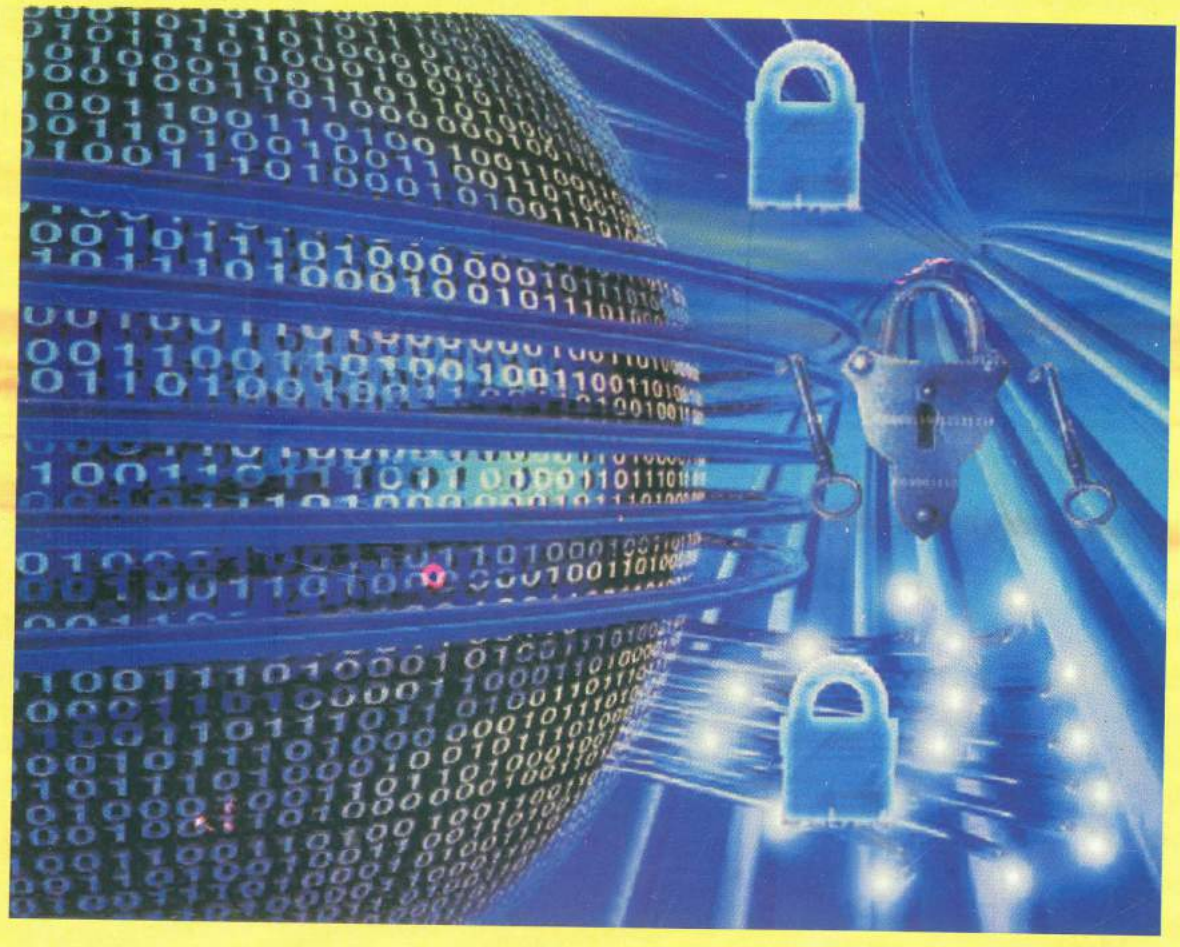
ignou
THE PEOPLE'S
UNIVERSITY

Indira Gandhi National Open University
School of Vocational Education and Training

MSEI-025
Application and Business Security Developments



Application Testing and Ethical Hacking

4

"शिक्षा मानव को बन्धनों से मुक्त करती है और आज के युग में तो यह लोकतंत्र की भावना का आधार भी है। जन्म तथा अन्य कारणों से उत्पन्न जाति एवं वर्गगत विषमताओं को दूर करते हुए मनुष्य को इन सबसे ऊपर उठाती है।"

– इन्दिरा गांधी

"Education is a liberating force, and in our age it is also a democratising force, cutting across the barriers of caste and class, smoothing out inequalities imposed by birth and other circumstances."

—Indira Gandhi

Indira Gandhi National Open University
School of Vocational Education and Training

MSEI-025
Application and Business Security Developments

Block

4

APPLICATION TESTING AND ETHICAL HACKING

Programme Expert/ Design Committee of Post Graduate Diploma in Information Security (PGDIS)

Prof. K.R. Srivathsan
Pro Vice-Chancellor, IGNOU

Mr. B.J. Srinath, Sr. Director & Scientist 'G',CERT-In, Department of Information Technology, Ministry of Communication and Information Technology Govt of India

Mr. A.S.A Krishnan, Director, Department of Information Technology, Cyber-Laws and E-Security Group, Ministry of Communication and Information Technology, Govt of India

Mr. S. Balasubramony, Dy. Superintendent of Police, CBI, Cyber Crime Investigation Cell, Delhi

Mr. B.V.C. Rao, Technical Director, National Informatics Centre, Ministry of Communication and Information Technology

Prof. M.N. Doja, Professor, Department of Computer Engineering, Jamia Milia Islamia
New Delhi

Dr. D.K. Lobiyal, Associate Professor, School of Computer and Systems Sciences, JNU
New Delhi

Mr. Omveer Singh, Scientist, CERT-In, Department of Information Technology, Cyber-Laws and E-Security Group, Ministry of Communication and Information Technology, Govt of India

Dr. Vivek Mudgil, Director, Eninov Systems
Noida

Mr. V.V.Subrahmanyam, Assistant Professor
School of Computer and Information Science
IGNOU

Mr. Anup Girdhar, CEO, Sedulity Solutions & Technologies, New Delhi

Prof. A.K. Saini, Professor, University School of Management Studies, Guru Gobind Singh Indraprastha University, Delhi

Mr. C.S. Rao, Technical Director in Cyber Security Division, National Informatics Centre, Ministry of Communication and Information Technology

Prof. C.G. Naidu, Director, School of Vocational Education & Training, IGNOU

Prof. Manohar Lal, Director, School of Computer and Information Science, IGNOU

Prof. K. Subramanian, Director, ACIIL, IGNOU
Former Deputy Director General, National Informatics Centre, Ministry of Communication and Information Technology, Govt. of India

Prof. K. Elumalai, Director, School of Law
IGNOU

Dr. A. Murali M Rao, Joint Director, Computer Division, IGNOU

Mr. P.V. Suresh, Sr. Assistant Professor, School of Computer and Information Science, IGNOU

Ms. Mansi Sharma, Assistant Professor, School of Law, IGNOU

Ms. Urshla Kant
Assistant Professor, School of Vocational Education and Training, IGNOU
Programme Coordinator

Block Preparation

Unit Writers

Ms. Rakhee Chhibber
Assistant Professor, Rukmini Devi Institute of Advanced Studies
Madhuban Chowk, Rohini, Delhi
(Unit 1)

Dr. Neeru Mundra
Associate Professor, Banarsidas Chandiwala Institute of Professional Studies, Dwarka, New Delhi
Ms. Renu Vashisth
Assistant Professor, Banarsidas Chandiwala Institute of Professional Studies, Dwarka, New Delhi (Unit 2)

Mr. Balraj Kumar
Head, Department of Computer Applications
School of Computer Applications, Lovely Professional University
Phagwara (Unit 3)

Mr. Arun Bakshi
Assistant Professor (Sr) (Information Technology) Gitarattan International Business School (giBS)
Madhuban Chowk, Delhi
(Unit 4)

Block Editor

Mr. P.V. Suresh
Sr. Assistant Professor
School of Computer and Information Science
IGNOU

Ms. Urshla Kant
Assistant Professor, School of Vocational Education and Training, IGNOU

Proof Reading and Format Editing

Ms. Urshla Kant
Assistant Professor, School of Vocational Education and Training, IGNOU

PRODUCTION

Mr. B. Natrajan	Mr. Jitender Sethi	Mr. Hemant Parida
Dy. Registrar (Pub.)	Asstt. Registrar (Pub.)	Proof Reader
MPDD, IGNOU	MPDD, IGNOU	MPDD, IGNOU

Feb, 2012

ISBN: 978-81-266-5892-3

Further information on the Indira Gandhi National Open University courses may be obtained from the University's office at Maidan Garhi, New Delhi-110 068 or the website of IGNOU www.ignou.ac.in

Printed and Published on behalf of the Indira Gandhi National Open University, New Delhi, by the Registrar, MPDD.

Printed at: Berry Art Press A-9, Mayapuri, Phase-I New Delhi-64

BLOCK INTRODUCTION

This block deals with application testing and ethical hacking. Application testing deals with tests for the entire application. This is driven by the scenarios from the analysis team. Application limits and features are tested here. The application must successfully execute all scenarios before it is ready for general customer availability. After all, the scenarios are a part of the requirement document and measure success. Application testing represents the bulk of the testing done by industry. The horizon of web applications is increasing day by day, thus its popularity and utility also. But security concerns are also very important components of the web applications. Also hacking is a threat for the users who travel across the cyber world for their personal and professional reasons. This block comprises of four units and is designed in the following way;

The **Unit One** is an effort towards answering some of the fundamental queries about assessment methodologies and tools. We have tried to give an overview of software testing and also benefits of performance testing. It covers Software Testing Life Cycle and application testing methodologies in detail.

The **Unit two** covers application security assessments. Organizations should ensure that appropriate security assessments are carried out on all custom, in-house developed, applications.

The **Unit three** covers WEB application scanning and vulnerability assessment. Web applications are vital components of any organization, but these applications can be dangerously weak links in your security framework. Hackers love to target web applications -especially when there's opportunity for financial gain. Web Application Scanner is an automated program to test web applications for common security vulnerabilities like Cross-Site Scripting, SQL Injection, directory traversal, insecure configurations, and remote command execution vulnerabilities by launching a series of Web attacks.

Unit four explains about the WEB application ethical hacking. This unit is all about the web applications and security concerns during use of web applications. The learners will learn about the web application and the respective security concerns. The topics like session ID is also covered. The threat risk modeling and web services challenges are also covered well.

Hope you benefit from this block.

ACKNOWLEDGEMENT

The material we have used is purely for educational purposes. Every effort has been made to trace the copyright holders of material reproduced in this book. Should any infringement have occurred, the publishers and editors apologize and will be pleased to make the necessary corrections in future editions of this book.

UNIT 1 ASSESSMENT METHODOLOGIES AND TOOLS

Structure

1.0 INTRODUCTION

Application testing deals with tests for the entire application, we have created. This is wholly depends upon scenarios of the analysis team. In this testing application limits and features are tested. The application must successfully execute all components before it is ready for general customer availability. All the components of application are a part of the requirement document and measure success.

These tests are done with a collection of parameters and collective results traditionally these tests may have been written by hand but in many modern systems this process become automated.

Mostly current applications are now have graphical user interfaces (GUI) and testing of a GUI to assure quality becomes a bit of a problem because most of them have event loops which contains signals for mouse, keyboard, window, and other related events. The coordinates on the screen are associated with each event. The screen coordinates are related back to the GUI object and then the event are serviced. Unfortunately, if some GUI object is positioned at a different location on the screen, then the coordinates change in the event loop and logically the events at the new coordinates should be associated with the same GUI object. This logical association can be accomplished by giving unique names to all of the GUI objects and providing the unique names as additional information in the events in the event loop. The GUI application reads the next event off of the event loop, locates the GUI object, and services the event.

Stress testing is designed to determine if the system can function when subject to large volume- larger than the normally expected. It also deals with the quality of the application in the environment. The areas that are stressed include input transactions, internal tables, disk space, output, communications, computer capacity and interaction with people. If the application functions properly under test, it can be assumed that it will function normally with normal volume of wok. This is the hardest and most complex category of testing to accomplish and it requires a joint effort from all teams.

A test environment is established with many testing stations. More and more stations are added, all simultaneous hammering on the system, until the system breaks. The system is repaired and the stress test is repeated until a level of stress is reached that is higher than expected to be present at a customer site.

Main problems of stress testing are **Race conditions** and **memory leaks.** A race condition is a conflict between at least two tests. Each test works correctly when done in isolation. When the two tests are run in parallel, one or both of the tests fail. This is usually due to an incorrectly managed lock.

A memory leak happens when a test leaves allocated memory behind and does not correctly return the memory to the memory allocation scheme. The test seems to run correctly, but after being exercised several times, available memory is reduced until the system fails.

1.1 OBJECTIVES

After studying this unit, you should be able to:

- define about the software testing;

- know about the type of performance testing of an application and also benefits of performance testing;
- know about the Software Testing Life Cycle;
- know about the application testing Methodology;
- assess application testing;
- explain the V-model of Software Testing;
- explain testing techniques / tools selection process; and
- identify tools for application testing.

1.2 SOFTWARE TESTING

Software testing is a process of *verifying* and *validating* that a software application or program satisfies its specified requirements or identifying differences between expected and actual results. It also identifies *defects*, flaws, or errors in the application code that must be fixed.

Software testing is a group activity and not one-person job. The testing team cannot improve quality; they can only measure it, although it can be argued that doing things like designing tests before coding begins will improve quality because the coders can then use that information while thinking about their designs and during coding and debugging.

Software testing has three main purposes: verification, validation, and defect finding.

- The *verification* process confirms that the software meets its technical specifications. A "specification" is a description of a function in terms of a measurable output value given a specific input value under specific preconditions. A simple specification may be along the line of "a SQL query retrieving data for a single account against the multi-month account-summary table must return these eight fields <list> ordered by month within 3 seconds of submission."
- The *validation* process confirms that the software meets the business requirements. A simple example of a business requirement is "After choosing a branch office name, information about the branch's customer account managers will appear in a new window. The window will present manager identification and summary information about each manager's customer base: <list of data elements>." Other requirements provide details on how the data will be summarized, formatted and displayed.

- A *defect* is a variance between the expected and actual result. The defect's ultimate source may be traced to a fault introduced in the specification, design, or development (coding) phases.

1.3 PERFORMANCE TESTING

Performance testing includes testing that the system will perform as specified at predetermined levels, including wait times, static processes, dynamic processes, transaction processes. It is also tested at the client/server level. Performance-related activities are concerned with achieving response times, throughput, and resource-utilization levels that meet the performance objectives for the application under test.

1.3.1 Types of Performance Testing

The following are the most common types of performance testing for applications.

Term	Purpose	Notes
Performance test	It determines the Speed , Scalability Stability and Confidence –	It is a technical investigation used to determine - • Does the application respond quickly enough for the intended users?(speed) • Will the application handle the expected user load and beyond? (scalability) • Is the application stable under expected and unexpected user loads?(Robustness) • Are you sure that users will have a positive experience on go-live day?
Load test	Load Tests are end to end performance tests under normal and anticipated peak production load	• This is a traditional Quality Assurance (QA) type test. It is conducted to verify that our application can meet our desired performance objectives which are specified in a service level agreement (SLA). This test enables us to measure response times, throughput rates, and resource-utilization levels, and to identify our application's breaking point, assuming that the breaking point occurs below the peak load condition. • It also determine, the minimum configuration that will allow the

		system to meet the formal stated performance expectations - so that extraneous hardware, software and the associated cost of ownership can be minimized. This is also a Business Technology Optimization (BTO) type test. • Endurance testing is also known as soak testing and it is a subset of load testing. it is a type of performance test focused on checks for memory leaks or other problems that may occur with prolonged execution. This type of testing may be used to calculate Mean Time Between Failure (MTBF) and Mean Time To Failure (MTTF).
Stress test	• Determines the load under which a system fails, and how it fails	• In a stress test, many more connections will be requested per minute than under normal levels of expected peak activity. The major aim of stress testing is to find application bugs that comes out only under high load conditions. These bugs can be related to synchronization issues, race conditions, and memory leaks. This testing enables us to identify our application's weak points, and shows how the application behaves under extreme load conditions. • Spike testing is a subset of stress testing. These tests are normally based on real world or projected work loads with focus on extreme load/performance conditions. • For example - in a web page number of visitors accessing the page suddenly increases to maximum then performance of the page breaks down. The process of testing a performance of web page due unexcepted traffic on the page is called as Spike Testing....
Capacity test	To determine how many users and/or transactions a given system will support and still meet	• Capacity testing is conducted in conjunction with capacity planning, which you use to plan for future growth, such as an increased user base or increased volume of data. For example, to accommodate future loads,

	performance goals.	you need to know how many additional resources (such as processor capacity, memory usage, disk capacity, or network bandwidth) are necessary to support future usage levels. • Capacity testing helps you to identify a scaling strategy in order to determine whether you should scale up or scale out.

1.3.2 Benefits by Key Performance Test Types

Term	Benefits	Challenges and Areas Not Addressed
Performance test	• Determines the speed, scalability and stability characteristics of an application, so that system can provide a good business decisions. • Focuses on whether the user of the system will be satisfied with the performance of the application. • It Identifies differences s between performance-related expectations and reality. • Supports tuning, capacity planning, and optimization efforts.	• May not detect some functional defects which only appear under load. • May only be indicative of performance characteristics if not carefully designed and validated, • There is always be a degree of uncertainty in the results.
Load test	• Determines the throughput required for peak load. • Determines the adequacy of a hardware environment. • Evaluates the load balancer. • Detects concurrency issues. • Detects functional errors under load. • It Collects data for scalability and capacity-planning purposes.	• Focus on speed of response. • Results could only be used for comparison with other load tests.

	• It determine how many users, an application can handle without compromising the performance • Helps to determine how much load the hardware can handle	
Stress test	• It find out if data can be corrupted by overstressing the system. • Provides an estimate of load on an application can handle before causing failures and errors with slowness. • Allows you to establish application-monitoring triggers to warn of impending failures. • Ensures that the security will opened up by stressful conditions. • the side effects of common hardware • determine different kinds of failures are most valuable to plan for.	• Because stress tests are unrealistic by design, so the stakeholders may dismiss test results. • Difficult to know how much stress is worth. • It is possible to cause application and /or network failures that may result in significant disruption if not isolated to the test environment.
Capacity test	• Provides information about how workload can be handled to meet business requirements. • Provides actual data that capacity planners can use to validate or enhance their models and/or predictions. • Enables you to conduct various tests to compare capacity-planning models and/or predictions. • Determines the current usage and capacity of the existing system to aid in capacity planning.	• Capacity model validation tests are complex to create. • Not all aspects of a capacity-planning model can be validated through testing at a time when those aspects would provide the most value.

	• Provides the usage and capacity trends of the existing system to aid in capacity planning	

Although the benefits are outweigh than the challenges related to performance testing, uncertainty over the relevance of the resulting data.

1.3.3 Additional Terms used in Performance Testing

There are some more terms when we talk about the performance testing. These tests are not necessarily performed in all industries or organizations. Some may perform them and others may not. These terms and concepts have been included because they are used frequently enough, and cause enough confusion, to make them worth knowing.

Term	Notes
Component test	A *component test* targets an architectural component of the application like servers, databases, networks, firewalls, clients, and storage devices.
Investigation	*Investigation* is based on collecting information related to the speed, scalability, and/or stability characteristics of the product. Investigation is frequently employed to prove or disprove hypotheses regarding the root cause of one or more observed performance issues.
Smoke test	A *smoke test* is the initial run of a performance test to see if your application can perform its operations under a normal load.
Unit test	A *unit test* is any test that targets a module of code where that module is any logical subset of the entire existing code base of the application for example - modules include functions, procedures, routines, objects, methods, and classes. This Performance unit tests are conducted by the developer who wrote the module of code.
Validation test	It compares the speed, scalability, and/or stability characteristics of the product against the expectations that have been set or presumed for that product.

Check Your Progress 1

Note: a) Space is given below for writing your answer.

b) Compare your answer with the one given at the end of the Unit.

1) What is software testing and why it is required?

.....

.....

.....

.....

2) Explain differences between verifications, validations and defects.

.....

.....

.....

.....

3) What are the challenges and benefits of the performance testing?

.....

.....

.....

.....

4) Explain the unit testing in details.

.....

.....

.....

.....

1.4 SOFTWARE TESTING LIFE CYCLE

STLC refers to a comprehensive group of testing related actions specifying details of every action along with the specification of the best time to perform such actions. There can not be a standardized testing process across various organizations, however every organization involved in software development business, defines and follows some sort of testing life cycle.

STLC comprises of following Seven Sequential Phases but it is not necessary that all the organization will follow these all steps:

1. **Planning of Tests**
2. **Analysis of Tests**
3. **Designing of Tests**
4. **Creation and Verification of Tests**
5. **Execution of Testing Cycles**
6. **Performance Testing, Documentation**
7. **Actions after Implementation**

So every company follows its own software testing life cycle to suit its own requirements, culture and available resources. The software testing life cycle interacts with the every phase of Software Development Life Cycle (SDLC). The main objective of the software testing life cycle is on manage and control all activities of software testing. It might be manual testing or an automated testing using some tool.

1.4.1 Planning of Tests

Ideas are tested now, not code. In this phase a project manager plans and identifies all the areas where testing efforts need to be applied within the boundaries of constraints like resources and budget. If the testing is not planned in the beginning, it can result in a poor quality product, dissatisfying the customer. Planning is not limited just to the initial phase, rather it is a continuous exercise extending till the end.

During this phase the reviewer read drafts of the planning documented. Then they gather data, using comparative product evaluations, focus groups or task analyses. The High Level Test Plan comprehensively describes the following:

- **Scope of Testing :**

  "Are these the right requirements "as to be tested, identification of features to be covered during testing.

- **Identification of Approaches for Testing:**

  "Are they achievable "

- **Defining Risks:**

  "Are they testable?"

- **Identification of resources :**

  Identification of resources like man, materials and machines which need to be deployed during Testing

- **Time schedule:**

  For performing the decided testing is aimed to deliver the end product as per the commitment made to the customer.

Involvement of software testers begins in the planning phase of the software development life cycle. During the design phase, testers work with developers in determining what aspects of a design are testable and with what parameters those tests will work. Factors that influence the software system include:

a) Size of the software

b) Percentage of the design and/or code that is new

c) Complexity of the software system

d) Difficulty level of design and coding

e) Quality

f) Languages to be used

g) Security classification level of the project

h) Target machine

i) Utilization of target hardware

1.4.2 Analysis of Tests

The test plan describes how testing will be accomplished. If the plan is developed carefully, test execution, analysis and reporting will flow smoothly. The objective of the test analysis is to describe all testing that is to be accomplished, together with the resources and schedule necessary for completion.

- Identification of Types of Testing to be performed during various stages of Software Development Life Cycle.
- Identification of extent to which automation needs to be done.
- Identification of the time at which automation is to be carried out.
- Identification of documentation required for automated testing

The Software project can't be successful unless there is frequent interaction among various teams involved in Coding and Testing with the active involvement of the Project Managers, Business Analysts or even the customer. Any deficiencies in the decided test plans come to the surface, during such meetings of cross-functional teams. This provides an opportunity to have a rethinking and refining the strategies decided for testing.

Based upon the customer requirements a detailed matrix for functional validation is prepared to cover the following areas:

- Ensure that each and every business requirement is getting covered through some test case or the other.
- Identification of the test cases best suited to the automated testing

- Identification of the areas to covered for performance testing and stress testing
- Carry out detailed review of documentation covering areas like Customer Requirements, Product Features and Specifications and Functional Design etc.

1.4.3 Designing of Tests

This phase involves the following:

- Further polishing of various Test Cases, Test Plans
- Revision and finalization of Matrix for Functional Validation.
- Finalization of risk assessment methodologies.
- In case line of automation is to be adopted, identification of test cases suitable for automation.
- Creation of scripts for Test cases decided for automation.
- Preparation of test data.
- Establishing Unit testing Standards including defining acceptance criteria
- Revision and finalization of testing environment.

1.4.4 Creation and Verification of Tests

This phase involves the following:

- Finalization of test plans and test cases
- Completion of script creation for test cased decided for automation.
- Completion of test plans for Performance testing and Stress testing.
- Providing technical support to the code developers in their effort directed towards unit testing.
- Bug logging in bug repository and preparation of detailed bug report.

Performing Integration testing followed by reporting of defects detected if any.

1.4.5 Execution of Testing Cycles

This phase involves the following:

- Completion of test cycles by executing all the test cases till a predefined stage reaches or a stage of no detection of any more errors reach.

- This is an iterative process involving execution of Test Cases, Detection of Bugs, Bug Reporting, Modification of test cases if felt necessary, Fixing of bugs by the developers and finally repeating the testing cycles.

1.4.6 Performance Testing and Documentation

- Execution of test cases pertaining to performance testing and stress testing.
- Revision and finalization of test documentation
- Performing Acceptance testing, load testing followed by recovery testing
- Verification of the software application by simulating conditions of actual usage.

1.4.7 Actions after Implementation

This phase involves the following:

- Evaluation of the entire process of testing.
- Documentation of TGR (Things Gone Right) and TGW (Things Gone Wrong) reports. Identification of approaches to be followed in the event of occurrence of similar defects and problems in the future.
- Creation of comprehensive plans with a view to refine the process of Testing.
- Identification and fixing of newly cropped up errors on continuous basis.
- Winding up of the Winding up of the test environment and restoration of all test equipment to the original base line conditions.

1.5 ASSESSMENT METHODOLOGIES

Application testing deals with tests for the entire application. This is driven by the scenarios from the analysis team. Application limits and features are tested here. The application must successfully execute all scenarios before it is ready for general customer availability. After all, the scenarios are a part of the requirement document and measure success. Application testing represents the bulk of the testing done by industry.

Due to fundamental new technologies in Windows 2000, you need to test your business applications for compatibility with the operating system as part of your Windows 2000 deployment project. Even if you currently use Windows NT, you should not assume that all your applications will work the same way with Windows 2000. Enhancements, such as improved security, mean that you must retest the applications that were developed for previous versions of Windows. These applications might not take full advantage of the new features available in

Windows 2000. They should, however, still perform as well on Windows 2000 as they do on the current platform.

Application Testing Process

First, you need to identify your Windows-based applications and prioritize them by how important they are to your application. As the inventory effort proceeds, you can start planning how you want to coordinate the testing. Then, as testing progresses, you need to report your status periodically to management and resolve compatibility problems as they arise.

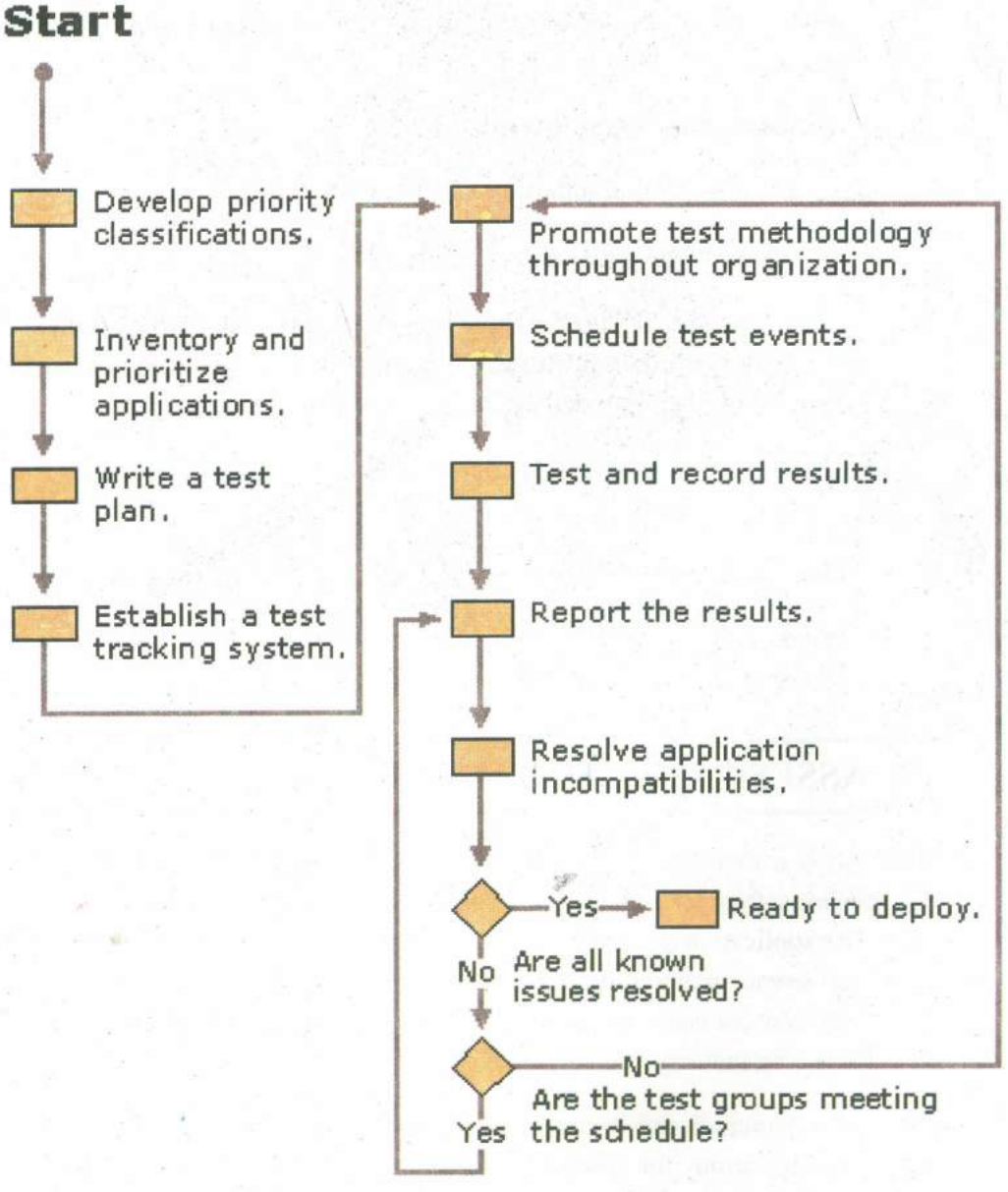


Fig. 1

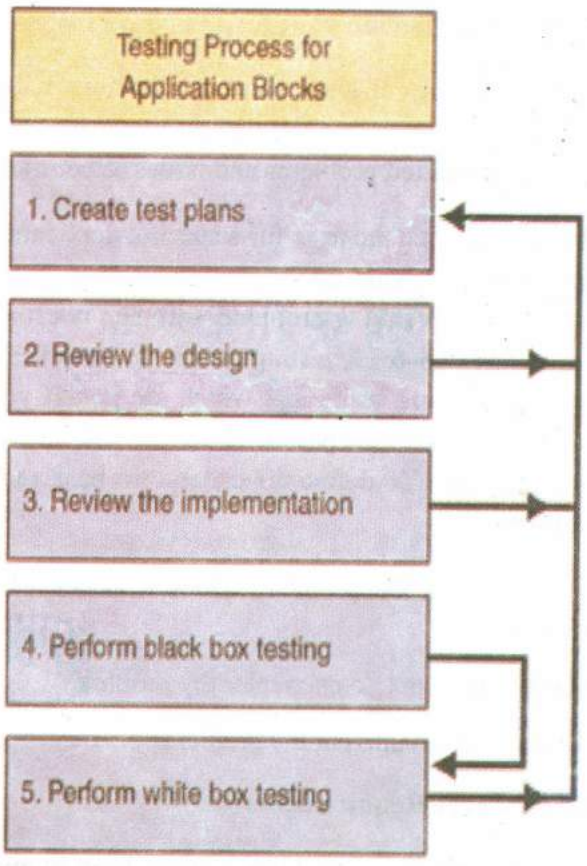


Fig. 2: Steps for Testing Applications

1) **Preparing an Application Test Plan**

One of the primary tasks in preparing for testing is to write a test plan. In the test plan, you specify the scope and objectives for the testing and describe the methodology to be used. Include the following information in your plan:

- **Scope:** The priority levels you address during testing.
- **Methodology:** Who does the testing and how you involve participants.
- **Requirements:** What hardware, software, personnel, training, and tools you need to perform the testing.
- **Criteria for pass-fail:** The factors that determine whether an application passes or fails.
- **Schedule:** How you plan to complete the testing by the scheduled rollout.

Depending on the number of applications and your test approach, application testing might require considerable cooperation from various units in your organization. Identify the application stakeholders early in the project and ask them to review and approve your test plan or to commit their resources to an agreed-upon level.

2) Defining Pass-Fail Criteria

As testers conduct a variety of tests, some applications will pass and some will fail. You should have a procedure defined so that participants know when and where they can log application problems and issues to be resolved.

When testers have completed the tests for a specific application, they need to enter the results in your test tracking and reporting system. You need, of course, to prioritize and track all outstanding problems and then retest applications when the problems are resolved. To track testing progress, however, you might want to know which applications are ready and which are not. If you plan to track your progress by whether applications passed or failed, you need to define the criteria for each category you use. To define the criteria for pass and fail, consider issues such as the following:

- How significant is the problem? Does it affect a critical function or a peripheral one?
- How likely is someone to encounter the problem?
- Is there a way to circumvent the problem?

3) Identifying Resource Requirements

As you plan for application compatibility testing, keep in mind the future state of your computing environment. Are you planning to upgrade some of your software to versions that fully use new Windows 2000 features? Are you planning to implement new standard desktop configurations or use Terminal Services? Issues such as these determine the resources that are required and the applications that are to be tested as a suite.

If you plan to deploy new applications with Windows 2000 during the rollout, test these applications with the current applications.

4) Defining the Testing Methodology

A major part of your test plan is describing the strategy for testing. When planning your methodology, consider:

- Where will the testing take place?
- Who will perform the tests?
- How will you communicate with and involve participants?
- How will you schedule the testing?
- How will you manage application problems?

Outsourcing is one option for application testing. To determine if you will use this option, consider the following:

- Do you have staff available for testing?
- Does your staff have the appropriate level of expertise?
- What are the internal costs compared to the outsourcing costs?
- What is your time frame? Can the testing get done faster if you outsource it?
- What are your security requirements? Would you need to provide confidential data to an external organization?

When you **test internally**, select experienced testers. If your organization has a group of application testers, it is recommended that you use them. If you do not have such a group or they are unavailable, look for ways to use a variety of resources to achieve the best results in a reasonable amount of time. For example, you can use a few experienced testers to develop a battery of test cases, which they can train others to run. Alternatively, you might have the experienced testers perform a core set of tests and then coordinate with business units to have their experts come to the lab to perform the functions they use in their work.

Devise a process for scheduling test days and communicating with the testers. For example, you might set up a Web site on your intranet where anyone can view test dates, status reports, contact names, and other relevant documents.

Establish a procedure for managing test results. Describe roles and responsibilities, including the following:

- Who enters problem reports in the incident tracking system?
- How problems are prioritized, assigned, and resolved?
- Who tracks the resolution of problems and retesting of applications?
- How do testers enter test results in the test tracking and reporting system?

1.6 THE V-MODEL OF SOFTWARE TESTING

The V-Model of testing identifies five software testing phases, each with a certain type of test associated with it.

Phase	Guiding Document	Test Type
Development Phase	Technical Design	Unit Testing
System and Integration Phase	Functional Design	System Testing Integration Testing
User Acceptance Phase	Business Requirements	User Acceptance Testing
Implementation Phase	Business Case	Product Verification Testing
Regression Testing applies to all Phases		

Fig. 3

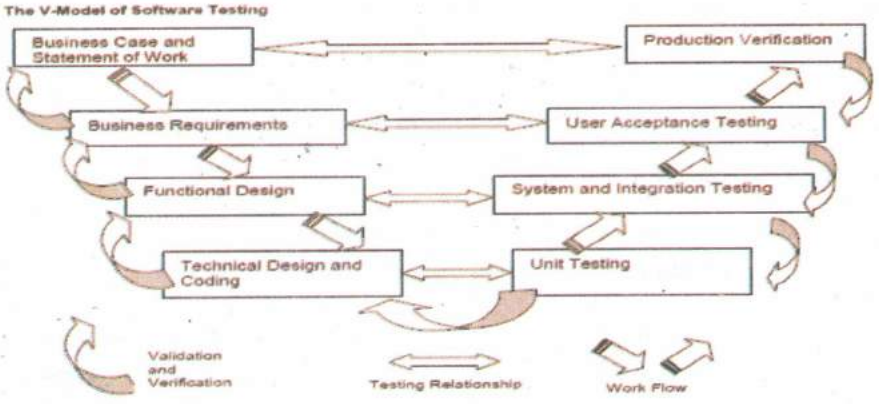


Fig. 4

Check Your Progress 2

Note: a) Space is given below for writing your answer.

b) Compare your answer with the one given at the end of the Unit.

1) What do you understand by the Software Testing Life Cycle?

.....

.....

.....

.....

2) Explain the steps for the testing process for application blocks.

.....

.....

.....

.....

3) What is the V model of Software Testing?

.....

.....

.....

.....

4) What is the pass-fail criteria?

.....

.....

.....

.....

1.7 TESTING TECHNIQUES/TOOLS SELECTION PROCESS

There are three testing concepts:

1.7.1 Structural Versus Functional Testing

Structural testing is also known as white box or open box testing which, is applied to sequential code and concentrates on checking that all executable statements within each module have been exercised and the corresponding branches and paths through that module have been covered. If there is a section of code that has never been exercised then there is a high possibility that it could contain an error that will remain undetected. One of the objectives of structural testing is to raise the designer's confidence level and ensure that it do not contain any untested areas and behaves in a manner that closely matches the original design specification. Structural testing can be considered as concentrating on checking that the control logic operates correctly. The coverage measurements are statement, branch, condition and expression, and path coverage.

Functional testing, also known as black box or closed box testing, which applied to code that operates concurrently and concentrates on checking the interaction between modules, blocks or functional boundaries. The objective here is to ensure that `correct results" are obtained when `good inputs" are applied to the various parts of the design, and when `bad inputs" are applied the design operates in a predictable manner. Functional testing can therefore be considered as concentrating on checking that the data paths operate correctly. The coverage measurements are toggle, triggering, and signal trace coverage.

1.7.2 Dynamic Versus Static Testing

Static testing is a form of software testing where the software isn't actually used. It is generally not detailed testing, but checks mainly for the algorithm, or document. It is mainly syntax checking of the code or manually reading of the code or document to find errors. This type of testing can be used by the developer who wrote the code. Code reviews, inspections and walkthroughs methods are also used for this type of testing. This is the **verification act**tivity. Code Reviews, inspection and walkthroughs are few of the static testing methodologies.

In **dynamic testing** the software must actually be compiled and run.

Dynamic analysis refers to the examination of the physical response from the system to variables that are not constant and change with time.

Some of dynamic testing methodologies include unit testing, integration testing, system testing and acceptance testing. Dynamic testing is the **validation** activities.

Unit Tests, Integration Tests, System Tests and Acceptance Tests are few of the Dynamic Testing methodologies.

1.7.3 Manual Versus Automatic Testing

Manual techniques are performed by people and automated techniques by the computers. Both Manual Testing and Automated Testing have several pros and cons.

Problems with Manual Testing: Some of the problems with manual testing are:

1) **Less Reliable:** Manual testing is not reliable, as there is no method available to find out whether the actual and expected results have been compared. We have to rely on the tester's words.

2) **High Risk:** A manual testing process is highly risky for oversights and mistakes. People get tired, they may be temporarily inattentive, they may have too many tasks on hand, they may be insufficiently trained and so on. Hence, unintentionally mistakes happen in entering data, in setting parameters, in execution and in comparisons.

3) **Incomplete Coverage:** Testing is quite complex when we have mix of multiple platforms, O.S. Servers, clients, channels, business processes etc. Testing is non-exhaustive. Full manual regression testing is impractical.

4) **Time Consuming:** Limited test resources makes manual testing simply too time consuming. As per a study done, 90% of all IT projects are delivered late due to manual testing.

5) **Facts and Fiction:** The fiction is that manual testing is done while the fact is only some manual testing is done depending upon the feasibility.

It is worth noting that the manual testing is used to do the documentation of tests, creating testing related guides according to data queries, providing structures for helping run the tests on temporary basis and measuring the test results.

Manual testing is considered to be costly and time-consuming; hence we use automated testing to cut down the time and cost.

Benefits of Automated Testing: On the contrary, automated testing is having many benefits.

Automated testing is the process of automating the manual testing process. We use automated testing to substitute or provide a supplement to manual testing with the use of a comprehensive suite of testing tools. Automated testing tools assist software testers to evaluate the quality of the software by automating the mechanical aspects of the software testing task. The benefits of automation are better software quality, lesser time for marketing, repeatability of testing

procedures and reduced cost of testing. We shall now list some more benefits of test automation. They are given below

1) Automated execution of test cases is faster than manual execution. This saves time. This time can also be utilized to develop additional test cases, thereby improving the coverage of testing.

2) Test automation can free test engineers from mundane tasks and make them focus on more creative tasks.

3) Automated tests can be more reliable. This is because manually running the tests may result in boredom and fatigue, more chances of human error. While automated testing overcomes all these shortcomings.

4) Automation helps in immediate testing, as it need not wait for the availability of test engineers. In fact,

   Automation = Lesser Person Dependence

5) Test cases for certain types of testing such as reliability testing, stress testing, load and performance testing cannot be executed without automation. For example, if we want to study the behavior of a system with millions of users logged in, there is no way one can perform these tests without using automated tools.

6) Manual testing requires the presence of test engineers but automated tests can be made to run round the clock, (24 x 7) environment. So, automated testing provides round the clock coverage.

7) Tests, once automated, take comparatively far less resources to execute. A manual test suite requiring 10 persons to execute it over 31 days i.e., 31 x 10 = 310 man days, may take just 10 man-days for execution, if automated. Thus, a ratio of 1: 31 is achieved.

8) Automation produces a repository of different tests, which helps us to train test engineers to increase their knowledge.

9) Automation does not end with developing programs for the test cases. Automation includes many other activities like selecting the right product build, generating the right test data, and analyzing results and so on.

Automation should have scripts that produce test data to maximize coverage of permutations and combinations of input and expected output for result comparison. They are called as test data generators.

It is important for automation to relinquish the control back to test engineers in situations where further sets of actions to be taken are not known.

As the objective of testing is to catch defects early, the automated tests can be given to developers so that they can execute them as part of unit testing.

Drawbacks of Automated Testing: Despite of many benefits, pace of test-automation is slow.

Some of its disadvantages are as under:

1) An average automated test suite development is normally 3-5 times the cost of a complete manual test cycle.
2) Automation is too cumbersome. Who would automate? Who would train? Who would maintain? These issues complicates the matter.
3) In many organizations, test automation is not even a discussion issue.
4) There are some organizations where there is practically no awareness or only some awareness on test automation.
5) Automation is not an item of higher priority for management. It does not make much difference to many organizations.
6) Automation would require additional trained staff. There is no staff for the purpose. Automation actually allows testing professionals to concentrate on their real profession of creating tests and test cases, rather than doing the mechanical job of test execution.

1.8 TOOLS FOR ASSESSMENT

1.8.1 Load Testing Tools

Tool Name	Company Name	Notes
AppLoader	NRG Global	Load and Performance testing Solution. Automates tests on the GUI level of the application. Can be used for unit, integration, and regression testing as well. Licensed.
blitz.io	Mu Dynamics	Blitz enables self-service load and performance testing for cloud and mobile applications. Solution is focused on continuous testing for DevOps that commonly make multiple changes every day.

IBM Rational Performance Tester	IBM	Eclipse based large scale performance testing tool primarily used for executing large volume performance tests to measure system response time for server based applications. Licensed.
JMeter	An Apache Jakarta open source project	Java desktop application for load testing and performance measurement.
Load Test (included with Soatest	Parasoft	Performance testing tool that verifies functionality and performance under load. Supports SOAtest tests, JUnits, lightweight socket-based components. Detects concurrency issues.
LoadRunner	HP	Performance testing tool primarily used for executing large numbers of tests (or a large number or virtual users) concurrently. Can be used for unit and integration testing as well. Licensed.
OpenSTA	Open System Testing Architecture	Open source web load/stress testing application, licensed under the Gnu GPL. Utilizes a distributed software architecture based on CORBA. OpenSTA binaries available for Windows.
SilkPerformer	Micro Focus	Performance testing in an open and sharable model which allows realistic load tests for thousands of users running business scenarios across a broad range of enterprise application environments.
SLAMD		Open source, 100% Java web application, scriptable, distributed with Tomcat.

Visual Studio Load Test	Microsoft	Visual Studio includes a load test tool which enables a developer to execute a variety of tests (web, unit etc...) with a combination of configurations to simulate real user load.[1]

1.8.2 Performance Testing Tools

These tools provide data that can be used to graph a "capacity curve" which determines a saturation point based on where the TPS (Transactions Per Second or throughput) flattens out and the Response Time becomes unacceptable (7 secs, desired is 3-4 secs). You use workload tools to provide stress on the web application and you use performance monitoring tools to determine the bottleneck.

- **Windows NT Resource Kits** by Microsoft
  - **Netmon** - Network Monitor - Sniff networking packets and analyze them from a remote network. Monitor network activity proactively.
  - **Perfmon** - Performance Monitor - Monitor performance of system objects in real time. Log to a file, send administrative alerts on thresholds, and generate reports.
  - **APIMon** - API Profiler - User mode debugging tool that supports attaching to applications remotely to trace API calls.
- **Vtune™** Performance Analyzer by Intel
  - http://developer.intel.com/vtune/analyzer/index.htm - Vtune home page.
  - Quotes from their site:
    - "The VTune Performance Analyzer collects, analyzes, and displays software performance data from the system-wide view down to a specific module, function or instruction in your code."
  - http://www.intel.com - Intel's home page.
- **Quantify** by Rational Software
  - http://www.rational.com/products/pqc/index.jsp - Perfmon home page.
  - Quotes from their site:
    - "Rational Purify, Quantify and PureCoverage are a complete set of automated runtime analysis tools for improving application performance and quality for software developers who need to build

and deploy resilient, reliable software applications in C/C++, Java and VB.".

- http://www.rational.com/products/index.jsp - Rational's Products. Click on the "System Testing" link. (Rational Suite TestStudio)

- **eGurkha**
  - www.egurkha.com/technology.htm - is an integrated monitoring application suite that addresses the infrastructure and business monitoring needs of eBusinesses.

Additional Performance Tools

- **InetMonitor** (http testing) by Microsoft.
  - http://www.microsoft.com/SiteServer/site/deployadmin/inetmonitor.htm - INetMonitor home page.
  - Quote from site:
    - "This load generation tool allows you to plan and maintain the optimal configuration for your site. The Simulator component of InetMonitor allows you to simulate user load on your site, as well as any type of client or user behavior. The Monitor component allows you to track the behavior of a typical user. Using the information gathered in these tests, you can assess your site's capacity, plan for growth, and track hardware resource utilization."
- **WCAT** - "Windows Capacity Analysis Tool" by Microsoft
  - http://msdn.microsoft.com/workshop/server/toolbox/wcat.asp - WCAT download and user guide site.
  - Quotes from site:
    - "..runs simulated workloads on client-server configurations. Using WCAT, you can test how your Internet Information Services and network configuration respond to a variety of different client requests for content, data, or Hypertext Markup Language (HTML) pages. The results of these tests can be used to determine the optimal server and network configuration for your computer. WCAT is specially designed to evaluate how Internet servers running Windows 2000 (or Windows NT®) and Internet Information Services respond to various client workload simulations."

- **Windows DNA Performance** Kit by Microsoft
  - Web Driver—The Web Driver simulates the actions of many Web clients making requests to your server. This allows you to test the performance of your Web applications in a manner similar to the Microsoft® Web Capacity Analysis Tool (WCAT). See the IIS Resource Kit for more information about WCAT.
  - Perfcol—The performance collector is a stand-alone application that you can use to monitor any number of performance counters across a large number of machines. The data is stored in a database to make further analysis easy. The toolkit also includes a way to automate performance collection for experiments run with the toolkit.
- **LoadSim by Microsoft** (to stress test Exchange 5.5 servers)
  - http://www.microsoft.com, then search for "LoadSim". Hard to find a product page but you will find white papers.
  - Quote from site:
    - You can also use the Microsoft Exchange Server Load Simulator (Loadsim.exe) tool to help you determine the level of performance that's acceptable for your organization's users. Load Simulator can help you determine how many users your Microsoft Exchange Server computer can support. It is designed to provide a realistic load on a Microsoft Exchange Server computer by simulating the behavior or users on one or more Microsoft Outlook computers. For more information on running the Load Simulator tool, see the *Microsoft Exchange Server Resource Guide.*
- **Open STA** - free tool.
  - http://www.opensta.org
- **NetBench® 6.0 by Ziff-Davis** (zdnet) (load generator to stress test file server I/O)
  - http://www.zdnet.com/zdbop/netbench/netbench.html
  - Quotes from site:
    - NetBench is a portable benchmark program that measures how well a file server handles file I/O requests from 32-bit Windows clients, which pelt the server with requests for network file operations.
- **BenchWeb**
  - http://www.netlib.org/benchweb

- Quotes from site:
  - BenchWeb is a starting point for finding information about computer system performance benchmarks, benchmark results, and benchmark code. The site is maintained at the University of Tennessee Computer Science Department by Innovative Computing Labs.

- **Systest Labs**
  - http://www.systest.com - SysTest Labs home page. Services - software testing and eBusiness testing

- **Automated Testing Specialists (ATS)**
  - http://www.sqa-test.com - ATS. Services - software testing

- **Web Performance Trainer by Web Performance**
  - http://www.webperfcenter.com - home page. Trial version available.
  - Quotes from site:
    - Web Performance Trainer 2 simulates multiple users hitting your web site so you can find performance bottlenecks, increase performance, or do capacity planning.
    - Runs on Windows NT, Linux, Solaris, and most UNIX variants
    - Supports any kind of back-end process, including Active Server Pages, Applets, servlets, plugins, Active X Components, ISAPI, and cgi-bin

- **PassMark Performance Test by PassMark Software (PC Benchmarking)**
  - http://www.passmark.com
  - Quotes from site:
    - Passmark Performance Test is an award winning PC hardware benchmark utility that allows everybody to quickly assess the performance of their computer and compare it to a number of standard 'baseline' computer systems.

Check Your Progress 3

Note: a) Space is given below for writing your answer.

b) Compare your answer with the one given at the end of the Unit.

1) What are the different types of software testing techniques?

.....

.....

.....

.....

2) What are the benefits of the automated testing techniques?

.....

.....

.....

.....

3) Give the name of 5-5 load testing and performance testing tools.

.....

.....

.....

.....

4) What is the difference between structural and functional testing?

.....

.....

.....

.....

1.9 LET US SUM UP

This unit is an effort towards answering some of the fundamental queries about assessment methodologies and tools. We have tried to give an overview of software testing and also benefits of performance testing. It covers Software Testing Life Cycle and application testing methodologies in detail.

Life Cycle of Software Testing (STLC)		
Phase	**Activities**	**Outcome**
Planning of Tests	• Creation of a Test Plan of High Level	Refined Test Plans and Specifications
Analysis of Tests	• Creation of fully descriptive Test Plan • Creation of Matrix for Functional Validation • Creation of Test Cases	Refined Test Plans, Test Cases and Matrix for Functional Validation
Designing of Tests	• Revision of Test Cases • Selection of Test Cases fit for automation	Refined Test Cases, Input Data Sets and Documents for Assessment of Risk
Creation and Verification of Tests	• Creation of scripts suitable for Test Cases for automation	Detailed Procedures for Testing, Testing Scripts, Test Reports and Bug-Reports
Execution of Testing Cycles	• Completion of Cycles of Testing	Detailed Test Reports and Bug-Reports.
Performance Testing, Documentation	• Execution of Test Cases related to performance tests and Stress Testing • ($) Detailed documentation	Test Reports, Documentation on various metrics used during testing
Actions after Implementation	• ($) Evaluation of all Processes of Testing	Detailed Plans for improving the process of testing

1.10 CHECK YOUR PROGRESS: THE KEY

Check Your Progress 1

1) Software testing is a process of *verifying* and *validating* that a software application or program.

   1. Meets the business and technical requirements that guided its design and development, and

   2. Works as expected.

   Software testing also identifies important *defects*, flaws, or errors in the application code that must be fixed.

2) ♦ The *verification* process confirms that the software meets its technical specifications. A "specification" is a description of a function in terms of a measurable output value given a specific input value under specific preconditions. A simple specification may be along the line of "a SQL query retrieving data for a single account against the multi-month account-summary table must return these eight fields <list> ordered by month within 3 seconds of submission."

   ♦ The *validation* process confirms that the software meets the business requirements. A simple example of a business requirement is "After choosing a branch office name, information about the branch's customer account managers will appear in a new window. The window will present manager identification and summary information about each manager's customer base: <list of data elements>." Other requirements provide details on how the data will be summarized, formatted and displayed.

   ♦ A *defect* is a variance between the expected and actual result. The defect's ultimate source may be traced to a fault introduced in the specification, design, or development (coding) phases.

3) The benefits and challenges of the performance testing

Benefits	Challenges
• Determines the speed, scalability and stability characteristics of an application, thereby providing an input to making sound business decisions. • Focuses on determining if the	• May not detect some functional defects that only appear under load. • If not carefully designed and validated, may only be indicative of performance

user of the system will be satisfied with the performance characteristics of the application. • Identifies mismatches between performance-related expectations and reality. • Supports tuning, capacity planning, and optimization efforts.	characteristics in a very small number of production scenarios. • Unless tests are conducted on the production hardware, from the same machines the users will be using, there will always be a degree of uncertainty in the results.

4) In computer programming, **unit testing** is a method by which individual units of source code are tested to determine if they are fit for use. A unit is the smallest testable part of an application. In procedural programming a unit may be an individual function or procedure. In object-oriented programming a unit is usually an interface, such as a class. Unit tests are created by programmers or occasionally by white box testers during the development process.

The primary goal of unit testing is to take the smallest piece of testable software in the application, isolate it from the remainder of the code, and determine whether it behaves exactly as you expect. Each unit is tested separately before integrating them into modules to test the interfaces between modules. Unit testing has proven its value in that a large percentage of defects are identified during its use.

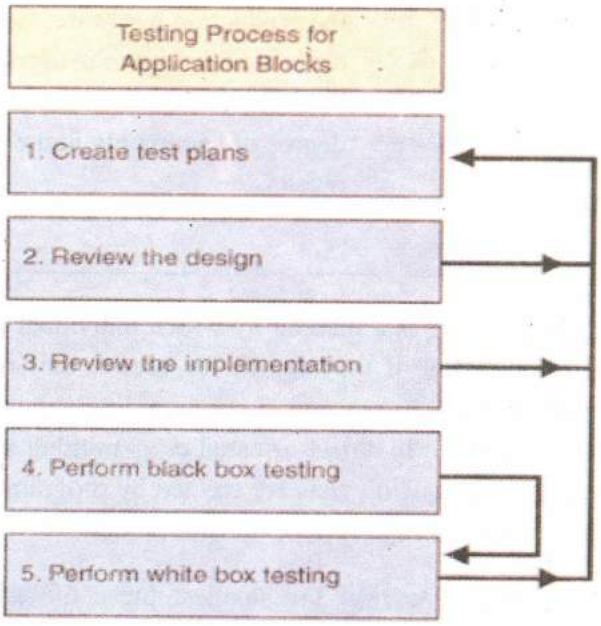
The most common approach to unit testing requires drivers and stubs to be written. The driver simulates a calling unit and the stub simulates a called unit. The investment of developer time in this activity sometimes results in demoting unit testing to a lower level of priority and that is almost always a mistake. Even though the drivers and stubs cost time and money, unit testing provides some undeniable advantages. It allows for automation of the testing process, reduces difficulties of discovering errors contained in more complex pieces of the application, and test coverage is often enhanced because attention is given to each unit.

Check Your Progress 2

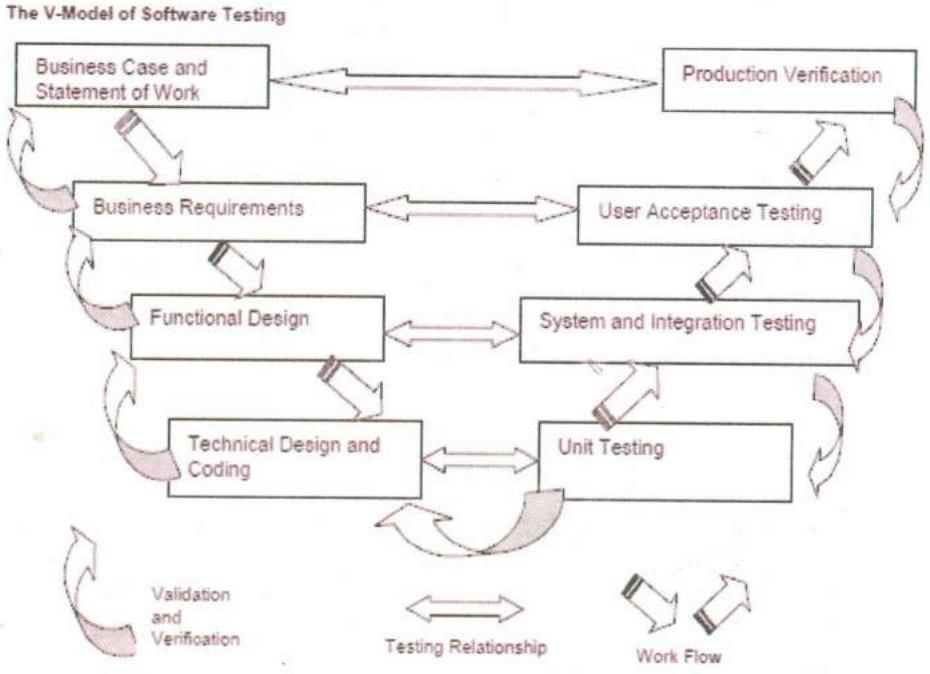
1) STLC by and large comprises of following Seven Sequential Phases:

a)Planning of Tests b) Analysis of Tests c) Designing of Tests d) Creation and Verification of Tests e) Execution of Testing Cycles f) Performance Testing, Documentation g) Actions after Implementation

2) The Steps for the testing process for application blocks are:



3) The V-Model of testing identifies five software testing phases, each with a certain type of test associated with it.



4) As testers conduct a variety of tests, some applications will pass and some will fail. You should have a procedure defined so that participants know when and where they can log application problems and issues to be resolved.

When testers have completed the tests for a specific application, they need to enter the results in your test tracking and reporting system. You need, of course, to prioritize and track all outstanding problems and then retest applications when the problems are resolved. To track testing progress, however, you might want to know which applications are ready and which are not. If you plan to track your progress by whether applications passed or failed, you need to define the criteria for each category you use. To define the criteria for pass and fail, consider issues such as the following:

- How significant is the problem? Does it affect a critical function or a peripheral one?
- How likely is someone to encounter the problem?
- Is there a way to circumvent the problem?

Check Your Progress 3

1) There are 3 different types of the software Testing :

   a) Structural versus Functional testing – structural testing also known as white box or open box testing, is normally applied to sequential HDL code and concentrates on checking that all executable statements within each module have been exercised and the corresponding branches and paths through that module have been covered. Functional testing, also known as black box or closed box testing, is normally applied to HDL code that operates concurrently and concentrates on checking the interaction between modules, blocks or functional boundaries. The objective here is to ensure that `correct results" are obtained when `good inputs" are applied to the various parts of the design, and when `bad inputs" are applied the design operates in a predictable manner.

   b) Dynamic versus static testing - **Static testing** is a form of software testing where the software isn't actually used. It is generally not detailed testing, but checks mainly for the sanity of the code, algorithm, or document. It is primarily syntax checking of the code or and manually reading of the code or document to find errors. This type of testing can be used by the developer who wrote the code, in isolation. Code reviews, inspections and walkthroughs are also used. This is the **verification** portion of Verification and Validation. These are verification activities. Code Reviews, inspection and walkthroughs are few of the static testing methodologies.

   In dynamic testing the software must actually be compiled and run.

   Dynamic analysis refers to the examination of the physical response from the system to variables that are not constant and change with time

Some of dynamic testing methodologies include unit testing, integration testing, system testing and acceptance testing. Dynamic testing is the validation portion of Verification and Validation. These are the Validation activities. Unit Tests, Integration Tests, System Tests and Acceptance Tests are few of the Dynamic Testing methodologies.

c) Manual versus automatic testing - Manual techniques are performed by people and automated techniques by the computers. Both Manual Testing and Automated Testing have several pros and cons.

2) **Benefits of Automated Testing:** On the contrary, Automated testing is having many benefits. Automated testing is the process of automating the manual testing process. We use automated testing to substitute or provide a supplement to manual testing with the use of a comprehensive suite of testing tools. Automated testing tools assist software testers to evaluate the quality of the software by automating the mechanical aspects of the software testing task. The benefits of automation are better software quality, lesser time for marketing, repeatability of testing procedures and reduced cost of testing. We shall now list some more benefits of test automation. They are given below

1. Automated execution of test cases is faster than manual execution. This saves time. This time can also be utilized to develop additional test cases, thereby improving the coverage of testing.

2. Test automation can free test engineers from mundane tasks and make them focus on more creative tasks.

3. Automated tests can be more reliable. This is because manually running the tests may result in boredom and fatigue, more chances of human error. While automated testing overcomes all these shortcomings.

4. Automation helps in immediate testing, as it need not wait for the availability of test engineers. In fact,

   Automation = Lesser Person Dependence

5. Test cases for certain types of testing such as reliability testing, stress testing, load and performance testing cannot be executed without automation. For example, if we want to study the behavior of a system with millions of users logged in, there is no way one can perform these tests without using automated tools.

6. Manual testing requires the presence of test engineers but automated tests can be made to run round the clock, (24 x 7) environment. So, automated testing provides round the clock coverage.

7. Tests, once automated, take comparatively far less resources to execute. A manual test suite requiring 10 persons to execute it over 31 days i.e., 31 x

10 = 310 man days, may take just 10 man-days for execution, if automated. Thus, a ratio of 1: 31 is achieved.

3) Load test tools are: = apploader, blitz.io, IBM Rational Performance Tester, JMeter, Load test (included with soatest) Performance Test tools are: = **InetMonitor** (http testing) by Microsoft, **WCAT** - "Windows Capacity Analysis Tool" by Microsoft, **Windows DNA Performance** Kit by Microsoft, **LoadSim by Microsoft** (to stress test Exchange 5.5 servers), **Open STA** - free tool from http://www.opensta.org.

4) Structural testing, also known as white box or open box testing, is normally applied to sequential HDL code and concentrates on checking that all executable statements within each module have been exercised and the corresponding branches and paths through that module have been covered. If there is a section of HDL code that has never been exercised then there is a high possibility that it could contain an error that will remain undetected. One of the objectives of structural testing is to raise the designer"s confidence level that the HDL code does not contain any untested areas and behaves in a manner that closely matches the original design specification. Structural testing can therefore be considered as concentrating on checking that the control logic operates correctly. The coverage measurements that fall into this category are: statement, branch, condition and expression, and path coverage. Functional testing, also known as black box or closed box testing, is normally applied to HDL code that operates concurrently and concentrates on checking the interaction between modules, blocks or functional boundaries. The objective here is to ensure that `correct results" are obtained when `good inputs" are applied to the various parts of the design, and when `bad inputs" are applied the design operates in a predictable manner. Functional testing can therefore be considered as concentrating on checking that the data paths operate correctly. The coverage measurements that fall into this category are: toggle, triggering, and signal trace coverage.

1.11 SUGGESTED READINGS

- *Better Software magazine.* Free issue at http://www.zinio.com/offer?issn= 1532-3579fandof=PF01andbd=1H
- Ergo/Gero http://www.ergogero.com/FAQ/Part4/cfaqPart4.htmlH (information on color blindness)
- Goldsmith, Robin F. (2002). *Software Development Magazine,* 4-part series, July-October. Information on the V-Model is at http://www.sdbestpractices.com/documents/s=8815/sdm0208e/
- International Institute for Software Testing. http://www.testinginstitute.com/H

- Marrik, Brian. *Classic Testing Mistakes* (1997) and *New Models for Test Development* (1999). http://www.testing.com/writings/writings.html
- Parasoft Corporation. *Automated Error Prevention.* http://www.parasoft.com/jsp/aep/aep.jsp?itemId=170
- Patton, Ron (2000). *Software Testing.*

UNIT 2 APPLICATION SECURITY ASSESSMENTS

Structure

2.0 INTRODUCTION

The past three decades have introduced many new factors in the Information Technology (IT) world that have changed the security landscape many times.

Introduction of UNIX has triggered the sprouting of multiple computers in a single location and introduction of Personal Computer (PC) has introduced the computing power on each desktop be it at home or at the office. The fact of the matter is that PC's have become so popular, that it is hard to imagine world without them today. As with any technological advancement, the good mixed with the bad like hackers, crackers, mischievous script kiddies, disgruntled employees and evil people. The road that has no limits or pre-defined routes, the road where every kid with a computer has the same power as any corporate CEO, where dangerousness of a person is measured in kilobits per second and megabytes of network traffic. Enter Internet in all of its capable-of-surviving-nuclear-attack glory. Just like in good old west days: "Have a PC, willing to hack".

2.1 OBJECTIVES

After studying this unit, you should be able to:

- define Application Security Assessment;
- describe Vulnerability Assessment;
- explain Security Assessment Methodology;
- describe Security Assessment Planning; and
- identify Assessment Techniques.

2.2 APPLICATION SECURITY ASSESSMENT

An application security assessment is the process of determining how effectively an entity being assessed (e.g., host, system, network, procedure, person—known as the assessment object) meets specific security objectives. Three types of assessment methods can be used to accomplish this—testing, examination, and interviewing.

- Testing is the process of exercising one or more assessment objects under specified conditions to compare actual and expected behaviors.
- Examination is the process of checking, inspecting, reviewing, observing, studying, or analyzing one or more assessment objects to facilitate understanding, achieve clarification, or obtain evidence.
- Interviewing is the process of conducting discussions with individuals or groups within an organization to facilitate understanding, achieve clarification, or identify the location of evidence. Assessment results are used to support the determination of security control effectiveness over time.

Application Security Assessment is designed to identify and assess threats to the organization through bespoke, proprietary applications or systems. Web application security is the security of all components – the web application being used, the web server running the web application and the modules running on the web server. All traffic directed to a web server is http or https traffic, which is legitimate traffic and is therefore not blocked by firewalls. Web servers are frequent targets of attack, prefer to attack web applications for the simple reason that no firewall blocks requests to the web server.

It is vital that they be assessed to ensure that

- The application doesn't expose the underlying servers and software to attack(s), and
- A malicious user cannot access, modify or destroy data or services within the system.

In a well-deployed and secured infrastructure, a weak application can expose the organization's information assets to unacceptable risk. It gives insight of the application security related research activities:

- Advisories of security vulnerabilities
- Security testing tools developed
- Highlight the innovative approach
- Presentations of security forums like on application security

2.2.1 NII Approach to Application Security Assessments

NII uses a number of software-testing techniques (including black-box testing, fault injection, and behavior monitoring), as well as real-world situations to test each application. The NII methodology is as described below:

High Level Design Audit

High Level Design Audit identifies and analyzes:

- Flow of information throughout the application environment
- Sensitive data in different sections of the organization
- Threats to the sensitive information in question

Source Code Audit

- In this step the code is reviewed for vulnerabilities and threats that belong to these categories:
  - Cryptography
  - Authentication

- Session Management
- Data Validation
- Exception Management
- Authorization
- Auditing and Logging

Black Box Testing

- Testing Communication Behavior
- Identifying Fault Injection Points
- Identifying Client-side behavior of the application
- Testing interactions with third-party applications
- File Interpretation
- Cryptanalysis

2.2.2 Benefits

Application Security Assessments help to:

- Secure the flow of information through the application
- Implement secure coding practices, remove logical, formatting flaws in the application code
- Embedding security right from the design to the execution stage
- Recognize the existing vulnerabilities and the extent of current and potential damages posed by the application
- Harden technologies keeping in mind the involvement of people which is a key criterion for any strategy to succeed

2.3 VULNERABILITY ASSESSMENT

Vulnerability Assessment is an integral part of any Information Security Risk Management program. 'Critical" vulnerabilities are present in software and systems and thus prone to exploitation, in other words IT systems are prone to **hacking**. Understanding and taking actions to these potential security breaches will help protect the business and build customer confidence.

2.3.1 Vulnerability Management

New vulnerabilities are discovered almost daily. Hence, it is very important that organizations identify them quickly and apply necessary patches. Managing vulnerabilities is a continual task.

There are three important factors that an organization needs to consider pertaining to Vulnerability Management:

1. The time taken to perform regular and periodical vulnerability assessments and the resources required;
2. The cost of vulnerability assessment and reporting exercise;
3. The transparency of the "vulnerability management" program to the management, stakeholders, customers and compliance auditors.

2.3.2 Automated Scanning and Reporting

The process of manually scanning and reporting the vulnerabilities in IT systems (operating systems, applications and networking devices) is labor intensive and is a tedious process. Such a manual process is also prone to errors due to variances in interpretation of scanning results. This provides automated scanning and reporting using regularly updated database of the published vulnerabilities. By utilizing pre-built reporting templates the system provides consistent reports.

2.3.3 On Demand and Cost Effective Scanning Through Internet

The system can be used to schedule scans at pre-defined interval such as daily, weekly, monthly and so on. Since the service is delivered through the centralized scanning server, the cost of scanning will be much lesser compared to manual scanning efforts.

2.3.4 Compliance and Third Party Certification

An organization need to demonstrate compliance with various regulations, legal requirements and information security standards as part of data protection / data privacy requirements. The vulnerability management program is an approved methodology and organization can secure a certification for the vulnerability management program from DNV or any other organisation giving greater confidence to the customers and users.

Check Your Progress 1

Note: a) Space is given below for writing your answer.

b) Compare your answer with the one given at the end of the Unit.

1) What do you mean by Application Security Assessment?

.....

.....

.....

.....

2) Explain any two NII methodologies.

.....

.....

.....

.....

3) What is Vulnerability Assessment?

.....

.....

.....

.....

4) Write short not on testing, examination, and interviewing in security assessmert.

.....

.....

.....

.....

2.4 INFORMATION SECURITY ASSESSMENT METHODOLOGY

A repeatable and documented security assessment methodology is beneficial in that it can:

- Provide consistency and structure to security testing, which can minimize testing risks.
- Expedite the transition of new assessment staff,
- Address resource constraints associated with security assessments

Because information security assessment requires resources such as time, staff, hardware, and software, resource availability is a limiting factor in the type and frequency of security assessments. Evaluating the types of security tests and examinations the organization will execute, developing an appropriate methodology, identifying the resources required, and structuring the assessment process to support expected requirements can mitigate the resource challenge. This gives the organization the ability to reuse pre-established resources such as trained staff and standardized testing platforms; decreases time required to conduct the assessment and the need to purchase testing equipment and software; and reduces overall assessment costs.

A phased information security assessment methodology offers a number of advantages. The structure is to follow and provides natural breaking points for staff transition. Its methodology should contain minimum the following phases:

Planning: Critical to a successful security assessment, the planning phase is used to gather information needed for assessment execution—such as the assets to be assessed, the threats of interest against the assets, and the security controls to be used to mitigate those threats—and to develop the assessment approach. A security assessment should be treated as any other project, with a project management plan to address goals and objectives, scope, requirements, team roles and responsibilities, limitations, success factors, assumptions, resources, timeline, and deliverables.

Execution: Primary goals for the execution phase are to identify vulnerabilities and validate them when appropriate. This phase should address activities associated with the intended assessment method and technique. Although specific activities for this phase differ by assessment type, upon completion of this phase assessors will have identified system, network, and organizational process vulnerabilities.

Post-Execution: The post-execution phase focuses on analyzing identified vulnerabilities to determine root causes, establish mitigation recommendations, and develop a final report. Several accepted methodologies exist for conducting different types of information security assessments.

2.5 SECURITY ASSESSMENT PLANNING

The core activities involved in planning for an assessment include:

Develop a security assessment policy: Organizations should develop an information security assessment policy to provide direction and guidance for their security assessments. This policy should identify security assessment requirements and hold accountable those individuals responsible for ensuring that assessments comply with the requirements. The approved policy should be disseminated to the appropriate staff, as well as third parties who are to conduct assessments for the organization. The policy should be reviewed at least annually and whenever there are new assessment-related requirements.

Prioritizing and scheduling assessments: Organizations should decide which systems should undergo assessments and how often these assessments should be done. This prioritization is based on system categorization, expected benefits, scheduling requirements, applicable regulations where assessment is a requirement, and resource availability. Technical considerations can also help determine assessment frequency, such as waiting until known weaknesses are

corrected or a planned upgrade to the system is performed before conducting testing.

Selecting and customizing technical testing and examination techniques: There are many factors for organizations to consider when determining which techniques should be used for a particular assessment. Factors include the assessment objectives, the classes of techniques that can obtain information to support those objectives, and the appropriate techniques within each class. Some techniques also require the organization to determine the assessors' viewpoint (e.g., internal versus external) so that corresponding techniques can be selected.

Determining the logistics of the assessment: This includes identifying all required resources, including the assessment team, selecting environments and locations from which to perform the assessment; and acquiring and configuring all necessary technical tools.

Developing the assessment plan: The assessment plan documents the activities planned for an assessment and other related information. A plan should be developed for every assessment to provide the rules and boundaries to which assessors must adhere. The plan should identify the systems and networks to be assessed, the type and level of testing permitted, logistical details of the assessment, data handling requirements, and guidance for incident handling.

Addressing any legal considerations: Organizations should evaluate potential legal concerns before commencing an assessment, particularly if the assessment involves intrusive tests (e.g., penetration testing) or if the assessment is to be performed by an external entity. Legal departments may review the assessment plan, address privacy concerns, and perform other functions in support of assessment planning.

2.6 APPLICATION SECURITY ASSESSMENT AND TESTING SERVICES (APPSATS)

Application Security is an integral part of an information security risk management program. Though organizations focus on 'quality assurance' aspects of the 'software code' during application development, but they seldom focus on 'risk assurance' aspects. This results in insecure products being developed and deployed in the IT infrastructure.

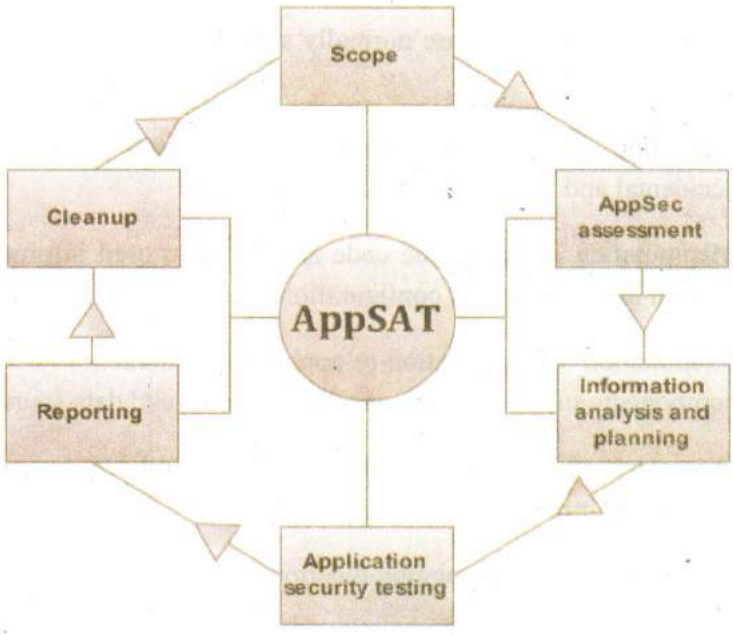


Fig. 1: Steps of Application Security Assessment and Testing Services

Insecure applications contain technical vulnerabilities or errors that are prone to exploitation. More so when the application is used in a banking environment that essentially handles very large amount of transactions. We provide a methodical application security assessment service to ascertain security weaknesses in the product architectural design, application logic and application code.

The importance of ensuring application security is emerging as a mandatory requirement and is insisted by clients, stakeholders as well as the regulators during the last few years. This is due to the fact that applications are compromised by malicious entities, such as hackers or malware to gain access to the confidential data as well as for perpetrating fraud. Surveys indicate that organizations have lost enormous amount of money and attracted customer wrath due to security breaches, exposure of confidential data and loss of critical and sensitive information.

2.6.1 Custom Application Assessment

Application security assessment is a unique area of assessment and penetration testing. Unlike infrastructure based assessments, the methodology utilised by a security professional for identifying security vulnerabilities and significant issues is highly dependent upon the type of application being assessed.

Although several high-level methodologies do exist (and some guides can indeed be quite comprehensive), they are often not generic or versatile enough to cope with the wide variety of custom applications commonly encountered. Many methodologies used by professional security assessment organisations are in fact highly guarded.

In general, the applications are normally subjected to the following groups of tests:

- Inspection of application validation and bounds checking for both accidental and mischievous input.
- Manipulation of client-side code and locally stored information such as session information and configuration files.
- Examination of application-to-application interaction between system components such as the web service and back-end data sources.
- Discovery of opportunities that could be utilized by an attacker to escalate their permissions
- Examination of event logging functionality.
- Examination of authentication methods in use for their robustness and resilience to various subversion techniques.

Regardless of whether it is a web-enabled client-server application or an n-tier compiled application, the methodology actually implemented by the security consultant to assess the security of all client-side functionality will also be subject to the consultants own experience and skill set.

Instead of focusing on an all-encompassing application security assessment methodology, many consultants may find it more practical to cycle through a check-list of questions. The emphasis of the questions is not so much on how you test the application, but more as to what the consultant should be looking for.

2.6.2 Experience Matters

The below list the types of questions commonly thought of or actually asked when assessing a custom application. These questions will aid a consultant in ensuring that all areas of an application have been covered, but the results will always unique to the custom development and the experience of the consultant to interpret the answers into valuable (non-cryptic) advice for the application owner.

For practical use, the questions have been divided into the following groupings:

- Authentication Processes
- Event Logging
- Data Validation and Manipulation
- Client Installation
- Cryptography

2.7 ASSESSMENT TECHNIQUES

The following assessment techniques are proposed (in assessment depth order):

- Application Security Threat Assessment
- Application Security Architecture Review
- Automated External Application Scanning (Tool Based)
- Automated Source Code Analysis (Tool-Based Static Code Analysis)
- Manual Penetration Testing (Security Test Specialist)
- Manual Security-Focused Code Review (Software Security Specialist)

2.7.1 Application Security Threat Assessment

It analyzes application architectural information to develop a threat profile for the application components.

- Identify the nature of the threats – the likely vulnerabilities of the given application and business.
- Estimate the probability that those vulnerabilities might lead to a disruption and the type of impact – both expected and worst case – that might arise.
- Analyze the different consequences and their likelihood of occurrence and determine which should be dealt with and what priority should be attached to mitigate.

While the threats to each asset within the Application can be individually developed and mapped, a more efficient approach is to develop a master list of threat types and identify how these can be used to launch an attack on the Application and/or its components and potentially in turn the business itself. A single threat may exploit a vulnerability to damage different types of assets. Conversely, several threat types may exploit disparate vulnerabilities in order to attack a single critical asset. Given the many-to-many relationships between threats and assets, it is best to use a simple representation of threat to asset mapping by listing threat types by each critical asset identified.

A threat assessment should categorize threat types and threat agents but should focus on malicious threats and will not cover accidental and natural threats. The source of Malicious Threats (the attacker or perpetrator, in this context) can be classified as unauthorized and authorized. Threat vectors and operating environment more fully define how and why a particular "user" falls into one of these categories:

Unauthorize

- An attacker who does not have credentials to legitimately access the application

Authorized

- A legitimate user of the application
- A malicious insider
- A previously unauthorized user who has gained authorized access through compromise of a valid account or user session

While the threat agents may have similar intent, varying reasons for engaging in illegal activities motivates them. Some authorized employees may willfully commit fraud for financial gain or engage in sabotage in order to disrupt routine operations. Theft, vandalism, intentional corruptions and alteration of information are also categorized as malicious threats.

Advantages

- Aids in focusing testing activities and defining true targets in large scale enterprise level applications.
- By using results from a threat assessment, the end-client is able to focus assessment (testing) activities based on business requirements and operational reality.
- Allows vulnerabilities discovered in other assessment types to be weighted and prioritized based on threat probability vs. raw vulnerability finding.

Disadvantages

- Cannot guarantee that all possible security threats will be uncovered.
- End-client must evaluate analysis and determine responses(s).
- Requires existence of application architecture documentation.
- Often confused with Risk Assessments, which weigh asset value, threat, and vulnerability in order to determine business risk. Threat assessments focus on identifying only the threat component (vector(s) and probability).

Recommended use

- Best as a precursor to development activities allowing best use of security resources (i.e., "bake in" security during application

development begets higher return on investment vs. "bolting on" application security fixes post production).

- Best utilized to determine the need and extent for further assessment (testing) activities (e.g., automated testing, expert testing, code review, etc.).

2.7.2 Application Security Architecture Review

Typically a table-top design level review and analysis of the application to identify critical assets, sensitive data stores and business critical interconnections. The purpose of architecture reviews is to also aid in determining potential attack vectors, which could be used in testing. Thus many organizations will blend this activity with Threat Assessment. The review of the application and its interconnections should include both a network- and application-level evaluation. Evaluation should identify areas of potential and actual vulnerability from these levels. This includes areas of vulnerability in the network and application architecture, in addition to a high-level measure of risk of each of those areas.

Advantages

- Aids in focusing testing activities and defining true targets in large scale enterprise level applications.
- Provides more depth to the development rationale of various functionality in the application life cycle.
- By using results from Architecture Review, the assessment (testing) team is able to follow development logic in the "how" and "why" business and security requirements were integrated into application behavior and security controls/constraints.

Disadvantages

- Cannot guarantee identification of all possible security threats.
- Because the scope of an architectural assessment is far more complex than the limited perspective of an application layer assessment, recommendations are similarly complex, sometimes involving "political" changes (i.e., security department's level of involvement in the SDLC).
- Requires on-site discussions with application principals.

Recommended use

- Best as a precursor to development activities allowing best use of security dollars (i.e., incorporate security in beginning begets higher return on investment vs. fixing applications post production).

- Best utilized to determine the need and extent for further assessment (testing) activities (e.g., automated testing, expert testing, code review, etc.).

2.7.3 Automated External Application Scanning

It is utilizing automated open source or commercial software to discover known application layer vulnerabilities.

Advantages

- Able to quickly identify default content, misconfigurations and error conditions resulting from improper input.
- Ability to be run on automated, regular basis to provide baseline and ongoing vulnerability management metrics.
- Can reduce the time required to assess large applications.
- Lower cost to operate/execute (excluding initial capital cost if using commercial testing software).

Disadvantages

- Not designed to determine the extent to which the vulnerabilities can be exploited to compromise the data the application handles.
- Requires appropriate skills/knowledge to use, configure, and interpret results (despite "point and click" vendor marketing claims).
- Requires extensive configuration and customization to emulate a user session.
- Limited ability to correlate events and correlate minor issues that may lead to larger exposure.
- Inability to adapt to dynamic responses in business logic flows.
- Unable to find vulnerabilities that only become apparent after performing a sequence of steps (e.g., business logic flow bypass, session state compromise, etc.).
- No means to interpret security implications, express extent of potential compromise, or identify seriousness of business and information exposure.
- Tools will often be confused by some aspect of an application's behavior, causing a particular class of test to generate a false positive. Because most tools lack intelligence and cannot adapt to such conditions, test results include a large number of identical false positives.

- For valid findings, the tools will often find dozens or hundreds of nearly identical findings, each with the same root cause.

- Tool based findings are only indications of problems. Manual review (by someone well versed in web application testing) is usually necessary to draw the necessary parallels between different findings, and determine which findings identify unique problems, and which represent repetition.

Recommended use

- Handling routine, manually intensive steps (e.g., input validation, field "fuzzing", etc.) in an assessment is useful, particularly in large applications. However, scanner-based results can be used as the starting point from which expert testing can take over.

- Testing less critical applications (i.e., applications that do not contain/handle business sensitive data and/or are not critical to business operation) where the perceived lower risk does not warrant performing hands-on testing.

- Testing critical applications as a prelude to expert testing or code reviews.

If a tool is used, it is important to define the technical limitations of the tool when the findings are presented. An example of this may involve noting the potential for SQL injection attacks based upon verbose SQL error messages, but also the tool's inability to produce the correct syntax needed to extract specific information from the vulnerable database. A failure to convey the differences between automated and expert testing increases the risk that the application assessment results will be interpreted (particularly by management) as being more "secure" than they actually are.

2.7.4 Automated Source Code Analysis

Automated source code analysis involves the use of special software "tools" to conduct Static Code Analysis on software source code to identify potential vulnerabilities within the code.

Advantages

- It can be very effective method for identifying defects in code functionality and syntax errors

- The automated source code analysis tools are highly scalable

- It expedites code review process (can review 400-500 LOC/hr via manual review vs. up to 10,000 LOC/hr using automated source code scanning tools)

- It helps teams focus their manual code review activities on flagged, high-risk areas

Disadvantages

- most automated source code analysis tools generate false positives
- some automated source code analysis tools have a significant learning curve
- some coding languages (such as ASP) are not supported by the tools
- results of automated scans are basically "blueprints" of source code vulnerabilities, so they must be properly protected
- Most of the automated source code analysis tools available on the market today can not identify logic errors, timing errors, resource conflicts, data errors and configuration errors.

Recommended use

- Automated source code analysis tools can be used to enhance and improve an organization's overall efforts to find and eliminate security vulnerabilities within software applications. However, they should not be viewed as a "silver bullet". Manual code reviews must still be conducted to identify false positives and security-related defects that tools cannot identify.

2.7.5 Manual Penetration Testing

Manual Penetration Testing involves application analysis performed by an experienced analyst, usually using a combination of open source automated utilities (either self created or through security community) for performing task-specific functions and hands-on analysis to attempt to further 'hack' the application as an attacker.

Advantages

- Open source tools utilized by the expert tester are developed usually for specific purposes rather than wrapping multiple functionality into a GUI.
- "Expert testing" is an intuitive approach (Business logic analysis), with the ability to identify flaws in business logic that automated scanners are usually inçapable of finding.
- Addresses many of the disadvantages noted in Automated Testing.
- Ability to correlate events and minor issues that may lead to larger exposure. This is a critical advantage of hands-on testing and simulates how a real attacker would operate.
- Ability to communicate complex technical findings in a manner that all can understand. The job is not done until the assessment end-client

understands the nature of the vulnerability discovered and its security implications.

- The analysis phase of expert testing provides "root cause" identification and recommendations based upon the unique, developmental characteristics of an application. Results from an automated tool cannot provide customized, specific recommendations tailored for each application and the end-client's requirements.

Disadvantages

- Often confused with Automated Testing – Marketing of commercial tools uses terms such as "expert" and "intuitive", normally associated with an assessment performed by an experienced analyst.
- Inability of most end-client organizations to determine if tester has appropriate application security skills required. There are no community accepted credentials/certifications established for this area of security.
- Cost to conduct will be higher though there is no upfront capital cost.
- Without establishing clear metrics up front, ongoing testing to determine trends within business environment or iterations of application is difficult.

Recommended use

- Testing critical applications (i.e., applications that do contain/handle business sensitive data and/or are critical to business operation) where the perceived higher risk does not warrant performing automated testing.
- Best suited for B2B, B2C and/or any transactional based and/or multi-level access application.

2.7.6 Manual Security-Focused Code Review

Review of the software source code to identify source code-level issues, which may enable an attacker to compromise an application, system, or business functionality. Web applications in particular are likely to have these vulnerabilities, as they are frequently developed quickly in an environment that does not allow for much security planning and testing. This should not be viewed as simply a generic software audit. Security-focused code reviews should be specifically tailored to find common vulnerabilities in applications.

Advantages

- If integrated into the Software Development Life Cycle (SDLC), coding issues can be resolved earlier in the development process. (less expensive to fix than a post-production revision of the code).

- Provides the ability to identify and address serious coding issues (e.g., input validation, hard-coded credentials, debugging features, etc.) prior to production.
- Enables development teams to identify and correct insecure coding techniques that could lead to security vulnerabilities or possible incidents.
- Helps developers learn from each other and share knowledge on secure coding techniques, best practices and common solutions.

Disadvantages

- It can be very time consuming and costly if automated tools are not available to help augment the review process
- If conducted solely by tools, results will be mostly static signature matching though some tools claim to walk codepath (follow nested loops and other regression through objects).
- Even applications that exhibit strong and thorough software development practices (e.g., documentation, seamless modularity, string checking to prevent buffer overflows, etc.) can fall victim to attack if the underlying business logic doesn't follow security requirements.
- Requires specific programming expertise (e.g., Java, .NET, PHP, etc.) that many application testers do not have.

Recommended use

- Security-focused code reviews should be conducted as part of the normal SDLC. Reviews may also be needed during or after security testing, prior to implementing system upgrades, prior to making system configuration changes, when new security bulletins are released, or immediately following any reported security incidents.

2.8 IDENTIFYING APPLICATION VULNERABILITIES TO PREVENT SECURITY BREACHES

Application security is a frequently overlooked component of a security plan. Developers are under pressure to bring custom applications of all kinds (such as Web applications, customer relationship management systems, accounting systems, etc.) online quickly. This often results in insufficient security testing and validation, leaving the applications vulnerable to exploitation by both internal and external attackers.

These applications are designed to be accessible by customers, partners and employees. They frequently house sensitive data that can be accessed across

networks, via extranets or by anyone over the Internet. Protecting the confidentiality, integrity and availability of this data is crucial. Recent events demonstrate that there is a flourishing underground marketplace for stolen personal information such as credit card numbers, account numbers and Social Security numbers. Much of this information is harvested from unsecured applications, as attackers are increasingly targeting such applications. Without proper security, applications are perhaps the most high-risk component of any network infrastructure. Due to the sensitivity of the information that applications may house, the security of applications can also impact compliance with government and industry regulations.

2.9 CUSTOM APPLICATION AND THEIR SECURITY THREATS

For many organizations, their internal curity professionals are adept at finding and responding to information about the latest vulnerabilities and threats to the software employed within business critical systems under their supervision. There are a great many security resources available, online and printed, ready to help explain and address potential vulnerabilities with the most common commercial software products. However, there exist two problems for those responsible for the security and integrity of your systems. Firstly, the "hit or miss" disclosure of vulnerabilities in commercial software, and secondly, how do you identify or address potential vulnerabilities in the custom (in-house developed) application that connects and runs atop the commercial software?

There is very little most organizations can do about the "hit or miss" disclosure of vulnerabilities inherent to the commercial software deployed throughout the business. You must rely upon the software provider to have initiated appropriate quality controls and security testing, and provide bug-fixes or patches as necessary. Due to various country specific laws or software agreements, it may actually be illegal for your organization to reverse-engineer or otherwise identify security flaws with the software you have purchased.

In the case of custom (in-house developed) applications, it is normally possible to conduct a more thorough security review of the system. There are two paths available to an organization wishing to review the security of their application; a code review and an application assessment. A code review tends towards more of an audit of coding practices utilized throughout the pre-compiled application, while an assessment reviews the functionality and resilience of the compiled application components to real-life threats. While a code review often initially appears to be more valuable, most organizations soon find to their amazement that such a review is extremely costly in both time and effort, and often fails to identify post compilation and deployment issues. An application assessment focuses on the compiled and installed elements of the entire system.

Attention is made to how the application components are deployed, communicate or otherwise interact with both the user and server environments. From past experience, the author has found that application assessments often represent the best value-for-money when identifying security threats and resolutions to distributed client-server applications.

Unfortunately, many organizations will find that they do not have sufficiently skilled resources available internally to cope with an application security assessment, and will have to seek security advice from a security specialist. Not all security organizations or consultancies can provide application security assessments, but those that do will normally classify the application or system into one of two classes; Web and Compiled.

This whitepaper aims to provide an organization with enough information to understand the potential threats to your application beyond classical "ethical hacking" methodologies, and provide helpful advice on steps that can be taken to limit such security threats.

2.10 HACKER'S ATTACKS AGAINST APPLICATIONS

Most initial attacks against your systems are likely to focus upon the systems infrastructure and commercial applications, as that is where most of the exploit material and methodologies are likely to work. Depending upon the skills or resources, if your systems are well secured and up to date with the latest bug-fixes or patches, the attacker may either resort to social engineering (Why try to crack a password when you can just call the helpdesk and have it reset?) or focus upon the manipulation of the vulnerabilities inherent in the application, or sub-components. Although custom applications can be vulnerable to a number of attack methodologies. Four attack categories are most prevalent:

- **Buffer overflow attacks** — Buffer overflow attacks are aimed at application components that take data as an input and pass it to memory buffers for later use and manipulation. Failure to adequately check the size of data before passing it into too small a buffer is commonplace. Attackers may be able to include their own embedded commands within the oversized data package and thus have their commands replace existing application code and execute on the system. If successfully executed, these commands will enable attackers to acquire privileges that exceed authorised permissions up to, and including, those of the system administrator — with corresponding control of the host system. In some circumstances, application components subject to a buffer overflow may suspend or crash, thus denying users further access to the service. In other cases such an attack may bring down the host server and deny access to any services provided by the system.

- **Race conditions** — Under certain circumstances, when an application requires access to specific files, variables or data, its programmers may not have correctly implemented multiple simultaneous accesses and installed the appropriate checks. This can often lead to an attacker enjoying unintended access to files or data through trusted and untrusted server application components.

- **Exploitation of application component privileges** — Server based application components run with specific group or user permissions, not necessarily with that of the user running them (such as an anonymous Web user). These application components, if they suffer additionally from buffer overflows or race conditions, can be used to increase access and escalate the potential damage to the system.

- **Client-side manipulation** — To speed up connectivity and reduce performance loads at the server end, client-side validation of input and manipulation of data is often required. It is often a relatively trivial exercise for attackers to bypass this checking and supply incorrect data or data formats to the server in an attempt to initiate any of the other three common attack formats or to reveal both confidential information and server application functionality. This method is also a popular means of conducting fraudulent transactions on commerce-enabled sites by changing the values of available products.

2.11 WEB BASED APPLICATIONS

As Internet businesses have matured, a new category of application has emerged that enables companies to interact with both potential and existing clients in this medium. These Web-based applications — while they rely on conventional corporate technologies such as database, mail or Web servers — are often subject to design compromises due to the limits of the medium. Such design compromises, unless carefully reviewed and analysed, can expose the integrity of the application components and corporate data to avoidable security risks. In many cases, in fact, application risks far outweigh the threats to the supporting environment from traditional hacking techniques.

To make the best use of client-side Internet bandwidth and to deal with high volumes of simultaneous connections and data requests, Web-based applications tend to be distributed over multiple servers using a tiered architecture model. Moreover, they frequently rely on client-side code to present and manage data. These design considerations, coupled with the use of scripting-type development languages and constant changes in authentication and certification procedures, frequently lead to security flaws in an applications implementation. Increasingly, Internet-based attacks exploit these security flaws to compromise sites and gain access to critical systems.

Direct attacks against custom Web applications through manipulation of their inherent vulnerabilities have become more popular due to their relative ease and the scope they offer for maintaining anonymity. Although companies may install various security mechanisms to strengthen security — firewalls, intrusion detection systems, operating system hardening procedures, etc. — they seldom expend much effort in securing and verifying the integrity of applications and coded pages against external attacks. In these circumstances, simple manipulation of client code or data, such as the price of goods in an online shopping basket application or sending corrupt and incorrect data to the server, can lead to fraudulent transactions or theft of confidential information.

It seems hardly a week goes by without the appearance of fresh newspaper stories describing how faulty application processes and input manipulation have led to the loss of confidential data such as banking details and credit card numbers. In almost all such cases, an understanding of manipulation techniques combined with rigorous client-side security testing would have identified the potential failure points and resulted in a more robust application, thus averting embarrassing, often dangerous compromises of system and data integrity.

2.11.1 The Web-Based Application Security Assessment Process

The process of assessing the security of a web-based application, although not technically complex, often relies upon a multi-facetted approach utilising a variety of technologies and techniques. Unfortunately, there is currently no quick shrink-wrapped solution available to automatically and comprehensively assess an application's security. Various vendors can supply testing products that will search for the most basic faults in non-complex applications/environments and provide advice on better coding practices. Based upon experience in assessing critical Web-enabled applications, automated tools should only be used for first-round security testing and preliminary identification of potential flaws.

Depending on your specific requirements and the type of web-based application, an application security assessment should typically consist of the following phases:

- Examination of external/client-side visible code for information that could be used for social engineering purposes or for information on how an application functions that might be used for a more focused attack.
- Discovery of information on the type of environment that exists at the server side (e.g., embedded SQL queries specific to a single database version).
- Inspection of application validation and bounds checking — both for accidental *and* mischievous input. The purpose of this exercise is to

ascertain the limits of correct server responses when handling unexpected data formats or sizes. This phase involves buffer overflow attempts to establish system resilience and performance continuity.

- Manipulation of client-side code and locally stored information such as cookies and session information. Client-side code is altered to subvert authentication checking and used to establish the bounds of server reliance on client data fields. URL request information and GET/PUT requests are altered to achieve unexpected system responses and access confidential information.
- Examination of application-to-application interaction between system components such as the Web service and back-end data sources. Attempts are made to reference system components by impersonating other system functions or sources. Redirection methods and messaging functions are closely examined.
- Discovery of techniques that could be employed by attackers to escalate their permissions by referencing application components with higher server-side permissions, or exploitation of race conditions to identify lax permission or authentication checking.
- Attempts to subvert in-transit data between the client and server system. Examination of data delivery methods and the likelihood of their subversion or use in a replay-type attack, or other session orientated attacks, including an analysis of system responses to such data.
- Authentication methods in use are examined for their robustness and resilience to various subversion techniques. Attempts are made to bypass authentication processes and/or impersonate valid logged-in users. Detailed studies of user segregation methods are undertaken and an analysis of server-side responses to failed attempts is made.
- Overall examination of the application's deployment and security configuration from perceived threat models. Advice is given on secure deployment methodologies for the application type, based upon market considerations, new vulnerability developments and attack methodologies.

2.11.2 Compiled Applications

Although not as visible or exposed as most web applications, the security threats to compiled applications are very similar and often share the same principles of data integrity and resilience. However, the methodologies and tools necessary for testing the compiled application and sub-components are quite different. While the emphasis of a web application assessment is on manipulation of external client data; for a compiled application, the emphasis is

on the communication methods and data streams between all internal system components.

The great diversity of compiled applications and their interaction with other commercial products (e.g. operating systems, databases, messaging systems and hardware) often means that assessment methodologies must be tuned to system at hand. For this reason, it is vital that both the organisation and the assessment supplier fully understand the scope and reasons for undertaking the security assessment.

2.11.3 General Advice on Securing Custom Applications

For securing applications to minimize the developer's time for correcting potential flaws that may be discovered during a security assessment. Several steps should be undertaken, or at least borne in mind, in order to maximize security and data integrity during the process of building a new application:

1. Assume that someone out there will intentionally try to break your application and may have access to greater resources than you expended in developing it.
2. Assume that you will receive erroneous data from authorised, authenticated users, and develop a way to deal with it.
3. If you choose to use client-side input validation, ensure that the input is also validated at the server end.
4. Avoid complex code and use minimal coding structures for handling performance related issues.
5. Be careful with application component privileges. Always apply the minimum possible permissions needed to carry out any particular task.
6. Utilise methods of load limiting to prevent overloading application components or servers.
7. Check every return code from system components and functions to prevent race conditions or malicious input.
8. Never allow passwords or user specific details to be passed in plain-text to the client browser or between application components.
9. Ensure that confidential system information is not encoded into documents that could potentially be accessed by a remote client, either directly or through escalated application component permissions and calls.
10. Commonly available libraries or sample code may not necessarily be secure. Review carefully any third-party code.

11. Always use full pathnames in system and application procedure calls to prevent redirection and unexpected use.
12. Take extreme care with file permissions and access rights.
13. Remove all unnecessary material from the hosting servers and only install services that are required for the application/system to function.
14. Remove all comments and unnecessary information from client-side code.
15. Distribute application functions between servers when possible to limit data access from a compromised host.
16. Use reasonable system timeouts for network reads and writes to help prevent race conditions being reached and denial of service attacks.
17. Ensure you are capable of logging and tracking all inbound requests for post attack analysis.
18. Where possible, only use shared resources you have direct control over. If this is not possible, ensure appropriate checks for data integrity are made.
19. Test the application thoroughly.
20. Get a third party to perform an independent assessment both of the application's security and the system's host servers before going live.

Check Your Progress 2

Note: a) Space is given below for writing your answer.

b) Compare your answer with the one given at the end of the Unit.

1) Which phases are required for an efficient security assessment methodology?

.....

.....

.....

.....

2) Write short note on Web based application.

.....

.....

.....

.....

3) How hackers execute attacks against application?

.....

.....

.....

.....

4) Write short note on Automated Source Code Analysis.

.....

.....

.....

.....

2.12 LET US SUM UP

The security threats and responses to your infrastructure and commercial software components are commonly understood, and classical security assessment or "ethical hacking" methodologies are adept at increasing the security level of your key infrastructure components. However, custom application code is often untested, and attackers are now focusing upon these security flaws to compromise system components or otherwise gain access to confidential data. For these reasons, organisations should ensure that appropriate security assessments are carried out on all custom, in-house developed, applications.

2.13 CHECK YOUR PROGRESS: THE KEY

Check Your Progress 1

1) Application Security Assessment is designed to identify and assess threats to the organization through bespoke, proprietary applications or systems. Web application security is the security of all components – the web application being used, the web server running the web application and the modules running on the web server. All traffic directed to a web server is http or https traffic, which is legitimate traffic and is therefore not blocked by firewalls. Web servers are frequent targets of attack, prefer to attack web applications for the simple reason that no firewall blocks requests to the web server.

2) **a) High Level Design** Audit identifies and analyzes:

- Flow of information throughout the application environment
- Sensitive data in different sections of the organization

- Threats to the sensitive information in question

b) **Source Code Audit**: In this step the code is reviewed for vulnerabilities and threats that belong to these categories:

- Cryptography
- Authentication
- Session Management
- Data Validation
- Exception Management
- Authorization
- Auditing and Logging

3) **Vulnerability Assessment** is an integral part of any Information Security Risk Management program. 'Critical" vulnerabilities are present in software and systems and thus prone to exploitation, in other words IT systems are prone to **hacking**.

4) Three types of assessment methods can be used to accomplish this—testing, examination, and interviewing.

a) Testing is the process of exercising one or more assessment objects under specified conditions to compare actual and expected behaviors.

b) Examination is the process of checking, inspecting, reviewing, observing, studying, or analyzing one or more assessment objects to facilitate understanding, achieve clarification, or obtain evidence.

c) Interviewing is the process of conducting discussions with individuals or groups within an organization to facilitate understanding, achieve clarification, or identify the location of evidence.

Check Your Progress 2

1) **Planning**: A security assessment should be treated as any other project, with a project management plan to address goals and objectives, scope, requirements, team roles and responsibilities, limitations, success factors, assumptions, resources, timeline, and deliverables.

Execution: Primary goals for the execution phase are to identify vulnerabilities and validate them when appropriate.

Post-Execution: The post-execution phase focuses on analyzing identified vulnerabilities to determine root causes, establish mitigation recommendations, and develop a final report.

2) As Internet businesses have matured, a new category of application has emerged that enables companies to interact with both potential and existing

clients in this medium. These Web-based applications — while they rely on conventional corporate technologies such as database, mail or Web servers — are often subject to design compromises due to the limits of the medium. Such design compromises, unless carefully reviewed and analysed, can expose the integrity of the application components and corporate data to avoidable security risks.

3) Four attack categories are most prevalent:

   a) **Buffer overflow attacks:** Attackers may be able to include their own embedded commands within the oversized data package and thus have their commands replace existing application code and execute on the system.

   b) **Race conditions:** Under certain circumstances, when an application requires access to specific files, variables or data, its programmers may not have correctly implemented multiple simultaneous accesses and installed the appropriate checks.

   c) **Exploitation of application component privileges:** Server based application components run with specific group or user permissions, not necessarily with that of the user running them (such as an anonymous Web user).

   d) **Client-side manipulation:** To speed up connectivity and reduce performance loads at the server end, client-side validation of input and manipulation of data is often required.

4) Automated source code analysis involves the use of special software "tools" to conduct Static Code Analysis on software source code to identify potential vulnerabilities within the code. Automated source code analysis tools can be used to enhance and improve an organization's overall efforts to find and eliminate security vulnerabilities within software applications. However, they should not be viewed as a "silver bullet". Manual code reviews must still be conducted to identify false positives and security-related defects that tools cannot identify.

2.14 SUGGESTED READINGS

- Chris McNab (2007). *Network Security Assessment.* O'Reilly Media, Inc. Second Edition.
- http://viewer.media.bitpipe.com/SecurityInnovation_sSecurity_SO2332654_E Guide_112410.pdf
- http://www.ncat.edu/~davisd/se.html
- Mark Dowd. *The Art of Software Security Assessment: Identifying and Preventing Software Vulnerabilities,* Paperback

UNIT 3 WEB APPLICATION SCANNING AND VULNERABILITY ASSESSMENT

Structure

3.0 INTRODUCTION

For the last decade, considerable resources have been directed at developing web-based applications. These range from simple applications that replace paper-based tasks to home/banking applications for customer convenience to complex applications that attempt to automate lengthy or difficult tasks. As web servers increasingly host more diverse applications, would-be attackers are focusing on them in attempts to gain access to information or resources. With the prevalence of many web application vulnerability classes, these attacks range from nuisance to full compromises of your organization. Since many of these applications are developed in-house, administrators typically cannot rely on public vulnerability databases to determine if their applications are vulnerable. Organizations must look at vulnerability tests specific to the in-house developed applications. Further, it is imperative that organizations analyze all the elements that support web applications.

A web server is a computer that delivers web pages. Every Web server has an IP address and possibly a domain name. For Example, http://www.example.com/path/file.html

Organizations in online industries such as e-commerce, banking and healthcare collect and provide access to data that can be classified as highly confidential. This extremely sensitive information makes a rather tempting target for hackers looking to make their fame by compromising corporate systems and thieving critical data. For the past three years, vulnerable web applications have lead to a number of dangerous exploits such as XSS (cross site scripting) and SQL injection, which count for a substantial amount of reported intrusions. While security begins with proper implementation and configuration, one tool that can help put your organization on a secure path is a software technology known as web application scanning.

3.1 OBJECTIVES

After studying this unit, you should be able to:

- know about web applications and their security issues;
- explain hacking methodology of web applications and different web attacks;
- elucidate the purpose of web application scanning and its importance;
- know about web application security scanners, their features, strengths and weaknesses;
- describe the web application security scanners evaluation criteria and the challenges of web application security scanners; and
- describe web application vulnerability assessment, its conduct and its impact on business organizations.

3.2 WEB APPLICATIONS

A Web application is an application that resides on a company's Web server, which any authorized user can access over a network, such as the World Wide Web or an Intranet.

Most importantly, modern websites and related web applications allow capturing, processing, storage and transmission of sensitive customer data (e.g., personal details, credit card numbers, social security information, etc.) for immediate and recurrent use. And, this is done through web applications. Such features as webmail, login pages, support and product request forms, shopping carts and content management systems, shape modern websites and provide businesses with the means necessary to communicate with prospects and customers. These are all common examples of web applications.

Webmail is a web-page interface used to access e-mail through a Web browser. In order to use Web mail you can get a subscription to an Internet Web mail service

Web applications are, therefore, computer programs allowing website visitors to submit and retrieve data to/from a database over the Internet using their preferred web browser. The data is then presented to the user within their browser as information is generated dynamically (in a specific format, e.g. in HTML using CSS) by the web application through a web server.

Since Web applications reside on a server, they can be updated and modified at any time without any distribution or installation of software on the client's machines – the main reason for the widespread adoption of Web applications in today's organizations.

As the number of businesses embracing the benefits of doing business over the web increases, so will the use of web applications and other related technologies continue to grow. Moreover, since the increasing adoption of intranets and extranets, web applications become greatly entrenched in any organization's communication infrastructures, further broadening their scope and possibility of technological complexity and competency. Web applications may either be purchased off-the-shelf or created in-house.

3.2.1 The Three-Layered Web Application Model

A web application consists of static (regular) and dynamic web pages. A regular web page is one that does not change when a user requests it; the web server sends the page to the requesting web browser without modifying it. In contrast, a dynamic web page is modified by the server before it is sent to the requesting browser. The changing nature of the page causes it to be dynamic.

Java Servlets are platform independent, server side components that provide a powerful mechanism for developing server side programs

The following fig.1 demonstrates the three-layered web application model that describes the working of a web application. The first layer is normally a web browser or the user interface; the second layer is the dynamic content generation technology tool such as Java servlets or ASPs, and the third layer is

the database containing content (e.g., news) and customer data (e.g., usernames and passwords, social security numbers etc.).

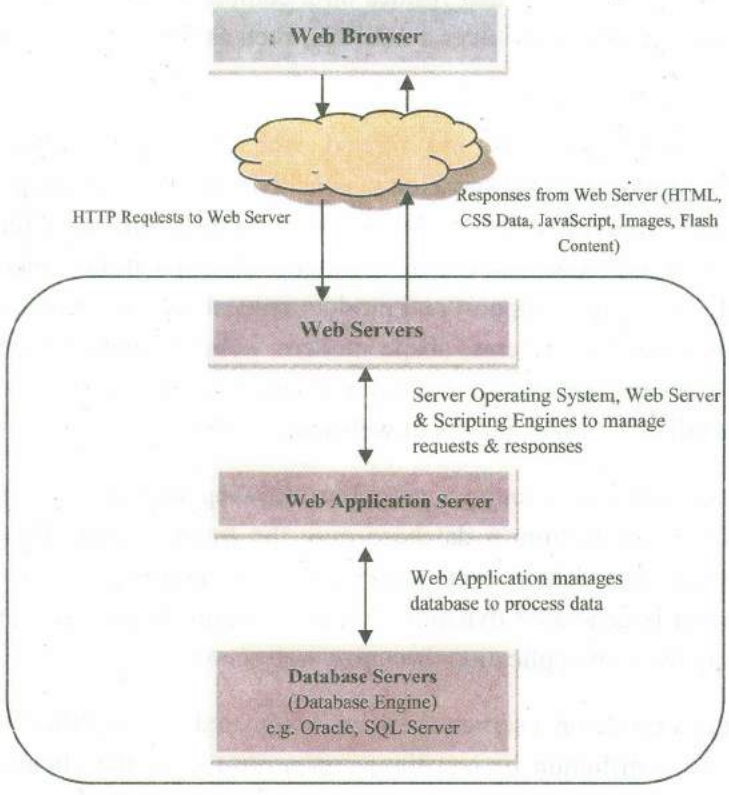


Fig. 1: The Three-Layered Web Application Model

According to this model, the initial request is generated by the user through the web browser over the Internet to the web application server. The web application accesses the database servers to perform the requested task updating and retrieving the information lying within the database. The web application then presents the information to the user through the browser. The step-by-step complete process of activation and successful completion of a transaction by the user over the Internet can be explained as follows:

1. The user submits the shopping cart order on website.
2. The web server receives shopping cart data from the user and transfers it to the web application server.
3. The web application server receives data from the web server and further sends it to the database server where the database is updated.
4. The web application server dynamically generates a web page and sends it to the web server.
5. The web server sends the generated page to the user informing him of his successful transaction.

3.2.2 Types of Web Applications

Two types of web applications can generally be defined:

- **Simple Web Application**
  - Client can request data
  - Usually client cannot modify data stored in back-end data provider
  - Oftentimes no authentication is required for these types of web apps.
- **Complex Web Application**
  - Client can request and modify data
  - Usually client cannot modify data stored in back-end data provider
  - Generally multiple levels of authentication and different authorization levels are applied to user sessions.

3.3 WEB APPLICATION SECURITY ISSUES

Web applications are vital components of any organization, but these applications can be dangerously weak links in your security framework. Hackers love to target web applications - especially when there's opportunity for financial gain.

Despite their advantages, web applications do raise a number of security concerns stemming from improper coding. Serious weaknesses or vulnerabilities, allow hackers to gain direct and public access to databases in order to churn sensitive data. Many of these databases contain valuable information (e.g., personal and financial details) making them a frequent target of hackers. Although such acts of vandalism as defacing corporate websites are still commonplace, nowadays, hackers prefer gaining access to the sensitive data residing on the database server because of the immense pay-offs in selling the data.

As discussed above, it is easy to see how a hacker can quickly access the data residing on the database through a dose of creativity and, with luck, negligence or human error, leading to vulnerabilities in the web applications. In fact, the research firm Gartner estimates that 75 percent of attacks on web security today are aimed straight at the application layer.

As stated, websites depend on databases to deliver the required information to visitors. If web applications are not secure, i.e., vulnerable to, at least one of the various forms of hacking techniques, then your entire database of sensitive

information is at serious risk. Some hackers, for example, may maliciously inject code within vulnerable web applications to trick users and redirect them towards phishing sites. This technique is called Cross-Site Scripting and may be used even though the web servers and database engine contain no vulnerability themselves.

The following fig.2 shows how hacker injects SQL command to database via custom web application: This is done in two ways: SQL commands are entered in the form fields on the webpage, or SQL queries are inserted into required input parameters. Thus, the hacker is able to run SQL queries and commands on the server.

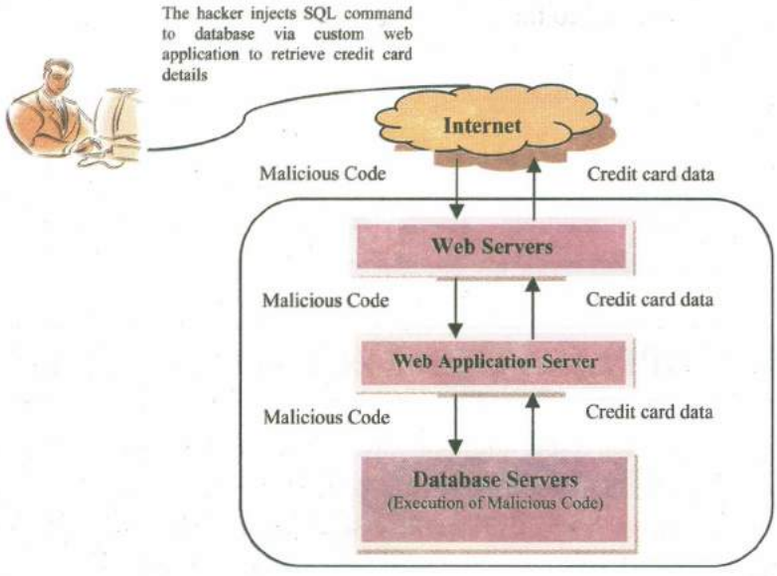


Fig. 2: SQL Injection

The following mentioned steps are performed during SQL injection:

1. The hacker finds vulnerability in the custom web application and sends an attack to the web server.
2. The web server receives the malicious code and sends it to the web application server.
3. The web application server receives the malicious code from web server and sends it to the database server.
4. The database server executes the malicious code on the database. The database returns data from credit cards table.
5. The web application server dynamically generates a page with data including credit card details from the database.
6. The web server sends credit card details to the hacker.

3.3.1 Web Applications Hacking

Unfortunately, hackers have been successful in finding a gaping hole in the corporate security infrastructure, one of which organizations were previously unaware – Web applications.

- By design, websites and related web applications are publically available on the Internet 24 hours a day, 7 days a week, to provide the required service to customers, employees, suppliers and other stakeholders.
- Firewalls and SSL provide no protection against web application hacking, simply because access to the website has to be made public – All modern database systems (e.g. Microsoft SQL Server, Oracle and MySQL) may be accessed through specific ports (e.g., port 80 and 443) and anyone can attempt direct connections to the databases effectively bypassing the security mechanisms used by the operating system. These ports remain open to allow communication with legitimate traffic and therefore constitute a major vulnerability.
- Web applications often have direct access to backend data such as customer databases and, hence, control valuable data and are much more difficult to secure. Those that do not have access will have some form of script that allows data capture and transmission. If a hacker becomes aware of weaknesses in such a script, he may easily reroute unwitting traffic to another location and illegitimately hive off personal details.
- Most web applications are custom-made and, therefore, involve a lesser degree of testing than off-the-shelf software. Consequently, custom applications are more susceptible to attack

Firewalls: A system designed to prevent unauthorized access to or from a private network and can be implemented in both hardware and software, or a combination of both.

SSL (*S*ecure *S*ockets *L*ayer), a protocol developed by Netscape for transmitting private documents via the Internet

Web applications, therefore, are a gateway to databases especially custom applications which are not developed with security best practices and which do not undergo regular security audits. In general, you need to answer the question: "Which parts of a website we thought are secure are open to hack attacks?" and "what data can we throw at an application to cause it to perform something it shouldn't do?"

3.3.2 Examples of Web Application Hacks

The gaping security loophole in Web applications is being exploited by hackers worldwide. According to a survey by the Gartner Group, almost three-fourths of all Internet assaults are targeted at Web applications.

The first reported instance of a Web application attack was perpetrated in 2000 by a 17 year-old Norwegian boy. While making online transactions with a large bank, he noticed that the URLs of the pages he was opening displayed his account number as one of the parameters. He then substituted his account

number with the account numbers of random bank customers to gain access to the customers' accounts and personal details.

On October 31, 2001, the website of Acme Art Inc. was hacked and all the credit card numbers from its online store's database were extracted and displayed on a Usenet newsgroup. This breach was reported to the public by the media and the company lost hundreds of thousands of dollars due to orders withdrawn by wary customers. The company also lost its second phase of funding by a venture capital firm.

Similarly, the 2002 turnover report of a Swedish company was accessed prior to its scheduled publication. The perpetrator simply changed the year parameter in the URL of the previous year's report to that of the present year to gain complete access.

In another 2002 incident, applicants to Harvard Business School accessed their admission status before the results were officially announced by manipulating the online Web application. This third-party Web application was also used by other universities. Upon receiving replies to their applications from these other schools, the applicants examined the URL of the reply and found two parameters that depicted the unique IDs of that school's students. Then, they simply substituted the values in those two parameters in the reply URL with their Harvard IDs, which returned the desired information. This procedure, posted on a businessweek.com online forum, was subsequently employed by over a hundred students eager to know their admission status. When the authorities detected this leakage, these students were denied admission.

In June 2003, hackers detected that the Web applications of the fashion label Guess and pet supply retailer PetCo contained SQL injection vulnerabilities. As a result, the credit card information of almost half a million customers was stolen.

Website defacement is another major problem resulting from Web application attacks. Hackers have learned to modify the source code of many websites. During the 2004 Christmas holidays, the "Santy" worm entered Web application servers, defacing 40,000 websites in a single day. On November 29, 2004, SCO's website logo was replaced by the text, "We own all your code, pay us all your money." Similarly, on December 6, 2004, the homepage of Picasa, the picture sharing facility from Google, was hacked and replaced with a totally blank page.

3.3.3 Liability

Companies face a number of legal implications from Web application attacks and lax security measures. Victoria's Secret, one of the world's leading lingerie manufacturers, was sued in 2005 when details about individual customers' purchases became accessible from its database. The company was directed to

pay a $50,000 fine to New York State and settle all monetary claims by customers. The same method was used in 2005 to access social security numbers and other details of a Tennessee payroll organization. The modus operandi was the same – change the value of the customer ID parameter in the URL.

In 2004, the Federal Trade Commission (FTC) filed judgments against a number of global organizations for privacy and security policy violations when it was discovered that there was a leakage of customer information from company databases caused by Web application intrusions.

For financial as well as legal reasons, it is imperative for companies to make their Web applications totally foolproof.

3.3.4 Hacking Methodology of Web Applications

Hackers have a wide arsenal of attack mechanisms, from which they choose the one most suited to a particular vulnerability. They use a very systematic plan of action. These steps can be classified as:

Study server infrastructure and server OS/type

The hacker first analyzes the properties of the server to be hacked, the operating system running on the server, and the server type. A port scan is then initiated to detect all open HTTP and HTTPS ports to single out the port to be attacked.

Survey the website/application

The hacker examines the website for any loopholes that can be exploited. Loopholes could take the form of feedback or inquiry forms that utilize GET and POST variables that hackers can use to their advantage. The hacker also inspects authentication and logon pages for any chances of accessing the server. The success of this method is evident from the 2000 incident involving the Norwegian boy. He was able to bypass required authentication by bookmarking the target page after going through authentication on his initial visit. A good hacker will go through almost every interactive element on a webpage or website in order to gain access to the server. The hacker also goes through the application script to check for any development glitches that can be exploited.

Authentication is the process of determining whether someone or something is, in fact, wh or what it is declared to be and is commonly don through the use of logon passwords.

Check for presence of input validation

Input validation consists of the validation that most Web applications incorporate to determine whether particular data input is safe and validated. Unsafe data is rejected and not processed further. Laxity in input validation is a prime access pathway for hackers. If they manage to outwit the input validation check post, they can use this path to send malicious inputs to the server.

Mount the attack

After examining the entire scenario, from the server to the application, and isolating all the loopholes and vulnerable target areas, the hacker now mounts the attack.

3.3.5 Types of Web Application Attacks

The various types of Web Attacks are:

SQL injection (SQLi)

The hacker transmits SQL query commands to the database residing on the server via the Web application. This is done in two ways: SQL commands are entered in form fields on the webpage, or SQL queries are inserted into required input parameters. Thus, the hacker is able to run SQL queries and commands on the server.

Cross-site scripting (XSS)

The hacker inserts malicious data into a dynamic webpage. Websites that include only static web-pages have control over user interaction because a static webpage is a "read-only" page that does not permit user interaction. Therefore, a would-be hacker can only view the page without being able to cause any damage. However, a dynamic webpage is open to user interaction, so a hacker can insert hazardous content without the website or Web application being able to differentiate this content from innocuous content. The key to the CSS vulnerability is that a hacker can cause the actual Web server to send a webpage with malicious content to the unsuspecting user. The hacker can then transfer the user's input to another server.

Directory traversal attacks

This attack is also called the ../ (dot dot slash) attack. With this attack, the Web application is manipulated to allow access to files or other resources on the server that are not normally accessible. The attack works by changing the parameter that an application would use to access a certain file. For instance, suppose the value of the parameter includes the path of a particular file. Placing ../ at the beginning of the parameter value forces the application to access the file in the parent directory. By placing a series of ../ and then giving a different file name at the end, a particular file in the root directory can be retrieved.

Parameter manipulation

This involves manipulating data transmitted between the browser and Web application. Parameter manipulation can be carried out in the following ways:

- **Cookie manipulation:** Cookies maintain a certain state in HTTP by storing user preferences and information related to session maintenance. All

cookies can be changed at the client end and then sent to the server with URL requests. Thus, a hacker can easily manipulate the data residing within a cookie.

- **HTTP header manipulation:** HTTP headers consist of control information that is sent from the Web client to the Web server during HTTP requests, and sent from Web servers to Web clients during HTTP responses. Since the HTTP request headers originate from the client, a hacker can easily modify them.
- **HTML form field manipulation:** Form fields contain values of all the check boxes checked, radio buttons selected, text fields filled or any other action by a user on a particular webpage. This data is then sent to the server. Moreover, there can be hidden fields not visible to the user on the page that are sent to the server. A potential hacker can manipulate the form fields to send any value. One example of this manipulation is to simply right-click the mouse on the webpage to view the source code, alter it, save the changes and then reload the page in the browser.
- **URL manipulation:** The HTML forms mentioned above are submitted in a process that requires a certain result to be displayed to the user before the result is displayed on a fresh webpage. The URL of this page will contain all the form field names and their respective values, which can be easily manipulated.

Authentication attacks

The hacker searches for valid authentication to access and enter the server from a Web application. For this kind of attack, a database of usernames and passwords is maintained in order to maximize authentication and thereby obtain access to restricted domains.

Known exploits

The hacker community is very close-knit; newly discovered Web application intrusions are posted on a number of community forums and websites known only to members of that group. These postings are updated on a daily basis and are used to facilitate further hacking.

Directory enumeration

Analyzing the website's entire directory structure, the hacker seeks out hidden directories. These hidden directories could contain administrative data that the hacker may find valuable when launching attacks.

Remote File Inclusion

This is the style of cross-server attack that manipulates an application to load executable content from a third-party server. This method allows an attacker to

execute arbitrary commands with the same privileges as the targeted web server.

Command Execution

A code injection flaw in which an application does not properly sanitize user input, allowing for the injection or arbitrary operating system commands. Such flaws typically allow an attacker to quickly and easily take complete control of a server.

3.4 WEB APPLICATION SCANNING

The rapid evolution of web applications has forced testing techniques to evolve more quickly in an attempt to not only find known vulnerabilities, but also to find the next threat. Early vulnerabilities in web applications such as cross-site scripting (XSS) were considered a novelty by many and not even rated as a serious risk. History has taught us that XSS attacks can cause serious widespread damage, can be trivial to carry out and can be the difference between achieving PCI DSS compliance or not.

Developing and implementing a proper web application assessment methodology can be an extremely laborious and expensive undertaking. No two web applications are the same, so every test must be performed thoroughly as a single vulnerability could lead to a system, network or organization compromise. To compound the problem, the web application resides on a server that must be examined just as thoroughly; a properly secured web application can be compromised just as quickly through an insecure service in the underlying operating system.

Thoroughly testing a web application requires following a detailed methodology based on years of experience. Examining the operating system platform and web server typically falls in the scope of a network assessment, but since they are crucial to the security of the applications they support, it is just as important to examine them. The web server itself may contain numerous application related vulnerabilities such as header information leaks, dangerous HTTP methods, directories that can be indexed, improper use of SSL certificates, weak ciphers and protocols for secure connections. Moving past the basics, an application must be reviewed from several different perspectives that correspond to types of users: unauthenticated user, guest, regular user, administrative user and any custom roles specific to the application.

The way an application performs authentication can be a very complex process; so complex in fact, that many web application scanners have elaborate systems that try to record the transactions that make up the authentication in order to

effectively repeat the process to perform authenticated testing. Even then, these scanners sometimes fail to properly maintain an authenticated state with the application causing hours of scanning results to be unreliable. The authentication sequence must be tested for a variety of issues such as username complexity and predictability, security of credentials, authentication method, password complexity; password reset security, account enumeration, self-provisioned account creation, brute force attacks, one-time passwords, multi-factor authentication, account lockout issues, challenge/response question security and much more.

Once authenticated, users often have access to an immense footprint of custom written application code that is designed to interact with backend systems, databases and users. Issues such as XSS and SQL injection typically become a bigger threat because the application has established a level of trust with the user. The assignment of privileges to the user must be fully tested to ensure the user cannot access portions of the application that are restricted such as administrative functions. The application must be tested for issues such as horizontal privilege escalation, vertical privilege escalation, split responsibility bypass, session termination, session concurrency, session fixation, cookie handling, URL re-writing, Referer header use, data caching, information disclosure, file upload, URL redirection and a number of input validation issues that must be tested for every part of the application for each defined user role.

Throughout this testing, it is important to consider the application's use of technology such as Java applets, Active-X, Flash, streaming media, videoconferencing, instant messaging, e-mail functionality and published documents. Each of these technologies has its own set of tests, concerns and potential vulnerabilities that vary greatly depending on the use and implementation.

The complexity and uniqueness of each web application makes it imprudent to rely solely on an automated vulnerability scanner. A skilled application tester typically finds dozens of vulnerabilities that scanners missed, many of them critical. Automated scanners cannot tell what a page or a variable controls, understand differences in account roles or intuitively guess mistakes a developer may have made. Automated scanners are a useful tool, but only in the hands of a skilled auditor with the experience to validate scanner findings and intuition to locate additional problems. Regardless of how accurate or thorough an application scanner is, it cannot perform all the testing required to perform a comprehensive audit of a web application.

3.4.1 Importance of Web Application Scanning

Organizations need a Web application scanning solution that can scan for security loopholes in Web-based applications to prevent would-be hackers from gaining unauthorized access to corporate applications and data. Web

applications are proving to be the weakest link in overall corporate security, even though companies have left no stone unturned in installing the better-known network security and anti-virus solutions. Quick to take advantage of this vulnerability, hackers have now begun to use Web applications as a platform for gaining access to corporate data.

Check Your Progress 1

Note: a) Space is given below for writing your answer.

b) Compare your answer with the one given at the end of the Unit.

1) What are the different layers of a web application?

.....

.....

.....

.....

2) What are the threats to web applications security?

.....

.....

.....

.....

3) What are different types of web attacks?

.....

.....

.....

.....

4) What is the importance of web application scanning?

.....

.....

.....

.....

3.5 WEB APPLICATION SECURITY SCANNERS

Clearly, Web applications are the biggest Achilles heel in an organization's security strategy. They are much more difficult to protect than traditional

applications that reside behind a firewall. Web application security needs to be stringently checked using an automated Web application security scanner.

A Web Application Security Scanner is an automated tool to test web applications for common security problems such as Cross-Site Scripting, SQL Injection, Directory Traversal, insecure configurations, and remote command execution vulnerabilities by launching a series of Web attacks. A Web application scanner checks for vulnerabilities on the Web server, proxy server, Web application server and even on other Web services. This tool crawls a web application, analyzes in-depth each file it finds, displays the entire website structure and locates application layer vulnerabilities and weaknesses, either by manipulating HTTP messages or by inspecting them for suspicious attributes.

More advanced systems even combine testing with simulated attacks during the scanning process. The average system is vulnerable to thousands of know security risks. A web application scanner identifies these risks and compares them against a continuously updated database.

A large number of web application scanning tools are available, both commercial and open source. Effective use of these tools is an important part of a thorough web application security assessment, and regular security scans are required to comply with security requirements.

3.5.1 Features of Web Application Scanners

The market for web application scanning solutions is expanding fast. While the features vary depending on the product, below are qualities found in almost all web application scanners:

Vulnerability Detection

The main goal a web application scanner is to mitigate the most common threats to web application security. This includes exploits such as cross site scripting that result in data theft and the execution of malicious code as well as techniques like SQL injection that lead to execution of unauthorized commands and tampering. Even the simplest of applications are susceptible to exploit when not properly secured and a web application scanner can help you quickly identify them before disaster strikes.

Vulnerability Prioritizing

Time is of the essence when it comes to protecting your system against sophisticated attacks. A web application scanner with the ability to identify security holes and prioritize the severity of those vulnerabilities can save precious time for researching and mitigating the problem. Today's smaller IT environments usually leave one individual to perform the duties of several. Automated assessment scans can serve benefits to the smallest IT team while reducing the costs and complexities of network security.

Analyze Web Application Infrastructure

Web applications are the most targeted components of a website. However, scanning traditional web applications alone is not enough. The applications of the underlying infrastructure must also be taken into account. A reliable web application scanner will perform critical assessment of vital components such as the operating system, web server, web services and neighboring systems as well.

3.5.2 Strengths and Weaknesses of Web Application Scanners

As with all testing tools, web application security scanners are not perfect, and have strengths and weaknesses too.

Strengths

- The tool can detect vulnerabilities of the finalized release candidate before shipping
- It simulates a malicious user by attacking and probing, and seeing what results are not part of the expected result set
- As a dynamic testing tool, it is not language dependent. A web application scanner is able to scan JAVA/JSP, PHP or any other engine driven web application.

Weaknesses

- Because the tool is implementing a dynamic testing method, it cannot cover 100% of the source code of the application and then, the application itself. The penetration tester should look at the coverage of the web application or of its attack surface to know if the tool was configured correctly or was able to understand the web application.
- It is really hard for a tool to find logical flaws such as the use of weak cryptographic functions, information leakage, etc.
- Even for technical flaws, if the web application doesn't give enough clue, the tool cannot catch them
- The tool cannot implement all variants of attacks for a given vulnerability. So the tools generally have a predefined list of attacks and do not generate the attack payloads depending on the tested web application.
- The tools are usually limited in their understanding of the behavior of applications with dynamic content such as JavaScript, Flash, etc.

3.5.3 Examples of Web Application Security Scanners

The following table *Table-1* shows a list of products and tools that provide web application security scanner functionality:

Table 1: Web Application Security Scanners

S. No.	Vendor Name	Product Name	Features
1	Acunetix	Web Vulnerability Scanner	Provides protection from the following attacks: • CRLF injection attacks • Code execution attacks • Directory traversal attacks • File inclusion attacks • Input validation attacks • Authentication attacks Creates professional security audit reports
2	Application Security Inc.	AppDetective	A network-based vulnerability assessment tool that rates the security strength of applications within your network.
3	Imperva	SecureSphere Dynamic Profiling Firewall	Provides total protection for Web application and Web service attacks, database breaches and worm infections. Incorporates a Web firewall, a database firewall, database auditing, Web services firewall, Intrusion Prevention System (IPS) and a network firewall. Includes one gigabit performance sub-millisecond latency.
4	Kavado	Scando Web application scanner	Detects and eliminates Web application vulnerabilities before exploitation by hackers and thieves. Compatible with every stage of the software life cycle — from development and installation right up to the auditing stage. Works together with the Kavado InterDo firewall to ensure continuous security monitoring of the Web application. Compatible with all Web technologies like Flash, ASP, JavaScript, XML and Web Services. Follows a structured three-stage scanning process: 1. Studies the structure and content of the Web application. 2. Executes dummy hacking instances to detect vulnerabilities.

			3. Displays scan results in systematic reports along with suggestions for remedial solutions.
5	Watchfire	AppScan	Security software that automates the complex, manual task of auditing Web applications.
		AppScan DE	Integrated seamlessly into VS.NET, AppScan DE is a powerful automated unit-testing tool that enables rapid development of secure Web applications.
6	SPI Dynamics	WebInspect	Efficiently detects vulnerabilities in Web applications. Ensures that there's no chance of an attack at any point in the Web application development and implementation of the lifecycle.

3.6 WEB APPLICATION SECURITY SCANNER EVALUATION CRITERIA

The Web Application Security Scanner Evaluation Criteria (WASSEC) is a set of guidelines to evaluate web application scanners on their ability to effectively test web applications and identify vulnerabilities. It covers areas such as crawling, parsing, session handling, testing, and reporting.

The goal of the WASSEC is to create a vendor-neutral document to help guide web application security professionals during web application scanner evaluations. This document provides a comprehensive list of features that should be considered when conducting a web application security scanner evaluation. Different users will place varying levels of importance on each feature, and the WASSEC provides the user with the flexibility to take this comprehensive list of potential scanner features, narrow it down to a shorter list of features that are important to the user, assign weights to each feature, and conduct a formal evaluation to determine which scanning solution best meets the user's needs.

Web Application Security Scanner Evaluation Criteria is given as follows:

3.6.1 Protocol Support

In order to test web applications, a scanner must support all communication protocols that are commonly used by web applications and intermediary network devices. The underlying communication protocol used by web applications is the Hypertext Transfer Protocol (HTTP).

Transport Support

A web application scanner should support the various protocols like HTTP 1.1, HTTP 1.0, SSL/TLS, HTTP Keep-Alive, HTTP Compression and HTTP User Agent Configuration

Proxy Support

It is not uncommon for a scanner to not have direct access to a website. In this situation the scanner should allow the configuration of a proxy. It should be noted that, while sometimes necessary, scanning a web application through proxy can result in false negatives and should be avoided if possible. The various proxy capabilities like HTTP 1.0 proxy, HTTP 1.1 proxy, Socks 4 proxy, and Socks 5 proxy should be supported by a scanner.

3.6.2 Authentication

This section covers support for standard or widely deployed authentication methods that web application scanners should support in order to effectively test applications that require authentication.

Authentication Schemes

A web application security scanner should support the different authentication schemes such as basic, digest, HTTP Negotiate (NTLM and Kerberos), HTML Form-based, Single Sign On, Client SSL Certificates, and Custom implementations within the extensible HTTP authentication framework.

3.6.3 Session Management

During a web application security scan, it is crucial that scanners will maintain a "living" and valid session with the application at all times. A valid application session is mainly required for:

- Web crawling
- Test phase
- Session Management Capabilities
- Session Management Token Type Support
- Session Token Detection Configuration
- Session Token Refresh Policy

3.6.4 Crawling

Crawling is the term used to describe the action taken by a program as it browses from page to page on a website. The crawler will visit a starting page and parse the provided links, crawling to those pages. This process will continue until some user-defined criteria are reached or the process is

completed and there are no more links to crawl. Crawling is essential to a web application security scan - it ensures that the scanner is aware of all linked pages that exist on the website. Crawlers should be as configurable as possible, allowing the user to define a large number of criteria to ensure a thorough and efficient crawl.

Web Crawler Configuration

With regards to crawling, a web application scanner should:

- Provide the user with the option to define a starting URL. While '/' is commonly seen as the home URL of a website, this is not always the case.
- Provide the user with the option to define additional hostnames (or IP addresses) in a list or a range. A website may consist of several domains or sub-domains
- Provide the user with the option to define exclusions for specific hostnames, URL patterns, specific file extensions, and specific parameters
- Provide the user with the ability to limit redundant requests. Large sites with catalogs or photo galleries will often have numerous redundant pages. The capability to tune a crawler to limit the requests to these redundant pages should exist.
- Provide the user with the option of supporting concurrent sessions. The capability should exist to enable support for multiple sessions (logins) or limit crawling to a single session.
- Provide the user with the ability to specify a request delay. Network limitations may require that requests be throttled, while a high bandwidth network may not require this. By setting a delay between requests, network traffic can be limited accordingly.
- Provide the user with the option to define a maximum crawl depth. The 'depth' of a website is a measurement of how many clicks it takes to reach a certain page from the starting page. If you clicked three links, then you are at a depth of three. The user may want to limit their scan to pages that are two or three levels deep.
- Provide the user with a method of training the crawler. The process of training a crawler involves accessing a browser (internal or external to the scanning solution) and manually browsing a website. As the website is browsed, valid pages and form inputs are recorded for the scanner to later use.

Web Crawler Functionality

During operation, a crawler should:

- Identify newly discovered hostnames
- Support automated form submission
- Detect error pages and custom 404 responses
- Redirect support
- Identify and accept cookies
- Support AJAX applications

3.6.5 Parsing

In order to thoroughly scan a web application for security problems, a web application scanner must first map out the web application's structure and functionality. The mapping process is done by the web crawler component, which makes use of different types of content parsers to extract information from web content. This information may include URLs, HTML forms, HTML form parameters, HTML comments, and so forth.

Web Content Types

A scanner should be capable of parsing the following content types to extract information about the application's structure and functionality:

- **HTML:** HTML is the most elementary building block of the World Wide Web. Web application scanners must be able to fully parse and understand HTML content.
- **JavaScript:** JavaScript is probably the most common web client-side scripting language in use today. Web application scanners should be able to parse JavaScript content to extract static and dynamic URLs.
- **VBScript:** Some web applications make use of VBScript as the language of choice for client-side scripting. Although less popular than JavaScript, VBScript content parsing should be supported by web application scanners.
- **XML:** XML content may appear in several scenarios when scanning web applications. For example, SOAP web services messages, Web Service Description Language (WSDL), XML Data Islands, WebDAV requests, XHTML, and so forth. Web application scanners should be able to parse XML content.
- **Plaintext:** Some web applications may store information in plaintext files. Web application scanners should be able to parse plaintext files and to extract relevant information. A common example of a plaintext file that

exists in most web applications and contains application structure information is the Robots.txt file.

- **ActiveX Objects:** Web applications may encapsulate client-side logic as ActiveX controls, which are downloaded and executed by the web browser. Web application scanners should be able to extract application information from such controls.
- **Java Applets:** Similar to ActiveX objects, web developers may choose to encapsulate client-side logic as Java Applets. Applets are small applications that are delivered as Java bytecode and executed on the client-side by the JVM (Java Virtual Machine). Web application scanners should be able to extract application information from Java Applets.
- **Flash:** Adobe Flash is a very popular client-side platform for delivering multimedia content to web browsers. Flash is often used to create interactive web applications with heavy animations and graphics, and was also adapted recently to support building RIAs (Rich Internet Applications). Web application scanners should support extraction of content and information from applications that make use of Flash.
- **CSS:** Cascading style sheets (CSS) is a language widely used in web applications to describe the presentation of HTML documents. Web application scanners should support the extraction of URLs from both inline CSS and external CSS files.

Character Encoding Support

Web applications often include content encoded in forms other than ASCII. A web application scanner should be able to parse and understand content encoded in the following encoding types:

- ISO-8859-1: Defined in RFC 2616 as the default encoding for HTTP content
- UTF-7: 7-bit Unicode Transformation Format
- UTF-8: 8-bit UCS/Unicode Transformation Format
- UTF-16: 16-bit Unicode Transformation Format

Parser Tolerance

Oftentimes, web content may not conform to appropriate standards, either by mistake or due to various types of problems, ranging from bad developer habits to communication problems. In such cases, it is crucial that content parsers will be able to cope with partial or non well-formed content and still be able to extract the relevant information from the application responses. It is important that web applications be at least as robust in their HTML parsing as web browsers are.

Parser Customization

Since web technologies and standards evolve rapidly, web applications may include links and other types of information which is encapsulated in various ways. In order to support web applications that make use of such technologies and standards, it is recommended that web application scanners' parsers will allow user customization for link and content extraction. For example, the following non-standard URL contains parameter names and values, which are concatenated by the string '::'. Each parameter name and value pair use the string '^^' to denote the equal sign:

http://www.some.site/appEntry?p1^^v1::p2^^v2...pN^^vN. In this scenario, the parser should be customizable to understand this non-standard URL format and properly parse the parameter names and values.

Extraction of Dynamic Content (Client-Side Logic Execution)

Due to the dynamic nature of client-side scripting languages, it is often not enough to be able to statically parse scripting languages such as JavaScript, VBScript, or even Flash applications in order to detect links and other relevant application information. In such cases, web application scanners should be able to emulate user interaction with client-side logic in order to dynamically extract this information.

3.6.6 Testing

Testing an application for vulnerabilities is the core functionality of a web application security scanner. This section lists the types of vulnerabilities that a web application scanner should be capable of detecting, as well as the testing-related configuration and customization options that a scanner should provide.

Testing Configuration

It is sometimes important to prevent the web application scanner from testing a given part of a web application. A scanner should provide the ability to reduce its visibility of the web application based on different criteria.

- **Host names or IPs**: The tester should be able to specify specific host names or IP addresses to be ignored by the web application scanner during the test phase. Note that the scanner might be able to crawl and parse these host names and IP addresses for reasons such as logging in and data gathering, but should not test them.
- **URL patterns:** Using patterns (e.g., regular expression, etc.), the tester should be able to specify which URLs should not be tested by the web application scanner.

- **File extensions:** The user should have the ability to specify which extensions to exclude from testing.
- **Parameters**: The user should have the ability to specify specific input parameters to exclude from testing.
- **Cookies:** It is sometimes easier to specify which part of the tested web application the scanner should ignore based on cookie specific values. The tester should then be able to specify one or many cookie values that would prevent the scanner from testing the web page.
- **HTTP headers:** The user should have the ability to configure the scanner to exclude certain pages from testing based on specific HTTP response headers.

Testing Capabilities

A web application scanner is looking for two major types of security problems - vulnerabilities and architectural weaknesses. The following list of problems to test was mostly extracted from the WASC Threat Classification v2.0.

- Authentication
- Authorization
- Client-side Attacks
- Command Execution
- Information Disclosure

Testing Customization

Even if they use the same basic technologies and communication protocols, web applications contain specific vulnerabilities that require very precise payloads to be enabled. A web application scanner should:

- Allow the user to modify existing tests
- Allow the user to create new customized tests

Test Policy

Most web application scanners have a large number of built-in tests, but in many situations, only a subset of these tests will be needed. A web application scanner should allow the user to create personalized test policies that specify which tests to include in a scan.

3.6.7 Command and Control

The Command and Control capabilities of web application scanners can have a significant influence on usability and therefore are an important aspect to

consider when conducting an evaluation. The types of Command and Control features most valued by an end user will vary based on the user's situation - some of the following features will be important to a large enterprise with many users and web applications to scan, but will not necessarily apply to a small company with a single user looking for an effective, low-cost scanning solution.

Scan Control Capabilities

The following control capabilities should be considered during a web application scanner evaluation:

- Ability to schedule scans
- Ability to pause and resume scans
- Ability to view real-time status of running scans
- Ability to define re-usable scan configuration templates
- Ability to run multiple scans simultaneously
- Ability to support multiple users
- Ability to support remote/distributed scanning

Control Interfaces Provided

Web application scanners can provide a variety of interfaces for the user to control the scanner:

- Client Application with GUI:
- Command Line Interface
- Web-Based Interface
- Extensibility and Interoperability
- Scan API
- Ability to integrate with common bug-tracking systems

3.6.8 Reporting

In order for scanning results to be viewed outside of the tool's interface, web application scanners should be able to generate reports of each scan. Because reports are often used by different groups within an organization, scanners should provide the ability to customize the format and information included in their reports.

Types of Reports

Although the specific reporting options may vary, scanners should provide the following types of reports:

- **Executive Summary:** An executive summary provides a concise picture of the scan results. This report allows a reader to able to determine high-level results at a glance.
- **Technical Detail Report:** Scanners must be able to provide all technical information required for readers to reproduce the identified issues.
- **Delta Report:** Scanners should provide the ability to compare results from two or more scans and show differences or trends over time.
- **Compliance Report:** Scanners should provide a report format that allows organizations to quickly determine whether they are in violation of regulatory requirements or other standards. These reporting capabilities should be considered if certain regulations are important to the organization.

Advisories for Each Unique Vulnerability Type

Scanners should have the capability to produce advisories for each unique vulnerability type they identify. These advisories should include vulnerability description, CVE or CWE ID, severity level, CVSS version 2 Score, Remediation guidance and Remediation code example(s).

Report Customization

Scanners should provide the ability to customize their reports, including the following:

- Ability to add custom notes to vulnerabilities
- Ability to mark vulnerabilities as false positives
- Ability to adjust the risk level of vulnerabilities
- Ability to identify and report vulnerabilities according to their content location
- Ability to include customizations

Report Format

Scanners should provide the capability to generate reports in both human and machine-readable formats, including PDF, HTML and XML.

Vendor Feedback

Scanners should provide the ability to automatically report false positives or other types of feedback to the scanner vendor to help improve the quality of future versions of the product. This information should be encrypted in transit and handled securely by the vendor.

3.7 CHALLENGES FACED BY WEB APPLICATION SECURITY SCANNERS

There are many challenges that web application security scanners face that are widely known within the industry however may not be so obvious to someone evaluating a product. For beginners if you think you can just download, install, and run a product against any site and get a report outlining all of its risks you'd be probably wrong. These sorts of tools are often configured as best as possible for most sites, however may not be configured 'out of the box' the best way for your site. In defense of these vendors they can't possibly know every single situation that can occur since site behaviors aren't strictly defined as industry standards, and are often improvised for each site's unique needs. Good tools should allow you to configure various options so that you can properly adjust them for your site.

Below are some of the top issues that these products face which may hinder you from performing an automated assessment against your own site:

Session State Management

Cookies and other state tracking mechanisms are used to track who you are and what you're doing on a site. This is by far one of the most difficult problems facing anyone auditing a web based application. For vendors this is particularly difficult being that developers implement session tracking in their own way. A common problem that automated products face trying to stay 'logged into' a website. When you send an attack against an application parameter you may end up invalidating your session token which can cause you to become logged out. Another problem arises when you have multiple requests sharing a 'session token' being sent at the same time. They often invalidate themselves during the attack phase and you end loosing attack requests, or resending them. Typically the best approach to this problem is to throttle down these products and ensure only 1 request is being sent at a time. The disadvantage of 1 request at a time is the length in which the scan runs for. If you have the patience and need to scan a site requiring logging in, this may be your best bet.

AJAX (Asynchronous Javascript and Xml) technology is a group of interrelated web development methods used on the client-side to create interactive and dynamic web

Script Parsing

As JavaScript, XML, and Flash use continue to grow the ability to identify how they are used gets more difficult. It isn't as simple as just griping a page for links and requesting them anymore. Code may contain conditional behaviour based on the user's actions or environment, dynamically generated links, and perform web based requests with technologies such as AJAX to download additional code. The downloaded code may change depending on the function performed and the order in which it was performed.

Logical Flow

Some websites require a visitor to traverse a site in a certain order before allowing them to perform a function. A good example of this would be the 'checking out' feature of a website and placing an order. Crawlers on the other hand simply fetch a page, identify links in it, and fetch them and don't have the concept of filling the shopping cart before attempting to check it out. The OWASP guys have released vulnerable website package known as WebGoat which points out some of these logical order flaws. At this time websites implementing these sorts of behaviours are best left to the humans.

Custom URLs

Certain websites contain simple URLs such as http://host/foo.php?id=1 while others contain URLs that seem to reach the bounds of one's imagination. One of the biggest challenges that you can face while auditing a web based application is identifying parameters to perform attacks against. Since there isn't any enforcement of what URLs should look like people do whatever they like. Examples of oddball things that people do may include using customer delimiters also known as parameter separators. An example url of index?name=foo&age=12 could be converted to index?name=fooQWEage=12 where QWE is the customer delimiter. Some applications may modify their url structure based off of the value of a single parameter. For example '/index?func=View' may perform a function however if changed to '/index?func=Sell' may require an additional parameter named "id" requiring the user to request '/index?func=Sell&id=12345. If your website doesn't follow 'normal' looking url structuring, manual work may be required.

Privilege Escalation

Wouldn't it would be nice to provide a list of site accounts to someone/thing with their access levels, and get a list of what each account can actually do? It sure would! Unfortunately 'at this time' the only way you're going to pull this off is by using a human. Applications are often to custom to be easily automated in this way which is a substantial limitation facing automated scanning products. Automating escalation testing of certain popular commercial packages (for starters) is something that we hope we'll see in the near future.

False Negatives/Positives

We admit it can be difficult writing a vulnerability signature that will accurately flag an issue without necessarily exploiting it. Certain vulnerabilities are tricky to reproduce every time and because of this may have signatures created based off of behavior, or a version banner. Depending on the vulnerability actively exploiting it could case an application to break or work in undesirable ways. General advice is to not scan production sites but

unfortunately not everyone has this luxury. Due to the nature of certain vulnerabilities, false positives are here to stay.

Check Your Progress 2

Note: a) Space is given below for writing your answer.

b) Compare your answer with the one given at the end of the Unit.

1) What is the purpose of a web Application Security Scanner?

.....

.....

.....

.....

2) What should be the qualities of a web application scanner?

.....

.....

.....

.....

3) Write the weaknesses of web application scanning?

.....

.....

.....

.....

4) What types of challenges are faced by web application scanners?

.....

.....

.....

.....

3.8 WEB APPLICATION VULNERABILITY ASSESSMENT

Sophisticated cyber-criminals supported by organized crime or by nation states are exploiting website vulnerabilities at an alarming rate. Hardened networks have caused the attackers to focus on more vulnerable web application targets. Beyond just stealing identity information (social security and credit card

numbers), cyber-criminals commonly penetrate one of any number of weak spots in an organization's website and silently lace the web pages with malicious code. When users visit the organization's website, their web browser is automatically exploited and their machine loaded with Trojan horses designed to steal passwords, send spam; attack other computers, and more. In April 2008, a single massive hack infected hundreds of thousands of web pages with malicious code using a sophisticated form of blind SQL injection. With 9 out of 10 websites possessing serious vulnerabilities, it's best to know what issues exist before they can be exploited.

Web application vulnerability assessment is performed by an experienced security practitioner who uses a combination of automated vulnerability scanning tools and manual testing techniques to assess the security of the target server, web site or network. Typically this will involve a manual review of the target, so that the automated tools are used in the most efficient and accurate way.

Results from the automated scans are then reviewed and systems can be manually tested to confirm the presence of any security vulnerabilities. This manual interaction significantly reduces the false positives. Finally a report is compiled that outlines the most significant issues along with re-mediation advice on how to resolve the issues discovered.

A web application vulnerability assessment is very different than a general vulnerability assessment where we focus on networks and hosts. In those we scan ports, connect to services and use other techniques to gather information revealing the patch levels, configurations, and potential exposures of our infrastructure. Even "standard" web applications are essentially custom; we need to dig a little deeper, examine application function and logic, and use more customized assessments to determine if a web application is vulnerable. With so much custom code and implementation, we have to rely less on known patch levels and configuration and more actually banging away at the application and testing attack pathways. Custom code means custom vulnerabilities.

3.8.1 Need

The need for web application vulnerability assessment can be realized from the following points:

- A realm of vulnerability scanners are available on the market which look for common and known vulnerabilities and patterns relating to web application security.
- However, the vulnerability scanners do not detect the unknown or less-known vulnerability scanners in specific web applications.

- Web application specific issues such as access control, temporary file, user-defined file, session state, identity credentials, and similar issues are very difficult if not impossible to detect using web application vulnerability scanners. This is where the Web Application Vulnerability Assessments come into play.
- Web Application Vulnerability Assessments rely on both automated and manual discovery in order to detect well-known and less-known security issues in web applications.

3.8.2 The Current Scenario

This situation has become all too familiar with today's e-business enabled enterprises that are at risk. Recent studies say 9 in 10 websites contain serious security issues are now the number one target for malicious hackers. The problem is: When website vulnerabilities are identified by a pen-tester, developer, outsider or whomever, there is always a certain amount of time required to determine the appropriate solution. Resolution could take the form of a software update, configuration change, Web application firewall rule, etc. In any case, the time to fix should be swift because hackers will exploit the websites' vulnerabilities when no immediate remedy is implemented. While the source code is being fixed or system configuration updated, an organization has three options:

1. Take the website down

2. Revert to an older version of the website/code (if it's secure)

3. Stay up while exposed.

The reality is vulnerabilities happen despite the most regimented software development lifecycle. Historically the option number 1 (taking down the website) is employed when an incident has occurred; option number 2 (rolling back the code) is preferable when a hot fix is not back-ported to development and is later overwritten. Practically speaking, the vast majority of website owners default to option number 3 (do nothing), essentially assuming the risk rather than halt business.

Why do so many companies choose not to act? While organizations and their security teams have good intentions, the challenges associated with remediation vulnerabilities in Web applications are daunting. For most, this involves the time consuming process of allocating the proper personnel, prioritization of tasks, QA / regression testing the fix, and finally scheduling a production release. The following figure *fig.-3* illustrates just how long it takes for the average organization to fix some of the most pervasive and widely exploited vulnerabilities.

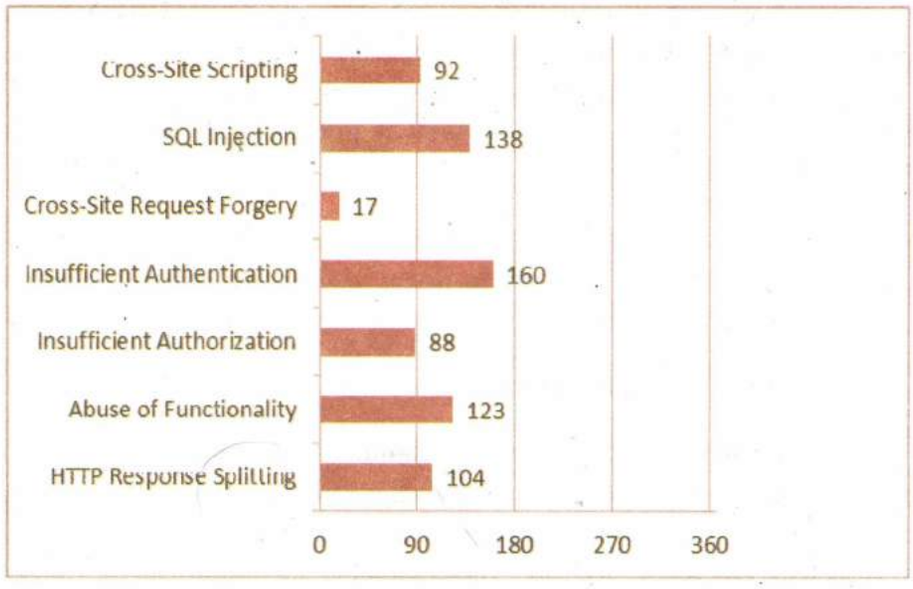


Fig. 3: Average Time to Fix by Class of Attack Measured in Days

Clearly, organizations must become more efficient at identifying Web application security problems, remediate more quickly, and adapt better to new attack techniques. When speaking with IT Security personnel, the issues they voice to disconnect between them and the software development groups. IT Security possesses little control over the security of the website in comparison to their control of the network or hosts. Patches cannot be applied to resolve custom Web application vulnerabilities. So, they must coordinate with development, which typically does not report to them, to get a code fix in place. Also, IT Security has a difficult time explaining the details and associated risk of a vulnerability to this less security savvy audience.

Overcoming these challenges requires a cutting-edge yet pragmatic approach: leveraging the tight integration of precise, comprehensive vulnerability assessments with Web application firewall technology. Such a solution:

1. Measurably improves security;
2. Drastically reduces the time-to-fix from months or years to days or hours;
3. Assists organizations meet industry and governmental regulations, such as PCI-DSS 6.6 compliance;
4. Enables vulnerability assessment results to be immediately actionable;
5. Eases WAF configuration and management, demonstrates due care. And, most importantly provides a revolutionary fourth remediation option to those discussed above:

"Virtual patching" allows IT security professionals to regain control over website security by eliminating vulnerabilities as they are detected without developer intervention.

3.8.3 Components of Web Application Vulnerability Assessment

Every effective vulnerability assessment program requires a cohesive combination of:

1. **People:** Qualified *people* are necessary to carry out day-to-day tasks, manage the technology, and interpret the results to make them meaningful to the business.
2. **Process:** *Process* is required for coordinated efforts between executive management, IT security, and software development groups to share information, prioritize vulnerability fixes, and enable organizational improvements.
3. **Technology:** The right *technology* is essential for consistency, efficiency, and comprehensiveness.

Whether an organization chooses to perform vulnerability assessments with internal resources, a consultancy, or a software-as-a-services vendor, the overall vulnerability program must always account for people, process, and technology. If not, the effort will cost more in time and dollars than it should. Or worse, simply not work.

Still, no matter how perfect any vulnerability assessment product or service, the challenge remains: Any identified custom web application issues must be resolved by the organization—a task inevitably falling to the software developers and not IT security. This is a problem that cannot be solved automatically with a vendor-supplied software patch or new network firewall rule. This is where web application firewalls play a powerful role.

3.8.4 Let's Execute a Web Application Vulnerability Assessment

If an organization is not taking a systematic and proactive approach to web security, and to running a web application vulnerability assessment in particular, then that organization isn't defended against the most rapidly increasing class of attacks. Web-based attacks can lead to lost revenue, the theft of customers' personally identifiable financial information, and falling out of regulatory compliance with a multitude of government and industry mandates: the Payment Card Industry Data Security Standard (PCI) for merchants, HIPAA for health care organizations or Sarbanes-Oxley for publicly traded companies.

For execution purpose, the vulnerability assessment program can be divided into three parts: The *first part* provides an overview of what you need to know to perform a vulnerability assessment to check for web security risks. It'll show you what you can reasonably expect a web application security scanner to accomplish, and what types of assessments still require expert eyes. The *second part* shows how to remedy the web security risks, a vulnerability assessment will uncover (and there'll be plenty to do). The *final segment* explains how to instill the proper levels of awareness, policies, and technologies required to keep web application security flaws to a minimum - from an application's conception, design, and coding, to its life in production. Orientation

Essentials of a Web Application Vulnerability Assessment

A web application vulnerability assessment is the way you go about identifying the mistakes in application logic, configurations, and software coding that jeopardize the *availability* (things like poor input validation errors that can make it possible for an attacker to inflict costly system and application crashes, or worse), *confidentiality* (SQL Injection attacks, among many other types of attacks that make it possible for attackers to gain access to confidential information), and *integrity* of your data (certain attacks make it possible for attackers to change pricing information, for example).

The only way to be as certain as you can be that you're not at risk for these types of vulnerabilities in web security is to run a vulnerability assessment on your applications and infrastructure. And to do the job as efficiently, accurately, and comprehensively as possible requires the use of a web application vulnerability scanner, plus an expert savvy in application vulnerabilities and how attackers exploit them.

Web application vulnerability scanners are very good at what they do: identifying technical programming mistakes and oversights that create holes in web security. These are coding errors, such as not checking input strings, or failure to properly filter database queries, that let attackers slip on in, access confidential information, and even crash your applications. Vulnerability scanners automate the process of finding these types of web security issues; they can tirelessly crawl through an application performing a vulnerability assessment, throwing countless variables into input fields in a matter of hours, a process that could take a person weeks to do manually.

Unfortunately, technical errors aren't the only problems you need to address. There is another class of web security vulnerabilities, those that lay within the business logic of application and system flow that still require human eyes and experience to identify successfully. Whether called an ethical hacker or a web security consultant, there are times (especially with newly developed and deployed applications and systems) that you need someone who has the expertise to run a vulnerability assessment in much the way a hacker will.

Just as is the case with technical errors, business logic errors can cause serious problems and weaknesses in web security. Business logic errors can make it possible for shoppers to insert multiple coupons in a shopping cart - when this shouldn't be allowed - or for site visitors to actually guess the usernames of other customers (such as directly in the browser address bar) and bypass authentication processes to access others' accounts. With business logic errors, your business may be losing money, or customer information may be stolen, and you'll find it tough to figure out why; these transactions would appear legitimately conducted to you.

Since business logic errors aren't strict syntactical slip-ups, they often require some creative thought to spot. That's why scanners aren't highly effective at finding such problems, so these problems need to be identified by a knowledgeable expert performing a vulnerability assessment. This can be an in-house web security specialist (someone fully detached from the development process), but an outside consultant would be preferable. You'll want a professional who has been doing this for awhile. And every company can benefit from a third-party audit of its web security. Fresh eyes will find problems your internal team may have overlooked, and since they'll have helped hundreds of other companies, they'll be able to run a vulnerability assessment and quickly identify problems that need to be addressed.

PCI DSS *(Payment Card Industry & Data Security Standard)*, is a standard that all organizations, including online retailers, must follow when storing, processing and transmitting their customer's credit card data

Conducting the Vulnerability Assessment

There are a number of reasons an organization may need to conduct a vulnerability assessment. It could be simply to conduct a checkup regarding your overall web security risk posture. But if your organization has more than a handful of applications and a number of servers, a vulnerability assessment of such a large scope could be overwhelming. The first thing you need to decide is what applications need to be assessed, and why. It could be part of your PCI DSS requirements, or to meet HIPAA (Health Insurance Portability and Accountability Act of 1996) requirements. Or the scope could be the web security of a single, ready-to-be-deployed application.

Once you've figured out the scope, you need to prioritize the applications that need to be assessed. If you're accessing a single, new application, that decision is easy. But if you're on the precipice of accessing every web application in your architecture, you have some decisions to make. Whether you're looking at the web security of applications you own, or only those that take part in online sales transactions, you need to inventory and prioritize the applications to be assessed.

Depending on the scope and purpose of your vulnerability assessment, it makes sense to start looking at the web security of your crucial applications first - for instance, those that conduct the most transactions or dollar volume - and work

down from there. Or it could be starting with all applications that touch those that process and store sales transactions.

No matter your scope, or the purpose of your vulnerability assessment, other aspects of your architecture always need to be considered when listing and prioritizing your applications. For instance, any externally facing applications - even those that don't contain sensitive information - need to be given high priority. The same is true for externally hosted applications, whether they are Internet-facing or directly connected to back-end systems. Any applications that are accessible by the Internet, or hosted by others, should be subject to a vulnerability assessment. You can't assume that an application is secure just because it is hosted by a third-party, just as you can't assume that just there is no risk just because a web application, form, or entire site doesn't handle sensitive information. In both cases, any web security vulnerabilities could very likely lead an attacker directly to your most critical network segments and applications.

Concluding the Vulnerability Assessment

Now you're ready for the vulnerability assessment. Believe it or not, much of the hard work is already done: deciding the scope, and then classifying and prioritizing your applications. Now, assuming you've already acquired a web security scanner and have identified who will conduct the manual scan for business logic errors, you're ready to take a whack at your application.

The resulting report, based on the security health of the application, will provide you a list of high, medium, and low priority vulnerabilities. At this point, you'll need someone to vet the automated vulnerability assessment results to find any false positives, or vulnerabilities identified by the scanner, but don't actually exist. If it seems overwhelming, don't fret; we'll delve into how to prioritize and remedy these web security vulnerabilities in the next installment. About the same time as your automated vulnerability assessment, the manual assessment will be underway. During the manual assessment, the expert will look for logic errors in the application: Is it possible for users to conduct transactions in ways the developers hadn't anticipated? Such as the ability of someone to tamper with application values that are being passed from the client to the server to alter the price of an item. The manual vulnerability assessment will end with a list of all vulnerabilities to web security found, and the assessor should prioritize the risks posed by each problem - based on the ease of exploiting the vulnerability, and the potential harm that could result if an attacker is successful.

Now you have your list of web security vulnerabilities, both technical and logic. And, if your organization is like most others, you have some remedying work to do. The challenge now is to prioritize what needs to be fixed, so that

your existing applications can be hardened, and those being built can be remedied and safely placed into production.

While the list of web security issues may be long, you've completed the first major phase on the road to a highly secure application. Take comfort in the fact that your vulnerability assessment has identified problems in your applications before they were attacked by competitors, lone-hackers, or organized crime.

3.8.5 Benefits to Users

- Automated tools help the user making sure the whole website is properly crawled, and that no input or parameter is left unchecked.
- Automated web vulnerability scanners also help in finding a high percentage of the technical vulnerabilities, and give you a very good overview of the website's structure, and security status.
- It is only because of automated scanners, you can have a better overview and understanding of the target website, which eases the manual penetration process.

3.8.6 Impacts on Business Organizations

Compliance

Like it or not, security controls are mandated by government regulation, industry standards & requirements, and contractual agreements. We like to break compliance into three separate justifications — industry or regulatory mandated controls (PCI web application security requirements), non-mandated controls that avoid other compliance violations (data protection to avoid breach disclosure), and investments to reduce the costs of compliance (lower audit costs or TCO). The average organization uses all three factors to gauge web application security investments.

Fraud Reduction

Depending on your ability to accurately measure fraud, it can be a powerful driver and justification for security investments. In some cases you can directly measure fraud rates and show how they can be reduced with specific security investments. Keep in mind that you may not have the right infrastructure to detect and measure this fraud in the first place, which might itself provide sufficient justification. Penetration tests are also useful for justifying investments to reduce fraud — a test may show previously unknown avenues for exploitation that could be under active attack or open to future attack. You can use the penetration test to estimate potential fraud and map that to security controls to reduce losses to acceptable levels.

Cost Savings

As we mentioned in the compliance section, some web application security controls can reduce the cost of compliance (particularly audit costs), but there are additional opportunities for savings. Using web application security tools and processes through development and maintenance can reduce the need for and costs of manual processes or controls to remediate software defects and flaws, and may help create general efficiency improvements. We can also calculate cost savings from incident reduction, including incident response and recovery costs.

Availability

When dealing with web applications, we look at both total availability (uptime), and service availability (loss of part of the application due to attack or to repair a defect). For example, while it's somewhat rare to see a complete site outage due to a web application security issue (although it definitely happens), it's not unusual to see an outage of a payment system or other functionality. We also see cases where due to active attack a site needs to shut down some of its own services to protect users, even if the attack didn't break the services directly.

User Protection

While this isn't quantifiable to a dollar amount, a major justification for investment in web security is to protect users from being compromised by their trust in you (yes, this has reputation implications, but we cannot precisely quantify them). Attackers frequently compromise trusted sites not to steal from that site, but to use it to attack the site's users. Even if you aren't concerned with fraud resulting in direct losses to your organization, it's a problem if your web application is used to defraud your users. Most organization derive value or direct revenue from customer data, and there is an implied custodial duty to protect the information you have gathered and use.

Reputation Protection

While many models attempt to quantify a company's reputation and potential losses due to reputation damage, the reality is all those models are bunk — there is no accurate way to measure the potential losses associated with a successful attack. Despite surveys indicating users switch to competitors if you lose their information, or that you'll lose future business, real world reports show that user behavior rarely aligns with survey responses. For example, TJX was the largest retail breach notification in history, yet sales went up after the widely reported incident. But just because we can't quantify reputation damage doesn't mean it isn't an important factor in justifying web application security. Just ask yourself (or management) how important that application is to the public image of your organization, and how willing you or they are to accept

the risk of losses ranging from defacement to lost customer information to downtime. User, investor, and partner trust in your company and services is complicated, and while it does not track directly with site security, trust remains important to the overall value of the business.

Breach Notification Costs

Aside from fraud, we also see direct losses associated with breach notifications (if sensitive information is involved). Ignore all the fluffy reputation/lost business/market value estimates and focus on the hard dollar costs of making a list, sending out notifications, and staffing the call center for customer inquiries. You might also factor in the cost of credit monitoring, if you'd offer that to your customers.

You will know which combination of these works best for you based on your own organizational needs and management priorities, but the key takeaway should be that you likely need to mix quantitative and qualitative assessments to prioritize your investments. If you're dealing with private information (financial/retail/healthcare), compliance drivers and breach notification mixed with cost savings are your best option. For general web services user protection & reputation, fraud reduction and availability are likely at the top of your list. And let's not forget that many of these justifications are just as relevant for internal applications. Whatever your application, there is no shortage of business (as opposed to technical) reasons to invest in web application security.

Check Your Progress 3

Note: a) Space is given below for writing your answer.

b) Compare your answer with the one given at the end of the Unit.

1) How a web application vulnerability assessment differs from a general vulnerability assessment?

.....

.....

.....

.....

2) What are the different components of a web application vulnerability assessment?

.....

.....

.....

.....

3) What steps are required to be performed while conducting a web application vulnerability assessment?

.....

.....

.....

.....

4) What type of benefits a business organization can reap by using web application scanning and vulnerability assessment tools?

.....

.....

.....

.....

3.9 LET US SUM UP

Web applications are vital components of any organization, but these applications can be dangerously weak links in your security framework. Hackers love to target web applications -especially when there's opportunity for financial gain.

Traditionally, the most common solution has been to test applications during the development stage. However, a large majority of these applications are developed by third-parities, not the organizations that are actually use them. This isn't all corporations must worry about as the underlying operating system platform, desktop applications and the databases that interact with those web applications all serve as entry points and potential security risks. Where the traditional testing methods fail, web application scanners offer a more robust and full testing measure.

Web Application Scanner is an automated program to test web applications for common security vulnerabilities like Cross-Site Scripting, SQL Injection, directory traversal, insecure configurations, and remote command execution vulnerabilities by launching a series of Web attacks. A Web application scanner checks for vulnerabilities on the Web server, proxy server, Web application server and even on other Web services.

The purpose of web application vulnerability assessment is to

- Proactively detect vulnerabilities in elements of deployable or deployed information systems and/or networks before those vulnerabilities are

exploited (by contrast with tools that are used to forensically analyze such systems/networks after an intrusion or compromise);

- Analyze all detected vulnerabilities to assess their potential impact on the security posture of the system/network element in which the vulnerabilities are found, and quantify the level of risk that impact poses on the overall system/network.

3.10 CHECK YOUR PROGRESS: THE KEY

Check Your Progress 1

1) A Web application is a three-layered application. Normally, the first layer would be a Web browser, the second would be a content generation technology tool such as Java servlets or ASP (Active Server Pages), and the third layer would be the company database.

2) In the absence of secure coding and testing practices during the development process, Web applications contain vulnerabilities that are easily exploited by attackers to bypass traditional security measures and damage your organization. Because of this, it is essential for organizations to find weaknesses in their Web applications and remediate them.

3) The various types of Web Attacks are:

   - SQL injection (SQLi)
   - Cross-site scripting (XSS)
   - Directory traversal attacks
   - Parameter manipulation
     - Cookie manipulation
     - HTTP header manipulation
     - HTML form field manipulation
     - URL manipulation
   - Authentication attacks
   - Known exploits
   - Directory enumeration
   - Remote File Inclusion
   - Command Execution

4) Organizations need a Web application scanning solution that can scan for security loopholes in Web-based applications to prevent would-be hackers from gaining unauthorized access to corporate applications and data. Web

applications are proving to be the weakest link in overall corporate security, even though companies have left no stone unturned in installing the better-known network security and anti-virus solutions. Quick to take advantage of this vulnerability, hackers have now begun to use Web applications as a platform for gaining access to corporate data.

Check Your Progress 2

1) A web application security scanner can facilitate the automated review of a web application with the expressed purpose of discovering security vulnerabilities, and are required to comply with various regulatory requirements. Web application scanners can look for a wide variety of vulnerabilities, including:

   - Input/Output validation: (Cross-site scripting, SQL Injection, etc.)
   - Specific application problems
   - Server configuration mistakes/errors/version

2) The market for web application scanning solutions is expanding fast. While the features vary depending on the product, below are qualities found in almost all web application scanners:

   - Vulnerability Detection
   - Vulnerability Prioritizing
   - Analyze Web Application Infrastructure

3) Weaknesses of web application scanning:

   - Because the tool is implementing a dynamic testing method, it cannot cover 100% of the source code of the application and then, the application itself.
   - The penetration tester should look at the coverage of the web application or of its attack surface to know if the tool was configured correctly or was able to understand the web application.
   - It is really hard for a tool to find logical flaws such as the use of weak cryptographic functions, information leakage, etc.
   - Even for technical flaws, if the web application doesn't give enough clue, the tool cannot catch them
   - The tool cannot implement all variants of attacks for a given vulnerability. So the tools generally have a predefined list of attacks and do not generate the attack payloads depending on the tested web application.

- The tools are usually limited in their understanding of the behavior of applications with dynamic content such as JavaScript, Flash, etc.

4) There are many challenges that web application security scanners have to face. Below are some of the top issues that these products face which may hinder you from performing an automated assessment against your own site:

   - Session State Management
   - Script parsing
   - Logical Flow
   - Custom URLs
   - Privilege Escalation
   - False Negatives/Positives

Check Your Progress 3

1) A web application vulnerability assessment is very different than a general vulnerability assessment where we focus on networks and hosts. In those we scan ports, connect to services and use other techniques to gather information revealing the patch levels, configurations, and potential exposures of our infrastructure. Even "standard" web applications are essentially custom; we need to dig a little deeper, examine application function and logic, and use more customized assessments to determine if a web application is vulnerable. With so much custom code and implementation, we have to rely less on known patch levels and configuration and more actually banging away at the application and testing attack pathways.

2) Components of a web application vulnerability assessment program:

   - People: Qualified *people* are necessary to carry out day-to-day tasks, manage the technology, and interpret the results to make them meaningful to the business.
   - Process: *Process* is required for coordinated efforts between executive management, IT security, and software development groups to share information, prioritize vulnerability fixes, and enable organizational improvements.
   - Technology: The right *technology* is essential for consistency, efficiency, and comprehensiveness.

3) Steps to be performed while conducting the web application vulnerability assessment:

- Deciding the scope
- Classifying and prioritizing the web applications
- Acquiring a web security scanner
- Identifying the expert/practitioner who will conduct the manual scan for business logic errors
- Readiness to take a whack at your application

4) Benefits to business organizations

- Compliance
- Fraud Reduction
- Cost Savings
- Service Availability
- User Protection
- Reputation Protection
- Breach Notification Costs

3.11 SUGGESTED READINGS

- http://www.hacker4lease.com/
- http://www.linuxforu.com/how-to/vulnerability-assessment-have-you-done-it-yet/
- http://www.lwn.net/Articles/381160/
- http://www.petri.co.il/importance_of_web_application_scanning.htm
- http://www.projects.webappsec.org/
- http://www.sectools.org/web-scanners.html

UNIT 4 WEB APPLICATION ETHICAL HACKING

Structure

4.0 INTRODUCTION

The horizon of web applications is increasing day by day, thus its popularity and utility also. But security concerns are also very important components of the web applications. Also hacking is a threat for the users who travel across the cyber world for their personal and professional reasons. This unit covers the concerns of security with web applications.

4.1 OBJECTIVES

After studying this unit, you should be able to describe:

- Web applications;
- Ethical hacking;
- Security concerns;
- Session ID; and
- Challenges of web services.

4.2 ABOUT WEB APPLICATIONS

From the past many years, the millions of businesses have started using web which is an inexpensive channel to communicate and exchange information with customers.

The web provides marketers' a way to know about the people who are trying to visit their sites and generally started communication with them. One of defined ways of doing so is to ask web visitors to subscribe newsletters or to submit an application form while requesting information for the products or may provide details to modify their browsing experience.

The web, an excellent sales channel for a countless organizations whether large or small having more than 1 billion Internet users today (source: Computer Industry Almanac, 2006), US e-commerce spending accounted for $102.1 billion in 2006 (Source: comScore Networks, 2007).

All this data is being captured, stored, processed and transmitted to be used immediately or later for some work. Web applications, in the form of submit fields, enquiry and login forms, shopping carts, and content management systems that make these widgets to happen.

This makes the businesses to influence the outside world with their presence online and creating long-lasting and profitable relationships with prospects and customers.

The web applications have become a ubiquitous phenomenon. Having highly technical and complex nature still they are widely unknown and grossly misunderstood fixture in our everyday cyber-life.

Web applications defined

From the technical point of view, web is a highly programmable environment that allows mass customization by deploying a large and diverse range of applications, to millions of global users. The two major components existing in a modern website are the flexible web browsers and the web applications which are available to all at no expense.

Users are able to get data and interact with web pages within a website through the help of Web browsers.

Today's websites are far more than the static text and graphics showcases of the early and mid-nineties in which modern web pages provide personalized dynamic content to users according to their individual preferences and settings. The web pages may also run client-side scripts that "change" the Internet browser into an interface for the web mail and interactive mapping software applications.(e.g., Yahoo Mail and Google Maps).

Modern web sites are importantly allowing to capture, process, store and transmit delicate data of customer (e.g., personal details, credit card numbers, social security information, etc.) for immediate and recurrent use by using web applications. The features such as web mailing, login pages, support and product request forms, shopping carts and content management systems, improves modern websites and make communication possible between businesses and their prospects and customers. These are all common examples of web applications.

Web applications are the computer programs which allow website visitors to submit and retrieve data to/from a database over the Internet using their preferred web browser. The web application using a web server present the data to the user within their browser as information is generated dynamically in a specific format i.e. in HTML using CSS.

The technical oriented Web applications query the content server and then dynamically generate the web documents to serve their users. The provided web documents are in a standard format supported by all browsers (e.g., HTML or XHTML). JavaScript is a client side script that provides dynamic elements on each page. The web browser is a key that interprets and runs all scripts, while displaying the requested pages and content. The web browser is described as the "universal client for any web application".

One more benefit of building and maintaining web applications is that they perform their function without worrying about the operating system and browsers running on the client side. Web applications gets easily deployed quickly anywhere at no cost and without any installation requirements at the user's end.

The number of businesses enjoys the benefits of doing business over the web as the use of web applications and related technologies grows and the increase in use of intranets and extranets helps web applications to firmly established in any organization's communication infrastructures, and increases the scope and possibility of technological complexity and prowess.

Web applications may either be purchased off-the-shelf or created in-house.

How do web applications work?

The three-layered web application model shows:

- The first layer - a web browser or the user interface;
- The second layer - the dynamic content generation technology tool such as Java servlets (JSP) or Active Server Pages (ASP), and

- The third layer- the database containing content (e.g., news) and customer data (e.g., usernames and passwords, social security numbers and credit card details).

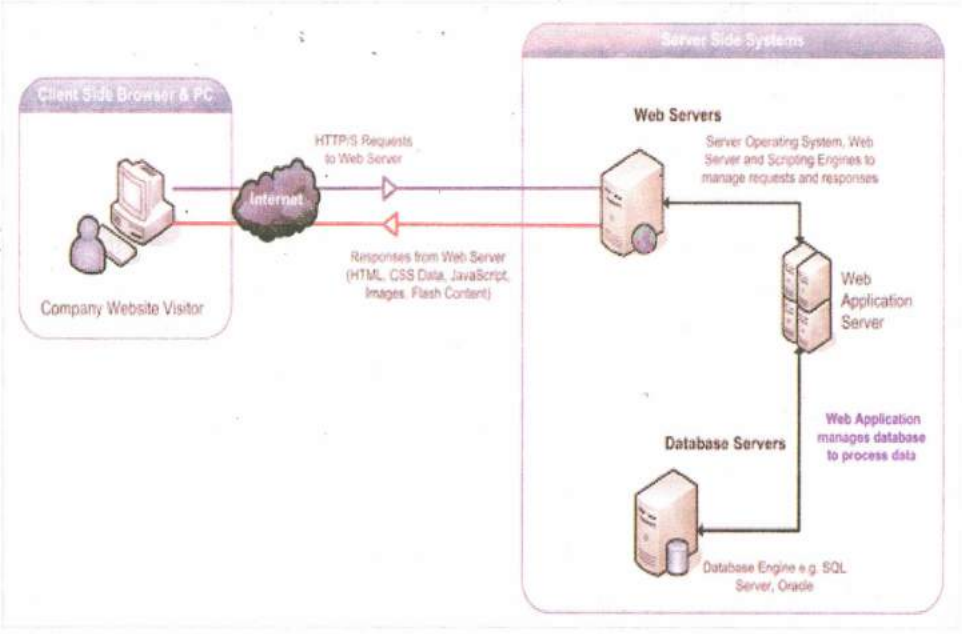


Fig. 1

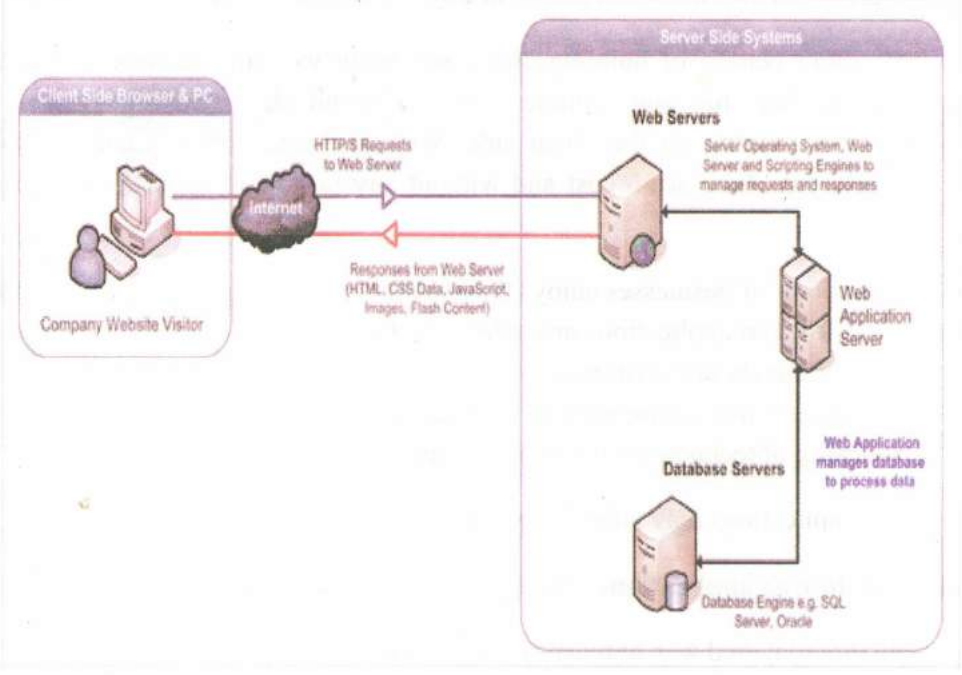


Fig. 2

In below figure the initial request is being triggered by the user using the browser over the Internet to the web application server. The web application accesses the databases servers to complete the required task and retrieves the

information which is their in the database and shows the available information to the user through the browser.

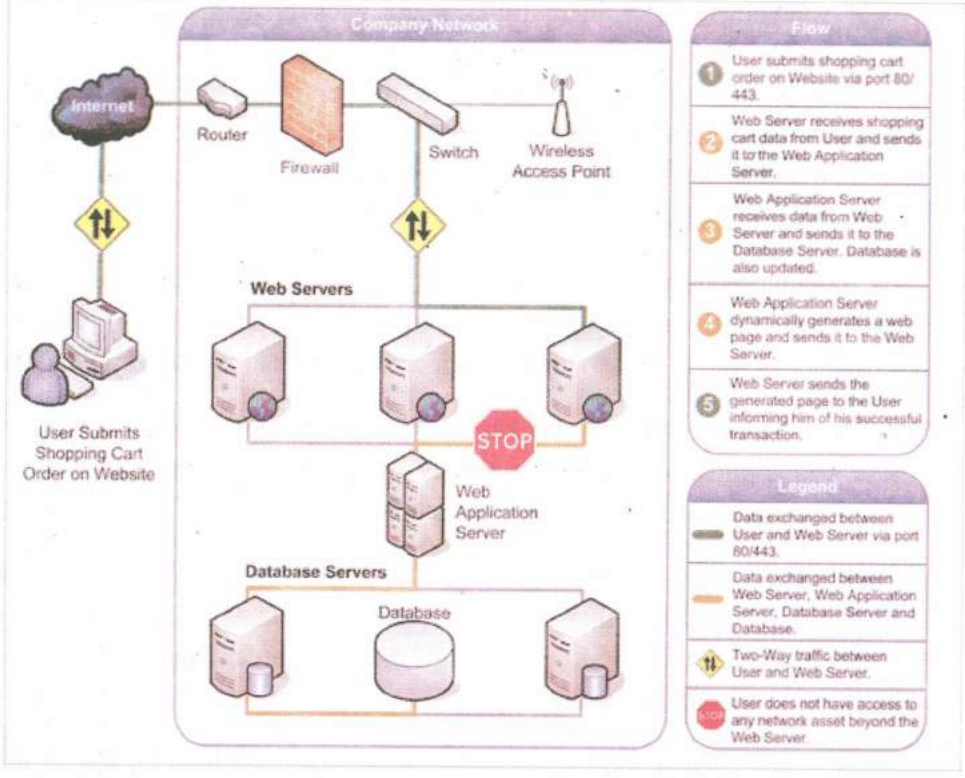


Fig. 3

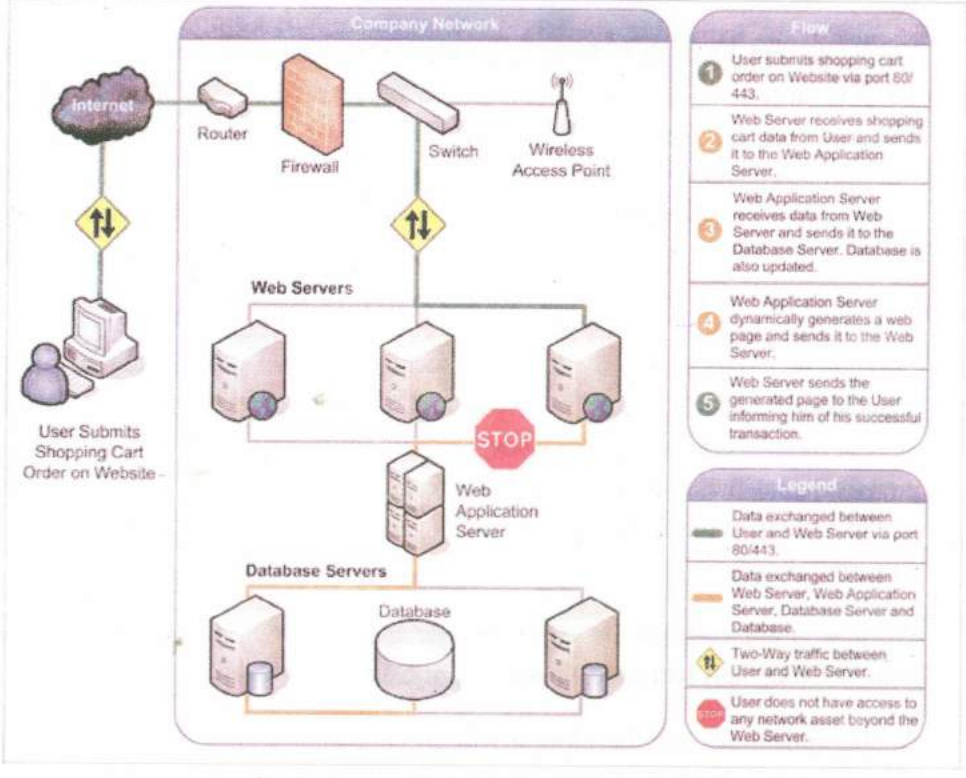


Fig. 4

Web Security Issues

Web applications also increase the security concerns due to improper coding. Major weaknesses or vulnerabilities make hackers to get direct and public access to databases in order to misuse the sensitive data. Many of these databases having valuable information i.e. the personal and financial details are targeted by hackers. These acts of vandalism as defacing corporate websites are common and hackers accesses the sensitive data residing on the database server due to the immense pay-offs in selling the data.

The framework shows the easiness of a hacker to get access the data quickly which is residing on the database through some creativity or by luck or by negligence or due to human error which creates mess in the web applications.

Websites are depending on databases to deliver the information to visitors. If these web applications are not secure by various forms of hacking techniques, then your entire database of sensitive data and information is in danger.

Some hackers may inject false code within vulnerable web applications to make users to redirect them towards fishy sites which is called Cross-Site Scripting and may be used even though the web servers and database engine having no vulnerability in themselves.

Websites and related web applications are available 24 hours a day, 7 days a week providing the services to customers, employees, suppliers and other stakeholders.

Firewalls and SSL do not make protection against web application hacking, because access to the website has to be made public – All modern database systems like Microsoft SQL Server, Oracle and MySQL may get accessed through specific ports (e.g., port 80 and 443) and anyone makes direct connections to the databases easily by passing the security mechanisms used by the operating system. These ports are always open to allow communication with legitimate traffic and therefore bring risk.

Web applications also provide direct access to backend data such as customer databases therefore it is difficult to secure and control valuable data. Those that do not have access will have some form of script that allows data capture and transmission. If a hacker becomes aware of these types of weaknesses in such a script, he easily reroute unwitting traffic to another location and illogically gets personal details.

Most web applications are custom-made and require a lesser degree of testing than off-the-shelf software. Custom applications are more prone to attacks.

Web applications are therefore a way to databases and to custom applications which do not have best security and which do not undergo regular security audits. Generally a question arises that "Which parts of a website are secure are

open to hack attacks?" and "what data can be thrown at an application to cause it to perform something it shouldn't do?" This is the case of a web vulnerability scanner.

4.3 ETHICAL HACKING

Ethical hacking is termed as hacking the confidential information. The information is not secure by ethical hacking. This is also known as the intrusion testing, penetration testing or red teaming. However it also gives professional certification to the certified ethical hacker where the hacking of the computer system or some other devices is done. The service is made available to the people by the international council of e-commerce consultants.

The user has to be an ethical hacker a name given to the person who does ethical hacking and for this the person has to be a penetration tester. The ethical hacker performs different activities; his main task is to work for an organization for penetrating the information from different networks or systems. The organization trusts him as he provides different services to the firm.

The ethical hacking service is same as hacking and the ethical hacker also works same as the hacker works for different purposes. The ethical hacker is said to be a computer expert and works for the networking systems. He is the person who works on behalf of the members of the organization. However, the hacking service that is being provided by the hacker can become dangerous for the firm and can exploit the systems of the company.

Check Your Progress 1

Note: a) Space is given below for writing your answer.

b) Compare your answer with the one given at the end of the Unit.

1) Write a note on web applications?

.....

.....

.....

.....

2) What are the three layers used in the working of web application.

.....

.....

.....

.....

3) Write a note on web security issues?

.....

.....

.....

.....

4) Define ethical hacking?

.....

.....

.....

.....

4.4 SECURITY

Security means protecting assets. The Assets may be tangible items, such as a Web page or your customer database or may be less tangible, such as your company's reputation.

Security is a path and it is not a destination. While analyzing the infrastructure and applications, potential threats are identified and present a degree of risk. Security is about risk management and implementing effective countermeasures.

Security relies on the following elements:

Authentication

Authentication speaks about who are you? This is the process of uniquely identifying the clients of your applications and services. These might be end users, other services, processes, or computers. Authenticated clients are referred to as principals in security parlance.

Authorization

Authorization says about what can you do? In this process the resources and operations authenticates the client to permitted access. These resources consists of files, databases, tables, rows, and so on, together with system-level resources such as registry keys and configuration data. Operations consist of doing transactions such as purchasing a product, transferring money from one account to another, or increasing a customer's credit rating.

Auditing

Effective auditing and logging refer to non-repudiation. Non-repudiation guarantees a user cannot deny performing an operation or initiating a

transaction. For an example, in an e-commerce system, non-repudiation mechanisms make sure that a consumer cannot deny ordering 100 copies of a particular book.

Confidentiality

Confidentiality is the process of making sure that data remains private and confidential, and that it is not in the hands of unauthorized users or eavesdroppers who monitor the flow of traffic across a network. Encryption is used to enforce confidentiality. Access control lists (ACLs) are termed as enforcing confidentiality.

Integrity

Integrity explains that data is saved from accidental or deliberate (malicious) modification. It is a key concern, particularly for data passed across networks. It is typically provided by using hashing techniques and message authentication codes.

Availability

Availability means that systems remain available for legitimate users. The goal for many attackers with denial of service attacks is to crash an application or to make sure that it is sufficiently overwhelmed so that other users cannot access the application.

Threats, Vulnerabilities, and Attacks Defined

A threat is any potential occurrence that is malicious and could harm an asset. It is said to be any bad thing that can happen to your assets.

Vulnerability is a flaw that makes a threat possible. It may be of poor design, configuration mistakes, or inappropriate and insecure coding techniques. Weak input validation is an example of an application layer vulnerability, which creates input attacks.

An attack is an action that exploits vulnerability or enacts a threat. Examples of attacks include sending malicious input to an application or flooding a network in an attempt to deny service.

In summarize form we may said that a threat is a serious event that can adversely affect an asset, whereas a successful attack exploits vulnerabilities in your system.

4.5 SESSION ID

The "strength" of the session ID is an important aspect of managing state within the web application. It is used to track an authenticated user through the

application. The organizations must know that this session ID must fulfill a particular set of criteria it is not to be compromised through predictive or brute-force type attacks. Randomness and length are the two major characteristics of a good session ID.

Session ID Randomness

The session ID is a non judgmental application consists of a strong method of generating random ID's. It is necessary to use a cryptographically strong algorithm to generate a unique session ID for an authenticated user. Ideally the session ID should be a random value. Do not use linear algorithms based upon predictable variables such as date, time and client IP address.

The session ID should fulfill the following criteria:

It must look random to pass statistical tests of randomness.

It must be unpredictable to predict what the next random value will be by giving complete knowledge of the computational algorithm or hardware producing the ID and all previous ID's.

It cannot be reliably reproduced, if the ID generator is used twice with exactly the same input criteria, the result will be an unrelated random ID.

Session ID Length

It is vital to have session ID to be of sufficient length and brute force method could be used to make it infeasible to derive a valid ID within a usable timeframe. With the Given current processor and bandwidth limitations, session ID's consists more than 50 random characters in length and can be make longer if the opportunity exists.

Factors on Which the Actual Length of the Session ID Depends

Speed of connection has a big difference between Internet client, B2B and internal network connections. An Internet client have less than 512 kbps connection speed while an internal user may be capable of connecting to the application server 200 times faster. Therefore an internal user could potentially obtain a valid session ID in 1/200th of the time.

Complexity of the ID implies the values and characters used within the session ID. Moving from numeric values (0-9) to a case-sensitive alpha-numeric (a-z, A-Z, 0-9) range means that, for the same address space, the session ID becomes much more difficult to predict. For example, the numeric range of 000000-999999 could be written as 0000-5BH7 using a case-sensitive alpha-numeric character set

Session Hijacking

Session ID's identifies and track a web application user. Any attacker who gets this unique identifier submits the same information and impersonates someone else and this type of attack is called Session Hijacking. The inherent stateless nature of the HTTP (and HTTPS) protocol in which the process of masquerading is an alternative to user using a hijacked session ID.

An attacker has three methods for gaining session ID information – observation, brute force and misdirection of trust

Observation

All HTTP traffic crosses the wire in an unencrypted, plain text, mode. Therefore, any device with access to the same wire or shared network devices is capable of "sniffing" the traffic and recording session ID information. In addition, many perimeter devices automatically log aspects of HTTP traffic – in particular the URL information.

A simple security measure to prevent "sniffing" or logging of confidential URL information is to use the encrypted form of HTTP – HTTPS.

Brute Force

If the session ID information is predictable then it becomes very easy for an attacker to repeatedly attempt to guess a valid ID. The process can take as little time as a few seconds, depending upon the randomness and the length of the session ID.

In ideal situation an attacker using a domestic DSL line can potentially conduct up to as many as 1000 session ID guesses per second. Thus it is very important to have a sufficiently complex and long session ID to ensure that any likely brute forcing attack will take many hundreds of hours to predict.

A paper by David Endler on the processes involved in brute forcing session ID's should be sought by readers requiring background information on this process.

Misdirected trust

A client's web browser would only ever disclose confidential session ID information to a single, trusted site. Unfortunately, there are numerous instances when this is not the case. For example – the HTTP REFERER field will send the full URL, and in some applications this URL may contain session ID information.

Another popular method, having common trust relationship flaws, is HTML embedded and Cross-site Scripting attacks. By cleverly embedding the HTML code or scripting elements, it is possible to steal session ID information – even

if it is held within the URL, POST fields and cookies. Readers needing more information about this class of attack and should review a copy of "HTML Code Injection and Cross-site scripting".

Common Failings

web based session management is important for tracking users and their navigation throughout an application, the most critical use is to maintain the state information of an authenticated user as he carries out his allowed functions. For online banking and retail environments, using an appropriately strong session management method is crucial to the success of the organization.

Some of the Most Common Failings and Assumptions

Predictable Session ID's

Being predictable has been the most regular error in the usage of session IDs. The two reasons, as stated earlier, are a lack of randomness, or length, or both.

Sequential allocation of Session ID's – Everyone who visits the site is provided a session ID in a sequential order. Therefore, by examining your own information of the session ID, the simple practice of replacing it with a different value a few steps up or down will permit the attacker to impersonate another user.

Session ID values are too short –During an automated attack, the whole series of valid session ID's could be enclosed prior to the time for the session to expire.

Common hashing techniques – Whilst a variety of the marketable websites have in-built functions for the calculation of hashed information, such methods are renowned and available for being reproduced. Great concern should be given to ensure that predictable information is not utilized in generating the hash, as the hashing function will duly create a session ID value that is unique. For example, there have been instances when the "unique" hash was based upon the local system time, and the IP address of the connecting host. The attacker was able to pre-calculate a huge number of time dependent hashes for a popular internet portal or a proxy service (i.e. AOL), using the same hashing function, and then used it to brute force any existing session from that service.

Session Obfuscation – Using a routine method of obscuring data and then it's usage for the purpose of session management. Including of the client or other confidential information within a session ID is not at all a feasible idea. For example, some organizations have even tried to encode the user's name and his password within the session ID using a shifted Unicode and hexadecimal representation of the information.

INSECURE TRANSMISSION

The use of these applications in banks and the retail sector require that all the confidential material and its session information be transmitted securely and shouldn't be at risk to observation or replay attacks. Regrettably many of the commercial packages have disastrously failed in past to secure the integrity of their session management due to insecure transmission.

Use Encryption when sending session information – As it was told previously, there are a lot of occurrences whereby a user's connection to the application server will be logged if it is not sent over an encrypted channel, such as HTTPS. This is majorly significant for applications that call for a high degree of confidentiality. Organizations should note that the client browser will submit the session ID with every request (and this includes pages and graphics) especially, if the cookie method is being used for managing session IDs, and it may also submit it to other servers within the same domain – which may or may not happen over a channel of secure data.

Using different session ID's when shifting between the secure and insecure application components – As a new user who finds the way through the web application as a "guest", use a different session ID than what would have been allocated in the secure part of that application. You should never use the same session ID information in the authenticated and unauthenticated sections of that web application. Make sure for a second time, that the session ID to be used in the secure part of that web application is not predictable and is based on the previous ID.

LENGTH OF SESSION VALIDITY

For the use of secure applications all session information should be time-limited and allow for client-side cancellation or server-side revocation.

Client Cancellation – Various web applications are unable to allow for client-side cancellation such as "log-out". If the purpose is to allow users to be able to interact with the applications from anywhere and everywhere, including in Internet Cafes, organizations need to be aware that other users can use that same machine and probe through the "history" and its cached page information. If the session has not been cancelled, it is a little exercise for the next user using that computer to "resume" the last connection.

Session Timeout – Yet again, when dealing with the prospect of shared client computers, it is very critical that there is a limited life (or a period of inactivity) after which the session will expire automatically. The expiry time should be kept to minimum and should be dependent upon the nature of that application. Preferably the application should have the ability to monitor the period of inactivity for each of the session IDs and be able to delete or revoke the session ID on reaching a threshold.

Server Revocation – In a few situations it may be indispensable to cancel a session at the server-side. Such likely events include when the user has left the insecure part of the application and has entered the secure part with new session ID. As an alternative, should some kind of attack be recorded by the server, it would be desirable to revoke the session associated with the attacker's system.

SESSION VERIFICATION

The handling and manipulating processes for session ID information should be robust and capable of handling correctly the attacks that are targeting the content within.

Session ID Length – It should be made sure that the session ID's content is of the expected type and size, and that the quality of the information has been verified before it is processed. For instance, one should be capable of identifying over-sized session IDs that will constitute a buffer overflow type attack. Additionally, ensure that the content of the session ID is not containing unexpected information – like, if the session ID will be used within the application's backend database, then one must ensure that the session ID does not have embedded data strings that maybe interpreted as an extension to the 'Select' SQL query.

Source of the Session ID – At the time of the usage of HTTP POST method for communicating session information, make sure that the application is able of discriminating that the session ID was delivered to the application from the client-browser through the HTTP POST method, and not through a manipulated GET request. Converting of the HTTP POST into a GET request is a very regular method of conducting cross-site scripting attacks and other distributed brute force attacks.

GOOD SESSION MANAGEMENT

Different means of session handling implementation are there for developers, as per the purpose of the application. For those applications which require the most highest level of session handling security, options are very less, and require a blend of methods described earlier in this text. The following example at present represents one of the most secure methods of handling sessions, but it's complex and difficult to implement successfully. This method is dependent on three sources of session ID information. These are the URL, the HTTP REFERER field and the cookies.

Once a client is initially connecting to the application as a guest, they are given unique personal identifier (ID1), and then this information is embedded within that URL to which they are redirected to. Also present within that same URL is a random identifier for page that was seen (ID2). The third personal identifier (ID3) is then sent as a session cookie, with a life period of the open-client

browser (i.e. the session cookie is held within the memory – and if the browser window or any of the child windows are closed, this information is gone).

If the application server notes no activity from the client-browser, the session information of ID3 is revoked.

1. Client connects to the site www.example.com over HTTP. http://www.example.com/

2. The Client is on its own, redirected through a server-side redirect to the home page with a URL that contains the unique session information - ID1 (user = ID93x7HeT7P4a9) and ID2 (current page = 3789264).

   http://www.example.com/page.jsp?user=ID93x7HeT7P4a9;cpage=3789264

3. Within the HTTP server response, a session cookie is delivered (user track = UT23dWT3nQi7n4).

   Set-Cookie:UserTrack=" UT23dWT3nQi7n4"; path="/"; domain="www.example.com"; expires="2000-01-01 00:00:00GMT"; version=0

   Within the page that is shown to the client, there will be a lot of hyperlinks to other content pages within the application. Each of the links has been dynamically created for the inclusion of the client ID1, and a randomly generated (but catalogued) page identifier. As the unauthenticated user navigates through the site, the current page identifier will change while ID1 and ID3 remain static. ID3 will change when the user has been successfully authenticated.

   Pages which contain user information submission areas, there, all HTML forms have hidden fields which include both ID1 and ID2. If the presented information is contains ANY confidential or personal information, then submission MUST be made securely over HTTPS.

4. Within the page, each hyperlink is uniquely addressed and contains an associated random identifier.

   Link 1

   Link 2

   Link 3

5. Within a page containing a user submission area, the form may look like the following (note that the ACTION specifies both HTTPS and the full URL):

   <FORM METHOD=POST ACTION="https://www.example.com/post/page.asp">

<INPUT TYPE="hidden" NAME="user" VALUE=" ID93x7HeT7P4a9">

<INPUT TYPE="hidden" NAME="cpage" VALUE="3789264">

<INPUT TYPE="text" NAME="data" MAXLENGTH="100">

<INPUT TYPE="submit" NAME="Send Data">

6. Every page or data submitted by the client browser will include the session cookie information (ID3).

7. The application must acquire each identifier (ID1, ID2 and ID3) and check each of it to see if they are valid for the client request, and that they have not been timed out or revoked. If this information is NOT correct, then the client is redirected to the application's first page with all new identifiers (ID1, ID2 and ID3) and also all the previous ID information is revoked.

8. When the client browser submits a request or if it trails a hyperlink, then a HTTP REFERER value is included. This value signifies the URL that was earlier presented to the client browser. The application should also be able to verify that ID2 within the REFERER URL is the correct precursor to the newly requested page (npage=). If it is not, then the client browser has not followed the right path to request the new page, and this may indicate an attack in progress.

   For example, the correct sequence to reach page 2 from the initial page is by following "link 1". Therefore, the request for the page http://www.example.com/page.asp?user=ID93x7HeT7P4a9;npage=8777623 must contain http://www.example.com/page.jsp?user=ID93x7HeT7P4a9;cpage=3789264 in the HTTP REFERER field.

9. If the identifiers are perfectly valid and correct, then a new page is presented. Also, ID2 is updated (e.g. current page = 8777623), while ID1 and ID3 remain the same.

   http://www.example.com/page.jsp?user=ID93x7HeT7P4a9;cpage=8777623

10. The page which is then returned back contains new random identifiers for all hyperlinks. There should always be a link to go "back" to the previous page. However, the previous page will have to be assigned a new random identifier. The client browsers "Back" button will no longer work then. For example:

    Original Page 1 was
    http://www.example.com/page.jsp?user=ID93x7HeT7P4a9;cpage=3789264

Page 2 is
http://www.example.com/page.jsp?user=ID93x7HeT7P4a9;cpage=8777623

to return to Page 1, the URL may be –
http://www.example.com/page.jsp?user=ID93x7HeT7P4a9;cpage=7322641

When the application wants the user to authenticate, all data submissions MUST be over an encrypted session like that of HTTPS. If the user gets successfully authenticated, a new session cookie (ID3) is issued to him, and the previous session cookie information is revoked at the server. All communication thereafter (till the time the user has finalized to "logout") must be over HTTPS.

11. If the user has successfully authenticated with the application, then the previous session cookie (ID3) is revoked and also a new ID3 is issued through the HTTPS session which has now been encrypted.

12. The application should be able to associate ID3 with the type of communication (i.e. HTTP or HTTPS), and instantly revoke all session information (ID1, ID2 and ID3) if the new ID3 has been used to access non-secure application resources. The usage of revoked or inappropriate session information should effect in the client browser being redirected to the start page and issued with an all new session identifiers as discussed earlier.

13. Now, yet again, just like the unsecured parts of the application, all pages are passed to the client in the authenticated and secure part of the application and should have randomly generated page identifiers.

14. The users should have the luxury to "logout" and cancel their session when they want. Logging out by the user results in the revocation of all session information and, if possible, the immediate closing of the client browser. Additionally, it is a good habit to ensure that both the HTML Meta tags associated with caching and the HTTP caching options are set to expire in the past so that no page content ought to be stored on the client system.

It is very crucial to note that it becomes nearly impossible to carry out any kind of URL embedded cross-site scripting attack, when you are utilising session information in the URL,. By assignment of unique random identifiers to each page and linking between pages with one-time identifiers, it is more or less impossible for an attacker to conduct any brute force or repetitive attacks. Nevertheless, it will not work with client browsers that have their cookies disabled, as this session method relies on the use of session cookies. In some of the cases, a client browser page request may not be containing any data in the HTTP REFERER field.

Check Your Progress 2

Note: a) Space is given below for writing your answer.

b) Compare your answer with the one given at the end of the Unit.

1) Write a note on security.

.....

.....

.....

.....

2) Define the terms authentication and authorization?

.....

.....

.....

.....

3) Write a note on threats, vulnerabilities and attacks?

.....

.....

.....

.....

4) What are the factors on which the length of the session id depends?

.....

.....

.....

.....

4.6 THREAT RISK MODELING

Threat risk modeling states to be an important process for secure web application development. It helps organizations to analyze the correct controls and to produce effective countermeasures within budget. For example, there is little point in spending $100,000 for fraud control for a system that has negligible fraud risk. Performing threat risk modeling using the Microsoft Threat Modeling Process

The threat risk modeling process has five steps, enumerated below and shown graphically in Figure 5 given below. They are:

- Identify Security Objectives
- Survey the Application
- Decompose it
- Identify Threats
- Identify Vulnerabilities

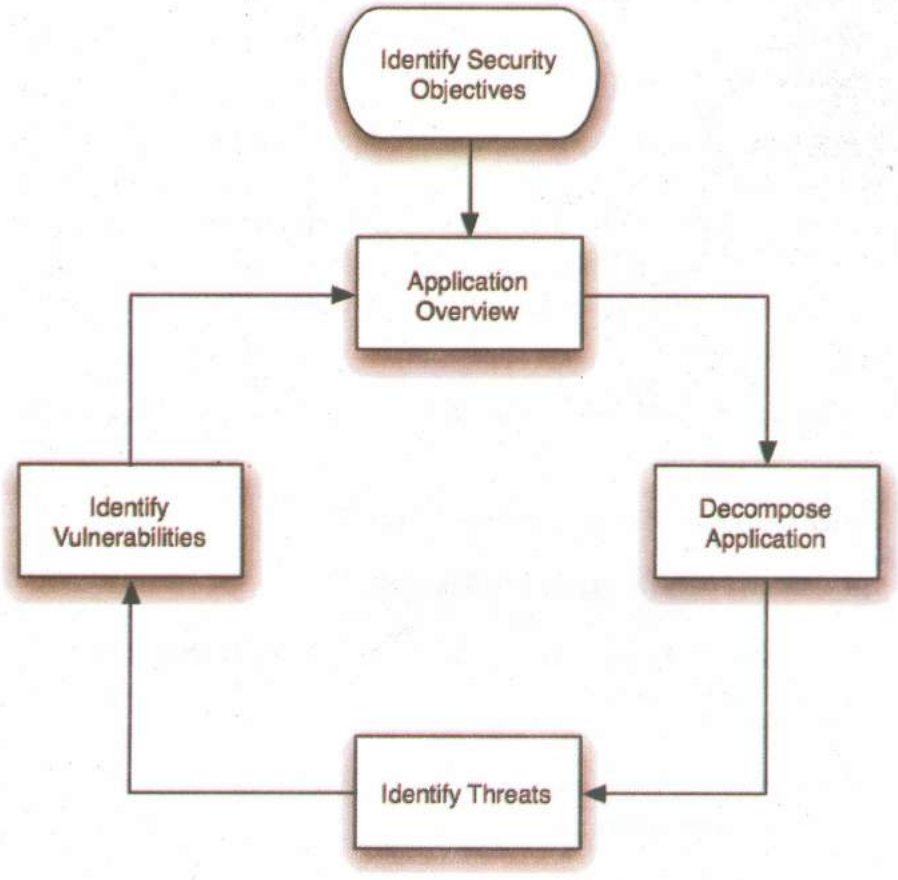


Fig. 5

Identify Security Objectives

The business leadership, in related to the software development and quality assurance teams, who all need to understand the security objectives. To allow this, break down the application's security objectives into the following categories:

Identity: Does the application protect user identity from abuse? Are there adequate controls in place to ensure evidence of identity (as required for many banking applications?)

Financial: a potential financial loss assesses the level of risk in the organization and is prepared to absorb in remediation. For example, forum software may have a lower estimated financial risk than an Internet banking application.

Reputation: estimation of the loss of reputation is being derived from the application being misused or successfully attacked.

Privacy and Regulatory: what will be the extent to which application will protect user data? Forum software is public by nature, but a tax preparation application is subject to contain tax regulations and privacy legislation requirements in most countries.

Availability Guarantees: Is the application required to be available per a Service Level Agreement (SLA) or similar guarantee? Is it a nationally protected infrastructure? To what level will the application have to be available? High availability techniques are very expensive therefore applying the correct controls up front will save a great deal of time, resources, and money.

It gives an idea of some of the business risk decisions leading into selecting and building security controls.

Other sources of risk guidance come from:

- Laws (such as privacy or finance laws)
- Regulations (such as banking or e-commerce regulations)
- Standards (such as ISO 17799)
- Legal Agreements (such as payment card industry standards or merchant agreements)
- Corporate Information Security Policy

Application Overview

When the security objectives have been defined, it is required to analyze the application design to identify the components, data flows, and trust boundaries.

It is done by surveying the application's architecture and design documentation by looking for UML component diagrams helps to understand how and why data flows to various places. For example, data movement across a trust boundary (such as from the Internet to the web tier, or from the business logic to the database server), needs to be carefully analyzed, whereas data that flows within the same trust level does not need as much scrutiny.

Decompose Application

Once the application architecture is understood it becomes easy to identify the features and modules with a security impact that need to be evaluated. For example, while investigating the authentication module, it is necessary to understand how data enters the module, how the module validates and processes the data, where the data flows, how the data is stored, and what fundamental decisions and assumptions are made by the module.

Identify Threats

Writing down of unknown threats is impossible, but it happens that new malware will be created to exploit new vulnerabilities within custom systems. Therefore, concentrate on known risks can be easily demonstrated using tools or techniques from Bugtraq.

Microsoft provides two types of approaches for writing up threats. One is a threat graph, as shown in Figure 6, and the other is a structured list.

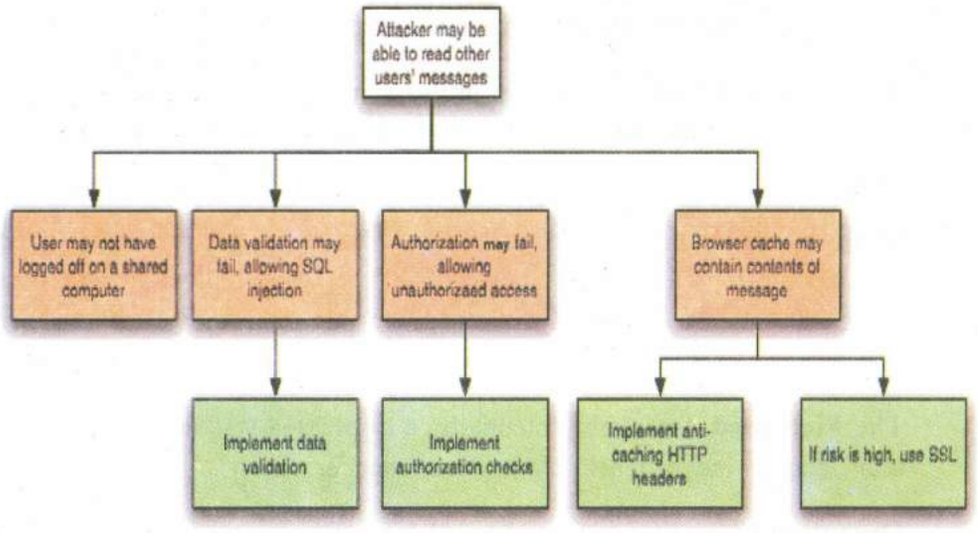


Fig. 6

Characteristically, greater information is given quickly by a threat graph but it takes too long time to construct, when compared to a structured list as in this it is easier to create but will take longer for the threat impacts to become apparent.

- Attacker may be able to read other user's messages
- User may not have logged off on a shared PC
- Data validation may allow SQL injection
- Implement data validation

- Authorization may fail, allowing unauthorized access
- Implement authorization checks
- Browser cache may contain contents of message
- Implement anti-caching directive in HTTP headers
- If eavesdropping risk is high, use SSL

Underline that it takes a motivated attacker to take advantage of a threat; they generally want something from your application or to hinder controls. For understanding the relevant threats, use the following categories to know about who might attack the application:

Accidental Discovery: It happens when access to privileged information or functionality is gained when an ordinary user using a web browser accidently stumbles across a functional mistake in your application.

Automated Malware: When programs or scripts, which are already in search for known vulnerabilities report them to a central collection site.

The Curious Attacker: when an ordinary user or a security researcher, who observes something wrong with the application, and then decides to track further.

Script Kiddies: Ordinary renegades, who are seeking to compromise or deface your applications for collateral gain, notoriety, or a political agenda, possibly using the attack categories described in the OWASP Web Application Penetration Checklist.

The Motivated Attacker: Potentially, a displeased staff member with inside knowledge or maybe a paid professional attacker.

Organized Crime: Criminals who seek high stake payouts, such as cracking e-commerce or corporate banking applications, for their monetary gains.

It is very important to understand the level of attacker that you are defending against. For example, a motivated attacker, who understands your internal processes is often more risky than script kiddies.

STRIDE

STRIDE is a classification scheme for the characterization of threats which are known according to the kinds of exploit that are used (or motivation of the attacker). The STRIDE acronym is formed from the first letter of each of the following categories.

Spoofing Identity – Spoofing of the identity for applications that have a lot of users but provide a single execution context at the application and database

level is a major risk. Especially, users should not be able to become any other user or assume the attributes of another user.

Tampering with Data Users can significantly change data which is delivered to them, return it, and by this means potentially manipulate client-side validation, GET and POST results, cookies, HTTP headers, and so on. The application should not send data to the user, such as interest rates or periods, which are obtainable only from within the application itself. The application should also examine with care the data received from the user and validate that it is sound and applicable before storing or using it.

Repudiation Users may disagree with transactions if there is unsatisfactory auditing or recordkeeping of their activity. For example, if a user says, "But I didn't transfer any money to this external account!", and you cannot track his/her activities through the application, then it is extremely possible that the transaction will have to be written off as a loss.

For that reason, deem if the application requires non-repudiation controls, such as web access logs, audit trails at each tier, or the same user context from top to bottom. If possible, the application should function with the user's privileges, not more, but this may not be possible with many off-the-shelf application frameworks.

Information Disclosure Users are rightfully cautious of giving off personal details to a system. If there is a possibility for an attacker to publically reveal user data at large, whether anonymous or as an authorized user, there will be an instantaneous loss of confidence and a substantial period of reputation loss. For that reason, applications must include fool proof controls to prevent the tampering of user ID and abuse, predominantly if they are using a single context to run the entire application.

What's more, think if the user's web browser may leak information. Some browsers may ignore the "no-caching directives" in HTTP headers or may handle them incorrectly. In a consequent fashion, every secure application has a responsibility to minimize the amount of information stored by the web browser, just in case it leaks or leaves information behind, which may be used by an attacker to learn the particulars about the application, the user, or to potentially become that user.

Lastly, in the implementation of persistent values, remember that the usage of hidden fields is insecure by nature. Such storage should not be relied upon to secure sensitive information or to provide adequate personal privacy safeguards.

Denial of Service - Application designers should be well aware that their applications may be subject to a denial of service (DOS) attack. As a result, the use of costly resources such as large files, complex calculations, heavy-duty

searches, or long queries should be reserved only for authenticated and authorized users, and should not be available to anonymous users.

For applications that do not have this luxury, every facet of the application should be engineered to perform as little work as possible, to use fast and few database queries, to avoid exposing large files or unique links per user, in order to prevent simple denial of service attacks.

Elevation of Privilege If there is a provision of distinct user and administrative roles by an application, then it is crucial to ensure that the user cannot lift up his/her role to a higher privileged one. Above all, simply not displaying privileged role links is not sufficient. As an alternative, all actions should be gated through an authorization matrix, for ensuring that only the permitted roles have access to privileged functionality.

DREAD

DREAD is a classification scheme for the quantification, comparison and prioritization of the amount of risk presented by each evaluated threat. The DREAD acronym is formed from the first letter of each of the below presented categories.

DREAD modeling influences the thinking behind the setting of risk rating, and is also used to directly sort out the risks. The DREAD algorithm, shown below, is used to compute a risk value, which is an average of all five categories.

Risk_DREAD = (DAMAGE + REPRODUCIBILITY + EXPLOITABILITY + AFFECTED USERS + DISCOVERABILITY) / 5

The calculation always produces a number between 0 and 10; the higher the number, the more serious the risk.

Here are some examples of how to quantify the DREAD categories.

Damage Potential

If a threat exploit occurs, how much damage will be caused?

- 0 = Nothing
- 5 = Individual user data is compromised or affected.
- 10 = Complete system or data destruction

Reproducibility

How easy is it to reproduce the threat exploit?

- 0 = Very hard or impossible, even for the application administrators.
- 5 = One or two steps re ed, may need to be an authorized user.

- 10 = Just a web browser and the address bar is sufficient, without authentication.

Exploitability

What is needed to exploit this threat?

- 0 = Advanced programming and networking knowledge, with custom or advanced attack tools.
- 5 = Malware exists on the Internet, or an exploit is easily performed, using available tools for attack.
- 10 = Just a web browser

Affected Users

How many users will be affected by this?

- 0 = None
- 5 = Some users, but not all
- 10 = All users

Discoverability

How easy is it to discover this threat?

- 0 = Very hard to impossible; requires source code or administrative access.
- 5 = Can figure it out by guessing or by monitoring network traces.
- 9 = Details of faults like this are already in the public domain and thus can be easily discovered using a search engine.
- 10 = The information is visible in the web browser address bar or in a form.

Note: When performing a security review of an existing application, "Discoverability" will often be set to 10 by convention, as it is always assumed that the threat issues will be revealed.

Note: Using DREAD can be difficult initially. But it may be helpful to think of Damage Potential and Affected Users in terms of Impact, while thinking of Reproducibility, Exploitability, and Discoverability in terms of Probability. Using the Impact vs Probability approach (which follows best practices such as defined in NIST-800-30), I would alter the formula to make the Impact score equal to the Probability score. Otherwise the probability scores have more weight in the total.

Alternative Threat Modeling Systems

OWASP is aware that the implementation of the Microsoft modeling process may not fit all organizations. If STRIDE and DREAD are unacceptable for some reason, it is recommended that your organization must "dry run" the other threat risk models talked about against an on hand application or design. This allows you to determine which approach works best, and to adopt the most appropriate threat modeling tools for your organization.

In a nutshell, performing threat modeling provides a far bigger return than most of the other controls in this guide. Hence, make threat risk modeling an early priority in your application design process.

Trike

Trike is a threat modeling framework similar to the Microsoft threat modeling process. Yet, Trike is different because it uses a risk based approach with distinct implementation, threat, and risk models, instead of the usage of STRIDE/DREAD aggregated threat model (attacks, threats, and weaknesses). From the Trike paper, Trike's goals are:

To make sure that the risk which this system entails to each of the assets is acceptable to all stakeholders, with the assistance from system stakeholders.

Empower stakeholders to understand and decrease the risks to them and other stakeholders implied by their actions within their respective domains.

For more information on Trike, please see Section 6.9, reference 8.

AS/NZS 4360:2004 Risk Management

The Australian/New Zealand Standard AS/NZS 4360, first issued in 1999, and revised in 2004, is the world's first formal standard given for the documentation and management of risk and is still one of the few formal standards for managing it. The standard's approach is simple (it's only 28 pages long), flexible, and iterative. Moreover, it does not lock organizations into a particular risk management methodology, given that the methodology fulfils the AS/NZS 4360 five steps. It also provides quite a few sets of risk tables as examples, and allows organizations to freely develop and adopt their own.

The five steps of the AS/NZS 4360 process are:

- Establish Context: Establish the risk domain, i.e., which assets/systems are important?
- Identify the Risks: Within the risk domain, what specific risks are evident?

- Analyze the Risks: Look at the risks and determine if there are any supporting controls in place.
- Evaluate the Risks: Determine the residual risk.
- Treat the Risks: Describe the method to treat the risks so that risks selected by the business will be mitigated.

AS/NZS 4360 assumes that an operational risk group will manage the risk, and that the organization has in-house adequate skills and risk management resources to identify, analyze, and treat the risks.

The advantages of AS/NZS 4360

- AS/NZS 4360 works well as a risk management methodology for organizations requiring Sarbanes-Oxley compliance.
- AS/NZS 4360 works well for organizations which prefer to manage risks in a traditional way, such as just using likelihood and consequence to determine an overall risk.
- AS/NZS 4360 is familiar to most risk managers worldwide, and your organization may already have implemented an AS/NZS 4360 compatible approach.
- You are an Australian organization, and you may require using it if you are audited on a continuous basis, or to justify why you aren't using it. Luckily, the STRIDE/DREAD model discussed earlier is AS/NZS 4360 compatible.

The limitations of AS/NZS 4360

- The AS/NZS 4360 approach works best for business or systemic risks than for technical risks.
- AS/NZS 4360 does not define the methodology to perform a structured threat risk modeling exercise.
- As AS/NZS 4360 is a generic framework for managing risk, it does not provide any structured method to enumerate web application security risks.

Even though AS/NZS 4360 may have been used to rank the risks for security reviews, lack of structured methods of enumerating threats for web applications makes it not so sought-after than other methodologies presented before.

CVSS

The US Department of Homeland Security (DHS) established the NIAC Vulnerability Disclosure Working Group, which takes input from Cisco

Systems, Symantec, ISS, Qualys, Microsoft, CERT/CC, and eBay. One of the group's outputs is the Common Vulnerability Scoring System (CVSS).

The advantages of CVSS

- You get notification from a security researcher or other source that your product has vulnerability, and wish to have an accurate and normalized severity rating, and alerts your customers to the appropriate level of action required when you release the patch.
- You are a security researcher, and found several threat exploits within an application. You would like to use the CVSS ranking system to produce reliable risk rankings, to ensure that the ISV will take the exploits seriously as indicated by their rating.
- CVSS has been recommended by the working group for use by US Government departments while it is not clear if it will become policy or be widely adopted at the time of this writing.

The limitations of CVSS

- CVSS does not overcomes the attack surface area (i.e. design flaws), or help enumerate risks within any arbitrary piece of code, as it is just a scoring system and is not a modeling methodology.
- CVSS is more complex than STRIDE/DREAD, and aims to calculate the risk of announced vulnerabilities as applied to deployed software and environmental factors.
- The CVSS risk ranking is complex i.e. a spreadsheet is required to calculate the risk components as the assumption behind CVSS is that a specific vulnerability has been identified and announced, or a worm or Trojan has been released targeting a small number of attack vectors.
- The overhead of calculating the CVSS risk ranking is quite high if applied to a thorough code review, which may have 250 or more threats to rank.

OCTAVE

OCTAVE is referred to as a heavyweight risk methodology approach originated from Carnegie Mellon University's Software Engineering Institute (SEI) in collaboration with CERT. OCTAVE's major concern is on organizational risk, not technical risk. OCTAVE has two versions: Full OCTAVE, for large organizations, and OCTAVE-S for small organizations, both of which have specific catalogs of practices, profiles, and worksheets to document the modeling outcomes.

OCTAVE is popular with many sites and is useful when

- Implementation of an organizational culture of risk management and control is necessary.
- Documenting and measuring business risk becomes timely.
- Documenting and measuring the overall IT security risk becomes necessary.
- When documenting, risks surrounding complete systems becomes necessary.
- To accommodate a fundamental reorganization, such as when an organization does not have a working risk methodology in place, and requires a robust risk management framework to be put in place.

The limitations of OCTAVE

- OCTAVE is incompatible with AS/NZS 4360, as it mandates Likelihood = 1 (i.e., It assumes a threat will always occur) and this is inappropriate for many organizations. OCTAVE-S makes the inclusion of this probability optional, but this is not part of the more comprehensive OCTAVE standard.
- Consisting of 18 volumes, OCTAVE is large and complex, with many worksheets and practices to implement.
- It does not provide a list of "out of the box" practices for assessing and mitigating web application security risks.

These issues do not anticipate OWASP that OCTAVE will be used at large by application designers or developers, as it fails to take threat risk modeling into consideration, which is useful during all stages of development, by all participants, to reduce the overall risk of an application.

4.7 WEB SERVICES CHALLENGES

The open standards communities that created Web services developed a number of security standards for Web services.

Challenges

This section discusses some of the important challenges in the area of web services security.

Discovery

In Web services discovery, participants identify and compose Web Services Description Language (WSDL) which is a specific service based on definitions in a UDDI registry. Due to the potentially large number of service candidates in the registry, performance rankings for algorithms used to search, match and compose services can vary from case to case.

End to End Quality of Service and Protection

Most Web services do not provide guarantees for Quality of Service (QoS) or Quality of Protection (QoP). QoS is defined as what the expected level of performance a particular Web service will have.

Availability and Protection from Denial of Service Attacks

Availability allows a Web Services Application to detect a Denial of Service (DOS) attack, the operation is continued as long as possible and then to gracefully recover and resume operations after a DOS attack.

4.8 RECOMMENDATIONS

Some recommendations are required for secure Web Services.

Replicate Data and Services to Improve Availability

Since Web Services are susceptible to Denial of Service (DOS) attacks, it is important to replicate data and applications in an effective manner.

Use Logging of Transactions to Improve Accountability

Non-repudiation and accountability require logging mechanisms involved in the entire SOA transaction. As of this writing, there are few implemented logging standards that can be used across an entire SOA. The level of logging provided by various UDDI registries, identity providers, and individual Web services varies greatly. Where the provided information is not sufficient to maintain accountability and non- repudiation, it may be necessary to introduce additional software or services into the SOA to support these security properties.

Use Threat Modeling and Secure Software Design

Techniques to Prevent Attacks

The secure software design techniques help to ensure that the design and implementation of Web Services Software does not contain defects that can be exploited. Threat modeling and risk analysis techniques are to be used to protect the Web Services application from attacks. When used effectively,

threat modeling can find security strengths and weaknesses, discover vulnerabilities and provide feedback into the security life cycle of the application. It must include security oriented code reviews and penetration testing. By using threat modeling and secure software design techniques Web services can be implemented to handle a variety of attacks.

Use Performance Analysis and Simulation Techniques for End to End Quality of Service and Quality of Protection

Queuing networks and simulation techniques played important roles in designing, developing and managing complex information systems. Similar techniques are used for quality assured and highly available web services. With QoS of a single service, end-to-end QoS is also becomes important for most composite services. For example, enterprise systems having many business partners complete their business processes on time to meet real time market conditions. The dynamic and compositional nature of web services makes end-to-end QoS management a major challenge for service oriented distributed systems.

Use XML Firewalls

To mitigate the risks from Web based applications is to use XML firewalls. This allows filtration of XML documents and uses contents in XML documents to make decisions. For example, it can filter malicious content, or specially crafted parameters that can be used to generate a SQL query by the attacker. And is configured properly to act like a WS-Security end point which verifies user signatures and validate user identity.

Use Penetration Testing

In Web Services interfaces penetration testing is done before they are opened up for external access. These interfaces are more complex and even experienced development teams make mistakes having security implications.

Check Your Progress 3

Note: a) Space is given below for writing your answer.

b) Compare your answer with the one given at the end of the Unit.

1) Discuss applications security objectives.

.....

.....

.....

.....

2) Write a note on web services challenges.

.....

.....

.....

.....

3) What are XML firewalls?

.....

.....

.....

.....

4) What is penetration testing?

.....

.....

.....

.....

4.9 LET US SUM UP

This unit is all about the web applications and security concerns during use of web applications. The learners will learn about the web application and the respective security concerns. The topics like session ID is also covered. The threat risk modeling and web services challenges are also covered well.

4.10 CHECK YOUR PROGRESS: THE KEY

Check Your Progress 1

1) The web provides marketers' a way to know about the people who are trying to visit their sites and generally start communication with them. One of the ways of doing so is asking web visitors to subscribe to newsletters or to submit an application form when they are requesting information for the products or may provide details to modify their browsing experience.

   From the technical point of view, web is a highly programmable environment that allows mass customization by deploying a large and diverse range of applications, to millions of global users. Two important components existing in a modern website are the flexible web browsers and the web applications; both are available to all and of no expense.

Users are able to get data and interact with web pages within a website through the help of Web browsers.

Today's websites are far more than the static text and graphics showcases of the early and mid-nineties: modern web pages provide personalized dynamic content to users according to their individual preferences and settings. Furthermore, web pages may also run client-side scripts that "change" the Internet browser into an interface for such applications as web mail and interactive mapping software (e.g., Yahoo Mail and Google Maps).

Modern web sites importantly allow to capture, process, store and transmit delicate data of customer (e.g., personal details, credit card numbers, social security information, etc.) for immediate and recurrent use. And, this is done using web applications. Such features as web mailing, login pages, support and product request forms, shopping carts and content management systems, enhance modern websites and allow businesses to communicate with prospects and customers. These are all common examples of web applications.

Web applications are computer programs which allow website visitors to submit and retrieve data to/from a database over the Internet using the preferred web browser.

The web application using a web server present the data to the user within their browser as information is generated dynamically (in a specific format, e.g. in HTML using CSS). For technically oriented, Web applications query the content server and dynamically generate web documents to serve the users. The documents are appeared in a standard format that is supported by all browsers (e.g., HTML or XHTML). JavaScript is a client side script that provides dynamic elements on each page. The web browser is a key that interprets and runs all scripts, while displaying the requested pages and content. The web browser is described as the "universal client for any web application".

2) The three-layered web application model shows:

   - The first layer is generally a web browser or the user interface;
   - The second layer is the dynamic content generation technology tool such as Java servlets (JSP) or Active Server Pages (ASP), and
   - The third layer is the database containing content (e.g., news) and customer data (e.g., usernames and passwords, social security numbers and credit card details).

3) Web applications also raise a number of security concerns due to improper coding. Serious weaknesses or vulnerabilities make hackers to gain direct

and public access to databases in order to misuse the sensitive data. Many of these databases having valuable information (e.g., personal and financial details) become target of hackers. Although such acts of vandalism as defacing corporate websites are still common and hackers prefer gaining access to the sensitive data residing on the database server because of the immense pay-offs in selling the data.

The framework shows how it is easy for a hacker to quickly access the data residing on the database through some creativity and luck or by negligence or human error which creates vulnerabilities in the web applications.

Websites are depending on databases to deliver the information to visitors. If web applications are not secure by various forms of hacking techniques, then your entire database of sensitive information is at serious risk.

4) Ethical hacking is the process of hacking the confidential information. The information is not secure by ethical hacking. This process is also known as the intrusion testing, penetration testing or red teaming. However it also gives professional certification to the certified ethical hacker where the hacking of the computer system or some other devices is done. This service had been made available to the people by the international council of e-commerce consultants.

The user has to be an ethical hacker a name given to the person who does ethical hacking and for this the person has to be a penetration tester. The ethical hacker is responsible for the performance of different activities; their main role is to work for an organization for penetrating the information from different networks or systems. The organization trusts him as he is responsible for providing different services to the firm.

The ethical hacking service is similar to that of the hacking and the ethical hacker also works in the same way as the hacker works for different purposes. The ethical hacker is found to be a computer expert and is also responsible for the working of the networking systems. He is the person who works on behalf of the members of the organization. However, the hacking service that is being provided by the hacker can become dangerous for the firm and can exploit the systems of the company.

Check Your Progress 2

1) Security means protecting assets. The Assets may be tangible items, such as a Web page or your customer database or may be less tangible, such as your company's reputation. Security is a path and it is not a destination. While analyzing the infrastructure and applications, potential threats are identified and present a degree of risk. Security is about risk management and implementing effective countermeasures.

2) **Authentication**

Authentication arise the question: who are you? It is the process of uniquely identifying the clients of your applications and services. These might be end users, other services, processes, or computers. Authenticated clients are referred to as principals in security parlance.

Authorization

Authorization arise the question: what can you do? It is the process that governs the resources and operations that the authenticated client is permitted to access. These resources include files, databases, tables, rows, and so on, together with system-level resources such as registry keys and configuration data. Operations include performing transactions such as purchasing a product, transferring money from one account to another, or increasing a customer's credit rating.

3) A threat is any potential occurrence that is malicious and could harm an asset. It is said to be any bad thing that can happen to your assets. Vulnerability is a weakness that makes a threat possible. This may be of poor design, configuration mistakes, or inappropriate and insecure coding techniques. Weak input validation is an example of an application layer vulnerability, which creates input attacks. An attack is an action that exploits vulnerability or enacts a threat. Examples of attacks include sending malicious input to an application or flooding a network in an attempt to deny service. In summarize form we may said that a threat is a potential event that can adversely affect an asset, whereas a successful attack exploits vulnerabilities in your system.

4) **Factors on Which the Actual Length of the Session Id Depends**

Speed of connection implies that there is typically a big difference between Internet client, B2B and internal network connections. An Internet client will typically have less than a 512 kbps connection speed, but an internal user may be capable of connecting to the application server at 200 times faster. Thus an internal user could potentially obtain a valid session ID in 1/200th of the time.

Complexity of the ID implies what values and characters are used within the session ID? Moving from numeric values (0-9) to a case-sensitive alpha-numeric (a-z, A-Z, 0-9) range means that, for the same address space, the session ID becomes much more difficult to predict. For example, the numeric range of 000000-999999 could be covered by 0000-5BH7 using a case-sensitive alpha-numeric character set.

Check Your Progress 3

1) **Identity:** Does the application protect user identity from abuse? Are there adequate controls in place to ensure evidence of identity (as required for many banking applications?)

   Financial: a potential financial loss assesses the level of risk in the organization and is prepared to absorb in remediation. For example, forum software may have a lower estimated financial risk than an Internet banking application.

   Reputation: estimation of the loss of reputation is being derived from the application being misused or successfully attacked.

   Privacy and Regulatory: To what extent will the application have to protect user data? Forum software is public by nature, but a tax preparation application is subject to contain tax regulations and privacy legislation requirements in most countries.

   Availability Guarantees: Is the application required to be available per a Service Level Agreement (SLA) or similar guarantee? Is it a nationally protected infrastructure? To what level will the application have to be available? High availability techniques are significantly more expensive, so applying the correct controls up front will save a great deal of time, resources, and money.

2) **Discovery**

   In Web services discovery, participants identify and compose Web Services Description Language (WSDL) which is a specific service based on definitions in a UDDI registry. Due to the potentially large number of service candidates in the registry, performance rankings for algorithms used to search, match and compose services can vary from case to case.

   End to End Quality of Service and Protection

   Most Web services do not provide guarantees for Quality of Service (QoS) or Quality of Protection (QoP). QoS is defined as what the expected level of performance a particular Web service will have.

   Availability and Protection from Denial of Service Attacks

   Availability allows a Web Services Application to detect a Denial of Service (DOS) attack, the operation is continued as long as possible and then to gracefully recover and resume operations after a DOS attack.

3) To mitigate the risks from Web based applications is to use XML firewalls. This allows filtration of XML documents and uses contents in XML

documents to make decisions. For example, it can filter malicious content, or specially crafted parameters that can be used to generate a SQL query by the attacker. Additionally, if it is configured properly, it can act like a WS-Security end point, to verify user signatures and validate user identity.

4) In Web Services interfaces penetration tested are done before they are opened up for external access. These interfaces are quite complex and even experienced development teams may make mistakes which can have security implications.

4.11 SUGGESTED READINGS

- http://msdn.microsoft.com
- http://www.acunetix.com
- http://www.owasp.org
- http://www.technicalinfo.net

Student Satisfaction Survey

Student Satisfaction Survey of IGNOU Students

ignou THE PEOPLE'S UNIVERSITY

Enrollment No.	
Mobile No.	
Name	
Programme of Study	
Year of Enrolment	
Age Group	☐ Below 30 ☐ 31-40 ☐ 41-50 ☐ 51 and above
Gender	☐ Male ☐ Female
Regional Centre	
States	
Study Center Code	

Please indicate how much you are satisfied or dissatisfied with the following statements

Sl. No.	Questions	Very Satisfied	Satisfied	Average	Dissati-sfied	Very Dissati-sfied
1.	Concepts are clearly explained in the printed learning material	☐	☐	☐	☐	☐
2.	The learning materials were received in time	☐	☐	☐	☐	☐
3.	Supplementary study materials (like video/audio) available	☐	☐	☐	☐	☐
4.	Academic counselors explain the concepts clearly	☐	☐	☐	☐	☐
5.	The counseling sessions were interactive	☐	☐	☐	☐	☐
6.	Changes in the counseling schedule were communicated to you on time	☐	☐	☐	☐	☐
7.	Examination procedures were clearly given to you	☐	☐	☐	☐	☐
8.	Personnel in the study centers are helpful	☐	☐	☐	☐	☐
9.	Academic counseling sessions are well organized	☐	☐	☐	☐	☐
10.	Studying the programme/course provide the knowledge of the subject	☐	☐	☐	☐	☐
11.	Assignments are returned in time	☐	☐	☐	☐	☐
12.	Feedbacks on the assignments helped in clarifying the concepts	☐	☐	☐	☐	☐
13.	Project proposals are clearly marked and discussed	☐	☐	☐	☐	☐
14.	Results and grade card of the examination were provided on time	☐	☐	☐	☐	☐
15.	Overall, I am satisfied with the programme	☐	☐	☐	☐	☐
16.	Guidance from the programme coordinator and teachers from the school	☐	☐	☐	☐	☐

After filling this questionnaire send it to:
Programme Coordinator, School of Vocational Education and Training,
Room no. 19, Block no. 1, IGNOU, Maidangarhi, New Delhi- 110068

MPDD-IGNOU/P.O.1T/Feb,2012

ISBN-978-81-266-5892-3