Certified Tester
Advanced Level Syllabus

Security Tester

Version 2016

International Software Testing Qualifications Board

Advanced Level Working Group: Mike Smith (chair)

Advanced Security Tester Syllabus Working Group: Randall Rice (chair), Tarun Banga, Taz Daughtrey, Frans Dijkman, Prof. Dr. Stefan Karsch, Satoshi Masuda, Raine Moilanen, Joel Oliveira, Alain Ribault, Ian Ross, Kwangik Seo, Dave van Stein, Dr. Nor Adnan Yahaya, Wenqiang Zheng.

Revision History

Version	Date	Remarks
0.1	24 April 2015	Baseline version created from existing Expert Security Tester draft syllabus version 3.9
0.2	15 June 2015	Consolidated author input after Oslo author meeting
1.0 - Beta	20 Sept 2015	Beta release – alpha release comments incorporated
1.0 – GA Candidate	4 March 2016	After Exam WG review, changed LO 4.1.2 from K2 and K3 and re-worded appropriately. Text already adequately supports a K3 LO.
1.0 - GA	18 March 2016	GA release – beta release comments incorporated

Table of Contents

Acknowledgements

This document was produced by a core team from the International Software Testing Qualifications Board Advanced Level Working Group.

The core team thanks the review team and all National Boards for their suggestions and input.

At the time the Advanced Level syllabus for this module was completed, the Advanced Level Working Group – Security Tester had the following membership:

The core team authors for this syllabus: Randall Rice (Chair), Hugh Tazwell Daughtrey (Vice Chair), Frans Dijkman, Joel Oliveira, Alain Ribault.

The following persons participated in the reviewing, commenting and balloting of this syllabus (alphabetical order): Tarun Banga, Clive Bates, Hugh Tazwell Daughtrey (Vice-Chair), Frans Dijkman (Author), Christian Alexander Graf, Wenda Hu, Matthias Hamburg, Prof. Dr. Stefan Karsch, Sebastian Malyska, Satoshi Masuda, Gary Mogyorodi, Raine Moilanen, Joel Oliveira, Meile Posthuma, Alain Ribault, Randall Rice (Chair), Ian Ross, Kwangik Seo, Dave van Stein, Ernst von Düring, Attila Toth, Wei Xue, Dr. Nor Adnan Yahaya, Xiaofeng Yang, Wenqiang Zheng, Ping Zuo.

In addition, we recognize and thank the leaders and members of the Expert Level Working Party for their early and continued guidance: Graham Bath (Chair, Expert Level Working Party), Judy McKay (Vice Chair, Expert Level Working Party).

This document was formally released by the General Assembly of ISTQB® on March 18, 2016.

0. Introduction to this Syllabus

0.1 Purpose of this Document

This syllabus forms the basis for the International Software Testing Qualification at the Advanced Level for the Specialist Module "Security Tester". The ISTQB® provides this syllabus as follows:

1. To National Boards, to translate into their local language and to accredit training providers. National boards may adapt the syllabus to their particular language needs and modify the references to adapt to their local publications.
2. To Exam Boards, to derive examination questions in their local language based on the learning objectives for each module.
3. To training providers, to produce courseware and determine appropriate teaching methods.
4. To certification candidates, as one source to prepare for the exam.
5. To the international software and system engineering community, to advance the profession of software and system testing, and as a basis for books and articles.

The ISTQB® may allow other entities to use this syllabus for other purposes, provided they seek and obtain prior written permission.

0.2 Overview

The Advanced Level Security Tester qualification is aimed at people who have already achieved an advanced point in their careers in software testing and wish to develop further their expertise in security testing. The modules offered at the Advanced Level cover a wide range of testing topics.

To receive Advanced Level certification in the module "Security Tester", candidates must hold the Certified Tester Foundation Level certificate and satisfy the Exam Board that examines them that they have sufficient practical experience to be certified at Advanced Level, which should be not less than three years of relevant academic, practical, or consulting experience. Refer to the relevant Exam Board to determine their specific practical experience criteria.

0.3 Examination

All examinations conducted at Advanced Level for this module are based on this Advanced Security Tester syllabus.

The format of the examination is defined by the Advanced Exam Guidelines of the ISTQB.

Exams may be taken as part of an accredited training course or taken independently (e.g., at an examination center). Exams may be taken on paper or electronically, but all exams must be proctored/ observed (supervised by a person mandated by a National or Examination Board).

0.4 How this Syllabus is Organized

There are ten chapters. The top-level heading shows the time for the chapter. For example:

1. The Basis of Security Testing 105 mins.

shows that Chapter 1 is intended to have a time of 105 minutes for teaching the material in the chapter.

Specific learning objectives are listed at the start of each chapter.

0.5 Definitions

Many terms used in the software literature are used interchangeably. Whereas candidates of the Foundation and Advanced Level exams may be asked questions based only on the ISTQB Standard Glossary of Terms, it is expected additionally that candidates at this level should be aware of and able to work with the differing definitions.

NOTE: “Information assurance” (IA) is called out only in section 2.4. A quotation in 2.4.3 is followed by the assertion that IA should be seen as broader than “security testing” in the same way as QA is broader than software testing.

“Information security” is used in sections 2.2, 2.3.1, 2.7.2, 6 (Background), 6.1.3, and throughout Chapter 9.

There is no use of the term “cybersecurity,” which in some quarters is now referred to as IA.

The keywords listed at the start of each chapter in this Advanced Level syllabus are defined either in the Standard Glossary of Terms used in Software Testing, published by the ISTQB, or are provided in the literature referenced.

0.6 Level of Detail

The level of detail in this syllabus allows internationally consistent teaching and examination. In order to achieve this goal, the syllabus consists of:

- General instructional objectives describing the intention of the Advanced Level
- Learning objectives for each knowledge area, describing the cognitive learning outcome and mindset to be achieved
- A list of information to teach, including a description, and references to additional sources if required
- A description of the key concepts to teach, including sources such as accepted literature or standards
- Certain tools, methods and trademarks may be mentioned in this syllabus. This syllabus is not intended to promote or recommend any particular security solution.

The syllabus content is not a description of the entire knowledge area for Advanced Security Testers; it reflects the level of detail to be covered in an Advanced Security Tester training course.

0.7 Learning Objectives / Level of Knowledge

This syllabus’ content, terms and the major elements (purposes) of all standards listed shall at least be remembered (K1) and understood (K2), even if not explicitly mentioned in the learning objectives.

The following learning objectives are defined as applying to this syllabus. Each topic in the syllabus will be examined according to the learning objective for it.

Level 1: Remember (K1)

The candidate will recognize, remember and recall a term or concept.
Keywords: Remember, recall, recognize, know.
Example
Can recognize the definition of "risk" as:

- "a factor that could result in future negative consequences; usually expressed as impact and likelihood."

Level 2: Understand (K2)
The candidate can select the reasons or explanations for statements related to the topic, and can summarize, differentiate, classify and give examples for facts (e.g., compare terms), the testing concepts, test procedures (explaining the sequence of tasks).
Keywords: Summarize, classify, compare, map, contrast, exemplify, interpret, translate, represent, infer, conclude, categorize.
Examples
Explain the reason why security tests should be designed as early as possible:

- To find security defects and vulnerabilities when they are cheaper to address
- To avoid building a system or application that is prone to continual patching of security vulnerabilities

Level 3: Apply (K3)
The candidate can select the correct application of a concept or technique and apply it to a given context. K3 is normally applicable to procedural knowledge. There is no creative act involved such as evaluating a software application, or creating a model for a given software program. When a model is supplied, the syllabus explains the procedural steps required to create test cases from that model, then it is K3.
Keywords: Implement, execute, use, follow a procedure, apply a procedure.
Example

- Use the generic procedure for security test case creation to select the test cases from a given state transition diagram in order to cover all transitions.

Level 4: Analyze (K4)
The candidate can separate information related to a procedure or technique into its constituent parts for better understanding, and can distinguish between facts and inferences. Typical application is to analyze a document, software, project situation and propose appropriate actions to solve a problem or task.
Keywords: Analyze, differentiate, select, structure, focus, attribute, deconstruct, evaluate, judge, monitor, coordinate, create, synthesize, generate, hypothesize, plan, design, construct, produce.
Example

- Analyze product security risks and propose preventive and corrective mitigation activities.
- Select security test tools that would be most appropriate in a given situation with past security failures.

Reference (For the cognitive levels of learning objectives)
Bloom, B. S. (1956). *Taxonomy of Educational Objectives, Handbook I: The Cognitive Domain,* David McKay, Co. Inc.
Anderson, L. W. and Krathwohl, D. R. (eds) (2001). *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*, Allyn & Bacon.

1 The Basis of Security Testing - 105 mins.

Keywords

data privacy, ethical hacker, information security, penetration testing, risk assessment, risk exposure, risk mitigation, security attack, security audit, security policy, security procedure, security risk

Learning Objectives for The Basis of Security Testing

1.1 Security Risks

AS-1.1.1 (K2) Understand the role of risk assessment in supplying information for security test planning and design and aligning security testing with business needs

AS-1.1.2 (K4) Identify the significant assets to be protected, the value of each asset and the data required to assess the level of security needed for each asset

AS-1.1.3 (K4) Analyze the effective use of risk assessment techniques in a given situation to identify current and future security threats

1.2 Information Security Policies and Procedures

AS-1.2.1 (K2) Understand the concept of security policies and procedures and how they are applied in information systems

AS-1.2.2 (K4) Analyze a given set of security policies and procedures along with security test results to determine effectiveness

1.3 Security Auditing and Its Role in Security Testing

AS-1.3.1 (K2) Understand the purpose of a security audit

Functional testing is based on a variety of items, such as risks, requirements, use cases, and models. Security testing is based on the security aspects of those specifications but also seeks to verify and validate security risks, security procedures and policies, attacker behavior and known security vulnerabilities.

1.1 Security Risks

1.1.1 The Role of Risk Assessment in Security Testing

Security test objectives are based on security risks. These risks are identified through performing a security risk assessment. General risk management techniques are described in [ISTQB_FL_SYL] and [ISTQB_ATM_SYL].

Risk is a measure of the extent to which an entity is threatened by a potential circumstance or event, and is typically a function of:

- The adverse impacts that would arise if the circumstance or event occurs, and
- The likelihood of occurrence.

Information security risks are those risks that arise from the loss of confidentiality, integrity, or availability of information or information systems and reflect the potential adverse impacts to organizational operations (i.e., mission, functions, image, or reputation), organizational assets, individuals, other organizations, and a country. [NIST 800-30]

The role of a security risk assessment is to allow an organization to understand which areas and assets may be at risk, and to determine the magnitude of each risk. For security testers, a security risk assessment can be a rich source of information from which security tests can be planned and designed. In addition, a security risk assessment can be used to prioritize security tests so that the highest level of test rigor and coverage can be focused on the areas with the great risk exposure.

By prioritizing security tests based on a security risk assessment, the tests become aligned with business security objectives. However, for this alignment to occur, the security risk assessment must accurately reflect the organization's security threats, the impacted stakeholders and the assets to be protected.

It is important to understand that any risk assessment (security or otherwise) is only a snapshot at a given point in time and based on limited information that may lead to invalid assumptions and conclusions. Security risks change continually within an organization and projects since new threats emerge daily. Therefore, security risk assessments should be performed at regular intervals. The exact time interval for performing security risk assessments varies based on the organization and the degree of change it experiences. Some organizations perform security risk assessments on a three to six month basis, while others perform them on an annual basis.

Another issue with risk assessments is the level of knowledge of the participants. Risks can be missed due to a lack of detailed information. In addition, risks can be missed if people do not understand security threats and risks. For this reason, it is good to solicit input from a variety of people and pay careful attention to the level of detail contained in the information they provide.

It is a realistic expectation that wrong assumptions may be made which can lead to important security risks being missed in the assessment. Ways to deal with the possibility of missing or incomplete risk information include using an established security risk assessment methodology as a checklist and getting input from multiple people. One such methodology can be found in [NIST 800-30].

1.1.2 Asset Identification

Not all information to be secured is in digital format, such as copied documents (contracts, plans, written notes, logins and passwords in written form). Although not in digital format, this information may have high value. Therefore, the question needs to be asked, which information is digital and which is not? Perhaps the asset to be protected exists both in digital and physical format. When identifying the assets to be secured, the following questions should be asked:

Which assets are valuable to the organization?
Examples of high value sensitive information include:

- Customer data
- Business plans
- Proprietary software developed by the company
- System documentation
- Pictures and diagrams that are the property of the company
- Intellectual property (e.g., processes, trade secrets)
- Financial spreadsheets
- Presentations and training courses
- Documents
- Emails
- Employee records
- Tax statements

While many assets are information-based, it is possible that some assets in an organization are of a physical or intangible nature. Examples of these assets would include:

- Physical prototypes of new devices under development
- The ability to deliver services
- Company reputation and trust

How valuable is the asset?
Many sensitive assets have a tangible value. Others are measured more in the costs and consequences of their loss. For example, what would a competitor do with a rival's business plan?

Value can be difficult to assess with certainty; however, some methods to determine the value of digital assets include:

- The future revenue to be generated by the asset
- The value to a competitor who may obtain the information
- The time and effort needed to recreate the asset
- Fines and penalties for not being able to produce the information when needed, e.g., for an audit or lawsuit
- Fines and penalties for the loss of customer data

Where are the digital assets located?
In the past, digital assets resided on servers, desktop computers or peripherals such as disks or CDs. While this is an outdated and disorganized approach, there may still be sensitive data on old CDs, DVDs and USB drives. A more secure means of storing digital assets is the use of secure enterprise servers, using strong encryption for all sensitive data. To access sensitive data stored on secure severs, authentication and authorization should be required. In addition, other security protection may be needed, such as digital certificates for accessing sensitive information over the Internet.

Storage is changing. Now large amounts of business data can exist on mobile devices such as smart phones, tablets and thumb drives. When digital information has been migrated to cloud storage, there is a new set of security concerns based on data access.

The importance of the issue of data storage arises from past cases where people entrusted with sensitive data simply walked out of a company building with a hard drive filled with private customer and business data. One such case in the United States involved a hard drive stolen from a secured area at a government security agency, which included payroll and bank information for over 100,000 current and former workers. [Washington Post, 2007].

How are the digital assets accessed?
Common methods for accessing digital assets include:
- Computer access over a Local Area Network or Wi-Fi networks
- Remote access through a Virtual Private Network (VPN) or Cloud drive
- Passing physical data stores (CDs, DVDs, USB drives) from person to person which is a low-tech but very common practice
- Sending files by email

How are the digital assets secured?
There are a number of ways of securing digital assets including:
- Encryption (What type and strength, who has the keys?)
- Authentication and tokens (Are digital certificates required? Are password policies adequate and followed?)
- Authorization (Which levels of privilege have been granted to users who deal with digital assets?)

1.1.3 Analysis of Risk Assessment Techniques

The security risk assessment process is very similar to a standard risk assessment, with the main difference being the focus on security-related areas.

A security risk assessment should include the perspectives of external security testing stakeholders (i.e., people or parties involved in the project or with the product who are outside of the company and have a clear stake in the security of the project/product). These stakeholders include:
- Customers and users – useful for understanding perspective, getting input for security testing, and establishing good communication
- The public and society – important to convey that information security is a community effort and responsibility
- Regulatory agencies – necessary for ensuring compliance with applicable laws regarding information security

Preparing for a risk assessment includes the following tasks [NIST 800-30]:
- Identify the purpose of the assessment
- Identify the scope of the assessment
- Identify the assumptions and constraints associated with the assessment
- Identify the sources of information to be used as inputs to the assessment
- Identify the risk model and analytic approaches (i.e., assessment and analysis approaches) to be employed during the assessment

Conducting risk assessments includes the following specific tasks [NIST 800-30]:
- Identify threat sources that are relevant to the organization
- Identify threat events that could be produced by those sources

- Identify vulnerabilities within the organization that could be exploited by threat sources through specific threat events and the predisposing conditions that could affect successful exploitation
- Determine the likelihood that the identified threat sources would initiate specific threat events and the likelihood that the threat events would be successful
- Determine the adverse impacts to organizational operations and assets, individuals, other organizations, and the nation resulting from the exploitation of vulnerabilities by threat

Communicating and sharing information consists of the following specific tasks [NIST 800-30]:

- Communicate the risk assessment results
- Share information developed in the execution of the risk assessment to support other risk management activities

1.2 Information Security Policies and Procedures

1.2.1 Understanding Security Policies and Procedures

It is common for information security policies to vary among organizations based on the business model, the specific industry and the unique security risks faced by the organization. Even with a wide range of variation, the goals of security policies are similar. The basis of all security policies should be a security risk assessment that examines specific security threats and how they impact the organization. [Jackson, 2010]

Security policy examples include, but are not limited to [Jackson, 2010]:

Acceptable Use – This policy defines practices that a user of a computer system must adhere to in order to be compliant with the organization's security policies and procedures. This policy covers both acceptable and non-acceptable behavior in using digital resources, such as networks, websites and data. Also, the policy can apply to both internal and external users of an organization's systems. It is important for users of the system to understand and follow the policy at all times. To prevent confusion and accidental violations of the policy, it should define specific rules concerning acceptable behavior, unacceptable behavior and required behavior.

Minimum Access – This policy defines the minimum levels of access needed to perform certain tasks. The goal of this policy is to prevent people from being granted access rights in excess of what is needed to perform their tasks. Having access rights higher than needed can provide opportunities for inadvertent or intentional abuse of user privileges.

Network Access – This policy defines criteria for accessing various types of networks such as local area networks (LANs) and wireless networks. In addition, this policy can define what is permissible and non-permissible while on the network. This policy often prohibits users from adding unauthorized devices such as routers and hot spots to the network.

Remote Access – This policy requires what is needed so that remote network access can be granted to both internal employees and external (non-employee) users. VPN usage is often covered in this policy.

Internet Access – This policy defines permissible usage of the Internet by employees and guests of an organization. The scope of this policy includes the types of websites that may and may not be accessed, such as gambling or pornography sites, and also addresses whether non-business use of the Internet is allowed. Although some of the items covered in the policy may also be addressed in the acceptable use policy, some organizations choose to define this policy separately because of the number of people doing business on the Internet.

User Account Management – This policy defines the creation, maintenance and deletion of user accounts. Regular auditing of user accounts is also covered in this policy to ensure compliance to the policy.

Data Classification – There are many ways to classify data from a security perspective. In this syllabus, the term "sensitive data" is used as a general term for any data that must be secured to avoid loss. A data classification policy defines the different types of data that are considered sensitive and must be secured. By having a data classification policy, an organization can create controls to protect the data based on its value to the organization and its customers. Typically, the business area that creates the data is responsible for its classification based on a standard classification structure.

The following is an example data classification structure (from a business context):

- Public: Anyone, either inside or outside of the organization, can view this data (e.g., outward facing documents and webpages).
- Confidential: This is normally the default classification for any internally created documents. These documents can include email, reports, and presentations that are used internally in the organization. An example of this would be a sales report. Only authorized users of this data should be able to work with this level of information. Non-disclosure agreements are often required before sharing this type of information with third parties such as consultants.
- Highly confidential: This is a higher level of confidentiality for sensitive information that should be available to only certain people in the organization. This would include information such as trade secrets, strategic plans, product designs and nonpublic financial data. Sharing of this type of data is not allowed except with explicit permission of the data owner.
- Private: This is information that is often restricted to officers of the organization who must be specifically authorized to have access to it. If disclosed, this information could have major negative impact, such as financial harm, to the organization. Because of the high risk associated with loss, private information must be protected with extreme care. This data can include research and development information, merger and acquisition plans, as well as customer information such as credit card and account information.
- Secret: In the corporate context, this is information an organization gets from an external party to make changes, but that is not allowed to become known inside or outside the organization. An example in the corporate context would be a design document created by a consultant working on a new type of technology that involves the collaboration of other companies, each of which must keep the information at a secret level until the technology is ready to be revealed. It is comparable to highly confidential with the difference that it may have no tangible value for the organization itself. In that regard, it is different from a trade secret. However, the disclosure of secret information could cause harm to the organization, other organizations or the country. In the military and governmental context, this is information that may be developed or obtained, but must be known only to people with certain levels of security clearance. In the military context, this would include particulars of scientific or research projects that incorporate new technological developments or techniques having direct military applications of vital importance to the national defense.

Configuration and Change Management – This policy can have a normal operational context, such as describing how changes to systems are managed and configured, in order to prevent outages due to unexpected impact. From a security perspective, configuration management controls how security

settings are applied to secure devices and applications. The risk is that an unauthorized change to a secure device could cause a security vulnerability that might go undetected.

Another risk is that an unauthorized change to the code or application configuration could create a security vulnerability. This policy includes standard configurations to be used, an approval process for all changes, and a rollback process if problems occur. This policy can apply to all IT services, applications and devices in an organization.

Server Security – This policy conveys responsibility to the server owner(s) to follow corporate security practices, as well as industry best practices for installing, configuring and operating servers and systems. In addition, baseline configurations are mandated to be defined and maintained. Examples of practices described in this policy include security requirements, backup and recovery, and limiting active services to those necessary for running applications. Also included in this policy may be monitoring and auditing requirements to ensure the server is configured and updated correctly.

Mobile Devices – Mobile devices have a unique set of security concerns, so therefore a separate policy may be needed just for mobile devices. For example, laptops and smart phones can be easily lost or stolen, which could result in loss of company and private data. These devices also have a high risk of contact with malware. These risks require specific rules and precautions that must be followed to mitigate the risks and limit the organization's exposure to security threats. This policy may include requirements for which data must be encrypted, the installation and maintenance of current versions of anti-malware software, and when passwords are needed to access the device. Also, the types of organizational information that can reside on mobile devices are defined in this policy. Physical security can also be addressed, such as having cable locks for laptop computers and having procedures for reporting lost or stolen devices.

Guest Access – This policy defines the practices that should be in place to protect the organization, while allowing the company to host guests and others on organizational networks. One aspect of this policy is to require guests to read and agree with acceptable use policies before granting network access to them. This policy can be implemented in a variety of ways, such as having guests sign an acceptable usage policy and then providing a code for temporary access. The main intent of this policy is to enforce the organization's security standards and still provide procedures for allowing guests to access the network or Internet.

Physical Security – This policy defines the controls needed for physical facilities since being in physical proximity to secure devices can increase the risk of a security breach. This policy can also cover other risks, such as power loss, theft, fire, and natural disasters. Also addressed here is which devices may be taken out of or brought into the company, especially for areas that house sensitive information.

Password Policy – This policy defines the minimum requirements for strong passwords and other secure password practices, such as the length of time allowed between mandatory password changes, how people protect the privacy of their passwords (such as not using the "remember password" feature in browsers, prohibiting the sharing of passwords, and prohibiting the transmission of passwords by email). This policy can apply to applications, user accounts and any other places where passwords are needed.

Malware Protection – This policy defines a framework of defenses and behaviors for preventing, detecting and removing malware. Since malware and spyware can be picked up from a variety of sources, this is an important policy for everyone in the organization to understand and follow. For example, this policy might restrict the use of USB drives.

Incident Response – This policy describes how to respond to a security-related incident. These incidents can range from the discovery of malware and violations of the acceptable use policy to the unauthorized access of sensitive data. It is important to have this policy in place before an incident happens to avoid having to determine appropriate responses on a case-by-case basis. This policy also addresses communication, including media responses and law enforcement notification.

Audit Policy – This policy authorizes auditors to request access to systems for the purpose of conducting an audit. The audit team might need access to log data, network traffic records and other forensic data.

Software Licensing – This policy addresses how the organization obtains and licenses the software it uses. If commercial software licenses are violated, the organization is at risk of fines and legal action. Because of this, it is important that licenses are identified and monitored. Downloading and installing unapproved software is a key prohibition often found in this policy.

Electronic Monitoring and Privacy – Organizations have a right and responsibility to monitor the electronic communications across company hardware and resources. This includes email correspondence and social media. This policy outlines which monitoring is performed by the organization and which data is subject to collection. Laws vary between countries, so legal counsel is needed before writing this policy. [Jackson, 2010]

Security Procedures
Security procedures specify the steps to be taken in implementing a specific policy or control, and the steps to be taken in response to a specific security incident. Formal, documented procedures facilitate implementation of the security policies and mandated controls.

Policies, standards and guidelines describe security controls that should be in place, while a procedure describes specifics, explaining how to implement security controls in a step by step manner. For example, a procedure could be written to explain how to grant user access levels, detailing each step that needs to be taken to ensure the correct level of access is granted so that the user rights satisfy the applicable policy, standards and guidelines.

1.2.2 Analysis of Security Policies and Procedures

Before evaluating a set of security policies and procedures it is important to determine the objective(s) of the evaluation and define a set of criteria by which to judge the adequacy of the policies and procedures. In some cases, the criteria may be defined by standards such as COBIT [COBIT], ISO27001 [ISO27001] or PCI [PCI].

In addition, it is necessary to define:

- Which resources are needed in terms of skills and knowledge in particular areas being evaluated
- How to measure the adequacy of the policies and procedures
- What to measure and assess (e.g., effectiveness, efficiency, usability, adoption)
- Where the policies and procedures are located in the organization
- A checklist to guide the assessment and add consistency

The checklist acts as a guide that directs the auditor where to look and what to expect. Tools, such as password audit tools, may be helpful in testing certain controls to determine if they are accomplishing their goals and to generate data that can be used later in risk assessment. The auditor looks to find “proof” compliance to policies, controls and standards. Some of the tasks in the following list are static

in nature while others, such as observing processes in action, are dynamic. The auditor does the following:

- Reviews system documentation
- Surveys people on their perception of the effectiveness of policies and procedures
- Interviews key personnel that are involved in the processes being controlled
- Witnesses systems and processes being performed
- Analyzes previous audit results to discover trends
- Analyzes logs and reports
- Reviews technical control configuration, such as firewall configuration and intrusion detection system configuration
- Samples data transactions for any anomalies or suspicious transactions [Jackson, 2010]

Controls

Security controls are technical or administrative safeguards or countermeasures to avoid, counteract or minimize loss or unavailability due to threats acting on their matching vulnerability, i.e., security risk. [Northcutt, 2014] For example, a security control in a payroll system might be that two people must separately approve a change to an employee's pay rate information. Security testers must be aware of the specific controls in their organization and include tests for them in the security tests.

The main security control types are administrative, technical, and physical. Under each category, the specific controls that can be implemented are preventive, detective, corrective, or recovery. These control types work together, and in general, controls must be provided from each category to effectively protect an asset. [Jackson, 2010]

A list of the Top 20 Critical Security Controls can be found at www.sans.org. [Web-1]

Security Tests

The primary difference in security testing as compared to a static analysis of security policies and procedures is the use of the results from tests designed specifically to verify or validate the effectiveness of security policies and procedures. These tests focus on the risk that a security policy may be in place, may be followed, but is not effective in protecting assets.

It is also possible to be told while performing security policy and procedure assessments that certain tasks are performed. A security test of those tasks can help determine how effective the security policies and procedures actually are in practice. For example, a password policy and procedure may seem reasonable and effective on paper, but when a password cracking tool is used, the procedure may fall short of its goals.

Security policies and procedures can be a source of security tests; however, the security tester must keep in mind that attacks are always evolving. New attacks emerge and just like in any software application, new defects can become evident – all of which are reasons to perform security tests from the attacker mindset.

1.3 Security Auditing and Its Role in Security Testing

Security auditing is a manual examination and evaluation that identifies weaknesses in an organization's security processes and infrastructure. Security audits at the procedural level (e.g., to review internal controls) may be performed manually. Security audits at the architectural level are often performed with security audit tools, which may be aligned with a particular vendor solution for networking, server architecture and workstations.

Just like security testing, a security audit does not guarantee all vulnerabilities will be found. However, the audit is one more activity in the security process to identify problem areas and indicate where remediation is needed.

In some security auditing approaches, testing is performed as part of the auditing process. However, the scope of security auditing is much larger than security testing. Security auditing often investigates areas such as procedures, policies and controls that are difficult to test in a direct way. Security testing is more involved with the technologies to support security, such as firewall configuration, correct application of authentication and encryption, and application of user rights.

There are five pillars to security auditing [Jackson, 2010]:

Assessment – Assessments document and identify potential threats, key assets, policies and procedures, and management's tolerance for risk. Assessments are not one-time events. Since the environment and business are constantly in flux, assessments must be performed on a regular basis. This also provides the opportunity to know if security policies are still relevant and effective.

Prevention – This extends beyond technology and includes administrative, operational, and technical controls. Prevention is not accomplished just through technology, but also through policies, procedures, and awareness. While the prevention of any and all attacks is unrealistic, the combination of defenses can help make it much more difficult for an attacker to succeed.

Detection – Detection is how a security breach or intrusion is identified. Without adequate detection mechanisms, there is the risk of not knowing whether the network has been compromised. Detective controls help to identify security incidents and provide visibility into activities on the network. Early incident detection enables an appropriate reaction to recover services quickly.

Reaction – Reaction time is greatly reduced with good security defenses and detection mechanisms. While security breaches are bad news, it is important to know if one has occurred. Fast reaction time is critical to minimize the exposure to the incident. Fast reaction requires good preventive defenses and detection mechanisms to provide the data and context needed for response. The speed and efficiency of incident response is a key indicator of the effectiveness of an organization's security efforts.

Recovery – Recovery starts with determining what occurred so that systems can be recovered without recreating the same vulnerability or condition that caused the incident in the first place. The recovery phase does not end with restoring the system. There is also the root cause analysis that determines what changes need to be made to processes, procedures, and technologies to reduce the likelihood of the same type of vulnerability in the future. An auditor must ensure that the organization has a plan for recovery that includes ways to prevent future similar incidents.

1.3.1 Purpose of a Security Audit

The following is a list of items that might be detected in a security audit:

- Inadequate physical security. A security policy might require encryption of all customer data, both in storage and in transmission. For example, during the audit, it is discovered that once every week, a customer information file is sent by physical report to all managers. This report is discarded each week, but it is discovered that some managers are carelessly throwing the physical reports in the trash where they can be found by anyone who might want to go through the trash (i.e., "dumpster diving").
- Inadequate password maintenance. A security policy might require each user to change their password every 30 days. A security audit reveals that passwords are changed, but many users simply alternate between "PasswordA" and "PasswordB" each month. (Password history is a common feature in password auditing tools.)

- Inadequate controls for user rights and sharing privileges. An example of a negative finding might be where users have been granted more access rights to items than they need to perform their job. Another example might be where an individual user's files are shared on the network when they should be private. This is a particular concern for users with notebook PCs and especially those that access the intranet through Wi-Fi connections at home or in public locations.
- Inadequate server-level security. Specific audit areas include:
  - Port allocation and security
  - Protection of data
  - Protection of user accounts (logins and other sensitive information)
- Inadequate application of vendor security updates
- Inadequate intrusion detection mechanisms
- Inadequate response plans in the event of a security breach

1.3.2 Risk Identification, Assessment and Mitigation

Once the problem areas have been identified by the audit, risk must be evaluated and an improvement plan put in place. The auditor's report may include recommendations as well as other areas of risk. From this point, risk identification, assessment and mitigation activities can be planned.

Risk identification is the process of documenting a risk or an area of risk. In the context of IT security, the risks are security-related. Risk assessment is the activity that assigns a value to the identified risks. It is important to understand that traditional IT risk assessment models are not sufficient to address IT security risks. Any security risk assessment model or approach should be specifically oriented to IT security risk profiles.

Security risks often are measured in terms of risk exposure. Risk exposure is computed by multiplying the potential impact or loss by the likelihood of that loss occurring. For example, if one customer's account information is compromised, what would be the impact? What if that customer had $100 million of assets on deposit?

The likelihood of occurrence can be determined by applying a security risk assessment model such as that found in NIST Publication 800-30, Guide for Conducting Risk Assessments [NIST 800-30]. Another excellent guide to performing security risk assessments is the OWASP Risk Rating Methodology [OWASP2]. The following information is abstracted from [NIST 800-30].

Risk models define the risk factors to be assessed and the relationships among those factors. Risk factors are characteristics used in risk models as inputs to determining levels of risk in risk assessments. Risk factors are also used extensively in risk communications to highlight what strongly affects the levels of risk in particular situations, circumstances, or contexts.

Typical risk factors include threat, vulnerability, impact, likelihood, and predisposing condition. Risk factors can be decomposed into more detailed characteristics (e.g., threats decomposed into threat sources and threat events). These definitions are important for organizations to document prior to conducting risk assessments because the assessments rely upon well-defined attributes of threats, vulnerabilities, impact, and other risk factors to effectively determine risk.

Threats

A threat is any circumstance or event with the potential to adversely impact organizational operations and assets, individuals, other organizations, or a country through an information system via unauthorized access, destruction, disclosure, or modification of information, and/or denial of service.

Threat events are caused by threat sources. A threat source is characterized as:
- The intent and method targeted at the exploitation of a vulnerability; or
- A situation and method that may accidentally exploit a vulnerability.

In general, types of threat sources include:
- Hostile cyber or physical attacks
- Human errors of omission or commission
- Structural failures of organization-controlled resources (e.g., hardware, software, environmental controls)
- Natural and man-made disasters, accidents, and failures beyond the control of the organization.

Various taxonomies of threat sources have been developed. Some taxonomies of threat sources use the type of adverse impacts as an organizing principle. Multiple threat sources can initiate or cause the same threat event—for example, a provisioning server can be taken offline by a denial-of-service attack, a deliberate act by a malicious system administrator, an administrative error, a hardware fault, or a power failure.

Vulnerabilities and Predisposing Conditions
A vulnerability is a weakness in an information system, system security procedures, internal controls, or implementation that could be exploited by a threat source.

Most information system vulnerabilities can be associated with security controls that either have not been applied (either intentionally or unintentionally), or have been applied but retain some weakness. However, it is also important to allow for the possibility of emergent vulnerabilities that can arise naturally over time as organizational missions/business functions evolve, environments of operation change, new technologies proliferate, and new threats emerge. In the context of such changes, existing security controls may become inadequate and may need to be reassessed for effectiveness. The tendency for security controls to potentially degrade in effectiveness over time reinforces the need to maintain risk assessments during the entire software lifecycle and also the importance of continuous monitoring programs to obtain ongoing situational awareness of the organizational security posture.

Vulnerabilities are not identified only within information systems. Viewing information systems in a broader context, vulnerabilities can be found in organizational governance structures (e.g., the lack of effective risk management strategies and adequate risk framing, poor intra-agency communications, inconsistent decisions about relative priorities of missions/business functions, or misalignment of enterprise architecture to support mission/business activities).

Vulnerabilities can also be found in external relationships (e.g., dependencies on particular energy sources, supply chains, information technologies, and telecommunications providers), mission/business processes (e.g., poorly defined processes or processes that are not risk-aware), and enterprise/information security architectures (e.g., poor architectural decisions resulting in lack of diversity or resiliency in organizational information systems).

Impact
The level of impact from a threat event is the magnitude of harm that can be expected to result from the consequences of unauthorized disclosure of information, unauthorized modification of information, unauthorized destruction of information, or loss of information or information system availability. Such harm can be experienced by a variety of organizational and non-organizational stakeholders including:
- Heads of agencies
- Mission and business owners
- Information owners/stewards

- Mission/business process owners
- Information system owners
- Individuals/groups in the public or private sectors relying on the organization—in essence, anyone with a vested interest in the organization's operations, assets, or individuals, including other organizations in partnership with the organization, or a country

The following information should be explicitly documented by an organization:
- The process used to conduct impact determinations
- Assumptions related to impact determinations
- Sources and methods for obtaining impact information
- The rationale for conclusions reached with regard to impact determinations

Organizations may explicitly define how established priorities and values guide the identification of high-value assets and the potential adverse impacts to organizational stakeholders. If such information is not defined, priorities and values related to identifying targets of threat sources and associated organizational impacts can typically be derived from strategic planning and policies. For example, security categorization levels indicate the organizational impacts of compromising different types of information.

Likelihood
The likelihood of occurrence addresses the probability (or possibility) that the threat event will result in an adverse impact, regardless of the magnitude of harm that can be expected. This is a weighted risk factor based on an analysis of the probability that a given threat is capable of exploiting a given vulnerability (or set of vulnerabilities). The likelihood risk factor combines an estimate of the likelihood that the threat event will be initiated with an estimate of the likelihood of impact (i.e., the likelihood that the threat event will result in adverse impacts).

For adversarial threats, an assessment of likelihood of occurrence is typically based on:
- Adversary intent
- Adversary capability
- Adversary targeting

For other than adversarial threat events, the likelihood of occurrence is estimated using historical evidence, empirical data, or other factors. Note that the likelihood that a threat event will be initiated or will occur is assessed with respect to a specific time frame (e.g., the next six months, the next year, or the period until a specified milestone is reached).

If a threat event is almost certain to be initiated or occur in the (specified or implicit) time frame, the risk assessment may take into consideration the estimated frequency of the event. The likelihood of threat occurrence can also be based on the state of the organization (including, for example, its core mission/business processes, enterprise architecture, information security architecture, information systems, and environments in which those systems operate. Predisposing conditions and the presence and effectiveness of deployed security controls to protect against unauthorized/undesirable behavior, detect and limit damage, and/or maintain or restore mission/business capabilities should also be taken into consideration.

Determining the Level of Security Risk
The likelihood of occurrence assessment and the impact assessment can be combined to calculate an overall severity for the risk. Specific assessment scores may be used as the basis of the completing the risk matrix. In other cases, estimates (low, medium or high) may be used.

The scoring for the risk matrix can be based on a scale of 0 – 9, where the numeric values are determined by specific criteria. For example, risk likelihood criteria could be evaluated for data privacy as:

0 - <3 (Low) Private data is not stored on local devices and is encrypted when stored on secure media.
3 - <6 (Medium) Private data may reside on devices such as notebook computers, but is encrypted.
6 - 9 (High) It is not known exactly if private data resides on local devices. Encryption cannot be assured.

Likewise, risk impact criteria could be assessed on the same 0 – 9 scale based on specific criteria. For example:

0 - <3 (Low) Compromise of private data would impact fewer than 200 people.
3 - <6 (Medium) Compromise of private data would impact between 200 and 1,000 people.
6 - 9 (High) Compromise of private data would impact over 1,000 people.

However the tester arrives at the likelihood and impact estimates, the estimates can be combined into a final severity rating for the risk item. If there is good business impact information, that should be used instead of the technical impact information. If there is no information about the business, then technical impact is the next best thing.

Below is a sample view of a risk matrix that can be used to determine the severity of individual risks.

Overall Risk Severity				
Risk Impact	High	Medium	High	Critical
	Medium	Medium	Medium	High
	Low	Low	Low	Medium
		Low	Medium	High
		Risk Likelihood		

In the example matrix above, if the likelihood is medium and the impact is high, the overall severity is high.

In addition, the risk assessment report should identify if the risk is ongoing. Continual risks indicate an increased likelihood that a loss will occur.

The severity of a risk determines the relative importance of mitigating the risk. The higher the risk severity, the more immediate the requirement for the response. The level of detail provided in any particular risk assessment is consistent with the purpose of the risk assessment and the type of inputs needed to support follow-on likelihood and impact determinations.
.

1.3.3 People, Process and Technology

There are also three components to an organization's IT practices: people, process and technology. All of these have an impact on security. According to Chris Jackson in his book, *Network Security Auditing* [Jackson, 2010], "All security incidents, from break-ins to lost customer records, can usually be traced back to a deficiency that can be attributed to people, process, or technology."

People: People can include end-users, system administrators, data owners, and managers of the organization. Each person has varying levels of skills, attitudes, and agendas, which has an influence on how security impacts them, and how they impact the effectiveness of security controls. Regardless of the presence of security policies, procedure and controls, they will be ineffective if people do not

follow them. If people do not follow security policies, there may be a need for remediation efforts, such as the need for security awareness training or penalties for noncompliance. Organizational structures and security policies are often driven by people, both internal and external to an organization.

Process: Processes define how IT services, including security-related services, are delivered. In a security context, processes include the procedures and standards that are put into place to protect valuable assets. To be effective, processes must be defined, up to date, consistent, and follow best practices for security. Processes define roles and responsibilities, controls, tools, and specific steps involved in performing a task.

Technology: Technology encompasses the facilities, equipment, computer hardware, and software that automate or support a business. Technology enables people to accomplish repetitive jobs faster and with fewer errors than if performed manually without it. In fact, some tasks such as password enforcement would be impossible without the right tools. The risk is that technology used incorrectly can help people make mistakes faster.

These three areas can be thought of as an “iron triangle” which together forms a complete IT solution. If any of the three areas are ignored, the entire IT delivery and security effort suffers.

When evaluating security controls, the auditor should look at a system from the perspective of an attacker and anticipate how people, process or technology could be exploited to gain unauthorized access to valuable assets. Management in organizations is often surprised that the security mechanisms they thought were secure, are not. The only way to know for sure if a particular security defense is working and effective is to test the system from an attacker perspective. This is often known as ethical hacking or penetration (pen) testing.

This is where the relationship between auditing and testing becomes the most direct. Auditing identifies deficiencies and areas of importance to test. Security testing is the means by which it is proven or disproven that the security controls are actually in place and working effectively.

Example scenario:

The tax agency for a country is the subject of a security audit. One of the audit findings is that it is possible for criminals to file a fraudulent tax return and obtain the tax refunds due the defrauded taxpayer. This audit finding is confirmed with security testing and risk is rated as “critical”. The tax agency acknowledges the possibility of such fraudulent risk, but decides not to act on the risk until the following year.

Defrauded taxpayers, who have followed all the prescribed security procedures, can file a claim with the tax agency that was aware of the defect in the tax filing process. In this case, the tax agency would be liable for the fraud.

2. Security Testing Purposes, Goals and Strategies - 130 mins.

Keywords
cross-site scripting, data obfuscation, denial of service, information assurance, security policy, security testing, security vulnerability, software lifecycle, test strategy

Learning Objectives for Security Test Purposes, Goals and Strategies

2.1 Introduction
No learning objectives for this section.

2.2 The Purpose of Security Testing
AS-2.2.1 (K2) Understand why security testing is needed in an organization, including benefits to the organization such as risk reduction and higher levels of confidence and trust

2.3 The Organizational Context
AS-2.3.1 (K2) Understand how project realities, business constraints, software development lifecycle, and other considerations affect the mission of the security testing team

2.4 Security Testing Objectives
AS-2.4.1 (K2) Explain why security testing goals and objectives must align with the organization's security policy and other test objectives in the organization
AS-2.4.2 (K3) For a given project scenario, demonstrate the ability to identify security test objectives based on functionality, technology attributes and known vulnerabilities
AS-2.4.3 (K2) Understand the relationship between information assurance and security testing

2.5 The Scope and Coverage of Security Testing Objectives
AS-2.5.1 (K3) For a given project, demonstrate the ability to define the relationship between security test objectives and the need for strength of integrity of sensitive digital and physical assets

2.6 Security Testing Approaches
AS-2.6.1 (K4) Analyze a given situation and determine which security testing approaches are most likely to succeed
AS-2.6.2 (K4) Analyze a situation in which a given security testing approach failed, identifying the likely causes of failure
AS-2.6.3 (K3) For a given scenario, demonstrate the ability to identify the various stakeholders and illustrate the benefits of security testing for each stakeholder group

2.7 Improving the Security Testing Practices
AS-2.7.1 (K4) Analyze KPIs (key performance indicators) to identify security testing practices needing improvement and elements not needing improvement

2.1 Introduction

Before applying specialized security test techniques, it is important to understand the broader context of security testing and its role within a particular organization. This understanding answers the following questions:

- Why is security testing needed?
- What is the purpose of security testing?
- How does security testing fit into the organization?

Security testing differs from other forms of functional testing in two significant areas [ISTQB_ATTA_SYL]:

1. Standard techniques for selecting test input data may miss important security issues
2. The symptoms of security defects are very different from those found with other types of functional testing

Security testing assesses a system's vulnerability to threats by attempting to compromise the system's security policy. The following is a list of potential threats, which should be explored during security testing [ISTQB_ATTA_SYL]:

- Unauthorized copying of applications or data
- Unauthorized access control (e.g., ability to perform tasks for which the user does not have rights). User rights, access and privileges are the focus of this testing. This information should be available in the specifications for the system.
- Software that exhibits unintended side effects when performing its intended function. For example, a media player that correctly plays audio but does so by writing files out to unencrypted temporary storage exhibits a side effect, which may be exploited by software pirates.
- Code inserted into a web page that may be exercised by subsequent users (cross-site scripting or XSS). This code may be malicious.
- Buffer overflow (buffer overrun) that may be caused by entering data strings into a user interface input field that are longer than the code can correctly handle. A buffer overflow vulnerability represents an opportunity for running malicious code instructions.
- Denial of service, which prevents users from interacting with an application (e.g., by overloading a web server with “nuisance” requests)
- The interception, mimicking and/or altering and subsequent relaying of communications (e.g., credit card transactions) by a third party such that a user remains unaware of that third party’s presence (“Man in the Middle” attack)
- Breaking the encryption codes used to protect sensitive data
- Logic bombs (sometimes called Easter Eggs), which may be maliciously inserted into code and which activate only under certain conditions (e.g., on a specific date). When logic bombs activate, they may perform malicious acts such as the deletion of files or formatting of disks.

Security testing must be integrated with all other development and testing activities. This requires considering the organization’s unique needs, any existing security policies, current security testing skill sets, and any existing test strategies.

2.2 The Purpose of Security Testing

Like software testing in general, security testing cannot guarantee a system or organization will be safe from attack. However, security testing can help identify risks and evaluate the effectiveness of existing

security defenses. There are other activities to supplement security testing, such as audits and reviews of security practices.

Security testing also shows that due diligence has been performed in protecting digital assets. In the event of a security breach, legal action may result. If a company can show that it took reasonable steps to protect digital assets, such as testing for vulnerabilities, there may be a defense in a court of law. Security testing can also be an assurance to clients and customers that an organization takes adequate steps to protect sensitive information.

2.3 The Organizational Context

Security is often one type of functional testing performed along with other types of testing. With only a given amount of time available for testing, a test manager must decide how much testing can be performed, including security testing. It is not uncommon for security testing to be considered a specialist role, and therefore outsourced to an organization that specializes in security testing. The extent of security testing is ultimately driven by business or organizational risks that are security-based. When security risks are numerous in an organization, more extensive security testing is needed.

Like software testing, information security is a lifecycle activity. Security needs must be defined in the requirements, expressed in design and implemented in code. Then, security testing can verify and validate the correctness and effectiveness of security implementation. Security can't be effectively patched into code or tested into code. Only when security is built into software using secure coding and design techniques can software be secure.

The realities of limited time, resources and scope, along with risk levels, security testing skill sets and lifecycle approaches greatly impact the success of a security testing team in an organization.

2.4 Security Testing Objectives

2.4.1 The Alignment of Security Testing Goals

The security testing policy can be written once the organization's security policy has been approved by senior management. It is important that security testing goals and objectives, as expressed in the security testing policy, are in alignment with the overall security policy of the organization. Otherwise, either unauthorized security tests will be performed, or security tests may not achieve the desired purposes.

2.4.2 Identification of Security Test Objectives

Security test objectives can be thought of in the same light as functional test objectives, but are focused on security goals. There should be one or more security test objectives for each security feature of the system or application.

Security test objectives must also be based on attributes of the technology (e.g., web, mobile, cloud, LAN) and known vulnerabilities, both in the application and generic vulnerabilities. For example, security test objectives might include:

- Verify that password authentication applies the correct rule for password strength
- Verify that all data entry fields are input validated to prevent SQL injection attacks
- Verify that customer data files are encrypted at the correct strength

2.4.3 The Difference Between Information Assurance and Security Testing

Information Assurance (IA) is defined as, “Measures that protect and defend information and information systems by ensuring their availability, integrity, authentication, confidentiality, and non-repudiation. These measures include providing for restoration of information systems by incorporating protection, detection, and reaction capabilities.” [NISTIR 7298]

Security testing is “A process used to determine that the security features of a system are implemented as designed and that they are adequate for a proposed application environment.” [MDA1]

When comparing the terms Information Assurance (IA) and Security Testing, IA is a wider, more encompassing term. This relationship is similar to that between Quality Assurance (QA) and Software Testing.

2.5 The Scope and Coverage of Security Testing Objectives

The greater the need for integrity of sensitive digital and physical assets, the greater the need is for coverage of the security test objectives. Security test objectives essentially describe the scope of security testing. If the scope is too small, the confidence that security is adequate will not be achieved. If the scope is too large, resources may be depleted before the test can be finished.

Security test objectives should describe what security testing is expected to achieve in regards to verifying and validating the protections in place for sensitive digital and physical assets. Security test objectives should relate directly to specific assets, protective measures, risks and the identification of security vulnerabilities.

2.6 Security Testing Approaches

The security testing strategy is defined to formalize and communicate an organization’s overall direction for security testing. Approaches are then defined that implement the security testing strategy.

2.6.1 Analysis of Security Test Approaches

Each organization has unique business and mission concerns, which in turn require unique security test strategies and approaches to identify and mitigate security risks. However, there are also some security concerns that are common to many organizations.

A security testing approach is defined at the project level and should be consistent with the organization’s test policy and strategy. A project’s security testing approach will be a unique blend of techniques, tools and skills to address the security test objectives for that project.

When analyzing a situation for the purpose of defining a security testing approach, consider the following:

- The source of the systems or applications
- Any previous security testing
- The security policy
- The security testing policy
- Any security risk assessments already performed in the organization
- The technical environment in use (e.g., software type and version, frameworks, programming languages, operating systems)
- Security testing skills in the test team
- Common security risks

- The test organization structure
- The project team structure
- The test team's experience with various security testing tools
- Constraints (e.g., limited resources, limited time, lack of access to environments)
- Assumptions (e.g., assumptions as to other prior forms of security testing performed)

Different technical environments and types of applications (e.g., client/server, web, mainframe) often require different security testing approaches and strategies. For example, software development may require code reviews to detect security vulnerabilities in the code, while software testing may require obfuscation of test data. Web-based applications have different vulnerabilities than mainframe systems and therefore require different kinds of security tests.

Some vulnerabilities are common to multiple technologies. For example, buffer overflow vulnerabilities can occur in client-server, web, and mobile applications with differences based on how memory management is handled in each technology. The result is the same in all environments, which is unpredictable software behavior that may allow an attacker to access an application and perform tasks that would normally not be allowed.

Inadequate data protection can occur in any technology or environment. However, encrypting data in the web and mobile environments is different than in the mainframe environment. The encryption algorithms may be the same (or similar), but the difference is that data must be protected in transit over the Internet in the case of web and mobile applications. In all technologies sensitive data should be stored in an encrypted format. There have been incidents where sensitive mainframe data was physically sent (using tape) to another party in an unencrypted format. "The Cattles Group, which specializes in personal loans and debt recovery, admitted losing two backup tapes containing information for about 1.4 million customers." [ComputerWeekly]

2.6.2 Analysis of Failures in Security Test Approaches

It is necessary to understand there are degrees of failure. Just because a security vulnerability is not detected and resolved, it doesn't necessarily mean the security testing approach failed. There are too many possible security vulnerabilities, with new ones being discovered daily. However, there are other cases where security test approaches have been inadequate to effectively identify security risks, which have led to sensitive data and other digital assets being compromised.

Root cause analysis can help identify why a security testing approach may have failed. Possible causes include:

- Lack of executive leadership in establishing security testing
- Lack of executive provision of resources needed to implement the security testing strategy (such as lack of funding, lack of time, lack of resources)
- Lack of effective implementation of the security testing approach (such as the lack of skills needed to perform the required tasks)
- Lack of organizational understanding and support of the security testing approach
- Lack of stakeholder understanding and support of the security testing approach
- Lack of understanding of security risks
- Lack of alignment between the testing approach and the organization's security policy
- Lack of alignment between the testing approach and the organization's security testing policy and strategy
- Lack of understanding of the purpose of the system
- Lack of technical information about the system (causing wrong assumptions)
- Lack of effective tools for security testing

- Lack of skills in security testing

2.6.3 Stakeholder Identification

In order for a security testing effort to be effective, a business case for it must be made to management. This business case must clearly define risks of security lapses and benefits of having an effective security testing approach for a particular project.

Different stakeholders will see different benefits of a security testing approach:

- Executive management will see business protection as a benefits
- Senior management may see due diligence
- Business customers may see protection from fraud
- Compliance officers (for internal corporate security policies) may see assurance that the organization is compliant in terms of legal obligations
- Regulatory officers (for external security laws) may see a benefit that security regulations are being followed
- Privacy officers may see the benefit that private data is kept secure and due diligence has been shown in the protection of digital assets

2.7 Improving the Security Testing Practices

To improve security testing practices, first an evaluation of existing practices is needed. There should be an objective way to evaluate security test practices. These are based on key metrics for security testing goals, from which it is possible to identify the degree of success for key strategy elements.

These practices must be evaluated as follows:

- From a short-term and long-term perspective
- Considering process and organization
- Considering people, tools, systems and techniques

The key metrics include, but are not limited to:

- Coverage levels of security risks by tests
- Coverage levels of security policies and practices by tests
- Coverage levels of security requirements by tests
- Effectiveness levels of past security testing efforts, based on when and where security vulnerabilities were identified. This includes both pre- and post-release security vulnerabilities.

3. Security Testing Processes - 140 mins.

Keywords
account harvesting, password cracking, social engineering, test approach, test plan, test process

Learning Objectives for Security Testing Processes

3.1 Security Test Process Definition
AS-3.1.1 (K3) For a given project, demonstrate the ability to define the elements of an effective security test process

3.2 Security Test Planning
AS-3.2.1 (K4) Analyze a given security test plan, giving feedback on strengths and weaknesses of the plan

3.3 Security Test Design
AS-3.3.1 (K3) For a given project, implement conceptual (abstract) security tests, based on a given security test approach, along with identified functional and structural security risks
AS-3.3.2 (K3) Implement test cases to validate security policies and procedures

3.4 Security Test Execution
AS-3.4.1 (K2) Understand the key elements and characteristics of an effective security test environment
AS-3.4.2 (K2) Understand the importance of planning and obtaining approvals before performing any security test

3.5 Security Test Evaluation
AS-3.5.1 (K4) Analyze security test results to determine the following:
- Nature of security vulnerability
- Extent of security vulnerability
- Potential impact of security vulnerability
- Suggested remediation
- Optimal test reporting methods

3.6 Security Test Maintenance
AS-3.6.1 (K2) Understand the importance of maintaining security testing processes given the evolving nature of technology and threats

3.1 Security Test Process Definition

Like software testing in general, security testing is also a lifecycle activity. Failure to implement and test security defenses throughout a project can lead to severe security defects which may never be fully resolved. The security test process must be aligned with the development process so that appropriate test activities are performed when needed.

Each organization's security testing risks and needs will be unique due to the nature of the organization, the technical environments, the software development process and the business risks. Therefore, the security testing process must be defined in the context of these factors.

3.1.1 ISTQB Security Testing Process

Table 3.1 shows the relationship between the general ISTQB testing process as described in the ISTQB Foundation Level and Advanced Level syllabi, and the ISTQB Security Testing Process. Example security testing tasks are shown for each step in the process.

Table 3.1 – ISTQB Security Testing Process

ISTQB Testing Process	ISTQB Security Testing Process	Example Security Testing Tasks
Test Planning and Control	Security Test Planning and Control – The goal is to define an appropriate scope of testing that corresponds to the security risks.	• Define security test objectives • Define the scope of security testing • Identify security test resources • Define security test estimates and schedules • Define security test metrics, entry and exit criteria • Monitor security test progress and results • Take actions as needed in response to information learned during other security test activities.

Test Analysis and Design	Security Test Analysis and Design - The goal is to gain understanding of specific security threats and risks based on security assessments, audits and standard sources of known vulnerabilities.	• Review items that serve as the basis of security testing, such as security risk assessments, security requirements and security policies
		• Define security test conditions based on:
		  • Test objectives
		  • Security risks
		  • Security standards and known vulnerabilities
		  • Defenses implemented to secure the system and its data

ISTQB Testing Process	ISTQB Security Testing Process	Example Security Testing Tasks
		• Scope of security testing • Applicability of security test tools
Test Implementation and Execution	Security Test Implementation and Execution – The goal is to translate conceptual tests into tests that can be executed either manually or with tools. Also, the goal is to perform these tests using a variety of security test perspectives – internal user, external user, malicious user, etc.	• Create security test cases, test scenarios, test scripts or other test specifications • Perform functional security tests based on defined security test specifications • Perform functional security testing and penetration based on tester knowledge and intuition • Perform security testing based on a model of a system • Set up or ready a testing environment to perform security testing
Evaluating Exit Criteria and Reporting	Evaluating and Reporting Results of Security Testing – This is often performed alongside test execution to evaluate individual tests and to report new threats as soon as possible.	• Determine specific security vulnerabilities based on test outcomes • Evaluate security risk levels based on the results from security tests performed • Report interim and final security test results to management and other authorized parties
Test Closure	Test Closure – The goal is to bring the security test activities to a point of closure so the tests can be maintained and performed on a regular basis to support any new security requirements and/or detect any new threats. In addition, all security testware and results are stored in a secured way, while being available for use if needed in future security tests.	• Ensure all planned security tests have been performed • Determine if security test deliverables (reports) have been delivered • Archive test results, test data, and other sensitive information in secure locations • Analyze security test results to improve system and application development in terms of security

It is important to understand that the ISTQB Security Testing Process is not necessarily sequential in nature. The security testing process should align with the organization's software lifecycle process. A

major implication of the process described in this section is that security test activities are performed alongside and during other project lifecycle activities and tests.

In addition, the security testing tasks shown in Table 3.1 are intended as examples, not prescriptive requirements for security testing tasks. The exact security testing tasks for an organization depend on the security test strategy and approach adopted by the organization as shown in Figure 3.1 below.

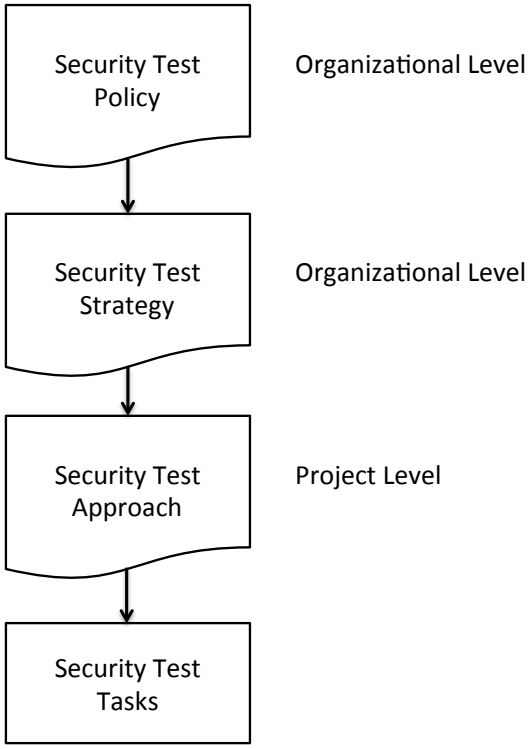

Figure 3.1 – Hierarchy of Security Test Planning

3.1.2 Aligning the Security Testing Process to a Particular Application Lifecycle Model

Each of the following types of lifecycle processes has security testing concerns. It is important to align the security testing to match the lifecycle.

Sequential Lifecycles

In these projects, the security tester should be aware of the following:

- Security needs and risks are defined early in a project and should be documented in software requirement specifications.
- Security needs may change during the project, but may not be reflected in updated software requirements. Security tests may appear to be very specific and complete, but may not be complete or current due to late-project risks.
- Security tests can be performed at any time, but it is common for these tests to be performed late in the project.

- It may be difficult to address the results of security testing at the end of a sequential lifecycle project.

Iterative/Incremental Lifecycles
Incremental projects deliver small and frequent releases of an application. Agile methods are an example of this approach. In these projects, the security tester should be aware of the following:

- Security needs and risks emerge throughout the project (normally in the context of an iteration or sprint) and can be defined in requirement specifications, user stories, models, acceptance criteria and/or prototypes.
- Security needs and risks may change during the project, and can (should) be addressed in the iteration in which they are identified.
- Security tests may be performed continuously throughout a project.
- Depending on the nature of the security risk, it may not be possible to completely mitigate and test it during one short release cycle.

Commercial Off-the-Shelf (COTS)
These projects are often black-box in nature and may or may not be customized. They often contain security vulnerabilities to the extent that frequent security updates and patches are required. There is no access to the code, so structural analysis and structural testing is not possible.

Open Source Software
This is a variant of COTS, but with one important distinction – the code is available for all the world to see. These products also have security vulnerabilities, so it is vitally important that security patches are kept up-to-date. Once a security vulnerability has been publicized, the users of that particular version of the software (and earlier) are at risk of an attack.

Example – Security Testing Process in a Sequential Lifecycle
It is important to note that security testing need not be confined to one phase or activity in a project. It is particularly important to avoid the situation where security testing (and other tests) is not performed until the acceptance phase of the project. At the end of the project, it is especially costly and risky to deal with any discovered defects. The following shows the proper security testing tasks that should be accomplished at each phase in the sequential lifecycle:

- **Requirements** – Security requirements are defined and reviewed as part of the overall requirements effort to express the needs of the organization. This is also where use cases may be written. It is at this point that a security test approach should be developed.

- **Analysis and design** – Normally someone in the Business Analyst role will examine the initial statement of requirements and refine them to fill gaps. A Systems Analyst and/or Architect will then analyze the requirements to propose the most optimal way to deliver a solution to meet the user needs. In this case, security would be one of the functional and non-functional needs, along with others such as usability and efficiency. At this point, the security test designers can get an idea of the architecture and what needs to be tested from both the structural and functional security perspectives. Major security test objectives should be defined at this time.

- **Detailed design** – At this point user interfaces and databases are designed. Functional rules are refined and security test design becomes more detailed. The first security tests may be performed based on models.

- **Coding/implementation** – This is when the design specifications are implemented as code. This is the first opportunity to test the structure of the application, including testing for security vulnerabilities such as buffer overflow defects and field-level edits that might permit SQL

injection to occur. Static analysis and code reviews are very valuable at this phase and should include examining the code from the security perspective. Component testing is also a key activity for verifying the code works as specified. Integration testing between components can also begin as components that interface with each other become available for testing in small assemblies.

- **System testing** – This is the test of systems and sub-systems. The system test includes software, hardware, data, procedures and how people interact with the system. These tests are often transactional in nature to test business processes. The basis for system testing can be requirements, design models, use cases and any other specifications that convey the system perspective. In addition, system integration testing may need to be performed to test how various (sub) systems communicate and exchange data. Security testing in this phase takes on a wider view because hardware and data exchanges are involved. Transaction security can be tested, which includes authentication, data storage, firewall implementation, as well as procedural security controls.

- **User Acceptance Testing** – This is when testing validates that a system supports real-world business processes and may span multiple systems in multiple organizations. The goal of this phase is not so much to find defects, but rather to validate that the system meets user needs in real-world conditions. This includes ensuring that security requirements have been implemented and met correctly. At this phase, security testing should have already been largely performed, but there are still opportunities to test security scenarios that occur at the business process level.

- **Deployment** – This is when the completed and tested system is deployed to users. There are many ways this can occur, such as in pilot deployments to selected groups, or a massive deployment to all users. Another approach is a parallel deployment where an old system and new system are in operation at the same time for a limited time. Much of the decision of a direct cut-over implementation depends on the risk of deploying to all users and the confidence gained during acceptance testing. Security is a concern during system deployment as all system components must be deployed in such as way that no new vulnerabilities are introduced. This can occur if security configurations are not correct in the target environment. An example of this would be if database access rights are not correct in the live environment.

- **Maintenance** – As new needs emerge or defects are discovered after deployment, maintenance is performed. Testing takes on a different dimension as the focus is on testing changes and performing regression testing. Security testing should also be performed to ensure new vulnerabilities are not introduced during changes. Part of the maintenance process is to keep firewalls and other security technology current. Continuous system monitoring can detect suspicious activity that may need to be addressed immediately.

Example – Security Testing Process in an Iterative/Incremental Lifecycle
There are a variety of methodologies that have been introduced over the last 20 years to define building software in smaller increments or iterations. In this example, releases of the software are delivered every four weeks. The basis of work (and of testing) is user stories, each with defined acceptance criteria.

The selection of which features to build and deliver is based on a prioritized backlog. The selected features should reflect the items that deliver the greatest value and are achievable in the sprint timeframe. The security tester works with business and/or product owner to have the proper and correct security requirements.

In this example, four major security functions are selected for the first iteration because they will be needed to develop many of the other features. The features are:

- User login
- SSL (Secure Socket Layer) enablement
- Reset lost password
- Lockout of account after three unsuccessful tries

Each of these features are written as user stories and refined into more detailed requirements, each with acceptance criteria.

From a security test perspective, the security tester works with the developer to ensure the correct policies and protocols are reflected in the code. The security tester will also work next to the developer to test the functions as they are developed.

In this example, the first release may just be the login page and associated functions for the login, such as resetting a lost password and the lockout control. In the next iteration, other functions will be developed, based on priority to the stakeholders. In each iteration, the security tester will test to make sure the security controls are working correctly and no new security vulnerabilities have been introduced. Iterations will continue until the backlog tasks have been completed.

In both examples (iterative/incremental and sequential), the security testing process steps can be seen as integral tasks to ensure a secure application.

3.2 Security Test Planning

3.2.1 Security Test Planning Objectives

Security testing in general should focus on two aspects:

- Verifying the designed security defenses are implemented and function as designed
- Verifying that no vulnerabilities are introduced during the development of the application

As mentioned earlier in this syllabus all security defenses to be implemented should be based on a risk analysis. This provides a starting point when planning security testing for a project.

Many of unintended developed vulnerabilities can be avoided using quality assurance activities and best practices during architecture, design and coding activities. Testing if vulnerabilities are introduced starts with an assessment of the practices used by the development team. Based on the outcome, it may be necessary to select and introduce additional security tests.

3.2.2 Key Security Test Plan Elements

Key elements of a security test plan are listed below. Each of these can be determined by asking the specified questions for a given project.

- Identifying the scope of security testing
  - What is in scope and out of scope?
  - What is achievable given project resources, security risks and time constraints?

- Identifying who should perform security testing
  - Does the organization have people with appropriate security testing skills?
  - Does the organization feel comfortable with outsourcing security testing?
  - In the case of commercial software and vendor-developed software, which security tests are the responsibility of the vendor and which are the responsibility of the customer?

    - Do security testers need training in the use of specific security test tools?

- Allocating the appropriate schedule for security testing given other project test scheduling requirements
    - Which security-related items need to be implemented and tested before other testing occurs? (e.g., access rights and logins)
    - When will security features be available for testing?
    - How long will it take to perform security testing given planned resources and scope?

- Defining the tasks to be performed and the time required for each
    - How much time is needed to design the appropriate security tests based on planned resources and scope?
    - How much time is needed to evaluate and report the results of security tests?
    - How much time is needed to perform regression tests as they relate to security?
    - How much time is needed to establish the security test environment?

- Defining the security test environment(s)
    - What is the extent of the environment? (platform, technology, size, location)
    - Is this a new environment?
    - Which security test tools and other test tools need to be installed in the environment?

- Procuring authorizations and approvals for security test activities
    - Who needs to authorize and approve security tests?
    - When is that authorization needed?
    - Are budget and funding sufficient?

Like any project deliverable, the security test plan should be reviewed to assess completeness and correctness. Since security tests are often technical in nature, a technical review session may be the most appropriate method. However, walkthroughs and inspections may also be suitable.

A standard checklist can help form the basis of what is covered in a review session. Like any other review, the feedback should be constructive and not aimed at the producer of the security test plan. The review team should include knowledgeable people from all areas impacted by the security aspects discussed in the security test plan. Review team members may not necessarily be security testers or possess security expertise. For example, the manager of a business unit may have information about security risks that should be recorded in the security test plan. IT auditors and security administrators are especially helpful in security test plan reviews because of their knowledge of security policies and procedures.

3.3 Security Test Design

There are several ways to start security test design. For instance, it might be started:

- Based on a performed risk analysis
- Based on an available threat model
- Based on an ad hoc origin categorization of security risks (see [ISTQB_ATTA_SYL])

Any of these can form a workable basis.

Depending on the type of project, it is important to ensure there are security tests in every applicable development phase.

3.3.1 Security Test Design

Detailed security tests are based on the security risks, a security test strategy and other sources such as threat models. Security tests can also be seen as functional and structural in nature. For example, in the case of the security testing of an e-commerce web site, the functional security risks can be SQL injection, account harvesting and password cracking. An example of a structural security risk would be a buffer overflow condition that would allow an attacker to gain access through a memory failure.

The following are essential attributes of detailed security tests:

- Prioritized by identified security risks and threat models
- Traced to defined security requirements
- Defined based on the intended audience (developers, functional testers, security testers)
- Defined based on security defect profiles
- Designed to be automated, if applicable

The basic flow of security test design can be seen as:

1. The security test approach (project level)
2. Security test risks, threat models and requirements (project level)
3. Security test design techniques (based on risks, requirements and application)
4. Security test cases and scenarios

In the rest of this chapter, common security risks and vulnerabilities are presented along with the associated security test design technique. New security risks and vulnerabilities emerge quickly, so it is advised that security test planners stay current with security standards and threat lists, as referenced in Chapter 9.

A key principle is that a security test design process should be able to create and implement tests based on any identified security risk, requirement, or threat.

Functional Security Controls (e.g., transaction controls)
These tests are designed to verify and validate that controls are in place, working correctly and are effective in detecting and preventing unauthorized actions.

Example: A bank teller may not authorize a cash withdrawal over a certain amount of money without the head teller's approval entered into the system.

Functional Access Controls (e.g., logins, passwords, tokens)
These tests are perhaps what most people immediately think of in terms of security testing. Tests include:

- Username and password policies are applied correctly
- The level of access control is appropriate for the risk
- The access controls are resistant to password cracking software

Example: Account harvesting is the practice of identifying a user name. Once the user name is guessed or identified, the password is the remaining piece needed to gain system access. A common test is to verify that when a correct user name is entered with an incorrect password, the error message does not indicate which of the items is incorrect.

Structural Access Controls (e.g., user access rights, encryption levels, authentication)
Tests for these controls are based on how user rights have been established for data access, functional access and privacy levels. Structural access controls are typically applied by a systems administrator,

security administrator or database administrator. In some cases, access rights are a configuration option in an application. In other cases, access rights are applied at a system infrastructure level.

Tests of structural access controls include creation of test user accounts for each level of security access and verifying that each level of access does not have access rights that are restricted for that level. For example, user accounts would be created for a minimal level of access, manager level access and administrator access. Testing should be conducted to ensure that a user with minimal access cannot perform manager level access activities.

Secure Coding Practices
This is primarily a static testing method to determine if software and system developers are following established security methods as they create applications.

A key principle is that many security attacks are accomplished through exploiting software defects to cause a system to behave in an unexpected way.

A very short list of secure coding practices includes:

- Proven session management algorithms and controls are used to create random session identifiers.
- Authorization decisions are made only by trusted system objects under control of the organization that provides authorization (for example, authorization should occur on the server side).
- Secure information should not appear in error messages. This information can include system details, session identifiers and account information.
- Application errors should be handled within the application as opposed to relying on the server configuration.
- HTTP GET requests should not include sensitive information.
- Error handlers should not display stack trace or other debugging information.
- All data input validation failures should be logged.
- Any sensitive information that may be stored temporarily on the server should be protected (for example, encryption should be used). This temporary secure information should be cleared when no longer needed.
- An application should not be able to issue commands directly to the operating system. Instead, built-in APIs should be used to conduct operating system tasks.
- Passwords, connection strings or other sensitive information should not be stored in clear text on client machines (for example, in cookies). Embedding such information in non-secure formats such as Adobe flash, compiled code and MS viewstate should be prohibited.
- Encryption should be used for the transmission of all sensitive information. Transport Layer Security (TLS) is a way to protect data in transit when using HTTP connections. For non-HTTP connections, encryption should be used for transmitting sensitive information.
- User supplied data should not be passed directly to any dynamic “include” function
- All user-provided data should be properly sanitized and validated before being used by the application.
- Variables should be strongly typed in languages that support type checking. That is, the variables should have a type of input defined. For example, a numeric field should not accept alpha characters. This restriction would be defined in the type definition of the variable, as well as in the database. It is possible to write secure code in JavaScript (Node JS) or other languages that do not support complier-enforced type checking.
- Instead of using new unmanaged code for common tasks, use tested, trusted and approved code that is under configuration management.
- Run services with the least possible privileges (never under root) and each service should have its own user account on the operating system.

A list of secure coding practices can be found in the OWASP Secure Coding Practices Quick Reference Guide [OWASP1] and Top 10 Secure Coding Practices [CERT1]. In addition, SANS compiles a list of the top 25 Most Dangerous Software Errors at [SANS1].

Dynamic tests can be performed to determine if practices such as data validation and error messaging have been followed by developers. In addition, one of the most common security vulnerabilities, the memory buffer overflow, can be identified with dynamic memory testing tools.

Operating System Access
Once access is gained to the operating system, an attacker can control data, network access and plant malware. Tests for this can include testing for the ability to plant rootkits and other malicious code into a system.

Language Vulnerabilities (e.g., Java)
According to security researchers at WhiteHat Security, an application security vendor, across the board there were no significant differences between languages when it comes to security vulnerabilities. [WhiteHat Security, 2014] In April of 2014, WhiteHat security issued a Website Security Statistics Report based on vulnerability assessments conducted against 30,000 customer websites using a proprietary scanner and the results indicated negligible differences in the relative security of languages such as .NET, Java, PHP, ASP, ColdFusion and Perl. Those six languages shared relatively similar mean numbers of vulnerabilities, and problems such as SQL injection and cross-site scripting vulnerabilities remained pervasive. [WhiteHat Security, 2014] It is important to recognize that secure code can be achieved with many languages, just as can non-secure code. The key factor is how an application is coded (implemented) in whichever language is used.

The CERT Division of the Software Engineering Institute provides publications [CERT2] and tools [CERT3] that address language-specific security issues. In addition, the Vulnerability Notes Database [CERT4] provides timely information about software vulnerabilities. Vulnerability notes include summaries, technical details, remediation information, and lists of affected vendors.

Platform Vulnerabilities (e.g., Windows, Linux, Mac OS, iOS, Android)
Each computing platform has its own set of security vulnerabilities. The concern for the security tester is to ensure that platform security updates are applied promptly and across all devices that run the affected platform.

External Threats
External security threats are the ones most people think of when they think of cyber attacks. Some external threats, such as exploiting application or language vulnerabilities can be detected, tested and prevented.

Denial of Service (DoS) is another type of external threat. In general, these attacks are based on overloading the system or application resources such that the system or application becomes inaccessible for legitimate users. DoS attacks can be targeted on the network's bandwidth, system or application connectivity or specific services or functions.

A Distributed Denial of Service (DDoS) attack is a type DoS were the attack is launched indirectly using other computer resources. Possible techniques are amplification or the use of botnets, which is a large number of earlier compromised computers under control or command of an attacker. An attacker may gain control by simply originating virus infections or the use of Trojans. The infected computers may be used as agents, each sending traffic to a specific victim (network) as targeted by the attacker.

When using amplification or reflection attacks, the attacker is using a vulnerability (or even desired functionality) in specific protocols (e.g., DNS or NTP). The attacker sends a large amount of traffic to IP broadcast addresses (multiple hosts) containing the spoofed source address of the victim. This results in the broadcast service echoing this traffic towards the victim's address and multiplying the original amount of traffic with the number of hosts. When an attacker sends these kinds of requests several times per second, the victim is suddenly confronted with the high number of responses it has to send.

Example: Attacker A sends a request to system B for a complete list of all known DNS records while impersonating Victim C, often with a spoofed IP address. System B will then send the complete list to Victim C that floods the Victim C server with an amplified amount of data.

Another form of DoS attacks are resource exhaustion attacks. These kinds of attacks abuse a desired functionality by consuming the computing resources (CPU, memory, disk storage, etc.) needed to provide the functionality.

Example: One of the functionalities in the SSL protocol is the option to generate new keys in an existing session if the client or server suspects a compromised session. Generating keys is a resource expensive process. When an attacker sends a request to generate new keys several times per second, a badly configured or unprotected system can end up in a situation where it is only generating new keys and does not have any resources left to do other things.

Lastly there are the so-called logical DoS attacks, where an attacker can abuse intended functionality to prevent other users from accessing the system.

Example: An application uses predictable user names and locks a user permanently after three failed login attempts. An attacker can guess the user names and lock many accounts in the system causing many users to be unable to access it (and indirectly DoS-ing the helpdesk).

There are four levels of testing for DDoS.

1. Test to ensure computers are not infected with known malware
2. Test the ability of the intrusion detection systems to quickly identify multiple requests from a single computer in a short period of time
3. Identify configurations that allow for functionalities that can be abused by an attacker (e.g., SSL, webserver, DNS)
4. Identify logic defects that can allow a DoS

Intrusions are another form of external attack. There are many ways to accomplish external intrusion into a system. These attacks are based on someone "breaking into" a system to get information. Some of the methods are described in the list below:

- Social engineering
- Injection attacks (SQL, malicious code)
- Account compromise (harvesting, password reset)
- Exploiting known vulnerabilities (firewall, OS, framework, application)
- Malware attacks
- Insecure configuration attacks
- Authorization defects
- Application logic attacks (taking advantage of application defects, especially in web-based applications, to misuse the function – for example, performing steps out of order in an e-commerce shopping application to achieve a discount or credit)

Intercepting a network transmission sent from inside an organization to someone in another organization is not considered an intrusion attack, but rather an internal breach.

Internal Threats
The greatest threats may be internal. Consider the following sources of internal attack:

- Corporate espionage where a trusted employee may sell corporate information, including customer account information, trade secrets, employee access information, etc.
- Information obtained by outsourced developers, testers, and other personnel (such as customer service representatives). Sometimes people leave the employment of an outsourcing company and take the information with them in their head.
- The theft of hard drives and other physical storage devices
- Disgruntled employees who seek to damage the company by leaking confidential information, or by committing acts of theft by paying themselves money under the guise of legitimate (but fabricated) invoices

Security Test Format and Structure
Each organization that performs security testing will have its own way of formatting detailed tests. It is often possible to use the same format for security test design as other types of testing, with the only difference being the target of the test and the test environment.

Even if an organization follows standards such as IEEE 829-2008 and ISO 29119 [ISO/IEC/IEEE 29119-3], the usage of that standard should be tailored to fit an organization's needs. These standards, however, form a standard understanding of what should be contained in various test planning documents. In many cases, test cases and test procedures (scripts) may be defined and implemented in a test management tool which often provides formatting structure.

Test cases are the most standalone form of test description. They do not require sequential execution. If sequential execution is needed to achieve a particular test objective, test cases are combined in a sequence expressed in a test procedure or script. Test cases are typically used for testing single conditions. For example, in security testing, testing the login function might consist of test cases designed to validate that the password formatting requirements are enforced correctly.

During test implementation the test cases are developed, prioritized and organized in the test procedure specification. The test procedure specifies the sequence of test case execution. If tests are run using a test execution tool, the sequence of actions is specified in a test script (which is an automated test procedure). Test procedures are used when sequence is important. For example, a test procedure would be helpful in testing the "recover lost password" process.

When experience-based tests, such as exploratory tests are needed, the test conditions and expected results are not defined in advance of the test, but the conditions tested and actual results should be recorded by the security tester for reporting.

3.3.2 Security Test Design Based on Policies and Procedures

When designing tests to validate security policies and procedures, these items become the basis of testing. From this perspective, security tests are almost a means of security auditing.

Security policies and procedures should not be the only basis of testing because other perspectives of security testing are needed. The goals of designing tests to validate security policies and procedures are to:

- Understand the purpose and scope of the policy or procedure
- Assess the testability of the policy/procedure
- Create tests related directly to the policy/procedure

For example, there might be a procedure that states: "*All XYZ's IT systems limit the number of unsuccessful login attempts to three. A specified lock-out period will occur after three unsuccessful login attempts. Individuals who do not have the appropriate local user account information will not be able to access our IT system and must contact IT support services to verify identity and obtain a temporary password.*"

This is a very testable procedure, which would require the following steps:

1. Attempt logging into an application three times unsuccessfully. A lock-out message should appear on the third unsuccessful attempt. Any further attempts to login to the account receive the lock-out message.
2. Contact IT support services and verify identity. A temporary password will be issued to a known email address.
3. Login with the temporary password. Access should be granted.
4. Create a new password that conforms to the password policy. The new password should be accepted.
5. Logout.
6. Login with the newly created password. Access should be granted.

Note that step 4 provides an opportunity to test the password policy as well.

Not all security policies are this testable. For example, "*The contents of XYZ, Inc. audit records contain all audited events with date/time stamp and are traceable to specific individuals. Manufacturer-specific logs that provide sufficient information to accomplish these requirements shall be considered adequate for auditing purposes.*"

While not impossible to test, a test would need to be defined and performed to cover all audited events. Actions would need to be performed to trigger a sample set of events to be logged to the audit records, and the accuracy of the information logged would need to be verified as correct, such as the user ID and the date and time stamp.

3.4 Security Test Execution

3.4.1 Key Elements and Characteristics of an Effective Security Test Environment

While many forms of testing can utilize a test environment located on the same server and network with other systems, security testing has unique risks that require a segregated approach to building the test environment. This is especially true when testing untrusted applications (such as from a third-party or open source provider).

Some security tests, such as testing functional controls and session management can be performed in a typical integrated test environment without high risk. However, when testing unknown and untrusted code, the possibility of malware corrupting a server and/or network makes it advisable to test in an isolated or virtual test environment.

The main attributes for a security test environment are as follows:

1. Isolated – from other systems (depending on the level of malware risk)
2. Complete – the total environment will need to reflect the target (production) environment in terms of:
   - Systems and applications under test
   - Operating systems (exact version and configuration)
   - Networking
   - Middleware

- Desktops (hardware brand, processor, memory)
- Mobile devices (manufacturer, processor, memory, power management)
- Databases
- Access rights
- Browsers and plug-ins
- Co-existing applications
- Data (engineered test data or production data that has been obfuscated)

3. Restorable – to repeat tests as needed and to recover from corruption should it occur

3.4.2 The Importance Of Planning and Approvals in Security Testing

There are several reasons why a security tester should have approval before executing security tests:

- In almost all countries, it is against the law to (try to) get access to data systems and their information. In some countries it is even against the law to have access to security testing tools. This means that in most security test activities you will be breaking one or more laws in doing such. The only possible way to perform the testing is to get a waiver from the system or data owner and approval from your management.
- Security tests may trigger intrusion detection alerts and the tester may appear to be a malicious insider. Penetration testing is a specific case when such authorization is especially needed.
- Security tests may lead to major system failures and black outs. The risk should be known and possible precautions should be in place.

Without prior and specific authorization for security testing, a tester may be violating security policies and procedures. This may make the tester subject to termination or prosecution.

An authorization form for security testing should contain the following information:

- Name of authorizing entity
- Names of testing personnel and/or entity
- Statement of work
- Dates of authorization (to/from)
- Other relevant details such as originating IP addresses, user accounts, and so forth
- Attestations:
  - Customer owns the system to be tested
  - Customer has the authority to authorize security testing
  - Customer has performed a backup of all systems and data
  - Customer has tested that the system can be restored from backups if needed
  - Customer understands the risks associated with security testing
- A “hold harmless” clause for the testing entity
- Signatures of customer representative authorized to enter into such agreements

A sample form can be found at [OWASP3].

3.5 Security Test Evaluation

Like much of testing, security test evaluation is performed during test execution as individual tests are performed. Security test evaluation is the evaluation of the outcome of a security test. When security defects (vulnerabilities) are identified, an incident report should be filed which states the following at a minimum:

- Name of tester observing the vulnerability
- Test environment where the vulnerability was observed
- Test steps performed (to facilitate re-creation of the test results)

- Nature of security vulnerability
- Extent of security vulnerability
- Potential impact of security vulnerability
- Suggested remediation course of action

Security test incident reports can be filed using the same incident management system as other forms of testing. Security test reports should be assigned a special category and may need to be secured to prohibit viewing by unauthorized personnel. Such situations might be when:

- Security testing is being performed by an independent organization and incidents are reported in a tool that has few restrictions on viewing incident reports
- Security vulnerabilities may be identified, but not immediately resolved
- Internal personnel may be considered a potential threat to take advantage of security vulnerabilities

The IT auditor should be able to make the determination whether or not to restrict access to security test results.

At the conclusion of a major security testing effort, such as at the conclusion of system testing, a final security test report may be issued. This report may also need to be considered confidential, depending on the status of vulnerability resolution.

3.6 Security Test Maintenance

In many cases, modifying the security testing process may only consist of adding new types of tests in response to new types of threats. One thing is certain, however. The security testing targets and threats change daily, so the security testing process needs to be designed to change easily.

New tools also appear on the market to help perform security tests. Security testers should keep up with these advances and evaluate which tools might add power and flexibility to security testing.

4. Security Testing Throughout the Software Lifecycle - 225 mins.

Keywords
abuse case, fuzz testing

Learning Objectives for Security Testing Throughout the Software Lifecycle

4.1 Role of Security Testing in a Software Lifecycle

AS-4.1.1 (K2) Explain why security is best achieved within a lifecycle process
AS-4.1.2 (K3) Implement the appropriate security-related activities for a given software lifecycle (e.g., iterative, sequential)

4.2 The Role of Security Testing in Requirements

AS-4.2.1 (K4) Analyze a given set of requirements from the security perspective to identify deficiencies

4.3 The Role of Security Testing in Design

AS-4.3.1 (K4) Analyze a given design document from the security perspective to identify deficiencies

4.4 The Role of Security Testing in Implementation Activities

AS-4.4.1 (K2) Understand the role of security testing during component testing
AS-4.4.2 (K3) Implement component level security tests (abstract) given a defined coding specification
AS-4.4.3 (K4) Analyze the results from a given component level test to determine the adequacy of code from the security perspective
AS-4.4.4 (K2) Understand the role of security testing during component integration testing
AS-4.4.5 (K3) Implement component integration security tests (abstract) given a defined system specification

4.5 The Role of Security Testing in System and Acceptance Test Activities

AS-4.5.1 (K3) Implement an end-to-end test scenario for security testing which verifies one or more given security requirements and tests a described functional process
AS-4.5.2 (K3) Demonstrate the ability to define a set of acceptance criteria for the security aspects of a given acceptance test

4.6 The Role of Security Testing in Maintenance

AS-4.6.1 (K3) Implement an end-to-end security retest/regression test approach based on a given scenario

4.1 The Role of Security Testing in a Software Lifecycle

Security is not tested or patched into an already-built application. Rather, it is achieved through security-oriented design and verification throughout the process of construction. Like software testing in general, security testing is also a process that must occur within the development lifecycle.

4.1.1 The Lifecycle View of Security Testing

A software lifecycle process provides a framework to perform certain activities at times that align with other activities. For example, the user needs should be obtained before application design occurs. The selection of the software lifecycle depends on the nature of the organization, the project and similar factors [IEEE 12207]. For the purpose of this syllabus and security testing, the concepts and techniques can be applied to any lifecycle process – sequential or iterative.

In Chapter 3 of this syllabus, a security test process was described which aligns with a sample generic software lifecycle. The reasons for integrating security testing into the software lifecycle are discussed in the following sections.

To provide a prescribed time in the lifecycle when security-related activities should occur
For example, when capturing and defining user needs, the business or systems analyst should be asking questions such as:

- Which levels of security access are needed?
- Are there any digital or physical assets that require special security defenses?
- How “open” is the application intended to be?
- What are the security risks?

Another example would be during coding. At this time, the developer has the best opportunity to apply secure coding practices to avoid attacks such as SQL injection and memory buffer overflow attacks. To find these kinds of vulnerabilities during later stages of the project is difficult and costly because many other software components might also need to be similarly addressed and fixed.

To provide checkpoints for review
For example, security requirements or user stories should be reviewed to ensure the security-related aspects of the user’s needs have been adequately investigated and documented. Code changes should also be reviewed to detect the presence of malicious coding by internal employees or contractors.

To provide checkpoints for testing
For example, in development, component tests should be documented and performed to verify that secure coding practices have been followed and successfully implemented.

To provide entry and exit criteria throughout the project
An example of this practice would be that no component may be accepted into an integrated test environment until it can be shown that all security-related activities (both development and testing) have been successfully completed. This is especially important in later stages of the project, where a security vulnerability might cause a security risk that impacts the entire system or application.

4.1.2 Security-Related Activities in the Software Lifecycle

The following security-related activities are performed alongside other project activities, as opposed to being performed in their own separate lifecycle.

Requirements: Requirements are gathered and defined in a variety of ways depending on the software lifecycle in use. It should be recognized that requirements might extend beyond user and stakeholder needs. For example, there may be regulatory requirements, technical requirements and business requirements, among others.

The requirements objectives include:

- Understanding and identifying security needs from all perspectives within the organization and outside the organization. For example, the customer of a business is not in the organization, but they have the need for their private information to remain secure.
- Documenting security needs in a way that is detailed and unambiguous. This allows for implementation and testing that is traceable to the requirements, allowing the requirements to be verified and validated.

The requirements activities include:

- Defining all impacted and knowledgeable people who might have input to the requirements.
- Using a variety of methods – interviews, workshops, etc., gathering the security needs expressed by each group. This can also be performed during the elicitation of other requirements.
- Documenting the requirements in a way that can be reviewed and traced.
- Reviewing the requirements for correctness, completeness, understandability and non-ambiguity.

Design: The system or application is designed based on the needs stated in the requirements. The requirements express the security needs while the design translates the needs into a workable solution approach.

The design objectives include:

- Creating a system or application design that meets the stated security requirements

The design activities include:

- Analyzing the documented requirements
- Arriving at the most workable approach to develop the application in a secure way
- Documenting the design using the appropriate techniques in line with the software lifecycle. For example, in an iterative approach, the design sessions may be conducted on a white board, while in other processes the design might be expressed in models.

Implementation: This is commonly known as the coding activity.

The implementation objectives include:

- Translating the requirements and design into secure code that meets the functional needs as stated in requirements
- Implementing any other needed procedures or technology (firewalls, tokens, etc.) to meet security needs

The implementation activities include:

- Creating code that meets security requirements
- Performing component testing to verify the correctness, efficiency and security of the implementation
- Performing component reviews to visually inspect the correctness, efficiency and security of the implementation

System Testing:
Note that some software lifecycle models, such as iterative delivery approaches, add new components or refine existing components over a shorter time span and are able to have system testing much more frequently than other, more sequential approaches.

The system testing objectives include:
- Performing an end-to-end test to observe the overall functioning and performance of the full system (hardware, software, data, people and procedures) after the various system components have been implemented and integrated into a complete system
- Testing that security requirements have been implemented correctly from a system perspective

The system testing activities include:
- Performing security tests in some approximation of the final target environment, necessitating a transition from the development environment in which previous implementation and integration activities have occurred

Acceptance Testing:
This is the final level of testing during which users or representatives of the users of the system build confidence that the system will deliver the necessary capabilities in the target environment.

The acceptance testing objectives include:
- Having the users, or agents acting on behalf of the users, perform security testing against the security-related acceptance criteria established for the system. Many times, the security-related acceptance criteria focus on functional security controls and processes.

The acceptance testing activities include:
- Installing the system into its operational environment
- Performing security tests based on acceptance criteria
- Determining acceptance based on test results

It should be noted that both system and acceptance testing are essentially “black-box” or stimulus-response tests without regard for the internal structure or behavior of components within the overall system. Previous component and integration testing provide complementary assessments by considering and exploiting the internal architecture of components and their interactions within the system.

Maintenance:
After a system has been placed into service, additional development effort may be required to correct defects in the released version (corrective maintenance), to adjust to other changes in the operating environment (adaptive maintenance), or to extend or enhance features (perfective maintenance).

The security testing perspective for system maintenance focuses on testing changes made to correct defects (confirmation testing) and core functionality (regression testing) to:
- Ensure no new vulnerabilities have been introduced to the system by the maintenance activities
- Verify that existing security defenses are still effective following a change

In this context, maintenance can include upgrades (e.g., operating system, databases), coding changes, data conversions and platform migrations.

In essence, any maintenance activity should be treated with the same care and attention as the original development. Otherwise, the risk of introducing new vulnerabilities can seriously jeopardize the security of the operational system.

4.2 The Role of Security Testing in Requirements

The following considerations need to be understood about requirements in general:

- Many organizations are challenged just to write basic user requirements that are clear, unambiguous, complete, correct and testable.
- Requirements are highly subject to change throughout a project, and therefore the maintenance of requirements can be a challenge.
- Special skills are needed to understand the user's needs and other needs, such as compliance and technical needs, before being able to draft them into documents or input them into requirements management tools.
- Requirements can contain gaps and errors. Therefore, both verification and validation are needed.
- Requirements *should* contain needs for quality characteristics such as security, performance, usability, and so forth. However, these attributes are often overlooked in favor of functionality only.

The challenge is to get the security perspective understood and expressed in the full set of requirements for a project. In evaluating requirements, an effective technique is to use a checklist as a guide. There can be many items contained in the checklist to cover a variety of topics. For the security-related attributes, the following is a good starting point for evaluation:

Privacy Needs

- Have all user groups and their related data privacy needs been identified and documented?
- Have all data types impacted by this requirement been identified and related privacy needs defined?
- Have user access rights been identified and defined?

Compliance Needs (to security policies)

- Have all relevant security policies been identified and documented?
- Have any exceptions to security policies been identified and documented?

Common Vulnerabilities – These will change over time as security attacks change, but should be defined as risks at the point of defining requirements. These also become the basis for security tests.

- Have all common and known security vulnerabilities for the feature being documented been identified as known risks?

Testability

- Is the requirement written in a way that security tests and other tests can be written based on the document?
- Are any ambiguous terms such as "processing must be secure" and "access is only granted to authorized personnel" identified and clarified to be specific and testable?

Usability – There are trade-offs between security and usability. For example, a user login on a web site might be so confusing and difficult that customers give up and go elsewhere.

- Do the requirements reflect an appropriate level of security process in relationship to the function being specified?
- Are the security procedures clear and understandable?
- Are remedies specified for legitimate users who may experience problems in accessing information?

Performance – There is a trade-off between security and performance. For example, it is possible for high levels of encryption to slow performance.

- Do the requirements reflect an appropriate level of security efficiency in relationship to the function being specified?

4.3 The Role of Security Testing in Design

Security-degrading design practices should be identified and avoided. Test-related activities contribute by recognizing software system designs that are likely to be vulnerable to compromise and directing design of software systems with strong, identifiable security properties.

The IEEE Computer Society Center for Secure Design [IEEE1] recommends these key design approaches:

- Earn or give, but never assume, trust
- Use an authentication mechanism that cannot be bypassed or tampered with
- Authorize after you authenticate
- Strictly separate data and control instructions, and never process control instructions received from untrusted sources
- Define an approach that ensures all data are explicitly validated
- Use cryptography correctly
- Identify sensitive data and how it should be handled
- Always consider the users
- Understand how integrating external components change your attack surface
- Be flexible when considering future changes to objects and actors

4.4 The Role of Security Testing in Implementation Activities

Security testing, as in other types of testing, begins at the lowest level of implementation, exercising separate software components that will be assembled into the overall system. After the static evaluation of these components, testing provides an additional level of appraisal examining dynamic behavior in response to both valid and invalid inputs.

4.4.1 Security Testing During Component Testing

4.4.1.1 White-box/Glass-box Testing Considerations

Static testing, involving the whole range of inspection, walkthrough, audit, and technical review activities, has already been noted.

So-called white-box and/or glass-box (structural) testing refers to tests designed on the basis of visibility into software design or implementation. In contrast, black-box (functional and non-functional) testing is not based on access to any such structural information and is simply stimulus-response testing.

White-box testing can target specific controls implemented within the module and determine their effectiveness. Visibility into the component structure also allows test coverage to be measured, as in terms of percentage of executable statements exercised, percentage of decision outcomes exercised, or percentage of logic paths traversed.

Structural security testing may be performed by automated static analysis tools and security scanning tools. Fuzz testing is a security testing technique used to discover security vulnerabilities by inputting massive amounts of random data, called fuzz, to the component or system under test. White-box fuzz testing (on small software blocks, functions, classes) might get usable results in much less time than a black-box fuzzing tool would do.

White-box fuzz testing tools are able to detect memory corruption, buffer overflow, etc., by instrumenting the code being tested.

The following security vulnerabilities can be identified and fixed during structural testing:

- Memory buffer overflows
- Malicious code inserted by an internal employee or contractor
- “Backdoor” access (access via an undocumented interface only known to the developer, that was intentionally implemented to bypass normal security controls)

4.4.1.2 Functional Security Testing Considerations

The adequacy of security testing at any level should be determined by confirming satisfaction of specified security requirements. This is in addition to observing responses to stresses not explicitly specified in security requirements, security risk assessments and similar documents. Creativity is required in testing for security weaknesses because testers are probing for what the software developers have overlooked.

4.4.2 Security Test Design at the Component Level

One example set of high-level coding best practices may be found in the article “Top 10 Secure Coding Practices” [CERT1] which states:

“Tests for any component should include assessment of possible violations of these practices:

- Validate input.
- Heed compiler warnings.
- Architect and design for security policies.
- Keep it simple.
- Default deny.
- Adhere to the principle of least privilege.
- Sanitize data sent to other systems.
- Practice defense in depth.
- Use effective quality assurance techniques.
- Adopt a secure coding standard.”

Tests performed against such best-practice checklists should include assessments of possible violations of these practices on the basis of a well-documented risk analysis incorporating realistic threat modeling. In other words, focus on the most crucial requirements in terms of the probability of attack and the consequences of compromise.

4.4.3 Analysis of Security Tests at the Component Level

One key measure of adequacy involves evaluating test coverage. Various measures of coverage derive from the nature of the testing perfomed.

Requirements-based testing exercises the system to provide assurance that it satisfies its requirements as specified. Without regard for implementation (black-box), coverage may be measured as any of the following:

- Percentage of total number of requirements tested
- Percentage of specified use/abuse cases tested
- Percentage of critical functions, scenarios, or mission threads tested

Data-driven testing exercises the system to provide assurance of its behavior across a range and combination of input data, attempting to choose as few test values as possible by dividing the data space into equivalence classes and selecting one representative from each class (with the expectation that the elements of this class are equivalent in terms of their ability to detect failures). Pairwise and N-wise coverage criteria are typical forms of data coverage criteria.

Model-based testing provides assurance of coverage in terms of a chosen modeling notation. When the model uses a pre-post notation, criteria may include cause-effect coverage and coverage of all disjuncts in the postcondition. For algebraic modeling notations, coverage of the axioms is a typical coverage criterion.

For transition-based models, which use explicit graphs containing nodes and arcs, graph coverage criteria include percentage of nodes (states), percentage of transitions, percentage of transition-pairs, and percentage of cycles.

Structural testing provides assurance based on visibility into, and analysis of, the actual implementation. By simple enumeration, test coverage is commonly reported as the percentage of the packages, classes, methods, decisions or lines of executable code in the application that are executed by the tests. The latter is referred to as statement coverage.

Cyclomatic complexity is a measure of how many different independent paths exist through an element and may be visualized in terms of a control flow graph with nodes (decision points) and arcs (paths). The strongest of the control flow-based criteria is path coverage, which measures against all entry-to-exit paths in the control flow graph. Since exhaustive path testing is generally not feasible because of loops, other less stringent criteria may be expressed in terms of selected logical paths considered critical (critical path coverage) or the percentage of decision outcomes exercised (branch coverage).

4.4.4 Security Testing During Component Integration Testing

As lower-level components are integrated into subsystems and eventually the full target system, the possibilities for security breaches are not simply the summation of the vulnerabilities in each of the components considered separately. Instead new attack vectors become possible due to interactions between components and with larger system and organizational elements.

On the other hand, some interactions between components might mitigate or block possible sequences leading to security breaches. Again, security testers need to be creative in looking for what has been otherwise overlooked by developers.

Integration testing can demonstrate the complexity of a system design and the stability of its behavior. The integration test approach (for example top-down or bottom-up) can affect the timing of revealing security concerns or the need for additional security-specific tests.

4.4.5 Security Test Design at the Component Integration Level

As with component testing, integration tests should be designed on the basis of well-documented risk analysis incorporating realistic threat modeling. As separate components are integrated together note that scaffolding (in the form of stubs and drivers) may be necessary to test less-than-complete paths through a system during integration. As more implemented components are added to the system this

scaffolding is incrementally removed, allowing for fuller assessment of functionality as well as new paths to vulnerabilities that might be exploited.

4.5 The Role of Security Testing in System and Acceptance Test Activities

4.5.1 The Role of Security Testing in System Testing

System testing is the first end-to-end exercising of the fully integrated components. Although usually done in a development environment, it should reveal emergent properties of the system that would not have been observed before integration was completed. Security requirements are typically considered in conjunction with one of more functional requirements.

For instance, “In the process of doing x the system should not allow y to happen.” As functional tests are conducted the tester should be probing for ways that security constraints might be violated.

Functional requirements, including those for security, typically address the imperatives. Other specifications, such as use cases, abuse cases, process models and state transition models describe procedures which can be used to define end-to-end test scenarios for security testing.

4.5.2 The Role of Security Testing in Acceptance Testing

Acceptance testing is distinguished from system testing in that it is performed in a realistic operational environment, if not in the actual setting where the system will become operational. Such testing allows for reasonable evaluation of performance and other behaviors based on interactions through external interfaces. It also finally places the system in the setting where external threat agents would be seeking to find weaknesses on a day-to-day basis.

Acceptance testing ideally should validate that the initial project goals have been delivered. This is accomplished by designing and performing tests to validate that acceptance criteria are met. Security needs should be documented in the acceptance criteria.

The best time to define and document acceptance criteria is before system development or purchase. Therefore, an initial understanding can be made between the supplier and acquirer, even if both are in the same organization. It is also common for acceptance criteria to change or emerge during a project, so these criteria should be analyzed for their impact in security testing.

In the context of security testing, the acceptance criteria may be global in nature. For example, there could be points of acceptance criteria that specify what is acceptable in terms of overall system security. This would include criteria that are applied to all system functions, such as user authentication, user rights, encryption levels, audit trails and so forth. In other cases, specific security acceptance criteria may be needed. For example, some functions, such as issuing payments in excess of a certain amount might require two people to approve the payment.

4.6 The Role of Security Testing in Maintenance

Regression testing is intended to confirm that all previously acceptable behaviors of the system remain intact after modifications have been made. In the negative aspects of security testing, such confirmation would involve checking that the system continues to successfully resist attempts to defeat established security controls. Enhancements to usability or efficency are especially prone to sacrificing security controls.

Security regression tests should focus on confirming satisfaction of all security requirements, as well as testing for new vulnerabilities that might have been introduced during maintenance activities.

Regression testing is often applied with a collection of test cases that are based on testing individual functions. However, for security testing, this is often insufficient to detect regression defects with security impact. End-to-end regression test scenarios are more robust and provide a higher level of confidence that complete transactions can be performed in a secure way.

For this type of regression testing, a set of security test scenarios should be defined and tested each time a change is made to the system. Keep in mind that system changes may be extended to include hardware, configuration files, operating systems, DBMSs, networking and software – and any other system components. Regression defects can appear from changes to any of these. Some of the regression defects may have security impact.

Example scenarios:
Users are able to login to a web site and complete a purchase securely without compromising their personal information.

Users are only able to perform actions defined in their user rights and privileges. (A user working in the payroll department may be able to add a new employee, but not have access to their banking information.)

5. Testing Security Mechanisms - 240 mins.

Keywords

anti-malware, authentication, authorization, demilitarized zone, encryption, firewall, hashing, insider threat, intrusion detection system, malware, malware scanning, network zone, pharming, phishing, salting, system hardening, vulnerability scanner

Learning Objectives for Testing Security Mechanisms

5.1 System Hardening

AS-5.1.1 (K2) Understand the concept of system hardening and its role in enhancing security
AS-5.1.2 (K3) Demonstrate how to test the effectiveness of common system hardening mechanisms

5.2 Authentication and Authorization

AS-5.2.1 (K2) Understand the relationship between authentication and authorization and how they are applied in securing information systems
AS-5.2.2 (K3) Demonstrate how to test the effectiveness of common authentication and authorization mechanisms

5.3 Encryption

AS-5.3.1 (K2) Understand the concept of encryption and how it is applied in securing information systems
AS-5.3.2 (K3) Demonstrate how to test the effectiveness of common encryption mechanisms

5.4 Firewalls and Network Zones

AS-5.4.1 (K2) Understand the concept of firewalls and the use of network zones and how they are applied in securing information systems
AS-5.4.2 (K3) Demonstrate how to test the effectiveness of existing firewall implementations and network zones

5.5 Intrusion Detection

AS-5.5.1 (K2) Understand the concept of intrusion detection tools and how they are applied in securing information systems
AS-5.5.2 (K3) Demonstrate how to test the effectiveness of existing intrusion detection tool implementations

5.6 Malware Scanning

AS-5.6.1 (K2) Understand the concept of malware scanning tools and how they are applied in securing information systems
AS-5.6.2 (K3) Demonstrate how to test the effectiveness of existing malware scanning tool implementations

5.7 Data Obfuscation

AS-5.7.1 (K2) Understand the concept of data obfuscation tools and how they are applied in securing information systems
AS-5.7.2 (K3) Demonstrate how to test the effectiveness of data obfuscation approaches

5.8 Training

AS-5.8.1 (K2) Understand the concept of security training as a software lifecycle activity and why it is needed in securing information systems

As-5.8.2 (K3) Demonstrate how to test the effectiveness of security training

5.1 System Hardening

Over the years, a variety of security mechanisms have emerged as key practices in securing digital and physical assets. Each of these mechanisms can be applied in various ways – some through tools and infrastructure, others through manual effort. None of these mechanisms alone is sufficient in most cases to secure information. Each mechanism has its own advantages and disadvantages.

Security testers need to understand the nuances of each line of defense so that appropriate tests can be designed to verify and validate their effectiveness. Advanced level security testers need to understand the implications of each of the mechanisms described in this chapter to design a test architecture that will provide a framework for continuous security testing.

5.1.1 Understanding System Hardening

Modern systems are becoming more and more complex, thus their attack surface is continuously growing. Vulnerabilities come from design errors (by design vulnerabilities), source code defects (by construction vulnerabilities) or lack of rigor in the configuration of these systems (by configuration vulnerabilities).

System hardening is the step-by-step process of reducing the attack surface by applying a security policy and different layers of protection. The main objective is to secure the system and reduce the risks of compromising security.

Depending on the context, hardening can be applied at different levels:

- Hardening a software or hardware component
- Hardening a product/application
- Hardening a system
- Hardening a system of systems

Organizational and technical security defenses to be applied should include:

- Removing unnecessary software (can contain defects)
- Removing unnecessary libraries and developer tools (can contain defects)
- Removing unnecessary accounts/logins (attack vectors)
- Removing unnecessary applications (can contain defects) and network services (attack vectors)
- Removing unnecessary peripherals and hardware ports (e.g., USB ports, card readers)
- Promptly patching systems and installing updates (e.g., activate automatic updates)
- Updating configurations
- Following coding rules (avoid “by-construction” vulnerabilities)
- Configure remote logging server (e.g., rsyslog) so that in case of a compromise, the attacker is only able to delete the log files on the compromised machine, but not on the remote log server

The following security mechanisms should be used:

- Strong authentication and efficient management of authorization (give only rights needed to achieve actions to dedicated roles)
- Encryption (communication and local hosted storage)
- Firewalls (personal, system or web application) and defined security zones (e.g., execution in a sand box)
- Intrusion detection system
- Anti-malware/anti-spyware
- Obfuscation of data and applications (e.g., protection against reverse-engineering)

System hardening is vital to protect the sensitive assets of an organization but the security rules must be applied at the right level and balanced with the usability of the system. At the extreme end of this trade-off, protections are disabled because they block the productivity of the company.

5.1.2 Testing the Effectiveness of System Hardening Mechanisms

Testing the effectiveness of system hardening mechanisms can be accomplished in a variety of ways. Tests will depend on the nature of the system or application being hardened, the sensitivity of the assets protected, and the identified threats. System hardening restricts the access of the system to the right roles, opens only the needed services, and monitors application updates. Therefore, to test the effectiveness of system hardening, tests should be designed so that it is known if the hardening efforts are working, applied in the right places, and applied in the right ways. It is also important to test for system hardening protections that are too restrictive and might be excessive in view of the security risks.

Some system hardening tests may be review-based or audit-based, while other tests may be based on the ability of certain user groups to perform certain actions, or access certain data.

Tests could include:

- The audit of the configuration of database and application servers to verify that default passwords have been changed
- The audit of the system configuration to identify unnecessary services and network ports
- The verification of components, libraries and application versions to check that they are not obsolete and vulnerable

A vulnerability scanner can be run to ease the tasks of vulnerabilities assessment, especially if the system is complex (e.g., a multi-site environment). Static analysis tools can be used to detect violations of coding rules that might introduce construction vulnerabilities. Security-oriented analyzers can be particularly useful for detecting vulnerabilities.

5.2 Authentication and Authorization

5.2.1 The Relationship Between Authentication and Authorization

Sensitive assets of an organization (e.g., bank account numbers of a list of customers, design of a new product) need to be protected and must be accessible only by the authorized person.

Authentication is based on the verification of a user identifier and a token to answer the questions:

- Login: who is the user?
- Password: is the user really who he pretends to be?

Different implementations of authentication mechanisms might be used depending on the need to protect against attacks to hijack authentication or to steal a password. These include detecting weak passwords, employing one-time passwords (OTP), fingerprinting, software certificates, certificates in hard tokens, and similar authentication means.

Depending on the architecture of a system, the applicative context, and the needs of an organization (ease to manage login/password), authentication mechanisms might include local authentication, server authentication, network authentication, Single Sign-On (SSO), and similar means.

Authorization is used for the following purposes:

- To verify if the authenticated user has the rights to do an action (e.g., the user can log into a server but cannot modify its data, or a user is authorized to use an FTP server but only on their dedicated space)
- To determine which level of access should be allowed to the system resources

There is a strong link between authentication and authorization based on the principle that a non-authenticated user has no rights or restricted rights on the system (not authorized to manipulate sensitive data). For example, in the context of a merchant website, a non-authorized user can see the list of products but, before buying the article of choice, the user must create a user account. The authenticated user can purchase an item, but cannot perform administrative functions.

5.2.2 Testing the Effectiveness of Authentication and Authorization Mechanisms

The aim of attackers is to steal passwords or to bypass systems in order to execute unauthorized actions. Usually, they exploit different types of weaknesses: coding errors (lack of input filtering), old vulnerable version of libraries, system configuration errors (keeping default passwords, default rights), and weak passwords (e.g., the most used password is “123456”).

An organization might have a set of password rules that must be followed, but if the user is not diligent in keeping the password secure, the password rules will not make a difference. In addition, the password rules must reflect current good practice in password definition. Such practices can be found in the Password Construction Guidelines from the SANS Institute [SANS2].

Tests for the authentication and authorization mechanisms could include:

- Brute force and dictionary attacks to attempt to discover user passwords. The first steps might be to try “123456”, “111111”, date of birth, name of pet, etc.
- Exploit lack of input filtering, for example, to inject SQL requests in order to be authenticated without any known login/password.
- Input unauthorized URI (../../ in an FTP account) or URL (site address/admin) to try to gain access to sensitive data.

Another example might be to exploit a vulnerability in the targeted system (perhaps because it has not been updated) to cause unintended behavior and usually resulting in gaining control of the system and allowing privilege escalation.

5.3 Encryption

5.3.1 Understanding Encryption

To avoid divulging sensitive data, even if it can be accessed when it is stored or exchanged between client and server, an encryption mechanism might be used. Hashing and salting are methods used during encryption.

Encryption is a process of encoding data (plain text) into cypher data (cypher text), using a cryptographic algorithm and secrets, in such a way that only authorized persons have the right to access by using a decryption mechanism. The secret is shared and only known to the authorized users. The goal is to have encryption that is strong enough to prevent an attacker, who may have succeeded in stealing encrypted data, from recovering the plain text. The use of cryptographic algorithms helps to ensure confidentiality, integrity, availability of sensitive assets and repudiation of manipulating them.

Cryptographic protocols can be used to protect information:

- Stored in a system, e.g., cyphered passwords in a database, logical cyphered drive, whole cyphered hard drive

- During communication, e.g., cyphered email, cyphered communication protocol (SSL, TLS)

The primary and well-known cryptographic protocols used are:
- Symmetric encryption: use of shared secret key
- Asymmetric encryption: use of private and public key

5.3.2 Testing the Effectiveness of Common Encryption Mechanisms

Some cryptographic mechanisms are known to be weak, especially due to the short size of the secret keys, or static keys. Other mechanisms are vulnerable because either they are not implemented with best practices or they embed coding defects (such as buffer overflow).

Tests for encryption mechanisms should include:
- Tests for design vulnerabilities:
  - Assessment that right modes are used in symmetric encryption
  - Verification that the size of the cryptographic keys are not too small (e.g., as of 2015 an RSA key less than 2048 bit is considered insecure)
  - Validation of the validity of certificates and the ability to raise an alert if the certificate is self-signed (SSL-trip can be used to avoid Man In The Middle attacks)
  - Replay-attack (e.g., attack against Wired Equivalent Privacy (WEP) protocols)
  - Attacks against cryptographic protocols to verify their level of strength [Bittau]
- Tests for construction vulnerabilities:
  - Code reviews (e.g., to verify that the default function random() is not used to generate random numbers (seed) because the random algorithm is relatively easy to crack)
  - Fuzzing to exploit unexpected behavior
  - Timing attacks (analyzing the time taken to execute cryptographic algorithms)
  - Power analysis (used for encrypted hardware devices)
- Tests for configuration vulnerabilities:
  - Cryptographic protocol configuration assessment (e.g., Transport Layer Security (TLS) server-side configuration, client-side authorized protocols, based on TLS configuration guide for administrators)
  - TLS cipher order on the server side, to see if any means exists to downgrade or renegotiate the cipher that is being used
- Tests for aging to verify encryption mechanisms that may have become weak and prone to being cracked

5.4 Firewalls and Network Zones

5.4.1 Understanding Firewalls

According to [Chapman 2000], “a firewall is a component or a set of components that restricts access between a protected network and the Internet, or between other sets of networks.” A firewall implements and enforces a security policy based on the definition of authorized and forbidden communications. A firewall could be host-based (software running on a single host that monitors application inputs/outputs) or network-based (software that monitors traffic between networks).

The main task of a firewall is to control the traffic between different trusted network zones by filtering the data flowing on the network. In this way, malicious traffic coming from an untrusted zone is detected and blocked.

A network zone is an identified sub-network with a defined level of trust:
- Internet/public zone which is considered as untrusted

- Several security zones called demilitarized zones or DMZs, with different levels of trust
- One or more private/internal networks considered as the most trusted

The network zones are parts of the configuration of the firewall: they are used to define the authorized flows between the different networks. All forbidden traffic is blocked.

Usually, a firewall filters communication based on:

- Source and target addresses and protocols (Ethernet or IP addresses, TCP/UDP ports, etc.)
- Protocol options (fragmentation, TTL, etc.)
- Size of data

Web Application Firewalls (WAFs) also filter communication based on:

- Users being the connections
- Data filtering (e.g., using pattern descriptions)

5.4.2 Testing Firewall Effectiveness

Due to the number of the protocols, their different options and the complexity of networks to protect, it is difficult to configure a firewall efficiently. Tests for firewall effectiveness should include:

- Port scanning to verify if the security policy is well implemented
- Using malformed network packets and network fuzzing to exploit an unexpected behavior (e.g., a Denial Of Service)
- Fragmentation attacks to bypass filtering features with the objective to carry on the attack behind the firewall

Another example of tests, targeting the WAF, is to encode and compress data or obfuscate it to hide the malicious information that conveys the attack.

5.5 Intrusion Detection

5.5.1 Understanding Intrusion Detection Tools

Each year, the number of attacks increases. Intrusion techniques evolve quickly and no system is 100% safe.

An Intrusion Detection System (IDS) is a system (standalone appliance or application) which monitors activities at different levels (from network to application, 7 layers of the OSI model) to detect violations of the security policy. If deviations from normal behavior are detected, alerts are raised that can be analyzed for further actions (e.g., traffic blocking, virtual patching).

Regarding IDS standardization, the Internet Engineering Task Force Working Group Intrusion Detection Exchange Format describes a design model for an IDS based on two security models:

- Negative security model (signature-based detection or blacklist detection): the rule is “all that is not explicitly forbidden is allowed”. The intrusion detection is based on a list of known attacks or patterns.
- Positive security model (behavior-based detection or white list detection): the rule is “all that is not explicitly allowed is rejected”. The intrusion detection is based on the specification of the behavior of the system to protect, e.g., the characteristics of an input in a form described as a regular expression. Intrusion is detected if the behavior deviates from the normal or expected behavior of the system. Trusted traffic might be used to generate the specification.

An IDS differs from a firewall in that a firewall looks outwardly at the traffic to stop intrusions whereas the IDS analyzes suspicious intrusions and raises an alert if they are confirmed.

5.5.2 Testing the Effectiveness of Intrusion Detection Tools

Scenario-based detection is easy to bypass because only known attacks are detected. Tests could include the following evasion techniques:

- Character encoding or modification of data (e.g., adding white space, end of lines, etc.)
- IP fragmentation, TCP segmentation
- Encryption, obfuscation
- URL encoding

Behavior-based detection generates a large number of false positive results and false negative results. A false negative result is any alert that should have reported but wasn't. False negatives can occur when a new attack is developed that an IDS is not aware of, or perhaps a rule might be written in such a way as to detect some attacks but miss others. Also, the accuracy of this detection method should be considered. It is possible for an attacker to deviate the IDS behavior from its normal behavior resulting in a new specification containing intrusive behavior. Thus, this new traffic is not considered as anomalous. Complementary tests should use malicious traffic in order to add new intrusive specifications considered as authorized traffic.

Some inputs can be used to define a set of tests for IDS, such as "Profile Protection Intrusion Detection System" [PP-IDS] and "Web Application Firewall Evaluation Criteria" [WAFEC].

5.6 Malware Scanning

5.6.1 Understanding Malware Scanning Tools

Malicious code might affect servers and end users' computers providing to its inventors the expected privileges and targeted sensitive data. The malicious code is placed at the target using different means such as email with malicious attachments, fake URLs, client-side code execution, etc.

An anti-malware application is software used to analyze, detect and remove malicious code received from different sources, with different targets of detection: malware, phishing and pharming.

The main detection feature used by anti-malware is a signature-based strategy. The principle is to search in a database for known patterns of data describing a suspicious piece of code. However, new malware or malware for which the signature is not present in the database will not be detected and could infect its victim. A heuristic mechanism often is embedded in anti-malware to identify slight variations of known malicious patterns to help combat this problem.

5.6.2 Testing the Effectiveness of Malware Scanning Tools

Developers of malware and backdoors use different techniques to protect their code against reverse-engineering and detection by anti-malware software. Some of these techniques include:

- Exploiting system library functions used by malware (e.g., FindWindow which can be used to close an anti-malware application)
- String obfuscation to disable the understanding of malicious code behavior (e.g., by using encryption). An example could be to store Java script in a PDF document. Another one is to use compression such as Ultimate Packer for executables (UPX).
- Dynamic loading of functions and libraries (e.g., to limit the analysis of the malicious code)
- Automatic update of applications (e.g., Skype Trojan)

Malware can also use other hardware resources such as the Graphics Processing Unit (GPU), to decompress malicious code and store it in memory to be executed by the processor. In this case the malware cannot be analyzed before execution.

From the functional testing perspective, a tool like “Eicar” [EICAR] (anti-malware test file) could be used to test the effectiveness of anti-malware without developing real malicious pieces of code.

An important consideration when implementing a new anti-malware application, or upgrading an existing anti-malware application is to test the implementation on a representative platform before deploying it to the entire organization. There have been cases where the anti-malware software falsely identified legitimate operating system files as malware and quarantined them, thus shutting down the entire computing capability of the organization.

5.7 Data Obfuscation

5.7.1 Understanding Data Obfuscation

Obfuscation (sometimes called data masking) is a mechanism to make data and source code non-understandable for a human.

This technique is used mainly to protect sensitive data against:

- Copying, to bypass license protection mechanisms
- Reverse-engineering, to understand code in order to exploit vulnerabilities

Data obfuscation also might be used to allow the employees of a company (support personnel, functional testers, etc.) to work with non-sensitive data, while hiding sensitive data from plain view. Some may refer to data obfuscation as “data anonymization”, in that it keeps an individual’s personal data anonymous.

Obfuscation also can be used to protect source code against simple copy-paste (e.g. to protect a new innovative algorithm) and future reuse after it has been reverse-engineered to understand it.

Sometimes developers need to optimize their code to make it more efficient. This may result in obfuscated source code (e.g., by coding some parts in assembly language). Some web application level attacks consist of script injection. To succeed, attackers need to know the structure of the website and HTML pages. Obfuscation can help in protecting sensitive and critical HTML pages (e.g., connection and administration pages).

Several obfuscation techniques can be used such as base64-encoding, XORing, randomly renaming functions, methods overriding, tab-return-space deletion, shuffling, etc. Encryption is also an obfuscation technique but with issues because the cyphered data will remain viewable to those with valid keys.

Note: Data obfuscation is often used by attackers to hide their malicious code and attacks.

5.7.2 Testing the Effectiveness of Data Obfuscation Approaches

Tight configuration control between the obfuscated data and keys used for the obfuscation is needed to ensure the correct versions of keys are used. Otherwise, the data can not be un-obfuscated for use.

Since private data could be involved in some tests, data obfuscation may be used for testing purposes to render production data used in a system test environment as anonymous. Sensitive data, such as user information used by a health information system, must not be divulged to testers. Tests could include:

- Brute force or dictionary attacks to attempt to get plain data from obfuscated data

Tests to verify obfuscation of code could include:

- Reverse-engineering of byte code Java (e.g., regenerate Java source code by using Java Decompiler) or .Net programs (e.g., retrieve .Net source code with .NET Reflector),
- Brute force attacks, because some obfuscation mechanisms are vulnerable (e.g., by using unXOR [Chopitea]).

In theory, code cannot protect itself against de-obfuscation, because debugging can always be used. While tools exist for the purpose of protecting code against de-compilation, there are still risks and limitations in protecting proprietary information represented by code.

5.8 Training

5.8.1 The Importance of Security Training

Humans are often the weakest link in the overall security picture. Therefore, consistent and continuous training is needed to remind people of the importance of following established security policies and to stress why the policies are needed. This training should occur throughout the software lifecycle process and be updated as new policies are added and new threats arise. Training should cover the identification of social engineering attacks and insider threats.

5.8.2 How to Test the Effectiveness of Security Training

For example, a security training program might address the importance of having strong user passwords that are kept confidential.

Tests could include:

- Social engineering to attempt to get a user to reveal their password during a phone conversation with a fake technical support person
- Looking around desks for sticky notes with passwords on them (especially under keyboards)
- Running password audit tools to identify weak passwords. One risk of this type of tool is that passwords may be viewable to the person running the test.

Another example would be that a developer fails to place a field level edit to prevent the entry of SQL commands in a data entry field. Because of this mistake, a security tester is able to inject a SQL command and see the contents of a customer database. This would indicate that the developer needs additional training in secure coding practices. It would also be good to examine the coding practices of other developers to see if this practice is widespread and a general process improvement initiative is needed.

A third example would be where a tester attempts to gain unauthorized physical access to an office and view documents that have been left in the open.

6. Human Factors in Security Testing - 105 mins.

Keywords

attacker, botnet, computer forensics, hacker, reconnaissance, script kiddies

Learning Objectives for Human Factors in Security Testing

6.1 Understanding the Attackers

AS-6.1.1 (K2) Explain how human behavior can lead to security risks and how it impacts the effectiveness of security testing

AS-6.1.2 (K3) For a given scenario, demonstrate the ability to identify ways in which an attacker could discover key information about a target and apply measures to protect the environment

AS-6.1.3 (K2) Explain the common motivations and sources for performing computer system attacks

AS-6.1.4 (K4) Analyze an attack scenario (attack performed and discovered) and identify possible sources and motivation for the attack

6.2 Social Engineering

AS-6.2.1 (K2) Explain how security defenses can be compromised by social engineering

6.3 Security Awareness

AS-6.3.1 (K2) Understand the importance of security awareness throughout the organization

AS-6.3.2 (K3) Given certain test outcomes, apply appropriate actions to increase security awareness

6.1 Understanding the Attackers

In the context of information security, the human being is both the largest threat and weakest point in defense.

Security attacks are performed by people with a variety of skills and motivations. Additionally, humans are the (biggest) enablers for most security attacks. Just understanding the technology of security and implementing it is not enough to defend against attacks. It is also important to understand the mindset, motivations and methods of the malicious attackers and be aware of the human weaknesses in the line of defense.

6.1.1 The Impact of Human Behavior on Security Risks

The key phase in any attack is the information gathering phase (reconnaissance) where the attacker tries to find and collect information about the target. All information that is published, sometimes unknowingly, about an organization, systems in use, etc., and stored on the Internet will be found and can/will be used in an attack. It is not a question of "if" but a question of "when". Besides the information published officially by the organization, employees are also publishing information about the company on their social networks. The amount and content of this information is continuously changing, often presenting some key information to the attackers.

Attackers do not use a security policy or predefined procedures when they attack a system. Based on the information they can collect they decide their strategy. They will update their knowledge base for each attack performing selective searches and 'visiting' already known IP addresses.

When the security policy for a company is formulated, it is usually based on the situation and facts that are available. Sometimes that does not include all the publicly accessible information and, even if it did, that information is likely to change. Security tests that were valid when they were created may not provide adequate coverage when published information changes.

6.1.2 Understanding the Attacker Mentality

During the reconnaissance or footprinting activity, the attacker will try to find all kinds of information about the target using passive and/or active means. Most IT equipment that faces public networks leaves a footprint on those networks. These footprints can and will be found. Google (including Google Earth and Street View) or other search engines, Shodan [Web-5], Facebook, LinkedIn and other social networks are the first sources used for finding information about the target. IP addresses, web pages, telephone numbers, names and email address structures, OS and applications can all provide information that is useful to the attacker.

A possible use of the Google search engine is to find specific information about a target. Hundreds of queries can be found in The Google Hacking Database [Web-4]. Shodan [Web-5] is another tool that is used to find specific information, e.g., which companies in a particular area are running an Apache server with a vulnerable version.

Most of this information can be found passively without actually connecting to the target system. Other tools used include:

- Whois [Web-13]
- Ripe (European IP Networks) database [Web-12]
- DNS searches [Web-25]

With active reconnaissance techniques the attacker uses tools to detect hosts, open ports, OS and applications by touching the system. Methods and tools used here include:

- Pinging - Fping [Web-15], Hping [Web-19]
- TCP/UDP scan – Nmap [Web-20], Zenmap [Web-21]
- OS detection – Nmap [Web-20], Xprobe2 [Web-22]
- Service fingerprinting (Nmap has capabilities to determine also the type and version of the service running on the discovered open port. This is done by comparing the "fingerprint" of the discovered service with Nmap's own database of fingerprints.)

As hacking into a system is prohibited by law in most, if not all countries, the hacker will try to destroy all evidence of the hack afterwards. Other reasons for destroying evidence are to lengthen the stay, continue the use of the system in the future, and to use the compromised system or network of systems (botnets) for attacking other systems. The attacker can deploy tools such as NetCat [Web-14] for this or use web sites like IP TRacer [Web-7], as well as tunneling or altering log files.

Other methods and tools used for hiding evidence include hide tools [Web-16], rootkits, and file streaming. All, or most, tools mentioned here are accessible using the Internet. Downloading the latest version of Kali Linux [Web-17] and a search on the OWASP site [OWASP1] will give access to many of those tools.

6.1.3 Common Motivations and Sources of Computer System Attacks

Many attacks and breaches on information systems come from inside the organization. Malicious system users (internal hackers or insider threats) will try to compromise systems while being an authorized network user. Most of the time revenge is the motivation but recent trends are showing an increase for economic espionage or theft.

External hackers are responsible for the minority of attacks. Curiosity for information was one of the first motivators to hack into information systems, and still is. To have some information from major companies or organizations and knowing that others do not is another motivator (prestige). Other motivators include notoriety or fame, challenge, boredom and revenge where the latter is considered the most dangerous form (highest motivation).

Attackers are often classified by their motivation and abilities. At the lower end of the spectrum of attackers are "script kiddies" who simply execute the attacks others create, while at the upper end of the spectrum are professional (governmental, hacktivism) organizations and individuals. Hacktivism is the attacking of systems based on mainly political but also economical or other demographical grounds.

Motivation can scale from playing around for fun to knocking a system or organization completely down for whatever reason (e.g., political, ideologically, economic, warfare, commercial, terrorism).

Hacking abilities vary from individuals having some system and networking knowledge working with a simple home computer to highly trained and educated professionals having access to labs, proxy networks and all other technical equipment needed. Having a picture about the potential attackers will help an organization implement the necessary protection and gives a guideline for the security test strategy.

6.1.4 Understanding Attack Scenarios and Motivations

A security incident is defined as a security relevant system event in which the system's security policy is disobeyed or otherwise breached. [RFC2828]

Finding out what happened and who was responsible for the security-related incident is a target for the discipline of computer forensics [Web-8] where one focuses on finding digital evidence of the attack.

The evidence recovering process is based on three phases:
1. Acquire and authenticate
2. Analyze
3. Report

6.1.4.1 Acquire and Authenticate

The organization's incident management process should restore the system to its original state (pre-attack) after evidence is collected and stored. It starts when the system administrator is alerted by the IDS or other monitoring means. Other typical symptoms of security incidents are:
- Suspicious log entries
- Unexplained user accounts
- Modified files/folders
- Unusual services running
- Unusual system behavior
- Unsuccessful login attempts

After being alerted, the process to perform is as follows:
1. Make a snapshot or copy of the system under investigation to collect all the evidence needed.
2. After authenticating the evidence (it is a genuine and complete copy), create a copy and store it in a safe location.
3. Analyze the evidence.
4. After the forensics process is finished, remove the cause for the incident (eradication).
5. The system is brought back to its normal state (recovery).

During these steps, any vulnerabilities are removed with patches or installing new software. In reporting the results, the process followed should be described along with the tools used during this process.

6.1.4.2 Analyze

After hacking attempts, it may be possible to find the source of the attacks by examining the system log files and active network connections. It is important to make copies of all log files and capture the process status information. During an active attack, it may make sense to gather system information related to the attacker(s) before locking them out.

Any attack via the Internet can be traced to the originated IP address whether it used email or Internet connections. It is only a matter of time, money and effort, and an assessment of the costs involved. Most attackers use proxies or chains of proxies, Tor network [Web-9], or other free anonymous options to cover their real IP address. The more proxies attackers use, the more time is needed to trace the originating address. Local laws based on the physical locations of the proxies may also obstruct this investigation.

Discovering intruders and tracing an IP address back to its origin can be done with tools such as Netstat (Windows) [Web-10], Tracert [Web-11] and the IP Tracer website [Web-7]. Netstat shows the connections to a machine, ports and services running. This tool can be used to search for any strange or unknown IP address or port number. Note: There is also a tracert utility in the Microsoft Windows Operating System (in Linux and OS/X, it is "traceroute"), but the web-based services mentioned above are independent from these O/S utilities.

In the header of an email containing viruses, the IP address of the ISP which sent the email may be shown. However, for most web-based email (Gmail, Yahoo mail, Outlook.com), this is the IP address of

the provider. To find the real IP address one has to look for the X-Originating-IP value. Using the Whois databases [Web-13] will lead to the details that can be used to contact the ISP organization to continue the investigation. It should be noted that email can originate from private servers and open relay mail servers. In that case, it may be very difficult to identify the actual source of an email message.

Investigating attacks that used a botnet is difficult. There is no need for the attacker to have an online connection with the botserver or the botclients so tracing is very difficult or nearly impossible. In this case, investigating the clients may lead to the server, but one must have access to the server in order to investigate the true source of the attack. Owners of these servers may not be aware that their machines are part of a botnet.

6.1.4.3 Report

The reporting of security vulnerabilities is described in Chapter 7.

6.2 Social Engineering

We can implement all the technical defenses we can think of to protect digital assets from the outside world, but in the end it all comes down to the fact that employees (users and administrators) need to have access to these assets to do their work. They may need to use authentication to get access from their desktops, notebooks, smart phones, tablets or other means. Any physical security defense to protect access to the office and office equipment is meaningless if the security to the working area of the IT manager at his home can easily be compromised.

It is the human being and his or her behavior that is the greatest threat to security. If people are sloppy with sensitive information, this leaves too many footprints to secure places, and broadcasts this information too loudly (both orally and electronically) in public places.

Social engineering is the art of exploiting the human using its general behavior as an attack vector. As social beings, people are willing to trust and help strangers. This creates a vulnerability to attack. By manipulating, influencing and persuading helpful people, an attacker will try to gather access, authorization details or other kinds of sensitive information.

The exploits can be performed by direct human interaction or by using computer/network equipment.

Direct human interaction can be done in person including:

- Tailgating or piggybacking (someone who lacks the proper authentication following an employee into a restricted area)
- Eavesdropping (listening in to someone else's private conversations without their knowledge)
- Shoulder surfing (looking over someone's shoulder without their knowledge as they perform tasks on the computer or in writing)
- Using the telephone (such as getting a password from an unsuspecting user by acting as another person, such as a manager or technical support person)

Computer-based social engineering can be done by:

- Sending emails infected with malware.
- Using chat or instant messaging applications. By using chat and instant messenging applications, any anonymous person may have a chat with another anywhere in the world, without knowing the other person's true identity. In addition, data through instant messengers can be easily sniffed.
- Using pop-up screens. For example, a window might appear on a user's computer screen with a message to the user that the network connection has been lost. At that point, the user is

prompted to re-enter their user name and password. A program previously installed by the intruder can then transmit the information to a remote site.
- Sending spam email. Spam emails are rife with fraudulent offers and links. Clicking on these links can install malware that can expose an entire network.
- Persuading people to visit infected (manipulated) websites. These phishing attempts may be broadly sent, or may be highly individualized (spear phishing).

There is no single defense against social engineering. Defenses can be implemented to control the damage (e.g., provide the least level of privilege that still allows someone to perform their work, separation of duties, rotation of duties), but the main defense is education and awareness at all levels in the organization.

6.3 Security Awareness

6.3.1 The Importance Of Security Awareness

The threat model is constantly changing as mentioned in other chapters in this syllabus. Networks are evolving, new applications are introduced, new interfaces are made operational, and new vulnerabilities are introduced and discovered.

Besides these technical aspects, there is the human factor. Risks that were once identified but did not become issues are easily forgotten and safeguards are withdrawn. This offers increased opportunities for both hacking and social engineering attacks. Regular security awareness training is needed to keep security administrators and all employees alert and informed about changes in the threat model. Security awareness training can focus on different user groups: developers, operations, management and general user staff.

6.3.2 Increasing Security Awareness

It is important to keep a “security aware” mindset. Besides general information about security defenses in the company, training should contain real case studies - discovered during security testing or in actual incidents. Building on these cases, it should be easier to discuss any defenses or changes to be implemented in the organization.

An outline for this section in the awareness training should include answers to the following questions:
- How did they (we) do it?
- What were the business consequences?
- What were the costs to investigate and process the incident?
- What were the costs to repair the problem?
- How could the incident have been avoided?
- Which changes will be implemented?

7. Security Test Evaluation and Reporting - 70 mins.

Keywords

acceptance criteria, attack vector, dashboard, exit criteria

Learning Objectives for Security Test Evaluation and Reporting

7.1 Security Test Evaluation

AS-7.1.1 (K2) Understand the need to revise security expectations and acceptance criteria as the scope and goals of a project evolve

7.2 Security Test Reporting

AS-7.2.1 (K2) Understand the importance of keeping security test results confidential and secure

AS-7.2.2 (K2) Understand the need to create proper controls and data-gathering mechanisms to provide the source data for the security test status reports in a timely, accurate, and precise fashion (e.g., a security test dashboard)

AS-7.2.3 (K4) Analyze a given interim security test status report to determine the level of accuracy, understandability, and stakeholder appropriateness

7.1 Security Test Evaluation

Measuring security test results and evaluating status with respect to security expectations, exit criteria, and/or acceptance criteria are required to determine test completion.

It is difficult to know all the security risks at the outset of a project. In addition, stakeholder and user expectations sometimes change regarding the level of security needed. For example, the awareness of a new threat might cause stakeholders to require higher levels of security than initially thought. This is one reason why security risk assessments need to be revisited throughout a project and the results incorporated into security test planning and execution.

7.2 Security Test Reporting

7.2.1 Confidentiality of Security Test Results

It is true that the average tester knows more about the test object after testing is finished compared to most developers or designers. When testing thoroughly one may find the most important weaknesses and strong points of the system. The same applies for security testing.

By testing the security implementation one may find hidden holes and security vulnerabilities. The difference lies in the possible negative impact in communicating these vulnerabilities to people other than the direct stakeholders. A generally good practice is that information should be made available only to those who need to know. This applies specifically to security test results; to be conservative with sharing this type of information is considered a good practice.

7.2.2 Creating Proper Controls and Data Gathering Mechanisms for Reporting Security Test Status

The impact and effect of a security vulnerability is normally judged to have a higher sensitivity compared to “normal” defects. This leads to the need to be more precise and accurate about reporting the nature of the defect and the implied risks. In most projects security defects are categorized with a higher severity than comparable functional defects.

The latter implies that management has a higher focus on security defects, their risks and possible resolutions. Security defect reporting must carefully assess the impact of a discovered problem, the accuracy of test results, and should be available in a well-defined and timely manner. It is good practice to discuss with the management how and when they would like to have access to the security defect reports.

7.2.3 Analyzing Interim Security Test Status Reports

Security test reports may be produced throughout the security testing process or only at the end of security testing (such as at the end of system security testing or at the end of security tests performed as part of acceptance testing). Early security test reporting is encouraged because it allows more time to remediate security vulnerabilities. If the security testing process is following the one described in this syllabus, the test team may discover vulnerabilities and document observations during all test activities.

The structure of a security test report should contain the following sections:

1. Report identifier
2. Summary
   a. Executive summary

   b. Key findings
3. Variances
   a. Test process followed
   b. Any variances from planned test process
   c. Methods and tools (configurations, policies) used
4. Comprehensive assessment
   a. Evaluation of test coverage based on the criteria indicated in the test plan
   b. Explanation for any items or features that were not tested
5. Summary of results
   a. Summarization of the results of security testing
   b. List of all resolved security vulnerabilities and their resolutions
   c. List of all unresolved vulnerabilities
6. Evaluation
   a. Evaluation of the observed test results and their status based on exit criteria
   b. Risks identified (classifications) and impact of unresolved security vulnerabilities
7. Summary of activities
8. Approvals

The effectiveness of security testing reporting depends on the following:

- The timing of the report
- The content of the report
- The recipients of the report
- The tuning of the content to match the recipients' need for the information

Multiple reports may be required to meet the needs of various stakeholders. For example, the content of a report for executive management will not be the same as the content for a system architect.

8. Security Testing Tools - 55 mins

Keywords
none

Learning Objectives for Security Testing Tools

8.1 Types and Purposes of Security Testing Tools
AS-8.1.1 (K2) Explain the role of static and dynamic analysis tools in security testing

8.2 Tool Selection
AS-8.2.1 (K4) Analyze and document security testing needs to be addressed by one or more tools
AS-8.2.2 (K2) Understand the issues with open source tools
AS-8.2.3 (K2) Understand the need to evaluate the vendor's capabilities to update tools on a frequent basis to stay current with security threats

8.1 Types and Purposes of Security Testing Tools

The exploits devised by the hacking community have driven the development of security testing tools to defend against these threats. Even from the early hacking activities (such as password cracking) simple tools were invented, created and improved by those using them. Tools that proved to be effective were shared in the hacker community and further improved and enhanced. At first these tools were developed for dedicated tasks and environments. Usability was not an issue as almost all users had a technical background. Eventually, some of the hacker tools became the basis for legitimate security testing tools used by information security administrators and testers.

As an example, “John the Ripper” was an early open source password cracking tool, originally used by hackers to guess (crack) passwords and gain access to Unix networks or applications. Today, this tool has been refined and is used for legitimate purposes to detect weak Unix passwords. [Web-26]

As major test and software development tool suppliers and specialized tool suppliers began to develop security testing tools, many of these tools achieved broader functional capabilities and improved usability. However, this broad functionality led to more complex tool configurations and implementation concerns.

At the same time that early security tools were emerging, the first versions of frameworks such as Nessus, Metasploit and others were developed as open source tools offering improved and expanded functionality and, in some cases, also an easy learnable GUI.

Today, the number of available security testing tools is massive. For almost any environment or task one is able to find a dedicated test tool, either as open source or licensed. The challenge with all these tools is that most of them are intelligent systems deploying non-standardized tests. All developers of these systems agree more or less about how to test security defenses or test for vulnerabilities. However, these tools may use different test data, different test implementations and different interpretations of the results.

Security test tools can be used to automate the evaluation of security defenses. Security testing tools can also be used to detect known types of vulnerabilities. With the understanding that the same type of security defense or vulnerability can be implemented in different ways, selecting and using security testing tools is a challenge for the security tester because the tools differ in how they find vulnerabilities and validate defenses.

The Web Application Security Consortium [Web-18] and OWASP [OWASP1] web sites offer lists of categorized tools. The penetration testing framework Backtrack [Web-23] (or Kali Linux [Web-17]) presents other ways to classify security testing tools.

The number of commercial security tools is rather limited compared to the number of open source tools. At the time this syllabus was developed (2016) we were able to find only a limited number of resources presenting a more or less complete overview of reliable and trustable open source security tools. One list of security tools can be found at https://sectools.org [Web-24]. It is expected that the advanced security tester maintains his or her own list of available tools and keeps that list updated as the tool market changes.

Both static and dynamic analysis tools are useful in security testing. The advantage of static testing is that it can be performed very early in the development lifecycle. Static analysis tools are available for most software languages and usually have an ability to report on security aspects.

The difference between dynamic and static testing tools in the security testing context is sometimes a little confusing compared to other test types. The definition for static testing is related to performing test activities while the system or object under test is not in operational mode. It is not unusual for the security dynamic testing tools to probe the system rather than the application under test. In this perspective these dynamic testing tools are used as a kind of static testing tools. For example, a dynamic security test tool may perform a static scan of a database. Of course, if the entire system is considered as the test object, then the tools truly are dynamic testing tools.

8.2 Tool Selection

8.2.1 Analyzing and Documenting Security Testing Needs

Among others, the following documents can form a test basis for security testing:

- The organization's security policy
- The organization's test policy
- Results of threat and risk analysis for the actual system/project
- Requirements and other system specifications
- System architecture and design
- Security (test) strategy
- The system or application under test
- Known security threats, exploits and vulnerabilities
- User profiles

All these and more can provide information on threats and on vulnerabilities which could be exploited. Requirements and design documents should indicate how the data or information is protected. This will lead to an overview of:

- Interfaces to be tested (including the GUI)
- Protocols and standards to be verified
- Web coding guidelines that promote secure coding practices to be used
- System component configurations to be verified (hardened)

It needs to be determined if security testing will be a development activity or a maintenance/operations activity. All this information will lead to the requirements for the security test tool set.

8.2.2 Issues with Open Source Tools

See [ISTQB_ATM_SYL] for a complete discussion of issues that may be encountered with open source tools.

As mentioned before, many security testing tools are found in the open source domain. These tools are distributed and can be used under a wide variety of licenses which both allow free use and modification of the source code. Not all companies or projects may consider using open source tools in their development processes. Based on regulatory compliancy issues organizations can be forced to use commercial or otherwise certified tools only.

There are many advantages and disadvantages related to tools under these licenses. In many cases, open source tools can be obtained for free, but the organization may need to have technical capacity available for support and specific configuration. If this capacity is lacking, then a cost may be incurred to obtain it from the developer of the software. Administration and user manuals, if any, are mostly written with a specific (technical) audience in mind and will most often not describe or cover all functionality in

the tool. Media channels like YouTube are lately an additional source for information regarding the use of these tools.

Aspects to consider when setting up the ROI calculation for any open source tool include:
- The limited scope of the tool (in most cases no further or other functionality is offered)
- The time to learn to administer, configure and use the tool
- The time to invest in the user forums and groups during the lifecycle
- The time needed to update and upgrade (and the internal policy on upgrades)
- The future direction of the tool (some tools may go away or go commercial)
- The level of responsiveness in the support community for the tool

For most businesses or projects the number of licenses needed for security testing tools is limited to one or a few. Only large companies will consider more licenses. The number of licenses will mainly be based on the total sum of functionality areas delivered by the tool (e.g., web application, web services, code analysis, others) and the assumed frequency, time of use for these services and the number of security testers using the tool.

8.2.3 Evaluating a Tool Vendor's Capabilities

If a tool is purchased from a vendor, that vendor should offer a number of services to allow the security testing service to start and grow to the needed level of internal support.

The following attributes can be used to evaluate vendor capabilities:
- Types of licenses offered (fixed/desk/floating/token)
- License scalability options (per functional area, number of licenses)
- Helpdesk/support facilities (hours of support)
- Forum/user community
- Update frequency
- Administration and user manuals
- Support and maintenance contracts

9. Standards and Industry Trends - 40 mins.

Keywords
consensus-based standard

Learning Objectives for Standards and Industry Trends

9.1 Understanding Security Testing Standards

9.1.1 (K2) Understand the benefits of using security testing standards and where to find them
9.1.2 (K2) Understand the difference in applicability of standards in regulatory versus contractual situations

9.2 Applying Security Standards

9.2.1 (K2) Understand the difference between mandatory (normative) and optional (informative) clauses within any standard

9.3 Industry Trends

9.3.1 (K2) Understand where to learn of industry trends in information security

9.1 Understanding Security Testing Standards

Standards of various types provide visibility into professional consensus or regulatory obligations. A consensus-based standard represents the considered opinion of a knowledgeable body of experts and is made available for voluntary use (in whole or in part) in contractual agreements between suppliers and customers. There are lesser types of so-called standards that arise from more informal or self-identified groups and may be vendor-specific.

In regulated industries (including the medical, financial, transportation, and energy sectors) government agencies may require compliance with their own regulations or their interpretations of otherwise voluntary standards.

9.1.1 The Benefits of Using Security Testing Standards

Standards, in general, provide guidance and consistency in performing a task. Typically, standards are developed by subject matter experts based on consensus of effective practices. The following are benefits of using security testing standards:

- They define a framework for security testing eliminating the need to start from a blank page.
- They describe effective defenses and how to test for the most common security attacks.
- The standards can be tailored to meet the needs of the project or organization.
- Due diligence in security testing can be demonstrated by following recognized security testing standards.

9.1.2 Applicability of Standards in Regulatory Versus Contractual Situations

In regulated activities, all parties need to be aware of their obligations to comply with imposed standards because failure to do so could delay or prevent approval of the product under development and, in extreme cases, result in financial or criminal sanctions.

In contractual situations, standards provide a reasonable and convenient basis for negotiating agreement on project and product requirements; they provide a starting point instead of the parties beginning with nothing. Consensus-based standards allow best practices to be communicated and adopted or adapted to the particular situation.

Unless imposed unilaterally by a regulator or in a non-negotiable contract, standards can be used as the basic framework for a negotiated agreement or self-imposed on the conduct of one's own work. If a contract is awarded on the basis of a claim or agreement to comply with specific standards, then the entity has the obligation to follow those standards strictly and document any deviations.

9.1.3 Selection of Security Standards

Certainly, not all security standards apply to all situations. It is the responsibility of an organization to research the most appropriate standard(s) for their systems, applications, sensitive digital assets, level of risk, and compliance requirements. It is also important to understand that many standards may be tailored to meet an organization's specific requirements.

A list on common security standards can be found in Chapter 10.

9.2 Applying Security Standards

Note the precise use of language within any standard: The word *shall* identifies the mandatory requirements to be followed to conform to the standard, while the words *should* and *may* indicate

optional tasks that are not required to claim conformance to the standard. One typical misuse is to confuse this distinction by either requiring an optional item or treating an obligatory item as optional.

Organization- or project-specific situations may dictate deviation from the strict sense of a standard in use. Justification for omissions, modifications, or additions to the content of the standard need to be documented and agreed upon by all parties.

9.3 Industry Trends

9.3.1 Where to Learn of Industry Trends in Information Security

Both general purpose and industry-specific news services (publications, websites, email distributions) and events (conferences, trade shows, professional society meetings) feature information and discussion of new or growing concerns. Belonging to a focused professional society or community of practice will likely provide timely and targeted updates. With the speed at which new exploits develop, electronic alerts can offer the most immediate actionable responses.

Periodic publicity of the most frequent or damaging exploits can identify widespread trends, but one should pay particular attention to issues more specific to the industry, application area, or products with which one is working. These issues are more likely to be communicated in specialized publications and news services or at technical conferences and professional events.

9.3.2 Evaluating Security Testing Practices for Improvements.

As new technologies or novel uses of existing technology are introduced, there is often a window of opportunity for misuse or exploitation of the technology until its risks and limitations are better understood.

For example, consider mobile devices with location-aware services. In exchange for convenience or other incitements, individuals seem willing to allow minute-by-minute tracking of their movements and activities.

A larger range of motivations and greater resources are emerging from criminal, hacktivist, economic, and political agents. Extortion and protection schemes have moved from physical threats to digital domains.

Large ad hoc networks of ideologically driven individuals might be directed on very short notice against targets of their ire. Corporate espionage is often well-funded and motivated. Nation-states seeking economic and military advantage are particularly well-resourced and may believe themselves immune to sanctions or retaliation.

Because the threats are consistently changing and evolving, security testers must always be ready to address the next threat. Awareness of the industry, close tracking of security trends and the acquisition of the most appropriate tools provide the best defense for an organization.

10. References

ISTQB Documents

[ISTQB_FL_SYL] ISTQB Foundation Syllabus, 2011

[ISTQB_ATM_SYL] ISTQB Advanced Test Manager Syllabus, 2012

[ISTQB_ATTA_SYL] ISTQB Advanced Technical Test Analyst Syllabus, 2012

Standards

[ISO/IEC/IEEE 29119-3] - Software and systems engineering -- Software testing -- Part 3: Test documentation

[IEEE 12207] - ISO/IEC/IEEE Standard for Systems and Software Engineering - Software Life Cycle Processes

COBIT - http://www.isaca.org

ISO27001 – Information Security Management - http://www.iso.org/iso/home/standards/management-standards/iso27001.htm

PCI – Payment Card Industry Standard - https://www.pcisecuritystandards.org/

Books

[Chapman, 2000] Chapman, Cooper, Zwicky, Building Internet Firewalls, O'Reilly & Associates, 2000.

[Jackson, 2010] Jackson, Christopher; Network Security Auditing, 2010.

Articles

[ComputerWeekly] http://www.computerweekly.com/news/2240113549/Cattles-lost-backup-tapes-highlight-risk-of-unencrypted-data-storage

[Northcutt, 2014] Northcutt, Stephen; Security Controls, SANS Institute.

[Washington Post, 2007] http://www.washingtonpost.com/wp-dyn/content/article/2007/05/04/AR2007050402152.html

Guides

[Bittau] Cryptographic protection of TCP Streams (tcpcrypt)
https://tools.ietf.org/html/draft-bittau-tcp-crypt-04

[CERT1] Top 10 Secure Coding Practices
https://www.securecoding.cert.org/confluence/display/seccode/Top+10+Secure+Coding+Practices

[CERT2] http://www.cert.org/secure-coding/publications/index.cfm

[CERT3] http://www.cert.org/secure-coding/tools/index.cfm

[IEEE1] Avoiding the Top 10 Security Flaws
http://cybersecurity.ieee.org/center-for-secure-design/avoiding-the-top-10-security-flaws.html

[MDA1] MDA Glossary, DoD Missile Defense Agency, www.mda.mil

[NIST 800-30] NIST Special Publication 800-30, Rev 1, *Guide for Conducting Risk Assessments* (2012)

[NISTIR 7298] Glossary of Key Information - Security Terms, Revision 2 (2013)

[OWASP1] OWASP Secure Coding Practices Quick Reference Guide
https://www.owasp.org/index.php/OWASP_Secure_Coding_Practices_-_Quick_Reference_Guide

[OWASP2] OWASP Risk Rating Methodology
https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology

[OWASP3] OWASP Sample Authorization Form
https://www.owasp.org/index.php?title=Authorization_form

[PP-IDS] US Government Protection Profile Intrustion Detection System for basic robustness environments, version 1.7, 25 July 2007.

[SANS1] 25 Most Dangerous Software Errors – http://www.sans.org

[SANS2] Password Construction Guidelines - https://www.sans.org/security-resources/policies/general/pdf/password-construction-guidelines

[WAFEC] Web Application Firewall Evaluation Criteria, wasc-wafec-v1.0.pdf, 2006.

Reports

[WhiteHat Security, 2014] https://www.whitehatsec.com

Web

[CERT4] Vulnerability Notes Database - http://www.kb.cert.org/vuls/

[Chopitea] tomchop.me/2012/12/yo-dawg-i-heard-you-like-xoring/

[EICAR] www.eicar.org

[RFC2828] Internet Security Glossary - http://www.rfc-archive.org/getrfc.php?rfc=2828

[Web-1] Top 20 Critical Security Controls - http://sans.org

[Web-2] National Vulnerability Database - https://web.nvd.nist.gov/view/ncp/repository

[Web-3] Website Security Statistics Report - https://www.whitehatsec.com/resource/stats.html

[Web-4] The Google Hacking Database – http://hackersforcharity.org/ghdb

[Web-5] Shodan - shodanhq.com

[Web-6] NetCat - http://sectools.org/tool/netcat/

[Web-7] IP Tracer - http://www.ip-adress.com/ip_tracer

[Web-8] Computer Forensics, Cybercrime and Steganography Resources – http://www.forensics.nl

[Web-9] Tor Project - https://www.torproject.org/

[Web-10] Netstat - https://technet.microsoft.com/en-us/library/Bb490947.aspx

[Web-11] Tracert – http://www.tracert.com

[Web-12] RIPE Scan - https://www.ripe.net

[Web-13] Whois - https://www.whois.net/

[Web-14] NetCat – http://netcat.sourceforge.net/

[Web-15] Fping – fping.org

[Web-16] Hidetools – http://hidetools.com/

[Web-17] Kali Linux – https://www.kali.org/

[Web-18] Web Application Security Consortium – http://www.webappsec.org/

[Web-19] Hping - http://www.hping.org/

[Web-20] Nmap - https://nmap.org/

[Web-21] Zenmap - https://nmap.org/zenmap/

[Web-22] Xprobe2 - http://null-byte.wonderhowto.com/how-to/hack-like-pro-conduct-os-fingerprinting-with-xprobe2-0148439/

[Web-23] BackTrack - http://www.backtrack-linux.org/

[Web-24] Top 125 Network Security Tools - at https://sectools.org

[Web-25] DNS Lookup - https://who.is/dns/

[Web-26] John the Ripper - http://www.openwall.com/john/