

Certified Tester Security Test Engineer Syllabus

Versione 1.0.1

International Software Testing Qualifications Board



Avviso di copyright

Avviso di Copyright © International Software Testing Qualifications Board (di seguito denominato ISTQB®)

ISTQB® è un marchio registrato di International Software Testing Qualifications Board.

Copyright © 2024, Dr. Frank Simon (chair), Alain Ribault, Gabriel Firmino Barjollo, Michael Pott, Beata Karpinska, Maria Kispal, Frans Dijkman.

Tutti i diritti riservati. Con la presente gli autori trasferiscono il copyright a ISTQB®. Gli autori (come attuali titolari del copyright) e ISTQB® (come futuro titolare del copyright) hanno concordato le seguenti condizioni di utilizzo:

È possibile copiare estratti di questo documento per uso non-commerciale, a condizione che venga riconosciuta la fonte. Qualsiasi Training Provider accreditato (azienda accreditata alla Formazione) può utilizzare il presente Syllabus come base per un corso di formazione, a condizione che gli autori e ISTQB® siano riconosciuti come fonte e proprietari del copyright del Syllabus, e a condizione che qualsiasi pubblicità di tale corso di formazione possa menzionare il Syllabus solo dopo l'accREDITAMENTO ufficiale dei materiali di formazione da parte di un Member Board riconosciuto da ISTQB®.

Qualsiasi individuo o gruppo di individui può utilizzare questo Syllabus come base per articoli e libri, a condizione che gli autori e ISTQB® siano riconosciuti come fonte e proprietari del copyright del Syllabus.

È proibito qualsiasi altro utilizzo di questo Syllabus senza avere prima ottenuto l'approvazione scritta di ISTQB®.

Qualsiasi Member Board riconosciuto da ISTQB® può tradurre questo Syllabus, a condizione che riproduca il suddetto avviso di copyright nella versione tradotta del Syllabus.

Storico delle review

Versione	Data	Osservazioni
1.0	18-10-2024	Versione finale
1.0.1	31-01-2025	Versione aggiornata e finale dopo review tecnica

Indice dei contenuti

Avviso di copyright	2
Storico delle review	3
Indice dei contenuti	4
Ringraziamenti	7
0 Introduzione.....	8
0.1 Scopo di questo Syllabus	8
0.2 Business Outcome	8
0.3 Obiettivi di Apprendimento Esaminabili e Livelli di Conoscenza	9
0.4 L'Esame per il Certificato Security Test Engineer	9
0.5 Accreditamento	10
0.6 Gestione degli Standard.....	10
0.7 Livello di Dettaglio	10
0.8 Come è Organizzato questo Syllabus	10
1 Paradigmi di Sicurezza - 135 minuti (K3)	12
1.1 Livelli di Sicurezza degli Asset	13
1.1.1. Gli asset e il relativo livello di protezione.....	13
1.1.2. Sensibilità delle informazioni e testing di sicurezza	14
1.2 Audit di Sicurezza	14
1.2.1. Audit di sicurezza e testing di sicurezza.....	15
1.3 Il Concetto di Zero Trust.....	16
1.3.1. Che cos'è lo Zero Trust?	16
1.3.2. Zero Trust nel testing di sicurezza.....	16
1.4 Software Open Source (OSS).....	17
1.4.1. Il concetto di OSS e il suo impatto sul testing di sicurezza.....	18
2 Tecniche di Test di Sicurezza - 150 minuti (K3)	19
2.1 Applicare i Tipi di Test di Sicurezza in Base al Contesto del Test.....	20
2.1.1. Testing black-box, testing white-box e testing grey-box	20
2.1.2. Testing di sicurezza statico e dinamico	20
2.2 Applicare il Testing di Sicurezza	22
2.2.1. Affrontare i rischi di sicurezza nella progettazione dei test.....	23
2.2.2. Testing di riconciliazione e testing di ricertificazione	24
2.2.3. Testing di identificazione, autenticazione e autorizzazione	26

2.2.4.	Crittografia (Encryption)	27
2.2.5.	Testing delle tecnologie di protezione	28
3	Il Processo di Test di Sicurezza - 120 minuti (K3)	31
3.1	Il Processo di Test di Sicurezza	32
3.1.1.	Il processo di test di sicurezza ISTQB	32
3.1.2.	L'ambiente di test di sicurezza	34
3.2	Progettare i Test di Sicurezza per i Livelli di Test	35
3.2.1.	Progettazione dei test di sicurezza a livello di test di componente	35
3.2.2.	Progettazione dei test di sicurezza a livello di integrazione dei componenti	38
3.2.3.	Il testing di sicurezza nel testing di sistema e nel testing di accettazione	39
4	Standard e Best Practice del Testing di Sicurezza - 195 minuti (K3)	41
4.1	Introduzione agli Standard e alle Best Practice	42
4.1.1.	Standard e best practice	42
4.2	Applicare Standard e Best Practice Importanti per il Testing di Sicurezza	43
4.2.1.	Standard industriali per il testing di sicurezza	43
4.3	Utilizzare gli Standard e le Best Practice del Testing di Sicurezza	47
4.3.1.	Applicazione obbligatoria degli standard e delle best practice	47
4.3.2.	Applicazione volontaria degli standard e delle best practice	48
4.3.3.	Oracoli del test estratti da standard e best practice	48
4.3.4.	Vantaggi e svantaggi dell'utilizzo degli standard e delle best practice per il testing di sicurezza	48
5	Adattare il Testing di Sicurezza al Contesto Organizzativo - 195 minuti (K4)	50
5.1	L'Impatto delle Strutture Organizzative nel Contesto del Testing di Sicurezza	51
5.1.1.	Analizzare un determinato contesto organizzativo e determinare quali aspetti specifici considerare per il testing di sicurezza	51
5.2	L'Impatto delle Normative sulle Politiche di Sicurezza e Come Testarle	53
5.2.1.	L'impatto delle normative governative sulle normative di sicurezza	54
5.3	Analizzare uno Scenario di Attacco	56
5.3.1.	Gli scenari di attacco comuni	56
6	Adattare il Testing di Sicurezza ai Modelli di Ciclo di Vita dello Sviluppo Software - 165 minuti (K4)	63
6.1	Gli Effetti dei Diversi Modelli di Ciclo di Vita dello Sviluppo Software sul Testing di Sicurezza	64
6.1.1.	Modelli di sviluppo sequenziale	66
6.1.2.	Sviluppo software Agile	66
6.1.3.	La metodologia DevOps	68
6.2	Il Testing di Sicurezza Durante le Operazioni e la Manutenzione	70

6.2.1.	Il regression testing e il testing confermativo di sicurezza	70
7	Il Testing di Sicurezza come Parte di un Information Security Management System - 105 minuti (K3)	72
7.1	Criteri di Accettazione per il Testing di Sicurezza	73
7.2	Input per un Information Security Management System (ISMS)	74
7.3	Migliorare un ISMS Tramite l'Adeguamento del Testing di Sicurezza	76
7.3.1.	Migliorare la visione olistica di un ISMS	76
7.3.2.	Migliorare la misurabilità all'interno di un ISMS	77
8	Il Reporting dei Risultati dei Test di Sicurezza - 135 minuti (K3)	79
8.1	Il Reporting dei Test di Sicurezza	80
8.2	Identificare e Analizzare le Vulnerabilità	81
8.3	Chiudere le Vulnerabilità Identificate	82
8.3.1.	Nascondere una vulnerabilità	83
8.3.2.	Evitare una vulnerabilità	84
9	Strumenti del Testing di Sicurezza - 90 minuti (K3)	86
9.1	Categorizzazione degli Strumenti del Testing di Sicurezza	87
9.1.1.	Strumenti del testing di sicurezza white-box	87
9.1.2.	Strumenti del testing di sicurezza black-box	87
9.1.3.	Strumenti del testing di sicurezza grey-box	87
9.1.4.	Strumenti del testing di sicurezza statico	88
9.1.5.	Strumenti del testing di sicurezza dinamico	88
9.1.6.	Considerazioni per la selezione degli strumenti del testing di sicurezza	89
9.2	Applicare gli Strumenti del Testing di Sicurezza	90
9.2.1.	Comprendere l'utilizzo e i concetti degli strumenti del testing di sicurezza statico	90
9.2.2.	Comprendere l'utilizzo e i concetti degli strumenti del testing di sicurezza dinamico	91
10	Riferimenti	92
	Appendice A - Obiettivi di Apprendimento/Livello Cognitivo di Conoscenza	97
	Livello 1: Ricordare (K1)	97
	Livello 2: Comprendere (K2)	97
	Livello 3: Applicare (K3)	98
	Livello 4: Analizzare (K4)	98
	Appendice B - Matrice di Tracciabilità dei Business Outcome rispetto agli Obiettivi di Apprendimento	100
	Appendice C - Release Note	104
	Appendice D - Termini Specifici del Dominio del Testing di Sicurezza	105
	Indice	107

Ringraziamenti

Questo documento è stato formalmente rilasciato dalla General Assembly di ISTQB® il 18-10-2024.

È stato prodotto dalla security test task force di International Software Testing Qualifications Board: Dr. Frank Simon (chair), Alain Ribault, Gabriel Firmino Barjollo, Michael Pott, Beata Karpinska, Maria Kispal, Frans Dijkman.

Le seguenti persone hanno partecipato alla revisione, ai commenti e alla votazione di questo syllabus:

Attila Toth, Claude Zhang, Dani Almog, Daniel Säther, Gary Mogyorodi, Haiying Liu (Sandy), Ingvar Nordström, Laura Albert, Nancy Thompson, Petr Zacek, Szilard Szell, Tal Pe'er, Tamás Gergely, Claude Zhang, Daniel van der Zwan, Dingguofu, Giancarlo Tomasig, Klaus Skafte, Markus Niehammer, Mattijs Kemmink, Meile Posthuma, Michael Stahl, Nicola De Rosa, Samuel Ouko, Tauhida Parveen, Tetsu Nagata

Traduzione italiana a cura di ITA-STQB (www.ita-stqb.org) – info@ita-stqb.org

Revisione della traduzione italiana a cura di: Raffaele Bozzini, Salvatore Reale

0 Introduzione

0.1 Scopo di questo Syllabus

Questo Syllabus costituisce la base per l'International Software Testing Qualification per il Security Test Engineer. ISTQB® fornisce questo Syllabus come segue:

1. Ai Member Board, per la traduzione nella loro lingua locale e per l'accreditamento dei Training Provider. I Member Board possono adattare il Syllabus alle loro specifiche esigenze linguistiche e modificare i riferimenti per adattarli alle loro pubblicazioni locali.
2. Agli organismi di certificazione, per derivare le domande d'esame nella loro lingua locale adattate agli obiettivi di apprendimento di questo Syllabus.
3. Ai Training Provider accreditati ISTQB®, per produrre materiale didattico e determinare i metodi di insegnamento appropriati.
4. Ai candidati alla certificazione, per prepararsi all'esame di certificazione (come parte di un corso di formazione o in modo indipendente).
5. Alla comunità internazionale di System Engineering e Software Engineering, per promuovere la professione del testing dei sistemi e del software, e come base per libri e articoli.

0.2 Business Outcome

Questo paragrafo elenca i Business Outcome (Risultati di Business) previsti per una persona che ha conseguito la certificazione di livello Specialist Security Test Engineer.

Un tester con la certificazione Security Test Engineer è in grado di...

STE-BO1	Comprendere i paradigmi fondamentali della sicurezza e il loro impatto sul testing di sicurezza
STE-BO2	Utilizzare e applicare tecniche di test di sicurezza appropriate e conoscerne i punti di forza e i limiti
STE-BO3	Contribuire alla pianificazione, alla progettazione e all'esecuzione dei test di sicurezza
STE-BO4	Comprendere come gli standard del testing di sicurezza e le best practice di sicurezza possano essere utilizzati per il testing di sicurezza
STE-BO5	Adattare ed eseguire attività di testing di sicurezza in base al contesto specifico dell'organizzazione
STE-BO6	Adattare ed eseguire attività di testing di sicurezza in base a metodi di sviluppo e cicli di vita dello sviluppo software specifici
STE-BO7	Inserire i risultati del testing di sicurezza in un information security management system (ISMS) per una gestione attiva dei rischi di sicurezza

STE-BO8	Raccogliere, valutare e aggregare i risultati dei test e redigere un test report dettagliato che includa tutte le prove e i risultati
STE-BO9	Sulla base di un approccio del testing di sicurezza richiesto, identificare i requisiti adeguati per gli strumenti e assistere nella selezione degli strumenti del testing di sicurezza

0.3 Obiettivi di Apprendimento Esaminabili e Livelli di Conoscenza

Gli Obiettivi di Apprendimento (LO, Learning Objective) supportano i Business Outcome e vengono utilizzati per creare gli esami di Certified Tester Security Test Engineer.

In generale, tutti i contenuti di questo Syllabus sono esaminabili a livello K1. Al candidato può quindi essere chiesto di riconoscere, ricordare o richiamare una parola chiave o un concetto menzionato in uno qualsiasi dei nove capitoli di questo Syllabus. I livelli specifici degli obiettivi di apprendimento sono specificati all'inizio di ogni capitolo e sono classificati come segue:

- K1: Ricordare
- K2: Capire
- K3: Applicare
- K4: Analizzare

Ulteriori dettagli ed esempi di obiettivi di apprendimento sono riportati nell'Appendice A.

Tutti i termini elencati come parole chiave e parole chiave specifiche del dominio della sicurezza devono essere ricordati (K1), anche se non esplicitamente menzionati negli obiettivi di apprendimento.

0.4 L'Esame per il Certificato Security Test Engineer

L'esame per il Certificato Security Test Engineer si baserà su questo Syllabus. L'altro Syllabus, Security Test Analyst, si focalizza sulla progettazione di test di sicurezza che vengono successivamente eseguiti dal Security Test Engineer.

Le risposte alle domande d'esame possono richiedere l'uso di materiale basato su più di un paragrafo di questo Syllabus. Tutti i paragrafi del Syllabus sono esaminabili, ad eccezione delle Appendici. Gli standard e i libri sono inclusi come riferimenti, ma il loro contenuto non è esaminabile, al di là di quanto sintetizzato in questo Syllabus relativamente a tali standard e libri.

Per ulteriori dettagli, fare riferimento al documento Exam Structures and Rules per il Syllabus Security Test Engineer.

Il criterio di accesso per sostenere l'esame di certificazione Security Test Engineer è che i candidati abbiano un interesse per il testing del software. Tuttavia, è consigliato che i candidati abbiano anche almeno una conoscenza di base dello sviluppo software, del testing del software o del testing di sicurezza.

Nota per i Requisiti di Ingresso: il certificato ISTQB® Foundation Level deve essere ottenuto prima di sostenere l'esame di certificazione Security Test Engineer.

Il completamento di un corso di formazione accreditato non è un prerequisito per l'esame.

0.5 Accreditamento

Un Member Board ISTQB® può accreditare Training Provider, il cui materiale didattico segue questo Syllabus. I Training Provider dovrebbero ottenere le linee guida per l'accREDITamento dal Member Board o dall'ente che effettua l'accREDITamento. Un corso accREDITato viene riconosciuto essere conforme a questo Syllabus ed è autorizzato a gestire un esame ISTQB® come parte del corso stesso.

Le linee guida per l'accREDITamento di questo Syllabus seguono le Accreditation Guidelines pubblicate dal Working Group ISTQB® Processes Management and Compliance.

0.6 Gestione degli Standard

Il Syllabus Security Test Engineer contiene riferimenti a standard (ad esempio, NIST e ISO). Lo scopo di questi riferimenti è fornire un framework o una fonte di informazioni aggiuntive, se il lettore lo desidera. Si noti che il Syllabus utilizza questi standard come riferimenti e che non sono oggetto di esame. Per ulteriori informazioni sull'uso degli standard, delle best practice e delle norme, fare riferimento al Capitolo 4.

0.7 Livello di Dettaglio

Il livello di dettaglio di questo Syllabus consente di organizzare corsi ed esami consistenti a livello internazionale. Per raggiungere questo obiettivo, il Syllabus consiste di:

- Obiettivi didattici generali che descrivono i propositi del livello Specialist
- Una lista di termini (parole chiave) che gli studenti devono essere in grado di ricordare
- Una lista di parole chiave specifiche del dominio associate alla sicurezza che gli studenti devono essere in grado di ricordare
- Obiettivi di apprendimento per ogni area di conoscenza, che descrivono i risultati cognitivi di apprendimento che devono essere raggiunti
- Una descrizione dei concetti chiave, che include i riferimenti a fonti riconosciute

Il contenuto del Syllabus non è una descrizione dell'intera area di conoscenza del testing di sicurezza, ma riflette il livello di dettaglio che deve essere coperto dai corsi di formazione Security Test Engineer a livello Specialist. Si focalizza sui concetti e sulle tecniche di test che possono essere applicati a tutti i progetti software, indipendentemente dal ciclo di vita dello sviluppo software utilizzato.

0.8 Come è Organizzato questo Syllabus

Sono presenti nove capitoli con contenuti esaminabili. Nell'intestazione principale di ogni capitolo è specificato il tempo di formazione del capitolo; la tempistica non viene fornita al di sotto del livello del capitolo. Per i corsi di formazione accREDITati, il Syllabus richiede un minimo di 1290 minuti (circa 22 ore) di lezione, distribuiti nei nove capitoli come segue:

- Capitolo 1: Paradigmi di Sicurezza - 135 minuti
- Capitolo 2: Tecniche di Test di Sicurezza - 150 minuti
- Capitolo 3: Il Processo di Test di Sicurezza - 120 minuti
- Capitolo 4: Standard e Best Practice del Testing di Sicurezza - 195 minuti
- Capitolo 5: Adattare il Testing di Sicurezza al Contesto Organizzativo - 195 minuti
- Capitolo 6: Adattare il Testing di Sicurezza ai Modelli di Ciclo di Vita dello Sviluppo Software - 165 minuti
- Capitolo 7: Il Testing di Sicurezza come Parte di un Information Security Management System (ISMS) - 105 minuti
- Capitolo 8: Il Reporting dei Risultati dei Test di Sicurezza - 135 minuti
- Capitolo 9: Strumenti del Testing di Sicurezza - 90 minuti

1 Paradigmi di Sicurezza - 135 minuti (K3)

Parole Chiave

disponibilità, confidenzialità, integrità, testing di sicurezza, vulnerabilità

Parole Chiave Relative alla Sicurezza

software open source, zero trust

Obiettivi di Apprendimento per il Capitolo 1:

1.1 Livelli di Sicurezza degli Asset

STE-1.1.1 (K2) Spiegare i diversi livelli di sicurezza degli asset e il corrispondente livello di protezione

STE-1.1.2 (K2) Spiegare la relazione tra la sensibilità delle informazioni e il testing di sicurezza

1.2 Audit di Sicurezza

STE-1.2.1 (K2) Descrivere il ruolo del testing di sicurezza nel contesto degli audit di sicurezza

1.3 Il Concetto di Zero Trust

STE-1.3.1 (K2) Spiegare il concetto di zero trust

STE-1.3.2 (K3) Applicare lo zero trust nel testing di sicurezza

1.4 Software Open Source

STE-1.4.1 (K2) Esempificare il concetto di riutilizzo del software open source nello sviluppo di software e il suo impatto sul testing di sicurezza

1.1 Livelli di Sicurezza degli Asset

Un asset è qualsiasi cosa che abbia valore in un'organizzazione e che, pertanto, richiede protezione. Gli asset consentono alle organizzazioni di gestire i processi di business e prendere decisioni. Ogni asset informativo e di dati è vulnerabile e dovrebbe quindi essere protetto. Gli asset sono oggetti di sicurezza informatica (information security). Gli asset possono essere persone, informazioni, software, hardware, funzioni, processi e strutture di reputazione aziendale [Chapple 2021]. Alcuni esempi:

- asset software: sistema operativo, applicazioni e database
- asset informativi: piani di business, documentazione, invenzioni, immagini e documenti personali
- asset hardware: sistemi informatici, archivio dati e dispositivi di comunicazione dei dati
- asset fisici: strutture

1.1.1. Gli asset e il relativo livello di protezione

Il valore di un asset è determinato da tre pilastri della sicurezza delle informazioni (noti come triade CIA): confidenzialità, integrità e disponibilità [Stallings18].

La **confidenzialità (Confidentiality)** mira a impedire la divulgazione non autorizzata delle informazioni. Si verifica una perdita di confidenzialità quando le informazioni vengono divulgate a soggetti o sistemi non autorizzati [Stallings18].

L'**integrità (Integrity)** mira a impedire che i dati vengano modificati o cancellati da soggetti non autorizzati [Stallings18].

La **disponibilità (Availability)** garantisce che le informazioni siano disponibili quando necessario [Stallings18].

I tre pilastri della sicurezza delle informazioni possono essere classificati nei seguenti livelli: alto, medio e basso. Più alto è il livello di sicurezza, maggiore è il requisito del livello di protezione (misure di sicurezza) da implementare. Le misure di sicurezza comprendono funzioni e vincoli di sicurezza, sicurezza del personale e sicurezza di strutture fisiche, aree e dispositivi.

La perdita di confidenzialità, integrità o disponibilità può avere un impatto sul funzionamento dell'organizzazione, sugli asset dell'organizzazione, sulle persone, sui clienti o persino sui paesi.

La classificazione degli asset definisce i livelli di sensibilità e confidenzialità degli asset. Questo aiuta le organizzazioni a implementare controlli di sicurezza e livelli di protezione appropriati.

Se l'asset ha una sensibilità bassa, il livello di protezione potrebbe essere impostato su un livello di sicurezza basso e il testing di sicurezza potrebbe non essere necessario. Il pilastro adatto a questo caso è disponibilità.

Un esempio di sensibilità bassa può essere quando la disponibilità delle informazioni è di maggiore importanza per un gran numero di persone, come nel caso delle informazioni sul traffico.

Se l'asset ha una sensibilità media, il livello di protezione sarà impostato su un livello di sicurezza medio e il testing di sicurezza sarà necessario. Il pilastro adatto a questo caso è integrità.

Un esempio di sensibilità media può essere quando le persone hanno bisogno di informazioni accurate da fonti affidabili, come le autorità.

Se l'asset ha una sensibilità alta, il livello di protezione sarà impostato su un livello di sicurezza alto e il testing di sicurezza sarà obbligatorio. Il pilastro adatto a questo caso è confidenzialità. Un esempio di sensibilità alta può essere rappresentato dalle informazioni personali sui dipendenti.

1.1.2. Sensibilità delle informazioni e testing di sicurezza

La sensibilità delle informazioni rappresenta il grado di protezione necessario per garantire la confidenzialità, l'integrità e la disponibilità delle informazioni [NIST Glossary]. Poiché le conseguenze della violazione di informazioni sensibili possono variare da lievi a disastrose, è necessario eseguire dei test di sicurezza prima che si verifichi qualsiasi violazione.

Lo scopo del testing di sicurezza è verificare che un'implementazione protegga i dati e mantenga la funzionalità prevista. Maggiore è la protezione necessaria per un asset, maggiori sono le azioni di sicurezza richieste.

Il testing di sicurezza aiuta a identificare le debolezze e le vulnerabilità e verifica se sono stati implementati adeguati controlli di sicurezza.

Il testing di sicurezza non può garantire che un sistema o un'organizzazione siano privi di vulnerabilità. Tali misure includono l'esecuzione del testing di sicurezza per raggiungere i seguenti obiettivi:

- Valutare l'efficacia dei controlli di sicurezza esistenti
- Individuare le debolezze e le vulnerabilità
- Definire una strategia di test di sicurezza che includa test confermativi per monitorare lo stato di avanzamento di eventuali patch del software e aggiornamenti a lungo termine

Per il Security Test Engineer (STE), una valutazione dei rischi di sicurezza effettuata dal punto di vista della sensibilità delle informazioni può essere una ricca fonte di informazioni da cui pianificare e progettare i test di sicurezza. Inoltre, una valutazione dei rischi di sicurezza può essere utilizzata per prioritizzare i test di sicurezza in modo da determinare i rischi e le priorità e sottoporre quelli con i livelli di rischio più elevati a un testing più rigoroso. La valutazione del rischio è solo un'istantanea del momento attuale e può basarsi su informazioni limitate.

1.2 Audit di Sicurezza

Un audit di sicurezza è una review e un esame indipendente della sicurezza del sistema informativo di un'organizzazione, che controlla il grado di conformità a una serie di criteri prestabiliti. Gli audit hanno lo scopo di determinare l'adeguatezza dei controlli di sistema e garantire la conformità a una politica e a procedure di sicurezza prestabilite. Ma anche di rilevare violazioni nei servizi di sicurezza e raccomandare eventuali modifiche indicate come contromisure [NIST Glossary].

1.2.1. Audit di sicurezza e testing di sicurezza

Gli audit di sicurezza hanno le seguenti caratteristiche:

- Possono essere eseguiti da auditor interni o esterni.
- Si focalizzano su aspetti dei processi e dei controlli di sicurezza di un'organizzazione, quali:
 - Componenti fisici del sistema informativo e dell'ambiente in cui sono archiviate le informazioni
 - Applicazioni e software, comprese le patch di sicurezza e le configurazioni
 - Controlli dei diritti e dei privilegi degli utenti
 - Vulnerabilità della rete, comprese le valutazioni della connessione tra diversi punti all'interno e all'esterno della rete dell'organizzazione
 - Modalità di raccolta, condivisione e archiviazione delle informazioni da parte dei dipendenti
 - Meccanismi di rilevamento delle intrusioni
 - Piani di risposta in caso di violazione
- Sono un tipo di testing statico (vedere la sezione 2.1.2) che prevede l'esame manuale dei prodotti di lavoro o review automatizzate con strumenti di audit di sicurezza.
- Esaminano aspetti delle politiche, delle procedure e dei controlli di sicurezza di un'organizzazione che sono difficili da testare dinamicamente.
- Verificano l'efficacia dei controlli di sicurezza installati e identificano dove i criteri stabiliti dall'organizzazione sono stati raggiunti o meno in un determinato momento.
- Non garantiscono che tutte le vulnerabilità vengano individuate, ma assicurano che le aree problematiche vengano identificate e indicano dove sono necessarie azioni correttive.

Gli audit di sicurezza dovrebbero far parte della routine regolare. Possono essere effettuati come:

- Valutazioni una tantum per circostanze speciali, ad esempio quando un'organizzazione introduce una nuova piattaforma software o una nuova integrazione. Dovrebbero essere eseguiti un test e un audit di sicurezza per individuare eventuali nuovi rischi e/o difetti.
- Audit schedulati regolarmente per verificare che i processi e le procedure di sicurezza siano seguiti e che siano adeguati al clima e alle esigenze di business attuali.

In alcuni approcci dell'audit di sicurezza, il testing di sicurezza viene eseguito come parte del processo di audit per determinare se i controlli di sicurezza sono effettivamente in atto e funzionano in modo efficace. Tuttavia, l'ambito di un audit di sicurezza è molto più ampio rispetto al testing di sicurezza.

Il testing di sicurezza e gli audit lavorano insieme. Gli audit identificano le carenze e le aree importanti da testare. Il testing di sicurezza è il mezzo attraverso il quale si dimostra o si smentisce che i controlli di sicurezza sono effettivamente in atto e funzionano in modo efficace.

1.3 Il Concetto di Zero Trust

1.3.1. Che cos'è lo Zero Trust?

Zero Trust è un modello di sicurezza creato da una serie di concetti e idee progettati per ridurre al minimo l'incertezza nell'applicazione di un accesso accurato e con privilegi minimi per ogni richiesta. Si basa sul principio di controlli di accesso rigorosi e di non fidarsi di nessuno a prescindere, anche se tutti sono già all'interno del perimetro della rete.

Il modello zero trust incarna il principio "non fidarti di nulla, verifica tutto, il volto di una rete è considerato compromesso".

L'aumento dell'uso dei servizi online ha portato a un corrispondente aumento delle vulnerabilità e degli attacchi. Le informazioni sono spesso distribuite tra i fornitori di servizi cloud, il che ha reso i concetti di sicurezza basati sul perimetro meno efficaci nel garantire la sicurezza di organizzazioni, datori di lavoro, utenti e clienti. La sicurezza tradizionale basata sul perimetro si fida di chiunque e di qualsiasi cosa all'interno della rete. Il problema di questo approccio è che, una volta che un attacker ottiene l'accesso alla rete, ha accesso a tutti gli asset al suo interno.

Utilizzando il modello zero trust, i sistemi e i servizi di informazioni operano partendo dal presupposto che le loro reti siano già compromesse e che la rete non abbia uno spazio affidabile. La prospettiva zero trust porta alla pratica che sposta le difese di sicurezza dai perimetri statici basati sulla rete per focalizzarsi su utenti, asset e risorse [NIST Glossary].

Vantaggi dello zero trust [Cloudflare]:

- Riduce la superficie di attacco di un'organizzazione
- Riduce al minimo i danni di un attacco limitando la violazione a un'area ristretta e abbassa i costi di ripristino
- Riduce l'impatto del furto delle credenziali degli utenti e degli attacchi di phishing richiedendo più fattori di autenticazione

1.3.2. Zero Trust nel testing di sicurezza

Le reti odierne sono prive di perimetri e stanno migrando da deployment piatti ad ambienti dinamici, distribuiti e ibridi. Le organizzazioni si adattano alla crescente complessità dei loro ambienti, che comprendono luoghi di lavoro ibridi, lavoratori remoti, interazioni con altre organizzazioni e fornitori, e devono proteggere persone, dispositivi, applicazioni, reti e dati ovunque si trovino. Il modello zero trust, basato sul presupposto "non fidarti di nulla, verifica tutto", richiede un completo cambiamento di paradigma nella sicurezza IT, in cui persone, dispositivi, applicazioni, reti e dati devono essere sottoposti a rigorose verifiche prima di poter accedere a un'applicazione o a una risorsa richiesta.

I principi dello zero trust [Micro22] [Cloudflare] includono:

- Monitoraggio e verifica continui: verificare sempre l'accesso a tutte le risorse. I login e le connessioni scadono periodicamente, costringendo gli utenti e i dispositivi a essere continuamente riverificati.

- Principio dell'"accesso con privilegi minimi": concedere agli utenti solo l'accesso necessario.
- Autenticazione a più fattori: richiede più di un elemento di prova per autenticare un utente. Il solo inserimento di una password non è sufficiente per consentire l'accesso.
- Monitoraggio della sicurezza degli endpoint dei dispositivi che accedono ai dati, come laptop, workstation, tablet, dispositivi mobile: ogni endpoint remoto può essere il punto di ingresso per un attacco.
- Microsegmentazione: dividere i perimetri di sicurezza in piccole zone per ottenere un accesso separato per parti separate della rete.
- Nessun movimento laterale all'interno delle reti senza una validazione continua: l'accesso zero trust è segmentato e deve essere ristabilito periodicamente. Un attacker non può spostarsi in altri microsegmenti all'interno della rete. Una volta rilevata la presenza dell'attacker, il segmento o l'account utente compromesso può essere messo in quarantena e isolato da ulteriori accessi.
- Protezione dei dati nei file e nei contenuti: crittografia e restrizioni di accesso basate sulle politiche dell'organizzazione.

Il modello zero trust influisce sul modo in cui dovrebbe essere gestito il testing di sicurezza. Ciò significa focalizzare i test case sui meccanismi di sicurezza zero trust. Il testing di sicurezza contribuisce a identificare le seguenti possibili debolezze:

- Rete: microsegmenti zero trust, riduzione dei danni ed evidenziazione delle violazioni
 - Testare il traffico tra segmenti con automazione e microsegmenti definiti dall'utente.
- Dati: lo zero trust richiede che i dati siano trasmessi in modo sicuro e crittografato.
 - Testare gli endpoint che espongono o archiviano dati utilizzando metodi non crittografati.
- Identità: persone, dispositivi e processi possono fare solo ciò che sono autorizzati a fare.
 - Testare se persone, dispositivi e/o processi dispongono di autorizzazioni di accesso superiori a quelle necessarie, che potrebbero compromettere gli asset della rete.
 - Verificare se utenti non autorizzati possono accedere a segmenti delle risorse di rete.
- I dispositivi dovrebbero eseguire solo software sicuro ed essere monitorati e gestiti centralmente.
 - Testare se il dispositivo è protetto con software di sicurezza ed eseguire il testing di penetrazione.
- Limitazione del raggio d'azione:
 - Vengono eseguiti test su base regolare per individuare le debolezze e mitigare l'impatto del rischio, al fine di limitare il raggio d'azione in caso di violazione esterna o interna.

1.4 Software Open Source (OSS)

L'OSS (Open-Source Software) è sviluppato e mantenuto tramite collaborazione aperta ed è disponibile, in genere gratuitamente, affinché chiunque possa utilizzarlo, modificarlo e ridistribuirlo a proprio piacimento.

1.4.1. Il concetto di OSS e il suo impatto sul testing di sicurezza

L'OSS è basato su codice open source che può essere liberamente utilizzato, modificato e condiviso da chiunque. L'OSS è spesso distribuito con licenze conformi alla definizione di open source fornita dall'Open-Source Initiative.

Il vantaggio principale dell'utilizzo dell'OSS è la trasparenza del codice e il presupposto che molti sviluppatori volontari abbiano controllato il codice per individuare e risolvere i difetti. Si presume che questa apertura porterà un numero maggiore di persone a partecipare all'identificazione rapida delle vulnerabilità e alla correzione dei difetti.

Tuttavia, il fatto che queste applicazioni, librerie e altri oggetti riutilizzabili siano disponibili in tutto il mondo rappresenta una sfida, poiché chiunque può aggiornare il codice e potenzialmente introdurre vulnerabilità e vettori di attacco [Shacklett]. L'OSS ha generalmente più vettori di attacco rispetto al software chiuso (proprietario), perché chiunque può aggiungere backdoor intenzionali, propagare vulnerabilità attraverso il riutilizzo e sfruttare vulnerabilità e errori umani divulgati pubblicamente. Una volta che una vulnerabilità è stata pubblicata, gli utenti di quella versione del software sono a rischio di attacco.

L'utilizzo di OSS significa che ogni exploit pubblicato per un componente specifico potrebbe potenzialmente influenzare migliaia di sistemi. Quando il codice sorgente è disponibile in versioni eseguibili, l'osservazione, il reverse engineering, la review del codice, il disassemblaggio e il testing esplorativo possono essere in grado di individuare vulnerabilità [Stallings18].

Il STE svolge le seguenti attività:

- Identificazione delle vulnerabilità open source
- Esecuzione di review del codice come parte dello shift left che mira al rilevamento dei difetti nelle prime fasi del processo di sviluppo

Quando si tratta di testare l'OSS per la sicurezza delle applicazioni, lo STE ragiona come un attacker. I test case catturano il comportamento di un'applicazione in diversi scenari di utilizzo e abuso e consentono agli sviluppatori di mettere in atto adeguate misure di mitigazione dei rischi.

Le review del codice vengono eseguite dagli sviluppatori e dai STE sia dal lato del produttore che da quello del consumatore per identificare il codice non sicuro.

L'Open Web Application Security Project (OWASP) ha reso disponibili strumenti automatizzati di rilevamento delle vulnerabilità gratuiti per i progetti open source [OWASP Top 10]. Il NIST offre una guida per la sicurezza dell'OSS [NIST SP 800-161].

2 Tecniche di Test di Sicurezza - 150 minuti (K3)

Parole Chiave

testing distruttivo, fuzz testing, scanning dei malware, scanning delle vulnerabilità

Parole Chiave Relative alla Sicurezza

autenticazione, autorizzazione, crittografia, firewall

Obiettivi di Apprendimento per il Capitolo 2:

2.1 Applicare i Tipi di Test di Sicurezza in Base al Contesto del Test

- STE-2.1.1 (K2) Fornire esempi di tipi di test di sicurezza in base a un contesto di sicurezza black-box, white-box e grey-box
- STE-2.1.2 (K2) Fornire esempi di tipi di test di sicurezza in base al testing di sicurezza statico o al testing di sicurezza dinamico

2.2 Applicare i Tipi di Test di Sicurezza in Base al Contesto Tecnico e al Progetto

- STE-2.2.1 (K3) Applicare test case di sicurezza, basati su un determinato approccio del test di sicurezza, insieme ai rischi di sicurezza funzionali e strutturali identificati
- STE-2.2.2 (K2) Descrivere come testare la riconciliazione e la ricertificazione delle identità e delle autorizzazioni
- STE-2.2.3 (K2) Descrivere come testare il controllo della gestione delle identità e degli accessi
- STE-2.2.4 (K2) Descrivere come testare il controllo della protezione dei dati
- STE-2.2.5 (K2) Descrivere come testare le tecnologie di protezione

2.1 Applicare i Tipi di Test di Sicurezza in Base al Contesto del Test

2.1.1. Testing black-box, testing white-box e testing grey-box

I tipi di test per la base di test sono classificati come testing black-box, testing grey-box e testing white-box [ISTQB FL]. Il Glossario ISTQB [ISTQB Glossary] definisce il testing black-box e il testing white-box.

Il testing di sicurezza grey-box è definito in [NIST] come "una metodologia di test che presuppone una certa conoscenza della struttura interna e dei dettagli di implementazione dell'oggetto di valutazione". Il STE ha accesso ad alcune informazioni sul sistema sottoposto a test (SUT) (ad esempio, un sottoinsieme di una mappa di indirizzamento di rete, un sottoinsieme della documentazione dell'architettura, un accesso utente e un accesso a una macchina interna).

Il STE ha accesso a tutte le informazioni necessarie sul SUT durante il testing di sicurezza:

- L'architettura del SUT
- Il codice sorgente
- I flussi di dati nel SUT
- Progettazione della rete e struttura delle zone
- Requisiti relativi alle password
- Regole del firewall
- Autenticazione
- Informazioni sull'archiviazione e sulla gestione dei log

La scelta delle tecniche di test si basa sugli obiettivi dell'approccio del test di sicurezza e sulla disponibilità del codice. Il STE dovrebbe decidere il livello di profondità dei test e se focalizzarsi sulle minacce esterne o interne.

2.1.2. Testing di sicurezza statico e dinamico

Sia il testing statico che il testing dinamico vengono utilizzati nel testing di sicurezza per proteggere il SUT durante il suo intero ciclo di vita.

Testing di sicurezza statico

Dal punto di vista del STE, tra i prodotti di lavoro che possono essere sottoposti a review durante il testing di sicurezza statico vi sono:

- Documentazione relativa all'analisi dei rischi di sicurezza
- Requisiti di sicurezza

Quando si cercano lacune nei requisiti, è necessario prendere in considerazione i seguenti meccanismi di sicurezza:

- Gestione degli utenti
- Autenticazione
- Autorizzazione
- Confidenzialità
- Integrità
- Responsabilità
- Gestione delle sessioni
- Sicurezza dei trasporti
- Segregazione del sistema a più livelli
- Conformità legislativa e normativa, inclusi gli standard relativi alla privacy, governativi e industriali
- Documentazione tecnica relativa alla sicurezza dell'architettura
- Codice sorgente
- Impostazione della configurazione e impostazione dell'infrastruttura/operativa

Testing di sicurezza dinamico

Rispetto al testing di sicurezza statico, lo scopo del testing di sicurezza dinamico è verificare che il SUT implementi e utilizzi correttamente le funzioni di sicurezza come richiesto o specificato e che tali funzioni di sicurezza non possano essere aggirate durante l'utilizzo del sistema o dell'applicazione.

Il testing di sicurezza dinamico può essere eseguito:

- Utilizzando il testing black-box per valutare le funzioni di sicurezza verificando che i risultati dei test siano quelli attesi quando vengono inviati input particolari
- Utilizzando il testing di penetrazione: cerca di individuare vulnerabilità che potrebbero essere sfruttate

È consigliato eseguire il testing di sicurezza dinamico nel modo seguente, se possibile, nei seguenti ambienti:

- L'ambiente di sviluppo
Questa è la prima occasione per il STE di eseguire test di sicurezza. In questo ambiente, il STE verifica che l'implementazione della sicurezza sia conforme ai requisiti di sicurezza (ad esempio, lo sviluppo corretto di una password controllando la dimensione minima, la dimensione massima e i tipi di caratteri obbligatori).
- L'ambiente di test e l'ambiente di preproduzione
Spesso noto come ambiente di staging o di test di accettazione, questo ambiente dovrebbe essere il più simile possibile all'ambiente di produzione e dovrebbe contenere tutte le misure di sicurezza previste che saranno applicate nell'ambiente di produzione.
- L'ambiente di produzione
Questo è l'ambiente più critico. Il STE deve fare attenzione a non disabilitare il SUT e dovrebbe condurre audit di sicurezza per mantenere il sistema sicuro. L'obiettivo del testing di sicurezza

sull'ambiente di produzione è verificare che le vulnerabilità appena scoperte e quelle già note siano state risolte.

L'utilizzo di uno strumento dynamic application security test (DAST) consente di rilevare le condizioni che potrebbero portare a vulnerabilità (vedere il capitolo 9). Il DAST può essere incluso in una pipeline di continuous integration/continuous delivery (CI/CD) nelle seguenti fasi:

- Durante il test dopo una build, funziona come uno scanner di sicurezza dinamico per rilevare i difetti di sicurezza
- Durante la produzione, funziona come uno scanner di vulnerabilità

2.2 Applicare il Testing di Sicurezza

La progettazione dei test di sicurezza può basarsi sulle seguenti fonti:

- Normative, standard e leggi (ad esempio, una nuova normativa che richiede che le automobili siano sottoposte a un test di sicurezza prima dell'approvazione)
- Un'analisi del rischio completa
- Modelli di minaccia disponibili
- Una categorizzazione ad hoc dei rischi di sicurezza (vedere [ISTQB_ATTA_SYL])
- Un approccio del test di sicurezza
- Requisiti di sicurezza delle funzioni e dei meccanismi di sicurezza
- Sistemi e prodotti nel ciclo di vita dello sviluppo software (SDLC)
- L'esperienza e le competenze di testing del STE
- Incidenti precedenti in cui la sicurezza è stata (quasi) violata
- Una guida di riferimento per i test, come la [OWASP Test Guide]

Di seguito sono riportati gli attributi di un test di sicurezza che dovrebbero essere presi in considerazione durante la progettazione dei test di sicurezza:

- Normative o leggi richieste (obbligatorie)
- Rischi di sicurezza identificati e modelli di minaccia prioritizzati dall'approccio del test di sicurezza
- Tracciato in base a requisiti di sicurezza definiti
- Definito in base ai tester previsti (ad esempio, sviluppatori, tester funzionali e STE)
- Definito in base ai profili dei difetti di sicurezza e alle vulnerabilità conosciute
- Progettato per essere automatizzato, se applicabile
- Testing distruttivi o testing non distruttivo
- Intrusivo o non intrusivo (ad esempio, l'obiettivo è quello di compromettere un sistema dall'interno o di metterlo fuori uso tramite un Distributed Denial of Service)

Il flusso di lavoro di base della progettazione dei test di sicurezza è:

1. Analisi dell'approccio del test di sicurezza (a livello di progetto)
2. Analisi dei rischi di sicurezza, dei modelli di minaccia e dei requisiti (a livello di progetto)

3. Applicazione delle tecniche di test di sicurezza

Nella maggior parte dei casi, una progettazione efficiente dei test di sicurezza si basa su una combinazione delle fonti sopra indicate. A seconda del tipo di progetto, è importante garantire che venga eseguito un test di sicurezza in ogni fase del SDLC del SUT.

2.2.1. Affrontare i rischi di sicurezza nella progettazione dei test

Un principio fondamentale è che il STE dovrebbe essere in grado di creare e implementare test case di sicurezza basati su qualsiasi rischio di sicurezza, requisito di sicurezza, minaccia ed esperienza identificati.

Il testing di sicurezza può basarsi su rischi di sicurezza esterni nell'ambiente di produzione (minacce a un sistema o a un prodotto), su un approccio del testing di sicurezza e su altre fonti, come i modelli di minaccia. I rischi di sicurezza possono anche essere considerati di natura funzionale e strutturale (cioè rischi dovuti alla mancanza di security by design).

Durante la progettazione dei test case di sicurezza, il STE deve identificare se un test è distruttivo o non distruttivo. Se un test viene identificato come distruttivo, deve garantire che non abbia alcun impatto negativo su altre attività di test, ambienti o sul business.

Di seguito vengono descritti i rischi di sicurezza e le vulnerabilità comuni a livello funzionale e strutturale, insieme alle tecniche di test di sicurezza appropriate.

Controlli di sicurezza, rischi o vulnerabilità funzionali

I test sono progettati per verificare e validare che i controlli siano in atto, che funzionino correttamente e che siano efficaci nel rilevare e prevenire azioni non autorizzate.

Le tecniche di test di sicurezza si basano sui requisiti relativi ai controlli di sicurezza funzionali e ai controlli di accesso funzionali.

Controlli di accesso, rischi o vulnerabilità strutturali

I test per questi controlli si basano sulle modalità di definizione dei diritti degli utenti per l'accesso ai dati, l'accesso funzionale e i livelli di privacy. I controlli di accesso strutturali in genere vengono applicati da un amministratore di sistema, un amministratore della sicurezza o un amministratore di database. In alcuni casi, i diritti di accesso sono un'opzione di configurazione in un'applicazione. In altri casi, i diritti di accesso vengono applicati a livello di infrastruttura di sistema.

Rischi o vulnerabilità di accesso al sistema operativo

Una volta ottenuto l'accesso al sistema operativo, un attacker può controllare i processi, i dati e l'accesso alla rete, il che può consentire l'inserimento di malware.

Rischi o vulnerabilità della piattaforma

Ogni piattaforma ha una propria serie di vulnerabilità di sicurezza.

Le tecniche di test di sicurezza si basano sull'esperienza del STE (ad esempio, nella gestione delle vulnerabilità) e sul testing delle procedure di sicurezza (ad esempio, il testing del mantenimento delle condizioni di sicurezza).

Minacce esterne e interne

Alcune minacce, come lo sfruttamento delle vulnerabilità delle applicazioni o dei linguaggi di programmazione, possono essere rilevate, testate e il loro impatto può essere limitato. Le minacce interne sono eseguite dai dipendenti interni. Le minacce esterne sono eseguite da persone esterne (ad esempio, attacker).

Le tecniche di test di sicurezza si basano sul testing esplorativo (ad esempio, per individuare potenziali obiettivi e punti di iniezione/vettori di attacco utili) e sull'esperienza del STE.

2.2.2. Testing di riconciliazione e testing di ricertificazione

Comprendere le preoccupazioni relative alla gestione delle identità e degli accessi

I servizi di business forniti dalle organizzazioni stanno diventando sempre più complessi. Sono implementati come un sistema di sistemi e ospitati in ambienti ibridi costituiti da elementi interni, forniti da partner, forniti da clienti e nel cloud. In questo ambiente distribuito, la gestione della sicurezza degli account utente e dei diritti dell'utente è fondamentale. Ad esempio:

- L'utente deve disporre dei privilegi adeguati e non di più
- I diritti devono essere revocati dopo che un dipendente ha lasciato l'organizzazione
- La gestione dei diritti deve essere conforme alle normative, ad esempio il General Data Protection Regulation (GDPR)

La gestione delle identità e degli accessi (Identity and Access Management o IAM) è una disciplina che consente di gestire e mantenere gli account e i diritti degli utenti definendo chi (identità) è autorizzato ad accedere a cosa (ruolo) per una risorsa specifica. L'IAM elenca due sottoprocessi direttamente correlati al testing di sicurezza:

- Riconciliazione:
Il confronto e l'aggiornamento dell'accesso, dei diritti e degli account privilegiati degli utenti tramite richieste di modifica e catene di approvazione a un database di gestione delle identità ufficiale e affidabile.
- Ricertificazione:
Review periodiche degli account e dei relativi diritti e privilegi per verificare se sono ancora necessari.

La riconciliazione è necessaria per garantire che tutti gli accessi alle applicazioni siano sincronizzati con la stessa "fonte affidabile". I livelli all'interno del processo di riconciliazione sono:

- Completo: confronto di tutti gli account e degli attributi di accesso degli utenti con il sistema di gestione delle identità e degli accessi, allo scopo di identificare eventuali differenze
- Incrementale: confronto solo delle modifiche agli account e ai diritti creati, aggiornati o eliminati

- Automatico: vengono utilizzate applicazioni in grado di schedulare confronti automatici di eventuali modifiche rilevanti per la sicurezza

La ricertificazione consiste nell'auditing di un account utente e dei privilegi di accesso per determinare se sono ancora giustificati, coerenti con le politiche interne dell'organizzazione e conformi alle normative. Ciò comporta l'esecuzione di un audit continuo per garantire che gli utenti abbiano accesso solo a ciò di cui hanno bisogno e per cui hanno l'autorizzazione. La valutazione potrebbe essere:

- Manuale
 - Estrarre e raccogliere le informazioni sugli account
 - Presentare le informazioni
 - Review svolte da parte dei manager
- Automatizzata
 - I messaggi vengono inviati ai manager per richiedere le richieste di ricertificazione.
 - Questo offre il vantaggio di poter pianificare audit su base regolare.

All'interno di una grande organizzazione, con sistemi ospitati su una vasta gamma di ambienti, l'IAM diventa complesso a causa della necessità di gestire diverse applicazioni, ognuna con accessi da concedere e revocare in base ai movimenti degli utenti (ad esempio, arrivi, partenze e trasferimenti). In alcune organizzazioni, gli account utente e i privilegi vengono gestiti al di fuori di un processo IAM formale per risparmiare tempo. Si tratta di una questione critica dal punto di vista della sicurezza, poiché gli account orfani e inutilizzati possono essere sfruttati da un attacker.

Come eseguire test di riconciliazione e test di ricertificazione per i meccanismi di identità e autorizzazioni

È necessario eseguire test di riconciliazione e test di ricertificazione per evitare incongruenze negli account utente in tutte le applicazioni a cui l'utente ha accesso. Questo include aspetti quali credenziali di login e privilegi.

Si applicano le seguenti condizioni di test:

- Gestione dei nuovi account
- Modifica delle credenziali e dei privilegi dell'account
- Trasferimento di un account, compresa la rimozione dell'accesso alle applicazioni e l'aggiunta dell'accesso a nuove applicazioni
- Review di tutti gli account

Gli obiettivi del test possono essere:

- Tentativo di passare, modificare o accedere a un altro ruolo
- Esecuzione di review della granularità dei ruoli e delle esigenze alla base delle autorizzazioni concesse
- Verifica che i requisiti di identità per la registrazione degli utenti siano in linea con i requisiti di business e di sicurezza.

Le tecniche di test di sicurezza per la riconciliazione e la ricertificazione includono:

- Eseguire review della documentazione relativa ai processi di riconciliazione e ricertificazione
- Verificare se le registrazioni sono state controllate da una persona prima dell'attribuzione o se vengono concesse automaticamente quando vengono soddisfatti determinati criteri
- Verificare, controllare e autorizzare le richieste di disassegnazione
- Verificare l'efficacia delle modifiche agli account
- Eseguire il fuzz testing sui possibili ruoli per assicurarsi che il sistema rifiuti i ruoli fuzzati
- Eseguire review delle autorizzazioni dei ruoli dopo aver raccolto tutte le modifiche applicate

Notare che la [OWASP Test Guide] fornisce un elenco di obiettivi di test di sicurezza e tecniche di test relative al testing dell'IAM.

2.2.3. Testing di identificazione, autenticazione e autorizzazione

Comprendere la relazione tra autenticazione e autorizzazione

Gli asset sensibili di un'organizzazione devono essere protetti e devono essere accessibili solo a persone autorizzate che sono state precedentemente autenticate. L'identificazione è il primo passo per ottenere l'accesso a una risorsa. È il processo di affermazione di un'identità.

L'autenticazione si basa sulla verifica di un identificativo utente e di un token per rispondere alle domande:

- Chi è l'utente? (identificatore utente)
- L'utente è davvero chi dice di essere? (token, come una password o un certificato)

A seconda del livello di protezione da garantire contro gli attacchi volti a dirottare un'autenticazione o a rubare il token, possono essere utilizzate diverse implementazioni dei meccanismi di autenticazione.

L'autorizzazione viene utilizzata per i seguenti scopi:

- Per verificare se l'utente autenticato ha i diritti per eseguire un'azione
- Per determinare quale livello di accesso dovrebbe essere consentito alle risorse di sistema

Esiste una stretta relazione tra autenticazione e autorizzazione basata sul principio che un utente non autenticato non deve avere privilegi o deve avere privilegi limitati sul sistema.

L'autenticazione, l'autorizzazione e l'accounting del soggetto danno origine all'acronimo AAA, che è un framework che aiuta a controllare e tracciare l'accesso all'interno di una rete di computer. L'accounting è la terza "A" nel framework AAA che considera la registrazione, il tracciamento e le attività di un utente.

Come testare i meccanismi di autenticazione e autorizzazione

Le tecniche di test di sicurezza (test di penetrazione) per i meccanismi di autenticazione potrebbero includere:

- Testing delle credenziali predefinite
- Testing delle politiche relative alle password deboli

- Ricerca di informazioni trapelate utilizzando l'open source intelligence (OSINT)
- Test di forza bruta utilizzando dizionari e tabelle arcobaleno (ovvero tabelle precalcolate di hash delle password invertiti) per eseguire attacchi che tentano di scoprire le password degli utenti. I primi passi potrebbero essere, ad esempio, provare "123456", "111111", la data di nascita o il nome di un animale domestico.

Le tecniche di test di sicurezza (test di penetrazione) per i meccanismi di autorizzazione potrebbero includere:

- Sfruttare la mancanza di filtraggio degli input, ad esempio iniettando richieste SQL da autenticare senza alcun login/password conosciuta o causando un overflow del buffer di input per ottenere l'accesso amministrativo a una sessione shell
- Inserire un Uniform Resource Identifier non autorizzato (ad esempio, ../.. in un account File Transfer Protocol) o un Uniform Resource Locator (URL) (ad esempio, indirizzo del sito/admin) per tentare di ottenere l'accesso a dati sensibili
- Testare le violazioni dell'escalation dei privilegi orizzontali e verticali

Notare che la [OWASP Test Guide] fornisce un elenco di tecniche di test di sicurezza per testare l'autenticazione e l'autorizzazione.

2.2.4. Crittografia (Encryption)

Comprendere la crittografia

È possibile utilizzare un meccanismo di crittografia per evitare la divulgazione di dati sensibili, anche se questi sono accessibili quando vengono archiviati da qualche parte o scambiati tra un client e un server. La crittografia è un processo di codifica dei dati (ad esempio, testo in chiaro) in dati cifrati (ad esempio, testo cifrato), utilizzando un algoritmo crittografico e chiavi segrete. Il segreto è condiviso e conosciuto solo dagli utenti autorizzati. L'obiettivo è utilizzare una crittografia sufficientemente forte da impedire a un attacker, che potrebbe essere riuscito a rubare i dati crittografati, di recuperare il testo in chiaro. L'uso di algoritmi crittografici aiuta a garantire la confidenzialità e l'integrità degli asset sensibili e a impedire che vengano manipolati.

I tipi di protocolli crittografici principali e tipicamente utilizzati sono:

- Crittografia simmetrica: basata sull'uso di una chiave segreta condivisa
- Crittografia asimmetrica: basata sull'uso di chiavi private, pubbliche e certificati, gestiti tramite un'infrastruttura a chiave pubblica

Come testare i meccanismi di crittografia comuni

Alcuni meccanismi crittografici sono noti per essere deboli, soprattutto a causa della lunghezza ridotta delle chiavi segrete. Altri meccanismi potrebbero essere vulnerabili perché non sono implementati utilizzando le best practice o perché incorporano difetti di codifica (ad esempio, overflow del buffer).

I test dei meccanismi di crittografia dovrebbero includere:

- Test di conformità (ad esempio, test basati sui requisiti di sicurezza) delle implementazioni dei meccanismi di crittografia
- Test per vulnerabilità "di progettazione"
- Test per vulnerabilità "di costruzione"
- Test per vulnerabilità "di configurazione"

Notare che la [OWASP Test Guide] fornisce un elenco di test di sicurezza per verificare le implementazioni crittografiche deboli.

2.2.5. Testing delle tecnologie di protezione

I STE devono comprendere le sfumature delle diverse linee di difesa in modo da poter progettare test appropriati per verificarne e validarne l'efficacia.

Come testare il system hardening

Il testing dell'efficacia del system hardening può essere effettuato in vari modi. Il system hardening limita l'accesso al sistema ai ruoli appropriati, apre solo i servizi necessari e monitora gli aggiornamenti delle applicazioni. Pertanto, per testare l'efficacia del system hardening, i test dovrebbero essere progettati per rilevare se le misure del system hardening funzionano, sono applicate nei posti giusti e nei modi corretti. È anche importante testare le protezioni del system hardening che sono troppo restrittive e potrebbero essere eccessive rispetto ai rischi di sicurezza.

Alcuni test di system hardening possono basarsi su una review o un audit, mentre altri possono basarsi sulla capacità di determinati gruppi di utenti di eseguire determinate azioni o di accedere a determinati dati.

Come testare i firewall

A causa del numero di protocolli, delle loro diverse opzioni e della complessità delle reti da proteggere, è difficile configurare un firewall in modo efficiente e coerente. I test di efficacia dei firewall dovrebbero includere:

- Esecuzione di un audit per verificare la configurazione del firewall
- Scansione delle porte per verificare se la politica di sicurezza è stata implementata correttamente
- Utilizzo di pacchetti di rete malformati e fuzz testing di rete per sfruttare comportamenti imprevisti
- Attacchi di frammentazione per aggirare le funzionalità di filtraggio con l'obiettivo di eseguire un attacco dietro il firewall
- Prendere di mira il firewall delle applicazioni web mediante la codifica e la compressione dei dati o il loro offuscamento per nascondere le informazioni dannose che rappresentano l'attacco. I criteri di valutazione del firewall delle applicazioni web (web application firewall evaluation criteria) [WAFEC] possono essere utilizzati per testare l'efficacia di un firewall delle applicazioni web.

Come eseguire il rilevamento delle intrusioni

Il rilevamento scenario-based si basa su uno scenario conosciuto o "firma". È facile da aggirare perché vengono rilevati solo gli attacchi conosciuti. I test potrebbero includere le seguenti tecniche di evasione:

- Codifica dei caratteri o modifica dei dati (ad esempio, aggiunta di spazi bianchi e indicatori di fine riga)
- Frammentazione dell'Internet protocol (IP), segmentazione del transmission control protocol (TCP)
- Crittografia o offuscamento
- Codifica dell'URL

Il rilevamento behavior-based si basa su un modello del comportamento del sistema e genera un gran numero di risultati falsi positivi e falsi negativi. Un risultato falso negativo è qualsiasi avviso di sicurezza che avrebbe dovuto essere segnalato ma non lo è stato. I risultati falsi negativi possono verificarsi quando viene sviluppato un nuovo attacco di cui un intrusion detection system (IDS) non è a conoscenza, o forse una regola potrebbe essere scritta in modo tale da rilevare alcuni attacchi ma tralasciare quelli non specificati nel modello.

È necessario mantenere l'accuratezza di questo metodo di rilevamento. È possibile che un attacker si discosti dal normale comportamento dell'IDS, il che comporta una nuova specifica per il comportamento intrusivo. I test complementari dovrebbero utilizzare il traffico dannoso per aggiungere nuove specifiche intrusive da considerare come traffico autorizzato.

Come eseguire la scansione alla ricerca di malware

Gli sviluppatori di malware utilizzano diverse tecniche per proteggere il loro codice dal reverse engineering e dal rilevamento da parte dei software anti-malware. Alcune di queste tecniche includono:

- Sfruttamento delle funzioni della libreria di sistema utilizzate dall'anti-malware
- Oscuramento delle stringhe per impedire la comprensione del comportamento del codice dannoso
- Permutazione del codice
- Inserimento di codice inutilizzato
- Caricamento dinamico di funzioni e librerie (ad esempio, per limitare l'analisi del codice dannoso)
- Aggiornamento automatico delle applicazioni

Dal punto di vista del testing di adeguatezza funzionale, gli strumenti anti-malware basati su firme potrebbero essere utilizzati per testare l'efficacia dell'anti-malware senza sviluppare vere e proprie parti di codice dannoso. Altri tipi di file dannosi devono essere testati in base al tipo di applicazioni.

Il testing dell'anti-malware behavior-based è difficile perché non esiste una chiara comprensione e definizione di cosa sia un comportamento dannoso. Le idee per il testing possono trarre vantaggio dalle tecniche utilizzate dagli sviluppatori di malware:

- File di esecuzione non firmati che tentano di utilizzare chiamate di sistema per apportare modifiche al sistema
- Tentativi di avviare processi insoliti con diritti concessi
- Tentativi di copiare file di esecuzione in posizioni non autorizzate

- Tentativi di chiamare API di sistema insolite

Una considerazione importante quando si implementa un nuovo anti-malware (che sia basato su firma o su comportamento) o si aggiorna un anti-malware esistente è di testare l'implementazione su una piattaforma rappresentativa prima di distribuirla all'intera organizzazione.

Testing dell'offuscamento dei dati

È necessario un rigoroso controllo della configurazione tra i dati offuscati e le chiavi utilizzate per l'offuscamento per garantire che vengano utilizzate le versioni corrette delle chiavi. In caso contrario, i dati non saranno comprensibili per l'uso.

Poiché alcuni test potrebbero coinvolgere dati privati, è possibile ricorrere all'offuscamento dei dati a fini di testing, per rendere anonimi i dati di produzione utilizzati in un ambiente di test di sistema. I dati sensibili, come le informazioni sugli utenti utilizzate da un sistema di informazione sanitaria, non devono essere divulgati ai tester. I test potrebbero includere attacchi di forza bruta o attacchi con dizionario per tentare di ottenere dati in chiaro da dati offuscati.

I test per verificare l'offuscamento del codice potrebbero includere:

- Reverse engineering del codice
- Attacchi di forza bruta, poiché alcuni meccanismi di offuscamento sono vulnerabili

3 Il Processo di Test di Sicurezza - 120 minuti (K3)

Parole Chiave

rischio, ambiente di test, test plan, processo di test

Parole Chiave Relative alla Sicurezza

nessuna

Obiettivi di Apprendimento per il Capitolo 3:

3.1 Il Processo di Test di Sicurezza

- STE-3.1.1 (K2) Spiegare le diverse attività, compiti e responsabilità all'interno di un processo di test di sicurezza
- STE-3.1.2 (K2) Comprendere gli elementi chiave e le caratteristiche di un ambiente di test di sicurezza efficace

3.2 Progettare i Test di Sicurezza per i Livelli di Test

- STE-3.2.1 (K2) Fornire esempi di test di sicurezza a livello di test di componente sulla base di una determinata base di codice
- STE-3.2.2 (K2) Fornire esempi di test di sicurezza a livello di integrazione dei componenti sulla base di una determinata specifica di progettazione
- STE-3.2.3 (K3) Implementare un test di sicurezza end-to-end che validi uno o più requisiti di sicurezza relativi a uno o più processi di business

3.1 Il Processo di Test di Sicurezza

Il testing di sicurezza è un processo all'interno del SDLC. Il processo di test di sicurezza è dedicato all'ambito della sicurezza e deve essere allineato al processo di sviluppo in modo che le attività di test appropriate vengano eseguite quando necessario.

I rischi e le esigenze di sicurezza di ogni organizzazione sono unici a causa della natura dell'organizzazione, degli ambienti tecnici, del SDLC e dei rischi di business. Pertanto, il processo di test di sicurezza deve essere definito e implementato nel contesto di questi fattori.

3.1.1. Il processo di test di sicurezza ISTQB

La tabella 3.1 mostra come tenere conto e gestire le attività di test di sicurezza nell'ambito del processo di test fondamentale di ISTQB.

Tabella 3.1 – Processo di test di sicurezza di ISTQB

Attività del processo di test di sicurezza di ISTQB	Compiti dei test di sicurezza	Responsabilità
Pianificazione dei test di sicurezza: L'obiettivo è definire un ambito appropriato del testing di sicurezza che corrisponda ai rischi di sicurezza.	• Tenere conto dei requisiti relativi alla sicurezza • Definire gli obiettivi dei test di sicurezza • Definire l'ambito del testing di sicurezza • Identificare le risorse per i test di sicurezza • Definire le stime e le schedulazioni del testing di sicurezza • Definire le metriche, i criteri di ingresso e i criteri di uscita dei test di sicurezza	Un security test analyst (STA) è responsabile di questa attività. Il STE contribuisce alla pianificazione stimando il carico di lavoro dei test e le risorse hardware e software necessarie
Analisi dei test di sicurezza: L'obiettivo è comprendere tutte le condizioni dei test di sicurezza e determinare cosa testare.	• Eseguire review della base di test per il testing di sicurezza, quali valutazioni dei rischi di sicurezza, requisiti relativi alla sicurezza, architettura basata sulla sicurezza e politiche di sicurezza • Definire le condizioni dei test di sicurezza in base a: • Obiettivi di sicurezza • Rischi di sicurezza	Un STA definisce l'ambito del testing di sicurezza

Attività del processo di test di sicurezza di ISTQB	Compiti dei test di sicurezza	Responsabilità
	• Standard di sicurezza e vulnerabilità conosciute • Difese implementate per proteggere il sistema e i suoi dati • Ambito del testing di sicurezza	
Progettazione dei test di sicurezza: L'obiettivo è identificare test case di alto livello, ovvero come testare	• Progettare test case e test suite di sicurezza • Prioritizzare i test case e le test suite • Identificare i dati di test necessari per la valutazione delle vulnerabilità o per il testing di penetrazione • Progettare l'ambiente di test di sicurezza (ovvero infrastruttura e strumenti) • Impostare la tracciabilità tra la base di test e i test case	Il STE progetta e prioritizza i test case di sicurezza Un STA esegue le review dei prodotti di lavoro del STE
Implementazione dei test di sicurezza:	• Organizzare i test case di sicurezza in procedure di test o test script. Configurare un ambiente di test per eseguire il testing di sicurezza	Un STE implementa i test case di sicurezza
Esecuzione dei test di sicurezza:	• Eseguire test di sicurezza sull'adeguatezza funzionale • Eseguire test di penetrazione • Determinare le vulnerabilità specifiche • Fornire report con informazioni dettagliate sui risultati provvisori dei test di sicurezza al management	Il STE esegue i test di sicurezza, produce risultati di test dettagliati e comunica le vulnerabilità identificate il prima possibile
Monitoraggio dei test e controllo dei test di sicurezza:	• Monitorare l'avanzamento dei test e i risultati dei test di sicurezza • Adottare le azioni correttive necessarie in risposta alle informazioni raccolte	Il STA è responsabile di questa attività.
Completamento dei test di sicurezza:	• Garantire che tutti i test di sicurezza pianificati siano stati eseguiti • Analizzare i risultati dei test di sicurezza per valutare i rischi residui	Il STA raccoglie tutte le informazioni prodotte durante l'esecuzione dei test di sicurezza e produce un test report

Attività del processo di test di sicurezza di ISTQB	Compiti dei test di sicurezza	Responsabilità
	• Analizzare i risultati dei test di sicurezza per migliorare lo sviluppo del software in termini di sicurezza • Fornire report dei risultati finali dei test di sicurezza al management e agli altri stakeholder autorizzati • Verificare se i deliverable del testing di sicurezza (ad esempio, i test report) sono stati consegnati • Archiviare i risultati dei test, i dati dei test e altre informazioni sensibili in luoghi sicuri	di alto livello per il management

Laddove sia stato eseguito il testing esplorativo, la progettazione, l'implementazione e l'esecuzione dei test di sicurezza si basano sui risultati dei test di test precedenti utilizzando tecniche standard di esplorazione quali sniffer, scanner, attacchi di forza bruta e bot. La progettazione, l'implementazione e l'esecuzione dei test vengono effettuate in modo continuativo.

3.1.2. L'ambiente di test di sicurezza

Sebbene molti tipi di test possano utilizzare un ambiente di test situato sullo stesso server o sulla stessa rete di altri sistemi, il testing di sicurezza presenta rischi specifici, anche se virtualizzati o basati su container. Ad esempio, l'esecuzione di test di sicurezza distruttivi, la contaminazione del SUT e la corruzione o la divulgazione dei dati richiedono un approccio separato alla creazione di un ambiente di test per il testing di sicurezza. Inoltre, nella maggior parte dei domini di business, le normative richiedono l'utilizzo di ambienti diversi per lo sviluppo, il testing e la produzione. Ad esempio, il requisito 6.4.2 dello standard di sicurezza dei dati del settore delle carte di pagamento (payment card industry data security standard) (PCI DSS) stabilisce che è necessaria la separazione delle attività tra gli ambienti di sviluppo, test e produzione. Analogamente, il requisito 6.4.3 del PCI DSS stabilisce che i dati di produzione (ad esempio, le reti personali) non devono essere utilizzati per i test o per lo sviluppo [vedere il PCI DSS, capitolo 6].

L'ambiente di test di sicurezza deve contenere tutte le funzioni necessarie per eseguire i test. Queste includono strumenti di test management, strumenti di test di sicurezza e strumenti di test automation. Questi sono necessari per consentire l'individuazione del maggior numero possibile di vulnerabilità entro il tempo assegnato e con il minor numero possibile di risultati falsi positivi e falsi negativi.

Potrebbe quindi essere necessario identificare, specificare e configurare un ambiente di test di sicurezza separato ed efficace per proteggere altri ambienti quali sviluppo, testing di componente, testing di integrazione dei componenti, testing di sistema, testing di accettazione e produzione. Tale efficacia deve coprire la tolleranza ai guasti relativi ai test di sicurezza distruttivi o fornire protezione contro altri sistemi in fase di test e per la produttività dei test di sicurezza.

Il STE deve analizzare e stimare l'architettura richiesta, le API e il comportamento del SUT per valutare l'impatto del testing di sicurezza e definire l'ambiente di test più efficace.

Le caratteristiche principali di un ambiente di test di sicurezza includono:

1. Isolamento al giusto livello (se necessario):
A seconda dei rischi, il SUT viene isolato tramite comunicazioni filtrate, oppure il SUT e tutti gli altri sistemi dipendenti vengono isolati da altri ambienti (ad esempio, un sito web commerciale necessita di un servizio di gestione dei pagamenti separato).
2. Rappresentativo dell'ambiente di destinazione:
Per ottenere il comportamento corretto del SUT, l'ambiente totale deve riflettere l'ambiente di produzione in termini di versione e configurazione esatte.
3. Produttivo:
Contiene tutti gli strumenti necessari per pianificare, preparare, eseguire e fornire report dei test di sicurezza, sia in modo manuale che (ove possibile) automatizzato. L'esecuzione dei test di sicurezza richiede strumenti di test specifici, come descritto nel capitolo 9.
4. Recuperabile:
Per ripetere i test secondo necessità e ripristinare eventuali danni qualora si verificassero.

3.2 Progettare i Test di Sicurezza per i Livelli di Test

La modellazione delle minacce è un'attività ripetitiva in cui ogni livello di test di sicurezza deve essere adattato in base ai risultati dei test degli ultimi risultati della modellazione delle minacce.

A seconda del tipo di progetto, è importante garantire che sia pianificato un test di sicurezza in ogni fase applicabile del SDLC.

3.2.1. Progettazione dei test di sicurezza a livello di test di componente

Base di test per la progettazione di test di sicurezza a livello di test di componente

Esempi di prodotti di lavoro che possono essere utilizzati per progettare test di sicurezza includono:

- Analisi dei rischi
- Requisiti delle funzioni e dei meccanismi di sicurezza
- Progettazione dettagliata delle funzioni e dei meccanismi di sicurezza (ad esempio API e algoritmi)
- Modelli di dati
- Regole di compilazione o costruzione
- Informazioni sul compilatore

Oggetti di test per la progettazione dei test di sicurezza a livello di test di componente

Gli oggetti di test tipici per il test di componente di sicurezza includono:

- Componenti
- Dipendenze (ad esempio, librerie di terze parti)
- Codice sorgente
- Moduli di database

Difetti e failure di sicurezza tipici a livello di test di componente

Esempi di difetti e failure di sicurezza tipici che possono essere rilevati a livello di test di componente includono:

- Codice e logica errati
- Comportamento errato
- Debolezze nel filtraggio degli input
- Problemi di flusso dei dati
- Problemi di flusso delle chiamate
- Codice irraggiungibile (morto)
- Codice malevolo inserito deliberatamente (ad esempio, bombe software)

Tipi di test di sicurezza a livello di test di componente

A livello di test di componente è possibile applicare test statici e test dinamici.

A seconda degli obiettivi dei test di sicurezza, della base di test, degli oggetti di test e dei tipi di test, è possibile utilizzare diversi approcci e tecniche di progettazione a livello di test di componente in base al codice fornito:

- Verificare che l'implementazione delle funzioni e dei meccanismi di sicurezza si comporti come previsto dai requisiti di sicurezza

La progettazione dei test di sicurezza si basa su requisiti dettagliati (ad esempio, specifiche dettagliate e progettazione dettagliata del SUT). È necessario utilizzare tecniche di test ben note basate sulle specifiche, come l'analisi ai valori limite e il partizionamento di equivalenza [ISTQB FL]. Il STE deve tracciare i test case di sicurezza alle specifiche dettagliate.

- Creare fiducia nella qualità del codice di sicurezza (cioè nella codifica sicura)

I test case di sicurezza devono focalizzarsi sull'applicazione delle regole di codifica sicura. Il STE deve anche verificare che il team di sviluppo non utilizzi istruzioni di codice pericolose ed eviti le debolezze dei linguaggi di programmazione e dei compilatori. Di solito, i team di sviluppo o le organizzazioni definiscono le proprie best practice di codifica sicura sulla base di riferimenti ben noti, che possono essere interni all'organizzazione o esterni, come la OWASP foundation. Il STE può progettare test case basati su queste regole che possono essere considerati come requisiti non funzionali (ad esempio, manutenibilità e altre caratteristiche di qualità non funzionali). Questi test case di sicurezza possono essere elaborati automaticamente utilizzando strumenti di analisi statica.

I test per qualsiasi componente dovrebbero includere la valutazione di possibili violazioni della seguente checklist delle best practice: [CERT1]

- Validare gli input
- Prestare attenzione agli avvisi del compilatore
- Architettura e progettazione delle politiche di sicurezza
- Principio "Keep it simple"
- Rifiuto predefinito, che definisce un elenco di "autorizzazioni"
- Aderenza al principio del privilegio minimo
- Sanificare i dati inviati ad altri sistemi
- Praticare la difesa in profondità
- Utilizzare tecniche efficaci di controllo della qualità
- Adottare uno standard di codifica sicuro

I test eseguiti sulla base di tali checklist delle best practice dovrebbero includere valutazioni di possibili violazioni di tali pratiche sulla base di un'analisi del rischio ben documentata che incorpori modelli di minaccia realistici.

- Individuare le vulnerabilità nei componenti

Dopo aver verificato la corretta implementazione delle funzioni e dei meccanismi di sicurezza e che siano state seguite le best practice di codifica sicura, il STE dovrebbe progettare test di sicurezza con l'obiettivo di individuare le vulnerabilità nei componenti sviluppati (ad esempio, eseguendo il fuzz testing sull'API di un componente).

Il STE può utilizzare strumenti di static application security testing (SAST) o di dynamic application security testing (DAST) per aiutare a individuare le vulnerabilità.

- Mitigare i rischi di sicurezza

Tutti i test di sicurezza sopra descritti aiutano a mitigare i rischi di sicurezza dell'applicazione o del sistema sviluppato.

Analisi della progettazione dei test di sicurezza a livello di test di componente

Una misura chiave dell'adeguatezza della progettazione dei test di sicurezza consiste nella valutazione della copertura. Diverse misure di copertura derivano dai test eseguiti.

La copertura può essere misurata in uno dei seguenti modi:

- Percentuale del numero totale di requisiti di sicurezza testati
- Percentuale degli use/abuse case di sicurezza specifici testati
- Percentuale di funzioni di sicurezza critiche, scenari o mission thread testati
- Percentuale di copertura del codice sorgente (ad esempio, per identificare codice morto o backdoor)
- Percentuale di copertura delle partizioni di equivalenza dei dati (ad esempio, per rilevare catture di eccezioni errate)
- Numero di scoperte relative alla sicurezza
- Efficienza degli strumenti di sicurezza utilizzati (ad esempio, numero di risultati falsi positivi e falsi negativi)

3.2.2. Progettazione dei test di sicurezza a livello di integrazione dei componenti

Secondo il Syllabus ISTQB Foundation [ISTQB FL], esistono due diversi livelli di integrazione: il testing di integrazione dei componenti e il testing di integrazione dei sistemi. I componenti e/o i sottosistemi da integrare possono provenire da una varietà di fonti diverse, come un altro team della stessa organizzazione, un subappaltatore, un prodotto commercial off-the-shelf (COTS), un servizio già disponibile nel cloud o un servizio open source. Durante queste attività di integrazione volte alla realizzazione finale del sistema di produzione completo, le possibilità di violazioni della sicurezza non sono semplicemente la somma delle vulnerabilità di ciascuno dei componenti. Al contrario, nuovi vettori di attacco diventano possibili a causa delle interazioni tra i componenti all'interno del sistema più ampio e a causa di elementi organizzativi.

Tuttavia, alcune interazioni tra i componenti potrebbero mitigare o bloccare possibili sequenze che portano a violazioni della sicurezza.

Il testing di integrazione dei componenti può dimostrare la complessità della progettazione di un sistema e la stabilità del suo comportamento. L'approccio dei test di integrazione dei componenti (ad esempio, top-down o bottom-up) può influire sui tempi di rivelazione dei problemi di sicurezza o sulla necessità di ulteriori test specifici per la sicurezza.

Come per i test di componente, i test di integrazione dei componenti dovrebbero essere progettati sulla base di un'analisi del rischio ben documentata che incorpori una modellizzazione realistica delle minacce. Poiché i componenti separati vengono integrati insieme, si noti che potrebbe essere necessario ricorrere a scaffolding o mocking sotto forma di stub e driver per testare percorsi incompleti attraverso un sistema durante l'integrazione. Man mano che vengono aggiunti al sistema più componenti implementati, lo scaffolding/il mocking viene rimosso in modo incrementale, consentendo una valutazione più completa dell'adeguatezza funzionale e aprendo nuovi percorsi verso vulnerabilità che potrebbero essere sfruttate.

A seconda del livello di affidabilità dei componenti/sottosistemi da integrare, la progettazione dei test di sicurezza a livello di integrazione dei componenti dovrebbe includere:

- Test di sicurezza dell'architettura di sicurezza globale basati sulla documentazione tecnica dell'architettura
- Test di sicurezza dei flussi integrati configurati (ad esempio, autorizzati o meno e il livello di confidenzialità, integrità e disponibilità)
- Test di sicurezza delle API integrate (ad esempio, per rilevare problemi di sicurezza nelle API dovuti alla mancanza di controlli o alla mancanza di conoscenza di tali API)
- Test di sicurezza relativi alla configurazione di sicurezza dei componenti integrati (ad esempio, filtraggio dell'accesso di un componente da parte di un altro, poiché i componenti non firmati dovrebbero avere un accesso limitato)
- Verifica che i componenti integrati che sono software esterni, open source o closed source siano privi di vulnerabilità

A livello di integrazione dei componenti, la copertura può essere misurata in uno dei seguenti modi:

- Percentuale di API utilizzate/testate
- Percentuale di interazioni testate tra componenti/sottosistemi sulla base della documentazione tecnica dell'architettura
- Numero di scoperte relative alla sicurezza a livello di integrazione dei componenti

- Numero di scoperte relative alla sicurezza che avrebbero dovuto essere rilevate a livello di test di componente
- Efficienza degli strumenti di sicurezza utilizzati (ad esempio, numero di risultati falsi positivi e falsi negativi)

3.2.3. Il testing di sicurezza nel testing di sistema e nel testing di accettazione

Testing di sicurezza del sistema

Si tratta del livello di test durante il quale viene testata l'implementazione dei requisiti di sicurezza per garantire che funzionino come previsto. Le attività di testing di sicurezza del sistema includono l'esecuzione di test di sicurezza in un ambiente approssimativamente simile a quello di produzione, il che richiede il passaggio dall'ambiente di sviluppo in cui sono state eseguite le precedenti attività di implementazione e integrazione.

Il ruolo del testing di sicurezza nel testing di sistema

Il testing di sicurezza del sistema è la prima opportunità per esercitare la funzionalità end-to-end dei componenti completamente integrati. Sebbene venga solitamente eseguito in un ambiente di test, dovrebbe rivelare le proprietà emergenti del sistema che non sarebbero state osservate prima del completamento dell'integrazione. In genere, i requisiti di sicurezza sono considerati insieme ai requisiti funzionali.

L'obiettivo del testing di sicurezza nel testing di sistema è testare:

- tutti i requisiti di sicurezza implementati nelle funzioni di sicurezza in un ambiente di test che rappresenta l'ambiente di produzione
- che la configurazione operativa sia sicura

Testing di sicurezza di accettazione

Si tratta dell'ultimo livello di testing durante il quale gli utenti, o i loro rappresentanti, acquisiscono la certezza che il sistema sia in grado di fornire le funzionalità necessarie nell'ambiente di produzione in modo sicuro. Gli obiettivi dei test di sicurezza di accettazione includono il testing di sicurezza rispetto ai criteri di accettazione relativi alla sicurezza stabiliti per il sistema. In genere, i criteri di accettazione relativi alla sicurezza si focalizzano sui controlli e sui processi di sicurezza funzionali. Le attività di testing di sicurezza di accettazione possono includere:

- Installare il sistema in un ambiente di preproduzione
- Eseguire i test di sicurezza sulla base dei criteri di accettazione
- Determinare l'accettazione sulla base dei risultati dei test di sicurezza

Il ruolo del testing di sicurezza nel testing di accettazione

Il testing di accettazione si distingue dal testing di sistema in quanto viene eseguito in un ambiente simile a quello di produzione. Esso colloca infine il sistema in un ambiente in cui gli agenti di minaccia esterni cercherebbero quotidianamente di individuare eventuali debolezze. Questi test consentono una valutazione ragionevole dell'efficienza delle prestazioni e di altri comportamenti basati sulle interazioni attraverso interfacce esterne.

Il testing di accettazione dovrebbe idealmente validare che gli obiettivi iniziali del progetto siano stati raggiunti e che i criteri di accettazione della sicurezza documentati siano soddisfatti. Questo si ottiene progettando ed eseguendo test per validare i processi/scenari di sicurezza come il controllo dei diritti, la gestione delle autorizzazioni e il filtraggio tramite firewall.

Il momento migliore per definire e documentare i criteri di accettazione è prima dello sviluppo o dell'acquisto del sistema. Nel contesto del testing di sicurezza, i criteri di accettazione possono essere di natura globale. Ad esempio, potrebbero esserci criteri di accettazione che specificano ciò che è accettabile in termini di sicurezza complessiva del sistema. Ciò includerebbe criteri applicati a tutte le funzioni del sistema, come l'autenticazione degli utenti, i diritti degli utenti, i livelli di crittografia e le tracce di audit.

Analisi della progettazione dei test di sicurezza a livello di test di accettazione

A livello di test di accettazione, la copertura può essere misurata come segue:

- Percentuale di processi/scenari di sicurezza testati
- Numero di scoperte relative alla sicurezza che avrebbero dovuto essere rilevate nei livelli di test precedenti con la loro severità
- Efficienza degli strumenti di sicurezza utilizzati (ad esempio, numero di risultati falsi positivi e falsi negativi)
- Percentuale dei requisiti di sicurezza testati
- Percentuale di elementi di configurazione sicura operativa testati

4 Standard e Best Practice del Testing di Sicurezza - 195 minuti (K3)

Parole Chiave

Common Attack Pattern Enumeration and Classification (CAPEC), Common Vulnerabilities and Exposures (CVE), Common Vulnerability Scoring System (CVSS), Common Weakness Enumeration (CWE), Common Weakness Scoring System (CWSS), vulnerabilità, debolezza

Parole Chiave Relative alla Sicurezza

nessuna

Obiettivi di Apprendimento per il Capitolo 4:

4.1 Introduzione agli Standard e alle Best Practice di Sicurezza

STE-4.1.1 (K3) Spiegare le diverse fonti degli standard e delle best practice e la loro applicabilità

4.2 Applicare Standard e Best Practice Importanti per il Testing di Sicurezza

STE-4.2.1 (K3) Applicare il concetto di Open Web Application Security Project, Common Vulnerability Enumeration, Common Weakness Enumeration, il Common Vulnerability Scoring System e il Common Weakness Scoring System e come utilizzarli per il testing di sicurezza

4.3 Utilizzare gli Standard e le Best Practice del Testing di Sicurezza

STE-4.3.1 (K2) Spiegare i vantaggi e gli svantaggi degli oracoli del test utilizzati per il testing di sicurezza

STE-4.3.2 (K3) Comprendere i vantaggi e gli svantaggi dell'utilizzo degli standard e delle best practice di sicurezza

4.1 Introduzione agli Standard e alle Best Practice

Gli standard e le best practice di vario tipo forniscono visibilità sul consenso professionale e sugli obblighi normativi. Uno standard basato sul consenso rappresenta l'opinione ponderata di un gruppo di esperti competenti.

Sebbene siano spesso utilizzati come sinonimi, esistono grandi differenze tra standard e best practice, che vengono spiegate nelle sottosezioni seguenti. Le differenze hanno un impatto significativo sul processo di selezione e sui possibili use case per il loro utilizzo.

4.1.1. Standard e best practice

Standard

Gli standard sono definiti come "un documento, stabilito dal consenso di esperti in materia e approvati da un organismo riconosciuto che fornisce una guida sulla progettazione, sull'uso o sulle prestazioni di materiali, prodotti, processi, servizi, sistemi o persone" ([ISO_Web_21] e Appendice D).

Esistono diversi livelli di "organismo riconosciuto", che consentono di distinguere tra diversi tipi di standard. Uno dei livelli più elevati di riconoscimento degli standard è rappresentato a livello mondiale dall'International Standard Organization (ISO). Di solito, ogni paese che fa parte della World Trade Organization (WTO) ha una propria rappresentanza locale. Gli standard hanno il massimo livello di riconoscimento per un STE grazie al loro elevato grado di maturità. Tuttavia, questa maturità ha lo svantaggio di richiedere molto tempo per essere completata e spesso porta alla creazione di standard con un carattere molto reattivo.

Gli organismi riconosciuti possono creare i propri standard. Questi possono essere classificati come standard industriali, standard de facto e standard specifici del produttore:

- **Standard industriali:**
Sono stati stabiliti nel corso di anni di applicazione in molti contesti e hanno dimostrato alcuni valori aggiunti risolvendo un problema particolare. L'Internet Engineering Task Force (IETF) è un attore importante nella creazione di standard a questo livello. Crea standard basati sul giudizio ingegneristico combinato dei loro partecipanti e sulla loro esperienza del mondo reale nell'implementazione e nel deployment delle loro specifiche [IETF23].
- **Standard de facto:**
Hanno spesso le loro radici negli standard industriali. Poiché la loro copertura e accettazione è elevata, soddisfano anche molti dei criteri per essere considerati al livello più alto degli standard. Un buon esempio è il protocollo TCP, che è stato stabilito come standard industriale ma oggi è considerato uno standard de facto [IETF23].
- **Standard specifici del produttore:**
Alcuni clienti/organizzazioni hanno compreso che seguire le specifiche proprietarie di un determinato produttore comporta un valore aggiunto.

Nella realtà questa chiara classificazione potrebbe presentare molte sovrapposizioni confuse e non è sempre semplice classificare in modo chiaro un determinato standard.

Best practice

Le best practice e il loro organismo di riferimento sono organizzati in modo meno formale. Il Glossario di Gartner [Gart21] definisce le best practice come un "insieme di attività che ottimizza l'efficienza (costi e rischi) o l'efficacia (livello di servizio) della disciplina o del processo di business a cui contribuiscono. Devono essere implementabili, replicabili, trasferibili e adattabili in tutti i settori". A questo livello, ogni gruppo di esperti, anche se lavora nello stesso contesto, può creare il proprio insieme di best practice.

4.2 Applicare Standard e Best Practice Importanti per il Testing di Sicurezza

Esistono diversi standard e best practice per la disciplina del testing di sicurezza. A causa dell'elevato livello di requisiti da soddisfare per essere considerati uno standard, la loro creazione e manutenzione è molto più lenta rispetto alle best practice. Questo consente un profondo riconoscimento all'interno del settore e include molti cicli di feedback per il miglioramento. Tuttavia, ciò impedisce il rapido adattamento alle nuove tendenze e ai nuovi rischi. In confronto, le best practice hanno un'elevata efficienza complessiva in termini di prestazioni, ma è più difficile che diventino ben note, raggiungano un alto livello di copertura e siano supportate da prove pratiche.

4.2.1. Standard industriali per il testing di sicurezza

Lo standard più consolidato nella sicurezza IT è la serie ISO 27000. Lo standard [ISO 27001] è accettato a livello internazionale e intitolato "Information technology – Security techniques – Information security management systems - Overview and vocabulary". L'obiettivo di questo standard è la gestione della sicurezza delle informazioni (information security), ovvero identificare i rischi, valutarli e gestirli attraverso controlli di sicurezza delle informazioni. Tutte queste attività sono combinate in un information security management system (ISMS) che costituisce il nucleo centrale dello standard ISO 27000. Lo standard ha un ambito ampio e si focalizza sul modo generale in cui un'organizzazione dovrebbe valutare i propri rischi, confrontarli con le proprie esigenze specifiche e affrontare i rischi più rilevanti. Lo standard fondamentale può essere applicato a tutte le organizzazioni.

La serie ISO 27000 è composta da oltre 40 standard individuali, che possono essere classificati come segue:

- Standard principale: panoramica generale e introduzione a un ISMS (a partire da ISO 27000 fino a ISO 27005)
- Standard specifici per argomento per coprire argomenti specifici, come la gestione dei servizi ISO 27013 e i fornitori di servizi cloud pubblici ISO 27017 (vedere [ITGOV23a])
- Standard specifici del dominio per focalizzarsi su domini specifici, come i fornitori di telecomunicazioni ISO 27011 e il settore finanziario ISO 27015 (vedere [ITGOV23a])

Gli standard più utilizzati che si applicano al contesto del testing di sicurezza e coprono gli oggetti di test e le condizioni di test rilevanti che il STE dovrebbe prendere in considerazione sono i seguenti:

- ISO 27000: questa parte spiega la struttura generale della serie ISO 27000 e introduce un ISMS e il ruolo che può svolgere il testing di sicurezza.

- ISO 27001: è lo standard più utilizzato, in quanto elenca una serie completa di raccomandazioni e controlli di sicurezza per strutturare e costruire un ISMS individuale. Il suo obiettivo è stabilire una visione completa degli asset rilevanti all'interno di un'organizzazione, dei rischi a cui sono esposti e delle possibili mitigazioni. [ISO 27001]
- ISO 27001, Appendice A:
La parte più importante di ISO 27001 per un STE è presentata in questa appendice. Elenca i controlli di sicurezza per diversi aspetti quali il controllo degli accessi, il ripristino di emergenza e la sicurezza della rete. Ognuno di questi controlli, se applicato in un contesto specifico, costituisce un input importante per un STE, poiché il suo compito è misurare l'efficacia di un controllo di sicurezza. [Cald11]
- ISO 27002: questo standard riprende i controlli di sicurezza generici da ISO 27001 e fornisce ulteriori indicazioni su come applicarli nella pratica e su come specificarli in modo più dettagliato per un contesto specifico. [Cald11]
- ISO 27003: questo standard supporta un'organizzazione nella creazione di un piano per stabilire un ISMS basato su ISO 27001.

Standard de facto per il testing di sicurezza

Esistono molti standard de facto che possono essere utilizzati da un STE. Una delle serie più importanti di standard de facto proviene dalla società MITRE, anche se la sua attività principale non consiste nel generare standard. MITRE è un'organizzazione privata senza scopo di lucro che fornisce consulenza ingegneristica e tecnica al governo federale degli Stati Uniti. I principali sponsor di MITRE sono il Department of Defense, la Federal Aviation Administration e il Department of Homeland Security [MITRE21].

Per quanto riguarda il testing di sicurezza, MITRE ospita e mantiene i seguenti standard ben noti che forniscono un valore aggiunto al STE:

Common Attack Pattern Enumeration and Classification (CAPEC™)

CAPEC™ fornisce un catalogo pubblicamente disponibile dei modelli di attacco comuni. L'idea è di comprendere meglio come gli attacker sfruttano le debolezze delle applicazioni e altre capacità abilitate digitalmente. I modelli di attacco si basano su modelli di progettazione software per gli attacker. Due modelli di attacco tipici sono l'SQL injection (CAPEC-66) e il relativo path traversal (CAPEC-139) [CAPEC21].

CAPEC fornisce diverse viste sui suoi set di dati. Le più rilevanti sono:

- Dominio dell'attacco, come software, social engineering e sicurezza fisica. Al livello più alto, CAPEC elenca nove domini di attacco.
- Meccanismi di attacco, come l'iniezione di elementi imprevisti e la manipolazione delle risorse di sistema. Al livello più alto, CAPEC elenca sei meccanismi di attacco.

Ogni oggetto di test che un STE testa dovrebbe essere inserito in questo catalogo. Spesso CAPEC è il punto di partenza per ottenere una panoramica iniziale dei possibili attacchi che potrebbero essere rilevanti per un determinato sistema.

I seguenti standard MITRE vengono utilizzati per un ulteriore affinamento efficace del testing di sicurezza:

Common Weakness Enumeration (CWE)

CWE è un elenco di debolezze software/hardware. Di solito, ogni modello di attacco comune presenta una o più debolezze che possono essere utilizzate per un modello di attacco CAPAC [CWE21]. CWE utilizza il concetto di viste, le più utilizzate delle quali sono:

- Sviluppo software, come API, pratiche di codifica errata e problemi di autorizzazione. Al livello più alto, CWE elenca 40 asset di sviluppo software.
- Progettazione hardware, come problemi di memoria e archiviazione, problemi con il core e il calcolo, periferiche, tessuto integrato e problemi di interfaccia I/O. Al livello più alto, CWE elenca 12 asset di progettazione hardware.

Ogni debolezza comune è un punto di partenza efficace per un STE per testare se il modello di attacco sottostante può essere sfruttato.

Open Web Application Security Project (OWASP)

È importante rendersi conto che CWE e OWASP [OWASP21] si sovrappongono ed entrambi elencano le debolezze comuni. OWASP è ben noto per la pubblicazione della sua classifica OWASP Top 10.

Common Weakness Scoring System (CWSS)

Più una debolezza diventa comune, più diventa importante disporre di uno schema di prioritizzazione. CWSS fornisce un meccanismo per prioritizzare le debolezze in modo coerente, flessibile e aperto [CWSS21]. La prioritizzazione viene calcolata in base a tre serie di metriche:

- **Base Finding Metric Group:**
Vengono calcolati il rischio intrinseco di una debolezza, l'affidabilità dell'accuratezza della scoperta e la forza dei controlli. Una metrica tipica è l'"impatto tecnico", che varia dal controllo completo su un sistema all'assenza di impatto tecnico.
- **Attack Surface Metric Group:**
Calcola le barriere che un attacker deve superare per sfruttare la debolezza. Una metrica tipica è il privilegio richiesto, che varia da nessun privilegio richiesto ai privilegi di amministratore.
- **Environmental Metric Group:**
Calcola le caratteristiche delle debolezze specifiche di un determinato ambiente o contesto operativo. Una metrica tipica è l'impatto sul business, che varia dal fallimento del business all'assenza di impatto.

Utilizzando pesi specifici predefiniti, tutte queste metriche possono essere aggregate in un unico valore CWSS complessivo per una specifica debolezza. Il CWSS è in grado di gestire metriche sconosciute utilizzando valori predefiniti o definendo/focalizzandosi su un singolo sottoinsieme di metriche. Inoltre, molte metriche del Base Finding Metric Group possono essere calcolate automaticamente da uno strumento di analisi statica.

Common Vulnerability Scoring System (CVSS)

Un meccanismo di prioritizzazione simile al CWSS è il CVSS [CVSS21], che segue un approccio simile, ma presuppone una vulnerabilità esistente e distribuita (vedere CVE di seguito). Sia il CVSS che il CWSS sono sistemi di punteggio per la sicurezza informatica: il CVSS è un approccio reattivo perché le vulnerabilità esistono già prima della classificazione. Il CWSS è un approccio proattivo, poiché si lavora con il software prima di rilasciarlo in produzione. Entrambi gli approcci sono spesso utilizzati insieme, anche se non sono completamente compatibili (cfr. [SecJour21]).

Common Vulnerabilities and Exposures (CVE)

CVE è un database di informazioni divulgate pubblicamente sui problemi di sicurezza [CVE21]. Un numero CVE identifica in modo univoco una particolare vulnerabilità dall'elenco. CVE è utile perché fornisce un identificatore standardizzato per una determinata vulnerabilità all'interno di un sistema specifico. Se un sistema è interessato da un CVE specifico, questa vulnerabilità è un'istanza specifica di una debolezza comune (common weakness) (CWE) che può essere utilizzata per eseguire un attacco specifico (CAPEC). Le nuove voci all'interno del repository CVE di solito provengono dal lavoro quotidiano dei STE. Se identificano una nuova vulnerabilità sconosciuta al CVE, possono pubblicarla sul CVE per coinvolgere la comunità di sicurezza nell'identificazione di contromisure.

Best practice per il testing di sicurezza

Le best practice devono solo soddisfare un criterio formale minimo per essere considerate tali. Molte best practice possono fallire dopo un certo periodo di tempo se non sono efficaci. Alcune smetteranno di essere utilizzate a causa della mancanza di pubblicazione/marketing, ma alcune potrebbero migliorare la loro maturità fino a essere considerate uno standard.

Una best practice matura tipica ancora oggi utilizzata è il modello STRIDE, inventato da Microsoft [Micro09]. STRIDE consente una modellizzazione sistematica delle minacce dal punto di vista dell'attacker. Il termine stesso è l'acronimo di sei categorie di minacce, che classificano le potenziali minacce: **s**poofing, **t**ampering, **r**epudiation, **i**nformation disclosure, **d**enial of service ed **e**levation of privilege.

- Spoofing dell'identità, ovvero dichiarare all'interno di un sistema di essere una persona o un sistema che non si è
- Tampering (manomissione) dei dati, ovvero la modifica dannosa dei dati
- Repudiation (ripudio), ovvero minacce che prendono di mira l'auditing e il tracciamento, assicurando che i comportamenti scorretti non possano essere provati
- Information disclosure (divulgazione di informazioni), ovvero l'esposizione di informazioni a persone che non dovrebbero avervi accesso
- Denial of service (negazione del servizio), ovvero negare il servizio a utenti validi
- Elevation of privilege (elevazione dei privilegi), ovvero quando un utente non privilegiato ottiene un accesso privilegiato.

In generale, STRIDE viene utilizzato per aiutare gli sviluppatori a considerare le minacce durante la progettazione e a colmare le lacune individuate. Il STE può utilizzare lo stesso approccio per focalizzarsi sul testing.

4.3 Utilizzare gli Standard e le Best Practice del Testing di Sicurezza

Esistono molti use case possibili per utilizzare gli standard e le best practice. In generale, questi use case possono essere suddivisi in applicazioni obbligatorie e applicazioni volontarie.

4.3.1. Applicazione obbligatoria degli standard e delle best practice

In questo tipo di use case, gli standard e le best practice sono obbligatori per un'altra parte:

- Requisiti di sicurezza per i contratti:
Le best practice sono un modo efficace per specificare i requisiti di sicurezza per lo sviluppo di software, in particolare quelli delegati a terzi. Aniché elencare tutti i requisiti specifici, è richiesto solo il rispetto di uno standard specifico, il che implica il rispetto di tutti i consigli e i requisiti di sicurezza in esso contenuti.
- Requisiti di sicurezza come regolamento:
Anche le istituzioni normative (ad esempio nel dominio bancario) utilizzano spesso standard e best practice, le quali sono facili da gestire.

4.3.2. Applicazione volontaria degli standard e delle best practice

In questo tipo di use case, l'applicazione di standard e best practice specifici è una decisione presa dal management per generare il seguente valore aggiunto:

- Stabilire un elevato livello di sicurezza riutilizzando le conoscenze consolidate in materia di sicurezza contenute negli standard e nelle best practice esistenti
- Prove ben note per dimostrare la consapevolezza in materia di sicurezza
- Finalità di marketing generale e creazione di punti di forza unici in un settore di business competitivo.

4.3.3. Oracoli del test estratti da standard e best practice

Uno use case generale per utilizzare gli standard e le best practice, indipendentemente dal fatto che siano obbligatori o volontari, è l'idea di utilizzare potenti oracoli del test. A livello di applicazione, l'oracolo del test è solitamente la sezione dei requisiti di sicurezza nella specifica. A livelli inferiori, ad esempio librerie incluse, sistema operativo sottostante, traffico di rete, standard e best practice possono essere facilmente utilizzati. In particolare, le best practice più volatili potrebbero elencare molte vulnerabilità conosciute per un determinato sistema e determinare i risultati attesi da utilizzare come prova della sicurezza o della non sicurezza. Si tratta di uno strumento potente per i STE, poiché devono solo definire i test case di basso livello corrispondenti, eseguirli e quindi confrontare i risultati dei test con quelli elencati nella best practice.

4.3.4. Vantaggi e svantaggi dell'utilizzo degli standard e delle best practice per il testing di sicurezza

L'utilizzo di standard e best practice per il testing di sicurezza presenta molti vantaggi, ma ci sono alcuni aspetti negativi da considerare attentamente in un contesto specifico. In generale, l'utilizzo di standard e best practice presenta i seguenti vantaggi:

- Terminologia coerente:
Nel settore IT esistono molti termini di marketing, sinonimi ed espressioni senza una definizione chiara o una distinzione tra loro. Gli standard e talvolta anche le best practice possono contribuire a chiarire la terminologia.
- Riutilizzo delle conoscenze di esperti:
La definizione di standard e best practice può essere un'attività che richiede molto tempo e che di solito viene svolta da esperti di sicurezza. Le loro conoscenze possono essere acquisite e riutilizzate negli standard e nelle best practice.
- Doppio controllo di benchmark e completezza:
Se un'azienda utilizza un proprio framework di test di sicurezza specifico, gli standard e le best practice esistenti possono essere utilizzati come benchmark per verificare la completezza delle proprie soluzioni.
- Miglioramento del commitment tra fornitore e cliente:
Più uno standard o una best practice sono consolidati e riconosciuti, più possono essere utilizzati

in modo efficiente come base per il commitment tra consumatore (ciò che desidera avere) e fornitore (ciò che deve fare).

- Facile comunicazione sul livello di sicurezza raggiunto:
Se un'organizzazione utilizza una propria serie di test di sicurezza senza alcun riferimento esterno, potrebbe essere difficile dimostrarne l'efficacia. L'utilizzo di standard e best practice contribuisce a creare un atteggiamento complessivamente positivo e semplifica notevolmente la comunicazione.

Tuttavia, l'utilizzo di standard e best practice per il testing di sicurezza può presentare alcuni aspetti negativi:

- Selezione errata:
Esistono molti standard e best practice disponibili, ciascuno con il proprio focus e le proprie precondizioni necessarie per essere applicabile. Utilizzare la fonte sbagliata riduce l'impatto nel raggiungere una maggiore sicurezza e potrebbe persino diminuire le risorse disponibili da destinare alla sicurezza.
- Best practice in un contesto specifico errato:
Mentre la maggior parte degli standard ha raggiunto un'elevata qualità grazie ai lunghi processi di creazione e ai lunghi cicli di feedback, le best practice possono avere una durata limitata. Soprattutto quando vengono proposte inizialmente, la loro correttezza e il loro valore aggiunto non sono sempre chiari e potrebbero non avere un forte legame con il contesto specifico. L'uso di una nuova best practice proprietaria che non ha ancora dimostrato di creare alcun valore aggiunto potrebbe persino ridurre il livello di sicurezza.
- Mancanza di personalizzazione:
Spesso gli standard e le best practice definiscono determinati parametri che devono essere soddisfatti per essere applicabili in un contesto specifico. Se questo viene ommesso o non viene fatto correttamente, la loro applicazione potrebbe produrre un valore aggiunto limitato.
- Considerazioni di commodity (utilità per unicità):
Più uno standard o una best practice diventano consolidati e popolari, meno possono essere utilizzati per creare unicità rispetto ad altri prodotti concorrenti (se richiesto).
- Cecità operativa:
Difetti o l'insorgere di nuove minacce potrebbero portare a una minore attenzione se gli standard o le best practice non vengono adottati tempestivamente.

5 Adattare il Testing di Sicurezza al Contesto Organizzativo - 195 minuti (K4)

Parole Chiave

debolezza

Parole Chiave Relative alla Sicurezza

rootkit

Obiettivi di Apprendimento per il Capitolo 5:

5.1 L'Impatto delle Strutture Organizzative nel Contesto del Testing di Sicurezza

STE-5.1.1 (K3) Analizzare un determinato contesto organizzativo e determinare quali aspetti specifici considerare per il testing di sicurezza

5.2 L'Impatto delle Normative sulle Politiche di Sicurezza e Come Testarle

STE-5.2.1 (K3) Analizzare l'impatto delle normative sulle politiche di sicurezza e come testarle

5.3 Analizzare uno Scenario di Attacco

STE-5.3.1 (K4) Analizzare uno scenario di attacco e identificare le possibili fonti e motivazioni dell'attacco

5.1 L'Impatto delle Strutture Organizzative nel Contesto del Testing di Sicurezza

La sicurezza delle informazioni (information security) non può essere garantita solo proteggendo l'infrastruttura e affidandosi a misure implementate tecnologicamente. È necessario considerare anche le persone e i processi di un'organizzazione.

5.1.1. Analizzare un determinato contesto organizzativo e determinare quali aspetti specifici considerare per il testing di sicurezza

Le persone spesso diventano vittime di attacchi di social engineering e processi importanti, come una risposta definita alle emergenze, possono essere assenti o implementati in modo improprio. Un STE deve quindi coprire questi aspetti durante il testing di sicurezza, perché entrambi influenzano la sicurezza delle informazioni di un'organizzazione. Le persone hanno un ruolo definito. A seconda del loro ruolo, sono coinvolte in processi diversi. Di solito, i ruoli e i processi dipendono in modo significativo dalla struttura organizzativa.

In genere, la struttura organizzativa può essere classificata nei seguenti tre tipi:

- Struttura funzionale: organizzata in base a funzioni comuni, quali produzione, marketing, risorse umane, IT e contabilità
- Struttura divisionale: organizzata come un insieme di funzioni che producono un prodotto
- Struttura a matrice: i dipendenti sono raggruppati contemporaneamente sia per funzione che per prodotto

Di seguito viene esaminato il modo in cui queste strutture organizzative influiscono sul flusso di informazioni e sull'implementazione delle decisioni amministrative:

- In un'organizzazione strutturata in modo funzionale, le informazioni devono essere scambiate tra i diversi reparti dell'organizzazione e le decisioni amministrative vengono implementate direttamente con un approccio top-down, dal management all'intera organizzazione
- Le organizzazioni strutturate in modo divisionale aggiungono un livello amministrativo al vertice di ogni divisione e, pertanto, le decisioni possono influire solo su una singola divisione. Inoltre, il flusso di informazioni tra le divisioni è leggermente ridotto, anche se le divisioni spesso contengono reparti simili, come quello dello sviluppo
- Le strutture a matrice cercano di unire entrambi gli aspetti. Sono separate dal punto di vista funzionale, ma adottano anche un approccio basato sul prodotto senza aggiungere il livello amministrativo separato che si trova nelle strutture divisionali. Tuttavia, vi è un rischio maggiore di conflitti, poiché le decisioni possono essere prese sia dal punto di vista del prodotto che da un punto di vista puramente amministrativo.

Trasferendo questo concetto al contesto del testing di sicurezza, un STE dovrebbe essere a conoscenza della struttura organizzativa per i seguenti motivi:

1. I risultati di un test di sicurezza saranno influenzati dal reparto che ha richiesto e pianificato il test.

2. A seconda della struttura complessiva dell'organizzazione, un STE può trarre vantaggio dalle debolezze nel flusso di informazioni.

Il primo di questi due motivi deriva dal fatto che i reparti IT e sicurezza sono solitamente autorizzati ad implementare e garantire la sicurezza, ma anche altri reparti potrebbero esserne consapevoli. Ad esempio, sapere che un dipendente del team di testing di penetrazione sta visitando un altro reparto aumenterà la probabilità che le persone di quel reparto rispettino effettivamente le politiche di sicurezza, almeno per il tempo della loro permanenza.

La seconda delle due ragioni sopra indicate deriva dalla probabilità che per ogni struttura organizzativa possano esistere particolari debolezze. In una struttura funzionale, un STE potrebbe trarne vantaggio come segue:

- I dipendenti di reparti diversi potrebbero non conoscere le persone di altri reparti, in particolare quelle responsabili dell'amministrazione IT.
- La consapevolezza della sicurezza e l'accettazione di determinate misure di sicurezza potrebbero essere significativamente inferiori nei reparti in cui lavora solo personale non tecnico.
- Le informazioni relative a un incidente potrebbero rimanere all'interno di un singolo reparto per un certo periodo di tempo, causando, nel peggiore dei casi, ritardi nei tempi di reazione.

Nelle organizzazioni strutturate in modo divisionale, si possono considerare potenziali debolezze simili, anche se queste possono essere trasferiti alle divisioni:

- I dipendenti di reparti diversi potrebbero non conoscere le persone di altri reparti. A seconda di come viene mantenuta l'infrastruttura IT dell'organizzazione, potrebbero non sapere chi è responsabile dell'amministrazione IT.
- Le informazioni relative a un incidente potrebbero rimanere all'interno di un singolo reparto per un certo periodo di tempo, causando ritardi nei tempi di reazione.
- A seconda delle dimensioni di una singola divisione, i team potrebbero anche essere suddivisi per funzionalità in reparti più piccoli all'interno della divisione, rendendo il punto sopra indicato relativo alla consapevolezza della sicurezza e all'accettazione di determinate misure di sicurezza valido anche per le strutture divisionali.

Sebbene le organizzazioni con una struttura divisionale distribuiscano funzionalità simili su diverse divisioni, alcuni servizi come l'amministrazione IT risiedono spesso in un reparto centrale.

Nel caso delle organizzazioni a matrice, una persona potrebbe trarre vantaggio da possibili conflitti tra l'administration management e il product management. Tuttavia, questo non è sempre vero e, come già menzionato per le organizzazioni strutturate in modo divisionale, alcuni servizi potrebbero essere centralizzati.

Alcune considerazioni

Le informazioni sopra menzionate offrono una prospettiva di alto livello delle strutture organizzative e delle loro potenziali debolezze. Nella pratica, molte organizzazioni non hanno strutture puramente

funzionali, divisionali o a matrice. In particolare, la funzione di gestione della sicurezza è spesso gestita da un dipartimento centrale che detta le misure di sicurezza, come una politica di sicurezza a livello di organizzazione.

Una politica di sicurezza può essere definita come "un documento di alto livello che descrive i principi, l'approccio e gli obiettivi principali dell'organizzazione che riguardano la sicurezza" [ISTQB Glossary]. Secondo il NIST, un servizio di sicurezza è definito come "una capacità che supporta uno o più obiettivi di sicurezza. Esempi di servizi di sicurezza sono la gestione delle chiavi, il controllo degli accessi e l'autenticazione" [NIST02].

Lo studio delle politiche di sicurezza dell'organizzazione può rivelare potenziali vettori di attacco, rendendo noti i vincoli imposti al comportamento dei suoi membri e quelli imposti agli avversari dai meccanismi di sicurezza. Le politiche di sicurezza aziendali potrebbero non essere disponibili al pubblico. Alcune politiche organizzative potrebbero essere accessibili solo ai dipendenti o anche solo ad alcuni membri del personale.

Un aspetto importante da considerare nel contesto organizzativo è il modo in cui l'organizzazione esternalizza parti della propria produzione o dei propri servizi. Anche i partner rilevanti dovrebbero essere considerati potenziali obiettivi di un test di sicurezza. Questo dipende dalla natura e dal contenuto del contratto tra le due organizzazioni che definisce gli obblighi legali. Ai partner esterni viene spesso concesso un accesso (limitato) alla virtual private network (VPN), lavorano sullo stesso repository di codice o dispongono di un token di accesso per un ufficio locale. Anche se spesso hanno account con restrizioni, questo potrebbe essere il primo passo verso un attacco riuscito. Un altro aspetto importante relativo ai partner nel contesto del testing di sicurezza è l'analisi della supply chain, poiché gli attacchi in questo settore possono avere gravi conseguenze (ad esempio, [WIRED21]).

Gli aspetti generali trattati in questa sezione potrebbero valere per quasi tutte le organizzazioni, ma trascurano questioni industriali specifiche, come il tipo di prodotto o servizio offerto da un'organizzazione, nonché il settore industriale in cui opera. Offrire un servizio web per la musica potrebbe avere requisiti di sicurezza diversi rispetto a un dispositivo medico utilizzato in un ospedale. Proprio per questo motivo, esistono normative per determinati settori che prescrivono requisiti per i processi, la safety, le misure di sicurezza o altri aspetti specifici del dominio (vedere la sezione 5.2). Questo potrebbe influire sulle organizzazioni che sviluppano i propri prodotti molto più che sulle organizzazioni che, ad esempio, vendono prodotti COTS.

5.2 L'Impatto delle Normative sulle Politiche di Sicurezza e Come Testarle

Le normative di sicurezza determinano il contenuto delle politiche di sicurezza, che a loro volta determinano il framework di controllo della sicurezza delle informazioni per il testing di sicurezza. Il STA sviluppa questo framework di controllo. Conoscendo il framework, un STE sviluppa e utilizza test case per testare i controlli.

5.2.1. L'impatto delle normative governative sulle normative di sicurezza

A causa della forte interconnessione della maggior parte dei settori industriali, gli attacchi alla sicurezza informatica possono avere un impatto profondo su una singola organizzazione e, nel peggiore dei casi, sull'infrastruttura complessiva di un intero paese. In risposta a ciò, i governi hanno definito delle normative per obbligare le organizzazioni critiche ad adeguare il loro livello di sicurezza almeno a un minimo. Le organizzazioni non conformi possono essere multate o temporaneamente chiuse fino all'implementazione delle misure di sicurezza richieste. Le organizzazioni interessate dalle normative hanno quindi un incentivo a migliorare le loro misure e politiche di sicurezza almeno al livello di sicurezza richiesto. Il NIST definisce le leggi e le normative nel contesto della sicurezza informatica come "leggi, normative, politiche, linee guida, standard e procedure a livello federale e specifiche per le organizzazioni che impongono requisiti per la gestione e la protezione delle risorse di tecnologia delle informazioni". Sebbene questo possa essere vero in alcuni casi, le normative generali possono essere definite a livello globale, sindacale o nazionale, come ad esempio:

- Normative globali definite, ad esempio, dalla WTO
- Normative specifiche dei sindacati, come il GDPR e le Network and Information Systems (NIS) Regulations definite dall'Unione Europea (UE)
- Normative nazionali come il Cyber Security Sharing Act negli Stati Uniti d'America (USA)

In aggiunta, ulteriori normative possono essere applicate a settori industriali specifici, come ad esempio:

- Health Insurance Portability and Accountability Act [HIPAA]
- UNECE WP.29 per il settore automotive [UNECE20]
- Implementing Regulation (DVO) (UE) 2019/1583 per la sicurezza aerea [BSI21]
- PCI DSS [PCI22]

La formulazione delle normative è affidata a istituzioni specifiche quali la Cybersecurity and Infrastructure Security Agency negli Stati Uniti o il Federal Office for Information Security in Germania. Nell'Unione Europea, la European Network and Information Security Agency ha definito la direttiva NIS, che è stata adottata come politica nel 2016. L'obiettivo è aumentare e standardizzare il livello di sicurezza informatica in tutti gli Stati membri. Le stesse istituzioni pubblicano spesso raccomandazioni (ad esempio, [TR02021]), best practice o almeno riferimenti ad altre pubblicazioni.

Questo è importante perché le normative possono essere molto vaghe riguardo alla tecnologia effettiva da utilizzare nella pratica. A causa della rapida evoluzione della tecnologia nel tempo, ciò richiederebbe un continuo adeguamento delle leggi definite. Tuttavia, esiste una visione comune della tecnologia all'avanguardia che viene pubblicata, ad esempio, da istituzioni come TeleTrust [TELETRUST] o NIST e che viene adattata nel tempo.

Le normative sono spesso poco specifiche perché mirano a coprire un ampio spettro di settori industriali. Attenersi a un insieme fisso di tecnologie di sicurezza IT potrebbe causare problemi, poiché alcune tecnologie potrebbero diventare inapplicabili per determinate organizzazioni.

Sebbene l'uso della tecnologia attuale sia una parte delle normative, è necessario considerare anche i seguenti tre pilastri fondamentali:

- Risorse (ad esempio, hardware, software e tecnologia all'avanguardia)

- Personale
- Processi di sicurezza delle informazioni

Per quanto riguarda il personale, ci sono quattro aspetti principali da considerare:

- Il personale deve essere consapevole dell'importanza della sicurezza delle informazioni e deve comprendere che è responsabile di garantirne la sicurezza
- Deve possedere le conoscenze necessarie per implementare e applicare le misure di sicurezza definite, tenendo presente che il tipo e la specificità possono variare a seconda dei diversi ruoli. Sono necessarie competenze e conoscenze relative alle politiche e alle procedure di sicurezza effettivamente definite
- Deve accettare e applicare i processi di sicurezza delle informazioni definiti (vedere la sezione successiva)
- Alcuni membri del personale necessitano di autorizzazioni speciali

I processi di sicurezza delle informazioni includono aspetti quali:

- Definizione delle responsabilità. Nel contesto normativo, questo influisce sui ruoli e sulle responsabilità all'interno di un'organizzazione, nonché sulle istituzioni responsabili del monitoraggio della conformità e del reporting degli incidenti, come il National Cyber Emergency Response. Spesso una persona viene nominata come unico punto di contatto all'interno di un'organizzazione. Deve definire quando e a chi deve presentare i report
- Come gestire i nuovi assunti o i dipendenti in uscita, il personale che cambia reparto o i dipendenti esterni
- In caso di allerta di sicurezza, chi deve essere informato, chi sarà responsabile delle decisioni e quando l'organizzazione deve segnalare l'incidente a un'istituzione (federale)
- Come comportarsi con i partner di business e i fornitori
- Review e rivalutazioni periodiche dei processi e delle misure attualmente definiti
- Procedure di auditing, compresi i preparativi e le correzioni dopo lo svolgimento di un audit

Gli aspetti sopra descritti sono parte integrante di un ISMS funzionante. Alcune normative mirano a stabilire e preservare un ISMS essenziale (vedere il capitolo 8).

Come testare le politiche di sicurezza

I STE hanno un'elevata responsabilità nel testare le politiche di sicurezza per le organizzazioni soggette alle normative. Le ragioni principali sono:

1. Dal punto di vista di un'organizzazione, è importante essere conformi. In caso contrario, l'organizzazione potrebbe dover pagare sanzioni o rischiare la chiusura temporanea della propria attività.
2. Poiché gli incidenti nei settori industriali regolamentati possono avere conseguenze molto più gravi rispetto ad altri settori, il testing deve essere effettuato in modo molto accurato per garantire un elevato livello di sicurezza.

Il passo successivo consiste nel valutare le best practice e le tecnologie all'avanguardia, poiché le normative potrebbero solo farvi riferimento. I STE devono validare che le misure di sicurezza previste per un SUT siano ancora sufficienti. Ad esempio, devono controllare se i dati sono crittografati e quale algoritmo viene utilizzato, poiché questo potrebbe già essere non sicuro.

Ai fini del testing, i risultati degli audit condotti in precedenza possono essere considerati parte integrante della base di test. Tuttavia, le scoperte di un audit precedente che ora sono state implementate devono essere sottoposte a test di conferma. Il STE non può presumere che l'implementazione sia stata eseguita correttamente. Inoltre, poiché le normative e gli obiettivi chiave di una politica di sicurezza includono persone e processi, anche le attività di test devono includere questi aspetti.

Il testing degli aspetti relativi al personale può essere eseguito applicando test quali simulazioni di attacchi di social engineering o tentativi di eludere i controlli di accesso (fisici). Questi tipi di test dipendono dal reparto dell'organizzazione che ha richiesto il test di sicurezza.

Il testing di sicurezza dei processi include il backup e il ripristino, la risposta alle emergenze e il reporting. Questi test possono essere molto elaborati e costosi, poiché molte persone potrebbero essere coinvolte durante il test.

5.3 Analizzare uno Scenario di Attacco

5.3.1. Gli scenari di attacco comuni

Gli incidenti di sicurezza variano notevolmente in termini di tecniche e strumenti di attacco utilizzati, tipo di attacker e motivazione dell'attacco. Di conseguenza, è difficile fornire una descrizione generica di un attacco. Tuttavia, alcune fasi sono comuni a quasi tutti gli attacchi. Queste fasi possono essere definite come segue:

1. Raccolta di informazioni
2. Sfruttamento/acquisizione dell'accesso
3. Persistenza/mantenimento dell'accesso
4. Cancellazione delle tracce

A titolo di confronto, la gestione degli incidenti di sicurezza è definita secondo il NIST [NIST03] dalle seguenti fasi:

1. Preparazione
2. Rilevamento e analisi
3. Contenimento, eradicazione e ripristino
4. Attività post-incidente

Sebbene entrambe le enumerazioni descrivano una sequenza comune di attacco e risposta, fattori quali la motivazione, le risorse, le competenze dell'attacker e l'approccio utilizzato hanno un forte impatto sia sul successo dell'attacco che sulle conseguenze per la parte attaccata.

Classificazione degli attacker e delle loro motivazioni

Il termine "hacker" è utilizzato in questa sezione come sinonimo di attacker. Tuttavia, il termine hacker si riferisce generalmente a una persona con elevate competenze tecniche.

Gli attacker possono essere suddivisi in diversi tipi, a seconda delle loro competenze tecniche e delle risorse a loro disposizione:

Tipo di attacker	Descrizione
script kiddies	Si tratta di persone con un basso livello di conoscenze tecniche, che utilizzano strumenti e script esistenti senza comprenderli appieno e che dispongono di risorse molto limitate.
scammer	Queste persone utilizzano tecniche semplici come il phishing, ma di solito prendono di mira molte persone, aumentando così le loro possibilità di successo.
hacker privati	Queste persone hanno un solido background tecnico e sono interessate alla sicurezza IT o sono molto curiose.
hacker professionisti	Queste persone hanno un background tecnico molto elevato e sono in grado di eseguire attacchi altamente sofisticati per guadagnarsi da vivere.
governi	Queste organizzazioni pagano interi team per spiare, hackerare o sabotare e dispongono di risorse notevolmente superiori rispetto a quelle di singoli individui o piccoli gruppi.

Sebbene si tratti di una categorizzazione approssimativa basata sulle competenze e sulle risorse, un altro aspetto importante è la motivazione di un attacker. Gli script kiddies potrebbero semplicemente voler giocare con qualcosa che hanno appena trovato online o impressionare i loro amici, mentre gli hacker professionisti guadagnano i loro soldi hackerando e quindi vogliono anche avere una buona reputazione. Di seguito sono elencate le categorie di motivazione:

- Motivazioni personali (ad esempio, fama, vendetta, gelosia e curiosità)
- Motivazioni politiche (ad esempio, "hacktivism", guerra e spionaggio)
- Motivazioni professionali (ad esempio, denaro, reputazione e spionaggio industriale)

A seconda del livello di motivazione, un attacco può essere sferrato contro una singola entità o contro più entità. Ad esempio, il ransomware è un malware che di solito crittografa i dati e blocca l'uso del sistema infetto fino a quando la vittima non paga una certa somma di denaro. Poiché infettare più sistemi aumenta le possibilità di guadagno per l'attacker, il ransomware di solito è scritto in modo da infettare il maggior numero possibile di sistemi.

Al contrario, i worm informatici come Stuxnet [WIKI02] sono sviluppati principalmente per il caso specifico di infettare i sistemi di controllo di supervisione e acquisizione dati. Questi sono stati utilizzati per sabotare i programmi nucleari di interi paesi.

Approccio comune di un attacker

Raccolta di informazioni

La prima fase di un attacco è la raccolta di informazioni, nota anche come ricognizione. Un attacker cerca informazioni sul bersaglio e cerca di trovare debolezze nelle seguenti aree:

- Infrastruttura IT (ad esempio, una vulnerabilità software conosciuta)
- Infrastruttura fisica e relativi meccanismi di controllo degli accessi (ad esempio, irrompere in un ufficio che potrebbe avere un sistema di allarme inadeguato e consentire l'accesso a informazioni sensibili)
- Dipendenti, che potrebbero avere una scarsa consapevolezza della sicurezza
- Processi interni a un'organizzazione che potrebbero essere sfruttabili

La raccolta di informazioni può essere passiva o attiva. La raccolta passiva di informazioni può essere svolta effettuando ricerche sul web utilizzando query di ricerca specializzate, come Google, che possono rivelare una quantità sorprendentemente grande di informazioni. Questa pratica è diventata nota come "Google hacking" o "Google dorking" [WIKI01]. Inoltre, le piattaforme dei social media sono un'importante fonte di informazioni sui dipendenti, in particolare per quanto riguarda i loro numeri di telefono, indirizzi e-mail e altre informazioni personali che possono essere utilizzate per il social engineering. L'utilizzo dei dettagli di una persona consente agli attacker di personalizzare i loro attacchi, come lo spear phishing o l'invio di e-mail personalizzate con allegati dannosi. Questi possono utilizzare, ad esempio, il nome, l'indirizzo o la data di nascita corretti di una persona per creare e-mail con contenuti che suonano molto familiari o intimi alla vittima, aumentando così la probabilità che la vittima apra l'allegato dannoso o clicchi su un link e visiti un sito web dannoso.

La raccolta attiva di informazioni include l'uso di strumenti e tecniche che interagiscono con l'obiettivo, ma aumenta il rischio di essere riconosciuti come attacker. La raccolta di informazioni può essere effettuata in diversi modi:

- Cercando di contattare il personale tramite e-mail o telefono (vishing)
- Cercare informazioni utili, come indirizzi e numeri di telefono, nella spazzatura della vittima [SENG22]. Questa pratica è nota come "dumpster diving"
- Scansioni delle porte
- Fingerprinting del sistema operativo
- Enumerazione (DNS) (vedi sotto)
- Scansione delle vulnerabilità
- Raccolta di informazioni utili disponibili pubblicamente (ad es. OSINT)

Tecniche come l'enumerazione DNS potrebbero rimanere inosservate, mentre altre tecniche come la scansione delle porte o la scansione delle vulnerabilità possono spesso essere facilmente identificate analizzando i file di log dei server o dei firewall. I network intrusion detection system (NIDS), che di solito vengono implementati come misura di sicurezza nell'infrastruttura di rete, analizzano il traffico in entrata alla ricerca di modelli sospetti e generano un allarme di sicurezza se riconoscono un possibile traffico dannoso. In particolare, gli scanner di vulnerabilità hanno una fingerprint di rete facilmente rilevabile. Sebbene l'uso degli scanner possa rivelare molto rapidamente informazioni utili all'attacker, il loro utilizzo aumenta anche il rischio di essere scoperti.

La durata della fase di raccolta delle informazioni può variare notevolmente e dipenderà dalla motivazione dell'attacker. Uno script kiddie può annoiarsi dopo pochi minuti o ore perché vuole solo giocare con uno strumento trovato su Internet, mentre gli attacker motivati politicamente potrebbero raccogliere informazioni per mesi prima di lanciare effettivamente un attacco. I metodi di raccolta attiva delle informazioni lasciano sempre tracce, ma a volte vengono rilevati solo attraverso indagini forensi dopo che un attacco è stato eseguito con successo.

Oltre alla motivazione, anche il tipo di attacco influenza la fase di raccolta delle informazioni. Un attacker che vuole eseguire un attacco denial-of-service che influisce solo sulla disponibilità potrebbe non aver bisogno di introdursi in una rete privata e quindi può ignorare alcune informazioni. Tuttavia, maggiori informazioni aumentano sempre le possibilità di successo di un attacco.

È possibile identificare un sistema vulnerabile enumerando tutti i servizi disponibili e le loro versioni e successivamente effettuando una ricerca nei database di exploit disponibili pubblicamente.

Va notato che i servizi pubblici legali possono automatizzare la scansione attiva e costante dell'intera rete Internet, aggiornando costantemente i propri database. Servizi come questi consentono di cercare dispositivi non protetti che utilizzano password predefinite e servizi vulnerabili. I servizi possono essere utilizzati sia da organizzazioni che da privati, rendendo più facile per chiunque trovare sistemi vulnerabili. Consentono di cercare indirizzi IP, domini e versioni di server web specifici, rendendoli un altro strumento utile per la raccolta di informazioni.

Tuttavia, la raccolta di informazioni può essere effettuata anche offline, attraverso il dumpster diving, l'osservazione, fingendo di essere un cliente o infiltrandosi in un'organizzazione come dipendente. Ciò aumenta notevolmente la possibilità di essere scoperti e quindi nella maggior parte dei casi viene evitato. Solo se un attacker ha una motivazione molto forte o se il tentativo di ottenere informazioni su un obiettivo è già fallito, potrebbero essere prese in considerazione queste attività come approccio per la raccolta di informazioni.

Infine, la raccolta di informazioni è un'attività ricorrente durante un attacco, perché una volta che un attacker ha ottenuto l'accesso a un'infrastruttura non disponibile al pubblico, ha bisogno di continuare ad acquisire informazioni su di essa.

Sfruttamento/Acquisizione dell'accesso

Una volta che gli attacker hanno acquisito una conoscenza ragionevole del loro obiettivo, passeranno all'attacco vero e proprio. In questo contesto, per "conoscenza ragionevole" si intende che hanno effettivamente trovato almeno una possibilità di attacco che avrà un'alta probabilità di successo. Un attacco riuscito può essere causato da:

- Vulnerabilità conosciute in un software non aggiornato
- Configurazione errata (ad esempio, configurazione mancante o sbagliata)
- Exploit zero-day
- Password deboli
- Social engineering

Se un attacco ha successo, l'attacker solitamente non dispone di un account amministratore. Questo potrebbe rappresentare un ostacolo, poiché vorrebbe modificare il sistema per i propri scopi (ad esempio, arrestando o modificando il software antivirus). Pertanto, dopo l'accesso iniziale è spesso necessario acquisire livelli di privilegio più elevati. L'escalation dei privilegi può avere successo per gli stessi motivi dell'attacco iniziale, come una vulnerabilità del software o una configurazione errata. La configurazione errata è una grave minaccia per l'escalation dei privilegi. Sui sistemi UNIX, ad esempio, molti programmi consentono l'accesso a una root shell e, se permettono l'uso di SUDO (acronimo di super user do), possono utilizzarlo per eseguire programmi come super user o altro utente [GTFO22].

Sebbene si tratti di attacchi che possono essere eseguiti da remoto, è anche possibile che un attacker ottenga l'accesso diretto dalla rete interna di un'organizzazione. Un attacker potrebbe essere un dipendente che vuole danneggiare l'organizzazione. Inoltre, un attacker potrebbe ottenere l'accesso fisico a un ufficio a causa di un controllo degli accessi insufficiente e da lì potrebbe essere in grado di connettersi alla rete interna.

Il social engineering è una minaccia che può portare l'attacker ad ottenere un accesso diretto. Secondo la definizione di NIST, si tratta dell'"atto di ingannare un individuo per indurlo a rivelare informazioni sensibili, ottenere un accesso non autorizzato o commettere una frode associandosi all'individuo per ottenere fiducia e affidabilità" [NIST05].

Persistenza/mantenimento dell'accesso

Per ottenere l'accesso non autorizzato a un sistema spesso vengono utilizzati exploit che possono essere difficili da applicare, hanno un'alta probabilità di fallimento o possono essere applicati solo in determinati momenti. Pertanto, gli attacker devono mantenere l'accesso al sistema compromesso fino a quando non hanno raggiunto i loro obiettivi. Anche in questo caso, ciò dipende dalla motivazione degli attacker, poiché alcuni di loro potrebbero essere interessati solo al successo dell'attacco, senza voler andare oltre.

L'accesso persistente viene solitamente ottenuto tramite rootkit, creati appositamente per questo scopo. I rootkit cercano di mantenere l'accesso anche se il sistema viene riavviato. Poiché i software anti-malware cercano di rilevare i software dannosi, i rootkit sono progettati per nascondere sé stessi e altri malware, come keylogger e sniffer di rete, nel sistema. Possono anche abilitare il controllo remoto automatizzato dei sistemi compromessi, consentendo agli attacker di creare botnet per eseguire attacchi distribuiti di tipo denial-of-service contro altri sistemi o di utilizzarli in modo improprio come server di spam.

Cancellazione delle tracce

Poiché gli attacchi possono avere gravi conseguenze legali, gli attacker vogliono rimanere anonimi o almeno non vogliono essere identificati personalmente. Di conseguenza, sono fortemente motivati a rimuovere tutte le tracce delle loro attività precedenti dopo aver raggiunto il loro obiettivo. Questo include la rimozione di tutti i programmi e i file che l'attacker ha copiato sui sistemi compromessi, la cancellazione o la rimozione dei file di log, delle cronologie dei comandi e forse la distruzione dell'hardware utilizzato per l'attacco.

Sebbene si tratti di attività che un attacker svolge al termine di un attacco, gli attacker utilizzano anche tecniche come il proxy chaining (concatenamento di proxy), una VPN o jump server già compromessi durante il loro accesso remoto per nascondere le loro tracce.

Un attacker deve essere consapevole che ogni attività che svolge contro un sistema può essere potenzialmente registrata e, nella maggior parte dei casi, non sarà in grado di cancellare completamente tutte le sue tracce.

Nel contesto della sicurezza e del testing di penetrazione, è necessario agire in modo simile, poiché il SUT potrebbe essere un IDS o un processo di risposta alle emergenze.

Risposta agli incidenti e analisi post-incidente

Le sezioni precedenti hanno descritto uno scenario di attacco dal punto di vista dell'attacker. D'altra parte, le organizzazioni investono in misure di sicurezza per rilevare e risolvere gli incidenti di sicurezza, un'attività nota come risposta agli incidenti. Si noti che la risposta agli incidenti e le fasi di attacco descritte in precedenza di solito avvengono contemporaneamente. In molti casi, un attacker è già stato rilevato in una fase precedente (ad esempio, la fase di persistenza) e non solo dopo aver già cercato di cancellare le proprie tracce.

Preparazione

Sebbene la preparazione faccia parte della gestione degli incidenti, non fa parte della risposta agli incidenti, ma costituisce la base per una procedura di risposta agli incidenti funzionante che verrà attuata in caso di incidente di sicurezza.

La risposta agli incidenti mira a raggiungere i seguenti obiettivi:

- Identificare un incidente e analizzare la situazione
- Contenimento, ad esempio isolare i sistemi compromessi e interrompere i servizi
- Eradicazione, ad esempio rimuovere gli account utente compromessi, rimuovere il malware e applicare patch al sistema
- Ripristino, ad esempio riportare i servizi online e ripristinare i dati

Una volta risolto l'incidente, è necessario eseguire una review per identificare le debolezze dell'infrastruttura e dei processi di risposta agli incidenti.

Rilevamento e analisi

Il rilevamento di un incidente può essere intenzionale o non intenzionale da parte dell'attacker. Nel caso di un defacement di un sito web, ad esempio, è intenzione dell'attacker che l'attacco venga riconosciuto. In altri casi, potrebbero lasciare un messaggio all'amministratore di sistema.

È possibile utilizzare diversi strumenti come aiuto nell'individuazione di attività sospette. Tra questi figurano i network/host intrusion detection system (NIDS/HIDS), gli scanner di malware e gli analizzatori di log. Nel migliore dei casi, è possibile identificare un attacco prima che l'attacker riesca nel suo intento. Ciò può verificarsi se l'attacker è troppo evidente durante la scansione o se i suoi primi tentativi di exploit falliscono. Inoltre, molti tentativi di login falliti o tentativi di login da parte di utenti inesistenti o che di solito non effettuano il login da remoto possono essere indice di un potenziale attacco. Se tutto questo può essere rilevato al di fuori della rete dell'organizzazione, ci sono buone possibilità di sventare l'attacco.

Se vengono rilevate attività sospette all'interno della rete dell'organizzazione, è necessario analizzare quali sistemi sono già stati compromessi e in che modo l'attacker ha ottenuto l'accesso. Questo può essere fatto dal reparto IT, ma più comunemente è un team di esperti forensi che inizia ad analizzare l'incidente. La risposta a un incidente è critica in termini di tempo e le azioni devono essere intraprese il prima possibile per evitare ulteriori danni.

Contenimento, eradicazione e ripristino

Se si ha un quadro chiaro dei sistemi compromessi, il passo successivo è contenere tali sistemi per impedire all'attacker di comprometterne altri. Questo può essere ottenuto, ad esempio, chiudendo temporaneamente i servizi, spostandoli su un'altra rete o bloccando gli account degli utenti.

Un aspetto importante da considerare durante questa fase è che alcune azioni potrebbero cancellare prove forensi utili per l'analisi post-incidente. Ad esempio, l'arresto di un sistema cancellerà la memoria principale, che potrebbe contenere dati utili come l'exploit iniziale utilizzato. Anche in questo caso, è fondamentale reagire rapidamente, poiché un attacker potrebbe compromettere altri sistemi. Le azioni da intraprendere devono essere decise in base al livello di rischio.

Il contenimento e l'analisi della situazione si alterneranno a un certo punto, poiché sarà necessario rivalutare se tutti i sistemi sono stati identificati correttamente e se l'attacker non è in grado di ottenere ulteriore accesso. Se vi è sufficiente certezza che tutti i sistemi interessati sono stati identificati e contenuti, è possibile procedere alla eradicazione. Un aspetto importante durante questa fase è fornire prove per ulteriori analisi. L'eradicazione può includere la cancellazione di un intero sistema e la sua successiva ricreazione da un backup. È anche possibile rimuovere solo componenti parziali del sistema e sostituirli durante la fase di ripristino con un nuovo componente. In entrambi i casi, è necessario assicurarsi che il backup o il componente utilizzato non contenga tracce dell'incidente precedentemente risolto.

Attività post-incidente

Dopo aver risolto un incidente, è necessario adottare diverse misure per valutare e migliorare le attuali routine di sicurezza. Queste includono:

- Indagini forensi, che nel migliore dei casi possono identificare l'attacker
- Chiudere le vulnerabilità che potrebbero essere state rivelate dall'attacco
- Rivalutare l'infrastruttura attuale
- Aumentare la consapevolezza dei dipendenti in materia di sicurezza
- Rivalutare e, eventualmente, adattare le politiche di sicurezza
- Affinare i processi di risposta agli incidenti
- Comunicare la situazione a clienti o consumatori, soprattutto se l'organizzazione ha obblighi di reporting. Si include la segnalazione all'istituzione competente o al governo.

6 Adattare il Testing di Sicurezza ai Modelli di Ciclo di Vita dello Sviluppo Software - 165 minuti (K4)

Parole Chiave

ciclo di vita dello sviluppo software

Parole Chiave Relative alla Sicurezza

nessuna

Obiettivi di Apprendimento per il Capitolo 6:

6.1 Gli Effetti dei Diversi Modelli di Ciclo di Vita dello Sviluppo Software sul Testing di Sicurezza

STE-6.1.1 (K2) Riassumere perché le attività di testing di sicurezza dovrebbero coprire il ciclo di vita dello sviluppo software

STE-6.1.2 (K4) Analizzare in che modo le attività di testing di sicurezza sono influenzate dai diversi modelli di ciclo di vita dello sviluppo software

6.2 Il Testing di Sicurezza Durante la Manutenzione

STE-6.2.1 (K3) Definire ed eseguire regression test e test confermativi di sicurezza sulla base di una modifica al sistema

STE-6.2.2 (K2) Analizzare i risultati dei test di sicurezza per determinare la natura di una vulnerabilità e il suo potenziale impatto tecnico

6.1 Gli Effetti dei Diversi Modelli di Ciclo di Vita dello Sviluppo Software sul Testing di Sicurezza

Il ciclo di vita dell'applicazione o del sistema può essere descritto come un modello con diversi SDLC. Le fasi del SDLC più utilizzate sono pianificazione, analisi, progettazione, sviluppo, test, implementazione, manutenzione e disattivazione.

Le attività e i compiti pianificati per ciascuna di queste fasi possono variare a seconda dell'applicazione, del sistema, del progetto o dell'organizzazione. Questi sono definiti nel modello di SDLC che può essere implementato utilizzando un approccio di sviluppo sequenziale o di sviluppo Agile.

Utilizzando un approccio di sviluppo sequenziale, è più facile riconoscere le diverse fasi del SDLC. Con l'approccio Agile potrebbe non essere così chiaro quando e quale attività o compito di quale processo del ciclo di vita viene eseguito. Le attività e i compiti possono essere ripetuti frequentemente, aggiungendo valore all'applicazione o al sistema con ogni singola iterazione.

Nel Syllabus ISTQB Certified Tester Foundation Level [ISTQB FL] vengono menzionati diversi modelli di sviluppo, tra cui il modello waterfall e i modelli di sviluppo software Agile (ad esempio, Rational Unified Process, Scrum, Kanban e Spiral). Si afferma che il testing di sicurezza dovrebbe essere adattato a questi modelli per essere più efficace. Come ci si può aspettare, anche gli approcci al testing di sicurezza devono essere adattati ai diversi modelli di SDLC. Questo Syllabus tratta DevOps oltre ai modelli sopra descritti.

Il STE dovrebbe conoscere le caratteristiche più importanti di questi modelli di SDLC e il modo in cui possono influire sulla sua capacità di eseguire il testing di sicurezza.

Confrontando queste categorie, si possono osservare differenze relative ai seguenti attributi. Una modifica di uno qualsiasi di questi attributi avrà anche un impatto sul modo in cui viene eseguito il testing di sicurezza.

Attributo	Descrizione
Durata dello sviluppo	Il tempo necessario dalla formulazione di un requisito fino al deployment • La durata influirà anche sul tempo consentito per eseguire il testing di sicurezza durante il SDLC • Meno tempo è disponibile, più difficili saranno le scelte da prendere e le priorità da stabilire
Dimensioni del deployment	Batch più grandi di funzionalità o una singola funzionalità per deployment • Spesso in relazione diretta con la durata dello sviluppo • Quanto più piccola è la dimensione di un deployment, tanto più specifico ci si può (e si deve) focalizzare sul testing di sicurezza

Attributo	Descrizione
	Con deployment più grandi, la superficie di attacco può aumentare notevolmente in una sola iterazione. La necessità di regression testing aumenta nei modelli di sviluppo incrementale
Tempo di testing consentito	La quantità risorse di tempo riservata per eseguire il testing - Nella maggior parte dei casi, il testing funzionale può essere pianificato, ma il testing non funzionale (compreso il testing di sicurezza) spesso non viene pianificato nei cicli di vita dei progetti - Se il tempo lo consente, è possibile creare opportunità per un testing non funzionale più approfondito, compreso il testing di sicurezza
Indipendenza del team	Il livello delle decisioni in materia di sicurezza che possono essere prese dal team Può consentire al team di assegnare o assumere competenze autonome in materia di testing di sicurezza, se necessario
Interfunzionalità del team	La disponibilità delle competenze necessarie (per lo sviluppo) nel team Il team può disporre di competenze di testing di sicurezza diverse e complementari
Livello di automazione	Quanto è automatizzato il processo di sviluppo Gran parte del testing di sicurezza può essere eseguito con la test automation utilizzando attività correlate a SAST e DAST
Principi di gestione	L'organizzazione è orientata alla linea, al progetto o al prodotto? I principi di gestione del team possono influenzare il modo in cui viene organizzato il testing di sicurezza. Questo può essere effettuato con team abilitati, attenzione continua alla sicurezza, solo durante la fase di progetto o solo durante la manutenzione.
Ambiente di test	È possibile stabilire un ambiente di test separato e dedicato per il testing di sicurezza distruttivo.

6.1.1. Modelli di sviluppo sequenziale

Il SDLC completo è spesso specificato da tutti i processi necessari per sviluppare, mantenere e supportare il sistema dal suo avvio iniziale fino al suo ritiro/smaltimento. Uno standard molto utilizzato che descrive tutti questi processi e le loro relazioni è [ISO 15288].

Il modello di sviluppo del sistema descrive l'implementazione (di parti) del SDLC necessario per sviluppare e implementare un sistema. Questa implementazione non include necessariamente tutti i processi del SDLC e dovrebbe essere adattata all'organizzazione (vedere il capitolo 5) e al contesto del progetto.

La sicurezza delle informazioni (information security) e la verifica della sicurezza dovrebbero essere un aspetto integrato coperto in tutti i processi del SDLC utilizzati nell'organizzazione. Solo così è possibile ottenere un approccio veramente olistico. Ciò consentirà inoltre di supportare o garantire che le attività richieste durante i processi di sviluppo possano essere condotte in modo coerente.

In [NIST 800-160] la sfida della sicurezza consiste nel descrivere la sicurezza del sistema come un problema di progettazione. Si osserva che "per una sicurezza completa è necessaria una combinazione di misure di sicurezza hardware, software, di comunicazioni, fisiche, di personale e amministrative-procedurali. Le misure di sicurezza software da sole non sono sufficienti".

Questo standard NIST presenta considerazioni e approcci che coprono tutti i processi del SDLC su come affrontare le attività relative alle informazioni di sicurezza.

Il modello di sviluppo sequenziale Waterfall o la sua implementazione come V-model sono ancora modelli molto utilizzati. Il V-model fa riferimento ad attività di testing generiche per ciascuna delle sue fasi con l'obiettivo di uno "shift left". Questi modelli sono più consolidati nelle organizzazioni con una netta separazione tra i diversi team e le fasi di sviluppo. In generale, quando si utilizza questo modello di sviluppo software, ci si può aspettare che il STE abbia il tempo di pianificare, preparare ed eseguire il testing di sicurezza. Le fasi del modello sono schedulate in sequenza, ma possono sovrapporsi tra loro.

In questi modelli, il STE dovrebbe essere consapevole di quanto segue:

- I requisiti e i rischi di sicurezza sono definiti nelle prime fasi di un progetto e dovrebbero essere documentati nelle specifiche dei requisiti software.
- I requisiti di sicurezza possono cambiare nel corso del progetto man mano che vengono scoperte nuove minacce, ma questi potrebbero non essere riflessi nei requisiti software aggiornati. Il testing di sicurezza può quindi sembrare molto specifico e completo, ma potrebbe non essere effettivamente completo o aggiornato a causa dei rischi legati alle fasi avanzate del progetto.
- Il testing di sicurezza può essere eseguito in qualsiasi momento o in qualsiasi fase di sviluppo, ma è comune che venga eseguito nelle fasi avanzate del progetto.
- Può essere difficile affrontare i risultati del testing di sicurezza e delle correzioni alla fine di un progetto con un modello di sviluppo sequenziale, poiché le scadenze saranno per lo più fissate in una fase iniziale del progetto.

6.1.2. Sviluppo software Agile

Lo sviluppo software Agile promuove il completamento del lavoro da svolgere da parte di team auto-organizzati e interfunzionali in brevi iterazioni. Gli enabler team che forniscono servizi specifici al progetto

possono essere a disposizione di questi team di sviluppo software Agile per assisterli con competenze specifiche del dominio, come il testing di sicurezza.

Una caratteristica generica dello sviluppo software Agile è che gli incrementi (di sistema e software) vengono forniti in una serie di iterazioni. Ognuna di queste iterazioni può richiedere da alcuni giorni ad alcune settimane. I modelli di sviluppo software Agile tendono ad essere utilizzati principalmente nella fase di sviluppo dell'applicazione/sistema, anche se alcuni modelli, come Kanban, possono essere applicati anche durante la fase operativa.

Il framework Scrum nelle varie implementazioni è il più utilizzato nello sviluppo software Agile. Tutte le attività di analisi, progettazione, codifica e testing vengono eseguite durante ogni iterazione, compreso il testing di sicurezza.

I product backlog fungono, almeno in parte, da specifiche dei requisiti. Sia i requisiti di sicurezza che altri requisiti non funzionali dovrebbero far parte del product backlog. Gli epic sono suddivisi in diverse user story e attività che vengono selezionate dal team o dai team per essere sviluppate o consegnate in uno degli sprint.

Le funzionalità sviluppate possono essere modificate o addirittura eliminate negli sprint futuri. La funzionalità "in crescita" e le modifiche future o addirittura le eliminazioni rendono instabile la base di test su cui lavorare. Lo sviluppo software Agile può essere visto come un mix di sviluppo (nuove funzionalità) e manutenzione (piattaforma e funzionalità esistenti) durante la fase di progetto. È quindi essenziale ripetere i test di sicurezza automatizzati.

È possibile adottare diversi approcci per eseguire le attività di testing di sicurezza. Alcuni esempi sono:

- Eseguire alcuni test di sicurezza e poi spostare l'attenzione su oggetti di test funzionali, tecnici o relativi alla piattaforma in ogni singolo sprint
- Eseguire alcuni test di sicurezza nella maggior parte degli sprint ed eseguire un test di sicurezza completo in uno sprint dedicato
- Eseguire tutti i test di sicurezza in un unico sprint (tardivo) che assomiglia al modello di sviluppo sequenziale

Il team di sviluppo Agile può coinvolgere un enabler team o assumere risorse per eseguire il testing di sicurezza, poiché queste competenze spesso non sono presenti nel team.

Poiché la soluzione cambia con ogni sprint, è necessario eseguire il regression testing. La sicurezza non viene testata o integrata in un'applicazione già costruita. Piuttosto, viene ottenuta attraverso una progettazione orientata alla sicurezza (cioè security by design) e la verifica durante tutto il processo di costruzione.

[Synopsys] ha descritto come il testing di sicurezza può essere applicato nello sviluppo software Agile applicando i seguenti quattro principi definiti nel Manifesto Agile:

- Sviluppatori e tester piuttosto che specialisti della sicurezza
- Mettere in sicurezza durante il lavoro piuttosto che mettere in sicurezza dopo il completamento
- Implementare le funzionalità in modo sicuro piuttosto che aggiungere funzionalità di sicurezza
- Mitigare i rischi piuttosto che correggere i difetti

Sviluppatori e tester piuttosto che specialisti della sicurezza

Gli specialisti della sicurezza con esperienza sono risorse preziose. I team Agile raramente hanno il lusso di poter contare su specialisti della sicurezza dedicati. Ciò significa che, nella maggior parte dei casi, i team Agile dovrebbero essere responsabili della propria sicurezza e non possono attendere una review esterna della sicurezza prima che il codice passi alla fase di sviluppo successiva. La sicurezza deve essere integrata nello sviluppo e nel testing del codice. I team devono occuparsi della sicurezza allo stesso modo in cui si occupano della user experience, dell'affidabilità, dell'efficienza delle prestazioni e di altri requisiti non funzionali.

Mettere in sicurezza durante il lavoro piuttosto che mettere in sicurezza dopo il completamento

È fondamentale applicare metodologie e pratiche sicure durante la creazione, il rilascio e la manutenzione di software funzionale. Allo stesso tempo, le attività di sicurezza non dovrebbero costringere gli sviluppatori a interrompere il loro lavoro, passare a un altro strumento per la correzione e poi tornare a quello che stavano facendo. L'alternativa è integrare il feedback e le informazioni sulla sicurezza negli strumenti degli sviluppatori. Le attività di sicurezza vengono presentate (ad esempio su lavagne) e vengono stabilite delle priorità per renderle visibili insieme alle altre attività.

Implementare le funzionalità in modo sicuro piuttosto che aggiungere funzionalità di sicurezza

L'attenzione dovrebbe focalizzarsi sulla realizzazione della missione di business del software, ma l'integrazione della sicurezza dovrebbe sempre essere presa in considerazione. Ciò significa che architetti, sviluppatori, tester e altri stakeholder devono considerare gli aspetti di sicurezza e collaborare per definire e costruire sistemi più sicuri. I sistemi sicuri dovrebbero essere progettati e costruiti fin dall'inizio.

Mitigare i rischi piuttosto che correggere i difetti

È necessario considerare i rischi specifici del business, degli utenti, dei dati e del software. Il risk management considera il modo corretto di affrontare un rischio. Questo può essere ottenuto adottando una visione di alto livello di ciò che potrebbe andare storto, invece di ridurre la sicurezza a un lungo elenco di singoli difetti che devono essere risolti. Sebbene la modellazione delle minacce sia più difficile della semplice localizzazione e correzione dei difetti, è un approccio efficace per individuare i problemi nelle prime fasi del SDLC. È sicuramente più economico quando i problemi possono essere risolti prima del rilascio del software.

6.1.3. La metodologia DevOps

La maggior parte dello sviluppo software Agile copre la consegna del sistema o del software al reparto operativo. DevOps va oltre includendo lo sviluppo e le operazioni. L'obiettivo principale di DevOps è fornire rapidamente (piccole) modifiche. Inoltre, la cultura del team ha un impatto molto maggiore sul successo del team. I team DevOps sono generalmente più autonomi e più orientati al prodotto rispetto ad altri team.

DevSecOps copre l'intero SDLC, inclusi sviluppo, sicurezza e operazioni. Durante lo sviluppo, la sicurezza si focalizza sull'identificazione e la prevenzione delle vulnerabilità, mentre nelle operazioni gli obiettivi principali sono il monitoraggio e la difesa dagli attacchi.

In generale, le iterazioni DevOps possono durare anche solo un'ora e le consegne sono tipicamente singole funzionalità/attività di sviluppo o piccoli rami. Il team DevOps mira a rendere disponibili i risultati

dei test quasi immediatamente dopo aver apportato una modifica in ogni fase del processo di sviluppo. Questo esercita una forte pressione sul testing e sulle metodologie in uso, il che a sua volta ha un effetto importante sulle possibilità di eseguire il testing di sicurezza.

Per ottenere iterazioni di sviluppo DevOps brevi, molte delle attività ripetitive e che richiedono molte risorse vengono automatizzate. Questo avviene sotto forma di una pipeline composta da una serie di fasi, ciascuna delle quali può contenere uno o più lavori relativi all'esecuzione di attività specifiche nel processo di costruzione e deployment. Una fase della pipeline può essere denominata testing di sistema e può includere un processo che esegue regression test automatizzati. Questa pipeline viene quindi eseguita per ogni funzionalità da consegnare.

DevOps è disponibile in diverse varianti. I due approcci più utilizzati sono:

- Utilizzo di un ramo principale o master. Le modifiche vengono sviluppate e testate in un ramo di funzionalità o trunk di breve durata e distribuite direttamente in produzione dopo l'approvazione. Questo approccio è noto anche come trunk-based development.
- L'utilizzo di un ramo di sviluppo separato consente di consegnare continuamente le modifiche in un ambiente di test e di distribuirle insieme in un piccolo batch dopo un breve periodo. Questo approccio è noto anche come feature-based development.

Le attività di testing di sicurezza richiedono tempo. Una domanda comune a cui rispondere riguarda la decisione se eseguire i test di sicurezza per ogni pipeline o se schedarli durante la notte dopo aver eseguito alcune pipeline.

DevSecOps è un concetto che indica l'importanza attribuita al testing di sicurezza in DevOps. È spesso considerato come una serie di attività di testing di sicurezza eseguite automaticamente nella pipeline e include sia l'analisi statica (cioè lo shift left) sia un'attenzione particolare al monitoraggio e alla prevenzione relativi alla sicurezza, come la formazione sulla sicurezza e la scrittura di codici sicuri.

L'enfasi è posta sulla sicurezza come responsabilità del team, considerando la sicurezza parte integrante di tutte le attività di sviluppo durante tutte le fasi per tutti i membri del team.

Le sfide comuni nell'implementazione del testing di sicurezza utilizzando la metodologia DevOps sono:

- Il testing di sicurezza è ancora considerato un'attività specialistica che deve essere svolta da risorse specifiche. Ciò ostacola la tanto necessaria integrazione del testing di sicurezza all'interno del team DevOps.
- La sicurezza può diventare una priorità eccessiva, con il risultato che altre caratteristiche di qualità, come l'efficienza delle prestazioni e l'usabilità, vengono trascurate. La sicurezza è importante, ma deve essere implementata con un approccio equilibrato.
- La sicurezza può portare all'inibizione della creatività degli sviluppatori, dell'autonomia del team e della possibilità di sperimentare. Questi attributi sono considerati essenziali per un'implementazione riuscita di DevOps.

Il Phoenix Project [TechTarget] descrive le seguenti pratiche necessarie per un'implementazione riuscita di DevOps. Queste pratiche dovrebbero essere applicate anche al testing di sicurezza:

- Creare e mantenere un flusso di attività
Il testing di sicurezza è spesso considerato come un'attività unica o di grandi dimensioni (ad

esempio, test delle applicazioni, scansione della rete e review dell'architettura). Quando si applica DevOps, è necessario pianificare, preparare ed eseguire il testing di sicurezza in attività più piccole. Queste dovrebbero avanzare (flusso) allo stesso modo delle altre attività di sviluppo, applicando concetti quali il rendere visibili le attività, la limitazione del lavoro in corso, l'assegnazione di singole attività a singole persone e l'automazione, ove possibile.

- **Garantire un feedback immediato**
I risultati dei test dovrebbero essere disponibili il più rapidamente possibile. Devono essere comprensibili e risolvibili. La capacità di raggiungere questo obiettivo è supportata dal flusso sopra menzionato e dall'utilizzo di attività più piccole. Questa pratica contribuisce anche a ridurre il debito tecnico.
- **Promuovere una cultura della sicurezza DevOps**
Il team deve essere aperto e trasparente riguardo alle questioni di sicurezza. I membri devono essere motivati a segnalare i problemi relativi alla sicurezza e considerare la sicurezza come una responsabilità del team.

Concettualmente, DevOps consente di fallire, imparare e migliorare. Quando si introduce DevOps, è considerato preferibile iniziare con alcuni piccoli miglioramenti e avviare il flusso.

Per essere efficiente, il Security Test Engineer dovrebbe integrare tutte le attività necessarie relative alla sicurezza nella pipeline DevOps (CI/CD) descritta sopra. Significa non solo focalizzarsi sul testing di sicurezza, ma anche formulare e migliorare epic, user story e attività durante la fase di pianificazione.

6.2 Il Testing di Sicurezza Durante le Operazioni e la Manutenzione

6.2.1. Il regression testing e il testing confermativo di sicurezza

Il testing di sicurezza continua anche dopo che un sistema è stato messo in produzione. Possono verificarsi cambiamenti nell'ambiente tecnico, nei sistemi esterni e nelle integrazioni del SUT. I cambiamenti possono essere dovuti a regolari aggiornamenti di sicurezza o ad altre modifiche nel middleware, nel firmware e nell'hardware. Inoltre, il SUT sarà soggetto a modifiche pianificate e non pianificate che potrebbero aprire nuove vulnerabilità e possibili attacchi.

Tutte queste modifiche richiedono almeno un regression testing della sicurezza periodico. A seconda dell'entità delle modifiche, potrebbe essere necessario un nuovo test di sicurezza.

I test di sicurezza potrebbero comportare la verifica che il sistema continui a resistere con successo ai tentativi di eludere i controlli di sicurezza stabiliti. I miglioramenti all'usabilità o all'efficienza delle prestazioni sono particolarmente soggetti a influire negativamente sui controlli di sicurezza.

Il regression testing di sicurezza dovrebbe focalizzarsi sulla conferma del soddisfacimento di tutti i requisiti di sicurezza e sul testing di nuove vulnerabilità che potrebbero essere state introdotte durante le attività di manutenzione.

Il regression testing viene spesso applicato con una serie di test case basati sul testing di singole funzioni. Tuttavia, per il testing di sicurezza, spesso non è sufficiente rilevare i difetti di regressione con un impatto sulla sicurezza. Gli scenari del regression testing end-to-end sono più robusti e forniscono un livello più elevato di fiducia nella possibilità di eseguire transazioni complete in modo sicuro. Per questo

tipo di regression test, è necessario definire una serie di condizioni di test di sicurezza e testarle ogni volta che viene apportata una modifica al sistema. I difetti di regressione possono derivare da modifiche a tutti i parametri rilevanti del sistema. Alcuni dei difetti di regressione possono avere un impatto sulla sicurezza.

Dopo che un sistema è stato messo in produzione, potrebbe essere necessario un ulteriore effort di sviluppo per correggere i difetti nella versione rilasciata (cioè manutenzione correttiva), per adattarsi ad altre modifiche nell'ambiente operativo (cioè manutenzione adattiva) o per estendere o migliorare le funzionalità (cioè manutenzione perfettiva).

La prospettiva dei test di sicurezza per la manutenzione del sistema si focalizza sul testing delle modifiche apportate per correggere i difetti e la funzionalità principale. Lo scopo è:

- garantire che non siano state introdotte nuove vulnerabilità nel SUT
- verificare che le difese di sicurezza esistenti siano ancora efficaci dopo una modifica

Parte del processo di manutenzione consiste nel mantenere aggiornati i firewall e le altre tecnologie di sicurezza. Il monitoraggio continuo del sistema può rilevare attività sospette che potrebbero richiedere un intervento immediato.

7 Il Testing di Sicurezza come Parte di un Information Security Management System - 105 minuti (K3)

Parole Chiave

nessuna

Parole Chiave Relative alla Sicurezza

Information Security Management System (ISMS)

Obiettivi di Apprendimento per il Capitolo 7:

7.1 Criteri di Accettazione per il Testing di Sicurezza

STE-7.1.1 (K2) Comprendere i criteri di accettazione del testing di sicurezza e come questi influenzano la scelta degli approcci e delle tecniche di test di sicurezza

7.2 Input per un Information Security Management System

STE-7.2.1 (K2) Comprendere il ruolo del testing di sicurezza per un information security management system efficace

7.3 Migliorare un Information Security Management System Tramite l'Adeguamento del Testing di Sicurezza

STE-7.3.1 (K3) Valutare la maturità dell'information security management system introducendo diversi approcci di test, nuovi oggetti di test o una copertura migliorata

STE-7.3.2 (K2) Comprendere la misurabilità all'interno di un information security management system

7.1 Criteri di Accettazione per il Testing di Sicurezza

Il testing di sicurezza può essere applicato come attività ad hoc una tantum per un sistema prima della messa in produzione o come processo continuo e sistematico durante lo sviluppo. Entrambi i tipi generano risultati di test, ma la loro capacità di fornire prove di rischi significativi per la sicurezza varia notevolmente. Lo stesso vale per le diverse tecniche di test di sicurezza. Ad esempio, il testing di sicurezza white-box genera risultati di test e potrebbe identificare vulnerabilità diverse da quelle generate dal testing di sicurezza black-box.

Proprio come per l'ingegneria del software in generale, i requisiti per i test di sicurezza sono raramente definiti in modo chiaro, completo, accurato o coerente. Avviare il testing di sicurezza in base a requisiti così poco definiti potrebbe generare alcuni risultati di test, ma il loro valore dipende dalla qualità dei requisiti che non è predefinita. Qualsiasi azione basata su tali requisiti è probabilmente rischiosa per i seguenti motivi:

- **Tecniche di test discutibili:**
Il testing di sicurezza utilizza una tecnica di test specifica o una combinazione di tecniche diverse, ciascuna delle quali presenta punti di forza e debolezze. Non esiste una tecnica di test migliore in assoluto, ma esistono alcune preferenze per il raggiungimento di un determinato obiettivo di test. Senza obiettivi di test, tutte le tecniche di test potrebbero soddisfare le aspettative, ma senza alcuna garanzia di creare un valore aggiunto.
- **Copertura discutibile:**
La maggior parte delle tecniche di test non ha una definizione predefinita di completezza, ma richiede metriche ben definite che devono essere soddisfatte affinché il test possa essere considerato completato. Il tipo di metriche e le loro soglie dipendono dagli obiettivi del test del test di sicurezza. Senza di esse, il STE potrebbe raggiungere un livello di copertura che potrebbe non soddisfare gli obiettivi del test.

Per evitare queste insidie, è essenziale definire i criteri di accettazione prima di qualsiasi test di sicurezza. Questi devono essere soddisfatti prima di utilizzare i risultati dei test come base per identificare eventuali deviazioni o azioni da intraprendere.

La parola "accettato" è fondamentale. ([WaCh90]) afferma che ciò significa "che i prodotti software provvisori e finali vengono esaminati per determinare se soddisfano criteri specifici. Se lo fanno, allora hanno superato l'accettazione". Naturalmente, i requisiti di sicurezza dovrebbero avere i propri criteri di accettazione (cfr. ([WaCh90])), che supportano la decisione di accettazione di rifiutare, accettare parzialmente o accettare.

Il testing di sicurezza può essere particolarmente adatto per controllare i criteri di accettazione della sicurezza definiti per un SUT. I risultati dei test, che di solito sono aggregati in un test report, dovrebbero contenere tutte le informazioni necessarie per consentire una decisione di accettazione. Per supportare questo processo decisionale, l'approccio del test di sicurezza selezionato dovrebbe basarsi sui criteri di accettazione specifici. Di solito questo viene fatto nei seguenti passaggi:

- Leggere attentamente i criteri di accettazione della sicurezza

- Elencare le possibili tecniche di test di sicurezza (vedere il capitolo 2) che possono essere utilizzate per supportare la decisione di accettazione. Va tenuto presente che alcune tecniche di test di sicurezza potrebbero coprire da zero a molti criteri di accettazione
- Creare una suite specifica di test di sicurezza per questi criteri di accettazione specifici. I principi guida per questo passaggio sono:
 - Possibilità di applicazione:
È possibile utilizzare una tecnica di test specifica per il SUT? (ad esempio, sono disponibili strumenti corrispondenti?)
 - Ottimizzazione di costi, tempi e qualità:
La sfida consiste nel definire una test suite specifica e strumenti che generino risultati di test significativi, che possano essere applicati entro un determinato lasso di tempo e che siano convenienti dal punto di vista economico.

Dopo aver selezionato le tecniche di test e gli strumenti di sicurezza migliori, è necessario applicare il test di sicurezza e analizzare i risultati del test riportandoli nel test report. Il test report stesso dovrebbe riflettere i criteri di accettazione e costituire la base per la decisione di accettazione della sicurezza.

7.2 Input per un Information Security Management System (ISMS)

Il testing di per sé non migliora la qualità. Il testing fornisce informazioni sulla qualità raggiunta per una specifica caratteristica di qualità, come la sicurezza. Un test report non migliora il livello di sicurezza. Se le scoperte del test report vengono analizzate e la maggior parte di esse viene risolta, potrebbe essere opportuno un test confermativo per dimostrare un aumento della sicurezza. Oltre a queste azioni di mitigazione del rischio specifiche del sistema, alcune scoperte potrebbero motivare una politica di sicurezza specifica per evitare tali vulnerabilità in futuro. D'altra parte, alcune di queste scoperte potrebbero avere origine in una mancanza di consapevolezza della sicurezza o nell'uso di tecniche immature/non sistematiche.

Per utilizzare il testing di sicurezza in modo efficace ed efficiente, è necessario integrarlo in un processo di sicurezza complessivo. È necessario cercare di ridurre al minimo i rischi e garantire la continuità del business limitando in modo proattivo l'impatto di una violazione della sicurezza. Questo è precisamente l'obiettivo di un ISMS. [ITGov23b] definisce un ISMS come segue:

- Un ISMS adotta un approccio sistematico per garantire la confidenzialità, l'integrità e la disponibilità degli asset informativi aziendali.
- Un ISMS ISO 27001 è costituito da politiche, procedure e altri controlli che coinvolgono persone, processi e tecnologia.
- Un ISMS è un modo efficiente per proteggere gli asset informativi, basato su valutazioni regolari dei rischi e approcci neutrali rispetto alla tecnologia e ai fornitori.

Un ISMS adotta una visione olistica della sicurezza e garantisce l'interazione efficace dei tre attributi chiave della sicurezza delle informazioni (information security):

- Processo
- Tecnologia

- Comportamento all'interno dell'organizzazione [Cald11]

Il testing di sicurezza delle applicazioni o dei sistemi è direttamente correlato alla tecnologia. Tuttavia, ogni vulnerabilità identificata dal testing di sicurezza potrebbe avere origine nel processo e/o nel comportamento e potrebbe essere mitigata in futuro modificando i processi e/o il comportamento.

Esistono almeno i seguenti motivi per cui un'organizzazione dovrebbe implementare un ISMS, direttamente supportati dal testing di sicurezza [Cald11]:

- Strategico, per gestire meglio la sicurezza delle informazioni nel contesto dei rischi di business complessivi
- Fiducia dei clienti, per dimostrare che un'organizzazione è conforme alle best practice di gestione della sicurezza delle informazioni
- Normativa, per soddisfare i vari requisiti normativi
- Efficacia interna, per gestire tatticamente le informazioni in modo più efficace all'interno di un'organizzazione.

Il testing di sicurezza svolge un ruolo importante nella definizione di un ISMS. Poiché è legato al testing, dimostra lo status quo di un sistema. Questo può essere interpretato come segue:

- Come prova a supporto di un obiettivo che si prevede di raggiungere
- Come prova di un punto di partenza, che stimola ulteriori azioni di sicurezza

Un noto ciclo di feedback per modellare l'obiettivo e il punto di partenza è il ciclo Plan-Do-Check-Act (PDCA). ISO 27001 "adotta il modello di processo PDCA, che viene applicato per strutturare tutti i processi ISMS" [Cald11]. Entrambi gli aspetti, obiettivo e punto di partenza, sono visibili all'interno di questo modello:

- Obiettivo:
Dopo le fasi Plan e Do, la fase Check stabilisce se l'obiettivo pianificato è stato raggiunto.
- Punto di partenza:
Il risultato della fase Check, supportata da attività di testing di sicurezza, viene analizzato nella fase Act per migliorare il processo complessivo. Le sue deviazioni costituiscono la base per il ciclo PDCA successivo.

Il testing di sicurezza fornisce il massimo valore aggiunto all'interno di un'organizzazione se considera entrambi gli aspetti (ovvero l'obiettivo e il punto di partenza)

Per misurare il raggiungimento delle fasi Plan e Do in termini di miglioramento della sicurezza, è necessario adeguare con precisione l'approccio ai test di sicurezza, in modo che questi ultimi corrispondano esattamente all'obiettivo pianificato. È buona pratica definire in anticipo i criteri di accettazione dei test di sicurezza per la fase Check al momento della definizione del piano di sicurezza.

Per utilizzare il testing di sicurezza come punto di partenza per il ciclo PDCA successivo, la tecnica di test, gli strumenti applicati, le test suite eseguite e tutti i risultati dei test (sia positivi che negativi) devono essere riportati nel test report.

È possibile utilizzare lo stesso approccio del test di sicurezza per impostare un nuovo punto di partenza e per misurare l'efficacia del ciclo PDCA successivo. Il testing di sicurezza deve essere applicato con un

approccio molto sistematico. Tutti i parametri rilevanti devono essere archiviati per consentire un testing di sicurezza ripetibile e oggettivo.

7.3 Migliorare un ISMS Tramite l'Adeguamento del Testing di Sicurezza

L'efficacia di un ISMS definito nel fornire una maggiore sicurezza dipende in larga misura dalle azioni proattive introdotte dall'ISMS stesso. Ciò significa che un ISMS dovrebbe derivare i controlli di sicurezza sulla base dello stato di sicurezza attuale e di una valutazione o di una situazione di business attuale (ad esempio, un incidente).

Oltre a qualsiasi azione diretta di mitigazione del rischio intrapresa per correggere i problemi identificati durante le normali operazioni, l'ISMS mira a derivare controlli che prevengano il ripetersi di tali problemi negli sviluppi futuri. Maggiore è il numero di iterazioni PDCA che un determinato ISMS ha attraversato, migliore sarà l'insieme dei controlli di sicurezza devianti e più elevato sarà il livello di sicurezza di tutte le applicazioni sviluppate.

7.3.1. Migliorare la visione olistica di un ISMS

Per migliorare un ISMS aumentando la sua visione olistica, il testing di sicurezza deve assumersi una nuova responsabilità oltre a stabilire la baseline per la fase Check all'interno del ciclo PDCA. Se l'obiettivo è migliorare la maturità dell'ISMS, il STE deve introdurre aspetti completamente nuovi che non erano ancora stati pianificati come parte del ciclo PDCA. Questi nuovi test di sicurezza potrebbero generare ulteriori approfondimenti sul SUT, che possono poi essere utilizzati per migliorare ulteriormente la maturità dell'ISMS derivando ulteriori controlli di sicurezza.

Le dimensioni tipiche che un STE può utilizzare per migliorare l'ambito dell'ISMS sono:

- Oggetti di test aggiuntivi:
Ogni sistema che deve raggiungere un determinato livello di sicurezza deve essere considerato nel suo ambiente tipico quando è in produzione. Il sistema può trovarsi dietro un firewall, essere collegato a un database centrale, avere un'interfaccia API con applicazioni/sistemi esterni, essere controllato da un processo di backup giornaliero o essere collegato a un sistema di gestione degli account privilegiati. Più sistemi sono connessi al SUT, più ampia è la superficie di attacco. Ogni sistema può essere attaccato e, se viene compromesso, è altamente probabile che anche l'intera rete di sistemi smetta di funzionare.

Un ISMS dovrebbe coprire il maggior numero possibile di aspetti per gestire la sicurezza delle informazioni nel modo più olistico possibile: ogni aspetto dovrebbe riflettere i possibili vettori di attacco. Per fare ciò, è essenziale che un STE si focalizzi sul maggior numero possibile di oggetti di test, perché ognuno di essi potrebbe essere un componente rischioso in una rete complessiva. Se uno dei test di sicurezza identifica ulteriori debolezze in un componente che non fa ancora parte dell'ISMS complessivo, è possibile migliorare la maturità dell'ISMS sulla base di queste nuove informazioni. Non è compito del STE definire contromisure (ad esempio, politiche di sicurezza o adeguamenti del processo), ma il suo compito include l'apertura mentale riguardo a componenti SUT aggiuntivi utili per la maturazione dell'ISMS.

- **Approcci di test aggiuntivi:**
Un'altra possibilità per un STE di fornire ulteriori approfondimenti su un sistema è utilizzare diversi tipi di testing. Anche se l'azione Check come parte del ciclo PDCA si focalizza su un test dinamico black-box del SUT, potrebbe essere utile eseguire anche un test statico. Questo potrebbe potenzialmente mostrare ulteriori vulnerabilità che non sono state identificate finora e che potrebbero essere utilizzate anche dagli attacker. Queste nuove informazioni dovrebbero essere utilizzate come input per un ISMS al fine di migliorarne ulteriormente la maturità.
- **Copertura migliorata:**
Anche mantenendo l'oggetto di test e l'approccio di test esistenti, il STE può generare informazioni preziose per la maturazione dell'ISMS. La semplice aggiunta di alcuni test case potrebbe generare informazioni completamente nuove. Questo può essere facilmente realizzato utilizzando strumenti di fuzz test strutturato o tabelle arcobaleno.

Un altro modo per migliorare la copertura potrebbe essere aumentare il numero di test case eseguiti per unità di tempo (ad esempio, automatizzando alcune test suite) o aumentare il numero di cicli di test eseguiti per forzare comportamenti insoliti che possono essere utilizzati per gli attacchi. Se queste misure di copertura migliorate identificano ulteriori vulnerabilità, ciò contribuirà a migliorare la maturità dell'ISMS.

7.3.2. Migliorare la misurabilità all'interno di un ISMS

Il STE può migliorare la maturità dell'ISMS introducendo un ciclo di feedback basato su metriche. Queste metriche sono solitamente denominate key performance indicator e supportano il miglioramento continuo utilizzando cicli PDCA. L'idea fondamentale del ciclo PDCA sottostante è verificare l'efficacia delle azioni Plan e Do precedenti. Quanto più oggettivamente può essere eseguita la fase Check/Act, tanto più oggettivo diventa il ciclo di feedback. Le politiche derivanti dall'ISMS includono metriche di copertura per le tecniche di test e il comportamento. Se una politica del portafoglio delle politiche complessivo di un'organizzazione include l'uso diretto di una tecnica di test, allora il testing di sicurezza può verificare direttamente se tale politica ha successo.

Anche se una politica riguarda solo indirettamente i processi utilizzati, il STE può comunque supportare la misurazione della sua efficacia. Ad esempio, potrebbe essere difficile misurare direttamente l'efficacia di alcuni corsi di formazione relativi alle convenzioni di codifica sicura. Un modo potrebbe essere condurre esami alla fine di ogni sessione di formazione. Un altro modo sarebbe impostare un test di sicurezza che verifichi con precisione il verificarsi di anti-pattern di sicurezza che sono stati trattati nella formazione. Quanto più frequentemente questi anti-pattern persistono dopo la formazione, tanto minore è il valore aggiunto della formazione in termini di sicurezza.

Lo stesso vale per la valutazione dell'efficacia dei comportamenti indesiderati. Ad esempio, il test di sicurezza potrebbe definire una simulazione di phishing che conta il numero di clic indesiderati effettuati all'interno di un'e-mail. Quando si considera il phishing, tassi di clic più elevati sono generalmente considerati negativi perché indicano che gli utenti non si accorgono che l'e-mail è di phishing, mentre tassi di clic bassi sono spesso considerati positivi. Tuttavia, per misurare l'efficacia di un'iniziativa di

sensibilizzazione, questo test di sicurezza deve essere ripetuto per consentire la misurazione dei cambiamenti prima e dopo. [StGrTh20]

Il testing di sicurezza può migliorare la maturità dell'ISMS perché i cicli di feedback si basano sui dati concreti generati dal testing di sicurezza. I miglioramenti avvengono più rapidamente e sono più affidabili di quelli basati su opinioni soggettive o sensazioni.

8 Il Reporting dei Risultati dei Test di Sicurezza - 135 minuti (K3)

Parole Chiave

hacker etico, mitigazione del rischio, test report

Parole Chiave Relative alla Sicurezza

nessuna

Obiettivi di Apprendimento per il Capitolo 8:

8.1 Il Reporting dei Test di Sicurezza

STE-8.1.1 (K2) Comprendere l'importanza dei risultati dei test di sicurezza e come questi influenzano la loro gestione e comunicazione

8.2 Identificare e Analizzare le Vulnerabilità

STE-8.2.1 (K3) Valutare i risultati del test di un determinato test di sicurezza per identificare le vulnerabilità

8.3 Chiudere le Vulnerabilità

STE-8.3.1 (K3) Valutare diverse tecniche per chiudere le vulnerabilità identificate

8.1 Il Reporting dei Test di Sicurezza

Ogni test di sicurezza si conclude con un test report [ISTQB Glossary]. Senza il reporting dei test, il test manca di prove che possano essere utilizzate per determinare azioni o decisioni basate sul risultato del test.

Le seguenti informazioni standard sono importanti quando si esegue il reporting di un test case di sicurezza fallito:

- Ambiente di test utilizzato: di solito include indirizzi IP specifici, whitelist IP applicate e account/password utilizzati
- Precondizioni dei test eseguiti. Includono tutte le attività di preparazione da applicare prima che la test suite preparata possa essere eseguita. Potrebbero includere attività quali dettagli di login, impostazioni specifiche dei file di configurazione o configurazioni specifiche del perimetro.
- Dati di test utilizzati
- Procedura di esecuzione del test
- Risultati attesi e risultati effettivi

Un test di sicurezza fallito significa che un test specifico ha rilevato la violazione di almeno un aspetto della sicurezza della triade CIA (vedere il capitolo 1). Un buon test report include un livello di dettaglio sufficiente per consentire la ripetizione del test. Gli strumenti utilizzati per eseguire i test potrebbero essere indicati nel test report e potrebbero essere inclusi screenshot per supportare i risultati del test con prove.

In generale, i test report di sicurezza dovrebbero essere trattati con un elevato livello di confidenzialità. Se questo tipo di informazioni viene divulgato al di fuori dell'organizzazione, potrebbe danneggiare gravemente la reputazione dell'organizzazione. Ancora peggio, le informazioni potrebbero essere utilizzate per attaccare qualsiasi sistema che presenti questa vulnerabilità.

Più un test report di sicurezza contiene test falliti, più il test report stesso e la sua comunicazione diventano critici e sensibili. In generale, ogni test report di sicurezza deve essere comunicato con cautela all'interno dell'organizzazione. Questo include le comunicazioni interne all'organizzazione che produce il SUT, poiché gli attacker potrebbero provenire dall'interno dell'organizzazione (cfr. [SwissCybInst20]). D'altra parte, i test report di sicurezza potrebbero essere importanti per molte persone all'interno di un'organizzazione. Questo paradosso influenza direttamente le attività di reporting del STE e di solito viene risolto creando diverse versioni dello stesso test report, ognuna delle quali contiene diversi livelli di dettaglio. Ogni versione del test report dovrebbe seguire il concetto di "necessità di sapere". Questo vale per coloro che hanno il compito di mitigare i rischi identificati e che quindi ricevono un test report completo o frammenti di esso in base al concetto di "necessità di sapere".

La sensibilità di un test report di sicurezza può essere modificata in base alle vulnerabilità identificate. Ciò è essenziale quando gli hacker etici identificano una vulnerabilità in un SUT e desiderano informare solo lo sviluppatore per dargli l'opportunità di mitigare questo rischio prima che il test report sia reso pubblico. Questa divulgazione responsabile è una delle caratteristiche di un hacker etico, in particolare per gli hacker white hat [Huneidy21]. Gli hacker grey hat spesso utilizzano la pubblicazione dei test report per aumentare la pressione su un'organizzazione affinché lavori sulle patch [Huneidy21].

8.2 Identificare e Analizzare le Vulnerabilità

È importante notare che l'assenza di una test suite fallita non significa che il sistema sia privo di difetti. Anche le test suite superate non significano necessariamente che il vettore di attacco esaminato non possa essere sfruttato. Significa semplicemente che con le test suite utilizzate non è possibile essere sfruttati da un vettore di attacco analizzato.

Se un test di sicurezza fallisce, viene identificata una potenziale vulnerabilità. Il test report dovrebbe fornire tutte le prove necessarie per ripetere il test case fallito. Un test report di sicurezza potrebbe evidenziare molte vulnerabilità. Prima di intraprendere qualsiasi azione correttiva, è necessario seguire i seguenti passaggi:

- **Demarcazione della vulnerabilità:**
Di solito un test fallito rappresenta un singolo test case fallito e una singola vulnerabilità. Tuttavia, potrebbero esistere diversi altri test case che mostrano la stessa vulnerabilità. Ad esempio, se un parametro di input vuoto può essere utilizzato per assumere il controllo di un'applicazione, forse lo stesso comportamento può essere ottenuto utilizzando un file da 100 MB come parametro di input. Durante la fase di demarcazione della vulnerabilità, vengono eseguiti test diversi ma simili per demarcare la vulnerabilità identificata. Questo è importante per la successiva valutazione del rischio e deve essere supportato dal STE.
- **Regolazione della probabilità del rischio:**
Questo passaggio viene eseguito per ricontrollare la probabilità del rischio di poter identificare una vulnerabilità nella produzione. Di solito, un test di sicurezza non viene eseguito su un sistema in produzione. Anche se l'ambiente di test è simile, non sarà mai completamente identico all'ambiente di produzione. In particolare, alcuni controlli di sicurezza potrebbero essere esplicitamente disabilitati per consentire l'esecuzione di un test di sicurezza. Se un SUT mostra una vulnerabilità, questa potrebbe essere offuscata da altri parametri messi in atto per la produzione. In tal caso, la vulnerabilità esiste ancora, ma non può essere sfruttata direttamente a causa di altri parametri. Questa regolazione potrebbe modificare il livello di rischio suggerito dal test di sicurezza e quindi modificare la necessità complessiva di pianificare azioni di mitigazione del rischio. Il STE ha il compito di considerare questa regolazione del rischio e di intraprendere le azioni corrispondenti.
- **Regolazione dell'impatto del rischio:**
Questo passaggio viene eseguito per ricontrollare il possibile impatto del rischio derivante dall'esposizione a una vulnerabilità. Di solito, l'attenzione del STE si focalizza sugli aspetti tecnici, il che rende difficile stimare l'impatto sul business. In particolare, se il STE riutilizza le valutazioni dell'impatto per le vulnerabilità identificate da una fonte esterna (vedere il capitolo 4, CVSS), queste possono essere imprecise per un contesto specifico. Se viene identificata una vulnerabilità, gli stakeholder di business affinano il possibile impatto del rischio. La vulnerabilità può essere stimata come priva di impatto dal punto di vista del business (ad esempio, se il componente impattato è usato raramente), oppure può essere considerata come avente un alto

livello di criticità di business. Questa regolazione potrebbe modificare il livello di rischio suggerito dal STE e giustificare un aggiornamento delle azioni di mitigazione del rischio pianificate.

Se vengono applicati tutti e tre i passaggi sopra indicati, è possibile ottenere una visione chiara della vulnerabilità identificata e del rischio identificato. L'adeguamento della probabilità del rischio e dell'impatto del rischio deve tenere conto di alcuni parametri importanti:

- Stretta comunicazione con gli stakeholder di business, poiché hanno l'ultima parola quando si discute del possibile impatto del rischio
- Stretta comunicazione con il team operativo, poiché ha l'ultima parola quando si considerano i parametri di produzione e come questi differiscono da quelli utilizzati nel testing di sicurezza
- Anche se l'esistenza della vulnerabilità nel SUT è chiara, gli stakeholder potrebbero cercare di influenzare attivamente la fase di regolazione del rischio (modificando la probabilità o l'impatto)

Se il livello di rischio residuo è considerato troppo elevato per avviare o mantenere la produzione, è necessario elaborare un piano di mitigazione del rischio. Il management è responsabile di decidere l'urgenza di tali piani di mitigazione del rischio. La decisione può essere presa tra i seguenti livelli di urgenza:

- Interrompere immediatamente l'operazione o interrompere ulteriori attività di messa in funzione:
Se il livello di rischio è considerato troppo elevato e non può essere accettato, l'unico modo per evitarlo è non eseguire il SUT. Il livello di rischio (ad esempio, alto, medio o basso) che influenza questa decisione dipende dalla propensione al rischio, definita come *"la quantità di rischio che un'organizzazione è disposta ad accettare per raggiungere i propri obiettivi"* [CARM22].
- Continuare a far funzionare il sistema con un monitoraggio intensivo:
Se il sistema è troppo critico o il rischio è troppo elevato, il SUT può continuare a funzionare ma viene monitorato intensamente.
- Aggiungere azioni di mitigazione del rischio al normale piano di release:
Se il rischio può essere gestito o il sistema ha molti vincoli di release rigorosi, le azioni di mitigazione del rischio vengono analizzate ed eseguite, ma non applicate direttamente al SUT. Invece, i componenti patchati vengono aggiunti al normale ciclo di release per garantire che la prossima release pianificata contenga le patch di sicurezza richieste.

Il STE deve garantire che vengano eseguiti test confermativi e regression test per ogni azione di mitigazione del rischio. Il test confermativo dovrebbe tenere conto delle prove fornite nel test report e dovrebbe anche utilizzare le lezioni apprese dalla fase di demarcazione della vulnerabilità.

8.3 Chiudere le Vulnerabilità Identificate

Se una vulnerabilità identificata è nota in termini di demarcazione della vulnerabilità e se si decide di mitigare il rischio, sono disponibili almeno due alternative di alto livello per mitigare le vulnerabilità identificate:

- Nascondere la vulnerabilità riducendo il rischio previsto
- Evitare la vulnerabilità applicando una patch al sistema interessato

Entrambe le alternative possono essere combinate. Nascondere la vulnerabilità può fornire una soluzione a breve termine e successivamente è possibile applicare la patch appropriata.

8.3.1. Nascondere una vulnerabilità

L'idea di nascondere le vulnerabilità è simile all'approccio applicato da un tester quando distingue tra un difetto che non causa un failure e un difetto che causa un failure (cioè che può avere un impatto sul cliente).

Anche se le fasi di regolazione del rischio hanno dimostrato che la vulnerabilità identificata è causata da un failure (ovvero espone a un rischio elevato), il sistema stesso potrebbe essere modificato in modo tale che il difetto non si manifesti al cliente, anche se il difetto rimane invariato. In termini di mitigazione del rischio, questo tipo di azione mira alla riduzione dell'impatto del rischio. Il difetto esiste ancora nel sistema, ma non può più essere sfruttato. Gli approcci tipici per nascondere le vulnerabilità in questo modo sono:

- **Blocco del traffico:**
I firewall moderni consentono analisi e meccanismi di blocco molto sofisticati. Se il traffico necessario per sfruttare la vulnerabilità è ben compreso, molti firewall possono essere configurati per bloccare tali modelli di traffico. In questo modo la vulnerabilità rimane invariata, ma non può più essere sfruttata.
- **Patching virtuale:**
Il patching virtuale non blocca necessariamente il traffico, ma lo converte in modo tale che la vulnerabilità non possa essere sfruttata. [OWASP11] lo definisce come "un livello di applicazione delle politiche di sicurezza che impedisce lo sfruttamento di una vulnerabilità conosciuta. La patch virtuale funziona poiché il livello di applicazione della sicurezza analizza le transazioni e intercetta gli attacchi in transito, in modo che il traffico dannoso non raggiunga mai l'applicazione. L'impatto risultante di una patch virtuale è che, sebbene il codice sorgente effettivo dell'applicazione stessa non sia stato modificato, il tentativo di sfruttamento non ha successo".
- **Disattivazione o riconfigurazione di specifiche funzionalità del sistema:**
Se la vulnerabilità ha una portata molto limitata all'interno del sistema, potrebbe essere possibile disattivare la funzionalità interessata dalla vulnerabilità identificata.
- **Ridurre la portata delle vulnerabilità:**
Potrebbe essere possibile accettare il rischio associato a una vulnerabilità riducendone la portata. Ad esempio, potrebbe essere possibile impostare filtri IP in modo che solo macchine conosciute con indirizzi IP dedicati possano connettersi alla macchina vulnerabile. Inoltre, potrebbe essere possibile disabilitare l'accesso esterno alla macchina vulnerabile e consentire solo l'accesso interno.

Durante il testing confermativo, il STE è responsabile dell'applicazione di tutti gli approcci relativi alle vulnerabilità nascoste per garantire che la vulnerabilità specifica non possa più essere sfruttata.

8.3.2. Evitare una vulnerabilità

In generale, evitare la vulnerabilità è un'azione molto dispendiosa in termini di tempo e denaro. Per evitare una vulnerabilità è necessario eseguire i seguenti passaggi:

Passaggio	Descrizione
Individuare la vulnerabilità	Poiché le azioni di mitigazione del rischio devono essere intraprese al livello utilizzato per implementare la funzionalità (ad esempio, codice, modelli e configurazioni), potrebbe essere necessario del tempo per identificare il componente interessato e, all'interno di tale componente, l'area interessata sulla base di una vulnerabilità identificata a livello di sistema.
Comprendere la vulnerabilità	Prima di effettuare una riparazione, è necessario comprendere appieno la vulnerabilità analizzandola nell'area interessata (ad esempio, snippet di codice).
Identificare le azioni di mitigazione del rischio	- È necessario sviluppare un approccio per la mitigazione del rischio. - L'azione di mitigazione potrebbe consistere in un algoritmo completamente nuovo, un nuovo componente, una piccola modifica alla configurazione o solo alcune piccole modifiche al codice (ad esempio, l'inclusione di un comportamento di eccezione specifico).
Eseguire l'azione di mitigazione del rischio	Viene applicata l'azione di mitigazione del rischio identificata.
Test confermativo	Eseguire un test confermativo sul sistema per testare se la vulnerabilità è stata eliminata.
Regression test	Il testing è una parte fondamentale per evitare le vulnerabilità e non dovrebbe focalizzarsi solo sul codice modificato. È essenziale eseguire la regression test suite completa per assicurarsi che il sistema funzioni ancora correttamente e che l'azione di mitigazione non abbia avuto effetti collaterali indesiderati.
Distribuzione	Distribuire il sistema patchato.

Passaggio	Descrizione
	Dopo che il sistema è stato distribuito, di solito viene effettuato un attento monitoraggio per un certo periodo al fine di assicurarsi che il sistema funzioni correttamente.

9 Strumenti del Testing di Sicurezza - 90 minuti (K3)

Parole Chiave

dynamic application security testing, interactive application security testing, static application security testing, scanning delle vulnerabilità

Parole Chiave Relative alla Sicurezza

software open source, software composition analysis (SCA)

Obiettivi di Apprendimento per il Capitolo 9:

9.1 Categorizzazione degli Strumenti del Testing di Sicurezza

STE-9.1.1 (K3) Analizzare diversi use case e applicare le categorizzazioni degli strumenti del testing di sicurezza

9.2 Selezionare gli Strumenti del Testing di Sicurezza

STE-9.2.1 (K2) Comprendere l'utilizzo e i concetti degli strumenti del testing di sicurezza statico

STE-9.2.2 (K2) Comprendere l'utilizzo e i concetti degli strumenti del testing di sicurezza dinamico

9.1 Categorizzazione degli Strumenti del Testing di Sicurezza

Esistono diverse possibilità per categorizzare gli strumenti del testing di sicurezza. Una selezione di queste categorie include:

- L'attività nel processo di test di sicurezza in cui lo strumento può essere utilizzato
- Strumenti del testing di sicurezza open source rispetto a quelli closed source
- Strumenti di analisi statica rispetto a strumenti di test dinamico
- Piattaforma/infrastruttura rispetto a applicazione
- Esecuzione dei test di sicurezza rispetto a management dei test di sicurezza
- Testing black-box rispetto a testing white-box rispetto a testing grey-box

Ognuna di queste categorie presenta vantaggi e svantaggi quando il STE seleziona lo strumento del testing di sicurezza più appropriato. Questo syllabus presenta tre categorie principali:

- Testing black-box rispetto a testing white-box
- Testing di sicurezza statico rispetto a testing di sicurezza dinamico
- Strumenti del testing di sicurezza open source rispetto a quelli closed source

Ci si aspetta che i STE creino una propria libreria di strumenti per il proprio dominio e contesto. Tuttavia, potranno comunque adottare le categorie suggerite.

9.1.1. Strumenti del testing di sicurezza white-box

Se il STE ha accesso a livello di codice, gli strumenti di test di sicurezza white-box forniscono loro informazioni relative al codice, alle configurazioni, alle librerie utilizzate, alle applicazioni, al sistema e alla piattaforma, all'architettura e ai dettagli di login. Un prerequisito importante è che il STE abbia il permesso di eseguire l'analisi, poiché cercherà di identificare e valutare le vulnerabilità del sistema e dei sistemi integrati.

9.1.2. Strumenti del testing di sicurezza black-box

Un prerequisito per eseguire il testing black-box è l'accesso a un'applicazione o a un sistema in esecuzione in un ambiente simile a quello di produzione. Questo è necessario per poter eseguire i test utilizzando strumenti di test di sicurezza black-box, che considerano il SUT come una black box e non richiedono alcuna conoscenza interna del software. Gli strumenti del testing di sicurezza black-box si focalizzano sull'identificazione delle vulnerabilità durante l'esecuzione dell'applicazione/del sistema.

9.1.3. Strumenti del testing di sicurezza grey-box

L'esecuzione del testing di sicurezza grey-box fornisce al STE alcune informazioni limitate sui componenti interni dell'applicazione/del sistema e l'accesso a una versione in esecuzione. Gli strumenti di test di sicurezza grey-box possono essere considerati una combinazione di strumenti del testing di sicurezza white-box e strumenti del testing di sicurezza black-box. Richiedono alcune informazioni interne e un sistema in esecuzione. L'obiettivo è identificare le vulnerabilità eseguendo l'applicazione/il sistema ed eseguendo test che tengono conto dei dettagli interni.

9.1.4. Strumenti del testing di sicurezza statico

Un aspetto importante per la categorizzazione degli strumenti di test di sicurezza si basa sulla differenza tra testing di sicurezza statico e testing di sicurezza dinamico. Le definizioni, le differenze e le descrizioni dei test di sicurezza statici e dinamici sono discusse nella sezione 2.1.2.

All'interno di questa categoria, gli strumenti possono essere considerati in base al loro utilizzo. Questi includono testing di rete, testing del sistema operativo, testing del database e testing delle applicazioni.

Il testing di sicurezza statico ha molto in comune con il testing di sicurezza white-box. La differenza principale è che gli strumenti del testing di sicurezza statico non necessitano che l'applicazione o il sistema siano in esecuzione. Lo strumento accede al codice, alle librerie o ai file di configurazione sotto esame e li analizza rispetto al repository interno dello strumento contenente vulnerabilità note legate alla sintassi, alla semantica o agli standard di codice per il linguaggio specifico in uso.

Esistono vari modi per eseguire test di sicurezza con gli strumenti. Per gli strumenti del testing di sicurezza statico questi includono, ma non sono limitati a, SAST e SCA. Questi sono spiegati più dettagliatamente di seguito.

Static Application Security Testing

SAST è una tipica attività di testing di sicurezza statico inclusa principalmente in una pipeline Dev(Sec)Ops, come discusso nel capitolo 6.1 e [NIST DevSecOps]. All'interno di queste pipeline, viene eseguito automaticamente ogni volta che parte del codice viene modificato e registrato. Utilizzare SAST in questo modo fornisce un feedback immediato. SAST si focalizza principalmente sull'applicazione e nella maggior parte dei casi non copre alcuna piattaforma o componente dell'infrastruttura. Inoltre, questi strumenti forniscono buone metriche di copertura del codice.

Esiste una forte dipendenza tra il testing di sicurezza statico e l'attributo "software open source" degli strumenti, poiché gli strumenti open source, per definizione, forniscono il codice sorgente. Ciò significa che SAST può essere eseguito su tutti i sistemi open source. Questo non è il caso delle applicazioni closed source. Potrebbe essere possibile eseguire alcuni test statici (ad esempio, identificare alcune librerie riutilizzate), tuttavia questo tipo di analisi non è possibile a causa dell'offuscamento o di accordi di licenza speciali che vietano le analisi statiche.

Software Composition Analysis

La software composition analysis (SCA) ha molti punti di contatto con la sicurezza. Gli strumenti SCA analizzano le vulnerabilità nel codice, comprese le dipendenze e i componenti open source utilizzati dall'applicazione. Questi componenti sono ben noti e possono presentare diverse vulnerabilità. Gli strumenti SCA possono suggerire rimedi alle vulnerabilità di sicurezza sulla base dei componenti identificati. Quando lo fanno, quasi tutti gli strumenti SCA utilizzano il database Common Vulnerabilities and Exposures [CVE21] delle vulnerabilità divulgate pubblicamente. (vedere la sezione 4.2.2)

9.1.5. Strumenti del testing di sicurezza dinamico

Gli strumenti del testing di sicurezza dinamico interagiscono con il SUT mentre è in esecuzione. Sia il testing di sicurezza black-box che il testing di sicurezza grey-box sono strettamente legati al testing di sicurezza dinamico. Gli strumenti possono essere considerati nelle categorie DAST e IAST.

Dynamic Application Security Testing

Come nel caso di SAST, il dynamic application security testing (DAST) è comunemente utilizzato in un contesto DevSecOps [NIST DevSecOps]. L'attività di testing viene eseguita automaticamente nella pipeline utilizzando uno strumento del testing di sicurezza black-box configurabile. Questo analizza l'applicazione o simula un attacker mentre il software è in esecuzione, alla ricerca di vulnerabilità quali la validazione degli input, il fuzz testing, l'autenticazione e l'autorizzazione, la configurazione e il deployment, la gestione delle sessioni, la gestione degli errori e la crittografia.

La scansione DAST simula diversi attacchi in tempo reale sul SUT in modo automatizzato per identificare eventuali vulnerabilità nell'applicazione. Le tecniche di test vengono utilizzate su un SUT in esecuzione durante l'esecuzione di test dinamici. Di conseguenza, questi richiedono più tempo e prestazioni rispetto al testing statico. Anche se il DAST si focalizza sull'applicazione, le vulnerabilità identificate spesso riguardano i componenti dell'infrastruttura necessari per eseguire l'applicazione.

Il DAST non copre in modo affidabile tutti i problemi OWASP Top 10 [OWASP Top 10] o SANS CWE Top 25 [CWE21]. Molti strumenti possono coprire aspetti specifici di ciascuna delle classi di vulnerabilità, ma l'utilizzo di questi strumenti può dare un falso senso di sicurezza.

Interactive Application Security Testing

L'interactive application security testing (IAST) è un approccio di test ibrido che utilizza sia il testing di sicurezza statico che quello dinamico. Gli strumenti vengono utilizzati per determinare se le vulnerabilità note nel codice sorgente sono sfruttabili durante il runtime.

Nell'ambiente in cui è in esecuzione l'applicazione viene installato un agente che monitora l'applicazione e identifica eventuali vulnerabilità mentre il STE (o lo strumento del testing di sicurezza dinamico) interagisce con il SUT.

9.1.6. Considerazioni per la selezione degli strumenti del testing di sicurezza

Il STE deve sapere quale strumento selezionare per quale contesto. Dovrebbe quindi conoscere i diversi schemi di categorizzazione e gli strumenti appartenenti a ciascuna categoria.

I cataloghi degli strumenti del testing di sicurezza possono aiutare a selezionare lo strumento giusto. Alcuni esempi sono disponibili in: [KALI], [OWASP], [SANS] e [NIST]. Si noti, tuttavia, che gli strumenti possono essere categorizzati in modo diverso in alcuni di questi cataloghi e possono essere presenti in uno e mancare in un altro.

Il STE non ha bisogno di conoscere e saper utilizzare tutti gli strumenti del testing di sicurezza disponibili. I STE che operano da più tempo in un particolare dominio/contesto in genere creano le proprie librerie specifiche.

Quando si seleziona uno strumento o si crea una libreria di strumenti, si raccomanda di procedere come segue:

- Focalizzarsi su ciò che deve essere verificato
- Non dipendere da un unico fornitore per i risultati dei test richiesti. Utilizzare più fornitori per la stessa funzionalità dello strumento.
- Esaminare periodicamente il mercato alla ricerca di nuovi strumenti emergenti.

Open source vs closed source

La differenza tra strumenti di test open source e closed source (con licenza) può essere un aspetto importante nella selezione dello strumento.

Chiunque può partecipare allo sviluppo di applicazioni o strumenti open source. Ciò contribuisce a eliminare le vulnerabilità di sicurezza il più rapidamente possibile, almeno se il progetto open source ha una comunità di sviluppo attiva. Inoltre, gli strumenti open source possono essere verificati (almeno in teoria), in modo che le backdoor siano visibili nel codice e possano essere escluse.

Il software open source può essere personalizzato e utilizzato per contesti specifici, il che gli conferisce un chiaro vantaggio.

Le seguenti caratteristiche potrebbero essere considerate svantaggi dell'utilizzo di strumenti open source:

- Mancanza di supporto professionale, soprattutto quando nessuna comunità attiva supporta un prodotto specifico. Tuttavia, alcune organizzazioni sono specializzate nella fornitura di supporto per le applicazioni OSS, il che è un aspetto importante per l'ambiente aziendale.
- Problemi di licenza (ad esempio, quando si utilizzano licenze pubbliche GNU Not Unix)
- È necessario disporre delle competenze di sviluppo necessarie (ad esempio, per adattarsi a contesti specifici)
- Se esiste una vulnerabilità in un sistema open source, c'è il rischio che qualcuno la identifichi e crei degli exploit
- L'ulteriore sviluppo delle applicazioni OSS è incerto, poiché sono per lo più guidate dalla comunità.

A differenza delle applicazioni OSS, le applicazioni closed source utilizzano codice proprietario che non è disponibile per l'utente. Ciò offre il vantaggio di poter usufruire di contratti di assistenza da parte del fornitore.

Le seguenti caratteristiche potrebbero essere considerate svantaggi dell'utilizzo di software closed source:

- È necessario pagare i diritti di licenza
- Non vi è alcuna garanzia contro le backdoor
- Eventuali vulnerabilità di sicurezza potrebbero rimanere sconosciute per lungo tempo
- Può verificarsi una forte dipendenza dal fornitore, in cui un cliente diventa fortemente dipendente dai prodotti o servizi di un fornitore specifico, rendendo difficile il passaggio a un fornitore diverso
- In generale, la possibilità di personalizzare lo strumento in base a un contesto specifico è limitata (ad esempio, il codice non può essere modificato).

9.2 Applicare gli Strumenti del Testing di Sicurezza

9.2.1. Comprendere l'utilizzo e i concetti degli strumenti del testing di sicurezza statico

I seguenti aspetti descrivono alcune delle caratteristiche principali degli strumenti del testing di sicurezza statico:

- Sono più efficaci se si conosce il SUT

- Possono essere applicati anche se l'applicazione stessa è ancora incompleta e contiene difetti
- Sono strettamente correlati al testing white-box
- Sono particolarmente adatti per individuare codice non sicuro o configurazioni errate nelle prime fasi del SDLC, poiché richiedono solo informazioni statiche sul codice sorgente
- Coprono l'intero codice sorgente e le configurazioni e richiedono quindi l'accesso in lettura agli stessi
- Sono in grado di leggere il determinato oggetto, come il codice sorgente di un'applicazione non compilata, e di confrontarlo con set di dati predefiniti di best practice e vulnerabilità note (ad esempio, comandi e catene non sicuri all'interno del codice sorgente)
- L'automazione li rende convenienti in termini di costi
- Spesso forniscono risultati falsi positivi poiché non sono sensibili al contesto. Ciò significa che non conoscono gli use case, lo stack di chiamate e la composizione di più righe di codice. Di conseguenza, potrebbero identificare vulnerabilità nel codice che non verranno mai eseguite nella realtà e quindi non hanno alcun impatto negativo sulla sicurezza.

Utilizzando, ad esempio, lo scanning di rete, lo strumento SAST valuterrebbe i file di configurazione per individuare configurazioni non sicure.

9.2.2. Comprendere l'utilizzo e i concetti degli strumenti del testing di sicurezza dinamico

La sicurezza continua delle applicazioni è importante nel contesto dell'approccio di continuous integration e delivery che si trova nello sviluppo software Agile. La sicurezza deve essere verificata in modo continuo e ripetuto per ogni incremento. La motivazione per l'utilizzo di strumenti di test dinamici è farli funzionare regolarmente e applicare l'automazione per garantire che le ultime vulnerabilità conosciute vengano immediatamente individuate e segnalate. Il monitoraggio e il miglioramento continuo della sicurezza è anche chiamato DevSecOps (vedere il capitolo 6 e [NIST DevSecOps]).

In genere, il DAST si focalizza sulla top 10 delle vulnerabilità di OWASP, quali: iniezioni (ad esempio, SQL injection, cross-site scripting e iniezioni di comandi), controlli di accesso non funzionanti, cross-site-request-forgery, condizioni di gara, errori della logica di business, perdite di memoria e vulnerabilità note.

È inoltre importante sottolineare che gli strumenti DAST automatizzati eseguono la scansione solo di una serie di vettori di attacco predefiniti. Pertanto, altri test e controlli di sicurezza rimangono obbligatori.

Utilizzando l'esempio dello scanning di rete, uno strumento di test dinamico esegue prima la scansione della rete alla ricerca di porte aperte, quindi esegue i servizi su tali porte.

10 Riferimenti

- [Bittau] Cryptographic protection of TCP Streams (tcpcrypt) <https://tools.ietf.org/html/draft-bittau-tcp-crypt-04>
- [BSI21] https://www.bsi.bund.de/EN/Themen/KRITIS-und-regulierte-Unternehmen/Weitere_regulierte_Unternehmen/LuSi/Luftsicherheit_node.html, ultima visita 7 novembre 2022
- [Bullock2017] Jessey Bullock: "Wireshark for Security Professionals: Using Wireshark and the Metasploit", Wiley 2017, disponibile online all'indirizzo https://computerscience.unicam.it/marcantoni/reti/laboratorio_wireshark/Wireshark%20for%20Security%20Professionals%20-%20Using%20Wireshark%20and%20the%20Metasploit%20Framework.pdf
- [BURP22] <https://portswigger.net/burp>, ultima visita 1 dicembre 2022
- [Cald11] Alan Calder, Jan van Bon: "Implementing Information Security based on ISO 27001/ISO 27002 – a management guide", Van Haren Publishing, 2011
- [CAPEC21] CAPEC: "Common Attack Pattern Enumeration and Classification: A Community Resource for Identifying and Understanding Attacks", disponibile online all'indirizzo <https://capec.mitre.org/>
- [CARM22] Mary Carmichael, CRISC, CISA, CPA, Member of ISACA Emerging Trends Working Group: "Risk Appetite vs. Risk Tolerance: What is the Difference?", disponibile online all'indirizzo <https://www.isaca.org/resources/news-and-trends/isaca-now-blog/2022/risk-appetite-vs-risk-tolerance-what-is-the-difference>
- [CERT]: <https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88487865>
- [CERT1]: <https://wiki.sei.cmu.edu/confluence/display/seccode/Top+10+Secure+Coding+Practices>
- [CIS22] <https://www.cisecurity.org/cis-benchmarks/>, ultima visita 1 dicembre 2022
- [CLAIR22] <https://github.com/quay/clair>, ultima visita 1 dicembre 2022
- [Cloudflare] <https://www.cloudflare.com/learning/security/glossary/what-is-zero-trust/>, ultima visita 1 dicembre 2023
- [CVE21] CVE® Program Mission: "Identify, define, and catalog publicly disclosed cybersecurity vulnerabilities.", disponibile online all'indirizzo cve.org
- [CVSS21] First, Improving Security together: "Common Vulnerability Scoring System SIG", disponibile online all'indirizzo <https://www.first.org/cvss/>
- [CWE21] CWE: "Common Weakness Enumeration: A Community-Developed List of Software & Hardware Weakness Types", disponibile online all'indirizzo <https://cwe.mitre.org>
- [CWSS21] CWE: "Scoring CWEs: Common Weakness Scoring System CWSS", disponibile online all'indirizzo https://cwe.mitre.org/cwss/cwss_v1.0.1.html
- [FUZZDB22] <https://github.com/fuzzdb-project/fuzzdb>, ultima visita 1 dicembre 2022

- [Gart21] Gartner Information Technology: “Glossary”, disponibile online all'indirizzo <https://www.gartner.com/en/information-technology/glossary>
- [GITLAB22] https://docs.gitlab.com/ee/user/application_security/sast/, ultima visita 1 dicembre 2022
- [GTFO22] <https://gtfobins.github.io/>, ultima visita 13 novembre 2022
- [Huneidy21] Mariangel Huneidy: “The Ultimate Guide to Ethical Hacking”, 21 settembre, disponibile online all'indirizzo [The Ultimate Guide to Ethical Hacking \(0x1.gitlab.io\)](https://0x1.gitlab.io/)
- [IETF23] Internet Engineering Task Force: “Introduction to the IETF”, disponibile online all'indirizzo <https://www.ietf.org/about/introduction/>
- [ISO 15288] ISO/IEC 15288 – System Life Cycle Processes.
- [ISO 25010] ISO/IEC 25010:2011
- [ISO 27001] ISO International Standards Organization: “ISO/IEC 27001:2013; Information technology — Security techniques — information security management system— Requirements”
- [ISO 31000], ISO 31000:2018 - Risk management
- [ISO/IEC/IEEE 29119-3], Software and systems engineering — Software testing – Part 3 – Test Documentation
- [ISO/IEC/IEEE 29119-4], Software and systems engineering — Software testing – Part 4 – Test Techniques
- [ISO_Web_21] International Standards Organization: “Consumers and standards: Partnership for a better world”, disponibile online all'indirizzo https://www.iso.org/sites/ConsumersStandards/1_standards.html
- [ISTQB FL]: ISTQB Certified Tester Foundation Level Syllabus v4.0 [Certified Tester Foundation Level \(CTFL\) v4.0 \[NEW!\] \(istqb.org\)](https://www.istqb.org/certification/foundation-level)
- [ISTQB Glossary]: International Software Test Qualification Board: Glossary <https://glossary.istqb.org/en/search/>
- [ISTQB_ATTA_SYL]: ISTQB Certified Tester Advanced Level Technical Test Analyst (CTAL-TTA) Syllabus [Technical Test Analyst \(istqb.org\)](https://www.istqb.org/certification/advanced-level-technical-test-analyst)
- [ISTQB_ATA_SYL]: ISTQB Certified Tester Advanced Level Test Analyst (CTAL-TA) Syllabus Test Analyst (istqb.org)
- [ITGOV23a] IT-Governance: “ISO 27000 Series of Standards”, disponibile all'indirizzo <https://www.itgovernance.co.uk/iso27000-family>
- [ITGov23b]: ISO/IEC 27001 – “Information Security Management: The international standard for information security”, in IT-Governance, disponibile all'indirizzo <https://www.itgovernance.co.uk/iso27001>
- [KALI] <https://www.kali.org/tools/>, ultima visita 2 dicembre 2022
- [KUBEAUDIT22] <https://github.com/Shopify/kubeaudit>, ultima visita 1 dicembre 2022

- [METASPLOIT22] <https://www.metasploit.com/>, ultima visita 1 dicembre 2022
- [Micro09] Microsoft: "The STRIDE Threat Model", 12/2009, disponibile all'indirizzo [The STRIDE Threat Model | Microsoft Docs](#)
- [Micro22] Microsoft: Zero Trust Essentials eBook , disponibile all'indirizzo [Zero Trust Essentials eBook \(microsoft.com\)](#)
- [MITRE21] The MITRE Corporation: "Media FAQs", 2018, disponibile all'indirizzo [media-FAQs-2018.pdf \(mitre.org\)](#)
- [NESSUS22] <https://tenable.com/products/nessus>, ultima visita 1 dicembre 2022
- [NIST DevSecOps] <https://csrc.nist.gov/Projects/devsecops>
- [NIST Glossary] <https://csrc.nist.gov/glossary>
- [NIST 800-160] <https://csrc.nist.gov/pubs/sp/800/160/v2/r1/final>
- [NIST SP 800-161] Supply Chain Risk Management Practices for Federal Information Systems and Organizations
- [NIST02] https://csrc.nist.gov/glossary/term/security_service, ultima visita 6 novembre 2022
- [NIST03] NIST Special Publication 800-61, Revision 2, Computer Security Incident Handling Guide
- [NIST05] https://csrc.nist.gov/glossary/term/social_engineering, ultima visita 13 novembre 2022
- [NIST06] [NIST SP 800-137, Information Security Continuous Monitoring \(ISCM\) for Federal Information Systems and Organizations](#), ultima visita settembre 2011
- [NISTIR 7007]: An Overview of Issues in Test Intrusion Detection Systems
- [NMAP22] <https://nmap.org/>, ultima visita 1 dicembre 2022
- [OPENVAS22] <https://www.openvas.org/>, ultima visita 1 dicembre 2022
- [OSI] <https://opensource.org/definition/>
- [OWASP byte code obfuscation]: https://owasp.org/www-community/controls/Bytecode_obfuscation
- [OWASP Test Guide]: <https://owasp.org/www-project-web-security-test-guide>
- [OWASP Top 10] <https://www.owasp.org>
- [OWASP] https://owasp.org/www-project-web-security-test-guide/v41/6-Appendix/A-Test_Tools_Resource, ultima visita 2 dicembre 2022
- [OWASP11] Ryan Barnett, Dan Cornell, Achim Hoffmann Martin Knobloch "Virtual patching Best Practices", 2011, condiviso da OWASP, disponibile online all'indirizzo https://owasp.org/www-community/Virtual_Patching_Best_Practices
- [OWASP21] OWASP: "The Open Web Application Security Project", disponibile all'indirizzo <https://owasp.org/>
- [OWASP24] OWASP Dependency-Check. <https://owasp.org/www-project-dependency-check/>

- [PCI DSS Chapter 6]: <https://www.pcidssguide.com/pci-dss-requirement-6>
- [PCI22] https://listings.pcisecuritystandards.org/documents/PCI-DSS-v4_0.pdf, ultima visita 7 novembre 2022
- [RFC4765] The Intrusion Detection Message Exchange Format (IDMEF), Network Working Group
- [SANS] <https://www.sans.org/tools/>, ultima visita 2 dicembre 2022
- [SecJour21] Security Journey: "Why vulnerability list methodologies matter". Disponibile all'indirizzo <https://www.securityjourney.com/post/why-vulnerability-list-methodologies-matter-and-why-we-trust-cwe-owasp>
- [SENG22] <https://www.social-engineer.org/framework/information-gathering/dumpster-diving/>, ultima visita 10 novembre 2022
- [Shacklett] Shacklett, M. E., *What is an attack vector?* <https://www.techtarget.com/searchsecurity/definition/attack-vector>
- [SNYK22] <https://snyk.io/>, ultima visita 1 dicembre 2022
- [SONAR22] <https://www.sonarqube.org/>, ultima visita 1 dicembre 2022
- [SQLMAP22] <https://sqlmap.org>, ultima visita 1 dicembre 2022
- [SSLSCAN22] <https://github.com/rbsec/ssllscan>, ultima visita 1 dicembre 2022
- [SSLYZE22] <https://github.com/nabla-c0d3/sslyze>, ultima visita 1 dicembre 2022
- [Stallings18] Stallings William, Brown Lawrie, 2018, Computer Security: Principles and Practice, ISBN 9781292220611
- [StGrTh20] Michelle P. Steves, Kristen K. Greene e Mary F. Theofanos: "*Categorizing Human Phishing Detection Difficulty: A Phish Scale*", Journal of Cybersecurity. Pubblicato online il 14 settembre 2020. Disponibile all'indirizzo <https://academic.oup.com/cybersecurity/article/6/1/tyaa009/5905453>
- [SwissCybInst20] Swiss Cyber Institute: "41 Insider Threat Statistics You Should Care About", 21 settembre, disponibile online all'indirizzo [41 Insider Threat Statistics You Should Care About - Swiss Cyber Institute](#)
- [SYNOPSIS] <https://go.synopsys.com/software-integrity-agile-security-manifesto.html>
- [TechTarget] TechTarget: "The three ways: The Phoenix project", disponibile all'indirizzo <https://www.techtarget.com/whatis/definition/The-Three-Ways>
- [TELETRUST] <https://www.stand-der-technik-security.de/startseite/>
- [TERRASCAN22] <https://runterrscan.io/>, ultima visita 1 dicembre 2022
- [TR02021] https://www.bsi.bund.de/EN/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/Technische-Richtlinien/TR-nach-Thema-sortiert/tr02102/tr02102_node.html
- [UNECE20] <https://unece.org/sustainable-development/press/un-regulations-cybersecurity-and-software-updates-pave-way-mass-roll>, ultima visita 7 novembre 2022

- [HIPAA] <https://www.cdc.gov/phlp/publications/topic/hipaa.html>, ultima visita 22 agosto 2023
- [WaCh90] Dolores R. Wallace, John C. Cherniavsky: "*Guide to Software Acceptance*", NIST Special Publication 500-180, 1990, disponibile online all'indirizzo <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication500-180.pdf>
- [WAFEC]: <https://owasp.org/www-project-wafec/>
- [WIKI01] https://en.wikipedia.org/wiki/Google_hacking, ultima visita 10 novembre 2022
- [WIKI02] <https://en.wikipedia.org/wiki/Stuxnet>, ultima visita 10 novembre 2022
- [WIRED21] <https://www.wired.com/story/solarwinds-hack-supply-chain-threats-improvements/>, ultima visita 6 novembre 2022
- [ZAP22] <https://www.zaproxy.org/>, ultima visita 1 dicembre 2022

Appendice A - Obiettivi di Apprendimento/Livello Cognitivo di Conoscenza

Gli obiettivi specifici di apprendimento che si applicano a questo Syllabus sono indicati all'inizio di ogni capitolo. Ogni argomento del Syllabus sarà esaminato in base al suo obiettivo di apprendimento.

Gli obiettivi di apprendimento iniziano con un verbo di azione corrispondente al livello cognitivo di conoscenza, come elencato di seguito.

Livello 1: Ricordare (K1)

Il candidato è in grado di ricordare, riconoscere e richiamare un termine o un concetto.

Verbi di azione: richiamare, riconoscere.

Esempi
Richiamare i concetti della piramide di test.
Riconoscere gli obiettivi tipici del testing.

Livello 2: Comprendere (K2)

Il candidato è in grado di selezionare le ragioni o le spiegazioni delle affermazioni relative all'argomento ed è in grado di riassumere, confrontare, classificare e fornire esempi per il concetto del testing.

Verbi di azione: classificare, confrontare, differenziare, distinguere, spiegare, fornire esempi, interpretare, riassumere.

Esempi	Note
Classificare gli strumenti di test in base al loro scopo e alle attività di test che supportano.	
Confrontare i diversi livelli di test.	Può essere utilizzato per cercare somiglianze, differenze o entrambe le cose.
Distinguere il testing dal debugging.	Cerca le differenze tra i concetti.
Distinguere tra rischi di progetto e rischi di prodotto.	Consente di classificare separatamente due (o più) concetti.
Spiegare l'impatto del contesto sul processo di test.	
Fornire esempi del perché il testing è necessario.	

Esempi	Note
Ricavare la root cause dei difetti da un determinato profilo di failure.	
Riassumere le attività del processo di review del prodotto di lavoro.	

Livello 3: Applicare (K3)

Il candidato è in grado di eseguire una procedura quando si trova di fronte a un compito che gli è familiare, oppure è in grado di selezionare la procedura corretta e applicarla a un determinato contesto.

Verbi di azione: applicare, implementare, preparare, utilizzare.

Esempi	Note
Applicare l'analisi ai valori limite per ricavare i test case da determinati requisiti.	Dovrebbe riferirsi a una procedura / tecnica / processo ecc.
Implementare i metodi di raccolta delle metriche per supportare i requisiti tecnici e di management.	
Preparare test di installabilità per le applicazioni mobile.	
Utilizzare la tracciabilità per monitorare l'avanzamento del test, in modo da poter verificare la completezza e la consistenza con gli obiettivi del test, la strategia di test e il test plan.	Potrebbe essere utilizzato in un LO che richiede che il candidato sia in grado di utilizzare una tecnica o una procedura. Simile ad "applicare".

Livello 4: Analizzare (K4)

Il candidato è in grado di separare le informazioni relative a una procedura o a una tecnica nelle sue parti costituenti per una migliore comprensione, ed è in grado di distinguere tra fatti e inferenze.

Un'applicazione tipica è analizzare un documento, un software o una situazione di progetto, e proporre azioni appropriate per risolvere un problema o completare un'attività.

Verbi di azione: analizzare, analizzare in modo critico, delineare, prioritizzare, selezionare.

Esempi	Note
Analizzare una determinata situazione di progetto per determinare quali tecniche di test black-box o basate	Esaminabile solo in combinazione con un obiettivo misurabile dell'analisi.

Esempi	Note
sull'esperienza dovrebbero essere applicate per raggiungere obiettivi specifici.	Dovrebbe essere della forma "Analizzare xxxx per xxxx" (o simile).
Prioritizzare i test case in una determinata test suite per l'esecuzione, in base ai rischi di prodotto correlati.	
Selezionare i livelli di test e i tipi di test appropriati per verificare un determinato insieme di requisiti.	Necessario quando la selezione richiede un'analisi.

Riferimenti

(Per i livelli cognitivi degli obiettivi di apprendimento)

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

Appendice B - Matrice di Tracciabilità dei Business Outcome rispetto agli Obiettivi di Apprendimento

La tabella seguente mostra i Business Outcome e il numero degli Obiettivi di Apprendimento del Syllabus.

Business Outcome: Security Test Engineer		Numero di LO
ID	Descrizione	
STE-BO1	Comprendere i paradigmi fondamentali della sicurezza e il loro impatto sul testing di sicurezza	6
STE-BO2	Utilizzare e applicare tecniche di test di sicurezza appropriate e conoscerne i punti di forza e i limiti	7
STE-BO3	Contribuire alla pianificazione, alla progettazione e all'esecuzione dei test di sicurezza	5
STE-BO4	Comprendere come gli standard del testing di sicurezza e le best practice di sicurezza possano essere utilizzati per il testing di sicurezza	4
STE-BO5	Adattare ed eseguire attività di testing di sicurezza in base al contesto specifico dell'organizzazione	3
STE-BO6	Adattare ed eseguire attività di testing di sicurezza in base a metodi di sviluppo e cicli di vita dello sviluppo software specifici	4
STE-BO7	Inserire i risultati del testing di sicurezza in un information security management system (ISMS) per una gestione attiva dei rischi di sicurezza	4
STE-BO8	Raccogliere, valutare e aggregare i risultati dei test e redigere un test report dettagliato che includa tutte le prove e i risultati	3
STE-BO9	Sulla base di un approccio del testing di sicurezza richiesto, identificare i requisiti adeguati per gli strumenti e assistere nella selezione degli strumenti del testing di sicurezza	3

La tabella seguente mostra la tracciabilità tra i Business Outcome e gli Obiettivi di Apprendimento:

		Business Outcome									
Numero LO	Obiettivo di Apprendimento	Livello K	STE-BO1	STE-BO2	STE-BO3	STE-BO4	STE-BO5	STE-BO6	STE-BO7	STE-BO8	STE-BO9
STE-1.1.1	Spiegare i diversi livelli di sicurezza degli asset e il corrispondente livello di protezione	K2	x								
STE-1.1.2	Spiegare la relazione tra la sensibilità delle informazioni e il testing di sicurezza	K2	x								

STE-1.2.1	Descrivere il ruolo del testing di sicurezza nel contesto degli audit di sicurezza	K2	x								
STE-1.3.1	Spiegare il concetto di zero trust	K2	x								
STE-1.3.2	Applicare lo zero trust nel testing di sicurezza	K3	x								
STE-1.4.1	Esemplificare il concetto di riutilizzo del software open source nello sviluppo di software e il suo impatto sul testing di sicurezza	K2	x								
STE-2.1.1	Fornire esempi di tipi di test di sicurezza in base a un contesto di sicurezza black-box, white-box e grey-box	K2		x							
STE-2.1.2	Fornire esempi di tipi di test di sicurezza in base al testing di sicurezza statico o al testing di sicurezza dinamico	K2		x							
STE-2.2.1	Applicare test case di sicurezza, basati su un determinato approccio del test di sicurezza, insieme ai rischi di sicurezza funzionali e strutturali identificati	K3		x							
STE-2.2.2	Descrivere come testare la riconciliazione e la ricertificazione delle identità e delle autorizzazioni	K2		x							
STE-2.2.3	Descrivere come testare il controllo della gestione delle identità e degli accessi	K2		x							
STE-2.2.4	Descrivere come testare il controllo della protezione dei dati	K2		x							
STE-2.2.5	Descrivere come testare le tecnologie di protezione	K2		x							
STE-3.1.1	Spiegare le diverse attività, compiti e responsabilità all'interno di un processo di test di sicurezza	K2			x						
STE-3.1.2	Comprendere gli elementi chiave e le caratteristiche di un ambiente di test di sicurezza efficace	K2			x						
STE-3.2.1	Fornire esempi di test di sicurezza a livello di test di componente sulla base di una determinata base di codice	K2			x						
STE-3.2.2	Fornire esempi di test di sicurezza a livello di	K2			x						

	integrazione dei componenti sulla base di una determinata specifica di progettazione										
STE-3.2.3	Implementare un test di sicurezza end-to-end che validi uno o più requisiti di sicurezza relativi a uno o più processi di business	K3			x						
STE-4.1.1	Spiegare le diverse fonti degli standard e delle best practice e la loro applicabilità	K3				x					
STE-4.2.1	Applicare il concetto di Open Web Application Security Project, Common Vulnerability Enumeration, Common Weakness Enumeration, il Common Vulnerability Scoring System e il Common Weakness Scoring System e come utilizzarli per il testing di sicurezza	K3				x					
STE-4.3.1	Spiegare i vantaggi e gli svantaggi degli oracoli del test utilizzati per il testing di sicurezza	K2				x					
STE-4.3.2	Comprendere i vantaggi e gli svantaggi dell'utilizzo degli standard e delle best practice di sicurezza	K3				x					
STE-5.1.1	Analizzare un determinato contesto organizzativo e determinare quali aspetti specifici considerare per il testing di sicurezza	K3					x				
STE-5.2.1	Analizzare l'impatto delle normative sulle politiche di sicurezza e come testarle	K3					x				
STE-5.3.1	Analizzare uno scenario di attacco e identificare le possibili fonti e motivazioni dell'attacco	K4					x				
STE-6.1.1	Riassumere perché le attività di testing di sicurezza dovrebbero coprire il ciclo di vita dello sviluppo software	K2						x			
STE-6.1.2	Analizzare in che modo le attività di testing di sicurezza sono influenzate dai diversi modelli di ciclo di vita dello sviluppo software	K4						x			
STE-6.2.1	Definire ed eseguire regression test e test confermativi di sicurezza sulla base di una modifica al sistema	K3						x			

STE-6.2.2	Analizzare i risultati dei test di sicurezza per determinare la natura di una vulnerabilità e il suo potenziale impatto tecnico	K2						X			
STE-7.1.1	Comprendere i criteri di accettazione del testing di sicurezza e come questi influenzano la scelta degli approcci e delle tecniche di test di sicurezza	K2							X		
STE-7.2.1	Comprendere il ruolo del testing di sicurezza per un information security management system efficace	K2							X		
STE-7.3.1	Valutare la maturità dell'information security management system introducendo diversi approcci di test, nuovi oggetti di test o una copertura migliorata	K3							X		
STE-7.3.2	Comprendere la misurabilità all'interno di un information security management system	K2							X		
STE-8.1.1	Comprendere l'importanza dei risultati dei test di sicurezza e come questi influenzano la loro gestione e comunicazione	K2								X	
STE-8.2.1	Valutare i risultati del test di un determinato test di sicurezza per identificare le vulnerabilità	K3								X	
STE-8.3.1	Valutare diverse tecniche per chiudere le vulnerabilità identificate	K3								X	
STE-9.1.1	Analizzare diversi use case e applicare le categorizzazioni degli strumenti del testing di sicurezza	K3									X
STE-9.2.1	Comprendere l'utilizzo e i concetti degli strumenti del testing di sicurezza statico	K2									X
STE-9.2.2	Comprendere l'utilizzo e i concetti degli strumenti del testing di sicurezza dinamico	K2									X

Appendice C - Release Note

ISTQB® Security Test Engineer è una nuova release. Per questo motivo, non sono disponibili release note dettagliate per capitolo e sezione.

Appendice D - Termini Specifici del Dominio del Testing di Sicurezza

Nome del termine	Definizione
autenticazione	Una procedura per determinare se una persona o un processo è, di fatto, colui o quello che ha dichiarato di essere.
autorizzazione	Permesso dato a un utente o processo per accedere alle risorse.
crittografia (encryption)	Il processo di codifica delle informazioni in modo che soltanto le parti autorizzate possano recuperare l'informazione originale, normalmente per mezzo di un processo o di una chiave di decryption specifica.
escalation dei privilegi	Lo sfruttamento di un bug o di un difetto che consente un livello di privilegio più elevato di quello normalmente consentito (secondo NIST)
firewall	Un componente o insieme di componenti che controllano il traffico di rete in entrata e in uscita, in base a predeterminate regole di sicurezza.
information security management system (ISMS)	Una parte di un sistema di management complessivo, basato su un approccio al rischio di business, volto a stabilire, implementare, operare, monitorare, sottoporre a review, mantenere e migliorare la sicurezza delle informazioni (information security).
Open Web Application Security Project (OWASP)	OWASP, l'Open Web Application Security Project, elenca le debolezze comuni ed è piuttosto famoso per la pubblicazione della classifica OWASP Top 10.
politica di sicurezza	Un documento di alto livello che descrive i principi, l'approccio e gli obiettivi principali dell'organizzazione, che riguardano la sicurezza.
rootkit	Una serie di strumenti utilizzati da un attacker per ottenere e mantenere l'accesso a livello di root a un host per nascondere le proprie attività con mezzi segreti. (secondo NIST)
sensibilità delle informazioni	Una misura dell'importanza di proteggere le informazioni assegnata dal proprietario (secondo NIST).
servizio di sicurezza	Una capacità che supporta uno o più obiettivi di sicurezza. (secondo NIST)
social engineering	Un tentativo di raggirare qualcuno nel rivelare informazioni (ad esempio, password) che possano essere usate per attaccare sistemi o reti.

Nome del termine	Definizione
software composition analysis (SCA)	Una pratica per identificare i componenti open source e closed source in uso in un'applicazione, le loro vulnerabilità di sicurezza note e le restrizioni di licenza ostili (secondo NIST)
software open source	Un Software accessibile, utilizzabile, modificabile e condivisibile da chiunque.
STRIDE	Un acronimo che indica sei categorie di minacce (ovvero spoofing, tampering, repudiation, information disclosure, denial of service ed elevation of privilege) utilizzate per modellare le potenziali minacce a un sistema.
zero-trust	Un modello progettato per ridurre al minimo l'incertezza nell'applicazione di decisioni di accesso accurate e con privilegi minimi per ogni richiesta in un componente, sistema e servizio se una rete è considerata compromessa. (Glossario NIST)

Indice

Tutti i termini sono definiti nel Glossario ISTQB® (<http://glossary.istqb.org/>).