Certified Tester

Security Test Engineer Syllabus

versão 1.0.1

International Software Testing Qualifications Board

BSTQB®
CT-STE 1.0

Direitos autorais

Aviso de direitos autorais © International Software Test Qualifications Board (doravante ISTQB®)

ISTQB® é uma marca registrada do International Software Test Qualifications Board.

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

Todos os direitos reservados. Os autores transferem os direitos autorais para o ISTQB®. Os autores (como atuais detentores dos direitos autorais) e o ISTQB® (como futuro detentor dos direitos autorais) concordaram com as seguintes condições de uso:

Extratos deste documento, para uso não comercial, podem ser copiados se a fonte for citada. Qualquer Provedor de Treinamento Credenciado pode usar este syllabus como base para um curso de treinamento se os autores e o ISTQB® forem reconhecidos como a fonte e os proprietários dos direitos autorais do syllabus e desde que qualquer anúncio de tal curso de treinamento possa mencionar o syllabus somente após a Acreditação oficial dos materiais de treinamento ter sido recebida de um Conselho de Membros reconhecido pelo ISTQB®

Qualquer indivíduo ou grupo de indivíduos pode usar este syllabus como base para artigos e livros, desde que os autores e o ISTQB® sejam reconhecidos como a fonte e os proprietários dos direitos autorais do syllabus.

É proibido qualquer outro uso deste syllabus sem antes obter a aprovação por escrito do ISTQB®

Qualquer Conselho Membro reconhecido pelo ISTQB® pode traduzir este syllabus desde que reproduza o Aviso de Direitos Autorais mencionado acima na versão traduzida do syllabus.

Histórico da Revisão

Versão	Data	Observações
1.0	18-10-2024	Versão final
1.0.1	01-31-2025	Versão final após a revisão técnica

Histórico da versão de tradução do BSTQB

Data	Observações
07/03/2025	Lançamento da versão na língua portuguesa

©International Software Testing Qualifications Board

Índice

©International Software Testing Qualifications Board

©International Software Testing Qualifications Board

©International Software Testing Qualifications Board

Agradecimentos

Este documento foi formalmente divulgado pela Assembleia Geral do ISTQB® em 31 de janeiro de 2025

Ele foi produzido pela força-tarefa de testes de segurança do Conselho Internacional de Qualificações de Testes de Software: Dr. Frank Simon (presidente), Alain Ribault, Gabriel Firmino Barjollo, Michael Pott, Beata Karpinska, Maria Kispal, Frans Dijkman

As seguintes pessoas participaram da revisão, dos comentários e da votação deste 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 Rosa, Samuel Ouko, Tauhida Parveen, Tetsu Nagata

O BSTQB® agradece à equipe do Grupo de Traduções do BSTQB pelo empenho em traduzir este material. Atuaram na tradução e revisão: George Fialkovitz, Irene Nagase, Osmar Higashi, Paula Oliveira, Rogério Athaide de Almeida, Stênio Viveiros.

O BSTQB® também agradece aos ISTQB® Accredited Training Providers no Brasil que contribuíram na revisão da tradução com sugestões e questionamentos. No momento deste trabalho os seguintes provedores estavam credenciados: Conecteseaqui, Exseed, Iterasys, Keeggo.

O BSTQB® também agradece às empresas brasileiras credenciadas no ISTQB® Partner Program que contribuíram na revisão da tradução com sugestões e questionamentos. No momento deste trabalho as seguintes empresas estavam credenciadas: Deltapoint, Keeggo, Venturus.

©International Software Testing Qualifications Board

0 Introdução

0.1 Objetivo deste syllabus

Este syllabus forma a base para o International Software Test Qualification Security Test Engineer. O ISTQB® fornece esse syllabus da seguinte forma:

1. Aos conselhos membros, para traduzir para seu idioma local e credenciar os provedores de treinamento. Os conselhos membros podem adaptar o syllabus às suas necessidades específicas de idioma e modificar as referências para adaptá-las às suas publicações locais.
2. Aos órgãos de certificação, para que elaborem questões de exame em seu idioma local, adaptadas aos objetivos de aprendizagem deste syllabus.
3. Aos provedores de treinamento, para produzir material didático e determinar métodos de ensino adequados.
4. Para candidatos à certificação, para se preparar para o exame de certificação (como parte de um curso de treinamento ou de forma independente).
5. Para a comunidade internacional de engenharia de software e sistemas, para promover a profissão de teste de software e sistemas, e como base para livros e artigos.

0.2 Resultados do negócio

Esta seção lista os resultados de negócio esperados de um candidato, que tenha obtido a certificação Security Test Engineer Specialist Level.

Um testador certificado em Security Test Engineer pode ...

STE-BO1	Compreender os paradigmas fundamentais de segurança e seu impacto nos testes de segurança.
STE-BO2	Usar e aplicar técnicas adequadas de teste de segurança e conhecer seus pontos fortes e limitações.
STE-BO3	Contribuir para o planejamento, a criação e a execução de testes de segurança.
STE-BO4	Entender como os padrões de teste de segurança e as práticas recomendadas de segurança podem ser utilizados para testes de segurança.
STE-BO5	Ajustar e executar atividades de teste de segurança de acordo com o contexto específico da organização.
STE-BO6	Ajustar e executar atividades de teste de segurança de acordo com métodos de desenvolvimento específicos e ciclos de vida de desenvolvimento de software.
STE-BO7	Alimentar os resultados dos testes de segurança em um sistema de gerenciamento de segurança da informação (ISMS) para um gerenciamento ativo dos riscos de segurança.
STE-BO8	Coletar, avaliar e agregar resultados de testes e redigir um relatório de teste detalhado que inclua todas as evidências e descobertas.
STE-BO9	Com base em uma abordagem de teste de segurança necessária, identificar os requisitos adequados para as ferramentas e ajudar na seleção das ferramentas de teste de segurança.

©International Software Testing Qualifications Board

0.3 Objetivos de Aprendizagem Examináveis e Nível Cognitivo de Conhecimento

Os objetivos de aprendizagem apoiam os resultados de negócio e são usados para criar os exames de nível de Engenheiro de Testes de Segurança de Testes Certificados.

Em geral, todo o conteúdo deste syllabus pode ser examinado no nível K1. Ou seja, o candidato pode ser solicitado a reconhecer, lembrar ou recordar uma palavra-chave ou conceito mencionado em qualquer um dos nove capítulos. Os níveis dos objetivos específicos de aprendizagem são mostrados no início de cada capítulo, e classificados da seguinte forma:

- K1: Lembrar
- K2: Compreender
- K3: Aplicar
- K4: Analisar

Mais detalhes e exemplos de objetivos de aprendizagem são fornecidos no Apêndice A.

Todos os termos listados como palavras-chave e palavras-chave específicas do domínio de segurança devem ser passíveis de exame (K1), mesmo que não sejam explicitamente mencionados nos objetivos de aprendizagem.

0.4 Exame de certificação do Security Test Engineer

O exame de certificação Security Test Engineer será baseado nesse syllabus. O outro syllabus, Security Test Analyst, concentra-se na elaboração de testes de segurança que são executados posteriormente pelo Security Test Engineer.

As respostas às questões do exame podem exigir o uso de material baseado em mais de uma seção deste syllabus. Todas as seções do syllabus são passíveis de exame, exceto os Apêndices. Normas e livros são incluídos como referências, mas seu conteúdo não é passível de exame, além do que está resumido no próprio syllabus a partir dessas normas e livros.

Consulte o documento Exam Structures and Rules para o documento Security Test Engineer para obter mais detalhes.

O pré-requisito para fazer o exame Security Test Engineer é que os candidatos tenham interesse em testes de software. No entanto, é recomendável que os candidatos também tenham pelo menos uma formação mínima em desenvolvimento de software, teste de software ou teste de segurança.

Observação sobre os requisitos de entrada: O certificado ISTQB® Foundation Level deve ser obtido antes de fazer o exame de certificação Security Test Engineer.

A conclusão de um curso de treinamento credenciado não é um pré-requisito para o exame.

0.5 Credenciamento

Um Conselho Membro do ISTQB® pode credenciar provedores de treinamento cujo material do curso siga este syllabus. Os provedores de treinamento devem obter as diretrizes de credenciamento do Conselho de Membros ou do órgão que realiza o credenciamento. Um curso credenciado é reconhecido como estando em conformidade com este syllabus e pode ter um exame ISTQB® como parte do curso.

As diretrizes de credenciamento para este syllabus seguem as diretrizes gerais de credenciamento publicadas pelo Grupo de Trabalho de Gerenciamento de Processos e Conformidade.

0.6 Manuseio de Normas

As normas técnicas são referenciadas no Security Test Engineer Syllabus (p. ex., NIST e ISO). O objetivo delas é fornecer uma estrutura ou uma fonte de informações adicionais, se o leitor desejar. Observe que o syllabus usa os

©International Software Testing Qualifications Board

padrões como referência e que eles não se destinam a exame. Consulte o capítulo 4 para obter mais informações sobre o uso de padrões, práticas recomendadas e normas.

0.7 Nível de detalhe

O nível de detalhamento desse syllabus permite a realização de cursos e exames consistentes em nível internacional. Para atingir esse objetivo, o syllabus consiste em:

- Objetivos gerais de instrução que descrevem a intenção do Nível de Especialista
- Uma lista de termos (palavras-chave) que os alunos devem ser capazes de lembrar
- Uma lista de palavras-chave específicas do domínio associadas à segurança que os alunos devem ser capazes de lembrar
- Objetivos de aprendizagem para cada área de conhecimento, descrevendo os resultados cognitivos de aprendizado a serem alcançados
- Uma descrição dos principais conceitos, incluindo referências a fontes reconhecidas

O conteúdo do syllabus não é uma descrição de toda a área de conhecimento de testes de segurança; ele reflete o nível de detalhes a ser abordado nos cursos de treinamento de Security Test Engineer no nível de Especialista. Ele se concentra em conceitos e técnicas de teste que podem ser aplicados a todos os projetos de software, independentemente do ciclo de vida de desenvolvimento de software empregado.

0.8 Como este syllabus está organizado

Há nove capítulos com conteúdo passível de exame. O cabeçalho de nível superior de cada capítulo especifica o tempo de treinamento para o capítulo; o tempo não é fornecido para os subcapítulos. Para cursos de treinamento credenciados, o syllabus exige um mínimo de 1.290 minutos (cerca de 22 horas) de instrução, distribuídos nos nove capítulos da seguinte forma:

- Capítulo 1: Paradigmas de Segurança - 135 minutos.
- Capítulo 2: Técnicas de Teste de Segurança - 150 minutos.
- Capítulo 3: O processo de Teste de Segurança - 120 minutos.
- Capítulo 4: Padrões e Práticas Recomendadas de Teste de Segurança - 195 minutos.
- Capítulo 5: Ajustando o Teste de Segurança ao Contexto Organizacional - 195 minutos.
- Capítulo 6: Ajustando o Teste de Segurança aos Modelos de Ciclo de Vida de Desenvolvimento de Software - 165 minutos.
- Capítulo 7: Teste de Segurança como Parte de um Sistema de Gerenciamento de Segurança da Informação (ISMS) - 105 minutos
- Capítulo 8: Relatório dos Resultados dos Testes de Segurança - 135 minutos.
- Capítulo 9: Ferramentas de Teste de Segurança - 90 minutos.

©International Software Testing Qualifications Board

1 Paradigmas de Segurança - 135 minutos (K3)

Palavras-chave

disponibilidade, confidencialidade, integridade, teste de segurança, vulnerabilidade

Palavras-chave de segurança

software de código aberto, confiança zero

Objetivos de aprendizagem para o Capítulo 1:

1.1 Níveis de segurança de ativos

STE-1.1.1 (K2) Explicar os diferentes níveis de segurança dos ativos e seu nível de proteção correspondente.

STE-1.1.2 (K2) Explicar a relação entre a sensibilidade das informações e segurança.

1.2 Auditorias de segurança

STE-1.2.1 (K2) Descrever a função dos testes de segurança no contexto das auditorias de segurança.

1.3 O conceito de confiança zero

STE-1.3.1 (K2) Explicar o conceito de confiança zero.

STE-1.3.2 (K3) Aplicar a confiança zero na segurança.

1.4 Software de código aberto

STE-1.4.1 (K2) Exemplificar o conceito de reutilização de software de código aberto no desenvolvimento de software e seu impacto nos testes de segurança.

©International Software Testing Qualifications Board

1.1 Níveis de Segurança de Ativos

Um ativo é qualquer coisa que tenha valor em uma organização e que, portanto, exija proteção. Os ativos permitem que as organizações operem processos de negócios e tomem decisões. Todo ativo de informações e dados é vulnerável e, portanto, deve ser protegido. Os ativos são objetos da segurança da informação. Os ativos podem ser pessoas, informações, software, hardware, funções, processos e instalações de reputação corporativa [Chapple 2021]. Alguns exemplos:

- ativos de software: sistema operacional, aplicativos e bancos de dados.
- ativos de informação: planos de negócios, documentação, invenções, fotos e registros pessoais.
- ativos de hardware: sistemas de computador, armazenamento de dados e dispositivos de comunicação de dados.
- ativos físicos: instalações.

1.1.1 Ativos e seu nível de proteção correspondente

O valor de um ativo é determinado por três pilares da segurança da informação (chamados de tríade CIA): confidencialidade, integridade e disponibilidade [Stallings18].

A **confidencialidade** (*confidentiality*) busca evitar a divulgação não autorizada de informações. A perda de confidencialidade ocorre quando as informações são divulgadas a uma parte ou a um sistema não autorizado [Stallings18].

A **integridade** (*integrity*) busca evitar que os dados sejam modificados ou excluídos por uma parte não autorizada. [Stallings18].

A **disponibilidade** (*availability*) garante que as informações estejam disponíveis quando necessário. [Stallings18].

Os três pilares da segurança da informação podem ser classificados nos seguintes níveis: alto, médio e baixo. Quanto mais alto for o nível de segurança, maior será a exigência de implementação do nível de proteção (salvaguardas). As salvaguardas são, por exemplo, funções e restrições de segurança, segurança de pessoal e segurança de estruturas físicas, áreas e dispositivos

A perda de confidencialidade, integridade ou disponibilidade pode afetar as operações, os ativos, as pessoas, os clientes ou até mesmo os países da organização.

A classificação dos ativos define os níveis de sensibilidade e confidencialidade dos ativos. Isso ajuda as organizações a implementarem controles de segurança e níveis de proteção adequados.

Se o ativo tiver baixa sensibilidade, um nível de proteção poderá ser definido como nível de segurança baixo, e o teste de segurança poderá não ser necessário. O pilar adequado para isso é a disponibilidade.

Um exemplo de baixa sensibilidade pode ser quando a disponibilidade de informações é de maior importância para muitas pessoas, como informações de trânsito.

Se o ativo tiver sensibilidade média, o nível de proteção será definido como nível de segurança médio, e serão necessários testes de segurança. O pilar adequado para isso é a integridade.

Um exemplo de sensibilidade média pode ser quando as pessoas precisam de informações precisas de fontes confiáveis, como autoridades.

Se o ativo tiver alta sensibilidade, um nível de proteção será definido como um nível de segurança alto, e os testes de segurança serão obrigatórios. O pilar adequado para isso é a confidencialidade. Um exemplo de alta sensibilidade são as informações pessoais dos funcionários.

1.1.2 Sensibilidade das informações e teste de segurança

A sensibilidade das informações representa o grau em que as informações exigem proteção para garantir sua confidencialidade, integridade e disponibilidade [Glossário do NIST]. Como as consequências da violação de

©International Software Testing Qualifications Board

informações confidenciais podem variar de pequenas a desastrosas, um teste de segurança deve ser feito antes de qualquer violação.

O objetivo dos testes de segurança é verificar se uma implementação protege os dados e mantém a funcionalidade conforme planejado. Quanto mais proteção um ativo precisar, mais ações de segurança serão necessárias.

Os testes de segurança ajudam a identificar pontos fracos e vulnerabilidades e verificam se os controles de segurança adequados foram implementados. Os testes de segurança não podem garantir que um sistema ou uma organização estará livre de vulnerabilidades.

Essas etapas incluem a realização de testes de segurança para atingir os seguintes objetivos:

- Avaliar da eficácia dos controles de segurança existentes
- Descobrir de pontos fracos e vulnerabilidades
- Estabelecer uma estratégia de teste de segurança que inclua testes de confirmação para acompanhar o progresso de quaisquer patches de software e atualizações de longo prazo

Para o Engenheiro de Testes de Segurança (STE), uma avaliação de risco de segurança feita a partir de uma perspectiva de sensibilidade da informação pode ser uma fonte rica de informações a partir da qual os testes de segurança podem ser planejados e criados. Além disso, uma avaliação de risco de segurança pode ser usada para priorizar os testes de segurança, de modo que os riscos e as prioridades possam ser determinados e aqueles com os níveis de risco mais altos sejam direcionados para testes mais rigorosos. A avaliação de risco é apenas um retrato do momento atual e pode se basear em informações limitadas.

1.2 Auditorias de Segurança

Uma auditoria de segurança é uma revisão e um exame independente da segurança do sistema de informações de uma organização, controlando a conformidade com um conjunto estabelecido de critérios. As auditorias têm o objetivo de determinar a adequação dos controles do sistema, garantir a conformidade com uma política e procedimentos de segurança estabelecidos. Mas também para detectar violações nos serviços de segurança e recomendar quaisquer alterações indicadas para contramedidas [Glossário do NIST].

1.2.1 Auditorias e testes de segurança

As auditorias de segurança têm as seguintes características:

- Elas podem ser realizadas por auditores internos ou externos.
- Elas se concentram em aspectos dos processos e controles de segurança de uma organização, tais como:
    - Componentes físicos do sistema de informações e o ambiente no qual as informações são armazenadas;
    - Aplicativos e software, incluindo patches e configurações de segurança;
    - Controles para direitos e privilégios do usuário;
    - Vulnerabilidades da rede, incluindo avaliações da conexão entre diferentes pontos dentro e fora da rede da organização;
    - Como os funcionários coletam, compartilham e armazenam informações;
    - Mecanismos de detecção de intrusão;
    - Planos de resposta no caso de uma violação.
- Elas são um tipo de teste estático (consulte a seção 2.1.2) que envolve o exame manual de produtos de trabalho ou revisões automatizadas com ferramentas de auditoria de segurança.
- Elas investigam aspectos das políticas, procedimentos e controles de segurança de uma organização que são difíceis de testar dinamicamente.

©International Software Testing Qualifications Board

- Elas verificam a eficácia dos controles de segurança instalados e identificam onde os critérios definidos pela organização foram ou não alcançados em um determinado momento.
- Elas não garantem que todas as vulnerabilidades serão encontradas, mas garantem que as áreas problemáticas sejam identificadas e indicam onde é necessário tomar medidas corretivas.

As auditorias de segurança devem fazer parte da rotina regular. Elas podem ser feitas como:

- Avaliações pontuais para circunstâncias especiais, como quando uma organização introduz uma nova plataforma de software ou uma nova integração. Um teste e uma auditoria de segurança devem ser realizados para descobrir novos riscos e/ou defeitos.
- Auditorias programadas regularmente para verificar se os processos e procedimentos de segurança estão sendo seguidos e se são adequados ao clima e às necessidades atuais dos negócios.

Em algumas abordagens de auditoria de segurança, o teste de segurança é realizado como parte do processo de auditoria para determinar se os controles de segurança estão realmente implementados e funcionando de forma eficaz. Entretanto, o escopo de uma auditoria de segurança é muito maior do que o teste de segurança.

Os testes de segurança e a auditoria trabalham juntos. A auditoria identifica as deficiências e as áreas importantes a serem testadas. O teste de segurança é o meio pelo qual se prova ou não que os controles de segurança estão realmente implantados e funcionando de forma eficaz.

1.3 O Conceito de Confiança Zero (Zero Trust)

1.3.1 O que é zero trust?

A confiança zero ou zero trust é um modelo de segurança criado a partir de uma coleção de conceitos e ideias projetados para minimizar a incerteza na aplicação de acesso preciso e com o mínimo de privilégio por solicitação. Ele se baseia no princípio de controles de acesso rigorosos e de não confiar em ninguém por padrão, mesmo que todos já estejam dentro do perímetro da rede.

O modelo de confiança zero incorpora um princípio de "não confiar em nada, verificar tudo, a face de uma rede é vista como comprometida".

O aumento do uso de serviços on-line resultou em um aumento correspondente de vulnerabilidades e ataques. As informações geralmente estão espalhadas pelos fornecedores de nuvem, o que fez com que os conceitos de segurança baseados em perímetro, se tornassem menos eficazes no fornecimento de segurança para organizações, empregadores, usuários e clientes. A segurança tradicional baseada em perímetro confia em qualquer pessoa e em qualquer coisa dentro da rede. O problema com essa abordagem é que, quando um invasor obtém acesso à rede, ele tem acesso a todos os ativos internos.

Usando o modelo de confiança zero, os sistemas e serviços de informação operam sob a suposição de que suas redes já estão comprometidas e que a rede não tem espaço confiável. A perspectiva de Confiança Zero faz com que a prática que move as defesas de segurança de perímetros estáticos e baseados em rede se concentre em usuários, ativos e recursos [Glossário do NIST].

Benefícios da confiança zero [Cloudflare]:

- Reduzir a superfície de ataque de uma organização;
- Minimizar os danos de um ataque ao restringir a violação a uma pequena área e reduz o custo de recuperação;
- Reduzir o impacto do roubo de credenciais do usuário e dos ataques de phishing, exigindo vários fatores de autenticação.

1.3.2 Confiança zero em testes de segurança

As redes atuais não têm perímetro, migrando de implantações planas para ambientes dinâmicos, distribuídos e híbridos. As organizações se adaptam à crescente complexidade de seus ambientes, que abrange locais de trabalho híbridos, funcionários remotos, interações com outras organizações e fornecedores, e precisam proteger pessoas, dispositivos, aplicativos, redes e dados onde quer que estejam. O modelo de Confiança Zero, com a

©International Software Testing Qualifications Board

premissa "não confie em nada, verifique tudo", impulsiona a necessidade de uma mudança completa de paradigma na segurança de TI, em que pessoas, dispositivos, aplicativos, redes e dados devem passar por validações rigorosas antes de obter acesso a um aplicativo ou recurso solicitado.

Os princípios de confiança zero [Micro22] [Cloudflare] incluem:

- *Monitoramento e verificação contínuos*: sempre verifique o acesso a todos os recursos. Os logins e as conexões são interrompidos periodicamente, forçando os usuários e os dispositivos a serem continuamente reverificados.
- *Princípio do "acesso com o mínimo de privilégios"*: dê aos usuários apenas o acesso necessário.
- *Autenticação multifatorial*: requer mais de uma evidência para autenticar um usuário. A simples digitação de uma senha não é suficiente para permitir o acesso.
- *Monitoramento de segurança de endpoints de dispositivos que acessam dados, como laptops, estações de trabalho, tablets, dispositivos móveis*: todo endpoint remoto pode ser o ponto de entrada para um ataque.
- *Microssegmentação*: dividir os perímetros de segurança em pequenas zonas para obter acesso separado para partes distintas da rede.
- *Nenhum movimento lateral dentro das redes sem validação contínua*: o acesso de confiança zero é segmentado e deve ser restabelecido periodicamente. Um invasor não pode se deslocar para outros microssegmentos da rede. Quando a presença do invasor é detectada, o segmento ou a conta de usuário comprometida pode ser colocada em quarentena e impedida de ser acessada.
- *Proteção dos dados em todos os arquivos e conteúdos*: criptografia e restrições de acesso com base em políticas organizacionais.

O modelo de confiança zero afeta a forma como os testes de segurança devem ser gerenciados. Isso significa concentrar os casos de teste nos mecanismos de segurança de confiança zero. Os testes de segurança contribuem para identificar os seguintes possíveis pontos fracos:

- Rede: Microssegmentos de confiança zero, reduzindo danos e destacando violações
  - Testar o tráfego de segmentos cruzados com automação e com microssegmentos definidos pelo usuário.
- Dados: A confiança zero exige que os dados sejam transmitidos de forma segura e criptografada.
  - Testar os pontos de extremidade que expõem ou armazenam dados usando métodos não criptografados.
- Identidade: pessoas, dispositivos e processos só podem fazer o que têm permissão para fazer.
  - Testar se as pessoas, os dispositivos e/ou os processos têm mais permissões de acesso do que as necessárias, o que pode comprometer os ativos da rede.
  - Verificar se usuários não autorizados podem acessar segmentos dos recursos da rede.
- Os dispositivos devem executar somente software seguro e serem monitorados e gerenciados de forma centralizada.
  - Testar se o dispositivo está protegido com software de segurança e realizar testes de penetração (*pentest*).
- Limitação do raio da explosão:
  - Os testes são realizados regularmente para mostrar os pontos fracos e mitigar o impacto do risco, a fim de limitar o raio da explosão caso ocorra uma violação externa ou interna.

1.4 Software de Código Aberto (OSS)

O software de código aberto ou *open-source software* (OSS) é desenvolvido e mantido por meio de colaboração aberta e está disponível, normalmente sem custo, para qualquer pessoa usar, alterar e redistribuir como quiser.

©International Software Testing Qualifications Board

1.4.1 O conceito de OSS e seu impacto nos testes de segurança

O OSS é desenvolvido com base em código-fonte aberto para ser usado, modificado e compartilhado livremente por qualquer pessoa. O OSS é frequentemente distribuído sob licenças que atendem à definição de código aberto, conforme fornecido pela Open-Source Initiative.

O principal benefício do uso do OSS é a transparência do código e a suposição de que muitos desenvolvedores voluntários verificaram o código para detectar e resolver defeitos. A suposição é que essa abertura fará com que mais pessoas se envolvam na identificação rápida de vulnerabilidades e na correção de defeitos.

No entanto, o fato de esses aplicativos, bibliotecas e outros objetos reutilizáveis estarem disponíveis em todo o mundo representa um desafio, pois qualquer pessoa pode atualizar o código e, possivelmente, introduzir vulnerabilidades e vetores de ataque [Shacklett]. O OSS geralmente tem mais vetores de ataque do que o software fechado (proprietário), porque qualquer pessoa pode adicionar backdoors intencionais, propagar vulnerabilidades por meio da reutilização e explorar vulnerabilidades divulgadas publicamente e erros humanos. Quando uma vulnerabilidade é publicada, os usuários dessa versão do software correm o risco de sofrer um ataque.

O uso de OSS significa que cada *exploit* publicado para um componente específico pode afetar milhares de sistemas. Quando o código-fonte está disponível em versões executáveis, a observação, a engenharia reversa, as revisões de código, a desmontagem e os testes exploratórios podem ser capazes de encontrar vulnerabilidades [Stallings18].

O STE (Security Test Engineer) executa as seguintes tarefas:

- Identificar vulnerabilidades de código aberto
- Realizar revisões de código como parte do *shift-left*, que visa à detecção de defeitos no início do processo de desenvolvimento.

Quando se trata de testar a segurança de aplicativos OSS, o STE pensa como um invasor. Os casos de teste capturam como um aplicativo se comporta em diferentes cenários de uso e uso indevido e permitem que os desenvolvedores implementem as mitigações de risco adequadas.

As revisões de código são realizadas por desenvolvedores e STEs, tanto do lado do produtor quanto do lado do consumidor, para identificar códigos não seguros.

O *Open Web Application Security Project* (OWASP) disponibilizou ferramentas automatizadas de detecção de vulnerabilidades que são gratuitas para projetos de código aberto [OWASP Top 10]. O NIST oferece orientação para a segurança de OSS [NIST SP 800-161].

©International Software Testing Qualifications Board

2 Técnicas de Teste de Segurança - 150 minutos (K3)

Palavras-chave

teste destrutivo, teste de fuzz, varredura de malware, varredura de vulnerabilidades

Palavras-chave de segurança

autenticação, autorização, criptografia,

Objetivos de aprendizagem para o Capítulo 2:

2.1 Aplicação de tipos de teste de segurança de acordo com um contexto de teste

STE-2.1.1 (K2) Exemplificar os tipos de teste de segurança de acordo com um contexto de segurança caixa-preta, caixa-branca e caixa-cinza.

STE-2.1.2 (K2) Exemplificar os tipos de testes de segurança de acordo com testes de segurança estáticos ou dinâmicos.

2.2 Aplicação de tipos de teste de segurança de acordo com um projeto e um contexto técnico

STE-2.2.1 (K3) Aplicar casos de teste de segurança, com base em uma determinada abordagem de teste de segurança, juntamente com os riscos de segurança funcionais e estruturais identificados.

STE-2.2.2 (K2) Descrever como testar a reconciliação e a recertificação de identidades e permissões.

STE-2.2.3 (K2) Descrever como testar o controle do gerenciamento de identidade e acesso.

STE-2.2.4 (K2) Descrever como testar o controle de proteção de dados.

STE-2.2.5 (K2) Descrever como testar as tecnologias de proteção.

©International Software Testing Qualifications Board

2.1 Aplicação dos Tipos de Teste de Segurança de Acordo com um Contexto de Teste

2.1.1 Testes caixa-preta, caixa-branca e caixa-cinza

Os tipos de teste para a base de teste são classificados como teste caixa-preta, teste caixa-cinza e teste caixa-branca [ISTQB FL]. O Glossário ISTQB [Glossário ISTQB] define teste caixa-preta e teste caixa-branca.

O teste caixa-cinza de segurança é definido no [NIST] como "uma metodologia de teste que pressupõe algum conhecimento da estrutura interna e dos detalhes de implementação do objeto de avaliação". O STE tem acesso a algumas informações sobre o sistema em teste (SUT) (p. ex., um subconjunto de um mapa de endereçamento de rede, um subconjunto de documentação de arquitetura, um acesso de usuário e um acesso a uma máquina interna).

O STE tem acesso a todas as informações necessárias sobre o SUT durante o teste de segurança:

- A arquitetura do SUT;
- O código-fonte;
- Fluxos de dados no SUT;
- Projeto de rede e estrutura de zonas;
- Requisitos de senha;
- Regras de firewall;
- Autenticação;
- Armazenamento de registros e informações de gerenciamento.

A escolha das técnicas de teste baseia-se nos objetivos da abordagem de teste de segurança, bem como na disponibilidade do código. O STE deve decidir sobre o nível de profundidade dos testes e se deve se concentrar em ameaças externas ou internas.

2.1.2 Teste de segurança estático e dinâmico

Tanto os testes estáticos quanto os dinâmicos são usados nos testes de segurança para proteger o SUT durante todo o seu ciclo de vida.

Teste estático de segurança

Do ponto de vista do STE, entre os produtos de trabalho que podem ser revisados durante o teste de segurança estático estão:

- Documentação de análise de risco de segurança;
- Requisitos de segurança;

Ao procurar por lacunas nos requisitos, os seguintes mecanismos de segurança devem ser considerados:

- Gerenciamento de usuários;
- Autenticação;
- Autorização;
- Confidencialidade;
- Integridade;
- Responsabilidade;
- Gerenciamento de sessões;

©International Software Testing Qualifications Board

- Segurança no transporte;
- Segregação de sistemas em camadas;
- Conformidade legislativa e normativa, incluindo privacidade, padrões governamentais e do setor;
- Documentação técnica de segurança arquitetônica;
- Código-fonte;
- Definição de configuração e instalação de infraestrutura/operacional.

Teste dinâmico de segurança

Em comparação com o teste de segurança estático, o objetivo do teste de segurança dinâmico é verificar se o SUT implementa e usa corretamente as funções de segurança conforme exigido ou especificado e se essas funções de segurança não podem ser contornadas durante o uso do sistema ou aplicativo.

Os testes dinâmicos de segurança podem ser realizados por:

- Usar o teste caixa-preta para avaliar as funções de segurança, verificando se os resultados do teste estão de acordo com o esperado quando determinadas entradas são enviadas
- Teste de penetração (*pentest*): Procurar vulnerabilidades que possam ser exploradas.

Recomenda-se executar testes dinâmicos de segurança da seguinte maneira nos seguintes ambientes, se possível:

- Ambiente de Desenvolvimento
  Essa é a primeira oportunidade para o STE executar testes de segurança. Nesse ambiente, o STE verifica se a implementação da segurança está em conformidade com os requisitos de segurança (p. ex., o desenvolvimento correto de uma senha controlando o tamanho mínimo, o tamanho máximo e os tipos obrigatórios de caracteres).
- Ambiente de Teste e Ambiente de Pré-Produção
  Geralmente conhecido como ambiente de teste de preparação ou de aceite, esse ambiente deve ser o mais semelhante possível ao ambiente de produção e deve conter todas as medidas de segurança pretendidas que serão aplicadas no ambiente de produção.
- Ambiente de Produção
  Esse é o ambiente mais crítico. O STE deve tomar cuidado para não desativar o SUT e deve realizar auditorias de segurança para manter o sistema seguro. O objetivo do teste de segurança no ambiente de produção é verificar se as vulnerabilidades recém-descobertas e conhecidas foram corrigidas.

O uso de uma ferramenta de teste dinâmico de segurança de aplicativos (DAST - *dynamic application security test*) permite a detecção de condições que podem levar a vulnerabilidades (consulte o capítulo 9). O DAST pode ser incluído em um pipeline de integração contínua/entrega contínua (CI/CD) nos seguintes estágios:

- Durante o teste após uma compilação, funcionará como um scanner de segurança dinâmico para detectar defeitos de segurança.
- Durante a produção, funcionará como um scanner de vulnerabilidade

2.2 Aplicação de Teste de Segurança

A modelagem do teste de segurança pode ser baseada nas seguintes fontes:

- Regulamentos, padrões, normas e leis (p. ex., um novo regulamento que exige que os carros tenham um teste de segurança antes da aprovação);
- Uma análise de risco concluída;
- Modelos de ameaças disponíveis;
- Uma categorização ad-hoc dos riscos de segurança (consulte [ISTQB_ATTA_SYL]);
- Uma abordagem de teste de segurança;

©International Software Testing Qualifications Board

- Requisitos de segurança de funções e mecanismos de segurança;
- Sistemas e produtos no ciclo de vida de desenvolvimento de software (SDLC);
- A experiência e as habilidades de teste do STE;
- Incidentes anteriores em que a segurança foi (quase) violada;
- Um guia de teste de referência, como o [OWASP Test Guide].

Veja a seguir os atributos de um teste de segurança que devem ser considerados durante o projeto do teste de segurança:

- Regulamentos, normas ou leis exigidos (obrigatórios);
- Riscos de segurança identificados e modelos de ameaças priorizados pela abordagem de teste de segurança;
- Rastreado até os requisitos de segurança definidos;
- Definido de acordo com os testadores pretendidos (p. ex., desenvolvedores, testadores funcionais e STE);
- Definido de acordo com perfis de defeitos de segurança e vulnerabilidades conhecidas;
- Projetado para ser automatizado, se aplicável;
- Testes destrutivos ou não destrutivos;
- Intrusivo ou não intrusivo (p. ex., o objetivo é quebrar um sistema por dentro ou derrubar um sistema por meio de uma negação de serviço distribuída).

O fluxo de trabalho básico do projeto de teste de segurança é:

1°) Analisar a abordagem de teste de segurança (no nível do projeto);

2°) Analisar os riscos de segurança, modelos de ameaças e requisitos (no nível do projeto);

3°) Aplicar as técnicas de teste de segurança.

Na maioria dos casos, um projeto de teste de segurança eficiente é baseado em uma combinação das fontes acima. Dependendo do tipo de projeto, é importante garantir que um teste de segurança seja realizado em todas as fases do SDLC do SUT.

2.2.1 Como lidar com os riscos de segurança no projeto de testes

Um princípio fundamental é que o STE deve ser capaz de criar e implementar casos de teste de segurança com base em qualquer risco de segurança identificado, requisito de segurança, ameaça e experiência.

Os testes de segurança podem se basear em riscos externos de segurança no ambiente de produção (ameaças a um sistema ou produto), em uma abordagem de testes de segurança e em outras fontes, como modelos de ameaças. Os riscos de segurança também podem ser vistos como de natureza funcional e estrutural (ou seja, riscos devidos à falta de segurança por projeto).

Durante o projeto do caso de teste de segurança, o STE deve identificar se um teste é destrutivo ou não destrutivo. Se um teste for identificado como destrutivo, ele deverá garantir que não tenha nenhum impacto negativo sobre outras atividades de teste, ambientes ou negócios.

A seguir, são descritos os riscos e as vulnerabilidades de segurança comuns nos níveis funcional e estrutural, juntamente com as técnicas de teste de segurança adequadas.

Controles funcionais de segurança, riscos ou vulnerabilidades

Os testes são projetados para verificar e validar se os controles estão em vigor, se funcionam corretamente e se são eficazes na detecção e prevenção de ações não autorizadas.

As técnicas de teste de segurança são baseadas em requisitos para controles de segurança funcionais e controles de acesso funcionais.

©International Software Testing Qualifications Board

Controles de acesso estrutural, riscos ou vulnerabilidades

Os testes para esses controles baseiam-se em como os direitos do usuário foram estabelecidos para acesso aos dados, acesso funcional e níveis de privacidade. Os controles de acesso estrutural são normalmente aplicados por um administrador de sistema, administrador de segurança ou administrador de banco de dados. Em alguns casos, os direitos de acesso são uma opção de configuração em um aplicativo. Em outros casos, os direitos de acesso são aplicados em um nível de infraestrutura do sistema.

Riscos ou vulnerabilidades de acesso ao sistema operacional

Uma vez obtido o acesso ao sistema operacional, o invasor pode controlar processos, dados e acesso à rede, o que pode permitir a inserção de malware.

Riscos ou vulnerabilidades da plataforma

Cada plataforma tem seu próprio conjunto de vulnerabilidades de segurança.

As técnicas de teste de segurança baseiam-se na experiência do STE (p. ex., ao lidar com vulnerabilidades) e no teste de procedimentos de segurança (p. ex., testar a manutenção das condições de segurança).

Ameaças externas e internas

Algumas ameaças, como a exploração de vulnerabilidades de aplicativos ou de linguagens de programação, podem ser detectadas, testadas e seu impacto limitado. As ameaças internas são executadas por funcionários internos. As ameaças externas são executadas por pessoas externas (p. ex., invasores).

As técnicas de teste de segurança são baseadas em testes exploratórios (p. ex., para encontrar alvos potenciais e pontos de injeção/vetores de ataque úteis) e na experiência do STE.

2.2.2 Teste de reconciliação e teste de recertificação

Entendendo as preocupações com o gerenciamento de identidade e acesso

Os serviços comerciais fornecidos pelas organizações estão aumentando em complexidade. Eles são implantados como um sistema de sistemas e hospedados em ambientes híbridos que consistem em elementos internos, fornecidos por parceiros, fornecidos por clientes e na nuvem. Nesse ambiente distribuído, o gerenciamento da segurança das contas e dos direitos dos usuários é fundamental. Por exemplo:

- O usuário deve ter os privilégios corretos, e não mais
- Os direitos devem ser revogados depois que o funcionário deixar a organização
- O gerenciamento de direitos deve estar em conformidade com os regulamentos, por exemplo, o Lei Geral de Proteção a Dados [LGPD 13.709].

O gerenciamento de identidade e acesso (IAM - *Identity and access management*) é uma disciplina para gerenciar e manter contas e direitos de usuários, definindo quem (identidade) está disposto a acessar o quê (função) para um recurso específico. O IAM lista dois subprocessos, que estão diretamente relacionados aos testes de segurança:

- Reconciliação:
  A comparação e a atualização do acesso do usuário, dos direitos e das contas privilegiadas por meio de solicitações de alteração e cadeias de aprovação para um banco de dados confiável e autorizado de gerenciamento de identidade.
- Recertificação:
  Revisões regulares de contas e direitos e privilégios relacionados para verificar se ainda são necessários.

A reconciliação é necessária para garantir que todos os acessos a aplicativos sejam sincronizados com a mesma "fonte confiável". Os níveis do processo de reconciliação são:

- *Completo*: comparação de todas as contas e atributos de acesso do usuário com o sistema de gerenciamento de identidade e acesso, com o objetivo de identificar quaisquer diferenças
- *Incremental*: Comparar apenas as alterações em contas e direitos criados, atualizados ou excluídos

©International Software Testing Qualifications Board

- *Automático*: Quando são usados aplicativos que podem programar comparações automáticas de quaisquer alterações relevantes de segurança feitas

A recertificação consiste em auditar uma conta de usuário e os privilégios de acesso para determinar se eles ainda são justificados, consistentes com as políticas internas da organização e em conformidade com as normas. Isso envolve a realização de uma auditoria contínua para garantir que os usuários tenham acesso apenas ao que precisam e para o que têm permissão. A avaliação pode ser:

- Manual
  - Extrair e agrupar informações contábeis
  - Apresentar informações
  - Revisão por gerentes
- Automatizado
  - Mensagens são enviadas aos gerentes para emitir solicitações de recertificação.
  - Isso tem a vantagem de poder planejar auditorias regularmente.

Em uma grande organização, com sistemas hospedados em uma grande variedade de ambientes, o IAM torna-se complexo devido à necessidade de gerenciar vários aplicativos, cada um com acessos a serem concedidos e revogados de acordo com os movimentos de um usuário (p. ex., chegada, saída e transferência). Em algumas organizações, as contas e os privilégios dos usuários são gerenciados fora de um processo formal de IAM para economizar tempo. Esse é um problema crítico do ponto de vista da segurança, pois contas órfãs e não utilizadas podem resultar em exploração por um invasor.

Como executar testes de reconciliação e testes de recertificação para mecanismos de identidades e permissões

Testes de reconciliação e testes de recertificação devem ser realizados para evitar inconsistências nas contas de usuários em todos os aplicativos aos quais o usuário tem acesso. Isso inclui aspectos como credenciais e privilégios de login.

Aplicam-se as seguintes condições de teste:

- Gerenciamento de novas contas;
- Modificação de credenciais e privilégios da conta;
- Realocação de uma conta, incluindo a remoção do acesso a aplicativos e a adição de acesso a novos aplicativos;
- Revisão de todas as contas.

Os objetivos do teste podem ser:

- Tentativa de trocar, alterar ou acessar outra função;
- Revisar a granularidade das funções e as necessidades por trás das permissões concedidas;
- Verificar se os requisitos de identidade para o registro de usuários estão alinhados com os requisitos comerciais e de segurança.

As técnicas de teste de segurança para reconciliação e recertificação incluem:

- Revisão da documentação dos processos de reconciliação e recertificação;
- Verificar se os registros foram examinados por uma pessoa antes do provisionamento ou se são concedidos automaticamente quando determinados critérios são atendidos;
- Verificação, habilitação e autorização de solicitações de desprovisionamento;
- Verificar se as modificações na conta são efetivas;
- Realização de testes de fuzz em possíveis funções para garantir que o sistema rejeite funções com fuzz;
- Revisar as permissões de função após reunir todas as modificações aplicadas.

Observe que o [OWASP Test Guide] fornece uma lista de objetivos de teste de segurança e técnicas de teste relacionadas ao teste de IAM.

©International Software Testing Qualifications Board

2.2.3 Teste de identificação, autenticação e autorização

Entendendo a relação entre autenticação e autorização

Os ativos confidenciais de uma organização devem ser protegidos e só podem ser acessados por uma pessoa autorizada que tenha sido previamente autenticada. A identificação é a primeira etapa para obter acesso a um recurso. É o processo de afirmação de uma identidade.

A autenticação é baseada na verificação de um identificador de usuário e de um token para responder às perguntas:

- Quem é o usuário (identificador de usuário);
- O usuário é realmente quem ele alega ser? (token, como uma senha ou certificado)

Diferentes implementações de mecanismos de autenticação podem ser usadas, dependendo do nível de proteção a ser dado contra-ataques para sequestrar uma autenticação ou roubar o token.

A autorização é usada para as seguintes finalidades:

- Para verificar se o usuário autenticado tem os direitos para executar uma ação;
- Para determinar qual nível de acesso deve ser permitido aos recursos do sistema.

Há uma forte relação entre autenticação e autorização com base no princípio de que um usuário não autenticado não deve ter privilégios ou ter privilégios restritos no sistema.

A autenticação (*authentication*), a autorização (*authorization*) e a contabilização (*accounting*) do assunto dão origem à abreviação AAA, que é uma estrutura que ajuda a controlar e rastrear o acesso em uma rede de computadores. A contabilização é o terceiro "A" da estrutura AAA, que considera o registro, o rastreamento e as atividades de um usuário.

Como testar os mecanismos de autenticação e autorização

As técnicas de teste de segurança (testes de penetração) para mecanismos de autenticação podem incluir:

- Teste de credenciais padrão;
- Teste de política de senha fraca;
- Busca de informações vazadas usando inteligência de código aberto (OSINT - *open source intelligence*);
- Testes de força bruta usando tabelas de dicionário e arco-íris (ou seja, tabelas pré-computadas de hashes de senha invertidos) para realizar ataques que tentam descobrir senhas de usuários. As primeiras etapas podem ser, por exemplo, tentar "123456", "111111", data de nascimento ou o nome de um animal de estimação.

As técnicas de teste de segurança (testes de penetração) para mecanismos de autorização podem incluir:

- Explorar a falta de filtragem de entrada, como injetar solicitações SQL para serem autenticadas sem nenhum login/senha conhecido ou causar estouro do buffer de entrada para obter acesso administrativo a uma sessão de shell.
- Inserção de um *Uniform Resource Identifier* (URI) não autorizado (ou seja, "../../" em uma conta de *File Transfer Protocol*) ou um *Uniform Resource Locator* (URL) (ou seja, endereço do site/admin) para tentar obter acesso a dados confidenciais
- Testes de violações de escalonamento de privilégios horizontal e vertical

Observe que o [OWASP Test Guide] fornece uma lista de técnicas de teste de segurança para testar a autenticação e a autorização.

©International Software Testing Qualifications Board

2.2.4 Criptografia

Entendendo a criptografia

Um mecanismo de criptografia pode ser usado para evitar a divulgação de dados confidenciais, mesmo que eles possam ser acessados quando armazenados em algum lugar ou trocados entre um cliente e um servidor. A criptografia é um processo de codificação de dados (p. ex., texto simples) em dados cifrados (p. ex., texto cifrado), usando um algoritmo criptográfico e chaves secretas. O segredo é compartilhado e conhecido apenas pelos usuários autorizados. O objetivo é usar uma criptografia forte o suficiente para impedir que um invasor, que possa ter conseguido roubar dados criptografados, recupere o texto simples. O uso de algoritmos criptográficos ajuda a garantir a confidencialidade e a integridade de ativos confidenciais e a impedir que eles sejam manipulados.

Os tipos de protocolos criptográficos principais e normalmente usados são:

- Criptografia simétrica: baseada no uso de uma chave secreta compartilhada;
- Criptografia assimétrica: baseada no uso de chaves privadas, públicas e certificados, gerenciada por meio de uma infraestrutura de chaves públicas.

Como testar mecanismos comuns de criptografia

Alguns mecanismos criptográficos são conhecidos por serem fracos, especialmente devido ao curto comprimento das chaves secretas. Outros mecanismos podem ser vulneráveis porque não são implementados usando as práticas recomendadas ou incorporam defeitos de codificação (p. ex., estouro de buffer).

Os testes dos mecanismos de criptografia devem incluir:

- Testes de conformidade (p. ex., testes baseados em requisitos de segurança) de implementações de mecanismos de criptografia;
- Testes de vulnerabilidade "por projeto";
- Testes de vulnerabilidade "por construção";
- Testes de vulnerabilidade "por configuração".

Observe que o [OWASP Test Guide] fornece uma lista de testes de segurança para verificar implementações criptográficas fracas.

2.2.5 Teste de tecnologias de proteção

Os STEs precisam entender as nuances das diferentes linhas de defesa para que os testes apropriados possam ser projetados para verificar e validar sua eficácia.

Como testar a proteção do sistema

O teste da eficácia da proteção do sistema pode ser realizado de várias maneiras. A proteção do sistema restringe o acesso ao sistema às funções certas, abre somente os serviços necessários e monitora as atualizações dos aplicativos. Portanto, para testar a eficácia do endurecimento do sistema, os testes devem ser projetados para detectar se as medidas de endurecimento do sistema estão funcionando, aplicadas nos locais certos e da maneira correta. Também é importante testar as proteções de reforço do sistema que são muito restritivas e podem ser excessivas em comparação com os riscos de segurança.

Alguns testes de proteção do sistema podem se basear em uma revisão ou auditoria, enquanto outros podem se basear na capacidade de determinados grupos de usuários de executar determinadas ações ou acessar determinados dados.

Como testar firewalls

Devido ao número de protocolos, suas diferentes opções e a complexidade das redes a serem protegidas, é difícil configurar um firewall de forma eficiente e consistente. Os testes de eficácia do firewall devem incluir:

- Realização de uma auditoria para verificar a configuração do firewall;
- Varredura de portas para verificar se a política de segurança está bem implementada;

©International Software Testing Qualifications Board

- Uso de pacotes de rede malformados e testes de fuzz de rede para explorar comportamentos inesperados;
- Ataques de fragmentação para contornar os recursos de filtragem com o objetivo de realizar um ataque por trás do firewall;
- Visar o firewall de aplicativos da Web codificando e compactando dados ou ofuscando-os para ocultar as informações maliciosas que representam o ataque. Os critérios de avaliação do firewall de aplicativos da Web [WAFEC] podem ser usados para testar a eficácia de um firewall de aplicativos da Web.

Como executar a detecção de intrusão

A detecção baseada em cenários baseia-se em um cenário conhecido ou em uma "assinatura". Ela é fácil de ser contornada porque somente ataques conhecidos são detectados. Os testes podem incluir as seguintes técnicas de evasão:

- Codificação de caracteres ou modificação de dados (p. ex., adição de espaço em branco e indicadores de fim de linha);
- Fragmentação do protocolo de Internet (IP), segmentação do protocolo de controle de transmissão (TCP);
- Criptografia ou ofuscação;
- Codificação de URL.

A detecção baseada em comportamento é baseada em um modelo do comportamento do sistema e gera muitos resultados falsos-positivos e falso-negativos. Um resultado falso-negativo é qualquer alerta de segurança que deveria ter sido relatado, mas não foi. Os resultados falso-negativos podem ocorrer quando um novo ataque é desenvolvido e um sistema de detecção de intrusão (IDS - *intrusion detection system*) não está ciente dele, ou talvez uma regra possa ser escrita de forma a detectar alguns ataques, mas não detectar aqueles não especificados no modelo.

A precisão desse método de detecção deve ser mantida. É possível que um invasor se desvie do comportamento normal do IDS, o que resulta em uma nova especificação de comportamento intrusivo. Testes complementares devem usar tráfego malicioso para adicionar novas especificações intrusivas a serem consideradas como tráfego autorizado.

Como executar a varredura de malware

Os desenvolvedores de malware usam diferentes técnicas para proteger seu código contra engenharia reversa e detecção por software antimalware. Algumas dessas técnicas incluem:

- Exploração das funções da biblioteca do sistema usadas pelo antimalware;
- Ofuscação de strings para desativar a compreensão do comportamento do código malicioso;
- Permutação de código;
- Inserção de código não utilizado;
- Carregamento dinâmico de funções e bibliotecas (p. ex., para limitar a análise do código malicioso);
- Atualização automática de aplicativos.

Do ponto de vista do teste de adequação funcional, as ferramentas antimalware baseadas em assinaturas podem ser usadas para testar a eficácia do antimalware sem desenvolver partes reais de código malicioso. Outros tipos de arquivos maliciosos devem ser testados em relação ao tipo de aplicativos

O teste do antimalware baseado em comportamento é difícil porque não há um entendimento e uma definição claros do que é um comportamento malicioso. As ideias para testes podem se beneficiar das técnicas usadas pelos desenvolvedores de malware:

- Arquivos de execução não assinados que tentam usar chamadas de sistema para fazer alterações no sistema;
- Tentativa de iniciar processos incomuns com direitos concedidos;

©International Software Testing Qualifications Board

- Tentativa de copiar arquivos de execução em locais não autorizados;
- Tentativa de chamar APIs incomuns do sistema.

Uma consideração importante ao implementar um novo antimalware (com base na assinatura ou no comportamento) ou atualizar o antimalware existente é testar a implementação em uma plataforma representativa antes de implementá-la em toda a organização.

Teste de ofuscação de dados

É necessário um controle rigoroso da configuração entre os dados ofuscados e as chaves usadas para a ofuscação para garantir que sejam usadas as versões corretas das chaves. Caso contrário, os dados não poderão ser compreendidos para uso.

Como dados privados podem estar envolvidos em alguns testes, a ofuscação de dados pode ser usada para fins de teste para tornar anônimos os dados de produção usados em um ambiente de teste de sistema. Dados confidenciais, como informações de usuários usadas por um sistema de informações de saúde, não devem ser divulgados aos testadores. Os testes podem incluir ataques de força bruta ou de dicionário para tentar obter dados simples a partir de dados ofuscados.

Os testes para verificar a ofuscação do código podem incluir:

- Engenharia reversa de código
- Ataques de força bruta, pois alguns mecanismos de ofuscação são vulneráveis

©International Software Testing Qualifications Board

3 O processo de teste de segurança - 120 minutos (K3)

Palavras-chave

risco, ambiente de teste, plano de teste, processo de teste

Palavras-chave de segurança

Nenhum

Objetivos de aprendizagem para o Capítulo 3:

3.1 O processo de teste de segurança

STE-3.1.1 (K2) Explicar as diferentes atividades, tarefas e responsabilidades em um processo de teste de segurança

STE-3.1.2 (K2) Compreender os principais elementos e características de um ambiente de teste de segurança eficaz

3.2 Criação de testes de segurança para níveis de teste

STE-3.2.1 (K2) Exemplificar os testes de segurança no nível de teste de componentes com base em uma determinada base de código

STE-3.2.2 (K2) Exemplificar os testes de segurança no nível de integração de componentes com base em uma determinada especificação de projeto

STE-3.2.3 (K3) Implementar um teste de segurança de ponta a ponta que valide um ou mais requisitos de segurança relacionados a um ou mais processos de negócio.

©International Software Testing Qualifications Board

3.1 O processo de teste de segurança

O teste de segurança é um processo dentro de um SDLC. O processo de teste de segurança é dedicado ao escopo de segurança e deve estar alinhado com o processo de desenvolvimento para que as atividades de teste apropriadas sejam realizadas quando necessário.

Os riscos e as necessidades de segurança de cada organização são únicos devido à natureza da organização, aos ambientes técnicos, ao SDLC e aos riscos de negócio. Portanto, o processo de teste de segurança deve ser definido e implementado no contexto desses fatores.

3.1.1 Processo de teste de segurança do ISTQB

A Tabela 3.1 mostra como levar em conta e lidar com as atividades de teste de segurança dentro do Processo Fundamental de Teste do ISTQB.

Tabela 3.1 - Processo de teste de segurança do ISTQB

Atividade do processo de teste de segurança do ISTQB	Tarefas de teste de segurança	Responsabilidades
Planejamento de testes de segurança: O objetivo é definir um escopo adequado de testes de segurança que corresponda aos riscos de segurança.	• Levar em conta os requisitos relacionados à segurança • Definir os objetivos do teste de segurança • Definir o escopo dos testes de segurança • Identificar recursos de teste de segurança • Definir estimativas e cronogramas de testes para testes de segurança • Definir métricas de teste de segurança, critérios de entrada e critérios de saída	Um analista de testes de segurança (STA) é responsável por essa tarefa. O STE contribui para o planejamento estimando a carga de trabalho do teste e os recursos de hardware e software necessários

Análise de teste de segurança:
O objetivo é compreender todas as condições de teste de segurança e determinar o que testar.
	• Revisar a base de teste para testes de segurança, como avaliações de risco de segurança, requisitos relacionados à segurança, arquitetura baseada em segurança e políticas de segurança
	• Definir condições de teste de segurança com base em:
	  • Objetivos de segurança
	  • Riscos de segurança
	  • Padrões de segurança e vulnerabilidades conhecidas
	  • Defesas implementadas para proteger o sistema e seus dados
	  • Escopo dos testes de segurança
		Uma STA define o escopo do teste de segurança

Projeto de teste de segurança: O objetivo é identificar casos de teste de alto nível, ou seja, como testar	• Projetar casos de teste de segurança e conjuntos de testes • Priorizar casos de teste e conjuntos de teste • Identificar os dados de teste necessários para avaliação de vulnerabilidade ou teste de penetração • Projetar o ambiente de teste de segurança (ou seja, infraestrutura e ferramentas) • Definir a rastreabilidade entre a base de teste e os casos de teste	O STE projeta e prioriza casos de teste de segurança Uma STA analisa os produtos de trabalho do STE

©International Software Testing Qualifications Board

Atividade do processo de teste de segurança do ISTQB	Tarefas de teste de segurança	Responsabilidades
Implementação de testes de segurança:	• Organizar os casos de teste de segurança em procedimentos de teste ou scripts de teste. Configurar um ambiente de teste para realizar testes de segurança	Um STE implementa os casos de teste de segurança
Execução de testes de segurança:	• Realizar testes de segurança de adequação funcional • Realizar testes de penetração • Determinar vulnerabilidades específicas • Relatar à gerência, com informações detalhadas, os resultados dos testes de segurança provisórios	O STE executa os testes de segurança, produz resultados detalhados dos testes e comunica as vulnerabilidades identificadas o mais rápido possível
Monitoramento e controle de testes de segurança:	• Monitorar o progresso e os resultados dos testes de segurança • Tomar medidas corretivas conforme necessário em resposta às informações coletadas	A STA é responsável por essa tarefa.
Conclusão do teste de segurança:	• Garantir que todos os testes de segurança planejados tenham sido realizados • Analisar os resultados dos testes de segurança para avaliar os riscos residuais • Analisar os resultados dos testes de segurança para aprimorar o desenvolvimento de software em termos de segurança • Informar os resultados do teste de segurança à gerência e a outras stakeholders autorizadas • Determinar se os resultados dos testes de segurança (ou seja, relatórios de testes) foram entregues • Arquivar resultados de testes, dados de testes e outras informações confidenciais em locais seguros	A STA coleta todas as informações produzidas durante a execução do teste de segurança e produz um relatório de teste de alto nível para a gerência

Nos casos em que o teste exploratório foi realizado, o projeto do teste de segurança, a implementação do teste de segurança e a execução do teste de segurança baseiam-se nos resultados de testes anteriores usando técnicas padrão de exploração, como sniffers, scanners, ataques de força bruta e bots. O projeto do teste, a implementação do teste e a execução do teste são alcançados continuamente.

3.1.2 O ambiente de teste de segurança

Embora muitos tipos de testes possam usar um ambiente de teste localizado no(s) mesmo(s) servidor(es) e rede(s) com outros sistemas, os testes de segurança têm riscos específicos, mesmo que sejam virtualizados ou baseados em contêineres. Por exemplo, a realização de testes de segurança destrutivos, a contaminação do SUT e a corrupção ou divulgação de dados exigem uma abordagem segregada para a criação de um ambiente de teste para testes de segurança. Além disso, na maioria dos domínios comerciais, as normas exigem que sejam usados ambientes diferentes para desenvolvimento, teste e produção. Por exemplo, o Requisito 6.4.2 do PCI DSS (Payment Card Industry Data Security Standard, padrão de segurança de dados do setor de cartões de pagamento) afirma que é necessária a separação de tarefas entre os ambientes de desenvolvimento, teste e produção. Da mesma forma, o Requisito 6.4.3 do PCI DSS afirma que os dados de produção (ou seja, redes de área pessoal) não devem ser usados para teste ou desenvolvimento [consulte o capítulo 6, PCI DSS].

©International Software Testing Qualifications Board

O ambiente de teste de segurança deve conter todas as funções necessárias para a realização dos testes. Isso inclui ferramentas de gerenciamento de testes, ferramentas de teste de segurança e ferramentas de automação de testes. Elas são necessárias para permitir a descoberta do maior número possível de vulnerabilidades dentro do tempo alocado e com o menor número possível de resultados falso-positivos e falso-negativos.

Portanto, pode ser necessário identificar, especificar e configurar um ambiente de teste de segurança separado e eficaz para proteger outros ambientes, como desenvolvimento, teste de componentes, teste de integração de componentes, teste de sistema, teste de aceite e produção. Essa eficácia deve abranger a tolerância a falhas em relação a testes de segurança destrutivos ou fornecer proteção contra outros sistemas em teste e para a produtividade dos testes de segurança.

O STE deve analisar e estimar a arquitetura necessária, as APIs e o comportamento do SUT para avaliar o impacto do teste de segurança e definir o ambiente de teste mais eficaz.

As principais características de um ambiente de teste de segurança incluem:

1. Isolado no nível certo (se necessário):
   Dependendo dos riscos, o SUT é isolado por meio de comunicações filtradas ou o SUT e todos os outros sistemas dependentes são isolados de outros ambientes (p. ex., um site comercial precisa de um serviço de gerenciamento de pagamentos separado).
2. Representante do ambiente de destino:
   Para obter o comportamento correto do SUT, o ambiente total deve refletir o ambiente de produção em termos de versão e configuração exatas.
3. Produtivo:
   Contém todas as ferramentas necessárias para planejar, preparar, executar e relatar testes de segurança, seja de forma manual ou (quando possível) automatizada. A execução de testes de segurança precisa de ferramentas de teste específicas, conforme descrito no capítulo 9.
4. Recuperável:
   Para repetir testes conforme necessário e para se recuperar de corrupção, caso ela ocorra

3.2 Criação de testes de segurança para níveis de teste

A modelagem de ameaças é uma atividade repetida na qual cada nível de teste de segurança deve ser ajustado em relação aos resultados do teste dos últimos resultados da modelagem de ameaças.

Dependendo do tipo de projeto, é importante garantir que haja um teste de segurança planejado em cada fase aplicável do SDLC.

3.2.1 Projeto de teste de segurança no nível de teste do componente

Base de teste para o projeto de teste de segurança no nível de teste de componentes

Exemplos de produtos de trabalho que podem ser usados para projetar testes de segurança incluem:

- Análise de risco;
- Requisitos de funções e mecanismos de segurança;
- Projeto detalhado de funções e mecanismos de segurança (p. ex., APIs e algoritmos);
- Modelos de dados;
- Compilação ou regras de construção;
- Informações sobre o compilador.

Objetos de teste para o projeto de teste de segurança no nível de teste de componentes

Os objetos de teste típicos para o teste de componentes de segurança incluem:

- Componentes;
- Dependências (p. ex., bibliotecas de terceiros);

©International Software Testing Qualifications Board

- Código-fonte;
- Módulos de banco de dados.

Defeitos e falhas de segurança típicos no nível de teste do componente

Exemplos de defeitos e falhas de segurança típicos que podem ser encontrados no nível de teste do componente incluem:

- Código e lógica incorretos;
- Comportamento incorreto;
- Pontos fracos da filtragem de entrada;
- Problemas de fluxo de dados;
- Problemas de fluxo de chamadas;
- Código inacessível (inoperante);
- Código malévolo inserido deliberadamente (ou seja, bombas de software).

Tipos de testes de segurança no nível de teste de componentes

Os testes estáticos e dinâmicos podem ser aplicados no nível de teste do componente.

Dependendo dos objetivos do teste de segurança, da base do teste, dos objetos de teste e dos tipos de teste, diferentes abordagens e técnicas de design podem ser usadas no nível do teste de componentes com base no código fornecido:

- Verificação de que a implementação de funções e mecanismos de segurança se comporta conforme o esperado pelos requisitos de segurança

  O projeto de testes de segurança é baseado em requisitos detalhados (p. ex., especificações detalhadas e projeto detalhado do SUT). Devem ser usadas técnicas de teste bem conhecidas baseadas em especificações, como análise de valor de limite e particionamento de equivalência [ISTQB FL]. O STE deve rastrear os casos de teste de segurança para as especificações detalhadas.

- Criar confiança na qualidade do código de segurança (ou seja, codificação segura)

  Os casos de teste de segurança devem se concentrar na aplicação de regras de codificação seguras. O STE também deve verificar se a equipe de desenvolvimento não usa instruções de código perigosas e evita pontos fracos em linguagens de programação e compiladores. Normalmente, as equipes ou organizações de desenvolvimento definem suas próprias práticas recomendadas de codificação segura com base em referências conhecidas, que podem ser internas à organização ou externas, como a fundação OWASP. O STE pode projetar casos de teste com base nessas regras que podem ser consideradas como requisitos não funcionais (p. ex., capacidade de manutenção e outras características de qualidade não funcionais). Esses casos de teste de segurança podem ser processados automaticamente usando ferramentas de análise estática.

  Os testes de qualquer componente devem incluir a avaliação de possíveis violações da seguinte lista de verificação de práticas recomendadas: [CERT1]

  - Validar entradas;
  - Atenção aos avisos do compilador;
  - Arquitetar e projetar políticas de segurança;
  - Princípio "mantenha a simplicidade” (Keep it simple);
  - Negação padrão, que define uma lista de “permissões”;
  - Aderência ao princípio do menor privilégio;
  - Sanitizar dados enviados a outros sistemas;
  - Pratique a defesa em profundidade;

©International Software Testing Qualifications Board

- Usar técnicas eficazes de controle de qualidade;
- Adotar um padrão de codificação seguro.

Os testes realizados com base nessas listas de verificação de práticas recomendadas devem incluir avaliações de possíveis violações dessas práticas com base em uma análise de risco bem documentada que incorpore uma modelagem realista de ameaças.

- Identificação de vulnerabilidades nos componentes

Depois de verificar a implementação correta das funções e dos mecanismos de segurança e que as práticas recomendadas de codificação segura foram seguidas, o STE deve projetar testes de segurança com o objetivo de encontrar vulnerabilidades nos componentes desenvolvidos (p. ex., teste de fuzzificação da API de um componente).

O STE pode usar ferramentas de teste estático de segurança de aplicativos (SAST - *static application security testing*) ou de teste dinâmico de segurança de aplicativos (DAST - *dynamic application security testing*) para ajudar a encontrar vulnerabilidades.

- Mitigação dos riscos de segurança

Todos os testes de segurança descritos acima ajudam a reduzir os riscos de segurança do aplicativo ou sistema desenvolvido.

Análise do projeto de teste de segurança no nível de teste de componentes

Uma medida importante da adequação do projeto de teste de segurança envolve a avaliação da cobertura. Várias medidas de cobertura provêm dos testes realizados.

A cobertura pode ser medida como uma das seguintes opções:

- Porcentagem do número total de requisitos de segurança testados;
- Porcentagem de casos de segurança de uso/abuso especificados testados;
- Porcentagem de funções críticas de segurança, cenários ou tópicos de missão testados;
- Porcentagem de cobertura do código-fonte (p. ex., para identificar códigos mortos ou backdoors);
- Porcentagem de cobertura de partição de equivalência de dados (p. ex., para detectar capturas de exceções ruins);
- Número de constatações de segurança;
- Eficiência das ferramentas de segurança usadas (p. ex., número de resultados falso-positivos e falso-negativos).

3.2.2 Projeto de teste de segurança no nível de integração de componentes

De acordo com o ISTQB Foundation Syllabus [ISTQB FL], há dois níveis diferentes de integração: teste de integração de componentes e teste de integração de sistemas. Os componentes e/ou subsistemas a serem integrados podem ser provenientes de várias fontes diferentes, como outra equipe da mesma organização, um subcontratado, um produto comercial pronto para uso (COTS - *commercial off-the-shelf*), um serviço já disponível na nuvem ou um serviço de código aberto. Durante essas atividades de integração para criar o sistema de produção completo, as possibilidades de violações de segurança não são simplesmente a soma das vulnerabilidades em cada um dos componentes. Em vez disso, novos vetores de ataque tornam-se possíveis devido às interações entre os componentes do sistema maior e aos elementos organizacionais.

No entanto, algumas interações entre os componentes podem atenuar ou bloquear possíveis sequências que levem a violações de segurança.

O teste de integração de componentes pode demonstrar a complexidade do projeto de um sistema e a estabilidade de seu comportamento. A abordagem do teste de integração de componentes (p. ex., *top-down* ou *bottom-up*) pode afetar o momento de revelar preocupações de segurança ou a necessidade de testes adicionais específicos de segurança

©International Software Testing Qualifications Board

Assim como nos testes de componentes, os testes de integração de componentes devem ser projetados com base em uma análise de risco bem documentada que incorpore uma modelagem realista de ameaças. À medida que os componentes separados são integrados, observe que o scaffolding ou mocking na forma de stubs e drivers pode ser necessário para testar caminhos incompletos em um sistema durante a integração. À medida que mais componentes implementados são adicionados ao sistema, o scaffolding/mocking é removido gradualmente, permitindo uma avaliação mais completa da adequação funcional e abrindo novos caminhos para vulnerabilidades que podem ser exploradas.

De acordo com o nível de confiança nos componentes/subsistemas a serem integrados, o projeto de teste de segurança no nível de integração de componentes deve incluir:

- Testes de segurança da arquitetura de segurança global com base na documentação da arquitetura técnica;
- Testes de segurança de fluxos integrados configurados (p. ex., autorizados ou não, e o nível de confidencialidade, integridade e disponibilidade);
- Testes de segurança de APIs integradas (p. ex., para detectar problemas de segurança em APIs devido à falta de controles ou à falta de conhecimento dessas APIs);
- Testes de segurança relacionados à configuração de segurança de componentes integrados (p. ex., filtragem de acesso de um componente por outro, já que componentes não assinados devem ter acesso limitado);
- Verificação de que os componentes integrados que são softwares externos, de código aberto ou de código fechado estão livres de vulnerabilidades;

No nível de integração de componentes, a cobertura pode ser medida como uma das seguintes opções:

- Porcentagem de APIs usadas/testadas;
- Porcentagem de interações testadas entre componentes/subsistemas com base na documentação da arquitetura técnica;
- Número de descobertas de segurança no nível de integração de componentes;
- Número de descobertas de segurança que deveriam ter sido encontradas no nível do teste de componentes;
- Eficiência das ferramentas de segurança usadas (p. ex., número de resultados falso-positivos e falso-negativos).

3.2.3 Teste de segurança em testes de sistema e testes de aceite

Teste do sistema de segurança

Esse é o nível de teste durante o qual a implementação dos requisitos de segurança é testada para garantir que eles funcionem conforme o esperado. As atividades de teste de segurança do sistema incluem a realização de testes de segurança em alguma aproximação do ambiente de produção, o que exige uma transição para fora do ambiente de desenvolvimento no qual ocorreram as atividades anteriores de implementação e integração.

A função do teste de segurança no teste de sistema

O teste do sistema de segurança é a primeira oportunidade de exercitar a funcionalidade de ponta a ponta dos componentes totalmente integrados. Embora geralmente seja feito em um ambiente de teste, ele deve revelar propriedades emergentes do sistema que não teriam sido observadas antes da conclusão da integração. Normalmente, os requisitos de segurança são considerados em conjunto com os requisitos funcionais.

O objetivo do teste de segurança no teste do sistema é testar:

- todos os requisitos de segurança implementados nas funções de segurança em um ambiente de teste que representa o ambiente de produção;
- que a configuração operacional esteja protegida.

©International Software Testing Qualifications Board

Teste de aceite de segurança

Esse é o nível final de teste durante o qual os usuários, ou seus representantes, criam confiança de que o sistema é capaz de fornecer os recursos necessários no ambiente de produção de forma segura. Os objetivos do teste de aceite de segurança incluem testes de segurança em relação aos critérios de aceite relacionados à segurança estabelecidos para o sistema. Normalmente, os critérios de aceite relacionados à segurança concentram-se nos controles e processos de segurança funcional. As atividades do teste de aceite de segurança podem incluir:

- Instalação do sistema em um ambiente de pré-produção;
- Realização de testes de segurança com base em critérios de aceite;
- Determinação do aceite com base nos resultados dos testes de segurança.

A função do teste de segurança no teste de aceite

O teste de aceite é diferente do teste de sistema, pois é realizado em um ambiente semelhante ao de produção. Finalmente, ele coloca o sistema no ambiente em que agentes de ameaças externas estariam procurando encontrar pontos fracos no dia a dia. Esses testes permitem uma avaliação razoável da eficiência do desempenho e de outros comportamentos baseados em interações por meio de interfaces externas.

O teste de aceite deve, idealmente, validar se as metas iniciais do projeto foram atingidas e se os critérios de aceite de segurança documentados foram cumpridos. Isso é feito projetando e executando testes para validar processos/cenários de segurança, como controle de direitos, gerenciamento de autorizações e filtragem de firewall.

O melhor momento para definir e documentar os critérios de aceite é antes do desenvolvimento ou da compra do sistema. No contexto dos testes de segurança, os critérios de aceite podem ser de natureza global. Por exemplo, pode haver critérios de aceite que especifiquem o que é aceitável em termos de segurança geral do sistema. Isso incluiria critérios que são aplicados a todas as funções do sistema, como autenticação de usuários, direitos de usuários, níveis de criptografia e trilhas de auditoria.

Análise do design do teste de segurança no nível do teste de aceite

No nível do teste de aceite, a cobertura pode ser medida da seguinte forma:

- Porcentagem de processos/cenários de segurança testados;
- Número de descobertas de segurança que deveriam ter sido encontradas em níveis de teste anteriores com sua gravidade;
- Eficiência das ferramentas de segurança usadas (p. ex., número de resultados falso-positivos e falso-negativos);
- Porcentagem de requisitos de segurança testados;
- Porcentagem de itens de configuração operacional segura testados.

©International Software Testing Qualifications Board

4 Padrões e práticas recomendadas de teste de segurança - 195 minutos (K3)

Palavras-chave

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), vulnerabilidade, pontos fracos

Palavras-chave de segurança

Nenhum

Objetivos de aprendizagem para o Capítulo 4:

4.1 Introdução aos padrões e práticas recomendadas de segurança

STE-4.1.1 (K3) Explicar as diferentes fontes de padrões e melhores práticas e sua aplicabilidade.

4.2 Aplicar padrões importantes e práticas recomendadas de segurança

STE-4.2.1 (K3) Aplicar o conceito do Open Web Application Security Project, Common Vulnerability Enumeration, Common Weakness Enumeration, Common Vulnerability Scoring System e Common Weakness Scoring System e como aproveitá-los para testes de segurança.

4.3Aproveitamento dos padrões e das práticas recomendadas de teste de segurança

STE-4.3.1 (K2) Explicar as vantagens e desvantagens dos oráculos de teste usados para testes de segurança.

STE-4.3.2 (K3) Compreender as vantagens e desvantagens do uso dos melhores padrões e práticas de segurança.

©International Software Testing Qualifications Board

4.1 Introdução aos padrões e práticas recomendadas

Padrões e práticas recomendadas de vários tipos dão visibilidade ao consenso profissional e às obrigações regulatórias. Um padrão baseado em consenso representa a opinião ponderada de um corpo de especialistas bem-informados.

Mesmo que muitas vezes sejam usados como sinônimos, há grandes diferenças entre padrões e práticas recomendadas, que são explicadas nas subseções a seguir. As diferenças têm um impacto significativo no processo de seleção e nos possíveis casos de uso para utilizá-los.

4.1.1 Padrões e práticas recomendadas

Padrões

As normas são definidas como "um documento, estabelecido por um consenso de especialistas no assunto e aprovado por um órgão reconhecido, que fornece orientação sobre o projeto, o uso ou o desempenho de materiais, produtos, processos, serviços, sistemas ou pessoas". ([ISO_Web_21] e Apêndice D).

Há vários níveis de um "órgão reconhecido", o que permite a distinção entre diferentes tipos de padrões. Um dos níveis mais altos de reconhecimento de normas é representado mundialmente pela International Standard Organization (ISO). Normalmente, cada país que faz parte da Organização Mundial do Comércio (OMC) tem sua própria representação local. As normas têm o mais alto nível de reconhecimento para um STE devido ao seu alto nível de maturidade. No entanto, essa maturidade tem a desvantagem de levar muito tempo para ser concluída e, muitas vezes, resulta em normas com um caráter muito reativo.

Os órgãos reconhecidos podem criar seus próprios padrões. Eles podem ser categorizados como padrões do setor, padrões de fato e padrões específicos do fabricante:

- Padrões do setor:
  Foram estabelecidos ao longo de anos de aplicação em muitos contextos e demonstraram alguns valores agregados ao resolver um problema específico. A Internet Engineering Task Force (IETF) é um ator importante na criação de padrões nesse nível. Eles estabelecem padrões com base no julgamento técnico combinado de seus participantes e na experiência prática deles na implementação e implantação de suas especificações [IETF23].
- Padrões de fato:
  Geralmente têm suas raízes nos padrões do setor. Como sua cobertura e aceite são altas, eles até preenchem muitos dos critérios para serem considerados no nível mais alto de padrões. Um bom exemplo é o protocolo TCP, que foi estabelecido como um padrão do setor, mas hoje é considerado um padrão de fato [IETF23].
- Padrões específicos do fabricante:
  Alguns clientes/organizações aprenderam que há um valor agregado em seguir as especificações proprietárias de um fabricante específico.

Na realidade, essa classificação clara pode ter muitas sobreposições difusas, e nem sempre é simples fazer uma classificação clara de um determinado padrão.

Práticas recomendadas

As melhores práticas e seu órgão reconhecido são menos organizados formalmente. O Glossário do Gartner [Gart21] define as melhores práticas como um "grupo de tarefas que otimiza a eficiência (custo e risco) ou a eficácia (nível de serviço) da disciplina ou do processo de negócios para o qual contribui. Ele deve ser implementável, replicável, transferível e adaptável em todos os setores". Nesse nível, cada grupo de especialistas, mesmo que esteja trabalhando no mesmo contexto, pode criar seu próprio conjunto de práticas recomendadas.

©International Software Testing Qualifications Board

4.2 Aplique padrões e práticas recomendadas importantes para o teste de segurança

Existem vários padrões e práticas recomendadas para a disciplina de testes de segurança. Devido ao alto nível de requisitos a serem cumpridos para ser considerado um padrão, sua criação e manutenção são muito mais lentas do que as práticas recomendadas. Isso permite um reconhecimento profundo no setor e inclui muitos ciclos de feedback para aprimoramento. No entanto, isso impede o ajuste rápido a novas tendências e riscos. Em comparação, as melhores práticas têm uma alta eficiência de desempenho geral, mas é mais difícil que elas se tornem bem conhecidas, atinjam um alto nível de cobertura e sejam protegidas por evidências práticas.

4.2.1 Padrões do setor para teste de segurança

O padrão mais estabelecido em segurança de TI é a série ISO 27000. O padrão [ISO 27001] é aceito internacionalmente e intitulado "*Information technology — Security techniques — Information security management systems — Overview and vocabulary*". O foco desse padrão é o gerenciamento da segurança da informação, ou seja, identificar riscos, avaliá-los e gerenciá-los por meio de controles de segurança da informação. Todas essas atividades são combinadas em um sistema de gerenciamento de segurança da informação (ISMS - *information security management system*), que é o núcleo geral da norma ISO 27000. O escopo da norma é amplo e se concentra na maneira geral pela qual uma organização deve avaliar seus riscos, compará-los com suas necessidades específicas e lidar com os riscos mais relevantes. A norma principal pode ser aplicada a todas as organizações.

A série ISO 27000 consiste em mais de 40 normas individuais, que podem ser assim classificadas:

- *Norma principal*: Visão geral e introdução a um ISMS (a partir da ISO 27000 até a ISO 27005)
- *Padrões específicos de tópicos*: Para cobrir tópicos específicos, como gerenciamento de serviços ISO 27013 e provedor de nuvem pública ISO 27017 (consulte [ITGOV23a])
- *Padrões específicos de domínio*: Para focar domínios específicos, como os provedores de telecomunicações ISO 27011 e o setor financeiro ISO 27015 (consulte [ITGOV23a])

Os padrões mais usados que se aplicam ao contexto de testes de segurança e abrangem os objetos de teste relevantes e as condições de teste que o STE deve considerar são os seguintes:

- *ISO 27000*: Esta parte explica a estrutura geral da série ISO 27000 e apresenta um ISMS e a função que os testes de segurança podem desempenhar.
- *ISO 27001*: É a norma mais usada, pois lista um conjunto abrangente de recomendações e controles de segurança para estruturar e criar um ISMS individual. Seu foco é estabelecer uma visão abrangente dos ativos relevantes em uma organização, seus riscos expostos e possíveis atenuações. [ISO 27001
- *ISO 27001, Apêndice A*: A parte mais importante da ISO 27001 para um STE é apresentada nesse apêndice. Ele lista os controles de segurança para diferentes aspectos, como controle de acesso, recuperação de desastres e segurança de rede. Cada um desses controles, se aplicado em um contexto específico, é uma entrada importante para um STE, pois é sua tarefa medir a eficácia de um controle de segurança. [Cald11]
- *ISO 27002*: Essa norma utiliza os controles de segurança genéricos da ISO 27001 e fornece mais orientações sobre como aplicá-los na prática e como especificá-los com mais detalhes para um contexto específico. [Cald11]
- *ISO 27003*: Essa norma oferece suporte para que uma organização crie um plano para estabelecer um ISMS com base na ISO 27001.

Padrões de facto para o teste de segurança

Há muitos padrões *de facto* que podem ser aproveitados por um STE. Uma das séries mais importantes destes padrões vem da corporação MITRE, embora seu principal negócio não seja gerar padrões. O MITRE é uma organização privada, sem fins lucrativos, que fornece engenharia e orientação técnica para o governo federal dos EUA. Os patrocinadores mais importantes do MITRE são o Departamento de Defesa, a Administração Federal de Aviação e o Departamento de Segurança Interna [MITRE21].

©International Software Testing Qualifications Board

Para a área de testes de segurança, o MITRE hospeda e mantém os seguintes padrões conhecidos que fornecem valor agregado para o STE:

Common Attack Pattern Enumeration and Classification (CAPEC™)

O CAPEC™ fornece um catálogo publicamente disponível de padrões de ataque comuns. A ideia é entender melhor como os invasores exploram os pontos fracos dos aplicativos e outros recursos cibernéticos. Os padrões de ataque são baseados em padrões de design de software para atacantes. Dois padrões de ataque de entrada típicos são *SQL injection* (CAPEC-66) e a *relative path traversal* (passagem de caminho relativo) (CAPEC-139) [CAPEC21].

A CAPEC oferece diferentes visões sobre seus conjuntos de dados. As mais relevantes são:

- Domínio de ataque, como software, engenharia social e segurança física. No nível mais alto, o CAPEC lista nove domínios de ataque.
- Mecanismos de ataque, como a injeção de itens inesperados e a manipulação de recursos do sistema. No nível mais alto, o CAPEC lista seis mecanismos de ataque.

Cada objeto de teste que um STE testa deve estar localizado nesse catálogo. Muitas vezes, o CAPEC é o ponto de partida para obter uma visão geral inicial dos possíveis ataques que podem ser relevantes para um determinado sistema.

Os seguintes padrões do MITRE são usados para aprimoramento adicional de testes de segurança eficazes:

Common Weakness Enumeration (CWE)

O CWE é uma lista de pontos fracos de software/hardware. Normalmente, cada padrão de ataque comum tem um ou mais pontos fracos que podem ser usados para alavancar um padrão de ataque CAPAC [CWE21]. O CWE usa o conceito de visualizações, sendo que as mais usadas são:

- Desenvolvimento de software, como uma API, práticas ruins de codificação e problemas de permissão. No nível mais alto, a CWE lista 40 ativos de desenvolvimento de software.
- Projeto de hardware, como problemas de memória e armazenamento, problemas de núcleo e computação e periféricos, tecido no chip e problemas de E/S de interface. No nível mais alto, a CWE lista 12 ativos de design de hardware.

Cada ponto fraco comum é um ponto de partida eficaz para um STE testar se o padrão de ataque subjacente pode ser explorado.

Open Web Application Security Project (OWASP)

É importante perceber que o CWE e o OWASP [OWASP21] se sobrepõem e ambos listam pontos fracos comuns. A OWASP é bem conhecida por publicar sua classificação OWASP Top 10.

Common Weakness Scoring System (CWSS)

Quanto mais comum se torna um ponto fraco, mais importante se torna ter um esquema de priorização em vigor. O CWSS oferece um mecanismo para priorizar os pontos fracos de forma consistente, flexível e aberta [CWSS21]. A priorização é calculada por três conjuntos de métricas:

- Grupo de métricas da descoberta básica:
  Calcula o risco inerente de um ponto fraco, a confiança na precisão da descoberta e a força dos controles. Uma métrica típica é o impacto técnico", que varia de controle total sobre um sistema a nenhum impacto técnico.
- Grupo de métricas de superfície de ataque:
  Calcula as barreiras que um invasor deve superar para explorar o ponto fraco. Uma métrica típica é o privilégio necessário, que varia de nenhuma, até de administrador.
- Grupo de métricas ambientais:
  Calcula as características dos pontos fracos que são específicos de um determinado ambiente ou contexto operacional. Uma métrica típica é o impacto nos negócios, que varia de o negócio poder falhar a nenhum impacto.

©International Software Testing Qualifications Board

Ao usar pesos específicos e predefinidos, todas essas métricas podem ser agregadas em um valor geral do CWSS para um ponto fraco específico. O CWSS pode lidar com métricas desconhecidas por meio de valores padrão ou definindo/focando em um subconjunto de métricas individuais. Além disso, muitas métricas do Grupo de Métricas de Determinação de Base podem ser calculadas automaticamente por uma ferramenta de análise estática.

Common Vulnerability Scoring System (CVSS)

Um mecanismo de priorização semelhante ao CWSS é o CVSS [CVSS21], que segue uma abordagem semelhante, mas pressupõe uma vulnerabilidade existente e implantada (veja CVE a seguir). Tanto o CVSS quanto o CWSS são sistemas de pontuação para segurança de computadores: o CVSS é uma abordagem reativa porque as vulnerabilidades já existem antes da classificação. O CWSS é uma abordagem proativa, pois você está trabalhando com o software antes de colocá-lo em produção. Ambas as abordagens são frequentemente usadas em conjunto, mesmo que não sejam totalmente compatíveis (cf. [SecJour21]).

Common Vulnerabilities and Exposures (CVE)

O CVE é um banco de dados de informações divulgadas publicamente sobre problemas de segurança [CVE21]. Um número CVE identifica de forma exclusiva uma determinada vulnerabilidade da lista. O CVE ajuda porque fornece um identificador padronizado para uma determinada vulnerabilidade em um sistema específico. Se um sistema for afetado por um CVE específico, essa vulnerabilidade é uma instância específica de um ponto fraco comum (CWE) que pode ser usada para realizar um ataque específico (CAPEC). As novas entradas no repositório CVE geralmente se originam do trabalho diário dos STEs. Se eles identificarem uma nova vulnerabilidade desconhecida do CVE, poderão publicá-la no CVE para envolver a comunidade de segurança na identificação de medidas de combate.

Práticas recomendadas para o teste de segurança

As melhores práticas só precisam atingir um critério formal baixo para serem consideradas como melhores práticas. Muitas práticas recomendadas podem fracassar depois de algum tempo se não ajudarem. Algumas deixarão de ser usadas devido à falta de publicação/marketing, mas outras poderão melhorar sua maturidade no caminho para serem consideradas um padrão.

Uma prática recomendada madura típica que ainda é usada atualmente é o modelo STRIDE, inventado pela Microsoft [Micro09]. O STRIDE permite a modelagem sistemática de ameaças do ponto de vista do invasor. O termo em si é um acrônimo para seis categorias de ameaças, que classifica as possíveis ameaças: falsificação (*spoofing*), adulteração (*tampering*), repúdio (*repudiation*), divulgação de informações (*information disclosure*), negação de serviço (*denial of service*) e elevação de privilégio (*elevation of privilege*).

- Falsificação de identidade (*spoofing*), ou seja, alegar em um sistema ser uma pessoa ou um sistema que não é
- Alteração de dados (*tampering*), ou seja, a modificação maliciosa de dados
- Repúdio (*repudiation*), ou seja, ameaças que visam à auditoria e ao rastreamento, garantindo que comportamentos ruins não possam ser comprovados
- Divulgação de informações (*information disclosure*), ou seja, a exposição de informações a indivíduos que não deveriam ter acesso a elas
- Negação de serviço (*denial of service*), ou seja, negar serviço a usuários válidos
- Elevação de privilégio (*elevation of privilege*), ou seja, um usuário sem privilégios obtém acesso privilegiado.

Em geral, o STRIDE é usado para ajudar os desenvolvedores a considerar as ameaças durante o projeto e a fechar as lacunas identificadas. O STE pode usar a mesma abordagem para se concentrar nos testes.

4.3 Aproveitamento dos padrões e das práticas recomendadas de teste de segurança

Há muitos casos de uso possíveis para aproveitar os padrões e as práticas recomendadas. Em geral, esses casos de uso podem ser divididos em aplicativos obrigatórios e aplicativos voluntários.

©International Software Testing Qualifications Board

4.3.1 Aplicativo obrigatório de padrões e práticas recomendadas

Nesse tipo de caso de uso, os padrões e as práticas recomendadas são obrigatórios para outra parte:

- Requisitos de segurança para contratos:
  As práticas recomendadas são uma maneira eficaz de especificar os requisitos de segurança para o desenvolvimento de software, especialmente aqueles que são delegados a terceiros. Em vez de listar todos os requisitos específicos, apenas o cumprimento de um padrão específico é exigido, o que implica o cumprimento de todos os conselhos e requisitos de segurança contidos.
- Requisitos de segurança como um regulamento:
  Até mesmo as instituições reguladoras (p. ex., no domínio bancário) costumam usar padrões e práticas recomendadas, que são fáceis de gerenciar.

4.3.2 Aplicativo voluntário de padrões e práticas recomendadas

Nesse tipo de caso de uso, a aplicação de padrões específicos e práticas recomendadas é uma decisão da gerência para gerar o seguinte valor agregado:

- Estabelecimento de um alto nível de segurança por meio da reutilização do conhecimento de segurança estabelecido e armazenado nos padrões e práticas recomendadas existentes
- Evidências bem conhecidas para demonstrar conscientização sobre segurança
- Objetivo geral de marketing e criação de pontos de venda exclusivos em uma área de negócios competitiva.

4.3.3 Oráculos de teste extraídos de padrões e práticas recomendadas

Um caso de uso geral para aproveitar os padrões e as práticas recomendadas, independentemente de ser obrigatório ou voluntário, é a noção de usar oráculos de teste poderosos. No nível do aplicativo, o oráculo de teste geralmente é a seção de requisitos de segurança na especificação. Em níveis mais baixos, por exemplo, bibliotecas incluídas, sistema operacional subjacente, tráfego de rede, padrões e práticas recomendadas podem ser facilmente usados. Especialmente o tipo mais volátil de práticas recomendadas pode listar muitas vulnerabilidades conhecidas de um determinado sistema e determinar os resultados esperados a serem usados como evidência de segurança ou não segurança. Essa é uma ferramenta poderosa para os STE, pois eles só precisam definir os casos de teste de baixo nível correspondentes, executá-los e, em seguida, comparar os resultados do teste com os listados na prática recomendada.

4.3.4 Vantagens e desvantagens de aproveitar os padrões e as práticas recomendadas de teste de segurança

Aproveitar os padrões e as práticas recomendadas para testes de segurança tem muitas vantagens, mas há alguns aspectos negativos que devem ser considerados cuidadosamente em um contexto específico. Em geral, as seguintes vantagens se aplicam ao aproveitamento de padrões e práticas recomendadas:

- Terminologia consistente:
  Em TI, há muitos termos de marketing, sinônimos e frases sem uma definição ou distinção clara entre eles. Os padrões e, às vezes, até mesmo as práticas recomendadas podem apoiar o esclarecimento da terminologia.
- Reutilização do conhecimento especializado:
  A definição de padrões e práticas recomendadas pode ser uma tarefa demorada, que geralmente é realizada por especialistas em segurança. Seu conhecimento pode ser capturado e reutilizado em padrões e práticas recomendadas.
- Referência e verificação dupla da integridade:
  Se uma empresa usar sua própria estrutura de teste de segurança específica, os padrões e as práticas recomendadas existentes poderão ser usados como referência para verificar a integridade de suas soluções.

©International Software Testing Qualifications Board

- Melhoria do compromisso entre fornecedor e cliente:
  Quanto mais estabelecido e reconhecido for um padrão ou uma prática recomendada, mais eficientemente ele poderá ser usado como base para o compromisso entre o consumidor (o que ele quer ter) e o fornecedor (o que ele deve fazer).
- Fácil comunicação sobre o nível de segurança alcançado:
  Se uma organização usar seu próprio conjunto de testes de segurança sem nenhuma referência externa, poderá ser difícil demonstrar sua eficácia. O uso de padrões e práticas recomendadas ajuda a obter uma atitude positiva geral e simplifica muito a comunicação.

Entretanto, alguns aspectos negativos são possíveis quando se usam padrões e práticas recomendadas para testes de segurança:

- Seleção incorreta:
  Há muitos padrões e práticas recomendadas disponíveis, cada um com seu próprio foco e condições prévias necessárias para ser aplicável. Aproveitar a fonte errada reduz o impacto da obtenção de uma segurança melhor e pode até diminuir os recursos disponíveis para serem gastos em segurança.
- Práticas recomendadas em um contexto específico errado:
  Enquanto a maioria dos padrões alcançou alta qualidade devido a seus longos processos de criação e longos ciclos de feedback, as práticas recomendadas podem surgir e desaparecer em um curto prazo. Especialmente quando propostas inicialmente, sua correção e valor agregado nem sempre são claros e podem não ter um forte vínculo com o contexto específico. O uso de uma nova prática recomendada proprietária que ainda não demonstrou nenhuma evidência de criação de valor agregado pode até mesmo diminuir o nível de segurança.
- Falta de personalização:
  Frequentemente, os padrões e as práticas recomendadas definem determinados parâmetros que precisam ser atendidos para serem aplicáveis em um contexto específico. Se isso for omitido ou não for feito corretamente, sua aplicação poderá gerar um valor agregado limitado.
- Considerações sobre commodities:
  Quanto mais estabelecido e popular um padrão ou prática recomendada se torna, menos ele pode ser usado para criar exclusividade em comparação com outros produtos concorrentes (se necessário).
- Cegueira operacional:
  As falhas ou o surgimento de novos tópicos podem levar a uma atenção menor se os padrões ou as práticas recomendadas não forem adotados em tempo hábil.

©International Software Testing Qualifications Board

5 Ajustando o teste de segurança ao contexto organizacional - 195 minutos (K4)

Palavras-chave

pontos fracos

Palavras-chave de segurança

rootkit

Objetivos de aprendizagem para o Capítulo 5:

5.1 O impacto das estruturas organizacionais no contexto do teste de segurança

STE-5.1.1 (K3) Analisar um determinado contexto organizacional e determinar quais aspectos específicos devem ser considerados nos testes de segurança

5.2 O impacto das regulamentações nas políticas de segurança e como testá-las

STE-5.2.1 (K3) Analisar o impacto das regulamentações sobre as políticas de segurança e como testá-las

5.3 Analisar um cenário de ataque

STE-5.3.1 (K4) Analisar um cenário de ataque e identificar possíveis fontes e motivações do ataque.

©International Software Testing Qualifications Board

5.1 O impacto das estruturas organizacionais no contexto do teste de segurança

A segurança das informações não pode ser obtida apenas com a proteção da infraestrutura e com medidas implementadas tecnologicamente. As pessoas e os processos de uma organização também devem ser considerados.

5.1.1 Analisar um determinado contexto organizacional e determinar quais aspectos específicos devem ser considerados nos testes de segurança

As pessoas frequentemente se tornam vítimas de ataques de engenharia social e processos importantes, como uma resposta de emergência definida, podem estar ausentes ou implementados de forma inadequada. Portanto, um STE precisa cobrir esses aspectos durante os testes de segurança, pois ambos afetam a segurança das informações de uma organização. As pessoas têm uma função definida. Dependendo de sua função, elas estão envolvidas em diferentes processos. As funções e os processos geralmente dependem muito da estrutura organizacional.

A estrutura organizacional pode ser amplamente classificada nos três tipos a seguir:

- Estrutura funcional: organizada por funções comuns, como produção, marketing, recursos humanos, TI e contabilidade
- Estrutura divisional: Organizada como um conjunto de funções que produzem um produto
- Estrutura matricial: os funcionários são agrupados por função e produto simultaneamente

A maneira pela qual essas estruturas organizacionais afetam o fluxo de informações e a implementação de decisões administrativas é considerada a seguir:

- Em uma organização estruturada funcionalmente, as informações precisam ser trocadas entre os diferentes departamentos da organização e as decisões administrativas são implementadas diretamente em uma abordagem de cima para baixo da gerência para toda a organização
- As organizações com estrutura divisional acrescentam uma camada administrativa no topo de cada divisão e, portanto, as decisões podem afetar apenas uma única divisão. Além disso, o fluxo de informações entre as divisões é ligeiramente reduzido, embora as divisões geralmente contenham departamentos semelhantes, como o de desenvolvimento
- As estruturas matriciais tentam unir os dois aspectos. Elas são separadas funcionalmente, mas também adotam um foco baseado no produto sem acrescentar a camada administrativa separada encontrada nas estruturas divisionais. No entanto, há um risco maior de conflitos, pois as decisões podem ser tomadas tanto da perspectiva do produto quanto de uma perspectiva puramente administrativa.

Levando isso para o contexto dos testes de segurança, um STE deve estar ciente da estrutura organizacional pelos seguintes motivos:

1. Os resultados de um teste de segurança serão influenciados pelo departamento que solicitou e planejou o teste.
2. Dependendo da estrutura geral da organização, um STE pode tirar proveito de pontos fracos no fluxo de informações

O primeiro desses dois motivos resulta do fato de que os departamentos de TI e de segurança geralmente estão autorizados a implementar e garantir a segurança, mas outros departamentos também podem estar cientes disso. Por exemplo, saber que um funcionário da equipe de testes de penetração está visitando outro departamento aumentará a chance de que as pessoas desse departamento realmente cumpram as políticas de segurança, pelo menos durante o período de sua estadia.

O segundo dos dois motivos acima resulta da probabilidade de que, para cada estrutura organizacional, existam pontos fracos específicos. Em uma estrutura funcional, um STE poderia tirar proveito disso da seguinte forma:

©International Software Testing Qualifications Board

- Os funcionários de diferentes departamentos podem não conhecer pessoas de outros departamentos, principalmente os responsáveis pela administração de TI.
- A conscientização sobre a segurança e o aceite de determinadas medidas de segurança pode ser significativamente menor nos departamentos em que trabalham apenas funcionários não técnicos.
- As informações sobre um incidente podem permanecer em um único departamento por um período, resultando, na pior das hipóteses, em atrasos no tempo de reação.

Em organizações estruturadas por divisões, é possível considerar os pontos fracos potenciais semelhantes, embora elas possam ser transferidas para as divisões:

- Os funcionários de diferentes departamentos podem não conhecer pessoas de outros departamentos. Dependendo de como a infraestrutura de TI da organização é mantida, eles podem não saber quem é o responsável pela administração de TI.
- As informações sobre um incidente podem permanecer em um único departamento por um período, resultando em atrasos no tempo de reação.
- Dependendo do tamanho de uma única divisão, as equipes também podem ser subdivididas por funcionalidade em departamentos menores dentro da divisão, o que faz com que o ponto acima, relativo à conscientização sobre segurança e o aceite de determinadas medidas de segurança, também seja válido para estruturas divisionais.

Embora as organizações com uma estrutura divisional distribuam funcionalidades semelhantes entre diferentes divisões, alguns serviços, como a administração de TI, geralmente residem em um departamento central.

Em organizações matriciais, uma pessoa pode tirar proveito de possíveis conflitos entre a gestão administrativa e a gestão de produtos. No entanto, esse nem sempre é o caso e, como já mencionado para organizações estruturadas em divisões, alguns serviços podem ser centralizados.

Algumas considerações

As informações acima apresentam uma perspectiva de alto nível das estruturas organizacionais e seus possíveis pontos fracos. Na prática, muitas organizações não têm estruturas puramente funcionais, divisionais ou matriciais. A função de gerenciamento de segurança, em particular, é muitas vezes gerenciada por um departamento central que dita as medidas de segurança, como uma política de segurança para toda a organização.

Uma política de segurança pode ser definida como "Um documento de alto nível que descreve os princípios, a abordagem e os principais objetivos da organização em matéria de segurança." [Glossário ISTQB]. Um serviço de segurança, de acordo com o NIST, é definido como "um recurso que oferece suporte a um ou vários dos objetivos de segurança. Exemplos de serviços de segurança são o gerenciamento de chaves, o controle de acesso e a autenticação." [NIST02].

O estudo das políticas de segurança da organização pode revelar possíveis vetores de ataque ao tornar conhecidas as restrições impostas ao comportamento de seus membros, bem como as restrições impostas aos adversários pelos mecanismos de segurança. As políticas de segurança da empresa podem não estar disponíveis publicamente. Algumas políticas organizacionais podem ser acessíveis apenas aos funcionários ou até mesmo a determinados membros da equipe.

Um aspecto importante a ser considerado no contexto organizacional é a maneira pela qual a organização terceiriza partes de sua produção ou serviços. O(s) parceiro(s) relevante(s) também deve(m) ser considerado(s) como alvo potencial para um teste de segurança. Isso depende da natureza e do conteúdo do contrato entre as duas organizações que definem as obrigações legais. Os parceiros externos geralmente recebem acesso (limitado) à rede privada virtual (VPN), trabalham no mesmo repositório de código ou têm um token de acesso para um escritório local. Embora eles geralmente tenham contas restritas, isso pode ser o primeiro passo para um

©International Software Testing Qualifications Board

ataque bem-sucedido. Outro foco relacionado aos parceiros no contexto dos testes de segurança é a análise da cadeia de suprimentos, pois os ataques nessa área podem ter consequências graves (p. ex., [WIRED21]).

Os aspectos gerais abordados nesta seção podem ser válidos para quase todas as organizações, mas negligenciam questões industriais específicas, como o tipo de produto ou serviço que uma organização oferece, bem como o setor industrial em que a organização opera. A oferta de um serviço da Web para música pode ter requisitos de segurança diferentes dos de um dispositivo médico usado em um hospital. Justamente por esse motivo, existem regulamentações para determinados setores que prescrevem requisitos para processos, segurança, medidas de segurança ou outros aspectos específicos do domínio. (consulte a seção 5.2). Isso pode afetar muito mais as organizações que desenvolvem seus próprios produtos do que as organizações que, por exemplo, vendem produtos COTS.

5.2 O impacto das regulamentações nas políticas de segurança e como testá-las

As normas de segurança orientam o conteúdo das políticas de segurança, que orientam a estrutura de controle de segurança da informação para testes de segurança. O STA desenvolve essa estrutura de controle. Com o conhecimento da estrutura, um STE desenvolve e usa casos de teste para testar os controles.

5.2.1 O impacto das regulamentações governamentais sobre as regulamentações de segurança

Devido à forte interconectividade da maioria dos setores industriais, os ataques à segurança cibernética podem ter um impacto profundo em uma única organização e, na pior das hipóteses, na infraestrutura geral de um país inteiro. Como reação a isso, os governos definiram regulamentações para forçar as organizações essenciais a adaptarem seu nível de segurança a um mínimo. As organizações que não estiverem em conformidade podem ser multadas ou temporariamente fechadas até que as medidas de segurança exigidas sejam implementadas. As organizações afetadas pelas regulamentações têm, portanto, um incentivo para aprimorar suas medidas e políticas de segurança para, no mínimo, o nível de segurança exigido. O NIST define leis e regulamentos no contexto da segurança de computadores como "leis, regulamentos, políticas, diretrizes, padrões e procedimentos específicos do governo federal e da organização que exigem requisitos para o gerenciamento e a proteção dos recursos de tecnologia da informação". Embora isso possa ser verdade em alguns casos, as regulamentações gerais podem ser definidas em nível global, sindical ou nacional, como, por exemplo:

- Regulamentações globais definidas, por exemplo, pela OMC
- Regulamentos específicos da União, como o GDPR e os Regulamentos de Sistemas de Informação e Rede (NIS) definidos pela União Europeia (UE)
- Normas nacionais, como a Lei de Compartilhamento de Segurança Cibernética dos Estados Unidos da América (EUA)

Além disso, outras regulamentações podem ser aplicadas a setores industriais específicos, como:

- Lei de portabilidade e responsabilidade do seguro de saúde [HIPAA]
- UNECE WP.29 para o setor automotivo [UNECE20]
- Regulamento de Implementação (DVO) (UE) 2019/1583 para a segurança da aviação [BSI21]
- PCI DSS [PCI22]

A formulação de regulamentos é feita por instituições específicas, como a Agência de Segurança Cibernética e de Infraestrutura nos EUA ou o Escritório Federal de Segurança da Informação na Alemanha. Na UE, a Agência Europeia de Segurança da Rede e da Informação definiu a diretiva NIS, que foi estabelecida como política em 2016. O objetivo é aumentar e padronizar o nível de segurança cibernética em todos os estados-membros. As mesmas instituições geralmente publicam recomendações (p. ex., [TR02021]), práticas recomendadas ou, pelo menos, referências a outras publicações.

©International Software Testing Qualifications Board

Isso é importante porque os regulamentos podem ser muito pouco específicos sobre qual tecnologia real deve ser usada na prática. Como a tecnologia muda rapidamente com o tempo, isso exigiria uma adaptação contínua das leis definidas. No entanto, há um entendimento comum sobre a tecnologia de ponta que é publicada, por exemplo, por instituições como a TeleTrust [TELETRUST] ou o NIST e é adaptada ao longo do tempo.

As regulamentações geralmente não são específicas porque visam abranger um amplo escopo de setores industriais. A manutenção de um conjunto fixo de tecnologias de segurança de TI pode causar problemas, pois algumas tecnologias podem se tornar inaplicáveis a determinadas organizações.

Embora o uso da tecnologia atual seja uma parte dos regulamentos, os três pilares a seguir também precisam ser considerados:

- Recursos (p. ex., hardware, software e tecnologia de ponta)
- Pessoal
- Processos de segurança da informação

Com relação ao pessoal, há quatro aspectos principais a serem considerados:

- Os funcionários devem estar cientes da importância da segurança das informações e devem entender que são responsáveis por garantir a segurança
- Eles devem ter o conhecimento necessário para implementar e aplicar as medidas de segurança definidas, observando que o tipo e a especificidade podem variar de acordo com as diferentes funções. São necessárias habilidades e conhecimentos sobre as políticas e os procedimentos de segurança efetivamente definidos.
- Eles devem aceitar e aplicar os processos de segurança da informação definidos (consulte a próxima seção)
- Alguns funcionários precisam de autorizações especiais

Os processos de segurança da informação incluem aspectos como:

- Definição de responsabilidades. No contexto das regulamentações, isso afeta as funções e responsabilidades dentro de uma organização e as instituições responsáveis pelo monitoramento da conformidade e pela comunicação de incidentes, como o National Cyber Emergency Response. Geralmente, uma pessoa é nomeada como o único ponto de contato dentro de uma organização. Ela precisa definir quando e para quem deve fazer os relatórios
- Como lidar com funcionários novos ou que estão saindo, pessoal que muda de departamento ou funcionários externos
- No caso de um alerta de segurança, quem precisa ser informado, quem será responsável pela tomada de decisões e quando a organização deve relatar o incidente a uma instituição (federal)?
- Como lidar com parceiros de negócios e fornecedores
- Revisões e reavaliações regulares dos processos e medidas atualmente definidos
- Procedimentos de auditoria, incluindo preparações e correções após a realização de uma auditoria

Os aspectos descritos acima são parte integrante de um ISMS em funcionamento. Algumas normas visam estabelecer e preservar um ISMS essencial (consulte o capítulo 8).

Como testar políticas de segurança

Os STEs têm um alto nível de responsabilidade no teste de políticas de segurança para organizações afetadas por regulamentações. Os principais motivos para isso são:

1. Do ponto de vista de uma organização, é importante estar em conformidade. Caso contrário, a organização poderá ter que pagar multas ou correr o risco de ter seus negócios temporariamente fechados.
2. Como os incidentes em setores regulamentados da indústria podem ter consequências muito mais graves do que em outros setores, os testes devem ser feitos de forma muito completa para garantir um alto nível de segurança.

©International Software Testing Qualifications Board

A próxima etapa é avaliar as práticas recomendadas e as tecnologias de ponta, já que os regulamentos podem se referir apenas a elas. Os STEs precisam validar se as medidas de segurança fornecidas para um SUT ainda são suficientes. Por exemplo, eles precisam verificar se os dados estão criptografados e qual algoritmo é usado, já que isso pode não estar seguro.

Para fins de teste, os resultados de auditorias realizadas anteriormente podem ser considerados como parte da base de teste. No entanto, as conclusões de uma auditoria anterior que foram implementadas agora precisam ser testadas para confirmação. O STE não pode presumir que a implementação foi realizada corretamente. Além disso, como os regulamentos e os principais objetivos de uma política de segurança incluem pessoas e processos, as atividades de teste também devem incluir esses aspectos.

O teste de aspectos pessoais pode ser realizado aplicando testes como fingir um ataque de engenharia social ou tentar contornar o controle de acesso (físico). Esses tipos de testes dependem do departamento organizacional que solicitou o teste de segurança.

O teste de segurança dos processos inclui backup e restauração, além de resposta e relatório de emergência. Esses testes podem ser muito elaborados e caros, pois muitas pessoas podem estar envolvidas durante o teste

5.3 Análise de um cenário de ataque

5.3.1 Cenário de ataque comum s

Os incidentes de segurança variam muito em termos de técnicas e ferramentas de ataque aplicadas, do tipo de invasor e da motivação para realizar um ataque. Como resultado, é difícil fornecer uma descrição genérica de um ataque. Entretanto, algumas etapas são comuns a quase todos os ataques. Essas etapas podem ser definidas como:

1. Coleta de informações
2. Exploração/obtenção de acesso
3. Persistência/manutenção do acesso
4. Limpeza de trilhas

Para fins de comparação, o tratamento de incidentes de segurança é definido de acordo com o NIST [NIST03] pelas seguintes etapas:

1. Preparação
2. Detecção e análise
3. Contenção, erradicação e recuperação
4. Atividade pós-incidente

Embora ambas as enumerações descrevam uma sequência comum de ataque e resposta, fatores como motivação, recursos, habilidades de um invasor e a abordagem usada têm um forte impacto sobre o sucesso de um ataque e as consequências para a parte atacada.

Classificação dos atacantes e suas motivações

A palavra "hacker" é usada nesta seção como sinônimo de invasor. No entanto, o termo hacker geralmente se refere a uma pessoa altamente capacitada tecnicamente.

Os atacantes podem ser divididos em diferentes tipos, dependendo de suas habilidades técnicas e dos recursos disponíveis:

Tipo de atacante	Descrição
script kiddies	São pessoas com baixo nível de conhecimento técnico, que usam as ferramentas e os scripts existentes sem compreendê-los totalmente e que têm recursos muito limitados.
golpistas	Essas pessoas usam técnicas simples, como phishing, mas geralmente visam muitas pessoas, o que aumenta suas chances
hackers privados	Essas pessoas têm uma sólida formação técnica e estão interessadas em segurança de TI ou são muito curiosas

©International Software Testing Qualifications Board

Tipo de atacante	Descrição
hackers profissionais	Essas pessoas têm uma formação técnica muito elevada e são capazes de realizar ataques altamente sofisticados para ganhar a vida
governos	Essas organizações pagam equipes completas para espionagem, hacking ou sabotagem e têm muito mais recursos disponíveis do que pessoas solteiras ou pequenos grupos.

Embora essa seja uma categorização grosseira que depende da habilidade e dos recursos, outro aspecto importante é a motivação de um invasor. As crianças que usam scripts podem querer apenas brincar com algo que acabaram de encontrar on-line ou impressionar os amigos, enquanto os hackers profissionais ganham dinheiro hackeando e, portanto, também querem ter uma boa reputação. Veja a seguir as categorias de motivação:

- Motivações pessoais (p. ex., fama, vingança, ciúme e curiosidade)
- Motivações políticas (p. ex., "hacktivismo", guerra e espionagem)
- Motivações profissionais (p. ex., dinheiro, reputação e espionagem industrial)

Dependendo do nível de motivação, um ataque pode ser realizado contra uma única entidade ou várias entidades. Por exemplo, o ransomware é um malware que geralmente criptografa dados e bloqueia o uso do sistema infectado até que a vítima pague uma determinada quantia. Como a infecção de mais sistemas aumenta a chance de um invasor ganhar dinheiro, o ransomware geralmente é escrito para infectar o maior número possível de sistemas.

Em contraste com isso, os worms de computador, como o Stuxnet [WIKI02], são desenvolvidos principalmente para o caso especial de infecção de sistemas de controle de supervisão e aquisição de dados. Eles foram usados para sabotar os programas nucleares de países inteiros.

Abordagem comum de um atacante

Coleta de informações

A primeira fase de um ataque é a coleta de informações, também conhecida como reconhecimento. Um invasor busca informações sobre o alvo e tenta encontrar pontos fracos nas seguintes áreas:

- Infraestrutura de TI (p. ex., uma vulnerabilidade de software conhecida)
- Infraestrutura física e os mecanismos de controle de acesso relacionados (p. ex., invadir um escritório, que pode ter um sistema de alarme deficiente e permitir o acesso a informações confidenciais)
- Os funcionários, que podem ter pouca consciência da
- Processos dentro de uma organização que podem ser explorados

A coleta de informações pode ser passiva ou ativa. A coleta passiva de informações pode ser feita por meio de pesquisas na Web usando consultas de pesquisa especializadas, como o Google, que podem revelar uma quantidade surpreendentemente grande de informações. Essa prática ficou conhecida como "Google hacking" ou "Google dorking" [WIKI01]. Além disso, as plataformas de mídia social são uma importante fonte de informações sobre os funcionários, especialmente no que se refere a números de telefone, endereços de e-mail e outras informações pessoais que podem ser usadas para engenharia social. O uso dos detalhes de uma pessoa permite que os invasores personalizem seus ataques, como no spear phishing ou no envio de e-mails personalizados com anexos maliciosos. Eles podem usar, por exemplo, o nome correto, o endereço ou a data de aniversário de uma pessoa para criar e-mails com conteúdo que pareça muito familiar ou íntimo para a vítima, aumentando assim a probabilidade de ela abrir o anexo malicioso ou clicar em um link e visitar um site malicioso.

A coleta ativa de informações inclui o uso de ferramentas e técnicas que interagem com o alvo, mas aumenta o risco de ser reconhecido como um invasor. A coleta de informações pode ser realizada de diferentes maneiras:

- Tentativa de contato pessoal por e-mail ou telefone (vishing).
- Busca de informações úteis no lixo da vítima, como endereços e números de telefone [SENG22]. Isso é conhecido como "dumpster diving".
- Varreduras de portas.

©International Software Testing Qualifications Board

- Impressão digital do sistema operacional.
- (DNS) enumeração (veja abaixo).
- Varredura de vulnerabilidades.
- Coleta de informações úteis que estão disponíveis publicamente (p. ex., OSINT).

Técnicas como a enumeração de DNS podem não ser detectadas, enquanto outras técnicas, como a varredura de portas ou a varredura de vulnerabilidades, podem ser facilmente identificadas pela análise dos arquivos de registro de servidores ou firewalls. Os sistemas de detecção de intrusão de rede (NIDS), que geralmente são implementados como uma medida de segurança na infraestrutura de rede, analisam o tráfego de entrada em busca de padrões suspeitos e emitem um alerta de segurança se reconhecerem um possível tráfego mal-intencionado. Em particular, os scanners de vulnerabilidade têm uma impressão digital de rede facilmente detectável. Embora o uso de scanners possa revelar informações úteis para o invasor muito rapidamente, seu uso também aumenta o risco de ser detectado.

A duração da fase de coleta de informações pode variar consideravelmente e dependerá da motivação do invasor. Um script kiddie pode ficar entediado depois de alguns minutos ou horas porque só quer brincar com uma ferramenta que encontrou na Internet, enquanto os atacantes com motivação política podem coletar informações durante meses antes de realmente lançar um ataque. Os métodos ativos de coleta de informações sempre deixam rastros, mas às vezes só são detectados por meio de investigações forenses depois que um ataque é realizado com sucesso.

Além da motivação, o tipo de ataque também influencia a fase de coleta de informações. Um invasor que deseja realizar um ataque de negação de serviço que afeta apenas a disponibilidade talvez não precise invadir uma rede privada e, portanto, pode ignorar determinadas informações. No entanto, mais informações sempre aumentam as chances de um ataque bem-sucedido.

A identificação de um sistema vulnerável pode ser feita enumerando todos os serviços disponíveis e suas versões e, posteriormente, fazendo uma pesquisa em bancos de dados de exploração disponíveis publicamente.

Deve-se observar que os serviços públicos legais podem automatizar a varredura ativa constante em toda a Internet, atualizando seus bancos de dados constantemente. Serviços como esses permitem a busca de dispositivos não seguros que usam senhas padrão e serviços vulneráveis. Os serviços podem ser usados tanto por organizações quanto por pessoas físicas, facilitando a localização de sistemas vulneráveis por qualquer pessoa. Eles permitem a pesquisa de endereços IP, domínios e versões de servidores da Web específicos, o que torna essa outra ferramenta útil para a coleta de informações.

No entanto, a coleta de informações também pode ser feita off-line por meio de mergulho em lixeiras, observação, alegação de ser um cliente ou infiltração em uma organização como funcionário. Isso aumenta muito a chance de ser detectado e, portanto, é evitado na maioria dos casos. Somente se um invasor tiver uma motivação muito alta ou se o aprendizado sobre um alvo já tiver falhado, isso poderá ser considerado como uma abordagem para a coleta de informações.

Por fim, a coleta de informações é uma tarefa recorrente realizada durante um ataque, pois, depois que um invasor obtém acesso à infraestrutura que não está disponível publicamente, ele precisa continuar aprendendo sobre ela.

Exploração/obtenção de acesso

Depois que os invasores obtiverem um conhecimento razoável sobre seu alvo, eles farão a transição para o ataque real. "Conhecimento razoável", neste contexto, significa que eles realmente encontraram pelo menos uma possibilidade de ataque que terá uma alta probabilidade de sucesso. Um ataque bem-sucedido pode ser causado por:

- Vulnerabilidades de softwares conhecidas em software não corrigido.
- Configuração incorreta (p. ex., configuração ausente ou errada).
- Explorações de dia zero.

©International Software Testing Qualifications Board

- Senhas fracas.
- Engenharia social.

Se um ataque for bem-sucedido, o invasor geralmente não terá uma conta de administrador. Isso pode representar um obstáculo para ele, pois gostaria de alterar o sistema para seus objetivos (p. ex., interrompendo ou modificando o software antivírus). Portanto, a aquisição de níveis de privilégios mais altos geralmente é necessária após o acesso inicial. O escalonamento de privilégios pode ser bem-sucedido pelos mesmos motivos do ataque inicial, como uma vulnerabilidade de software ou uma configuração incorreta. A configuração incorreta é uma séria ameaça ao escalonamento de privilégios. Nos sistemas UNIX, por exemplo, muitos programas permitem o acesso a um shell raiz e, se permitirem o uso do SUDO (acrônimo de super user do), podem usá-lo para executar programas como superusuário ou outro usuário [GTFO22].

Embora esses sejam ataques que podem ser executados remotamente, também é possível que um invasor obtenha acesso direto da rede interna de uma organização. Um invasor pode ser um funcionário que deseja prejudicar a organização. Além disso, um invasor pode obter acesso físico a um escritório devido a um controle de acesso insuficiente e, a partir daí, conseguir se conectar à rede interna.

A engenharia social é uma ameaça que pode fazer com que o invasor obtenha acesso direto. De acordo com o NIST, ela é definida como "o ato de enganar um indivíduo para que ele revele informações confidenciais, obtenha acesso não autorizado ou cometa fraude associando-se a ele para ganhar confiança". [NIST05].

Persistência/manutenção do acesso

A obtenção de acesso não autorizado a um sistema geralmente usa exploits que podem ser difíceis de aplicar, têm alta probabilidade de falha ou só podem ser aplicados em determinados momentos. Portanto, os atacantes precisam manter o acesso ao sistema comprometido até atingirem seus objetivos. Novamente, isso depende da motivação dos atacantes, pois alguns podem estar interessados apenas em um ataque bem-sucedido, mas não querem ir além.

O acesso persistente geralmente é obtido por meio do rootkit, que é criado especificamente para essa finalidade. Os rootkits tentam manter o acesso, mesmo que o sistema seja reinicializado. Como o software antimalware tenta detectar softwares mal-intencionados, os rootkits são criados para ocultar a si mesmos e a outros malwares, como key loggers e sniffers de rede no sistema. Eles também podem permitir o controle remoto automatizado de sistemas comprometidos, permitindo que os invasores criem botnets para realizar ataques distribuídos de negação de serviço contra outros sistemas ou usá-los indevidamente como um servidor de spam.

Limpando trilhas

Como o ataque pode ter sérias consequências legais, os atacantes querem permanecer anônimos ou, pelo menos, não querem ser identificados pessoalmente. Como resultado, eles têm uma grande motivação para remover todos os vestígios de suas atividades anteriores depois de atingirem seu objetivo. Isso inclui remover todos os programas e arquivos que o invasor copiou para o(s) sistema(s) comprometido(s), limpar ou remover arquivos de registro, históricos de comandos e talvez destruir o hardware usado para o ataque.

Embora essas sejam tarefas que um invasor executa ao concluir um ataque, os invasores também usarão técnicas como encadeamento de proxy, VPN ou servidores de salto já comprometidos durante o acesso remoto para ocultar seus rastros.

Um invasor deve estar ciente de que cada atividade que ele realiza contra um sistema pode ser potencialmente registrada e, na maioria dos casos, ele não conseguirá excluir completamente todos os seus rastros.

No contexto dos testes de segurança e penetração, é necessário agir de maneira semelhante, pois o SUT pode ser um IDS ou um processo de resposta a emergências.

Resposta a incidentes e análise pós-incidente

As seções anteriores descreveram um cenário de ataque da perspectiva de um invasor. Por outro lado, as organizações investem em medidas de segurança para detectar e resolver incidentes de segurança, uma tarefa conhecida como resposta a incidentes. Observe que a resposta a incidentes e as fases de ataque descritas

©International Software Testing Qualifications Board

anteriormente geralmente ocorrem ao mesmo tempo. Em muitos casos, um invasor já foi detectado em uma fase anterior (p. ex., fase de persistência), e não apenas depois de ter tentado apagar seus rastros.

Preparação

Embora a preparação faça parte do gerenciamento de incidentes, ela não faz parte da resposta a incidentes, mas constrói a base para um procedimento de resposta a incidentes que funcionará no caso de um incidente de segurança.

A resposta a incidentes tem o objetivo de alcançar o seguinte:

- Identificar um incidente e analisar a situação
- Contenção, por exemplo, isolar os sistemas comprometidos e encerrar os serviços
- Erradicação, por exemplo, remover contas de usuário comprometidas, remover malware e corrigir um sistema
- Recuperação, por exemplo, colocar os serviços on-line novamente e restaurar

Após a resolução de um incidente, deve haver uma revisão para identificar os pontos fracos da infraestrutura e dos processos de resposta a incidentes.

Detecção e análise

A detecção de um incidente pode ser intencional ou não intencional por parte do invasor. No caso da desfiguração de um site, por exemplo, a intenção dos invasores é que o ataque seja reconhecido. Em outros casos, eles podem deixar uma mensagem para um administrador do sistema.

Diferentes ferramentas podem ser usadas para ajudar na detecção de atividades suspeitas. Elas incluem sistemas de detecção de intrusão de rede/host (NIDS/HIDS), scanners de malware e analisadores de logs. Na melhor das hipóteses, um ataque pode ser identificado antes que o invasor seja bem-sucedido. Esse pode ser o caso se um invasor for muito visível durante a varredura ou se suas primeiras tentativas de exploração falharem. Além disso, muitas tentativas de login com falha ou tentativas de login de usuários que não existem ou que normalmente não fazem login remotamente podem ser uma indicação de um possível ataque. Se isso puder ser detectado fora da rede da organização, há uma boa chance de derrotar o ataque.

Se forem detectadas atividades suspeitas dentro da rede da organização, é preciso analisar quais sistemas já estão comprometidos e como o invasor obteve acesso. Isso pode ser feito pelo departamento de TI, mas é mais comum que uma equipe forense comece a analisar o incidente. A resposta a um incidente é crítica em termos de tempo e as ações devem ocorrer o mais cedo possível para evitar maiores danos.

Contenção, erradicação e recuperação

Se houver um quadro claro sobre quais sistemas estão comprometidos, a próxima etapa é conter esses sistemas para evitar que um invasor comprometa outros sistemas. Isso pode ser feito, por exemplo, desligando serviços temporariamente, movendo-os para outra rede ou bloqueando contas de usuários.

Um aspecto importante a ser considerado durante essa fase é que algumas ações podem excluir evidências forenses, que podem ser úteis para a análise pós-incidente. Por exemplo, o desligamento de um sistema excluirá a memória principal, que pode conter dados úteis, como o exploit inicial que foi usado. Novamente, é fundamental reagir rapidamente, pois um invasor pode comprometer outros sistemas. As ações executadas devem ser decididas com base no nível de risco.

A contenção e a análise da situação se alternarão em um determinado ponto, pois é preciso reavaliar se todos os sistemas foram identificados corretamente e se o invasor não consegue obter mais acesso. Se houver certeza suficiente de que todos os sistemas afetados foram identificados e contidos, a erradicação poderá ser realizada. Um aspecto importante durante essa fase é fornecer evidências para análise posterior. A erradicação pode incluir a exclusão de um sistema inteiro e sua posterior recriação a partir de um backup. Também é possível remover apenas componentes parciais do sistema e substituí-los durante a fase de recuperação por um novo componente. Em ambos os casos, é preciso garantir que o backup ou o componente usado não contenha rastros do incidente resolvido anteriormente.

©International Software Testing Qualifications Board

Atividade pós-incidente

Após a resolução de um incidente, várias medidas devem ser tomadas para avaliar e melhorar as rotinas de segurança atuais. Isso inclui:

- Investigações forenses que, na melhor das hipóteses, podem identificar o invasor
- Fechamento de vulnerabilidades que podem ter sido reveladas por meio do ataque
- Reavaliação da infraestrutura atual
- Aumentar a conscientização de segurança dos funcionários
- Reavaliar e talvez adaptar as políticas de segurança
- Refinamento dos processos de resposta a incidentes
- Fazer comunicados a clientes ou consumidores, especialmente se a organização tiver obrigações de comunicação. Isso inclui a apresentação de relatórios à instituição correspondente ou ao governo.

©International Software Testing Qualifications Board

6 Ajustando o teste de segurança aos modelos de ciclo de vida de desenvolvimento de software - 165 minutos (K4)

Palavras-chave

ciclo de vida de desenvolvimento de software

Palavras-chave de segurança

Nenhum

Objetivos de aprendizagem do Capítulo 6:

6.1 Os efeitos de diferentes modelos de ciclo de vida de desenvolvimento de software no teste de segurança

STE-6.1.1 (K2) Resumir por que as atividades de teste de segurança devem abranger o ciclo de vida do desenvolvimento de software

STE-6.1.2 (K4) Analisar como as atividades de teste de segurança são afetadas por diferentes modelos de ciclo de vida de desenvolvimento de software

6.2 Teste de segurança durante a manutenção

STE-6.2.1 (K3) Definir e executar testes de regressão de segurança e testes de confirmação com base em uma alteração em um sistema

STE-6.2.2 (K2) Analisar os resultados dos testes de segurança para determinar a natureza de uma vulnerabilidade e seu possível impacto técnico

©International Software Testing Qualifications Board

6.1 Os efeitos de diferentes modelos de ciclo de vida de desenvolvimento de software no teste de segurança

O ciclo de vida do aplicativo ou do sistema pode ser descrito como um modelo com diferentes SDLCs. As fases mais usadas do SDLC são planejamento, análise, projeto, desenvolvimento, teste, implementação, manutenção e encerramento.

As atividades e tarefas planejadas para cada uma dessas fases podem diferir de acordo com o aplicativo, o sistema, o projeto ou a organização. Elas são definidas no modelo SDLC, que pode ser implementado usando um desenvolvimento sequencial ou uma abordagem de desenvolvimento ágil.

Usando uma abordagem de desenvolvimento sequencial, é mais fácil reconhecer as diferentes fases do SDLC. Com a abordagem ágil, pode não ser tão claro quando e qual atividade ou tarefa de qual processo do ciclo de vida é executada. As atividades e tarefas podem ser repetidas com frequência, agregando valor ao aplicativo ou sistema a cada iteração.

No syllabus ISTQB Certified Tester Foundation Level [ISTQB FL], são mencionados vários modelos de desenvolvimento, incluindo o modelo em cascata e os modelos ágeis de desenvolvimento de software (p. ex., Rational Unified Process, Scrum, Kanban e Spiral). É mencionado que os testes de segurança devem ser adaptados a esses modelos para serem mais eficazes. Como é de se esperar, as abordagens dos testes de segurança também precisam ser adaptadas aos diferentes modelos de SDLC. Este syllabus aborda o DevOps além dos modelos descritos acima.

O STE deve ter conhecimento das características mais importantes desses modelos de SDLC e de como elas podem afetar sua capacidade de realizar testes de segurança.

Ao comparar essas categorias, podem ser observadas diferenças com relação aos seguintes atributos. Uma mudança em qualquer um desses atributos também terá um impacto sobre como os testes de segurança são realizados.

Atributo	**Descrição**
Duração do desenvolvimento	O tempo necessário desde a formulação de um requisito até a implementação • A duração também afetará o tempo permitido para executar testes de segurança durante o SDLC. • Quanto menos tempo disponível, mais desafiadoras serão as escolhas e as prioridades a serem definidas
Tamanho da implementação	Lotes maiores de funcionalidade ou um único recurso por implementação • Frequentemente em relação direta com a duração do desenvolvimento • Quanto menor o tamanho de uma implantação, mais específico pode (e deve) ser o foco dos testes de segurança • Com implantações maiores, a superfície de ataque pode ser substancialmente aumentada em uma iteração. A necessidade de testes de regressão aumenta nos modelos de desenvolvimento incremental.
Tempo de teste permitido	A quantidade de recursos de tempo reservados para realizar testes • Na maioria dos casos, o teste funcional pode ser planejado, mas o teste não funcional (incluindo o teste de segurança) geralmente não é planejado nos ciclos de vida do projeto • Se o tempo permitir, isso pode criar oportunidades para testes não funcionais mais extensos, incluindo testes de segurança

©International Software Testing Qualifications Board

Atributo	Descrição
Independência da equipe	O nível das decisões de segurança que podem ser tomadas pela equipe • Isso pode permitir que a equipe atribua ou contrate competência autônoma de teste de segurança, se necessário
Funcionalidade cruzada da equipe	A disponibilidade das habilidades necessárias (para o desenvolvimento) na equipe • A equipe pode ter competências de teste de segurança diferentes e complementares disponíveis
Nível de automação	Quanto do processo de desenvolvimento é automatizado. • Grande parte dos testes de segurança pode ser realizada com a automação de testes usando atividades relacionadas ao SAST e ao DAST
Princípios de gerenciamento	A organização é uma organização de linha, de projeto ou orientada para o produto? Os princípios de como a equipe é gerenciada podem influenciar a organização dos testes de segurança. Ele pode ser realizado com equipes capacitadas, com foco contínuo na segurança, apenas durante a fase do projeto ou apenas durante a manutenção.
Ambiente de teste	Um ambiente de teste separado e dedicado pode ser estabelecido para testes destrutivos de segurança.

6.1.1 Modelos de desenvolvimento sequencial

O SDLC completo geralmente é especificado por todos os processos necessários para desenvolver, manter e dar suporte ao sistema desde o início até sua desativação/eliminação. Um padrão muito usado que descreve todos esses processos e suas relações é o [ISO 15288].

O modelo de desenvolvimento do sistema descreve a implementação de (partes do) SDLC necessário para desenvolver e implementar um sistema. Essa implementação não inclui necessariamente todos os processos do SDLC e deve ser ajustada à organização (consulte o capítulo 5) e ao contexto do projeto.

A segurança da informação e a verificação da segurança devem ser um aspecto integrado coberto por todos os processos do SDLC usados na organização. Somente assim será possível obter uma abordagem verdadeiramente holística. Isso também dará suporte ou garantirá que as atividades necessárias durante os processos de desenvolvimento possam ser conduzidas de forma consistente.

No [NIST 800-160], o desafio de segurança é descrever a segurança do sistema como um problema de design. Ele observa que "uma combinação de hardware, software, comunicações, proteções físicas, pessoais e administrativas-procedimentais é necessária para uma segurança abrangente. As proteções de software por si só não são suficientes".

Esse padrão do NIST apresenta considerações e abordagens que abrangem todos os processos do SDLC sobre como lidar com as atividades de segurança da informação.

O modelo de desenvolvimento sequencial em cascata ou sua implementação como modelo em V ainda é um modelo muito usado. O modelo em V refere-se a atividades de teste genéricas para cada uma de suas fases, com o objetivo de mudar para a esquerda. Esses modelos são mais estabelecidos em organizações com separação distinta entre as diferentes equipes e fases de desenvolvimento. Em geral, podemos esperar que o STE tenha tempo para planejar, preparar e executar testes de segurança ao usar esse modelo de desenvolvimento de software. As fases do modelo são programadas em sequência, mas podem se sobrepor umas às outras.

Nesses modelos, o STE deve estar ciente do seguinte:

- Os requisitos e riscos de segurança são definidos no início de um projeto e devem ser documentados nas especificações de requisitos de software.

©International Software Testing Qualifications Board

- Os requisitos de segurança podem mudar no decorrer do projeto à medida que novas ameaças são descobertas, mas isso pode não se refletir nos requisitos de software atualizados. Portanto, os testes de segurança podem parecer muito específicos e completos, mas podem não estar completos ou atualizados devido aos riscos tardios do projeto.
- Os testes de segurança podem ser realizados a qualquer momento ou em qualquer fase de desenvolvimento, mas é comum que sejam realizados no final do projeto.
- Pode ser difícil abordar os resultados dos testes de segurança e das correções no final de um projeto de modelo de desenvolvimento sequencial, pois os prazos serão definidos, em sua maioria, em uma fase inicial do projeto.

6.1.2 Desenvolvimento ágil de software

O desenvolvimento ágil de software promove a conclusão do trabalho a ser feito por equipes auto-organizadas e multifuncionais em iterações curtas. Equipes capacitadoras que fornecem serviços específicos ao projeto podem estar disponíveis para essas equipes de desenvolvimento ágil de software para ajudar com competências de domínio específicas, como testes de segurança.

O que é genérico no desenvolvimento ágil de software é que os incrementos (de sistema e software) são entregues em uma série de iterações. Cada uma dessas iterações pode levar de dias a algumas semanas. Os modelos de desenvolvimento ágil de software tendem a ser usados principalmente na fase de desenvolvimento do aplicativo/sistema, embora alguns modelos, como o Kanban, também possam ser aplicados durante a fase de operação.

A estrutura Scrum, em várias implementações, é a mais usada no desenvolvimento ágil de software. Toda a análise, o projeto, a codificação e os testes são feitos durante cada iteração, inclusive os testes de segurança.

Os backlogs de produtos funcionam, pelo menos parcialmente, como uma especificação de requisitos. Espera-se que tanto os requisitos de segurança quanto outros requisitos não funcionais façam parte do backlog do produto. Os épicos são divididos em várias histórias de usuários e tarefas que são selecionadas pela(s) equipe(s) para serem desenvolvidas ou entregues em um dos sprints

A funcionalidade desenvolvida pode ser alterada ou até mesmo excluída em sprints futuros. A funcionalidade "crescente" e as futuras alterações ou até mesmo exclusões criam uma base de teste instável para se trabalhar. O desenvolvimento ágil de software pode ser visto como uma combinação de desenvolvimento (nova funcionalidade) e manutenção (plataforma e funcionalidade existente) durante a fase do projeto. Portanto, a repetição de testes de segurança automatizados é essencial.

Várias abordagens podem ser adotadas para realizar as atividades de teste de segurança. Alguns exemplos são:

- Faça alguns testes de segurança e, em seguida, mude o foco para objetos de teste funcionais, técnicos ou relacionados à plataforma em cada sprint diferente
- Faça alguns testes de segurança na maioria dos sprints e realize um teste de segurança completo em um sprint dedicado
- Realizar todos os testes de segurança em um sprint (final) que se assemelha ao modelo de desenvolvimento sequencial

A equipe de desenvolvimento ágil pode envolver uma equipe capacitadora ou contratar recursos para realizar os testes de segurança, pois essas competências geralmente não residem na equipe.

Como a solução muda a cada sprint, é necessário realizar testes de regressão. A segurança não é testada ou corrigida em um aplicativo já criado. Em vez disso, ela é obtida por meio de um projeto orientado à segurança (ou seja, segurança por projeto) e verificação durante todo o processo de construção

A [Synopsys] descreveu como os testes de segurança podem ser aplicados no desenvolvimento ágil de software, aplicando os quatro princípios a seguir definidos no Manifesto Ágil:

- Desenvolvedores e testadores em vez de especialistas em segurança
- Proteger enquanto você trabalha em vez de proteger depois de terminar
- Implementar recursos de forma segura em vez de adicionar recursos de segurança
- Mitigar riscos em vez de corrigir defeitos.

©International Software Testing Qualifications Board

Desenvolvedores e testadores em vez de especialistas em segurança

Especialistas em segurança experientes são recursos valiosos. As equipes ágeis raramente se dão ao luxo de ter seus próprios especialistas em segurança dedicados. Isso significa que, na maioria das vezes, as equipes ágeis devem ser responsáveis por sua própria segurança e não podem esperar por uma revisão externa de segurança antes que o código passe para a próxima fase de desenvolvimento. A segurança deve ser integrada ao desenvolvimento e aos testes do código. As equipes devem se apropriar da segurança da mesma forma que se apropriam da experiência do usuário, da confiabilidade, da eficiência do desempenho e de outros requisitos não funcionais.

Proteger enquanto você trabalha em vez de proteger depois de terminar

A aplicação de metodologias e práticas seguras na criação, liberação e manutenção de software funcional é imprescindível. Ao mesmo tempo, as atividades de segurança não devem forçar os desenvolvedores a interromper o que estão fazendo, ir para outra ferramenta para correção e depois voltar para o que estavam fazendo. A alternativa é integrar o feedback e as informações de segurança às ferramentas do desenvolvedor. As tarefas de segurança são apresentadas (p. ex., em quadros brancos) e as prioridades são definidas para torná-las visíveis juntamente com outras tarefas.

Implementação de recursos de forma segura sobre a adição de recursos de segurança

O foco deve ser o cumprimento da missão de negócio do software, mas a integração da segurança deve sempre ser levada em consideração. Isso significa que arquitetos, desenvolvedores, testadores e outras partes interessadas devem considerar os aspectos de segurança e trabalhar juntos para definir e criar sistemas mais seguros. Os sistemas seguros devem ser projetados e criados desde o início.

Mitigando os riscos sobre a correção de defeitos

Devem ser considerados os riscos específicos da empresa, dos usuários, dos dados e do software. O gerenciamento de riscos considera a maneira correta de lidar com um risco. Isso pode ser obtido com uma visão de alto nível do que pode dar errado, em vez de reduzir a segurança a uma longa lista de defeitos individuais que precisam ser resolvidos. Embora a modelagem de ameaças seja mais difícil do que simplesmente localizar e corrigir defeitos, ela é uma abordagem eficaz para detectar problemas no início do SDLC. É definitivamente mais barato quando os problemas podem ser resolvidos antes do lançamento do software.

6.1.3 A metodologia DevOps

A maior parte do desenvolvimento ágil de software abrange a entrega do sistema ou software ao departamento de operações. O DevOps vai além, incluindo o desenvolvimento e as operações. O principal objetivo do DevOps é fornecer (pequenas) mudanças rapidamente. Além disso, a cultura da equipe tem um impacto muito maior sobre o sucesso da equipe. As equipes de DevOps geralmente são mais autônomas e mais orientadas para o produto do que as outras equipes.

O DevSecOps abrange todo o SDLC, incluindo desenvolvimento, segurança e operações. Durante o desenvolvimento, a segurança se concentra na identificação e prevenção de vulnerabilidades, enquanto nas operações, o monitoramento e a defesa contra-ataques são os principais objetivos.

Em geral, as iterações de DevOps podem ser tão curtas quanto uma hora e as entregas são tipicamente recursos/tarefas de desenvolvimento único ou pequenas ramificações. A equipe de DevOps tem como objetivo disponibilizar os resultados dos testes quase imediatamente após fazer uma alteração em cada etapa do processo de desenvolvimento. Isso exerce grande pressão sobre os testes e as metodologias em uso, o que, por sua vez, tem um grande efeito sobre as possibilidades de realizar testes de segurança.

Para obter iterações de desenvolvimento de DevOps curtas, muitas das tarefas repetitivas e com uso intensivo de recursos são automatizadas. Isso é feito na forma de um pipeline que consiste em várias fases do pipeline, cada uma das quais pode conter um ou mais trabalhos relacionados à execução de tarefas específicas no processo de compilação e implantação. Uma fase do pipeline pode ser chamada de teste do sistema e pode ter um trabalho que executa testes de regressão automatizados. Esse pipeline é então executado para cada recurso a ser entregue.

©International Software Testing Qualifications Board

O DevOps é oferecido em diferentes “sabores”. As duas abordagens mais usadas são:

- Usando uma ramificação principal ou master. As alterações são desenvolvidas e testadas em uma ramificação (branch) ou na linha principal (trunk) de recursos de curta duração e implantadas diretamente na produção após a aprovação. Isso também é conhecido como desenvolvimento baseado em tronco.
- O uso de uma ramificação de desenvolvimento separada permite que as alterações sejam entregues em um ambiente de teste continuamente e implantadas juntas em um pequeno lote após um curto período. Isso também é conhecido como desenvolvimento baseado em recursos.

As atividades de teste de segurança levam tempo. Uma pergunta comum a ser respondida envolve decidir se os testes de segurança devem ser realizados para cada pipeline ou se devem ser programados durante a noite, após a execução de alguns pipelines.

DevSecOps é um conceito que indica a importância dada aos testes de segurança em DevOps. Geralmente é considerado como vários trabalhos de teste de segurança executados automaticamente no pipeline e inclui tanto a análise estática (ou seja, *shift left*) quanto o foco no monitoramento e na prevenção relacionados à segurança, como treinamento em segurança e codificação segura.

A ênfase é colocada na segurança como uma responsabilidade da equipe, com a segurança sendo considerada parte de todas as atividades de desenvolvimento durante todas as fases para todos na equipe.

Os desafios comuns na implementação de testes de segurança usando a metodologia DevOps são:

- O teste de segurança ainda é considerado uma tarefa especializada a ser realizada por recursos específicos. Isso inibe a tão necessária integração dos testes de segurança na equipe de DevOps.
- A segurança pode se tornar excessivamente prioritária, fazendo com que outras características de qualidade, como eficiência de desempenho e usabilidade, sejam negligenciadas. A segurança é importante, mas precisa ser implementada em uma abordagem equilibrada.
- A segurança pode levar à inibição da criatividade do desenvolvedor, da autonomia da equipe e da possibilidade de fazer experiências. Esses atributos são considerados essenciais para uma implementação bem-sucedida do DevOps.

O Projeto Phoenix [TechTarget] descreve as seguintes práticas necessárias para uma implementação bem-sucedida do DevOps. Elas também devem ser aplicadas aos testes de segurança:

- Criar e manter um fluxo de tarefas
  O teste de segurança é geralmente considerado uma tarefa grande ou única (p. ex., teste de aplicativo, verificação de rede e revisão de arquitetura). Ao aplicar o DevOps, é necessário planejar, preparar e executar testes de segurança em tarefas menores. Elas devem progredir (fluir) da mesma forma que outras tarefas de desenvolvimento, aplicando conceitos como tornar as tarefas visíveis, limitar o trabalho em andamento, atribuir tarefas individuais a pessoas individuais e automatizar sempre que possível.
- Garantir feedback instantâneo
  Os resultados dos testes devem estar disponíveis o mais rápido possível. Eles devem ser compreensíveis e solucionáveis. A capacidade de fazer isso é apoiada pelo fluxo mencionado acima e pelo uso de tarefas menores. Isso também ajuda a reduzir o débito técnico.
- Incentivar uma cultura de segurança de DevOps
  A equipe deve ser aberta e transparente em relação às questões de segurança. Eles devem ser motivados a relatar problemas relacionados à segurança e considerar a segurança como uma responsabilidade da equipe.

Em termos conceituais, o DevOps permite falhar, aprender e melhorar. Ao introduzir o DevOps, considera-se melhor começar com algumas pequenas melhorias e colocar o fluxo em funcionamento.

Para ser eficiente, o engenheiro de testes de segurança deve integrar todas as tarefas necessárias relacionadas à segurança no pipeline de DevOps (CI/CD) descrito acima. Isso significa não apenas concentrar-se nos testes de segurança, mas também formular e aprimorar épicos, histórias de usuários e tarefas durante a fase de planejamento.

©International Software Testing Qualifications Board

6.2 Teste de segurança durante operações e manutenção

6.2.1 Teste de Regressão de Segurança e Teste de Confirmação

Os testes de segurança continuam depois que um sistema é colocado em produção. Podem ocorrer alterações no ambiente técnico, nos sistemas externos e nas integrações do SUT. As alterações podem se dever a atualizações regulares de segurança ou a outras alterações em middleware, firmware e hardware. Além disso, a SUT estará sujeita a alterações planejadas e não planejadas que podem abrir novas vulnerabilidades e possíveis ataques.

Todas essas alterações exigem, no mínimo, testes periódicos de regressão de segurança. Dependendo do tamanho das alterações, pode ser necessário um novo teste de segurança.

Os testes de segurança podem envolver a verificação de que o sistema continua a resistir com sucesso às tentativas de burlar os controles de segurança estabelecidos. Os aprimoramentos na eficiência da usabilidade ou do desempenho são especialmente propensos a afetar negativamente os controles de segurança.

O teste de regressão de segurança deve se concentrar na confirmação da satisfação de todos os requisitos de segurança e no teste de novas vulnerabilidades que possam ter sido introduzidas durante as atividades de manutenção.

O teste de regressão geralmente é aplicado com uma coleção de casos de teste que se baseiam no teste de funções individuais. No entanto, para testes de segurança, muitas vezes não é suficiente detectar defeitos de regressão com impacto na segurança. Os cenários de teste de regressão de ponta a ponta são mais robustos e oferecem um nível mais alto de confiança de que as transações completas podem ser realizadas de forma segura. Para esse tipo de teste de regressão, um conjunto de condições de teste de segurança deve ser definido e testado sempre que uma alteração for feita no sistema. Os defeitos de regressão podem surgir de alterações em todos os parâmetros relevantes do sistema. Alguns dos defeitos de regressão podem ter um impacto na segurança.

Depois que um sistema é colocado em produção, pode ser necessário um esforço adicional de desenvolvimento para corrigir defeitos na versão lançada (ou seja, manutenção corretiva), para se ajustar a outras mudanças no ambiente operacional (ou seja, manutenção adaptativa) ou para estender ou aprimorar recursos (ou seja, manutenção perfectiva).

A perspectiva do teste de segurança para a manutenção do sistema concentra-se em testar as alterações feitas para corrigir defeitos e a funcionalidade principal. O objetivo disso é:

- para garantir que nenhuma nova vulnerabilidade tenha sido introduzida na SUT
- para verificar se as defesas de segurança existentes ainda são eficazes após uma alteração

Parte do processo de manutenção é manter os firewalls e outras tecnologias de segurança atualizados. O monitoramento contínuo do sistema pode detectar atividades suspeitas que talvez precisem ser tratadas imediatamente.

©International Software Testing Qualifications Board

7 Teste de Segurança como parte de um Sistema de Gerenciamento de Segurança da Informação - 105 minutos (K3)

Palavras-chave

Nenhum

Palavras-chave de segurança

Sistema de gerenciamento de segurança da informação (ISMS)

Objetivos de aprendizagem para o Capítulo 7:

7.1 Critérios de aceite para testes de segurança

STE-7.1.1 (K2) Compreender os critérios de aceite dos testes de segurança e como eles influenciam a seleção de abordagens e técnicas de teste de segurança.

7.2 Entrada para um sistema de gerenciamento de segurança da informação

STE-7.2.1 (K2) Compreender o papel dos testes de segurança para um sistema eficaz de gerenciamento de segurança da informação.

7.3 Aprimoramento de um sistema de gerenciamento de segurança da informação por meio do ajuste dos testes de segurança

STE-7.3.1 (K3) Avaliar a maturidade do sistema de gerenciamento da segurança da informação, introduzindo diferentes abordagens de teste, novos objetos de teste ou cobertura aprimorada.

STE-7.3.2 (K2) Compreender a mensurabilidade em um sistema de gestão de segurança da informação.

©International Software Testing Qualifications Board

7.1 Critérios de aceite para o teste de segurança

Os testes de segurança podem ser aplicados como uma atividade pontual e ad hoc para um sistema antes de entrar em produção ou como um processo contínuo e sistemático no desenvolvimento. Ambos os tipos geram resultados de testes, mas sua capacidade de fornecer evidências de riscos significativos à segurança varia muito. Ele se aplica a diferentes técnicas de teste de segurança. Por exemplo, o teste de segurança caixa-branca gerará resultados de teste e poderá identificar outras vulnerabilidades além daquelas geradas pelo uso do teste de segurança caixa-preta

Assim como na engenharia de software em geral, os requisitos para testes de segurança raramente são claramente definidos, completos, precisos ou consistentes. O início dos testes de segurança com base nesses requisitos mal definidos pode gerar alguns resultados de teste, mas seu valor depende da qualidade dos requisitos, que não é predefinida. Qualquer ação baseada em tais requisitos provavelmente será arriscada pelos seguintes motivos:

- Técnicas de teste questionáveis:
  O teste de segurança usa uma técnica específica ou uma combinação de diferentes técnicas de teste, cada uma delas com pontos fortes e fracos. A melhor técnica de teste não existe por si só, mas existem algumas preferências para atingir um determinado objetivo de teste. Sem objetivos de teste, todas as técnicas de teste podem corresponder às expectativas, mas sem nenhuma garantia de criação de valor agregado.
- Cobertura questionável:
  A maioria das técnicas de teste não tem uma definição predefinida de completo, mas precisa de métricas bem definidas que devem ser atendidas para que o teste possa ser considerado concluído. O tipo de métrica e seus limites dependem dos objetivos do teste de segurança. Sem eles, o STE pode atingir um nível de cobertura que talvez não atenda aos objetivos do teste.

Para evitar essas armadilhas, é essencial definir critérios de aceite antes de qualquer teste de segurança. Esses critérios devem ser cumpridos antes de usar os resultados do teste como base para identificar quaisquer desvios ou itens de ação.

A palavra aceito é fundamental. ([WaCh90]) afirma que isso significa "que os produtos de software intermediários e finais são examinados para determinar se atendem a critérios específicos. Se atenderem, terão sido aceitos". Naturalmente, os requisitos de segurança devem ter seus próprios critérios de aceite (cf. ([WaCh90]), que apoiam a decisão de aceite: rejeitar, aceitar parcialmente ou aceitar.

Os testes de segurança podem ser bem adequados para controlar os critérios de aceite de segurança definidos para um SUT. Os resultados do teste, que geralmente são agregados em um relatório de teste, devem conter todas as informações necessárias para permitir uma decisão de aceite. Para apoiar essa tomada de decisão, a abordagem de teste de segurança selecionada deve se basear nos critérios de aceite específicos. Normalmente, isso é feito nas etapas a seguir:

- Leia atentamente os critérios de aceite de segurança
- Liste as possíveis técnicas de teste de segurança (consulte o capítulo 2) que podem ser usadas para apoiar a decisão de aceite. Lembre-se de que algumas técnicas de teste de segurança podem abranger de zero a muitos critérios de aceite
- Crie um conjunto específico de testes de segurança para esses critérios de aceite específicos. Os princípios orientadores para isso são:
  - Possibilidade de aplicação:
    É possível usar uma técnica de teste específica para o SUT? (p. ex., há ferramentas correspondentes disponíveis?)
  - Otimização de custo, tempo e qualidade:
    O desafio é definir um conjunto de testes e ferramentas específicos que gerem resultados de teste significativos, que possam ser aplicados em um determinado período e que sejam econômicos.

©International Software Testing Qualifications Board

Depois de selecionar as melhores técnicas e ferramentas de teste de segurança, o teste de segurança deve ser aplicado e os resultados do teste devem ser analisados e relatados no relatório de teste. O próprio relatório de teste deve refletir os critérios de aceite e formar a base para a decisão de aceite da segurança.

7.2 Entrada para um sistema de gerenciamento de segurança da informação (ISMS)

O teste em si não melhora a qualidade. Os testes fornecem informações sobre a qualidade alcançada para uma característica de qualidade específica, como a segurança. Um relatório de teste não melhora o nível de segurança. Se as conclusões do relatório de teste forem analisadas e a maioria delas for resolvida, um teste de confirmação poderá ser apropriado para demonstrar um aumento na segurança. Além dessas ações de atenuação de riscos específicos do sistema, algumas das descobertas podem motivar uma política de segurança específica para evitar tais vulnerabilidades no futuro. Por outro lado, algumas dessas descobertas podem ter suas raízes em uma falta de conscientização sobre segurança ou no uso de técnicas imaturas/não sistemáticas.

Para aproveitar os testes de segurança de forma eficaz e eficiente, eles devem ser integrados em um processo geral de segurança. Ele deve tentar minimizar os riscos e garantir a continuidade dos negócios, limitando proativamente o impacto de uma violação de segurança. Esse é exatamente o objetivo de um ISMS (*Information Security Management System*). O [ITGov23b] define um ISMS da seguinte forma:

- O ISMS adota uma abordagem sistemática para garantir a confidencialidade, a integridade e a disponibilidade dos ativos de informações corporativas
- Um ISMS ISO 27001 consiste em políticas, procedimentos e outros controles que envolvem pessoas, processos e tecnologia.
- Um ISMS é uma maneira eficiente de manter os ativos de informação seguros, com base em avaliações de risco regulares e abordagens neutras em relação à tecnologia e ao fornecedor

Um ISMS adota uma visão holística da segurança e garante a interação efetiva dos três principais atributos da segurança da informação:

- Processo
- Tecnologia
- Comportamento dentro da organização [Cald11]

O teste de segurança de aplicativos ou sistemas está diretamente relacionado à tecnologia. No entanto, cada vulnerabilidade identificada pelos testes de segurança pode ter suas raízes no processo e/ou no comportamento e pode ser atenuada no futuro por meio de mudanças nos processos e/ou no comportamento.

Há pelo menos os seguintes motivos para uma organização implementar um ISMS que são diretamente apoiados por testes de segurança [Cald11]:

- Estratégico, para gerenciar melhor a segurança das informações no contexto dos riscos gerais do negócio
- Confiança do cliente, para demonstrar que uma organização está em conformidade com as práticas recomendadas de gerenciamento de segurança da informação
- Regulamentar, para atender a vários requisitos regulamentares
- Eficácia interna, para gerenciar taticamente as informações de forma mais eficaz em uma organização.

Os testes de segurança desempenham um papel importante no estabelecimento de um ISMS. Como está relacionado a testes, ele demonstra o status quo de um sistema. Isso pode ser entendido da seguinte forma:

- Como evidência de uma meta que está planejada para ser alcançada
- Como evidência de um ponto de partida, que motiva outras ações de segurança

Um ciclo de feedback bem conhecido para modelar a meta e o ponto de partida é o ciclo Planejar-Fazer-Verificar-Agir (PDCA). A ISO 27001 "adota o modelo de processo PDCA, que é aplicado para estruturar todos os processos do SGSI" [Cald11]. Ambos os aspectos, meta e ponto de partida, são visíveis nesse modelo:

©International Software Testing Qualifications Board

- Meta:
  Após as etapas Plan (Planejar) e Do (Fazer), a etapa Check (Verificar) estabelece se a meta planejada foi atingida.
- Ponto de partida:
  O resultado da etapa Verificar, que é apoiada por atividades de teste de segurança, é analisado na etapa Agir para melhorar o processo geral. Seus desvios são a base para o próximo ciclo PDCA.

O teste de segurança oferece o maior valor agregado em uma organização se considerar os dois aspectos (ou seja, meta e ponto de partida)

Para medir a realização das etapas Planejar e Realizar em termos de segurança aprimorada, uma abordagem de teste de segurança deve ser ajustada com precisão, de modo que o teste de segurança corresponda exatamente à meta planejada. É uma boa prática definir os critérios de aceite dos testes de segurança para a etapa Verificar com antecedência ao definir o plano de segurança.

Para aproveitar os testes de segurança como ponto de partida para o próximo ciclo PDCA, a técnica de teste, as ferramentas aplicadas, os conjuntos de testes executados e todos os resultados dos testes (tanto os positivos quanto os negativos) devem ser relatados no relatório de teste.

Pode ser possível aproveitar a mesma abordagem de teste de segurança para definir um novo ponto de partida e para medir a eficácia do próximo ciclo PDCA. Os testes de segurança devem ser aplicados em uma abordagem muito sistemática. Todos os parâmetros relevantes devem ser armazenados para permitir um teste de segurança objetivo e repetível.

7.3 Aprimoramento de um ISMS por meio do ajuste do teste de segurança

A eficácia de um ISMS (*Information Security Management System*) definido para oferecer maior segurança depende muito das ações proativas introduzidas por um ISMS. Isso significa que um ISMS deve derivar controles de segurança com base no status de segurança atual e em uma avaliação ou situação de negócio atual (p. ex., um incidente).

Além de qualquer ação direta de mitigação de riscos realizada para corrigir problemas identificados em operações normais, o ISMS tenta derivar controles que impeçam a ocorrência de tais problemas em desenvolvimentos futuros. Quanto mais iterações PDCA um ISMS específico tiver experimentado, melhor será o conjunto de controles de segurança desviados e melhor será o nível de segurança de todos os aplicativos desenvolvidos.

7.3.1 Aprimorando a visão holística de um ISMS

Para aprimorar um ISMS aumentando sua visão holística, os testes de segurança devem assumir uma nova responsabilidade, além de definir a linha de base para a etapa de verificação dentro do ciclo PDCA. Se o objetivo for melhorar a maturidade do ISMS, o STE deverá trazer aspectos completamente novos que ainda não foram planejados como parte do ciclo PDCA. Esses novos testes de segurança podem gerar percepções adicionais sobre o SUT, que podem então ser usadas para melhorar ainda mais a maturidade do ISMS, derivando controles de segurança adicionais.

As dimensões típicas que um STE pode usar para aprimorar o escopo do ISMS são

- Objetos de teste adicionais:
  Cada sistema que precisa atingir um nível específico de segurança deve ser considerado em seu ambiente típico quando estiver em produção. O sistema pode estar localizado atrás de um firewall, ou conectado a um banco de dados central, ou ter uma interface API para um sistema/aplicativos externos, ou ser controlado por um processo de backup diário, ou ter uma conexão com um sistema de gerenciamento de contas privilegiadas. Quanto mais sistemas estiverem conectados à SUT, mais ampla será a superfície de ataque. Cada sistema pode ser atacado e, se for quebrado, há uma grande probabilidade de que a rede geral de sistemas também falhe.

  Um ISMS deve abranger o maior número possível de aspectos para gerenciar a segurança das informações da forma mais holística possível: Cada aspecto deve refletir os possíveis vetores de ataque.

©International Software Testing Qualifications Board

Para fazer isso, é essencial que um STE se concentre no maior número possível de objetos de teste, pois qualquer um deles pode ser um componente de risco em uma rede geral. Se um dos testes de segurança identificar pontos fracos adicionais em um componente que ainda não faz parte do ISMS geral, ele poderá melhorar a maturidade do ISMS com base nessas novas informações. Não é tarefa do STE definir contramedidas (p. ex., políticas de segurança ou ajustes de processos), mas sua tarefa inclui ter a mente aberta em relação a componentes adicionais do SUT que sejam úteis para amadurecer o ISMS.

- Abordagens de teste adicionais:
  Outra possibilidade para que um STE traga insights adicionais sobre um sistema é usar diferentes tipos de testes. Mesmo que a ação de verificação como parte do ciclo PDCA se concentre em um teste dinâmico de caixa preta do SUT, pode ser vantajoso realizar também um teste estático. Isso poderia mostrar vulnerabilidades adicionais que não foram identificadas até agora e que também poderiam ser usadas por invasores. Esses novos insights devem ser aproveitados como entrada em um ISMS para melhorar ainda mais sua maturidade.

- Cobertura aprimorada:
  Mesmo quando permanecer com o objeto de teste e a abordagem de teste existentes, o STE pode gerar percepções valiosas para o amadurecimento do ISMS. O simples fato de adicionar mais alguns casos de teste pode gerar percepções completamente novas. Isso pode ser feito facilmente com o uso de ferramentas de teste de fuzz estruturadas ou tabelas rainbow.

  Outra maneira de melhorar a cobertura pode ser aumentar o número de casos de teste executados por unidade de tempo (p. ex., automatizando alguns conjuntos de testes) ou aumentar o número de ciclos de teste realizados para impor um comportamento incomum que possa ser usado para ataques. Se essas medidas de cobertura aprimoradas identificarem vulnerabilidades adicionais, isso ajudará a melhorar a maturidade do ISMS.

7.3.2 Melhorando a mensurabilidade em um ISMS

O STE pode melhorar a maturidade do ISMS introduzindo um ciclo de feedback baseado em métricas. Essas métricas geralmente são chamadas de indicadores-chave de desempenho e dão suporte à melhoria contínua usando ciclos PDCA. A ideia fundamental do ciclo PDCA subjacente é verificar a eficácia das ações anteriores de Planejar e Executar. Quanto mais objetiva for essa ação de verificação, mais objetivo será o ciclo de feedback. As políticas derivadas do ISMS incluem métricas de cobertura para técnicas e comportamentos de teste. Se uma política do portfólio geral de políticas de uma organização incluir o uso direto de uma técnica de teste, os testes de segurança poderão verificar diretamente se essa política foi bem-sucedida

Mesmo que uma política atinja apenas indiretamente os processos usados, o STE ainda pode apoiar a medição de sua eficácia. Por exemplo, pode ser difícil medir diretamente a eficácia de algum treinamento dado em relação às convenções de codificação segura. Uma maneira poderia ser a realização de exames ao final de cada sessão de treinamento. Outra maneira seria um teste de segurança que verifique com precisão as ocorrências de anti-padrões de segurança que foram abordados no treinamento. Quanto mais frequentemente esses anti-padrões ainda existirem após o treinamento, menor será o valor agregado pelo treinamento em termos de segurança.

O mesmo pode ser dito ao avaliar a eficácia de um comportamento indesejado. Por exemplo, o teste de segurança pode definir uma simulação de phishing que conta o número de cliques indesejados feitos em um e-mail. Ao considerar o phishing, as taxas de cliques mais altas geralmente são vistas como ruins porque significam que os usuários não percebem que o e-mail é phishing, enquanto as taxas de cliques baixas geralmente são vistas como boas. Entretanto, para medir a eficácia de uma iniciativa de conscientização, esse teste de segurança deve ser repetido para permitir a medição de uma mudança antes e depois. [StGrTh20]

Os testes de segurança podem melhorar a maturidade do ISMS porque os ciclos de feedback são baseados em fatos gerados pelos testes de segurança. Os aprimoramentos ocorrem mais rapidamente e são mais confiáveis do que os baseados em sentimentos ou subjetivos.

©International Software Testing Qualifications Board

8 Relatório dos Resultados dos Testes de Segurança - 135 minutos (K3)

Palavras-chave

hacker ético, mitigação de riscos, relatório de teste

Palavras-chave de segurança

Nenhum

Objetivos de aprendizagem para o Capítulo 8:

8.1 Relatório de teste de segurança

STE-8.1.1 (K2) Compreender a importância dos resultados dos testes de segurança e como isso afeta seu manuseio e comunicação

8.2 Identificação e análise de

STE-8.2.1 (K3) Avaliar os resultados de um determinado teste de segurança para identificar vulnerabilidades

8.3 Fechar vulnerabilidades

STE-8.3.1 (K3) Avaliar diferentes técnicas para eliminar as vulnerabilidades identificadas

©International Software Testing Qualifications Board

8.1 Relatórios de testes de segurança

Cada teste de segurança termina em um relatório de teste [Glossário ISTQB]. Sem o relatório de teste, o teste não tem evidências que possam ser usadas para determinar ações ou decisões com base no resultado do teste.

As informações padrão a seguir são importantes ao relatar um caso de teste de segurança com falha:

- Ambiente de teste usado: Isso geralmente inclui endereços IP específicos, lista de permissões de IP aplicada e contas/senhas usadas.
- Condições prévias dos testes executados. Isso inclui todas as atividades de preparação a serem aplicadas antes que o conjunto de testes preparado possa ser executado. Isso pode incluir atividades como detalhes de login, definições específicas de arquivos de configuração ou configurações específicas de perímetro.
- Dados de teste usados.
- Procedimento de execução do teste.
- Resultados esperados e resultados reais.

Um teste de segurança com falha significa que um teste específico detectou a violação de pelo menos um aspecto de segurança da tríade CIA (consulte o Capítulo 1). Um bom relatório de teste inclui um nível de detalhe suficiente para permitir que o teste seja repetido. As ferramentas usadas para executar os testes podem ser nomeadas no relatório de teste e as capturas de tela podem ser incluídas para apoiar os resultados do teste com evidências.

Em geral, os relatórios de testes de segurança devem ser tratados com alto nível de confidencialidade. Se esse tipo de informação vazar para fora da organização, isso poderá reduzir drasticamente a reputação da organização. Pior ainda, as informações podem ser usadas para atacar qualquer sistema que inclua essa vulnerabilidade.

Quanto mais testes falhos um relatório de teste de segurança contiver, mais crítico e sensível será o relatório de teste e sua comunicação. Em geral, todo relatório de teste de segurança deve ser comunicado com cuidado dentro da organização. Isso inclui comunicações internas dentro da organização que produz o SUT, pois os invasores podem vir de dentro da organização (cf. [SwissCybInst20]). Por outro lado, os relatórios de testes de segurança podem ser importantes para muitas pessoas em uma organização. Esse paradoxo influencia diretamente as atividades de relatório do STE e geralmente é resolvido com a criação de diferentes versões do mesmo relatório de teste, cada uma contendo diferentes níveis de detalhes. Cada versão do relatório de teste deve seguir o conceito de "necessidade de saber". Isso se aplica àqueles cuja tarefa é mitigar os riscos identificados e, portanto, receber um relatório de teste completo ou fragmentos dele com base no conceito de "necessidade de conhecimento".

A sensibilidade de um relatório de teste de segurança pode ser modificada de acordo com as vulnerabilidades identificadas. Isso é essencial quando os hackers éticos identificam uma vulnerabilidade em um SUT e desejam informar apenas o desenvolvedor para dar a ele a oportunidade de mitigar esse risco antes que o relatório de teste seja disponibilizado publicamente. Essa divulgação responsável é uma das características de um hacker ético, especialmente para os *white-hat hackers* [Huneidy21]. Os *grey-hat hackers* geralmente usam a publicação de relatórios de teste para aumentar a pressão sobre uma organização para que ela trabalhe em correções [Huneidy21].

8.2 Identificação e análise de vulnerabilidades

É importante observar que a ausência de um conjunto de testes com falha não significa que o sistema não tenha defeitos. Mesmo os conjuntos de testes aprovados não significam necessariamente que o vetor de ataque examinado não possa ser explorado. Isso simplesmente afirma que, com os conjuntos de testes usados, não é possível ser explorado por um vetor de ataque analisado.

Se um teste de segurança falhar, será identificada uma possível vulnerabilidade. O relatório de teste deve fornecer todas as evidências necessárias para repetir o caso de teste que falhou. Um relatório de teste de segurança pode demonstrar muitas vulnerabilidades. As etapas a seguir devem ser executadas antes de qualquer ação corretiva:

- Demarcação de vulnerabilidades:
  Normalmente, um teste com falha representa um único caso de teste com falha e representa uma vulnerabilidade. No entanto, podem existir vários outros casos de teste que mostram a mesma

©International Software Testing Qualifications Board

vulnerabilidade. Por exemplo, se um parâmetro de entrada vazio pode ser usado para assumir o controle de um aplicativo, talvez o mesmo comportamento possa ser obtido ao usar um arquivo de 100 MB como parâmetro de entrada. Durante a fase de demarcação da vulnerabilidade, testes diferentes, mas semelhantes, são executados para demarcar a vulnerabilidade identificada. Isso é importante para a avaliação de risco subsequente e deve ser apoiado pelo STE.

- Ajuste da probabilidade de risco:
  Essa etapa é realizada para verificar novamente a probabilidade de risco de conseguir identificar uma vulnerabilidade na produção. Normalmente, um teste de segurança não é realizado em um sistema de produção. Mesmo que o ambiente de teste seja semelhante, ele nunca será totalmente idêntico ao ambiente de produção. Em particular, alguns controles de segurança podem ser explicitamente desativados para permitir a realização de um teste de segurança. Se um SUT demonstrar uma vulnerabilidade, ela poderá ser ofuscada por outros parâmetros implementados para a produção. Se esse for o caso, a vulnerabilidade ainda existe, mas não pode ser explorada diretamente devido a outros parâmetros. Esse ajuste pode alterar o nível de risco sugerido pelo teste de segurança e, portanto, pode alterar a necessidade geral de planejar ações de mitigação de risco. O STE tem a tarefa de considerar esse ajuste de risco e tomar as medidas correspondentes.

- Ajuste do impacto do risco:
  Essa etapa é realizada para verificar novamente o possível impacto do risco decorrente da exposição a uma vulnerabilidade. Normalmente, o foco do STE está nos aspectos técnicos, o que torna difícil estimar o cálculo do impacto nos negócios. Especialmente se o STE reutilizar avaliações de impacto para vulnerabilidades identificadas de uma fonte externa (consulte o capítulo 4, CVSS), elas podem ser imprecisas para um contexto específico. Se uma vulnerabilidade for identificada, as partes interessadas do negócio refinam o possível impacto do risco. A vulnerabilidade pode ser estimada como não tendo impacto de uma perspectiva de negócio (p. ex., se o componente afetado for raramente usado) ou pode ser considerada como tendo um alto nível de criticidade do negócio. Esse ajuste pode alterar o nível de risco sugerido pelo STE e garantir uma atualização das ações planejadas de mitigação de riscos.

Se todas as três etapas acima forem aplicadas, será possível obter uma visão clara da vulnerabilidade identificada e do respectivo risco identificado. O ajuste da probabilidade do risco e do impacto do risco deve considerar alguns parâmetros importantes:

- Comunicação próxima com os stakeholders do negócio, pois eles têm a palavra final ao discutir o possível impacto do risco
- Comunicação próxima com a equipe de operações, pois ela tem a palavra final ao considerar os parâmetros de produção e como eles diferem daqueles usados nos testes de segurança
- Mesmo que a existência da vulnerabilidade no SUT seja clara, os stakeholders podem tentar influenciar ativamente a etapa de ajuste de risco (ajustando a probabilidade ou o impacto)

Se o nível de risco restante for considerado muito alto para entrar em produção ou permanecer em produção, deverá ser criado um plano de mitigação de riscos. A gerência é responsável por decidir sobre a urgência de tais planos de mitigação de riscos. A decisão pode ser tomada entre os seguintes níveis de urgência:

- Interromper a operação imediatamente ou interromper outras atividades de entrada em operação
  Se o nível de risco for considerado muito alto e não puder ser aceito, ele só poderá ser evitado se o SUT não for executado. O nível de risco (p. ex., alto, médio ou baixo) que influencia essa decisão depende do apetite pelo risco, que é definido como "*a quantidade de risco que uma organização está disposta a aceitar para atingir seus objetivos*" [CARM22]
- Continuar a executar o sistema com monitoramento intensivo:
  Se o sistema for muito crítico ou o risco for muito crítico, o SUT poderá continuar em execução, mas será monitorado intensivamente.

©International Software Testing Qualifications Board

- Adicionar ações de mitigação de riscos ao plano de liberação normal:
  Se for possível lidar com o risco ou se o sistema tiver muitas restrições rigorosas de versão, as ações de mitigação de risco serão analisadas e executadas, mas não serão aplicadas diretamente ao SUT. Em vez disso, os componentes corrigidos são adicionados ao ciclo de versão normal para garantir que a próxima versão planejada contenha as correções de segurança necessárias.

O STE deve garantir que testes de confirmação e testes de regressão sejam realizados para cada ação de mitigação de risco. O teste de confirmação deve considerar as evidências fornecidas no relatório de teste e deve usar as lições aprendidas na etapa de demarcação de vulnerabilidade.

8.3 Fechar vulnerabilidades identificadas

Se uma vulnerabilidade identificada for conhecida em termos de demarcação de vulnerabilidade e se for decidido mitigar o risco, pelo menos duas alternativas de alto nível estarão disponíveis para mitigar as vulnerabilidades identificadas:

- Ocultar a vulnerabilidade reduzindo o risco esperado
- Evitar a vulnerabilidade aplicando patches no sistema afetado

Ambas as alternativas podem ser combinadas. Ocultar a vulnerabilidade pode ser uma solução de curto prazo e a correção adequada pode ser feita posteriormente.

8.3.1 Escondendo uma vulnerabilidade

A ideia de ocultar vulnerabilidades é semelhante à abordagem que um testador aplica quando diferencia entre um defeito que não causa uma falha e um defeito que causa uma falha (ou seja, que pode afetar um cliente).

Mesmo que as etapas de ajuste de risco tenham mostrado que a vulnerabilidade identificada é causada por uma falha (ou seja, expõe um alto risco), o próprio sistema pode ser alterado de forma que o defeito não seja revelado ao cliente, mesmo que o defeito permaneça inalterado. Em termos de mitigação de risco, esse tipo de ação visa à redução do impacto do risco. O defeito ainda existe no sistema, mas não pode mais ser explorado. As abordagens típicas para ocultar vulnerabilidades dessa forma são:

- Bloqueio de tráfego:
  Os firewalls modernos permitem análises e mecanismos de bloqueio muito sofisticados. Se o tráfego necessário para explorar a vulnerabilidade for bem compreendido, muitos firewalls podem ser configurados para bloquear esses padrões de tráfego. Ao fazer isso, a vulnerabilidade permanece inalterada, mas não pode mais ser explorada.
- Aplicação de patches virtuais:
  A aplicação de patches virtuais não necessariamente bloqueia o tráfego, mas o converte de forma que a vulnerabilidade não possa ser explorada. A [OWASP11] define isso como "uma camada de aplicação de política de segurança que impede a exploração de uma vulnerabilidade conhecida". O patch virtual funciona porque a camada de aplicação de segurança analisa as transações e intercepta os ataques em trânsito, de modo que o tráfego mal-intencionado nunca chega ao aplicativo. O impacto resultante de um patch virtual é que, embora o código-fonte real do aplicativo em si não tenha sido modificado, a tentativa de exploração não é bem-sucedida."
- Desligamento ou reconfiguração de recursos específicos do sistema
  Se a vulnerabilidade tiver um escopo muito limitado no sistema, talvez seja possível desativar a funcionalidade afetada pela vulnerabilidade identificada.
- Reduzir o escopo das vulnerabilidades
  Pode ser possível aceitar o risco associado a uma vulnerabilidade reduzindo seu escopo. Por exemplo, pode ser possível configurar filtros de IP para que somente máquinas conhecidas com endereços IP dedicados possam se conectar à máquina vulnerável. Além disso, pode ser possível desativar o acesso externo à máquina vulnerável e permitir apenas o acesso interno.

©International Software Testing Qualifications Board

O STE é responsável, durante o teste de confirmação, por aplicar todas as abordagens de vulnerabilidade oculta para garantir que a vulnerabilidade específica não possa mais ser explorada.

8.3.2 Como evitar uma vulnerabilidade

Em geral, evitar a vulnerabilidade é uma ação cara e que consome muito tempo. As etapas a seguir devem ser executadas para evitar uma vulnerabilidade:

Etapa	Descrição
Localize a vulnerabilidade	• Como a ação de redução de riscos deve ser tomada no nível usado para implementar a funcionalidade (p. ex., código, modelos e configurações), pode levar algum tempo para identificar o componente afetado e, dentro desse componente, a área afetada com base em uma vulnerabilidade identificada no nível do sistema
Entenda a vulnerabilidade	• Antes de fazer um reparo, é necessário obter um entendimento completo da vulnerabilidade por meio da análise da vulnerabilidade na área afetada (p. ex., trecho de código)
Identificar a ação de mitigação de riscos	• Deve-se desenvolver uma abordagem para a mitigação de riscos. • A ação de atenuação pode consistir em um algoritmo completamente novo, um novo componente, uma pequena alteração na configuração ou apenas alguns pequenos ajustes no código (p. ex., incluir um comportamento de exceção específico).
Executar a ação de mitigação de riscos	• A ação de mitigação de risco identificada é aplicada.
Teste de confirmação	• Realize um teste de confirmação no sistema para verificar se a vulnerabilidade foi eliminada.
Teste de regressão	• O teste é uma parte fundamental para evitar vulnerabilidades e não deve se concentrar apenas no código alterado. É essencial que o conjunto completo de testes de regressão seja executado para garantir que o sistema ainda esteja funcionando corretamente e que a ação de atenuação não tenha tido efeitos colaterais indesejados.
Implementar	• Implemente o sistema corrigido • Depois que o sistema é implantado, geralmente há um monitoramento rigoroso por um período para garantir que o sistema esteja funcionando bem.

©International Software Testing Qualifications Board

9 Ferramentas de teste de segurança - 90 minutos (K3)

Palavras-chave

Dynamic Application Security Testing, Interactive Application Security Testing, Static Application Security Testing, varredura de vulnerabilidades

Palavras-chave de segurança

software de código aberto, análise de composição de software (SCA)

Objetivos de aprendizagem do Capítulo 9:

9.1 Categorização das ferramentas de teste de segurança

STE-9.1.1 (K3) Analisar diferentes casos de uso e aplicar categorizações para ferramentas de teste de segurança

9.2 Seleção de teste de segurança

STE-9.2.1 (K2) Compreender o uso e os conceitos das ferramentas de teste de segurança dinâmica

STE-9.2.2 (K2) Compreender o uso e os conceitos das ferramentas de teste de segurança estática

©International Software Testing Qualifications Board

9.1 Categorização das ferramentas de teste de segurança

Há várias possibilidades de categorizar as ferramentas de teste de segurança. Uma seleção dessas categorias inclui:

- A atividade no processo de teste de segurança em que a ferramenta pode ser usada
- Ferramentas de teste de segurança de código aberto versus código fechado
- Análise estática versus ferramentas de teste dinâmico
- Plataforma/infraestrutura versus aplicativo
- Execução de testes de segurança versus gerenciamento de testes de segurança
- Teste caixa-preta versus teste caixa-branca versus teste caixa-cinza

Cada uma dessas categorias tem suas vantagens e desvantagens quando o STE está selecionando a ferramenta de teste de segurança mais adequada. Este syllabus apresenta três categorias principais:

- Teste caixa-preta versus caixa branca
- Teste de segurança estático versus teste de segurança dinâmico
- Ferramentas de teste de segurança de código aberto versus código fechado

Espera-se que STE crie sua própria biblioteca de ferramentas para seus domínios e contextos. No entanto, eles ainda podem adotar as categorias sugeridas

9.1.1 Ferramentas de teste de segurança caixa-branca

Se o STE tiver acesso no nível do código, as ferramentas de teste de segurança de caixa branca fornecerão a ele conhecimento relacionado ao código, às informações de configuração, às bibliotecas usadas, aos aplicativos, ao sistema e à plataforma, à arquitetura e aos detalhes de login. Um pré-requisito importante é que o STE tenha permissão para realizar a análise, pois ele tentará identificar e avaliar as vulnerabilidades do sistema e dos sistemas integrados.

9.1.2 Ferramentas de teste de segurança caixa-preta

Um pré-requisito para a realização de testes de caixa preta é o acesso a um aplicativo ou sistema em execução em um ambiente semelhante ao de produção. Isso é necessário para que seja possível executar os testes usando ferramentas de teste de segurança de caixa preta, que consideram o SUT como uma caixa-preta e não precisam de nenhum conhecimento interno do software. As ferramentas de teste de segurança caixa-preta concentram-se na identificação de vulnerabilidades durante a execução do aplicativo/sistema.

9.1.3 Ferramentas de teste de segurança caixa-cinza

A realização de testes de segurança de caixa-cinza fornece ao STE algumas informações limitadas sobre os componentes internos do aplicativo/sistema e acesso a uma versão em execução. As ferramentas de teste de segurança caixa-cinza podem ser vistas como uma mistura de ferramentas de teste de segurança caixa-branca e caixa-preta. Elas precisam de algumas informações internas, bem como de um sistema em execução. O foco é identificar vulnerabilidades executando o aplicativo/sistema e executando testes que considerem detalhes internos.

9.1.4 Ferramentas de teste de segurança estática

Uma dimensão importante para a categorização da ferramenta de teste de segurança baseia-se na diferença entre testes de segurança estáticos e testes de segurança dinâmicos. As definições, diferenças e descrições dos testes de segurança estáticos e dinâmicos são discutidas na seção 2.1.2.

Dentro dessa categoria, as ferramentas podem ser consideradas de acordo com seu uso. Isso inclui testes de rede, testes de sistema operacional, testes de banco de dados e testes de aplicativos.

Os testes estáticos de segurança têm muito em comum com os testes de segurança caixa-branca. A principal diferença é que as ferramentas de teste de segurança estática não precisam que o aplicativo ou o sistema esteja

©International Software Testing Qualifications Board

em execução. A ferramenta acessa o código, as bibliotecas ou os arquivos de configuração no escopo e os analisa em relação ao repositório interno da ferramenta de vulnerabilidades conhecidas de sintaxe, semântica ou padrão de código para a linguagem específica em uso.

Há várias maneiras de realizar testes de segurança com ferramentas. No caso das ferramentas de teste de segurança estática, elas incluem, entre outras, SAST e SCA. Elas são explicadas em mais detalhes a seguir

Teste estático de segurança de aplicativos
O SAST (Static Application Security Testing) é uma atividade típica de teste de segurança estática incluída principalmente em um pipeline de Dev(Sec)Ops, conforme discutido no capítulo 6.1 e em [NIST DevSecOps]. Nesses pipelines, ele é executado automaticamente sempre que algum código é alterado e é verificado. O uso do SAST dessa forma fornece feedback imediato. O foco principal do SAST é o aplicativo e, na maioria dos casos, não abrange nenhum componente de plataforma ou infraestrutura. Além disso, essas ferramentas fornecem boas métricas de cobertura de código.

Há uma forte dependência entre o teste de segurança estática e o atributo de ferramenta do software de código aberto, pois as ferramentas de código aberto, por definição, fornecem o código-fonte. Isso significa que o SAST pode ser realizado em todos os sistemas de código aberto. Esse não é o caso dos aplicativos de código fechado. Pode ser possível realizar alguns testes estáticos (p. ex., identificar algumas bibliotecas reutilizadas), mas esse tipo de análise não é possível devido à ofuscação ou a contratos de licença especiais que proíbem análises estáticas

Análise de composição de software
A análise de composição de software (SCA) tem muitos pontos de contato com a segurança. As ferramentas de SCA analisam as vulnerabilidades no código, incluindo as dependências e os componentes de código aberto usados pelo aplicativo. Esses componentes são bem conhecidos e podem ter várias vulnerabilidades. As ferramentas de SCA podem sugerir correções de vulnerabilidades de segurança com base nos componentes identificados. Ao fazer isso, quase todas as ferramentas de SCA usam o banco de dados Common Vulnerabilities and Exposures [CVE21] de vulnerabilidades divulgadas publicamente. (consulte a seção 4.2.2)

9.1.5 Ferramentas de teste de segurança dinâmica

As ferramentas de teste de segurança dinâmica interagem com o SUT enquanto ele está em execução. Os testes de segurança caixa-preta e caixa-cinza estão intimamente ligados aos testes dinâmicos de segurança. As ferramentas podem ser consideradas nas categorizações de DAST e IAST.

Teste dinâmico de segurança de aplicativos
Assim como o SAST, o Dynamic Application Security Testing (DAST) é comumente utilizado em um contexto de DevSecOps [NIST DevSecOps]. A atividade de teste é realizada automaticamente no pipeline usando uma ferramenta de teste de segurança caixa-preta configurável. Ela analisa o aplicativo ou simula um invasor enquanto o software está em execução, procurando vulnerabilidades como validação de entrada, teste de fuzz, autenticação e autorização, configuração e implantação, gerenciamento de sessão, tratamento de erros e criptografia.

A varredura DAST simula diferentes ataques em tempo real no SUT de forma automatizada para identificar quaisquer vulnerabilidades no aplicativo. As técnicas de teste são usadas em um SUT em execução ao realizar testes dinâmicos. Como resultado, esses testes consomem mais tempo e desempenho em comparação com os testes estáticos. Embora o DAST se concentre no aplicativo, as vulnerabilidades identificadas geralmente estão relacionadas aos componentes da infraestrutura necessários para executar o aplicativo.

A DAST não cobre de forma confiável todos os problemas do OWASP Top 10 [OWASP Top 10] ou do SANS CWE Top 25 [CWE21]. Muitas ferramentas podem abranger aspectos específicos de cada uma das classes de vulnerabilidade, mas o uso dessas ferramentas pode gerar uma falsa sensação de segurança

Teste interativo de segurança de aplicativos

O teste interativo de segurança de aplicativos (IAST) é uma abordagem de teste híbrida que aproveita os testes de segurança estáticos e dinâmicos. As ferramentas são usadas para determinar se as vulnerabilidades conhecidas no código-fonte são exploráveis durante o tempo de execução.

Um agente é instalado no ambiente em que o aplicativo está sendo executado, o qual monitora o aplicativo e identifica quaisquer vulnerabilidades no aplicativo enquanto o STE (ou ferramenta de teste de segurança dinâmica) está interagindo com o SUT.

©International Software Testing Qualifications Board

9.1.6 Considerações sobre a seleção de ferramentas de teste de segurança

O STE deve saber qual ferramenta selecionar para cada contexto. Portanto, ele deve ter conhecimento dos diferentes esquemas de categorização e das ferramentas pertencentes a cada categoria.

Os catálogos de ferramentas de teste de segurança podem ajudar a selecionar a ferramenta certa. Alguns exemplos podem ser encontrados em: [KALI], [OWASP], [SANS] e [NIST]. Observe, entretanto, que as ferramentas podem ser categorizadas de forma diferente em alguns desses catálogos e podem estar presentes em um e ausentes em outro

O STE não precisa conhecer e ser capaz de operar todas as ferramentas de teste de segurança disponíveis. Os STEs que estão ativos há mais tempo em um determinado domínio/contexto geralmente criam suas próprias bibliotecas específicas.

Ao selecionar uma ferramenta ou criar uma biblioteca de ferramentas, é recomendável fazer o seguinte:

- Concentre-se no que precisa ser verificado
- Não se torne dependente de um único fornecedor para obter os resultados de teste necessários. Use vários fornecedores para a mesma funcionalidade da ferramenta.
- Examine periodicamente o mercado em busca de novas ferramentas emergentes.

Código aberto vs. código fechado

A diferença entre ferramentas de teste de código aberto e de código fechado (licenciadas) pode ser um aspecto importante da seleção de ferramentas.

Qualquer pessoa pode participar do desenvolvimento de aplicativos ou ferramentas de código aberto. Isso ajuda a eliminar as vulnerabilidades de segurança o mais rápido possível, pelo menos se o projeto de código aberto tiver uma comunidade de desenvolvimento ativa. Além disso, as ferramentas de código aberto podem ser testadas (pelo menos teoricamente), de modo que os backdoors fiquem visíveis no código e possam ser excluídos.

O software de código aberto pode ser personalizado e usado em contextos específicos, o que lhe confere uma clara vantagem

As características a seguir podem ser consideradas desvantagens do uso de ferramentas de código aberto:

- Falta de suporte profissional, especialmente quando nenhuma comunidade ativa oferece suporte a um produto específico. No entanto, algumas organizações são especializadas em fornecer suporte para aplicativos OSS, o que é um aspecto importante para o ambiente corporativo.
- Problemas de licenciamento (p. ex., ao usar as licenças públicas GNU's Not Unix)
- É necessário ter as habilidades de desenvolvimento necessárias disponíveis (p. ex., para se ajustar a contextos específicos)
- Se houver uma vulnerabilidade em um sistema de código aberto, há o risco de que alguém a identifique e crie explorações
- O desenvolvimento futuro dos aplicativos OSS é incerto, pois eles são, em sua maioria, orientados pela comunidade.

Em contraste com os aplicativos OSS, os aplicativos de código fechado usam código proprietário que não está disponível para o usuário. Isso oferece a vantagem de serem oferecidos contratos de suporte pelo fornecedor.

As características a seguir podem ser consideradas desvantagens do uso de software de código fechado:

- As taxas de licença devem ser pagas
- Não há garantia contra backdoors
- Quaisquer vulnerabilidades de segurança podem permanecer desconhecidas por um longo período.
- Uma forte dependência do fornecedor pode ocorrer quando um cliente se torna altamente dependente dos produtos ou serviços de um fornecedor específico, dificultando a mudança para um fornecedor diferente
- Em geral, há apenas uma capacidade limitada de personalizar a ferramenta para um contexto específico (p. ex., o código não pode ser modificado).

©International Software Testing Qualifications Board

9.2 Aplicação de ferramentas de teste de segurança

9.2.1 Compreender o uso e os conceitos das ferramentas de teste de segurança estática

Os aspectos a seguir descrevem algumas das principais características das ferramentas de teste de segurança estática:

- São mais eficazes se houver conhecimento do SUT
- Pode ser aplicado mesmo que o aplicativo em si ainda esteja incompleto e contenha defeitos
- Estão fortemente relacionados a testes caixa-branca
- Adequado para encontrar códigos inseguros ou configurações incorretas no início do SDLC, pois requer apenas informações estáticas do código-fonte
- Abrange o código-fonte e as configurações completas e, portanto, requer acesso de leitura a eles
- Capacidade de ler o objeto fornecido, como o código-fonte de um aplicativo não compilado, e compará-lo com conjuntos de dados predefinidos de práticas recomendadas e vulnerabilidades conhecidas (p. ex., comandos e cadeias não seguros no código-fonte)
- A automação os torna econômicos
- Frequentemente fornecem resultados falso-positivos, pois não estão cientes do contexto. Isso significa que eles não conhecem os casos de uso, a pilha de chamadas e a composição de várias linhas de código. Como resultado, eles podem identificar vulnerabilidades em códigos que nunca serão executados na realidade e, portanto, não têm nenhum impacto negativo sobre a segurança.

Usando, por exemplo, a varredura de rede, a ferramenta SAST avaliaria os arquivos de configuração para encontrar configurações não seguras.

9.2.2 Compreender o uso e os conceitos das ferramentas de teste de segurança dinâmica

A segurança contínua de aplicativos é importante no contexto da abordagem de integração e entrega contínuas encontrada no desenvolvimento ágil de software. A segurança precisa ser verificada contínua e repetidamente para cada incremento. A motivação para usar ferramentas de teste dinâmico é tê-las em execução regularmente e aplicar a automação para garantir que as últimas vulnerabilidades conhecidas sejam imediatamente encontradas e relatadas. O monitoramento e o aprimoramento contínuos da segurança também são chamados de DevSevOps (consulte o capítulo 6 e [NIST DevSecOps]).

As DAST normalmente se concentram nas 10 principais vulnerabilidades da OWASP, como: injeções (p. ex., injeção de SQL, scripts entre sites e injeção de comandos), controles de acesso quebrados, falsificação de solicitações entre sites, condições de corrida, erros de lógica de negócio, vazamentos de memória e vulnerabilidades conhecidas.

Também é importante mencionar que as ferramentas DAST automatizadas examinam apenas um conjunto de vetores de ataque predefinidos. Portanto, outros testes e verificações de segurança ainda são obrigatórios.

Usando o exemplo da varredura de rede, uma ferramenta de teste dinâmico primeiro varre a rede em busca de portas abertas e, em seguida, executa serviços nessas portas.

©International Software Testing Qualifications Board

10 Referências

- [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, accessed on November 7, 2022
- [Bullock2017] Jessey Bullock: “Wireshark for Security Professionals: Using Wireshark and the Metasploit”, Wiley 2017, online available via 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, last accessed on 1. December 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”, online available via 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?”, online available at 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 last accessed on 1. December 2022
- [CLAIR22] https://github.com/quay/clair last accessed on 1. December 2022
- [Cloudflare] https://www.cloudflare.com/learning/security/glossary/what-is-zero-trust/ last accessed on 1. December 2023
- [CVE21] CVE® Program Mission: “Identify, define, and catalog publicly disclosed cybersecurity vulnerabilities.”, online available via cve.org
- [CVSS21] First, Improving Security together: “Common Vulnerability Scoring System SIG”, online available via https://www.first.org/cvss/
- [CWE21] CWE: “Common Weakness Enumeration: A Community-Developed List of Software & Hardware Weakness Types”, online available via https://cwe.mitre.org
- [CWSS21] CWE: “Scoring CWEs: Common Weakness Scoring System CWSS”, online available via https://cwe.mitre.org/cwss/cwss_v1.0.1.html
- [FUZZDB22] https://github.com/fuzzdb-project/fuzzdb last accessed on 1. December 2022
- [Gart21] Gartner Information Technology: “Glossary“, online available via https://www.gartner.com/en/information-technology/glossary
- [GITLAB22] https://docs.gitlab.com/ee/user/application_security/sast/ last accessed on 1. December 2022
- [GTFO22] https://gtfobins.github.io/, accessed on November 13, 2022
- [Huneidy21] Mariangel Huneidy: “The Ultimate Guide to Ethical Hacking “, Sept 21, online available via The Ultimate Guide to Ethical Hacking (0x1.gitlab.io)
- [IETF23] Internet Engineering Task Force: “Introduction to the IETF”, online available via 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”

©International Software Testing Qualifications Board

- [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] ISO International Standards Organization: “Consumers and standards: Partnership for a better world”, online available via 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)
- [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)
- [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”, available via https://www.itgovernance.co.uk/iso27000-family
- [ITGov23b]: “ISO/IEC 27001 – Information Security Management: The international standard for information security”, in IT-Governance, available via https://www.itgovernance.co.uk/iso27001
- [KALI] https://www.kali.org/tools/ last accessed 2. December 2022
- [KUBEAUDIT22] https://github.com/Shopify/kubeaudit, last accessed on 1. December 2022
- [LGPD 13.709] https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm
- [METASPLOIT22] https://www.metasploit.com/ last accessed on 1. December 2022
- [Micro09] Microsoft: “The STRIDE Threat Model”, 12/2009, via The STRIDE Threat Model | Microsoft Docs
- [Micro22] Microsoft: Zero Trust Essentials eBook, via Zero Trust Essentials eBook (microsoft.com)
- [MITRE21] The MITRE Corporation: “Media FAQs”, 2018, via media-FAQs-2018.pdf (mitre.org)
- [NESSUS22] https://tenable.com/products/nessus last accessed on 1. December 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, accessed on November 6, 2022
- [NIST03] NIST Special Publication 800-61, Revision 2, Computer Security Incident Handling Guide
- [NIST05] https://csrc.nist.gov/glossary/term/social_engineering, accessed on November 13, 2022
- [NIST06] NIST SP 800-137, Information Security Continuous Monitoring (ISCM) for Federal Information Systems and Organizations, accessed on September 2011
- [NISTIR 7007]: An Overview of Issues in Test Intrusion Detection Systems
- [NMAP22] https://https://nmap.org/_last accessed on 1. December 2022
- [OPENVAS22] https://www.openvas.org/ last accessed on 1. December 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

©International Software Testing Qualifications Board

- [OWASP Top 10] https://www.owasp.org
- [OWASP] https://owasp.org/www-project-web-security-test-guide/v41/6-Appendix/A-Test_Tools_Resource, last accessed 2. December 2022
- [OWASP11] Ryan Barnett, Dan Cornell, Achim Hoffmann Martin Knobloch “Virtual patching Best Practices”, 2011, shared by OWASP, online available at https://owasp.org/www-community/Virtual_Patching_Best_Practices
- [OWASP21] OWASP: “The Open Web Application Security Project”, via 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, accessed on November 7, 2022
- [RFC4765] The Intrusion Detection Message Exchange Format (IDMEF), Network Working Group
- [SANS] https://www.sans.org/tools/, last accessed 2. December 2022
- [SecJour21] Security Journey: “Why vulnerability list methodologies matter”. Available via 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/, accessed on November 10, 2022
- [Shacklett] Shacklett, M. E., *What is* an attack vector*?* https://www.techtarget.com/searchsecurity/definition/attack-vector
- [SNYK22] https://snyk.io/ last accessed on 1. December 2022
- [SONAR22] https://www.sonarqube.org/, last accessed on 1. December 2022
- [SQLMAP22] https://sqlmap.org last accessed on 1. December 2022
- [SSLSCAN22] https://github.com/rbsec/sslscan last accessed on 1. December 2022
- [SSLYZE22] https://github.com/nabla-c0d3/sslyze last accessed on 1. December 2022
- [Stallings18] Stallings William, Brown Lawrie, 2018, Computer Security: Principles and Practice, ISBN 9781292220611
- [StGrTh20] Michelle P. Steves, Kristen K. Greene and Mary F. Theofanos: “*Categorizing Human* Phishing Detection Difficulty: A Phish Scale”, Journal of Cybersecurity. Published online Sept. 14, 2020. Available via https://academic.oup.com/cybersecurity/article/6/1/tyaa009/5905453
- [SwissCybInst20] Swiss Cyber Institute: “41 Insider Threat Statistics You Should Care About”, Sept 21, online available via 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”, available via https://www.techtarget.com/whatis/definition/The-Three-Ways
- [TELETRUST] https://www.stand-der-technik-security.de/startseite/
- [TERRASCAN22] https://runterrascan.io/, last accessed on 1. December 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, accessed on November 7, 2022
- [HIPAA] https://www.cdc.gov/phlp/publications/topic/hipaa.html, last accessed on 22 August 2023
- [WaCh90] Dolores R. Wallace, John C. Cherniavsky: “*Guide to Software Acceptance*”, NIST Special Publication 500-180, 1990, online available via https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication500-180.pdf

©International Software Testing Qualifications Board

- [WAFEC]: https://owasp.org/www-project-wafec/
- [WIKI01] https://en.wikipedia.org/wiki/Google_hacking, accessed on November 10, 2022
- [WIKI02] https://en.wikipedia.org/wiki/Stuxnet, accessed on November 10, 2022
- [WIRED21] https://www.wired.com/story/solarwinds-hack-supply-chain-threats-improvements/, accessed on November 6, 2022
- [ZAP22] https://www.zaproxy.org/ last accessed on 1. December 2022

©International Software Testing Qualifications Board

Apêndice A - Objetivos de aprendizagem/nível cognitivo de conhecimento

Os seguintes objetivos de aprendizagem são definidos como aplicáveis a este syllabus. Cada tópico do syllabus será examinado de acordo com seu objetivo de aprendizado.

Os objetivos de aprendizagem começam com um verbo de ação correspondente ao seu nível cognitivo de conhecimento, conforme listado abaixo.

Nível 1: Lembrar (K1)

O candidato se lembrará, reconhecerá e recordará um termo ou conceito.

Verbos de ação: Recordar, reconhecer.

Exemplos
Relembre os conceitos da pirâmide de teste.
Reconhecer os objetivos típicos do teste.

Nível 2: Compreender (K2)

O candidato pode selecionar as razões ou explicações para declarações relacionadas ao tópico e pode resumir, comparar, classificar e dar exemplos para o conceito do teste.

Verbos de ação: Classificar, comparar, diferenciar, distinguir, explicar, dar exemplos, interpretar, resumir

Exemplos	Notas
Classificar as ferramentas de teste de acordo com sua finalidade e as atividades de teste que elas suportam.	
Compare os diferentes níveis de teste.	Pode ser usado para procurar semelhanças, diferenças ou ambos.
Diferencie teste de depuração.	Procura diferenças entre os conceitos.
Distinguir entre riscos do projeto e do produto.	Permite que dois (ou mais) conceitos sejam classificados separadamente.
Explicar o impacto do contexto no processo de teste.	
Dê exemplos de por que o teste é necessário.	
Inferir a causa raiz dos defeitos a partir de um determinado perfil de falhas.	
Resumir as atividades do processo de revisão do produto de trabalho.	

Nível 3: Aplicar (K3)

O candidato pode executar um procedimento quando confrontado com uma tarefa familiar ou selecionar o procedimento correto e aplicá-lo a um determinado contexto.

Verbos de ação: Aplicar, implementar, preparar, usar

Exemplos	Notas
Aplicar a análise de valor limite para derivar casos de teste a partir de determinados requisitos.	Deve se referir a um procedimento / técnica / processo etc.
Implementar métodos de coleta de métricas para dar suporte aos requisitos técnicos e gerenciais.	
Prepare testes de instalação para aplicativos móveis.	
Use a rastreabilidade para monitorar o progresso do teste quanto à integridade e à consistência com os objetivos, a estratégia e o plano de teste.	Pode ser usado em um LO que deseja que o candidato seja capaz de usar uma técnica ou um procedimento. Semelhante a "aplicar".

©International Software Testing Qualifications Board

Nível 4: Analisar (K4)

O candidato pode separar as informações relacionadas a um procedimento ou técnica em suas partes constituintes para melhor compreensão e pode distinguir entre fatos e inferências. Uma aplicação típica é analisar um documento, software ou situação de projeto e propor ações apropriadas para resolver um problema ou tarefa.

Verbos de ação: Analisar, desconstruir, delinear, priorizar, selecionar.

Exemplos	Notas
Analisar uma determinada situação de projeto para determinar quais técnicas de teste caixa-preta ou baseadas em experiência devem ser aplicadas para atingir objetivos específicos.	Examinável somente em combinação com um objetivo mensurável da análise. Deve ser do tipo 'Analyze xxxx to xxxx' (ou similar).
Priorizar os casos de teste em um determinado conjunto de testes para execução com base nos riscos relacionados ao produto.	
Selecionar os níveis de teste e os tipos de teste adequados para verificar um determinado conjunto de requisitos.	Necessário quando a seleção requer análise.

©International Software Testing Qualifications Board

Apêndice B - Referência

(Para os níveis cognitivos dos objetivos de aprendizagem)

Anderson, L. W. e Krathwohl, D. R. (eds) (2001) A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives, Allyn & Bacon B - Business Outcomes traceability matrix with Learning Objectives.

A tabela abaixo mostra os Resultados de Negócios do syllabus e o número de Objetivos de Aprendizagem

Resultados de negócio: Engenheiro de testes de segurança		**Número de LOs**
ID	**Descrição**	
STE-BO1	Compreender os paradigmas fundamentais de segurança e seu impacto nos testes de segurança	6
STE-BO2	Usar e aplicar técnicas adequadas de teste de segurança e conhecer seus pontos fortes e limitações	7
STE-BO3	Contribuir para o planejamento, a criação e a execução do teste de segurança	5
STE-BO4	Entender como os padrões e as práticas recomendadas de segurança do Teste de Segurança podem ser utilizados para o Teste de Segurança	4
STE-BO5	Ajustar e executar atividades de teste de segurança de acordo com o contexto específico da organização	3
STE-BO6	Ajustar e executar atividades de teste de segurança de acordo com métodos de desenvolvimento específicos e ciclos de vida de desenvolvimento de software	4
STE-BO7	Alimentar os resultados dos testes de segurança em um sistema de gerenciamento de segurança da informação (ISMS) para um gerenciamento ativo dos riscos de segurança	4
STE-BO8	Coletar, avaliar e agregar resultados de testes, escrever um relatório de teste detalhado com todas as evidências e descobertas	3
STE-BO9	Com base em uma abordagem de teste de segurança necessária, identificar os requisitos adequados para ferramentas e auxiliar na seleção de ferramentas de teste de segurança	3

©International Software Testing Qualifications Board

A tabela a seguir mostra a rastreabilidade entre os Resultados do Negócio e os Objetivos de Aprendizagem:

			Resultados de Negócio								
# LO	**Objetivo de aprendizagem**	**Nível K**	**STE-BO1**	**STE-BO2**	**STE-BO3**	**STE-BO4**	**STE-BO5**	**STE-BO6**	**STE-BO7**	**STE-BO8**	**STE-BO9**
STE-1.1.1	Explicar os diferentes níveis de segurança dos ativos e seu nível de proteção correspondente	K2	x								
STE-1.1.2	Explicar a relação entre a sensibilidade das informações e os testes de segurança	K2	x								
STE-1.2.1	Descrever a função dos testes de segurança no contexto das auditorias de segurança	K2	x								
STE-1.3.1	Explicar o conceito de confiança zero	K2	x								
STE-1.3.2	Aplicar confiança zero nos testes de segurança	K3	x								
STE-1.4.1	Exemplificar o conceito de reutilização de software de código aberto no desenvolvimento de software e seus impactos nos testes de segurança	K2	x								
STE-2.1.1	Dar exemplos de tipos de testes de segurança de acordo com um contexto de segurança de caixa preta, caixa branca e caixa cinza	K2		x							
STE-2.1.2	Dar exemplos de tipos de testes de segurança de acordo com os testes de segurança estáticos ou dinâmicos	K2		x							
STE-2.2.1	Aplicar casos de teste de segurança, com base em uma determinada abordagem de teste de segurança, juntamente com os riscos de segurança funcionais e estruturais identificados	K3		x							
STE-2.2.2	Descrever como testar a reconciliação e a recertificação de identidades e permissões	K2		x							
STE-2.2.3	Descrever como testar o controle do gerenciamento de identidade e acesso	K2		x							
STE-2.2.4	Descrever como testar o controle de proteção de dados	K2		x							
STE-2.2.5	Descrever como testar a tecnologia de proteção	K2		x							
STE-3.1.1	Explicar as diferentes atividades, tarefas e responsabilidades em um processo de teste de segurança	K2			x						

©International Software Testing Qualifications Board

# LO	Objetivo de aprendizagem	Nível K	STE-BO1	STE-BO2	STE-BO3	STE-BO4	STE-BO5	STE-BO6	STE-BO7	STE-BO8	STE-BO9
STE-3.1.2	Compreender os principais elementos e características de um ambiente de teste de segurança eficaz	K2			x						
STE-3.2.1	Dar exemplos de testes de segurança no nível de teste de componentes com base em uma determinada base de código	K2			x						
STE-3.2.2	Dar exemplos de testes de segurança no nível de integração de componentes com base em uma determinada especificação de projeto	K2			x						
STE-3.2.3	Implementar um teste de segurança de ponta a ponta que valide um ou mais requisitos de segurança relacionados a um ou mais processos de negócios	K3			x						
STE-4.1.1	Explicar diferentes fontes de padrões de teste e práticas recomendadas e sua aplicabilidade	K3				x					
STE-4.2.1	Aplicar o conceito do Open Web Application Security Project, Common Vulnerability Enumeration, Common Weakness Enumeration, Common Vulnerability Scoring System e Common Weakness Scoring System e como aproveitá-los para testes de segurança	K3				x					
STE-4.3.1	Explicar os prós e contras dos oráculos de teste usados para testes de segurança	K2				x					
STE-4.3.2	Entender os prós e contras do uso dos melhores padrões e práticas recomendadas de segurança	K3				x					
STE-5.1.1	Analisar um determinado contexto organizacional e determinar quais aspectos específicos devem ser considerados nos testes de segurança.	K3					x				
STE-5.2.1	Analisar o impacto das regulamentações governamentais nas políticas de segurança.	K3					x				
STE-5.3.1	Analisar um cenário de ataque (ataque realizado e descoberto) e identificar possíveis fontes e motivação do ataque.	K4					x				
STE-6.1.1	Resumir por que as atividades de teste de segurança devem abranger o ciclo de vida de desenvolvimento de software	K2						x			
STE-6.1.2	Analisar como as atividades de teste de segurança são afetadas por diferentes modelos de ciclo de vida de desenvolvimento de sistemas	K4						x			

©International Software Testing Qualifications Board

# LO	Objetivo de aprendizagem	Nível K	STE-BO1	STE-BO2	STE-BO3	STE-BO4	STE-BO5	STE-BO6	STE-BO7	STE-BO8	STE-BO9
STE-6.2.1	Definir e executar testes de regressão de segurança e testes de confirmação com base em uma alteração em um sistema	K3						x			
STE-6.2.2	Analisar os resultados dos testes de segurança para determinar a natureza de uma vulnerabilidade de segurança e seu possível impacto técnico	K2						x			
STE-7.1.1	Compreender os critérios de aceite dos testes de segurança e como eles influenciam a seleção de abordagens e técnicas de testes de segurança	K2							x		
STE-7.2.1	Compreender a função dos testes de segurança para um sistema eficaz de gerenciamento de segurança da informação	K2							x		
STE-7.3.1	Avaliar a maturidade do ISMS, introduzindo diferentes abordagens de teste, novos objetos de teste ou cobertura aprimorada	K3							x		
STE-7.3.2	Compreender a mensurabilidade em um ISMS	K2									
STE-8.1.1	Compreender a importância dos resultados dos testes de segurança e como isso afeta seu manuseio e comunicação	K2								x	
STE-8.2.1	Avaliar os resultados de um determinado teste de segurança para identificar vulnerabilidades de segurança	K3								x	
STE-8.3.1	Avaliar diferentes técnicas para eliminar as vulnerabilidades identificadas	K3								x	
STE-9.1.1	Analisar diferentes casos de uso e aplicar categorizações para ferramentas de teste de segurança	K3									x
STE-9.2.1	Compreender o uso e os conceitos das ferramentas de teste de segurança dinâmica	K2									x
STE-9.2.2	Compreender o uso e os conceitos das ferramentas de teste de segurança estática	K2									x

©International Software Testing Qualifications Board

Apêndice C - Notas de versão

O ISTQB® Security Test Engineer é uma nova versão. Por esse motivo, não há notas de versão detalhadas por capítulo e seção.

©International Software Testing Qualifications Board

Apêndice D - Termos específicos do domínio de teste de segurança

Nome do termo	Definição
autenticação	Um procedimento que determina se uma pessoa ou um processo é, de fato, quem ou o que se declara ser.
autorização	Permissão dada a um usuário ou processo para acessar recursos.
criptografia	O processo de codificação de informações para que somente as partes autorizadas possam recuperar as informações originais, geralmente por meio de uma chave ou processo de descriptografia específico.
firewall	Um componente ou conjunto de componentes que controla o tráfego de entrada e saída da rede com base em regras de segurança predeterminadas.
information security management system (ISMS)	Uma parte de um sistema de gerenciamento geral, com base em uma abordagem de risco de negócio, para estabelecer, implementar, operar, monitorar, revisar, manter e aprimorar a segurança das informações.
sensibilidade das informações	Uma medida da importância de proteger as informações atribuída por seu proprietário (segundo o NIST).
software de código aberto	Software que pode ser acessado, usado, modificado e compartilhado por qualquer pessoa.
Open Web Application Security Project (OWASP)	O OWASP, Open Web Application Security Project, lista os pontos fracos comuns e é bastante famoso por publicar sua classificação OWASP Top 10.
escalonamento de privilégios	A exploração de um bug ou falha que permite um nível de privilégio mais alto do que o que normalmente seria permitido (segundo o NIST)
rootkit	Um conjunto de ferramentas usadas por um invasor para obter e manter o acesso em nível de raiz a um host para ocultar as atividades do invasor por meios secretos. (Após o NIST)
política de segurança	Um documento de alto nível que descreve os princípios, a abordagem e os principais objetivos da organização com relação à segurança.
serviço de segurança	Um recurso que oferece suporte a um ou vários objetivos de segurança. (Após o NIST)
engenharia social	Uma tentativa de enganar alguém para que revele informações (p. ex., uma senha) que possam ser usadas para atacar sistemas ou redes.
software composition analysis (SCA)	Uma prática para identificar componentes de código aberto e de código fechado em uso em um aplicativo, suas vulnerabilidades de segurança conhecidas e restrições de licença adversárias (After NIST)
STRIDE	Um acrônimo para seis categorias de ameaças (ou seja, *spoofing* (falsificação), *tampering* (adulteração), *repudiation* (repúdio), *information disclosure* (divulgação de informações), *denial of service* (negação de serviço) e *elevation of privilege* (elevação de privilégio) usado para modelar possíveis ameaças a um sistema.
confiança zero	Um modelo projetado para minimizar a incerteza na aplicação de decisões de acesso precisas e menos privilegiadas por solicitação em um componente, sistema e serviços se uma rede for considerada comprometida. (Glossário do NIST)

©International Software Testing Qualifications Board

Glossário

Todos os termos estão definidos no ISTQB Glossary em https://glossary.istqb.org

©International Software Testing Qualifications Board