CIBERSEGURANÇA
E
HACKING ÉTICO

Versão 1.0
Data 13/10/2025
SérgioRicardo de Souza Rezende

Sumário

Conceitos Básicos de SI, Cibersegurança e Hacking Ético

Os princípios básicos que sustentam a Segurança da Informação são; confidencialidade, integridade, disponibilidade, autenticidade e legalidade.

- **Confidencialidade:** garantir que somente pessoas autorizadas possuam acesso às informações.
- **Integridade:** garantir que a informação não será manipulada ou alterada por pessoas não autorizadas.
- **Disponibilidade:** garantir que a informação esteja sempre acessível quando necessária.
- **Autenticidade:** garantir a identificação incontestável da informação e o não repúdio pela identidade.
- **Legalidade:** garantir o cumprimento das leis vigentes relacionadas à proteção de dados pessoais.

Cibersegurança é o conjunto de processos, ferramentas e melhores práticas usados para proteger sistemas, redes e dados contra ataques digitais, roubo de informação, e outras ameaças cibernéticas.

Os Principais Tópicos de Cibersegurança:

- Segurança de Redes
- Segurança de Aplicações
- Segurança de Dados
- Segurança Física
- Gerenciamento de Acesso e Autenticação
- Análise de Intrusão e Resposta a Incidentes
- Criptografia e Proteção de Dados
- Gestão de Vulnerabilidades
- Auditoria e Testes de Segurança
- Conformidade e Regulamentação
- Conscientização de usuários

Cada um desses tópicos requer conhecimento especializado para implementar medidas eficazes de proteção contra ameaças cibernéticas.

Cibersegurança segundo profissionais de segurança da ISACA (Information Systems Audit and Control Association) é a proteção dos ativos da informação, por meio do tratamento das ameaças que trazem riscos ao processamento, armazenamento e transmissão digital de informações entre sistemas computacionais.

Segundo a norma ISO 27001, um ativo de informação é definido por qualquer elemento que sustente um ou mais processos de negócios, ou seja, tudo aquilo que tenha valor para uma organização.

Cibersegurança é um termo que abrange um conjunto de estratégias e técnicas de segurança que são recomendadas para proteger e mitigar diversos tipos de ameaças cibernéticas. A cibersegurança é focada em manter protegidos os recursos de TI, como por exemplo, sistemas operacionais, serviços de rede, aplicativos, dispositivos de armazenamento de dados, computadores, servidores, além de outros dispositivos encontrados na infraestrutura de redes de computadores de uma organização.

A principal tarefa da cibersegurança é garantir proteção contra as principais ameaças e vulnerabilidades que podem vir a prejudicar a segurança dos ativos e da infraestrutura de TI.

Técnicas Essenciais de Cibersegurança:

- **Proteger dispositivos de hardware:** atualizar firmwares de dispositivos, utilizar senhas de BIOS de computadores, utilizar senhas fortes de acesso para interfaces de gerenciamento de roteadores e pontos de acesso Wi-Fi.

- **Proteger sistemas operacionais:** utilizar senhas fortes de acesso ao sistema, atribuir contas de usuários com privilégios mínimos, aplicar atualizações de patchs de segurança, habilitar firewall baseado em host, habilitar sistema anti-malware, executar somente serviços necessários, realizar hardening de SO.

- **Proteger aplicações:** utilizar senhas fortes de acesso, aplicar atualizações de patchs e versões, configurações de segurança, utilizar fator duplo de autenticação, aplicar o conceito de Segurança por Design no desenvolvimento de aplicativos.

- **Proteger perímetro de rede:** implantar firewalls de borda, sistemas de detecção de intrusão IDS/IPS, proxy, VPN, segmentação de redes, protocolos robustos de segurança para redes Wi-Fi.

Hacking Ético

O Hacking Ético é o uso de técnicas de invasão de sistemas com objetivo de descobrir, entender e corrigir vulnerabilidades de segurança em uma rede ou sistema de computador de forma legalizada.

Red Team e Blue Team são duas equipes distintas de profissionais de cibersegurança que trabalham juntas para garantir a segurança da informação em uma organização.

O Red Team é a equipe responsável por simular ataques cibernéticos contra a organização, com o objetivo de identificar vulnerabilidades e testar a eficácia dos sistemas de defesa. Eles utilizam técnicas de hacking ético para simular ataques de hackers mal-intencionados, bem como para testar a capacidade de detecção e resposta da organização.

O Blue Team é a equipe responsável por defender a organização contra ataques cibernéticos, usando ferramentas e técnicas de detecção e prevenção de ameaças. Eles monitoram constantemente as redes e sistemas da organização, procurando por sinais de ataques e respondendo rapidamente a incidentes de segurança.

A colaboração entre Red Team e Blue Team é essencial para garantir a segurança da informação. Enquanto o Red Team simula ataques, o Blue Team trabalha para detectar e responder a esses ataques, bem como para melhorar os sistemas de defesa da organização. Essa colaboração ajuda a identificar vulnerabilidades e a implementar medidas preventivas para proteger a organização contra ataques cibernéticos.
Em resumo, Red Team e Blue Team são duas equipes fundamentais para a segurança da informação, trabalhando juntas para identificar e mitigar ameaças cibernéticas.

https://www.hackingisnotacrime.org

Ferramentas e Técnicas Utilizadas para Hacking Ético

Ambas as equipes utilizam uma variedade de ferramentas e técnicas para realizar suas tarefas.
Alguns exemplos incluem:

- **Ferramentas de reconhecimento:** Shodan, ReconNG, Maltego, DNSRecon, Whatweb
- **Ferramentas de varredura (Scanning):** Nmap, Masscan, Dirb, Gobuster
- **Ferramentas de identificação de vulnerabilidades:** Nessus, OpenVAS, Nikto, BurpSuite, ZAP, Wpscan, SQLmap
- **Ferramentas de simulação de ataques:** Metasploit, Cobalt Strike, Empire
- **Ferramentas de quebra de senhas:** John The Ripper, Hashcat, Hydra, Ophcrack, Rainbow Tables
- **Ferramentas de engenharia social:** SET, Gopish
- **Ferramentas de detecção de ameaças:** SIEM (Security Information and Event Management), EDR (Endpoint Detection and Response), IDS/IPS (Intrusion Detection/Prevention System), DLP (Data Loss/Leak Prevention)

Engenharia Social

Engenharia Social é uma técnica utilizada para manipular indivíduos ou grupos de pessoas, com o objetivo de obter informações confidenciais ou acesso a sistemas e dados sensíveis. Essa técnica é baseada no uso de manipulação psicológica, persuasão e enganação, visando induzir as vítimas a revelar senhas, compartilhar informações confidenciais ou executar ações específicas que comprometam a segurança da organização.

A Engenharia Social é frequentemente utilizada em ataques cibernéticos, como phishing, spear-phishing e whaling, onde o atacante tenta enganar a vítima para que ela revele informações confidenciais ou execute ações que comprometam a segurança da organização.

Alguns exemplos de técnicas de Engenharia Social incluem:

Phishing: enviar emails falsos ou mensagens instantâneas que parecem ser de fontes confiáveis, com o objetivo de induzir a vítima a clicar em links ou baixar arquivos maliciosos.
Spear-Phishing: enviar emails personalizados para indivíduos específicos, utilizando informações confidenciais ou pessoais para aumentar a credibilidade do ataque.

Whaling: enviar emails ou mensagens dirigidas a indivíduos de alta posição na organização, como CEOs ou diretores, com o objetivo de obter informações confidenciais ou acesso a sistemas sensíveis.
Pretexting: criar situações fictícias ou pretextos para induzir a vítima a revelar informações confidenciais ou executar ações específicas.

Para se proteger contra ataques de Engenharia Social, é importante:

- Educar funcionários sobre as técnicas utilizadas pelos atacantes.
- Implementar políticas e procedimentos para verificar a autenticidade de e-mails e mensagens.
- Utilizar ferramentas de detecção de phishing e outras ameaças cibernéticas.
- Realizar treinamentos regulares para funcionário, visando aumentar a conscientização sobre as ameaças cibernéticas e a importância da segurança da informação.

Em resumo, a Engenharia Social é uma técnica poderosa utilizada em ataques cibernéticos, que exige muita atenção e cuidado para se proteger contra esse tipo de ameaça.

Certificações e Frameworks de Cibersegurança e Pentest

Existem várias certificações de hacking ético disponíveis para profissionais da área de cibersegurança. Algumas das mais conhecidas incluem:

- **Certified in Cybersecurity (CC)** - Primeiro passo para uma carreira em cibersegurança da ISC2, a principal organização profissional de segurança cibernética do mundo, conhecida pelo CISSP. https://www.isc2.org/certifications/cc

- **Certified Information Systems Security Professional (CISSP)** - Embora não seja exclusivamente focada em hacking ético, essa certificação abrange um conjunto amplo de habilidades em segurança da informação, incluindo técnicas de hacking ético. https://www.isc2.org/certifications/cissp

- **Certified Ethical Hacker (CEH)** - Oferecida pela EC-Council, essa certificação é uma das mais reconhecidas e abrange uma ampla gama de habilidades em hacking ético. https://www.eccouncil.org/train-certify/certified-ethical-hacker-ceh

- **Offensive Security Certified Professional (OSCP)** - Uma certificação que testa habilidades práticas em hacking ético, exigindo que os candidatos solucionem desafios reais de segurança. https://www.offsec.com/courses/pen-200

- **Certificate Cisco in Ethical Hacker** - Este certificado é um passo essencial para se especializar em funções de segurança ofensivas, aprimorando ainda mais seus conhecimentos e credenciais. https://www.netacad.com/courses/ethical-hacker?courseLang=pt-BR

Essas certificações ajudam profissionais da área de cibersegurança a demonstrar suas habilidades e competências em hacking ético, bem como em outras áreas relacionadas à segurança da informação.

Frameworks de Cibersegurança

O **NIST** (National Institute of Standards and Technology) desenvolve padrões, diretrizes, melhores práticas e recursos de segurança cibernética e privacidade para atender às necessidades da indústria, agências federais e do público em geral dos EUA.
https://www.nist.gov/cyberframework

O **CIS** (Center for Internet Security) é uma organização independente e sem fins lucrativos cuja missão é criar confiança no mundo conectado. Na CIS são disponibilizados vários guias que compartilham o poder da comunidade global de TI para proteger organizações públicas e privadas contra ameaças cibernéticas.

https://www.cisecurity.org

O **NCSC** (National Cyber Secure Center) recomenda o Cyber Essentials como o padrão mínimo de segurança cibernética para todas as organizações. O Cyber Essentials é um programa de certificação apoiado pelo governo do Reino Unido que ajuda a manter os dados da sua organização e dos seus clientes protegidos contra ataques cibernéticos.

https://www.ncsc.gov.uk

A **GCA** (Global Cyber Alliance) é uma organização internacional sem fins lucrativos que mobiliza ações coletivas para enfrentar os maiores desafios da Internet e construir um mundo digital mais seguro para todos.

https://globalcyberalliance.org

O **CERT.br** é um Time de Resposta a Incidentes de Segurança (CSIRT) de Responsabilidade Nacional de último recurso, mantido pelo NIC.br. O centro presta serviços da área de Gestão de Incidentes de Segurança da Informação para qualquer rede que utilize recursos administrados pelo NIC.br.

https://www.cert.br

Frameworks de Hacking Ético

O **“Penetration Testing Execution Standard”** consiste em sete (7) seções principais. Elas abrangem tudo relacionado a um pentest – desde a comunicação inicial e o raciocínio por trás de um pentest, passando pelas fases de coleta de inteligência e modelagem de ameaças, nas quais os testadores trabalham nos bastidores para obter um melhor entendimento da organização testada, passando pela pesquisa de vulnerabilidades, exploração e pós-exploração, onde a expertise técnica em segurança dos testadores entra em jogo e se combina com a compreensão do negócio do projeto, e, finalmente, até o relatório, que captura todo o processo de uma maneira que faça sentido para o cliente e agregue o máximo valor a ele. http://www.pentest-standard.org

OSINT é a sigla para **"Open Source Intelligence"**, que significa Inteligência de Fontes Abertas em português. Trata-se de uma técnica de coleta de informações disponíveis publicamente, através de fontes como sites, redes sociais, documentos públicos, entre outros. O objetivo do OSINT é reunir dados relevantes sobre uma pessoa, empresa ou organização, sem necessariamente ter acesso a sistemas ou bases de dados restritos.

O OSINT é amplamente utilizado em investigações de segurança da informação, bem como em análises de risco e prevenção de ameaças cibernéticas. Pode ser realizado manualmente ou através de ferramentas automatizadas que buscam e processam grandes volumes de dados.

Em termos de aplicação prática, o OSINT pode ser usado para identificar vulnerabilidades em sistemas, detectar atividades suspeitas de hackers ou terroristas, bem como para monitorar a reputação de marcas e empresas.

Exemplos de ferramentas de OSINT incluem o Shodan, que permite buscar dispositivos conectados à internet; o Maltego, que ajuda a visualizar conexões entre diferentes tipos de dados; e o theHarvester, que coleta informações de e-mails, subdomínios e hosts associados a um domínio específico.

Em resumo, o OSINT é uma técnica fundamental para a coleta de informações em cibersegurança, permitindo que profissionais identifiquem potenciais ameaças e tomem medidas preventivas para proteger sistemas e dados sensíveis.

https://osintframework.com

MITRE ATT&CK é uma base de conhecimento globalmente acessível sobre táticas e técnicas adversárias, baseada em observações do mundo real. A base de conhecimento ATT&CK é usada como base para o desenvolvimento de modelos e metodologias de ameaças específicas no setor privado, no governo e na comunidade de produtos e serviços de segurança cibernética.

https://attack.mitre.org

MITRE D3FEND é uma base de conhecimento de contramedidas de segurança cibernética. Várias fontes de literatura sobre pesquisa e desenvolvimento, incluindo uma amostra específica de mais de 500 patentes de contramedidas dos EUA.

https://d3fend.mitre.org

Cyber Kill Chain desenvolvido pela Lockheed Martin, o framework faz parte do modelo Intelligence Driven Defense para identificação e prevenção de atividades de intrusão cibernética. O modelo identifica o que os adversários devem realizar para atingir seus objetivos.

As sete etapas do Cyber Kill Chain® aumentam a visibilidade de um ataque e enriquecem a compreensão do analista sobre as táticas, técnicas e procedimentos do adversário.

https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html

IA, Distribuições Linux e Ferramentas de Pentest

Deep Hat é um LLM "Large Language Model" de segurança cibernética de código aberto desenvolvido para solucionar desafios do mundo real. Enquanto a maioria dos modelos é treinada em código genérico ou benchmarks sintéticos, o Deep Hat se concentra em dados de cibersegurança ricos e reais para impulsionar fluxos de trabalho autônomos ofensivos e defensivos.

https://www.deephat.ai

Parrot Security oferece um vasto arsenal de ferramentas, utilitários e bibliotecas que profissionais de TI e cibersegurança podem usar para testar e avaliar a segurança de seus ativos de forma confiável, compatível e reproduzível. Da coleta de informações ao relatório final, o sistema Parrot oferece o ambiente mais flexível possível.

https://parrotsec.org

Black Arch Linux é uma distribuição de testes de penetração baseada no Arch Linux para testadores de penetração e pesquisadores de segurança. O repositório contém 2.860 ferramentas. Você pode instalar ferramentas individualmente ou em grupos. O BlackArch Linux é compatível com instalações existentes do Arch.

https://blackarch.org

Kali Linux é uma distribuição Linux de código aberto, baseada em Debian, voltada para diversas tarefas de segurança da informação, como testes de penetração, pesquisa de segurança, computação forense e engenharia reversa.

https://www.kali.org

Aqui está a lista das principais ferramentas de hacking ético no Kali Linux:

- **Nmap:** Ferramenta de varredura de rede que permite identificar hosts ativos e serviços disponíveis em redes locais e Internet.
- **DirBuster/GoBuster:** Ferramentas de brute force de diretórios que permite testar e explorar diretórios em servidores web.
- **Recon-ng/Maltego:** Ferramenta de reconhecimento de informações OSINT que permite coletar e analisar informações sobre alvos.
- **Nessus/OpenVAS:** Ferramentas para scanner de vulnerabilidades utilizado para detectar falhas de segurança em redes e sistemas.
- **Metasploit:** Ferramenta de exploração de vulnerabilidades que permite testar e explorar vulnerabilidades descobertas em sistemas de forma automatizada.
- **Burp Suite/ZedAttackProxy:** Ferramentas de interceptação de tráfego web que permite testar e explorar vulnerabilidades em aplicações web.
- **Wireshark:** Ferramenta de análise de tráfego de rede que permite capturar e analisar pacotes de dados.

- **John the Ripper/Hashcat:** Ferramentas de cracking de senhas que permite testar e quebrar senhas em sistemas.
- **Aircrack-ng:** Ferramenta de cracking de senhas WiFi que permite testar e quebrar senhas em redes sem fio.
- **SQLmap:** Ferramenta de exploração de vulnerabilidades SQL que permite testar e explorar vulnerabilidades em bancos de dados.
- **Hydra:** Ferramenta de cracking de senhas que permite testar e quebrar senhas em serviços e sistemas web.

Essas ferramentas são amplamente utilizadas por profissionais de hacking ético para testar e explorar vulnerabilidades em sistemas e aplicações. É importante lembrar que o uso dessas ferramentas deve ser feito de forma ética e responsável, sempre com autorização prévia do proprietário do sistema ou aplicação sendo testada.

https://www.kali.org/tools

OWASP Top 10

As empresas devem adotar este documento e iniciar o processo para garantir que suas aplicações web minimizem riscos de segurança. Utilizar o OWASP Top 10 é provavelmente o primeiro passo mais eficaz para mudar a cultura de desenvolvimento de software em sua organização para produzir código mais seguro.

https://owasp.org/www-project-top-ten

1° Quebra de Controle de Acesso

Descrição:

O controle de acesso impõe a política de modo que os usuários não possam agir fora de suas permissões pretendidas. As falhas normalmente levam à divulgação, modificação ou destruição de informações não autorizadas de todos os dados ou ao desempenho de uma função comercial fora dos limites do usuário. Vulnerabilidades comuns de controle de acesso incluem:

- Violação do princípio de privilégio mínimo ou negação por padrão, onde o acesso deve ser concedido apenas para determinados recursos, funções ou usuários, mas está disponível para qualquer pessoa.

- Ignorar verificações de controle de acesso modificando a URL (adulteração de parâmetros ou navegação forçada), o estado interno do aplicativo, a página HTML ou usando uma ferramenta de ataque que modifica as requisições de API.

- Permitir a visualização ou edição da conta de outrem, mediante a disponibilização do seu identificador único (referências diretas não seguras a objetos).

- Acessando API sem controles de acesso para POST, PUT e DELETE.

- Elevação de privilégio. Agir como um usuário sem estar logado ou agir como um administrador quando logado como um usuário.

- Manipulação de metadados, como reproduzir ou adulterar um token de controle de acesso JSON Web Token (JWT), um cookie ou campo oculto manipulado para elevar privilégios ou abusar da invalidação de JWT.

- A configuração incorreta do CORS permite o acesso à API de origens não autorizadas / não confiáveis.

- Força a navegação para páginas autenticadas como um usuário não autenticado ou para páginas privilegiadas como um usuário padrão.

https://owasp.org/Top10/pt-BR/A01_2021-Broken_Access_Control/

2° Falhas Criptográficas

Descrição:

A primeira coisa é determinar as necessidades de proteção dos dados em trânsito e armazenados. Por exemplo, senhas, número de cartão de crédito, registros de saúde, informações pessoas e segredos de negócios que requerem proteção extra, principalmente se esses dados se enquadrarem nas leis de privacidade, alguns exemplos são a da Europa General Data Protection Regulation (GDPR), no Brasil Lei Geral de Proteção de Dados (LGPD) ou regulamentos de proteção de dados financeiros, como PCI Data Security Standard (PCI DSS). Para todos esses dados:

- Todos os dados são transmitidos em texto não criptografado? Isso diz respeito a protocolos como HTTP, SMTP, FTP também usando atualizações TLS como STARTTLS. O tráfego externo da Internet é perigoso. Verifique todo o tráfego interno, por exemplo, entre balanceadores de carga, servidores da web ou sistemas back-end.

- Algum algoritmo ou protocolo criptográfico antigo ou fraco é usado por padrão ou em código mais antigo?

- As chaves criptográficas padrão em uso, são chaves criptográficas geradas fracas ou reutilizadas, faltando o gerenciamento ou rotação de chaves adequado? As chaves criptográficas são verificadas nos repositórios de código-fonte?

- A criptografia não é aplicada, por exemplo, há alguma diretiva de segurança de cabeçalhos HTTP (navegador) ou cabeçalhos ausentes?

- O certificado do servidor recebido e a cadeia de confiança estão devidamente validados?

- Os vetores de inicialização são ignorados, reutilizados ou não gerados suficientemente seguros para o modo criptográfico de operação? Está em uso um modo de operação inseguro, como o ECB? A criptografia é usada quando a criptografia autenticada é a mais apropriada?

- As senhas estão sendo usadas como chaves criptográficas na ausência de uma função de derivação de chave de base de senha?

- A aleatoriedade é usada para fins criptográficos que não foram projetados para atender aos requisitos criptográficos? Mesmo se a função correta for escolhida, ela precisa ser propagada pelo desenvolvedor e, se não, o desenvolvedor sobrescreveu a forte funcionalidade de propagação incorporada a ela com uma semente que carece de entropia/imprevisibilidade suficiente?

- Estão em uso funções hash obsoletas, como MD5 ou SHA1, ou funções hash não criptográficas usadas quando funções hash criptográficas são necessárias?

- Estão em uso métodos de preenchimento criptográfico obsoletos, como PKCS número 1 v1.5?

- As mensagens de erro criptográficas ou as informações do canal lateral podem ser exploradas, por exemplo, na forma de ataques oracle de preenchimento?

https://owasp.org/Top10/pt-BR/A02_2021-Cryptographic_Failures/

3° Injeção

Descrição:

Uma aplicação é vulnerável a ataques quando:

- Os dados fornecidos pelo usuário não são validados, filtrados ou higienizados pelo aplicativo.

- Consultas dinâmicas ou chamadas não parametrizadas sem escape ciente do contexto são usadas diretamente no interpretador.

- Dados hostis são usados nos parâmetros de pesquisa de mapeamento relacional de objeto (ORM) para extrair registros confidenciais adicionais.

- Dados hostis são usados diretamente ou concatenados. O SQL ou comando contém a estrutura e os dados maliciosos em consultas dinâmicas, comandos ou procedimentos armazenados.

Algumas das injeções mais comuns são SQL, NoSQL, injeção de comando shell, Mapeamento Relacional de Objeto (ORM), LDAP e Linguagem de Expressão (EL) ou injeção de Biblioteca de Navegação de Gráfico de Objeto (OGNL). O conceito é idêntico entre todos os intérpretes. A revisão do código-fonte é o melhor método para detectar se os aplicativos são vulneráveis a injeções. O teste automatizado de todos os parâmetros, cabeçalhos, URL, cookies, JSON, SOAP e entradas de dados XML são fortemente encorajados. As organizações podem incluir ferramentas de teste de segurança de aplicações estáticos (SAST), dinâmicos (DAST) e interativos (IAST) no pipeline de CI/CD para identificar as falhas de injeção introduzidas antes da implantação da produção.

https://owasp.org/Top10/pt-BR/A03_2021-Injection/

4° Design Inseguro

Descrição:

O design inseguro é uma categoria ampla que representa diferentes pontos fracos, expressos como "design de controle ausente ou ineficaz". O design inseguro não é a fonte de todas as outras 10 categorias principais de risco de segurança. Há uma diferença entre design inseguro e implementação insegura. Nós diferenciamos entre falhas de design e defeitos de implementação por um motivo, eles têm diferentes causas raízes e remediação. Um design seguro ainda pode ter defeitos de implementação que levam a vulnerabilidades que podem ser exploradas. Um design inseguro não pode ser corrigido por uma implementação perfeita, pois, por definição, os controles de segurança necessários nunca foram criados para a defesa contra ataques específicos. Um dos fatores que contribuem para um design inseguro é a falta de perfis de risco de negócios inerentes ao software ou sistema que está sendo desenvolvido e, portanto, a falha em determinar o nível de design de segurança necessário.

Gerenciamento de Requisitos e Recursos: Colete e negocie os requisitos de negócios para uma aplicação com a empresa, incluindo os requisitos de proteção relativos à confidencialidade, integridade, disponibilidade e autenticidade de todos os ativos de dados e a lógica de negócios esperada. Leve em consideração a exposição da sua aplicação e se você precisa de segregação de tenants (além do controle de acesso). Compile os requisitos técnicos, incluindo requisitos de segurança funcionais e não funcionais. Planeje e negocie o orçamento cobrindo todo o projeto, construção, teste e operação, incluindo atividades de segurança.

Design Seguro: O design seguro é uma cultura e metodologia que avalia constantemente as ameaças e garante que o código seja desenvolvido e testado de forma robusta para evitar métodos de ataque conhecidos. A Modelagem de Ameaças deve ser integrada às sessões de refinamento (ou atividades semelhantes); procure por mudanças nos fluxos de dados e controle de acesso ou outros controles de segurança. No desenvolvimento da história do usuário, determine o fluxo correto e os estados de falha, certifique-se de que sejam bem compreendidos e aceitos pelas partes responsáveis e afetadas. Analise suposições e condições para fluxos esperados e de falha, assegure-se de que eles ainda sejam precisos e desejáveis. Determine como validar as suposições e fazer cumprir as condições necessárias para comportamentos adequados. Certifique-se de que os resultados sejam documentados na história do usuário. Aprenda com os erros e ofereça incentivos positivos para promover melhorias. O design seguro não é um add-on nem uma ferramenta que você pode adicionar ao software.

Ciclo de Vida de Desenvolvimento Seguro: O software seguro requer um ciclo de vida de desenvolvimento seguro, alguma forma de padrão de projeto seguro, metodologia de paved road, bibliotecas de componentes protegidos, ferramentas e modelagem de ameaças. Procure seus especialistas em segurança no início de um projeto de software, durante todo o projeto e durante a manutenção de seu software. Considere aproveitar o OWASP Software Assurance Maturity Model (SAMM) para ajudar a estruturar seus esforços de desenvolvimento de software seguro.

https://owasp.org/Top10/pt-BR/A04_2021-Insecure_Design/

5° Configuração Incorreta de Segurança

Descrição:

A aplicação pode ser vulnerável se for:

- Falta de proteção de segurança apropriada em qualquer parte da stack das aplicações ou permissões configuradas incorretamente em serviços em nuvem.
- Recursos desnecessários são ativados ou instalados (por exemplo, portas, serviços, páginas, contas ou privilégios desnecessários).
- As contas padrão e suas senhas ainda estão ativadas e inalteradas.
- O tratamento de erros revela stack traces ou outras mensagens de erro excessivamente informativas aos usuários.
- Para sistemas atualizados, os recursos de segurança mais recentes estão desabilitados ou não estão configurados com segurança.
- As configurações de segurança nos servidores das aplicações, nos frameworks (por exemplo, Struts, Spring, ASP.NET), bibliotecas, bancos de dados, etc., não estão definidas para proteger os valores.
- O servidor não envia cabeçalhos ou diretivas de segurança, ou eles não estão configurados para proteger os valores.
- O software está desatualizado ou vulnerável (consulte A06: 2021-Componentes Vulneráveis e Desatualizados).
- Sem um processo de configuração de segurança de aplicações que seja integrado e repetível, os sistemas correm um risco maior.

6° Componentes Vulneráveis e Desatualizados

Descrição:

Você provavelmente está vulnerável:

- Se você não souber as versões de todos os componentes que usa (tanto do lado do cliente quanto do lado do servidor). Isso inclui componentes que você usa diretamente, bem como dependências aninhadas.
- Se o software for vulnerável, sem suporte ou desatualizado. Isso inclui o sistema operacional, servidor web/application, sistema de gerenciamento de banco de dados (DBMS), aplicações, APIs e todos os componentes, ambientes de tempo de execução e bibliotecas.

- Se você não faz a varredura de vulnerabilidades regularmente e não assina os boletins de segurança relacionados aos componentes que você usa.

- Se você não corrigir ou atualizar a plataforma, as estruturas e as dependências subjacentes de maneira oportuna e baseada em riscos. Isso geralmente acontece em ambientes em que a correção é uma tarefa mensal ou trimestral sob controle de alterações, deixando as organizações abertas a dias ou meses de exposição desnecessária a vulnerabilidades corrigidas.

- Se os desenvolvedores de software não testarem a compatibilidade de bibliotecas atualizadas, atualizações ou com patches.

- Se você não proteger as configurações dos componentes (consulte A05: 2021-Configuração Incorreta de Segurança).

https://owasp.org/Top10/pt-BR/A06_2021-Vulnerable_and_Outdated_Components/

7° Falhas de Identificação e Autenticação

Descrição:

Confirmação da identidade, autenticação e sessão do usuário gerenciamento é fundamental para proteger contra autenticação relacionada ataques. Pode haver pontos fracos de autenticação se o aplicativo:

- Permite ataques automatizados, como preenchimento de credenciais, onde o invasor tem uma lista de nomes de usuários e senhas válidos.

- Permite força bruta ou outros ataques automatizados.

- Permite senhas padrão, fracas ou conhecidas, como "Senha1" ou "admin/admin".

- Usa recuperação de credenciais fraca ou ineficaz e esqueci a senha processos, como "respostas baseadas em conhecimento", que não podem ser feitas de modo seguro.

- Usa armazenamento de dados e senhas em texto simples, criptografadas ou com hash fraco (consulte A02:2021-Falhas Criptográficas).

- Possui multifator de autenticação ausente ou ineficaz.

- Expõe o identificador de sessão na URL.

- Reutiliza o identificador de sessão após o login bem-sucedido.

- Não invalida corretamente IDs de sessão. Sessões de usuário ou tokens de autenticação (principalmente tokens de logon único (SSO)) não são devidamente invalidado durante o logout ou um período de inatividade.

https://owasp.org/Top10/pt-BR/A07_2021-Identification_and_Authentication_Failures/

8° Falhas de Software e Integridade de Dados

Descrição:

Falhas na integridade de software e dados estão relacionadas a código e infraestrutura que não protegem contra violações de integridade. Um exemplo disso é quando um aplicativo depende de plugins, bibliotecas ou módulos de fontes, repositórios e redes de entrega de conteúdo (CDNs) não confiáveis. Um pipeline de CI/CD inseguro pode introduzir a possibilidade de acesso não autorizado, código malicioso ou comprometimento do sistema. Por último, muitos aplicativos agora incluem funcionalidade de atualização automática, onde as atualizações são baixadas sem verificação de integridade suficiente e aplicadas ao aplicativo previamente confiável. Atacantes podem potencialmente fazer upload de suas próprias atualizações para serem distribuídas e executadas em todas as instalações. Outro exemplo é quando objetos ou dados são codificados ou serializados em uma estrutura que um atacante pode ver e modificar, o que torna a deserialização insegura.

https://owasp.org/Top10/pt-BR/A08_2021-Software_and_Data_Integrity_Failures/

9° Falhas de Registro e Monitoramento de Segurança

Descrição:

Retornando à lista OWASP Top 10 de 2021, essa categoria ajuda a detectar, escalonar e responder a violações ativas. Sem o monitoramento e registro, as violações não podem ser detectadas. A falta de registro, detecção, monitoramento e resposta ativa ocorre sempre que:

- Eventos auditáveis, como logins, logins falhos e transações de alto valor, não são registrados.
- Avisos e erros geram mensagens de log inexistentes, inadequadas ou confusas.
- Logs de aplicativos e APIs não são monitorados quanto a atividades suspeitas.
- Logs são armazenados apenas localmente.
- Limiares de alerta apropriados e processos de escalonamento de resposta não estão em vigor ou são eficazes.
- Testes de penetração e varreduras por ferramentas de teste de segurança de aplicativos dinâmicos (DAST), como OWASP ZAP, não acionam alertas.
- A aplicação não pode detectar, escalonar ou alertar para ataques ativos em tempo real ou quase em tempo real.

Você está vulnerável a vazamento de informações tornando eventos de registro e alerta visíveis para um usuário ou um atacante (veja A01:2021-Quebra de Controle de Acesso).

https://owasp.org/Top10/pt-BR/A09_2021-Security_Logging_and_Monitoring_Failures/

10° Falsificação de Solicitação do Lado do Servidor (SSRF)

Descrição:

As falhas de SSRF ocorrem sempre que um aplicativo da web busca um recurso remoto sem validar a URL fornecida pelo usuário. Ele permite que um invasor force o aplicativo a enviar uma solicitação criada para um destino inesperado, mesmo quando protegido por um firewall, VPN ou outro tipo de lista de controle de acesso à rede (ACL).

Como os aplicativos da web modernos fornecem aos usuários finais recursos convenientes, buscar uma URL se torna um cenário comum. Como resultado, a incidência de SSRF está aumentando. Além disso, a gravidade do SSRF está se tornando mais alta devido aos serviços em nuvem e à complexidade crescente das arquiteturas.

https://owasp.org/Top10/pt-BR/A10_2021-Server-Side_Request_Forgery_%28SSRF%29/

Prática Zed Attack Proxy/Juice-Shop

Teste de segurança de software é o processo de avaliar e testar um sistema para descobrir riscos de segurança e vulnerabilidades do sistema e de seus dados. Não existe uma terminologia universal, mas, para nossos propósitos, definimos avaliações como a análise e descoberta de vulnerabilidades sem a tentativa de explorá-las.

https://www.zaproxy.org

Considerações Essenciais sobre Varreduras com o ZAP

Automated Scan: Primeiro o ZAP realiza uma varredura passiva no aplicativo web ou site com seu módulo Spider, verificando cada página que encontrar na URL do alvo. Assim que finaliza a varredura o ZAP usará o Active Scan para simular um ataque real em todas as páginas, funcionalidades e parâmetros descobertos, não use essa varredura contra alvos que você não tem permissão para testar.

Manual Explore: O ZAP fornece 2 Spiders para rastrear aplicativos web e indexar o conteúdo de sites.

Spider Tradicional: Descobre links examinando linguagem HTML nas respostas da aplicação web. Esse Spider é rápido, mas nem sempre é eficaz ao explorar um aplicativo web interativo que gera links usando JavaScript.

AJAX Spider: Para aplicativos web interativos provavelmente será mais eficaz. Esse spider que explora o aplicativo web invocando navegadores que seguem links gerados via JavaScript. O AJAX Spider é mais lento que o tradicional e requer configuração adicional para uso em determinados ambientes. O ZAP verificará passivamente via Spider todas as solicitações e respostas enviadas por meio dele. A varredura passiva não altera as respostas de forma alguma e é considerada segura. A verificação também é realizada em um “thread” em segundo plano para não retardar a exploração. A

varredura passiva é boa para encontrar algumas vulnerabilidades e também como uma forma de ter uma ideia do estado básico de segurança de um aplicativo web ou site.

Active Scan via HUB: A varredura ativa via HUB busca encontrar outras vulnerabilidades usando ataques bem conhecidos contra os alvos selecionados. A varredura ativa é um ataque real a esses alvos e pode colocá-los em risco, portanto, não use a varredura ativa contra alvos que você não tem permissão para testar.

Para informações detalhadas acesse o link: https://www.zaproxy.org/getting-started/

https://github.com/sergiorez-cmd/secbitrez/blob/main/owasp/OWASP_ZAP_Basic.pdf

Considerações Essenciais sobre o Web-App Juice-Shop

O Juice Shop é escrito em Node.js, Express e Angular. Foi o primeiro aplicativo escrito inteiramente em JavaScript listado no Diretório OWASP VWA.

O aplicativo contém um vasto número de desafios de hacking de dificuldade variada, nos quais o usuário deve explorar as vulnerabilidades subjacentes. O progresso do hacking é monitorado em um placar. Encontrar esse placar é, na verdade, um dos desafios (fáceis)!

Além do caso de uso para hackers e treinamento de conscientização, proxies de pentesting ou scanners de segurança podem usar o Juice Shop como um aplicativo "cobaia" para verificar o desempenho de suas ferramentas com frontends de aplicativos com uso intensivo de JavaScript e APIs REST.

https://github.com/juice-shop/juice-shop

Exemplos Práticos:

1° Inspecionar o Código do Web-App Juice-Shop via Browser

- Acesse a web page http://localhost:3000 e clique com o botão direito para selecionar a opção “Inspect”, clique em “Debugger”, “Sources” e localize o arquivo main.js, clique no símbolo { } para organizar o código.
- Clique em “Search” e pesquise utilizando a palavra “path”.
- Acesse os caminhos descobertos na inspeção de código
- Encontre a página "Score Board" e “Sand Box”cuidadosamente escondida.

2° Brute Force em Diretórios Web

- Execute um terminal no Kali Linux e digite o seguinte comando

***$ dirb http://localhost:3000* -z 30**

- Provocar um erro que não é tratado de forma nem muito elegante nem consistente
- Acessar diretórios encontrados para obter um documento confidencial
- Use o ataque “Poison Null Byte” para baixar arquivos não permitidos do diretório ftp, digite após a URL %2500 seguido de . e o tipo do arquivo permitido (.pdf, .md).

https://wiki.zacheller.dev/web-app-pentest/upload-download/error-only-.md-and-.pdf-files-are-allowed

3° Brute Force em Login Web com ZAP

- Acesse o produto Apple Juice e visualize sua avaliação para descobrir o nome de usuário.
- Criar uma wordlist customizada ou faça download de uma wordlist

  $ cat /usr/share/wordlists/sqlmap.txt | grep admin > admin-passwd.txt

  https://github.com/danielmiessler/SecLists/blob/master/Passwords/Common-Credentials/best1050.txt

- Execute o ZAP e selecione Manual Explore e digite A URL do Juice-Shop http://localhost:3000 e clique em Launch Browser
- Na webpage do juice_Shop acesse Conta > Login e digite o usuário e qualquer senha.
- No ZAP clique em History > URL de login > clique com o botão direito e selecione Attack > FUZZ
- Marque a string de password Add > Add > File > Select admin-passwd.txt Open > Add > OK
- Start Fuzzer
- Em Fuzzer procure por Code 200 OK
- Efetue login com o e-mail e a senha se você obteve via ZAP Fuzzer
- Acesse o diretório /administration descoberto na inspeção de código
- Acesse o carrinho de outros usuários via ZAP “Open/Resend with Request Editor”

4° Ataques de Injeção

Injeção SQL Bypass Auth

- Acesse o produto Banana Juice e visualize sua avaliação para descobrir o nome de usuário.
- Efetue login com o e-mail bender@juice-sh.op digitando o parâmetro ‘ -- logo após e a qualquer senha para executar um ataque de injeção de bypass de autenticação.

https://github.com/sergiorez-cmd/secbitrez/blob/main/owasp/sql-inject-basic.md

Execute um ataque DOM XSS (Cross-site-scripting)

- Cole a string de ataque <iframe src="javascript:alert('xss')"> no campo Pesquisar….
- Pressione a tecla Enter.
- Uma caixa de alerta com o texto "xss" deverá aparecer.
- Use payload bonus DOM XSS.

<iframe width="100%" height="166" scrolling="no" frameborder="no" allow="autoplay" src="https://w.soundcloud.com/player/?url=https%3A//api.soundcloud.com/tracks/771984076&color=%23ff5500&auto_play=true&hide_related=false&show_comments=true&show_user=true&show_reposts=false&show_teaser=true"></iframe>

https://owasp.org/www-community/attacks/DOM_Based_XSS

Realizar um ataque automatizado de Injeção SQL

- No ZAP clique em History > URL de search > clique com o botão direito e selecione Attack > Active Scan
- Start Scan
- Alerts > SQL Injection

Exfiltrate Database Schema

- Execute o ZAP e selecione Manual Explore e digite A URL do Juice-Shop http://localhost:3000 e clique em Launch Browser
- Na webpage do Juice-Shop faça uma pesquisa digitando uma letra
- No ZAP clique em History > URL de search > clique com o botão direito e selecione Open/Resend with request editor

- Altere a requisição adicionando o parâmetro ‘)

  GET http://localhost:3000/rest/products/search?q=') HTTP/1.1

- Essa requisição provocou um erro ao expor a consulta SQL e qual mecanismo de banco de dados ela está usando.

- Continue adicionando os seguintes payloads para obter os nomes do database, schema, tables para obter as credenciais dos usuários.

```
'))%20--%2

'))%20order%20by%209%20--%20

'))%20union%20all%20select%201,2,3,4,5,6,7,8,9%20--%20

a'))%20union%20all%20select%201,2,3,sql,5,6,7,8,9%20from%20sqlite_master%20--%20

'))%20union%20all%20select%201,email,username,4,password,6,7,8,9%20from%20Users%20--%20
```

https://github.com/sergiorez-cmd/secbitrez/blob/main/owasp/sql-inject-basic.md

Realize um brute force no hash de senha do usuário jim

$ hashid -m jim-hash.txt

$ hashcat -a 0 -m 0 jim-hash.txt /usr/share/wordlists/rockyou.txt

$ john --wordlist=/usr/share/wordlists/rockyou.txt --format=Raw-MD5 hash-5.txt --fork=2

Obter o Database Schema Juice-Shop via SQLMap

- Execute um terminal no Kali Linux e digite os seguintes comados:

```
$ sqlmap -u "http://localhost:3000/rest/products/search?q=q" --dbms=sqlite --level=3 --risk=3 --technique=B --threads=4 --schema

$ sqlmap -u "http://localhost:3000/rest/products/search?q=q" --dbms=sqlite --level=3 --risk=3 --technique=B --threads=4 --tables

$ sqlmap -u "http://localhost:3000/rest/products/search?q=q" --dbms=sqlite -D SQLite_masterdb -T Users --columns --threads=4

$ sqlmap -u "http://localhost:3000/rest/products/search?q=q" --dbms=sqlite -D SQLite_masterdb -T Users -C email,password,role,topSecret,username --dump --threads=4
```

https://github.com/sergiorez-cmd/secbitrez/blob/main/owasp/sqlmap.md

5° Alterar Senha de Usuário via Função Esqueceu a Senha

- Descubra a pergunta de segurança do usuário bjoern@owasp.org
- Pesquise no Google pelo endereço de e-mail do usuário
- Acesse o link do X https://x.com/bkimminich/status/1594985736650035202
- Descubra a pergunta de segurança do usuário emma@juice-sh.op
- Acesse a seção Parede de Fotos do Juice-Shop e procure pela foto postada por Emma
- Abra a foto em outra janela do navegador ou baixe a foto para poder executar um zoom e descobrir o departamento anterior onde Emma trabalhava.

Bônus OSINT

- Descubra a senha do usuário mc.safesearch@juice-sh.op
- Acesse o painel Score-Board e visualize as dicas sugeridas para descobrir s senha
- Pesquise no Google pelo nome da música indicada nas dicas
- Assista o vídeo no You Tube e habilite o modo de legendas para descobrir a senha

6° Reportar Problemas de Segurança Encontrado

Comporte-se como qualquer "white-hat" deveria antes de entrar em ação.

https://securitytxt.org

Quando riscos de segurança em serviços web são descobertos por pesquisadores de segurança independentes que compreendem a gravidade do risco, eles geralmente não dispõem dos canais necessários para divulgá-los adequadamente. Como resultado, problemas de segurança podem não ser reportados. O security.txt define um padrão para ajudar as organizações a definir o processo para que pesquisadores de segurança divulguem vulnerabilidades de segurança com segurança.

7° Bonus Easter Egg

- Use o ataque Poison Null Byte para baixar http://localhost:3000/ftp/eastere.gg%2500.md
- Obtenha a string criptografada do arquivo eastere.gg no desafio "Encontre o easter egg escondido":
- Decodifique em Base64 e depois para ROT13

  https://gchq.github.io/CyberChef

- Visite a URL decodificada para acessar a página com o Easter Egg

CTF (Capture The Flag)

Um CTF (Capture The Flag) é um desafio de segurança cibernética em que os participantes devem resolver quebra-cabeças de segurança para encontrar sinalizadores ou vulnerabilidades ocultas em sistemas. Os CTFs são comumente usados para treinamento, competições e avaliação de habilidades em segurança cibernética. Eles simulam cenários de ataque do mundo real e ajudam a aprimorar habilidades de segurança defensivas e ofensivas.
Categorias de desafios CTF

Web Hacking: Exploração de vulnerabilidades em aplicações web, como SQL injection, Cross-Site Scripting (XSS), etc.

Criptografia: Decifrar mensagens ou códigos usando técnicas criptográficas.

Engenharia reversa: Analisar e entender o funcionamento interno de programas ou sistemas para encontrar vulnerabilidades ou informações.

Exploração de binários do SO: Encontrar e explorar vulnerabilidades em programas executáveis.
Forensics: Análise de dados forenses para encontrar pistas e informações relevantes.

Networking: Análise de tráfego de rede, protocolos e configuração de redes.

Misc (Diversos): Desafios que não se encaixam em categorias específicas, muitas vezes combinando diferentes áreas.

Links:

https://tryhackme.com

https://www.hackthebox.com

https://www.hacker101.com

https://pentest-ground.com

http://vulnweb.com

https://www.vulnhub.com

https://websploit.org

https://github.com/The-Art-of-Hacking

https://github.com/sergiorez-cmd/secbitrez