ETHICAL HACKING CODE REVIEW AND EXPLOITATION (EHCRX)

DESCRIÇÃO

(Nível Intermediário)

Carga Horária: 40 hrs

Em um cenário onde vulnerabilidades desconhecidas continuam a ameaçar sistemas críticos, o curso *Ethical Hacking Code Review and Exploitation* (EHCRX) oferece uma imersão nas técnicas de análise de código-fonte, tanto ofensiva quanto defensiva, revelando como vulnerabilidades ocultas nas etapas de desenvolvimento de software podem ser exploradas para contornar lógicas de negócio e causar grandes impactos.

O treinamento aborda formas de identificar falhas de segurança em códigos-fonte utilizando as técnicas como análises *sources-to-sink* e *sink-to-source*, que podem ser utilizadas para verificar pacotes de código, bem como criar regras para detectar vulnerabilidades em massa, criando Provas de Conceito (PoC) para explorá-las. Além de análise de vulnerabilidades comuns, você aprenderá, na prática, o conceito de combinação estratégica de falhas, aparentemente isoladas, para maximizar o impacto de um ataque, procedimento conhecido como *vulnerability chain*.

Muitos recursos são gastos com soluções de SAST/DAST e, mesmo assim, várias vulnerabilidades são encontradas diariamente. Não espere até que uma brecha no código comprometa sua aplicação. Proteja seus projetos e sua reputação aprendendo como detectar e corrigir falhas que ferramentas tradicionais não conseguem identificar. O intuito deste curso é ensinar como encontrar e realizar esses ataques, para que a detecção básica dessas ferramentas seja complementada com comportamentos de atacantes reais, gerando *exploits* funcionais.

Como as empresas conseguem se proteger desses ataques? Como evitar que um erro de programação seja explorado para exfiltrar todos os dados de suas aplicações? Ou como criar técnicas onde os desenvolvedores não precisam se preocupar com segurança e simplesmente programar utilizando códigos pré-definidos? Durante o curso, os alunos irão aprender técnicas de proteção para todos os ataques demonstrados, desenvolvendo uma aplicação vulnerável de início e depois corrigindo todas as falhas. Diversas técnicas de *Secure by Design* serão apresentadas, tais como *Design Domain Driven*, proporcionando um entendimento de como publicar aplicações seguras por padrão.

Ao final do curso, os alunos poderão empregar todos os conhecimentos adquiridos para executar análises de vulnerabilidades reais, estudando e utilizando CVEs recentes, e ter a possibilidade de encontrar novas falhas e executar outros caminhos de ataque.

Serão abordados diversos temas: fundamentos de Pentest Whitebox, evasão de filtros, contorno (*bypass*) de mecanismos de autenticação e autorização, pesquisa de vulnerabilidades em código-fonte, provas de conceito, relatório de falhas, análise de causa raiz, desenvolvimento seguro, *secure by design*, entre outros.

MÓDULO 1

1. Introdução ao Pentest Whitebox
2. Análise de códigos-fonte
3. Sources and Sinks
4. Análise *Source-to-Sink*
5. Análise *Sink-to-Source*
6. *Debugging*
7. Abstração e entendimento de códigos-fonte
8. *Frameworks* de desenvolvimento
9. Entrada (*input*) de usuário
10. Cadeia de Vulnerabilidades (*Vulnerability Chain*)
11. *Tracing* de vulnerabilidades
12. Confecção de relatório

MÓDULO 2

1. Atacando aplicações inseguras
2. Desenvolvimento de aplicações inseguras
3. Atacando entrada (*input*) do usuário
   a. *Cross-Site Scripting* (DOM, *Stored* e *Reflected*)
   b. *Server-Side Request Forgery* (SSRF)
   c. *Command Injection*
   d. XML *External Entities* (XXE)
   e. *Template Injection*
   f. *Cross-Site Request Forgery* (CSRF)
   g. *Open Redirection*
   h. SQL *Injection*
   i. *Mass Assignment*
   j. IDOR (BOLA, BFLA)
   k. Deserialização
   l. *Insecure File Uploads*
4. Testes de autenticação e autorização
   a. *Bypass* de autenticação e autorização
   b. *Type Juggling*
   c. *Magic Hashes*
   d. *Secrets* expostos

    e. *Race Condition*
    f. Mecanismos de autorização e autenticação no código-fonte
5. Subvertendo lógicas de negócio
6. Criação de casos de uso
7. Análises de Causa Raiz
    a. Encontrando a causa principal de vulnerabilidades
    b. Encontrando vulnerabilidades em massa
    c. Automatização e descoberta de vulnerabilidades customizadas

MÓDULO 3

1. Defesa contra as vulnerabilidades
2. Desenvolvimento de aplicações seguras
3. Melhores práticas de desenvolvimento
4. O que não fazer quando projetar uma aplicação
5. Defendendo entrada (*input*) do usuário
    a. *Cross-Site Scripting* (DOM, *Stored* e *Reflected*)
    b. *Server-Side Request Forgery* (SSRF)
    c. *Command Injection*
    d. XML *External Entities* (XXE)
    e. *Template Injection*
    f. *Cross-Site Request Forgery* (CSRF)
    g. *Open Redirection*
    h. SQL *Injection*
    i. *Mass Assignment*
    j. IDOR (BOLA, BFLA)
    k. Deserialização
    l. *Insecure File Uploads*
6. Protegendo mecanismos de autenticação e validação
7. Protegendo objetos e instanciações
8. Protegendo *secrets*
9. *Secure by Design*
    a. Casos de uso
    b. Casos de abuso
    c. Mitigações
    d. Modelagem de dados segura

e. *Read Once Object Pattern*
f. *Domain Driven Design*
g. Outros *Design Patterns* relacionados a segurança

MÓDULO 4

1. Pesquisa de vulnerabilidades em código-aberto
2. Processos de análise de códigos-fonte
3. Análise de múltiplos pacotes
4. Escrita de *Exploits*
   a. Escrita de *exploits* para aplicações web
   b. Escrita de *exploits* de rede
5. Estudos de caso
   a. *Common Vulnerabilities* and *Exposures* (CVE)
   b. *Java Name and Directory Interface* (JNDI) *Injection*
   c. *Spring Expression Language* (SpEL) *Injection*
   d. Envio e obtenção de CVEs

PRÉ-REQUISITOS

- Conhecimentos básicos de Rede de Computadores e Serviços de Rede: Protocolo TCP/IP, FTP, HTTP, HTTPS, SMB, SSH, DNS, ICMP.
- Conhecimentos básicos de Penetration Testing: Metodologia e Procedimentos.
- Familiaridade com a distribuição Kali Linux.
- Familiaridade com ferramentas de Virtualização – VMWare.
- Conhecimentos básicos de Lógica de Programação.
- Conhecimentos básicos em Python e Java.

PÚBLICO-ALVO

- Desenvolvedores de Software
- Especialistas em Segurança da Informação
- Analista de Segurança da Informação
- Pentesters
- Membros de Red Team / Blue Team
- Membros de CSIRT
- Analista de SOC
- Profissionais de TI com interesse e afinidade na área de Segurança da Informação

MATERIAL NECESSÁRIO

- Os alunos devem utilizar suas próprias estações de trabalho (Windows, Linux ou Mac OS), com a capacidade de executar de 02 a 03 Máquinas Virtuais (VM) simultaneamente.
- Configuração mínima de 8GB de RAM, 60GB de espaço livre em disco, porta USB e placa de rede (RJ45 ou *wireless*) para acesso à Internet.
- Software de Virtualização: VMWare, versão mais atualizada, de preferência *Workstation* (para hosts Windows ou Linux) ou *Fusion* (para host Mac OS).

MATERIAL RECEBIDO

- Apostila do Curso no formato PDF.
- Gravações das sessões ao vivo, que ficam concentradas no portal do aluno, o GoHacking Academy.
- Certificado de Conclusão do Curso no formato PDF (com a carga horária e ementa).

CAPACIDADES ALCANÇADAS

No final do curso, o aluno estará apto a:

- Entender os principais aspectos de um Pentest Whitebox.
- Identificar vulnerabilidades pela análise de código-fonte de aplicações.
- Definir a causa principal (raiz) de uma vulnerabilidade.
- Encontrar falhas complexas em código-fonte.
- Explorar os mecanismos de entrada de usuário em aplicações.
- Explorar o processo de autenticação e autorização de aplicações.
- Analisar repositórios de código para encontrar vulnerabilidades.
- Automatizar a descoberta de vulnerabilidades em massa em repositórios de código.
- Criar *exploits* capazes de combinar múltiplas vulnerabilidades (*vulnerability chain*) para atingir um objetivo definido.
- Proteger aplicações contra diversos ataques modernos.
- Encontrar vulnerabilidades em aplicações durante o processo de desenvolvimento.
- Entender as melhores práticas do modelo *Secure by Design*.
- Defender os mecanismos de entrada de usuário em aplicações.
- Proteger o processo de autenticação e validação de aplicações.
- Ajudar desenvolvedores no entendimento das melhores formas de correção de falhas.
- Identificar novos ataques e obter CVEs.

JOÃO PAULO DE ANDRADE FILHO