INSTITUTO DO EMPREGO E FORMAÇÃO PROFISSIONAL	**Delegação Regional do Centro** **Centro de Emprego e Formação Profissional de Viseu**
	Programador/a de Informática (AÇÃO 24/2023)
	Formando/a: Judite da Silva Miguel **N.º: 3824316** **Data: 22 / 05 / 2024**

Portefólio Reflexivo de Aprendizagens (PRA)

UFCD: 10795 – Segurança no Desenvolvimento de Software

Número de Horas: 25 horas

Formador/a: Gisela Marilda Martins Firmino **Assinatura:** _____

Mediadora: Carla David **Assinatura:** _____

Objetivos da UFCD:

- Identificar os conceitos essenciais em programação.
- Minimizar riscos de segurança no desenvolvimento de software.
- Aplicar os princípios de segurança na programação.

Nesta UFCD começamos por fazer um teste disgnóstico. Posteriormente abordámos o ciclo de vida de desenvolvimento do software (SDLC-Software Development Life Cycle), quais os seus objetivos, importância, vantagens e o seu funcionamento.

Abordámos as etapas do ciclo de vida do software, planear, analisar, desenhar, implementar, testar, instalar e manter.

De seguida abordámos os modelos de SDLC, modelo em Cascata, modelo Iterativo, modelo Espiral, modelo Ágil, modelo V-Shape, modelo Big Bang e as vantagens e desvantagens de cada modelo. Realizámos uma ficha de trabalho de avaliação de consolidação destes conceitos.

Visualizámos um curso de desenvolvimento de software seguro e segurança na informação. O curso teve como base o OWASP (Open Web Application Security Project) mais concretamente o OWASP Top 10 do 2017 que evidencia as 10 vulnerabilidades exploradas em ambiente WEB. A OWASP importa-se com a segurança em

aplicações WEB e procura melhorar a qualidade do software em ambientes WEB. As 10 vulnerabilidades são: injeção, quebra de autenticação, exposição de dados sensíveis, entidades externas XML (XXE), quebra de controlo de acessos, configurações de segurança incorretas, Cross-Site Scripting (XSS), desserialização insegura, utilização de componentes vulneráveis, registo e monitorização insuficiente. Em cada um abordamos os vetores de ataque a falha de segurança e o impacto.

Aa falhas de injeção ocorrem quando trechos maliciosos são injetados em entradas de dados legítimos e interpretadas como comando pelo interpretador. Podem ser de vários tipos SQL, NoSQL, LDAP, XPath ou mesmo Shell. Os vetores de ataque são as variáveis de ambiente, formulários WEB, parâmetros de URL, manipulação de sessão e qualquer entrada de dados.

A quebra de autenticação é todo o tipo de acesso não autorizado ao sistema. Os principais vetores de ataque são: credenciais padrão (credenciais conhecidas), vazamento de dados, força bruta (adivinhar a senha do utilizador), engenharia social (fishing..)e falhas de implementação (atacante explora a falha e faz o login), credential stuffing, permissão de senhas fracas, recuperação de credenciais ou senha fracos, senhas armazenadas em texto plano, criptografadas ou hash fraco, não implementa a autenticação multi-factor, expõe ID de sessão no URL, não mudar ID de sessão após um ID bem sucedido.

A exposição de dados sensíveis é a publicação de dados não autorizada na internet (ex:SQL injection). Dados pessoais são exemplo saúde, religião, filiação partidária…., senhas. Os vetores de ataque são: ataque MitM (entre um ponto a e c existe um ponto b escutando a encriptação), ataque de injeção (obtenção de dados sensíveis), quebra de autenticação, engenharia social (alguém se passa por alguém para obter informação) e falhas de implementação. Transmissão de dados em texto plano (empregar protocolos http, smtp, ftp) a aplicação está vulnerável a dados sensíveis porque está a ser escutada quando há dados sensíveis incluindo cópias de segurança, ataque ao backup, algoritmos criptográficos antigos ou fracos, chaves criptográficas padrão ou fracas estão a ser usadas/geradas/reutilizadas, ou não estão sendo geridas convenientemente nem existe rotatividade de chaves, quando a encriptação não está a ser forçada por diretivas ou no cabeçalho HTTP, o cliente não valida corretamente o certificado do servidor. Deve-se classificar os dados entre sensíveis e não sensíveis e não se devem armazenar desnecessariamente, criptografia em repouso para dados sensíveis, algoritmos, protocolos, chaves fortes, uso de protocolos seguros e forcar o uso de encriptação recorrendo a diretivas como HTTP Strict Transport Security (HSTS), dados não acesso a toda a internet, deve-se criptografar todos os dados em trânsito, desativar os cache para respostas com dados sensíveis.

Entidades externas de XML, muitos processadores de XML mais antigos ou mal configurados processam referencias a entidades externas dentro dos documentos XML. Essas entidades externas podem ser usadas para

obter acesso a arquivos internos, pesquisar portas de comunicação abertas, executar código remoto e perpetrar ataques de navegação de serviço, como o ataque Billion Laughs. Os vetores de ataque desse tipo de vulnerabilidade são: APIs que recebem formato XML e processam conteúdo, formulários com upload, injeção de conteúdo XML, engenharia social, falhas de implementação. O atacante pode ler o arquivo de um ou mais servidores. A vulnerabilidade existe se a aplicação aceita XML diretamente, ou carregamento de arquivos XML sem a devida validação, se qualquer processador de XML em uso na aplicação ou nos serviços web baseados em SOAP permite Definição de Tipo de Documento (DTD), quando a aplicação implementa SAML (Security Assertion Markup Language) para processamento de identidade no contexto de segurança federada ou Single Sign-on(SSO), SAML usa XML para validação da identidade e pode por isso ser vulnerável. Se possível não usar XML. Optar por um formato de dados mais simples JSON (por ex.). Corrige ou atualize todos os processadores e bibliotecas de XML usados pela aplicação, nas dependências ou sistema operacional. Atualizar o SOAP para a versão 1.2 ou superior.

A vulnerabilidade quebra de controle de acesso dá-se quando alguém ou sistema acede a um recurso que não está autorizado. Há três tipos de controle de acesso: vertical (função papel do utilizador no sistema), horizontal(recurso), dependente do contexto. Os vetores de ataque são: manipulação de parâmetros em URLs (quando altera), manipulação de sessão (quando altera), injeção (escala de privilégio), engenharia social e manipulação de protocolo HTTP. Fica-se vulnerável quando a aplicação aceita parâmetros de API´s ou URL´s sem ver se o utilizador logado possui permissão para ler o valor do parâmetro recebido (através do ID). Quando permite elevação de privilégios por manipulação de parâmetros, permite manipulação de metadados, repetição ou adulteração do JSON Web Token (JWT) do controlo de acesso, cookie ou campo hidden para elevação de privilégios, não implementa corretamente a política de compartilhamento de recursos entre origens (CORS), não protege devidamente todos os verbos HTTP como POST, PUT ou DELETE. Deve-se ter um bom controle de acesso, implementar um mecanismo único de controle de acesso, os programadores devem declarar permissões em cada função, negue acesso a todos os recursos por padrão, desativa-se a listagem de diretórios no servidor e assegure-se que nenhum metadado está presente na raiz do servidor web (.git), registe todas as falhas no controle de acesso e alerte os administradores quando necessário, invalide Ids de sessão e tokens JWT após cada logout.

Para testar conhecimentos fizemos alguns questionários.

Esta UFCD foi interessante, não tinha grandes conhecimentos de segurança de desenvolvimento de software. Permitiu ficar com uma ideia genérica dos vetores de ataque e a maneira de precaver. Gostei bastante do curso que fizemos está claro e objetivo. Gostei da maneira como a formadora geriu a UFCD.