Faculdade de Tecnologia de Franca "Dr. Thomaz Novelino"
Curso Tecnológico Superior em Desenvolvimento de *Software* Multiplataforma

SEGURANÇA NO DESENVOLVIMENTO DE APLICAÇÕES – 2025/2

Prof. Me. Fausto Gonçalves Cintra – professor@faustocintra.com.br

PROVA 2 (P2)

1 INSTRUÇÕES GERAIS

1. A prova é ***estritamente individual***.

2. A provas idênticas, com alto grau de semelhança ou indícios de utilização de Inteligência Artificial, será atribuída a nota ZERO.

3. O valor da prova é 10,0 (dez), conforme explicado no documento *[ISG022-00] Apresentação*.

2 INSTRUÇÕES ESPECÍFICAS

1. No decorrer do semestre, estudamos a lista das dez maiores vulnerabilidades de API publicada pela OWASP (OWASP Top 10 API Security Risks – 2023), quais sejam:

   a) **API1:2023 – Falha de autenticação a nível de objeto**. APIs tendem a expor *endpoints* que lidam com identificadores de objetos, criando uma ampla superfície de ataque para questões de Controle de Acesso a Nível de Objeto. Verificações de autorização a nível de objeto devem ser consideradas em cada função que acessa uma fonte de dados usando um ID fornecido pelo usuário.

   b) **API2:2023 – Falha de autenticação**. Mecanismos de autenticação são frequentemente implementados de forma incorreta, permitindo que atacantes comprometam *tokens* de autenticação ou explorem falhas na implementação para assumir temporariamente ou permanentemente as identidades de outros usuários. Comprometem a capacidade de um sistema de identificar o cliente/usuário compromete a segurança da API como um todo.

   c) **API3:2023 – Falha de autenticação a nível de propriedade**. Esta categoria combina API3:2019 - Exposição Excessiva de Dados e API6:2019 - Atribuição em Massa, focando na causa raiz: a falta ou a

validação inadequada de autorização ao nível das propriedades do objeto. Isso resulta em exposição ou manipulação de informações por partes não autorizadas.

d) **API4:2023 – Consumo irrestrito de recursos**. Satisfazer requisições de API requer recursos como largura de banda de rede, CPU, memória e armazenamento. Outros recursos, como emails/SMS/chamadas telefônicas ou validação biométrica, são disponibilizados por provedores de serviços via integrações de API e pagos por solicitação. Ataques bem-sucedidos podem levar a uma Negação de Serviço ou ao aumento dos custos operacionais.

e) **API5:2023 – Falha de autenticação a nível de função**. Políticas de controle de acesso complexas com diferentes hierarquias, grupos e funções, além de uma separação pouco clara entre funções administrativas e regulares, tendem a levar a falhas de autorização. Explorando essas questões, atacantes podem obter acesso aos recursos de outros usuários e/ou às funções administrativas.

f) **API6:2023 – Acesso irrestrito a fluxos de negócio sensíveis**. APIs vulneráveis a esse risco expõem um fluxo de negócios - como comprar um ingresso ou postar um comentário - sem compensar o impacto que a funcionalidade pode causar ao negócio se usada de forma excessiva e automatizada. Isso não necessariamente resulta de *bugs* na implementação.

g) **API7:2023 – Falsificação de requisição do lado do servidor**. Falhas de *Server-Side Request Forgery* (SSRF) podem ocorrer quando uma API busca um recurso remoto sem validar a URI fornecida pelo usuário. Isso permite que um atacante force a aplicação a enviar uma requisição manipulada para um destino inesperado, mesmo quando protegida por um *firewall* ou uma VPN.

h) **API8:2023 – Má configuração de segurança**. APIs e os sistemas que as suportam geralmente contêm configurações complexas, destinadas a tornar as APIs mais personalizáveis. Engenheiros de *software* e DevOps

podem não perceber essas configurações ou não seguir as melhores práticas de segurança em relação à configuração, abrindo a porta para diferentes tipos de ataques.

i) **API9:2023 – Gerenciamento inapropriado do inventário**. APIs tendem a expor mais *endpoints* do que aplicações *web* tradicionais, tornando a documentação adequada e atualizada altamente importante. Um inventário apropriado de *hosts* e versões de API implantadas também é crucial para mitigar problemas como versões de API obsoletas e *endpoints* de depuração expostos.

j) **API10:2023 – Consumo inseguro de APIs**. Os desenvolvedores tendem a confiar mais nos dados recebidos de APIs de terceiros do que na entrada do usuário, adotando assim padrões de segurança mais fracos. Para comprometer APIs, atacantes visam serviços de terceiros integrados em vez de tentar comprometer a API alvo diretamente.

2. Identifique, seja no código do projeto *back-end*, seja no código do projeto *front-end*, pontos onde, ***pelo menos***, ***três das vulnerabilidades*** listadas anteriormente foram evitadas ou deveriam ter sido evitadas. Registre a identificação em um comentário no código, ***especificando qual a vulnerabilidade***, de acordo com os exemplos a seguir:

```
/*
  Vulnerabilidade: API9:2023 - Gerenciamento inapropriado do inventário
  Esta vulnerabilidade foi evitado no código ao fazer X, Y, Z, etc.
*/
```

ou então

```
/*
  Vulnerabilidade: API5:2023 - Falha de autenticação a nível de função
  Esta vulnerabilidade deveria ter sido evitada no código fazendo A, B, C, etc.
*/
```

3 ENTREGA

1. Salve todos os arquivos em que tiver inserido os comentários.
2. Feche todos os terminais porventura abertos e abra um novo terminal, para fazer *commit:*

```
git add .
git commit -m "(26/11) PROVA 2"
git push
```

3. Faça *login* no GitHub.

4. Acesse a página do seu repositório: github.com/SEU_USUARIO/vulcom-main-2025-2.

5. Clique sobre a aba `Pull requests`. Clique sobre `New pull request` e, em seguida, sobre `Create pull request`.

6. No título da pull request, preencha com "PROVA 2". Na descrição, ***coloque o seu nome completo***. Finalize clicando sobre o botão `Create pull request`.