Curso de Extensão em Tecnologias Microsoft

INF0998 Programação segura (segurança de software)

Atividades práticas de testes de segurança automatizados com OWASP ZAP (Segurança de APIs)

Testes de segurança manuais são demorados e, quando realizados em grande quantidade, podem causar atrasos em projetos de desenvolvimento de software, ou mesmo não estarem disponíveis em tempo hábil para que correções sejam efetuadas.

Testes de segurança auxiliados por ferramentas aceleram a geração de resultados que podem ser tratados rapidamente. Porém, vale lembrar que ferramentas automáticas de verificações de segurança não substituem o testador experiente.

Esta aula usa a ferramenta de testes de intrusão ***OWASP Zed Attack Proxy (ZAP)*** para auxiliar a descoberta e a exploração de vulnerabilidades presentes nas aplicações estudadas nas aulas anteriores.

Apresentando o OWASP API Security Risks 2019

OWASP API Security https://owasp.org/www-project-api-security/

As explorações dos ***OWASP Top 10 API Security risks*** são realizadas sobre a aplicação **Juice Shop**.

Descobrindo a API da aplicação Juice Shop

Na tela/página inicial da aplicação Juice Shop no browser.

- Ferramentas do desenvolvedor → Sources
- Visualizar o arquivo main-es2015.js
    - Usar ***Pretty printing*** ou botão ***{}***
- Buscar (menu três pontos verticais → buscar) pela palavra *path.*
    - O resultado da busca é toda a API da aplicação!
- Identificar a API do dashboard de desafios e acessá-la.
    - Spoiler alert! https://localhost:3000/#/score-board

API 1: 2019 Broken Object Level Authorization

Na Juice Shop → desafio ***View another user's shopping basket***
O objetivo é visualizar a cesta de compras de um usuário, qualquer um, sem estar logado como ele.

- Criar dois usuários. Por exemplo, user1/user1 e user2/user2
- Encher a cesta do user2 com diversos produtos do juice shop.
- Descobrir e explorar a API da cesta de compras
    - Exploração 1
        - abrir as ferramentas de desenvolvedor
        - Entrar o App Juice Shop logado como user1
        - acessar a cesta de compras do user1
        - em DevTools, em Network, observar que a API REST da cesta é muito simples
            - P.ex., https://try-juice-shop.herokuapp.com/rest/basket/7
            - O que é o número 7?
        - em DevTools, Application → Session Storage
            - aparecem dois elementos bid = 7 e ItemTotal = 0.
                - será o mesmo número 7?
        - Conclusão deduzida: bid quer dizer BasketID
        - Alterar bid para algum valor qualquer e atualizar a página.
            - por exemplo, o valor bid = 6 !
            - o que acontece?
        - Spoiler Alert!
            - Aparece a cesta de compras do user2.
    - Exploração 2
        - no OWASP Zap Proxy do JuiceShop
        - interceptar e reenviar a requisição do basket
            - botão direito na requisição do histórico;
            - Na tela de reenvio, substituir o valor de bid na requisição por outro valor qualquer até achar um valor válido.
        - Esta exploração pode ser automatizada para baixar as cestas de compras de todos os usuários. (como?)
        - Por que este ataque acontece?

API 2: 2019 Broken User Authentication (e API 8: 2019 Injection)

- Exploração 1
  - Logar na aplicação com email `' or 1=1--` e qualquer senha
    - a aplicação vai logar como a primeira entrada na tabela de usuários Users
    - que é o Admin
- Exploração 2
  - Logar na aplicação com email `admin@juice-sh.op'--` e qualquer senha
    - 2a. parte da query SQL é comentada e a senha não é verificada
- Exploração 3
  - Logar na aplicação com email `admin@juice-sh.op` se senha `admin123`
  - Senha default descoberta por tentativa e erro ou busca exaustiva com dicionário
- Exploração 4
  - Explorando a API da aplicação
    - Logar na aplicação como user1@juice
    - no OWASP Zap Proxy do JuiceShop
      - interceptar e reenviar a requisição de products/search

`GET http://try-juice-shop.herokuapp.com/rest/products/search?q=`

      - botão direito na requisição do histórico
      - Na tela de reenvio, substituir `q=` por `q='))--`
        - Por que não `q='--` ?
      - Spoiler alert! A requisição products/search sofre de SQLi!
      - Na tela de reenvio, substituir `q=` pelo seguinte

```
')) UNION SELECT id, email, password, '4', '5', '6', '7', '8', '9' FROM
Users--
```

      - Reenviar a requisição injetada!
      - Pergunta 1: Como sei que o nome da tabela é Users?
        - ***Educated guess***! na API REST, serviços = tabelas?!
      - Pergunta 2: Como sei a quantidade de campos?
        - Tentativa e erro...

- Descobrindo a senha dos usuários
  - Na resposta da requisição, procurar pelo user1@juice (botão direito, buscar)
  - A senha desse usuário está no campo description
    - por exemplo, 24c9e15e52afc47c225b757e7bee1f9d
    - No website https://crackstation.net/
  - Digitar o hash e apertar o botão CrackHashes
  - deu um match perfeito com md5 e user1
    - Descobrimos o algoritmo de hash usado na criptografia da senha!
  - Repetindo o processo para o usuário admin
    - O hash é 0192023a7bbd73250516f069df18b500
    - O website https://crackstation.net/ diz que a senha é admin123
  - Tarefa: Descobrir a senha dos outros administradores!

API 3: 2019 Excessive Data Exposure

- Logar na aplicação com email `' or 1=1--` e qualquer senha
  - a aplicação vai logar como a primeira entrada na tabela de usuários Users
  - que é o Admin

- Em DevTools do browser
  - Visualizar o arquivo main-es2015.js
    - Usar ***Pretty printing*** ou botão ***{}***
    - Buscar (menu três pontos verticais → buscar) pela palavra *admin.*
      - *Existe um path administration*
    - Buscar (menu três pontos verticais → buscar) pela palavra *user.*
      - *Existe uma api rest user*

  - O resultado da busca dá os serviços administrativos da loja

- Na aplicação Juice Shop com OWASP Zap Proxy
  - Logar como admin
    - Logar na aplicação com email `' or 1=1--` e qualquer senha
  - Acessar http://localhost:3000/#/administration
    - aparece uma lista de usuários
  - clicar em qualquer um dos usuários da lista (ícone de olho)
- No OWASP ZAP
  - Verificar no histórico de requisições que a API Users foi ativada
    - `p.ex., http://localhost:3000/api/Users/2`
    - o que significa o 2?
  - Reenviar esta requisição com outros parâmetros
    - por exemplo, 3, 5, 8
      - devolve o Json do usuário correspondente
    - sem parâmetros
      - Esta opção devolve o json de toda a base de dados de usuário!

EXTRAS do OWASP ZAP

Attack Mode e geração de relatórios

No OWASP, ativar a opção de menu Editar/Edit → submenu Modo ZAP → Attack Mode.

O modo de ataque do OWASP ZAP é usado para que a ferramenta realize explorações simples das vulnerabilidades à medida que o testador navega pela aplicação.

- O OWASP ZAP aprende a navegação e adapta seus ataques para a navegação aprendida.
- Para ser efetivo, o attack mode precisa que o testador passe pelo máximo de caminhos da aplicação.
    - Fazer a exploração manual da aplicação
- Quando terminar de navegar pela aplicação, faça uma varredura ativa da aplicação.
    - No frame de sites, selecionar a url da aplicação, clicar com o botão direito do mouse e selecionar a opção Ataque/Attack → Active Scan/Varredura Ativa.
    - A varredura ativa pode demorar vários minutos ou até algumas horas.
- Ao final da varredura ativa, gere um relatório, por exemplo, em HTML
    - No menu Relaórios/Reports → Gerar relatório em HTML...
- Alguns ataques podem ser destrutivos. Use com moderação, com autorização e apenas em ambiente de teste.

Fuzzing e bruteforce

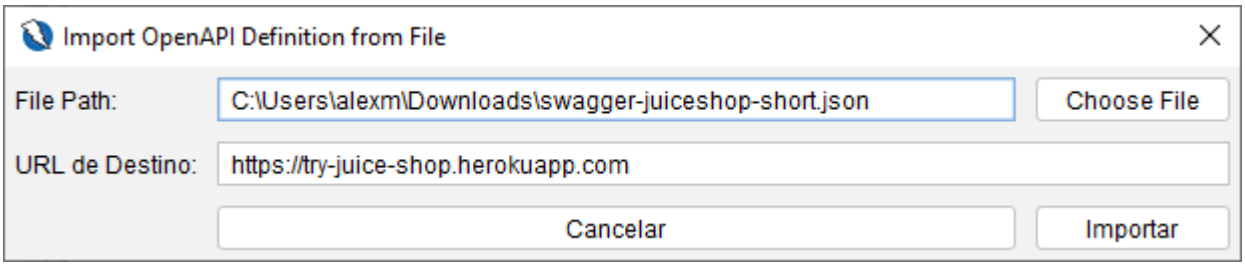
No bWAPP usando ZAP proxy

- Tentar um login sem sucesso;
- Interceptar a requisição de login no histórico do ZAP;
- Clicar com o botão direito, selecionar a opção Attack/Ataque → Fuzz..
- Na Janela do Fuzzer, Fuzzing locations, selecionar o valor de senha digitada
    - Clicar no botão adicionar (para adicionar payloads)
- Na janela de payloads, clicar no botão Adicionar
    - Na janela de adição de payloads
        - Selecionar *Strings*
        - Digitar na caixa de texto as strings de payload da força bruta da senha, por exemplo:
            - bag
            - beg
            - big
            - bog
            - bug
        - Clicar no botão adicionar
    - Clicar no botão OK
- Clicar no botão Start Fuzzer
- O frame do Fuzzer aparece na barra de tarefas
- Refazer o exercício com uma opção diferente de *Strings.* Por exemplo, a opção de *File Fuzzers → Injection.*
    - Este fuzzing pode demorar vários minutos ou até algumas horas.
    - Alguns ataques podem ser destrutivos. Use com moderação, com autorização e apenas em ambiente de teste.

Automated Scan de APIs do Juice Shop

No OWASP, ativar a opção de menu ***Import*** → submenu ***Import an OpenAPI Definition ...***.

- Escolher a opção de arquivo ou de URL
    - Deve apontar para o json com a definição da API
- Na opção de arquivo
    - entrar com o caminho do arquivo json
    - entrar com a url da aplicação
    - Clicar no botão ***Importar.***
        - Verificar se a importação é bem sucedida
- No quadro de Sites do ZAP
    - Selecionar a API importada
    - Realizar um Spider ou vareedura ativa na API



OBSERVAÇÃO: O OWASP Juice Shop não tem uma api documentada para swagger.
Uma documentação extra-oficial, incompleta e possivelmente defeituosa pode ser achada a seguir.

https://github.com/apox64/RestSec/blob/master/restsec-samples/src/main/resources/docs_swagger/swagger-juiceshop-short.json

ZAP como proxy do Postman

Este exercício usa uma instalação local do Postman e a configuração de proxy manual do ZAP.

No OWASP ZAP

- Acessar o menu Ferramentas → Opções → Proxies Locais
- No quadro do Proxy local
    - Endereço: localhost
    - Porta: 8081 (isto evita conflitos com outros usuários da porta 8080)

No PostMan

- Clicar o ícone de engrenagem → Settings → Proxy.
    - Selecionar a opção ***Add Custom Proxy Configuration***
        - Proxy type: HTTP e HTTPS
        - Proxy Server: localhost :8081 *(mesma configuração do ZAP!)*
        - Opcionalmente, fazer as configurações de login/senha