SÉRIE WEBAPP PARA PENTESTER E APPSEC

NMAP

CRIANDO SCRIPTS PARA IDENTIFICAR ORACLE VULNERÁVEIS

O MANUAL PASSO A PASSO

de como criar seus próprios scripts para identificar e tratar vulnerabilidades

FERNANDO MENGALI

SUMÁRIO

INTRODUÇÃO

Muitas vezes, profissionais de segurança ofensiva precisam varrer sua rede em busca de uma vulnerabilidade recém-descoberto ou 0 day (zero day), mas as ferramentas comerciais de scannings dinâmicos não conseguem adicionar com agilidade ou a tempo uma assinatura de verificação de servidores vulneráveis e exploráveis a 0 days.

Quando isso acontece, e o crescente aumento de ataques, colocam os servidores em risco.
Com NMAP o profissional pode criar rapidamente e aproveitar as features de rede do NMAP, automatizando o processo de identificação de servidores com vulnerabilidades do tipo 0 days.
O exemplo acima não está limitado a vulnerabilidades do tipo 0 days, mas há vários tipos de outras vulnerabilidades e diferentes tecnologias. Mas o que você precisa para aprender a criar um script do NMAP?
A resposta é simples e a criação do script também, você apenas precisa saber como funciona o processo para identificar a vulnerabilidade no host remoto, ou seja, requisição e validação da resposta do servidor... e um pouquinho de LUA, isso você aprenderá lendo esse artigo.

Nesse artigo, desenvolveremos um simples script para o NMAP, o intuito é ganharmos o banner do servidor Oracle e associarmos a um exploit, para esse artigo.
O que apreenderemos nesse artigo será fundamental para você criar scripts para o NMAP, com diversas funcionalidades de automatização que ajudará a identificar vulnerabilidades durantes os testes de invasões nos alvos.
Utilizaremos a linguagem LUA que foi padronizada para a criação de scripts do NMAP.

Primeiro, iremos apresentar o processo de identificação de banners com parametrizações especificadas do NMAP e depois como podemos utilizar scripts para automatizar nosso processo de identificação de vulnerabilidades.
Futuramente, você aprenderá como desenvolver um **script em LUA**, e utilizá-lo na rotina do seu trabalho de pentest.

Esse artigo não apresenta técnicas avançadas para o desenvolvimento dos scripts em LUA.
Para a elaboração desse artigo, utilizamos conceitos básicos, mas eficiente para identificar falhas de segurança, seja para um alvo específico ou vários alvos.
O principal foco nesse artigo, será identificar o banner de um serviço Oracle, consequentemente associar a versão do banner a um exploit.

O conteúdo apresentado nesse artigo para identificar um banner de serviço Oracle e associá-lo a uma vulnerabilidade não é o equivalente as grandes ferramentas de mercado, como por exemplo, ferramentas que atendem a metodologia DAST (Dynamic application security testing). Com o tempo você poderá continuar aperfeiçoando suas técnicas de desenvolvimento de scripts para a identificação de vulnerabilidades.

Não vamos ensinar a desenvolver algoritmos sofisticados que são utilizadas pelas ferramentas de análise dinâmica disponíveis comercialmente, mas compartilharemos informações suficientes para você começar a criar seus primeiros scripts e identificar vulnerabilidades.

Uma informação muito importante, varreduras com NMAP não são indicadas em sistemas de varreduras multithreadings, pois a ferramenta carece de muito tempo para validar as requisições conduzidas sobre um alvo e entregar um resultado consistente.

Com o tempo você poderá continuar aperfeiçoando suas técnicas de desenvolvimento de scripts para a identificação de vulnerabilidades.

2.0 PRÉ-REQUISITOS

Recomendamos a criação de dois ambientes, um ambiente com um servidor Oracle disponível ou acessível por um usuário.
Após criar o ambiente com Windows Server, podemos utilizar uma máquina com a distribuição Kali Linux (pode ser sua máquina):

- **Download do Kali Linux:**
  https://www.kali.org/get-kali/#kali-installer-images/

- **Download do Windows Server:**
  https://archive.org/download/ver_202Ppbdjkabd10806/Windows_2000_Professional_PT-PT.iso
- **GlassFish Server Open Source Edition 4.1**
  https://download.oracle.com/glassfish/4.1/release/index.html

- **Download do VMWARE:**
  https://customerconnect.vmware.com/en/downloads/info/slug/desktop_end_user_computing/vmware_workstation_pro/15_0

Após fazer download de cada ferramenta, apenas faça o simples processo de instalação e configurações necessárias para funcionamento.

3.0 CRIANDO O LABORATÓRIO/AMBIENTE

Nessa seção instalaremos um servidor Oracle vulnerável numa máquina Windows Server.

Como o processo de instalação é muito simples, não abordaremos o processo de instalação.

Vamos considerar que você concluiu a instalação do Windows Server e do GlassFish Server Open Source Edition 4.1.

Se desejar acessar somente a seção sobre o desenvolvimento do script em LUA para o NMAP, acesse a **seção 5**.

4.0 ACESSANDO O LABORATÓRIO

Acesse a máquina com Windows Server e inicie o GlassFish Server Open Source Edition 4.1.

Com o servidor operando normalmente, vamos iniciar o processo de construção do nosso script em LUA para o NMAP.

Para realizar o teste é obrigatório instalar uma distribuição Kali Linux, se desejar reproduzir o laboratório.

5.0 CRIAÇÃO DO SCRIPT

Nessa seção vamos criar o primeiro script que fará uma verificação de banner de serviço do Oracle que instalamos.

Se você não conhece a linguagem LUA, não se preocupe, porque vamos apresentar cada parte do script e como funciona.

5.1 AS LIBRARIES

Quando criamos um script para o NMAP, precisamos pensar nas libraries para que o script consiga executar suas funções.

Nosso script precisará utilizar três tipos de libraries:

5.1.1 NMAP

A library NMAP será importante, pois vamos utilizar esse módulo na coleta de informação sobre o nosso host remoto ou nosso alvo.

Existe diversas funções que permitem manipularmos nossas conexões, quando sondamos ou varremos um servidor alvo.

5.1.2 COMM

Essa library trabalha muito bem, quando precisamos coletar informações como banner do servidor de destino.

5.1.3 STDNSE

Essencial para apresentarmos ou renderizarmos os dados na tela do usuário, além de ser muito útil para debugarmos erros e status de verificações dos serviços no servidor alvo.

5.1.4 SHORTPORT

Nessa biblioteca, podemos utilizar quais serão as portas especificadas para executarmos o script.

Você pode especificar uma única porta ou um conjunto de portas que devem ser definidas para o script executar, caso contrário, as verificações não serão feitas.

Essas serão as quatro principais bibliotecas que vamos utilizar no script de verificação de banners.

Portanto, nosso script terá a seguinte estrutura:

```
local comm = require "comm"
local nmap = require "nmap"
local shortport = require "shortport"
```

Depois de entendermos de forma resumida a utilidade de cada biblioteca e como elas serão fundamentais em nossa verificação de banners, vamos estruturar a apresentação das informações de como usar o script.

Vamos começar entendendo como funciona o "***description***". Nesse contexto, vamos descrever qual a utilidade e finalidade do script.

É importante ressaltar, que as informações ou "***description***", deverão ser adicionadas após definirmos as bibliotecas que serão necessárias para o script.

```
description = [[
Script para verificação de banners em servidores oracle GlassFish.
]]
```

Após entendermos o “***description***”, vamos definir os direitos autorais do script, ou seja, quem foi o criador do script.

```
author = "Fernando Mengali"
license = "Same as Nmap--See https://nmap.org/book/man-legal.html"
categories = {"discovery", "safe"}
```

No exemplo, acima temos três variáveis “***author***”, “***license***” e “***categories***”.

A variável “**author**” é onde será mencionado o autor do script, nesse exemplo, adicionei meu nome, mas você precisará adicionar o seu nome.

Na variável “***license***”, definimos os direitos autorias do NMAP.
Na variável “***categories***”, definimos as categorias que o script atende, nesse caso, descoberta de hosts alvos.

```
portrule = shortport.port_or_service({4848}, {“oracle-tns”})
```

O bloco acima, define que o script será executado, apenas nas portas 4848 e serviço Oracle. Se a porta 4848 não for definida no input do terminal para o NMAP varrer o alvo, o script não será executado.

```
action = function( host, port )

    local out = grab_banner_oracle(host, port)
    return out

end
```

A variável action, tem como objetivo estabelecer uma conexão com o host alvo, nesse exemplo, passamos o endereço do host e a porta a ser verificada, depois passamos esses dois dados para fazer um banner grab, ou ganhar a versão do servidor.

Agora vamos entender, como a função grab_banner_oracle trabalha e como podemos ganhar a versão do serviço remoto.

```
function grab_banner_oracle(host, port)
local status, response = comm.get_banner(host, port)

 if not status then
    return nil
  end

 if response:match("GlassFish Server Open Source Edition  4.1") then
  return "ORACLE VUNERABLE";
 else
  return "ORACLE NOT VULNERABLE"
 end

end
```

A função grab_banner_oracle, possui a seguinte estrutura:

1º utiliza a função get_banner reservada do NMAP para ganhar o banner do servidor Oracle.

2º Utilizamos a biblioteca comm, juntamente com a função reservada get_banner, passando o endereço do servidor alvo e a porta de destino.

3º Se o status não for definido ou não tenha algum conteúdo, fazemos o retorno.

4º Caso a variável status tenha um conteúdo, seguimos com a execução do script.

5º Com o conteúdo da variável, seguimos para o bloco condicional que verifica se o servidor é vulnerável.
Para considerarmos o servidor como vulnerável, a resposta do servidor deve ter um conteúdo parecido com a string “GlassFish Server Open Source Edition 4.1”.

6º Se a resposta do servidor não conter a string “GlassFish Server Open Source Edition 4.1”, simplesmente retornamos com a menagem: “Not vulnerable”.

Agora, vamos salvar o arquivo no diretório usr/share/nmap/scripts com a extensão lua ou a extensão, nse.

Se você salvar o script com a extensão lua e não nse, o NMAP irá gerar um alerta, mas não se preocupe, ambas as extensões funcionam.

Como podemos observar é muito simples a estrutura do script e pode ser de grande utilidade.

7.0 EXECUTANDO O SCRIPT

Depois de salvar o nosso primeiro script com o nome de banneroracle.lua, vamos executar o NMAP com o argumento --script e especificar nosso script banneroracle.lua com o intuito de capturarmos o banner do servidor Oracle. Você precisa utilizar o comando:

```
nmap -p 4848 --script=banneroracle.lua 192.168.0.10
```

Vejamos o resultado do script em nosso terminal do Kali Linux:

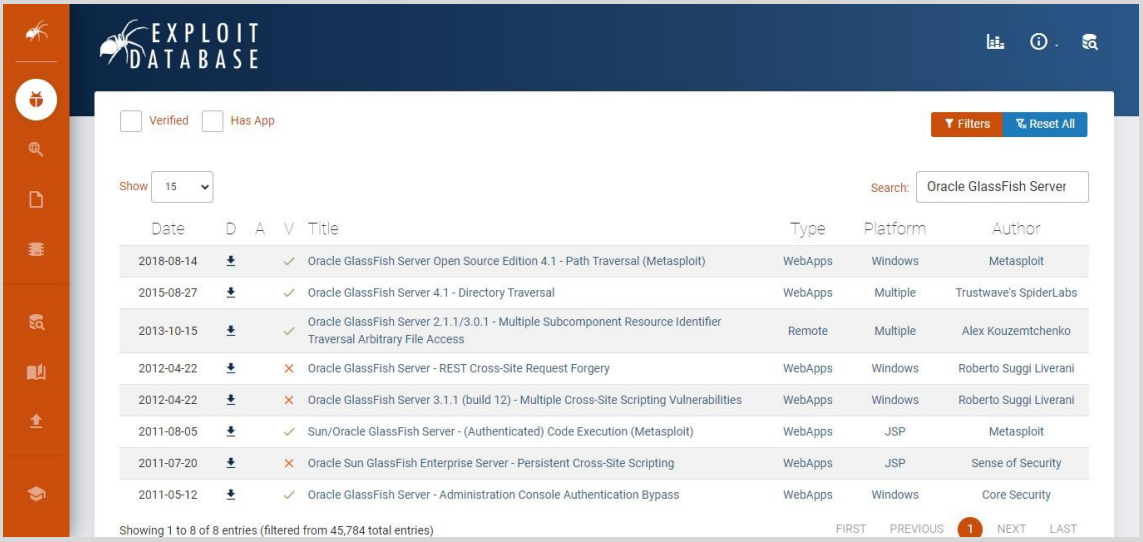
```
nmap -p 4848 --script=banneroracle.lua 192.168.176.131
Starting Nmap 7.94 ( https://nmap.org ) at 2023-11-29 18:10 EST
Nmap scan report for 192.168.176.131
Host is up (0.00039s latency).

PORT   STATE SERVICE
4848/tcp open  oracle
|_banneroracle: VUNERABLE
MAC Address: 00:0C:29:9B:56:97 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.43 seconds
```

8.0 BUSCANDO EXPLOITS

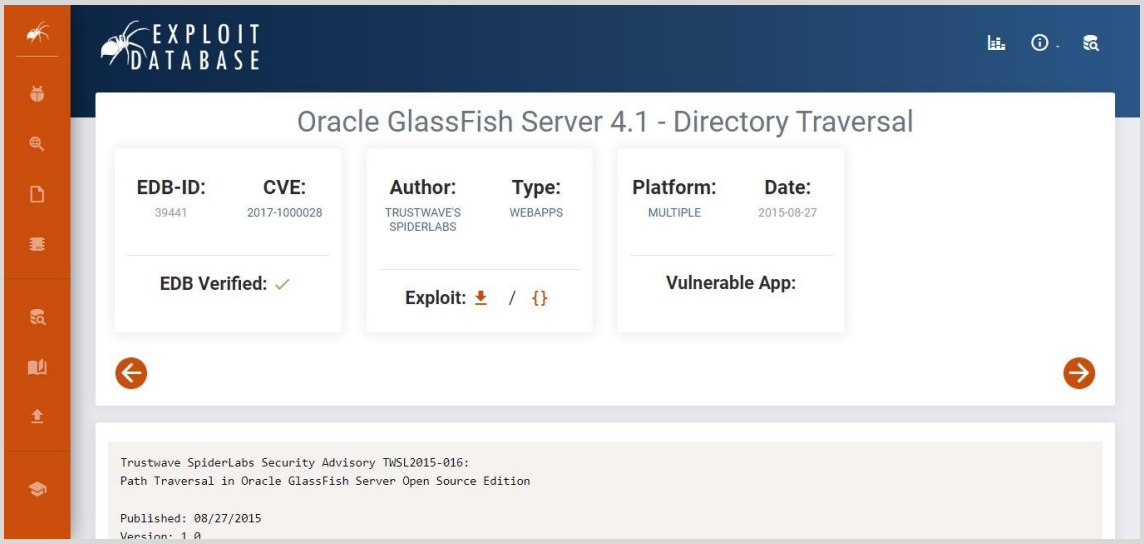
Depois de observarmos o resultado do script, um pesquisador de segurança ou pentester pode buscar por exploits nos repositórios da internet.

Nesse exemplo de demonstração temos um banner que apresenta a versão do servidor Oracle.

Vamos pesquisar se a versão do Oracle Server identificado pelo NMAP é vulnerável e possui algum exploit disponível na internet.



Encontramos 8 exploits, que exploram vulnerabilidades diferentes no servidor Oracle Glassfish.



Um exploit disponível no exploit-db: https://www.exploit-db.com/exploits/39441

Nessa seção, o objetivo foi identificar o banner do servidor Oracle com NMAP e depois apresentar como funciona as buscas por exploits em repositórios da internet.

Como o intuito é didático e o objetivo sé apresentarmos o processo de codificação do script do NMAP, não vamos seguir com a exploração da vulnerabilidade no serviço de Oracle.

9.0 SCRIPT COMPLETO

Nessa seção, disponibilizamos a estrutura completo do script que você pode testar.

```
local comm = require "comm"
local nmap = require "nmap"
local shortport = require "shortport"

description = [[
 Script para verificação de banners em servidores oracle GlassFish.
 ]]


 author = "Fernando Mengali"
 license = "Same as Nmap--See https://nmap.org/book/man-legal.html"
 categories = {"discovery", "safe"}

 portrule = shortport.port_or_service({4848}, {“oracle-tns”})

action = function( host, port )

  local out = grab_banner_oracle(host, port)
  return out

end

function grab_banner_oracle(host, port)
  -- Did the service engine already do the hard work?

  local status, response = comm.get_banner(host, port)

   if not status then
      return nil
    end


   if response:match("GlassFish Server Open Source Edition  4.1") then
    return "VUNERABLE";
   else
    return "NOT VULNERABLE"
   end

end
```

10.0 AUTOMATIZANDO O SCRIPT

Aproveitando o suporte para Shell Script do próprio Kali Linux, podemos criar um script de automatização que utiliza o NMAP e o script banneroracle.lua para buscar apenas os servidores na minha rede que possuem algum tipo de versionamento de banner e que represente um perigo.

```
#!/bin/bash

hosts="/home/kali/Desktop/listUrl.txt"
destinationFiles="/home/kali/Desktop/targetVulnerable.txt"

# Check file exist
if [ ! -f "$hosts" ]; then
    echo "File not exist!"
    exit 1
fi

keyword=" GlassFish Server Open Source Edition  4.1"

while IFS= read -r lineUrl
do
    scanning=$(nmap -p 4848 -sV --script=banneroracle.lua $linha)

    if [[ "$scanning" == *"$keyword "* ]]; then
        echo "Target $lineUrl vulnerable"
        echo "$lineUrl" >> "$destinationFiles"
    fi

done < "$hosts"
```

Como o código funciona?

A primeira linha especifica o arquivo que contém todos os endereços que serão analisados e a segunda linha o arquivo de destino que armazenará os alvos “vulneráveis”:

```
hosts="/home/kali/Desktop/listUrl.txt"
destinationFiles="/home/kali/Desktop/targetVulnerable.txt"
```

A próxima linha, verifica se o arquivo que contém todos os endereços que analisados existe, se não existe o arquivo script será encerrado.

```
if [ ! -f "$hosts" ]; then
    echo "File not exist!"
    exit 1
fi
```

A próxima linha possui uma variável de controle, que será responsável por comparar os resultados do NMAP. Ou seja, se o resultado do NMAP contém o valor da variável o alvo será considerado vulnerável.

```
keyword=" GlassFish Server Open Source Edition  4.1"
```

Na última parte do código, temos o comando de interações while. Ele será responsável por verificar cada linha do arquivo hosts e depois executar o NMAP para analisar o alvo.

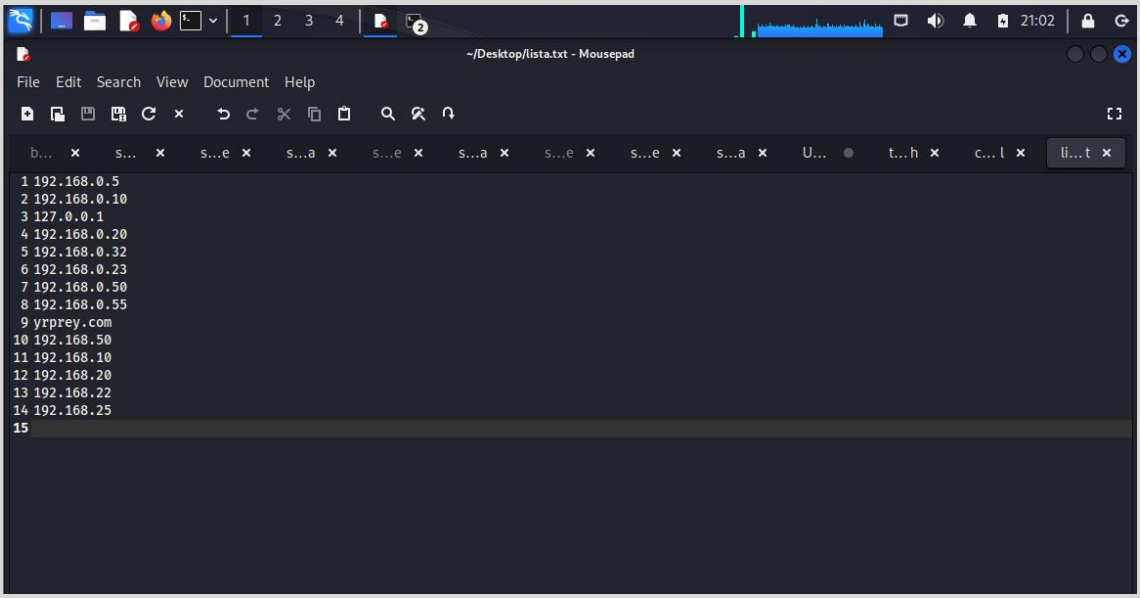
Após o NMAP responder, verificamos se o conteúdo do resultado do NMAP possui algum valor relacionado com o conteúdo da variável "**keyword**", caso a resposta seja positiva, abrimos o arquivo que está localizado em home/kali/Desktop/targetVulnerable.txt e escrevemos o alvo que está vulnerável.

```
while IFS= read -r lineUrl
do
    scanning=$(nmap -p 4848 -sV --script=banneroracle.lua $linha)

    if [[ "$scanning" == *"$keyword "* ]]; then
        echo "Target $lineUrl vulnerable"
        echo "$lineUrl" >> "$destinationFiles"
    fi
done < "$hosts
```

Antes executarmos o script, precisamos criar uma lista com os alvos a serem analisados.

No arquivo Shell Script definimos o arquivo listUrl.txt e a path home/kali/Desktop/ com os nossos alvos ou urls a serem analisadas pelo NMAP e a resposta tratada pelo script do NMAP.

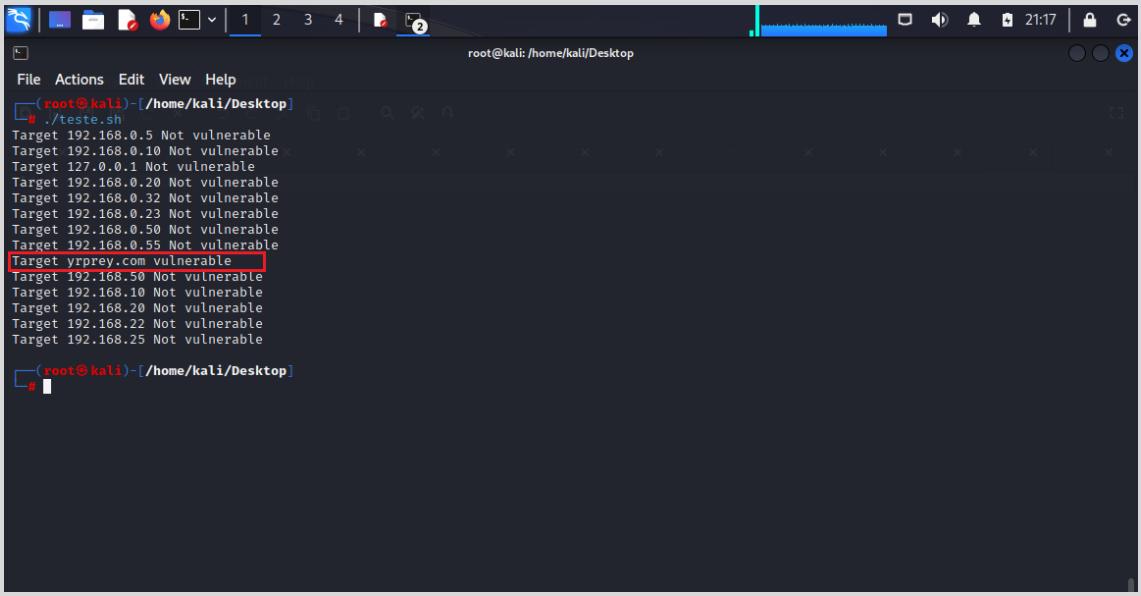
A minha lista possui a seguinte estrutura:



A lista acima, possui os alvos vulneráveis a serem analisados. A linha 9 possui o endereço do framework yrprey.com que estava rodando um Windows Server. O Yrprey é um framework com vulnerabilidade de aplicações, principalmente nas APIs e está disponível para download no endereço: https://owasp.org/yrprey

Quando os targets vulneráveis forem identificados como vulneráveis, cada url será escrita no arquivo targetVulnerable.txt.
Sendo assim, saberemos quais alvos são vulneráveis para serem explorados e corrigido as vulnerabilidades.

Vejamos na prática, como funciona a execução do script no terminal.



Observe que nesse teste, o framework yrprey.com, estava com o serviço Oracle GlassFih vulnerável instalado.

Futuramente com o aparecimento de vulnerabilidades do dia zero ou (Zero Day/0day) você pode utilizar os conhecimentos de desenvolvimento de scripts do NMAP e o uso shellscript para verificar se existe algum servidor em sua rede que esteja vulnerável, portanto, esse script pode auxiliar na identificação de servidores vulneráveis, contribuindo de forma ágil para correção das falhas de segurança e evitando um vazamento de dados, antes que os hackers façam.

11.0 APPLICATION SECURITY

No contexto de Segurança de Aplicações precisamos adotar algumas medidas de segurança, a fim de proteger de futuros ataques, como por exemplo:

1. Atualização dos patches de segurança;
2. Instalação de dispositivos de rede, como IPs, WAF, Firewall etc
3. Instalação de sistemas a nível de sistema operacional, visando a integridade de proteção de sistemas operacionais.
4. Revisão de políticas de segurança, por exemplo, políticas de acesso etc.
5. Pentest regularmente ao sistema alvo
6. Análise de vulnerabilidade contínuo.

Nesse cenário, a recomendação mais relevante é a remoção das informações de banners, principalmente as informações sobre o versionamento do serviço.

E se houver atualizações, atualize. Caso não haja atualizações, busque serviços equivalentes que atendam suas demandas diárias.

E para complementar, siga os 7 itens acima para garantir maior proteção do seu ambiente.

As informações contidas nessa seção, são recomendações padrões, mas uma análise e um estudo profundo do ambiente deve ser realizado para melhores recomendações mais assertivas e precisas.

12.0 SOBRE O AUTOR

Paper criado por Fernando Mengali no dia 14 de março de 2025.

LinkedIn: https://www.linkedin.com/in/fernando-mengali-273504142/