Registro de Consultas DNS em Tempo Real

Gabriel Peruzzi

Abril de 2025

1 - Introdução

1.1 - Motivação

Em um mundo cada vez mais conectado, o monitoramento de redes tornou-se uma das principais ferramentas para garantir a segurança cibernética. Uma das estratégias utilizadas é a análise de consultas DNS, que podem revelar comportamentos anômalos de dispositivos infectados por malware.

Malwares modernos dependem fortemente da comunicação com servidores externos para receber comandos ou exfiltrar dados. Essa comunicação, muitas vezes, inicia-se por meio de requisições DNS para domínios controlados pelos atacantes. Monitorar essas requisições permite detectar rapidamente atividades suspeitas e tomar medidas proativas de contenção.

Neste projeto, desenvolvemos uma aplicação simples em Python que escuta consultas DNS em tempo real e registra o IP de origem das solicitações, proporcionando uma ferramenta útil para estudos de segurança de redes.

1.2 - Sobre a Organização do Projeto

O projeto é constituído por um único arquivo Python denominado dns_logger.py. Ele utiliza a biblioteca padrão socket para criar um servidor que escuta a porta 53 via protocolo UDP. Essa escolha é justificada pelo fato de que consultas DNS geralmente são feitas via UDP por sua leveza e velocidade.

A porta 53 é a porta padrão para serviços DNS. Como ela é uma porta considerada privilegiada (número abaixo de 1024), é necessário executar o programa como administrador/root para que o socket possa ser vinculado corretamente.

2 - Código-Fonte

2.1 - Descrição do Funcionamento do Código

O código desenvolvido possui a função principal de capturar e registrar consultas DNS que chegam até a máquina.

Inicialmente, a biblioteca socket é importada para possibilitar a criação e manipulação de conexões de rede. A função iniciar_sniffer_dns() é responsável por criar um socket UDP (SOCK_DGRAM) e vinculá-lo ao endereço IP 0.0.0.0 na porta 53, o que permite que o programa receba consultas DNS de qualquer dispositivo na rede.

Dentro de um laço infinito, a função recvfrom() aguarda passivamente a chegada de pacotes DNS, capturando tanto os dados enviados quanto o endereço IP do remetente. Cada vez que um pacote é recebido, o endereço IP de origem é exibido no terminal através da função print().

No bloco principal (if __name__ == "__main__"), a função iniciar_sniffer_dns() é chamada dentro de um bloco try-except, permitindo tratar dois tipos de exceções:

- PermissionError: ocorre se o programa não tiver permissões suficientes para abrir a porta 53. - KeyboardInterrupt: permite a finalização segura do programa quando o usuário pressiona Ctrl+C.

Essa organização torna o código robusto e preparado para execuções contínuas em ambientes de rede controlada.

2.2 - Explicação Detalhada do Código

Linha 1

import socket

Importa a biblioteca padrão 'socket', que permite a criação de conexões de rede em baixo nível.

Essa biblioteca é essencial para trabalhar com protocolos como UDP e TCP.

Linhas 2 a 10

```
def iniciar_sniffer_dns():
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    sock.bind(('0.0.0.0', 53))
    print("Sniffer DNS iniciado. Aguardando consultas...")

    while True:
        data, addr = sock.recvfrom(512)
        print(f"Consulta DNS recebida de {addr[0]}")
```

:

- Cria a função principal do sniffer.
- Cria um socket UDP com socket.AF_INET e socket.SOCK_DGRAM.
- Vincula o socket a todas as interfaces IP locais na porta 53.
- Exibe mensagem 'Sniffer DNS iniciado'.
- Entra em um loop infinito aguardando pacotes.
- Ao receber o pacote, imprime o IP de origem.

Linhas 11 a 18

```
if __name__ == "__main__":
    try:
        iniciar_sniffer_dns()
    except PermissionError:
        print("Permissão negada: execute como administrador/root.")
    except KeyboardInterrupt:
        print("\nSniffer encerrado.")
```

- Garante que o script só execute diretamente.
- Tenta iniciar o sniffer DNS.
- Trata PermissionError: falta de permissão para usar porta 53.
- Trata KeyboardInterrupt: encerra o programa com uma mensagem amigável.

Resumo Geral O programa escuta consultas DNS na rede e registra o IP de origem.

Utiliza sockets UDP e trata exceções comuns para garantir uma execução estável.

2.3 - Código-Fonte Completo

Abaixo está a imagem contendo o código completo utilizado no programa:

```python
import socket

def iniciar_sniffer_dns():
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    sock.bind(('0.0.0.0', 53))
    print("Sniffer DNS iniciado. Aguardando consultas...")

    while True:
        data, addr = sock.recvfrom(512)
        print(f"Consulta DNS recebida de {addr[0]}")

if __name__ == "__main__":
    try:
        iniciar_sniffer_dns()
    except PermissionError:
        print("PermissÃ£o negada: execute como administrador/root.")
    except KeyboardInterrupt:
        print("\nSniffer encerrado.")
```

Figura 1 - Código-fonte completo em Python.

3 - Análise de Complexidade

O projeto opera de forma altamente eficiente.

Cada iteração do laço principal apenas aguarda a chegada de um pacote UDP, operação realizada com baixa utilização de CPU, caracterizando uma complexidade de tempo constante O(1) para cada consulta recebida.

Considerando o tempo total de execução, a complexidade do programa é O(n), onde n representa o número de pacotes DNS capturados.

Além disso, como o programa apenas exibe IPs no terminal e não realiza processamento profundo de dados, o consumo de memória também se mantém constante durante a execução.

4 - Possíveis Melhorias Futuras

- **Decodificação de Pacotes DNS:** Permitir a extração do domínio consultado, aumentando a precisão da análise.
- **Armazenamento de Logs:** Salvar as consultas recebidas em arquivos para auditoria e estudo posterior.
- **Filtros Personalizados:** Implementar filtros para capturar somente determinados tipos de consultas ou domínios suspeitos.
- **Suporte a IPv6:** Expandir a capacidade do sniffer para funcionar em redes modernas.
- **Interface Gráfica:** Criar uma aplicação GUI para exibir as consultas em tempo real de maneira visual e acessível.

Essas melhorias ampliariam significativamente o potencial da ferramenta, possibilitando seu uso em ambientes corporativos ou acadêmicos.

5 - Conclusão

Este trabalho apresentou o desenvolvimento de uma ferramenta simples e eficiente para captura de consultas DNS em tempo real. Apesar da simplicidade de sua implementação, o programa é funcional e cumpre seu objetivo de auxiliar na análise de tráfego DNS.

O projeto reforça a importância de entender protocolos básicos de rede e suas aplicações na segurança cibernética. Como trabalho futuro, vislumbram-se expansões que tornarão a ferramenta ainda mais robusta, integrando análise de domínios, salvamento de logs e interface gráfica.

O uso de Python como linguagem de desenvolvimento garantiu rapidez na implementação e fácil compreensão do código, fatores relevantes em projetos educacionais e prototipagem.