Projeto e implementação de um filtro DNS em Python utilizando as bibliotecas Scapy, PyDivert e DNSPython

Carlos Gabriel de Araujo Gewehr[1]

[1]Engenharia de Computação – Universidade Federal de Santa Maria (UFSM)

carlos.gewehr@ecomp.ufsm.br

Abstract. *In this paper an implementation of a DNS filter targeting Windows platforms is discussed, along with other additional functionalities aiming usability and performance assessments through external tools.*

Resumo. *Neste relatório é discutida a implementação de um filtro DNS para plataformas Windows, assim como funcionalidades adicionais permitindo fácil usabilidade e avaliações de performance a partir de ferramentas externas.*

1. Introdução

Dito de forma simples, um filtro de DNS consiste em uma ferramenta que intercepta as requisições de DNS realizadas por uma máquina e encaminhá-las ao seu destino apenas aquelas feitas a domínios previamente estabelecidos, seja através de uma *whitelist*, onde apenas as requisições associadas a domínios cadastrados são encaminhadas; ou a uma *blacklist*, onde apenas as requisições associadas a domínios cadastrados não são encaminhadas. Para este trabalho, a metodologia escolhida foi a de *blacklist*.

2. Metodologia

O funcionamento de um filtro DNS em muito se assemelha com um ataque de *DNS spoofing*, da área de segurança de redes. Em tal, um agente malicioso se coloca entre cliente e servidor, interceptando requisições associadas a domínios de interesse a tal, e então fabricando (portanto, *spoofing*) respostas não-autênticas a estas requisições. Através destas respostas fabricadas, o cliente é direcionado a outro servidor, não correspondente ao real servidor associado a suas requisições, de acordo com os interesses deste agente atacante.

É possível então afirmar que um filtro DNS pode ser interpretado como uma parte de um ataque de *DNS spoofing*, pois neste estão abarcadas todas as funcionalidades necessárias a aquele. Destas funcionalidades, identifica-se duas essenciais: Agir sobre a interface de rede, interceptando pacotes DNS, encaminhando-os ou não, de acordo; e Decodificar e modificar estes pacotes, de modo a tomar uma decisão quanto a seu encaminhamento.

São facilmente encontradas referências de código aberto para a implementação de ataques de *DNS spoofing*, sendo citadas na seção de Bibliografia as de maior proveito ao leitor. Tais referências em sua totalidade referem-se a implementações voltadas a sistemas Unix, tomando proveito da fácil definição de regras de encaminhamento a nível de usuário através do comando *iptables*.

Tal comando não é disponível em sistemas Windows, o que serve em parte como motivação para este trabalho. Para tal, optou-se pelo uso da biblioteca PyDivert,

uma interface em Python para o *driver* de código aberto WinDivert, que consegue prover a primeira das funcionalidades anteriormente mencionadas (“Agir sobre a interface de rede, interceptando pacotes DNS, encaminhando-os ou não, de acordo”).

Com esta biblioteca, um *handle* é aberto na interface de rede padrão, interceptando pacotes com a porta UDP 53 como sua porta de entrada ou de saída (porta *well-known* para o protocolo DNS).

A segunda funcionalidade essencial é obtida através das bibliotecas Scapy e DNSPython (“Decodificar e modificar estes pacotes, de modo a tomar uma decisão quanto a seu encaminhamento”). Com o uso destas, é possível a manipulação dos pacotes DNS interceptados, de modo a extrair destes o domínio associado a uma requisição, e compará-lo com os domínios contidos na *blacklist*.

Adicionalmente, através da biblioteca Scapy é realizada a confecção de pacotes de resposta às requisições bloqueadas. Enquanto que em um ataque de *DNS spoofing* esta resposta seria uma resposta valida com o endereço de interesse ao agente atacante, no filtro DNS está é simplesmente uma mensagem informando ao cliente que o endereço requisitado não foi encontrado. (No futuro é interessante implementar uma abordagem mais similar ao *DNS spoofing*, onde o endereço da mensagem de resposta aponta o cliente para uma página que explicita que sua requisição ao domínio de interesse foi bloqueada).

Estes novos pacotes de resposta são adicionados à interface de rede por meio do *handle* previamente aberto com a biblioteca PyDivert e então direcionados a suas aplicações de destino.

Das funcionalidades adicionais mencionadas na Introdução, estas foram a de geração de um *log*, relatando em um arquivo de texto os eventos relevantes que ocorrem ao longo do funcionamento do filtro, assim como *timestamps* associados a cada evento, para análises de performance; além de *flags* para verbosidade, imprimindo no console as mesmas mensagens escritas no arquivo de *log*; e para realização de *flush* no *cache* de DNS na inicialização do filtro, de modo a invalidar quaisquer endereços associados a domínios na *blacklist* e assegurar o correto funcionamento do filtro.

3. Avaliação Experimental

Ao inicializar o filtro em um console PowerShell com o *flag* de verbosidade ativo, as seguintes mensagens deve ser exibidas (lembrando que estas também serão escritas no *log* de eventos):

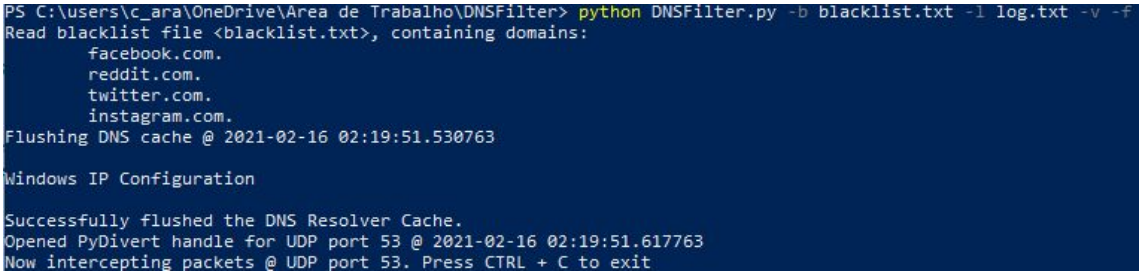

Figura 1. Mensagens de inicialização do filtro

Submetendo o filtro a tráfego real, os seguintes eventos são observados:

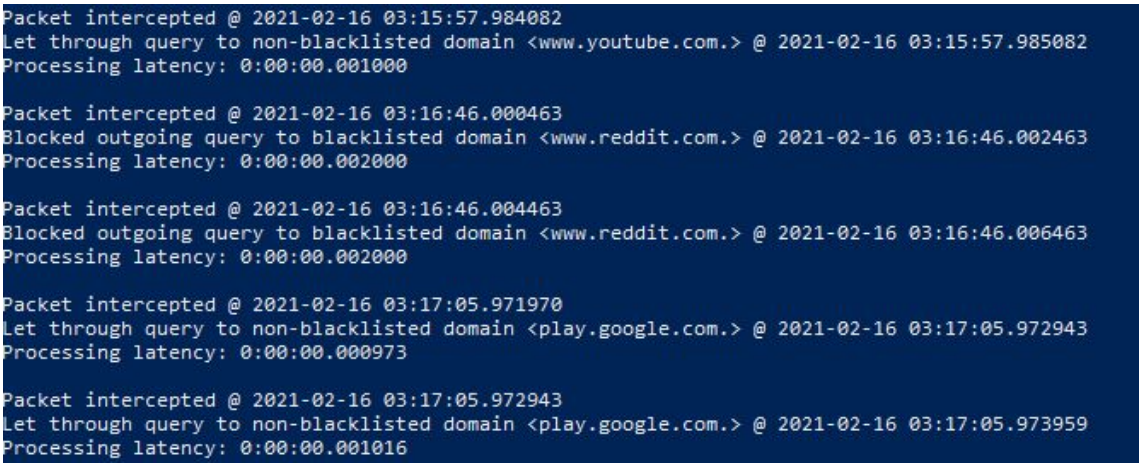

Figura 2. Mensagens de eventos do filtro

É possível perceber 5 requisições DNS associadas a 3 dominios diferentes: "www.youtube.com", "www.reddit.com" e "play.google.com". Destes, somente o domínio "www.reddit.com" está presente na *blacklist*, e portanto, somente este é bloqueado, e os demais, encaminhados normalmente. É notável também a diferença de latência de processamento para ambos os casos, com valores de 2000 microsegundos para o domínio bloqueado, e, um máximo de 1016 microsegundos para os domínios não-bloqueados.

Ao finalizar a execução do filtro, as seguintes mensagens são exibidas:

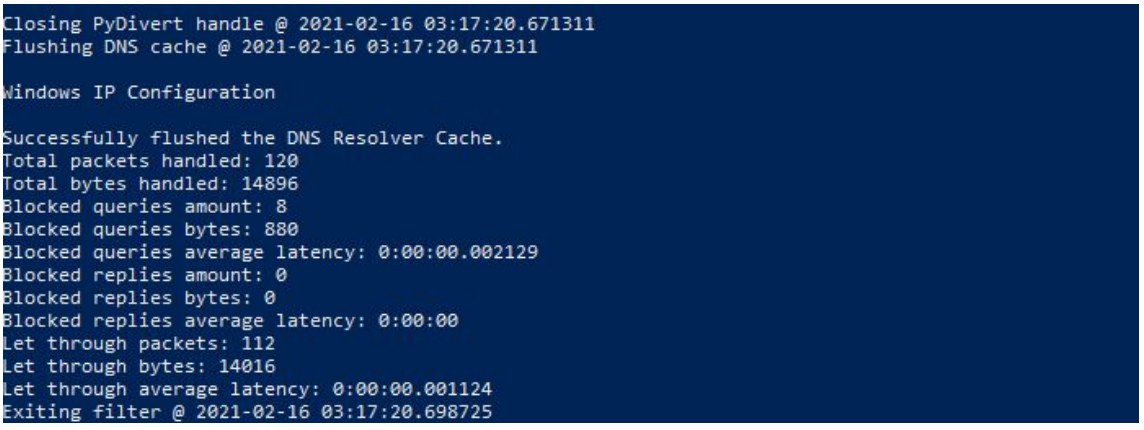

Figura 3. Mensagens de saída do filtro

São informados ao usuário os eventos de fechada do *handle* PyDivert e o *flush* do *cache* de DNS, invalidando as respostas inseridas pelo filtro. Mais abaixo, valores relevantes à execução do filtro são explicitados, sendo eles a quantidade de pacotes, quantidade de bytes e latência média de processamento, para requisições bloqueadas, respostas interceptadas e requisições não-bloqueadas, respectivamente.

4. Conclusão

A implementação do filtro DNS foi bem-sucedida, apesar dos desafios envolvidos em seu desenvolvimento. Seu correto funcionamento foi provado através de uma breve exposição quanto a performance do filtro em um cenário de tráfego real. Ademais, uma avaliação sistemática considerando tráfego sintético, apesar de interessante, não foi efetuada, apesar da implementação de recursos que tornem possível, fácil e conveniente tal curso de ação, através de uma simples análise dos logs gerados pelo filtro.

5. Bibliografia

Implementação de *DNS spoofing* 1:
https://www.cs.dartmouth.edu/~sergey/netreads/local/reliable-dns-spoofing-with-python-scapy-nfqueue.html

Implementação de *DNS spoofing* 2:
https://www.thepythoncode.com/code/make-dns-spoof-python

Biblioteca Scapy: https://scapy.net/

Biblioteca PyDivert: https://pypi.org/project/pydivert/

Biblioteca DNSPython: https://pypi.org/project/dnspython/