Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Ataque Distribuído de Negação de Serviço por Reflexão Amplificada usando Network Time Protocol

Alexander André de Souza Vieira - 13/0039853

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia de Computação

Orientador
Prof. Dr. João José Costa Gondim - 139751

Brasília
2019

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Ataque Distribuído de Negação de Serviço por Reflexão Amplificada usando Network Time Protocol

Alexander André de Souza Vieira - 13/0039853

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia de Computação

Prof. Dr. João José Costa Gondim - 139751 (Orientador)
CIC/UnB

Prof. Dr. Marcos Fagundes Caetano
CIC/UnB

Prof. Dr. Marcelo Grandi Mandelli
CIC/UnB

Prof. Dr. José Edil Guimarães de Medeiros
Coordenador do Curso de Engenharia de Computação

Brasília, 22 de Março de 2019

Dedicatória

Dedico este trabalho à Maria Aparecida, Alexandra Andréia, Antônia Taiza, Elydia Guimarães e José Zacarias.

Agradecimentos

Agradeço à minha mãe Maria Aparecida, que sempre me proporcionou todas as condições e ferramentas, além de todo o apoio para que eu pudesse trilhar o meu caminho, me incentivando sempre a buscar alcançar os meus próprios objetivos. Agradeço à minha irmã Alexandra Andréia, por sempre ser contrária às minhas opiniões, me provocando e promovendo boas brincadeiras e discussões. Agradeço à Antônia Taiza por todo o apoio nos momentos mais difíceis, e por sempre me apoiar nas tomadas das minhas decisões. Agradeço aos meus familiares pelas palavras de apoio e encorajamento. A todos vocês, meu mais singelo muito obrigado!

Agradeço à todos os meus amigos que diretamente ou indiretamente me auxiliaram em algum momento na vida acadêmica. Agradeço em especial aos meus amigos Gabriel Ferreira, Rodrigo Saldanha, Felipe Ofugi e Guilherme Manno. Em vários momentos ao longo de todos esses anos vocês fizeram os meus dias mais felizes, e proporcionaram vários momentos de descontração, os quais já sinto saudades.

Agradeço ao meu professor orientador João Gondim, por partilhar seu tempo e conhecimento, bem como todos os professores que tive ao longo da graduação. O trabalho de vocês foi essencial para o meu desenvolvimento.

Agradeço à Universidade de Brasília, por me proporcionar esses longos e inesquecíveis anos de graduação.

Agradeço à todos os brasileiros e brasileiras, que direta ou indiretamente financiaram a minha graduação.

Resumo

Durante os últimos anos pudemos presenciar vários casos de Ataques de Negação de Serviço (DoS). Devido à sua grande capacidade, os atacantes se utilizam desse artifício para os mais diversos fins, como ganhar dinheiro através de extorsão ou até mesmo mascarar outros ataques. Em razão do seu grande poder, alcance e eficácia, se faz necessário o estudo do ataque para que possamos ter uma melhor compreensão sobre o mesmo. Neste trabalho, estudaremos as várias opções de ataques distribuídos de negação de serviço, DDoS, mas daremos ênfase aos ataques por amplificação e reflexão, analisando o potencial da exploração da vulnerabilidade presente no *Network Time Protocol (NTP)*, um protocolo para sincronização de relógios de dispositivos de uma rede a partir de referências de tempo confiáveis. Será apresentada a ferramenta utilizada para a realização do teste de funcionalidade e performance, e consequentemente, serão analisados os dados obtidos através desse teste, para verificar a viabilidade do uso do protocolo para esse tipo de ataque.

Palavras-chave: Negação de serviço, amplificação, reflexão, vulnerabilidade, NTP, negação de serviço distribuída

Abstract

During the past years we were able to witness a large amount of *Denial of Service* (DoS) attacks. Because of its great capacity, the attackers use this artifice for different purposes such as making money through extortion or even masking other attacks. In reason of its big power, reach and effectiveness, it is necessary that we study the attack, so that we can have a better comprehension about it. In this work, we will study the various options of *Distributed Denial of Service* (DDoS) attacks, but we will give emphasis to the attacks by reflection and amplification, analyzing the potential of the exploitation of the vulnerability present in the NTP, a protocol used for synchronization of clocks of network devices from trusted time references. The tool used to perform the test of functionality and performance will be presented, and consequently, will be analyzed the data obtained through that test, to check the viability of the use of the protocol for this type of attack.

Keywords: denial of service, amplification, reflection, vulnerability, NTP, distributed denial of service

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

CISA *Cybersecurity and Infrastructure Security Agency.*

DDoS *Distributed Denial of Service.*

DNS *Domain Name System.*

DoS *Denial of Service.*

DTSS *Digital Time Synchronization Service.*

GPS *Global Position System.*

ICMP *Internet Control Message Protocol.*

IP *Internet Protocol.*

NTP *Network Time Protocol.*

RFC *Request For Comments.*

SNMP *Simple Network Management Protocol.*

SNTP *Simple Network Protocol.*

SOAP *Simple Object Access Protocol.*

SSDP *Simple Service Discovery Protocol.*

TCP *Transmission Control Protocol.*

UDP *User Datagram Protocol.*

UPnP *Universal Plug and Play.*

Capítulo 1

Introdução

Ao longo dos anos, a evolução da Internet permitiu várias melhorias na vida dos seres humanos. Acesso rápido a informações, uso de serviços de *streaming*, automações residenciais, entre outros. Contudo, além das coisas boas, a Internet proveu um ambiente onde usuários mal intencionados podem se aproveitar da inocência das pessoas, bem como de falhas presentes em diversos pontos ao longo da Internet. Ao explorarem essas falhas, os usuários mal intencionados podem fazer com que usuários legítimos sejam alvos de ataques; ou até mesmo sejam utilizados como ferramentas de ataque, sem que estejam a par do risco aos quais estão expostos, ou do papel que estão desempenhando.

1.1 Motivação

Os serviços disponíveis na Internet passaram a se tornar essenciais para as pessoas. Hoje, o acesso rápido às informações bem como a otimização de tempo e de funções no dia a dia, fazem com que a grande maioria da população esteja a maior parte do tempo conectada a dispositivos que possuem acesso a Internet.

O acesso a um extrato bancário, uma reunião à distância, uma transmissão ao vivo de um evento, se tornaram eventos do cotidiano. Contudo, esses serviços disponibilizados podem nem sempre estarem disponíveis. Os mesmos podem sofrer ataques de usuários mal intencionados, que buscam obter informações para benefício próprio ou em busca de um bem que pra eles seria considerado "coletivo"(como ataques com viés políticos).

Alguns exemplos de motivações para os atacantes são:

- Roubar dados de usuários;
- Ganhos por extorsão;
- Derrubar servidores em competições;

- Ações políticas[1];
- Comportamento imaturo;
- Prestar serviços de ataque[2].

Um tipo de ataque que pode ocasionar essa indisponibilidade é o ataque de negação de serviço. Ele busca interromper o acesso de usuários legítimos a algum serviço ou recurso compartilhado ao longo da Internet. Basicamente, o atacante busca sobrecarregar a vítima, de forma que ela esgote os seus recursos computacionais, e não consiga mais receber tráfego, ou processar as requisições que chegam nela.

Dentro das subcategorias do ataque de negação de serviço, há aqueles por reflexão e amplificação. O qual consiste no redirecionamento de um pacote(reflexão), e no aumento da quantidade de pacotes, acarretando na sua potencialização(amplificação). Para isso, os atacantes fazem uso dos chamados refletores, os quais podem ser encontrados em alguns protocolos que respondem à requisições com pacotes muito maiores do que o pacote inicial, como o *Domain Name System* (DNS) e o *Simple Network Management Protocol* (SNMP) por exemplo. Outro protocolo com essa característica é o *Network Time Protocol* (NTP), presente em diversos dispositivos da rede como servidores, computadores pessoais e roteadores. Assim, nessa monografia estudaremos o ataque de negação de serviço por reflexão amplificada explorando o protocolo NTP, a fim de compreender melhor esse tipo de ataque, bem como a vulnerabilidade presente no protocolo.

1.2 Justificativa

Manter um sistema sempre ativo e estável é o objetivo de toda grande empresa. Quando nos voltamos à serviços essenciais como transações bancárias, ou consultas que dependem de uma prévia autorização realizada através de sistemas, nos damos conta de quão importante é que esses serviços funcionem sempre de forma rápida e da maneira correta. Contudo, nem sempre os sistemas funcionam conforme esperamos.

Em Fevereiro de 2014 por exemplo, a companhia americana provedora de segurança CloudFlare sofreu um dos maiores ataques DDoS da história, recebendo um tráfego de aproximadamente 400Gbps[3]. Nesse ataque, foram utilizados 4529 servidores NTP rodando em 1298 redes diferentes.

Já em Fevereiro de 2018, a companhia Akamai relatou que o GitHub, um dos seus clientes, sofreu o maior ataque DDoS da história, o qual proveu um tráfego de 1.3 Tbps[4]. Esse ataque ocorreu sob o uso de servidores *Memcached* como refletores, e superou o antes registrado maior ataque sobre a companhia Dyn, que sofreu um tráfego de 1.2 Tbps[5], e tornou o acesso a sites como Netflix, Twitter, Reddit e CNN indisponíveis.

Sendo o *Network Time Protocol* utilizado por uma grande quantidade de dispositivos conectados à rede, para manterem os seus relógios sincronizados, encontra-se aqui um alvo que poderia facilmente ser utilizado para ataques de negação de serviço.

Ao mostrarmos aos leitores algumas consequências provenientes da ampla quantidade de vulnerabilidades na Internet, mais especificamente vulnerabilidades relativas à exploração de protocolos, se faz necessário que sejam conduzidos estudos que visam obtermos uma melhor compreensão dessas vulnerabilidades. Por consequência, encorajá-los a no futuro buscar a tomada de ações que possam promover a segurança da Internet.

1.3 Objetivo

O objetivo deste trabalho consiste em prover aos leitores, um conhecimento prévio acerca das variantes do ataque de negação de serviço e como elas funcionam; detalhar o funcionamento do protocolo NTP bem como as suas vulnerabilidades, e instigar ao estudo de outros protocolos e suas respectivas vulnerabilidades.

Através da ferramenta desenvolvida Linderhof, poderemos realizar ataques em ambientes controlados a servidores NTP, a fim de coletar dados estatísticos, para demonstrar o uso da vulnerabilidade do protocolo. Atualmente, essa ferramenta além de trabalhar com o protocolo NTP, pode explorar vulnerabilidades dos protocolos *Memcached* e SSDP.

1.4 Organização

A monografia está disposta do seguinte modo:

- Capítulo 2: introdução teórica necessária para compreender o funcionamento do ataque;
- Capítulo 3: apresentação da estrutura do Linderhof, bem como seu funcionamento;
- Capítulo 4: análise dos dados coletados;
- Capítulo 5: conclusão e trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Nesse capítulo será feita a explicação do funcionamento do ataque de negação de serviço, bem como será tratado a fundo o funcionamento do protocolo NTP.

2.1 DoS - *Denial of Service*

O ataque de negação de serviço, ou em inglês, *Denial of Service*, ocorre quando um atacante envia uma quantidade excessiva de tráfego ilegítimo para algum sistema, serviço ou dispositivo, até que se chegue ao ponto de o alvo que está sendo atacado sobrecarregar, e não conseguir mais responder a todas as requisições, tornando o seu acesso indisponível. Assim, o objetivo principal do atacante é sobrecarregar um sistema, até que ele exaure os seus recursos, e os usuários legítimos não consigam fazer uso dele, sofrendo uma negação de serviço.

Ferramentas que executam esse ataque como *Low Orbit Ion Cannon* (LOIC) [6] ou *R-U-Dead-Yet* (RUDY) [7] são simples de serem encontradas na Internet. Por não exigirem um grau muito grande de preparo e serem relativamente simples de se manusearem, nas mãos de usuários mal intencionados e até mesmo inexperientes, se configuram como ameaças potenciais.

2.2 DDoS - *Distributed Denial of Service*

O ataque DDoS é um DoS mais estruturado. Ele age de maneira similar ao DoS, entretanto, o ataque parte não somente de um *host*, mas sim de múltiplos, de maneira distribuída. Geralmente esses ataques dependem da estrutura de *Botnets*, onde temos alguns *Handlers* (máquinas com pacotes de softwares mal intencionados distribuídos ao longo da Internet), que coordenam vários *hosts* comprometidos (comumente chamados de *Agents* ou *Zombies*)[8]. O atacante então se comunica com os seus *Handlers*, que ordenam

aos *agents*(*zombies*) a realizarem de fato o ataque. Assim, o ataque parte de vários pontos distintos, evitando a saturação do atacante, e que ele seja identificado.

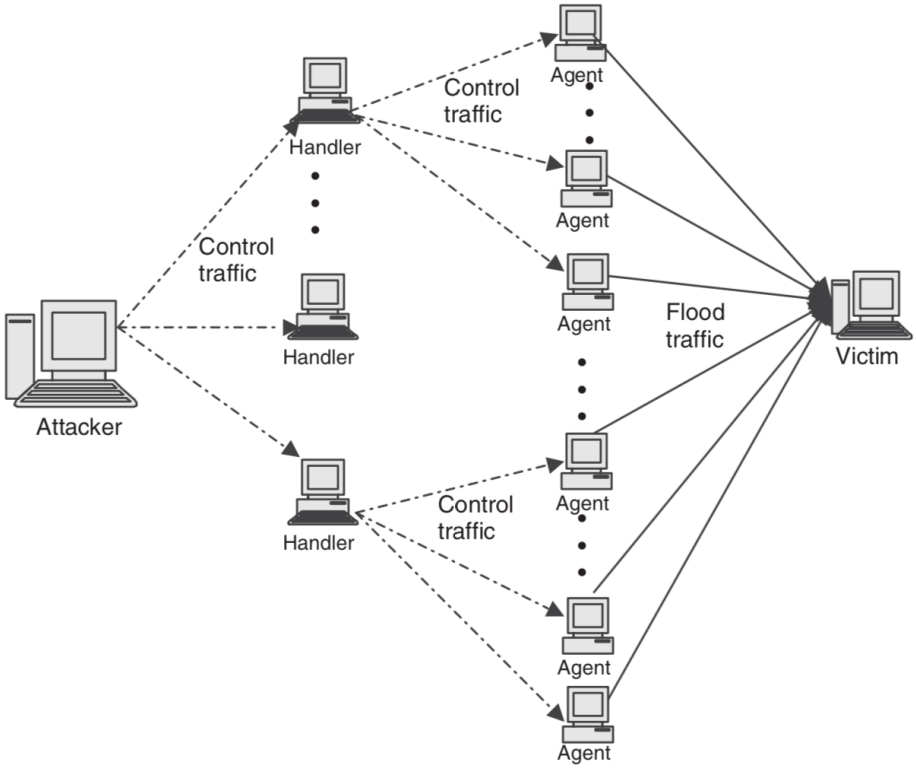

Figura 2.1: Estrutura de um ataque DDoS.

Na Figura 2.1 temos um exemplo de como o ataque se configura. Nela podemos ver a grande diferença entre um ataque DoS e um DDoS, pois uma *botnet*, ou seja uma grande quantidade de máquinas distribuídas (*agents/zombies*), e não somente uma, geram tráfego malicioso através da Internet para a vítima, além do tráfego de dados legítimo dos usuários reais. O grande tráfego de dados sobrecarrega a vítima, esgota os seus recursos, e torna assim o acesso ao serviço prestado por ela indisponível.

Segundo Muhammad Aamir e Mustafa Ali Zaidi, os ataques DDoS são classificados sumariamente como uma exploração de vulnerabilidade[9]. Partindo desse pressuposto, os ataques podem então serem subclassificados conforme as seguintes categorias, presentes na Figura 2.2, e melhor detalhados nas seções subsequentes:

- ***Malformed Packet Attack:*** pacotes IP mal formados são enviados às vítimas, buscando corrompê-las.
- ***Protocol Exploit Attack:*** os atacantes se aproveitam de alguma característica do protocolo ou de sua implementação para gerar uma alocação de recursos na vítima.

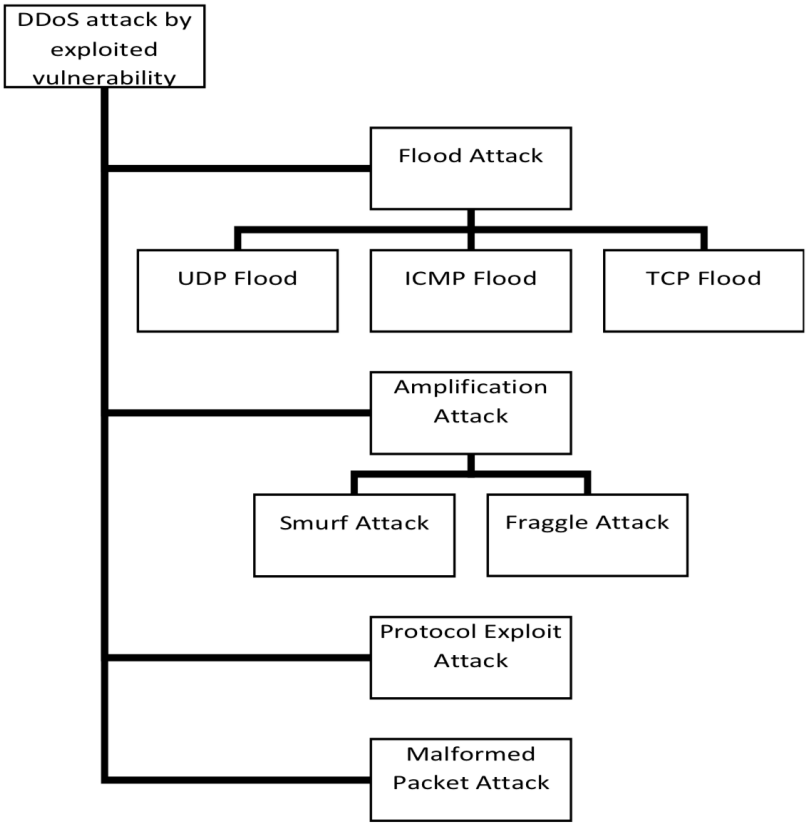

Figura 2.2: Classificação ataque DDoS.

- ***Flood Attack:*** *zombies* enviam grandes volumes de tráfego à vítima, em busca de congestionar a sua rede com o tráfego.
- ***Amplification/Reflection Attack:*** os ataques de reflexão/amplificação, tomam vantagem de serviços UDP públicos e acessíveis, para sobrecarregar as vítimas com respostas de requisições[10].

2.2.1 *Malformed Packet Attack*

Como exemplo do ataque de pacote mal formado, temos o caso do *Ping of Death* (POD). O ataque ocorre quando um *host* envia vários fragmentos de pacotes, que devem ser reconstruídos no sistema que os recebe após a transmissão. Se os fragmentos adicionarem mais dados do que o permitido para o tamanho de um pacote, ocorrerá uma sobrecarga nos buffers de memória, e assim o sistema receptor ficará lento, ou até mesmo inacessível.

O tamanho máximo de um pacote IP é de 65535 *bytes*, contudo a camada de enlace só permite pacotes do tamanho máximo de até 1500 *bytes*. Assim, para que o pacote fosse transmitido, ele teria que ser fragmentado em vários pacotes menores, que

vão ser reconstruídos pelo sistema receptor. Na teoria, o tamanho de uma requisição **ICMP_ECHO_REQUEST** (*ping*) é de no máximo 65507 *bytes*. Entretanto, em decorrência do jeito que a sua fragmentação é realizada, é possível que no último fragmento combinemos um valor de *offset* junto com um tamanho de fragmento adequado, de modo que o offset mais o tamanho do fragmento resultará em um pacote maior que 65535 *bytes*[11]. Como a maioria das máquinas não processam os pacotes até que ele tenha chegado completamente e tenha sido reconstruído, ocorre a sobrecarga dos *buffers* de memória, e por consequência, as mesmas ficam inacessíveis.

2.2.2 *Protocol Exploit Attack*

No ataque de exploração de protocolo, é explorada algum erro de implementação, ou alguma funcionalidade de um protocolo que se encontra na máquina da vítima. Um bom exemplo desse ataque é o *TCP SYN*, pois ele explora a fraqueza do *three way handshake* presente no *handshake* do protocolo TCP.

Em um cenário ideal, o cliente iniciaria a comunicação com o servidor enviando um sinal *SYN* para o servidor, requisitando que fosse estabelecida uma conexão. O servidor então responderia com um sinal *ACK* que seria o reconhecimento de que o servidor está pronto para estabelecer a solicitação de conexão. Após isso, o servidor fica esperando pelo *ACK* do cliente, e quando ele chega, a conexão é estabelecida[9].

Já no ataque *SYN flood*, um atacante iniciaria o ataque enviando inúmeros pacotes *SYN* com seus endereços IPs forjados. Dessa maneira, os servidores enviariam os *ACKs* do estabelecimento da conexão para endereços que não estavam tentando estabelecer uma conexão, e por consequência, não responderiam aos servidores, de modo que os servidores nunca receberiam o *ACK* final do cliente. Assim, inúmeras conexões que foram parcialmente estabelecidas ficariam na fila do servidor, congestionando o sistema[12], e os clientes legítimos que buscavam estabelecer uma conexão não conseguiriam sucesso, pois o sistema estaria inacessível a eles em decorrência da negação de serviço.

Como o *SYN flood* apresenta uma grande quantidade de pacotes enviados, gerando um grande tráfego de dados, ele poderia ser classificado como um ataque de inundação. Contudo, como ele explora a funcionalidade do *three way handshake* do TCP, ele é classificado como uma ataque de exploração de protocolo. Na Figura 2.3, temos uma demonstração de como esse ataque funciona, onde a vítima responde ao pacote SYN para *hosts* que não estavam tentando estabelecer a conexão, e assim, o *ACK* final do cliente nunca chega.

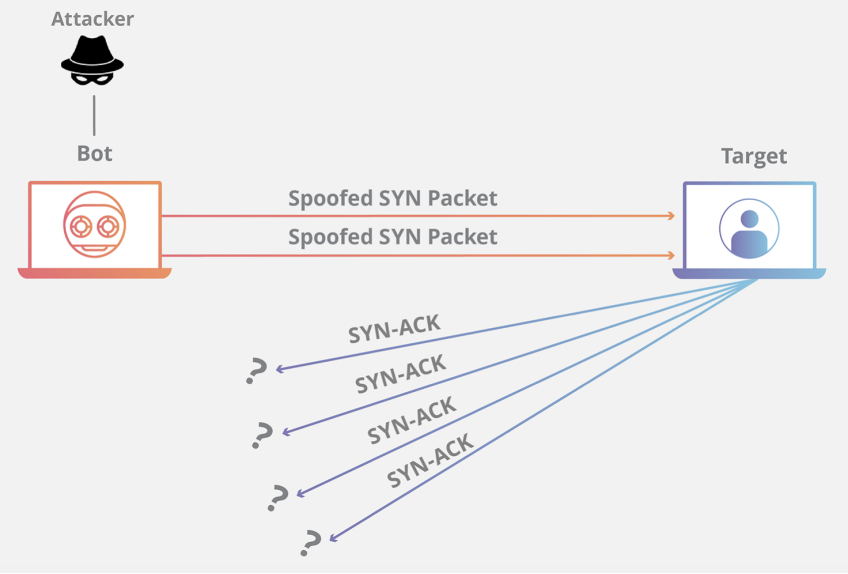

Figura 2.3: Demonstração do ataque *SYN flood*.

2.2.3 *Flood Attack*

Nos ataques de inundação, são enviados inúmeros pacotes a fim de saturar a rede e esgotar a banda da vítima. Para exemplificarmos, vamos tratar dos ataques UDP *flood* e ICMP *flood*. Em um ataque de inundação UDP, inúmeros pacotes UDP são enviadas para portas aleatórias ou específicas no sistema das vítimas. A vítima então processa os dados recebidos, e checa se há algum programa rodando que está escutando naquela porta específica, esperando por requisições naquela porta em questão. Caso não haja nenhum programa esperando por requisições naquela porta, o sistema da vítima envia um pacote ICMP (*Internet Control Message Protocol*) informando um *'Destination host unreachable'* (*host* de destino não está acessível) ao remetente (atacante)[13].

Para ocultar a identidade do atacante e prevenir que sua máquina receba qualquer resposta, o atacante forja o endereço IP de origem dos pacotes de ataque, e assim, as respostas não chegam aos destinos das requisições de origem. Como a máquina da vítima fica constantemente ocupada processando e tentando responder as requisições, o sistema fica sobrecarregado e resulta em uma negação de serviço para o tráfego legítimo. Ataques de inundação UDP também podem esgotar a largura de banda da rede em torno do sistema da vítima. Assim, os sistemas em torno da vítima também são afetados devido ao ataque de inundação do UDP. Na Figura 2.4 temos um exemplo bem simplificado desse ataque.

Já o ataque *ICMP flood* funciona de maneira quase semelhante ao *UDP flood*. Conforme podemos ver na Figura 2.5, neste ataque inúmeros pacotes ***ICMP_ECHO_REQUEST*** *(ping)* com o cabeçalho IP forjado são enviados para as vítimas da maneira mais rá-

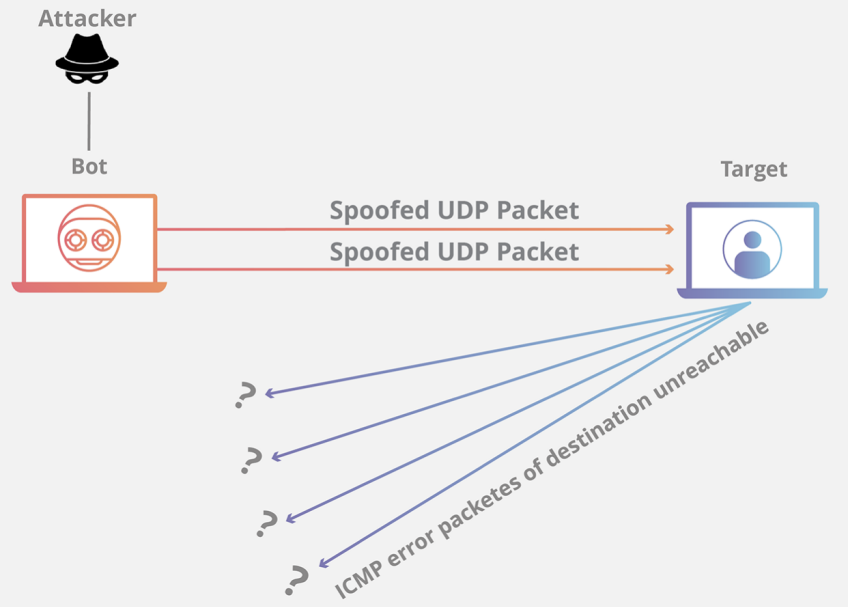

Figura 2.4: Demonstração do ataque *UDP flood*[14].

pida possível, sem esperar pelas respostas. A vítima então gera o pacote de resposta ***ICMP_ECHO_RESPONSE***, que nunca chegam na origem, em decorrência do cabeçalho IP forjado. Como ele continua recebendo as requisições, e fica gerando as respostas para os pacotes, as quais nunca obtem êxito ao enviá-los, a banda da vítima é consumida, e impossibilita que requisições legítimas sejam atendidas.

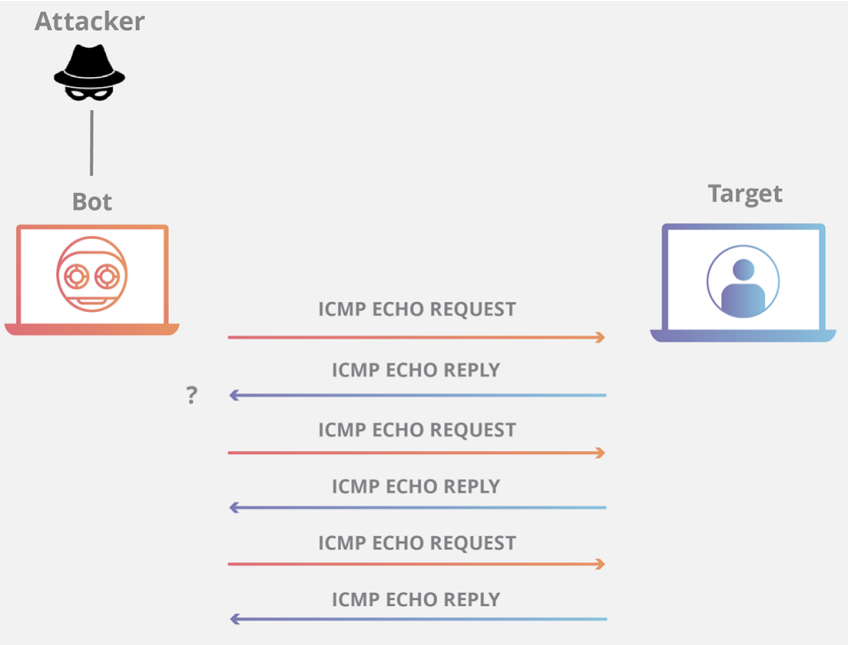

Figura 2.5: Demonstração do ataque *ICMP flood*[15].

2.2.4 *Amplification/Reflection Attack*

Para que tenhamos um ataque de amplificação e reflexão bem sucedido, dependemos de dois fatores. O primeiro é que a mensagem seja amplificada, que ocorre quando um atacante envia um pacote de solicitação de serviço de tamanho pequeno, e recebe como resposta um pacote muitas vezes maior. Já a reflexão ocorre quando o atacante forja o endereço IP do solicitante, como sendo da vítima, para que a vítima receba as mensagens. Juntando esses dois fatores, conseguimos elaborar o ataque pois o atacante forja os endereços IPs das mensagens que solicitam algum serviço, que respondem com pacotes muitos maiores para as vítimas. A estrutura desse ataque pode ser vista na Figura 2.6, onde além da presença do computador atacante, dos *Masters* (*Handlers*) e dos *Slaves* (*Agents/Zombies*), adicionamos os refletores, que são os responsáveis pela amplificação e reflexão do tráfego.

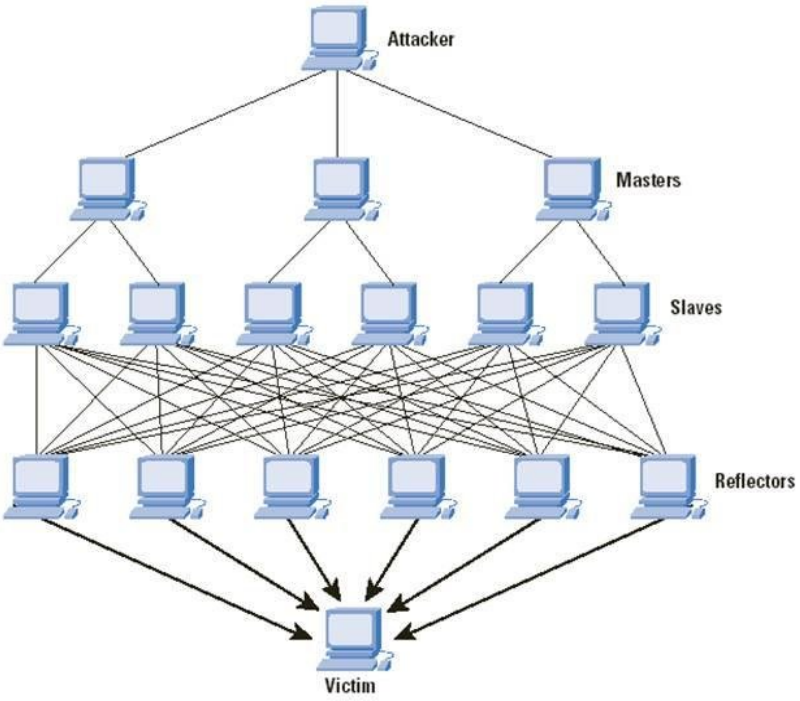

Figura 2.6: Estrutura do DDoS com reflexão.

Como a amplificação é de suma importância para o desenvolvimento do ataque, se faz necessário que compreendamos como a amplificação é calculada. Basicamente, divide-se o tamanho do pacote de resposta pelo tamanho do pacote de uma requisição. Alguns pesquisadores como Rossow[16] não utilizam todos os campos do pacote para o cálculo da amplificação, usando somente o *payload* da resposta e da requisição, de modo que o

fator de amplificação de banda (*Bandwidth Amplification Factor* - BAF) seria calculado da seguinte maneira:

$$BAF = \frac{SizePayload(resp)}{SizePayload(req)} \tag{2.1}$$

Onde,

- **BAF** = Fator de amplificação de banda;
- ***SizePayload(resp)*** = Tamanho em *bytes* do *payload* da resposta;
- ***SizePayload(req)*** = Tamanho em *bytes* do *payload* da requisição.

Entretanto, como todos os cabeçalhos dos pacotes foram resultados de um esforço computacional para que fossem gerados, vamos incluir para o cálculo da amplificação os cabeçalhos IP e UDP, resultando na equação explicitada logo a seguir:

$$BAF = \frac{SizeIP(resp) + SizeUDP(resp) + SizePayload(resp)}{SizeIP(req) + SizeUDP(req) + SizePayload(req)} \tag{2.2}$$

Onde, variando entre requisição (req) e resposta (resp),

- **BAF** = Fator de amplificação de banda;
- ***SizeIP*** = Tamanho em *bytes* do cabeçalho IP;
- ***SizeUDP*** = Tamanho em *bytes* do cabeçalho UDP;
- ***SizePayload*** = Tamanho em *bytes* do *payload*.

Inicialmente a amplificação foi muito explorada nos ataques *Smurf* e *Fraggle*. Entretanto, com o passar dos anos e a descoberta de novas vulnerabilidades, muitos atacantes começaram a utilizar esse princípio da amplificação e reflexão explorando outros protocolos ou serviços como o *Memcached*, DNS, SSDP e NTP, sendo este último o alvo do trabalho, e que será discutido mais adiante.

Ataque *Smurf* e *Fraggle*

O ataque *Smurf* consiste no envio de inúmeras requisições *ECHO* (ping) à endereços *broadcast* da rede, para que o pacote ICMP seja retransmitido para todos os dispositivos situados após o roteador. O atacante então forja o endereço IP dos pacotes, para que a vítima receba as respostas das solicitações. Como o roteador repassa essa requisição para os dispositivos que se encontram após ele, todos os dispositivos que receberem essa

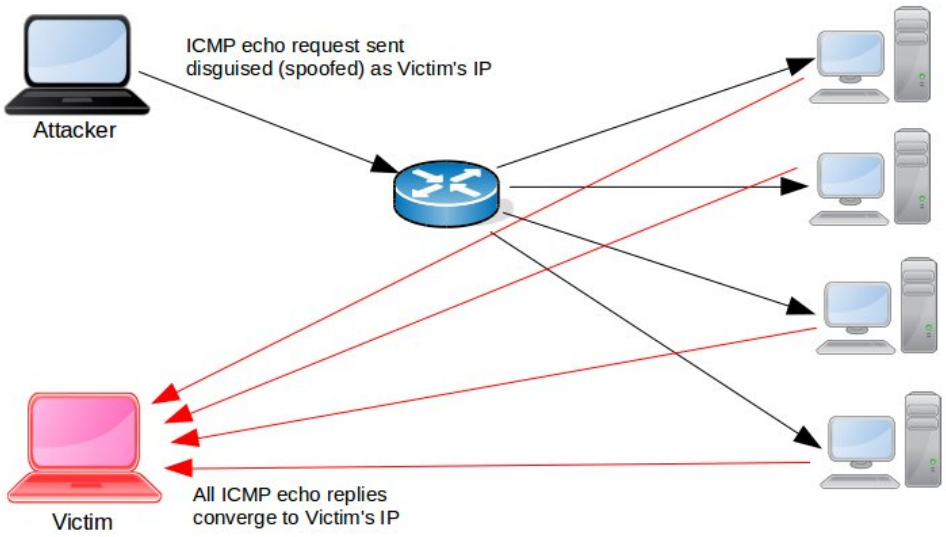

Figura 2.7: Demonstração do ataque *Smurf*.

requisição irão responder para a vítima, refletindo e amplificando o ataque. Podemos ver um exemplo simplificado na Figura 2.7.

Já o ataque *Fraggle* ocorre de maneira quase que semelhante ao Smurf. A única diferença é que ao invés de utilizar requisições *ECHO* do tipo ICMP, o ataque utiliza as requisições *ECHO* do tipo UDP. Segundo A. Mitrokotsa e C. Douligeris, o ataque *Fraggle* pode ter um impacto muito mais severo que o *Smurf*[13].

Ataque usando o *Memcached*

O *Memcached* é um sistema geral de armazenamento em cache de memória distribuída, geralmente usado para acelerar aplicativos dinâmicos da *Web*, armazenando dados e objetos em *cache* na *RAM* e reduzindo o banco de dados de *backend* ou os ciclos da API[17]. Esse protocolo permite que o servidor *Memcached* seja submetido a consultas que que buscam informação sobre os valores de chaves armazenadas, e como não há autenticação no *memcached*, ele só deveria ser usado em sistemas que não estão expostos na Internet. Contudo, vários administradores de rede deixam as portas 11211 do TCP e do UDP abertas para consultas oriundas na Internet. Ao passo que no dia 05-03-2018, o serviço de busca de servidores conectados à Internet, *Shodan*, relatou que mais de 105,000 servidores estavam conectados à Internet, e responderam a requisições TCP ou UDP na porta 11211[18].

Para realizar o ataque, a primeira coisa que o atacante faz é forjar o cabeçalho IP de uma requisição HTTP GET com o endereço da vítima, e envia para o servidor. Quando o servidor processa a requisição, ele responde ao alvo com múltiplos pacotes UDP, cada

um com o tamanho de até 1400 *bytes*. O alvo então não consegue processar a grande quantidade de dados fornecida pelo servidor *Memcached*, e resulta em uma negação de serviço para as requisições legítimas. Se o atacante quiser gerar um ataque ainda maior, ele pode inserir dados extra no servidor *memcached* exposto, antes de realizar o ataque, obtendo assim, taxas de amplificação maiores ainda.

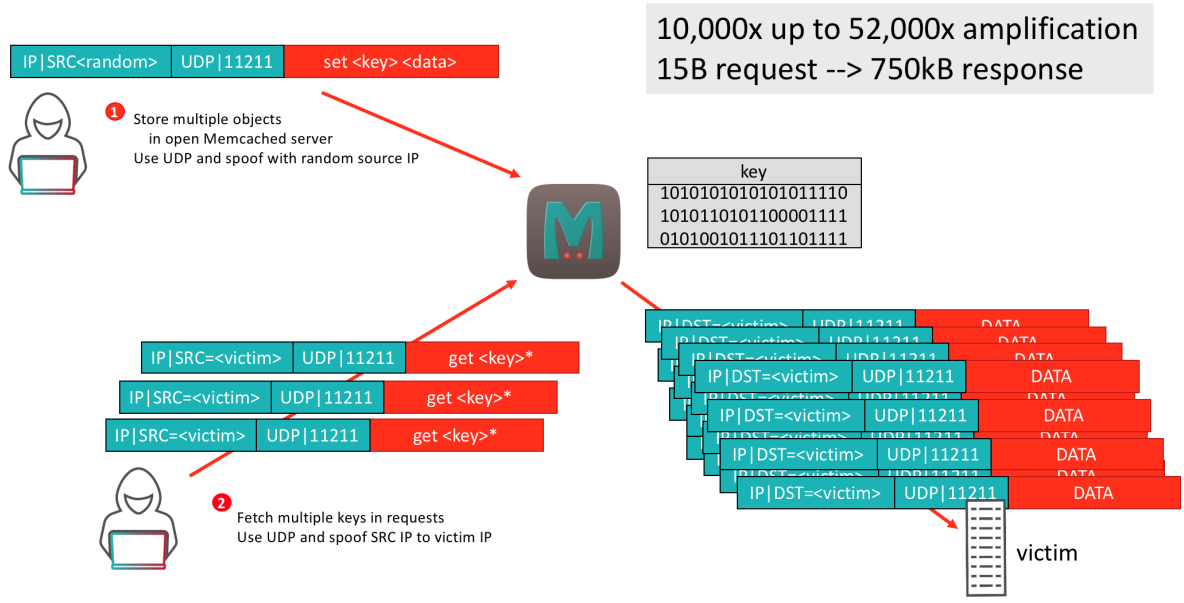

Figura 2.8: Demonstração do ataque *Memcached*[17].

Na Figura 2.8 temos um exemplo do ataque. Primeiro, o atacante armazena vários dados no servidor *memcached*. Logo depois, ele faz várias requisições ao servidor, acerca das chaves armazenadas, sendo que o seu IP está forjado apontando para a vítima. O servidor as processa, e então responde a vítima a sobrecarregando. É importante notar a informação presente na imagem de que nesse ataque, uma requisição tem o tamanho de 15 *bytes*, e obtém uma várias respostas que somadas geram até 750 kb, conseguindo assim uma taxa de amplificação de até 52,000x.

Ataque usando o SSDP

O protocolo SSDP é um protocolo da camada de aplicação que permite que dispositivos conectados à Internet descubram uns aos outros, e estabeleçam serviços funcionais como troca de dados, comunicação e entretenimento[19]. Este protocolo trabalha com o método multicast de notificar e descobrir rotas. As requisições deste protocolo podem ser divididos em dois tipos: a do tipo *NOTIFY*, que pode ser usada pelos dispositivos para anunciar a

sua presença na rede; e a do tipo *M-SEARCH*, a qual o dispositivo pode descobrir serviços disponíveis na rede.

Já o protocolo SOAP fornece um padrão sob o qual diferentes aplicativos rodam em sistemas operacionais diferentes e usa várias tecnologias e linguagens de programação que podem se comunicar entre si. Ele é usado para entregar mensagens de controle para dispositivos UPnP e para passar informações de volta dos dispositivos. Os invasores descobriram que as requisições SOAP podem ser criadas para obter uma resposta que reflita e amplifique um pacote, para direcioná-lo a um destino. Quando o cliente envia uma requisição, a quantidade de pacotes de resposta é muito maior do que a quantidade de pacotes de solicitação.

Com essas informações podemos compreender o funcionamento do ataque, conforme bem ilustrado na Figura 2.9. Antes de tudo, o atacante escaneia a rede procurando por dispositivos *plug-and-play* que podem ser usados como refletores. O atacante, que está controlando a sua *botnet*, orienta às máquinas comprometidas a enviarem pacotes UPnP do tipo *discovery (M-SEARCH)*, para os dispositivos *plug-and-play* obtidos anteriormente, entretanto com o IP forjado com o endereço da vítima. Os dispositivos então respondem com os pacotes *NOTIFY* para a vítima, que sobrecarrega e sofre uma negação de serviço para os pacotes legítimos. É interessante notarmos que este pacote consegue taxas de amplificação de até 30.8x.

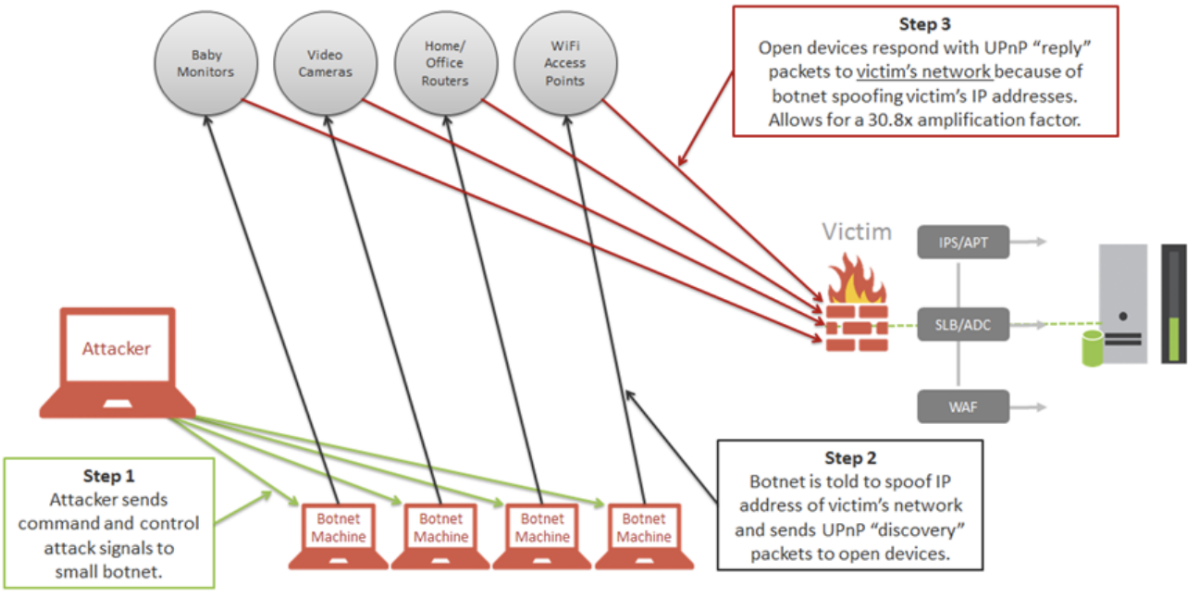

Figura 2.9: Demonstração do ataque SSDP[20].

Ataque usando o DNS

Os servidores *Domain Name System* (DNS) são os responsáveis por localizar e traduzir os endereços dos sites que digitamos nos navegadores para números IP, através de uma base de dados distribuída e hierárquica que armazena o endereço IP corresponde para cada nome do domínio. Segundo a *Cloudflare*[21], existem quatro categorias de servidores DNS que são: servidores resolvedores recursivos, servidores de nomes raiz, servidores de nomes de TLD e servidores de nomes oficiais. Os servidores utilizados para o ataque de amplificação/reflexão são os servidores recursivos abertos, que respondem a consultas de qualquer tipo de cliente, e não somente os locais.

Na Figura 2.10, podemos acompanhar o andamento do ataque, que está descrito logo a seguir. O atacante, sob controle de sua *botnet*, ordena aos seus *zombies* que enviem requisições com o IP forjado do tipo "*ANY*"para o servidor DNS recursivo aberto. Após receber as requisições, o servidor DNS responde à vítima, que sobrecarrega com a grande quantidade de tráfego, e resulta em uma negação de serviço. É interessante notarmos que o pacote de requisição geralmente tem um tamanho médio de 100 *bytes*, e a resposta do servidor geralmente tem um tamanho 4,000 *bytes*, o que representa uma taxa de amplificação de 40x[22].

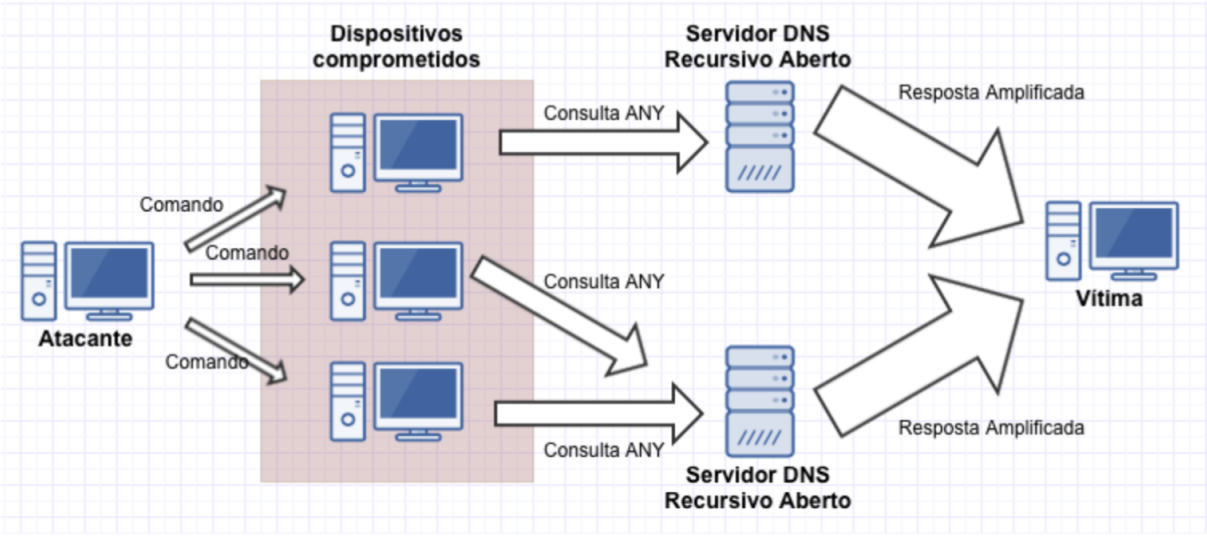

Figura 2.10: Demonstração do ataque DNS com servidor recursivo[23].

2.3 Ataque usando o NTP - *Network Time Protocol*

O NTP (*Network Time Protocol*), é o protocolo que sincroniza um conjunto de relógios na rede, através do sistema de clientes e servidores, com a precisão de nanosegundos[24].

Como o NTP é o protocolo que será utilizado para o ataque com reflexão amplificadora, ele será explicado a seguir.

2.3.1 Histórico do NTP

Segundo David Mills[25], o criador do *Network Time Protocol* (NTP), a primeira implementação ocorreu por volta dos anos 1980, mas só foi documentada no ano de 1981, na já extinta "*Internet Engineering Note*"como IEN-173 [26]. Já a primeira especificação apareceu na RFC778[27], entretanto, ainda era intitulada como *Internet Clock Service*. Na primeira vez em que recebeu o nome de NTP no ano de 1985, ela foi descrita na RFC958[24], e reconhecida como Versão 0, onde sumariamente eram descritos os pacotes, bem como o cálculo do *offset/delay*.

A especificação da Versão 1 (*NTPv1*) foi documentada no ano de 1988, na RFC1059[28]. Nessa versão já era possível encontrar a especificação do protocolo e algoritmos, bem como a estrutura de cliente/servidor, e modelos simétricos. Um ano depois foi lançada a Versão 2 (*NTPv2*) do NTP através da RFC1119 [29], a qual introduzia o sistema de autenticação criptográfica baseado na criptografia de chaves simétricas, e o protocolo de mensagens de controle NTP para a administração dos clientes e servidores.

Nessa mesma época foi lançado pela compania *Digital Equipment Corporation* o *Digital Time Synchronization Service* (DTSS), que era outro protocolo de sincronização, com objetivos parecidos com o do NTP, mas estratégias de implementação divergentes. Várias ideias do DTSS e NTP foram reunidas no que gerou então, em 1992, a Versão 3 (*NTPv3*) do protocolo, descrita na RFC1305[30]. Nessa versão foram descritos alguns princípios de corretude, foi adicionado o modo *broadcast*, e vários algoritmos foram revisados.

Com o passar do tempo, tanto a especificação quanto a implementação foram alvos constantes de melhorias, até que em 1994, na RFC2030[31] foi lançada uma variante mais simples do protocolo, intitulada como *Simple Network Protocol* (SNTP), que é recomendada para aplicações mais simples, onde a acurácia e a confiabilidade não são de extrema importância. O NTP continuou sendo aperfeiçoado, até que em 2010 foi lançada a Versão 4 (*NTPv4*), disponível na RFC5905[32]. Desde então, várias melhorias foram implementadas no protocolo, deixando pendente mais uma publicação de uma RFC. Por fim, com a aposentadoria do professor David L. Mills da Universidade de *Delaware*, o projeto continua sendo mantido por Harlan Stenn através da sua empresa *Network Time Foundation*.

2.3.2 Arquitetura do NTP

No NTP temos um conjunto de vários servidores que funcionam de forma hierárquica. Esses servidores estão divididos ao longo de 17 camadas, chamadas de estratos, ou em inglês, *stratum*. Nessa hierarquia, quanto menor o seu *stratum* mais alta é a sua posição na tabela de hierarquias, ou seja, a camada mais alta é o *stratum* 0, e a mais baixa, é o *stratum* 16. No *stratum* 0 podemos encontrar as referências primárias de tempo, como os relógios atômicos e os receptores de GPS . A medida que nos distanciamos do topo, perdemos a precisão dos servidores, até que cheguemos no *stratum* 16, que indica que um servidor está inoperante.

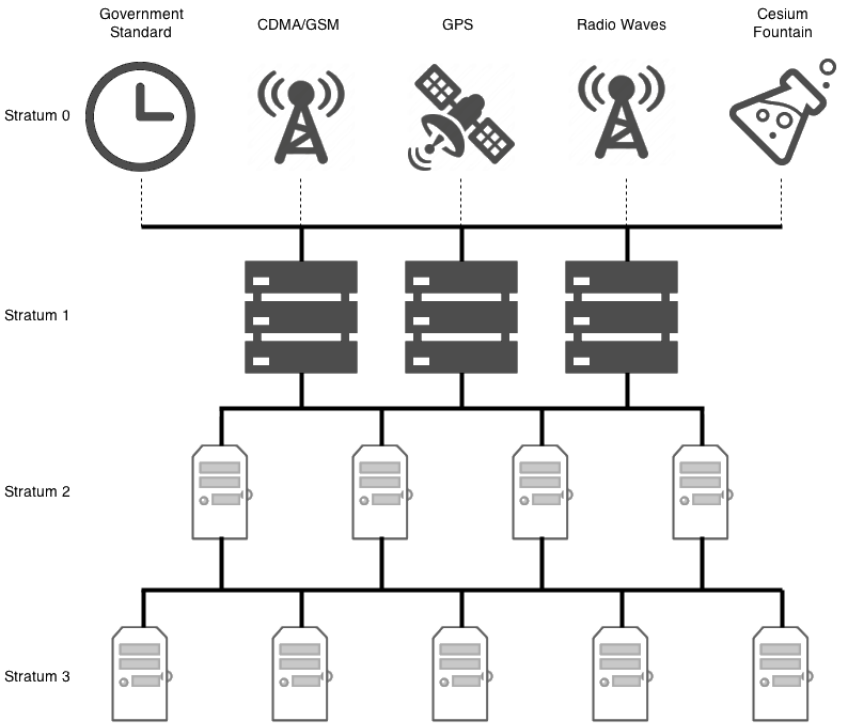

Figura 2.11: Topologia em camadas (*stratum*) do NTP[33].

Os servidores do *stratum* 1 são conectados fisicamente aos servidores do *stratum* 0[34], pois os mesmos não estão acessíveis via Internet. A partir do *stratum* 1 bem como os seus subsequentes, a conexão entre os servidores se dá totalmente via Internet. O *stratum* 0 fornece o tempo corretamente para o *stratum* 1, que fornece o tempo para o *stratum* 2, e assim por diante, ou seja, os *hosts* do NTP são ao mesmo tempo cliente (consulta o tempo ao *stratum* superior) e servidor (fornece o tempo ao *stratum* subsequente).

Segundo o Projeto NTP.br[35], os dispositivos NTP podem se associar das seguintes maneiras: cliente-servidor; modo simétrico e *broadcast* ou *multicast*. As três associações são explicadas logo a seguir:

- **Cliente-Servidor:** um cliente solicita informações sobre o tempo a um servidor, sendo que o cliente tem conhecimento das associações com os servidores e do estado da troca de pacotes. Um outro dispositivo faz o papel de servidor, pois responde à solicitação do cliente com informações sobre o tempo. O processo do cliente de enviar uma solicitação ao servidor e aguardar pela resposta é descrito como uma operação do tipo *pull*. Um cliente pode criar associações com vários servidores simultaneamente, e um servidor pode fornecer tempo a diversos clientes simultaneamente, por isso, dizemos que um *host* NTP normalmente é cliente e servidor ao mesmo tempo.
- **Modo Simétrico:** dois ou mais dispositivos que estejam no mesmo stratum são configurados como pares (*peers*), de forma que podem tanto buscar o tempo como fornecê-lo, garantindo uma redundância mútua. Dessa maneira, caso um dos pares perca a sua referência de tempo, o outro pode funcionar como referência, e além disso, os pares podem verificar quem tem o tempo mais preciso, para que possam se sincronizar a ele. O modo simétrico pode ser ativo, onde os dois pares configuram um ao outro como par, criando uma associação permanente; e também pode ser passivo, onde somente um dos pares configura o outro como par, criando uma associação transitória.
- ***Broadcast ou Multicast:*** neste modo, o cliente NTP, ao receber o primeiro pacote de um servidor, busca os dados por um curto período de tempo como se estivesse no modo cliente - servidor, a fim de conhecer o atraso envolvido. Ou seja, durante alguns instantes há troca de pacotes entre cliente e servidor, depois disso o cliente passa apenas a receber os pacotes *broadcast* ou multicast do servidor. Essa configuração é vantajosa no caso de redes locais com poucos servidores alimentando uma grande quantidade de clientes.

2.3.3 Comando *MONLIST*

Dentre os vários comandos disponíveis presentes no protocolo NTP para fins de monitoramento, um deles se faz especial e de suma importância para o andamento desse trabalho, que é o *monlist* (*MON_GETLIST* ou *MON_GETLIST_1*). O comando *monlist* opera no modo privado, que é o modo utilizado para realizar a troca de informações entre cliente e servidor, e é enviado em uma mensagem UDP na porta 123, que é a porta utilizada pelo protocolo NTP. Quando o cliente envia esse comando ao servidor, o servidor retorna ao cliente uma lista com os últimos 600 *hosts* que se conectaram a ele[36], além de dados como endereço IP de origem e destino, versão do NTP e modo do pacote[37].

Cada requisição *monlist* pode receber como resposta até 100 datagramas UDP, cada um com o tamanho de 440 *bytes* de *payload*[16]. O fator de amplificação de banda do Protocolo NTP pode ser muito grande, já que segundo Russow[16], uma requisição pode ter um tamanho mínimo de 8 *bytes*, enquanto a sua resposta pode conter milhares de *bytes*. Nos experimentos realizados por ele, a requisição *monlist* teve uma média de amplificação de 556.9x, sendo que em seus experimentos, a maior taxa de amplificação de banda foi de 4670x.

A ferramenta Nmap[38] pode nos dar um belo exemplo do funcionamento desse comando. Ao executarmos o *script* "nmap -sU -pU:123 -Pn -n –script=ntp-monlist <target>"[37], sendo *target* o servidor NTP que seria usado como amplificador, recebemos uma resposta no formato da presente na Figura 2.12. Ao acompanharmos a requisição desse script em um analisador de rede como o *Wireshark*[39], podemos ver o envio de uma pequena requisição, e como resposta vários pacotes bem maiores que a requisição inicial, comprovando a amplificação.

```
PORT    STATE SERVICE REASON
123/udp open  ntp     udp-response
| ntp-monlist:
|   Target is synchronised with 127.127.38.0 (reference clock)
|   Alternative Target Interfaces:
|       10.17.4.20
|   Private Servers (0)
|   Public Servers (0)
|   Private Peers (0)
|   Public Peers (0)
|   Private Clients (2)
|       10.20.8.69      169.254.138.63
|   Public Clients (597)
|       4.79.17.248     68.70.72.194    74.247.37.194   99.190.119.152
|       ...
|       12.10.160.20    68.80.36.133    75.1.39.42      108.7.58.118
|       68.56.205.98
|       2001:1400:0:0:0:0:0:1 2001:16d8:dd00:38:0:0:0:2
|       2002:db5a:bccd:1:21d:e0ff:feb7:b96f 2002:b6ef:81c4:0:0:1145:59c5:3682
|   Other Associations (1)
|_      127.0.0.1 seen 1949869 times. last tx was unicast v2 mode 7
```

Figura 2.12: Exemplo de requisição *monlist* com a ferramenta Nmap[37].

Para que a amplificação possa ser realizada pelo comando *monlist*, temos a necessidade de três fatores. Que o servidor NTP esteja habilitado a receber consultas (*queries*), que o comando *monlist* esteja ativado nele, e que o mesmo esteja aberto, ou seja, ele está configurado para responder requisições de qualquer cliente, e não somente os locais. É possível verificar se tem algum servidor NTP aberto em sua rede através do serviço *web Ntp Scan Project* [40].

O ataque DDoS explorando o protocolo NTP se daria então da seguinte maneira. Primeiro o atacante precisaria de uma lista com servidores NTP abertos. É possível obter listas como essa através de ferramentas que escaneiam a Internet como o Zmap[41]. Em

posse dessa lista, o atacante iria forjar os IPs dos pacotes a serem enviados, colocando o IP da vítima como IP de origem, para que os pacotes pudessem ser respondidos para a vítima. Após isso, ele orientaria a sua *botnet* a enviar as requisições do tipo *monlist* para os servidores NTP abertos que foram encontrados, para que eles agissem como refletores e amplificadores dos pacotes. A vítima então iria receber uma quantidade muito grande de tráfego de dados, iria sobrecarregar, e não conseguiria responder as requisições legítimas que estivessem chegando para ela, sofrendo assim, uma negação de serviço. Um exemplo básico do ataque pode ser visualizado na Figura 2.13.

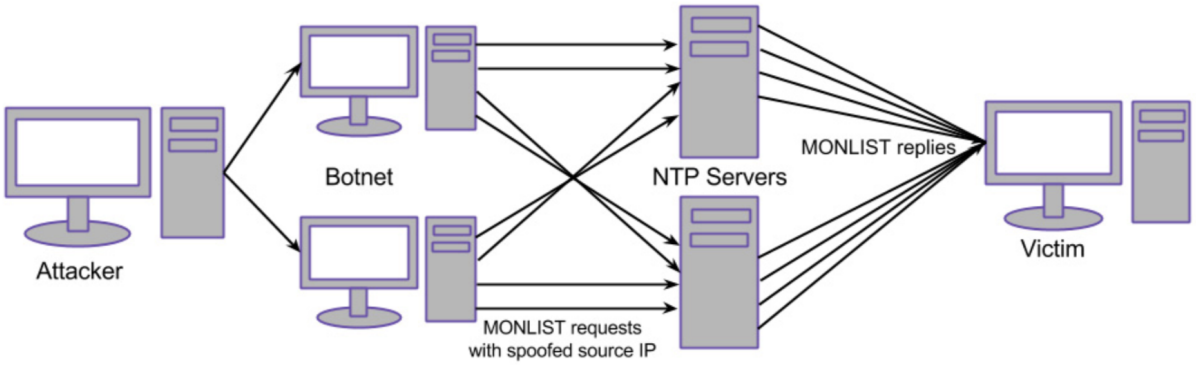

Figura 2.13: Ataque DDoS usando servidores NTP como refletores[42].

2.4 Considerações Finais

Podemos ver na Figura 2.14, uma tabela com os valores do fator de amplificação de banda de vários protocolos baseados no UDP, disponível no *US-CERT Alert (TA14-013A)*[43]. É interessante notarmos que o NTP só perde em fator de amplificação para o protocolo *Memcached*, se mostrando assim, um poderoso amplificador.

Em decorrência da capacidade de amplificação de banda oriunda da utilização do comando *monlist*[44], optou-se pelo uso do protocolo NTP, para conduzir o estudo do ataque distribuído de negação de serviço. Com as altas taxas de amplificação que podemos obter com ele, o mesmo se mostra bastante poderoso em relação aos outros protocolos, e assim, um ótimo refletor.

Protocol	Bandwidth Amplification Factor
DNS	28 to 54
NTP	556.9
SNMPv2	6.3
NetBIOS	3.8
SSDP	30.8
CharGEN	358.8
QOTD	140.3
BitTorrent	3.8
Kad	16.3
Quake Network Protocol	63.9
Steam Protocol	5.5
Multicast DNS (mDNS)	2 to 10
RIPv1	131.24
Portmap (RPCbind)	7 to 28
LDAP	46 to 55
CLDAP [7]	56 to 70
TFTP [23]	60
Memcached [25]	10,000 to 51,000

Figura 2.14: Relação de amplificação por Protocolo[43].

Capítulo 3

Ferramenta de Ataque Linderhof

Neste capítulo será explicada a ferramenta utilizada para a realização do ataque DDoS utilizando o protocolo NTP.

3.1 Ferramenta de Ataque

A ferramenta de ataque utilizada para esse estudo foi o Linderhof, que foi foi desenvolvida em Linguagem C. A ferramenta Linderhof possui a seguinte arquitetura disponibilizada na Figura 3.1, que é dividida em três componentes: *Oryx*, *Commander* e Netuno.

3.1.1 Componente *Oryx*

Esse módulo é responsável pela disponibilização de interface ao usuário e pelo *parser* dos dados inseridos, criando a estrutura do ataque que será executado. Na Figura 3.2, temos um exemplo de como utilizar a ferramenta, obtida através do argumento - h (*helper*). O *helper* orienta ao usuário acerca dos argumentos a serem utilizados junto a ferramenta.

- **-m:** Obrigatório, *mirror*/amplificador que será utilizado no ataque;
- **-t:** Obrigatório, IP da vítima;
- **-a:** Obrigatório, IP do amplificador;
- **-h:** Opcional, mostra o helper do programa;
- **-p:** Opcional, porta para entrada dos pacotes no amplificador;
- **-g:** Opcional, porta para direcionamento dos pacotes da vítima;
- **-l:** Opcional, nível do ataque;
- **-c:** Opcional, tempo do ataque;

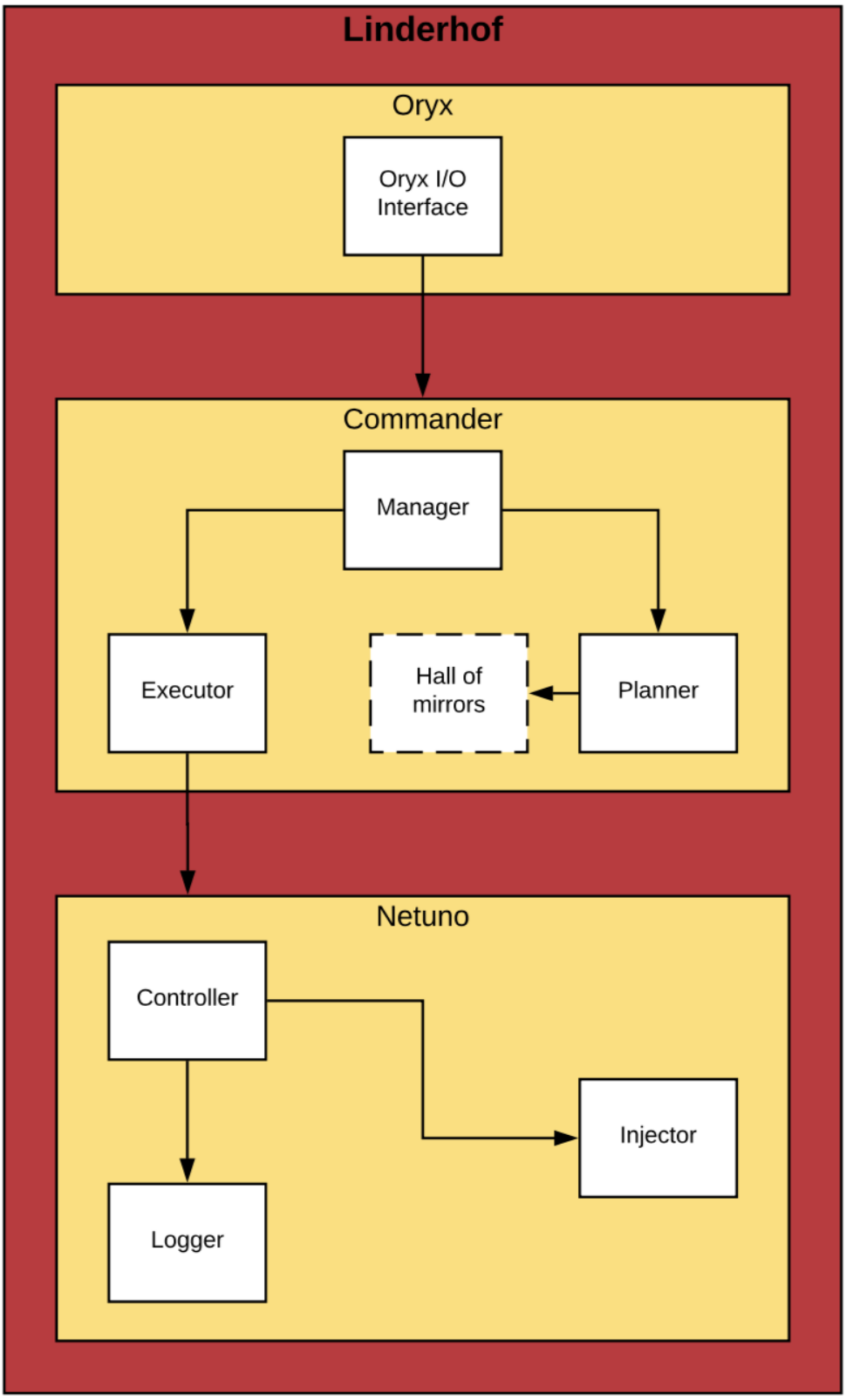

Figura 3.1: Arquitetura da Ferramenta Linderhof.

- **-f:** Opcional, nome do arquivo para escrita do log do ataque;
- **-i:** Opcional, informa que será um ataque incremental.

```
alexander@alexander-Inspiron-N4010:~/Linderhof$ ./bin/lhf -h
Usage: lhf [ARGS]

                Linderhof (lhf) CLI

Mandatory arguments:
    -m    --mirror=INPUT   Mirror type
    -t    --target=INPUT   Target IPV4
    -a    --amplifier=INPUT   Amplifier IPV4

Optional arguments:
    -h    --help         Show usage
    -p    --amport=INPUT   Amplifier port
    -g    --targport=INPUT   Target port
    -l    --level=INPUT   Attack level
    -c    --timer=INPUT   Attack timer
    -f    --log=INPUT   Log file name
    -i    --inc=INPUT   Increment attack
```

Figura 3.2: Uso do argumento -h (*help*).

3.1.2 Componente *Commander*

O componente Commander é de suma importância, pois é nele que ocorre o planejamento do ataque e a ordem de execução. Nesse componente podemos encontrar quatro módulos: Manager, Executor, *Hall of Mirrors* e *Planner*.

O módulo *Planner* é o responsável por verificar qual será o *mirror* utilizado no ataque, e assim, montar a estrutura do plano de ataque, que contém a função de chamada do *mirror*. O módulo *Hall of Mirrors* contém as funções de chamada para cada ataque, também denominadas de *mirrors*. Cada mirror deve fazer a criação dos pacotes(será discutida logo em breve) correspondentes ao seu ataque que serão injetados, e fazer a chamada do injetor de pacotes.

O módulo *Manager*, sob posse da função de chamada do *mirror*, recebida pelo *Planner*, valida as informações recebidas, e a repassa para o módulo *Executor*. O módulo *Executor* após receber o pacote pronto do *Manager*, dá a ordem de execução do ataque, repassando-o para o componente Netuno.

No programa foram implementados os *mirrors*/amplificadores dos ataques NTP, *Memcached* e SSDP; mas como esse trabalho discute a implementação do ataque DDoS explorando o protocolo NTP, toda a explicação das estruturas bem como do ataque será voltada para esse protocolo, mesmo que as outras implementações sejam muito semelhantes.

Montagem do Pacote

Como o *Hall of Mirrors* é o responsável pela criação do pacote, ele primeiro reúne as informações necessárias, para poder criar o cabeçalho IP, logo depois o UDP, e por último, a mensagem NTP *monlist* é anexada. Portanto, o cabeçalho *Ethernet* não é criado pela ferramenta.

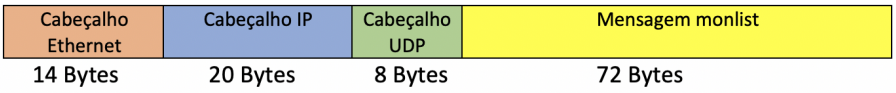

Figura 3.3: Estrutura do Pacote utilizado no ataque.

Na criação do cabeçalho IP, utilizamos a técnica chamada de *IP Spoofing*. Em condições normais, o cabeçalho IP levaria no campo *Source IP Address* do *host* que está enviando o pacote. Contudo, ao realizarmos o *IP Spoofing*, alteramos esse campo e o preenchemos com o valor do IP da vítima. Assim, quando o refletor receber o pacote de requisição, ele irá responder para a vítima, possibilitando assim o ataque.

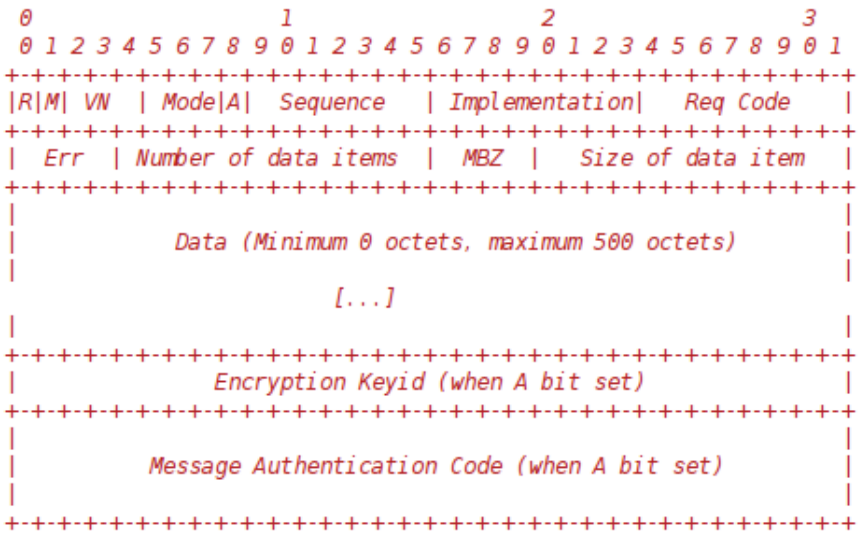

Figura 3.4: Estrutura do Pacote NTP.

Conforme podemos ver na Figura 3.4, o pacote NTP possui muitos campos que podem ser preenchidos com diversas combinações. Para a criação da nossa mensagem *monlist*, precisamos nos preocupar somente com os campos R (*Response Bit*),VN (*Version Num-*

ber), Mod(*Mode*), Implementation(*Implementation Number*) e Req. Code(*Request Code*). Devemos preencher esses campos em decimal da seguinte maneira:

- **R = 0:** indica que o pacote é uma requisição;
- **VN = 2:** indica que utilizamos a versão 2 do NTP;
- **Mod = 7:** indica que é uma operação no modo privado;
- **Implementation = 3:** indica a implementação do XNTPD.
- **Req. Code = 42:** indica o código da requisição *monlist*(MON_GETLIST_1)[45].

Que resultaria em hexadecimal na mensagem 0x1700032a. Todos os outros campos restantes seriam zerados. Após isso, todos os cabeçalhos e a mensagem são juntados para formar um só pacote.

3.1.3 Componente Netuno

O componente Netuno é o responsável por de fato iniciar o ataque, pois é nele que se faz a injeção dos pacotes. O módulo *Injector* é o responsável por criar e destruir as threads que fazem a injeção dos pacotes, regulando a quantidade de pacotes que devem ser enviados. O módulo *Controller* é o responsável por delegar ao Injector quando criar ou destruir novas *threads*, de acordo com as especificações passadas pelo usuário, como intensidade, tempo e se o ataque é incremental ou não. Além disso, o módulo *Logger*, se solicitado, cria um arquivo de *log* do ataque que foi executado, que mostra o tempo de ataque, o nível do ataque, a quantidade de pacotes esperados e a quantidade de pacotes fornecidos.

Nível do Ataque

A quantidade de pacotes injetadas pelo Netuno depende do nível do ataque. O nível varia de 1 a 10, mais uma categoria após o nível 10 considerada como infinito. A quantidade de pacotes por segundo que tentam ser inseridos pelo Netuno, surge através da seguinte equação:

$$QP = 10^{(N-1)} \tag{3.1}$$

Onde,

- **QP** = Quantidade de pacotes;
- **N** = Nível do ataque.

Assim, através da Equação 3.1, podemos construir a Tabela 3.1, onde temos a relação entre o nível e a quantidade de pacotes injetados. É importante ressaltarmos que esses são os valores teóricos esperados que o atacante consiga injetar, não podendo serem garantidos.

Tabela 3.1: Quantidade de pacotes por segundo inseridos a cada Nível.

Nível	**Quantidade de Pacotes**
1	1
2	10
3	100
4	1000
5	10000
6	100000
7	1000000
8	10000000
9	100000000
10	1000000000
>10	>1000000000

3.1.4 Execução

Para executarmos o ataque, devemos rodar o programa com privilégios elevados (sudo), e como dito anteriormente, informar ao menos o *mirror* que será utilizado, o IP do amplificador e o IP da vítima. Um exemplo de execução seria através do comando: "sudo ./bin/lhf -m ntp -a 192.168.0.102 -p 123 -t 192.168.0.103 -g 123", que é a base para o ataque com a ferramenta Linderhof utilizando o mirror NTP. Na Figura 3.5, podemos ver a tela de inicialização desse ataque.

```
alexander@alexander-MacBookPro:~/Desktop/Linderhof$ sudo ./bin/lhf -m ntp
 -a 192.168.0.102 -p 123 -t 192.168.0.103 -g 123
Welcome to Linderhof!
Attack configuration
Mirror:         NTP
Target ip:      192.168.0.103
Target port:    123
Amplifier ip:   192.168.0.102
Amplifier port: 123
################################################################
```

Figura 3.5: Tela de inicialização de ataque.

Se não for atribuído nenhum valor às *flags* dos parâmetros nível, timer ou inc, o ataque ocorre no estilo pré-definido, na sua maneira mais simples, que é o ataque funcionando

no nível 1, onde a cada segundo, 1 pacote é enviado pela ferramenta. O ataque no modo *default* está presente na Figura 3.6.

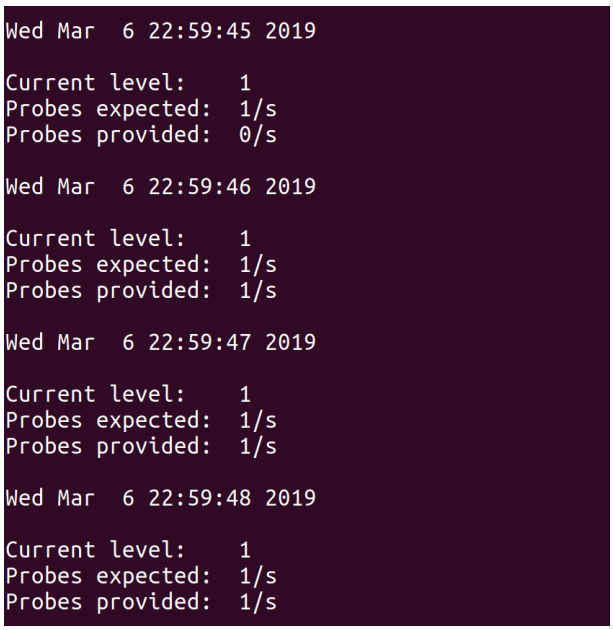

Figura 3.6: Ataque no modo *default*.

Se atribuirmos algum valor apenas ao parâmetro nível, o ataque ocorre de maneira semelhante ao *default*, só que com a quantidade de pacotes do nível correspondente sendo injetados, ao invés de apenas um. Se atribuirmos algum valor apenas o parâmetro timer, o programa executa durante o tempo estabelecido como parâmetro, e encerra a execução quando atingir o tempo. Se for atribuído algum valor apenas ao parâmetro inc, o ataque ocorre de maneira semelhante ao default, só que com o valor do nível sendo incrementado em uma unidade após a quantidade de vezes que foi passada no argumento inc, e os pacotes injetados por consequência sendo reajustados ao nível correspondente, até que se passe do nível 10, onde a quantidade de pacotes injetados pode ser de qualquer valor, mas necessariamente maior que a anterior.

Capítulo 4

Testes e Resultados

Neste capítulo são explicitados os dados obtidos nos testes realizados, através do uso da ferramenta de ataque Linderhof, utilizando o módulo de ataque NTP. Primeiramente vamos expor a especificação dos equipamentos utilizados durante toda a simulação, e logo após teremos a explicação e resultado dos testes que foram executados.

4.1 Equipamentos Utilizados

Para a realização deste trabalho, todos os testes foram realizados em um ambiente controlado, onde não haveria nenhuma ameaça para o ambiente exterior. Por isso, seguindo a disposição de máquinas presente na Figura 4.1, foi necessário que uma rede paralela fosse criada. Os dispositivos utilizados durante toda a abordagem foram:

- **Atacante:**

Tabela 4.1: Especificação do Dispositivo Atacante.

	Configuração
Marca	Apple
Modelo	MacBook Pro
Endereço IP	192.168.0.101
Função	Executar o programa Linderhof, originando o ataque
Sistema Operacional	Ubuntu 16.04 LTS
Processador	3,5 GHz Intel Core i7
HD	SSD 500 GB
Memoria RAM	16 GB 2133 MHz LPDDR3
Adaptador de Rede	Type C Hub with Ethernet Adapter - Velocidade 100 Mbps

- **Refletor/Amplificador:**

Tabela 4.2: Especificação do Dispositivo Refletor e Amplificador.

	Configuração
Marca	Dell
Modelo	Inspiron 14R 690
Endereço IP	192.168.0.102
Função	Refletir e Amplificar o ataque, hospeda um servidor NTP
Sistema Operacional	Ubuntu 14.04 LTS
Processador	2,40 GHz Intel Core i3-520M
HD	500 GB
Memoria RAM	4 GB 1333 MHz DDR3 SDRAM
Adaptador de Rede	AR8152 PCE-e Fast Ethernet Controller - Velocidade 100 Mbps

- **Vítima:**

Tabela 4.3: Especificação do Dispositivo Vítima.

	Configuração
Marca	Dell
Modelo	Inspiron 14R 3040
Endereço IP	192.168.0.103
Função	Vítima, receber todos os pacotes gerados pelo refletor
Sistema Operacional	Windows 10 Professional
Processador	2,30 GHz Intel Core i5-2410M
HD	1 TB
Memoria RAM	4 GB 1333 MHz DDR3 SDRAM
Adaptador de Rede	RTL 8139/810x Fast Ethernet - Velocidade 100 Mbps

- **Roteador:**

Tabela 4.4: Especificação do Roteador.

	Especificação
Marca	Multilaser
Modelo	RE024 Wireless N
Velocidade	150 Mbps

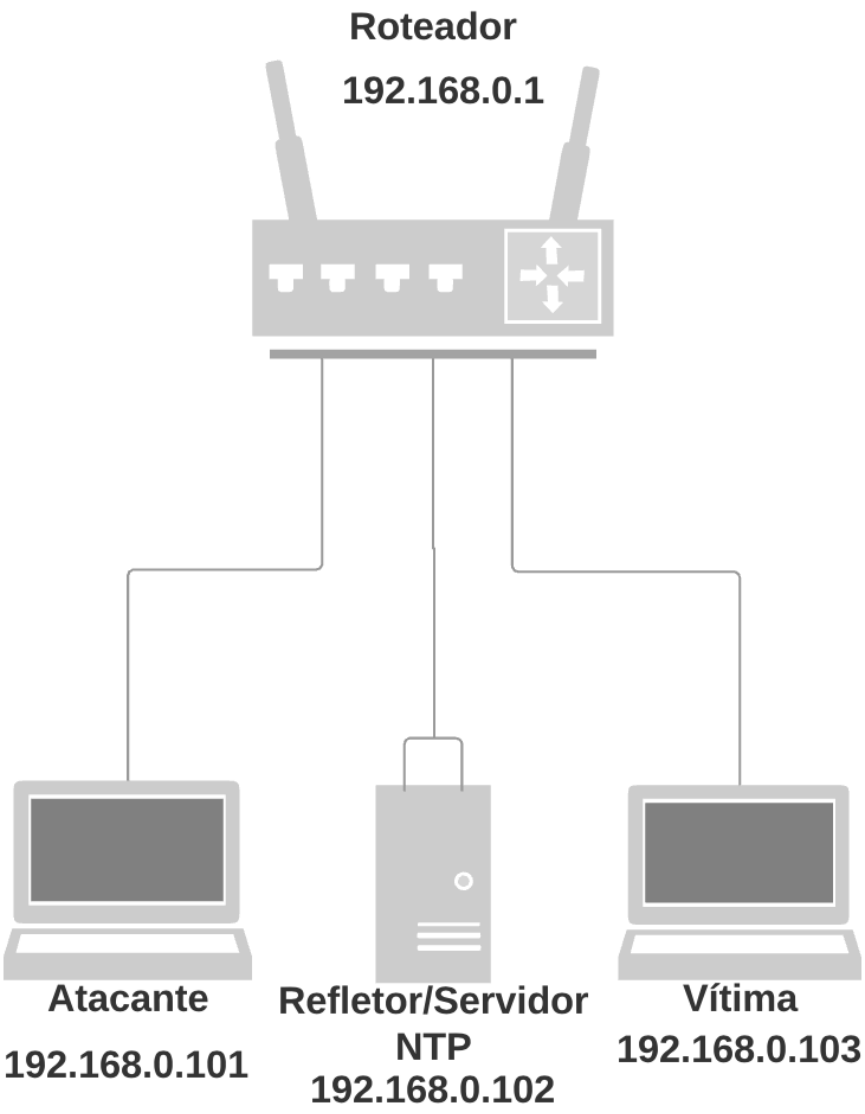

Figura 4.1: Estrutura da rede utilizada para os ataques.

4.2 Teste de Funcionalidade e Performance

O teste de funcionalidade e performance visa comprovar os princípios do ataque de reflexão e amplificação, bem como os limites de envio de pacotes pela ferramenta. A reflexão ocorre quando o *IP Spoofing* é bem sucedido, ou seja, a vítima irá receber o pacote de resposta de uma requisição, pois o cabeçalho IP da requisição foi adulterado. A amplificação ocorre quando envia-se um pacote com um tamanho X, e a vítima o recebe com um tamanho X vezes Y, onde Y seria o fator de amplificação, sendo qualquer valor maior que 1.

Já os limites de envio de pacotes ocorrem quando o atacante ou o amplificador não conseguem injetar a quantidade de pacotes que era esperada que eles injetassem na rede, e

conseguimos visualizar esse limite. Se esses princípios puderem ser observados, bem como a visualização da limitação de injeção dos pacotes pelas ferramentas, podemos concluir que o a ferramenta passou no teste, e assim ela pode ser utilizada para um ataque de negação de serviço explorando o protocolo NTP.

4.2.1 Configuração e Execução

Foram realizados testes baseados em três configurações. Na primeira, o servidor NTP possuía em sua tabela de últimos dispositivos conectados 60 *hosts*. Na segunda, 300 *hosts*, e por último, na terceira, 600 *hosts*. Nos testes realizados os argumentos utilizados junto ao Linderhof foram "-i 5 -c 50", para que ele executasse o ataque do nível 1 ao 10, e de forma gradual (a cada 5 segundos), fizesse a troca de nível para o subsequente, aumentando assim o número de pacotes injetados na rede. Para que o tráfego pudesse ser capturado e analisado, foi utilizada a ferramenta de Capturas Wireshark.

Um servidor NTP real teria uma lista com inúmeros *hosts* que se conectaram a ele. Entretanto, ficaria impraticável em um ambiente de testes configurarmos 60, 300 ou no pior dos casos 600 *hosts* para sincronizarem seus relógios com esse servidor. Por esse motivo, utilizamos de scripts[46] que enviam falsas requisições de updates, para que a tabela do servidor fosse preenchida conforme as nossas necessidades.

Para cada configuração foram realizadas 4 rodadas de ataques, as quais obtemos as estatísticas, e realizamos a média aritmética para a obtenção dos resultados. Assim, os valores apresentados subsequentemente serão sempre baseados na média dos quatro ataques e foram considerados valores com até duas casas decimais.

4.2.2 Configuração com 60 *Hosts*

Nessa configuração, a tabela do servidor NTP foi preenchida com 60 *hosts*, ou seja, apenas dez por cento da capacidade total de *hosts* que a tabela pode armazenar.

Atacante

Na Tabela 4.5 temos a quantidade de pacotes esperados que fossem injetados pelo atacante, bem como a quantidade real de pacotes e *bytes* que foram injetados por ele nessa primeira configuração durante 50 segundos.

Tabela 4.5: Pacotes e *Bytes* Injetados pelo computador atacante durante 50 segundos.

Nível	**Quant. Teorica de Pacotes Enviados**	**Pacotes Enviados**	**Bytes Enviados**
1	1	1	114
2	10	10	1140
3	100	100	11400
4	1000	1000	114000
5	10000	9999	1139886
6	100000	17224	1963604
7	1000000	73105	8333998
8	10000000	86479	9858674
9	100000000	87270	9948803
10	1000000000	81850	9330951

Refletor

A Tabela 4.6 mostra a quantidade de pacotes e *bytes* recebidos no refletor, bem como os pacotes e *bytes* que ele enviou como resposta às solicitações durante os 50 segundos de ataque.

Tabela 4.6: Quantidade de Pacotes recebidos e enviados pelo refletor.

Nível	**Pacotes Recebidos**	**Bytes Recebidos**	**Pacotes Enviados**	**Bytes Enviados**
1	1	114	10	4820
2	10	1140	100	48200
3	100	11400	533	256761
4	1000	114000	575	277174
5	9999	1139886	828	399144
6	17112	1950836	1310	631589
7	72930	8314037	2221	1070835
8	86572	9869253	2241	1080403
9	87337	9956418	2380	1147473
10	82059	9354743	2489	1200059

Vítima

A Tabela 4.7 mostra a quantidade de pacotes e *bytes* recebidos pela vítima.

Tabela 4.7: Quantidade de Pacotes e *Bytes* recebidos pela vítima.

Nível	Pacotes Recebidos	Bytes Recebidos
1	10	4940
2	91	43814
3	491	236614
4	559	269341
5	813	391842
6	1241	598330
7	2158	1040156
8	2272	1095104
9	2401	1157523
10	2411	1162102

Estatísticas

Na Figura 4.2 e na Figura 4.3, podemos ver as representações gráficas em *bytes* por nível e *frames* por nível do ataque, respectivamente.

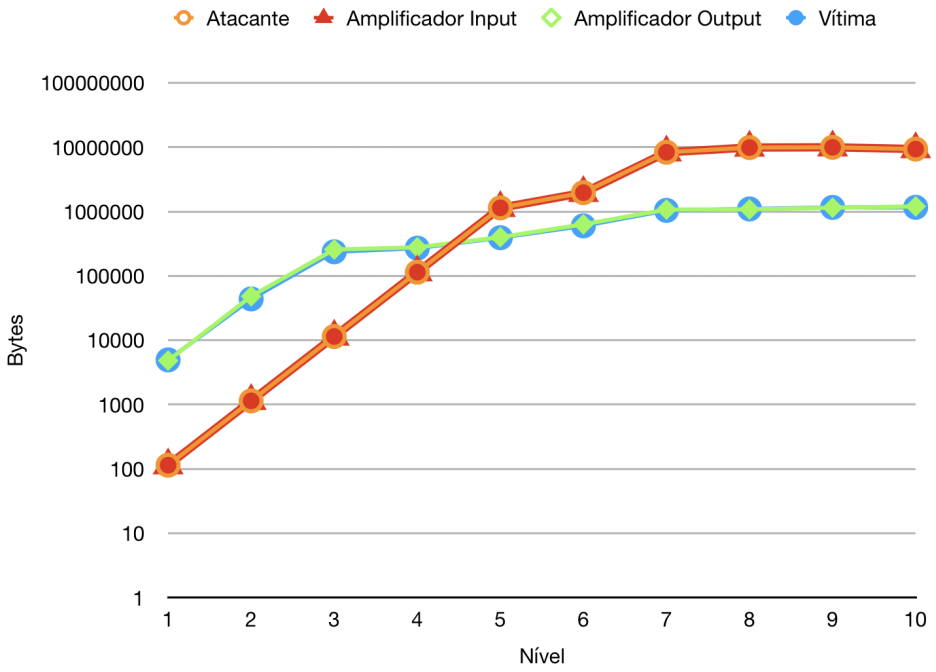

Figura 4.2: Representação do ataque em *bytes* por nível para 60 *hosts*.

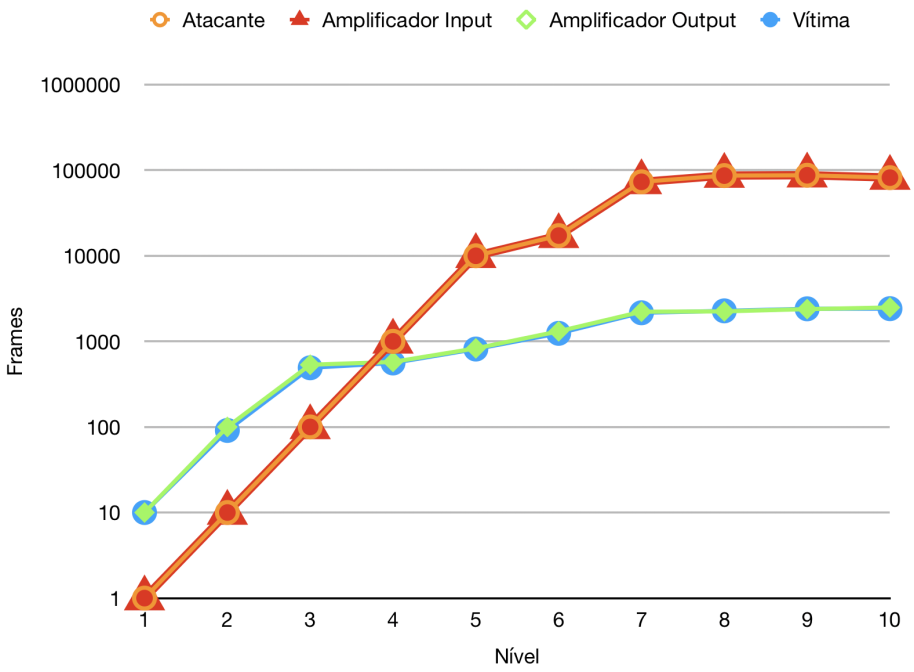

Figura 4.3: Representação do ataque em *frames* por nível para 60 *hosts*.

4.2.3 Configuração com 300 *Hosts*

Nessa configuração, a tabela do NTP foi preenchida com 300 *hosts*. Teoricamente, nessa configuração o refletor deveria gerar um fluxo de pacotes maior, pelo fato das respostas às requisições *monlist* se dividirem em uma quantidade maior de pacotes.

Atacante

A tabela Tabela 4.8 mostra a quantidade de pacotes e *bytes* injetados pelo computador atacante nessa configuração durante 50 segundos. É interessante notarmos que a partir do nível 6, o computador atacante não consegue injetar a quantidade de pacotes que eram esperados que fossem injetados por ele.

Tabela 4.8: Quantidade de Pacotes e *Bytes* Injetados pelo computador atacante na configuração com 300 *hosts*.

Nível	Quant. Teorica de Pacotes Enviados	Pacotes Enviados	Bytes Enviados
1	1	1	114
2	10	10	1140
3	100	100	11400
4	1000	1000	114000
5	10000	9999	1139886
6	100000	17816	2031058
7	1000000	71870	8193248
8	10000000	86537	9865218
9	100000000	87162	9936513
10	1000000000	81731	9317305

Refletor

A Tabela 4.9 nos mostra os dados relativos ao refletor nessa segunda configuração. Podemos observar que a partir do nível 4, o refletor passa a receber uma quantidade maior de pacotes do que consegue de fato enviar.

Tabela 4.9: Quantidade de Pacotes recebidos e enviados pelo refletor na configuração com 300 *hosts*.

Nível	Pacotes Recebidos	Bytes Recebidos	Pacotes Enviados	Bytes Enviados
1	1	114	50	24100
2	10	1140	417	201163
3	100	11400	551	265750
4	1000	114000	593	285874
5	9999	1139886	826	398035
6	17816	2031058	1259	607199
7	71844	8190227	2060	992920
8	86543	9865936	2146	1034540
9	85642	9763222	2108	1016297
10	78788	8981849	1974	951733

Vítima

A Tabela 4.10 mostra a quantidade de pacotes e *bytes* recebidos pela vítima.

Tabela 4.10: Quantidade de Pacotes e *Bytes* recebidos pela vítima na configuração com 300 *hosts*.

Nível	Pacotes Recebidos	Bytes Recebidos
1	67	32221
2	422	203452
3	559	269390
4	603	290863
5	849	409604
6	1309	631034
7	2197	1058906
8	2158	1040348
9	2161	1041819
10	1801	868251

Estatísticas

Na Figura 4.4 podemos ver a representação gráfica em *bytes* por nível do ataque.

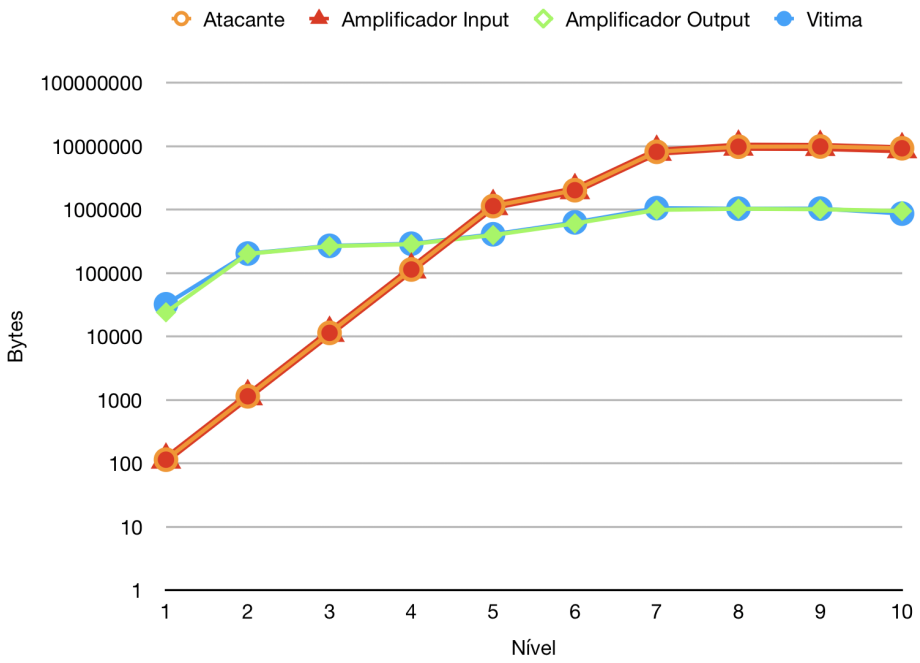

Figura 4.4: Representação do ataque em *bytes* por nível para 300 *hosts*.

Já na Figura 4.5 podemos ver a representação gráfica em *frames* por nível do ataque.

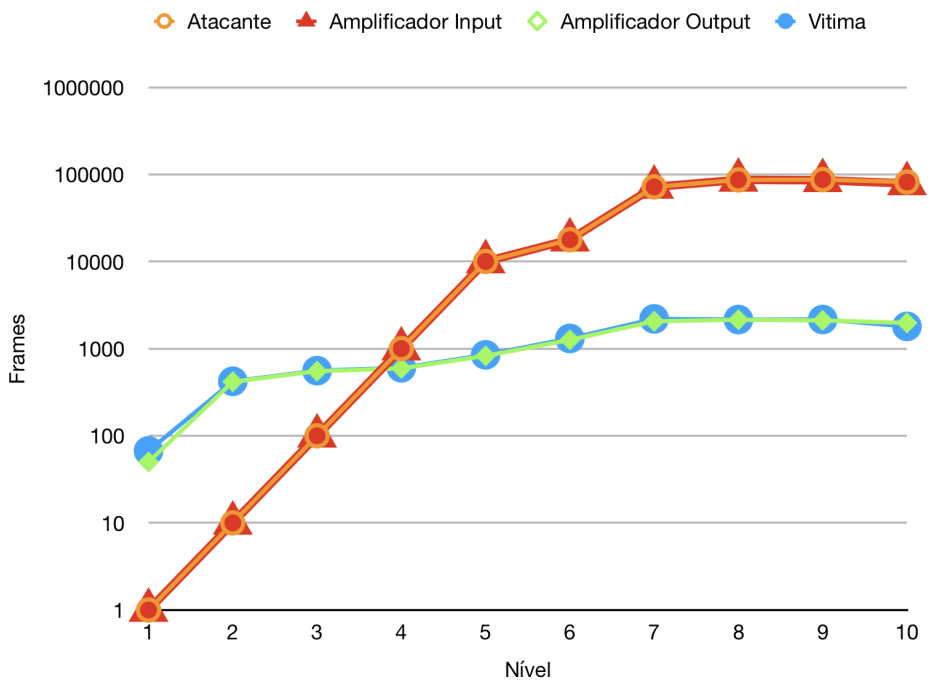

Figura 4.5: Representação do ataque em *frames* por nível para 300 *hosts*.

4.2.4 Configuração com 600 *Hosts*

Na última configuração, a tabela do NTP foi preenchida com 600 *hosts*. Teoricamente, seria essa a configuração mais ameaçadora, onde se poderia obter as melhores taxas de amplificação, resultando em ataques mais volumosos.

Atacante

Na Tabela 4.11, através dos dados explicitados, podemos ver que se repetiu o comportamento das configurações com 60 e 300 *hosts*, onde o computador atacante não consegue injetar a quantidade de pacotes que eram esperados que fossem injetados por ele a partir do nível 6.

Tabela 4.11: Quantidade de Pacotes e *Bytes* Injetados pelo computador atacante na configuração com 600 *hosts*.

Nível	Quant. Teorica de Pacotes Enviados	Pacotes Enviados	Bytes Enviados
1	1	1	114
2	10	10	1140
3	100	100	11400
4	1000	1000	114000
5	10000	9999	1139886
6	100000	17247	1966152
7	1000000	71835	8189247
8	10000000	86731	9887368
9	100000000	87337	9956395
10	1000000000	82165	9366878

Refletor

Na terceira configuração, a Tabela 4.12 mostra que analogamente à configuração com 60 e 300 *hosts*, a partir do nível 4, o refletor não consegue enviar mais pacotes do que a quantidade que chegou a ele.

Tabela 4.12: Quantidade de Pacotes recebidos e enviados pelo refletor na configuração com 600 *hosts*.

Nível	Pacotes Recebidos	Bytes Recebidos	Pacotes Enviados	Bytes Enviados
1	1	114	100	48200
2	10	1140	511	246205
3	100	11400	562	271173
4	1000	114000	586	282524
5	9999	1139886	816	393336
6	17247	1966152	1170	564205
7	71808	8186100	2561	1234402
8	84240	9603343	2633	1269178
9	87343	9957096	2785	1342370
10	82188	9369460	2851	1374375

Vítima

A Tabela 4.13 mostra a quantidade de pacotes e *bytes* recebidos pela vítima.

Tabela 4.13: Quantidade de Pacotes e *Bytes* recebidos pela vítima na configuração com 600 *hosts*.

Nível	Pacotes Recebidos	Bytes Recebidos
1	103	49892
2	492	237602
3	551	265775
4	588	283608
5	807	388901
6	1131	545431
7	2597	1251754
8	2745	1323331
9	2827	1362686
10	2681	1292435

Estatísticas

Na Figura 4.6 e na Figura 4.7, podemos ver as representações gráficas em *bytes* por nível e *frames* por nível do ataque, respectivamente.

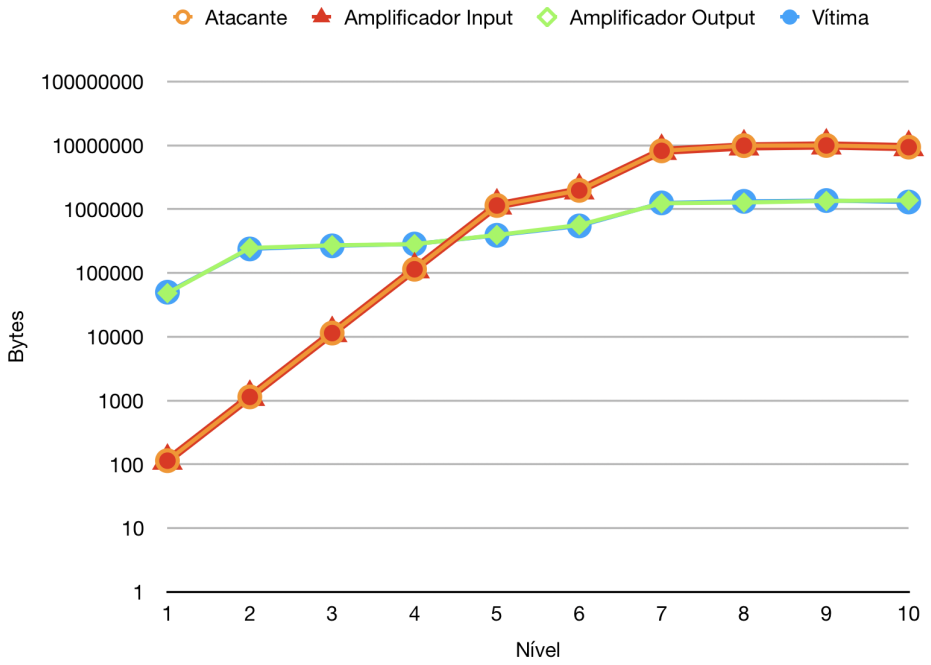

Figura 4.6: Representação do ataque em *bytes* por nível para 600 *hosts*.

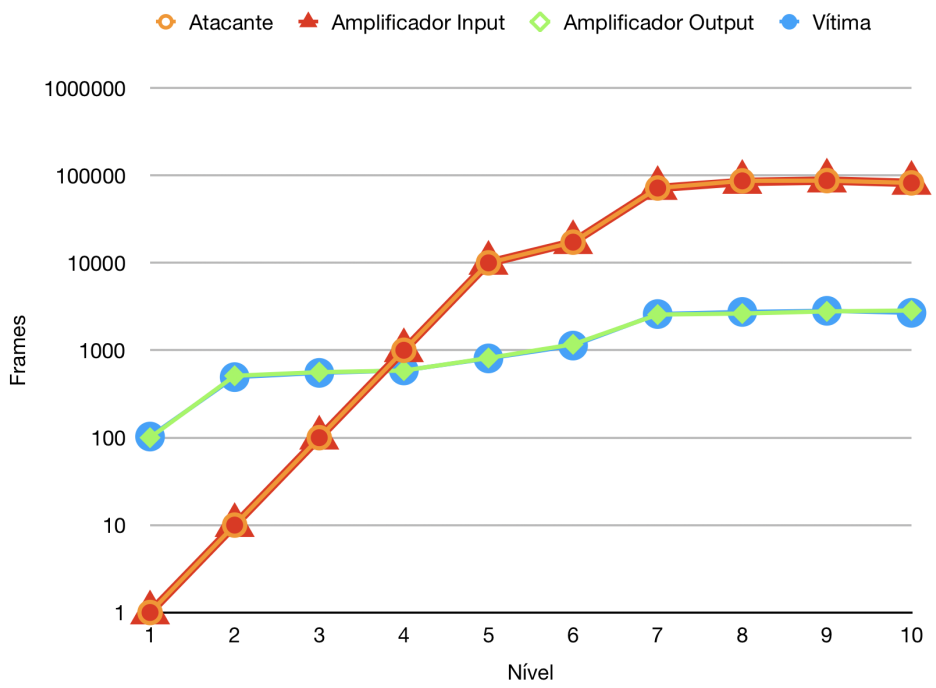

Figura 4.7: Representação do ataque em *frames* por nível para 600 *hosts*.

4.3 Amplificação

A seguir, tem-se a tabela com os valores correspondentes da amplificação em cada um dos níveis do ataque.

4.3.1 Amplificação dos *Bytes*

No.	Time	Source	Destination	Protocc	Length	Info
1	0.000000	192.168.0.103	192.168.0.102	NTP	114	NTP Version 2, private
2	0.001321	192.168.0.102	192.168.0.103	NTP	482	NTP Version 2, private
3	0.001324	192.168.0.102	192.168.0.103	NTP	482	NTP Version 2, private
4	0.001326	192.168.0.102	192.168.0.103	NTP	482	NTP Version 2, private
5	0.001328	192.168.0.102	192.168.0.103	NTP	482	NTP Version 2, private
6	0.001330	192.168.0.102	192.168.0.103	NTP	482	NTP Version 2, private
7	0.001331	192.168.0.102	192.168.0.103	NTP	482	NTP Version 2, private
8	0.001333	192.168.0.102	192.168.0.103	NTP	482	NTP Version 2, private
9	0.001335	192.168.0.102	192.168.0.103	NTP	482	NTP Version 2, private
10	0.001336	192.168.0.102	192.168.0.103	NTP	482	NTP Version 2, private
11	0.001338	192.168.0.102	192.168.0.103	NTP	482	NTP Version 2, private

Figura 4.8: Captura no refletor no nível 1.

Produziu-se a Tabela 4.14 através da Equação 2.2. Tomemos como exemplo a configuração com 60 *hosts*, no nível 1 do ataque. Através da captura mostrada na Figura 4.8,

podemos ver que o tamanho do pacote NTP enviado ao refletor é de 114 *bytes*, onde dos 114 *bytes*:

- 14 são cabeçalhos *Ehternet*;
- 20 são cabeçalhos IP;
- 8 são cabeçalhos UDP;
- 74 são da requisição *MON_GETLIST_1*.

Além disso, na própria Figura 4.8 podemos ver a resposta à requisição *MON_GETLIST_1* dividida em 10 pacotes de 482 *bytes*, onde em cada um desses:

- 14 são cabeçalhos *Ethernet*;
- 20 são cabeçalhos IP;
- 8 são cabeçalhos UDP;
- 440 são dados referentes aos *hosts* presentes na tabela do servidor NTP.

Assim, tendo como base a Equação 2.2, entretanto modificando-a pois a resposta é não somente em um pacote, mas sim dividida entre dez, podemos calcular a amplificação da seguinte maneira:

$$\begin{aligned} BAF &= \frac{10 * (SizeIP(resp) + SizeUDP(resp) + SizePayload(resp))}{SizeIP(req) + SizeUDP(req) + SizePayload(req)} \\ &= \frac{10 * (20 + 8 + 440)}{20 + 8 + 72} \\ &= \frac{4680}{100} \\ &= 46,8 \end{aligned} \tag{4.1}$$

E assim, repetindo esse cálculo para os dez níveis, nas três configurações, podemos produzir a Tabela 4.14 que se segue.

Tabela 4.14: Tabela de Amplificação de banda pelo refletor.

	Número de Hosts		
Nível	**60**	**300**	**600**
1	46,80	234,00	468,00
2	46,80	195,32	239,05
3	24,93	25,80	26,33
4	2,69	2,78	2,74
5	0,39	0,39	0,38
6	0,36	0,33	0,32
7	0,14	0,13	0,17
8	0,12	0,12	0,15
9	0,13	0,12	0,15
10	0,14	0,12	0,16

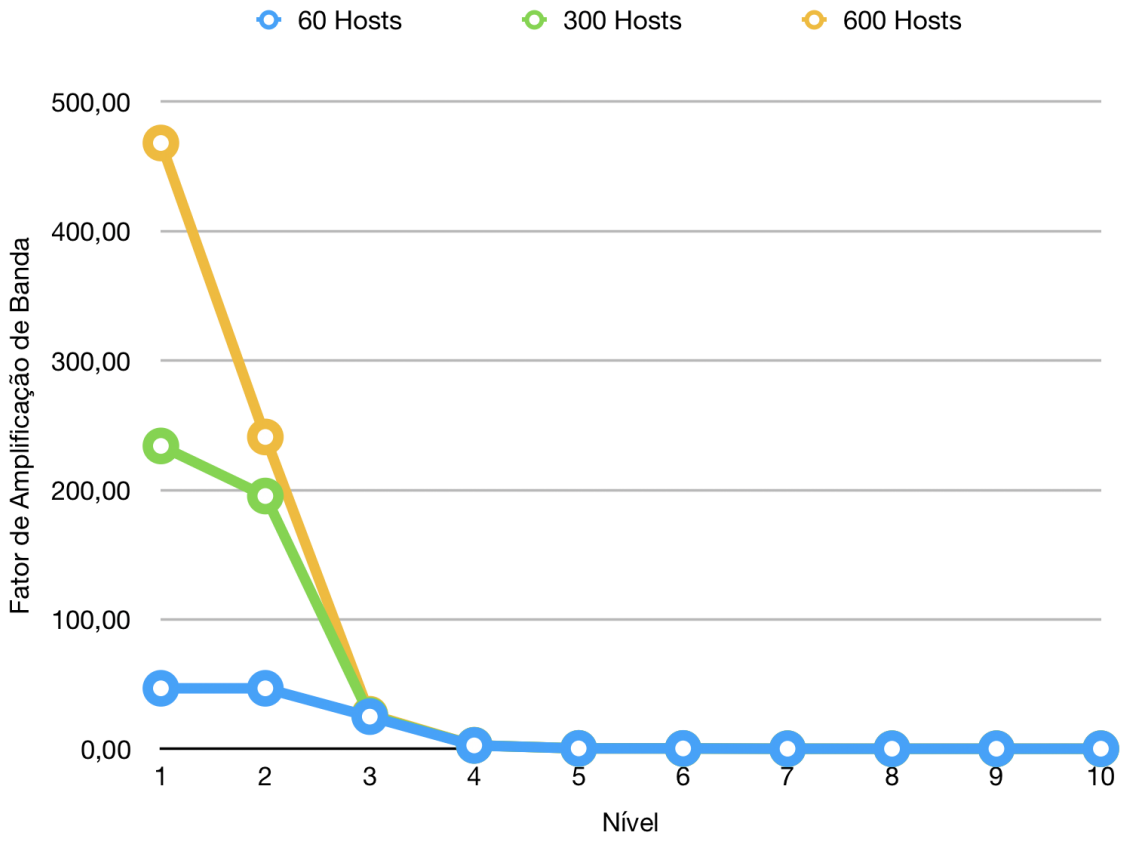

Figura 4.9: Amplificação de banda por nível.

4.3.2 Amplificação dos Pacotes

Podemos calcular a amplificação dos pacotes dividindo a quantidade de pacotes enviados à vítima pelo refletor, pela quantidade de pacotes recebidos pelo refletor, em um mesmo nível. O que poderia ser representado pela Equação 4.3:

$$FAF = \frac{QF(resp)}{QF(rec)} \tag{4.2}$$

Onde,

- **FAF** = Fator de amplificação de pacotes;
- **QF(rec)** = Quantidade de pacotes recebidos pelo refletor;
- **QF(resp)** = Quantidade de pacotes enviados pelo refletor

Tomemos como exemplo a configuração com 300 *hosts*, no nível 2 do ataque. O refletor recebe 10 pacotes, e envia 417. Assim, através da Equação 4.3, temos:

$$\begin{aligned} FAF &= \frac{QF(resp)}{QF(rec)} \\ &= \frac{417}{10} \\ &= 41,70 \end{aligned} \tag{4.3}$$

E assim, repetindo esse cálculo para os dez níveis, nas três configurações, podemos produzir a Tabela 4.15 que se segue.

Tabela 4.15: Tabela de Amplificação dos Pacotes pelo refletor.

	Número de *Hosts*		
Nível	**60**	**300**	**600**
1	10,00	50,00	100,00
2	10,00	41,70	51,10
3	5,33	5,51	5,62
4	0,58	0,59	0,59
5	0,08	0,08	0,08
6	0,08	0,07	0,07
7	0,03	0,03	0,04
8	0,03	0,02	0,03
9	0,03	0,02	0,03
10	0,03	0,03	0,03

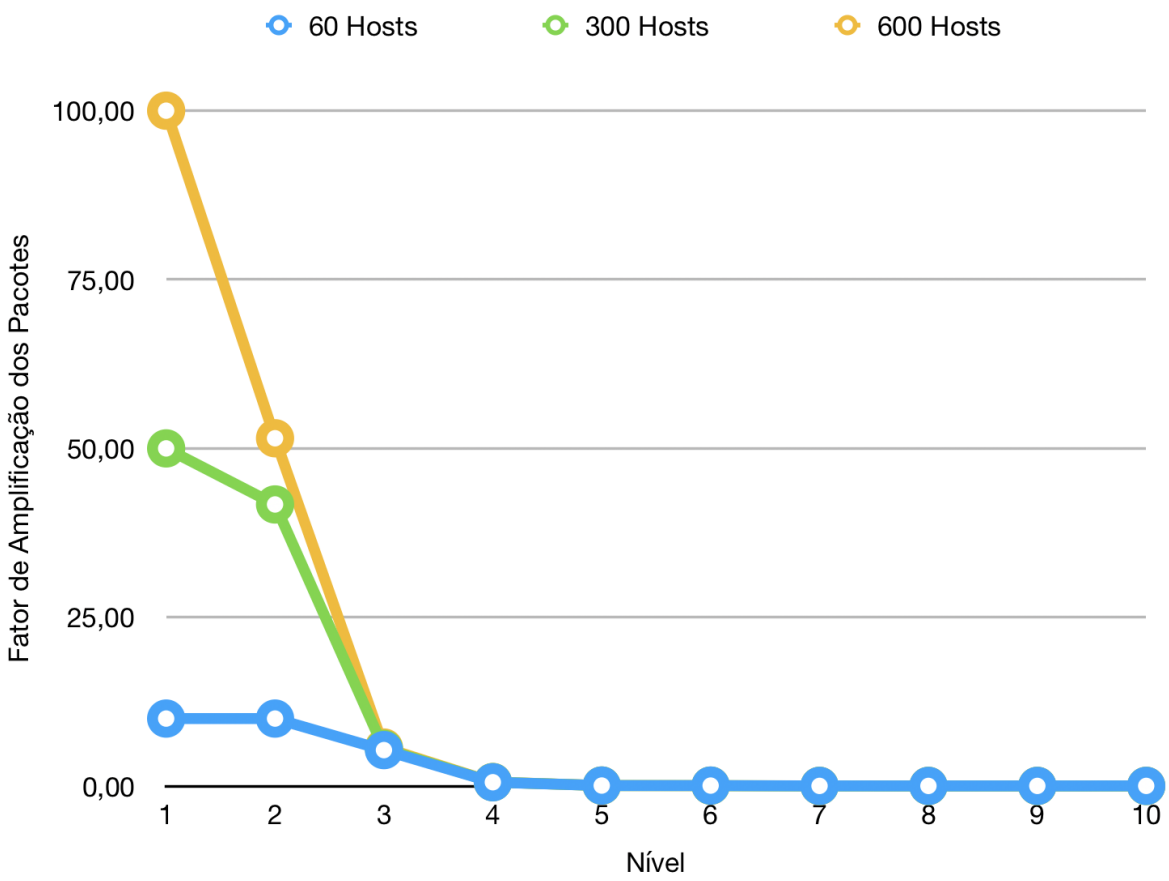

Figura 4.10: Amplificação de banda por nível.

4.3.3 Análise da Amplificação

Como uma amplificação só se tem quando aumentamos a quantidade de um valor, consideramos que houve amplificação quando o valor da mesma for maior ou igual a um. Partindo desse pressuposto, podemos concluir que para todas as configurações só houve amplificação da largura de banda pelo refletor até o nível 4, conforme disposto na Tabela 4.14. No nível inicial, melhor visto na Figura 4.9, foi onde obteve-se os melhores valores de amplificação, pois após ele pode ser observado que houve uma atenuação muito forte na amplificação, até chegar ao ponto da saturação no nível 5, onde não houve mais a amplificação de banda.

Na melhor configuração (600 *hosts*), o nosso refletor obteve uma amplificação máxima de 468,00x, o que, frente ao valor de 556.9x obtido por Russow[16], é um excelente número, pois se tivéssemos utilizado o mesmo método de cálculo de fator de amplificação de banda que Russow utilizou, presente na Equação 2.1, teríamos obtido um fator de amplificação de banda de 611.11x, ou seja, superior ao fator obtido por ele.

De maneira quase análoga, conforme a Tabela 4.15 e a Figura 4.10, houve amplificação

dos pacotes em todas as configurações, entretanto, somente entre os níveis um ao três. Do quarto em diante, houve saturação, e o refletor não conseguiu injetar uma quantidade maior de pacotes do que a recebida. A saturação tanto na amplificação dos pacotes quando da banda pode ser atribuída a uma limitação de hardware no computador refletor, pois o mesmo dentre os três computadores, é aquele que dispõe de menos recursos computacionais.

4.4 Análise da Performance

A análise de performance será dividida da seguinte maneira:

- Análise da performance do computador de ataque;
- Análise da performance do refletor;
- Análise dos dados na vítima;
- Análise da quantidade de *hosts*.

4.4.1 Análise de performance do Atacante

Não pode-se dizer que o atacante exerce um mesmo esforço computacional que o refletor, pois ele não recebe pacotes. Entretanto, ele possui um esforço maior que a vítima, pois além de injetar os pacotes NTP com as requisições *monlist*, ele precisa os preparar, envolvendo um certo esforço computacional que não pode ser descartado.

Conforme pode ser observado nas três configurações, através das tabelas apresentadas anteriormente, o atacante a partir do nível 6 já não consegue injetar a quantidade de pacotes esperados que fossem injetados por ele. Contudo, mesmo sem injetar a quantidade de pacotes ideal, o atacante só chega de fato a saturar a partir do nível 8 em diante, pois a injeção dos pacotes começa a apresentar valores muito próximos, ficando praticamente constante.

4.4.2 Análise de performance do Refletor

O refletor é o responsável por receber o tráfego dos pacotes com as requisições *monlist* geradas pelo atacante, processar as requisições e consequentemente amplificá-las, enviando as respostas para a vítima. Nas três configurações, o refletor conseguiu fazer a amplificação dos pacotes até o nível 3. A partir do nível 4 em diante, a quantidade de pacotes que chegava no refletor foi maior do que a quantidade que saia dele para todas as configurações.

Assim, conforme podemos ver na Tabela 4.15, não houve amplificação dos pacotes a partir do nível 4.

De maneira quase análoga, conforme os dados presentes na Tabela 4.14, para todas as configurações, houve amplificação de banda pelo refletor, só que agora, até o nível 4. Embora o refletor tenha uma amplificação de banda maior que um, ele possui um limite para a amplificação dos pacotes, pois o processamento da requisição *monlist* não é feita de maneira instantânea, e consequentemente ele perde tempo as processando. Dessa maneira, a amplificação dos pacotes não consegue acompanhar a amplificação dos *bytes*, e satura um nível antes.

Outro ponto importante é que a ferramenta de ataque Linderhof tenta injetar uma grande quantidade de pacotes da maneira mais rápida possível. Por consequência, o servidor NTP, por já estar sobrecarregado processando os pacotes que já haviam chegado anteriormente, e preparando as suas respostas, pode acabar descartando vários pacotes.

As amplificações mais significativas podem ser vistas até o nível 3, pois depois disso, as mesmas caíram substancialmente. Já a saturação do refletor ocorreu por volta do nível 8, quando os refletores passaram a responder com uma quantidade de pacotes praticamente iguais até o final do ataque.

4.4.3 Análise da Vítima

A vítima somente recebe os pacotes provenientes do refletor. Conforme podemos ver nos gráficos que representam os *bytes* por nível de cada configuração, presentes na Figura 4.2, Figura 4.4 e Figura 4.6; a maioria dos *bytes* que saem do refletor, chegam na vítima.

4.4.4 Análise da quantidade de *Hosts*

Nas Figuras 4.11 a 4.16 pode ser observado como a quantidade de *hosts* presentes no servidor NTP, dependendo do nível utilizado para o ataque, pode influenciar na proporção do ataque gerado. Se utilizarmos a ferramenta de ataque entre os níveis 1 a 2, ou do 7 em diante, a configuração mais vantajosa de ser utilizada é a com 600 *hosts* na tabela do servidor. Como nessa configuração o pacote de resposta de uma requisição *monlist* é dividido em 100 pacotes, conseguimos obter as melhores taxas de amplificações, tanto de bytes quanto de *frames*, conforme pode ser visto na Figura 4.13 e na Figura 4.16, .

Entretanto, se o ataque fosse executado em um nível constante entre 3 e 6, os valores de amplificações gerados por cada configuração seriam muito próximos, permitindo assim uma opção de escolha que poderia ser vantajosa ao atacante. Como nem sempre os servidores NTP estão com suas tabelas totalmente preenchidas, ao realizar o scanner de

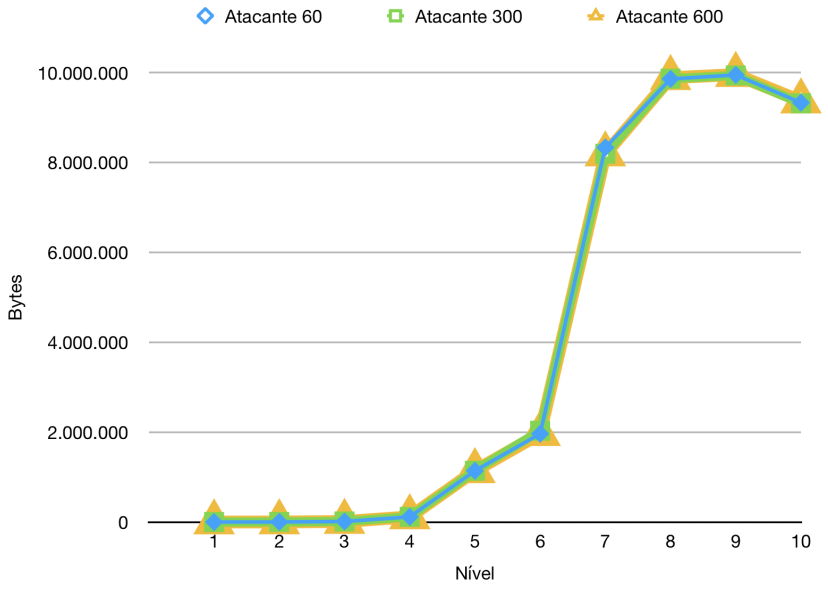

Figura 4.11: Quantidade de *bytes* por nível que saem do atacante para 60, 300 e 600 *Hosts*.

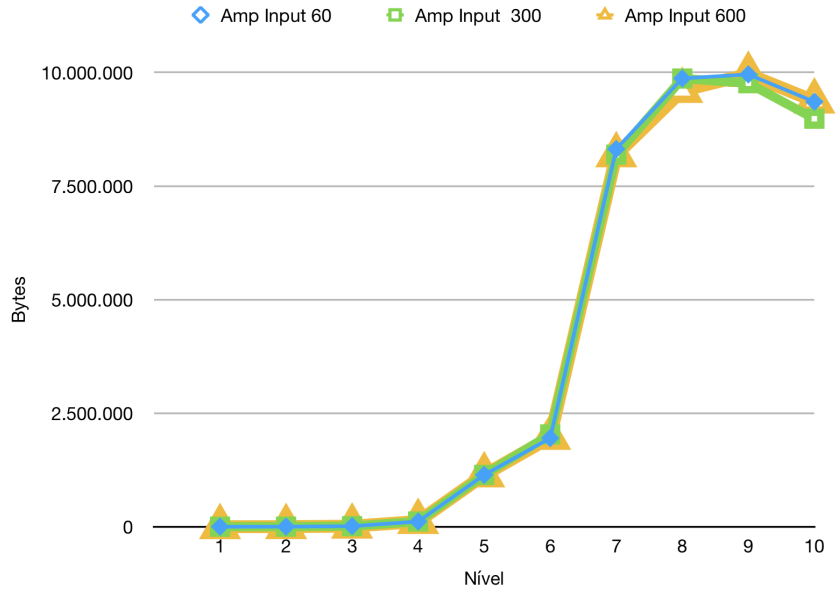

Figura 4.12: Quantidade de *bytes* por nível que chegam no refletor para 60, 300 e 600 *Hosts*.

rede em busca de servidores abertos, o atacante não teria a obrigatoriedade de somente encontrar servidores que estivessem com a tabela completa.

Assim, segundo os dados obtidos através do teste executado, um servidor que tivesse 60, 300 ou 600 *hosts* em sua tabela, para os níveis de 3 a 6, geraria praticamente a mesma

quantidade de amplificação de *bytes* e pacotes. Por consequência, o atacante teria uma maior facilidade para a preparação do ataque, já que o êxito na geração de tráfego do seu ataque não dependeria de uma quantidade alta de *hosts* nos servidores encontrados.

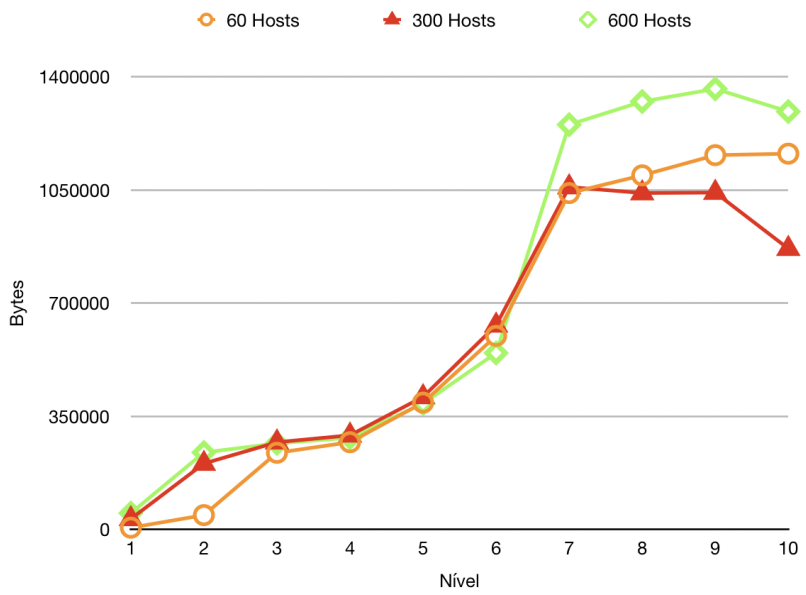

Figura 4.13: Quantidade de *bytes* por nível que chegam na vítima para 60, 300 e 600 *Hosts*.

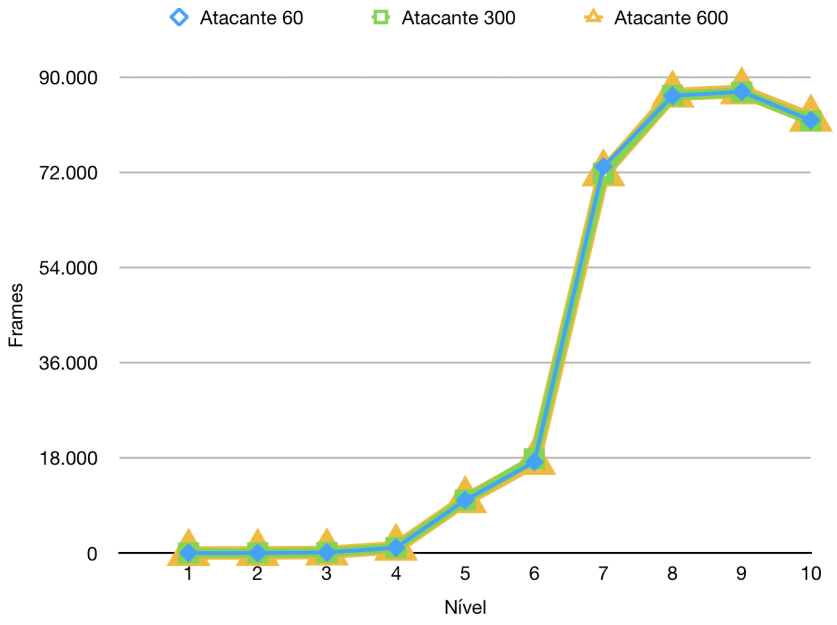

Figura 4.14: Quantidade de pacotes por nível que saem do atacante para 60, 300 e 600 *Hosts*.

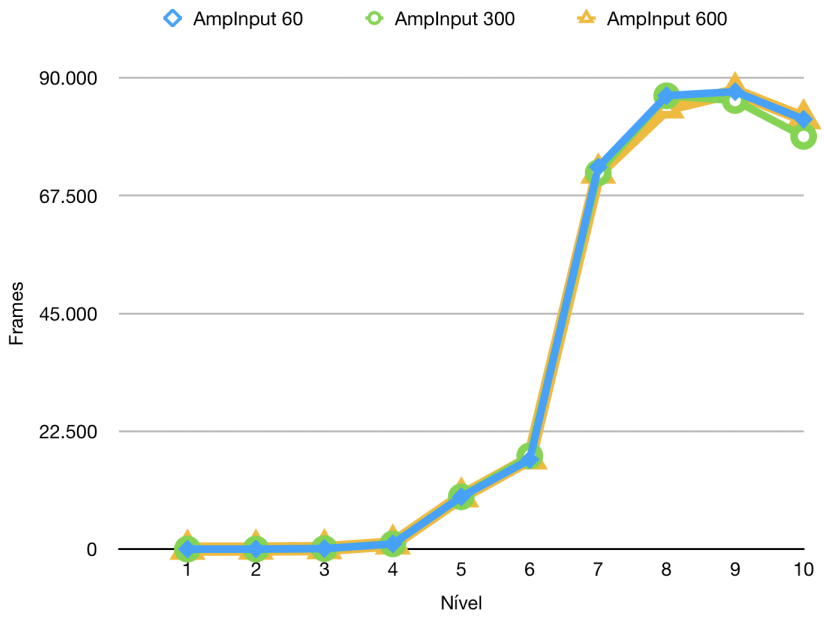

Figura 4.15: Quantidade de pacotes por nível que chegam no refletor para 60, 300 e 600 *Hosts*.

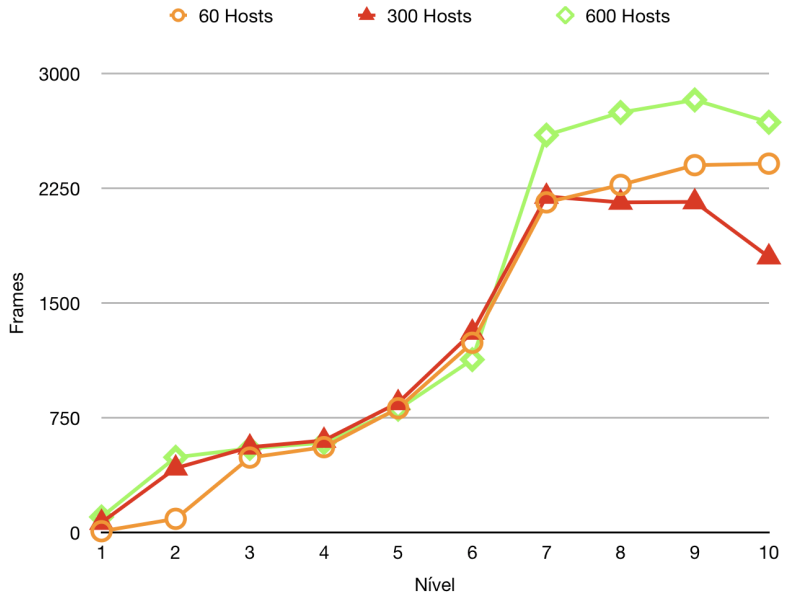

Figura 4.16: Quantidade de pacotes por nível que chegam na vítima para 60, 300 e 600 *Hosts*.

4.5 Considerações Finais

Com o teste de funcionalidade e performance realizado em laboratório, pode-se concluir que o ataque explorando o protocolo NTP pode ser utilizado, pois o mesmo apresenta

as características necessárias de amplificação e reflexão, podendo até mesmo em algumas configurações atingir altos níveis de amplificação. Contudo, pode se observar nos testes realizados que a amplificação não consegue ser mantida, em decorrência da sobrecarga e consequente saturação do refletor, processando as requisições *monlist*.

Capítulo 5

Conclusão e Trabalhos Futuros

5.1 Conclusão

Essa monografia apresentou a viabilidade da utilização do protocolo NTP para ataques distribuídos de negação de serviço por reflexão amplificadora, que é o protocolo utilizado para a sincronização de relógios de dispositivos de uma rede a partir de referências de tempo confiáveis.

Primeiramente foram apresentados vários tipos de ataque de negação de serviço, discutido o conceito de amplificação e reflexão, e proposto um embasamento teórico acerca do protocolo NTP. Nesse embasamento, foi explicitada a vulnerabilidade do protocolo com a mensagem de requisição *monlist*, a qual pode gerar uma resposta várias vezes maior que a requisição inicial, e assim, foi considerada a possibilidade de utilizar esse protocolo para ataques de negação de serviço por reflexão e amplificação.

Em seguida foi apresentada a ferramenta desenvolvida para que pudesse ser verificada a viabilidade do ataque. A ferramenta se destaca por ser extremamente flexível e versátil, onde somente temos a necessidade de criar um mirror para a preparação do ataque, e uma chamada do injetor, para a inicialização do ataque. Em decorrência dessa sua flexibilidade, podemos incluir nela qualquer tipo de ataque de negação de serviço; entretanto, a sua real ênfase é para ataques com características reflexivas e amplificadoras.

Com a ferramenta desenvolvida, em um ambiente controlado foi executado um teste de funcionalidade e performance, o qual buscava-se verificar as propriedades do ataque. Para o teste foram utilizadas três máquinas e um roteador, o qual foi utilizado para conectar as três máquinas entre si. Um computador agiu como atacante, executando a ferramenta Linderhof, outro computador foi configurado como um servidor NTP para refletir e amplificar os pacotes gerados pelo atacante, e um último computador foi utilizado como vítima do ataque. Os testes foram realizados em três configurações diferentes, onde podemos encontrar a resposta à requisição *monlist* com os dados referentes à 60, 300 e 600

hosts. Os ataques tiverem a duração de 50 segundos, onde cada nível durava 5 segundos, e consequentemente, foram utilizados os níveis de 1 a 10. Os dados referentes ao nível 1 ao 10 em sua respectiva configuração foram disponibilizados por meio de gráficos e tabelas.

Após processamento e análise dos dados coletados em decorrência dos testes, algumas particularidades puderam ser observadas. No nível inicial, para todas as configurações, obteve-se excelentes níveis de amplificação, os quais os valores eram esperados. Entretanto, do nível 2 em diante, foi presenciada uma queda acentuada nas taxas de amplificação. Essa queda na amplificação foi justificada com dois fatores, sendo o primeiro a sobrecarga do refletor com o processamento das requisições que haviam chegado, preparando as respectivas respostas, e consequentemente, dispensando alguns pacotes; e o segundo, foi a alta taxa de injeção do atacante, pois o Linderhof tenta injetar a maior quantidade de pacotes possível, em um curto espaço de tempo, auxiliando assim na sobrecarga do refletor.

O atacante conseguiu desempenhar um papel muito bom realizando a injeção dos pacotes até o nível 8, onde saturou, e manteve uma taxa constante de injeção. Entretanto, essa saturação do atacante em nada atrapalhou na análise do refletor, uma vez que o refletor deixou de fazer uma amplificação significativa dos pacotes a partir do nível 3, ou seja, bem antes da saturação do atacante.

Pelos resultados do teste, podemos concluir que o ataque de negação de serviço utilizando o NTP é viável, e pode entregar altas taxas de amplificação. Para que tenhamos um ataque bem sucedido, deve-se injetar uma quantidade menor de pacotes no refletor, aumentando o tempo de envio entre os pacotes, para que assim, o refletor possa os processar, e consequentemente, não saturar e manter as altas taxas de amplificação.

Com esse trabalho, conseguiu-se entender melhor o ataque de reflexão amplificada por NTP. Algumas atitudes podem ser tomadas para impedir que um servidor NTP seja utilizado em um ataque como refletor em um ataque, as quais segundo o alerta TA14-013A[44] do *Cybersecurity and Infrastructure Security Agency* (CISA), são:

- Atualizar a versão do NTP para no mínimo a 4.2.7p26.
- Desabilitar a requisição *monlist* no arquivo de configuração ntp.conf, adicionando a linha “disable monitor”.
- No arquivo de configuração ntp.conf, adicionar a diretiva “noquery” à linha “restrict default” como na Figura 5.1

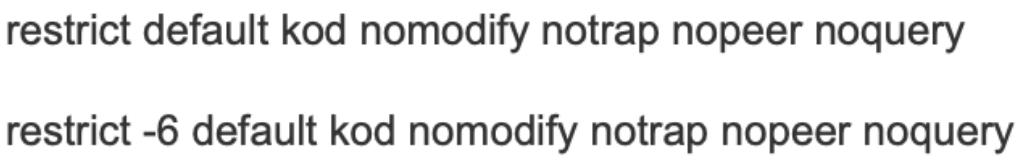

Figura 5.1: Comando para desabilitar a requisição "*get monlist*".

5.2 Trabalhos Futuros

Devido ao vasto campo de estudo que há nos ataques de negação de serviço, alguns trabalhos futuros são propostos:

- **Otimização do Linderhof:** Por decorrência do comportamento presenciado no refletor NTP, poderiam ser feitas algumas modificações no programa, para que se possa ter um controle sobre a injeção de pacotes, ou seja, poder realizar um escalonamento da injeção. Outro ponto importante para ser realizado no futuro é a implementação da possibilidade da utilização de mais de um refletor no ataque, para que o mesmo tenha as devidas características de um ataque DDoS.
- **Extensão do Linderhof:** Hoje a ferramenta desenvolvida nos permite realizar ataques com o NTP, o Memcached e o SSDP. Entretanto, a mesma pode utilizar outros vetores de ataque como o SNMP, DNS, entre outros.
- **Execução de testes com dispositivos diferentes:** Ao realizarmos testes com outros dispositivos, poderíamos verificar como a configuração de hardware dos dispositivos influenciam no resultado do ataque, especialmente para refletores diferentes.

Referências

[1] Ottis, R: *Analysis of the 2007 cyber attacks against estonia from the information warfare perspective*. 7th European Conference on Information Warfare and Security 2008, ECIW 2008, páginas 163–168, janeiro 2008. 2

[2] Francis, Ryan: *Hire a ddos service to take down your enemies*. Online, March 2017. `https://www.csoonline.com/article/3180246/hire-a-ddos-service-to-take-down-your-enemies.html`. 2

[3] Prince, Matthew: *Technical details behind a 400gbps ntp amplification ddos attack*. Online, February 2014. `https://blog.cloudflare.com/technical-details-behind-a-400gbps-ntp-amplification-ddos-attack/`. 2

[4] Alerts, Akamai SIRT: *Memcached-fueled 1.3 tbps attacks*. Online, March 2018. `https://blogs.akamai.com/2018/03/memcached-fueled-13-tbps-attacks.html`. 2

[5] Woolf, Nicky: *Ddos attack that disrupted internet was largest of its kind in history, experts say*. Online, October 2016. `https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet`. 2

[6] Abatishchev: *Loic*, 2014. `https://sourceforge.net/projects/loic/`, acesso em 2019/02/03. 4

[7] Orbitalsatelite: *r-u-dead-yet*, 2016. `https://sourceforge.net/projects/r-u-dead-yet/`, acesso em 2019/02/3. 4

[8] Bawany, Narmeen, Jawwad Shamsi e Khaled Salah: *Ddos attack detection and mitigation using sdn: Methods, practices, and solutions*. Arabian Journal for Science and Engineering, 42(2):425–441, 2017, ISSN 2193-567X. 4

[9] Aamir, M. e Mustafa Ali Zaidi: *Ddos attack and defense: Review of some traditional and current techniques*. CoRR, abs/1401.6317, 2014. 5, 7

[10] Correa, Andre: *Ddos reflection and amplification attacks*. Online. `https://www.malwarepatrol.net/ddos-reflection-and-amplification-attacks/`. 6

[11] Kenney, Malachi: *Ping of death*, October 1996. `https://insecure.org/sploits/ping-o-death.html`. 7

[12] Rana, Deepak, Naveen Garg e Sushil Chamoli: *A study and detection of tcp syn flood attacks with ip spoofing and its mitigations*. International Journal of Computer Technology and Applications, 03, julho 2012. 7

[13] Mitrokotsa, A. e C. Douligeris: *Network Security: Current Status and Future Directions*, capítulo Denial of Service Attacks, páginas 117–134. John Wiley and Sons. John Wiley and Sons, June 2007. 8, 12

[14] *Udp flood attack*. https://www.cloudflare.com/learning/ddos/udp-flood-ddos-attack/. ix, 9

[15] *Ping (icmp) flood ddos attack*. https://www.cloudflare.com/learning/ddos/ping-icmp-flood-ddos-attack/. ix, 9

[16] Rossow, Christian: *Amplification hell: Revisiting network protocols for ddos abuse*. February 2014. 10, 19, 45

[17] *Memcached ddos attacks*, February 2018. https://security.radware.com/ddos-threats-attacks/threat-advisories-attack-reports/memcached-under-attack/. ix, 12, 13

[18] *Public memcached servers*. Online, March 2018. https://www.shodan.io/report/poJtt1i9. 12

[19] Arukonda, Srinivas e S. Sinha: *The innocent perpetrators : Reflectors and reflection attacks*. 2015. 13

[20] *Ssdp amplification*. https://www.corero.com/resources/ddos-attack-types/ssdp-amplication-ddos.html. ix, 14

[21] *Dns server types*. Online. https://www.cloudflare.com/learning/dns/dns-server-types/. 15

[22] Liu, Cricket: *Distributed-denial-of-service attacks and dns*. Online, November 2017. https://www.forbes.com/sites/forbestechcouncil/2017/11/15/distributed-denial-of-service-attacks-and-dns/#1c7139276076. 15

[23] Medeiros, Tiago Fonseca: *Ataque distribuído de negação de serviço por reflexão amplificada usando simple network management protocol*. 2015. http://bdm.unb.br/handle/10483/11152. ix, 15

[24] Mills, David L.: *Network time protocol (ntp)*. Rfc 958, RFC Editor, September 1985. https://tools.ietf.org/html/rfc958. 15, 16

[25] L Mills, David: *A brief history of ntp time: confessions of an internet timekeeper*. Computer Communication Review - CCR, janeiro 2003. https://www.eecis.udel.edu/~mills/database/papers/history.pdf. 16

[26] Mills, David L.: *Time synchronization in dcnet hosts*. Ien 173, COMSAT Laboratories, https://www.eecis.udel.edu/ mills/database/rfc/ien-173.txt, February 1981. 16

[27] Mills, David L.: *Dcnet internet clock service*. Rfc 778, RFC Editor, April 1981. https://tools.ietf.org/html/rfc778. 16

[28] Mills, David L.: *Network time protocol (version 1) - specification and implementation*. Rfc 1059, RFC Editor, July 1988. https://tools.ietf.org/html/rfc1059. 16

[29] Mills, David L.: *Network time protocol (version 2) - specification and implementation*. Rfc 1119, RFC Editor, 1989. https://tools.ietf.org/html/rfc1119. 16

[30] Mills, David L.: *Network time protocol (version 3) - specification, implementation and analysis*. Rfc 1305, RFC Editor, March 1992. https://tools.ietf.org/html/rfc1305. 16

[31] Mills, David L.: *Simple network time protocol (sntp) version 4 for ipv4, ipv6 and osi*. Rfc 2030, RFC Editor, October 1996. https://tools.ietf.org/html/rfc2030. 16

[32] Mills, D. L., U. Delaware e Ed. J. Martin: *Network time protocol version 4: Protocol and algorithms specification*. Rfc 5905, RFC Editor, June 2010. https://tools.ietf.org/html/rfc5905. 16

[33] Toponce, Aaron: *Real life ntp*. Online, November 2013. https://pthree.org/2013/11/05/real-life-ntp/. ix, 17

[34] *How ntp works*. Online, March 2014. https://www.eecis.udel.edu/~mills/ntp/html/warp.html. 17

[35] *O ntp*. Online. https://ntp.br/ntp.php. 17

[36] Graham-Cumming, John: *Understanding and mitigating ntp-based ddos attacks*. Online, January 2014. https://blog.cloudflare.com/understanding-and-mitigating-ntp-based-ddos-attacks/. 18

[37] jah: *File ntp-monlist*. Online. https://nmap.org/nsedoc/scripts/ntp-monlist.html. ix, 18, 19

[38] NMAP.ORG: *Downloading nmap*. Online. https://nmap.org/download.html. 19

[39] Wireshark: *Download wireshark*. https://www.wireshark.org/download.html. 19

[40] *Openntpproject.org - ntp scanning project*. Online. http://openntpproject.org. 19

[41] *The zmap project*. Online. https://zmap.io. 19

[42] Rudman, Lauren e Barry Irwin: *Characterization and analysis of ntp amplification based ddos attacks*. 2015 Information Security for South Africa (ISSA), páginas 1–5, 2015. ix, 20

[43] Team, United States Computer Emergency Readiness: *Alert (ta14-017a) - udp-based amplification attacks*. Alert, March 2018. https://www.us-cert.gov/ncas/alerts/TA14-017A. ix, 20, 21

[44] Team, United States Computer Emergency Readiness: *Alert (ta14-013a) - ntp amplification attacks using cve-2013-5211*. Online, October 2016. https://www.us-cert.gov/ncas/alerts/TA14-013A. 20, 53

[45] *Build a trigger to monitor responses to ntp monlist requests*. Online, February 2019. https://docs.extrahop.com/7.4/walkthrough-upa/. 26

[46] *Proj 6x: Packet amplification with ntp*. Online, October 2014. https://samsclass.info/124/proj14/p6x-NTP-DrDOS.htm. 32