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

Ataque Distribuído de Negação de Serviço por Reflexão Amplificada usando Simple Network Management Protocol

Tiago Fonseca Medeiros

Brasília
2015

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

Ataque Distribuído de Negação de Serviço por Reflexão Amplificada usando Simple Network Management Protocol

Tiago Fonseca Medeiros

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Engenharia da Computação

Orientador
Prof. Ms. João José Costa Gondim

Brasília
2015

Universidade de Brasília - UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharelado em Engenharia da Computação

Coordenador: Prof. Dr. Ricardo Zelenovsky

Banca examinadora composta por:

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

Prof. Dra. Aletéia Patrícia Favacho de Araújo - CIC/UnB

Prof. Dr. Robson de Oliveira Albuquerque - ENE/UnB

CIP - Catalogação Internacional na Publicação

Medeiros, Tiago Fonseca

Ataque Distribuído de Negação de Serviço por Reflexão Amplificada usando Simple Network Management Protocol / Tiago Fonseca Medeiros. Brasília : UnB, 2015.

Monografia (Graduação) - Universidade de Brasília, Brasília, 2015

1. Negação de Serviço
2. Negação de Serviço Distribuída
3. Amplificação
4. Reflexão
5. Simple Network Management Protocol
6. Segurança
7. Redes
8. IP Spoofing
9. SNMP4J
10. GetBulkRequest

CDU 004.4

Endereço: Universidade de Brasília

Campus Universitário Darcy Ribeiro - Asa Norte

CEP 70910-900

Brasília-DF - Brasil

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

Ataque Distribuído de Negação de Serviço por Reflexão Amplificada usando Simple Network Management Protocol

Tiago Fonseca Medeiros

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Engenharia da Computação

Prof. Ms. João José Costa Gondim (Orientador)
CIC/UnB

Prof. Dra. Aletéia Patrícia Favacho de Araújo
CIC/UnB

Prof. Dr. Robson de Oliveira Albuquerque
ENE/UnB

Prof. Dr. Ricardo Zelenovsky
Coordenador do Bacharelado em Engenharia da Computação

Brasília, 9 de Julho de 2015

AGRADECIMENTOS

Agradeço aos meus pais que me apoiaram, incentivaram e ensinaram durante todos os anos da minha vida. Obrigado por estarem sempre ao meu lado e me fornecerem todas as ferramentas necessárias para que eu possa trilhar o meu próprio caminho. Agradeço aos meus irmãos, à Socorro e a outros familiares que também estiveram sempre presentes.

Agradeço aos meus amigos que compartilharam comigo todos esses anos e que, com certeza, vão continuar fazendo parte da minha vida. Obrigado pelas milhares de horas de companhia e por tudo que vencemos, e vamos vencer ainda, juntos.

Agradeço à Raissa pela ajuda e compreensão em todos os momentos difíceis. Obrigado por tudo.

Agradeço a todos os professores que participaram da minha formação. Muito obrigado por todo o conhecimento e pelas valiosas lições. Agradeço, em especial, ao meu orientador, João Gondim, por todas as suas orientações e por ter me possibilitado realizar um trabalho com um tema de que gosto tanto.

Agradeço à Universidade de Brasília por me proporcionar essa experiência única que é a graduação.

RESUMO

Ataques de negação de serviço existem na internet a muitos anos. Sua principal finalidade é exaurir os recursos de uma vítima impedindo que ela disponibilize os seus serviços ou utilize outros. Recentemente, negação de serviço pode ser utilizada como uma forma de distração ou cobertura para outros ataques. Devido aos altos prejuízos que empresas sofrem com os ataques, principalmente as que possuem a disponibilidade como um pilar de seus negócios, o seu estudo se mostra cada vez mais importante.

Este trabalho apresenta um estudo do ataque distribuído de negação de serviço por amplificação e reflexão usando o *Simple Network Management Protocol* (SNMP), um protocolo para gerenciamento de redes muito utilizado. A versão do SNMP estudada foi a SNMPv2c, por combinar a fraca segurança, utilizada na versão 1, com o comando GetBulkRequest, adicionado a partir da versão 2 e que é capaz de proporcionar uma amplificação significativa.

O estudo foi realizado com o desenvolvimento de uma ferramenta capaz de detectar possíveis refletores e de realizar o ataque em diferentes intensidades. A ferramenta foi desenvolvida nas linguagens de programação Java e C, com uma arquitetura simples e extensível. Sua utilização se dá por uma interface gráfica e comandos simples como a maioria das ferramentas para ataque de negação de serviço encontradas na internet.

Foram realizados dois tipos de teste para estudar as características do ataque. O primeiro foi um teste de funcionalidade que analisa as propriedades de reflexão e amplificação. O segundo teste, de performance, verificou a capacidade da ferramenta desenvolvida além da saturação dos dispositivos envolvidos.

Palavras-chave: negação de serviço, negação de serviço distribuída, amplificação, reflexão, Simple Network Management Protocol, segurança, redes, IP spoofing, SNMP4J, getbulkrequest.

ABSTRACT

Denial-of-service attacks exist on the internet for many years. Its primary goal is to exhaust a victim resources preventing it from making their own services available or use others. Recently, denial-of-service may be used as a smokescreen for others attacks. Studies about denial-of-service are important due to the high losses companies suffer, especially those with availability as a pillar of its business.

This paper presents a study about the Simple Network Management Protocol (SNMP) reflected amplification distributed denial-of-service. SNMP is a protocol for managing networks and is widely used. The SNMP version studied was SNMPv2c. This version combines the weak security, used in version 1, with the GetBulkRequest command, added in version 2, that is able to provide significant amplification.

To better understand the properties of the attack a software was developed. The software was able to detect possible reflectors and to perform the attack with different intensities. Java and C programming languages were used to develop the program. A simple and extensible architecture was used. The program works with an user graphic interface and simple commands like most denial-of-service programs found on the internet.

Two types of tests were performed to understand the attack properties. The first one was a functionality test and examined the reflective and amplification property. The second, a performance test, verified the capacity of the developed software and the saturation point of all devices involved. Finally, tests results were analyzed and explained.

Keywords: denial-of-service, distributed denial-of-service, amplification, reflected, Simple Network Management Protocol, security, network, IP spoofing, SNMP4J, getbulkrequest.

SUMÁRIO

Sumário i

Lista de Figuras iv

Lista de Tabelas vi

Glossário vii

Capítulo 1 – Introdução 1

1.1 Motivação 1
1.2 Justificativa 2
1.3 Objetivos 2
1.4 Organização do texto 3

Capítulo 2 – Quadro Conceitual 4

2.1 Negação de serviço 4
2.2 Negação de serviço distribuída 4
2.2.1 Abuso de protocolo 6
2.2.1.1 Exploração de protocolo 6
2.2.1.2 Exploração de vulnerabilidade 7
2.2.2 Volumétrico 8
2.2.2.1 Ataques de inundação 8
2.2.2.2 Ataques de reflexão e amplificação 8
2.3 Ataque usando o *Simple Network Management Protocol* 11
2.3.1 Histórico do SNMP 11
2.3.2 Tipos de dispositivos 12
2.3.3 *Management Information Base* 13
2.3.4 Modos de operação 15
2.4 Considerações finais 17

Capítulo 3 – Ferramenta - SNMP Striker 19

3.1 *Design* . . . 19
3.1.1 Gui - *Graphical user interface* . . . 20
3.1.1.1 Procurando dispositivos . . . 20
3.1.1.2 Importar e exportar dispositivos . . . 21
3.1.1.3 Selecionar dispositivos . . . 21
3.1.2 Scanner . . . 21
3.1.2.1 Reconhecimento ou detecção . . . 21
3.1.2.2 Cálculo da amplificação . . . 21
3.1.3 Striker . . . 23
3.1.3.1 *Ip spoofing* . . . 23
3.1.3.2 Taxa de envio . . . 23
3.2 Implementação . . . 23
3.2.1 Gui . . . 24
3.2.1.1 Adicionar e remover . . . 24
3.2.1.2 Tabela de dispositivos vulneráveis . . . 26
3.2.1.3 Importar e exportar . . . 27
3.2.1.4 Realizar ataque . . . 27
3.2.2 Scanner . . . 28
3.2.2.1 SNMP4J . . . 29
3.2.3 Striker . . . 31
3.2.3.1 JNI . . . 32
3.2.3.2 Rate Limiter . . . 33
3.3 Considerações finais . . . 34

Capítulo 4 – Testes e Resultados 35

4.1 Configuração de testes . . . 35
4.2 Teste de funcionalidade . . . 36
4.2.1 Descoberta de dispositivos . . . 36
4.2.1.1 Captura - Computador de ataque . . . 37
4.2.1.2 Captura - Computador refletor . . . 38
4.2.1.3 Captura - Computador vítima . . . 39
4.2.2 Ataque . . . 39
4.2.2.1 Captura - Computador de ataque . . . 40
4.2.2.2 Captura - Computador refletor . . . 40
4.2.2.3 Captura - Vítima . . . 41
4.2.3 Análise do teste de funcionalidade . . . 41
4.2.3.1 Procura de dispositivos vulneráveis . . . 41

4.2.3.2 Princípio da reflexão 41
4.2.3.3 Princípio da amplificação 42
4.3 Teste de performance 43
4.3.1 Computador de ataque 44
4.3.2 Computador refletor 45
4.3.3 Computador vítima 46
4.3.4 Análise do teste de performance 47
4.3.4.1 Análise da performance do computador de ataque 47
4.3.4.2 Análise da performance do *switch* 48
4.3.4.3 Análise da performance do refletor 51
4.3.4.4 Análise dos dados na vítima 53
4.4 Considerações finais 53

Capítulo 5 – Conclusão 55

5.1 Trabalhos Futuros 57

Referências Bibliográficas 59

LISTA DE FIGURAS

2.1 Arquitetura ataques de negação de serviço distribuídos. 5

2.2 *Three way handshake* na negação de serviço TCP SYN. 7

2.3 Esquema simplificado do ataque *Smurfing*. 9

2.4 Ataque de negação de serviço por amplificação e reflexão utilizando DNS. 10

2.5 Arquitetura SNMP simplificada. 13

2.6 Árvore MIB simplificada. 15

3.1 Diagrama de pacotes. 20

3.2 SNMP Gui. 24

3.3 SNMP Scanner. 25

3.4 Mensagem - Nenhum dispositivo encontrado. 26

3.5 Mensagem - Um dispositivo encontrado. 26

3.6 Tabela com um dispositivo vulnerável. 26

3.7 Json com dispositivo vulnerável. 27

3.8 Diagrama de classes do módulo Scanner. 28

3.9 Diagrama de sequência na descoberta de novos dispositivos. 30

3.10 Diagrama de sequência no cálculo de amplificação. 31

3.11 Diagrama de classes do módulo Striker. 32

4.1 Disposição de computadores para o teste de funcionalidade. 36

4.2 Tela para adicionar novos dispositivos vulneráveis através de varredura na rede. 37

4.3 Captura no computador de ataque da primeira parte da varredura. 38

4.4 Captura no computador de ataque da segunda parte da varredura para cálculo de amplificação. 38

4.5 Captura de varredura no computador refletor. 38

4.6 Captura de varredura no computador vítima. 39

4.7 Tela principal antes do início do ataque. 39

4.8 Captura nível 1 no computador de ataque. 40

4.9 Mensagens trocadas entre os dispositivos durante o ataque. 40

4.10 Captura nível 1 no computador refletor. 41

4.11 Captura nível 1 na vítima. 41

4.12 GetBulkRequest no refletor. 42

4.13 GetResponse no refletor. 42

4.14 Bits e pacotes por segundo enviados pelo atacante. 48

4.15 Quantidade de bits enviados pelo computador de ataque e recebidos pelo refletor durante o teste 1. 49

4.16 Quantidade de bits enviados pelo computador de ataque e recebidos pelo refletor durante o teste 2. 49

4.17 Quantidade de bits enviados pelo refletor e recebidos pela vítima durante o teste 1. 50

4.18 Quantidade de bits enviados pelo refletor e recebidos pela vítima durante o teste 2. 51

4.19 Quantidade de bits recebidos e enviados pelo refletor durante os testes 1 e 2. . . 52

4.20 Valor da amplificação para bits e pacotes. 54

LISTA DE TABELAS

4.1 Tabela com dados da captura no computador de ataque durante 30 segundos no teste 1. 44

4.2 Tabela com dados da captura no computador de ataque durante 30 segundos no teste 2. 45

4.3 Tabela com dados da captura no computador refletor durante 30 segundos no teste 1. 45

4.4 Tabela com dados da captura no computador refletor durante 30 segundos no teste 2. 46

4.5 Tabela com dados da captura na vítima durante 30 segundos no teste 1. 46

4.6 Tabela com dados da captura na vítima durante 30 segundos no teste 2. 47

4.7 Tabela com a amplificação do refletor em bits e pacotes durante os teste 1 e 2. . 52

GLOSSÁRIO

DOS	Denial of Service
DDOS	Distributed Denial of Service
SNMP	Simple Network Management Protocol
DNS	Domain Name System
NTP	Network Time Protocol
ICMP	Internet Control Message Protocol
SSDP	Simple Service Discovery Protocol
IP	Internet Protocol
UDP	User Datagram Protocol
TCP	Transmission Control Protocol

CAPÍTULO 1

INTRODUÇÃO

É notável o crescimento da importância da internet, para a sociedade, durante os últimos anos. Diariamente pessoas têm acesso à diversos serviços, como redes sociais, bancos e compras, que estão disponíveis na internet. Entretanto, a disponibilidade desses serviços pode ser ameaçada por vários tipos de ataques como, por exemplo, a negação de serviço.

1.1 Motivação

Em computação um ataque de negação de serviço (DoS - *Denial of Service*) pretende esgotar os recursos de um equipamento ou infraestrutura que fornece ou utiliza um serviço. Esse equipamento pode ser um servidor, um *switch*, um roteador, um usuário e muitos outros. Os ataques consistem no envio de um número muito grande de requisições que esgotam o limite de tráfego ou não conseguem ser processadas, esgotando algum recurso vital da vítima como memória ou banda. Dessa forma, a vítima não consegue acessar serviços disponíveis ou disponibilizar os seus [1].

Ataques de negação de serviço distribuídos (DDoS - *Distributed Denial of Service*) utilizam os mesmos princípios da negação de serviço, mas com um número de máquinas comprometidas para aumentar consideravelmente sua eficiência. Essas máquinas, além de potencializar o efeito do ataque, dificultam o isolamento da fonte de tráfego, dificultando o seu bloqueio pela vítima. Outra estratégia de ataque distribuído é o uso de refletores. Refletor é qualquer dispositivo que ao receber um datagrama IP retorna outro como resposta. Ao contrário de máquinas comprometidas, que geram tráfego sozinhas, refletores são úteis, na visão da negação de serviço, quando possuem um fator de amplificação. Este fator indica quanto de tráfego o refletor gera em relação ao que recebe. Com a amplificação, o tráfego gerado pelo atacante será potencializado no refletor [2].

Dois protocolos muito utilizados para negação de serviço por amplificação e reflexão são:

Domain Name System (DNS) e *Netwrok Time Protocol* (NTP). Ambos possibilitam que pequenas mensagens gerem respostas muitas vezes maiores. Um terceiro protocolo com essa característica é o *Simple Network Management Protocol* (SNMP). O SNMP roda em diversos dispositivos de rede e permite obter status desses dispositivos além de alterar configurações. Essa monografia visa estudar a viabilidade do SNMP como um protocolo para a negação de serviço distribuída a fim de entender melhor esse tipo de ataque.

Este trabalho está sendo desenvolvido com fins acadêmicos para efeito de demonstração das vulnerabilidades estudadas. O autor e o seu orientador não se responsabilizam por qualquer uso inapropriado, inadequado ou ilegal que venha a extrapolar tais objetivos. O código desenvolvido está sob a custódia do autor e orientador para a continuidade de estudos futuros.

1.2 Justificativa

Requisitos de disponibilidade são muito importantes para vários serviços disponíveis na internet. Bancos, empresas de cartão de crédito e instituições financeiras em geral, por exemplo, possuem modelos de negócios fundamentados no imediato acesso a seus serviço e sistemas. A redução da qualidade ou a interrupção desses serviços podem trazer danos financeiros significativos, além de prejudicar a imagem da empresa. É nesse sentido que o estudo e a análise dos ataques de negação de serviço é importante. A partir de um entendimento mais completo pode-se desenvolver técnicas de mitigação e controle mais eficientes.

O protocolo SNMP é um protocolo muito utilizado para o gerenciamento de dispositivos em uma rede. Por esse motivo pode existir um número muito alto de refletores para o ataque de negação de serviço. A identificação das principais características do SNMP que possibilitam o ataque é importante, pois servem como base para soluções que visam impedir o uso do SNMP como vetor de negação de serviço.

1.3 Objetivos

O objetivo da monografia é desenvolver uma ferramenta para detectar possíveis refletores e avaliar quantitativamente os efeitos da negação de serviço com características amplificadoras e refletoras, utilizando o protocolo de gerenciamento *Simple Network Management Protocol*.

A ferramenta desenvolvida deve possuir funcionalidades para calcular a amplificação máxima possível em cada refletor. Além de encontrar refletores, deve realizar o ataque, com eficiência, em níveis graduais de intensidade.

Por fim, uma série de testes deve ser elaborado para uso da ferramenta em ambientes controlados, avaliando assim as propriedades do ataque por SNMP.

1.4 Organização do texto

Este trabalho está organizado com a seguinte estrutura:

- No Capítulo 2 aborda-se todo o fundamental teórico necessário para o entendimento do trabalho;
- No Capítulo 3 apresentam-se o *design* e a implementação da ferramenta desenvolvida;
- No Capítulo 4 são discutidos os resultados e a análise de todos os testes realizados durante o trabalho;
- Por fim, as conclusões e as propostas para trabalhos futuros são abordadas no Capítulo 5.

CAPÍTULO 2

QUADRO CONCEITUAL

Apresentam-se nesse capítulo os conceitos e fundamentos dos ataques de negação de serviço desde sua forma mais simples e o protocolo *Simple Network Management Protocol*. No final do capítulo é feita uma análise sobre os motivos que levam esse protocolo a ser um candidato para ataques de negação de serviço por reflexão e amplificação.

2.1 Negação de serviço

Negação de serviço (DOS - *Denial of Service*) é um tipo de ataque onde um atacante visa exaurir os recursos de uma vítima a fim de impedir que ela disponibilize ou use um serviço. Essa vítima pode ser um servidor, um usuário, um provedor de serviços (ISP), uma rede, uma empresa e diversos outros. Ressalta-se que um ataque de negação só é caracterizado quando um serviço, que deveria estar disponível, fica fora do ar por causa de ações intencionais e maliciosas. Caso o serviço caia devido a ações não intencionais não podemos dizer que ele sofreu um ataque apesar da negação de serviço.

Ataques de negação de serviço surgiram como um jogo entre atacantes de comunidades ocultas na internet. Um atacante, por exemplo, ganhava reconhecimento ao derrubar sites populares. Ferramentas de negação de serviço como o *High Orbit Ion Cannon* [3] podem ser facilmente adquiridas na internet e junto com sua simplicidade de uso tornam qualquer usuário um potencial atacante [4].

2.2 Negação de serviço distribuída

Ataques de negação de serviço distribuídos utilizam os mesmo princípios da negação de serviço não distribuída. A grande diferença é que o ataque é realizado de forma coordenada por diversos dispositivos que foram comprometidos. Com a utilização de dispositivos comprome-

tidos um atacante pode realizar uma negação de serviço muito mais eficaz além de se manter anônimo pois quem realiza o ataque são esses dispositivos vulneráveis. A Figura 2.1 demonstra a arquitetura básica dos ataques distribuídos.

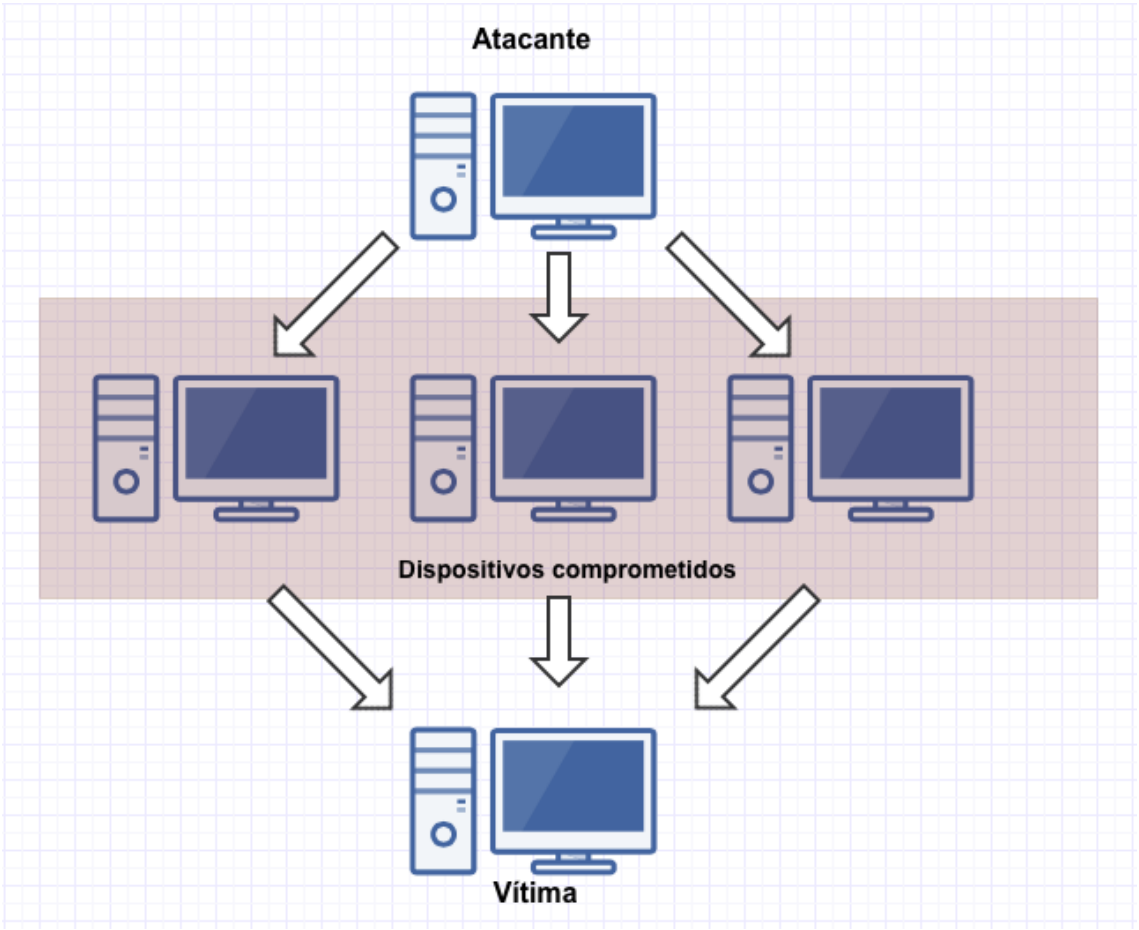

Figura 2.1. Arquitetura ataques de negação de serviço distribuídos.

Existe uma variedade muito grande de ataques de negação de serviço distribuídos. Entretanto, é possível organizá-los em duas grandes categorias: volumétricos e abuso de protocolo. A primeira categoria diz respeito aos ataques que enviam uma quantidade muito grande de tráfego para a vítima, exaurindo algum recurso, como banda, ou impedindo que o tráfego legítimo passe. A segunda compreende os ataques que, com a ajuda de características específicas de protocolos e implementações, esgotam os recursos da vítima de modo que ela não consegue mais disponibilizar seus serviços [5].

Uma característica que alguns ataques de negação compartilham é o *IP spoofing*. Normalmente, pacotes IP são utilizados por aplicações que usam a internet para se comunicar. Esses pacotes são gerados automaticamente pelo sistema operacional e possuem os valores corretos de

origem e destino em seu cabeçalho para que a troca de informações entre as aplicações funcione. Entretanto, o usuário pode criar seu próprio cabeçalho do pacote IP e informar ao sistema operacional. Desse modo, um usuário que queira permanecer anônimo na internet pode alterar o IP de origem de seu pacote. Assim, mesmo equipamentos que mantêm *log* das conexões não serão capazes de dizer de onde o pacote realmente veio. Além disso, *IP spoofing* é uma característica muito importante para ataques de amplificação e reflexão como será visto a seguir [6].

2.2.1 Abuso de protocolo

Em ataques de negação por abuso de protocolo, um atacante se utiliza maliciosamente de alguma característica do protocolo de comunicação usado ou de sua implementação. Dessa forma, é possível classificá-los em exploração de protocolo e exploração de vulnerabilidade.

2.2.1.1 Exploração de protocolo

Na exploração de protocolo, um atacante se utiliza de protocolos que geram uma alocação de recursos na vítima. Por exemplo, tem-se a exploração do *three-way handshake* do protocolo TCP no ataque *SYN flooding*. Na negação por *SYN flooding* computadores controlados pelo atacante enviam muitos pedidos de conexão TCP com o IP de origem alterado, na forma de segmentos SYN, para um servidor. Em seguida, o servidor responde para cada mensagem recebida com um segmento SYN ACK e espera pelo segmento ACK. Como o IP de origem no pacote SYN era falso o ACK de resposta nunca chega, pois o SYN ACK foi enviado para o IP alterado. Desse modo, com uma alta quantidade de segmentos SYN sendo enviados, os recursos do servidor se esgotam e usuários legítimos não conseguem usar seus serviços.

Embora o *SYN flooding* gere uma grande quantidade de tráfego, marcante dos ataques de negação do tipo volumétrico, ele foi classificado como um abuso de protocolo, pois sua principal caraterística é a exploração do *three-way handshake* do protocolo TCP. A figura 2.2 demonstra um esquema simplificado do ataque.

Uma ferramenta que consegue realizar esse ataque é o Hping na versão 3. Essa ferramenta permite enviar qualquer tipo de pacote TCP/IP e demonstra, mais uma vez, a facilidade com que um ataque de negação de serviço pode ser feito [7].

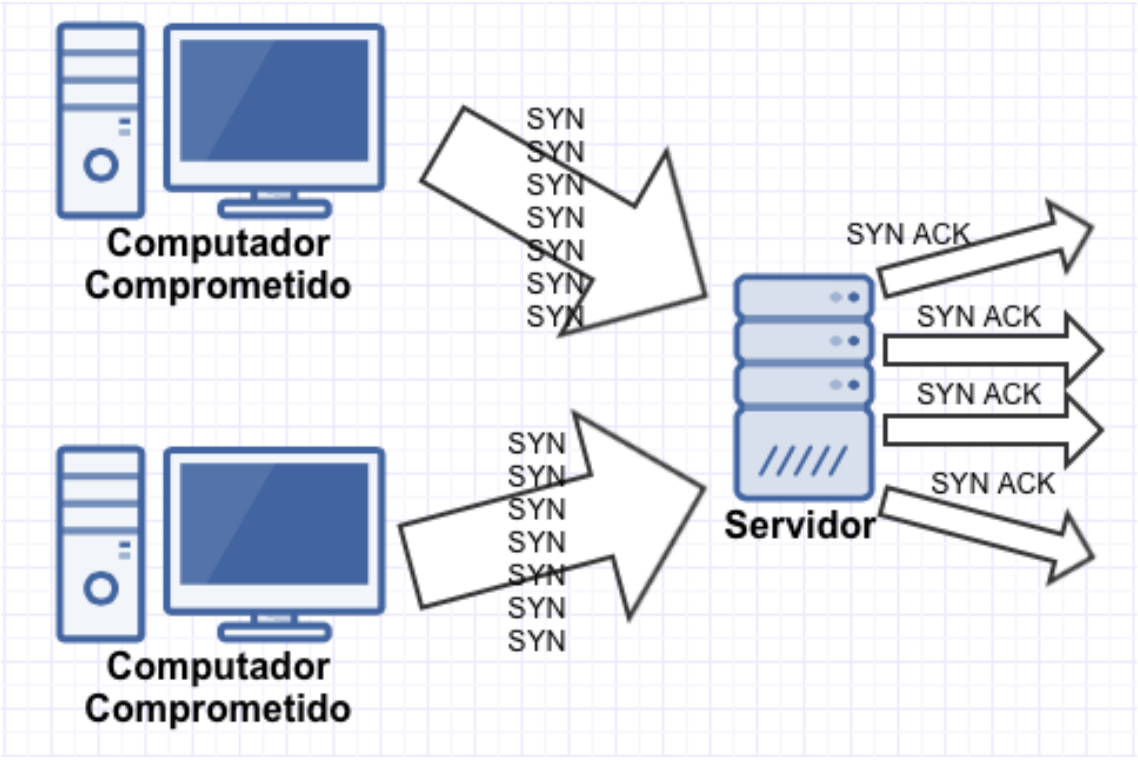

Figura 2.2. *Three way handshake* na negação de serviço TCP SYN.

2.2.1.2 Exploração de vulnerabilidade

Negação de serviço por exploração de vulnerabilidade pode ocorrer de duas maneiras. A primeira é encontrar alguma vulnerabilidade que provoca a situação do sistema ficar inoperante.Como exemplo pode-se citar a vulnerabilidade no Windows NT de 1997 onde, por causa da implementação do TCP, ao enviar um pacote TCP específico o sistema encerrava [8]. O *Ping of Death* é também um ataque desse tipo e consiste em enviar datagramas `ICMP_ECHO_REQUEST` ("*ping*") com tamanho maior que 65536 bytes que é o máximo permitido pelo protocolo IP [9].

A segunda maneira é explorar alguma característica que leve a vítima a gastar mais tempo de processamento ou mais memória. Essa condição pode ser alcançada ao se randomizar os campos opcionais do cabeçalho IP, o que leva a um gasto maior de processamento. Outra forma seria enviar diversos fragmentos de um pacote IP com números de identificação diferentes a fim de exaurir os recursos de memória, utilizada para armazenar cada fragmento até que todo o pacote chegue, ou processamento, cada fragmento deve ser processado para verificar sua posição no pacote.

2.2.2 Volumétrico

Ataques de negação de serviço volumétrico podem ser divididos em ataques de inundação (*flooding*) e ataques de reflexão e amplificação. Ambos os ataques visam sobrecarregar algum recurso da vítima, geralmente, a banda disponível, pelo envio de grandes quantidades de tráfego.

2.2.2.1 Ataques de inundação

Ataques de negação por inundação ocorrem quando computadores comprometidos enviam altas quantidades de pacotes para uma vítima a fim de esgotar sua banda. Uma técnica conhecida para esse ataque é através do envio de `ICMP_ECHO_REQUEST` ("*ping*"). Quando vários sistemas comprometidos enviam essa mensagem, a banda da vítima fica sobrecarregada com todas as mensagens que chegam, os *requests*, além das mensagens de resposta, os *replays*. Caso seja utilizado *IP spoofing* os sistemas atacantes podem maximizar o ataque, pois utilizam todo o seu recurso para enviar mensagens, e nenhuma banda será perdida com respostas da vítima.

Outro exemplo de ataque por inundação é o envio de mensagens UDP com portas aleatórias. Quando uma mensagem UDP chega em um servidor em uma determinada porta esse servidor processa a mensagem e determina qual aplicação está escutando nessa porta. Se não existir nenhuma aplicação, uma mensagem ICMP é retornada. Como no exemplo anterior, além da banda da vítima ser ocupada por todo o tráfego chegando ela também é tomada pelas respostas ICMP [10].

2.2.2.2 Ataques de reflexão e amplificação

Os ataques de reflexão e amplificação se utilizam de dois princípios. O primeiro é o de refletor que é qualquer dispositivo IP que ao enviar uma mensagem responde com outra. A ideia básica do ataque é enviar mensagens com *IP spoofing* para o refletor, e ele responder para a vítima. O ataque não tem muito sentido sem o segundo princípio que é a amplificação. Com a amplificação, uma mensagem enviada para um refletor gera uma mensagem muitas vezes maior. A razão entre a mensagem gerada e a mensagem que a originou é o fator de amplificação.

Esse tipo de ataque começou com o *Smurfing*, discutido a seguir. Ao longo dos anos outras técnicas de amplificação foram utilizadas explorando os protocolos *Domain Name System*

(DNS) e *Network Time Protocol* (NTP), por exemplo. Outro protocolo que também pode ser utilizado é o SNMP, foco deste trabalho, e que será discutido adiante.

Amplificação usando o *Internet Control Message Protocol - Smurfing*

O *Smurfing* se utiliza do endereço de *broadcast* para enviar `ICMP_ECHO_REQUEST` ("ping") com *IP spoofing*. Desse modo, ao atingir um roteador a mensagem é distribuída a todos os dispositivos conectados. Quando um dispositivo recebe a mensagem ele gera uma resposta e envia para o endereço de origem, que foi modificado. Um atacante consegue então amplificar e refletir o seu ataque. A Figura 2.3 demonstra o esquema simplificado do *Smurfing* [10].

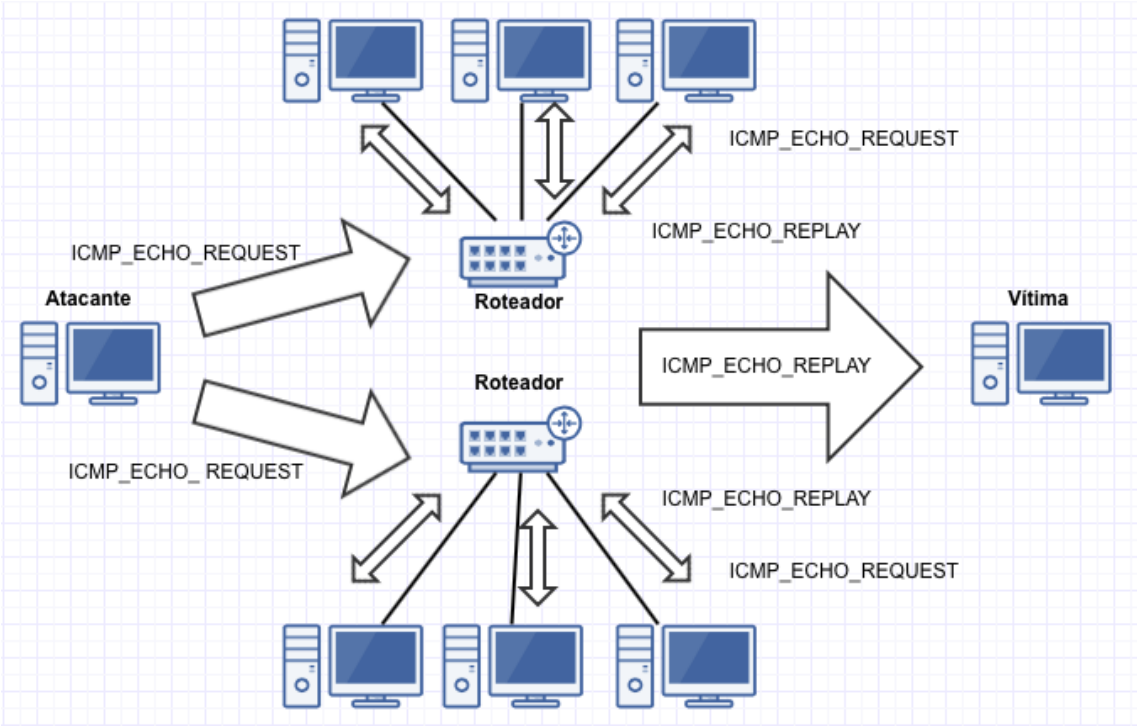

Figura 2.3. Esquema simplificado do ataque *Smurfing*.

Amplificação usando o *Domain Name System*

O *Domain Name System* (DNS) é um sistema de busca de dados distribuído e hierárquico que traduz os nomes de domínio, que são mais facilmente lembrados por humanos, em endereços IP. O sistema é composto por diversos servidores DNS que prestam serviços a clientes. Um modo de operação de um servidor utilizado no DNS é o recursivo. Esse tipo de servidor é responsável por receber consultas de clientes locais e consultar servidores externos para obter

a resposta. Quando um servidor recursivo está configurado para responder qualquer cliente e não somente os locais se diz que esse servidor é aberto [11] [12].

Para realizar o ataque, envia-se uma mensagem para um servidor DNS recursivo aberto com o IP de origem alterado, *IP spoofing*. A mensagem, com uma consulta tipo "*ANY*", pede que o servidor responda com todos os seus registros sobre uma determinada zona. A resposta gerada pelo servidor será muitas vezes maior, em bytes, que a consulta, e assim obtêm-se a amplificação. Para potencializar o ataque, atacantes podem utilizar mais de um servidor DNS e mais de um computador comprometido para enviar as consultas [13]. A figura 2.4 mostra a arquitetura do ataque.

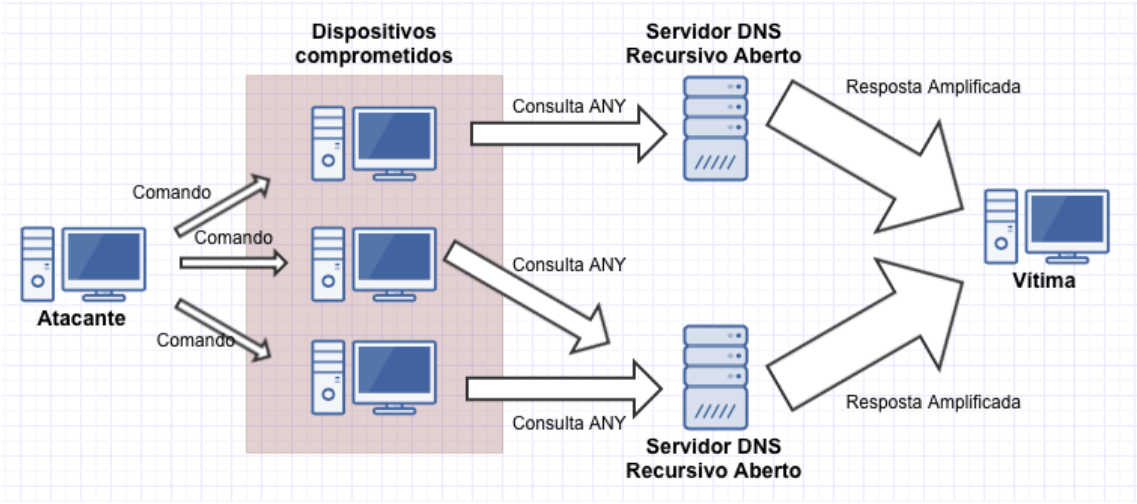

Figura 2.4. Ataque de negação de serviço por amplificação e reflexão utilizando DNS.

Ataque usando o *Network Time Protocol*

O *Network Time Protocol* (NTP) é um protocolo utilizado para sincronizar o relógio de sistemas computacionais. Ele funciona sobre uma arquitetura cliente-servidor onde os clientes sincronizam os seus relógios com os servidores. Cada servidor, que informa o tempo, é ranqueado baseado em sua precisão (*stratum*) [14].

O ataque de negação de serviço utilizando o NTP é bastante parecido com o ataque do DNS. A grande diferença é que em vez de se utilizar servidores DNS recursivos aberto utiliza-se servidores NTP que têm a funcionalidade "*monolist*" habilitada. Essa funcionalidade faz com que o servidor NTP responda os últimos 600 IPs que se conectaram ao servidor.

O ataque consiste em o atacante enviar uma mensagem "*get monolist*" com o IP de origem alterado para um servidor NTP que permita essa funcionalidade. Em seguida, o servidor NTP

gera a resposta com os últimos 600 IPs, que é muito maior que a consulta, e envia para a vítima [15].

2.3 Ataque usando o *Simple Network Management Protocol*

O ataque usando o *Simple Network Management Protocol* (SNMP) é o foco do estudo e é explicado a seguir.

O SNMP é um protocolo para gerenciamento de redes que define uma maneira universal para que informações de controle possam ser definidas para algum dispositivo e como transferir essas informações entre esse dispositivo e o dispositivo de gerenciamento [16].

2.3.1 Histórico do SNMP

O SNMP versão 1 ou SNMPv1 foi desenvolvido no começo de 1988 e está documentado nas RFCs 1155, 1156, 1157 e 1213. Esses documentos contêm as definições básicas das principais estruturas do SNMP: *Structure of Management Information* (SMI), *Management Information Bases* (MIB) e o próprio protocolo. Embora muito utilizado, o SNMPv1 foi criticado por sua segurança. Ele utilizava uma forma de autenticação fraca por *community string* que são palavras de segurança, igual senhas, enviadas junto com as mensagens do protocolo para identificar dispositivos autorizados [17] [18] [16] [19].

A segunda versão do SNMP ou SNMPv2 foi publicada em 1993. Essa nova versão modificou o modelo de segurança, fez algumas alterações nas operações do protocolo e definiu a nova versão do SMI (SMIv2) [20]. Entretanto, essa nova versão do SNMP não foi utilizada em grande escala. Um dos motivos foi sua segurança mais complexa e exigir maior uso de poder computacional. Como resultado foram desenvolvidas modificações para o SNMPv2. Duas dessas versões são: o SNMPv2c [21] que é a segunda versão do SNMP com todas as suas alterações mas com a segurança por *community string* da versão um e o SNMPv2u que utiliza segurança por usuários que é mais simples que a segurança da versão 2 original [22] [23].

Outras versões foram desenvolvidas, mas devido a falta de um padrão nenhuma obteve aceitação igual ao SNMPv1. Desse modo, muitos usuários do protocolo continuaram usando a

versão 1 ou optaram pela versão SNMPv2c.

Por fim, em 1998 a terceira versão do SNMP ou SNMPv3 foi desenvolvida. A terceira versão fez melhorias ao protocolo, adicionou algumas ferramentas para ajudar a administração da rede e permitiu a escolha de diferentes formas de segurança a fim de evitar os problemas que ocorreram com a segunda versão.

2.3.2 Tipos de dispositivos

Existem dois tipos de dispositivos em uma rede com SNMP que são os nós gerenciados e as estações de gerenciamento. Cada dispositivos roda um software diferente dependendo do seu tipo. Os nós gerenciados podem ser qualquer dispositivo de rede que se comunique pelo protocolo IP e que tenha o software de agente SNMP adequado rodando. Nós gerenciados típicos são: computadores, roteadores, *switches*, *bridges* e *hubs*. Outros exemplos podem ser impressoras, *scanners* e diversos dispositivos eletrônicos. Cada nó gerenciado deve conter um *SNMP Agent* que é um software que implementa o protocolo SNMP e permite que informações sejam enviadas para as estações de gerenciamento. Além disso, o software agente permite receber instruções das estações. O outro software necessário é o *SNMP Management Information Base* (MIB) que define quais informações do nó gerenciado que podem ser coletadas e usadas como controle pela estação de gerenciamento. A troca de informações no SNMP acontece pela troca de objetos da MIB.

As estações de gerenciamento também são qualquer dispositivo IP. Entretanto, devido ao seu papel de controle de dispositivos e monitoramento, costumam ser computadores ou similares. Cada estação roda o *SNMP Manager* que é um software que implementa o protocolo SNMP e que permite que as estações enviem instruções e obtenham as informações dos nós gerenciados. A Figura 2.5 demonstra um exemplo simplificado da arquitetura SNMP.

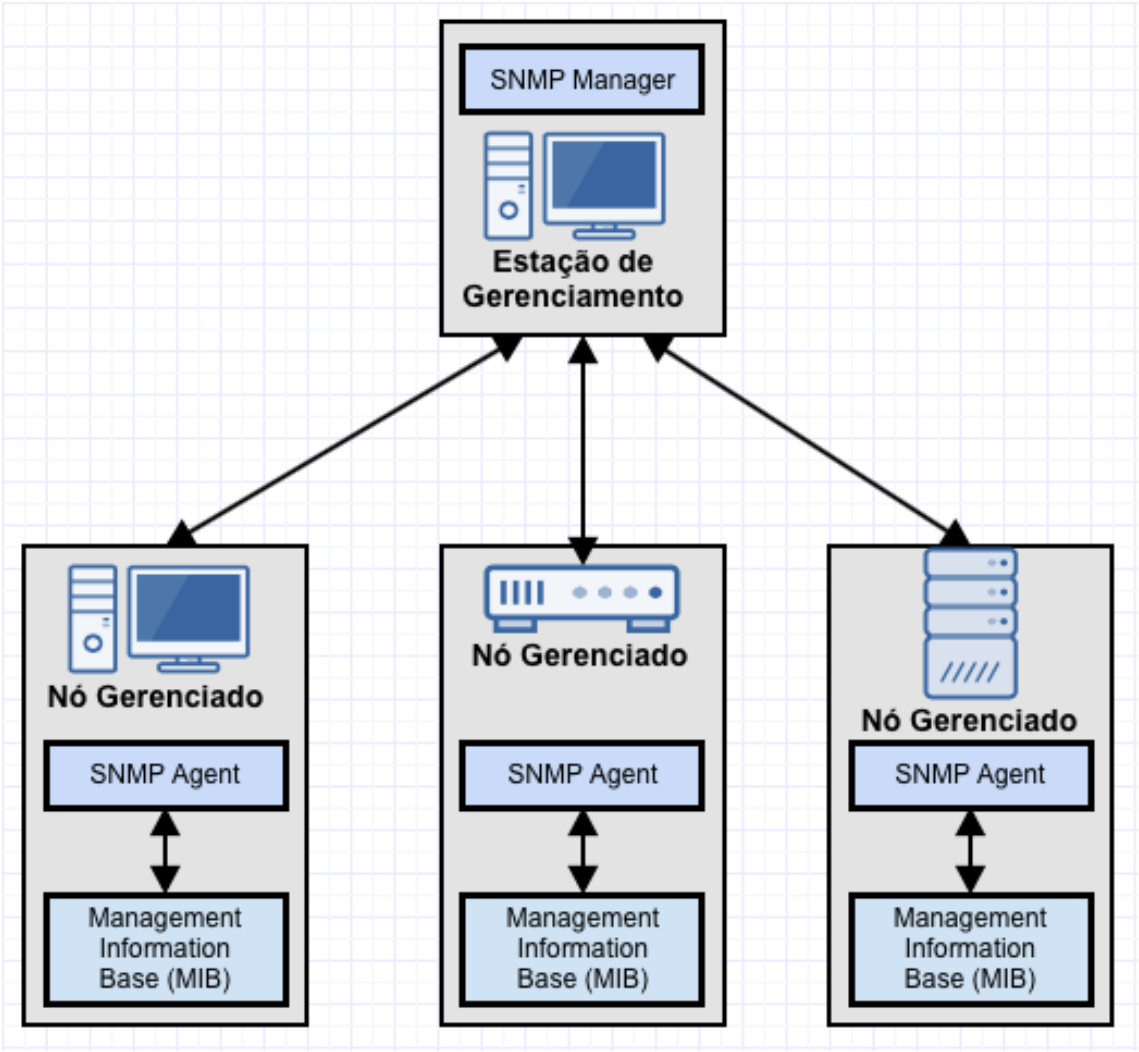

Figura 2.5. Arquitetura SNMP simplificada.

2.3.3 *Management Information Base*

Em gerenciamento de redes, existem duas operações essenciais. Uma delas é obter informações dos dispositivos gerenciados e a outra é modificar o comportamento desses dispositivos. Uma opção para prover essas funcionalidades é um protocolo orientado a comandos, onde comandos são definidos para recuperar informação e para controlar os dispositivos. Entretanto, devido a heterogeneidade dos dispositivos em uma rede tal protocolo teria que prover inúmeros comandos. Isso ocorre pois cada tipo de dispositivo pode necessitar de diferentes comandos e toda vez que novos dispositivos fossem criados novos comandos também teriam que aparecer.

O SNMP, para solucionar esse problema, é um protocolo orientado a informação. Todos os dispositivos são definidos em termos de variáveis que podem ser lidas e escritas. Em vez de enviar comandos, a estação de gerenciamento lê os valores de variáveis para obter as informações

do dispositivo, e escreve em variáveis, para controlar o dispositivo.

As variáveis trocadas entre os dispositivos gerenciados e as estações de gerenciamento são denominadas objetos. Cada objeto descreve uma característica do dispositivo e podem ser bem genéricos, e utilizados por todos, como o endereço IP, ou bem específicos, só existindo em determinados tipos de dispositivos. O conjunto dessas variáveis é chamado de *Management Information Base* ou MIB [24]. A MIB é composta por módulos e cada equipamento pode implementar os módulos apropriados.

Outro problema que surge com a heterogeneidade de equipamentos é que cada um pode representar informações de diferentes formas. O padrão *Structure of Management Information* (SMI) é quem define as regras de como os objetos do SNMP, ou os objetos MIB são construídos. No SMI os objetos são descritos utilizando a ISO *Abstract Syntax Notation 1* (ANS.1) [25]. O SMI também define informações como o padrão de descrição dos objetos, os tipos de objeto que podem ser criados e como nomeá-los.

Cada objeto da MIB possui dois nomes. Um é a descrição do objeto que é um texto, e deve ser fácil de entender. O segundo é o identificador do objeto, uma sequência de inteiros separados por ponto que especifica a posição do objeto em uma classificação hierárquica que contém todos os objetos. Essa classificação é mantida pela ISO e ITU e funciona de forma idêntica ao DNS, indo do nível mais genérico ao mais específico. O identificador do objeto é composto do identificador de cada nível, separados por ponto, até o seu próprio identificador. A Figura 2.6 exemplifica a classificação hierárquica e o caminho até a variável *sysName* que armazena o nome do sistema.

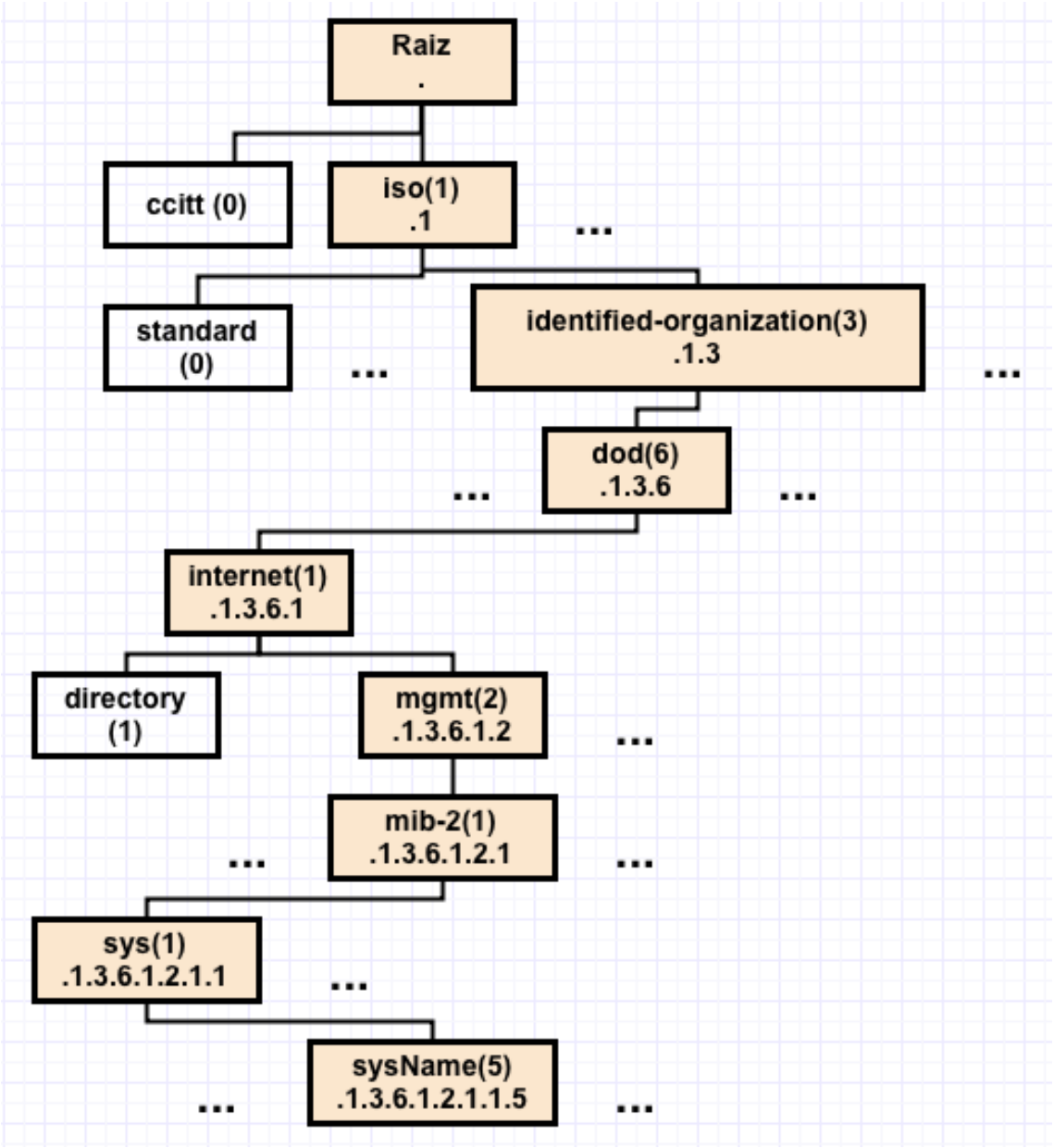

Figura 2.6. Árvore MIB simplificada.

2.3.4 Modos de operação

O SNMP possui duas formas de troca de informação. A primeira se dá quando a estação de gerenciamento pede informação para os nós gerenciados. Na segunda, os nós gerenciados que enviam informação para as estações.

Em sua primeira versão foram definidas cinco operações entre estações de gerenciamento e nós gerenciados. Na segunda e terceira versões mais duas operações foram definidas e algumas modificações foram feitas. Assim, as principais operações são:

- **GetRequest**: Mensagem enviada por uma estação de gerenciamento para um nó gerenciado a fim de receber o valor de determinadas variáveis. As variáveis são enviadas como parâmetro da operação pela estação de gerenciamento;
- **Response**: Na versão 1 essa mensagem chamava GetResponse. Ela serve para retornar os valores das variáveis pedidas pela estação de gerenciamento ou erros que indicam problema com a solicitação feita;
- **GetNextRequest**: Na MIB podem existir variáveis que representam dados em forma de tabelas. Por exemplo, não faz sentido existir uma variável diferente para cada IP de um roteador. Por esse motivo uma variável do tipo tabela é usada onde em cada entrada da tabela se tem um IP. Esse tipo de dado causa um problema para o GetRequest pois uma estação de gerenciamento deve saber o identificador de todas as entradas caso deseje recuperar essas informações. Para solucionar esse problema o SNMPv1 criou o GetNextRequest. Esse comando é parecido com o GetRequest, mas em vez de retorna a variável que chega como parâmetro ele retorna a próxima variável. Desse modo a estação pode ir percorrendo a tabela, sempre atualizando o valor da variável de parâmetro do GetNextRequest com a última recebida;
- **GetBulkRequest**: Quando uma estação de gerenciamento necessita de uma tabela ela deve pedir um a um as variáveis usando o GetNextRequest. Esse processo é mais demorado e resulta em mais tráfego na rede do que uma mensagem só. Por isso, o SNMPv2 introduziu o comando GetBulkRequest. Com esse comando variáveis normais e tabulares podem ser consultadas com uma única mensagem. Ele possui como parâmetro a lista de variáveis, sendo que as variáveis comuns devem vir primeiro e as tabulares depois, um parâmetro chamado *Non Repeaters* que especifica quantas das variáveis da lista são normais e um parâmetro *Max Repetitions* que diz quantas iterações devem ser feitas nas demais variáveis da lista. Observa-se que com o GetBulkRequest uma estação deve saber quantas entradas da tabela pedir, caso ela ponha um número maior para o *Max Repetitions* as próximas variáveis depois da tabela serão retornadas;
- **SetRequest**: Quando uma estação de gerenciamento quer alterar o funcionamento de um nó gerenciado ele envia um SetRequest. Esse tipo de mensagem contém como parâmetro as variáveis que devem ser alteradas e os valores que devem ser escritos nessas variáveis. O nó gerenciado, por sua vez, cria uma mensagem de Reponse que contém dados indicando

o sucesso da operação ou códigos de erro e envia para a estação;

- **Trapv2**: Na versão 1 esse tipo de mensagem chamava apenas Trap. Trap é um conjunto de condições que um nó gerenciado monitora. Quando um evento, configurada nos agentes, ocorre que ativa essa Trap uma mensagem do tipo Trapv2 é enviado do nó para a estação. Desse modo, a estação é informada de acontecimentos sem precisar fazer o GetRequest. Quando uma estação recebe uma Trapv2 ela não gera nenhuma mensagem de resposta, dizemos então que a Trapv2 é uma mensagem não confirmada;
- **InformRequest**: O InformRequest é um tipo de mensagem trocada entre estações de gerenciamento para informar sobre ocorrências na rede. Por exemplo, caso um nó gerenciado envie uma Trap, a estação pode escolher informar outras estações sobre esse acontecimento. Esse tipo de mensagem gera um Response então podemos dizer que é uma mensagem confirmada.

2.4 Considerações finais

Na primeira versão do SNMP, a sua segurança era realizada por *community strings* que funcionam como senhas. Todas as mensagens trocadas deveriam conter esse nome de comunidade e as que contém nomes errados eram simplesmente descartadas. Essa estratégia, embora sirva como um tipo de proteção, é bem limitada. Os nomes de comunidade são passados em claro na rede e podem ser capturados por todos que vierem a escutar a rede. Como se isso já não bastasse, muitos dispositivos que utilizam SNMP são configurados com os nomes padrões de comunidade: "*public*" para leitura e "*private*" para escrita. Isso já é grave pois abre caminho para que usuários não autorizados alterem o funcionamento dos dispositivos vulneráveis.

Na segunda versão do SNMP foi introduzida a mensagem GetBulkRequest para facilitar e agilizar o pedido de variáveis pelas estações de gerenciamento. Entretanto, esse comando possui características semelhantes ao "ANY" do DNS e ao "Monolist" do NTP. O GetBulkRequest também pode ser usado como um comando de amplificação pois uma mensagem pequena pode gerar uma outra, substancialmente maior, em resposta. Como o GetBulkRequest só existe nas versões 2 e 3, o ataque só pode ser realizado nessas versões e precisaria burlar as respectivas seguranças. Entretanto, no SNMPv2c existe o GetBulkRequest e a segurança implementada é a mesma da versão 1 que é muito falha. Além disso, a versão SNMPv2c ainda é muito usada.

Portanto, a combinação do comando GetBulkRequest e da pouca segurança da versão SNMPv2c fazem com que o protocolo SNMP seja um candidato para ataques de negação de serviço por reflexão e amplificação.

A ferramenta a ser desenvolvida neste trabalho deve descobrir possíveis refletores, utilizando um comando que peça apenas uma única variável, e realizar o ataque pelo GetBulkRequest com os melhores argumentos para potencializar a amplificação gerada.

CAPÍTULO 3

FERRAMENTA - SNMP STRIKER

Neste capítulo, o *design* e a implementação da ferramenta são apresentados. O *design* diz respeito às escolhas de funcionalidade e à arquitetura utilizadas. A implementação explica como que essas funcionalidades foram implementadas.

As funcionalidades gerais da ferramenta são:

- Realizar uma varredura em sub-redes definidas para encontrar possíveis refletores;
- Importar e exportar as informações dos refletores encontrados;
- Escolher quais refletores devem participar do ataque;
- Configurar a intensidade do ataque em uma escala de 1 a 10;
- Escolher uma vítima e realizar o ataque.

3.1 *Design*

A ferramenta de negação de serviço desenvolvida utiliza uma arquitetura baseada em três módulos.

Como demonstrado na Figura 3.1, o módulo Gui, que é a interface com o usuário, utiliza serviços dos módulos Scanner e Striker. Esses dois últimos são independentes e podem ser atualizados sem impactar um ao outro.

O módulo Scanner tem como função encontrar os dispositivos vulneráveis e obter os parâmetros que venham maximizar a amplificação para cada dispositivo. Assim, entende-se como dispositivos vulneráveis os dispositivos que rodam o SNMPv2c e que estão configurados com os nomes de comunidade padrões.

O Striker é responsável por realizar o ataque em si. Ele recebe os parâmetros do Scanner, realiza *IP Spoofing* e envia pelo canal.

Por fim, o Gui realiza a integração dos outros dois módulos com as configurações do usuário.

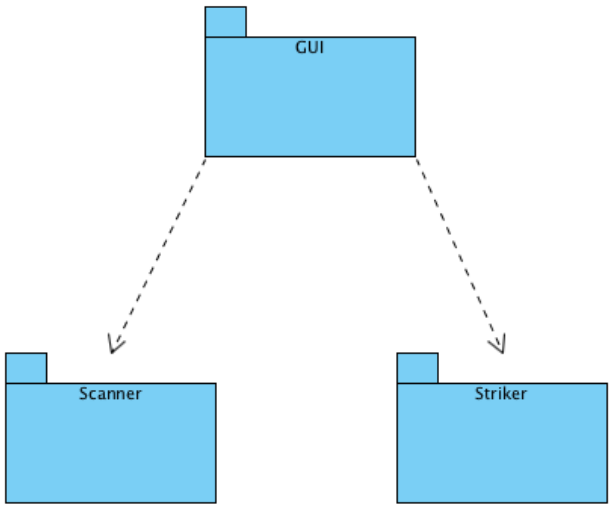

Figura 3.1. Diagrama de pacotes.

3.1.1 Gui - *Graphical user interface*

A ferramenta interage com o usuário por uma interface gráfica, e através dela é possível:

- Procurar dispositivos vulneráveis;
- Importar endereços e configurações de dispositivos que foram descobertos anteriormente;
- Exportar endereços e configurações de dispositivos descobertos;
- Selecionar quais dispositivos devem ser utilizados no ataque e quais suas configurações;
- Escolher a potência do ataque;
- Realizar o ataque.

3.1.1.1 Procurando dispositivos

Para a descoberta de dispositivos vulneráveis o módulo Gui utiliza serviços do Scanner. Assim, configura-se pela interface gráfica qual o IP base que vai ser utilizado; qual a sub rede; quais as comunidades que devem ser testadas; quais portas enviar; qual o *timeout* dos pacotes enviados; quantos pacotes enviar caso ocorra *timeout*; e qual o tempo de espera total. Para cada novo dispositivo encontrado é calculado sua respectiva amplificação. Ao final do processo se tem a lista dos dispositivos refletores encontrados.

3.1.1.2 Importar e exportar dispositivos

A ferramenta possui funcionalidades para importar e exportar dispositivos. Desse modo, a descoberta de possíveis refletores e os testes do ataque podem ocorrer em momentos diferentes.

3.1.1.3 Selecionar dispositivos

O usuário da ferramenta é capaz de escolher quais dispositivos utilizar como amplificadores. Além disso pode-se configurar o valor do *Max Repetitions* do GetBulkRequest.

3.1.2 Scanner

O módulo Scanner é responsável por reconhecer dispositivos refletores no conjunto de IPs selecionados pelo usuário; e calcular, para cada dispositivo encontrado, qual o tipo de mensagem com melhor amplificação.

3.1.2.1 Reconhecimento ou detecção

O reconhecimento de equipamentos vulneráveis é feito através da mensagem GetNextRequest do protocolo SNMP. O GetNextRequest irá enviar um identificador de objeto da MIB e solicitar ao equipamento a próxima variável disponível. Dessa forma, garante-se uma resposta do equipamento, caso ele seja vulnerável, pois sempre será gerada uma resposta, mesmo de erro.

A mensagem é enviada para todos os conjuntos IP, comunidade e porta únicos, ou seja, todos os dispositivos são testados com todas as comunidades e portas que o usuário definiu.

O módulo irá então esperar por respostas. Caso o tempo de *time out* escolhido pelo usuário se esgote, outras mensagens serão enviadas. O número de mensagens também é escolha do usuário. Por fim, quando algum dispositivo responder ele será adicionado à lista de equipamentos vulneráveis e sua amplificação calculada na próxima etapa.

3.1.2.2 Cálculo da amplificação

A amplificação é feita através da mensagem GetBulkRequest do protocolo SNMP que é enviada de uma estação de gerenciamento para um nó gerenciado. Pelo GetBulkRequest envia-

se uma lista de identificadores de variável e os parâmetros *Non Repeaters* e *Max Repetitions*. O nó gerenciado pega o valor das primeiras n variáveis da lista, onde n é o valor de *Non Repeaters*, e para as demais variáveis pega o valor das próximas m iterações a partir delas, onde m é o valor de *Max Repetitions*. Para gerar uma mensagem com o maior fator de amplificação envia-se o identificador de apenas uma variável, escolhe-se o zero para o *Non Repeaters*, indicando que é para fazer iterações nessa variável com o valor de *Max Repetitions*.

A decisão do valor correto de *Max Repetitions* é um grande desafio. Quando um nó gerenciado recebe um GetBulkRequest, este coloca, em uma única resposta, todas as variáveis que foram pedidas. Entretanto, caso essa mensagem extrapole o tamanho máximo de um datagrama IP (65.535 bytes) nada será retornado. Logo, o melhor valor é aquele que gera uma resposta próxima dos 65 mil bytes em tamanho, mas sem ultrapassar o limite. Um agravante dessa situação é que dispositivos possuem informações diferentes em sua MIB. Desse modo, um mesmo valor de *Max Repetitions* irá formar dois tamanhos diferentes de reposta. Para contornar essa situação é utilizado um cálculo progressivo.

O cálculo progressivo se baseia em enviar mais de uma mensagem de GetBulkRequest e ir aumentando o valor do *Max Repetitions* progressivamente. Dessa forma, testa-se os limites do dispositivo. Essa abordagem gera mais tráfego do que escolher um único valor, mas garante que todos os dispositivos tenham uma melhor amplificação. Além disso, esse cálculo é transparente para o usuário que irá ver apenas o melhor valor de amplificação, isso é um diferencial comparando com outras ferramentas [26]. O cálculo da amplificação é dado pela Fórmula (3.1):

$$A = \frac{RespIpSize + RespUdpSize + RespSNMPSize}{ReqIpSize + ReqUdpSize + ReqSNMPSize} \tag{3.1}$$

Onde,

- A = Amplificação;
- RespIpSize = Tamanho em bytes do cabeçalho IP da resposta;
- RespUdpSize = Tamanho em bytes do cabeçalho UDP da resposta;
- RespSNMPSize = Tamanho em bytes da mensagem SNMP de resposta;
- ReqIpSize = Tamanho em bytes do cabeçalho IP da requisição;
- ReqUdpSize = Tamanho em bytes do cabeçalho UDP da requisição;
- ReqSNMPSize = Tamanho em bytes da mensagem SNMP da requisição.

Tendo em vista que o tamanho dos cabeçalhos IP e UDP são o mesmo para ambos a resposta e a requisição tem-se:

$$A = \frac{20 + 8 + RespSNMPSize}{20 + 8 + ReqSNMPSize} \tag{3.2}$$

3.1.3 Striker

O módulo Striker recebe as configurações para a mensagem SNMP, constrói o pacote UDP e IP realizando o *IP Spoofing*. Em seguida, executa o ataque na potência estipulada. Assim, o usuário pode, através da interface gráfica, utilizar o Max Repetition calculado pelo módulo Scanner, que é o padrão da ferramenta, ou escolher um outro valor.

3.1.3.1 *Ip spoofing*

O *IP spoofing* é realizado modificando o campo *Source IP Address* do cabeçalho IP. Normalmente, esse campo leva o IP do computador enviando o pacote. Entretanto, nos ataques de reflexão e amplificação, esse campo é alterado com o valor do IP da vítima. Desse modo, quando o refletor recebe o pacote ele irá enviar a resposta amplificada para a vítima, caracterizando o ataque.

3.1.3.2 Taxa de envio

A taxa de envio é a quantidade de pacotes que a ferramenta vai enviar para os refletores. Essa taxa é escolhida pelo usuário por meio da interface gráfica. Quanto maior for a taxa, mais pacotes serão enviados e maior deverá ser o efeito do ataque.

3.2 Implementação

Para a implementação da ferramenta foi escolhida a linguagem de programação Java, integrada com a linguagem C. Java foi utilizado devido a sua portabilidade e facilidade de criação de interfaces gráficas. Por outro lado, a linguagem C provê rápida execução e manipulação de todas as partes da mensagem enviada pela rede.

Desse modo, a linguagem Java foi utilizada na implementação do Scanner e Gui enquanto

Java e C foram usados no Striker.

3.2.1 Gui

O módulo Gui foi implementado em Java usando o Swing [27], que é uma API para construção de interfaces de usuário gráficas. A Figura 3.2 mostra a tela principal da interface em seu estado inicial.

Figura 3.2. SNMP Gui.

A partir dessa interface é possível adicionar e remover novos dispositivos, acompanhar os dispositivos vulneráveis encontrados, importar e exportar dispositivos, escolher o alvo do ataque, selecionar a taxa de envio e realizar o ataque.

3.2.1.1 Adicionar e remover

Para adicionar novos dispositivos o usuário deve clicar no botão Add. Uma nova tela será aberta com as configurações de descoberta como mostra a Figura3.3

Em cada campo o usuário deve:

- Base IP - Definir o IP base para a descoberta de novos dispositivos. O IP deve ser no formato xxx.xxx.xxx.xxx;
- Range - Definir em qual sub rede deve-se procurar dispositivos. Caso nenhuma das duas opções seja selecionada, a ferramenta irá testar apenas o IP base.
- Community - Definir quais comunidades a ferramenta deve usar. Comunidades devem ser separadas por vírgula. A comunidade "public"é definida como padrão.

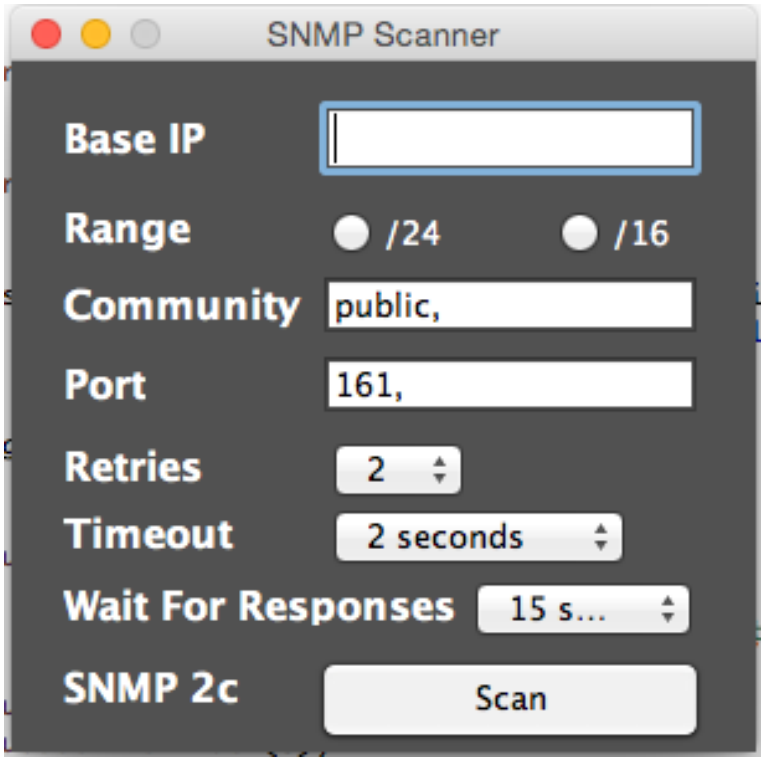

Figura 3.3. SNMP Scanner.

- Port - Definir quais portas devem ser testadas. Do mesmo modo que as comunidades, as portas devem ser separadas por vírgulas e a porta 161 é o padrão.
- Retries - Definir o número de mensagens, além da primeira, que devem ser enviadas. As mensagens são enviadas caso nenhuma resposta seja recebida. Os valores possíveis são: 0,1,2,3,4,5,10;
- Timeout - Definir por quanto tempo a ferramenta deve esperar antes de enviar uma nova mensagem. Os valores possíveis são: 0.5, 1, 1.5, 2, 2.5, 3 segundos.
- Wait For Responses - Definir o tempo total de espera. O tempo selecionado é dividido em dois, sendo o primeiro para o reconhecimento de novos dispositivos e o segundo para o cálculo da amplificação.

Depois de definido as configurações, basta apertar o botão Scan. Finalizado o processo, a interface mostrará uma mensagem informando quantos dispositivos foram encontrado. As Figuras 3.4 e 3.5 mostram a mensagem quando nenhum dispositivo é encontrado e quando um dispositivo é encontrado, respectivamente.

Para remover um dispositivo basta selecioná-lo na tabela e clicar Remove.

Figura 3.4. Mensagem - Nenhum dispositivo encontrado.

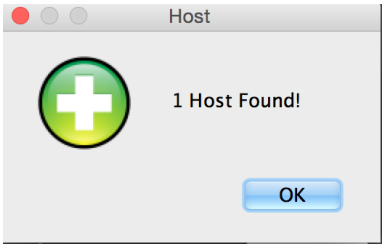

Figura 3.5. Mensagem - Um dispositivo encontrado.

3.2.1.2 Tabela de dispositivos vulneráveis

Quando a ferramenta encontra um novo dispositivo ele aparece na tabela. Essa tabela mostra:

- IP: O endereço IP do dispositivo;
- Community: A comunidade que o dispositivo responde;
- Port: A porta que o dispositivo escuta;
- MaxAmplification - VB: A amplificação obtida pelo Scanner e o número de variáveis que vieram no pacote de resposta;
- Custom MaxRep: Um campo para alterar o número de *MaxRepetitions* que será usado no ataque;
- Select: Campo para escolher se aquele dispositivo deve ou não ser utilizado no ataque.

A Figura 3.6 mostra a tabela com um dispositivo.

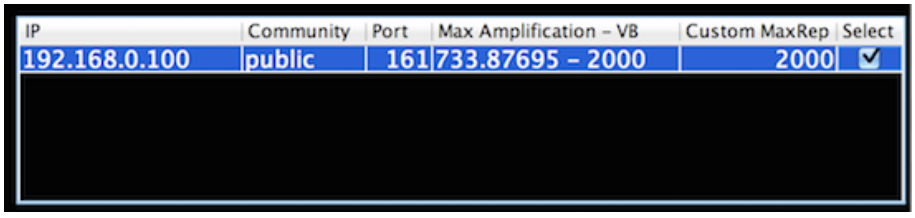

Figura 3.6. Tabela com um dispositivo vulnerável.

3.2.1.3 Importar e exportar

A ferramenta possui funcionalidades para exportar e importar listas de dispositivos. A lista é salva em um arquivo .json, como mostrado na Figura 3.7.

ListaExemplo.json

```
[
    {
        "port":161,
        "ipAddress":
        {
            "port":161,
            "inetAddress":"XXX.XXX.XXX.XXX"
        },
        "community":"public",
        "version":1,
        "amplificationPower":123.44615,
        "requestPacket":[48,35,2,1,1,4,6,112,117,98,108,105,99,
        -91,22,2,4,78,22,-24,-45,2,1,0,2,2,1,82,48,7,48,5,6,1,
        43,5,0],
        "select":false
    }
]
```

Figura 3.7. Json com dispositivo vulnerável.

3.2.1.4 Realizar ataque

Para realizar o ataque, deve existir pelo menos um dispositivo vulnerável selecionado na tabela. Além disso, o usuário deve colocar o IP do dispositivo que será a vítima no campo Set Target.

A potência é ajustada pela barra Power que varia do valor 1, valor padrão, até 10. Cada valor de potência significa o número de pacotes por segundo que é enviado para todos os dispositivos selecionados. O aumento de potência é dado pela Fórmula 3.3,

$$Pps = 10^{power-1} \tag{3.3}$$

Onde,

- Pps = pacotes por segundo;
- power = valor da potência.

Desse modo, com a potência em 1, será enviado 1 pacote por segundo para cada refletor. No nível dois são enviados 10 pacotes e assim por diante.

Para iniciar o ataque basta apertar o botão Strike. Após iniciado, o botão Strike será substituído por um botão Stop que para o ataque.

A capacidade de envio máxima acontece com a potência 10. Nesse caso, seriam enviados um bilhão de pacotes por segundo. Sabe-se que não é possível alcançar esse valor, dadas as limitações de hardware e do protocolo de enlace, entretanto, a escala de dez níveis foi criada para que o efeito de saturação possa ser melhor observado. Além disso, a ferramenta estaria preparada para futuras evoluções de hardware e infraestrutura.

3.2.2 Scanner

O módulo Scanner foi implementado em java utilizando a API SNMP4J [28]. A Figura 3.8 traz o diagrama de classes simplificado desse módulo.

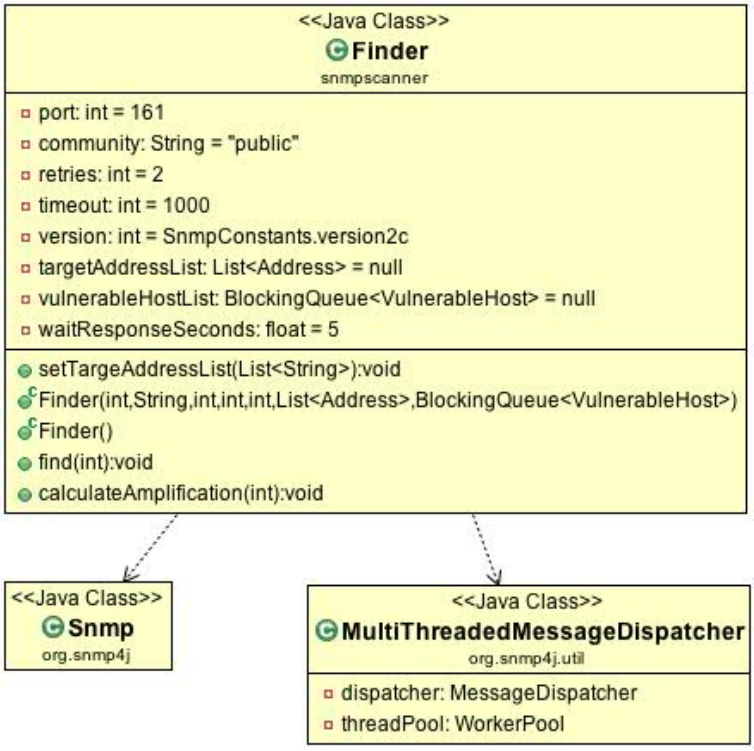

Figura 3.8. Diagrama de classes do módulo Scanner.

A classe Finder é a principal classe do módulo e disponibiliza funções para descoberta de novos dispositivos e para o cálculo da amplificação. A classe recebe os parâmetros porta de destino, comunidade, número de *retries*, tempo de *timeout* para cada *request*, versão do SNMP e tempo máximo para esperar por respostas no construtor. Todos os parâmetros contêm valores *default*.

Para realizar a descoberta de novos dispositivos, deve-se definir a lista de IPs a ser usado, com aquelas configurações, e ativar a função *find()* no objeto. A função *find()* recebe como

parâmetro um inteiro que representa o número de *threads* criadas para tratar as respostas.

A descoberta de novos dispositivos é feita pelo GetNextRequest com a variável *.iso.org.dod.internet.m 2.system.sysContact*. Desse modo, os sistemas vulneráveis, ou seja, que estão escutando na porta e respondem àquela comunidade, irão enviar como resposta a variável *.iso.org.dod.internet.mgmt.m 2.system.sysName* que é a próxima. Logo, todas as respostas recebidas pela ferramenta serão de dispositivos vulneráveis. Caso o dispositivo não possua a variável, sysName, ele responderá com a próxima variável disponível. O importante é que a mensagem GetNextRequest sempre induzirá uma resposta em equipamentos vulneráveis.

A função *calculateAmplification()* calcula a amplificação dos dispositivos através da mensagem GetBulkRequest. Para implementar o efeito da amplificação progressiva são enviados 6 GetBulkRequest com os valores de *Max Repetitions* 10, 50, 100, 300, 1000 e 2000. A resposta de maior tamanho fica registrada como melhor amplificação. A variável utilizada no GetBulkRequest é a *.iso.org*. Embora a MIB do SNMP não inicie nessa variável, os nós gerenciados conseguem utilizar esse identificador. Isso é feito pois, quanto maior o valor da variável de parâmetro, maior o tamanho da mensagem GetBulkRequest, o que diminui a amplificação total. Desse modo, foi escolhido o menor valor possível para otimizar a amplificação.

Caso a função para cálculo de amplificação seja chamada depois da função *find()* só os dispositivos já encontrados terão sua amplificação calculada. Por outro lado, se *calculateAmplification()* for chamada sozinha, as mensagens de GetBulkRequest serão enviadas a todos os dispositivos definidos na lista.

3.2.2.1 SNMP4J

SNMP4J é uma API que implementa o protocolo SNMP para java [28]. Na implementação da ferramenta foram utilizadas as funcionalidades de enviar e receber mensagens assíncronas para agentes SNMP. Tais funções são disponibilizadas pela classe Snmp (Figura 3.8). A classe Finder do módulo Scanner cria um objeto Snmp e, através dele, envia as mensagens.

A classe MultiThreadMessageDispatcher é a responsável por tratar as mensagens de resposta. A classe Finder utiliza seus serviços e entrega as *threads* criadas para tratar de respostas.

A Figura 3.9 demonstra a interação entre o usuário, a ferramenta e os dispositivos durante

a descoberta.

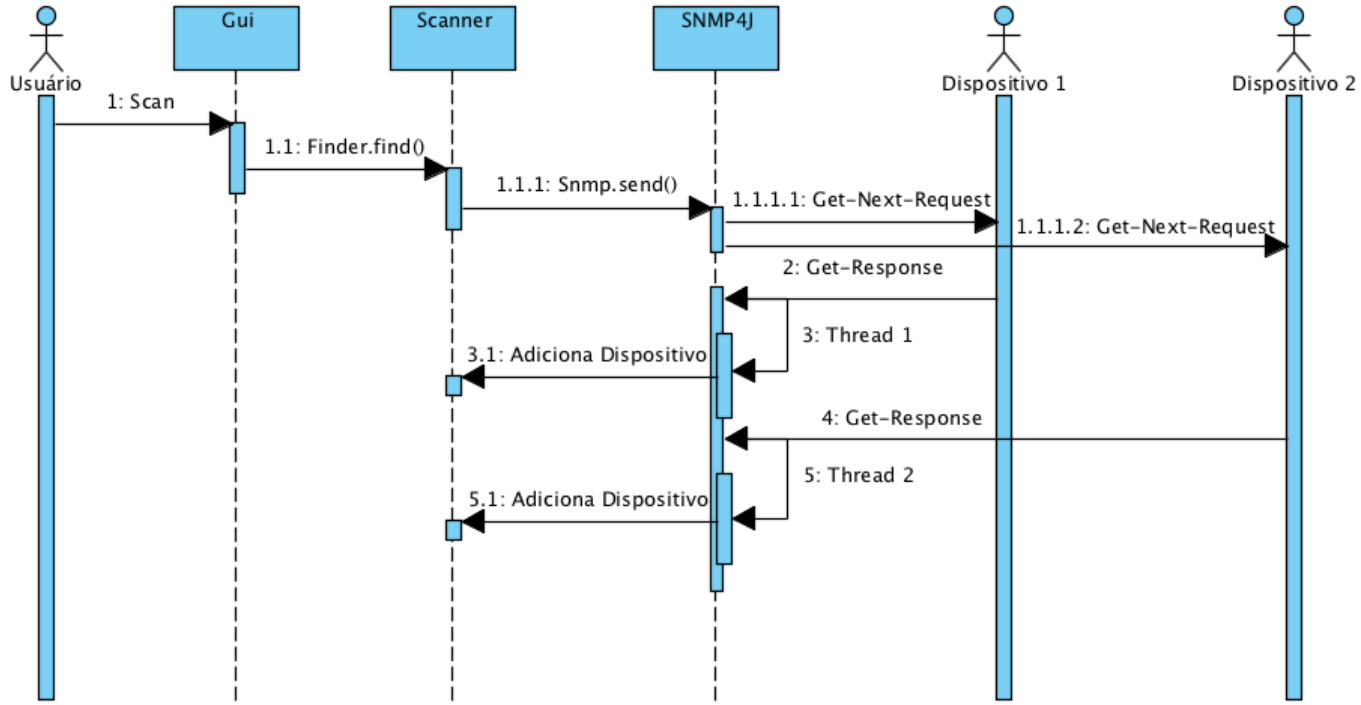

Figura 3.9. Diagrama de sequência na descoberta de novos dispositivos.

Primeiro o usuário ativa a função *Scan* pela interface com o usuário (passo 1). O módulo Gui, então, instancia o objeto Finder e chama seu método *find()* (passo 1.1). O módulo Scanner cria o objeto Snmp e envia os pacotes pelo método *send()* (passo 1.1.1). Os dispositivos recebem as mensagens GetNextRequest e respondem, se vulneráveis, para a ferramenta (passos 1.1.1.1, 1.1.1.2, 2, 4). O módulo SNMP4J recebe cada mensagem assincronamente e passa seu tratamento para uma *thread* disponível (passos 3, 5). Cada *thread* trata a resposta e adiciona o novo dispositivo (passos 3.1, 5.1).

Assim, observando a Figura 3.10 é possível ver o processo de cálculo de amplificação. Ele começa quando o módulo Gui chama a função *calculateAmplification()* do objeto Finder.

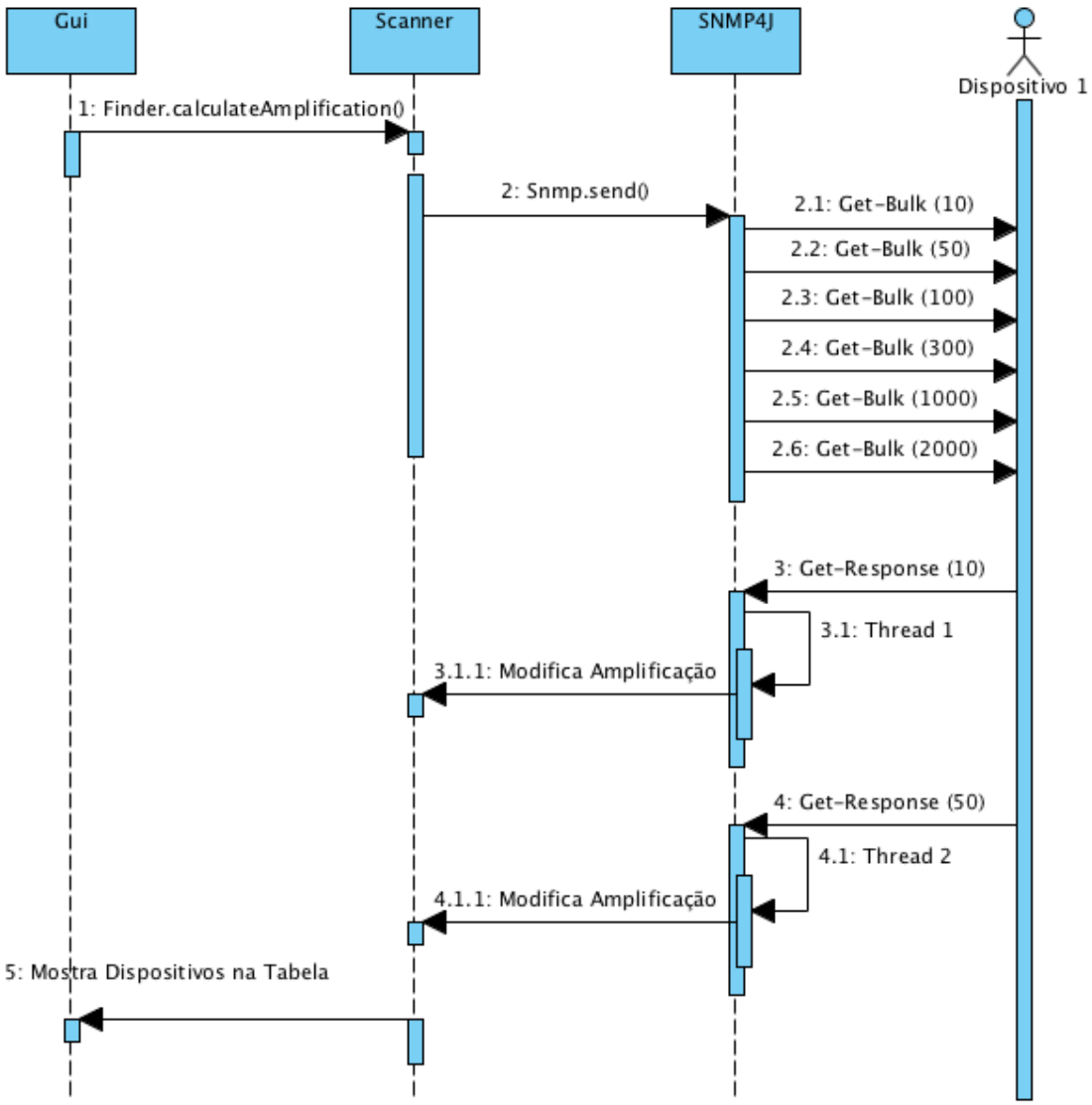

Figura 3.10. Diagrama de sequência no cálculo de amplificação.

Na Figura 3.10 é possível observar a interação apenas com um dispositivo. Vale notar que o dispositivo representado supostamente não teria respondido as mensagens de GetBulkRequest pedindo mais de 50 variáveis. Logo, a sua amplificação seria com 50 de *Max Repetitions*.

3.2.3 Striker

O módulo Striker foi implementado em Java e C. Ele é composto de uma única classe, mostrada na Figura 3.11.

Para cada refletor utilizado no ataque é instanciado um objeto da classe Striker. Cada objeto recebe os parâmetros para o ataque em seu construtor. Quando o ataque é iniciado os objetos rodam em *threads*, fazendo-se necessário que a classe Striker implemente Runnable.

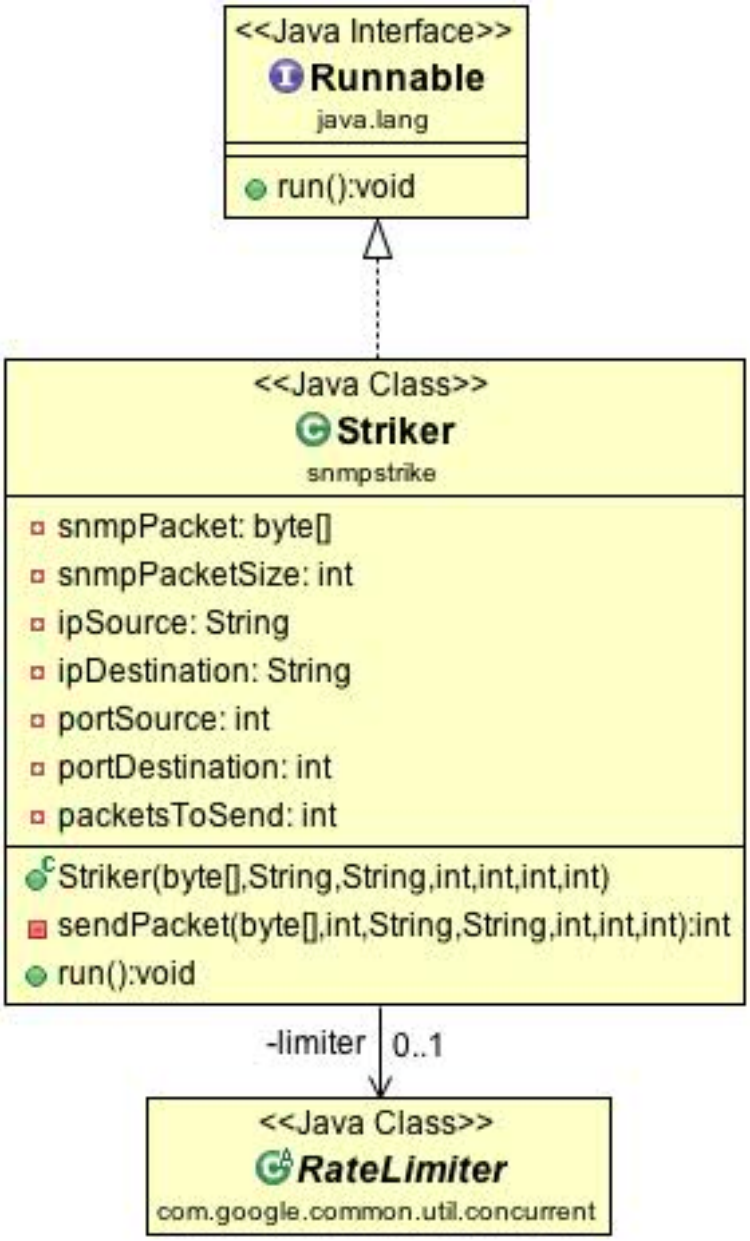

Figura 3.11. Diagrama de classes do módulo Striker.

3.2.3.1 JNI

A classe Striker possui dois métodos: *run()* e *sendPacket()*. A assinatura do método sendPacket é feita em java, enquanto sua implementação é em C. Isso ocorre porque java não tem acesso ao cabeçalho do protocolo IP sem o uso de bibliotecas extras. Desse modo, a solução mais simples e eficiente foi implementar o corpo da função em C e usar JNI [29].

JNI é um *framework* que permite ao código java, rodando em uma máquina virtual, chamar código nativo em C ou C++. Para integrar as duas linguagens foi necessário: escrever o código java com a assinatura do método, criar o arquivo *header* em C a partir do código em java,

implementar o código em C e, por fim, criar uma biblioteca compartilhada.

A função *sendPacket()* recebe os dados SNMP prontos e, utilizando a biblioteca socket.h, cria um *raw socket*. Em cima do *raw socket* a função monta o UDP e o IP. O sistema operacional se encarrega de completar os dados da camada de enlace.

A montagem do datagrama UDP ocorre normalmente. Ambas as portas de destino e de origem são parâmetros recebidos no construtor do objeto. A porta de origem é gerada aleatoriamente, o que impede que regras baseadas em portas sejam utilizadas para essa ferramenta [26]. Após encapsular os dados SNMP, o tamanho do datagrama UDP é atualizado e seu *checksum* é calculado.

O *IP Spoofing* ocorre durante a montagem do IP. No campo IP de destino, em vez de se colocar o IP da máquina que vai enviar o pacote, se coloca o IP da vítima. Esse IP também é um parâmetro recebido no construtor do objeto. O restante do segmento é montado normalmente. Por fim, o pacote é enviado por um número determinado de vezes. Esse número é um parâmetro que é dependente da potência em que a ferramenta está sendo utilizada.

3.2.3.2 Rate Limiter

Para controlar o número de pacotes enviados por segundo, que é definido pela potência da ferramenta, utiliza-se o Guava [30]. O projeto Guava contém diversas bibliotecas desenvolvidas e utilizadas pelo Google em projetos java. O Rate Limiter é uma classe presente nesse projeto que auxilia no controle da frequência de eventos diversos [31].

Existem, portanto, dois parâmetros que controlam o número de pacotes enviados. Um é o parâmetro *packetsToSend* que a função *sendPacket()* recebe. Esse número controla um *loop* onde a ferramenta envia todos os pacotes o mais rápido que puder. O segundo é a configuração do Rate Limiter, que limita a chamada da função *sendPacket()*. Por exemplo, em sua configuração de potência 1, o Rate Limiter é configurado para permitir uma chamada da função *sendPacket()* por segundo. Além disso, a função *sendPacket()* recebe o valor 1 para *packetsToSend*. Desse modo, a cada um segundo um pacote é enviado. Na configuração de nível 2, o Rate Limiter permite 10 chamadas da função por segundo, e esta ainda recebe como 1 o valor de *packetsToSend*. A partir do nível 2, o Rate Limiter sempre permitirá 10 chamadas da função *sendPacket()*, mas o valor de *packetsToSend* será aumentado em uma ordem de grandeza. Logo,

no nível 3 serão executadas 10 chamadas a função *sendPacket()* por segundo e ela enviará 10 pacotes a cada execução o que significa 100 pacotes por segundo.

3.3 Considerações finais

Ao longo do capítulo 3 explicou-se o *design* e a implementação da ferramenta. De modo geral, a ferramenta será capaz de executar em todo sistema em que sua biblioteca nativa, parte em C do módulo Striker, for compilada. A interface gráfica construída é intuitiva e fácil de usar. Isso reflete como as ferramentas de negação de serviço não são exclusivas das pessoas com muitos conhecimentos técnicos. As duas principais funcionalidades da ferramenta são encontrar possíveis refletores e realizar o ataque em diferentes intensidades. Com essas duas funcionalidades é possível realizar um bom estudo das características da negação de serviço por reflexão amplificada, que é o tema do próximo capítulo.

CAPÍTULO 4

TESTES E RESULTADOS

Neste capítulo apresentam-se os dois testes elaborados para a ferramenta. O primeiro, de funcionalidade, visa verificar se os princípios do ataque estão sendo observados. O segundo, de performance, demonstra os limites da ferramenta e dos refletores a medida em que o ataque de negação de serviço se intensifica.

4.1 Configuração de testes

Os dois testes foram realizados em um ambiente controlado. Foi utilizado o analisador de protocolos Wireshark [32] para a captura e análise do tráfego gerado durante os testes. Durante os testes a ferramenta esteve sempre em sua configuração padrão.

Os testes de funcionalidade e performance utilizaram a disposição de máquinas mostrada na Figura 4.1.

Os equipamentos correspondentes são:

- IP: 192.168.0.1 - *Switch* - Roteador Wireless N 150 Mbps da Multilaser para o teste de funcionalidade e performance e *Switch* 48 portas Enterasys C-Series C5G124-48P2 1 Gbps para o teste de performance;
- IP: 192.168.0.101 - Computador de ataque - MAC OSX 10.10.1, 2.3GHz Intel Core i7, 16GB 1600MHz DDR3;
- IP: 192.168.0.100 - Refletor e amplificador - Windows 8.1 Single Language 64 bits, 1.6GHz Intel Core i5-4200, 4GB DDR2;
- IP: 192.168.0.102 - Vítima - Windows 7 Professional 64 Bits, 3.4GHz Intel Core i7-4770, 8GB DDR2.

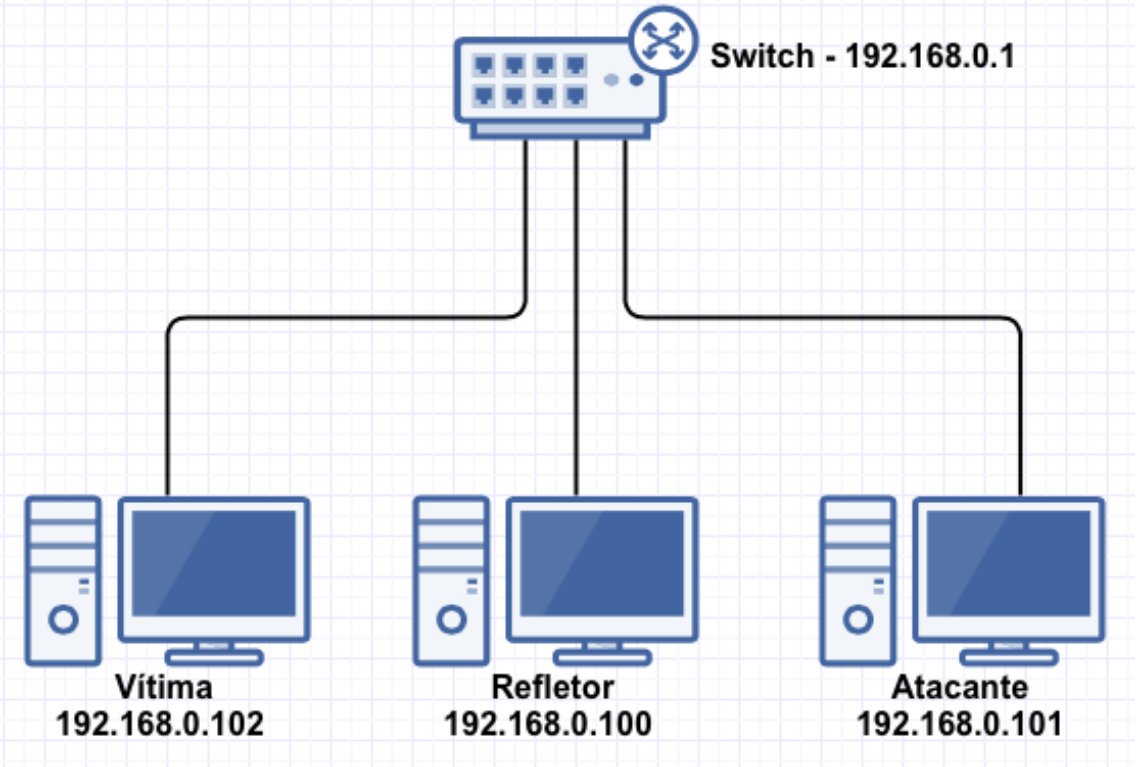

Figura 4.1. Disposição de computadores para o teste de funcionalidade.

4.2 Teste de funcionalidade

O teste de funcionalidade tem como objetivo descobrir dispositivos vulneráveis na rede e comprovar dois princípios do ataque, os quais são a reflexão e a amplificação. A reflexão diz respeito ao *IP spoofing*, ou seja, enviar um pacote de um dispositivo A para um dispositivo B com o IP de origem de um outro dispositivo que não seja A, por exemplo C. Desse modo, o dispositivo B irá responder para o dispositivo C. A amplificação é o fenômeno de se enviar um pacote com tamanho T e receber um pacote com tamanho T vezes F, onde F é o fator de amplificação com valor maior que 1.

Caso os dois princípios sejam observados, pode-se dizer que a ferramenta passou no teste e que o ataque de negação de serviço por SNMP ocorre com o seu uso.

4.2.1 Descoberta de dispositivos

O primeiro passo do teste é a varredura de rede procurando dispositivos vulneráveis. Esse processo é realizado pela funcionalidade de adicionar novos dispositivos da ferramenta. Para o teste, o computador de IP 192.168.0.100, que é o refletor, contêm um nó gerenciado SNMP

ativado e com a comunidade para leitura com valor de "*public*". A seguir é possível observar a captura de pacote nos três dispositivos durante essa etapa.

4.2.1.1 Captura - Computador de ataque

O computador de ataque é quem roda a ferramenta e realiza a varredura. A ferramenta é configurada com as opções padrões. O endereço IP base utilizado foi 192.168.0.1 e a sub-rede /24. A Figura 4.2 mostra a tela de adicionar dispositivos com essa configuração.

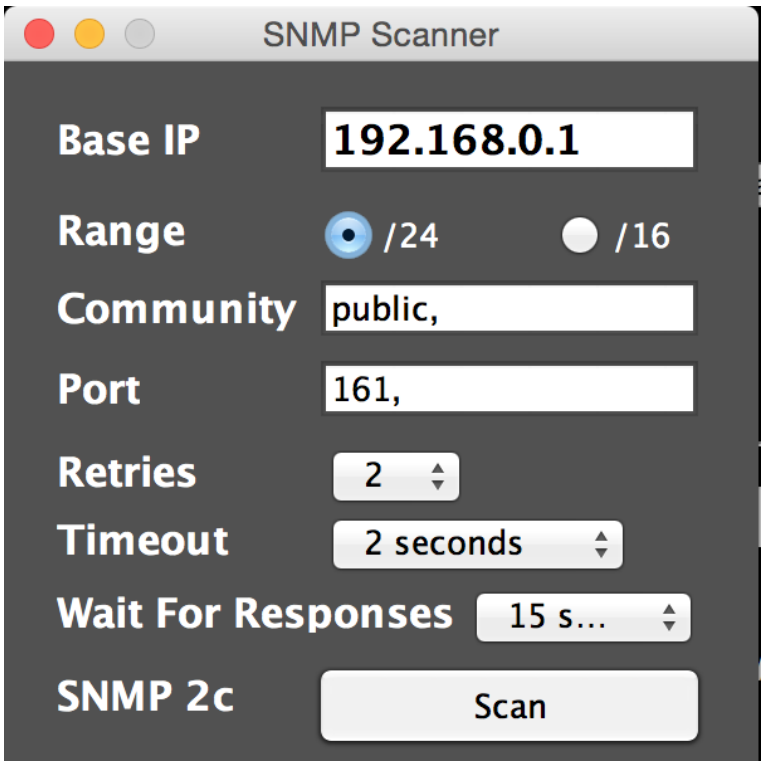

Figura 4.2. Tela para adicionar novos dispositivos vulneráveis através de varredura na rede.

Assim, é possível observar pela opção Retries que a ferramenta deve mandar mais duas mensagens caso não se tenha resposta no período de *timeout* de 2 segundos. O tempo total para se esperar por respostas é de 15 segundos, sendo metade para a primeira parte da varredura, utilizando GetNextRequest, e a outra metade para a segunda, com GetBulkRequest.

As Figuras 4.3 e 4.4 demonstram a captura no computador de ataque. Como esperado tem-se a mensagem GetNextRequest do IP 192.168.0.101, que é o computador de ataque, para todos os dispositivos da rede. Como o computador 192.168.0.100 é o refletor, e o único com SNMP ligado, ele produz a única resposta para o GetNextRequest. Seguindo a configuração, o computador de ataque envia mais 2 pacotes para os endereços IP que não responderam quando

o *timeout* de 2 segundos passa.

No.	Time	Source	Destination	Protocol	Length	Info
268	20:06:14.283987000	192.168.0.101	192.168.0.1	SNMP	85	get-next-request 1.3.6.1.2.1.1.4.0
369	20:06:14.295119000	192.168.0.101	192.168.0.102	SNMP	85	get-next-request 1.3.6.1.2.1.1.4.0
371	20:06:14.295209000	192.168.0.101	192.168.0.100	SNMP	85	get-next-request 1.3.6.1.2.1.1.4.0
385	20:06:14.296302000	192.168.0.100	192.168.0.101	SNMP	91	get-response 1.3.6.1.2.1.1.5.0
547	20:06:16.282633000	192.168.0.101	192.168.0.1	SNMP	85	get-next-request 1.3.6.1.2.1.1.4.0
647	20:06:16.301382000	192.168.0.101	192.168.0.102	SNMP	85	get-next-request 1.3.6.1.2.1.1.4.0
820	20:06:18.282940000	192.168.0.101	192.168.0.1	SNMP	85	get-next-request 1.3.6.1.2.1.1.4.0
921	20:06:18.303046000	192.168.0.101	192.168.0.102	SNMP	85	get-next-request 1.3.6.1.2.1.1.4.0

Figura 4.3. Captura no computador de ataque da primeira parte da varredura.

Passada a metade dos 15 segundos, a ferramenta parte para a segunda etapa da varredura que é o cálculo da amplificação em dispositivos encontrados. Como a única resposta foi do computador 192.168.0.100 só ele recebe as mensagens de GetBulkRequest. A Figura 4.4 mostra essa captura.

No.	Time	Source	Destination	Protocol	Length	Info
1097	20:06:21.326862000	192.168.0.101	192.168.0.100	SNMP	78	getBulkRequest 1.3
1098	20:06:21.327079000	192.168.0.101	192.168.0.100	SNMP	78	getBulkRequest 1.3
1099	20:06:21.327257000	192.168.0.101	192.168.0.100	SNMP	78	getBulkRequest 1.3
1100	20:06:21.327424000	192.168.0.101	192.168.0.100	SNMP	79	getBulkRequest 1.3
1101	20:06:21.327595000	192.168.0.101	192.168.0.100	SNMP	79	getBulkRequest 1.3
1102	20:06:21.327758000	192.168.0.101	192.168.0.100	SNMP	79	getBulkRequest 1.3
1103	20:06:21.362495000	192.168.0.100	192.168.0.101	SNMP	376	get-response 1.3.6.1.2.1.1.1.0
1104	20:06:21.362807000	192.168.0.100	192.168.0.101	SNMP	1157	get-response 1.3.6.1.2.1.1.1.0
1108	20:06:21.363613000	192.168.0.100	192.168.0.101	SNMP	122	get-response 1.3.6.1.2.1.1.1.0
1114	20:06:21.394959000	192.168.0.100	192.168.0.101	SNMP	847	get-response 1.3.6.1.2.1.1.1.0
1129	20:06:21.402752000	192.168.0.100	192.168.0.101	SNMP	171	get-response 1.3.6.1.2.1.1.1.0
1161	20:06:21.447520000	192.168.0.100	192.168.0.101	SNMP	1465	get-response 1.3.6.1.2.1.1.1.0

Figura 4.4. Captura no computador de ataque da segunda parte da varredura para cálculo de amplificação.

4.2.1.2 Captura - Computador refletor

Durante a varredura, o computador refletor recebe o GetNextRequest do computador de ataque e responde. Na segunda parte, chegam os GetBulkRequest. A Figura 4.5 mostra a captura correspondente.

No.	Time	Source	Destination	Protocol	Length	Info
214	20:06:30.767905000	192.168.0.101	192.168.0.100	SNMP	85	get-next-request 1.3.6.1.2.1.1.4.0
225	20:06:30.768734000	192.168.0.100	192.168.0.101	SNMP	91	get-response 1.3.6.1.2.1.1.5.0
872	20:06:37.834034000	192.168.0.101	192.168.0.100	SNMP	78	getBulkRequest 1.3
873	20:06:37.834034000	192.168.0.101	192.168.0.100	SNMP	78	getBulkRequest 1.3
874	20:06:37.834035000	192.168.0.101	192.168.0.100	SNMP	78	getBulkRequest 1.3
875	20:06:37.834035000	192.168.0.101	192.168.0.100	SNMP	79	getBulkRequest 1.3
876	20:06:37.834035000	192.168.0.101	192.168.0.100	SNMP	79	getBulkRequest 1.3
877	20:06:37.834036000	192.168.0.101	192.168.0.100	SNMP	79	getBulkRequest 1.3
878	20:06:37.834375000	192.168.0.100	192.168.0.101	SNMP	376	get-response 1.3.6.1.2.1.1.1.0 1.3.6.1.
879	20:06:37.834640000	192.168.0.100	192.168.0.101	SNMP	1157	get-response 1.3.6.1.2.1.1.1.0 1.3.6.1.
883	20:06:37.835125000	192.168.0.100	192.168.0.101	SNMP	122	get-response 1.3.6.1.2.1.1.1.0 1.3.6.1.
889	20:06:37.866055000	192.168.0.100	192.168.0.101	SNMP	847	get-response 1.3.6.1.2.1.1.1.0 1.3.6.1.
904	20:06:37.873010000	192.168.0.100	192.168.0.101	SNMP	171	get-response 1.3.6.1.2.1.1.1.0 1.3.6.1.
936	20:06:37.915647000	192.168.0.100	192.168.0.101	SNMP	1465	get-response 1.3.6.1.2.1.1.1.0 1.3.6.1.

Figura 4.5. Captura de varredura no computador refletor.

4.2.1.3 Captura - Computador vítima

Como o computador vítima faz parte da rede em que ocorreu a varredura, ele também recebeu mensagens GetNextRequest. Em uma situação normal, a vítima não estaria na mesma rede dos refletores e não receberia nenhum pacote SNMP antes das mensagens amplificadas. Entretanto, a sua captura é interessante nesse momento para mostrar a situação de computadores sem SNMP que estariam na rede sendo varrida. A Figura 4.6 mostra a captura, na qual é possível ver que como o computador não respondeu à primeira mensagem de GetNextRequest, ele recebeu mais duas que são os *retries* do computador de ataque. O intervalo entre cada mensagem é de dois segundos, que é a configuração padrão do *timeout* da ferramenta.

No.	Time	Source	Destination	Protocol	Length	Info
785	20:11:28.283287000	192.168.0.101	192.168.0.102	SNMP	85	get-next-request 1.3.6.1.2.1.1.4.0
1041	20:11:30.289363000	192.168.0.101	192.168.0.102	SNMP	85	get-next-request 1.3.6.1.2.1.1.4.0
1306	20:11:32.290938000	192.168.0.101	192.168.0.102	SNMP	85	get-next-request 1.3.6.1.2.1.1.4.0

Figura 4.6. Captura de varredura no computador vítima.

4.2.2 Ataque

Para testar o ataque foi utilizado a ferramenta em potência 1. O *target* foi o computador 192.168.0.102 e o refletor foi o computador 192.168.0.100. A Figura 4.7 mostra a tela principal da ferramenta antes de se iniciar o ataque.

Figura 4.7. Tela principal antes do início do ataque.

4.2.2.1 Captura - Computador de ataque

O computador 192.168.0.101 roda a ferramenta e realiza o ataque. Em sua captura é possível ver pacotes GetBulkRequest direcionados ao refletor e com o IP de origem da vítima. A Figura 4.8 mostra um trecho da captura. Observa-se que o intervalo entre cada mensagem enviada é de aproximadamente 1 segundo, correspondendo ao nível 1 da ferramenta.

No.	Time	Source	Destination	Protocol	Length	Info
4717	20:37:53.760345000	192.168.0.102	192.168.0.100	SNMP	79	getBulkRequest 1.3
4742	20:37:54.759562000	192.168.0.102	192.168.0.100	SNMP	79	getBulkRequest 1.3
4744	20:37:55.760061000	192.168.0.102	192.168.0.100	SNMP	79	getBulkRequest 1.3
4745	20:37:56.760322000	192.168.0.102	192.168.0.100	SNMP	79	getBulkRequest 1.3
4756	20:37:57.759773000	192.168.0.102	192.168.0.100	SNMP	79	getBulkRequest 1.3
4766	20:37:58.760338000	192.168.0.102	192.168.0.100	SNMP	79	getBulkRequest 1.3
4775	20:37:59.760497000	192.168.0.102	192.168.0.100	SNMP	79	getBulkRequest 1.3
4783	20:38:00.760701000	192.168.0.102	192.168.0.100	SNMP	79	getBulkRequest 1.3
4792	20:38:01.760479000	192.168.0.102	192.168.0.100	SNMP	79	getBulkRequest 1.3
4805	20:38:02.760400000	192.168.0.102	192.168.0.100	SNMP	79	getBulkRequest 1.3
4828	20:38:03.759797000	192.168.0.102	192.168.0.100	SNMP	79	getBulkRequest 1.3
4831	20:38:04.760653000	192.168.0.102	192.168.0.100	SNMP	79	getBulkRequest 1.3
4839	20:38:05.760866000	192.168.0.102	192.168.0.100	SNMP	79	getBulkRequest 1.3
4842	20:38:06.760161000	192.168.0.102	192.168.0.100	SNMP	79	getBulkRequest 1.3
4864	20:38:07.761392000	192.168.0.102	192.168.0.100	SNMP	79	getBulkRequest 1.3

Figura 4.8. Captura nível 1 no computador de ataque.

Como esperado, não foi observado mais nenhum tipo de pacote SNMP. Todos os pacotes estão sendo enviados ao computador refletor, 192.168.0.100, que está enviando as respostas ao computador 192.168.0.102, a vítima, como mostrado na Figura 4.9.

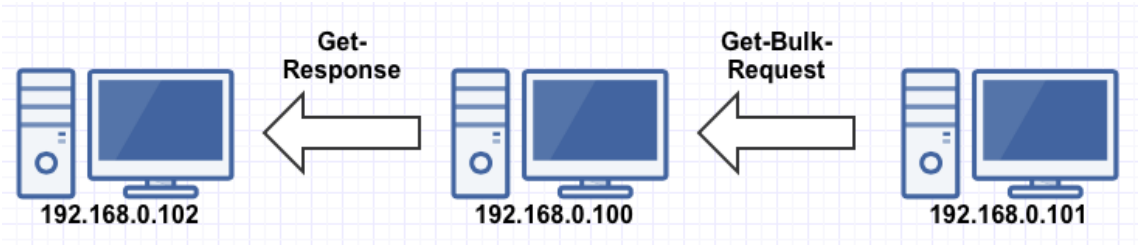

Figura 4.9. Mensagens trocadas entre os dispositivos durante o ataque.

4.2.2.2 Captura - Computador refletor

Na captura do computador refletor 192.168.0.100, Figura 4.10, é possível verificar tanto os GetBulkRequest, que aparentemente chegam da vítima, quanto os GetResponse.

No.	Time	Source	Destination	Protocol	Length	Info
1587	20:38:12.869511	192.168.0.102	192.168.0.100	SNMP	79	getBulkRequest 1.3
1620	20:38:12.981198	192.168.0.100	192.168.0.102	SNMP	360	get-response 1.3.6.1.2.1.1.1.0 1.3.6.1.2.1.1.2
1623	20:38:13.869302	192.168.0.102	192.168.0.100	SNMP	79	getBulkRequest 1.3
1656	20:38:13.895504	192.168.0.100	192.168.0.102	SNMP	360	get-response 1.3.6.1.2.1.1.1.0 1.3.6.1.2.1.1.2
1659	20:38:14.868737	192.168.0.102	192.168.0.100	SNMP	79	getBulkRequest 1.3
1692	20:38:14.911171	192.168.0.100	192.168.0.102	SNMP	360	get-response 1.3.6.1.2.1.1.1.0 1.3.6.1.2.1.1.2
1693	20:38:15.883064	192.168.0.102	192.168.0.100	SNMP	79	getBulkRequest 1.3
1726	20:38:15.913114	192.168.0.100	192.168.0.102	SNMP	360	get-response 1.3.6.1.2.1.1.1.0 1.3.6.1.2.1.1.2
1728	20:38:16.869686	192.168.0.102	192.168.0.100	SNMP	79	getBulkRequest 1.3
1761	20:38:16.911679	192.168.0.100	192.168.0.102	SNMP	360	get-response 1.3.6.1.2.1.1.1.0 1.3.6.1.2.1.1.2
1762	20:38:17.869011	192.168.0.102	192.168.0.100	SNMP	79	getBulkRequest 1.3
1795	20:38:17.912087	192.168.0.100	192.168.0.102	SNMP	360	get-response 1.3.6.1.2.1.1.1.0 1.3.6.1.2.1.1.2
1801	20:38:18.870222	192.168.0.102	192.168.0.100	SNMP	79	getBulkRequest 1.3
1834	20:38:18.929178	192.168.0.100	192.168.0.102	SNMP	360	get-response 1.3.6.1.2.1.1.1.0 1.3.6.1.2.1.1.2

Figura 4.10. Captura nível 1 no computador refletor.

4.2.2.3 Captura - Vítima

Por fim, na captura do computador 192.168.0.102, é possível ver as mensagens de GetResponse vindas do refletor. Vale observar que não existe mensagem de GetBulkRequest, pois estão sendo geradas pelo computador de ataque, mas apenas as respostas. A captura é apresentada na Figura 4.11.

No.	Time	Source	Destination	Protocol	Length	Info
2620	20:42:59.650191	192.168.0.100	192.168.0.102	SNMP	360	get-response 1.3.6.1.2.1.1.1.0 1.3.6.1.2.1.
2655	20:43:00.652107	192.168.0.100	192.168.0.102	SNMP	360	get-response 1.3.6.1.2.1.1.1.0 1.3.6.1.2.1.
2697	20:43:01.649301	192.168.0.100	192.168.0.102	SNMP	360	get-response 1.3.6.1.2.1.1.1.0 1.3.6.1.2.1.
2738	20:43:02.682117	192.168.0.100	192.168.0.102	SNMP	360	get-response 1.3.6.1.2.1.1.1.0 1.3.6.1.2.1.
2776	20:43:03.679363	192.168.0.100	192.168.0.102	SNMP	360	get-response 1.3.6.1.2.1.1.1.0 1.3.6.1.2.1.
2815	20:43:04.732305	192.168.0.100	192.168.0.102	SNMP	360	get-response 1.3.6.1.2.1.1.1.0 1.3.6.1.2.1.
2854	20:43:05.646515	192.168.0.100	192.168.0.102	SNMP	360	get-response 1.3.6.1.2.1.1.1.0 1.3.6.1.2.1.
2889	20:43:06.661607	192.168.0.100	192.168.0.102	SNMP	360	get-response 1.3.6.1.2.1.1.1.0 1.3.6.1.2.1.
2924	20:43:07.664062	192.168.0.100	192.168.0.102	SNMP	360	get-response 1.3.6.1.2.1.1.1.0 1.3.6.1.2.1.
2958	20:43:08.662219	192.168.0.100	192.168.0.102	SNMP	360	get-response 1.3.6.1.2.1.1.1.0 1.3.6.1.2.1.
2993	20:43:09.662898	192.168.0.100	192.168.0.102	SNMP	360	get-response 1.3.6.1.2.1.1.1.0 1.3.6.1.2.1.
3037	20:43:10.679802	192.168.0.100	192.168.0.102	SNMP	360	get-response 1.3.6.1.2.1.1.1.0 1.3.6.1.2.1.
3076	20:43:11.647227	192.168.0.100	192.168.0.102	SNMP	360	get-response 1.3.6.1.2.1.1.1.0 1.3.6.1.2.1.

Figura 4.11. Captura nível 1 na vítima.

4.2.3 Análise do teste de funcionalidade

4.2.3.1 Procura de dispositivos vulneráveis

Como demonstrado pelos resultados, a descoberta de dispositivos vulneráveis, para serem utilizados como refletores, foi bem sucedida. As Figuras 4.3, 4.4, 4.5 e 4.6 mostram que as mensagens foram enviadas para os destinatários corretos nos tempos certos.

4.2.3.2 Princípio da reflexão

A reflexão é comprovada pelas capturas mostradas anteriormente onde é possível ver que o computador de ataque gera mensagens GetBulkRequest com o endereço IP de origem da vítima. Além disso, pelo fato de todos os dispositivos estarem na mesma sub rede, verifica-se

a reflexão pelo endereço MAC nas mensagens.

A Figura 4.12 mostra um GetBulkRequest, gerado pelo computador de ataque, que foi recebido no refletor. Logo em seguida é gerada a resposta, um GetResponse, mostrado na Figura 4.13. Dessa forma, é possível observar que o IP de origem do GetBulkRequest supostamente é o computador 192.168.0.102, que é a vítima, e que o IP de destino da resposta também é esse endereço. Entretanto, o endereço MAC de origem do GetBulkRequest e de destino do GetResponse são diferentes. Isso ocorre, pois quem de fato enviou a primeira mensagem foi o computador de ataque 192.168.0.101 e não a vítima. Logo, o endereço MAC de origem do GetBulkRequest é do computador de ataque e não da vítima. Os valores de endereço MAC foram parcialmente ocultados.

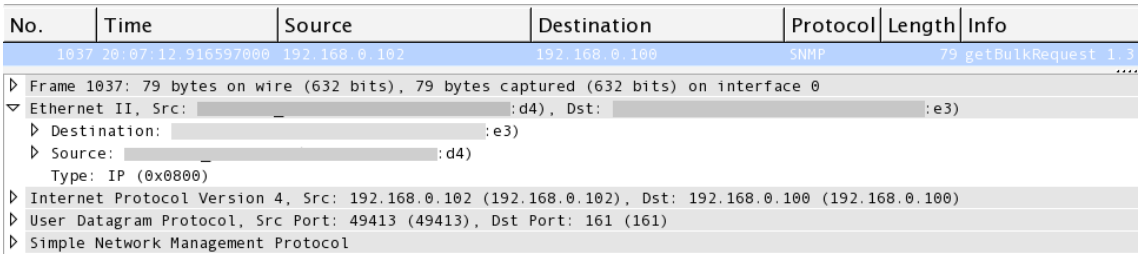

Figura 4.12. GetBulkRequest no refletor.

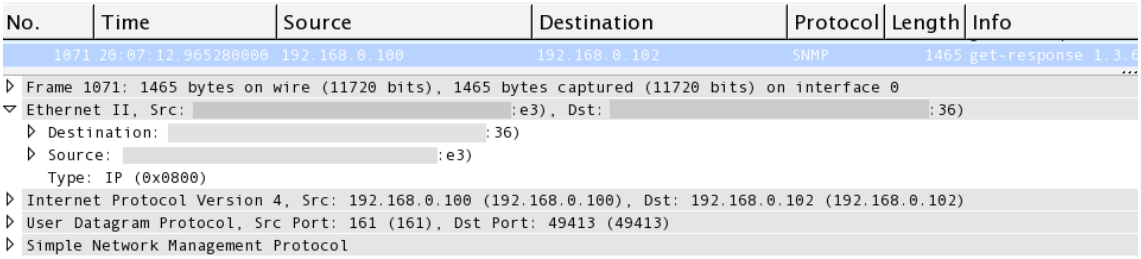

Figura 4.13. GetResponse no refletor.

4.2.3.3 Princípio da amplificação

Como pode ser observado pela captura mostrada na Figura 4.8, o tamanho do pacote SNMP enviado é de 79 bytes. Desses 79 bytes:

- 14 são cabeçalho Ethernet;
- 20 são cabeçalho IP;
- 8 são cabeçalho UDP;
- 37 são PDU SNMP.

A Figura 4.4 demonstra as respostas enviadas pelo refletor durante a medição de sua amplificação. A maior mensagem é a última, que é a resposta do GetBulkRequest com 2000 de MaxRepetition. Nessa mensagem, que é a junção de vário fragmentos IP, tem-se 47.374 bytes onde:

- 14 são cabeçalho Ethernet;
- 20 são cabeçalho IP;
- 8 são cabeçalho UDP;
- 47332 são PDU SNMP.

Pela Equação (4.1) tem-se uma amplificação de:

$$\begin{aligned} A &= \frac{RespIpSize + RespUdpSize + RespSNMPSize}{ReqIpSize + ReqUdpSize + ReqSNMPSize} \\ &= \frac{20 + 8 + 47332}{20 + 8 + 37} \\ &= 728.615 \end{aligned} \tag{4.1}$$

Esse valor condiz com o mostrado pela ferramenta antes do ataque na Figura 4.7, lembrando que os 14 bytes do Ethernet são desconsiderados. As Figuras 4.4 e 4.8, juntamente com a Equação (4.1), demonstram a amplificação do refletor utilizado e confirmam o princípio da amplificação esperado. A amplificação de 728 vezes é muito alta. Ela significa que para cada bit gerado pelo atacante o refletor irá gerar 98 bytes para a vítima. Assim, é possível concluir que a ferramenta obteve todos os resultados esperados durante o teste de funcionalidade e, de fato, observa-se o ataque de reflexão e amplificação pelo SNMP.

4.3 Teste de performance

O teste de performance foi elaborado com a mesma configuração do teste de funcionalidade. Entretanto, ele foi executado duas vezes. A primeira, a partir de agora denominada teste 1, com o Roteador Wireless N da Multilaser e outra, teste 2, com o *Switch* 48 portas da Enterasys. Essa comparação foi feita para verificar uma possível saturação no *switch* da Multilaser devido sua modesta capacidade. Durante o teste, a potência da ferramenta foi aumentada gradualmente passando por todos os níveis. Desse modo, esperava-se observar os limites de envio de pacotes

da ferramenta, do *switch* e do refletor além do limite de amplificação do refletor. Vale ressaltar que o teste foi elaborado com apenas um computador como refletor, diferentemente de uma situação real, onde muitos refletores seriam utilizados.

Uma tabela com o nível do ataque, quantos pacotes foram recebidos e enviados e quantos bytes foram recebidos e enviados foi elaborada para cada teste e para cada dispositivo do esquema mostrado na Figura 4.1. A duração medida foi de 30 segundos para cada nível, contado a partir do primeiro pacote recebido ou enviado.

4.3.1 Computador de ataque

A Tabela 4.1 demonstra os dados para o computador de ataque durante o teste 1. Esse computador não recebe nenhum pacote. Ele apenas envia os GetBulkRequest para o refletor.

Tabela 4.1. Tabela com dados da captura no computador de ataque durante 30 segundos no teste 1.

Nível	**Pacotes Recebidos**	**Bytes Recebidos**	**Pacotes Enviados**	**Bytes Enviados**
1	0	0	31	2449
2	0	0	310	24490
3	0	0	3091	244189
4	0	0	30630	2419770
5	0	0	309518	24451922
6	0	0	3099925	244894075
7	0	0	8816056	696468424
8	0	0	9423659	744469061
9	0	0	9144787	722438173
10	0	0	9034007	713686553

Por outro lado, a Tabela 4.2 contêm os dados durante o teste 2.

Tabela 4.2. Tabela com dados da captura no computador de ataque durante 30 segundos no teste 2.

Nível	Pacotes Recebidos	Bytes Recebidos	Pacotes Enviados	Bytes Enviados
1	0	0	30	2370
2	0	0	300	23700
3	0	0	3091	244189
4	0	0	30943	2444497
5	0	0	309942	24485418
6	0	0	3100000	244900000
7	0	0	9005199	711410721
8	0	0	9179288	725163752
9	0	0	9077176	717096904
10	0	0	9203830	727102570

4.3.2 Computador refletor

As Tabelas 4.3 e 4.4, mostram os números do computador refletor durante os teste 1 e 2, respectivamente. Ao contrário do computador de ataque e da vítima, o refletor possui pacotes chegando e saindo. Nota-se que o número de pacotes enviados é muito maior que o número de pacotes recebido devido a fragmentação dos pacotes amplificados.

Tabela 4.3. Tabela com dados da captura no computador refletor durante 30 segundos no teste 1.

Nível	Pacotes Recebidos	Bytes Recebidos	Pacotes Enviados	Bytes Enviados
1	31	2449	992	1500386
2	311	24569	9952	15063725
3	3091	244189	25920	39237633
4	30588	2416452	26169	38467637
5	265061	20939819	26103	38380533
6	969923	76623917	23552	35639814
7	1833870	144875730	20992	31749744
8	1826179	144268141	16401	24150593
9	2206824	174339096	15048	22148981
10	1875600	148172400	15630	22995078

Tabela 4.4. Tabela com dados da captura no computador refletor durante 30 segundos no teste 2.

Nível	Pacotes Recebidos	Bytes Recebidos	Pacotes Enviados	Bytes Enviados
1	30	2370	990	1494072
2	300	23700	9933	14982644
3	3091	244189	26268	39273843
4	30875	2439125	27324	41339398
5	266695	21068905	25179	37624293
6	1326824	104819096	21516	32268784
7	2106893	166444547	20460	30787340
8	2106898	166444942	20559	31005579
9	1763377	139306783	20460	30858640
10	2115243	167104197	16433	24521703

4.3.3 Computador vítima

As Tabelas 4.5 e 4.6 contêm os dados do computador vítima. A vítima apenas recebe os pacotes amplificados do refletor.

Tabela 4.5. Tabela com dados da captura na vítima durante 30 segundos no teste 1.

Nível	Pacotes Recebidos	Bytes Recebidos	Pacotes Enviados	Bytes Enviados
1	992	1500386	0	0
2	9952	15063725	0	0
3	25920	39237633	0	0
4	26169	38467637	0	0
5	26200	38525854	0	0
6	23552	35639814	0	0
7	20979	31730111	0	0
8	16389	24133793	0	0
9	15048	22148981	0	0
10	16071	23644370	0	0

Tabela 4.6. Tabela com dados da captura na vítima durante 30 segundos no teste 2.

Nível	Pacotes Recebidos	Bytes Recebidos	Pacotes Enviados	Bytes Enviados
1	990	1494072	0	0
2	9933	14982644	0	0
3	25821	38605863	0	0
4	27324	41339398	0	0
5	25278	37772226	0	0
6	21516	32268784	0	0
7	20460	30787340	0	0
8	20559	31005579	0	0
9	20460	30858640	0	0
10	16533	24671671	0	0

4.3.4 Análise do teste de performance

A análise do teste de performance irá comparar os resultados obtidos nos dois testes e será dividia da seguinte forma:

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

4.3.4.1 Análise da performance do computador de ataque

A performance do computador de ataque é facilmente medida pelo gráfico da Figura 4.14, onde tem-se a relação de megabit por segundo e pacotes por segundo enviados pela ferramenta durante os dez níveis do ataque. Pelo gráfico de barras, que representa os bits enviados, pode-se ver que a ferramenta satura sua capacidade de envio perto dos 200 Mbps. Os pacotes por segundo, representados pelas linhas, saturam em 300 mil pacotes a cada segundo. Não houve diferenças significativas entre os teste 1 e 2, em ambos pacotes e bits por segundo.

A taxa de envio do computador de ataque é limitada pela interface de saída. No caso desse computador, essa interface possui capacidade de até 100 Mbps. Pelo gráfico da Figura 4.14, encontramos valores perto dos 200 Mbps. Isso ocorre porque a ferramenta de análise de protocolo, Wireshark, captura os pacotes enviados entre o sistema operacional e o hardware de

rede. Logo, a ferramenta de ataque está tentando enviar perto dos 200 Mbps, mas existe uma saturação do hardware de, no melhor caso, 100 Mbps e, também, uma possível saturação do barramento.

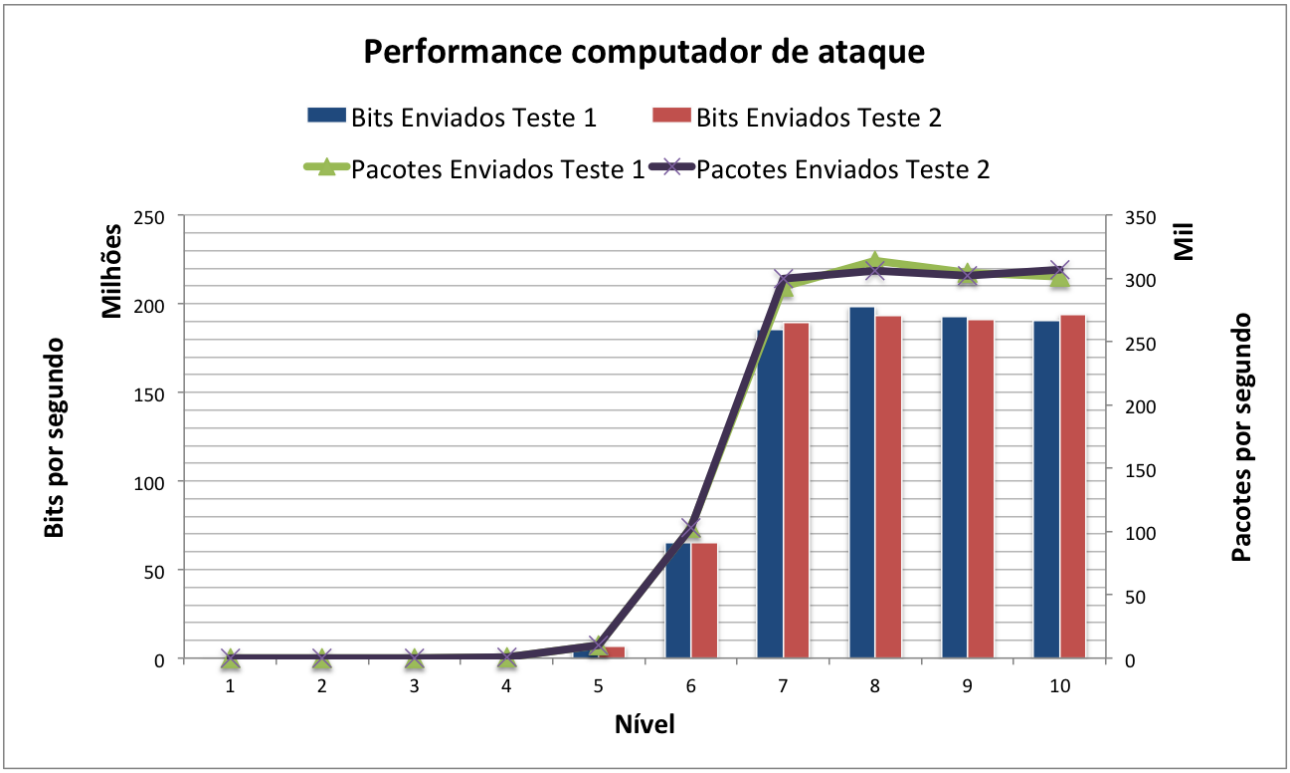

Figura 4.14. Bits e pacotes por segundo enviados pelo atacante.

4.3.4.2 Análise da performance do *switch*

O *switch* foi o equipamento responsável por transferir os dados entre os três computadores do teste. A performance do *switch* foi analisada em duas etapas. Primeiramente, foi comparada a quantidade de bits enviado pelo computador de ataque que chegou no refletor e depois se comparou a quantidade de bits enviadas pelo refletor que chegou até a vítima.

Os gráficos das Figuras 4.15 e 4.16 representam a primeira parte da análise. Neles é possível verificar que, a partir do nível 6, devido a grande quantidade de dados, a diferença entre quantos bits o computador de ataque envia e quantos bits o computador refletor recebe é muito grande. Claramente está ocorrendo uma saturação. Entretanto, a saturação pode ser do computador de ataque, que não está conseguindo enviar essa quantidade de bits, ou do *switch*, que não consegue encaminhar todo esse tráfego para o refletor.

A quantidade de bits recebida pelo refletor nos dois testes é muito semelhante. Entre

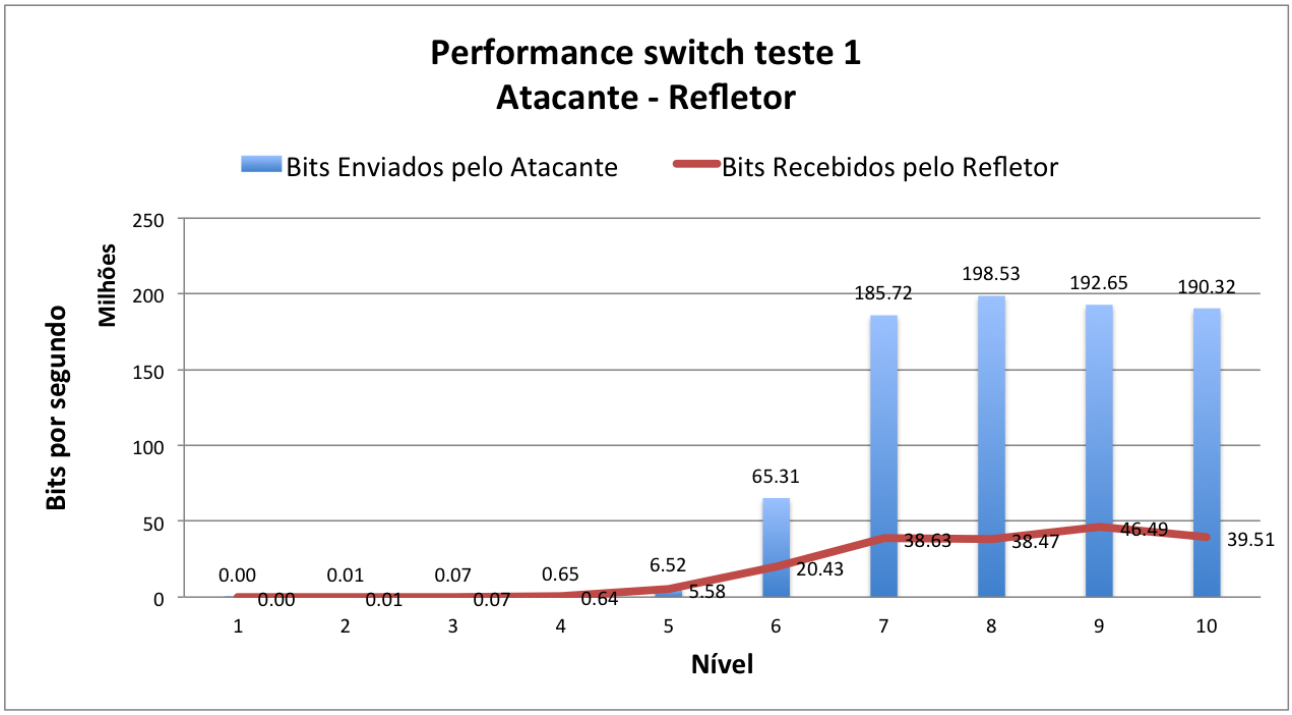

Figura 4.15. Quantidade de bits enviados pelo computador de ataque e recebidos pelo refletor durante o teste 1.

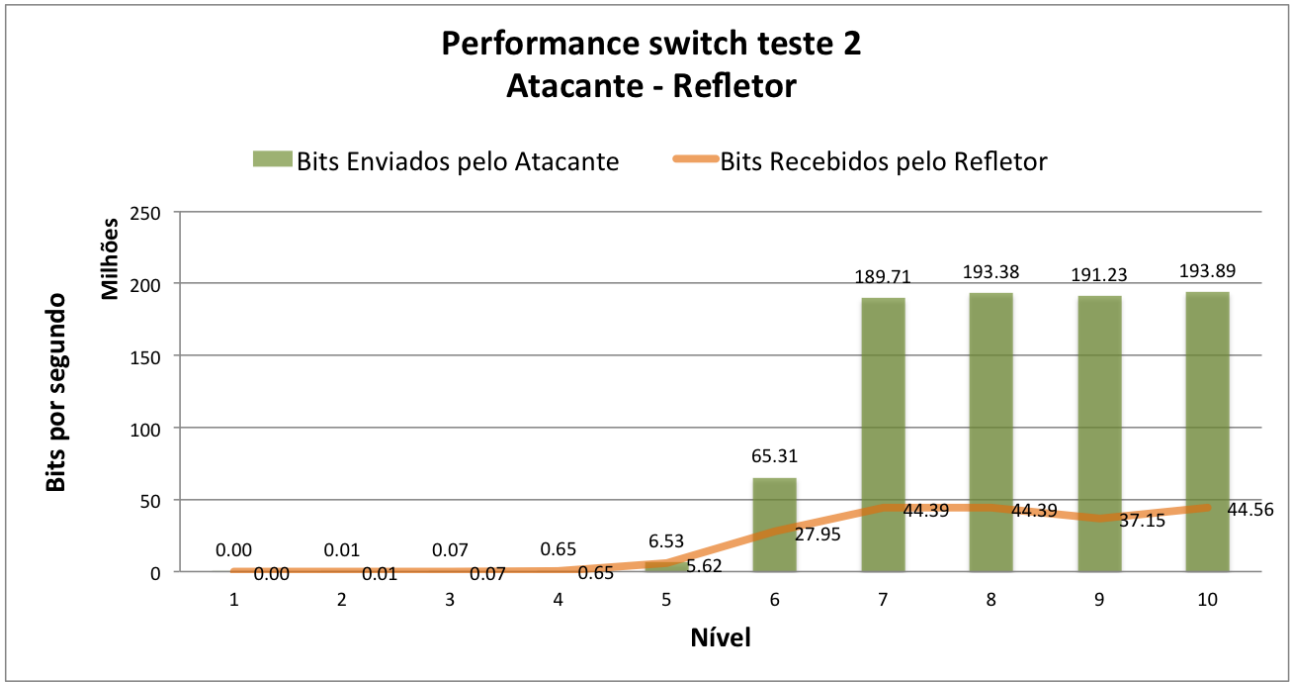

Figura 4.16. Quantidade de bits enviados pelo computador de ataque e recebidos pelo refletor durante o teste 2.

os níveis 7 e 10, onde ocorreu maior diferença, o *switch* do teste 1 entregou uma média de 40,78 Mbps e o *switch* do teste 2, uma média de 42,62 Mbps. Essa diferença, de 4,51%, é muito pequena, levando em consideração a diferença de capacidade dos dois *switches*. Por esse motivo, espera-se uma saturação maior, se não for total, no computador de ataque do que nos

switches.

Para a segunda parte da análise, tem-se os gráficos das Figuras 4.17 e 4.18, que demonstram a quantidade de bits enviada pelo refletor e recebida pela vítima. Pode-se observar que são valores praticamente iguais. Isso demonstra que não houve efeito de saturação dos *switches* na passagem de bits do refletor para a vítima.

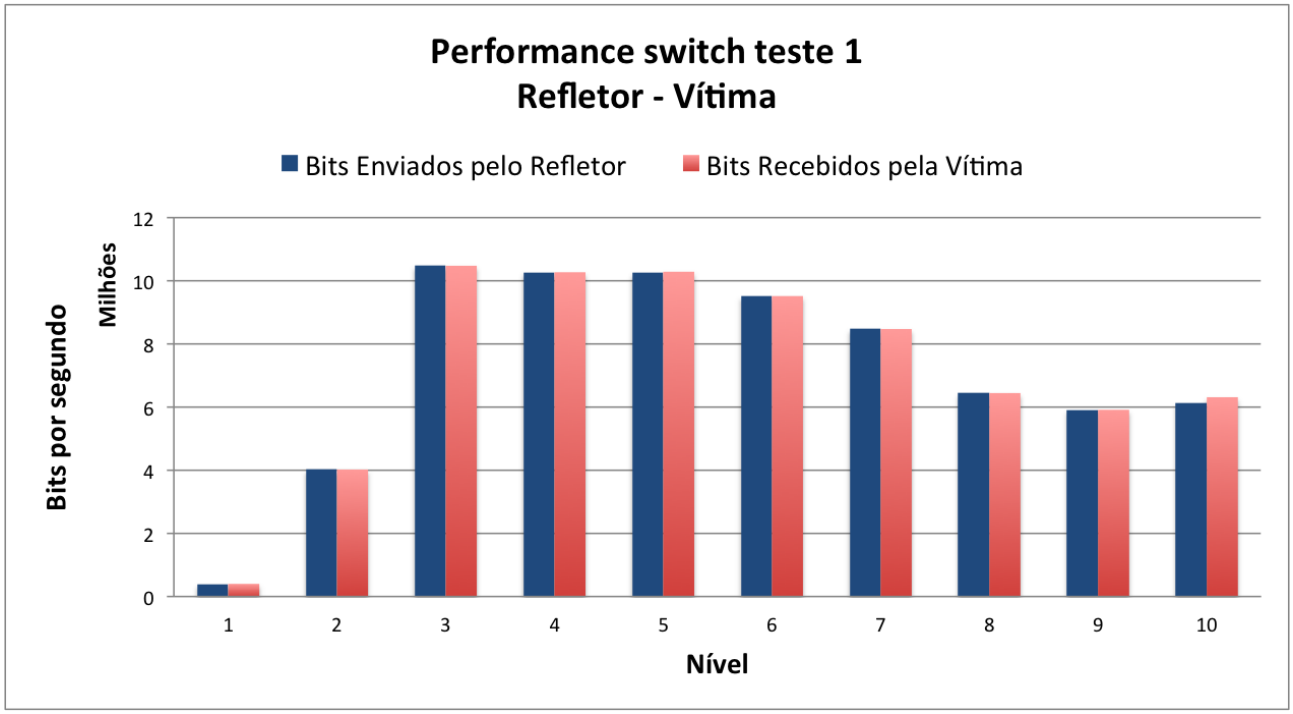

Figura 4.17. Quantidade de bits enviados pelo refletor e recebidos pela vítima durante o teste 1.

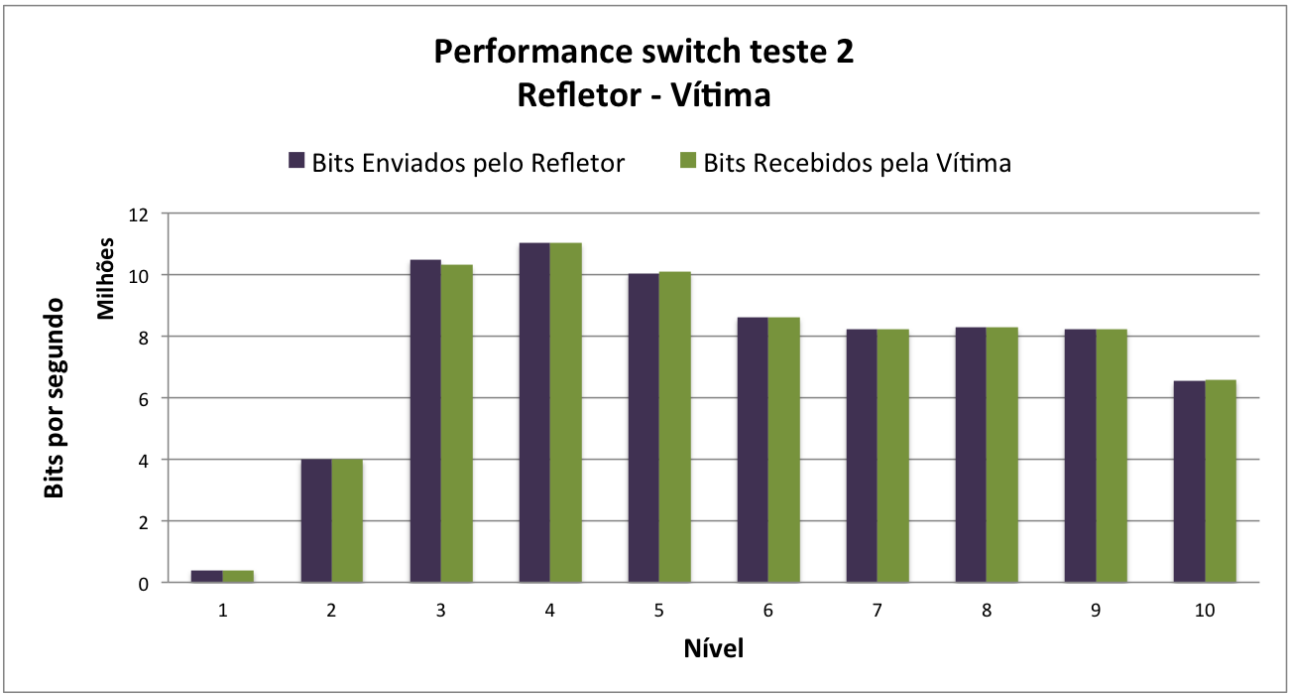

Figura 4.18. Quantidade de bits enviados pelo refletor e recebidos pela vítima durante o teste 2.

4.3.4.3 Análise da performance do refletor

O refletor é o dispositivo que recebe o tráfego gerado pelo atacante, amplificando-o e enviando para a vítima. O gráfico da Figura 4.19 mostra a performance do refletor durante os dois testes. Nesse gráfico tem-se a comparação da quantidade de bits recebidos pelo refletor, barra azul para o teste 1 e linha verde para o teste 2, com a quantidade de bits enviadas pelo refletor, barra vermelha para o teste 1 e linha roxa para o teste 2. Observa-se que até o nível 5, a quantidade de bits enviada pelo refletor era maior do que a recebida, caracterizando uma amplificação. A partir do nível 6, o refletor passou a receber mais bits do que envia.

A Tabela 4.7 mostra a amplificação realizada pelo refletor nos testes 1 e 2 para bits e pacotes. Como já visto no gráfico da Figura 4.19, a partir do nível 6 essa amplificação fica abaixo de 1, indicando que o refletor está recebendo mais pacotes do que enviando. O motivo desse fenômeno é que, embora o refletor tenha uma amplificação maior que 1, ele tem um limite para a quantidade de pacotes que consegue amplificar. Isso ocorre porque o processamento do GetBulkRequest não é instantâneo e o refletor perde um determinado tempo coletando todas as informações pedidas para então enviar o GetResponse. Assim, mesmo o atacante enviando mais bits o refletor não consegue acompanhar e sua amplificação satura. Essa é uma informação importante pois mostra que a amplificação não é o único parâmetro referente sobre os refletores.

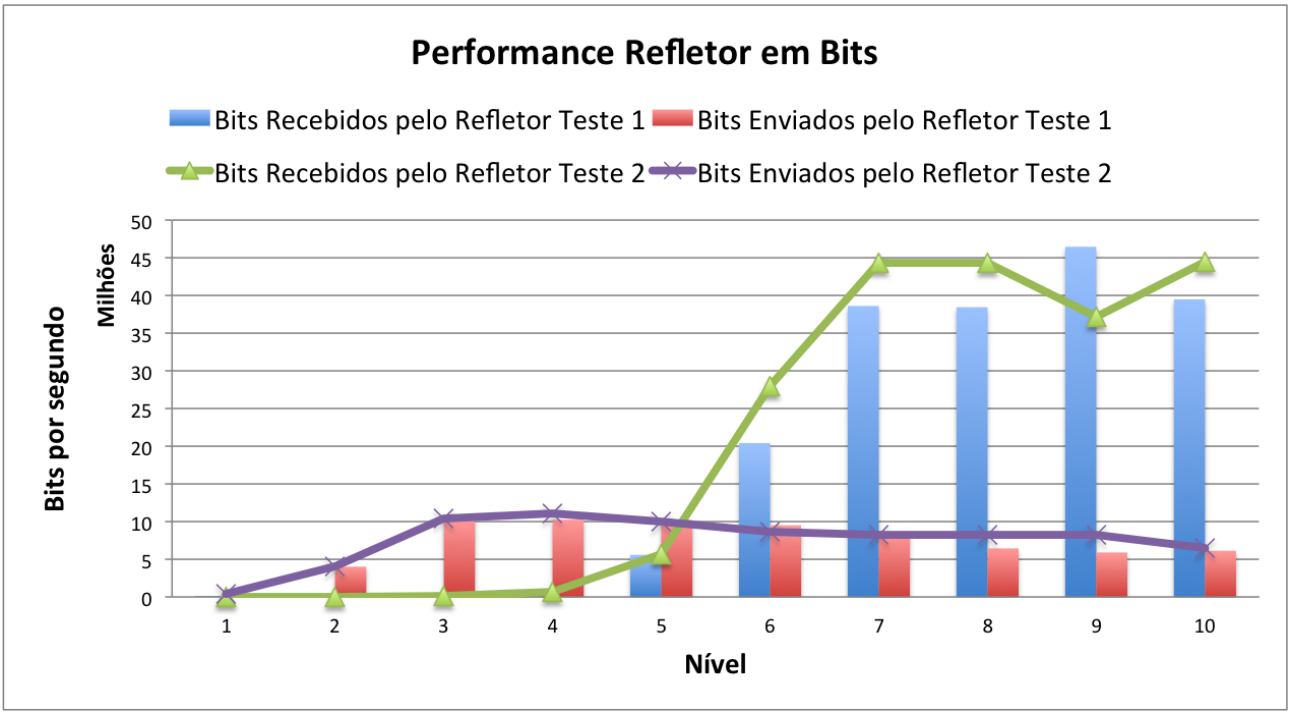

Figura 4.19. Quantidade de bits recebidos e enviados pelo refletor durante os testes 1 e 2.

Assim, deve se considerar também a capacidade de cada refletor em coletar as informações da MIB e responder.

Tabela 4.7. Tabela com a amplificação do refletor em bits e pacotes durante os teste 1 e 2.

Nível	Amplificação Teste 1 Bits	Amplificação Teste 1 Pacotes	Amplificação Teste 2 Bits	Amplificação Teste 2 Pacotes
1	612.65	32.00	630.41	33.00
2	613.12	32.00	632.18	33.11
3	160.69	8.39	160.83	8.50
4	15.92	0.86	16.95	0.88
5	1.83	0.10	1.79	0.09
6	0.47	0.02	0.31	0.02
7	0.22	0.01	0.18	0.01
8	0.17	0.01	0.19	0.01
9	0.13	0.01	0.22	0.01
10	0.16	0.01	0.15	0.01

Outra importante observação é que enquanto o refletor não satura, níveis 1 e 2, a amplificação não é somente de bits mas também de pacotes. Isso ocorre pois enquanto o GetBulkRequest ocupa apenas um pacote o GetResponse é dividido em diversos pacotes TCP. Ou seja, o ataque além de causar uma amplificação em número de bits também causa uma amplificação em número de pacotes devido a fragmentação.

O dispositivo utilizado como refletor foi um computador pessoal. Esse é um dispositivo atípico para uso do SNMP. Ele possui um hardware mais potente do que dispositivos típicos, como roteadores e *switches*. A implementação do protocolo também é menos otimizada em computadores pessoais. Dessa forma, acredita-se que a saturação, devido à busca de informações da MIB no refletor, possa demorar mais a ocorrer em dispositivos otimizados para o uso do SNMP.

4.3.4.4 Análise dos dados na vítima

O computador vítima apenas recebe as mensagens amplificadas do refletor. O total de bits recebido em cada nível do ataque pode ser visto nos gráficos das Figuras 4.17 e 4.18, apresentados na performance do *switch*. Esses gráficos demonstram que a vítima recebe a mesma quantidade de bits que o refletor gera e que não existe nenhum efeito de saturação.

4.4 Considerações finais

Com os testes de funcionalidade e performance, percebe-se que o ataque utilizando o SNMP possui as características de um ataque de amplificação e reflexão e pode alcançar altos valores de amplificação. Com o refletor utilizado, obteve-se uma amplificação de 728 vezes, desconsiderando os 14 bytes do Ethernet. Entretanto, mostrou-se que a amplificação não é sustentável devido à saturação dos refletores ao compilar todos os dados da MIB, pedidos no GetBulkRequest.

O gráfico da Figura 4.20 apresenta os valores de amplificação alcançados, nos dois testes, para bits e para pacotes. Esse gráfico foi construído com os valores da Tabela 4.7 e mostra como a amplificação não se sustenta a medida que os níveis vão aumentando.

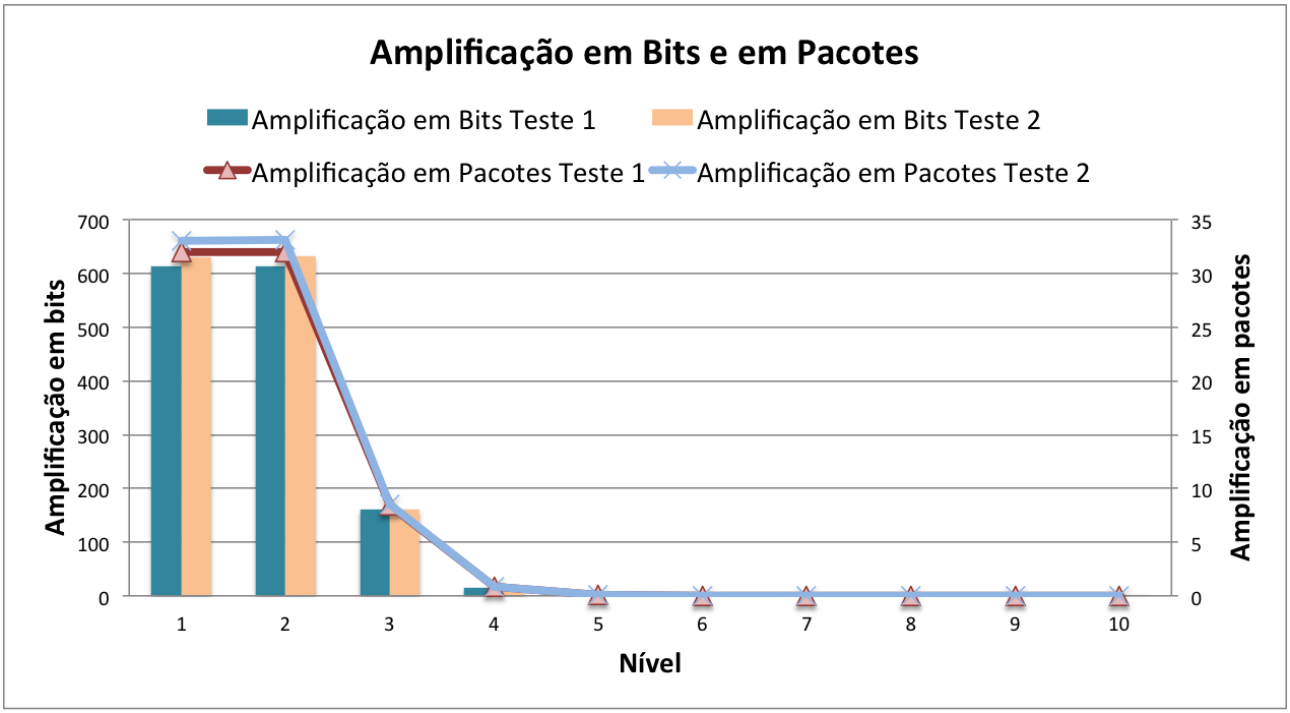

Figura 4.20. Valor da amplificação para bits e pacotes.

CAPÍTULO 5

CONCLUSÃO

Esta monografia apresentou a viabilidade e as principais características para ataques de negação de serviço por reflexão e amplificação utilizando o protocolo *Simple Network Management Protocol* (SNMP), que é um protocolo para efetuar o gerenciamento de dispositivos em uma rede.

Primeiramente, foram apresentados os principais ataques de negação de serviço e o protocolo de gerenciamento SNMP. Pelo estudo do protocolo, observou-se uma mensagem que pode gerar uma resposta muitas vezes maior. Além disso, um tipo específico da segunda versão utiliza uma segurança baseada em strings de comunidade, que é considerada fraca. A partir dessas informações, criou-se a hipótese de que o protocolo poderia ser mais um candidato à realização de ataques de negação de serviço do tipo reflexão e amplificação.

Em seguida, foi desenvolvida uma ferramenta que pudesse realizar o ataque para comprovar sua viabilidade. A ferramenta desenvolvida foi composta por três módulos: uma interface com o usuário, um módulo Scanner e um módulo Striker. A interface com o usuário foi implementada para prover acesso às funcionalidades da ferramenta além de aumentar a usabilidade. O módulo Scanner é utilizado para encontrar dispositivos vulneráveis que servem como refletores no ataque e para calcular qual a maior amplificação possível nesses dispositivos. O módulo Striker é o responsável por realizar o ataque, ele recebe os parâmetros calculados pelo módulo Scanner e, quando indicado pelo usuário, envia as mensagens para os refletores que amplificam e enviam para a vítima.

Com a ferramenta desenvolvida foi realizado um teste de funcionalidade e um teste de performance em um ambiente controlado para verificar as propriedades do ataque. Os testes utilizaram três computadores conectados por um equipamento que serviu como *switch*. Um computador agiu como atacante e utilizou a ferramenta, outro computador foi configurado com um nó de gerenciamento SNMP para servir como refletor e amplificador do ataque e, por

fim, um outro computador serviu como vítima. Dois tipos diferentes de equipamentos foram utilizados para a função de *switch*.

O teste de funcionalidade teve como objetivo verificar se dispositivos vulneráveis poderiam ser encontrados pela ferramenta e comprovar os princípios de amplificação e reflexão. A ferramenta obteve sucesso na descoberta de dispositivos. A amplificação pode ser verificada pela relação entre o GetBulkRequest gerado pela ferramenta e o GetResponse gerado pelo refletor. A reflexão foi observada pelas capturas que mostraram mensagens sendo geradas no computador de ataque com o IP de origem da vítima. Além disso, pode-se verificar a reflexão pelos endereços MAC utilizados.

Para o teste de performance foi executado o ataque durante 30 segundos para cada nível de potência da ferramenta. Os pacotes enviados e recebidos foram capturados e expostos em tabelas. A partir da análise verificou-se que:

- O limite de envio da ferramenta foi de aproximadamente 200 Mbps alcançado nos níveis de 7 a 10. Ocorreu, entretanto, uma saturação devido ao hardware indicando que grande parte do tráfego não chegou ao refletor;
- Os *switches* utilizados tiveram um desempenho semelhante;
- O refletor não conseguiu amplificar todo o tráfego destinado a ele devido ao tempo de processamento do GetBulkRequest. Uma saturação da amplificação ocorreu do nível 3 em diante;
- A amplificação causada pelo ataque não é somente em valores absolutos de bits mas também em número de pacotes.

Pelos resultados dos testes, o ataque de negação de serviço por reflexão e amplificação usando o SNMP foi demonstrado como viável. Para o ataque ser bem sucedido, deve-se configurar a ferramenta para que os refletores utilizados não saturem e sua amplificação se mantenha.

Com esse estudo, conseguiu-se entender melhor o ataque de reflexão e amplificação por SNMP. Vários aspectos que impedem ou podem amenizar o ataque foram discutidos e são a contribuição mais valiosa do trabalho. São eles:

- Utilizar versões do SNMP que implementem protocolos de segurança mais eficazes do que os presentes na versão SNMPv1 e SNMPv2c ou, caso isso não seja possível, modificar o

valor da string de comunidade para um valor diferente do default;

- Configurar os nós de gerenciamento para responder mensagens apenas dos endereços IP das estações de gerenciamento;
- Os *Internet Service Probviders* devem tomar medidas para impedir que *IP Spoofing* ocorra em suas redes, pois, sem a modificação do IP de origem nas mensagens utilizadas no ataque, não será possível redirecionar o tráfego para as vítimas;
- Usuários domésticos que não têm necessidade do uso do SNMP devem ser encorajados a desabilitar o protocolo em seus dispositivos;
- Informar os donos de dispositivos vulneráveis que seus aparelhos podem ser usados como refletores no ataque, a exemplo do que é feito com os servidores DNS e NTP vulneráveis [33].

5.1 Trabalhos Futuros

Devido ao grande valor que o estudo de ataques de negação de serviço podem trazer, os seguintes trabalhos são propostos:

- **Otimização da ferramenta** - A otimização da ferramenta é importante para melhorar o desempenho do ataque e verificar outras propriedades. Essa otimização pode ser feita utilizando-se estruturas de dados mais rápidas e eliminando o tempo gasto com informações enviadas para a tela. Outra otimização é a utilização de mais GetBulkRequest na hora de calcular a amplificação máxima para novos dispositivos. Desse modo uma amplificação mais precisa pode ser alcançada. Outra forma de otimização é transformar a ferramenta em linha de comando. Assim, a execução de testes por *script* será possível agilizando os testes;
- **Extensão da ferramenta** - A ferramenta desenvolvida abre espaço para a implementação de outros vetores de ataque como o DNS, o NTP e o *Simple Service Discovery Protocol* (SSDP). Desse modo, pode-se obter um entendimento de como os ataques se relacionam quando executados juntos;
- **Teste com combinações de variáveis** - O efeito de saturação observado no trabalho se deve ao tempo de processamento do GetBulkRequest. Testes podem ser elaborados para pedir diferentes combinações de variáveis da MIB. Dessa forma, o valor da amplificação

será menor. Entretanto, o tempo de processamento pode também diminuir, levando a uma amplificação mais sustentável.

- **Teste com vários refletores** - O teste com mais refletores talvez seja um dos mais importantes para trabalhos futuros. Esse teste deve identificar o comportamento do ataque e da ferramenta com muitos refletores;
- **Teste com dispositivos diferentes** - Uma importante contribuição para a análise dos resultados encontrados nesse trabalho seria a realização dos testes com dispositivos diferentes. Isso seria necessário para verificar quanto que a configuração de hardware do dispositivo e sua implementação do SNMP influenciam no resultado. Esse teste seria especialmente útil para refletores diferentes;
- **Teste na internet** - Um teste realizado na internet com dispositivos próprios seria interessante, pois a velocidade de transmissão da internet entre os dispositivos pode influenciar nos resultados e valores mais reais para os limites da ferramenta poderiam ser obtidos.

REFERÊNCIAS BIBLIOGRÁFICAS

1 MCDOWELL, M. *Understanding Denial-of-Service Attacks*. [S.l.], 2009. Citado na página 1.

2 PAXSON, V. *An Analysis of Using Reflectors for Distributed Denial-of-Service Attacks*. 2001. AT&T Center for Internet Research at ICSI. Citado na página 1.

3 SMURFTROLL. *High Orbit Ion Cannon*. 2014. Disponível em: <http://sourceforge.net/projects/highorbitioncannon/>. Citado na página 4.

4 GU, Q.; LIU, P. *Denial of Service Attacks*. 2007. Texas State University. Disponível em: <http://s2.ist.psu.edu/paper/DDoS-Chap-Gu-June-07.pdf>. Citado na página 4.

5 SPECHT, S. M.; LEE, R. B. *Distributed Denial of Service: Taxonomies of Attacks, Tools and Countermeasures*. 2004. Princeton University. Citado na página 5.

6 ALI, F. *IP Spoofing*. 2007. <http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_10-4/104_ip-spoofing.html>. Citado na página 6.

7 SANFILIPPO, S. *Hping3*. 2005. Disponível em: <http://www.hping.org/manpage.html>. Citado na página 6.

8 MICROSOFT CORPORATION. *Stop 0A in Tcpip.sys When Receiving Out Of Band (OOB) Data*. 2013. Disponível em: <https://support.microsoft.com/en-us/kb/kbview/143478>. Citado na página 7.

9 KENNEY, M. *Ping of Death*. 1996. Disponível em: <http://insecure.org/sploits/ping-o-death.html>. Citado na página 7.

10 CIRSCUOLO, P. J. *Distributed Denial of Service Trin00, Tribe Flood Network, Tribe Flood Network 2000, And Stacheldraht CIAC-2319*. 2000. Department of Energy Computer Incident Advisory Capability (CIAC). Citado 2 vezes nas páginas 8 and 9.

11 MOCKAPETRIS, P. *Domain names - concepts and facilities*. IETF, 1987. RFC 1034 (INTERNET STANDARD). (Request for Comments, 1034). Updated by RFCs 1101, 1183, 1348, 1876, 1982, 2065, 2181, 2308, 2535, 4033, 4034, 4035, 4343, 4035, 4592, 5936. Disponível em: <http://www.ietf.org/rfc/rfc1034.txt>. Citado na página 10.

12 HOEPERS, C.; STEDING-JESSEN, K.; MURILO, N.; OBELHEIRO, R. R. *Recomendações para Evitar o Abuso de Servidores DNS Recursivos Abertos*. 2013. CERT.BR. Citado na página 10.

13 US-CERT. *DNS Amplification Attacks*. 2013. Alert (TA13-088A). Disponível em: <https://www.us-cert.gov/ncas/alerts/TA13-088A>. Citado na página 10.

14 ISO. *ISO Transport Protocol specification ISO DP 8073*. IETF, 1984. RFC 905. (Request for Comments, 905). Disponível em: <http://www.ietf.org/rfc/rfc905.txt>. Citado na página 10.

15 US-CERT. *NTP Amplification Attacks Using CVE-2013-5211*. 2014. Alert (TA14-013A). Disponível em: <https://www.us-cert.gov/ncas/alerts/TA14-013A>. Citado na página 11.

16 CASE, J.; FEDOR, M.; SCHOFFSTALL, M.; DAVIN, J. *Simple Network Management Protocol (SNMP)*. IETF, 1990. RFC 1157 (Historic). (Request for Comments, 1157). Disponível em: <http://www.ietf.org/rfc/rfc1157.txt>. Citado na página 11.

17 ROSE, M.; MCCLOGHRIE, K. *Structure and identification of management information for TCP/IP-based internets*. IETF, 1990. RFC 1155 (INTERNET STANDARD). (Request for Comments, 1155). Disponível em: <http://www.ietf.org/rfc/rfc1155.txt>. Citado na página 11.

18 MCCLOGHRIE, K.; ROSE, M. *Management Information Base for network management of TCP/IP-based internets*. IETF, 1990. RFC 1156 (Historic). (Request for Comments, 1156). Disponível em: <http://www.ietf.org/rfc/rfc1156.txt>. Citado na página 11.

19 MCCLOGHRIE, K.; ROSE, M. *Management Information Base for Network Management of TCP/IP-based internets: MIB-II*. IETF, 1991. RFC 1213 (INTERNET STANDARD). (Request for Comments, 1213). Updated by RFCs 2011, 2012, 2013. Disponível em: <http://www.ietf.org/rfc/rfc1213.txt>. Citado na página 11.

20 MCCLOGHRIE, K.; PERKINS, D.; SCHOENWAELDER, J. *Structure of Management Information Version 2 (SMIv2)*. IETF, 1999. RFC 2578 (INTERNET STANDARD). (Request for Comments, 2578). Disponível em: <http://www.ietf.org/rfc/rfc2578.txt>. Citado na página 11.

21 CASE, J.; MCCLOGHRIE, K.; ROSE, M.; WALDBUSSER, S. *Introduction to Community-based SNMPv2*. IETF, 1996. RFC 1901 (Historic). (Request for Comments, 1901). Disponível em: <http://www.ietf.org/rfc/rfc1901.txt>. Citado na página 11.

22 MCCLOGHRIE, K. *An Administrative Infrastructure for SNMPv2*. IETF, 1996. RFC 1909 (Historic). (Request for Comments, 1909). Disponível em: <http://www.ietf.org/rfc/rfc1909.txt>. Citado na página 11.

23 WATERS, G. *User-based Security Model for SNMPv2*. IETF, 1996. RFC 1910 (Historic). (Request for Comments, 1910). Disponível em: <http://www.ietf.org/rfc/rfc1910.txt>. Citado na página 11.

24 PRESUHN, R. *Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)*. IETF, 2002. RFC 3418 (INTERNET STANDARD). (Request for Comments, 3418). Disponível em: <http://www.ietf.org/rfc/rfc3418.txt>. Citado na página 14.

25 LEGG, S. *Abstract Syntax Notation X (ASN.X)*. IETF, 2007. RFC 4912 (Experimental). (Request for Comments, 4912). Disponível em: <http://www.ietf.org/rfc/rfc4912.txt>. Citado na página 14.

26 PROLEXIC. *Threat Advisory: SNMP Reflection DDoS Attacks*. 2015. Citado 2 vezes nas páginas 22 and 33.

27 ORACLE. *Swing (Java Foundation Classes)*. 2015. Disponível em: <http://docs.oracle.com/javase/7/docs/technotes/guides/swing/>. Citado na página 24.

28 SNMP4J. *SNMP4J - The Object Oriented SNMP API for Java Managers and Agents*. 2015. Disponível em: <http://www.snmp4j.org>. Citado 2 vezes nas páginas 28 and 29.

29 ORACLE. *Java Native Interface*. 2015. Disponível em: <http://docs.oracle.com/javase/7/docs/technotes/guides/jni/>. Citado na página 32.

30 GOOGLE INC. *Guava: Google Core Libraries for Java*. 2015. Disponível em: <https://github.com/google/guava>. Citado na página 33.

31 GUAVA PROJECT. *Rate Limiter*. 2015. Disponível em: <http://docs.guava-libraries.googlecode.com/git/javadoc/com/google/common/util/concurrent/RateLimiter.html>. Citado na página 33.

32 WIRESHARK FOUNDATION. *Wireshark*. 2015. Disponível em: <https://www.wireshark.org>. Citado na página 35.

33 OPEN Resolver Project. 2015. Disponível em: <http://openresolverproject.org>. Citado na página 57.