

Universidade Federal da Paraíba
Centro de Informática
Programa de Pós-Graduação em Informática

Defesas Seletivas para Mitigar Ataques de Negação de
Serviço às Aplicações de VoIP

Marcilio Olinto de Oliveira Lemos

João Pessoa, Paraíba, Brasil

Fevereiro de 2017

Universidade Federal da Paraíba
Centro de Informática
Programa de Pós-Graduação em Informática

Defesas Seletivas para Mitigar Ataques de Negação de Serviço
às Aplicações de VoIP

Marcilio Olinto de Oliveira Lemos

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Informática da Universidade Federal da Paraíba como parte dos requisitos necessários para obtenção do grau de Mestre em Informática.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Sinais, Sistemas Digitais e Gráficos

Prof. Dr. Iguatemi E. da Fonseca
(Orientador)

Prof. Dr. Vivek Nigam
(Co-Orientador)

João Pessoa, Paraíba, Brasil

©Marcilio Olinto de Oliveira Lemos, 20 de Fevereiro de 2017

L557d Lemos, Marcilio Olinto de Oliveira.
Defesas seletivas para mitigar ataques de negação de
serviço às aplicações de VoIP / Marcilio Olinto de Oliveira
Lemos - João Pessoa, 2017.
84 f. : il. -

Orientador: Iguatemi E. da Fonseca.
Coorientador: Vivek Nigam.
Dissertação (Mestrado) - UFPB/CI

1. Informática. 2. VoIP. 3. SIP. 4. DDoS. I. Título.

UFPB/BC

CDU: 004(043)



UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA



Ata da Sessão Pública de Defesa de Dissertação de Mestrado de MARCILIO OLINTO DE OLIVEIRA LEMOS, candidato ao título de Mestre em Informática na Área de Sistemas de Computação, realizada em 20 de fevereiro de 2017.

1 Aos vinte dias do mês de fevereiro, do ano de dois mil e dezessete, às dez horas, no Centro
2 de Informática da Universidade Federal da Paraíba, em Mangabeira, reuniram-se os
3 membros da Banca Examinadora constituída para julgar o Trabalho Final do Sr. Marcilio
4 Olinto De Oliveira Lemos, vinculado a esta Universidade sob a matrícula nº 2015104060,
5 candidato ao grau de Mestre em Informática, na área de "Sistemas de Computação", na linha
6 de pesquisa "Sinais, sistemas digitais e gráficos", do Programa de Pós-Graduação em
7 Informática, da Universidade Federal da Paraíba. A comissão examinadora foi composta
8 pelos professores: Iguatemi Eduardo Da Fonseca (PPGI-UFPB), Orientador e Presidente da
9 Banca, Gustavo Henrique Matos Bezerra Motta (PPGI-UFPB), Examinador Interno ao
10 Programa, Examinador interno ao programa, e Divanilson Rodrigo De Sousa Campelo
11 (UFPE), Examinador externo à instituição. Dando início aos trabalhos, o Presidente da
12 Banca cumprimentou os presentes, comunicou aos mesmos a finalidade da reunião e
13 passou a palavra ao candidato para que o mesmo fizesse a exposição oral do trabalho de
14 dissertação intitulado "Defesas seletivas para mitigar ataques de negação de serviço às
15 aplicações de VoIP". Concluída a exposição, o candidato foi arguido pela Banca
16 Examinadora que emitiu o seguinte parecer: "**aprovado**". Do ocorrido, eu, Claurton de
17 Albuquerque Siebra, Coordenador do Programa de Pós-Graduação em Informática, lavrei a
18 presente ata que vai assinada por mim e pelos membros da banca examinadora. João
19 Pessoa, 20 de fevereiro de 2017.

Prof. Dr. Claurton de Albuquerque Siebra

Prof. Dr. Iguatemi Eduardo Da Fonseca
Orientador (PPGI-UFPB)

Prof. Dr. Gustavo Henrique Matos Bezerra Motta
Examinador interno (PPGI-UFPB)

Prof. Dr. Divanilson Rodrigo De Sousa Campelo
Examinador externo à instituição (UFPE)

Resumo

Assim como outros serviços baseados em redes IP, aplicações VoIP estão vulneráveis aos perigosos ataques distribuídos de negação de serviço (DDoS). Comumente, esses ataques eram efetuados enviando inúmeras requisições a um servidor SIP (*SIP flooding*), de modo a sobrecarregá-lo. Embora bastante prejudiciais, tais ataques podem ser facilmente identificados por defesas que detectam mudanças abruptas no volume de tráfego da rede. Ataques mais recentes, classificados como *Low-Rate*, conseguem sobrepujar essas defesas, atacando aplicações e gerando tráfego bastante similar ao de clientes normais, tornando tais defesas ineficazes. Em uma nova forma de DDoS denominada Negação de Serviço de Telefonia (TDoS - *Telephony Denial of Service*), atacantes vem fazendo uso de chamadas maliciosas para impedir que clientes legítimos consigam receber ou realizar chamadas. Um exemplo de TDoS *Low-Rate* é o *Coordinated Call Attack*, onde atacantes simplesmente realizam chamadas entre si para esgotar os recursos do servidor VoIP alvo. Uma defesa seletiva eficiente contra ataques *Low Rate* que exploram o protocolo HTTP é a ferramenta SeVen. O presente trabalho demonstra que estratégias seletivas também podem ser usadas com sucesso para mitigar ataques TDoS, em particular, o *Coordinated Call Attack*. As contribuições deste trabalho são de três tipos: (1) Uma defesa seletiva adequada para aplicações VoIP; (2) A Formalização da defesa na ferramenta Maude e condução de simulações usando um verificador estatístico de modelos (PVeStA); (3) Resultados experimentais mostram que sem defesa, menos de 41% dos usuários conseguem acessar o serviço VoIP alvo, enquanto que com a defesa seletiva, a disponibilidade fica em torno de 98%.

Palavras-chave: VoIP, SIP, DDoS.

Abstract

Similar to others IP network based services, VoIP applications are quite vulnerable to dangerous distributed denial of service attacks (DDoS). Commonly, these attacks had been carried out by sending numerous requests to a SIP server (SIP flooding) in order to overload it. Although still harmful, such attacks can be easily identified by defenses that detect sudden changes in the network traffic volume. Most recent attacks, classified as Low-Rate, have bypassed such defense mechanisms attacking applications by generating traffic very similar to normal client traffic rendering such monitoring defenses ineffective. In a new form of DDoS attack called Telephony Denial of Service (TDoS), attacker has been using malicious calls to prevent legitimate clients to receive or to make calls. An example of a Low-Rate TDoS attack is the Coordinated Call Attack, where attackers simply make calls to each other to exhaust the target VoIP server's resources. An efficient selective defense against Low-Rate attacks exploiting the HTTP protocol is the tool SeVen. This work demonstrates that selective strategies can also be successfully used to mitigate TDoS attacks, in particular, the Coordinated Call Attack. The contributions of this work are three-fold: (1) A selective strategy suitable for VoIP applications; (2) Formulating the defense in the Maude tool and conducting simulations using statistical model checker (PVeStA); (3) Experimental results show that without defense, less than 41% of users have access to the target VoIP service, whereas with the selective defense, availability is around 98%.

Keywords: VoIP, SIP, DDoS.

Agradecimentos

Agradeço, primeiramente, aos meus pais, Mariano e Ivanete, por todo amor, carinho, compreensão e apoio incondicional em todos os momentos. Aos meus irmãos, pelo incentivo, em especial a José Marcelo, por ter me dado a oportunidade de ter uma educação básica de qualidade, e Paulo, pelas palavras certas no momento certo.

Aos meus amigos de fé e irmãos camaradas: Daniel, Eduardo, Iron, Matheus, Rodrigo, Rubens e Túlio, que ao longo da graduação e mestrado, demonstraram união, companheirismo e amizade.

A muitos dos meus mestres, pelo conhecimento passado de bom grado, em especial meus orientadores, Dr. Iguatemi E. Fonseca e Dr. Vivek Nigam, pela imensurável contribuição para minha formação acadêmica e por me guiarem na condução deste trabalho.

Agradeço aos meus colegas do LaR, pelas oportunidades e experiência adquiridas, particularmente ao Me. Yuri Gil Dantas, pela ajuda com a concepção do modelo formal e condução das simulações.

Por fim, agradeço ao Google, pela gratuidade e infinidade de informação, Stack Overflow, pela ajuda com diversos problemas relacionados a programação, e a Babagge, Ada, Boole, Shannon, Zuse, Turing e Neumann, pela Ciência da Computação.

Dedico este trabalho à memória de meu pai, Mariano Lemos Sobrinho (1936-2016)

Conteúdo

1	Introdução	1
1.1	Motivação	1
1.2	Objetivos	4
1.2.1	Objetivo Geral	4
1.2.2	Objetivos Específicos	4
1.3	Metodologia	5
1.4	Publicações Relacionadas	6
1.5	Estrutura da Dissertação	6
2	Fundamentação Teórica	8
2.1	Voice over Internet Protocol (VoIP)	8
2.2	Session Initiation Protocol (SIP)	9
2.2.1	Componentes	10
2.2.2	Mensagens	10
2.2.3	Sinalização	12
2.3	Ataques de Negação de Serviço	13
2.3.1	SIP <i>Flooding</i>	15
2.3.2	<i>Dial Attack</i>	16
2.3.3	<i>Coordinated Call Attack</i>	17
2.4	Métodos Formais	18
2.4.1	Lógica de Reescrita	18
2.4.2	Maude	20
2.4.3	PVeStA	22
2.5	Trabalhos Relacionados	23

2.5.1	Defesas contra o SIP <i>Flooding</i>	23
2.5.2	Defesas contra o <i>Dial Attack</i>	27
2.5.3	Métodos Formais e DDoS	28
2.6	Considerações	29
3	Estratégia Proposta para Mitigar Ataques às Aplicações VoIP	32
3.1	Estratégia Seletiva contra Ataques TDoS <i>Low-Rate</i>	32
3.1.1	Exemplo de Execução	37
3.1.2	Intuição, Limitações e Extensões	39
3.2	Especificação Formal	42
3.2.1	Sorts e Funções	43
3.2.2	Regras de Reescrita	46
3.3	Simulações	49
3.3.1	Resultados sem Defesa	52
3.3.2	Resultados com Defesa	53
3.4	Considerações	54
4	Resultados Experimentais	56
4.1	Plano do Experimento	56
4.2	Parâmetros	58
4.3	Métricas de Qualidade	59
4.4	Resultados Experimentais	60
5	Conclusão e Trabalhos Futuros	66
	Referências Bibliográficas	74

Lista de Símbolos

VoIP : *Voice Over IP*

IP : *Internet Protocol*

DDoS : *Distributed Denial of Service*

TCP : *Transmission Control Protocol*

ADDoS : *Application-Layer Distributed Denial of Service*

HTTP : *Hypertext Transfer Protocol*

DNS : *Domain Name System*

SIP : *Session Initiation Protocol*

TDoS : *Telephony Denial of Service*

RTP : *Real-time Transport Protocol*

UDP : *User Datagram Protocol*

TSL : *Transport Layer Security*

PSTN : *Public Switched Telephone Network*

DoS : *Denial of Service*

NAT : *Network Address Translation*

Lista de Figuras

2.1	Exemplo de uma mensagem do protocolo SIP.	11
2.2	Exemplo de uma sessão multimédia SIP.	13
2.3	Ataque SIP <i>flooding</i> usando mensagens INVITE.	15
2.4	<i>Dial Attack</i>	16
2.5	Ilustração do <i>Coordinated Call Attack</i> . Os <i>slots</i> representam recursos sendo usados pelas chamadas $C_1, C_2, C_3, C_4, C_5, \dots, C_n$	17
2.6	Exemplo do problema <i>Blocks World</i> com três blocos.	19
2.7	Formalização do problema <i>Blocks World</i> na ferramenta Maude.	21
3.1	Gráfico ilustrando o comportamento da estratégia seletiva de acordo com o estado da chamada e sua duração. p_{WAIT} representa a chance de uma chamada com estado WAITING ser descartada, enquanto p_{IN} indica a chance de uma chamada com estado INCALL ser interrompida.	36
3.2	Ilustração da eficácia de estratégias seletivas contra o <i>Coordinated Call Attack</i> . Círculos (em preto) denotam clientes legítimos e as cruzeiras (em vermelho) denotam chamadas de atacantes.	40
3.3	Regras de reescrita especificando a estratégia seletiva proposta no presente trabalho.	48
3.4	Grau de Sucesso do Cliente: Resultados das simulações sob ataque e defesa não sendo executada.	53
3.5	Grau de Sucesso do Cliente: Resultados das simulações sob ataque e usando a defesa.	54
4.1	Topologia da infraestrutura VoIP empregada nos experimentos.	57

4.2	Arquitetura da aplicação <i>proxy</i> que implementa a defesa proposta. O módulo <i>Core</i> é responsável por lidar com as mensagens SIP, enquanto que uma biblioteca compartilhada fica consultando o estado do servidor Asterisk e encerrando chamadas	58
4.3	Disponibilidade do servidor VoIP.	61
4.4	Índice de duração médio das chamadas incompletas.	63
4.5	Ocupação do atacante ao longo do tempo, quando a taxa de chamadas do atacante é de 30 chamadas por minuto.	64

Lista de Tabelas

2.1	Métodos de requisição básicos presentes no SIP.	12
-----	---	----

Capítulo 1

Introdução

Nos últimos anos é notável a crescente popularidade do uso de serviços de Voz sobre IP (VoIP - *Voice over Internet Protocol*) para comunicação de voz através da Internet, motivada pelos seus baixos custos, maior flexibilidade e recursos de telefonia mais sofisticados. Nesse tipo de serviço, usuários se comunicam por meio do envio de voz codificada em pacotes de dados, que são transportados através de redes IP (*Internet Protocol*) (Ferdous 2014).

Por operarem em uma rede IP, serviços VoIP estão expostos às vulnerabilidades de segurança herdadas do protocolo IP, como aquelas exploradas pelos Ataques Distribuídos de Negação de Serviço (DDoS - *Distributed Denial of Service*). Tradicionalmente, ataques DDoS são direcionados à camada de transporte do modelo TCP/IP (Socolofsky 1991). Entretanto, ataques mais recentes, denominados *Application-Layer DDoS* (ADDoS), estão sendo conduzidos na camada de aplicação, objetivando afetar uma aplicação específica (ex: HTTP, DNS, VoIP). O presente trabalho investiga ataques de negação de serviço direcionados ao componente chave da maioria das infraestruturas VoIP, o **servidor SIP**.

1.1 Motivação

O Protocolo de Inicialização de Sessão (SIP - *Session Initiation Protocol*) é amplamente usado para gerenciar chamadas em serviços VoIP. Apesar da infraestrutura SIP ser constituída por diversos componentes (Rosenberg et al. 2002), servidores SIP são os alvos mais frequentes de ataques DDoS (Stanek 2010), pois a sua desativação acarreta na indisponibilidade de todo o serviço. Além disso, o servidor SIP é especial-

mente propenso a DDoS (Marchal et al. 2015). Um ataque do tipo inundação (SIP *flooding*) (Zargar, Joshi e Tipper 2013) lançado contra servidores SIP da CallCentric foi capaz de, em apenas 1 minuto, aumentar a carga dos servidores em 100% (Bode 2012). Ademais, apesar de o ataque ter durado dois dias, os efeitos dele perduraram por 12 dias. Outro caso semelhante ocorreu com a TelePacific Communications um ano antes (Messmer 2011). Portanto, ataques DDoS a servidores SIP representam um problema severo à integridade do serviço VoIP.

Várias soluções têm sido propostas para detectar e mitigar ataques DDoS a servidores SIP, como em Tang et al. 2014, Ehlert et al. 2008, Wan, Li e Fan 2010, Zi-Fu, Jun-Rong e Xiao-Yu 2010¹. Muitas destas soluções se baseiam em análise do fluxo de tráfego da rede, de modo que sempre que houver um aumento incomum de tráfego, alguma contra-medida é aplicada, como o bloqueio de IPs. Um problema desse tipo de abordagem é que a defesa se torna ineficaz caso o ataque DDoS seja capaz de gerar tráfego similar ao de clientes habituais, como é o caso dos ataques *Low-rate*.

Em uma nova forma de DDoS denominada Ataque de Negação de Serviço de Telefonia (TDoS - *Telephony Denial of Service*) (Guri, Mirsky e Elovici 2016), atacantes usam chamadas comuns para impedir que usuários honestos consigam utilizar o serviço VoIP. Por exemplo, um simples *script* automatizado que repetidamente inicia chamadas para todos os *links* de comunicação VoIP da vítima pode, eficazmente, impedir que ela seja capaz de receber ou realizar outras chamadas (*Dial Attack*) (Kapravelos et al. 2010).

Para muitas organizações que utilizam o VoIP, os efeitos do TDoS podem ser devastadores. Para serviços públicos vulneráveis, esses ataques podem implicar atrasos fatais em lidar com emergências (Guri, Mirsky e Elovici 2016). Em 2013, o Departamento de Segurança Interna dos EUA e o FBI emitiram um alerta afirmando que vários serviços públicos estavam suscetíveis ao TDoS (Center 2013), enquanto ataques a centrais 911 (serviço de emergência americano) foram registrados em 2016 (Carman 2016).

Um exemplo de TDoS *Low-rate* é o *Coordinated Call Attack* (Lemos et al. 2016). Em vez de ocupar todas as linhas de um usuário, como é feito no *Dial Attack*, o *Coordinated Call Attack* tenta ocupar todas as "linhas" do servidor SIP. Nesse caso, uma linha representa

¹A Seção 2.5 do Capítulo 2 descreve uma visão geral dos principais mecanismos de defesa contra DDoS ao VoIP.

os recursos do servidor que são alocados para cada chamada. Assumindo que o servidor alvo é do tipo *stateful*²(Rosenberg et al. 2002), e que pares de atacantes, Alice e Bob, conspiram para esgotar os recursos do servidor, o ataque consiste simplesmente em Alice realizar uma chamada para Bob e manter esta chamada ativa pelo máximo de tempo possível (mantendo assim a linha ocupada). Deste modo, utilizando-se um número significativo de chamadas, é possível ocupar todas as "linhas", *i.e.*, esgotar todos os recursos do servidor. Provedores VoIP de baixo custo, geradores de chamadas difíceis de rastrear (Guha, Daswani e Jain 2006) e tutoriais na Internet (Carman 2009) facilitam a concepção de *Coordinated Call Attack* automatizado.

Identificar e mitigar o *Coordinated Call Attack* não é uma tarefa fácil, pois a quantidade de tráfego gerado por este é bastante pequena (se comparado ao ataque de inundação) e similar ao de clientes honestos, de modo que é difícil detectá-lo com métodos que monitoram variações abruptas no tráfego da rede, como em Tang et al. 2014, Ehlert et al. 2008, Wan, Li e Fan 2010, Zi-Fu, Jun-Rong e Xiao-Yu 2010.

Existem contramedidas conhecidas e empregadas para lidar com ataques TDoS (High-rate). Uma abordagem empregada é bloquear usuários que abusam do serviço através da implementação e aplicação de uma lista negra (CISCO 2014). Se o ataque for realizado a partir de aparelhos infectados por algum tipo de *malware*, esse método impedirá o proprietário legítimo do dispositivo de fazer chamadas, o que pode levar a implicações éticas em serviços críticos, como os de emergência. Outra opção é tentar detectar silêncio (PINDROP 2014), encerrando chamadas que não apresentam áudio. Tal abordagem é ineficaz contra atacantes que injetam áudio gravado nas ligações. Uma terceira medida preventiva é a detecção de presença humana. Por exemplo, solicitar o pressionamento de dígitos ou uso de *captcha* (Polakis, Kontaxis e Ioannidis 2011). Entretanto, esta técnica ainda pode levar a sobrecarga da rede, além de não ser amigável para as necessidades da comunidade surda. Portanto, é evidente a necessidade de desenvolver métodos mais eficazes em combater o *Coordinated Call Attack*.

Recentemente, Dantas, Nigam e Fonseca 2014 propuseram uma ferramenta chamada SeVen, que utiliza de estratégias seletivas para mitigar ataques ADDoS. O SeVen é regido por funções de probabilidade que especificam as chances de uma requisição ser descartada

²Os usuários ficam usando os recursos da aplicação durante toda a duração da chamada.

quando o serviço está sobrecarregado. Dantas, Nigam e Fonseca 2014 mostram que usando uma simples distribuição uniforme, o SeVen pode ser operado contra uma série de ataques que exploram o protocolo HTTP (Slowloris e POST). No entanto, nesse trabalho só consideraram ataques HTTP contra servidores web, de modo que é relevante investigar se, com o uso de defesas seletivas, é possível mitigar ataques DDoS a serviços VoIP.

O presente trabalho propõe a utilização de estratégias seletivas para mitigar ataques DDoS a serviços VoIP. Intuitivamente, a distribuição uniforme não é adequada para lidar com o *Coordinated Call Attack* devido aos requisitos de qualidade exigidos em comunicações VoIP. Em muitos casos, é verdade que a interrupção de uma chamada existente é considerada mais prejudicial à reputação do serviço do que finalizar uma requisição parcialmente processada em um servidor web. Para serviços VoIP, é desejável que os clientes legítimos tenham mais chances de continuar usando os recursos do servidor. Deste modo, a função de distribuição de probabilidade que rege o comportamento da defesa foi alterada de distribuição uniforme para distribuições mais sofisticadas.

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo geral desse trabalho é propor um mecanismo de defesa baseado em estratégias seletivas que seja capaz de mitigar ataques DDoS do tipo *Low-rate* a serviços VoIP.

1.2.2 Objetivos Específicos

Objetivo 1: Propor uma nova estratégia seletiva apropriada para serviços VoIP.

Objetivo 2: Empregar distribuições dependentes de estado para decidir qual chamada em andamento deve ser selecionada para continuar e qual deve ser descartada.

Objetivo 3: Formalizar o *Coordinated Call Attack* e a defesa seletiva na ferramenta computacional, baseada em Lógica de Reescrita, Maude (Clavel et al. 2007).

Objetivo 4: Implementar uma aplicação que use a estratégia seletiva proposta e que seja capaz de operar em conjunto com servidores VoIP.

Objetivo 5: Validar o mecanismo de defesa proposto por meio de experimentos realizados sobre um arquitetura VoIP usual.

Objetivo 6: Verificar a eficácia da defesa proposta em lidar com ataques ao VoIP, como por exemplo, o *Coordinated Call Attack*.

1.3 Metodologia

A elaboração deste trabalho seguiu as seguintes atividades:

- Estudo da tecnologia VoIP e do protocolo SIP: Realização de estudo acerca dos fundamentos da tecnologia VoIP e das principais características do protocolo SIP;
- Levantamento Bibliográfico: Coleta de artigos científicos que propõem mecanismos de defesa contra ataques DDoS a serviços VoIP. A partir daí, foi realizada uma filtragem das publicações mais recentes que fossem relevantes;
- Elaboração da Estratégia Seletiva: Definição de uma estratégia seletiva baseada em funções de probabilidade capaz de mitigar o *Coordinated Call Attack* a servidores VoIP;
- Aprendizado acerca de Métodos Formais baseados em Lógica de Reescrita: Efetuação de estudo sobre a ferramenta Maude e do verificador estatístico de modelos PVeStA;
- Modelagem Formal e Verificação Estatística: Modelagem do *Coordinated Call Attack* e da estratégia seletiva proposta no Maude e condução de simulações usando o PVeStA, a fim de obter uma primeira impressão sobre qual seria a eficácia do método proposto em lidar com ataques ao VoIP;
- Desenvolvimento da Aplicação: Desenvolvimento de uma aplicação em C++ que implementa a estratégia seletiva proposta e que opera como um *proxy* para o servidor Asterisk;
- Modelagem de Ataques ao VoIP: Configuração da ferramenta de geração de chamadas SIPp (GAYRAUD e JACQUES 2014) de modo que ela seja capaz de realizar diversos ataques DDoS, como por exemplo, o *Coordinated Call Attack* ao servidor Asterisk;

- Realização de Experimentos: Realização de uma série de experimentos para avaliar a eficácia da solução proposta em mitigar o *Coordinated Call Attack*.

1.4 Publicações Relacionadas

Este trabalho resultou em duas publicações em anais de congresso e uma submissão para um periódico. A primeira publicação apresentou a formalização da defesa na ferramenta computacional Maude e a validação desta por meio da ferramenta de verificação estatística de modelos PVeStA. O projeto foi aceito para o *11th International Workshop on Rewriting Logic and its Applications* realizado em Eindhoven na Holanda. O trabalho enviado ao congresso foi publicado como artigo completo na base de dados Springer e intitulado de "*Formal Specification and Verification of a Selective Defense for TDoS Attacks*".

As etapas de implementação deste trabalho, mais especificamente o desenvolvimento de uma aplicação que implementa a defesa proposta e que opera em arquiteturas VoIP, bem como os experimentos em rede para validá-la, foi aceito e publicado como artigo completo intitulado "*A Selective Defense for Mitigating Coordinated Call Attacks*" no 34th Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC) que se realizou na cidade de Salvador – BA em 2016.

Um terceiro artigo intitulado "*On the Accuracy of Formal Verification of Selective Defenses for TDoS Attacks*" foi produzido e submetido para o periódico "*Journal of Logical and Algebraic Methods in Programming*", da editora Elsevier. Tal artigo apresenta uma série de simulações e experimentos com diferentes configurações para o atacante e para a defesa. Também é traçado um comparativo entre os resultados experimentais e aqueles produzidos pela modelagem formal, demonstrando que os métodos formais podem eficazmente ser usados para avaliar novas defesas a fim de mitigar ataques TDoS com menos esforços que a implementação e realização de experimentos na rede.

1.5 Estrutura da Dissertação

Este documento está dividido em 5 capítulos. O **Capítulo 1** apresenta o trabalho de maneira geral descrevendo sua motivação, objetivos, metodologia e estrutura da dissertação.

No **Capítulo 2**, se encontra o embasamento teórico com os conceitos e definições relacionados ao VoIP, protocolo SIP, métodos formais e *Coordinated Call Attack*. Nesse capítulo também é descrito o estado da arte em defesas contra ataques DDoS direcionados a serviços VoIP. A nova estratégia seletiva usada para mitigar o *Coordinated Call Attack* é apresentada no **Capítulo 3**, juntamente com sua especificação formal e validação preliminar por meio de simulações. O **Capítulo 4** descreve os experimentos, conduzidos na rede, para legitimar os resultados obtidos por simulação, bem como os resultados destes experimentos. As conclusões, discussão, trabalhos futuros e considerações finais se encontram no **Capítulo 5**.

Capítulo 2

Fundamentação Teórica

O objetivo deste capítulo é apresentar os conceitos fundamentais para entendimento do trabalho proposto. Primeiramente, na Seção 2.1, são apresentadas algumas definições básicas sobre VoIP. Em seguida, a Seção 2.2 introduz os principais conceitos do protocolo SIP, enfatizando suas características e componentes. Na Seção 2.3, são apontados alguns conceitos sobre DDoS e os principais ataques ao VoIP são discutidos, dando destaque ao *Coordinated Call Attack*, ataque abordado nesse trabalho. A Seção 3.2 discorre as técnicas utilizadas na especificação formal e simulações: Lógica de Reescrita, Maude e PVeStA. Por fim, a Seção 2.5 apresenta o estado da arte dos métodos para detecção e mitigação de ataques DDoS a serviços VoIP, apontando como os trabalhos presentes na literatura podem ser usados para mitigar o *Coordinated Call Attack*. Tal Seção também lista alguns trabalhos relacionados a aplicação de métodos de Lógica de Reescrita para formalizar técnicas de mitigação de DDoS.

2.1 Voice over Internet Protocol (VoIP)

VoIP é o termo usado para protocolos e mecanismos que possibilitam o uso de redes IP como meio de transmissão para comunicação multimídia. Termos como *Internet telephony* ou *IP telephony* são muitas vezes utilizados e têm o mesmo significado. No VoIP, os dados de voz (e outras mídias) fluem sobre uma rede de comutação de pacotes, em vez de uma rede de comutação de circuito, como acontece na telefonia tradicional. As vantagens da primeira em relação a segunda são: taxas mais baratas para as chamadas, integração com vários serviços,

melhor escalabilidade e melhor recuperação a falhas. Devido a estes benefícios, serviços VoIP são amplamente utilizados para fins comerciais e pessoais (Chiappetta et al. 2013).

Protocolos de sinalização e protocolos de transferência de mídia desempenham um papel fundamental no VoIP. O primeiro tem a função de estabelecer, manter e encerrar uma chamada entre usuários. Já o segundo é responsável pela transmissão em tempo real de dados, voz e vídeo. Em termos de aspecto técnico, a voz é digitalizada, convertida em um fluxo de pacotes e, em seguida, transportada por uma rede IP. Os principais protocolos de sinalização usados no VoIP são o H.323 (Jones 2016) e o SIP (Rosenberg et al. 2002), enquanto que o RTP (Schulzrinne et al. 2003) é o principal protocolo de transferência de mídia.

O SIP é o protocolo de sinalização mais facilmente adotado e o mais utilizado em serviços VoIP. Hoje em dia, os ambientes VoIP são geralmente baseados na combinação de SIP e RTP. Por esse motivo, o presente trabalho aborda tópicos de segurança relacionados ao protocolo SIP. É importante destacar que os protocolos mencionados não representam todos os que foram usados no VoIP. Existem centenas de outros protocolos com foco em diferentes aspectos do VoIP, mas estes estão fora do escopo deste trabalho. Para uma visão mais geral sobre o VoIP, Camp 2003 apresenta um bom ponto de partida.

2.2 Session Initiation Protocol (SIP)

O Protocolo de Inicialização de Sessão (SIP - *Session Initiation Protocol*) é um protocolo de sinalização da camada de aplicação usado para gerenciar sessões de multimídia entre dois ou mais participantes em uma rede IP. No contexto do VoIP, uma sessão pode representar desde uma simples chamada por voz até uma conferência multimídia colaborativa. Assim como o HTTP, o SIP é um protocolo baseado em texto e orientado a requisição-resposta.

O padrão SIP define três tipos de transporte utilizáveis em comunicações SIP. Os dois primeiros são os clássicos UDP e TCP, já o terceiro é o transporte criptografado usando TLS (Dierks. e Rescorla 2008). As seções seguintes deste capítulo descrevem brevemente a arquitetura do protocolo e apresentam alguns conceitos necessários para o entendimento da estratégia proposta neste trabalho.

2.2.1 Componentes

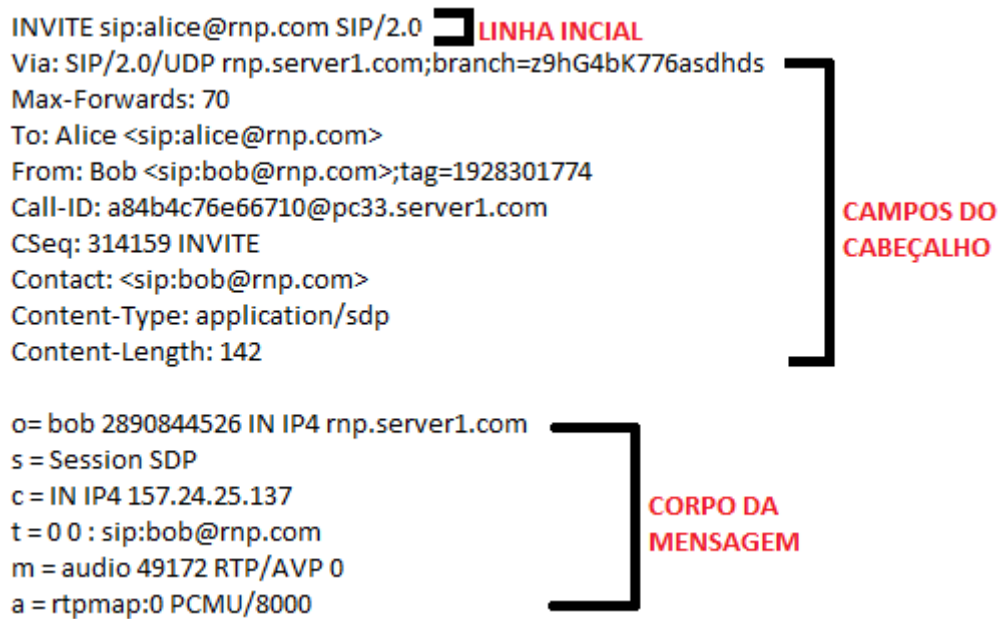
A seguir, são listados os principais componentes de uma arquitetura SIP:

- **Agente do Utilizador:** são elementos da rede capazes de criar e enviar, bem como receber e processar mensagens SIP. Em geral, agentes do utilizador podem ser dispositivos de usuário-final, tais como telefones celulares e *tablets*, ou aplicações *desktop* (por exemplo, Skype (Microsoft 2016));
- **Servidor Proxy:** O servidor proxy encaminha requisições e respostas entre elementos da rede, sendo capaz de processar e alterar as mensagens trocadas por estes elementos. Esse servidor pode ser *stateless* ou *stateful*, dependendo do fato de que eles devem ou não manter informações sobre o estado da sessão;
- **Registrador:** agentes do utilizador contatam esse servidor para anunciar sua presença na rede. O registrador consiste basicamente de um banco de dados contendo a localização e preferências do usuário;
- **Servidor de Redirecionamento:** O servidor de redirecionamento é usado durante o início de sessão para determinar o endereço do dispositivo chamado e auxilia o agente do utilizador do chamador a acessar domínios externos.

Os três últimos componentes listados geralmente são implementados na forma de um único componente referido como o **servidor SIP**. Nesse trabalho, o termo servidor SIP é usado para se referir ao componente de integração de todos estes três componentes de servidor.

2.2.2 Mensagens

A Figura 2.1 ilustra um exemplo de mensagem usada no protocolo SIP. Existem dois tipos de mensagem usadas nesse protocolo: requisições e respostas. Ambas são formadas por um cabeçalho e um corpo, sendo este último opcional. O cabeçalho é composto por uma linha inicial e um ou mais campos. Os campos especificam informações, como quem originou a mensagem (*From header*), para quem a mensagem deve ser encaminhada (*To header*), um identificador único para a sessão (*Call-ID header*) e etc.



The diagram illustrates the structure of a SIP message. It is divided into two main parts: the header and the body. The header section, labeled 'CAMPOS DO CABEÇALHO' (Header Fields), includes the initial line 'INVITE sip:alice@rnp.com SIP/2.0' and several key-value pairs: 'Via: SIP/2.0/UDP rnp.server1.com;branch=z9hG4bK776asdhds', 'Max-Forwards: 70', 'To: Alice <sip:alice@rnp.com>', 'From: Bob <sip:bob@rnp.com>;tag=1928301774', 'Call-ID: a84b4c76e66710@pc33.server1.com', 'CSeq: 314159 INVITE', 'Contact: <sip:bob@rnp.com>', 'Content-Type: application/sdp', and 'Content-Length: 142'. The message body section, labeled 'CORPO DA MENSAGEM' (Message Body), contains SDP data: 'o= bob 2890844526 IN IP4 rnp.server1.com', 's= Session SDP', 'c= IN IP4 157.24.25.137', 't= 0 0 : sip:bob@rnp.com', 'm= audio 49172 RTP/AVP 0', and 'a= rtpmap:0 PCMU/8000'. Brackets on the right side of the text group the lines into these two categories.

```

INVITE sip:alice@rnp.com SIP/2.0
Via: SIP/2.0/UDP rnp.server1.com;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: Alice <sip:alice@rnp.com>
From: Bob <sip:bob@rnp.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.server1.com
CSeq: 314159 INVITE
Contact: <sip:bob@rnp.com>
Content-Type: application/sdp
Content-Length: 142

o= bob 2890844526 IN IP4 rnp.server1.com
s= Session SDP
c= IN IP4 157.24.25.137
t= 0 0 : sip:bob@rnp.com
m= audio 49172 RTP/AVP 0
a= rtpmap:0 PCMU/8000
  
```

Figura 2.1: Exemplo de uma mensagem do protocolo SIP.

A linha inicial é ligeiramente diferente para requisições e respostas. Nas requisições, a linha inicial é formada por uma palavra-chave, especificando o tipo da requisição, uma URI, relacionada ao participante requisitado, e pela especificação da versão do SIP utilizada. Basicamente, essa linha informa o que é requisitado e de quem. Observando a Figura 2.1 é possível notar que o formato de uma SIP URI é semelhante a um endereço de e-mail. Nas respostas, a linha inicial é composta pela versão do SIP, pelo código da resposta e por uma frase relacionada ao código da resposta.

Existem cerca de 14 diferentes métodos de requisição no SIP, sendo 6 básicos e 8 adicionados como uma extensão. A Tabela 2.1 lista os tipos básicos de requisição que devem ser implementados por qualquer aplicação SIP. A implementação das extensões é opcional. Informações sobre os 8 métodos opcionais, bem como sobre todos os campos que podem ser inseridos nos cabeçalhos, podem ser encontrados em Rosenberg et al. 2002.

Os códigos de resposta do SIP são bastantes similares aos usados pelo HTTP. Esses códigos podem ser agrupados em 6 diferentes classes, definidas a seguir, em que x representa um dígito (0-9):

- **Provisórias (1xx):** A requisição foi recebida, mas o resultado de seu processamento ainda é desconhecido. Por exemplo, 180 (Ringing), enviada para indicar que o agente

Tabela 2.1: Métodos de requisição básicos presentes no SIP.

Método	Descrição
INVITE	Indica que um cliente está sendo convidado a participar de uma sessão
ACK	Confirma que o cliente recebeu uma resposta final a um INVITE
BYE	Encerra uma sessão
REGISTER	Registra o endereço listado no campo "To header" do cabeçalho
CANCEL	Cancela uma requisição pendente
OPTIONS	Obtém informações sobre todas as funcionalidades do servidor

do utilizador do destinatário recebeu a requisição INVITE e está alertando o usuário para a chamada;

- **Sucesso (2xx):** A requisição foi processada com sucesso e a ação pretendida foi aceita. Por exemplo, 200 (Ok), enviada quando um participante aceita um pedido para iniciar uma sessão (INVITE);
- **Redirecionamento (3xx):** Outra ação precisa ser tomada, a fim de completar a requisição. Por exemplo, 302 (Alternative Service), enviada quando uma chamada falha, mas alternativas são detalhadas no corpo da mensagem;
- **Erro no Cliente (4xx):** A requisição contém sintaxe inválida ou não pode ser cumprida neste servidor. Por exemplo, 401 (Unauthorized), enviada quando uma requisição necessita de autenticação por parte do usuário;
- **Erro no Servidor (5xx):** O servidor não cumpriu uma requisição aparentemente válida. Por exemplo, 503 (Service Unavailable), enviada quando o servidor está em manutenção ou está temporariamente sobrecarregado e por isso não pode processar a requisição;
- **Erro Global (6xx):** A requisição não pode ser cumprida em qualquer servidor. Por exemplo, 603 (Decline), enviada quando o destino não aceita o pedido para iniciar uma sessão.

2.2.3 Sinalização

Um exemplo do fluxo básico de comunicação entre dois usuários de um serviço VoIP é ilustrado na Figura 2.2. Uma sessão é iniciada quando o agente de Alice (chamador) envia

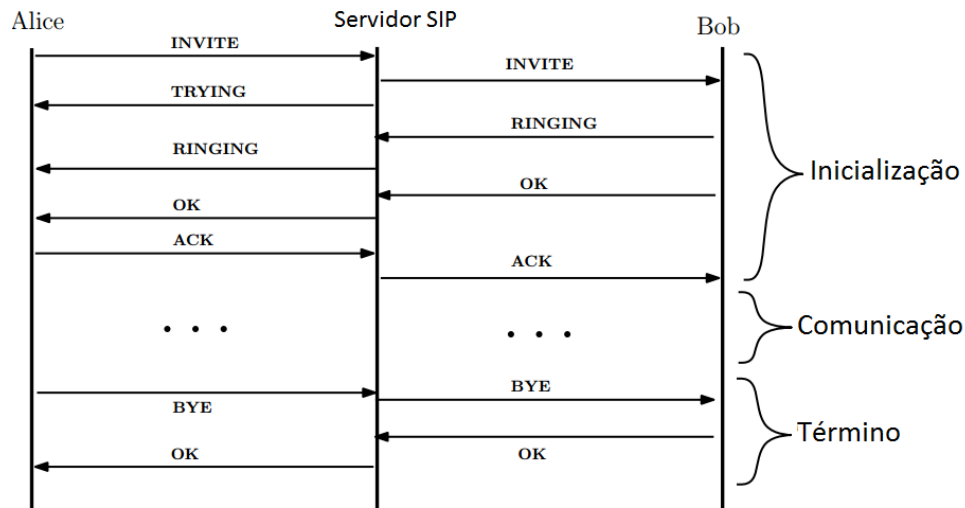


Figura 2.2: Exemplo de uma sessão multimídia SIP.

uma mensagem INVITE para o servidor SIP, indicando que Alice deseja se comunicar com Bob (receptor). Ele então verifica o registro de Alice no banco de dados e a localização de Bob. Em seguida, a mensagem INVITE é encaminhada para Bob e o servidor envia uma mensagem Trying (100) para Alice. O agente de Bob envia uma resposta Ringing (180) quando o telefone começa a tocar. Se o mesmo decidir atender a chamada, o agente dele envia uma mensagem OK (200). Finalmente, o agente de Alice envia uma mensagem ACK e a **sessão é estabelecida**.

Nesse ponto, Alice e Bob começam a se comunicar por meio de fluxos de mídia trocados entre os agentes de Bob e Alice (isto é representado pelas reticências na Figura 2.2). A chamada é então terminada quando uma das partes (Alice) envia uma mensagem BYE para o servidor, que a encaminha para a outra parte (Bob), que, em seguida, responde com a mensagem OK, que é encaminhado para Alice.

2.3 Ataques de Negação de Serviço

Ataques de Negação de Serviço (DoS - *Denial of Service*) são ataques que objetivam interromper a disponibilidade de um serviço da rede, fazendo com que usuários legítimos não possam acessá-lo. Quando estes ataques possuem uma natureza distribuída (múltiplas fontes), são chamados de DDoS (acrônimo para *Distributed Denial of Service*). Em geral,

ataques DDoS são realizados por um conjunto de *hosts*¹ controlados pelo atacante. Além disso, ataques nos quais endereços IP de origem são mascarados para escapar à detecção podem ser considerados um tipo de ataque distribuído.

Ataques DDoS podem ocorrer na camada de transporte e na camada de aplicação do modelo TCP/IP (Socolofsky 1991).

Quando o DDoS acontece na camada de transporte, o atacante tenta tornar o serviço indisponível por meio do consumo de toda a largura de banda e outros recursos da rede. Para isso, o atacante inunda o serviço com incontáveis requisições, como no SYN *flooding attack* (TCP) e RTP *flooding attack* (UDP) (Zargar, Joshi e Tipper 2013). Apesar de ainda perigosos, a grande quantidade de tráfego gerada pelo excessivo número de requisições faz com que esses ataques possam ser facilmente identificados por meio da análise do tráfego.

No DDoS que sucede na camada de aplicação, o alvo pode ser uma única aplicação (ex: servidor SIP) hospedada em uma máquina da rede. Nesse caso, o atacante objetiva esgotar algum recurso (ex: memória, *sockets*) da máquina hospedeira. Ao explorar os protocolos (ex: HTTP, DNS, SIP) usados pela aplicação alvo, o atacante é capaz de atingir seu objetivo usando uma quantidade bem menor de requisições, consumindo bem menos largura de banda e consequentemente tornando o ataque silencioso e impossível de ser detectado apenas analisando o tráfego.

Exemplos de ataques DDoS na camada de aplicação incluem o Slowloris (slowloris 2013), POST (yet 2013) e Slowread (slowread 2013), direcionados ao HTTP, e o SIP *flooding* (Zargar, Joshi e Tipper 2013), *Dial Attack* (Kapravelos et al. 2010) e *Coordinated Call Attack* (Securelogix 2013), direcionados ao VoIP.

Diferente do SIP *flooding*, os ataques *Coordinated Call Attack* e *Dial Attack* não exploram alguma fraqueza do protocolo SIP. Os atacantes seguem corretamente o protocolo, enviando as mensagens esperadas para estabelecer uma chamada (como descrito na Seção 2.2.3). Nesses ataques, são as chamadas maliciosas dos atacantes que provocam a indisponibilidade no serviço. Devido a esta característica, o *Coordinated Call Attack* e o *Dial Attack* também são chamados de ataques de Negação de Serviço de Telefonia (TDoS - *Telephony Denial of Service*) (Securelogix 2013).

¹Geralmente, esses *hosts* estão infectados por algum tipo de *malware* que os obriga a realizar tarefas para o atacante.

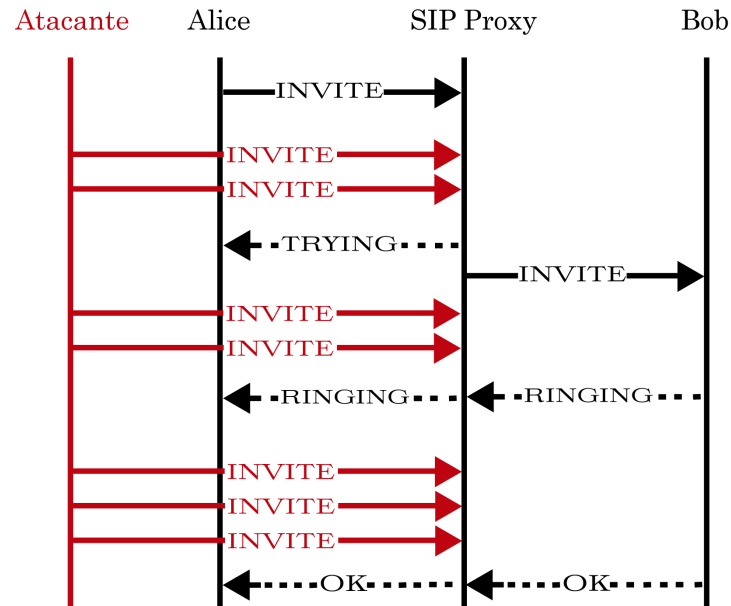


Figura 2.3: Ataque SIP *flooding* usando mensagens INVITE.

A seguir, são descritos brevemente cada um dos ataques mencionados que objetivam tornar indisponível um serviço VoIP. No Capítulo 3 é apresentado uma estratégia seletiva que pode ser utilizada para mitigar o *Coordinated Call Attack*.

2.3.1 SIP Flooding

O SIP *flooding* funciona de maneira similar a outros ataques de inundação realizados na camada de transporte. O servidor *proxy* (ou outro componente) é inundado com incontáveis requisições INVITE/REGISTER em um curto período de tempo. Ele fica tão ocupado processando as requisições do atacante que se torna incapaz de processar aquelas enviadas por usuários legítimos.

O ataque segue o padrão mostrado na Figura 2.3. No SIP, uma sessão é estabelecida com um aperto de mão de três vias (INVITE, OK e ACK). Portanto, cada mensagem INVITE cria um estado no servidor SIP, consumindo memória. Explorando esse fato, o atacante utiliza um fluxo rápido de mensagens INVITE para consumir toda memória disponível, fazendo com que as requisições legítimas não possam ser atendidas.

2.3.2 *Dial Attack*

No *Dial Attack*, o atacante injeta um grande volume de chamadas direcionadas a um serviço de telefonia alvo, como *call centers*, órgãos governamentais e serviços de emergência. A Figura 2.4 ilustra a abordagem seguida pelo atacante no *Dial Attack*. O ataque é comumente realizado usando ferramentas VoIP (Carman 2009) que permitem a realização de chamadas de forma automática usando alguma taxa pré-definida (normalmente 3 chamadas por segundo). Uma vez que a chamada é feita e a vítima atende o telefone, o atacante reproduz algum áudio pré-definido. Percebendo que esta é uma falsa chamada (trote), o atendente desliga o telefone. No entanto, uma vez que o número de chamadas é muito grande, o telefone toca novamente. Como as linhas estão ocupadas, os clientes legítimos não podem ser atendidos.

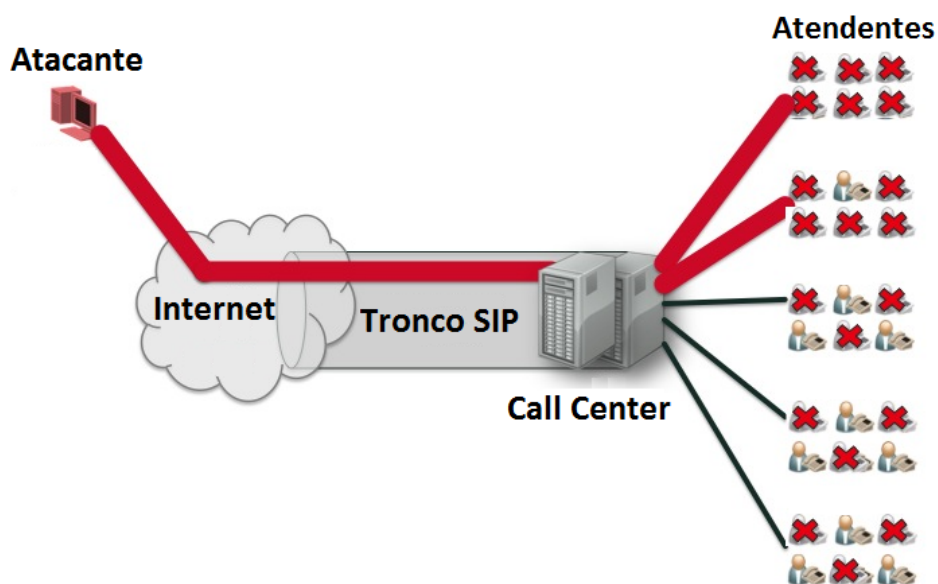


Figura 2.4: *Dial Attack*

Contudo, diferentemente dos ataques de inundação habituais, onde o objetivo é simplesmente negar serviço, o *Dial Attack* almeja não apenas ocupar as linhas da vítima, mas também exaustar os atendentes do serviço alvo. Tal comportamento aumenta o poder de barganha do atacante, para, por exemplo, chantagear os responsáveis pelo serviço. Portanto, o objetivo do atacante pode ser alcançado mesmo se o serviço for apenas parcialmente negado. Construir defesas contra o *Dial Attack* é duplamente desafiador: quando sob ataque, a defesa deve manter alta disponibilidade e evitar que as falsas chamadas cheguem ao atendente.

2.3.3 Coordinated Call Attack

Um par de agentes do utilizador maliciosos ou comprometidos, A_1 e A_2 , que estão registrados no serviço VoIP (ou dois usuários honestos que tiveram seus dispositivos infectados pelo atacante), realizam uma chamada entre si e mantêm esta chamada ativa pelo máximo de tempo que puderem. Os agentes trocam com o servidor as mensagens necessárias para inicializar a chamada, como em uma sessão normal do SIP. Uma vez que a chamada é estabelecida, os atacantes permanecem na chamada por tempo indeterminado. Eles podem ser desligados por algum mecanismo de *timeout* que estabelece um limite (amplo) de tempo para a duração da chamada. Durante o tempo que A_1 e A_2 estão se comunicando, eles estão usando recursos do servidor. Como servidores VoIP habituais têm recursos limitados, estes só podem manter uma quantidade finita de chamadas simultâneas antes que interrupções no serviço comecem a ocorrer. Utilizando um número suficiente de pares de agentes do utilizador, o servidor VoIP torna-se alvo do que é chamado *Coordinated Call Attack*.

A Figura 2.5 ilustra o motivo deste ataque ser tão eficaz. O atacante pode lentamente ocupar os *slots* de chamadas disponíveis no servidor SIP. Dado que os atacantes permanecem na chamada por tempo indefinido, uma vez que um *slot* é ocupado, ele nunca será liberado (sem algum mecanismo de defesa) e, portanto, em um dado momento, o atacante estará ocupando todos os *slots*, impedindo que os clientes legítimos usem o serviço VoIP.

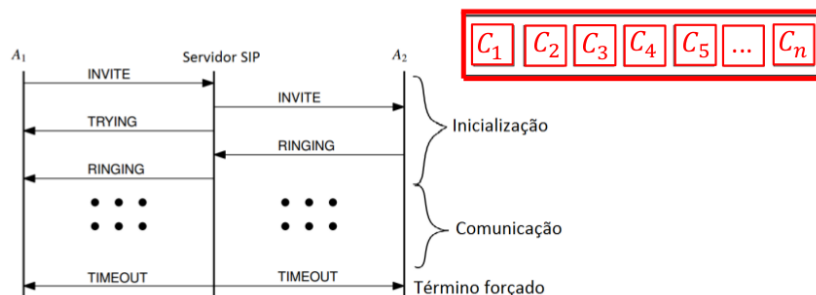


Figura 2.5: Ilustração do *Coordinated Call Attack*. Os *slots* representam recursos sendo usados pelas chamadas $C_1, C_2, C_3, C_4, C_5, \dots, C_n$.

Uma diferença importante entre o *Coordinated Call Attack* e o *SIP flooding* é o fato de que o tráfego do atacante não é significativamente diferente do tráfego legítimo de costume. O atacante não tem de gerar um grande número de chamadas a uma instância específica. Ele pode produzir as chamadas a uma taxa semelhante a de clientes habituais, de modo que o

tráfego total corresponda ao tráfego esperado e nenhum mecanismo de defesa seja ativado. Como demonstrado na Seção 4.4, mesmo uma taxa relativamente baixa de chamadas pode reduzir a disponibilidade do serviço para níveis próximos de 6%.

O atacante dispõe de muitas possibilidades para que um agente do utilizador se torne participante de um *Coordinated Call Attack*. Primeiramente, pares de usuários descontentes podem conspirar para prejudicar a reputação da empresa que presta o serviço. Atacantes também podem utilizar uma *botnet* de dispositivos VoIP infectados por algum tipo de *malware*. Evidências da utilização de redes VoIP para construir *botnets* já foram relatadas em Hines 2007. Tarifas fixas de baixo custo oferecidas por provedores VoIP e ferramentas que permitem a geração e recepção de diversas chamadas concorrentes de maneira automática (SIPp (GAYRAUD e JACQUES 2014)) são outros fatores que facilitam a concepção do *Coordinated Call Attack*.

2.4 Métodos Formais

2.4.1 Lógica de Reescrita

A Lógica de Reescrita (RL - *Rewriting Logic*) (Meseguer 2012) é um modelo natural de computação e um expressivo *framework* semântico para concorrência, paralelismo, comunicação e interação. Ela pode ser utilizada para especificar uma vasta gama de sistemas em várias áreas de aplicação. O presente trabalho foca nos aspectos dinâmicos da Lógica de Reescrita, empregando-a como um mecanismo para formalizar defesas contra ataques DDoS (Agha et al. 2005, AlTurki, Meseguer e Gunter 2009, Eckhardt et al. 2012, Dantas, Nigam e Fonseca 2014). Esse tipo de formalização oferece algumas vantagens:

- Nível de especificação bastante detalhado;
- Prototipação rápida;
- Depuração executando diretamente as especificações do mecanismo utilizado.

Teorias de reescritas são compostas por tuplas $\langle \Sigma, E, R \rangle$, onde Σ é uma assinatura que define a sintaxe do sistema e de seus estados, E é um conjunto de equações que definem os

estados do sistema como um tipo de dados algébrico e R é um conjunto de regras de reescrita da forma:

$$t \rightarrow t'$$

especificando as transições no sistema. O lado esquerdo t descreve um padrão de ativação, e o lado direito t' descreve um modelo de substituição correspondente. Isto é, qualquer fragmento de um estado que é uma instância do padrão de ativação pode executar uma transição na qual é substituída pela instância correspondente do padrão de substituição t' .

Para ilustrar como a Lógica de Reescrita funciona na prática, a seguir é formalizado um dos problemas de planejamento mais populares na inteligência artificial, o *Blocks World* (Clavel et al. 2007), ilustrado pela Figura 2.6. Tal problema considera que existem alguns blocos, colocados sobre uma mesa, que podem ser movidos por um braço robótico, que tem por objetivo formar pilhas verticais desses blocos.

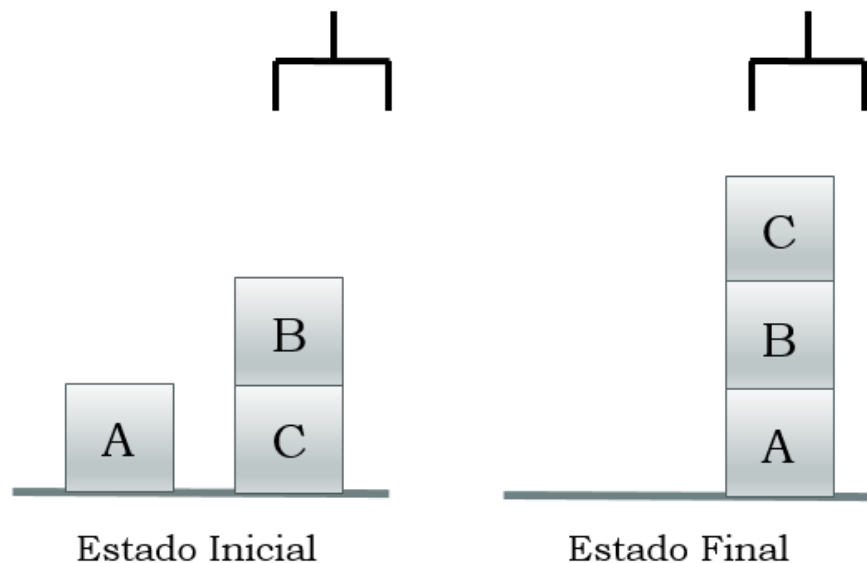


Figura 2.6: Exemplo do problema *Blocks World* com três blocos.

Após familiarizar-se com o problema, são descritos os predicados dos blocos, do robô e suas reescritas.

Predicados pertencentes aos blocos: Atribui-se três predicados aos blocos, dois com aridade um e outro com aridade dois:

OnTable(X): O bloco X está sobre a mesa;

Clear(X): O bloco X está livre, *i.e.*, não existe outro bloco acima dele;

On(X,Y): O bloco X está sobre o bloco Y.

Predicados pertencentes ao robô: É assumido que o robô possui dois predicados, um unário e outro sem argumento:

Hold (X): O robô está segurando um bloco X

Empty(): O robô está livre

Reescritas do Blocks World: A seguir são especificados quatro estados e suas transições utilizando os predicados descritos:

PickUp: Robô pega o bloco que está sobre a mesa;

Putdown: Robô solta o bloco sobre a mesa;

UnStack: Robô desempilha um bloco que está sobre outro;

Stack: Robô empilha um bloco sobre outro.

Assim, temos:

PickUp: $Empty \text{ and } Clear(X) \text{ and } OnTable(X) \rightarrow Hold(X);$

Putdown: $Hold(X) \rightarrow Empty \text{ and } Clear(X) \text{ and } OnTable(X);$

UnStack: $Empty \text{ and } Clear(X) \text{ and } On(X, Y) \rightarrow Hold(X) \text{ and } Clear(Y);$

Stack: $Hold(X) \text{ and } Clear(Y) \rightarrow Empty \text{ and } Clear(X) \text{ and } On(X, Y).$

Na primeira regra (**PickUp**), é verificado se o robô e o bloco estão livres e se o bloco está sobre a mesa. Após os três predicados serem verdadeiros, o robô pega o bloco X. A segunda reescrita é o inverso da primeira, o robô vai soltar um bloco X sobre a mesa. Na terceira (**UnStack**), a finalidade é desempilhar um bloco que está sobre outro, caso o robô esteja livre, o bloco X encontra-se sobre Y e não exista outro bloco sobre X, o robô pega X deixando Y livre. A quarta e última reescrita, é o oposto da anterior, o intuito é empilhar um bloco X sobre o bloco Y, caso não exista outro bloco sobre Y.

Sob algumas condições, regras de reescrita podem ser executadas em motores de reescrita, tal como o Maude, descrito na seção seguinte. Uma visão mais aprofundada sobre os fundamentos da Lógica de Reescrita pode ser encontrada em Meseguer 2012.

2.4.2 Maude

O Maude é uma ferramenta computacional de alto desempenho que dá suporte à especificação e programação de lógica equacional e de reescrita em uma ampla gama de aplicações.

```

mod BLOCKS-WORLD is
  protecting QID .
  sorts BlockId Prop State .
  subsort Qid < BlockId .
  subsort Prop < State .

  op table : BlockId -> Prop [ctor] .      *** O BLOCO ESTÁ NA MESA
  op on : BlockId BlockId -> Prop [ctor] . *** O BLOCO A ESTÁ SOBRE O BLOCO B
  op clear : BlockId -> Prop [ctor] .      *** O BLOCO ESTÁ LIVRE
  op hold : BlockId -> Prop [ctor] .      *** O ROBO ESTÁ SEGURANDO UM BLOCO
  op empty : -> Prop [ctor] .             *** O ROBO ESTÁ LIVRE
  op 1 : -> State [ctor] .
  op _&_ : State State -> State [ctor assoc comm id: 1] . *** AND LÓGICO

  vars X Y : BlockId .

  rl [pickup] : empty & clear(X) & table(X) => hold(X) .
  rl [putdown] : hold(X) => empty & clear(X) & table(X) .
  rl [unstack] : empty & clear(X) & on(X, Y) => hold(X) & clear(Y) .
  rl [stack] : hold(X) & clear(Y) => empty & clear(X) & on(X, Y) .
endm

```

Figura 2.7: Formalização do problema *Blocks World* na ferramenta Maude.

A Figura 2.7 apresenta a especificação formal do problema *Blocks World* implementado no Maude.

A primeira informação que uma especificação formal precisa declarar são os tipos (chamados de *sort*) dos dados que estão sendo definidos e as operações correspondentes. No Maude, *sorts* podem ser organizados em hierarquias (*subsort*). A relação entre *sort* e *subsort* é paralela à relação de conjunto e subconjunto da álgebra. *Sort* e *subsort* são declarados usando a seguinte sintaxe:

```

sort <Sort-1> .
subsort <Sort-2> < <Sort-1> .

```

Um operador é declarado com a palavra-chave *op* seguido de seu nome, dois pontos, uma lista de *sorts* para seus argumentos, seguido por $\rightarrow$, pelo tipo de seu resultado e opcionalmente seguido por uma declaração de atributo. Esses operadores podem ter atributos como: associativo [*assoc*], comutativo [*comm*], identidade [*id*] e construtores [*ctor*]. Assim, o esquema geral para declaração de operadores tem a forma:

```

op <Nome> : <Sort-1> ... <Sort-k> -> <Sort> [<Atributos>] .

```

Há também uma notação alternativa para os operadores, utilizando sublinhados (), onde os argumentos são destinados a substituí-los durante a etapa de verificação.

Variáveis são restritas a um determinado *sort* e podem ser definidas de duas maneiras:

```
var <Identificador> : <Sort> .
vars <Identificador-1> ... <Identificador-k> : <Sort> .
```

As regras de reescrita podem ser do tipo incondicional e condicional. Matematicamente, uma regra de reescrita incondicional tem a forma $l : t \rightarrow t'$, onde t e t' são termos e l é o rótulo da regra. Uma regra incondicional é introduzida em Maude com a seguinte sintaxe:

```
rl [<Rotulo>] : <Termo-1> => <Termo-2> [<Atributos>] .
```

As regras de reescrita condicionais podem ter condições muito gerais envolvendo equações, associações e outras reescritas. Em sua representação Maude, as regras condicionais são declaradas com sintaxe:

```
crl [<Rotulo>] : <Termo-1> => <Termo-2>
  if <Condicao-1> ... <Condicao-k> [<Atributos>] .
```

Por fim, equações são definidas da seguinte forma:

```
eq <Termo-1> = <Termo-2> [<Atributos>] .
```

Após formalizado, um modelo pode ser executado a partir de um estado inicial e verificado usando, por exemplo, o PVeStA, abordado na próxima seção.

2.4.3 PVeStA

PVeStA (acrônimo para *A Parallel Statistical Model Checking and Quantitative Analysis Tool*) é uma extensão da ferramenta VeSta, originalmente usada para verificação estatística de modelos e análise quantitativa de sistemas probabilísticos. PVeStA emprega uma abordagem cliente-servidor onde amostras aleatórias são computadas em paralelo com a tarefa de realizar simulações de Monte Carlo (Landau e Binder 2005).

PVeStA pode ser utilizado para verificar teorias de reescritas probabilísticas expressas em Maude. Esse é o motivo principal para sua utilização neste trabalho, pois PVeSta possibilita que as reescritas tenham um comportamento estocástico. Deste modo, agora as reescritas podem ter a seguinte forma:

$$t \rightarrow t' \quad \text{com probabilidade } P$$

O termo P modifica o comportamento das reescritas determinísticas, tornado-as estocásticas.

Além da verificação de modelos, PVerStA suporta avaliação estatística dos valores esperados para expressões escritas em uma linguagem de consulta denominada QuaTEx (Agha et al. 2005). "Qual é a fração esperada de clientes que se conectam com êxito a um servidor sob ataque DoS?" é um exemplo de consulta que pode ser codificada em QuaTEx.

O valor esperado de uma expressão QuaTEx é estatisticamente avaliado em relação a dois parâmetros, α e δ , fornecidos como entrada. Especificamente, o valor esperado é calculado por amostragem até que o tamanho do intervalo de confiança de $100\%(1 - \alpha)$ para o valor esperado seja limitado por δ . Uma visão mais detalhada sobre o QuaTEx é apresentada em Agha et al. 2005.

2.5 Trabalhos Relacionados

O objetivo principal desta seção é apresentar o estado da arte dos métodos para detecção e mitigação de ataques DDoS a serviços VoIP. Em levantamento bibliográfico realizado, foram considerados apenas os trabalhos que abordam os ataques descritos na seção anterior. Alguns trabalhos que empregam o uso de métodos formais para avaliar defesas contra ataques DDoS também são contextualizados.

2.5.1 Defesas contra o SIP Flooding

Ataques *SIP flooding* geram uma quantidade significativa de tráfego na rede. Portanto, uma ideia natural para detectar tais ataques é identificar variações abruptas no volume de tráfego da rede. Diferentes métodos estatísticos são usados em Ha et al. 2009, Tang et al. 2014, Jeyanthi, Thandeeswaran e Vinithra 2014, Zargar e Moghaddam 2014, Zhang et al. 2009, Tang, Cheng e Zhou 2009, Tang, Cheng e Hao 2012, Li et al. 2010 para modelar o fluxo de tráfego normal da rede. Se o comportamento observado desviar significativamente desse modelo, então suspeita-se que o sistema está sob ataque.

Os autores Ha et al. 2009 coletam atributos do tráfego SIP de modo a classificar seu comportamento. Esses atributos foram definidos de acordo com três critérios: comportamento do usuário, comportamento da chamada e estado da rede/servidor. Quando um fluxo

é classificado como suspeito, um mecanismo de detecção de anomalia baseado em um limiar pré-definido é aplicado para confirmação do ataque.

Em Jeyanthi, Thandeeswaran e Vinithra 2014, é empregado um modelo matemático, denominado Análise de Quantificação de Recorrência (RQA), para monitorar de forma contínua o tráfego da rede e detectar qualquer desvio do comportamento normal do tráfego. RQA é uma técnica para analisar o comportamento dos dados de tráfego não-lineares. Ela usa diferentes parâmetros, como taxa de recorrência, entropia, determinismo e etc. A partir dos dados de tráfego, esses parâmetros são calculados. Quando os valores obtidos diferem dos valores normais, o ataque é detectado. Uma vantagem dessa abordagem é que o modelo matemático garante que a estratégia nunca será comprometida, mesmo que os atacantes tenham conhecimento dela.

Um algoritmo de soma acumulativa (CUMSUM) é usado em Zhang et al. 2009 para comparar os parâmetros do tráfego observado com o tráfego normal da rede. Este algoritmo emprega uma análise estatística sobre a correlação entre o número de tentativas de estabelecimento de uma chamada (INVITE) e os apertos de mão de três vias concluídos (INVITE/200 OK/ ACK).

Os autores Tang, Cheng e Zhou 2009, Tang, Cheng e Hao 2012, Tang et al. 2014, Akbar, Basha e Sattar 2015 detectam e mitigam ataques combinando a técnica de *sketch* e Distância de Helling. A técnica de *sketch* sumariza o tráfego da rede em um conjunto de dados compacto e de tamanho constante. A Distância de Helling é uma medida estatística que mensura o desvio entre duas distribuições de probabilidade.

Com base nos dados armazenados no *sketch*, os autores Tang, Cheng e Zhou 2009 obtêm uma distribuição de probabilidade para o tráfego da rede em um determinado momento. A evolução da distribuição de probabilidade é monitorada pela Distância de Helling, onde o alerta de ataque é dado quando a variação da distribuição excede um limiar. Em adicional, é proposto um esquema para evitar que o limiar, que é atualizado em tempo real, seja impactado pelos ataques.

Em Tang, Cheng e Hao 2012, a defesa é aperfeiçoada para lidar com casos em que são usados mais de um tipo de mensagem para compor o ataque. Para isso, eles propõem o uso de um *sketch* tridimensional que permite que anomalias sejam analisadas separadamente para cada tipo de mensagem. Ademais, é proposta uma estratégia para mitigar o ataque.

Primeiro, as mensagens que compõem o ataque são identificadas por meio da observação de variações bruscas na função massa de probabilidade de cada entrada do *sketch*. Em seguida, estas mensagens são seletivamente descartadas para evitar a continuação do ataque.

Os autores Tang et al. 2014 propõem que seja mantida no SIP *proxy* uma distribuição dos dados do *sketch* para o tráfego normal da rede. Quando um usuário legítimo envia um pedido para se registrar no serviço VoIP, o SIP *proxy* define sua posição no *sketch* de acordo com a distribuição escolhida. Uma chave é enviada para o usuário, que é utilizada em suas futuras requisições para recuperar sua posição no *sketch*. Como os atacantes não tem conhecimento dessas chaves, suas posições no *sketch* são aleatórias, fazendo com que a distribuição resultante seja a uniforme ao invés da distribuição escolhida.

Em Akbar, Basha e Sattar 2015 a estratégia *sketch* e Distância de Helling é combinada com outros recursos para mitigação de DDoS presentes em alguns balanceadores de carga. Os autores Zargar e Moghaddam 2014 empregam a variação da entropia sobre o *sketch* para identificar um ataque. Observando as células do *sketch* que mais contribuíram para a modificação no valor de entropia, é possível identificar as mensagens dos atacantes e descartá-las.

Máquinas de estado que modelam as transações realizadas por um SIP *proxy* são utilizadas em Ehlert et al. 2008 para detectar fluxos de tráfego comuns em ataques SIP *flooding*. As máquinas de estado monitoram o tráfego em três diferentes escopos: transação, usuário e global. O monitoramento global é usado para detecção, enquanto que os demais escopos proveem informações para identificar a fonte do ataque. Essas informações podem ser usadas por um *firewall* de rede para decidir se uma requisição pode passar por ele ou se ela deve ser rejeitada.

Os autores Hussain et al. 2013 introduzem um mecanismo simples e computacionalmente barato para combater um ataque de inundação ao servidor SIP. Um limiar específico do usuário, denominado número crítico, controla a quantidade de pacotes que podem ser enviados ao usuário pelo servidor. O número crítico é configurado no momento do registro do usuário na rede. Uma vez que o número crítico para um determinado usuário é atingido, o SIP *proxy* coloca as futuras requisições em uma lista de espera.

Em Zi-Fu, Jun-Rong e Xiao-Yu 2010 é empregado um esquema de filas de prioridade para mitigar ataques SIP *flooding* que fazem uso de mensagens INVITE. As requisições com INVITE são postas em filas de prioridade e só são processadas após as demais. Quanto

mais requisições INVITE forem recebidas, menor a classe de prioridade atribuídas a elas. Quando a prioridade atribuída extrapola uma faixa de prioridades pré-definida, a mensagem é descartada.

A técnica de mitigação apresentada em Bansal e Pais 2015 foi baseada na análise de chamadas e mensagens SIP. Os autores assumem que um usuário legítimo envia de uma a duas mensagens INVITE quando estes querem iniciar uma chamada. Para gerar outra chamada, o usuário primeiro se desconecta da chamada permeável, logo uma mensagem do tipo BYE deve passar pelo servidor. Entretanto, um atacante envia múltiplos INVITE sem desconectar as chamadas. Portanto, ao se contar a quantidade de mensagens INVITE e BYE vindas de um certo IP e porta, é possível identificar os atacantes e mitigar o ataque.

Soluções baseadas em filtros são apresentadas em Stanek e Kencl 2012, Huici, Niccolini e d’Heureuse 2009, Geneiatakis, Vrakas e Lambrinoudakis 2009.

Os autores Stanek e Kencl 2012 propõem uma arquitetura composta de filtros de tráfego de entrada e saída e de um servidor de redirecionamento. Para alcançar o SIP *proxy*, uma requisição enviada passa por três estágios: um servidor de redirecionamento, um *firewall* com regras de limitação de tráfego e um servidor NAT. O servidor de redirecionamento é usado para alterar virtualmente a localização do SIP *proxy* de acordo com o endereço IP e porta do requisitante. O *firewall* monitora a taxa de requisições dos clientes em um intervalo de tempo recente. O servidor NAT verifica se o endereço IP e porta de destino são aqueles fornecidos pelo servidor de redirecionamento. Além disso, é verificado se o número de requisições direcionadas ao SIP *proxy* excede um limiar pré-definido. Em caso afirmativo, as requisições são descartadas.

Em Geneiatakis, Vrakas e Lambrinoudakis 2009, os autores propuseram uma solução baseada em Filtros de Bloom. Um Filtro de Bloom é um vetor que armazena elementos indexados por funções de *hash*. Três Filtros de Bloom são usados para verificar se o número de mensagens INVITE recebidas é igual ao número de apertos de mão de três vias (INVITE, OK, ACK) estabelecidos.

Uma arquitetura distribuída de filtros é projetada em Huici, Niccolini e d’Heureuse 2009 para defender o servidor contra grandes ataques de inundação. Tal arquitetura permite que as vítimas enviem solicitações de filtragem para pontos de filtragem denominados controladores SIP, que são postos perto da fonte do ataque.

2.5.2 Defesas contra o *Dial Attack*

Fuchs et al. 2008 aplica técnicas de detecção de anomalia para proteger centrais de atendimento públicas contra TDoS originados em uma rede VoIP. Especificamente, os autores analisam traços de chamadas para determinar o nível de ligações provenientes das redes PSTN, GSM e VoIP. A partir da análise, perfis são traçados e utilizados para discriminar os ataques baseados em VoIP, limitando o número de chamadas que podem ser originadas desse domínio, com base em trabalhos anteriores que identificaram a rede de origem como um potencial discriminador.

Kapravelos et al. 2010 propõe uma defesa a ser implantada em provedores VoIP, afim de impedir que usuários abusem do serviço. O tráfego de entrada é monitorado em busca de usuários que iniciam um grande número de chamadas em um curto espaço de tempo. Após a detecção, todos os pedidos para iniciar chamadas vindos do suspeito são rejeitados ou um limite fixo de pedidos é imposto ao usuário de uma forma que seu tráfego corresponda ao de um usuário com comportamento legítimo.

Guri, Mirsky e Elovici 2016 sugerem algumas medidas para prever e mitigar ataques TDoS anônimos a centros de emergência 911. Uma delas é forçar o dispositivo a enviar um identificador confiável para rede. Esse identificador deve ficar armazenado numa região segura da memória, para que este não possa ser alterado por um *malware*. Outra abordagem proposta é implementar em cada dispositivo um *firewall* obrigatório onde componentes confiáveis de software de baixo nível são usados para identificar e bloquear atividades TDoS (ex: chamadas frequentes a um mesmo serviço).

Polakis, Kontaxis e Ioannidis 2011 aborda a possibilidade de uso de métodos de detecção de presença humana, como solicitar o pressionamento de dígitos ou outros tipos de *captcha*.

Devido aos efeitos devastadores que o *Dial Attack* pode causar à empresas e outras organizações, algumas defesas encontradas por meio de revisão bibliográfica são ferramentas comerciais (CISCO 2014, TransNexus 2015, PINDROP 2014).

Mecanismos de lista negra bloqueiam dispositivos de usuários que abusam do serviço (CISCO 2014, TransNexus 2015). Por exemplo, em TransNexus 2015 é apresentada uma ferramenta que pontua todas as requisições INVITE de acordo com o padrão de tráfego do destinatário, de modo a descartar aquelas potencialmente fraudulentas. O padrão de cada cliente é modelado por meio da aplicação de técnicas de aprendizagem por reforço sobre da-

dos do seu histórico de chamadas. Para classificar uma requisição como sendo parte de um *Dial Attack*, são analisados atributos como: endereço IP de origem, número do chamador, número do destinatário e dispositivo SIP utilizado.

Por fim, a ferramenta apresentada em PINDROP 2014 analisa amostras de áudio para diferenciar as chamadas fraudulentas das legítimas.

2.5.3 Métodos Formais e DDoS

A formalização dos ataques DDoS e suas defesas tem sido objeto de outros trabalhos. Em Meadows 1999, é proposto um *framework* que identifica ações em um protocolo e atribui custos (ex: custo computacional) para elas. Os custos gerados pela participação de clientes legítimos são comparados com custos incorridos pela participação de atacantes. Um ataque DDoS é caracterizado quando os esforços realizados pelo cliente excedem um dado limiar.

Outra abordagem é o uso de Lógica de Reescrita, que é o método empregado neste trabalho. Técnicas de reescrita foram aplicadas com sucesso na análise da disponibilidade contra ameaças DDoS. Exemplos disto na literatura incluem a análise de TCP SYN *floods* (Agha et al. 2005), e verificação de algumas das propriedades do protocolo de verificação seletiva adaptativa (ASV) (Khanna et al. 2008) contra ataques DDoS, ambas usando um modelo de compartilhamento de largura de banda entre atacantes e clientes. Estes trabalhos possuem uma abordagem similar, usando verificação estatística de modelos em conjunto com uma lógica temporal quantitativa (QuaTEx) para realizar análises acerca do desempenho de ataques e defesas.

Enquanto que em Agha et al. 2005, Khanna et al. 2008 são modelados ataques DDoS tradicionais que exploram protocolos sem estado nas camadas de transporte/rede do modelo TCP/IP, os autores de Dantas, Nigam e Fonseca 2014 formalizam ataques na camada de aplicação que exploram o protocolo HTTP. A contra-medida abordada é o SeVen, no qual conexões são descartadas ou interrompidas de acordo com um modelo de distribuição uniforme.

Em Shankesi et al. 2009 é apresentada uma nova técnica de verificação de modelos, denominada verificação de medidas, para analisar ataques de amplificação no protocolo SIP. A técnica baseia-se na ideia de definir medidas de custo não apenas em objetos individuais, como um invasor ou um servidor, mas também em todo o sistema de rede. A verificação de

modelos analisa se certas comparações de medidas que caracterizam um ataque de amplificação são possíveis ou não. Esse trabalho também emprega Lógica de Reescrita e Maude.

2.6 Considerações

Neste capítulo, foi mostrado que o VoIP não é uma única tecnologia, mas um conjunto delas, que juntas possibilitam a comunicação por voz sobre o protocolo de internet. Foi dado enfoque ao SIP, destacando os componentes da sua arquitetura e as principais características de suas mensagens, indicando quais mensagens são trocadas entre os diferentes componentes para inicializar e encerrar uma sessão.

No contexto dos ataques de negação de serviço, foram vistos três ataques que tem por alvo um serviço VoIP: *SIP flooding*, *Dial Attack* e *Coordinated Call Attack*. Entre esses três ataques, o *Coordinated Call Attack* é o que mais produz tráfego similar ao tráfego legítimo de costume. Isso faz com que ele seja o mais difícil de detectar e mitigar. Entretanto, o *SIP flooding* e o *Dial Attack* são bastante perigosos e já foram registrados diversas ocorrências desses dois ataques (Bode 2012, Messmer 2011, FBI 2014, DHS 2013).

Foram apresentados alguns conceitos introdutórios acerca de Lógica de Reescrita e como ela pode ser usada para formalizar sistemas com o auxílio da ferramenta computacional Maude. Após formalizado, o modelo pode ser verificado usando o PVeStA.

A verificação estatística de modelos pode ser usada para explorar em maior profundidade o impacto dos ataques DDoS e mecanismos de defesas em relação a medidas de desempenho, como a disponibilidade. O presente trabalho segue os passos dos autores de Agha et al. 2005, Khanna et al. 2008, onde se formaliza o sistema em Maude e se utiliza o verificador estatístico de modelos PVeStA para realizar análises quantitativas (QuaTE_x) do desempenho de ataques e defesas.

Tópicos relacionados a DDoS ao VoIP têm atraído muita atenção de pesquisadores, a fim de lidar com ataques mencionados nesse capítulo. Como resultado, várias soluções têm sido propostas para detectar e mitigar esses ataques.

Por meio do levantamento bibliográfico realizado, pode-se concluir que o estado da arte da detecção e mitigação de ataques *SIP flooding* adota estratégias de limitação de tráfego baseadas em limiares, no qual as mensagens recebidas são contadas de forma dependente ou

independente de suas fontes.

Alguns autores aplicam um limiar estático para distinguir uma situação de ataque de uma situação real (Ha et al. 2009, Jeyanthi, Thandeeswaran e Vinithra 2014, Hussain et al. 2013, Zi-Fu, Jun-Rong e Xiao-Yu 2010, Bansal e Pais 2015, Ehlert et al. 2008, Stanek e Kencl 2012, Geneiatakis, Vrakas e Lambrinoudakis 2009), enquanto que outros adotam um limiar dinâmico para refletir as variações do padrão de tráfego que podem ocorrer em diferentes períodos do dia (diurno contra noturno) e dias da semana (dias de trabalho contra feriados) (Tang, Cheng e Zhou 2009, Tang, Cheng e Hao 2012, Tang et al. 2014, Akbar, Basha e Sattar 2015, Zargar e Moghaddam 2014).

Uma limitação do uso de limiar estático é que a defesa precisa passar por uma etapa de treinamento antes de realmente ser testada em uma situação real. Um limiar dinâmico pode ser enganado por atacantes inteligentes que expandem gradualmente o tráfego do ataque, aumentando o limiar para acompanhar o aparente crescimento do tráfego legítimo.

Portanto, é possível concluir que a detecção e mitigação de ataques SIP *flooding* continua sendo um tema de pesquisa relevante e desafiador e que as técnicas atuais devem ser melhoradas para prover uma defesa mais robusta contra esses ataques.

Por fim, foram descritas algumas defesas contra o *Dial Attack*. Fuchs et al. 2008 e Kapravelos et al. 2010 defendem a imposição de limites no número de chamadas de usuários que abusam do serviço. Para dimensionar o tráfego do cliente, é utilizado seu IP de origem. Em muitos casos, é assumido que um endereço IP está associado a um único cliente. No entanto, tal hipótese não pode ser mantida quando vários clientes são multiplexados por trás de um NAT ou proxy SIP. Um ataque realizado por poucos clientes mal-intencionados pode resultar na restrição do tráfego para todas as solicitação por trás do endereço IP do roteador NAT ou proxy SIP.

As defesas comentadas por Guri, Mirsky e Elovici 2016 exigem o uso de componentes de hardware e software robustos contra investidas de *malware*. Ademais, os autores não detalham as defesas com profundidade e não discutem como os métodos poderiam ser implementados e implantados em um ambiente real.

As poucas soluções comerciais encontradas atuam como *firewalls* que monitoram todo tráfego de sinalização (CISCO 2014, TransNexus 2015) ou de mídia (PINDROP 2014) para classificar chamadas como legítimas ou fraudulentas. Tais ferramentas comerciais podem ser

caras para as pequenas empresas comprarem e manterem e requerem conhecimentos técnicos para implantação.

O *Coordinated Call Attack* emula o tráfego de clientes legítimos, não causando um aumento repentino e inesperado no tráfego da rede, de modo que todas as defesas que lidam com o SIP *flooding* não são eficazes para detectar e mitigar o *Coordinated Call Attack*. As defesas propostas por Fuchs et al. 2008 e Kapravelos et al. 2010 também empregam análise do tráfego, o que as tornam inadequadas para o contexto do *Coordinated Call Attack*. Um atacante pode transpassar a defesa apontada em PINDROP 2014 simplesmente injetando áudio pré-gravado no tráfego de mídia da chamada. Em relação à ferramenta apresentada em TransNexus 2015, o atacante pode engana-lá caso ele tenha controle dos dispositivos SIP de clientes legítimos e conheça seus padrões de tráfego.

No próximo capítulo é apresentada uma estratégia seletiva capaz de eficientemente mitigar o *Coordinated Call Attack* mesmo quando o ataque é realizado a partir dos dispositivos de usuários legítimos.

Capítulo 3

Estratégia Proposta para Mitigar Ataques às Aplicações VoIP

O objetivo deste capítulo é apresentar uma nova defesa contra ataques DDoS a serviços VoIP, bem como descrever a modelagem formal dessa defesa na ferramenta Maude em conjunto com simulações conduzidas no PVeStA para validá-la. A Seção 3.1 deste capítulo introduz uma nova estratégia seletiva que pode ser incorporada a arquiteturas VoIP para mitigar o *Coordinated Call Attack*. Na Seção 3.2, são destacados alguns pontos-chave da formalização em Maude. Na Seção 3.3, são explanados os resultados obtidos por simulação.

3.1 Estratégia Seletiva contra Ataques TDoS *Low-Rate*

Estratégias seletivas têm sido usadas para mitigar os ataques DDoS de camada de transporte (*High-Rate*) (Khanna et al. 2008, Khanna et al. 2012) e de aplicação (*Low-Rate*) (Dantas, Nigam e Fonseca 2014). Uma vantagem principal das estratégias seletivas é que, uma vez que são algoritmos probabilísticos, é muito mais difícil para um atacante furtivo evita-las quando comparadas com defesas determinísticas, ex., utilizando algum limiar, etc.

Uma estratégia seletiva é composta por duas funções:

Select : Request $\times$ State $\rightarrow$ bool

Drop : State $\rightarrow$ Request

na qual State é o conjunto de estados do sistema que está sendo defendido, o que normalmente inclui as requisições que estão sendo atendidas atualmente e seus metadados, como seus estados internos, o número de requisições recebidas, etc. Request é o conjunto de requisições, que, entre outros dados, contém um identificador único id .

A função Select especifica quando uma requisição de entrada deve ser selecionada para ser servida, e Drop especifica qual requisição que está sendo processada deve ser descartada. O que torna as estratégias seletivas poderosas é o fato de que as funções Select e Drop são **governadas por algumas funções de probabilidade** que podem depender do estado do sistema.

O algoritmo geral funciona da seguinte maneira. Assumindo que N_T é o número máximo de requisições que a aplicação pode manipular simultaneamente. Para os servidores web, N_T é o número de *workers* disponíveis para processar requisições, já no VoIP, N_T é o número de chamadas concorrentes que podem ser mantidas. Dado que uma requisição req_ν chegue à aplicação que tem estado atual $\mathcal{S}$ e seja N_S o número de requisições processadas no estado $\mathcal{S}$:

1. Se $N_S < N_T$, aceitar req_ν ;
2. Caso contrário, a capacidade máxima do aplicativo será atingida. Dado que $b = \text{Select}(\mathcal{S}, req_\nu)$:
 - (a) Se b tem valor "falso", então a defesa seletiva não seleciona a requisição req_ν para ser atendida e, portanto, ela é descartada e uma mensagem é enviada para o usuário correspondente informando que o serviço não está disponível;
 - (b) Caso contrário, se b tem valor "verdadeiro", então a defesa deve decidir qual das requisições parcialmente processadas deve ser descartada e substituída por req_ν . Seja $req_i = \text{Drop}(\mathcal{S})$. A requisição req_i é descartada de $\mathcal{S}$ e req_ν toma seu lugar.

Nota-se que as requisições (tanto de clientes como de atacantes) são descartadas **somente quando a capacidade máxima da aplicação é alcançada**. Isto significa que em condições normais, *i.e.*, não sob ataque, estratégias seletivas não geram falso positivo.

Observa-se também que o passo 2(b) pode descartar requisições legítimas que estão sendo processadas. Este é um mal necessário em ataques do tipo ADDoS *Low-Rate* onde é difícil distinguir o tráfego legítimo do cliente do tráfego do atacante.

Finalmente, ajustando adequadamente as distribuições de probabilidade empregadas por Select e Drop de acordo com o contexto do VoIP, é possível mitigar ataques *Low-Rate*, como o *Coordinated Call Attack*.

3.1.0.0.1 Estado do Sistema de Defesa: O estado do sistema de defesa tem a seguinte forma:

$$\mathcal{S} = \langle \mathcal{C}, \text{PMod} \rangle$$

em que $\mathcal{C} = \{\text{call}_1, \dots, \text{call}_n\}$ é o conjunto de chamadas sendo mantidas por um servidor SIP e PMod é um número natural indicando a taxa de chegada de novas chamadas uma vez que a aplicação tenha atingido sua capacidade máxima. Na prática, PMod é um contador que é incrementado sempre que $N_S = N_T$ e ocorre uma nova chamada.

Assume-se que as chamadas são representadas por tuplas da forma $\langle id, status, dur \rangle$, onde id é um identificador único da chamada, $status$ representa o estado atual da chamada (ex: tocando, atendida, etc) e $dur \in \mathbb{R}^*$ é um número não negativo que especifica a duração da chamada.

3.1.0.0.2 Funções da Estratégia Seletiva A função Select para uma nova chamada call_ν e estado $\langle \mathcal{C}, \text{PMod} \rangle$ é definida a seguir:

$$\text{Select}(\text{call}_\nu, \langle \mathcal{C}, \text{PMod} \rangle) \quad := \quad \text{Passo 1: Computar } \gamma \leftarrow \frac{N_T}{N_T + \text{PMod}};$$

Passo 2: Gerar um número aleatório $\beta \leftarrow]0, 1[$ com distribuição uniforme;

Paso 3: se $\beta \leq \gamma$, então retorne "verdadeiro";

Caso contrário retorne "falso".

Intuitivamente, a equação definida no Passo 1 é específica para penalizar novas chamadas de acordo com a taxa de tráfego de entrada, de modo que as chamadas sendo processadas não fiquem sendo constantemente substituídas. Assim, quanto maior a taxa de requisições de entrada armazenada em PMod, menor é a chance de selecionar uma nova chamada, ou seja, a probabilidade de aceitar novas chamadas diminui progressivamente quando o *buffer* contém N_T chamadas. Portanto, PMod precisa ser redefinido periodicamente para evitar que novas chamadas tenham chances muito baixas de serem processadas. Motivado pelo trabalho

de mitigação de ataques ADDoS (Dantas, Nigam e Fonseca 2014), PMod é redefinido a cada 100 ms.

Objetivando mitigar o *Coordinated Call Attack*, fez-se com que a função Drop dependa de dois parâmetros: (1) o estado da chamada e (2) e sua duração. Dois possíveis estados a serem atribuídos a uma chamada foram considerados:

- **WAITING:** Uma chamada é WAITING se a fase de inicialização da chamada do protocolo SIP (Figura 2.2) ainda não foi completada. Isto é, o chamador ainda está aguardando o respondedor atender a chamada.
- **INCALL:** Uma chamada é INCALL se o processo de inicialização de chamada foi completado e ambas as partes (chamador e respondedor) já estão se comunicando (ou simplesmente estão em uma chamada).

Assim, qualquer requisição INVITE assume o estado WAITING, e este pode mudar para INCALL uma vez que a parte de inicialização de chamada do SIP (OK e ACK) estiver concluída.

Em muitos casos, é verdade que interromper uma chamada existente é considerado mais danoso à reputação do serviço VoIP que apenas não permitir que um usuário realize uma nova chamada. Logo, é assumido que é preferível que um servidor SIP, quando sobrecarregado, rejeite requisições WAITING em vez de requisições INCALL referentes a chamadas de **não tão longa duração**. Tal comportamento pode ser modelado por meio da configuração adequada da função Drop. Para determinar se uma chamada está demorando muito, é assumido que o servidor sabe a duração média, t_M ¹, das chamadas.

A chance de uma chamada INCALL ser descartada aumenta exponencialmente de acordo com a função DropFactor baseada na distribuição de Poisson², uma vez que sua duração for maior que t_M . A Figura 3.1 representa aproximadamente como a chance de uma chamada ser interrompida aumenta de acordo com sua duração. A equação atual é da forma:

¹O valor de t_M pode ser obtido pelos logs de chamadas mantidos por um provedor VoIP.

²É usada a distribuição de Poisson porque tal distribuição é normalmente utilizada para modelar a chegada de chamadas telefônicas (Brown et al. 2002).

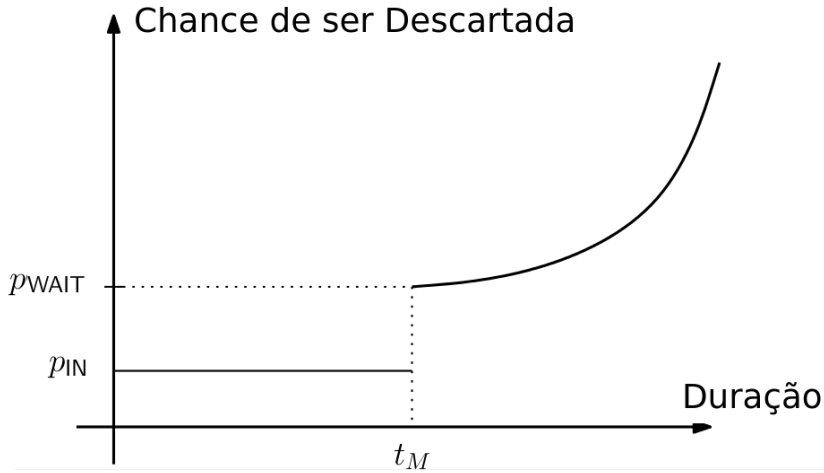


Figura 3.1: Gráfico ilustrando o comportamento da estratégia seletiva de acordo com o estado da chamada e sua duração. p_{WAIT} representa a chance de uma chamada com estado WAITING ser descartada, enquanto p_{IN} indica a chance de uma chamada com estado INCALL ser interrompida.

$$\text{DropFactor}(\text{call}) = \begin{cases} p_{WAIT} & \text{se } (\text{call} = \langle id, \text{WAITING}, dur \rangle) \\ p_{IN} & \text{se } (\text{call} = \langle id, \text{INCALL}, dur \rangle) \\ & \wedge (0 \leq dur \leq t_M) \\ p_{WAIT} + e^{\alpha \times dur / t_M} & \text{se } (\text{call} = \langle id, \text{INCALL}, dur \rangle) \\ & \wedge (dur > t_M) \end{cases} \quad (3.1)$$

Lembrando que chamadas com baixo valor de DropFactor são preferíveis para não serem descartadas. A tendência em descartar chamadas que estão em WAITING é constante, enquanto que a preferência em descartar chamadas nas quais os usuários já estão se comunicando, *i.e.*, em INCALL, não muda até a duração atingir t_M . Desse ponto em diante, DropFactor cresce exponencialmente, capturando assim a preferência de que chamadas de longa duração tendem a ser selecionadas. O parâmetro α na equação 3.1 indica se o termo exponencial cresce ($\alpha > 0$) ou decai ($\alpha < 0$) à medida que dur aumenta. Quanto maior o valor de α , mais rápido DropFactor aumenta. Nos experimentos conduzidos no presente trabalho, foi constatado que 2.55 é um bom valor para α .

Dada a função DropFactor, a função Drop, que seleciona qual das chamadas que estão sendo processadas será descartada, é definida como:

Drop($\mathcal{S}$) := Passo 1: Computar $\mathcal{T} \leftarrow \{\text{call}_1, \dots, \text{call}_k\}$ usando probabilidade uniforme, com $\mathcal{T} \subseteq \mathcal{S}$ e $|\mathcal{T}| = k$;
 Passo 2: Se $\text{DropFactor}(\text{call}_i) \geq \text{DropFactor}(\text{call}_j)$ para todo $\text{call}_j \in \{\mathcal{T} - \{\text{call}_i\}\}$, então retorne call_i .

De acordo com a definição de Drop, primeiro são selecionadas k chamadas usando probabilidade uniforme para fazer parte de um torneio (Blickle e Thiele 1995). Então, a chamada a ser descartada será aquela com o maior DropFactor entre as k chamadas escolhidas. Caso haja duas ou mais chamadas com o maior valor de DropFactor, uma delas é selecionada aleatoriamente. Por exemplo, considere $k = 2$ e que o servidor esteja atendendo as chamadas $\text{call}_1, \text{call}_2, \text{call}_3$ e call_4 , com DropFactor de 2, 3, 1 e 6, respectivamente. Serão escolhidas aleatoriamente duas dessas quatro chamadas. Suponha que call_2 e call_3 são selecionadas para participar do torneio. Nesse caso, a chamada call_2 é escolhida para ser descartada, pois tem o maior DropFactor. O tamanho do torneio, k , é um parâmetro que possui uma forte influência no comportamento do esquema de seleção. Se $k = 1$, então o torneio se torna equivalente ao descarte uniforme apresentada em Dantas, Nigam e Fonseca 2014, que não é adequado ao contexto de aplicações VoIP. Por outro lado, se k é escolhido para ser elevado demais, então o torneio se comporta mais próximo de uma estratégia determinística, que sempre seleciona a chamada com o maior DropFactor. De fato, se $k \gg N_T$, então a seleção ocorre de maneira determinística. Portanto, nas simulações e experimentos descritos no presente trabalho, foi empregado $k = N_T/2$, ou seja, uma estratégia entre a uniforme e a determinística. Uma vantagem do k -torneio para outros métodos, como a roleta (Lipowski e Lipowska 2011), é o seu baixo custo computacional. Tal característica faz do torneio o mecanismo de seleção mais usado na literatura (Noraini e Geraghty 2011) para problemas envolvendo algoritmos genéticos.

3.1.1 Exemplo de Execução

Considere-se que $N_T = 3$, PMod = 0 e que existem duas chamadas sendo processadas no instante de 9 unidades de tempo:

$$\mathcal{S}_0 = \langle \{ \langle \text{id}_1, \text{WAITING}, \text{undef} \rangle, \langle \text{id}_2, \text{INCALL}, 8.5 \rangle \}, 0 \rangle$$

O estado $\mathcal{S}_0$ especifica que o id_1 está esperando que seu convite para iniciar uma chamada seja respondido (com uma mensagem OK), enquanto que id_2 está atualmente em uma chamada. Suponha que uma mensagem associada a id é uma simples tupla $\langle id, M \rangle$, onde M representa o tipo da mensagem, e que a duração média de uma chamada é $t_M = 5$ unidades de tempo. Isso significa que id_2 já extrapolou essa medida.

Assumi-se que, no tempo 9.5, uma mensagem $\langle id_1, OK \rangle$ surge explicitando que a parte com que id_1 deseja se comunicar respondeu seu convite para iniciar uma chamada. O estado do sistema é atualizado da seguinte maneira:

$$\mathcal{S}_1 = \langle \{ \langle id_1, INCALL, 0 \rangle, \langle id_2, INCALL, 9.0 \rangle \}, 0 \rangle$$

com PMod ainda 0, pois a aplicação até então possui a capacidade de manter mais uma chamada. Então a mensagem $\langle id_3, INVITE \rangle$ chega. Uma vez que o número de chamadas em $\mathcal{S}_1$ é menor que $N_T = 3$, a aplicação pode processar com segurança esta requisição, levando ao estado:

$$\mathcal{S}_2 = \langle \{ \langle id_1, INCALL, 0 \rangle, \langle id_2, INCALL, 9.0 \rangle, \langle id_3, WAITING, undef \rangle \}, 0 \rangle$$

Considere-se agora que outra mensagem $m_1 = \langle id_4, INVITE \rangle$ chega no tempo 10.5. Desde que o número de chamadas em $\mathcal{S}_2$ é igual a $N_T = 3$, PMod é incrementado para 1 e a aplicação tem que decidir se vai processar m_1 , ou não. Ela usa a função Select, que gera um número aleatório no intervalo $]0, 1[$ e verifica se ele é menor ou igual a:

$$\frac{N_T}{N_T + \text{PMod}} = \frac{3}{3 + 1} = \frac{3}{4} \geq 0.75$$

Portanto, a nova requisição tem 75% de chance de ser selecionada para ser servida. Suponha que a nova requisição é selecionada. A defesa precisa, agora, decidir qual das chamadas sendo processadas deve ser descartada. Ela executa Drop, que realiza um torneio para escolher uma das chamadas:

- id_1 tem DropFactor = p_{IN} pois este possui uma duração (1.0 unidade de tempo) menor

que t_M ;

- id_2 tem boas chances de ser descartada, pois sua duração tem o dobro de t_M : (2×5) ;
- id_3 tem $\text{DropFactor} = p_{\text{WAIT}}$ pois este está com estado WAITING.

Assumindo que a defesa decide descartar id_2 , o que significa que a chamada é interrompida pela aplicação. O novo estado é:

$$\mathcal{S}_3 = \langle \{ \langle id_1, \text{INCALL}, 1 \rangle, \langle id_4, \text{WAITING}, \text{undef} \rangle, \langle id_3, \text{WAITING}, \text{undef} \rangle \}, 1 \rangle$$

3.1.2 Intuição, Limitações e Extensões

3.1.2.0.1 Intuição para a Eficácia de Estratégias Seletivas: Para melhor compreender o porque da estratégia seletiva apresentada ser eficaz contra o *Coordinated Call Attack* (como observado nas simulações e experimentos na rede), considera-se a Figura 3.2. Lembre-se que o objetivo desse tipo de ataque é ocupar por longos períodos de tempo os recursos de um servidor VoIP. Isso é representado pelas cruzes vermelhas na Figura 3.2. Uma vez que a aplicação está em sua capacidade máxima, é provável que ela esteja sob ataque, e é presumível que haja um maior número de atacantes consumindo recursos do que clientes legítimos fazendo o mesmo. Portanto, há uma maior chance da função Drop encerrar a chamada de um atacante. Isto é ilustrado na Figura 3.2 pelas linhas tracejadas verticais representando momentos em que a defesa precisa encerrar alguma chamada em andamento. Durante um ataque, o número de cruzes vermelhas que representam chamadas de atacantes sendo processadas é muito maior do que o número de pontos pretos que representam chamadas legítimas. Assim, a chance de interromper uma ligação maliciosa é maior.

3.1.2.0.2 Limitações da Estratégia Seletiva Proposta: Uma das consequências do uso do parâmetro PMod na função Select para penalizar novas chamadas de acordo com a taxa de tráfego de entrada é que um atacante pode potencialmente inundar a aplicação com muitas novas requisições INVITE em um curto espaço de tempo, de modo que o valor PMod ficasse suficientemente grande para fazer com que Select rejeite todos os novos pedidos para iniciar chamadas (tanto para clientes legítimos, quanto atacantes). No entanto, ataques baseados em

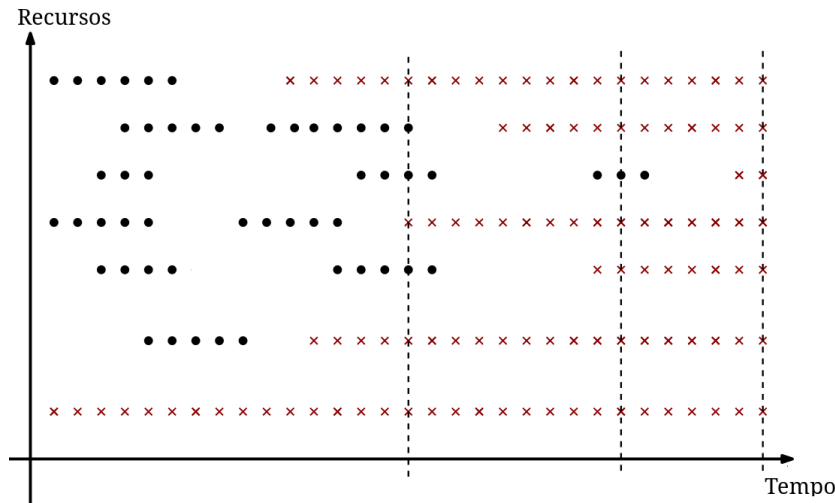


Figura 3.2: Ilustração da eficácia de estratégias seletivas contra o *Coordinated Call Attack*. Círculos (em preto) denotam clientes legítimos e as cruzes (em vermelho) denotam chamadas de atacantes.

inundação já foram amplamente abordados na literatura (Seção 2.5) e estão fora do âmbito do presente trabalho. Estratégias seletivas foram projetadas para evitar ataques DDoS do tipo *Low-Rate* na camada de aplicação. Os mecanismos de prevenção de ataques de inundação podem ser combinados com estratégias seletivas para prevenir coletivamente ambos os tipos de ataques.

Outra preocupação é que atacantes furtivos podem reduzir a duração das chamadas maliciosas objetivando aumentar as chances destas não serem selecionadas por Drop, devido às suas curtas durações ($dur \leq t_M$). No entanto, ao reduzir a continuidade da chamada para evitar a detecção, o atacante também reduz o tempo que ele fica usando os recursos do servidor VoIP, que é o objetivo desejado pela defesa.

Uma última preocupação é que as chamadas legítimas que estão se comunicando por um longo período de tempo podem ser erroneamente encerradas pela estratégia seletiva. No entanto, vale a pena lembrar que a estratégia seletiva só descarta chamadas quando a aplicação está operando em sua capacidade máxima. Quando o servidor VoIP experiencia uma sobrecarga de uso dos seus recursos locais, um efeito colateral da defesa proposta é que as chamadas consumindo mais recursos (*i.e.* chamadas onde as partes estão se comunicando por muito tempo) são mais suscetíveis de serem interrompidas. Pode-se argumentar que este comportamento da estratégia seletiva é desejável no que diz respeito à equidade. Outro argumento

a ser feito é que em uma situação de emergência, um cliente legítimo que teve sua ligação encerrada tende a re-discar para o mesmo destino a fim de dar continuidade à comunicação. Graças à estratégia seletiva, o cliente tem uma nova oportunidade de continuar falando por meio de uma nova chamada, sendo que esta nova ligação terá bem menos chances de ser descartada por um período razoável de tempo (t_M). Ademais, resultados experimentais e por simulação mostram que poucas chamadas legítimas foram equivocadamente encerradas.

3.1.2.0.3 Estratégias Seletivas Alternativas: Embora não esteja no escopo deste trabalho, são descritos a seguir outras possíveis estratégias seletivas que podem ser usadas, desde que se tenha mais conhecimento sobre o tráfego legítimo e do atacante. Uma análise mais aprofundada dessas estratégias é deixada para o trabalho futuro.

- **Estratégia baseada em Listas:** É possível incorporar listas negras em uma estratégia seletiva, especificando para quais IPs o serviço deve ser negado, e listas brancas, especificando para quais IPs o serviço deve ser fornecido. É fácil adaptar esses mecanismos na estratégia proposta. A defesa seletiva ajustará a probabilidade de uma nova chamada ser servida para 0, quando esta vier de um IP na lista negra, e 1 para as requisições na lista branca, ou seja, nunca selecionar clientes que estão em listas negras (mesmo se a aplicação não estiver em sua capacidade máxima) e sempre selecionar requisições que estão na lista branca. Além disso, a aplicação pode manter dois conjuntos de chamadas sendo processadas, S_W e S_N , contendo, respectivamente, as chamadas que estão vinculadas na lista branca e que não estão na lista branca. Então, Drop escolheria uma chamada a ser encerrada apenas no conjunto S_N . Se $S_N = \emptyset$, logo, Drop também escolheria chamadas do conjunto S_W . Desta forma, as chamadas na lista branca teriam maior chance de continuar a comunicação por todo o tempo desejado.
- **Estratégia baseada em Autenticação/Credenciais:** Se a aplicação possui algum mecanismo de autenticação ou algum sistema de credenciais (*i.e.*, classes de usuários), a estratégia seletiva pode incorporar esses sistemas para dar prioridade aos usuários com credenciais mais altas. Isso é feito de forma semelhante à descrita para a incorporação de listas negras e brancas. A probabilidade de selecionar/manter chamadas de usuários

com credenciais mais elevadas pode ser definida como maior do que a probabilidade de selecionar usuários com credenciais mais baixas.

- **Estratégia baseada em Histórico:** A aplicação pode manter um registro de interação com os usuários e essas informações podem ser usadas para configurar uma estratégia seletiva de diferentes maneiras. Por exemplo, pode ser desejável priorizar os usuários que usaram menos o serviço do que os usuários que estão usando em excesso. Isso também pode ser incorporado na estratégia seletiva, estabelecendo uma maior probabilidade de selecionar/manter a chamada anterior do que as últimas chamadas.

3.2 Especificação Formal

As especificações formais executáveis mostraram-se capazes de refletir o comportamento real de sistemas probabilísticos complexos como as estratégias seletivas (Khanna et al. 2008, Dantas, Nigam e Fonseca 2014). Portanto, uma vez que foi definida uma estratégia seletiva adequada ao VoIP, é relevante conceber sua especificação formal, afim de obter uma maneira eficiente e economizadora de tempo para aperfeiçoar o projeto da defesa contra ataques *Low-rate* ao VoIP antes mesmo que ela seja implementada ou avaliada experimentalmente.

A modelagem formal da estratégia seletiva proposta segue aquela apresentada em Dantas, Nigam e Fonseca 2014, especificando cenários de teste usando atores onde atacantes, clientes e servidor enviam e recebem mensagens. Essas mensagens são armazenadas em um *scheduler* que mantém uma fila de mensagens. Os atacantes não assumem o controle sobre o canal. Em vez disso, eles compartilham um canal com os clientes.

Todos os atores foram formalizados no Maude e simulações foram conduzidas usando o verificador estatístico de modelos PVeStA. Por razões de simplicidade, assumamos que tanto a defesa quanto a aplicação defendida são representadas por um mesmo ator. Tal decisão não afeta a análise dos resultados, pois na prática, a defesa e o servidor VoIP estão na mesma máquina, compartilhando um canal de comunicação rápido. A seguir, alguns pontos-chave da especificação formal em Maude são discutidos. O modelo completo pode ser encontrado em (SeVen <<https://github.com/ygdantas/SeVen.git>> 2016).

3.2.1 Sorts e Funções

3.2.1.0.1 Actor Os elementos do tipo `Actor` são construídos pelo operador

```
op <name:_|_> : Address AttributeSet -> Actor .
```

que toma como argumento um `Address`, que pode ser um `string`, e um conjunto de atributos, `AttributeSet`. Foram considerados os seguintes atributos:

```
op req-cnt:_ : Float -> Attribute .
```

```
op b-set:_ : NBuffer -> Attribute .
```

```
op server:_ : Address -> Attribute .
```

```
op status:_ : Status -> Attribute .
```

o atributo `req-cnt` armazena o valor do `PMoD`, `b-set` guarda o *buffer* interno do servidor, do tipo `NBuffer` e `server` armazena o endereço do servidor. Finalmente, o atributo `status` especifica o estado da chamada, que pode assumir uma das seguintes constantes:

- `none` – uma chamada que ainda não foi realizada;
- `waiting` – uma chamada que foi realizada e está esperando que o destinatário atenda o telefone;
- `incall` – uma chamada onde os participantes estão se comunicando;
- `complete` – uma chamada que foi completada, *i.e.*, as partes conseguiram se comunicar por todo tempo desejado;
- `incomplete` – uma chamada que foi interrompida pela defesa seletiva.

Os três primeiros atributos (`req-cnt`, `b-set` e `server`) são de uso exclusivo pelo ator servidor, enquanto que `status` é empregado por ambos os três tipos de atores (servidor, clientes e atacantes).

O *buffer* da defesa tem tipo `NBuffer`, sendo que este é construído emparelhando um número e um *buffer*:

```
op [_|_] : Nat Buffer -> NBuffer .
```

O número indica os elementos no *buffer*, que nada mais é que uma lista de elementos do tipo `EleBuf`. Tais elementos são criados usando uma tripla da forma:

```
op <___> : Address State Float -> EleBuf .
```

A primeira posição armazena o endereço da chamada. Já a segunda posição, o estado da mesma. A terceira posição guarda o tempo da primeira mensagem e é usada para computar a duração da chamada.

3.2.1.0.2 Mensagens Atores processam mensagens do tipo `Msg`, que são instanciadas usando o operador

```
op _<_ : Address Contents -> Msg .
```

O primeiro parâmetro, do tipo `Address`, indica para qual ator a mensagem é endereçada e o segundo argumento, do tipo `Contents`, é o *payload* da mensagem. Um `ActiveMsg` é um *timestamp* para a `Msg`, construído usando o operador:

```
op {_,_} : Float Msg -> ActiveMsg .
```

onde o primeiro parâmetro especifica o tempo quando a mensagem deve ser processada.

É assumido que uma mensagem $\{gt1, msg1\}$ será processada antes de uma mensagem $\{gt2, msg2\}$ sempre que $gt1 < gt2$. Isto é conseguido usando um *scheduler*, como em Dantas, Nigam e Fonseca 2014, Eckhardt et al. 2012, Eckhardt et al. 2012. Um *scheduler* tem tipo `Scheduler` e é instanciado pelo seguinte operador:

```
op [_|_] : Float ActiveMsgList -> Scheduler .
```

sendo que o primeiro parâmetro é o tempo global enquanto o segundo argumento contém a lista de mensagens ativas ordenadas por seu tempo de entrega. A função a seguir retorna o novo estado do *scheduler*, obtido pela inclusão de uma lista de mensagens ativas:

```
op insert : Scheduler ActiveMsgList -> Scheduler .
```

3.2.1.0.3 Configuração Uma configuração do tipo `Config` é uma coleção de `Actors` e `Scheduler`:

```
subsort Actors < Config .
subsort Scheduler < Config .
op ___ : Config Config -> Config [assoc comm id: none] .
```

Por exemplo, a configuração inicial é definida pela equação:

```
eq initState =
<name: server | req-cnt: 0.0 , b-set: [0 | none], none >
<name: client-generate | server: server, cnt: 0 , none >
<name: attacker-generate | server: server, cnt: 0 , none >
  [0.0 | {0.0, (attacker-generate <- spawn )} ;
    {0.0, (client-generate <- spawn )} ;
    {Ts, server <- ROUND}}] .
```

que especifica uma configuração com três atores: um gerador de atacantes, um gerador de clientes e um servidor. O tempo global é `0.0` e há três mensagens ativas no *scheduler*, as duas primeiras direcionadas para os atores *attacker-generate* e *cliente-generate*, respectivamente. Essas mensagens, intuitivamente, criam um novo ator atacante e um novo ator cliente. A terceira mensagem, agendada no momento `Ts`, é direcionada ao servidor *server*, que executa a estratégia seletiva quando uma rodada termina (`0.1s`).

A função a seguir extrai a primeira mensagem ativa no *scheduler* e retorna um novo *scheduler* com o tempo global avançado para seu tempo de entrega:

```
op mytick : Scheduler -> Config .
```

Por exemplo, seja *msg* uma mensagem e *SL* um `ActiveMsgList`. Então

```
mytick([0.0 | {1.0,msg} ; SL])
```

retorna `msg [1.0 | SL]` contendo a mensagem *msg* e o *scheduler* `[1.0 | SL]`. Intuitivamente, a mensagem *msg* vai ser processado em seguida. O processamento de mensagens é formalizado por regras de reescrita.

3.2.1.0.4 Estratégia Seletiva

Finalmente, a função

```
op select : Float Buffer -> ActorBuffer .
```

implementa a estratégia seletiva descrita na Seção 3.1. Dado o tempo global e um *buffer*, esta função retorna um par de sort *ActorBuffer* com o nome do ator do elemento que foi selecionado para ser removido e o *buffer* resultante obtido retirando esse elemento do *buffer* fornecido.

A estratégia seletiva formalizada usa a seguinte função:

```
op sampleUniWithInt : Nat -> Nat
```

que para uma dada entrada n , retorna um número aleatório no intervalo entre 0 e n .

O mecanismo de seleção por k -torneio é implementado pela função:

```
op creatingTour : NBuffer Nat Buffer -> Buffer .
```

Ela toma como argumentos um *NBuffer* e um número natural, especificando o tamanho do torneio, e acumula os elementos selecionados para o torneio em um terceiro parâmetro. Uma vez que o torneio é criado, é usado a seguinte função para selecionar aquele com o maior *DropFactor*:

```
op selectGreatest : Buffer -> EleBuf .
```

3.2.2 Regras de Reescrita

As regras de reescrita modificam elementos de *Conf* e especificam a semântica operacional do sistema. São descritos a seguir as principais regras de reescrita usadas no presente trabalho.

A primeira ação descrita é quando um ator recebe uma mensagem *pool* indicando que ele deve iniciar uma chamada no momento $gt + delay$.

```
rl [CLIENT-RECEIVE-POOL] :
  <name: c(i) | server: Ser, status: none, AS >
  {c(i) <- poll} [gt | SL]
=>
  <name: c(i) | server: Ser , status: waiting, AS >
  mytick(insert([gt | SL], { gt + delay, Ser <- INVITE(c(i))})) .
```

A regra de reescrita a seguir especifica o comportamento de um cliente ao receber uma mensagem OK do servidor. Ele muda o seu estado de *waiting* para *incall* e gera uma mensagem BYE, programada para ser enviada depois de algum tempo. Isto significa que o cliente chamado $c(i)$ deverá terminar a sua chamada no momento $gt + callDur(tMedio)$.

```

rl [CLIENT-RECEIVE-OK] :
  <name: c(i) | server: Ser, status: waiting, AS >
  {c(i) <- OK} [gt | SL]
=>
  <name: c(i) | server: Ser , status: incall, AS >
  mytick(insert([gt | SL],
    { gt + callDur(tMedio), (Ser <- BYE(c(i)))})) .

```

A aplicação pode, no entanto, encerrar uma chamada antes que esta seja concluída. Tais chamadas são classificadas como incompletas, *i.e.*, o status da chamada descartada é alterado para *incomplete*:

```

rl [CliEnd-Incomplete-drop] :
  flag(true,r) { gt | SL }
  (c(i) <- poll)
  <name: c(i) | server: Ser , retries: j ,
    status: incall, AS >
=> flag(gt <= LIMIT * r, r)
  <name: c(i) | server: Ser , atime: gt, retries: j ,
    status: incomplete, AS >
  mytick({ gt | SL }) .

```

As regras de reescrita para os atacantes são semelhantes às regras do cliente. A única diferença é que nenhuma mensagem BYE é gerada, portanto, especificando o *Coordinated Call Attack* onde os atacantes tentam permanecer na chamada por tempo indeterminado.

A Figura 3.3 descreve as regras que implementam a estratégia seletiva proposta no presente trabalho. Para cada mensagem INVITE recebida por algum ator (*Actor*), a regra *RECEIVE-INVITE* verifica se o *buffer* do servidor atingiu seu máximo. Se não, então a

```

cr1 [RECEIVE-INVITE] :
  <name: Ser | req-cnt: pmod , b-set: [lenB | B], AS >
  {Ser <- INVITE(Actor)} [gt | SL]
=> if (lenB >= lenBufSeVen) then
  if p1 then ConfAcc myTick(SchAcc)
  else ConfRej myTick(SchRej)
  fi
  else ConfAcc2 mytick(SchAcc2)
  fi
if p1 := sampleBerWithP(accept-prob(pmod))
/\ { ActorDr, bufDr } := select(gt,B)
/\ nBuf := add([lenB + (- 1) | bufDr], < Actor invite gt >)
/\ ConfAcc := <name: Ser | req-cnt: (pmod + 1.0),
  b-set: nBuf, AS >
/\ SchAcc := insert([gt | SL],
  {gt, Actor <- TRYING} ;
  {gt, ActorDr <- poll}})
/\ ConfRej := <name: Ser | req-cnt: (pmod + 1.0),
  b-set: [lenB | B], && AS >
/\ SchRej := insert([gt | SL],
  {gt + delay , Actor <- poll}})
/\ b-setNu := add( [lenB | B], < Actor invite gt > )
/\ ConfAcc2 := <name: Ser | req-cnt: pmod , b-set: b-setNu,
  AS >
/\ SchAcc2 := insert([gt | SL],
  {gt + delay, Actor <- TRYING})) .

r1 [APP-ROUND] :
  <name: Ser | req-cnt: pmod , b-set: [lenB | B], AS >
  {Ser <- ROUND} [gt | SL]
=>
  <name: Ser | req-cnt: 0.0, b-set: [lenB | B], AS >
  mytick(insert([gt | SL],
    {gt, reply(Ser, B, gt)}
    {gt + Ts, Ser <- ROUND})) .

```

Figura 3.3: Regras de reescrita especificando a estratégia seletiva proposta no presente trabalho.

requisição é adicionada ao *buffer* do servidor. Caso contrário, então a requisição é inserida no *buffer* da defesa (ConfAcc2) e uma mensagem TRYING para o ator correspondente é gerada. Se o *buffer* do servidor atingiu a capacidade máxima, a defesa joga uma moeda ($p1$) para decidir se a requisição de entrada será servida. Se ela decidir atender a nova requisição, então alguma chamada sendo processada será selecionada para ser descartada usando

a função `select`. Ela retorna o nome do ator `ActorDr` e o *buffer* resultante `nBuf`. A nova requisição é adicionada a `nBuf` e `pmod` é incrementado, resultando na configuração `ConfAcc`. Além disso, uma mensagem `poll` para `ActorDr` e uma `TRYING` destinada a `Actor` são geradas. Caso contrário, a requisição de entrada é rejeitada e `pmod` é incrementado sem afetar o *buffer* do servidor, resultando na configuração `ConfRej`. Uma mensagem `poll` para `Actor` também é criada.

A regra `APP-ROUND` especifica que quando a rodada termina, todas as requisições que sobreviveram no *buffer* do servidor são respondidas e `pmod` é reajustado.

3.3 Simulações

A especificação formal da estratégia seletiva proposta foi validada usando o verificador estatístico de modelos `PVeStA`. As simulações conduzidas são parametrizadas nos seguintes valores:

- **Tempo médio de uma chamada** – t_M : Este é o tempo médio estimado de duração das chamadas de usuários honestos. Foi assumido $t_M = 5$ unidades de tempo;
- **Fator de Descarte** - `DropFactor` - São pressupostos os seguintes valores para as constantes da função `DropFactor` (Equação 3.1):
 - $p_{IN} = 2$;
 - $p_{WAIT} = 8$;
 - $\alpha = 2.55$.

Esses valores foram escolhidos para fazer com que a chance de uma chamada ser interrompida aumente consideravelmente após sua duração ser maior que t_M . Alguns valores de amostra são os seguintes, lembrando que $t_M = 5$:

Duração da chamada (unid.)	DropFactor
6	12.37
8	17.31
10	27.84

Ou seja, o fator de descarte de uma chamada com duração de 10 unidades de tempo é aproximadamente 3 vezes maior que uma chamada cujo *status* é WAITING (27.84/8). Esta é uma razão coerente. No entanto, de acordo com a aplicação, outros valores podem ser ajustados para p_{IN} , p_{WAIT} e α . Finalmente, foi definido $p_{WAIT} = 4$ vezes p_{IN} de modo que as chamadas com duração inferior a t_M têm muito menos chance de serem descartadas do que aquelas que ainda estão à espera da resposta;

- **Capacidade do servidor** – N_T : Este é o limite superior do tamanho do *buffer* do servidor que indica a capacidade de processamento do aplicativo. $N_T = 24$;
- **Taxa de Chamadas (R)**: A taxa total de chamadas foi fixada como R , que é o resultado da soma do tráfego das chamadas legítimas, R_L , com o tráfego das chamadas de atacantes, R_A . Portanto, $R_L + R_A = R$.

O valor de R é computado usando técnicas padrão (Jewett, Shrago e Yomtov 1980)³, de modo que se $R_L = R$, *i.e.*, o servidor não está sob ataque, então o mesmo não será sobrecarregado.

Com R fixo, foi definido R_L e R_A como $R_L + R_A = R$, mas considerando cenários com diferentes proporções para R_A e R_L . Isso reflete o fato de que *Coordinated Call Attack* usa baixo tráfego e, portanto, pode sobrepujar as defesas habituais baseadas em análise de tráfego de rede que normalmente monitoram a taxa total de chamadas R .

Foram considerados 5 diferentes proporções para R_L e R_A , expressas na porcentagem do número total de chamadas R :

Chamadas legítimas (R_L)	Chamadas de atacantes (R_A)
83%	17%
67%	33%
50%	50%
33%	67%
17%	83%

³Usando o modelo de Erlang que calcula R tendo em conta N_T e t_M

- **Duração das chamadas legítimas:** Enquanto os atacantes tentam permanecer em uma chamada por longos períodos de tempo, clientes legítimos não se comportam assim. A literatura modela a duração da chamada legítima, em sistemas VoIP, como sendo uma distribuição assimétrica e possivelmente de cauda longa (Meyer 1970). Baseado nos trabalhos de Sheluhin, Smolskiy e Osin 2007, Waters e Engineering 2007, Galiotos, Dagiuklas e Kotsopoulos 2015, a duração da chamadas legítimas é definida por uma distribuição lognormal do tipo:

$$f(x, \mu, \sigma) = \frac{1}{x\sigma\sqrt{2\pi}} \exp \left[-\frac{(\ln(x) - \mu)^2}{2\sigma^2} \right]. \quad (3.2)$$

onde os parâmetros σ e μ são computados de acordo com o tempo médio assumido t_M (ver Jewett, Shrago e Yomtov 1980 para mais informações);

- **Tempo total das simulações - total:** Cada uma das simulações tem duração de 40 segundos, similar aquele usado em Eckhardt et al. 2012;
- **Atraso na Rede:** Foi usado um atraso de 0.1 segundo das mensagens na rede;
- **Grau de confiança para simulação:** As simulações foram realizadas com grau de confiança de 99% (ver Agha, Meseguer e Sen 2006, Sen, Viswanathan e Agha 2005 para mais detalhes acerca de verificação estatística de modelos).

3.3.0.0.1 Métricas de Qualidade Nas simulações, empregaram-se métricas intrínsecas para serviços VoIP. Estas são especificadas por expressões quantitativas QuaTEX baseadas em lógica temporal probabilística definidas por Agha, Meseguer e Sen 2006. A verificação estatística de modelos foi aplicada a estratégia seletiva, no sentido de: uma vez que uma fórmula QuaTEX e um grau desejado de confiança são especificados, um número suficientemente grande de simulações de Monte Carlo são realizadas (pelo Maude) permitindo a verificação da fórmula QuaTEX.

As fórmulas QuaTEX, *i.e.*, as métricas de qualidade, empregadas nas simulações são definidas abaixo. O operador $\bigcirc$ é uma modalidade temporal que especifica o avanço do tempo global para o tempo do próximo evento (ver Agha, Meseguer e Sen 2006 para mais detalhes).

- **Chamadas Completas:** quantas chamadas honestas foram capazes de permanecer em INCALL por sua duração esperada.

$$complete(total) = \text{if } time > total \text{ then } \frac{countComplete}{countHonest} \\ \text{else } \bigcirc complete(total)$$

onde $countComplete$ é um contador que é incrementado sempre que uma chamada honesta é completada.

- **Chamadas Incompletas:** quantas chamadas honestas foram capazes de obter o estado INCALL mas foram descartadas antes que a chamada fosse completada, *i.e.* que não permaneceram com estado INCALL durante toda a duração esperada.

$$incomplete(total) = \text{if } time > total \text{ then } \frac{countIncomplete}{countHonest} \\ \text{else } \bigcirc incomplete(total)$$

onde $countIncomplete = countIncall - countComplete$, e $countIncall$ é um contador que é incrementado sempre que uma chamada honesta muda do estado WAITING para INCALL.

- **Chamadas Malsucedidas:** quantas chamadas honestas não foram capazes de chegar ao estado INCALL.

$$unsuccessful(total) = \text{if } time > total \text{ then } \frac{countUnsuccess}{countHonest} \\ \text{else } \bigcirc unsuccessful(total)$$

onde $countUnsuccessful = countHonest - countIncall$.

3.3.1 Resultados sem Defesa

Foram realizadas simulações do *Coordinated Call Attack* em um cenário sem defesa objetivando quantificar o potencial desse ataque e seu efeito em um serviço VoIP vulnerável. Os resultados de tais simulações são ilustrados na Figura 3.4. Eles sugerem que o *Coordinated Call Attack* é efetivamente eficaz na redução da disponibilidade de um serviço VoIP. Aumentar a proporção de atacantes aumenta rapidamente a proporção de chamadas sem êxito,

i.e., chamadas que nem sequer iniciaram uma conversa, enquanto a proporção de chamadas completas diminui. Como esperado, não há chamadas incompletas, pois o servidor VoIP não interrompe chamadas.

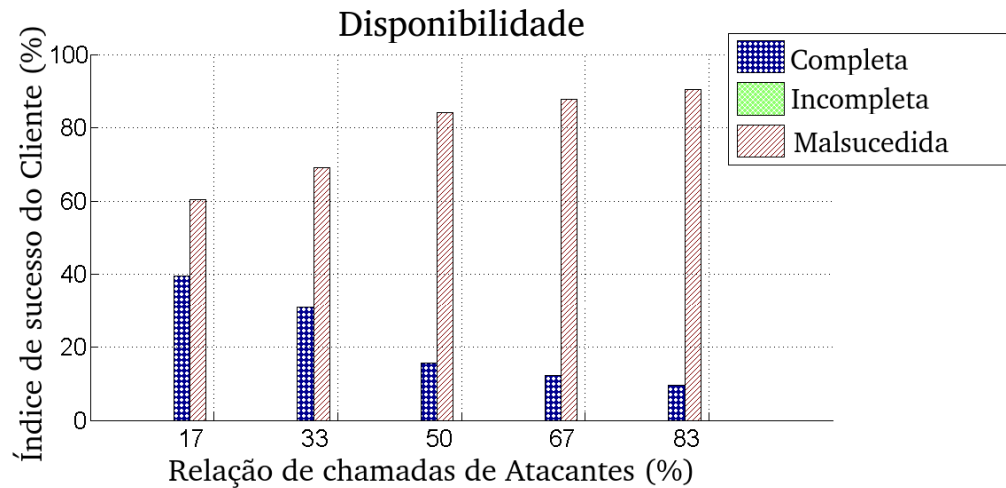


Figura 3.4: Grau de Sucesso do Cliente: Resultados das simulações sob ataque e defesa **não** sendo executada.

3.3.2 Resultados com Defesa

A Figura 3.5 ilustra os resultados de disponibilidade ao usar a estratégia seletiva proposta. Eles sugerem que estratégias seletivas podem realmente mitigar o *Coordinated Call Attack*. A proporção de chamadas completas permanece em níveis elevados, acima de 86% das chamadas legítimas.

Como a defesa pode interromper chamadas no meio de uma conversa, há chamadas incompletas, *i.e.*, chamadas onde as partes começaram a se comunicar, mas foram interrompidas antes de se comunicarem pelo tempo esperado. Cerca de 10% das chamadas legítimas foram interrompidas.

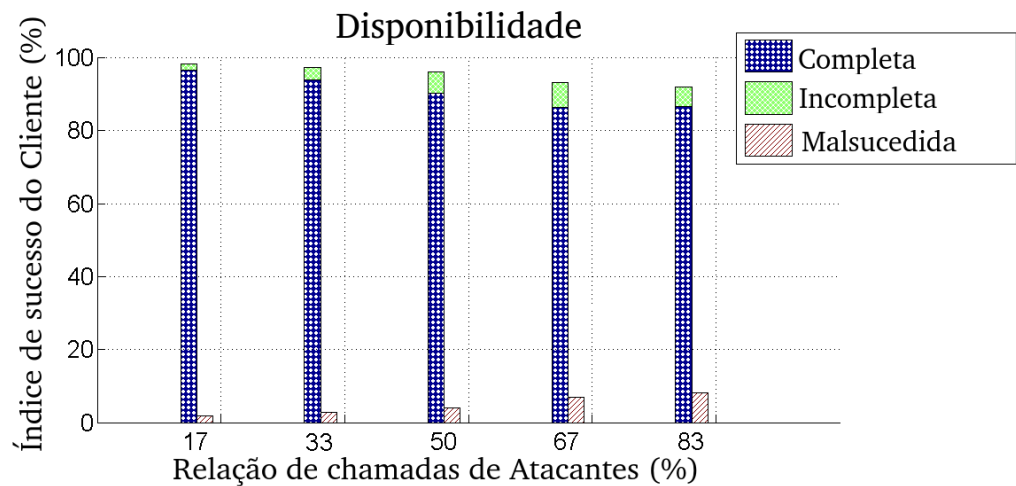


Figura 3.5: Grau de Sucesso do Cliente: Resultados das simulações sob ataque e usando a defesa.

3.4 Considerações

Neste capítulo, foi apresentada uma nova estratégia seletiva adequada para lidar com ataques de negação de serviço a sistemas VoIP. Existe uma principal diferença entre a estratégia seletiva proposta pelo presente trabalho com respeito aquela usada no trabalho de Dantas, Nigam e Fonseca 2014. Essa diferença é na função Drop, usada para selecionar a requisição que deve ser descartada. Em Dantas, Nigam e Fonseca 2014, nenhum DropFactor foi utilizado e as requisições são escolhidas de maneira uniforme, *i.e.*, com a mesma chance. No presente trabalho, foi apresentada a ideia de DropFactor, baseado no estado e duração das chamadas. O DropFactor é então usado por um mecanismo de seleção por torneio para eleger a chamada que será encerrada. A combinação de DropFactor com seleção por torneio também pode ser empregada no contexto dos servidores Web (Sampaio et al. Submetido). Por exemplo, existem ataques DDoS que tentam degradar o serviço solicitando um número suficiente de tarefas intensivas de CPU e memória. Acredita-se na possibilidade de mitigação dessa classe de ataques DDoS, computando o DropFactor considerando a memória e ciclos de CPU usados por um processo associado à requisição. De fato, estratégias seletivas que fazem uso dessa abordagem se mostraram eficazes em mitigar ataques de exaustão de memória a web-sites WordPress, como descrito em Sampaio et al. Submetido.

Antes de iniciar a experimentação da estratégia seletiva com chamadas reais, realizou-se uma série de simulações de Monte Carlo em cima do modelo formal da defesa implementado na ferramenta Maude. Normalmente, cada simulação leva cerca de 40 segundos para ser concluída. Em contraste, cada experimento realizado na rede demorou 60 minutos. Isto significa que, no presente trabalho, a formalização representou um caminho mais rápido para se obter uma primeira validação da estratégia seletiva proposta. As simulações mostraram que o *Coordinated Call Attack* é eficaz se o servidor não possui nenhum mecanismo de defesa. Todos os resultados de simulações usando cenários com defesa indicam que estratégias seletivas são de fato uma boa defesa para mitigar o *Coordinated Call Attack*. A maior proporção de chamadas foram completadas, enquanto uma menor proporção das chamadas foram interrompidas e uma minoria de chamadas não foram bem sucedidas. No próximo capítulo, esses resultados obtidos por simulação são validados através de experimentos realizados na rede usando uma arquitetura SIP usual.

Capítulo 4

Resultados Experimentais

O objetivo deste capítulo é descrever os experimentos conduzidos que validam a defesa proposta para mitigar o *Coordinated Call Attack*. Na Seção 4.1, explicamos as ferramentas desenvolvidas e utilizadas nos experimentos e explanamos como estas foram configuradas. Nas seções 4.2 e 4.3, são descritos os parâmetros empregados nos experimentos e as métricas de qualidades empregadas, respectivamente. Os resultados são apresentados e discutidos na Seção 4.4.

4.1 Plano do Experimento

Para os experimentos, foi utilizado o Asterisk (Digium 2015) em sua versão 13.6.0, o qual é um servidor SIP amplamente utilizado por empresas de pequeno e médio porte. Assumi-se a presença de usuários honestos e atacantes que tentam fazer com que o serviço VoIP fique indisponível. Ambos os tráfegos de usuários honestos e atacantes maliciosos foram emulados usando a ferramenta SIPp (GAYRAUD e JACQUES 2014), com versão 3.4.1. SIPp gera chamadas que podem ser configuradas como chamador e receptor. Portanto, no ensaio, são usados pares de instâncias do SIPp, um par gerando as chamadas dos usuários honestos e o outro gerando as do atacante. Finalmente, foi desenvolvida uma aplicação em C++ que opera como um *proxy* e implementa a estratégia seletiva descrita na Seção 3.1.

A Figura 4.1 ilustra a infraestrutura empregada nos experimentos. Para fazer uma chamada, pares de SIPp enviam mensagens para o *proxy*, que por outro lado as encaminha para o Asterisk. Da mesma forma, todas as mensagens geradas pelo Asterisk são enviadas para

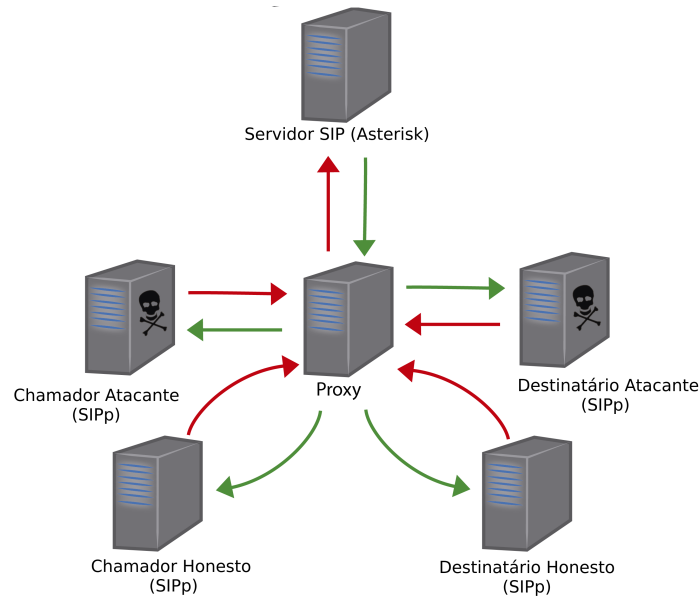


Figura 4.1: Topologia da infraestrutura VoIP empregada nos experimentos.

o *proxy* que então as encaminha para seus respectivos usuários. A arquitetura empregada na implementação da aplicação *proxy* pode ser vista na Figura 4.2. A aplicação é basicamente dividida em dois módulos: *Core* e *Shared Libraries*. Enquanto que o *Core* é responsável por transmitir e processar as mensagens SIP recebidas, uma biblioteca compartilhada fica se comunicando com o módulo AMI HTTP (Asterisk 2007) do Asterisk, obtendo informações sobre o estado interno do servidor e enviando comandos para encerrar/rejeitar chamadas. Quando uma nova chamada chega, o *proxy* envia o comando **'core show calls'** para o AMI HTTP e recebe como resposta o número de chamadas sendo atualmente processadas. Se esse valor for igual a sua capacidade máxima, o *proxy* utiliza a função *Select* para decidir se a nova chamada será servida. Caso *Select* retorne "true", o *proxy* usa o comando **'cdr show active'** para obter informações sobre as chamadas sendo processadas, como o *id* e duração (em segundos). Essas informações são usadas por *Drop* para escolher a chamada que será encerrada. Finalmente, o comando **'channel request hangup'** é enviado ao Asterisk, instruindo-o a interromper a chamada selecionada.

Para os experimentos, é suficiente utilizar uma única máquina. Foi utilizado uma máquina com configuração Intel(R) Core(TM) i7-4510U CPU @ 2.00GHz e 8 GB de RAM.

Finalmente, foi definida a duração das chamadas geradas pelo SIPp como se segue:

- **Duração Total das Chamadas dos Clientes:** Como mencionado na Seção 3.3, sem-

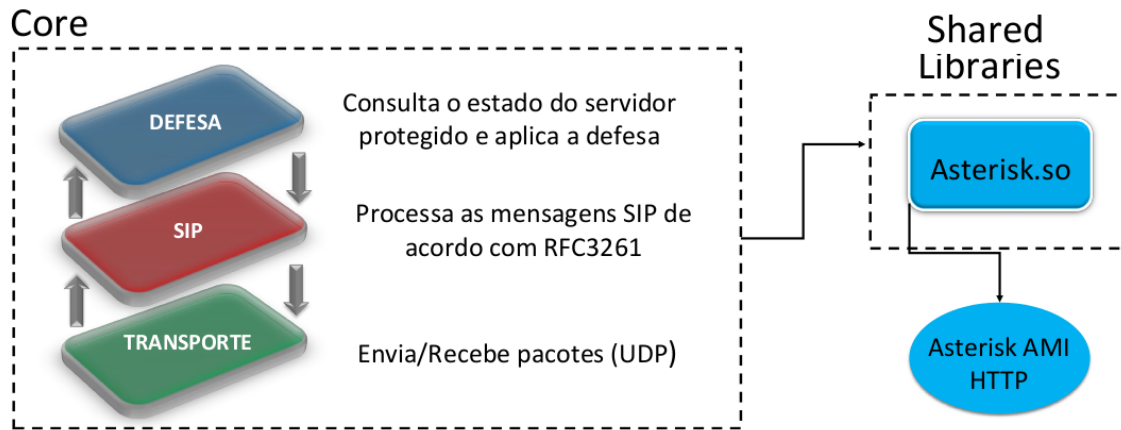


Figura 4.2: Arquitetura da aplicação *proxy* que implementa a defesa proposta. O módulo *Core* é responsável por lidar com as mensagens SIP, enquanto que uma biblioteca compartilhada fica consultando o estado do servidor Asterisk e encerrando chamadas

pre que uma nova chamada é realizada, sua duração é definida por um número real gerado por uma distribuição lognormal com média igual a t_M . SIPp encerra a chamada quando o tempo de duração da mesma é atingido;

- **Duração Total das Chamadas dos Atacantes:** Seguindo o *Coordinated Call Attack*, a duração das chamadas do atacante não foi limitada. Ele se comunica por tempo indefinido.

4.2 Parâmetros

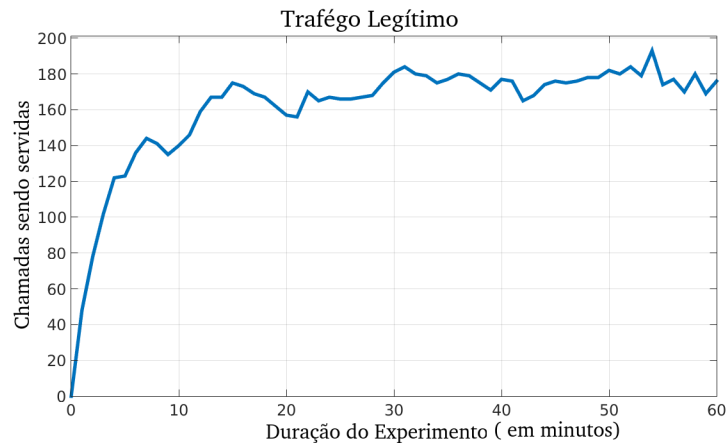
Foram usados os seguintes parâmetros para configurar os experimentos:

- **Tempo Médio de Duração das Chamadas (t_M)** - É pressuposto que a média de duração das chamadas é conhecida. Essa informação pode ser obtida por análise do histórico de chamadas. Nos experimentos é assumido $t_M = 162$ segundos¹;
- **Capacidade do Servidor SIP (N_T)** - Representa o número de chamadas simultâneas que um servidor SIP pode manter. Foi empregado $N_T = 200$, que é uma capacidade realista para uma pequena empresa, permitindo que 400 usuários (2×200) usem o serviço ao mesmo tempo;

¹Esse valor de t_M foi escolhido com base em estatísticas do serviço VoIP fone@RNP (RNP 2016)

- **Tempo Total de Experimento (T)** - Cada um dos experimentos tem duração de 60 minutos, isto é, 22 vezes a média de duração das chamadas dos clientes. Com essa duração, já é possível testemunhar os danos causados pelo *Coordinated Call Attack* bem como a eficiência da solução proposta em mitigar esse ataque;
- **Taxa de Tráfego (R)** - Usando um servidor com capacidade de 200 chamadas concorrentes, foram empregadas técnicas padrão (Jewett, Shrago e Yomtov 1980) para calcular o que seria um tráfego típico para tal servidor. Foi obtido o valor $R = 60$ chamadas por minuto. No ensaio, essa taxa foi dividida entre clientes e atacantes. Lembrando que o *Coordinated Call Attack* não gera tráfego diferente do esperado, de modo que ele é capaz de passar por defesas baseadas na análise do mesmo. Portanto, o tráfego total nos experimentos será sempre menor ou igual a R .

O gráfico a seguir ilustra o uso dos clientes em condições normais, isto é, sem sofrer um ataque, usando os parâmetros especificados acima.



Isso mostra que o servidor é bem dimensionado para essa taxa de chamadas (legítimas). Os clientes ocupam em média aproximadamente 85% da capacidade do Asterisk, portanto, não sobrecarregando o servidor, mas ainda sendo fortemente usado.

4.3 Métricas de Qualidade

Para os experimentos, foram utilizadas as seguintes métricas de qualidade para as chamadas:

- **Chamadas Completas:** Uma chamada é completa se seu estado mudar de WAITING para INCALL e se a mesma for capaz de permanecer nesse estado por toda sua duração. Isto é, o chamador é capaz de se comunicar com o receptor por toda duração prescrita;
- **Chamadas Incompletas:** Uma chamada é incompleta se ela muda seu estado de WAITING para INCALL, mas não sendo capaz de permanecer nesse estado por toda sua duração. Isto é, o chamador é interrompido antes de completar a chamada;
- **Chamadas Malsucedidas:** Uma chamada é malsucedida se ela não foi capaz de mudar seu estado de WAITING para INCALL. Isto é, o chamador não têm sequer a oportunidade de falar com o receptor.

Intuitivamente, chamadas completas são melhores que as incompletas que, por sua vez, são melhores que as malsucedidas. A fim de apoiar esta ideia, foi computada a duração média das chamadas incompletas, isto é, o tempo médio que o usuário é capaz de ficar se comunicando antes que sejam interrompidos pela estratégia seletiva.

4.4 Resultados Experimentais

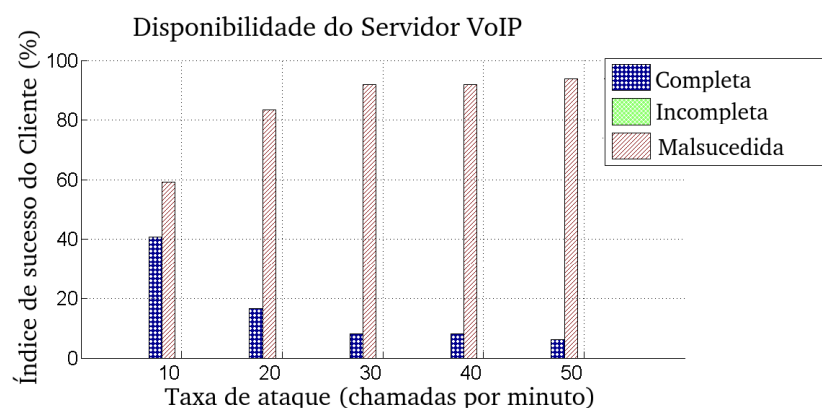
Foram realizados uma série de experimentos que analisam o ataque e a eficácia da defesa proposta. Três tipos de experimentos foram executados:

- **Tipo 1:** Atacando o servidor Asterisk sem utilizar o *proxy* para defendê-lo;
- **Tipo 2:** Atacando o servidor Asterisk e utilizando o *proxy* para defendê-lo;
- **Tipo 3:** Não atacando o servidor Asterisk mas ainda assim usando o *proxy*.

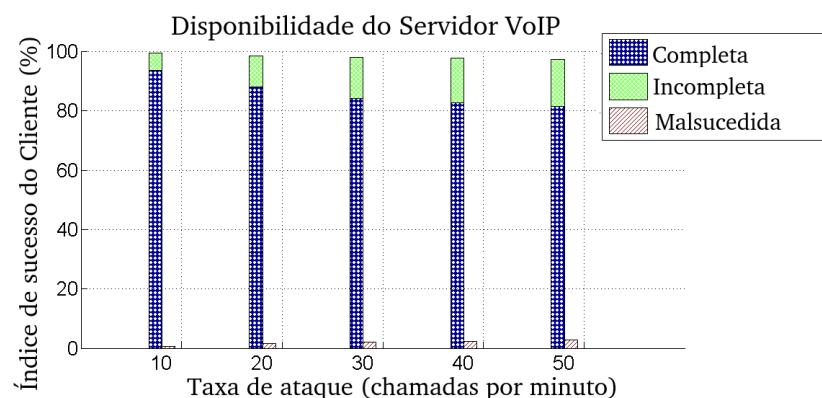
A Figura 4.3 apresenta os resultados de disponibilidade para clientes legítimos em cada tipo de experimento. Nos experimentos com ataque, a taxa de chamadas do atacante variou de 10 chamadas por minuto até 50 por minuto. A taxa de chamadas dos clientes foi ajustada a fim de preservar um total de 60 por minuto, que é uma taxa de chamada típica para o servidor Asterisk como detalhado na Seção 4.2. Para o cenário com uma taxa de ataque de 10 chamadas por minuto, a quantidade de chamadas dos clientes foi $60 - 10 = 50$ chamadas por minuto, o que corresponde a uma proporção de 5 chamadas de clientes para cada chamada

do atacante. Da mesma forma, quando a taxa de chamadas utilizada pelo atacante é de 50 por minuto, foram geradas 10 por minuto para clientes legítimos, que corresponde a uma proporção de 5 chamadas do atacante para cada chamada do cliente.

Quando sob ataque e não executando a defesa (Figura 4.3(a)), observou-se que o total de chamadas malsucedidas foi bastante elevado (59-94% do total de chamadas do cliente). Isso significa que esses clientes não foram capazes de iniciar uma chamada. As restantes (6-41% do total de chamadas do cliente) foram todas bem sucedidas, ou seja, cada um pôde ficar usando o serviço VoIP pelo tempo total que lhes foi atribuído.

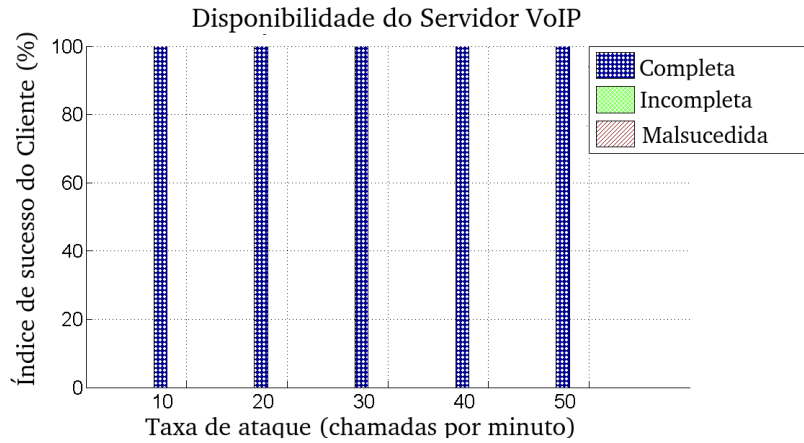


(a) Grau de sucesso do cliente quando sob ataque e a estratégia seletiva **não** está sendo executada.



(b) Grau de sucesso do cliente quando sob ataque e a estratégia seletiva está sendo executada.

Figura 4.3: Disponibilidade do servidor VoIP.



(c) Grau de sucesso do cliente quando não se está sob ataque e a estratégia seletiva está sendo executada.

Figura 4.3 (cont.): Disponibilidade do servidor VoIP.

Por outro lado, quando sob ataque e executando a estratégia seletiva (Figura 4.3(b)), observou-se que os clientes tinham muito mais chamadas bem-sucedidas (81-93% do total de chamadas de clientes), enquanto alguns (0.6- 3% do total de chamadas do cliente) não foram capazes de sequer iniciar a chamada. As demais (6-16% do total de chamadas de clientes) foram interrompidas devido ao comportamento da estratégia seletiva. Portanto, a proporção de chamadas legítimas que foram capazes de atingir o estado INCALL é próximo de 100% do total de chamadas de clientes. Isso significa que quase todos os clientes foram capazes de usar (mesmo que alguns de forma incompleta) o serviço VoIP.

Finalmente, nota-se que a estratégia seletiva não afetou a disponibilidade do Asterisk quando este não está sob ataque, como representado na Figura 4.3(c), onde todas as chamadas de clientes foram bem sucedidas.

Objetivando entender melhor o perfil das chamadas incompletas no cenário com ataque e usando a estratégia seletiva para defender o serviço, foi mensurada a razão para cada chamada incompleta:

$$\frac{t_D}{t_T}$$

em que t_D é a duração da chamada, *i.e.*, o tempo entre o momento em que a chamada começou até o momento em que ela foi interrompida, e t_T é o tempo total de duração pretendido. A Figura 4.4 ilustra a razão média obtida em cada teste executado. Como pode

ser observado, as chamadas que são interrompidas pela defesa estão conseguindo ficar utilizando o serviço VoIP por cerca de 51-58% do tempo total pretendido. Acredita-se que esses tempos médios alcançados são razoavelmente satisfatórios, especialmente em situações críticas, como em emergências, dado que os clientes interrompidos estão conseguindo ficar se comunicando por mais de 50% do tempo pretendido. Após ser interrompido, o cliente tem a oportunidade de realizar uma nova chamada e continuar falando pelo resto do tempo desejado. Como trabalho futuro pretende-se investigar como mudar as distribuições de probabilidades utilizadas para configurar a estratégia seletiva de modo a melhorar esses números.

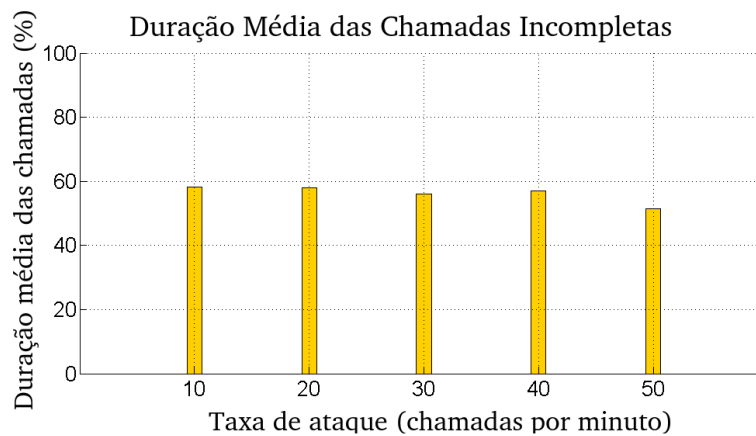
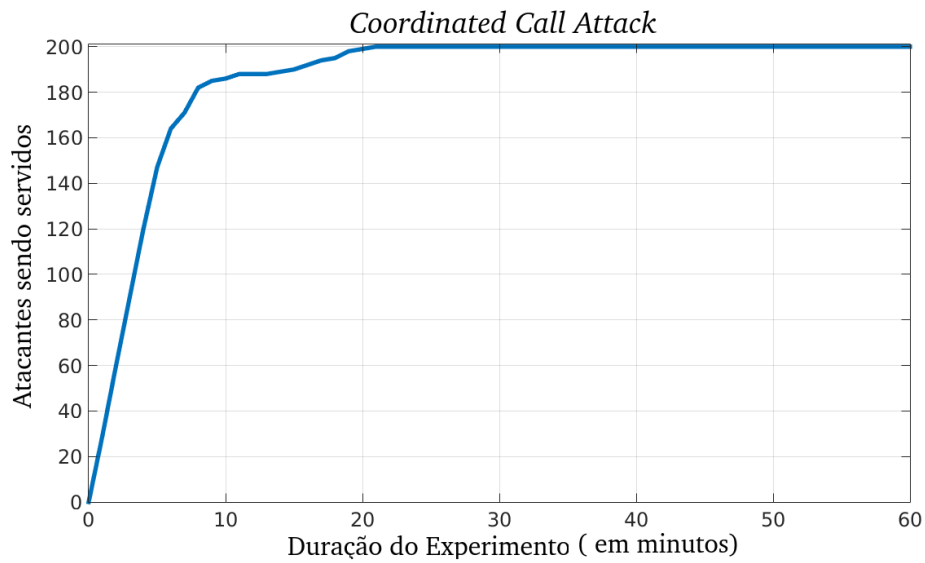
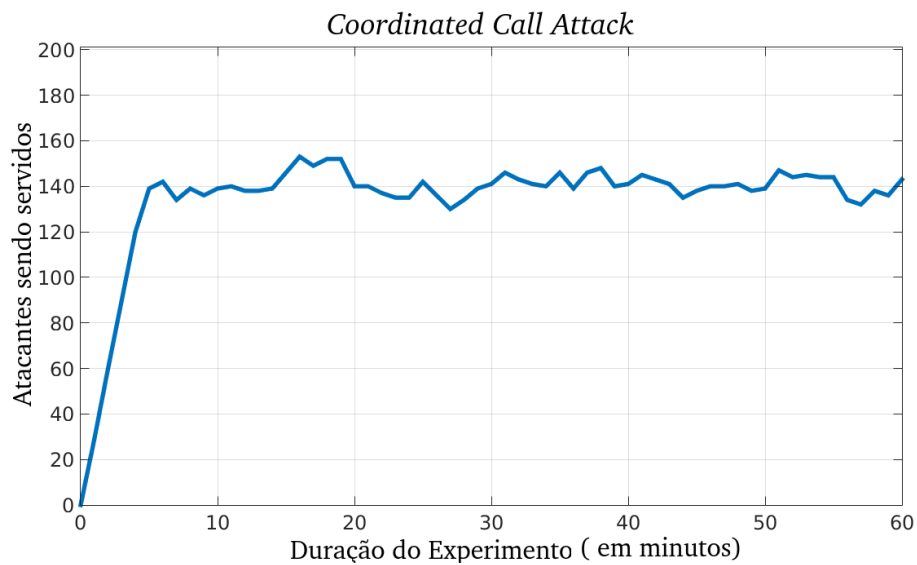


Figura 4.4: Índice de duração médio das chamadas incompletas.

Finalmente, a fim de compreender melhor o comportamento do *Coordinated Call Attack* e, em particular, o quão eficaz é a defesa proposta, foi analisada quantas chamadas de atacantes são servidas ao longo do experimento. A Figura 4.5 apresenta os resultados para o caso em que a taxa de chamadas do atacante é 30 por minuto, ou seja, a mesma que a taxa de chamada dos clientes ($60 - 30 = 30$ chamadas por minuto). Os casos com outras taxas tiveram resultados semelhantes. Como esperado do perfil do *Coordinated Call Attack*, o atacante é capaz de negar serviço lentamente (após 20 minutos) ocupando todas os recursos disponíveis no servidor. Após este ponto, nenhum cliente legítimo é capaz de usar o serviço VoIP. Isto significa que, se considerarmos experimentos ainda mais longos, a disponibilidade para os clientes (Figura 4.3(a)) tenderia a zero, pois estes só são capazes de acessar o serviço VoIP



(a) Sem utilizar a estratégia seletiva.



(b) Usando a estratégia seletiva.

Figura 4.5: Ocupação do atacante ao longo do tempo, quando a taxa de chamadas do atacante é de 30 chamadas por minuto.

nos primeiros 20 minutos do experimento. **Na verdade, foi realizado um experimento onde primeiro os atacantes foram inicializados, deixando-os ocupar todos os recursos disponíveis no servidor VoIP para que só então os clientes sejam inicializados. O resultado deste experimento foi de 100% das chamadas não concretizadas.** Em contraste, ao

usar a estratégia seletiva, o atacante só é capaz de ocupar cerca de 70% dos recursos do servidor. Isso pode parecer um alto valor, mas o gráfico esconde o fato de que os atacantes são descartados pela estratégia de defesa e, portanto, altos níveis de disponibilidade são obtidos.

Capítulo 5

Conclusão e Trabalhos Futuros

Este trabalho apresentou uma nova estratégia seletiva para mitigar o *Coordinated Call Attack* em serviços VoIP. Mostrou-se que o uso de distribuições de probabilidade dependentes do estado para selecionar quais chamadas devem ser processadas resulta em altos níveis de disponibilidade. Foram realizadas simulações sobre um modelo formal e experimentos em uma rede VoIP. Em ambos os casos, observou-se que a estratégia seletiva foi capaz de mitigar o *Coordinated Call Attack*.

Uma das principais vantagens da solução proposta é que ela não foi concebida usando muitas suposições específicas sobre o tipo de serviço. O único pressuposto usado é um conhecimento prévio da duração média das chamadas, que pode ser facilmente deduzida a partir do histórico de chamadas de serviço. Além disso, a solução proposta pode ser facilmente integrada com outros mecanismos de defesa, como aqueles abordados na Seção 2.5.

Enquanto se deixa como trabalho futuro testar a estratégia seletiva contra outros ataques ADDoS *Low-Rate* ao VoIP (Hussain et al. 2015), acredita-se fortemente na possibilidade de mitigação de ataques que saturam recursos através da ocupação do *buffer* do servidor alvo, como o *Ringling-based attack* (Conner e Nahrstedt 2008), que tenta ocupar o *buffer* da aplicação com transações INVITE que nunca são respondidas, objetivando consumir toda a memória disponível. Essas transações INVITE podem ser modeladas como chamadas WAITING e podem ser descartadas configurando p_{WAIT} adequadamente. Tal investigação é deixada como trabalho futuro.

Como outro projeto póster, está sendo investigado de que modo estratégias seletivas podem ser usadas contra ataques de amplificação no protocolo SIP (Shankesi et al. 2009).

Finalmente, pretende-se incorporar a estratégia seletiva proposta em serviços VoIP existentes, como o `fone@RNP` (RNP 2016).

Bibliografia

Agha et al. 2005 AGHA, G. et al. Formal modeling and analysis of dos using probabilistic rewrite theories. In: *Foundations of Computer Security (FCS)*. [S.l.: s.n.], 2005.

Agha, Meseguer e Sen 2006 AGHA, G.; MESEGUER, J.; SEN, K. PMAude: Rewrite-based Specification Language for Probabilistic Object Systems. *Electron. Notes Theor. Comput. Sci.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 153, n. 2, p. 213–239, maio 2006. ISSN 1571-0661.

Akbar, Basha e Sattar 2015 AKBAR, A.; BASHA, S.; SATTAR, S. A. Leveraging the sip load balancer to detect and mitigate ddos attacks. In: *Green Computing and Internet of Things (ICGCIoT), 2015 International Conference on*. [S.l.: s.n.], 2015. p. 1204–1208.

Alturki, Meseguer e Gunter 2009 ALTURKI, M.; MESEGUER, J.; GUNTER, C. A. Probabilistic modeling and analysis of dos protection for the asv protocol. *Electr. Notes Theor. Comput. Sci.*, v. 234, p. 3–18, 2009.

Asterisk 2007 ASTERISK. *Allow Manager Access via HTTP*. 2007. <<https://wiki.asterisk.org/wiki/display/AST/Allow+Manager+Access+via+HTTP>>. Accessed: 2015-27-09.

Bansal e Pais 2015 BANSAL, A.; PAIS, A. Mitigation of flooding based denial of service attack against session initiation protocol based voip system. In: *Computational Intelligence Communication Technology (CICT), 2015 IEEE International Conference on*. [S.l.: s.n.], 2015. p. 391–396.

Blickle e Thiele 1995 BLICKLE, T.; THIELE, L. A Mathematical Analysis of Tournament Selection. In: *Proceedings of the 6th International Conference on Genetic Algorithms*. San Francisco, CA, USA: [s.n.], 1995. p. 9–16.

Bode 2012 BODE, K. *CallCentric Victim of Devastating Two-Week DDoS Attack*. 2012. <<http://www.dslreports.com/shownews/CallCentric-Victim-of-Devastating-MonthLong-DDoS-Attack-121667>>. [Online; Acessado 28-Setembro-2015].

Brown et al. 2002 BROWN, L. et al. The title of the work. *Statistical Analysis of a Telephone Call Center: A Queueing Science Perspective*, n. 100, p. 36–50, 2002.

Camp 2003 CAMP, K. *IP Telephony Demystified*. [S.l.]: The McGraw-Hill Professional, 2003.

Carman 2009 CARMAN, A. *Performing a Denial of Service (DoS) Attack on a Phone Line*. 2009. <http://neverfear.org/blog/view/89/Performing_a_Denial_of_Service_DoS_Attack_on_a_Phone_Line>. [Online; Acessado 22-Novembro-2016].

Carman 2016 CARMAN, A. *Police arrested a hacker who allegedly triggered a DDoS attack on the 911 emergency call system*. 2016. <<http://www.theverge.com/2016/10/30/13471128/meetkumar-hiteshbhai-desai-arrest-911-exploit>>. [Online; Acessado 22-Novembro-2016].

Center 2013 CENTER, N. C. . C. I. *National Public Safety Telecommunications Council*. 2013. <http://www.npstc.org/download.jsp?tableId=37&column=217&id=2699&file=NCCIC_NCC_TDoS_Info_Bulletin_130328.pdf&page=NPSTCHome>. [Online; Acessado 22-Novembro-2016].

Chiappetta et al. 2013 CHIAPPETTA, S. et al. An anomaly-based approach to the analysis of the social behavior of voip users. *Comput. Netw.*, Elsevier North-Holland, Inc., v. 57, n. 6, p. 1545–1559, abr. 2013.

CISCO 2014 CISCO. *TDoS Attack Mitigation*. 2014. <http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/voice/cube_proto/configuration/15-mt/cube-proto-15-mt-book/voi-cube-tdos-attack-mitigation.pdf>. [Online; Acessado 28-Setembro-2015].

Clavel et al. 2007 CLAVEL, M. et al. *All About Maude: A High-Performance Logical Framework*. [S.l.]: Springer, 2007. (LNCS, v. 4350).

Conner e Nahrstedt 2008 CONNER, W.; NAHRSTEDT, K. Protecting sip proxy servers from ringing-based denial-of-service attacks. In: *Multimedia, 2008. ISM 2008. Tenth IEEE International Symposium on*. [S.l.: s.n.], 2008. p. 340–347.

Dantas, Nigam e Fonseca 2014 DANTAS, Y.; NIGAM, V.; FONSECA, I. A selective defense for application layer ddos attacks. In: *Intelligence and Security Informatics Conference (JISIC), 2014 IEEE Joint*. [S.l.: s.n.], 2014. p. 75–82.

DHS 2013 DHS. *Division of Homeland Security and Emergency Management: Situational Awareness Update*. 2013. <<http://krebsonsecurity.com/wp-content/uploads/2013/04/DHSEM-16-SAU-01-LEO.pdf>>. [Online; Acessado 28-Setembro-2015].

Dierks. e Rescorla 2008 DIERKS., T.; RESCORLA, E. *The Transport Layer Security (TLS) Protocol*. 2008. <<https://tools.ietf.org/html/rfc5246>>. [Online; Acessado 12-Janeiro-2016].

Digium 2015 DIGIUM, I. *SAsterisk Private Branch eXchange*. 2015. <<http://www.asterisk.org/>>. [Online; Acessado 28-Setembro-2015].

Eckhardt et al. 2012 ECKHARDT, J. et al. Stable availability under denial of service attacks through formal patterns. In: *FASE*. [S.l.: s.n.], 2012. p. 78–93.

Eckhardt et al. 2012 ECKHARDT, J. et al. Statistical model checking for composite actor systems. In: *WADT*. [S.l.: s.n.], 2012. p. 143–160.

- Ehlert et al. 2008 EHLERT, S. et al. Specification-based denial-of-service detection for sip voice-over-ip networks. In: *Internet Monitoring and Protection, 2008. ICIMP '08. The Third International Conference on*. [S.l.: s.n.], 2008. p. 59–66.
- FBI 2014 FBI. *Cyber Criminals Use Telephony Denial of Service (TDoS) Attacks to Extort Healthcare and Public Health Sector Employee*. 2014. <<http://voipsecurityblog.typepad.com/files/pin-20140702-001.pdf>>. [Online; Acessado 28-Setembro-2015].
- Ferdous 2014 FERDOUS, R. *Analysis and Protection of SIP based Services*. Tese (Doutorado) — Department of Information Engineering and Computer Science, University of Trento, Italy, 2014.
- Fuchs et al. 2008 FUCHS, C. et al. Detecting voip based dos attacks at the public safety answering point. In: *Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security*. [S.l.: s.n.], 2008. p. 148–155.
- Galiotos, Dagiuklas e Kotsopoulos 2015 GALIOTOS, P.; DAGIUKLAS, T.; KOTSOPOULOS, S. Call-level voip traffic modelling based on data from a real-life voip service provider. In: *2015 IEEE Globecom Workshops (GC Wkshps)*. [S.l.: s.n.], 2015. p. 1–7.
- GAYRAUD e JACQUES 2014 GAYRAUD, R.; JACQUES, O. *SIPp – SIP Traffic Generator*. 2014. <<http://sipp.sourceforge.net>>. [Online; Acessado 28-Setembro-2015].
- Geneiatakis, Vrakas e Lambrinoudakis 2009 GENEIATAKIS, D.; VRAKAS, N.; LAMBRINOUDAKIS, C. Utilizing bloom filters for detecting flooding attacks against {SIP} based services. *Computers Security*, v. 28, n. 7, p. 578 – 591, 2009.
- Guha, Daswani e Jain 2006 GUHA, S.; DASWANI, N.; JAIN, R. An Experimental Study of the Skype Peer-to-Peer VoIP System. In: *Proceedings of The 5th International Workshop on Peer-to-Peer Systems (IPTPS)*. [S.l.: s.n.], 2006.
- Guri, Mirsky e Elovici 2016 GURI, M.; MIRSKY, Y.; ELOVICI, Y. 9-1-1 ddos: Threat, analysis and mitigation. *CoRR*, abs/1609.02353, 2016.
- Ha et al. 2009 HA, D.-Y. et al. Design and implementation of sip-aware ddos attack detection system. In: *Proceedings of the 2Nd International Conference on Interaction Sciences: Information Technology, Culture and Human*. New York, NY, USA: ACM, 2009. (ICIS '09), p. 1167–1171. ISBN 978-1-60558-710-3.
- Hines 2007 HINES, M. *Attackers Get Chatty on VOIP*. 2007. <<http://www.pcworld.com/article/132389/article.html>>. Accessed: 2015-27-09.
- Huici, Niccolini e d'Heureuse 2009 HUICI, F.; NICCOLINI, S.; D'HEUREUSE, N. Protecting sip against very large flooding dos attacks. In: *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*. [S.l.: s.n.], 2009. p. 1–6.
- Hussain et al. 2013 HUSSAIN, I. et al. A lightweight countermeasure to cope with flooding attacks against session initiation protocol. In: *Wireless and Mobile Networking Conference (WMNC), 2013 6th Joint IFIP*. [S.l.: s.n.], 2013. p. 1–5.

- Hussain et al. 2015 HUSSAIN, I. et al. A comprehensive study of flooding attack consequences and countermeasures in session initiation protocol sip. *Sec. and Commun. Netw.*, John Wiley & Sons, Inc., New York, NY, USA, v. 8, n. 18, p. 4436–4451, dez. 2015. ISSN 1939-0114. Disponível em: <<http://dx.doi.org/10.1002/sec.1328>>.
- Jewett, Shrago e Yomtov 1980 JEWETT, J.; SHRAGO, J.; YOMTOV, B. *Designing Optimal Voice Networks for Businesses, Government, and Telephone Companies*. [S.l.: s.n.], 1980.
- Jeyanthi, Thandeeswaran e Vinithra 2014 JEYANTHI, N.; THANDEESWARAN, R.; VINITHRA, J. Rqa based approach to detect and prevent ddos attacks in voip networks. *Cybernetics and Information Technologies*, v. 14, p. 11–24, April 2014.
- Jones 2016 JONES, P. E. *SIPp – SIP Traffic Generator*. 2016. <<https://www.packetizer.com/ipmc/h323/standards.html>>. [Online; Acessado 11-Janeiro-2016].
- Kapravelos et al. 2010 KAPRAVELOLOS, A. et al. D(eli)aling with voip: Robust prevention of dial attacks. In: _____. *Computer Security – ESORICS 2010: 15th European Symposium on Research in Computer Security, Athens, Greece, September 20-22, 2010. Proceedings*. [S.l.]: Springer Berlin Heidelberg, 2010. p. 663–678.
- Khanna et al. 2012 KHANNA, S. et al. Adaptive selective verification: An efficient adaptive countermeasure to thwart dos attacks. *Networking, IEEE/ACM Transactions on*, v. 20, n. 3, p. 715–728, June 2012.
- Khanna et al. 2008 KHANNA, S. et al. Adaptive selective verification. In: *IEEE INFOCOM 2008 - The 27th Conference on Computer Communications*. [S.l.: s.n.], 2008. ISSN 0743-166X.
- Landau e Binder 2005 LANDAU, D.; BINDER, K. *A Guide to Monte Carlo Simulations in Statistical Physics*. New York, NY, USA: Cambridge University Press, 2005. ISBN 0521842387.
- Lemos et al. 2016 LEMOS, M. O. O. et al. A selective defense for mitigating coordinated call attacks. In: *34th Brazilian Symposium on Computer Networks and Distributed Systems (SBRC)*. [S.l.: s.n.], 2016.
- Li et al. 2010 LI, W. et al. On sliding window based change point detection for hybrid sip dos attack. In: *Services Computing Conference (APSCC), 2010 IEEE Asia-Pacific*. [S.l.: s.n.], 2010. p. 425–432.
- Lipowski e Lipowska 2011 LIPOWSKI, A.; LIPOWSKA, D. Roulette-wheel selection via stochastic acceptance. *CoRR*, abs/1109.3627, 2011.
- Marchal et al. 2015 MARCHAL, S. et al. Mitigating mimicry attacks against the session initiation protocol. *Network and Service Management, IEEE Transactions on*, v. 12, n. 3, p. 467–482, Sept 2015.
- Meadows 1999 MEADOWS, C. A Formal Framework and Evaluation Method for Network Denial of Service. In: *CSFW*. [S.l.: s.n.], 1999. p. 4–13.

Meseguer 2012 MESEGUER, J. Twenty years of rewriting logic. *J. Log. Algebr. Program.*, v. 81, n. 7-8, p. 721–781, 2012. Disponível em: <<http://dx.doi.org/10.1016/j.jlap.2012.06.003>>.

Messmer 2011 MESSMER, E. *Massive DDoS attacks a growing threat to VoIP services*. 2011. <<http://www.networkworld.com/article/2181743/voip/massive-ddos-attacks-a-growing-threat-to-voip-services.html>>. [Online; Acessado 28-Setembro-2015].

Meyer 1970 MEYER, P. *Introductory Probability and Statistical Applications*. Addison-Wesley Publishing Company, 1970. (Addison-Wesley series in statistics). ISBN 9780201047103. Disponível em: <<https://books.google.com.br/books?id=99NQAAAAMAAJ>>.

Microsoft 2016 MICROSOFT. *Skype*. 2016. <<http://www.skype.com/>>. [Online; Acessado 12-Janeiro-2016].

Noraini e Geraghty 2011 NORAINI, M. R.; GERAGHTY, J. Genetic algorithm performance with different selection strategies in solving tsp. 2011.

PINDROP 2014 PINDROP. *PINDROP: PROTECTING YOUR CALL CENTERS AGAINST PHONE FRAUD SOCIAL ENGINEERING*. 2014. <https://www.pindrop.com/wp-content/uploads/2016/01/pindrop_overview_whitepaper_fi_20141121_v2.pdf>. Accessed: 2015-27-09.

Polakis, Kontaxis e Ioannidis 2011 POLAKIS, I.; KONTAXIS, G.; IOANNIDIS, S. Captchuring automated (smart)phone attacks. In: *2011 Seventh European Conference on Computer Network Defense*. [S.l.: s.n.], 2011. p. 60–60.

RNP 2016 RNP. *Estatísticas fone@RNP*. 2016. <<http://estatisticasfone.rnp.br/Dashboard.html>>. Accessed: 2016-27-09.

Rosenberg et al. 2002 ROSENBERG, J. et al. *Session Initiation Protocol*. 2002. <<http://http://www.ietf.org/rfc/rfc3261.txt>>. [Online; Acessado 28-Setembro-2015].

Sampaio et al. Submetido SAMPAIO, G. et al. Mitigating asymmetric application-layer ddos attacks. Submetido.

Schulzrinne et al. 2003 SCHULZRINNE, H. et al. *RTP: A Transport Protocol for Real-Time Applications*. 2003. <<https://tools.ietf.org/html/rfc3550>>. [Online; Acessado 11-Janeiro-2016].

Securelogix 2013 SECURELOGIX. *The Surging Threat of Telephony Denial of Service Attacks*. 2013. <http://voipsecurityblog.typepad.com/files/tdos_paper_4-11-13.pdf>. [Online; Acessado 28-Setembro-2015].

Sen, Viswanathan e Agha 2005 SEN, K.; VISWANATHAN, M.; AGHA, G. On statistical model checking of stochastic systems. In: *CAV*. [S.l.: s.n.], 2005. p. 266–280.

SeVen <<https://github.com/ygdantas/SeVen.git>> 2016 SeVen <<https://github.com/ygdantas/SeVen.git>>. 2016.

- Shankesi et al. 2009 SHANKESI, R. et al. Model-checking dos amplification for voip session initiation. In: *Computer Security - ESORICS 2009, 14th European Symposium on Research in Computer Security, Saint-Malo, France, September 21-23, 2009. Proceedings*. [s.n.], 2009. p. 390–405. Disponível em: <http://dx.doi.org/10.1007/978-3-642-04444-1_24>.
- Sheluhin, Smolskiy e Osin 2007 SHELUHIN, O.; SMOLSKIY, S.; OSIN, A. *Self-Similar Processes in Telecommunications*. Wiley, 2007. ISBN 9780470062104. Disponível em: <<https://books.google.com.br/books?id=FfbMh1j4EbIC>>.
- slowloris 2013 SLOWLORIS. <<http://ha.ckers.org/slowloris/>>. 2013.
- slowread 2013 SLOWREAD. <<https://code.google.com/p/slowhttpstest/>>. 2013.
- Socolofsky 1991 SOCOLOFSKY, T. *A TCP/IP Tutorial*. 1991. <<https://tools.ietf.org/html/rfc1180>>. [Online; Acessado 04-Janeiro-2016].
- Stanek 2010 STANEK, J. *DoS and DDoS attacks on the SIP protocol*. Dissertação [U+FFFD] [U+FFFD] o (Mestrado) — Faculty of Mathematics and Physics, Charles University in Prague, 2010.
- Stanek e Kencl 2012 STANEK, J.; KENCL, L. Sip protector: Defense architecture mitigating ddos flood attacks against sip servers. In: *Communications (ICC), 2012 IEEE International Conference on*. [S.l.: s.n.], 2012. p. 6733–6738. ISSN 1550-3607.
- Tang, Cheng e Hao 2012 TANG, J.; CHENG, Y.; HAO, Y. Detection and prevention of sip flooding attacks in voice over ip networks. In: *INFOCOM, 2012 Proceedings IEEE*. [S.l.: s.n.], 2012. p. 1161–1169. ISSN 0743-166X.
- Tang et al. 2014 TANG, J. et al. Sip flooding attack detection with a multi-dimensional sketch design. *Dependable and Secure Computing, IEEE Transactions on*, v. 11, n. 6, p. 582–595, Nov 2014. ISSN 1545-5971.
- Tang, Cheng e Zhou 2009 TANG, J.; CHENG, Y.; ZHOU, C. Sketch-based sip flooding detection using hellinger distance. In: *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*. [S.l.: s.n.], 2009. p. 1–6. ISSN 1930-529X.
- TransNexus 2015 TRANSNEXUS. *Telephony Denial of Service Attacks*. 2015. <<http://transnexus.com/telephony-denial-service-attacks/>>. [Online; Acessado 16-Janeiro-2016].
- Wan, Li e Fan 2010 WAN, X.-Y.; LI, Z.; FAN, Z.-F. A sip dos flooding attack defense mechanism based on priority class queue. In: *Wireless Communications, Networking and Information Security (WCNIS), 2010 IEEE International Conference on*. [S.l.: s.n.], 2010. p. 428–431.
- Waters e Engineering 2007 WATERS, M.; ENGINEERING, O. S. U. E. *Traffic Characteristics of Optical Burst Switch Assembled Aggregated VOIP Traffic*. Oklahoma State University, 2007. ISBN 9780549470823. Disponível em: <<https://books.google.com.br/books?id=pnNoKQNXtHAC>>.

yet 2013 YET r-u-dead. <<https://code.google.com/p/r-u-dead-yet/>>. 2013.

Zargar e Moghaddam 2014 ZARGAR, R.; MOGHADDAM, M. An entropy-based voip flooding attacks detection and prevention system. In: *Computer and Knowledge Engineering (ICCKE), 2014 4th International eConference on*. [S.l.: s.n.], 2014. p. 691–696.

Zargar, Joshi e Tipper 2013 ZARGAR, S.; JOSHI, J.; TIPPER, D. A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks. *Communications Surveys Tutorials, IEEE*, v. 15, n. 4, p. 2046–2069, Fourth 2013. ISSN 1553-877X.

Zhang et al. 2009 ZHANG, H. et al. Detecting voip-specific denial-of-service using change-point method. In: *Advanced Communication Technology, 2009. ICACT 2009. 11th International Conference on*. [S.l.: s.n.], 2009. v. 02, p. 1059–1064. ISSN 1738-9445.

Zi-Fu, Jun-Rong e Xiao-Yu 2010 ZI-FU, F.; JUN-RONG, Y.; XIAO-YU, W. A sip dos flooding attack defense mechanism based on custom weighted fair queue scheduling. In: *Multimedia Technology (ICMT), 2010 International Conference on*. [S.l.: s.n.], 2010. p. 1–4.