Universidade Federal de São Carlos– UFSCar
Centro de Ciências Exatas e de Tecnologia– CCET
Departamento de Computação– DC
Programa de Pós-Graduação em Ciência da Computação– PPGCC

Emerson Rogério Alves Barea

Avaliação de Riscos de Segurança BGP na Internet com Emulação Escalável

São Carlos
2021

Emerson Rogério Alves Barea

Avaliação de Riscos de Segurança BGP na Internet com Emulação Escalável

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências Exatas e de Tecnologia da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação.

Área de concentração: Sistemas Distribuídos e Redes de Computadores

Orientador: Hermes Senger

São Carlos
2021

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Defesa de Tese de Doutorado do candidato Emerson Rogério Alves Barea, realizada em 15/12/2021.

Comissão Julgadora:

Prof. Dr. Hermes Senger (UFSCar)

Prof. Dr. Helio Crestana Guardia (UFSCar)

Prof. Dr. Paulo Matias (UFSCar)

Prof. Dr. Cesar Augusto Cavalheiro Marcondes (ITA)

Prof. Dr. Italo Fernando Scota Cunha (UFMG)

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.
O Relatório de Defesa assinado pelos membros da Comissão Julgadora encontra-se arquivado junto ao Programa de Pós-Graduação em Ciência da Computação.

Dedico o resultado desse trabalho à minha família, orientador, amigos e a Asgard.

Agradecimentos

Agradeço primeiramente à Deus pela minha vida, saúde e integridade física. À minha família, que ao mesmo tempo são pilares de sustentação, combustível e pousada para meu corpo e mente. A toda equipe de servidores do Departamento de Computação da Universidade Federal de São Carlos, que direta ou indiretamente são o alicerce para tudo que se produz de conhecimento nesse departamento. Aos amigos de Asgard e GSDR, de todas gerações que passaram por aqueles laboratórios, pois todos contribuíram de alguma forma com esse resultado final. Tenho certeza que sem vocês, nada ou muito pouco seria capaz.

Resumo

O *Border Gateway Protocol* (BGP) é o protocolo de roteamento predominante que permite a comunicação dos diferentes *Autonomous Systems* (AS) na Internet. Os ataques ao BGP geralmente têm um impacto significativo, pois podem causar indisponibilidade e comprometer dados confidenciais em grandes áreas geográficas. Devido à relevância do tema, diversos estudos foram desenvolvidos abordando diferentes áreas de pesquisa em segurança no BGP. Uma área de estudo que se destaca é a reprodução de *testbeds* para análise detalhada do protocolo, entretanto uma característica comum na validação dessas propostas é o uso de uma representação limitada em escala e fidelidade da estrutura da Internet. Essa limitação torna impossível ter uma visão global do risco associado aos ataques e a identificação dos métodos de mitigação. Diante desse cenário, tornou-se imprescindível realizar uma análise aprofundada dos riscos ao protocolo BGP no nível da infraestrutura real e global da Internet. Essa análise exige soluções que representem a topologia real da Internet e a reprodução precisa dos eventos de ataque em implementações reais de software BGP. Nessa linha, este trabalho apresenta a *Minimalistic Security* BGP (MiniSecBGP), uma *testbed* baseada em emulação, e uma metodologia para análise de risco da infraestrutura de roteamento das redes interdomínios. Nossa solução pode usar dados reais ou sintéticos sobre ataques, possibilitando um melhor entendimento dos ataques anteriores e em cenários hipotéticos, para identificar estratégias de mitigação mais robustas. Os testes de validação demonstraram que a MiniSecBGP reproduz fielmente cenários reais usando dados de topologia e eventos de ataque extraídos automaticamente de conjuntos de dados públicos ou criados sinteticamente pelo usuário.

Palavras-chave: BGP. Ataque. *Testbed*. Tecnologia de emulação leve e distribuída.

Abstract

Border Gateway Protocol (BGP) is the predominant routing protocol that allows different Autonomous Systems (AS) to communicate on the Internet. Attacks on BGP typically have a significant impact as they can cause unavailability and compromise sensitive data over large geographic areas. Due to the topic's relevance, several studies were developed that address different research areas on security in BGP. An area of study that stands out is the reproduction of testbeds for detailed protocol analysis. However, a common characteristic of validating the proposals is using a limited representation in scale and fidelity of the Internet structure. This limitation makes it impossible to have a global view of the risk associated with the attacks and identify the mitigation methods. Given this scenario, it became essential to carry out an in-depth analysis of the risks to the BGP protocol at the real and global Internet infrastructure level. This analysis demands solutions representing the real Internet topology and the accurate reproduction of the attack events on real BGP software implementations. This work presents the Minimalistic Security BGP (MiniSecBGP), an emulation-based testbed, and a methodology for risk analysis of interdomain network routing infrastructures. Our solution can use either real or synthetic data about attacks, which allows a better understanding of past attacks to "what-if" scenarios to identify more robust mitigation strategies. The validation tests demonstrated that MiniSecBGP faithfully reproduces real scenarios using topology data and attack events extracted automatically from public datasets or synthetically created by the user.

Keywords: BGP. Attack. Testbed. Lightweight and distributed emulation technology..

Lista de ilustrações

Lista de tabelas

Lista de siglas

API *Application Programming Interface*

ARIN *American Registry for Internet Numbers*

Ark *Archipelago*

ARP *Address Resolution Protocol*

AS *Autonomous System*

ASN AS *Number*

BGP *Border Gateway Protocol*

BRT BGP *Replay Tool*

BRITE *Boston university Representative Internet Topology gEnerator*

c2p *customer-to-provider*

CAIDA *Center for Applied Internet Data Analysis*

DoS *Denial of Service*

DDoS *Distributed* DoS

eBGP *external* BGP

FIB *Forwarding Information Base*

GUI *Graphical User Interface*

IANA *Internet Assigned Numbers Authority*

iBGP *internal* BGP

IEEE *Institute of Electrical and Electronics Engineers*

IPC *Interprocess Communication*

IXP *Internet Exchange Point*

IS-IS *Intermediate System to Intermediate System*

JSON *JavaScript Object Notation*

LG *Looking Glass*

LXC LinuX *Container*

LXD LinuX *container Daemon*

MiniSecBGP *Minimalistic Security* BGP

OSPF *Open Shortest Path First*

p2p *provider-to-provider*

PIB *Policy Information Base*

PID *Process Identification*

RAM *Random Access Memory*

RIB *Routing Information Base*

RIP *Routing Information Protocol*

RIPE *Réseaux IP Européens*

RPKI *Resource Public Key Infrastructure*

s2s *sibling-to-sibling*

SDN *Software Defined Networks*

SO Sistema Operacional

UTS UNIX *Timesharing System*

VLAN *Virtual Local Area Network*

VM *Virtual Machine*

Sumário

Capítulo 1

Introdução

1.1 Contexto

O *Border Gateway Protocol* (BGP) é o protocolo de roteamento que possibilita a comunicação entre as diferentes redes que formam a Internet, denominadas *Autonomous System* (AS) (REKHTER; LI; HARES, 2006). Por meio dele, são anunciados os blocos de endereçamento IP de cada rede, evitados *loops* e aplicadas as políticas de roteamento que são parte do negócio dos provedores dessas redes.

A segurança do BGP tem sido alvo de estudos recorrentes, entretanto eventos de falhas de segurança ainda ocorrem. Como exemplo, Shi et al. (2012) apresentaram um estudo sobre ataques de sequestro de prefixos de ASs, denominados *hijacking*, mostrando que aproximadamente 220 ataques desse tipo foram detectados em apenas um ano. Esse cenário piorou ao longo do tempo, com Mathurin (2021) apontando um crescimento de 11 vezes no número de ocorrências desse tipo de ataque em menos de dez anos, atingindo 2.255 ataques de sequestro em 2020. Além disso, ataques de *Denial of Service* (DoS), *Worms*, *Ransomware*, *Malwares*, *Spam*, entre outros, também são utilizados para explorar diretamente o protocolo e suas implementações (AKAMAI, 2018; ARBOR, 2019).

Ataques ao BGP normalmente causam grande impacto, pois geralmente estão relacionados à indisponibilidade e ao comprometimento de dados sensíveis em grandes áreas geográficas. Como exemplo, um caso amplamente divulgado na mídia ocorreu em 24 de fevereiro de 2008, onde o YouTube ficou inacessível para mais de uma centena de provedores asiáticos e europeus por 2 horas. Análise forense pós-incidente mostrou que Pakistan Telecom e PCCW, dois provedores da Ásia, passaram a anunciar o prefixo de endereços IP do Youtube, tornando inacessíveis seus serviços. Apesar da proporção desse evento, tal anúncio foi considerado acidental, sendo denominado como um evento de vazamento

de rotas (*route leak*) (RIPE, 2008).

Outros casos com menor repercussão, porém, não menos críticos, ainda acontecem. Um exemplo foi o ocorrido em 06 de junho de 2019, onde Safe Host SA e China Telecom anunciaram incorretamente mais de 70.000 rotas por até 2 horas, abrangendo cerca de 368 milhões de endereços IP vazados. Apesar desse evento não ter impossibilitado o acesso aos prefixos, todo tráfego destinado a eles foi desviado pela rede da China Telecom, gerando degradação no desempenho do acesso e representando um possível caso de interceptação de tráfego. Oficialmente não há conclusão se esta ocorrência foi acidental ou intencional (MADORY, 2019).

1.2 Problema

Devido a relevância do tema, diversos estudos que abordam áreas de pesquisa variadas sobre segurança em BGP vêm sendo desenvolvidos. Há estudos que focam no monitoramento de mensagens BGP (CLAFFY, 2012) ou utilizam essas mesmas mensagens para análise forense (LI et al., 2005; ZHAO et al., 2002). Há também trabalhos relacionados à validação da configuração BGP (WANG et al., 2012), correlação de informações de controle e plano de dados para identificar possíveis ataques (SHI et al., 2012), entre outros.

Nessa linha de estudos relacionados aos riscos associados ao BGP, uma área que se destaca é a que propõe a reprodução de ambientes para análise detalhada do protocolo. Por exemplo, existem trabalhos que empregam algum grau de simulação para prever os ataques possíveis e propor meios de mitigação (Karlin; Forrest; Rexford, 2006; ZHENG et al., 2007; ZHANG et al., 2008; COHEN et al., 2016), enquanto outros propõem a utilização de ambientes de testes, denominados *testbeds*, na reprodução de cenários com topologias BGP (ZHANG et al., 2005; LI; LILJENSTAM; LIU, 2009; SCHLINKER et al., 2014; AL-MUSAWI; BRANCH; ARMITAGE, 2017b), além de trabalhos que fazem uso de BGP para testes diretamente na Internet (KATZ-BASSETT et al., 2011), apesar dessa abordagem ser mais rara devido ao alto risco envolvido.

Apesar das boas iniciativas apresentadas, algumas características observadas nesses trabalhos, como: (*i*) reprodução de apenas uma área bastante limitada e não realista da topologia da Internet; (*ii*) utilização de tecnologias de roteadores BGP que não correspondem às empregadas em ambientes de produção; e (*iii*) a não reprodução fiel dos eventos de ataques e mecanismos de mitigação verificados nas ocorrências reais; mesmo não inviabilizando os resultados observados nesses trabalhos, limitam sua utilização na obtenção de uma visão do efeito dos ataques ao BGP no ambiente real da Internet, pois cada evento pode resultar em consequências e áreas de abrangência diferentes no ambiente da Internet, bem como os mecanismos utilizados para recuperar as áreas comprometidas e mitigar os riscos podem mudar em cada caso.

1.3 Hipótese

Diante desse cenário, tornou-se extremamente necessário realizar a análise profunda dos riscos ao protocolo BGP no nível da estrutura real e global da Internet. Esse tipo de estudo demanda:

a) Soluções que representem a topologia real da Internet;

b) Reprodução mais precisa dos eventos e da dinâmica dos ataques BGP;

c) Uso de implementações de software BGP reais (entregues como software aberto ou produtos comerciais) para experimentação de alta fidelidade.

Para isso, deve-se fazer uso de ambientes que garantam a representação da topologia dos ASs na Internet e reproduza os eventos de ataques ao BGP em uma base tecnológica compatível com a real, a fim de observar as consequências, identificar os pontos críticos e a área comprometida nos ataques ao BGP na topologia da Internet. Como resultado, espera-se que essas análises proporcionem mecanismos mais eficientes de mitigação do risco associado.

1.4 Objetivo e Motivação

Buscando preencher essa lacuna, este trabalho propõe uma metodologia e o desenvolvimento de uma *testbed* baseada em emulação que pode ser usada para análise de risco na infraestrutura de roteamento de redes interdomínios em escala global. Essa solução pode usar dados reais ou sintéticos sobre ataques, possibilitando uma melhor compreensão dos ataques anteriores até cenários do tipo "e se", possibilitando identificar estratégias de mitigação mais robustas. As principais contribuições desse trabalho podem ser detalhadas da seguinte forma:

a) Proposta de uma metodologia para estudo das vulnerabilidades a ataques ao BGP. Essa metodologia suporta tanto a análise de risco de cenários sintéticos, a reprodução de ataques reais ou uma combinação de ambos, em um ambiente controlado;

b) Combinar informações reais coletadas de conjuntos de dados públicos para inferir uma visão realista da topologia da Internet com a emulação leve, para reproduzir ataques em escala global com alta precisão;

c) Proposta e avaliação da *Minimalistic Security* BGP (MiniSecBGP), uma testbed que dá suporte à metodologia apresentada. A MiniSecBGP faz uso de emulação leve baseada em tecnologia de *containers* em um ambiente distribuído e controlado. Reproduz automaticamente topologias realistas da Internet a partir de dados históricos de rotas extraídos dos repositórios públicos dos projetos AS-Relationship, do *Center for Applied Internet Data Analysis* (CAIDA), e RIPEstat, da *Réseaux IP Européens* (RIPE), além de possibilitar a especificação de topologias sintéticas.

Para alcançar tal objetivo, este trabalho trouxe consigo quatro questões de pesquisa que nortearam as ações desenvolvidas e que estão apresentadas a seguir:

1. Quais são os requisitos de um ambiente controlado destinado ao estudo dos ataques ao BGP e mitigação dos riscos para a estrutura da Internet?

Primeiramente, por ser um ambiente controlado, entende-se que ele não deve fazer uso dos recursos reais da Internet, seja total ou parcial, de forma a não representar risco ao seu correto funcionamento e segurança. Ao mesmo tempo, por ser destinado ao estudo dos ataques ao BGP com a consequente identificação dos mecanismos de mitigação do risco, deve-se garantir meios para criação de topologias de rede compostas pelos ASs, com suas relações de vizinhanças devidamente representadas, passando pelos eventos de anúncios e retiradas de rotas BGP nesses ASs, até a execução de eventos de ataque ao protocolo.

A partir disso, esse ambiente deve suportar a criação de cenários compostos pela topologia, eventos BGP considerados normais e de ataque, com possibilidade de serem executados automaticamente ou inseridos em tempo de execução numa linha de tempo, além de disponibilizar mecanismos de monitoramento do ambiente em tempo de execução e com manutenção de histórico.

2. Quais características da estrutura da Internet devem ser reproduzidas nos ambientes de estudo do BGP, e com qual grau de precisão com relação aos ambientes reais?

É necessário reproduzir a estrutura da Internet com o maior realismo possível, para que os estudos possam ser realizados sobre uma base confiável.

A Internet é um conjunto de ASs, portanto, redes autônomas, descentralizadas, heterogêneas e totalmente distribuídas em diferentes áreas geográficas. Reproduzi-la não é uma tarefa trivial, pois, devem ser consideradas as limitações de escalabilidade das tecnologias disponíveis e a ausência de detalhes técnicos sobre a estrutura de cada AS real. Mesmo assim, a estrutura reproduzida deve conter a representação fidedigna da topologia real dos ASs, interligados através de enlaces de comunicação de dados que considerem a direção do fluxo da informação estabelecidos nos acordos comerciais existentes entre eles (GAO, 2001) ou que pelo menos representem de forma correta a relação de vizinhança existente entre os ASs. Os enlaces de dados também devem considerar, sempre que possível, as características da banda suportada, atraso e utilização média verificada nos ambientes reais.

Quanto ao negócio dos ASs, esses podem fornecer serviços variados, como hospedagem de aplicações, e-mail, acesso à navegação Web, voz sobre IP (VoIP), entre outros. O público alvo dos ASs também pode variar de acordo com a área geográfica onde o AS está instalado ou o serviço é oferecido.

Todas essas características são relevantes para a correta reprodução da estrutura da Internet, pois, servem como base de informação que possibilitam identificar o real risco associado a um evento de ataque ao BGP.

3. Como garantir a reprodução dos eventos de ataques ao BGP de maneira fidedigna aos casos reais?

Nesse caso, os eventos de ataque ao BGP podem ser considerados como toda ocorrência com potencial de interferir no correto funcionamento do protocolo, seja através do anúncio de prefixos por ASs que não têm direitos sobre eles, com a ação de *hijack* ou *route leak*; da retirada de anúncios de rotas no BGP, conhecido como *withdraw*; ou a inclusão indevida de ASs como prefixo no caminho da rota BGP, denominado *prepend*.

De maneira geral, pode-se acompanhar os detalhes dos casos de ataques ao BGP em portais Web de notícias ou notas técnicas de empresas especializadas no assunto, onde normalmente são reportados os detalhes dos ataques, como os métodos utilizados pelos atacantes, dia e hora dos eventos, área comprometida, entre outras informações, assim como o reportado em RIPE (2008). Porém, esse tipo de base de dados pode não conter todos detalhes técnicos da ocorrência, a ponto de inviabilizar a reprodução fidedigna do ataque. Além disso, pode ser difícil encontrar ou até mesmo inexistir esse tipo de informação para casos de ataques com menor repercussão.

Baseado nessa realidade, ganha relevância a utilização de fontes de dados que disponibilizam *traces* reais de atualizações dos anúncios do BGP ao longo do tempo na Internet, como o disponibilizado por RIPE (2021). Esse tipo de base de dados históricos dos anúncios BGP reporta as mudanças ocorridas nas rotas entre os ASs monitorados, possibilitando até mesmo inferir as políticas de roteamento internas de cada AS no caminho, ao ponto que os anúncios são aceitos ou negados por eles.

4. Como estudar a Internet com fidelidade, realismo e escalabilidade para entender o impacto de um ataque sobre sua estrutura?

O melhor meio para obter resultados confiáveis na análise de um ambiente é observando o comportamento do próprio ambiente em sua estrutura original. Porém, essa abordagem é cara e gera riscos que devem ser evitados. Nesse caso, a emulação de rede é uma opção que se mostra viável, pois suporta o uso de aplicações reais, como o complexo software de roteamento de borda existente nos ASs da Internet, sem o ônus do alto custo e da complexidade do ambiente físico. Paralelamente, por estar susceptível às nuances das implementações das aplicações de roteadores BGP utilizados em ambientes reais de produção, a emulação também pode refletir maior precisão nos resultados da reprodução de ambientes complexos e instáveis quando comparada à simulação (WHITE et al., 2002), (HUANG et al., 2014) e (YAN; MCKEOWN, 2017).

Ainda assim, emular uma estrutura tão grande e complexa, como a topologia da Internet, pode ser inviável sob o ponto de vista do poder e custo computacional envolvidos. Dessa forma, torna-se necessário analisar a necessidade ou não da utilização de soluções de emulação leve com suporte ao processamento distribuído em máquinas distintas na rede.

1.5 Contribuições da tese

Abaixo estão apresentadas as principais contribuições resultantes do desenvolvimento dessa tese. Também segue a relação dos trabalhos submetidos e publicados com os resultados intermediários e final:

a) Em Barea et al. (2018) nós definimos uma arquitetura para *testbed* capaz de reproduzir o grafo da topologia da Internet, com bom grau de realismo e com emulação automática dos roteadores BGP baseada em tecnologia de emulação leve;

b) Em Barea et al. (2019) nós realizamos a análise comportamental de soluções de emulação leve e distribuída para serem utilizadas como tecnologia base da *testbed*. Essa análise comparou o consumo de recursos computacionais (memória, disco e processador) das soluções Docker, LinuX *Container* (LXC), LinuX *container Daemon* (LXD) e Mininet, além dos recursos de rede e grau de automação da gestão de todo ambiente de emulação nessas tecnologias;

c) Em Barea et al. (2020) nós expandimos o trabalho anterior mensurando os tempos de instanciação de todo ambiente de emulação leve e distribuído para as soluções Docker, LXC, LXD e Mininet;

Além das contribuições citadas até aqui, disponibilizamos publicamente todo código necessário para a instalação e utilização da *testbed* em ambiente próprio do utilizador, além da documentação com instruções de uso e os dados crús com os códigos necessários para reprodução dos cenários de validação da *testbed* apresentados no Capítulo 7 deste documento.

A versão final dessa tese resultou na escrita de um artigo que será submetido ao IEEE *Access*, periódico multidisciplinar de acesso aberto do *Institute of Electrical and Electronics Engineers* (IEEE). Além disso, estamos analisando junto à Agência de Inovação da UFSCar a possibidade e viabilidade de registro de patente da metodologia proposta, registro do código do software da *testbed* e da marca *Minimalistic Security* BGP (MiniSecBGP).

1.6 Estrutura do Documento

O restante deste documento está organizado da seguinte forma:

a) *Capítulo 2*: apresenta conceitos fundamentais e tecnologias relevantes ao desenvolvimento deste trabalho;

b) *Capítulo 3*: apresenta outros trabalhos que tratam de problemas relacionados, discutindo suas limitações e como essas soluções diferem desta proposta;

c) *Capítulo 4*: detalha a arquitetura da *testbed*;

d) *Capítulo 5*: apresenta as tecnologias habilitadoras e os detalhes técnicos que nortearam a implementação da *testbed*;

e) *Capítulo 6*: apresenta as funções da MiniSecBGP;

f) *Capítulo 7*: apresenta a sequência de testes realizados para validar a MiniSecBGP através da reprodução de um caso real de ataque ao BGP;

g) *Capítulo 8*: conclui o trabalho e sugere trabalhos futuros para evolução dos estudos do tema principal.

Capítulo 2

Fundamentação Teórica

Neste capítulo são apresentados conceitos fundamentais e tecnologias relevantes ao desenvolvimento desta tese. A seção 2.1 trata do protocolo de roteamento BGP e apresenta detalhes da implementação de um roteador BGP amplamente utilizado em ambientes reais. A seção 2.2 apresenta informações sobre a reprodução de topologias realísticas da Internet; enquanto a seção 2.3 detalha as soluções de emulação leve e distribuída baseadas em *containers*, utilizadas como base do ambiente proposto para estudo do BGP.

2.1 BGP

O BGP é um protocolo de roteamento que tem como principal função trocar informações de acesso e alcançabilidade entre ASs. A comunicação é feita com a troca de mensagens entre pares de roteadores previamente configurados (*peers*) . Essas mensagens são processadas e os resultados armazenados em tabelas de roteamento que mantêm a informação do melhor caminho para cada destino (REKHTER; LI; HARES, 2006).

As mensagens BGP são transportadas em conexões TCP estabelecidas entre os *peers*, podendo ser dos seguintes tipos:

a) `OPEN`: são mensagens enviadas pelos *peers* logo após o estabelecimento da conexão TCP. Notificam sobre a intenção do roteador trocar informações BGP. Os *peers* devem confirmar o recebimento e aceitação desse tipo de mensagem para sequência do processo;

b) `UPDATE`: são mensagens utilizadas para transmitir informações de roteamento entre os *peers*. Essas mensagens são apenas incrementais, isso é, não são realizados reenvios periódicos. Podem anunciar rotas viáveis que compartilham um determinado

caminho ou retirar rotas inviáveis. Uma única mensagem `UPDATE` pode conter simultaneamente informações para inclusão (*announcement*) e exclusão (*withdraw*) de rotas. Os *peers* utilizam as mensagens `UPDATE` para construir um grafo descrevendo as relações entre todos os ASs.

Além dos prefixos de endereços IP a serem incluídos ou excluídos, as mensagens de `UPDATE` também podem conter um conjunto de atributos que são utilizados durante o processamento do algoritmo de decisão do BGP, denominado *Path Vector* (MEYER; PATEL, 2006), para determinar o melhor caminho. Os atributos mais utilizados são:

- `ORIGIN`: define se a origem do anúncio é um *peer* interno ao AS - *internal* BGP (iBGP), externo ao AS - *external* BGP (eBGP), ou por redistribuição de outros protocolos de roteamento - *INCOMPLETE*.
- `AS_PATH`: contém a sequência de ASs (*path*) até um determinado destino. É preenchido com o AS *Number* (ASN) do emissor do anúncio no lado mais a esquerda da lista. Esse atributo pode ser modificado pela técnica de *prepend* nos roteadores BGP.
- `NEXT_HOP`: endereço IP do próximo roteador para alcançar determinada rede.
- `MULTI_EXIT_DISC` (`MED`): sugere aos *peers* externos o caminho preferido de um AS com múltiplos pontos de entrada.
- `LOCAL_PREF`: propagado apenas entre *peers* iBGP. Determina a preferência de um caminho sobre outros.
- `COMMUNITY`: propagado entre *peers* eBGP. Identifica os anúncios que podem ser tratados de forma diferenciada por outros ASs seguindo suas políticas de roteamento internas (CHANDRA; TRAINA, 1996).

c) `KEEPALIVE`: mensagens trocadas periodicamente entre os *peers* informando que ainda estão em funcionamento. O não recebimento consecutivo de mensagens `KEEPALIVE` por parte de um *peer*, representa sua indisponibilidade. Nesse caso, as informações de rotas previamente aprendidas dessa origem são descartadas e o roteador envia novas mensagens `UPDATE` aos outros *peers* informando sobre a necessidade de exclusão dessas rotas;

d) `NOTIFICATION`: mensagens de notificação de erros trocadas entre os *peers*. A conexão BGP é encerrada imediatamente após seu envio.

De modo geral, tanto as mudanças no `AS_PATH` quando o *withdraw* de rotas podem ser causados por motivos variados, como alteração nas regras de filtro de algum AS intermediário, problemas nos enlaces físicos de dados ou até mesmo nas conexões TCP entre os *peers* BGP.

Quanto às mudanças no `AS_PATH` que incluem *prepend*, podem ocorrer no momento em que um `UPDATE` é recebido pelo roteador, conhecido como *prepend in*. Nesse caso, o

roteador que recebeu a atualização inclui uma ou mais repetições do último AS contido no `AS_PATH` recebido. No *prepend in* o caminho é modificado antes do roteador escolher a melhor rota, portanto, a mudança pode interferir no roteamento do próprio roteador que efetuou o *prepend*.

O outro tipo de *prepend*, conhecido como *prepend out*, ocorre quando o roteador inclui uma ou mais repetições de seu próprio ASN no `UPDATE` que vai anunciar. Nesse caso, o *prepend out* não afeta a escolha do caminho no roteador que realizou o *prepend*, mas é propagado aos seus *peers* no BGP.

Apesar dessas formas padronizadas para configurar o *prepend*, o roteador BGP de um AS também pode modificar o caminho incluindo qualquer sequência e ASNs no `AS_PATH`, com repetições ou de forma alternada. Além disso, também é possível excluir ASNs do `AS_PATH` em qualquer posição, modificando completamente o caminho na rede.

Ainda segundo Rekhter, Li e Hares (2006), ao receber uma mensagem de `UPDATE`, o *peer* inicia o processo de escolha do melhor caminho para os prefixos atualizados. Abaixo são apresentados os atributos e valores mais significativos na ordem de escolha do melhor caminho:

a) `LOCAL_PREF` com valor mais alto;

b) caminho gerado localmente por redistribuição ou agregação;

c) menor `AS_PATH` (menor número de ASs existentes no *path*);

d) menor `ORIGIN` (iBGP < eBGP < *INCOMPLETE*);

e) rota com `MED` mais baixo;

f) havendo rotas eBGP, descartar todas rotas iBGP;

g) rota com menor métrica para o roteador;

h) roteador com menor identificador (*Router* ID). O *Router* ID é o endereço IP mais alto do roteador, com preferência aos endereços de *loopback*; e

i) rota recebida do *peer* com menor endereço IP de interface.

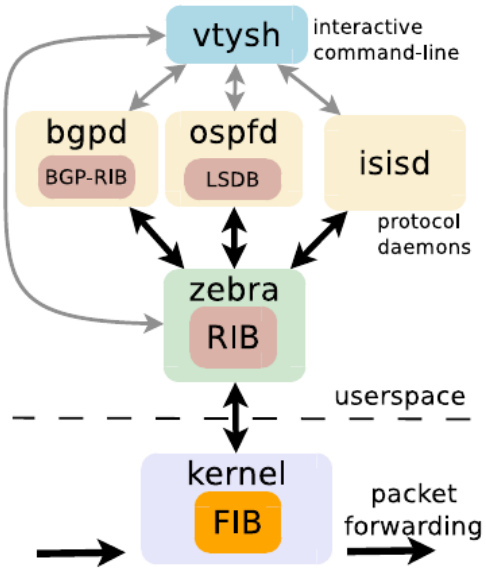
Apesar de todo detalhamento dos critérios para escolha do melhor caminho, os autores argumentam que essas são sugestões para o processo de troca de mensagens BGP e definição do melhor caminho, mas que as implementações podem conter as adaptações que os desenvolvedores julgarem necessárias para o correto funcionamento do BGP. Dessa forma, buscando entender o comportamento e escalabilidade suportados por uma implementação de roteador BGP amplamente utilizada em ambientes reais, a seção 2.1.1 traz o detalhamento da implementação do protocolo BGP na suíte de roteamento Quagga[1].

[1] Quagga Routing Suite. https://www.nongnu.org/quagga/

2.1.1 Detalhes de Implementação do BGP - Quagga

A suíte de roteamento Quagga consiste no conjunto de processos de protocolos de roteamento que se comunicam por chamadas de sistema. Implementa protocolos de roteamento comuns, como o BGP, o *Open Shortest Path First* (OSPF), o *Routing Information Protocol* (RIP), o *Intermediate System to Intermediate System* (IS-IS), entre outros. Sua arquitetura está centrada no serviço `Zebra`, que age como uma interface entre o plano de dados do kernel do Linux e os processos dos protocolos de roteamento (JAKMA; LAMPARTER, 2014). Figura 1.

Figura 1 – Arquitetura Quagga



Fonte: Jakma e Lamparter (2014)

O Quagga utiliza o protocolo `Zserv` para comunicação entre os processos, além de possuir uma ferramenta de interação baseada em linha de comando, o `vtysh`, que possibilita a configuração e o monitoramento dos protocolos de roteamento.

O processo `Zebra` mantém uma cópia da base de informações de encaminhamento, *Forwarding Information Base* (FIB), contendo o estado de encaminhamento de pacotes do kernel com as informações das interfaces de rede e a tabela de rotas atualmente ativas.

O `Zebra` também reúne as informações recebidas dos processos dos protocolos de roteamento (`bgpd`, `ospfd` e `isisd`, Figura 1), armazenando-as juntamente com sua imagem da FIB em sua própria base de informações de roteamento, denominada *Routing Information Base* (RIB). Com a RIB populada, o `Zebra` seleciona a melhor rota entre todas disponíveis para cada destino e atualiza a FIB. Além disso, a informação sobre as melhores rotas atuais e as alterações no estado das interfaces de rede podem ser encaminhadas aos protocolos de roteamento para atualização.

No Quagga, o processo `bgpd` do protocolo de roteamento BGP realiza três ações básicas (JAKMA; LAMPARTER, 2014; RAMANATH, 2004),

a) recebe anúncios de prefixos salvando-os em sua RIB;

b) anuncia a descoberta de novas rotas (atualizações) para seus *peers*; e

c) comunica o `Zebra` sobre os caminhos que foram adicionados à RIB.

Para isso, o `bgpd` replica toda estrutura de tabelas utilizadas para armazenar as rotas recebidas de cada *peer*. Uma forma de reduzir o consumo de memória do roteador local é colocando todos *peers* que compartilham as mesmas políticas de roteamento, em um mesmo grupo, pois, dessa forma, todos compartilharão a mesma estrutura de tabelas.

No Quagga, a tabela RIB do `bgpd` é uma coleção de RIBs independentes para cada família e tipo de endereços (IPv4, IPv6, unicast, multicast etc), encabeçadas por uma tabela raiz denominada `bgp_table`, que armazena referências para os três tipos de RIBs. Os tipos são:

a) *Adjacent RIB Incoming* (`Adj-RIB-In`): armazena as rotas recebidas nos anúncios de outros *peers* em mensagens `UPDATE`. Ao receber um prefixo, esse deve ser incluído ou alterado, caso já exista previamente na `Adj-RIB-In`. O processo oposto ocorre para mensagens de remoção.

b) *Local RIB* (`Loc-RIB`): armazena as rotas selecionadas da `Adj-RIB-In` de acordo com as definições contidas na base de políticas de roteamento, *Policy Information Base* (PIB), predefinidas no BGP ou manualmente inseridas; e

c) *Adjacent RIB Outcoming* (`Adj-RIB-Out`): armazena as rotas recebidas da `Loc-RIB`, que serão propagadas para *peers* específicos nas mensagens `UPDATE`, respeitando as políticas existentes na PIB.

2.1.2 Tabela de Roteamento *versus* Utilização de Memória RAM

O entendimento dos requisitos de memória, vazão de rede e processamento dos roteadores que suportam o protocolo BGP é algo que preocupa os responsáveis por ASs há tempos. Devido a relevância do tema, que afeta diretamente a escalabilidade de todo ambiente BGP, foram desenvolvidos modelos cujo objetivo era identificar e prever o consumo dos recursos baseados em métricas, como a quantidade de prefixos divulgados, o número de ASs participantes da rede, entre outras (REKHTER; WATSON, 1991; MEYER; PATEL, 2006).

Como exemplo, a Equação 1 ilustra os requisitos de memória (MR) de um roteador executando BGP (MEYER; PATEL, 2006). Nela, o número médio de rotas anunciadas por cada *peer* é identificado como N, o número total de `AS_PATH` únicos é A, a distância média de ASs da Internet como M (quantidade média de ASs nos `AS_PATH`), o número de octetos necessários para armazenar uma rede é R, o número de bytes necessários para

armazenar um AS num `AS_PATH` como P e a fração do número de *peers* que podem acessar cada rede como S.

$$MR = (((N * R) + (M * A) * P) * S) \quad (1)$$

Porém, devido a natureza dinâmica das relações comerciais entre os ASs, a não obrigatoriedade dos desenvolvedores em seguir rigorosamente as definições formais do BGP em suas implementações, a possibilidade de customização das configurações nos roteadores BGP e a falta de uma base contendo informações atualizadas e completas sobre os relacionamento de cada AS, tornam esses modelos imprecisos.

Por esse motivo, os próprios desenvolvedores de soluções de roteamento BGP, muitas vezes publicam exemplos de casos de uso com os valores de consumo de recursos de seus roteadores em relatórios técnicos ou postagens em fóruns relacionados às suas tecnologias.

Como exemplo, segundo dados da Cisco, o tamanho das tabelas de rotas BGP variam dependendo do número de prefixos aprendidos, e exemplos apontam utilização entre 28MB a 71MB de memória para cada 100K prefixos armazenados (CISCO, 2008). Da mesma forma, resultados de testes empíricos realizados junto ao *Looking Glass* (LG) do *Internet Exchange Point* (IXP) de São Paulo, por exemplo, apontaram um consumo aproximado de 74MB de memória para o armazenamento de pouco menos de 110K prefixos IPv4[2].

A princípio esse consumo de memória pode não parecer relevante, porém, quando pensamos em uma *testbed* responsável por mimetizar a topologia da Internet, devemos considerar a instanciação de centenas ou até milhares de roteadores, cada um com sua respectiva tabela de roteamento BGP, o que pode tornar o consumo total de memória impeditivo à esse tipo de implementação.

2.2 Representação da Topologia da Internet

Um ponto relevante para a representação de um ambiente BGP realista é a correta definição da topologia reproduzida, pois, ela deve ser uma réplica fidedigna da porção da topologia da Internet analisada. Infelizmente, mapear a topologia real, caracterizá-la e desenvolver modelos que capturam o comportamento emergente da Internet não é algo trivial, visto que há mais do que simples ligações físicas e meios de comunicação bem definidos que precisam ser considerados.

2.2.1 Geradores de Topologias Sintéticas

Historicamente alguns modelos vêm sendo utilizados com a finalidade de reproduzir a topologia da Internet. Alguns são baseados em pesquisa empírica por *traceroute*

[2] Acesso ao LG SP via telnet na data de 20 de dezembro de 2017 e captura dos dados com os comandos `show ip bgp` e `show bgp memory`.

(MADHYASTHA et al., 2006; SHAVITT; ZILBERMAN, 2010), mas de maneira geral apresentam problemas de escalabilidade e de ordem técnica, como por exemplo, em ambientes que suportam compartilhamento de carga ou mudanças de rotas durante o *traceroute* (WILLINGER, 2010).

Outro modelo bastante utilizado é o baseado em grafos aleatórios, iniciando com Erdos-Renyi e Gilbert (GILBERT, 1959), passando por Waxman (WAXMAN, 1988) e, posteriormente, os grafos de lei de potência (MEDINA; MATTA; BYERS, 2000).

Na linha de geradores de topologias, algumas ferramentas se destacam, como por exemplo o GT-ITM (CALVERT; DOAR; ZEGURA, 1997) e Inet (JIN; CHEN; JAMIN, 2000), considerados geradores de topologia orientados a modelos, ou seja, utilizam algum dos modelos apresentados anteriormente na geração da topologia; e o *Boston university Representative Internet Topology gEnerator* (BRITE) (MEDINA et al., 2001), reconhecido como precursor dos geradores de topologia universal, isto é, não está limitado apenas a um modelo específico de topologia. O BRITE suporta a leitura dos parâmetros de geração da topologia a partir de um arquivo de configuração que pode ser escrito diretamente pelo usuário, gerado automaticamente em sua *Graphical User Interface* (GUI), importado de outros geradores de topologias ou a partir de dados topológicos coletados diretamente da Internet.

2.2.2 Bases de Dados de Rotas como Geradores de Topologias

Outra abordagem que possibilita a representação de um ambiente BGP real é a inferência da topologia a partir de análises diretas das tabelas de rotas disponíveis em LGs espalhados pela internet (BRITO et al., 2015), porém esse procedimento é dispendioso e não contempla a totalidade de caminhos existentes entre os ASs.

Uma segunda alternativa é obter a topologia a partir de inferências oriundas dos *traces* de rotas armazenadas da Internet em projetos como o RouteViews[3]. Nesse caso, além de possibilitar consultas diretas à sua base de dados históricos, o Projeto RouteViews também é utilizado como fonte de dados para outros projetos mais especializados, como o Projeto AS-Relationship[4] do CAIDA (DIMITROPOULOS et al., 2007).

O Projeto AS-Relationship disponibiliza um repositório de dados de relacionamento entre ASs com vasta topologia mapeada. Os dados são organizados em bases atualizadas periodicamente, totalizando 72.813 ASs registrados na base de setembro de 2021, com a informação de conectividade entre eles composta por 501.913 enlaces de comunicação identificados com o tipo de relacionamento comercial acordado.

Os tipos de relacionamento são tipificados seguindo o acordo comercial firmado entre pares de ASs seguindo os modelos propostos por Luckie et al. (2013), que especializa o modelo de cone de clientes inicialmente proposto por Gao (2001), com os relacionamentos

[3] University of Oregon RouteViews Project. http://www.routeviews.org
[4] AS Relationship. http://www.caida.org/data/as-relationships/

entre ASs extraído dos projetos *Archipelago* (Ark)[5] e Peering Multilateral, de Giotsas et al. (2013).

Como resultado, temos os seguintes tipos de acordos comerciais identificados:

a) *customer-to-provider* (c2p): corresponde aos ASs que compram serviço de trânsito para tráfego destinado a partes da Internet em que eles não possuem conexão direta nem podem alcançar através de seus clientes. Esse tipo de relacionamento permite apenas tráfego com origem no cliente (*customer* - `c`), passando pelo provedor de trânsito (*provider* - `p`), para ser encaminhado ao destino, e nunca o inverso. A direção do tráfego respeita o acordo comercial firmado entre os ASs;

b) *provider-to-provider* (p2p): são os links que conectam dois provedores que concordam em trocar tráfego, próprio e de seus clientes, sem custo para ambos. A direção do tráfego é duplo nesse tipo de relacionamento, porém, a origem é controlada;

c) *sibling-to-sibling* (s2s): corresponde aos links que conectam dois ASs pertencentes administrativamente à mesma empresa. A direção do tráfego é dupla nesse tipo de relacionamento e a origem permitida é apenas entre os próprios ASs e seus clientes.

Seguindo a definição dos tipos de acordos comerciais apresentados, cada registro existente nas bases do Projeto AS-Relationship podem seguir o modelo `<provider-as>|<customer-as>|-1`, para relacionamentos do tipo `c2p`, e `<peer-as>|<peer-as>|0|<source>`, para relacionamentos do tipo `p2p`.

Ainda na linha de utilizar *traces* de rotas armazenadas da Internet para inferir a topologia, outro projeto de destaque é o RIPEstat[6], da RIPE (RIPE, 2021). Esse projeto oferece uma *Application Programming Interface* (API) que suporta consultas parametrizadas à sua base de dados por meio de requisições Web. Alguns dos parâmetros suportados nessa consulta são: o intervalo inicial e final dos eventos BGP observados na rede, os ASs que anunciam as atualizações de rotas, os prefixos IP anunciados, entre outros. As consultas à API retornam um arquivo do tipo *JavaScript Object Notation* (JSON) contendo as seguintes informações:

a) `START_TIME` e `END_TIME`: informam o intervalo entre a data de início e final dos eventos reportados. É formada pelo ano, mês, dia, hora, minuto e segundo;

b) `INITIAL_STATE`: apresenta as informações do estado inicial dos ASs coletores de rotas da RIPE, contendo o `AS_PATH` e `COMMUNITY` relacionados com cada prefixo IP informado como parâmetro na consulta;

c) `EVENTS`: composto pela relação dos eventos ocorridos numa linha de tempo entre o `START TIME` e `END TIME`. Contém o momento da ocorrência na linha do tempo; o tipo de evento, que pode estar entre o anúncio de um novo `AS_PATH` simples ou um `AS_PATH` com *prepend*, ou ainda a retirada de uma rota; o identificador do coletor

[5] Archipelago (Ark) Measurement Infrastructure. https://www.caida.org/projects/ark/
[6] RIPEstat Project. https://stat.ripe.net/

que recebeu a atualização na rede, o prefixo IP atualizado, o `AS_PATH` e `COMMUNITY` relacionados com cada prefixo IP informado como parâmetro na consulta;

d) `NODES`: contém a lista dos ASs que participam da topologia de alguma forma, podendo ser ASs coletores da RIPE ou ASs que participam algum `AS_PATH` anunciado no BGP;

e) `SOURCES`: contém a lista dos ASs coletores da RIPE, com suas informações de identificação na rede.

2.3 Soluções de Emulação Leve e Distribuída

Para a completa reprodução de um ambiente BGP capaz de representar a Internet, além do perfeito entendimento sobre o funcionamento do protocolo BGP e da definição de uma topologia fidedigna à real, é necessário que a tecnologia utilizada como base do ambiente seja suficientemente escalável.

Essa tecnologia deve ser capaz de emular topologias extensas e dinâmicas, suportando um grande número de roteadores BGP e enlaces de comunicação que repliquem características reais, como banda suportada, comportamento de atraso e estabilidade dos enlaces. Para isso, são considerados requisitos a utilização de soluções de baixo custo e sem restrição de uso, além de suporte a automação do *setup* do ambiente, contribuindo para a agilidade das análises e maior acurácia nos resultados.

O consumo de recursos inerentes à própria emulação é um ponto relevante a ser considerado, por isso, deve-se dar preferência ao uso de tecnologias de emulação leve baseadas em *containers* que suportem aplicações de rede reais e de processamento distribuído.

2.3.1 Virtualização Leve

O conceito de virtualização não é algo recente. Tanto as plataformas quanto a camada de software responsável por abstrair e compartilhar o hardware com os níveis de isolamento necessário entre as funções virtualizadas evoluíram, passando da necessidade de ter S.Os. completos virtualizados sobre hipervisores com alta sobrecarga de funções, para a virtualização leve baseada no isolamento de processos na área de usuário sobre um mesmo kernel (*containers*) (GANESH et al., 2012), fazendo uso do conceito de *namespaces* no isolamento de funções e *Control Groups* (CGroups) no controle de recursos (DANIELS, 2009).

De maneira geral, os *containers* são processos Linux atrelados a *namespaces* com funções específicas e com os requisitos de recursos garantidos pelo *Control Group* (CGroup). A finalidade de cada *namespace* é envolver um recurso de sistema global em uma abstração que faça parecer aos processos que eles têm sua própria instância isolada desse recurso, dando a ilusão de que são os únicos no sistema. Para isso, cada *namespace* faz

uso da atribuição de um valor ao parâmetro `CLONE_NEW()`, onde `()` recebe o identificador exclusivo do *namespace*, que é passado às três chamadas de sistema (`clone()`, `unshare()` e `setns()`) responsáveis pela criação ou ligação de cada processo ao *namespace* correspondente (KERRISK, 2013). Já o CGroup é uma solução de gerenciamento de recursos que fornece um mecanismo para agregar ou particionar conjuntos de processos, e processos filhos, em grupos hierárquicos com comportamento especializado. Este comportamento está relacionado à limitação, priorização, contabilização e controle da alocação dos recursos da máquina, como memória, CPU, dispositivos de I/O, aos respectivos grupos (BROWN, 2014).

Devido suas características, a virtualização por *container* mostrou-se viável como base para a reprodução de redes completas, principalmente por suportar a emulação de ambientes inteiros que possibilitam o estudo de cenários variados e complexos, com custo geral bastante reduzido e precisão nos resultados (YAN; MCKEOWN, 2017). Atualmente existe uma variedade de soluções que implementam esse tipo de virtualização, sendo que algumas são bem conhecidas e amplamente utilizadas nos mais variados cenários.

O LXC é um exemplo clássico de virtualização leve baseada em *container*. Faz uso dos *namespaces*, *Interprocess Communication* (IPC), UNIX *Timesharing System* (UTS), *Process Identification* (PID), rede e usuários, garantindo controle dos recursos do *container* mesmo quando criado em modo não-privilegiado. Implementa *chroot* para criar uma abstração do ponto de montagem do sistema de arquivos raiz do *container* (*pivot_root*), utiliza CGroup e outros mecanismos de segurança e isolamento do processo, possui uma API responsável pela integração com a *liblxc* no gerenciamento de todo ciclo de vida e controle dos *containers*, além de suportar a criação de novas instâncias a partir de imagens de S.O. diferentes da utilizada na máquina hospedeira, desde que suportada pelo kernel e respeitada a arquitetura (CANONICAL, 2018?a).

Outra solução baseada em *container* é o LXD, que consiste num serviço que expõe uma API REST através de *socket* Unix ou rede, fazendo uso de uma ferramenta de linha de comando para gerenciar *containers* LXC via *liblxc*. O LXD é basicamente uma alternativa às ferramentas de administração do LXC, com algumas funções extras, como acesso direto a dispositivos da máquina hospedeira, transferência de imagens de S.O. e *containers* entre máquinas distintas, gerenciamento de rede e armazenamento. Além da API padrão, o LXD ainda é suportado pelo projeto *nova-lxd* do OpenStack (CANONICAL, 2018?b).

Assim como o LXD, o Docker também surgiu como um serviço acessível via API REST através de *socket* Unix e rede para administração de *containers* LXC. Passou a utilizar a *libcontainer* no gerenciamento direto de *containers* a partir da versão 0.9, tornando opcional o uso de *libvirt*, LXC ou *systemd-nspawn* (HYKES, 2014). Suporta o download de imagens pré-configuradas de seu repositório (*Docker Hub*) e o empacotamento de *containers* com suas dependências para criação de novas imagens. Suporta *copy-on-write* com o compartilhamento do sistema de arquivos entre *containers* (UnionFS), onde

somente o conteúdo modificado é replicado entre as diferentes instâncias, aumentando seu desempenho enquanto diminui o consumo de disco da máquina hospedeira (MERKEL, 2014).

Além das soluções baseadas em *containers* que implementam um grande conjunto de mecanismos de isolamento, como apresentado anteriormente, existem outras ferramentas desenvolvidas com foco específico na emulação para experimentação em rede, como o Mininet.

O Mininet diferencia-se de outras implementações de *container* principalmente por possuir apenas os *namespaces* de rede e de ponto de montagem atrelados a um processo Linux, conferindo menor sobrecarga de funções a cada *container*. Faz uso de CGroups para controle de recursos (banda e latência), suporta o gerenciamento de switches e controladores de *Software Defined Networks* (SDN) com uso do *Open vSwitch*, interconectando todos elementos da rede emulada com interfaces virtuais (*veth*). Suporta aplicações de rede reais e possui uma API acessível por linha de comando, e programável via Python, possibilitando a administração de todo ciclo de vida e controle dos elementos de um experimento.

Como resultado final das características dessas soluções, de maneira geral elas apresentam consumo de recursos computacionais reduzido, porém, devido as diferenças de implementação, aliado aos recursos de *namespaces* utilizados por cada uma, essas soluções podem apresentar requisitos computacionais diferentes uma das outras.

2.3.2 Virtualização Leve e Distribuída

Quando o conceito de virtualização leve é expandido para ambiente distribuído, é necessário analisar os mecanismos de conexão entre *containers* criados em máquinas distintas, assim como as técnicas para ajuste dos parâmetros de banda e atraso dos enlaces virtuais entre eles.

O LXD, por exemplo, suporta a criação de túneis GRE e VXLAN para comunicação entre *containers* em máquinas distintas. O processo de criação dos túneis e distribuição dos *containers* não é automatizado, havendo necessidade de ser definido e solicitado pelo utilizador através da API. Outra forma de gestão desse ambiente distribuído é integrando o LXD com o projeto *Neutron* do OpenStack, porém, esse procedimento representa aumento significativo de complexidade e consumo de recursos computacionais do ambiente como um todo. A API do LXD não suporta a configuração dos parâmetros de banda e atraso de links (GRABER, 2016).

O Docker possui um ecossistema para gerenciamento de servidores em *cluster* (Docker Cloud), que suporta a automação da estratégia de distribuição de *containers* em seus nós. Para isso, faz uso dos algoritmos *Emptiest node*, que distribui os *containers* nos nós menos utilizados no momento da criação; *High availability*, que cria os *containers* de um mesmo serviço num único nó considerado menos utilizado no momento da criação; e o *Every node*,

que cria um *container* em cada nó participante do *cluster*. A interconexão dos *containers* em máquinas distintas é feita utilizando *plugins*, através da *libnetwork*, suportando a configuração de parâmetros de rede de acordo com a necessidade do utilizador; ou redes *overlay*, através de túneis VXLAN gerenciados pelo *Swarm*[7]. O *Swarm* não suporta parametrização de banda e atraso dos links nativamente (DOCKER, 2018).

Quanto ao Mininet, passou a suportar processamento distribuído utilizando túneis SSH na versão 2.2.0 de sua distribuição oficial, e túneis GRE na versão 2.3.0, suportando maior vazão por não depender do TCP no transporte de dados. Essa versão distribuída do Mininet é denominada Mininet Cluster.

No Mininet Cluster, a criação dos túneis nas máquinas hospedeiras é feita automaticamente quando informada a existência de um enlace entre dois elementos Mininet em sua API, porém, esse procedimento não suporta a configuração dos parâmetros de banda e atraso dos enlaces (LANTZ; O'CONNOR, 2015). Possui algoritmos predefinidos para distribuição automática dos elementos nos nós do *cluster*, sendo que, o *SwitchBinPlacer* distribui switches e controladores em blocos de tamanho uniforme baseado no tamanho do *cluster*, tentando alocar os *containers* de hosts no mesmo servidor em que estão os switches; e o *RandomPlacer* faz a distribuição randômica dos elementos pelo *cluster* (BURKARD, 2014).

Outra implementação de Mininet distribuído é o MaxiNet (WETTE et al., 2014), que consiste em uma API atuando como uma camada sobre o Mininet padrão. No MaxiNet, um nó central, denominado *Frontend*, invoca comandos nos nós remotos, denominados *workers*, gerenciando os elementos Mininet localizados na rede através de um servidor de nomes para objetos remotos do Python, o *PYthon Remote Object* versão 4 (Pyro4).

O MaxiNet utiliza túneis GRE para conexão remota entre elementos Mininet, porém, esses túneis são suportados apenas por elementos do tipo *switch*. Suporta a configuração dos parâmetros de banda e latência dos links diretamente em sua API, e cria automaticamente os túneis GRE nas máquinas hospedeiras. Faz uso da biblioteca METIS (KARYPIS; KUMAR, 1995) para particionamento do grafo da topologia emulada, criando partições com pesos equivalentes em todos nós do *cluster*, agregando a maior parte do tráfego emulado localmente nos *workers* através do critério de corte mínimo, baseado na banda dos links da topologia.

[7] https://docs.docker.com/engine/swarm/

Capítulo 3

Trabalhos Relacionados

Neste capítulo são apresentados o resumo dos trabalhos relacionados ao tema principal desta tese. A seção 3.1 apresenta os tipos de ataques mais comuns ao BGP; a seção 3.2 apresenta os mecanismos de proteção e mitigação dos riscos associados a esses ataques; a seção 3.3 traz exemplos de ambientes que suportam a reprodução de topologias BGP para o estudo do protocolo; e a seção 3.4 apresenta as limitações e sumário dos trabalhos relacionados.

3.1 Tipos de Ataques ao BGP

As ameaças ao protocolo de roteamento BGP são variadas, assim como são numerosos os estudos que discutem o tema. Realizar um levantamento exaustivo de todos riscos associados ao BGP não é uma tarefa trivial, portanto, é necessário delimitar o escopo da abordagem. Devido as diferenças entre os procedimentos e técnicas de ataque ao BGP, é comum que as ameaças sejam organizadas em classes para melhor identificá-las. Como exemplo, Mitseva, Panchenko e Engel (2018) criaram uma taxonomia que divide os ataques ao BGP em três grupos principais: ataque de falsificação de dados, manipulação do protocolo e mau uso dos dados.

Quanto à falsificação de dados, um ataque com grande potencial de dano é o *hijacking*. Ele pode ser intencional, quando um atacante anuncia indevidamente o prefixo ou sub-prefixo do alvo (também chamado de sequestro ou vazamento); ou não intencional, quando esses anúncios ocorrem por erros de configuração ou problemas no roteador BGP (CHO et al., 2019). Estatísticas mostram que aproximadamente 20% dos eventos de sequestros de prefixos de endereço IP e configurações incorretas em ambientes BGP duram menos

de 10 minutos, porém, são capazes de poluir até 90% da Internet em menos de 2 minutos (SHI et al., 2012).

O ataque de *hijacking* tem como foco principal atrair ou interceptar o tráfego do AS alvo. Sob essa abordagem, o tráfego atraído não precisa obrigatoriamente ser entregue ao destinatário original, podendo causar o isolamento do alvo e perda dos dados. Já na interceptação, a entrega dos dados ao alvo deve ser garantida, portanto, é necessário que o atacante mantenha rotas válidas ao AS destino (GOLDBERG et al., 2010).

Algumas consequências dos ataques de *hijacking* são:

a) causar *Distributed* DoS (DDoS) ao inundar o enlace do alvo, ou de algum AS de trânsito, com o tráfego desviado. Nesse caso, o número de ASs afetados pode ser grande se o enlace saturado for utilizado por muitos clientes (SRIRAM; MONTGOMERY, 2018);

b) desviar o tráfego para interceptar dados em algum ponto da rede que o atacante tenha acesso (*man-in-the-middle*) (BALLANI; FRANCIS; ZHANG, 2007; PILOSOV; KAPELA, 2008);

c) propagar *phishing* e *spam*. Nesse caso o atacante explora um prefixo de rede considerado legítimo, propagando campanhas durante o período do sequestro (VERVIER; THONNARD, 2013).

Com relação aos protocolos, tanto o TCP quanto o próprio BGP são susceptíveis a ataques. No caso do TCP, por exemplo, os ataques de inundação de *SYN* (EDDY, 2007) e *RST* (WATSON, 2003) podem gerar DoS nos roteadores de borda por esgotamento de recursos, assim como possibilitar o redirecionamento do tráfego. Além disso, o atacante pode aproveitar a ausência de mecanismos que garantam a confidencialidade no TCP para observar, de forma passiva, os dados transmitidos no BGP, a fim de conhecer as políticas de roteamento dos ASs.

Quanto ao BGP, é possível modificar os parâmetros da sessão estabelecida entre os *peers*, atrasando os anúncios de atualizações de rotas e deixando os roteadores dessincronizados quanto ao estado da rede naquele momento (HUSTON; ROSSI; ARMITAGE, 2011). Além disso, é possível manipular os temporizadores *Routing Flap Damping* (RFD), que suprime os anúncios de rotas intermitentes, e o *Minimum Route Advertisement Interval* (MRAI), que controla o tempo que um roteador deve aguardar entre anúncios de uma mesma rota, desviando o tráfego ou tornando o alvo inacessível. A manipulação indevida desses temporizadores pode tornar uma rota inválida permanentemente (SONG; VENKATARAMANI; GAO, 2016).

Quanto aos ataques de mau uso dos dados, estão relacionados principalmente à violação da política de roteamento de um AS, geralmente por erros na configuração dos roteadores de borda. Nesse tipo de ataque, há vazamento de rotas verdadeiras, recebidas de um AS de nível superior, para outro AS em sentido oposto ao permitido pelos acordos comerciais estabelecidos entre eles. Nesses casos, apesar dos dados de rotas serem

verdadeiros, o anúncio é indevido e pode resultar em redirecionamento de tráfego e DoS em pontos específicos da rede (MAHAJAN; WETHERALL; ANDERSON, 2002). Um exemplo desse ataque é o vazamento de rotas ocorrido em junho de 2019 pela Safe Host SA e China Telecom, apresentado na seção 1.1.

3.2 Mecanismos de Segurança do BGP

Assim como os tipos de ataques ao BGP são variados, existem também diversos mecanismos de segurança disponíveis para sua proteção. Contra os ataques do tipo de falsificação de dados, por exemplo, as técnicas para defesa podem ser proativas ou reativas. Alguns dos mecanismos proativos utilizados para evitar e prever ataques, são:

a) *Resource Public Key Infrastructure* (RPKI). O RPKI pode ser utilizado para verificar a legitimidade da posse de determinado prefixo de rede por um AS. Nessa técnica, os certificados RPKI formam um relacionamento hierárquico que segue a alocação do espaço de endereços. Por exemplo, a *Internet Assigned Numbers Authority* (IANA) aloca um prefixo para a *American Registry for Internet Numbers* (ARIN), a ARIN para os ASs, os ASs para os clientes. Dessa maneira, as autorizações de origem de rota (*Route Origin Authorizations* - ROAs) podem ser assinadas criptograficamente e publicadas em repositórios. Conseguinte, ferramentas de terceiros baixam essas informações, validam e enviam para o roteador BGP. A partir desse momento, o roteador que receber mensagens de atualização do BGP poderá compará-las com os ROAs validados. Esse método é chamado de validação de origem BGP (*BGP Origin Validation*) (WäHLISCH; MAENNEL; SCHMIDT, 2012);

b) validação do caminho com o BGPsec. O BGPsec é uma extensão do BGP que fornece segurança a todo caminho de rede pelo qual passa uma mensagem de `UPDATE`. É implementado utilizando um atributo opcional e não transitivo do BGP, que transporta assinaturas digitais produzidas por cada AS que propaga a mensagem `UPDATE`. As assinaturas digitais garantem que todos os ASs listados no `AS-Path` da mensagem `UPDATE` autorizaram explicitamente o anúncio da rota (LEPINSKI; SRIRAM, 2017).

Assim como o BGPsec, outras soluções também colaboram de alguma forma para validação do caminho no BGP, como o S-BGP (KENT; LYNN; SEO, 2000) e o soBGP (WHITE, 2006). Apesar de possuírem algumas diferenças conceituais e de implementação, juntas essas técnicas são conhecidas como S*BGP (LYCHEV; GOLDBERG; SCHAPIRA, 2013).

Na defesa proativa, as técnicas apresentam resultados efetivos somente quando adotadas globalmente, pois baseiam-se na confiança das informações geradas em cada AS. Porém, desafios técnicos e o elevado custo e complexidade operacionais envolvidos nessa

abordagem ainda inviabilizam sua adoção em grande escala na Internet. A utilização dessas técnicas também não elimina totalmente o risco de comprometimento dos ASs em eventos de ataques propagados na Internet (GOLDBERG et al., 2010).

Quanto à forma reativa de defesa, consiste no monitoramento para detecção de eventos de ataques que estão ocorrendo, ou já ocorreram, utilizando sistemas de detecção de anomalias em mensagens BGP direcionadas aos próprios ASs (HU; MAO, 2007; VEENMAN, 2019). Também podem fazer uso de sistemas mais avançados que suportam o monitoramento de mensagens BGP disponíveis em servidores públicos na Internet (SCHLAMP et al., 2016; SERMPEZIS et al., 2018). Como a técnica reativa identifica ataques reais, atuais ou passados, há possibilidade de comprometimento das operações de um AS mesmo com sua utilização, devido à diferença do tempo entre a identificação do evento e sua efetiva mitigação.

Para proteção contra ataques ao protocolo, há técnicas que vão desde a inclusão de números de sequência para garantir a ordenação das mensagens BGP (SMITH; GARCIA-LUNA-ACEVES, 1996), até a encriptação das camadas inferiores da pilha de protocolos TCP/IP (MURPHY, 2002).

Quanto às contramedidas ao comprometimento dos dados, é importante que cada AS proteja-se dos anúncios suspeitos recebidos de seus *peers* BGP, bem como também cuide para que seus anúncios de rotas não representem riscos à rede. Para isso, é necessário utilizar filtros de anúncios suspeitos nas políticas de roteamento de cada AS, impedindo o recebimento e propagação de rotas válidas por caminhos incorretos, ou prefixos inválidos recebidos de outros ASs ou gerados localmente.

São exemplos de prefixos inválidos, que geram consequências indesejadas às redes quando propagados no BGP, portanto, devem ser filtrados, os endereços de *loopback* e *bogons* (FEAMSTER; JUNG; BALAKRISHNAN, 2005). Nessa linha, Karlin, Forrest e Rexford (2006) propuseram o *Pretty Good* BGP (PGBGP), um mecanismo que identifica rotas suspeitas consultando uma tabela confiável de informações de rotas aprendidas historicamente. No PGBGP, caso o anúncio contenha uma rota não cadastrada previamente, será aplicado um atraso ao anúncio, possibilitando que os administradores da rede tenham tempo de verificar a validade da informação antes do BGP propagá-la em larga escala.

Além disso, podem ser filtradas as rotas para sub-redes pequenas (com prefixo de máscara maiores de 24 bits), evitando o aumento demasiado das tabelas de rotas BGP (BELLOVIN et al., 2001), e os ASs também podem impor um limite máximo para o número de prefixos recebidos nos anúncios de seus *peers*, fechando a sessão BGP caso a quantidade ultrapasse o limite estabelecido (CHANG; GOVINDAN; HEIDEMANN, 2002).

Além dos estudos que abordam formas de ataques e mecanismos de proteção para casos específicos no BGP, como os apresentados até aqui, existem também pesquisas mais abrangentes (*surveys*) que apresentam a evolução do panorama dos riscos e das

soluções indicadas para cada caso. Essas pesquisas servem como um bom repositório de informação que possibilita entender como foi a evolução do cenário de segurança do BGP com o passar do tempo.

Nessa linha de estudos mais generalizados, alguns exemplos são os trabalhos de Butler et al. (2010), que apresenta vulnerabilidades correntes da época (confidencialidade e integridade do TCP e BGP) e explora práticas operacionais, padrões estabelecidos e pesquisas em andamento na segurança do BGP (criptografia, filtragem de anúncios, registro histórico de rotas e gerenciamento seguro do roteador de borda), expondo semelhanças e diferenças entre as técnicas apresentadas; e o trabalho de Al-Musawi, Branch e Armitage (2017a), que inicialmente classificam os ataques em intencionais e não intencionais, apresentando exemplos específicos, seguidos pelos mecanismos de identificação desses ataques (aprendizado de máquina, padrões estatísticos de reconhecimento e dados históricos) .

3.3 Ambientes para Análise do BGP

Com relação aos ambientes para estudo e análise do BGP, Li, Liljenstam e Liu (2009) apresentaram uma *testbed* que integra um roteador baseado em software de código aberto (XORP[1]) sobre uma plataforma de virtualização leve (OpenVZ[2]), que é responsável por conduzir exercícios específicos de ataque e defesa no BGP. Nesse ambiente o tráfego gerado é capturado e desviado para um simulador que calcula os atrasos e perdas correspondentes à travessia da rede, aumentando o realismo do ambiente. Apesar de empregar tecnologias de emulação leve, a *testbed* utiliza roteadores BGP predominantemente acadêmicos, prejudicando a representação realista das tecnologias utilizadas na Internet.

Schlinker et al. (2014) apresentaram a *testbed* PEERING, um ambiente que possibilita testes diretamente conectados na Internet. Para isso, tanto o prefixo quanto o AS utilizados pelo experimentador são definidos e controlados pelos mantenedores da *testbed*. Além disso, é imposto controle da taxa de mensagens de atualização do BGP. Apesar das vantagens em realizar testes diretamente conectados na Internet, a *testbed* restringe e controla algumas ações dos utilizadores, limitando o escopo da análise.

Al-Musawi, Branch e Armitage (2017b) apresentaram o BGP *Replay Tool* (BRT), um módulo de sistema capaz de ler mensagens BGP de uma base local e injetá-las em uma *testbed* composta por roteadores físicos e virtuais, identificando anomalias na rede. As mensagens BGP podem ser geradas sinteticamente ou extraídas dos projetos RouteViews ou RIPEstat. Apesar dessa característica, o funcionamento do ambiente depende diretamente do módulo de programa desenvolvido pelos autores. Além disso, não suporta a representação automática da topologia de ASs da Internet e necessita da estrutura de roteadores BGP configurada à parte.

[1] eXtensible Open Router Platform: http://www.xorp.org
[2] Open Virtuozzo: https://openvz.org

Outro trabalho baseado na utilização de *testbeds* para análise dos efeitos de ataques ao BGP foi o desenvolvido por Ekparinya, Gramoli e Jourjon (2018), que realizaram um estudo empírico sobre o risco de um atacante roubar *coins* Ethereum[3] executando um ataque do tipo *man-in-the-middle*, seguido de um ataque de gasto duplo, onde um único registro de moeda digital é gasto mais de uma vez. Para isso, os autores criaram uma *testbed* baseada na representação das 10 maiores mineradoras Ethereum emuladas numa plataforma OpenStack[4], onde foram realizados ataques de *hijacking* no BGP e *spoofing* do *Address Resolution Protocol* (ARP) para particionar a rede antes do ataque ao Ethereum.

3.4 Limitações e Sumário dos Trabalhos Relacionados

As tabelas 1 e 2 trazem o sumário dos trabalhos apresentados neste capítulo. Embora todos tenham contribuído para a evolução da segurança do BGP de alguma forma, ainda existem lacunas a serem exploradas.

De maneira geral, os estudos apresentados sobre as técnicas de monitoramento e defesa dos riscos do BGP, seções 3.1 e 3.2, tiveram seus resultados obtidos e comprovados através de simulação, emulação em *testbeds* de uso geral, testes em ambientes reais ou utilizando *traces* históricos de mensagens BGP. Apesar de suficientes para validação dos trabalhos, as abordagens apresentam limitação na reprodução de novos cenários para análise de ataques envolvendo representações fidedignas da topologia da Internet, bem como da reprodução dos eventos de ataques reais.

Da mesma maneira, alguns autores apontam claramente para a necessidade de utilizar ambientes próprios de testes, como as *testbeds*, para determinar o esforço necessário na adoção de soluções de segurança para o BGP, auxiliando na parametrização ou hibridização de cenários que combinem soluções diferentes (BUTLER et al., 2010; HUSTON; ROSSI; ARMITAGE, 2011). Além disso, apontam também para a necessidade desses ambientes suportarem aplicações de roteador de borda BGP reais, visto que, as implementações dos roteadores utilizados nesses trabalhos podem não seguir fielmente as especificações da literatura e boas práticas. Um exemplo desse caso foi o apresentado por Song, Venkataramani e Gao (2016), que validaram sua proposta em um laboratório com roteadores BGP Cisco e Quagga, após constatarem que as implementações não seguiam as definições da RFC 2439.

Quanto aos ambientes utilizados para reprodução de topologias BGP, seção 3.3, os trabalhos apresentados não possuem mecanismos que suportam a emulação de topologias reais da Internet. Apesar de Schlinker et al. (2014) apresentarem uma *testbed* que utiliza nós reais, a topologia fica limitada apenas às conexões desses nós.

[3] Ethereum: https://www.ethereum.org/
[4] OpenStack: https://www.openstack.org/

Tabela 1 – Sumário dos trabalhos relacionados aos ataques ao BGP

Trabalho	Tema	Técnica	Validação dos Resultados
(AL-MUSAWI; BRANCH; ARMITAGE, 2017a)	Técnicas de identificação de ameaças e classificação	Análise de séries temporais, aprendizado de máquina, reconhecimento de padrões estatísticos, validação de atualizações de BGP com base no registro de histórico e verificação de alcançabilidade	Análise de dados históricos e literatura
(BALLANI; FRANCIS; ZHANG, 2007)	Tipo de ameaça	Interceptação de tráfego	Teste direto na Internet
(BELLOVIN et al., 2001)	Defesa proativa	Exclusão de redes pequenas (máscaras maiores que 24 bits) dos anúncios BGP	Simulação
(BUTLER et al., 2010)	Técnicas de identificação de ameaças e contramedidas	Confidencialidade e integridade. Criptografia, filtragem de anúncios, registro histórico de rotas, gerenciamento seguro do roteador de borda	Análise de literatura e conceitos técnicos
(CHANG; GOVINDAN; HEIDEMANN, 2002)	Defesa proativa	Limitação do número de rotas aprendidas em uma atualização BGP	Laboratório limitado com equipamentos reais
(CHO et al., 2019)	Classificação de ameaças	Mudança da origem da rota, manipulação de AS-Path, erros em prefixos e *prepends*	Aprendizado de máquina *vs.* classificação humana
(EDDY, 2007)	Tipo de ameaça	Ataque de inundação de SYN	Especificação técnica
(FEAMSTER; JUNG; BALAKRISHNAN, 2005)	Tipo de ameaça	Análise de tráfego para detecção de *bogons*	Análise empírica direto na Internet
(GOLDBERG et al., 2010)	Abrangência de ataques e defesa proativa	Ataques de interceptação, atração e validação do caminho	Baseado em conceitos técnicos e simulação
(HU; MAO, 2007)	Defesa reativa (uso de mensagens locais)	*Hijacking* de prefixos de IP	Correlacionou resultados com dados históricos
(HUSTON; ROSSI; ARMITAGE, 2011)	Avaliação de ameaças e propostas de segurança ao BGP	Classificação de ameaças e métodos de segurança	Análise de literatura e conceitos técnicos
(Karlin; Forrest; Rexford, 2006)	Defesa proativa	Consulta a tabela de informações históricas	Simulação
(KENT; LYNN; SEO, 2000)	Defesa proativa	Validação do caminho	Avaliação em *testbed*
(LEPINSKI; SRIRAM, 2017)	Defesa proativa	Validação do caminho	Especificação técnica
(LYCHEV; GOLDBERG; SCHAPIRA, 2013)	Técnicas de defesa	BGPSec, S-BGP e soBGP	Simulação
(MAHAJAN; WETHERALL; ANDERSON, 2002)	Erros de configuração BGP	Anúncio ou encaminhamento de rotas inválidas	Pesquisa junto aos ASs analisados
(MITSEVA; PANCHENKO; ENGEL, 2018)	Avaliação de ameaças e propostas de segurança ao BGP	Classificação de ameaças e métodos de segurança	Análise de literatura e conceitos técnicos
(PILOSOV; KAPELA, 2008)	Tipo de ameaça	Interceptação de tráfego	Teste direto na Internet
(SCHLAMP et al., 2016)	Defesa reativa (uso de mensagens locais)	*Hijacking* de prefixos de IP	Correlacionou resultados com dados históricos
(SERMPEZIS et al., 2018)	Defesa reativa (uso de mensagens locais e de servidores públicos)	*Hijacking* de prefixos de IP	Teste direto na Internet
(SHI et al., 2012)	Detector de *hijacking*	Correlação das informações do plano de controle e dados	Teste direto na Internet
(SMITH; GARCIA-LUNA-ACEVES, 1996)	Integridade dos dados no BGP	Inclusão de números de sequência	Análise de literatura e conceitos técnicos
(SONG; VENKATARAMANI; GAO, 2016)	Tipo de ameaça	Manipulação dos temporizadores RFD e MRAI do BGP	Laboratório limitado com equipamentos reais e simulação
(SRIRAM; MONTGOMERY, 2018)	Tipos de ameaças e mitigação	DoS, *spoofing*, amplificação. Monitoramento, filtragem, RPKI, validação de caminho	Baseado em conceitos técnicos
(VEENMAN, 2019)	Defesa reativa (uso de mensagens locais)	*Hijacking* de prefixos de IP	Correlacionou resultados com bases públicas de rotas BGP
(VERVIER; THONNARD, 2013)	Tipo de ameaça	Spam	Correlacionou resultados com dados históricos
(WATSON, 2003)	Tipo de ameaça	Ataque de TCP *Reset*	Teste limitado em laboratório
(WäHLISCH; MAENNEL; SCHMIDT, 2012)	Defesa proativa	Autenticação da origem com RPKI	Correlacionou resultados com bases públicas de rotas BGP
(WHITE, 2006)	Defesa proativa	Validação do caminho	Especificação técnica

Tabela 2 – Sumário dos trabalhos relacionados aos ambientes para análise do BGP

Trabalho	Platforma	Topologia	Tecnologia do Roteador BGP
(AL-MUSAWI; BRANCH; ARMITAGE, 2017b)	Física	Sintética / Estática	Necessita estrutura à parte
(EKPARINYA; GRAMOLI; JOURJON, 2018)	Virtual	Parte da real / Configurável	Compatível com mercado e de acesso livre
(LI; LILJENSTAM; LIU, 2009)	Virtual	Sintética / Configurável	De uso acadêmico
(SCHLINKER et al., 2014)	Física	Parte da real / Limitada aos nós do *cluters*	Limitada aos nós do *cluters*

Outro ponto importante é a escalabilidade dos ambientes, que ficam limitados apenas ao poder computacional da *testbed* física de cada proposta. Butler et al. (2010) já apontavam a dificuldade dos simuladores e *testbeds* escalarem suficientemente para suportar a representação da topologia global da Internet, o que dificultava o entendimento detalhado do impacto e o comportamento das técnicas de segurança propostas. Além disso, não é fácil adaptar ou trocar o software do roteador BGP quando necessário pelo utilizador. Por fim, as *testbeds* não suportam a automação de todo *setup* do ambiente, com criação de topologias dinâmicas, definição dos parâmetros de rede, configuração dos roteadores BGP, efetivação dos procedimentos de ataque e coleta dos resultados.

Capítulo 4

Arquitetura da *testbed*

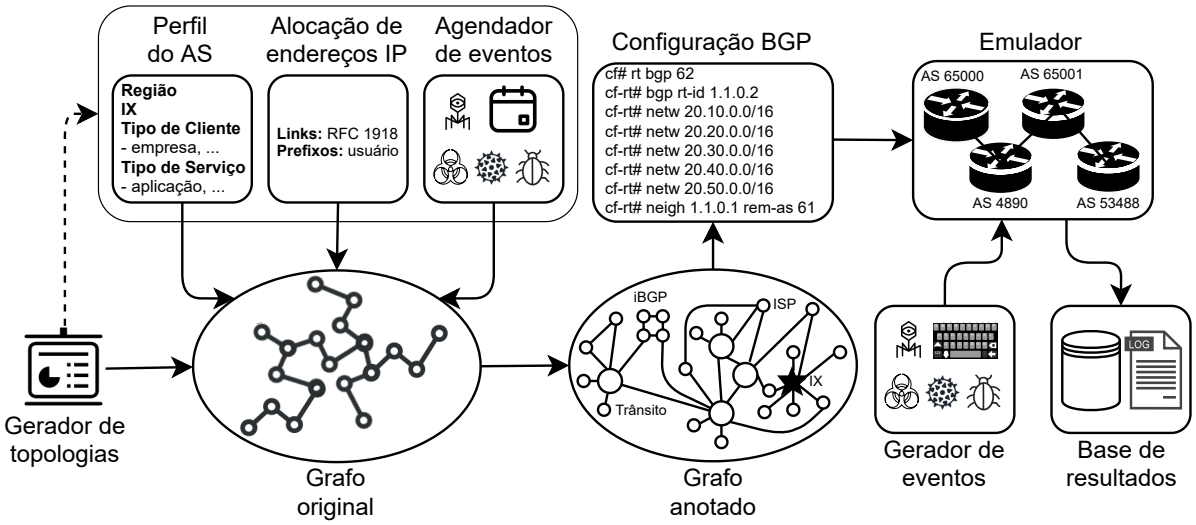
Motivados pela primeira questão de pesquisa apresentada na seção 1.4, que trata dos requisitos de um ambiente controlado para estudo dos ataques ao BGP e mitigação dos riscos para a estrutura da Internet, optou-se pela elaboração de uma arquitetura de *testbed* modularizada capaz de suportar todo ciclo de vida de um experimento BGP, conforme apresentado na Figura 2. Esta arquitetura foi planejada para guiar o utilizador desde a fase de definição da topologia, passando pela configuração dos detalhes técnicos de endereçamento IP, informação das características dos links e serviços oferecidos pelos ASs, identificação dos eventos de ataques ao BGP, configuração automática do ambiente de testes, além de disponibilizar mecanismos de interação em tempo de execução e extração dos resultados finais para análise posterior pelo utilizador.

A Figura 2 apresenta os módulos formadores da arquitetura e a relação entre eles. Abaixo são descritas suas funcionalidades:

a) **Gerador de topologias:** módulo responsável pela definição do grafo que corresponde à topologia que será emulada. Suporta a criação de topologias específicas definidas pelo utilizador, baseadas em modelos de grafos já estabelecidos ou importadas já prontas a partir de fontes externas, como apresentado na seção 2.2.

   A topologia resultante nessa etapa tem sua estrutura representada na Figura 3. Nela é possível observar que o grafo de topologia é composto por um arranjo de ASs (`autonomous_ systems`[, `autonomous_system 1`, ..., `autonomous_system n`]) e outro arranjo de links (`links`[, `link 1`, ..., `link n`]). Cada AS contém atributos que podem ser recorrentes ou não, dependendo dos valores do objeto. Quanto aos links, são compostos por atributos que identificam os ASs vizinhos, bem como traz detalhes técnicos do próprio link de dados. Na Figura 3 também é possível

Figura 2 – Elementos formadores da arquitetura da *testbed*



Fonte: Produzido pelos autores

observar que os objetos de AS e de link podem receber novos elementos sempre quando identificados novos ASs ou relações de vizinhança entre os roteadores BGP.

Os atributos de um `AS` podem ser divididos entre básicos e complementares. Como atributos básicos podemos citar:

- `router_id`: que especifica o valor do parâmetro identificador do roteador BGP que será utilizado na configuração do ambiente; e
- `prefixes`: que corresponde ao arranjo dos prefixos IP e máscaras de rede que serão divulgados pelo AS no BGP.

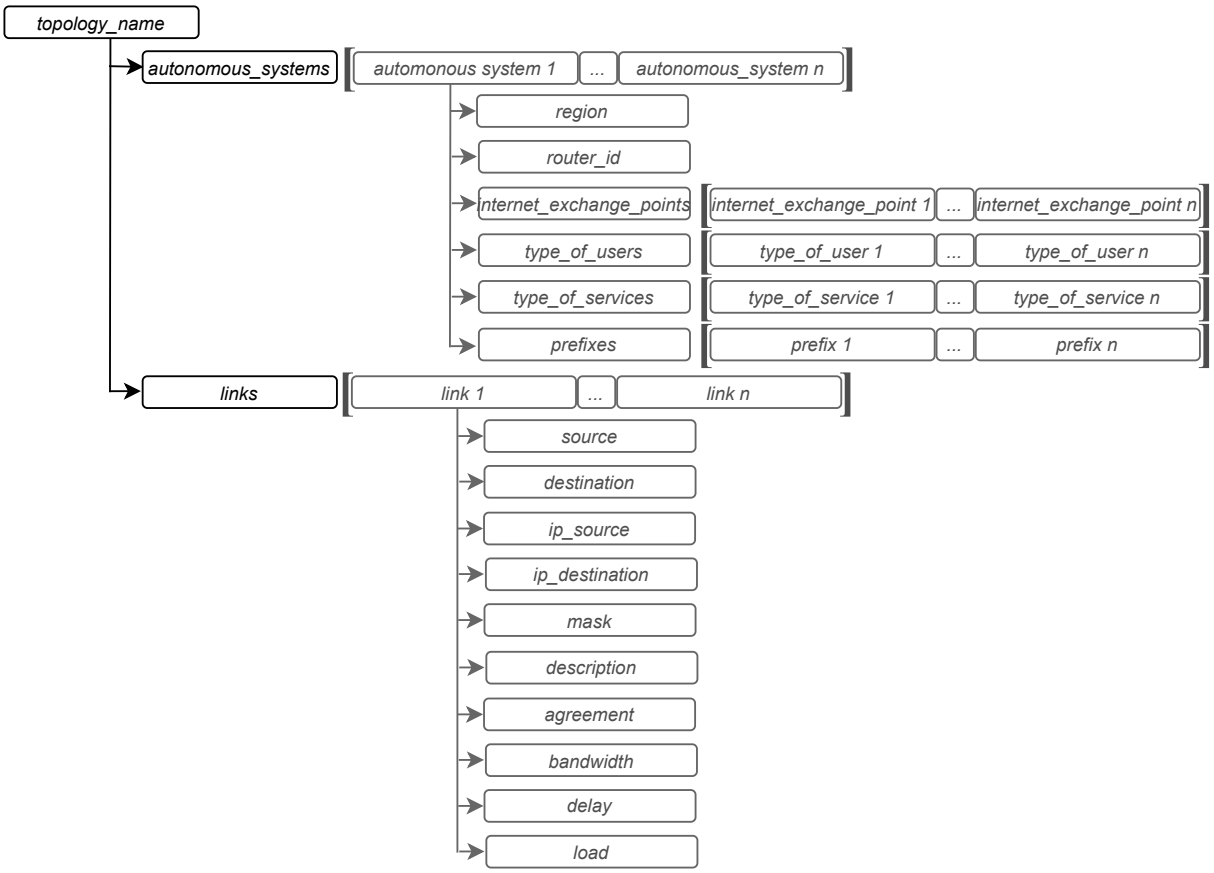
Por sua vez, os atributos complementares são:

- `region`: que indica a região geográfica onde o AS está localizado;
- `internet_exchange_points`: que corresponde ao arranjo dos IXPs com os quais o AS faz vizinhança;
- `type_of_users`: que especifica os tipos e quantidade de usuários que utilizam os serviços daquele AS (exemplo: `empresa`, `domicílio` e `governo`);
- `type_of_services`: que apresenta a relação de tipos de serviços oferecidos pelo AS (exemplo: `Aplicação`, `Hospedagem`, `Acesso`, `Conteúdo` e `Trânsito`).

De modo geral, os atributos básicos são responsáveis pelo funcionamento do BGP, enquanto os atributos complementares podem ser utilizados na identificação da criticidade dos ataques BGP a determinados ASs.

Assim como nos ASs, os links também possuem atributos básicos, que são responsáveis por identificar os ASs envolvidos na relação de vizinhança do BGP, e

Figura 3 – Representação gráfica da estrutura de dados utilizada para representar os ASs e a relação de vizinhança BGP na *testbed*



Fonte: Produzido pelos autores

os complementares, que contêm dados técnicos que especializam cada link. Nessa linha, os parâmetros básicos são:

- `source` e `destination`: que identificam os ASN vizinhos;
- `ip_source`, `ip_destination` e `mask`: que contêm os endereços IP e máscara das interfaces dos roteadores que formam os pontos extremos do link.

Quanto aos atributos com dados técnicos e complementares de cada link, temos:

- `description`: que corresponde ao nome identificador do link;
- `agreement`: que especifica o tipo de relação comercial existente entre os ASs vizinhos (seção 2.2);
- `bandwidth`: define a banda do link e está representada em kilobits por segundo;
- `delay`: corresponde ao atraso em milissegundos do link;
- `load`: que representa a porcentagem de utilização do link para *download*.

Um ponto importante é que os valores dos parâmetros da topologia gerada neste módulo podem ser modificados posteriormente pelo utilizador, ao interagir com as interfaces de acesso disponibilizadas nos módulos **Perfil do AS**, **Alocação de endereços IP** e **Agendador de eventos**.

b) **Perfil do AS:** este módulo é responsável por possibilitar a definição de novos valores ou alterar os atributos básicos e complementares já existentes dos ASs e links no grafo da topologia.

Vale ressaltar que, mesmo não interferindo diretamente no comportamento e funcionamento do BGP na rede, a inclusão de algumas informações, como `region`, `internet_exchange_points`, `type_of_users` e `type_of_services` são relevantes, pois, auxiliam os utilizadores da *testbed* a mensurar os impactos no caso de ataques em regiões específicas da topologia. Por exemplo, se uma determinada região é composta predominantemente por clientes do tipo governamental e sofre um ataque de *hijack* de um prefixo responsável por algum serviço importante ao governo, o dano pode assumir grandes proporções. Da mesma forma, se uma determinada região possui uma grande quantidade de clientes empresariais e sofre ataques DDoS em um prefixo importante para serviços corporativos, isso pode representar grande prejuízo financeiro.

c) **Alocação de endereços IP:** este módulo é responsável por alocar endereços IP nas interfaces dos links de dados e definir os prefixos de sub-rede divulgados no BGP. Para isso, o processo descrito abaixo deve ocorrer:

- **links de dados:** quando não informados explicitamente na topologia resultante do módulo **Gerador de topologias**, recebem endereços IPv4 privados (RFC 1918) atribuídos automaticamente, com 30 bits na máscara de sub-rede. Os endereços são definidos sequencialmente, em ordem crescente, através do parâmetro identificador dos links e não há controle de alocação por área geográfica da topologia. Os endereços IP dos **links de dados** não são divulgados no BGP.
- **prefixos divulgados:** os prefixos divulgados no BGP podem ser definidos na topologia resultante do módulo **Gerador de topologias** ou serem atribuídos pelo utilizador manualmente pela interface deste módulo. Não há atribuição automática de prefixos para serem divulgados no BGP.

d) **Agendador de eventos:** este módulo é responsável por possibilitar o agendamento de eventos de ataque ao BGP no grafo da topologia definida no módulo **Gerador de topologias**, com todas as especificidades definidas nos módulos **Perfil do AS** e **Alocação de endereços IP**. Essa ação estabelece um novo conceito

na arquitetura, denominado **cenário de ataque ao BGP**. Esse conceito compreende a definição de uma linha de tempo, com data, hora, minuto e segundo de início e fim para validade do cenário, onde são executados eventos de ataques ao protocolo BGP na topologia em tempos pré-estabelecidos. Para serem considerados válidos, esses eventos devem ser programados para ocorrerem em um período válido entre a data de início e fim de execução do cenário. São considerados eventos de ataque as seguintes ações:

- **anúncio de prefixos sequestrados (*announcement*):** corresponde ao agendamento do anúncio de prefixos por ASs durante a execução do cenário. Este anúncio deve conter a definição de um tempo válido (dia, hora, minuto e segundo) para sua execução, além de conter o ASN que divulgará o prefixo e o próprio prefixo e máscara que serão divulgados;
- **retirada ou bloqueio de anúncios (*withdraw*):** correspondem à exclusão do anúncio de um determinado prefixo no BGP. A retirada ou exclusão podem ser realizadas diretamente no AS emissor, cessando a divulgação do referido prefixo, ou em algum ponto da rede, bloqueando a propagação a partir de um AS específico. Tecnicamente, a retirada ou bloqueio de um anúncio devem ser observadas sob o ponto de vista do AS onde a ação é executada, podendo ser realizadas sobre os anúncios BGP que o AS recebe ou propaga na rede. Essa ação também precisa ser agendada com uma data válida no cenário;
- **inclusão de AS como prefixo no `AS_PATH` (*prepend*):** corresponde à inclusão ou exclusão de ASs no `AS_PATH` de um determinado anúncio BGP. Também deve ser observado a partir do ponto de vista de um AS específico na topologia, onde o `AS_PATH` pode ser modificado incluindo novos ASs, repetidos ou não, na chegada de um anúncio BGP ou nos anúncios publicados por ele. O agendamento do *prepend* deve conter a informação do AS alvo, o AS que será prefixado ou sufixado, a quantidade de vezes que isso ocorrerá e a data da ação válida no cenário.

e) **Grafo anotado:** esse módulo corresponde à fusão do grafo da topologia contendo os perfis dos ASs, a estrutura de endereçamento IP, prefixos divulgados no BGP e os eventos de ataque agendados. Mantém uma representação do grafo da topologia finalizado para consulta e visualização pelo utilizador. Nessa fase, toda topologia e eventos BGP estão finalizados e prontos para serem codificados na tecnologia de emulação.

f) **Configuração BGP:** converte as informações que compõem o **Grafo Anotado** em codificação válida no **Emulador**. O resultado é o conjunto dos comandos necessários para instanciação de toda topologia no ambiente de emulação, seguida da

execução dos eventos BGP e o armazenamento dos resultados do processamento do cenário.

g) **Emulador:** tecnologia de emulação leve que suporta toda *testbed*. Nele são criadas as instâncias de cada roteador; configurados os IPs de conectividade e prefixos de rede divulgados no BGP; configurados os parâmetros de banda e atraso dos links de dados entre os roteadores; executados os eventos de ataques agendados; realizadas as interações a partir de alterações dos parâmetros do ambiente em tempo de execução do cenário; e capturado o estado do ambiente a cada mudança ocorrida, para posterior análise pelo utilizador.

h) **Gerador de eventos:** esse módulo pode ser considerado uma extensão do **Agendador de eventos**, porém com uma interface que possibilita a interação do usuário com os parâmetros do cenário em tempo de execução, possibilitando modificar o valor de atributos do ambiente e coletar resultados durante a execução do cenário. Importante ressaltar que as ações executadas nesse módulo não paralisam obrigatoriamente a execução dos eventos agendados previamente, podendo ser executadas em paralelo.

i) **Base de resultados:** esse módulo é responsável por coletar, armazenar e disponibilizar os resultados do processamento dos eventos BGP previamente agendados ou inseridos em tempo de execução em um cenário. Ele permite que o utilizador tenha acesso aos logs de todo ambiente para análise detalhada da execução do cenário. Fazem parte desse módulo os logs:

   - **da tecnologia de emulação:** possibilita o utilizador analisar os detalhes da emulação do cenário, como tempos de instanciação do ambiente, consumo dos recursos computacionais das máquinas hospedeiras da *testbed*, erros de execução ou qualquer outro evento relacionado ao funcionamento da tecnologia de emulação da topologia de rede;
   - **do roteador BGP:** traz detalhes sobre o funcionamento do BGP no ambiente, com os tempos de adjacência do protocolo, relação de vizinhança entre os ASs, prefixos divulgados, anúncios realizados e aceitos pelos ASs, entre outros;
   - **dos eventos de ataques ao BGP:** apresenta os resultados dos eventos de ataques ao BGP (*announcement*, *withdraw* e *prepend*), pré-agendados ou definidos em tempo de execução.

Capítulo 5

Tecnologias Habilitadoras e Detalhes de Implementação

A partir da definição da arquitetura apresentada no Capítulo 4, passou-se à fase de identificação das tecnologias e métodos necessários para garantir o funcionamento das funções de cada módulo na *testbed*, bem como, quando necessário, na criação dos módulos de sistema responsáveis por atender essas funções específicas. Nessa linha, as próximas seções trazem o detalhamento dos recursos e procedimentos envolvidos na criação e reprodução das topologias de rede na *testbed* (seção 5.1); na geração de eventos de ataques ao BGP (seção 5.2); e na definição da solução de emulação leve e distribuída utilizada como base da *testbed* (seção 5.3).

5.1 Criação e Reprodução da Topologia

Como abordado na segunda questão de pesquisa (seção 1.4), a reprodução da topologia realista da Internet é um requisito para atingir o objetivo deste trabalho. Para isso, foi necessário identificar a solução que melhor atende esse requisito dentre os métodos e tecnologias geradoras de topologias apresentadas na seção 2.2. Além disso, nessa fase do trabalho buscou-se estender as funções da *testbed* a fim de possibilitar que o utilizador modifique as topologias reproduzidas ou proponha novas topologias sintéticas.

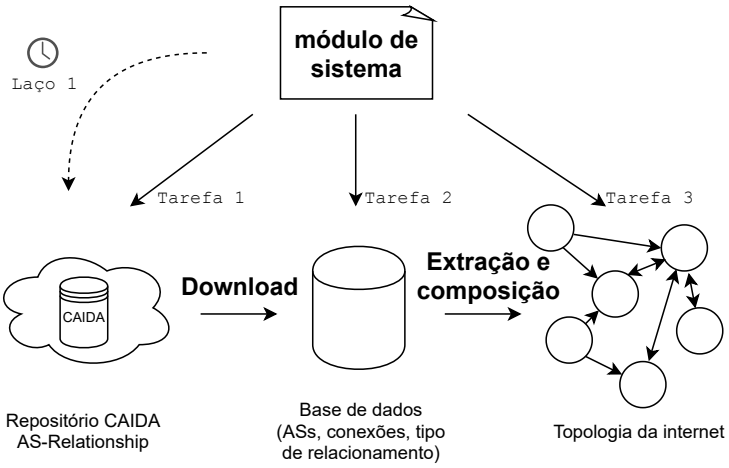
5.1.1 Topologia Realista - Projeto AS-Relationship (CAIDA)

Das soluções analisadas anteriormente, as bases de dados de relacionamento entre ASs disponibilizadas pelo CAIDA no repositório do projeto AS-Relationship (seção 2.2.2), são

fontes de informação confiável para serem utilizadas na reprodução da topologia realista da Internet, pois contêm uma relação relevante dos ASs existentes na Internet, suas conexões com outros ASs e os tipos de relacionamentos comerciais que definem a direção do tráfego BGP entre eles. Essas informações possibilitam reproduzir a topologia da Internet com bom grau de realismo, além de propiciar que a direção do tráfego seja respeitada na análise da propagação de um ataque ao BGP.

Nessa linha, para garantir que a *testbed* suporte esses dados, o módulo **Gerador de topologias** da arquitetura passou a contar com um módulo de sistema desenvolvido em Python 3, que é responsável por ler cada registro da base de dados, capturando e armazenando as informações que identificam os ASs e seus relacionamentos, como apresentado na Figura 4. Essas informações são utilizadas para montar o grafo da topologia da Internet, cujos vértices representam os ASs e as arestas direcionadas denotam os enlaces de conexão entre eles.

Figura 4 – Processo de criação da topologia realista da internet baseada nos dados do projeto AS-Relationship (CAIDA)



Fonte: Produzido pelos autores

Quanto ao tipo de relacionamento comercial entre os ASs (`c2p` e `p2p`), é utilizado para identificar a direção das atualizações de rotas, ordem de precedência dos ASs afetados e a velocidade de propagação dos ataques no BGP. Para isso, as arestas do grafo da topologia são direcionadas de acordo com o fluxo de dados informado no relacionamento.

Esse módulo de sistema também tem a função de pesquisar por novas versões da base de dados no repositório do Projeto AS-Relationship. A pesquisa por atualizações pode ser manual, com intervenção do utilizador da *testbed*, ou automaticamente, com o agendamento prévio da pesquisa por novos dados. Essa função possibilita que as atualizações nas informações de ASs, conexões e mudanças no tipo de relacionamento comercial entre eles sejam consideradas em análises posteriores.

O grafo resultante desse processo, considerando também o tipo de relacionamento entre os ASs, possibilita identificar as regiões afetadas por um ataque na Internet, considerando a sequência de propagação e quantidade de ASs comprometidos em um evento de ataque.

A Figura 4 também apresenta a sequência de eventos executadas pelo módulo de sistema para atender ao processo descrito anteriormente. A `Tarefa 1` corresponde às ações de pesquisa e download da base de dados atualizada e disponível no repositório do projeto AS-Relationship do CAIDA; a `Tarefa 2` é o momento em que a base de dados é lida e as informações de ASs, conexões e tipo de relacionamento são capturadas e armazenadas na base de dados da *testbed*; enquanto a `Tarefa 3` corresponde à reprodução do grafo direcionado da topologia da Internet para a execução de um cenário de ataque ao BGP na *testbed*. O `Laço 1` representa o processo de verificação por atualizações periódica da base de dados, que pode ser agendado ou executado manualmente pelo utilizador.

5.1.2 Topologia Realista - Projeto RIPEstat (RIPE)

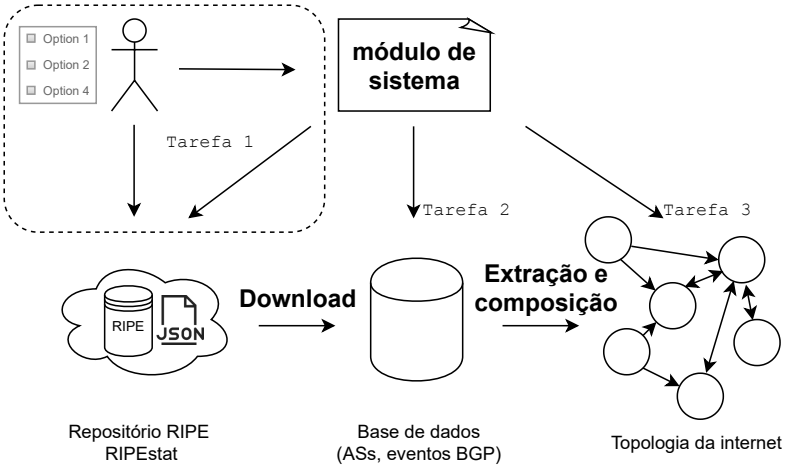
Outra base de dados utilizada como fonte de informação para reprodução da topologia da Internet é a disponibilizada no Projeto RIPEstat da RIPE (seção 2.2.2).

Para isso, o **Gerador de topologias** também passou a contar com um módulo de sistema desenvolvido em Python 3, que possibilita ao utilizador informar os parâmetros de consulta à base, definindo o período inicial e final dos dados desejados, os prefixos e ASs relacionados. A partir desse ponto, o sistema pesquisa, captura e armazena automaticamente os dados resultantes da consulta realizada ao repositório RIPEstat, conforme apresentado na `Tarefa 1` da Figura 5. Além disso, o sistema também suporta a entrada dos dados a partir de arquivos JSON resultantes das pesquisas manuais realizadas pelo próprio utilizador da *testbed*.

A partir desse ponto, na `Tarefa 2`, o módulo de sistema lê a base de dados recebida, extrai a lista de ASs que fazem parte da topologia da rede e identifica a relação de vizinhança entre eles no tempo inicial monitorado. Esse procedimento é realizado analisando os registros contidos no parâmetro `INITIAL_STATE` do arquivo JSON. Soma-se a isso a relação dos eventos BGP ocorridos posteriormente, que são encontrados no parâmetro `EVENTS` (seção 2.2.2), que permitem identificar as novas adjacências ocorridas no BGP durante a execução do cenário. Feito isso, a `Tarefa 3` corresponde à criação do grafo da topologia da Internet com as informações armazenadas na base de dados da *testbed*.

Além de possibilitar a reprodução da topologia realista da Internet, a base RIPEstat também viabiliza a identificação dos eventos ocorridos no BGP, sejam pelas atualizações de rotas consideradas normais ao ambiente, como também os eventos de ataque ao protocolo no período analisado. Essa função é detalhada na seção 5.2.

Figura 5 – Processo de criação da topologia realista da internet baseada nos dados do projeto RIPEstat (RIPE)



Fonte: Produzido pelos autores

5.1.3 Topologia Sintética

Além de garantir a reprodução realista da topologia da Internet, o escopo do projeto foi estendido ao incluir um módulo de sistema no **Gerador de topologias**, também desenvolvido em Python 3, que suporta a inserção de topologias criadas manualmente pelo utilizador, ou importá-las a partir das tecnologias geradoras apresentadas na seção 2.2.1. Tal função se mostra relevante ao viabilizar a utilização da *testbed* também na fase de projeto de novas redes, possibilitando a homologação de novas topologias e implementações BGP.

Para isso, foi definido um modelo de dados para representar toda topologia de rede baseado em objetos JSON, contendo a relação dos ASs e todos parâmetros de conexão necessários entre eles. A Figura 6 traz um exemplo da representação do AS 65000, que possui um enlace de conexão com o AS 65001, com os parâmetros do enlace (banda, atraso, carga e tipo de relacionamento comercial existente entre os ASs) devidamente preenchidos. Nesse exemplo também é possível observar os dados que identificam a localização geográfica do AS, os serviços oferecidos, a quantidade e tipo de clientes, bem como os endereços IP do enlace e prefixos divulgados no BGP.

Os parâmetros descritos nos objetos JSON da Figura 6 seguem a mesma estrutura da representação gráfica dos ASs e relação de vizinhança BGP na *testbed*, apresentadas na Figura 3.

Figura 6 – Representação em objeto JSON da estrutura de dados utilizada para representar os ASs e a relação de vizinhança BGP na *testbed*


```
{
    "topology_name": "Topology",
    "autonomous_systems": [{
        "autonomous_system": "65000",
        "region": "North",
        "router_id": "200.1.1.1",
        "internet_exchange_points": [{
            "internet_exchange_point": "IX-A",
            "region": "North" }
        ],
        "type_of_users": [{
            "type_of_user": "government",
            "number": 100 }
        ],
        "type_of_services": ["access"],
        "prefixes": [{
            "prefix": "200.2.0.0",
            "mask": 20 }
        ]},
     ...],
    "links": [{
        "source": "65000",
        "destination": "65001",
        "ip_source": "200.1.0.1",
        "ip_destination": "200.1.0.2",
        "mask": "30",
        "description": null,
        "agreement": "p2p",
        "bandwidth": 1000000,
        "delay": 10,
        "load": 30 },
    ...]}
    ]
}
```


Fonte: Produzido pelos autores

5.1.4 Topologia Resumida

Para toda topologia disponível na *testbed*, independente do método de criação, o utilizador deve ser capaz de escolher se deseja reproduzi-la de forma completa ou resumida durante a execução dos cenários de ataque ao BGP. Nas topologias resumidas, os ASs *stub*, que são os ASs que possuem apenas um enlace de dados para conexão com toda Internet, são retirados da topologia final juntamente com seus enlaces, permanecendo apenas os ASs que possuem duas ou mais relações de vizinhança no BGP ou que anunciam algum prefixo na rede.

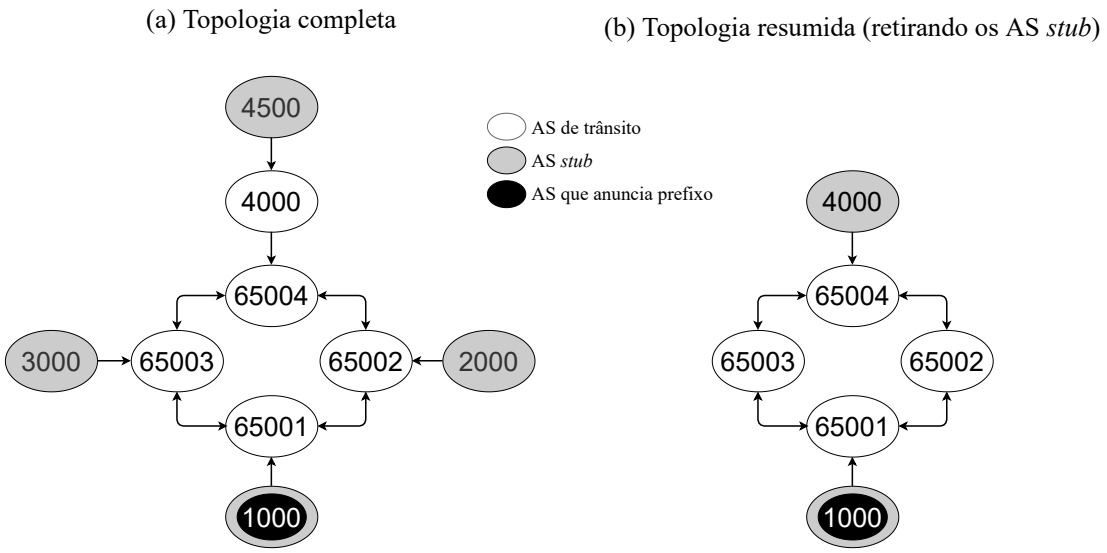
A manutenção dos ASs com dois ou mais vizinhos justifica-se pela possibilidade dele ser utilizado como trânsito na rede ou utilizar `AS_PATH` diferentes para alcançar os destinos desejados, enquanto os ASs que divulgam algum prefixo são os responsáveis diretos por modificar as tabelas de rotas do BGP na topologia.

Essa ação visa diminuir os requisitos de recursos computacionais na reprodução da topologia completa na *testbed*, sem comprometer o realismo da reprodução. Para isso, o **Gerador de topologias** conta com um módulo de sistema, desenvolvido em Python

3, que permite o utilizador informar se deseja retirar os ASs *stub* da topologia antes de iniciar a reprodução dos cenários de ataque.

A Figura 7 apresenta um exemplo de topologia onde foram retirados os ASs *stub*. Nela é possível observar que a topologia original continha 9 ASs (Figura 7a) e, ao executar o procedimento de retirada dos ASs *stub*, passou a contar com apenas 6 ASs (Figura 7b). Nesse caso, a topologia final teve uma diminuição aproximada de 33% no número de ASs emulados, resultando uma redução significativa dos requisitos computacionais para reprodução do ambiente. Vale ressaltar que esse procedimento apenas deixa de emular os ASs *stub*, porém não modifica a base de dados da topologia armazenada na *testbed*.

Figura 7 – Comparação entre a topologia completa (a) e a topologia sem os ASs *stub* (b)



Fonte: Produzido pelos autores

Comparando as Figuras 7a e 7b, podemos observar que, apesar do AS `1000` ser *stub*, ele não foi retirado da topologia resumida porque divulga um prefixo IP no BGP. Outro ponto relevante que merece destaque, é que o AS `4000` permanece como *stub* na topologia resumida. Isso ocorre porque o módulo de sistema verifica por ASs *stub* apenas uma vez, nesse caso, retirando o AS `4500` da topologia resumida e deixando o AS `4000` apenas com um vizinho na nova representação.

Para se ter uma noção mais clara dos benefícios de resumir a topologia, no caso específico da topologia criada a partir da base de dados de setembro de 2021 do Projeto AS-Relationship (CAIDA) (subseção 2.2.2), o processo resultou na diminuição de aproximadamente 36% no número de ASs e 5% no número de enlaces na topologia final. Inicialmente a base possuía 72.813 ASs e 501.913 enlaces, sendo que, desse total, 26.450 são ASs *stub*.

5.1.5 Endereçamento IP e Especialização do AS

Seja qual for o tipo de topologia criada, assim que o módulo **Gerador de topologias** encerra suas atividades, ele entrega o processamento ao sistema que desempenha as funções do módulo de **Alocação de endereços IP**, para identificação dos prefixos que serão divulgados no BGP e atribuição dos endereços IP nos links de comunicação entre os ASs que possuem relação de vizinhança no BGP. Em paralelo, o sistema do módulo **Perfil do AS** especializa o AS com as informações do tipo de serviço, tipo de cliente e regionalidade.

Alguns métodos de criação da topologia suportam o preenchimento automático desses dados, como nas topologias sintéticas, apresentadas na seção 5.1.3. Nesse caso, o módulo de sistema lê os dados existentes no JSON e popula a base de dados da *testbed* com os dados disponíveis. Por outro lado, os sistemas dos módulos **Perfil do AS** e **Alocação de endereços IP** também permitem que o utilizador inclua valores posteriormente à criação da topologia quando eles não existirem, bem como modifiquem os valores já existentes.

Os sistemas dos módulos **Perfil do AS** e **Alocação de endereços IP** foram desenvolvidos em Python3.

5.2 Reprodução dos Eventos de Ataques ao BGP

Como apontado na terceira questão de pesquisa, apresentada na seção 1.4, a reprodução fidedigna dos eventos observados nos casos de ataques reais ao BGP é considerada um requisito para a análise completa das técnicas utilizadas pelos atacantes, além de possibilitar identificar as contramedidas possíveis por parte do utilizador da *testbed*.

Visando atender esse requisito, a arquitetura da *testbed* conta com os módulos **Agendador de eventos** e **Gerador de eventos** (Figura 2). Apesar de seguirem abordagens diferentes quanto ao método utilizado para definição dos eventos de ataque ao BGP, possibilitam informar os eventos ocorridos durante a execução dos cenários analisados. Nessa linha, as seções 5.2.1 e 5.2.2 detalham a implementação dos módulos de sistema responsáveis por desempenhar essas funções.

5.2.1 Reprodução dos eventos reais ocorridos no BGP

A *testbed* faz uso dos registros existentes nas bases de dados do Projeto RIPEstat (seção 2.2.2) para identificar os eventos reais ocorridos no BGP na topologia da Internet. Para isso, o módulo **Agendador de eventos** atua como uma extensão do **Gerador de topologias**, sendo acionado automaticamente quando uma topologia é criada na *testbed*, a partir dos dados extraídos dos arquivos JSON resultantes das consultas à base do projeto.

Inicialmente, o sistema que desempenha as funções do módulo **Agendador de eventos**, que foi desenvolvido em Python 3, lê os registros contidos no objeto `INITIAL_STATE`

do arquivo JSON, identificando o estado inicial do ambiente ao apresentar os eventos que devem ser executados no BGP logo no início da reprodução da topologia na *testbed*. Os parâmetros lidos são:

a) `target_prefix`: apresenta o prefixo IP e a máscara relacionados ao evento;

b) `source_id`: especifica o AS coletor que observou o evento;

c) `path`: arranjo contendo o `AS_PATH` para o prefixo de que trata o evento.

A partir desse ponto, o módulo **Agendador de eventos** também lê os registros do objeto `EVENTS` do arquivo JSON, identificando os valores dos parâmetros citados anteriormente, somados aos parâmetros apresentados a seguir:

a) `timestamp`: contém a informação do momento da ocorrência do evento na linha de tempo do cenário;

b) `type`: apresenta o tipo de evento, que pode ser o anúncio de um novo prefixo IP, a mudança do `AS_PATH` ou um *withdraw*.

Dessa forma, cada evento observado no objeto `EVENTS` pode representar o anúncio de um novo prefixo IP (*announcement*); alteração do `AS_PATH` para um prefixo IP específico, contendo um caminho simples ou com *prepend*; ou a retirada de uma rota do BGP (*withdraw*).

Importante ressaltar que um AS pode conter múltiplos roteadores BGP, portanto, eventos com `source_id` diferentes podem pertencer ao mesmo AS. Como esses roteadores normalmente compartilham as mesmas políticas de roteamento interna do AS, pelo iBGP, o módulo de sistema do **Agendador de eventos** reproduz apenas um roteador por AS mesmo nessa situação.

5.2.1.1 Anúncios de novos prefixos (*announcements*)

O anúncio de um novo prefixo no BGP (`target_prefix`) pode ocorrer tanto nos eventos observados no objeto `INITIAL_STATE` quanto no `EVENTS` do arquivo JSON. Basicamente, a diferença entre eles é que os eventos do `INITIAL_STATE` são executados no BGP assim que o cenário é criado no **Emulador**, antes da execução de qualquer outro evento agendado. Já os anúncios identificados no `EVENTS`, são configurados no AS correspondente da topologia apenas no tempo exato constante do parâmetro `timestamp` do próprio evento. Nas duas situações, o anúncio de um novo prefixo IP só é configurado no AS anunciante uma única vez, na sua primeira ocorrência.

A Figura 8 traz o exemplo do anúncio do prefixo IP 208.65.153.0/24, encontrado no objeto `EVENTS` do arquivo JSON.

Primeiramente, pode-se afirmar que o evento representado na Figura 8 trata de um anúncio, devido o valor A contido no parâmetro `type`. Além disso, é possível verificar que o coletor RIPE que observou o anúncio tem código identificador 00-12.0.1.63 (`source_id`). Esse código pode ser traduzido para seu ASN consultando o objeto `sources`, existente no

Figura 8 – Exemplo de evento observado no objeto EVENTS do arquivo JSON resultante da pesquisa nos repositórios do RIPEstat


```
"events": [
    {
        "seq": 898,
        "timestamp": "2008-02-24T18:47:53",
        "type": "A",
        "attrs": {
            "source_id": "00-12.0.1.63",
            "target_prefix": "208.65.153.0/24",
            "path": [2497,3491,3491,17557],
            "community": [
                "2497:5000"
            ]
        }
    }
]
```


Fonte: Produzido pelos autores

mesmo arquivo, ou observando o ASN mais à esquerda no parâmetro `path` (ASN 2497). Também é possível identificar o `timestamp`, que representa o momento exato em que o prefixo IP deve ser anunciado pelo AS correspondente. Já o AS anunciante do prefixo é identificado analisando o arranjo contido no parâmetro `path`, onde o AS mais à direita representa o anunciante do prefixo IP (AS 17557).

Apesar do exemplo apresentado na Figura 8 corresponder ao objeto `EVENTS`, sua estrutura é praticamente igual ao objeto `INITIAL_STATE`, mudando apenas alguns parâmetros como apresentado anteriormente.

5.2.1.2 Alteração do AS_PATH (simples ou com *prepend*)

As mudanças de caminho para rotas BGP podem ser verificadas nas alterações do `AS_PATH` observadas para um mesmo prefixo IP no decorrer do tempo no arquivo JSON. Se por um lado o anúncio de um novo prefixo deve ser considerado apenas na ocorrência do primeiro evento, toda mudança no `AS_PATH` deve ser examinada, pois representa o dinamismo do protocolo BGP em escolher novos caminhos na rede. Esse tipo de evento também pode representar a inclusão de *prepend*, conforme observado na Figura 8, onde o parâmetro `path` contém a repetição do AS 3491.

Infelizmente, os registros contidos no arquivo JSON não expõem os motivos da mudança de um `AS_PATH`, impossibilitando inferir de forma macro a abrangência e consequências das alterações observadas por um determinado coletor BGP. Apesar disso, cada alteração é registrada como um evento individual pelo módulo de sistema do **Agendador de eventos**, possibilitando reproduzi-las quando necessário.

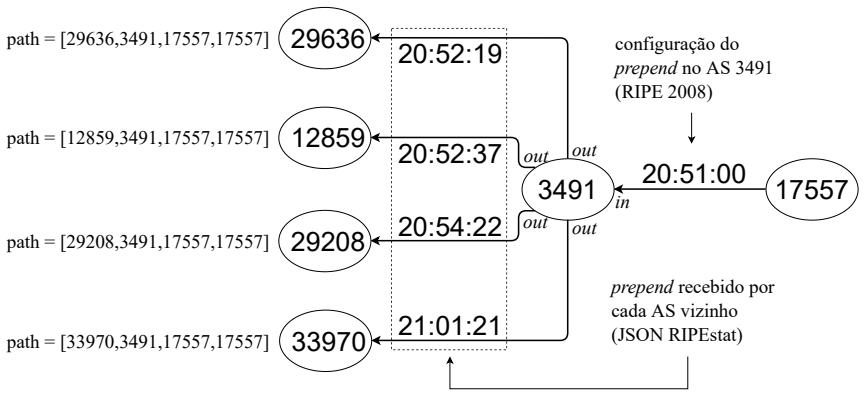
Outra limitação constatada, é que os registros do arquivo JSON não informam se o *prepend* é *in* ou *out*, nem onde ele ocorreu. Informam apenas quais ASs receberam o `AS_PATH` já modificado. Nesse caso, a configuração incorreta do tipo de *prepend* e qual

AS modifica o caminho podem gerar um comportamento diferente no BGP no ambiente reproduzido, quando comparado aos registros do RIPEstat.

Devido às limitações das informações disponíveis, o módulo de sistema do **Agendador de eventos** deve inferir em qual AS o *prepend* será configurado e seu tipo, se *in* ou *out*. Para entendimento do método utilizado na *testbed*, a Figura 9 traz a representação de uma parte da topologia extraída do repositório RIPEstat, correspondendo ao cenário descrito em RIPE (2008).

A topologia representada na Figura 9, corresponde à relação de vizinhança existente entre os ASs 17557, que divulgava incorretamente parte do prefixo IP do Youtube (208.65.153.0/24), e o AS 3491, que tinha a função de trânsito nessa relação. Por sua vez, o AS 3491 tinha como *peer* BGP os ASs 29636, 12859, 29208, 33970, entre outros que não foram representados na figura.

Figura 9 – Parte da topologia retirada do repositório RIPEstat utilizada para análise do método de inferência do tipo de *prepend* e em qual AS ele será configurado



Fonte: Produzido pelos autores

Nesse caso, o relatório da RIPE (2008) informa que o AS 3491 realizou o *prepend* dos anúncios originados no AS 17557, às 21 horas e 51 minutos, inserindo uma repetidão desse ASN no `AS_PATH`. Porém, os registros contidos no arquivo JSON, obtido no repositório RIPEstat, somente apresentam eventos relacionados a esse *prepend* a partir das 20 horas, 52 minutos e 19 segundos, quando o AS 29636 recebeu o primeiro `UPDATE` contendo essa informação. Por sua vez, o último *peer* BGP a receber o `UPDATE` informando o *prepend* foi o AS 33970, às 21 horas, 01 minuto e 21 segundos, totalizando 10 minutos e 21 segundos de tempo decorrido entre o momento teórico da configuração do *prepend* ao efetivo recebimento do `UPDATE` pelo *peer* BGP.

Como discutido anteriormente, existem motivos variados que podem gerar diferença de tempo entre os `UPDATEs` recebidos na topologia, mas essa informação não está disponível no arquivo JSON. Nesse caso, se o *prepend* for implementado na interface de entrada do roteador do AS 3491 (*prepend in*), seus *peers* receberão o `UPDATE` com essa informação

quase que simultaneamente, causando discrepância nas rotas da topologia reproduzida, quando comparadas aos registros do arquivo JSON. Por isso, optamos por programar o módulo de sistema do **Agendador de eventos**, para configurar o *prepend* nas interfaces de saída do roteador do próximo AS, após a ocorrência do *prepend* no `AS_PATH` (*prepend out*). Essa abordagem suporta configurações independentes de *prepend* para cada *peer* do AS, respeitando a relação de tempo das ocorrências no arquivo JSON.

A exceção ocorre quando o *prepend* é observado apenas no último AS do caminho. Como exemplo, o `AS_PATH` [31323, 3549, 3549, 3491, 17557] indica que o ASN 3549 sofreu *prepend*, mas esse evento foi observado apenas pelo roteador do AS 31323. Nesse caso, como o AS 31323 não repassará essa informação para nenhum outro roteador no BGP, por ser o último do `AS_PATH`, o *prepend* é configurado como *in* em sua interface de entrada.

Considerando a topologia apresentada na Figura 9, como o *prepend* observado foi do ASN 17557 (exemplo do parâmetro `path` do AS 29636 [29636, 3491, 17557, 17557]), o roteador do AS 3491 deve receber uma nova configuração de *prepend out* do ASN 17557, a cada novo evento observado no arquivo JSON para cada *peer*. Dessa forma, a *testbed* garante que os `UPDATES` sejam emitidos pelos roteadores BGP, nos momentos exatos contantes dos eventos no arquivo JSON.

5.2.1.3 Exclusão de rotas (*withdraw*)

No arquivo JSON resultante da consulta ao repositório RIPEstat, os eventos relacionados ao *withdraw* de um prefixo IP recebem o valor W no parâmetro `type`. Além disso, o registro contém o código identificador do AS coletor que observou o *withdraw* (`source_id`), o momento da ocorrência (`timestamp`) e o prefixo IP removido (`target_prefix`).

Quanto ao *withdraw* de um prefixo causado pelos filtros configurados nos roteadores, pode ser gerado na interface de entrada do roteador (*withdraw in*) ou na interface de saída (*withdraw out*). Basicamente, a diferença entre eles é que, no *withdraw in*, o próprio roteador que realiza o filtro passa a não contar mais com aquela rota. Já no *withdraw out*, o roteador que faz o filtro ainda possui um `AS_PATH` válido para aquele prefixo, mas não divulga aos seus vizinhos no BGP.

Os registros de *withdraw* existentes no arquivo JSON são apenas do tipo *in*, pois os eventos correspondem aos anúncios recebidos pelos ASs coletores da RIPE, indicando a ocorrência do *withdraw*. É importante ressaltar que os registros não informam qual AS originou o evento, portanto, o bloqueio das atualizações para o prefixo IP deve ser realizado em todas interfaces do roteador BGP.

5.2.1.4 Modos de restrição dos anúncios BGP

Ao identificar todas ocorrências, o módulo **Agendador de eventos** solicita que o utilizador defina o modo de restrição que será respeitado nos anúncios de rotas no BGP. Os modos disponíveis são **permissivo** e **restritivo**. De modo geral, no modo **permissivo**

cada roteador recebe apenas as configurações que correspondem aos eventos agendados; enquanto no modo **restritivo**, todos roteadores recebem uma configuração prévia bloqueando o anúncio e recebimento de atualizações no BGP, necessitando que cada evento executado posteriormente seja liberado explicitamente nos roteadores que fazem parte do `AS_PATH`.

Apesar de mais complexo para administrar, o modo **restritivo** garante maior controle de como os eventos BGP são recebidos e processados em cada roteador da topologia. Esse tipo de restrição permite, por exemplo, reproduzir o comportamento dos ASs que possuem políticas internas de filtro baseadas em `COMMUNITY` (seção 2.1). Apesar do conteúdo das `COMMUNITY` não ser de conhecimento público, portanto, não constarem nos repositórios do RIPEstat, seu comportamento pode ser inferido a partir do `AS_PATH` observado nos objetos do arquivo JSON e reproduzido com as restrições que o modo **restritivo** suporta.

Outra vantagem do modo **restritivo**, é tornar a topologia reproduzida a partir dos dados do Projeto RIPEstat, aderente à topologia criada com os dados do Projeto AS-Relationship, do CAIDA (seção 2.2.2). Isso é possível porque o controle dos caminhos implementado na rede pelo modo **restritivo**, onde são autorizados apenas os `AS_PATH` observados na topologia real, traduzem os acordos comerciais reportados na base do AS-Relationship.

5.2.1.5 Implementando os eventos no roteador BGP

Tecnicamente, o método utilizado para configurar os eventos no roteador BGP depende da solução utilizada. Apesar disso, os conceitos envolvidos geralmente podem ser replicados na maioria das tecnologias atuais.

Como exemplo, o método utilizado para anunciar os prefixos IP agendados no **Agendador de eventos**, depende da alteração da configuração do roteador BGP durante a execução do cenário de ataque. Nesse caso, no tempo correspondente ao anúncio cadastrado, deve-se incluir o novo prefixo IP na configuração do processo BGP do roteador que irá anunciá-lo. Como apresentado na seção 5.2.1.1, a configuração de um novo prefixo específico deve ser realizada apenas uma vez durante a execução do cenário.

Quanto às alterações do `AS_PATH`, também dependem da solução de roteador utilizada e do modo de restrição dos anúncios BGP escolhido. No modo de restrição **permissivo**, por exemplo, as alterações no `AS_PATH`, verificadas no arquivo JSON, não geram mudanças nas configurações dos roteadores BGP, já que não há restrição prévia configurada em nenhuma máquina. Nesse caso, espera-se que toda e qualquer mudança de caminho na rede ocorra naturalmente no cenário reproduzido, apenas baseando-se nas convergências e métricas do próprio protocolo.

Já no modo **restritivo**, os roteadores possuem configurações prévias que restringem o tráfego BGP na rede. Como uma atualização de `AS_PATH` traz basicamente as informações do prefixo IP divulgado e o `AS_PATH` correspondente, pode-se utilizar as técnicas

de verificação de prefixo e `AS_PATH`, associadas à filtragem do tráfego de entrada e saída nas sessões com os vizinhos BGP. Como exemplo, considerando o roteador Quagga, os filtros de prefixo podem ser implementados utilizando o *ip prefix list*, enquanto os filtros de `AS_PATH` podem ser implementados com o *ip as-path access-list*. Por sua vez, a implementação dos filtros nas interfaces com os vizinhos BGP, podem empregar *route-map* para o tráfego *in* e *out*.

Finalizando, o *withdraw* de prefixo segue o mesmo método de configuração do modo **restritivo**, utilizando filtros baseados em *ip prefix list*, *ip as-path access-list* e *route-map* *in* ou *out*, de acordo com os eventos cadastrados no **Agendador de eventos**.

5.2.2 Geração de eventos sintéticos no BGP

Além de possibilitar o agendamento de eventos BGP extraídos automaticamente das ocorrências reais observadas na Internet, o módulo **Agendador de eventos** também permite a inclusão de ocorrências geradas sinteticamente. Nesse caso, o utilizador da *testbed* deve especificar o tipo de evento que será agendado e o momento em que deseja executá-lo na linha de tempo do cenário. Os tipos possíveis são:

a) *hijack* ou *route leak*: nesse caso o utilizador deve informar o prefixo que será anunciado e o AS que divulgará;

b) *prepend* de AS: nesse tipo de evento o utilizador deve informar qual AS anunciará o *prepend*; a direção, podendo ocorrer na entrada do AS (*in*) ou na saída (*out*); qual AS sofrerá *prepend* e a quantidade de repetições;

c) *withdraw* de prefixo: esse evento corresponde ao bloqueio ou retirada dos anúncios de prefixos IP em um determinado AS. O *withdraw* pode ocorrer na entrada ou na saída do AS (*in* ou *out*).

A *testbed* também conta com o módulo **Gerador de eventos**, que é uma interface que possibilita o utilizador acessar diretamente o terminal de configuração dos roteadores BGP criados no **Emulador**. Esse módulo possibilita a inclusão de qualquer tipo de evento suportado pela solução de roteador BGP utilizada, além disso, os eventos podem ser inseridos ou retirados a qualquer momento na execução do cenário de ataque. Nesse caso, um ponto relevante é que o utilizador deve conhecer os comandos correspondentes ao roteador BGP utilizado.

5.3 Tecnologia de Emulação da *Testbed*

Como discutido na quarta questão de pesquisa (seção 1.4), a emulação de rede reúne as condições necessárias para ser utilizada como tecnologia base da *testbed* no estudo da segurança do BGP. Nessa linha, as seções seguintes apresentam os procedimentos utilizados na avaliação das tecnologias de emulação leve e distribuída apresentadas na

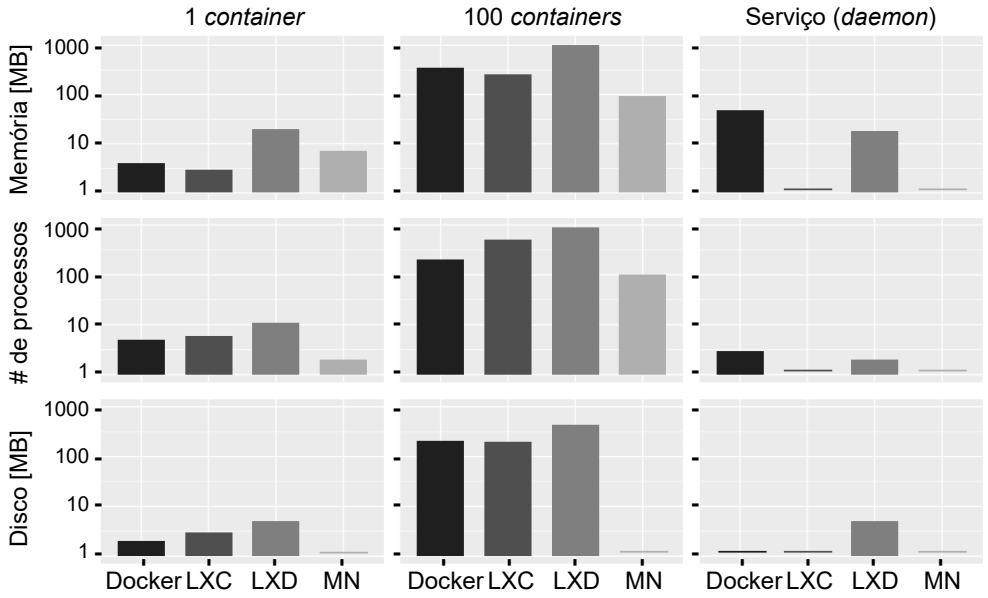
seção 2.3, identificando a que melhor atende aos requisitos de escalabilidade e realismo da *testbed* para ser utilizada como base do módulo **Emulador**.

5.3.1 Desempenho das Tecnologias de Emulação Leve

Por se tratar de um ambiente que deve suportar a reprodução de topologias de rede contendo centenas ou até milhares de roteadores BGP, as soluções Mininet, LXC, LXD e Docker passaram por avaliações para caracterizar seu comportamento quanto à escalabilidade e o consumo de recursos computacionais na emulação de *containers* básicos. Nessa linha, a Figura 10 apresenta a utilização de *Random Access Memory* (RAM), número de processos Linux criados e espaço em disco consumido em cenários com a criação de 1 e 100 *containers* básicos com cada solução.

Como o LXC, LXD e Docker utilizam imagens de Sistema Operacional (SO) na criação dos *containers*, optamos por utilizar as imagens menores e mais simples disponíveis em seus repositórios oficiais, consumindo o mínimo de recursos da máquina hospedeira possível na criação de *containers* básicos. Para isso, foram utilizadas as imagens do Linux Alpine 3.4 AMD641 com LXD e Linux BusyBox 1.29 AMD642 com LXC e Docker.

Figura 10 – Utilização de recursos computacionais pelas soluções de virtualização leve



Fonte: Produzido pelos autores

Analisando os dados apresentados na Figura 10, é possível observar que o Mininet utiliza aproximadamente 90 MB de memória RAM da máquina hospedeira quando instanciados 100 *containers*, enquanto o LXC utiliza aproximadamente 260MB, o Docker 350MB e o LXD 1.02GB. Nesse caso, o consumo observado está diretamente relacionado

às características das imagens de SO utilizadas na criação dos *containers*, pois ela carrega consigo todo conjunto de códigos, bibliotecas, variáveis de ambiente e arquivos de configuração necessários para seu funcionamento. Quanto à quantidade de processos criados para instanciação de 100 *containers*, o Mininet cria apenas 101 processos, o Docker 202, o LXC 501 e o LXD 902 processos.

Quanto ao consumo de disco, o Mininet não faz uso algum, visto que seus *containers* utilizam um mesmo ponto de montagem comum a todos usuários no sistema de arquivos da máquina hospedeira, enquanto o LXC e LXD consomem respectivamente 210 MB e 430 MB cada, devido à criação de sistemas de arquivos virtuais independentes para cada *container*. Já o Docker, utiliza aproximadamente 140MB de espaço em disco para criação dos 100 *containers*, devido a técnica de *copy-on-write*.

Com base nos dados apresentados, é possível observar que o Mininet consome menos recursos computacionais da máquina hospedeira no processo de criação de *containers* básicos quando comparado às outras tecnologias apresentadas.

5.3.2 Validação do Mininet como Base da *Testbed*

Concluído que o Mininet é a solução de emulação leve que consome menos recursos computacionais para instanciação de um número elevado de *containers*, restava avaliar seu comportamento quando utilizado como base para reprodução de topologias de rede complexas contendo sistemas de roteadores BGP reais. Para essa etapa do trabalho, foi desenvolvido um protótipo da *testbed*.

De modo geral, esse protótipo fez uso da plataforma de emulação Mininet versão 2.2.2 (seção 2.3) para suporte dos *containers* dos ASs; utilizou o roteador BGP baseado em *software* Quagga 0.99.24; e um conjunto de outras aplicações responsáveis pelas funções restantes. Utilizou como base um servidor com processador de 3.50GHz e 8 núcleos, além de 16GB de memória RAM e SO Ubuntu 16.04.4 LTS Server.

O acesso ao protótipo da *testbed* foi realizado através de conta de usuário válido no SO e toda interação com a *testbed* aconteceu através de uma aplicação desenvolvida em Python[1] 3.6.3 que fez a interface entre os módulos da arquitetura e as aplicações externas.

Ao iniciar essa aplicação o utilizador tinha a opção de importar a topologia da rede de fonte externa, seguindo o modelo JSON apresentado na Figura 6. Após a importação da topologia na *testbed*, os módulos **Perfil do AS** e **Alocação de endereços IP** foram acionados para continuidade do processo de criação da topologia como apresentado na seção 5.1.5.

Quando finalizadas as funções desses dois módulos, o **Grafo Anotado** foi então gerado com a inserção de todos atributos correspondentes na base de dados da *testbed*. Após essa fase, o módulo **Configuração BGP** consultou os atributos dos objetos de ASs, gerou

[1] Python. https://www.python.org/

e executou os comandos correspondentes à plataforma de emulação utilizada, no caso o Mininet. Esse processo instanciou e configurou todos *containers* de ASs existentes na topologia.

A interação pôde ser feita diretamente nas instâncias dos *containers*, através do emulador de terminais *xterm*, pela interface do próprio Mininet ou através dos arquivos de configuração dos *containers*. Nessas interfaces foi possível acompanhar a divulgação das atualizações BGP, acessar as tabelas de roteamento de cada AS, além de incluir e modificar os parâmetros dos testes de ataques ao BGP.

5.3.2.1 Testes de Validação e Discussão de Resultados

Para validação do protótipo da MiniSecBGP baseada na solução Mininet foram propostos dois testes principais. O primeiro analisou o comportamento da *testbed* com relação à escalabilidade do ambiente, enquanto o segundo avaliou seu funcionamento em um caso específico de ataque ao BGP.

Como resultados, com relação a quantidade de instâncias, o ambiente suportou a criação de até 8188 *containers* sem modificação dos parâmetros de limites do Kernel, levando aproximadamente 1 segundo a cada grupo de 100 novos *containers*.

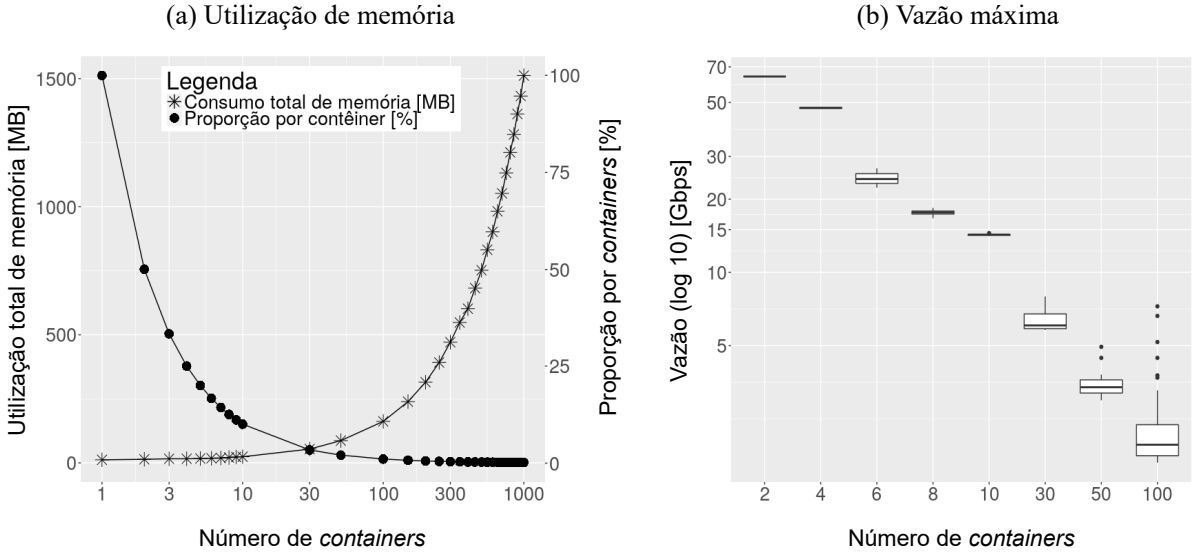
A Figura 11a apresenta a utilização de memória RAM total observada na ativação de até 1000 *containers* (eixo Y esquerdo). Apresenta também a proporção de memória consumida por cada *container* (em porcentagem) sobre a memória total utilizada pelo Mininet (eixo Y direito). Apesar de cada instância utilizar 1,5MB de memória RAM em média, notou-se a alocação de aproximadamente 12MB de memória pelo Mininet durante a instanciação do primeiro *container*. Como parte dessa memória é compartilhada com outras instâncias, a proporção de memória utilizada por *container* diminui com o incremento do número de instâncias, chegando a 0,1% aproximadamente em 1000 *containers*.

Quanto ao número de interfaces de rede, analisou-se a quantidade máxima suportada por um único *container*, onde foram ativadas até 20.000 interfaces com sucesso. Constatou-se também que a inclusão de novas interfaces gera um aumento considerado insignificante no consumo de memória total, pois, a utilização aumenta apenas 0,11% (aproximadamente 10MB) para cada novo grupo de 100 interfaces criadas num mesmo *container*.

Quanto à vazão de rede, inicialmente foram realizados testes utilizando o iPerf3[2] para identificar a sobrecarga gerada pelo Mininet no ambiente. Foram comparadas a vazão máxima do Linux no servidor hospedeiro com um par de *containers* Mininet devidamente instanciados e conectados. Foram realizadas 3 sequências de 10 testes cada um, onde constatou-se uma queda de 5,5% na vazão do ambiente Mininet. Além disso, observou-se também a degradação da vazão com o aumento do número de *containers*. Na Figura 11b é possível observar que há uma degradação elevada na vazão mesmo com um número não

[2] iPerf/iPerf3. https://iperf.fr/

Figura 11 – Consumo de recursos (a) e escalabilidade da vazão (b) do MiniSecBGP em máquina única



Fonte: Produzido pelos autores

tão elevado de instâncias. O ambiente passa de uma vazão inicial de 63,9 Gbps, entre um par de *containers*, para uma vazão média de 2,39 Gbps com 100 *containers* trocando dados entre pares. Além disso, nota-se uma grande dispersão no conjunto das vazões observadas, representando certa instabilidade no ambiente que deve ser considerada na definição dos cenários de testes.

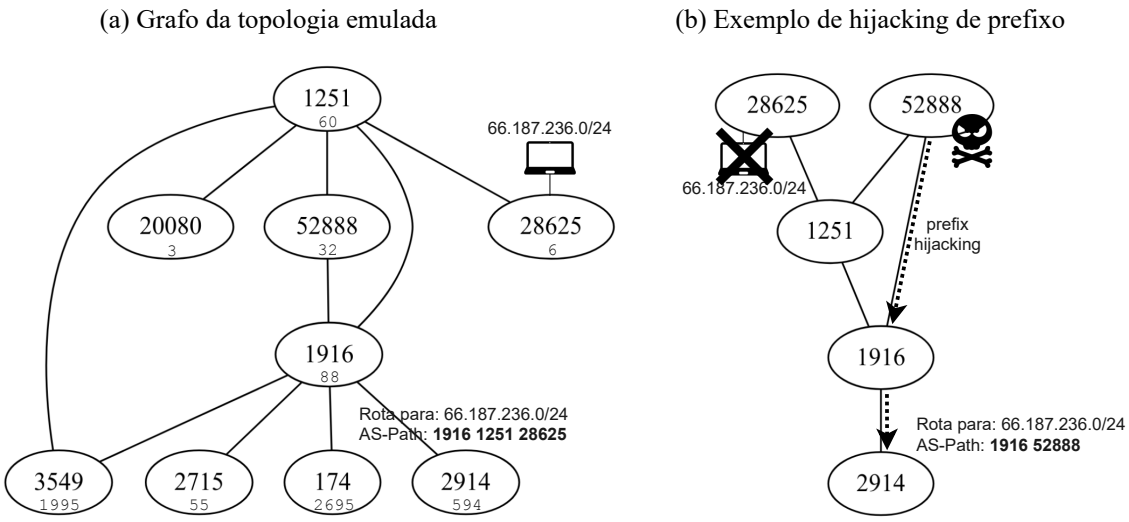
Tal comportamento pode representar uma deficiência relevante da *testbed* em cenários que dependam de altas taxas de transmissão de dados, porém, alguns artifícios como a mudança da base de grandeza da vazão, de Gbps para Mbps ou Kbps, por exemplo, podem diminuir o erro. Outra possível abordagem é a utilização de processamento distribuído da *testbed*, de forma a diminuir a carga de processamento em cada servidor.

Na segunda fase de testes, foi avaliado o comportamento do protótipo da *testbed* na emulação de uma topologia que representa um conjunto de ASs contidos na Figura 12a. A topologia é composta por 9 ASs, representados pelas elipses e identificados pelo ASN (número maior ao centro da elipse), com um total de 5528 prefixos divulgados no BGP (números menos representados na parte inferior de cada AS). Todos ASs estão divulgando e recebendo todas rotas conhecidas (*full routing*).

Nesse cenário foi avaliada a integração dos módulos da arquitetura da MiniSecBGP, partindo da importação da topologia até a instanciação de todo ambiente no Mininet.

Como a topologia foi importada de fonte externa, seu tempo de criação não foi computado, porém, foi observado o tempo de conversão do modelo JSON para código Mininet (3 segundos) e o tempo de ativação do ambiente no Mininet (5,5 segundos).

Figura 12 – Representação gráfica de conjunto de ASs e suas conexões (a), com exemplo de ataque de *hijacking* de prefixo no BGP (b)



Fonte: Produzido pelos autores

Quanto à utilização de memória, somente a instanciação da topologia consumiu 21MB da máquina, enquanto que todas tabelas de roteamento devidamente populadas consumiram mais 75MB de memória para um total de 10.466 prefixos aprendidos por cada AS. Desse modo, o consumo total de memória do protótipo da *testbed* foi de 96MB.

Se compararmos o total de prefixos mantidos nas tabelas de rotas dos 9 ASs (94.194 rotas) e o consumo de memória correspondente (75MB), com a estimativa apresentada na seção 2.1, observamos que o ambiente segue o conceito apresentado quanto ao consumo de memória por número de prefixos existentes na tabela de roteamento.

5.3.3 Avaliação de Soluções de Mininet Distribuído

Como observado nos resultados dos testes realizados com o protótipo MiniSecBGP, a *testbed* com processamento centralizado em uma única máquina apresentou limitações de escalabilidade que inviabilizam sua utilização na reprodução da topologia global da internet. Buscando resolver essa deficiência, foram identificadas opções de soluções de emulação leve com suporte a processamento distribuído (Mininet Cluster e MaxiNet - seção 2.3), que pudessem ser utilizadas como base para a *testbed*.

Nesta seção são avaliadas as duas soluções de Mininet distribuído. A avaliação foi baseada na análise do consumo de recursos computacionais em um *cluster* com as tecnologias Mininet Cluster e MaxiNet. Para isso, foram emuladas redes de *data center* com topologias compostas pelos elementos *host* do Mininet, representando os servidores da rede; *switch*, correspondendo aos switches utilizados na interligação dos servidores; e

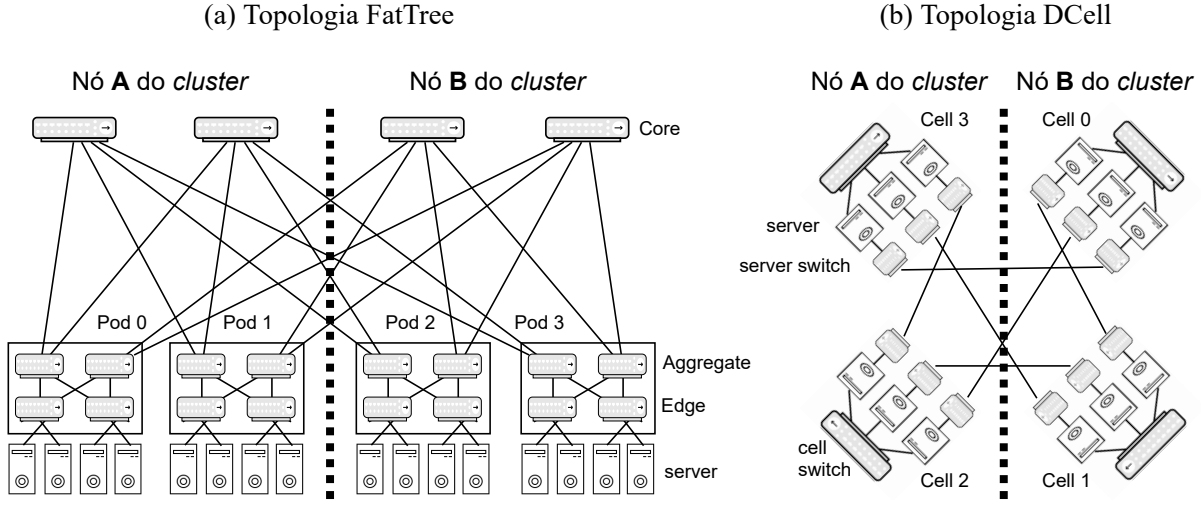
link, responsáveis pela conexão entre servidores e switches. Nos cenários testados houve variação da quantidade de elementos Mininet criados, nós do *cluster* alocados e tecnologia Mininet utilizada para processamento distribuído.

Durante todo processo de criação e ativação dos elementos da topologia, monitorou-se o efeito percebido pela variação de um único fator, ou um conjunto deles, na utilização de memória RAM, quantidade de processos Linux gerados, conexões de rede estabelecidas entre os nós físicos do *cluster* e tempo de ativação de cada serviço.

5.3.3.1 Topologias e Particionamento

Foram emuladas duas topologias de *data center* com diferentes requisitos computacionais. A primeira é a FatTree, Figura 13a, reconhecida pela alta ocorrência de switches e enlaces de dados por servidor na topologia. A segunda é a topologia DCell, Figura 13b, onde a proporção de servidores é maior quando comparada à quantidade de enlaces e switches na topologia. Tal comparação tem como objetivo identificar se há relação direta entre o consumo de recursos computacionais do *cluster* e o tipo de elemento Mininet instanciado.

Figura 13 – Representação das topologias FatTree (a) e DCell (b) utilizadas na avaliação das soluções de Mininet distribuído



Fonte: Produzido pelos autores

Buscando garantir a distribuição igualitária da carga de processamento no *cluster*, foi utilizada a técnica de *Round Robin* no particionamento dos k elementos da topologia entre seus nós, alocando também todos elementos de um único *pod*, na topologia FatTree, e mesma *cell*, na topologia DCell, num mesmo servidor.

Devido ao fato do MaxiNet suportar comunicação remota apenas entre elementos Mininet do tipo *switch*, como apresentado na seção 2.3, foram incluídos switches nos

enlaces entre servidores na topologia DCell, identificados como *server switch* na Figura 13b.

5.3.3.2 Metodologia de Testes

A avaliação das soluções de Mininet distribuído depende do ajuste dos parâmetros de fatores de controle, que são testados simultaneamente e podem assumir diferentes níveis. Para cada combinação dos possíveis níveis, é necessário executar uma ou mais sequências de testes que geram resultados para serem analisados, porém, a definição do ajuste em cada parâmetro não é algo trivial, bem como o aumento na quantidade de fatores e níveis testados aumenta consideravelmente o tamanho do experimento, podendo até inviabilizar a obtenção de resultados significativos.

Uma alternativa para experimentação em ambientes com essa característica é a utilização da técnica de planejamento fatorial (MONTGOMERY, 2006). Planejamento fatorial possibilita medir os efeitos, ou influências, de uma ou mais variáveis na resposta de um processo. Essa técnica consiste na identificação de um conjunto finito de fatores que influenciam o comportamento do ambiente, atribuindo valores específicos e válidos aos níveis de cada fator, o que altera o resultado das variáveis monitoradas. A relação entre os fatores e níveis é do tipo exponencial, $(niveis)^{fatores}$, e sua ocorrência mais comum é a 2^k fatorial, onde um conjunto de k fatores assume dois níveis de valores possíveis cada.

Partindo deste princípio, foi definido uma sequência de 3 fatores de controle com 2 níveis de variação cada para a análise experimental das soluções de Mininet distribuído, formando a representação de um experimento do tipo 2^3 fatorial. Figura 14a. O primeiro fator corresponde ao tipo de tecnologia Mininet distribuído ($Mininet\ Distribuído$), com níveis variando entre MaxiNet (MN) e Mininet Cluster com túneis GRE (MC). O segundo fator é o tamanho da topologia ($Tamanho\ Topologia$), que corresponde à quantidade de *pods*, na topologia FatTree, e *cells*, na topologia DCell, com níveis 4 e 12. Tal variação visa garantir a análise experimental em topologias compostas por poucos e muitos elementos. O terceiro fator é a quantidade de nós do *cluster* ($Tamanho\ Cluster$), com níveis 2 e 4, dessa forma, todo ambiente é testado em um *cluster* composto por 2 e 4 nós. Cada fator recebe uma denominação do tipo $x1$, $x2$, ..., xn, que simplifica sua identificação.

No planejamento de um experimento fatorial 2^k, os dois níveis de cada fator são denominados nível baixo e nível alto, podendo ser identificados com os valores (-1) no nível baixo e (+1) no nível alto, conforme apresentado na coluna de níveis da Figura 14a. A partir dessa definição foi possível identificar as combinações de testes do experimento, Figura 14b, cuja definição da distribuição dos níveis dos fatores no plano seguiu a definição:

a) Para $x1$, o sinal da coluna de (1) alterna em grupos de $2^0 = 1$, ou seja, seguidamente.

b) Para $x2$, o sinal da coluna de (1) alterna em grupos de $2^1 = 2$, ou seja, em pares.

Figura 14 – Tabela de fatores (a) e Matriz de planejamento (b) para experimentação das soluções de Mininet distribuído utilizando 2^3 fatorial

(a) Tabela de fatores

	fatores	níveis -1	níveis +1	resultados	
x1	Mininet Distribuído	MN	MC	RAM	proc.
x2	Tamanho Topologia	4	12	RAM	proc.
x3	Tamanho Cluster	2	4	RAM	proc.

(b) Matriz de planejamento

	fator de controle			
exp.	x1	x2	x3	seq.
1	-1	-1	-1	5
2	+1	-1	-1	7
3	-1	+1	-1	1
4	+1	+1	-1	8
5	-1	-1	+1	2
6	+1	-1	+1	3
7	-1	+1	+1	4
8	+1	+1	+1	6

Fonte: Produzido pelos autores

c) Para $x3$, o sinal da coluna de (1) alterna em grupos de $2^2 = 4$, ou seja, 4 vezes (-1) seguidos de 4 vezes (+1).

Por fim, foi definido a sequência aleatória para execução de cada teste do experimento, minimizando a possibilidade de interferência de fontes de variação não assinaláveis no ambiente. Concluídas essas etapas, o experimento pôde então ser executado.

5.3.3.3 Ambiente e Tecnologias

O *cluster* utilizado no experimento era composto por 4 servidores. Cada um contendo 1 processador com 8 núcleos de 2.40GHz, 8GB de memória RAM e duas interfaces de rede de 1Gbps cada. O SO era o Ubuntu Server 16.04.5 LTS, kernel 4.4.0-138, e os principais programas instalados eram o Mininet 2.3.0d4, MaxiNet 1.2, *Open vSwitch* 2.5.5 e Pyro4.

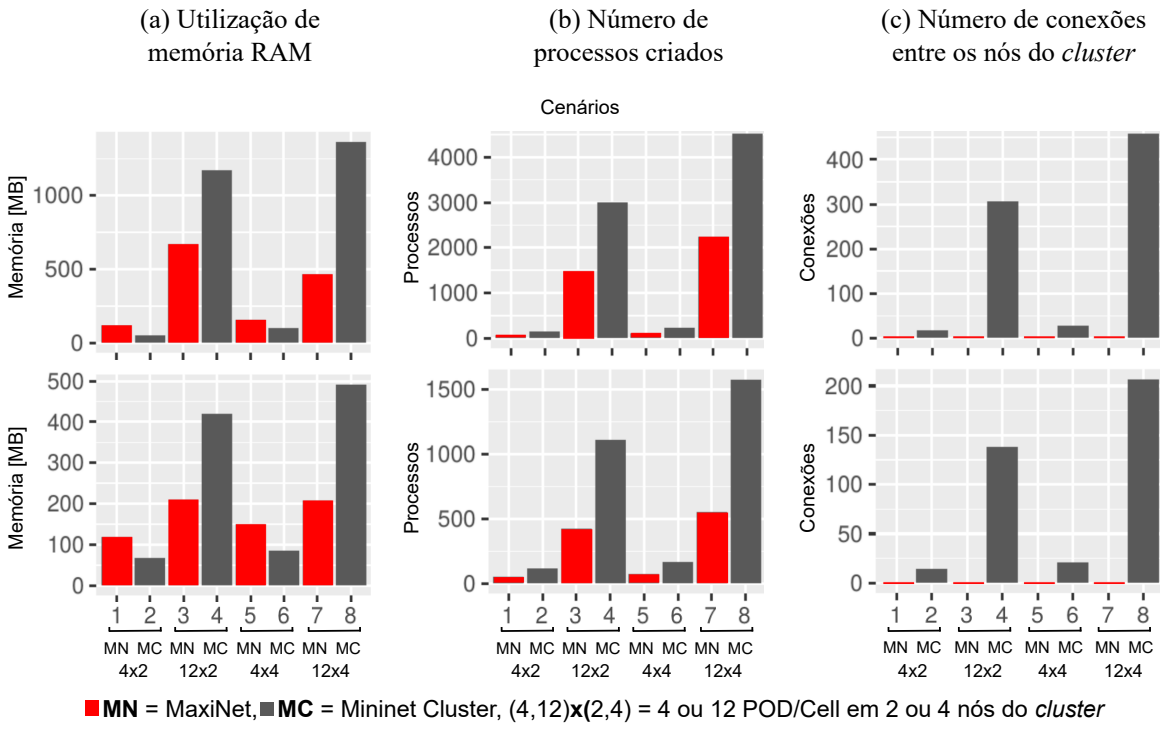
Cada servidor estava conectado em uma rede de experimentação totalmente isolada, composta por um *switch* com interfaces de 1Gbps em topologia estrela. Essa rede era utilizada exclusivamente para troca de mensagens das soluções de Mininet distribuído na gestão do ciclo de vida do experimento. A segunda interface de rede de cada servidor estava conectada em um segundo *switch* com interfaces de 1Gbps, também em topologia estrela, numa rede independente para administração dos servidores. A topologia física utilizada na interligação dos nós do *cluster* buscou impedir a interferência de fatores externos não previstos nos resultados do experimento.

5.3.3.4 Testes de Validação e Discussão dos Resultados

Após as definições apresentadas, foram realizados os testes para avaliação das soluções de Mininet distribuído seguindo a sequência definida durante o planejamento do experimento, Figura 14b, com um total de 10 replicações para cada cenário.

A Figura 15 apresenta os resultados das medições realizadas, apontando a utilização total de memória RAM do ambiente, número de processos Linux criados e de conexões de rede estabelecidas entre os nós do cluster durante todo processo de criação dos elementos Mininet e recursos adicionais necessários em cada cenário avaliado.

Figura 15 – Utilização de memória (a), número de processos Linux criados (b) e conexões de rede estabelecidas entre os nós do *cluster* (c) para as topologias FatTree e DCell



Fonte: Produzido pelos autores

De forma geral, analisando a Figura 15a é possível observar que o total de memória utilizada no *cluster* variou consideravelmente entre todos cenários contendo 4 e 12 *pods* ou *cells*, partindo de aproximadamente 55MB de memória RAM utilizada no cenário 2 da topologia FatTree, cenário composto por 4 *pods* distribuídos em 2 nós de *cluster* utilizando Mininet Cluster, para até 1.4GB de memória utilizada no cenário 8 da mesma topologia, que conta com 12 *pods* distribuídos em 4 nós de *cluster* utilizando Mininet Cluster.

Por outro lado, também é possível observar variação nos resultados, mesmo em menor escala, entre cenários contendo o mesmo número de nós de *cluster*, *pods* ou *cells*, variando apenas a tecnologia de Mininet distribuído.

Um exemplo desse comportamento pode ser observado entre os cenários 7 e 8 da topologia DCell. O cenário baseado em MaxiNet apresenta consumo de memória RAM equivalente à 42% quando comparado ao Mininet Cluster. Tal fato ocorreu devido à forma como as duas tecnologias administram os elementos Mininet remotos. O Maxinet, como

apresentado na seção 2.3, utiliza o Pyro4 para gerenciar os objetos remotos do Python, portanto, novos elementos Mininet são solicitados diretamente ao servidor Pyro4 que se encarrega de enviar a solicitação ao Mininet da máquina remota para execução. Já o Mininet Cluster cria uma nova conexão SSH independente entre o nó *master*, máquina onde o Mininet está sendo executado inialmente, e o nó *slave*, máquina onde o elemento deve ser criado, carregando o código *mnexec* responsável pela criação do elemento remoto. Esse túnel SSH permanece estabelecido enquando o elemento remoto existir. Tal procedimento consome memória, processos e conexões entre os nós do *cluster* envolvidos no processo.

O fato do MaxiNet utilizar o Pyro4 no gerenciamento de objetos Python, gera a necessidade de que esse serviço esteja em execução permanente em todos nós do *cluster*, antes mesmo que qualquer elemento Mininet local ou remoto seja criado. Isso justifica o consumo superior de memória RAM do MaxiNet em cenários contendo poucos elementos Mininet, como por exemplo, entre os cenários 1 e 2 das topologias FatTree e DCell. Nesse caso, o serviço Pyro4 consome aproximadamente 55MB de memória RAM, no nó *FrontEnd*, e 25MB de memória RAM em cada nó *worker*, enquanto o Mininet Cluster não depende ou necessita de serviço prévio.

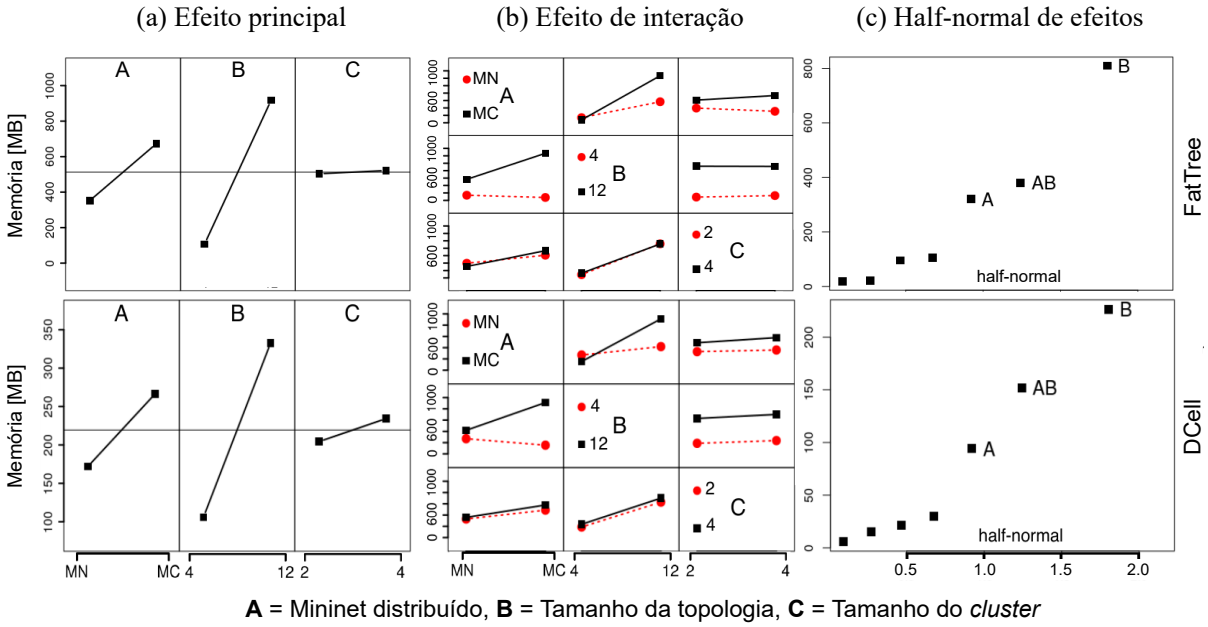
Quanto aos processos Linux criados em cada cenário, Figura 15b, sua variação também tem relação direta com o número de elementos Mininet da topologia, pois, cada elemento local ou remoto faz uso de pelo menos 1 novo processo Linux.

Já o número de conexões entre os nós do *cluster*, Figura 15c, tem relação direta com a tecnologia Mininet utilizada. Nesse caso, como o Mininet Cluster cria túneis SSH independentes entre os nós *master* e *slave* a cada novo elemento remoto, esse comportamento reflete no grande número de processos Linux e conexões em topologias maiores. O MaxiNet, por sua vez, cria um número de processos Linux diretamente proporcional ao número de elementos existentes na topologia, além de apresentar poucas conexões remotas devido ao fato delas serem responsáveis apenas pela troca de mensagens entre objetos do Pyro4. Esse comportamento explica a grande diferença no número de processos e conexões em cenários com o mesmo número de elementos e nós de *cluster*, onde varia apenas a tecnologia Mininet utilizada, como demonstrado nos cenários 7 e 8 das topologias FatTree e DCell.

Outra análise importante é o efeito causado pela variação isolada de cada fator, ou um conjunto deles, no resultado. Essa análise ajuda identificar a melhor estratégia para escalar as topologias de rede emuladas, consumindo o mínimo de recursos do *cluster*. A Figura 16a apresenta o efeito de cada fator independe, também conhecido como efeito principal. Esse efeito é analisado observando a variação do consumo de memória, que corresponde ao grau de inclinação da reta que representa o fator, quando passado do nível (-1) para o nível (+1). Como resultado, é possível notar no gráfico apresentado na Figura 16a, que o fator com maior efeito é o *Tamanho Topologia*, seguido pelo *Mininet Distribuído*. Já o

Tamanho Cluster tem pouco efeito no consumo de memória do ambiente.

Figura 16 – Relação do efeito principal (a), efeito de interação (b) e *half-normal* de efeitos (c) observados na experimentação para as topologias FatTree e DCell



Fonte: Produzido pelos autores

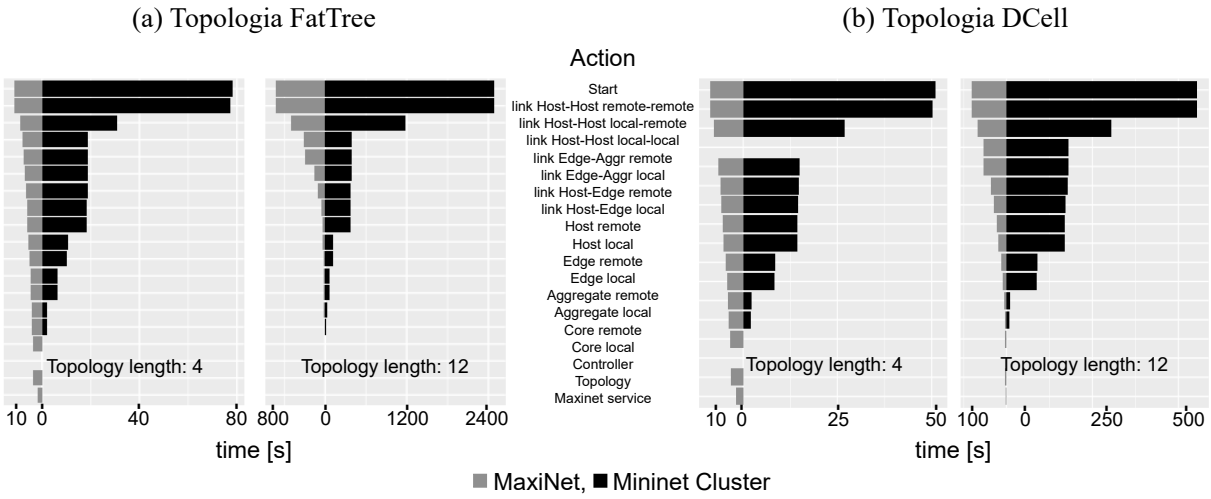
Quanto ao efeito de interação entre os fatores avaliados, pode ser identificado pela variação do ângulo entre as retas dos fatores na Figura 16b. Os resultados apresentados permitem concluir que a interação entre os fatores *Tamanho Topologia* e *Mininet Distribuído* causa oscilação no consumo de memória, comportamento não observado de forma relevante na interação dos outros fatores.

Para mensurar o efeito de um fator, ou conjunto deles, sobre o resultado, pode-se fazer uso do gráfico de *half-normal* de efeitos, Figura 16c. Como exemplo, nele é possível observar que o fator *Mininet Distribuído* causou uma variação de aproximadamente 380MB no consumo de memória, enquanto sua interação com o fator *Tamanho Topologia* causou um efeito de 400MB e o fator *Tamanho Topologia* causou um efeito de 880MB de consumo de memória na topologia FatTree.

Outra informação importante que deve ser considerada é o tempo gasto para executar cada ação necessária para a conclusão do experimento. A Figura 17 detalha o tempo utilizado para executar todas ações de configuração dos cenários baseados em 4 nós de *cluster* e 4 ou 12 *pods* ou *cells*, para a topologia FatTree (Figura 17a) e DCell (Figura 17b). Os resultados mostraram que o Mininet Cluster gastou mais tempo para concluir as ações de configuração para todos os cenários. Este comportamento é uma consequência do tempo gasto pelo Mininet Cluster para abrir conexões SSH entre os nós do *cluster*

e enviar os comandos Mininet responsáveis por gerenciar os elementos remotos e links, conforme discutido na seção 2.3. Como exemplo, no cenário da topologia FatTree com comprimento de topologia de 4 *pods* (Figura 17a), o Mininet Cluster gastou 8 vezes mais tempo para completar todas as ações de configuração quando comparado ao MaxiNet.

Figura 17 – Tempo gasto por ação de configuração na avaliação da solução Mininet distribuída nas toplogias FatTree (a) e DCell (b)



Fonte: Produzido pelos autores

Com a análise dos resultados apresentados, é possível concluir que, apesar do Mininet Cluster ser a implementação oficial de Mininet distribuído, apresenta desempenho inferior ao MaxiNet. O MaxiNet possui estrutura de gerência de elementos remotos mais eficiente com a utilização do Pyro4, consumindo menos recursos computacionais do *cluster*. Todas essas características levam a concluir que o MaxiNet é a solução de emulação leve e distribuída mais indicada para ser utilizada como base do módulo **Emulador** da *testbed*.

5.3.4 Administração do roteador BGP nos *containers* MaxiNet

Após definido que o MaxiNet é a solução de emulação leve e distribuída mais indicada como base dos roteadores na *testbed*, iniciou-se os trabalhos para avaliar se os *containers* Mininet suportariam o *software* de roteador Quagga, com todo isolamento necessário em um ambiente de *cluster*, suportando a troca de anúncios de rotas BGP entre as instâncias do roteador distribuídas no *cluster* MaxiNet.

Para atender esse requisito, o módulo **Emulador** da arquitetura passou a contar com um sistema que distribui os arquivos de configuração do Mininet e do roteador Quagga nos nós do *cluster* MaxiNet. Esses arquivos são necessários para instanciar os *containers* com as configurações dos ASs, além da lista de prefixos divulgados no BGP e os *links* utilizados para comunicação entre os ASs que possuem relação de vizinhança.

Na *testbed*, cada AS é representado por um roteador BGP Quagga. Por sua vez, cada roteador corresponde a uma instância independente do elemento Mininet do tipo *host*, que deve suportar a execução dos serviços `Zebra` e `bgpd` em processos independentes por *container*, garantindo o isolamento do Quagga conforme apresentado na seção 2.1.1.

Ao criar a topologia Mininet, inicialmente são instanciados os elementos *host* de cada roteador, inseridos os *links* de dados de acordo com a relação de vizinhança estabelecida no BGP, e iniciado os processos `Zebra` e `bgpd` internos do *container*. Como o MaxiNet não suporta conexão direta entre elementos do tipo *host* hospedados em máquinas diferentes (seção 2.3.2), são inseridos elementos do tipo *switch* entre os roteadores vizinhos no BGP que estão hospedados em nós diferentes do *cluster*.

Finalizada a etapa de criação dos roteadores BGP, passou-se à fase que visava garantir a execução dos eventos de ataques ao BGP previamente cadastrados no **Agendador de eventos**. Para isso, todo evento BGP é inserido em um arquivo no nó *master* do *cluster* MaxiNet, que é lido continuamente em forma de *loop* durante a execução do cenário de ataque. Quando chega o momento de executar um evento específico, o nó *master* conecta no nó onde o elemento do roteador foi criado via SSH, executando o comando correspondente ao evento no Quagga. Esse processo é realizado com uma chamada Python ao SO local (*os.popen*) no nó *master*, utilizando o módulo Python *pexpect*[3] para executar o comando no Quagga diretamente na console do roteador BGP remoto.

[3] Pexpect. https://pexpect.readthedocs.io/en/stable/

Capítulo 6

MiniSecBGP

Após todas definições apresentadas até aqui, foi desenvolvida a *Minimalistic Security* BGP (MiniSecBGP). A MiniSecBGP é uma *testbed* que une um arcabouço de aplicações de rede e emulação leve, algumas pré-existentes e já bem conhecidas, enquanto outras foram desenvolvidas especialmente para esse fim.

De modo geral, a MiniSecBGP é um sistema desenvolvido em Python 3, acessível via Web e baseado no *framework* Pyramid[1] 1.10. Utiliza a plataforma de emulação leve e distribuída MaxiNet versão 1.2 (seção 2.3.2) que é utilizada como base dos *containers* para os roteadores BGP dos ASs; utiliza o roteador BGP baseado em *software* Quagga versão 1.2.4 (seção 2.1.1); e um conjunto de outras aplicações responsáveis pelas funções restantes. Pode ser instalada em um *cluster* de servidores baseados em Linux Ubuntu 18 LTS, Server ou Desktop, e suporta a reprodução de cenários de ataques em topologias de rede que mimetizam a estrutura dos roteadores BGP dos ASs existentes na Internet, com posterior análise dos mecanismos de mitigação dos riscos associados.

As seções seguintes detalham os procedimentos de instalação da MiniSecBGP (seção 6.1) e as funções suportadas pela *testbed* (seção 6.2).

6.1 Instalação e administração da *testbed*

A MiniSecBGP foi planejada para ser um sistema de fácil instalação, retirando do utilizador a necessidade de conhecer os detalhes da implementação e das tecnologias envolvidas em sua concepção. Essa característica mostrou-se ainda mais relevante devido ao fato da MiniSecBGP ser um sistema distribuído, portanto, que deve ser instalado em um *cluster* de servidores.

[1] Pyramid. https://trypyramid.com/

Para realizar a instalação, basta que o utilizador defina qual nó do *cluster* será utilizado como nó *master* da *testbed*, confirme que o SO em uso é o Ubuntu 18 (Server ou Desktop), faça login com um usuário válido no sistema com direitos de *sudo*, baixe o MiniSecBGP diretamente de seu repositório do GitHub[2] e siga o procedimento de instalação descrito no arquivo README do instalador.

Ao iniciar a instalação, todo procedimento de atualização do SO, download e configuração de novos pacotes são realizados automaticamente pelo instalador do sistema. Esse procedimento pode ser acompanhado através das mensagens informativas apresentadas na tela durante o processo de instalação.

A instalação do sistema deve ser realizada apenas no nó *master*, pois toda construção do *cluster* é realizada através das interfaces disponíveis no próprio sistema.

6.2 Funções e organização da *testbed*

As principais funções da *testbed* são: administração dos usuários, gerenciamento do *cluster* de servidores, reprodução das topologias e a execução dos cenários de ataques ao BGP.

6.2.1 Administração dos usuários

Este módulo é responsável pelas ações de criação e exclusão de usuários com permissão de acesso à interface de gerência da MiniSecBGP, através de uma tela de login do sistema. Além disso possibilita alterar a senha de autenticação dos usuários existentes.

Cada usuário do sistema deve ter associado um dos perfis disponíveis na *testbed*: `Admin` ou `Viewer`. Os usuários com perfil `Admin` possuem acesso de administração de todas funções da *testbed*, enquanto os usuários com perfil `Viewer` apenas podem executar os cenários de ataques ao BGP com os dados previamente cadastrados por um usuário do tipo `Admin`.

6.2.2 Gerenciamento do *cluster*

Este módulo de sistema é responsável por incluir novos servidores ao *cluster* da *testbed*. A instalação é feita de forma automatizada e os nós incluídos são denominados *workers*.

Para a instalação ocorrer corretamente, basta o utilizador informar o endereço IP do servidor que deseja incluir ao *cluster*, com o usuário e senha de login no SO com direito de *sudo*. Feito isso, o nó *master* inicia a instalação do sistema no nó *worker*, inicialmente verificando se há acesso de rede entre eles, através de testes com os serviços *ping* e SSH. Caso o acesso via rede esteja funcionando corretamente, é validado o *hostname* do nó *worker*, a fim de garantir que não há outro nó do *cluster* com o mesmo nome de máquina,

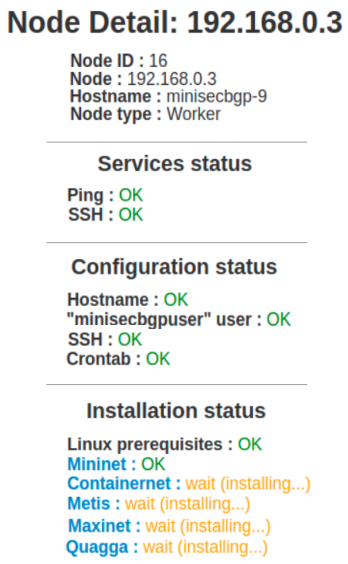
[2] GitHub MiniSecBGP. https://github.com/MiniSecBGP/MiniSecBGP

o que inviabiliza o funcionamento de alguns módulos internos do sistema. Após isso, é criado o usuário `minisecbgpuser` no nó *worker*, configurado o `crontab` no nó *master* para incluir verificações periódicas desses serviços e monitorar a saúde da *testbed* durante seu funcionamento normal. Após esses passos, o processo de instalação passa à fase de atualização do SO do servidor que está sendo incluído ao *cluster*, finalizando com a instalação dos sistemas periféricos, como o MaxiNet e Quagga.

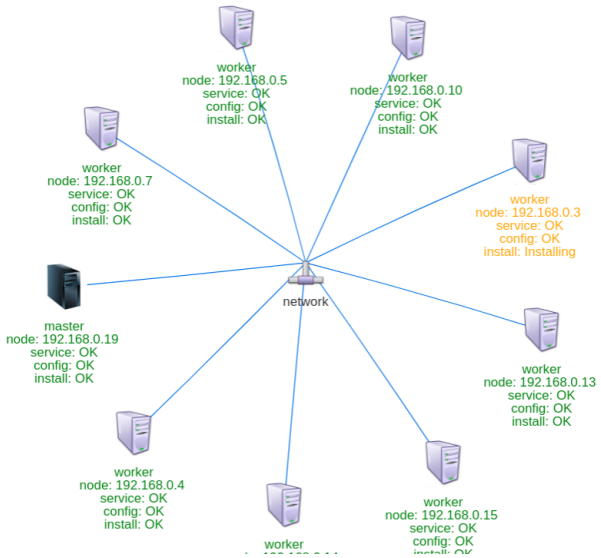
Durante todo esse processo, a MiniSecBGP disponibiliza ao utilizador uma tela de sistema para acompanhar a instalação, Figura 18a. Nela é possível verificar o *status* da operação que está sendo executada naquele momento, onde `OK`, na cor verde, representa que o processo já foi executado com sucesso; e o `wait (installing...)`, na cor laranja, apresenta os processos que ainda estão sendo executados. Caso ocorra algum erro num determinado processo, o log retornado é apresentado na cor vermelha ao lado da identificação do processo que falhou, e toda instalação é então paralisada para possibilitar a intervenção do utilizador do sistema, caso necessário.

Figura 18 – Interfaces para gerência do *cluster* na MiniSecBGP

(a) Detalhe de um nó do *cluster* MiniSecBGP



(b) Visão geral do *cluster* MiniSecBGP



Fonte: Produzido pelos autores

Além de acompanhar o *status* da instalação individualmente por nó do *cluster*, como apresentado na Figura 18a, o utilizado da MiniSecBGP também tem uma visão global da saúde do *cluster* de maneira consolidada, Figura 18b. Nessa visão, o utilizador consegue identificar facilmente a quantidade de nós participantes do *cluster* e sua situação. Caso desejar detalhes sobre um nó específico, basta clicar sobre a imagem do nó correspondente que será redirecionado à tela de detalhes apresentada na Figura 18a.

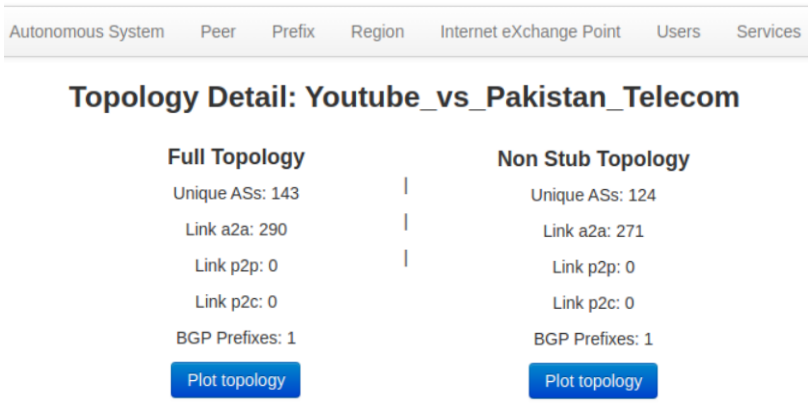
Além de possibilitar a inclusão de nós do *cluster*, o sistema também possibilita o utilizador excluir servidores que já não lhe são necessários. Para isso, basta informar o endereço IP da máquina correspondente que deseja retirar do *cluster*.

6.2.3 Reprodução das topologias

Como detalhado na seção 5.1, a MiniSecBGP suporta a reprodução de topologias realistas da Internet baseada nas bases de dados dos projetos AS-Relationship, do CAIDA, e RIPEstat, da RIPE, além de topologias sintéticas definidas pelo próprio utilizador.

Para isso, o sistema conta com interfaces independentes para criação das topologias baseadas nessas fontes de dados. A MiniSecBGP também conta com interfaces de visualização dos detalhes da topologia, Figura 19, que possibilita o utilizador identificar a quantidade de ASs, relação de vizinhança e lista de prefixos divulgados no BGP, tanto para as topologias normais quanto nas topologias resumidas.

Figura 19 – Interface para gestão das topologias na MiniSecBGP



Fonte: Produzido pelos autores

O sistema também conta com interfaces para gerenciamento das topologias, possibilitando incluir, excluir e modificar os ASs, relação de vizinhança, prefixos divulgados no BGP, tipos de usuários, tipos de negócios dos ASs e relação com IXP. Além disso, o utilizador também pode solicitar a duplicação de topologias existentes ou excluir as que não são mais necessárias.

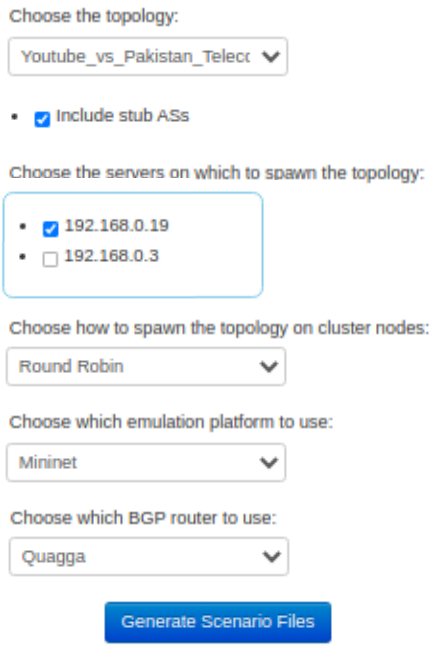
6.2.4 Execução dos cenários de ataque

Basicamente, a execução dos cenários de ataque ao BGP contemplam atividades relacionadas à configuração do cenário, que envolvem a escolha da topologia que será reproduzida, a retirada ou não dos ASs *stub* (seção 5.1.4), a definição do método de distribuição da topologia no *cluster*, a escolha da plataforma de emulação e do roteador BGP utilizados.

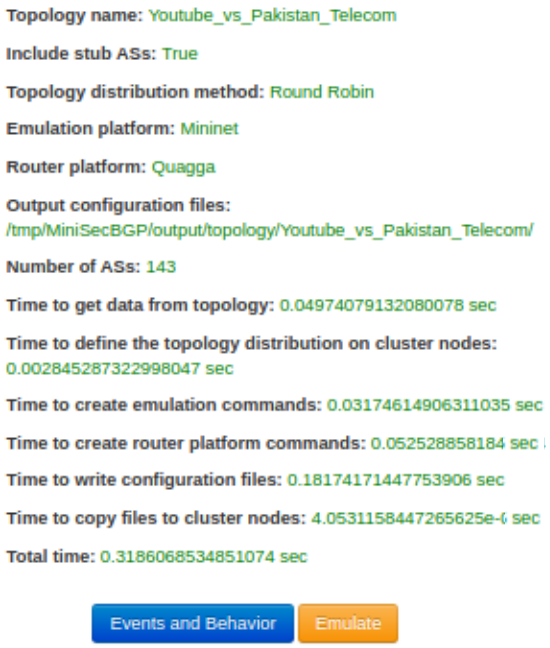
Essas definições são realizadas em uma tela do sistema, conforme apresentado na Figura 20a. Atualmente a MiniSecBGP suporta a opção *Round Robin* como método de distribuição da topologia, Mininet como plataforma de emulação e o Quagga como roteador BGP, porém, o sistema suporta a inclusão de outras tecnologias e métodos que podem contribuir com sua evolução futura.

Figura 20 – Configuração e acompanhamento da criação dos cenários de ataques ao BGP na MiniSecBGP

(a) Tela para configuração do cenário na MiniSecBGP



(b) Relatório contendo *status* e tempo para configuração do cenário na MiniSecBGP



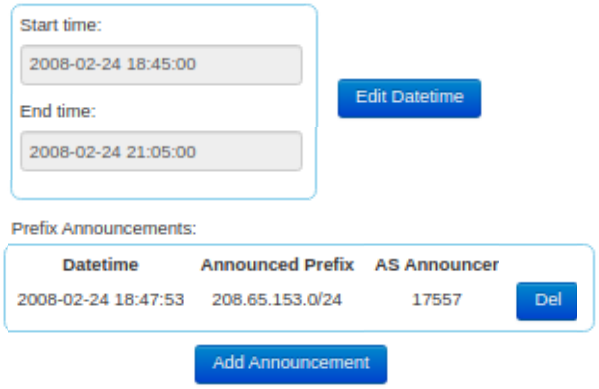
Fonte: Produzido pelos autores

Ao finalizar a configuração do cenário, o utilizador pode acompanhar o *status* da criação dos arquivos de configuração dos *containers* e dos roteadores BGP em cada nó correspondente do *cluster* (seção 5.3.4), além de verificar o tempo de execução de cada tarefa, conforme apresentado na Figura 20b.

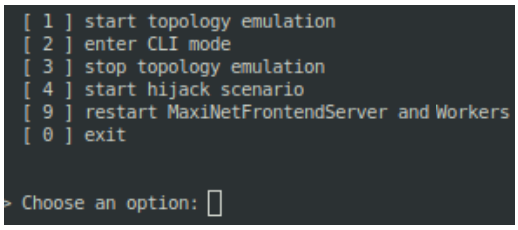
Feito isso, o utilizador pode iniciar a emulação do ambiente, conforme definido, ou ainda especializar o cenário gerenciando os eventos de ataque que serão executados automaticamente durante os testes, conforme apresentado na Figura 21a. Nessa etapa, o utilizador pode alterar a data de início e fim da execução do cenário, modificar os eventos relacionados ao anúncio de prefixos BGP por ASs, incluir *withdraw* de rotas e *prepends* em `AS-PATH` que ocorrerão durante a execução do cenário. Além disso, o utilizado pode definir se o ambiente seguirá o tipo de restrição **permissivo** ou **restritivo** quanto aos anúncios de rotas no BGP (seção 5.2.1).

Figura 21 – Especialização da configuração do cenário de ataque ao BGP e menu de condução da execução do cenário

(a) Tela para especialização de eventos na MiniSecBGP



(b) Menu de condução para execução do cenário de ataque na MiniSecBGP



Fonte: Produzido pelos autores

Na emulação do ambiente, a MiniSecBGP conta com um módulo de sistema que disponibiliza um menu para conduzir o utilizador de maneira automatizada nas etapas de reprodução do cenário de ataque ao BGP, conforme apresentado na Figura 21b. A utilização desse menu visa garantir que todo processo seja executado com possibilidade de erro reduzido.

O menu conta com a opção de iniciar a topologia, criando os elementos Mininet nos nós do *cluster* MaxiNet e iniciando os roteadores BGP; possibilita acesso à console dos roteadores BGP, viabilizando a interação com o ambiente em tempo de execução, conforme descrito no módulo **Gerador de eventos** da arquitetura da *testbed*; permite a chamada para execução do cenário de ataque com os eventos previamente cadastrados no **Agendador de eventos**, além de possibilitar que o *cluster* MaxiNet seja reiniciado, caso necessário recuperar o ambiente ao seu estado inicial.

Capítulo 7

Testes de Validação e Discussão dos Resultados

Este capítulo apresenta os procedimentos para validação da MiniSecBGP. A seção 7.1 descreve o ambiente computacional utilizado para instalação e homologação da *testbed*, enquanto a seção 7.2 apresenta os resultados do teste de reprodução do cenário de um ataque real, quando o Youtube foi vítima de *route leak* originado por Pakistan Telecom no ano de 2008.

7.1 Ambiente de homologação da MiniSecBGP

Para validação das funções propostas na MiniSecBGP, a *testbed* foi instalada em um *cluster* contendo 10 (dez) servidores virtuais Linux Ubuntu Server 18.04. Cada servidor contava com 16 GB de RAM, 6 vCPUs e 160 GB de espaço em disco, com acesso direto à Internet para download e atualização dos sistemas necessários ao MiniSecBGP, bem como para acesso e administração remota das máquinas.

Esses servidores eram instâncias de *Virtual Machine* (VM) criadas em um *cluster* OpenStack Ansible[1], versão 18.1.1, Figura 22a, composto por quatro servidores físicos, conforme apresentado na Tabela 3. Tal cluster foi montado para apoiar esse projeto, facilitando a criação e remoção de recursos, e também como plataforma de ensino e pesquisa genérico.

O *cluster* OpenStack tinha um total de recursos físicos de 96 vCPUs, 1,3 TB de RAM e 3,3 TB de disco. O servidor *Infra0* é responsável pelo armazenamento e gerenciamento de toda infraestrutura de serviços internos necessários ao funcionamento do OpenStack,

[1] OpenStack Ansible. https://github.com/openstack/openstack-ansible

Tabela 3 – *Cluster* OpenStack utilizado para homologação da MiniSecBGP

Nó do *cluster*	Função do nó no *cluster*	Configuração
Infra0	*Infrastructure*	8 vCPUs, 32 GB de RAM, 1,0 TB de disco e 1,0 Gbps de rede
Compute00	*Compute*	24 vCPUs, 284 GB de RAM, 1,8 TB de disco e 4,0 Gbps de rede
Compute01	*Compute*	32 vCPUs, 504 GB de RAM, 273 GB de disco e 4,0 Gbps de rede
Compute02	*Compute*	32 vCPUs, 504 GB de RAM, 273 GB de disco e 4,0 Gbps de rede

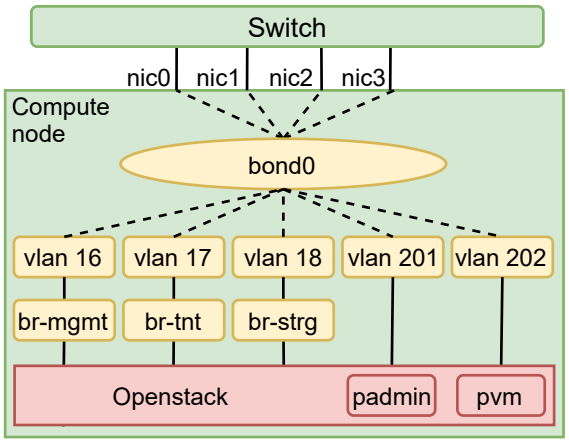
enquanto os outros servidores (*compute00*, *compute01* e *compute02*) compunham o ambiente para criação das instâncias das VMs utilizadas no *cluster* MiniSecBGP.

Os quatro servidores pertencentes ao *cluster* OpenStack estavam interconectados por um *switch* com interfaces de 1 Gbps. O servidor *Infra0* contava apenas com uma interface de rede de 1 Gbps, enquanto os servidores do tipo *compute* possuíam quatro interfaces de rede cada um. Para aumentar a vazão dos servidores *compute*, as quatro interfaces de rede foram conectadas ao *switch* com agregação de tráfego, representada pela interface *bond0* na Figura 22b, totalizando 4 Gbps de banda agregada por servidor físico, e o mesmo valor agregado para as VMs.

Figura 22 – Representação física e lógica da rede nos servidores *compute* do OpenStack

(a) Servidores físicos do *Cluster* Openstack

(b) Representação da rede lógica



Fonte: Produzido pelos autores

Foram configuradas *Virtual Local Area Network* (VLAN) na interface *bond0* dos servidores *compute* para separação lógica do tráfego de rede, com as interfaces *br-mgmt*, *br-tnt* e *br-strg* responsáveis pela comunicação entre os módulos internos do OpenStack, enquanto as interfaces *padmin* e *pvm* eram responsáveis pela comunicação entre as VMs, e com a Internet.

Todo ambiente contava com acesso direto à Internet através de um roteador conectado ao *switch*, utilizado para instalação e atualização dos sistemas.

7.2 Teste de ataque: Youtube vs. Pakistan Telecom

Para validar a capacidade da MiniSecBGP em executar um cenário de ataque real, criado a partir dos eventos observados na Internet, reproduzimos o caso de *route leak* sofrido pelo Youtube em 24 de fevereiro de 2008. Nessa ocasião, Pakistan Telecom e PCCW Global anunciaram o prefixo do Youtube inadvertidamente, tornando-o inacessível a mais de uma centena de provedores asiáticos e europeus por 2 horas. Esse caso teve grande repercussão, ao ponto do próprio RIPE ter disponibilizado um relatório detalhando os eventos ocorridos no ataque.

Segundo RIPE (2008), o Youtube (AS 36561) anunciava seu prefixo (208.65.152.0/22) normalmente, quando, às 18 horas e 47 minutos de 24 de fevereiro de 2008, Pakistan Telecom (AS 17557) anunciou o mesmo prefixo com uma máscara mais restrita (208.65.153.0/24), passando a ter prioridade na escolha do caminho pelo BGP.

Como contramedida, o Youtube também anunciou o prefixo com a mesma máscara mais restrita às 20 horas e 7 minutos (208.65.153.0/24). Essa ação criou dois caminhos para o mesmo prefixo no BGP, fazendo o protocolo utilizar o tamanho do `AS-PATH` na escolha da rota, quando o menor caminho tem prioridade. Dessa forma, parte dos ASs na Internet ainda consideravam que Pakistan Telecom era o anunciante do prefixo.

Outra medida adotada pelo Youtube, às 20 horas e 18 minutos, foi anunciar o prefixo de forma ainda mais restrita, com máscara de 25 bits (208.65.153.0/25 e 208.65.153.128/25). Porém poucos ASs aceitaram esses anúncios por não atenderem às boas práticas de restringir os prefixos no máximo com máscara de 24 bits no BGP.

Às 20 horas e 51 minutos, PCCW Global (AS 3491), que era trânsito de Pakistan Telecom, passou a incluir um *prepend* do AS 17557 nos anúncios originados de Pakistan Telecom, aumentando o `AS-PATH` para esse anunciante. Além disso, às 21 horas e 01 minuto, PCCW Global removeu todos anúncios dos prefixos originados no AS 17557, com um `UPDATE` de *withdraw* que também excluía o anuncio do prefixo sequestrado do Youtube, cessando definitivamente o ataque.

7.2.1 Reprodução do cenário de ataque

Para reproduzir esse cenário na MiniSecBGP, utilizamos os dados da topologia e eventos BGP extraídos do Projeto RIPEstat (seção 5.2.1). Para isso, configuramos os parâmetros de consulta ao repositório do RIPEstat da seguinte forma na tela de criação da topologia no sistema:

a) Prefixos consultados:

- 208.65.152.0/22;
- 208.65.153.0/24;
- 208.65.153.0/25

- 208.65.153.128/25.

b) Período dos eventos analisados:

- iniciando às 18 horas e 45 minutos de 24 de fevereiro de 2008;
- finalizando às 21 horas e 05 minutos de 24 de fevereiro de 2008.

A consulta retornou os dados da topologia e dos eventos ocorridos no BGP, conforme discutido nas seções 5.1.2 e 5.2.1. As Tabelas 4, 5 e 6 trazem o detalhamento das ocorrências agrupadas por tipo de evento.

7.2.1.1 Anúncios de novos prefixos IP (*announcement*)

A Tabela 4 apresenta os eventos ocorridos, que correspondem ao anúncio de prefixos IP no BGP do cenário consultado. As duas colunas da esquerda trazem o horário da ocorrência e a descrição do evento, conforme disponibilizados no relatório da RIPE. Por sua vez, as três colunas da direita trazem os dados do momento da ocorrência, o prefixo IP anunciado e o AS anunciante, conforme extraídos do repositório de dados do Projeto RIPEstat.

Tabela 4 – Relação dos anúncios de prefixos por ASs na topologia (*announcement*)

Relatório RIPE (RIPE, 2008)		JSON RIPEstat		
Horário	Descrição	Horário	Prefixo	Anunciante
estado inicial	Antes, durante e depois de 24 de fevereiro de 2008: AS 36561 (YouTube) anuncia 208.65.152.0/22	estado inicial	208.65.152.0/22	36561
18:47:00	AS 17557 (Pakistan Telecom) começa anunciar 208.65.153.0/24. AS 3491 (PCCW Global) propaga o anúncio. Roteadores em todo mundo recebem o anúncio e o tráfego do YouTube é redirecionado para o Paquistão	18:47:53	208.65.153.0/24	17557
20:07:00	AS 36561 (YouTube) começa anunciar 208.65.153.0/24. Com dois prefixos idênticos no sistema de roteamento, o BGP passa a seguir a política de escolha do caminho mais curto para determinar a rota. Isso significa que o AS 17557 (Pakistan Telecom) continua atraindo parte do tráfego do YouTube	20:07:25	208.65.153.0/24	36561
20:18:00	AS 36561 (YouTube) começa anunciar 208.65.153.0/25 e 208.65.153.128/25. Prefixo mais longo tem prioridade	20:18:43	208.65.153.0/25	36561
		20:18:43	208.65.153.128/25	36561

Ao analisar os dados contidos na Tabela 4, é possível observar que, apesar da relação do prefixo IP anunciado e AS anunciante, reportados em RIPE (2008), condizerem com a informação extraída do repositório RIPEstat, o horário de ocorrência dos eventos apresentam pequena diferença de segundos. Como exemplo, o relatório RIPE informa que o AS 17557 passou a divulgar o prefixo 208.65.153.0/24 às 18 horas e 47 minutos exatos, enquanto a primeira ocorrência desse evento foi observada nos registros do RIPEstat somente às 18 horas, 47 minutos e 53 segundos.

7.2.1.2 Alteração do caminho (*prepend*)

A Tabela 5 apresenta a relação de *prepends* identificados no cenário. As duas colunas da esquerda trazem a relação do horário da ocorrência e a descrição do evento, conforme

disponibilizados no relatório da RIPE. Já as sete colunas da direita trazem os dados extraídos do repositório RIPEstat contendo o momento da ocorrência do evento; o local de aplicação do *prepend* no roteador, se na interface de entrada (*in*) ou saída (*out*); o prefixo IP relacionado ao evento; o AS que anuncia o *prepend*; o ASN que será inserido repetidas vezes no `AS_PATH`; o AS *peer* que receberá os anúncios com *prepend*; e a quantidade de repetições que o ASN sofrerá.

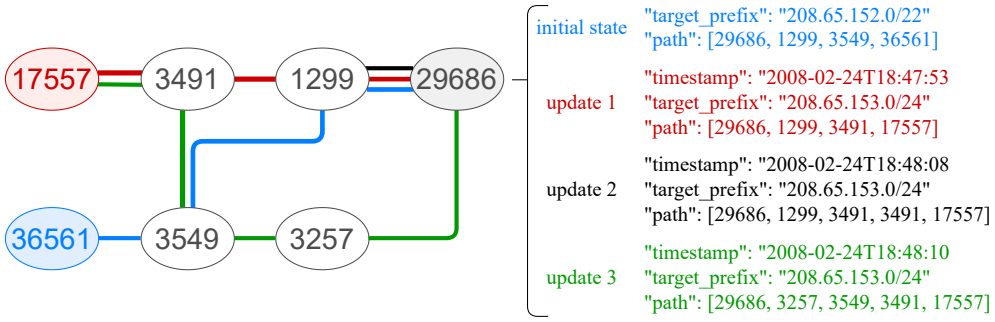
Tabela 5 – Relação dos anúncios contendo *prepend* na topologia

Relatório RIPE (RIPE, 2008)		JSON RIPEstat						
Horário	Descrição	Horário	In/Out	Prefixo	*Prepender*	*Prepended*	*Peer*	Inserções
Eventos que não constam do relatório da RIPE		18:48:08	out	208.65.153.0/24	1299	3491	29686	1
		18:48:09	out	208.65.153.0/24	1299	3491	28917	1
		18:48:09	out	208.65.153.0/24	1299	3491	34225	1
		18:48:09	out	208.65.153.0/24	1299	3491	12676	1
		18:48:13	in	208.65.153.0/24	31323	3549	3549	1
		18:48:27	in	208.65.153.0/24	16186	21318	21318	1
		18:48:35	in	208.65.153.0/24	30890	6453	6453	2
		18:48:51	in	208.65.153.0/25	31323	3549	3549	1
		18:48:51	in	208.65.153.128/25	31323	3549	3549	1
		18:48:57	in	208.65.153.0/25	30890	21318	21318	1
		18:48:57	in	208.65.153.128/25	30890	21318	21318	1
20:51:00	Todos os anúncios de prefixo, incluindo o sequestrado /24 originado pelo AS 17557 (Pakistan Telecom) via AS 3491 (PCCW Global), recebem um prepend do AS 17557. O caminho do AS mais longo significa que mais roteadores preferem o anúncio originado pelo YouTube	20:52:19	out	208.65.153.0/24	3491	17557	29636	1
		20:52:25	out	208.65.153.0/24	3491	17557	286	1
		20:52:25	out	208.65.153.0/24	3491	17557	9009	1
		20:52:26	out	208.65.153.0/24	3491	17557	12989	1
		20:52:26	out	208.65.153.0/24	3491	17557	25232	1
		20:52:26	out	208.65.153.0/24	3491	17557	16243	1
		20:52:27	out	208.65.153.0/24	3491	17557	8419	1
		20:52:28	out	208.65.153.0/24	3491	17557	6320	1
		20:52:29	out	208.65.153.0/24	3491	17557	6067	1
		20:52:33	out	208.65.153.0/24	3491	17557	39202	1
		20:52:34	out	208.65.153.0/24	3491	17557	3741	1
		20:52:36	out	208.65.153.0/24	3491	17557	8447	1
		20:52:36	out	208.65.153.0/24	3491	17557	13237	1
		20:52:36	out	208.65.153.0/24	3491	17557	29686	1
		20:52:36	out	208.65.153.0/24	3491	17557	12676	1
		20:52:37	out	208.65.153.0/24	3491	17557	39196	1
		20:52:37	out	208.65.153.0/24	3491	17557	24963	1
		20:52:37	out	208.65.153.0/24	3491	17557	12859	1
		20:52:38	out	208.65.153.0/24	3491	17557	41692	1
		20:52:42	out	208.65.153.0/24	3491	17557	30844	1
		20:52:42	out	208.65.153.0/24	3491	17557	8881	1
		20:52:49	out	208.65.153.0/24	3491	17557	16034	1
		20:52:49	out	208.65.153.0/24	3491	17557	8468	1
		20:52:49	out	208.65.153.0/24	3491	17557	2857	1
		20:52:57	out	208.65.153.0/24	3491	17557	20640	1
		20:53:06	out	208.65.153.0/24	3491	17557	8763	1
		20:53:20	out	208.65.153.0/24	3491	17557	3327	1
		20:54:22	out	208.65.153.0/24	3491	17557	29208	1
		21:01:21	out	208.65.153.0/24	3491	17557	33970	1

Observando os dados contidos na Tabela 5, chama a atenção os registros das ocorrências identificadas no repositório RIPEstat que não constam no relatório da RIPE. São 11 eventos de *prepend* nessa condição, com o primeiro ocorrendo às 18 horas, 48 minutos e 08 segundos, enquanto o último ocorreu às 18 horas, 48 minutos e 57 segundos. Todos esses eventos foram posteriores ao *route leak* causado pelo Pakistan Telecom e às contramedidas adotadas pelo Youtube, portanto, são relevantes e devem ser consideradas, pois podem alterar o comportamento da rede.

Como exemplo, a Figura 23 detalha as consequências do *prepend* apresentado no primeiro registro na Tabela 5.

Figura 23 – Mudança do AS_PATH causado pela inclusão de *prepend* do ASN 3491 ao anúncio originado no AS 17557



Fonte: Produzido pelos autores

A parte da topologia reproduzida na Figura 23 conta com 7 ASs, incluindo o Youtube (AS 36561) e o Pakistan Telecom (AS 17557). Nela é possível observar que o AS 36561 anunciava o prefixo IP 208.65.152.0/22, que era recebido com o `AS_PATH` [29686, 1299, 3549, 36561] pelo AS 29686 (caminho identificado com a cor azul na Figura 23). A partir das 18 horas, 47 minutos e 53 segundos, o AS 17557 passou a anunciar o prefixo 208.65.153.0/24, recebido com `AS_PATH` [29686, 1299, 3491, 17557] pelo mesmo AS 29686 (`UPDATE` identificado com a cor vermelha). Esse caminho passou a ter prioridade na escolha da rota por ter máscara de sub-rede mais restrita.

Nesse mesmo cenário, às 18 horas, 48 minutos e 08 segundos, o AS 1299 inseriu um *prepend* do ASN 3491 no `AS_PATH` anunciado ao *peer* 29686, que passou a contar com o `AS_PATH` [29686, 1299, 3491, 3491, 17557] para o prefixo 208.65.153.0/24 (`UPDATE` identificado com a cor preta). Como o caminho foi modificado com a inserção do *prepend*, aumentando o `AS_PATH`, às 18 horas, 48 minutos e 10 segundos, o AS 29686 recebeu um novo `UPDATE` para esse prefixo, alterando o `AS_PATH` para [29686, 3257, 3549, 3491, 17557], o que modificou o caminho na rede (`UPDATE` identificado com a cor verde na Figura 23).

Apesar dos eventos observados nesse exemplo específico não terem gerado grande alteração na estrutura da rede, como por exemplo, modificando o AS anunciante do prefixo IP sequestrado, eles foram capazes de mudar o `AS_PATH` correspondente. Tal situação pode ter consequências maiores em topologias que envolvam uma grande quantidade de ASs. Nesse caso, estima-se que o AS 1299 tenha se adiantado em uma mitigação mais intensa baseado em suas políticas internas de roteamento, adicionando o *prepend* ao perceber o sequestro.

Outro ponto relevante observado na Tabela 5, está relacionado ao *prepend* ocorrido às 20 horas e 51 minutos, conforme reportado em RIPE (2008). Nesse evento, o AS

3491 inseriu uma repetição do ASN 17557 no `AS_PATH` anunciado por ele, aumentando o caminho na rede com intuito de mitigar o ataque. Porém, ao observar os registros extraídos do RIPEstat, é possível notar que esse *prepend* foi recebido pelos *peers* do AS 3491 em momentos alternados, com o primeiro `UPDATE` ocorrendo às 20 horas, 52 minutos e 19 segundos, enquanto o último *peer* recebeu o `UPDATE` apenas às 21 horas, 01 minuto e 21 segundos, totalizando 09 minutos e 02 segundos de diferença entre as ocorrências.

7.2.1.3 Exclusão de rotas (*withdraw*)

Para finalizar essa etapa de análise dos eventos de ataque identificados na MiniSecBGP, podemos comparar as ocorrências reportadas em RIPE (2008) aos eventos extraídos automaticamente do repositório do Projeto RIPEstat. Nessa linha, a Tabela 6 traz a relação das solicitações de exclusão de rotas para prefixos IP específicos, observados no cenário.

Tabela 6 – Relação de anúncios removendo prefixo IP em ASs da topologia (*withdraw*)

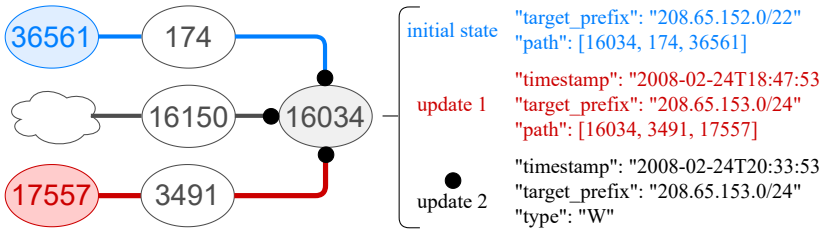
Relatório RIPE (RIPE, 2008)		JSON RIPEstat				
Hora	Descrição	hora	In/Out	Prefixo	*Withdrawer*	*Peer*
Eventos que não constam do relatório da RIPE		20:33:53	in	208.65.153.0/24	16467	2914
		20:33:53	in	208.65.153.0/24	16467	3356
		20:52:18	in	208.65.153.0/24	16034	16150
		20:52:18	in	208.65.153.0/24	16034	174
		20:52:18	in	208.65.153.0/24	16034	3491
21:01:00	O AS 3491 (PCCW Global) remove todos os prefixos originados pelo AS 17557 (Pakistan Telecom), interrompendo o sequestro de 208.65.153.0/24. O AS 17557 não foi completamente desconectado pelo AS 3491. Prefixos originados por outros ASs paquistaneses ainda eram anunciados pelo AS 17557 ao AS 3491	21:00:59	in	208.65.153.0/24	12989	3356
		21:00:59	in	208.65.153.0/24	12989	174
		21:00:59	in	208.65.153.0/24	12989	3491
		21:00:59	in	208.65.153.0/24	12989	36561
		21:00:59	in	208.65.153.0/24	12989	4436
		21:00:59	in	208.65.153.0/24	12989	3549
		21:01:14	in	208.65.153.0/24	20640	174
		21:01:14	in	208.65.153.0/24	20640	3491
		21:01:21	in	208.65.153.0/24	13237	174
		21:01:21	in	208.65.153.0/24	13237	3320
		21:01:21	in	208.65.153.0/24	13237	3491
		21:01:21	in	208.65.153.0/24	13237	25512
		21:01:21	in	208.65.153.0/24	13237	3549
		21:01:22	in	208.65.153.0/24	8763	174
		21:01:22	in	208.65.153.0/24	8763	3491

Assim como ocorrido nos casos de *prepend* (seção 7.2.1.2), nota-se na Tabela 6 a ocorrência de eventos de *withdraw* não reportados em RIPE (2008). Nesse caso, os 5 primeiros registros da tabela apresentam situações em que 2 ASs específicos, o AS 16467 e o AS 16034, receberam mensagens de `UPDATE` contendo *withdraw* do prefixo IP sequestrado.

As consequências do *withdraw* devem ser analisadas com mais atenção, quando comparadas ao *prepend*, pois significa o término do ataque pelo menos aos ASs que o recebeu. Como exemplo, a Figura 24 traz parte da topologia analisada nesse cenário. Ela contém a representação de 6 ASs e está relacionada ao terceiro, quarto e quinto registros contidos na Tabela 6. Esses registros representam o *withdraw* que cessou o *route leak* sofrido pelo Youtube no AS 16034.

Ao analisar a Figura 24, é possível observar que o AS 16034 recebia corretamente o anúncio do prefixo IP 208.65.152.0/22, originado no AS 36561, com o `AS_PATH` [16034,

Figura 24 – Mudança do AS_PATH causado pelo *withdraw* recebido pelo AS 16034 para o prefixo IP sequestrado



Fonte: Produzido pelos autores

174, 36561] (caminho identificado com a cor azul). Quando, às 18 horas, 47 minutos e 53 segundos, o AS 17557 anunciou o prefixo 208.65.153.0/24, sendo recebido pelo AS 16034 com `AS_PATH` [16034, 3491, 17557] (caminho identificado com a cor vermelha). Esse caminho passou a ter prioridade na escolha da rota por ter máscara mais restrita, iniciando o *route leak* sofrido pelo Youtube.

Às 20 horas, 33 minutos e 53 segundos, o AS 16034 recebeu um `UPDATE` contendo o *withdraw* do prefixo sequestrado (208.65.153.0/24), passando a bloquear as novas atualizações recebidas para esse prefixo nas interfaces com seus 3 *peers* no BGP (ASs 174, 16150 e 3491) (filtros nas interfaces identificados com a cor preta). Tal ação foi suficiente para excluir a entrada para o prefixo IP sequestrado na tabela de rotas do roteador do AS 16034, quando ele passou a contar novamente apenas com uma entrada para o prefixo original do Youtube (208.65.152.0/22), que era anunciado pelo AS 36561. Nesse caso, o *withdraw* cessou o *route leak* para esse AS específico.

7.2.2 Fidelidade do cenário reproduzido ao caso real

Para validar a capacidade da MiniSecBGP em reproduzir com fidelidade o comportamento do BGP nesse caso de ataque real, executamos o cenário de ataque nos modos **permissivo** e **restritivo**, baseado nos eventos criados pelo **Agendador de eventos** durante o processo de importação da topologia. Os resultados foram comparados aos registros existentes na base de dados do repositório RIPEstat, confrontando as seguintes informações:

a) AS origem do anúncio de prefixo IP aprendido;

b) `AS_PATH` para acessar o prefixo IP;

c) Momento da ocorrência do evento.

Na MiniSecBGP, as medições dos parâmetros foram realizadas automaticamente durante a execução do cenário de ataque, em intervalos de tempo pré-definidos no sistema.

Como a base de dados do Projeto RIPEstat conta apenas com as informações históricas dos eventos observados pelos coletores da RIPE, optamos por utilizar somente os dados coletados nos roteadores BGP do ambiente reproduzido que correspondem a esses ASs. Por esse motivo, apesar da topologia contar com um total de 143 roteadores, foram utilizados dados de apenas 88 deles para validação.

7.2.2.1 Reprodução do cenário no modo permissivo

Como apresentado na seção 5.2.1.4, no modo **permissivo** os roteadores recebem apenas as configurações dos eventos de *announcement*, *prepend* e *withdraw* previamente agendados na MiniSecBGP, não incluindo qualquer filtro que interfira nas métricas do protocolo para escolha das rotas.

A Tabela 7 apresenta os dados resultantes da execução desse cenário. Nela é possível observar a relação dos ASs aprendidos como anunciantes dos prefixos IP, confrontando os valores obtidos na MiniSecBGP aos esperados em RIPEstat. As linhas apresentam os valores de referência, enquanto as colunas trazem os resultados observados na execução do cenário. Cada registro apresenta a quantidade de ASs da topologia que aceitaram os anúncios originados nos ASs 36561, 17557 ou que não aceitaram o anúncio (NA). Os dados estão agrupados por prefixo IP anunciado (208.65.152.0/22, 208.65.153.0/24, 208.65.153.0/25 e 208.65.153.128/25) e evento de monitoramento (t0 à t9).

Um dado relevante que pode ser observado na Tabela 7, é a quantidade de anúncios que deveriam ter sido originados em um determinado AS (RIPEstat) mas foram anunciados por outro (MiniSecBGP). Essa relação também informa quais são os ASs envolvidos nesses casos.

Como exemplo, observando os dados contidos na primeira linha da célula destacada em vermelho na tabela, era previsto que 3 ASs não recebessem anúncio do prefixo IP 208.65.153.0/24 no momento t6 de monitoramento. Porém, ao analisar os dados da execução do cenário na MiniSecBGP, constatou-se que esses 3 ASs receberam anúncios para o prefixo correspondente, originados pelo AS 36561 (2 anúncios aceitos) e o AS 17557 (1 anúncio aceito). Essa mesma situação ocorreu com anúncios dos ASs 36561 e 17557, onde 4 anúncios que deveriam ser originados no AS 36561, foram divulgados pelo AS 17557; e 15 anúncios que deveriam ser originados no AS 17557, foram divulgados pelo AS 36561.

Essa situação ocorre justamente devido à não utilização de filtros para reproduzir as políticas de roteamento dos ASs no modo **permissivo** na MiniSecBGP. Nesse caso, o BGP considera apenas suas métricas básicas na escolha dos caminhos na rede, causando distorções nos dados observados no ambiente de reprodução frente aos eventos reais.

Esse fato não pode ser desconsiderado na reprodução de um ataque real ao BGP, pois todo comportamento do ambiente pode ser alterado e os resultados podem não refletir a realidade. Como exemplo, a Figura 25 apresenta a relação entre os erros nos anúncios dos prefixos IP, onde os ASs da topologia reproduzida aceitaram anúncios originados em

Tabela 7 – Relação dos ASs aprendidos como anunciantes de prefixos IP por episódio de monitoramento (RIPEstat *versus* MiniSecBGP (modo **permissivo**))

		MiniSecBGP												
	AS	NA	36561	17557	NA	36561	17557	NA	36561	17557	NA	36561	17557	
RIPEstat	NA	**0**	1	0	**88**	0	0	**88**	0	0	**88**	0	0	t0
	36561	0	**87**	0	0	**0**	0	0	**0**	0	0	**0**	0	
	17557	0	0	**0**	0	0	**0**	0	0	**0**	0	0	**0**	
	NA	**0**	1	0	**0**	0	4	**88**	0	0	**88**	0	0	t1
	36561	0	**87**	0	0	**0**	0	0	**0**	0	0	**0**	0	
	17557	0	0	**0**	0	0	**84**	0	0	**0**	0	0	**0**	
	NA	**0**	1	0	**0**	0	4	**88**	0	0	**88**	0	0	t2
	36561	0	**87**	0	0	**0**	0	0	**0**	0	0	**0**	0	
	17557	0	0	**0**	0	0	**84**	0	0	**0**	0	0	**0**	
	NA	**0**	1	0	**0**	0	4	**88**	0	0	**88**	0	0	t3
	36561	0	**87**	0	0	**0**	0	0	**0**	0	0	**0**	0	
	17557	0	0	**0**	0	0	**84**	0	0	**0**	0	0	**0**	
	NA	**0**	1	0	**0**	0	4	**88**	0	0	**88**	0	0	t4
	36561	0	**87**	0	0	**0**	0	0	**0**	0	0	**0**	0	
	17557	0	0	**0**	0	0	**84**	0	0	**0**	0	0	**0**	
	NA	**0**	1	0	**0**	0	4	**88**	0	0	**88**	0	0	t5
	36561	0	**87**	0	0	**0**	0	0	**0**	0	0	**0**	0	
	17557	0	0	**0**	0	0	**84**	0	0	**0**	0	0	**0**	
	NA	**0**	1	0	**0**	2	1	**88**	0	0	**88**	0	0	t6
	36561	0	**87**	0	0	**34**	4	0	**0**	0	0	**0**	0	
	17557	0	0	**0**	0	15	**32**	0	0	**0**	0	0	**0**	
	NA	**0**	1	0	**0**	2	1	**0**	73	0	**0**	73	0	t7
	36561	0	**87**	0	0	**41**	5	0	**15**	0	0	**15**	0	
	17557	0	0	**0**	0	8	**31**	0	0	**0**	0	0	**0**	
	NA	**0**	1	0	**1**	2	1	**0**	72	0	**0**	72	0	t8
	36561	0	**87**	0	0	**41**	5	0	**16**	0	0	**16**	0	
	17557	0	0	**0**	0	7	**31**	0	0	**0**	0	0	**0**	
	NA	**0**	1	0	**6**	2	1	**0**	72	0	**0**	72	0	t9
	36561	0	**87**	0	0	**47**	32	0	**16**	0	0	**16**	0	
	17557	0	0	**0**	0	0	**0**	0	0	**0**	0	0	**0**	
		208.65.152.0/22			208.65.153.0/24			208.65.153.0/25			208.65.153.128/25			

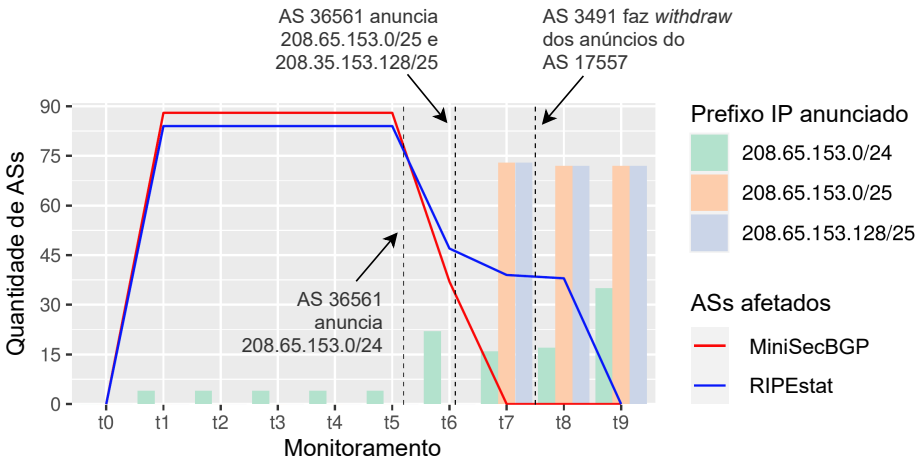
ASs diferentes do que era previsto em RIPEstat. Tal situação teve como consequência a mitigação do ataque de *route leak* em momento errado e com diferente causa no ambiente reproduzido, o que pode induzir o utilizador da *testbed* a confiar em mecanismos ineficientes de mitigação do ataque.

Na Figura 25, as barras indicam a quantidade de ASs da topologia reproduzida que aceitaram anúncios de prefixo IP originados em ASs diferentes do informado em RIPEstat. As linhas contínuas indicam a quantidade de ASs que eram afetados pelo *route leak*, observados nos períodos de monitoramento do ambiente, enquanto as linhas verticais tracejadas apresentam as ocorrências dos eventos relevantes para as mudanças no BGP.

Durante todo cenário, é possível observar que há diferença na quantidade de ASs afetados pelo *route leak* nos resultados da topologia reproduzida frente ao caso real (linhas contínuas azul e vermelha). Isso ocorre porque há erro na relação de ASs anunciantes de prefixos IP em todo ponto de monitoramento (barras verticais). Essa situação é detalhada na Tabela 7.

Nesse ponto, é importante observar o comportamento do *route leak* quando o AS 36561 divulga os prefixos do Youtube em sub-redes com máscara de 25 bits (t6). Como

Figura 25 – Relação da quantidade de ASs afetados com o *route leak* e os ASs anunciantes de prefixos IP aprendidos incorretamente (RIPEstat *versus* MiniSecBGP (modo **permissivo**))



Fonte: Produzido pelos autores

a *testbed* não implementa filtros no modo **permissivo**, todos ASs da topologia recebem esses anúncios, convergindo a rota para esse anunciante devido a máscara ser mais restrita, encerrando o ataque no t7 (linha vermelha). Porém, como no caso real apenas 15 ASs aceitaram esse anúncio, devido as políticas de roteamento de cada AS, os efeitos do ataque ainda foram sentidos até o monitoramento t8, quando o AS 3491 realizou *withdraw* do prefixo sequestrado.

Se por um lado o modo **permissivo** pode levar a MiniSecBGP a ter comportamento diferente dos casos reais reproduzidos, ela garante ao utilizador a possibilidade de testar mecanismos de mitigação do ataque de forma mais autônoma. Como a topologia reproduzida não implementa qualquer tipo de filtro nos roteadores, todos ASs estarão sujeitos apenas às condições de funcionamento e métricas do próprio BGP. Nesse caso, o utilizador da *testbed* tem a certeza de que os métodos identificados por ele para uma possível mitigação de ataque, independem das configurações de filtros de outros ASs.

7.2.2.2 Reprodução do cenário no modo restritivo

Diferentemente do modo de funcionamento da *testbed* apresentado na seção anterior, o modo **restritivo** implementa filtros em todos roteadores BGP, liberando apenas o tráfego correspondente aos eventos observados no RIPEstat.

A Tabela 8 apresenta a relação dos ASs anunciantes dos prefixos IP resultantes da reprodução desse cenário na MiniSecBGP. Nela é possível observar que houve alguma inconsistência nos eventos em apenas três momentos específicos do monitoramento (células destacadas em vermelho). Todos esses casos estão relacionados à divulgação do prefixo IP

208.65.153.0/24, onde 7 ASs receberam atualizações incorretas originadas no AS 17557 no momento de monitoramento t6, 8 atualizações incorretas no momento t7 e 7 atualizações incorretas no momento t8.

Tabela 8 – Relação dos ASs aprendidos como anunciantes de prefixos IP por episódio de monitoramento (RIPEstat *versus* MiniSecBGP (modo **restritivo**))

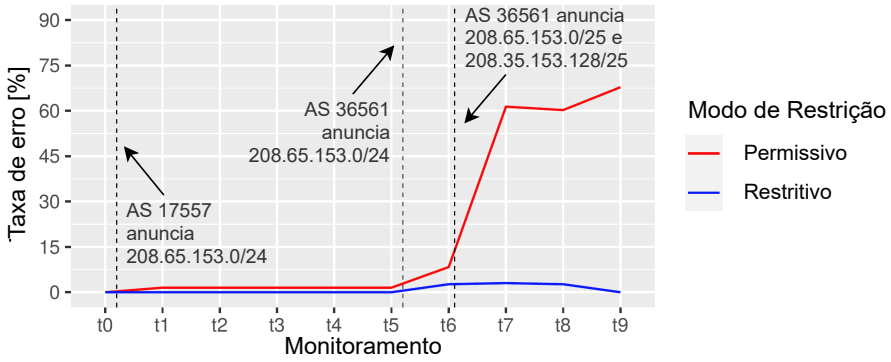
		MiniSecBGP												
	AS	NA	36561	17557	NA	36561	17557	NA	36561	17557	NA	36561	17557	
RIPEstat	NA	**1**	0	0	**88**	0	0	**88**	0	0	**88**	0	0	t0
	36561	0	**87**	0	0	**0**	0	0	**0**	0	0	**0**	0	
	17557	0	0	**0**	0	0	**0**	0	0	**0**	0	0	**0**	
	NA	**1**	0	0	**4**	0	0	**88**	0	0	**88**	0	0	t1
	36561	0	**87**	0	0	**0**	0	0	**0**	0	0	**0**	0	
	17557	0	0	**0**	0	0	**84**	0	0	**0**	0	0	**0**	
	NA	**1**	0	0	**4**	0	0	**88**	0	0	**88**	0	0	t2
	36561	0	**87**	0	0	**0**	0	0	**0**	0	0	**0**	0	
	17557	0	0	**0**	0	0	**84**	0	0	**0**	0	0	**0**	
	NA	**1**	0	0	**4**	0	0	**88**	0	0	**88**	0	0	t3
	36561	0	**87**	0	0	**0**	0	0	**0**	0	0	**0**	0	
	17557	0	0	**0**	0	0	**84**	0	0	**0**	0	0	**0**	
	NA	**1**	0	0	**4**	0	0	**88**	0	0	**88**	0	0	t4
	36561	0	**87**	0	0	**0**	0	0	**0**	0	0	**0**	0	
	17557	0	0	**0**	0	0	**84**	0	0	**0**	0	0	**0**	
	NA	**1**	0	0	**4**	0	0	**88**	0	0	**88**	0	0	t5
	36561	0	**87**	0	0	**0**	0	0	**0**	0	0	**0**	0	
	17557	0	0	**0**	0	0	**84**	0	0	**0**	0	0	**0**	
	NA	**1**	0	0	**3**	0	0	**88**	0	0	**88**	0	0	t6
	36561	0	**87**	0	0	**31**	7	0	**0**	0	0	**0**	0	
	17557	0	0	**0**	0	0	**47**	0	0	**0**	0	0	**0**	
	NA	**1**	0	0	**3**	0	0	**73**	0	0	**73**	0	0	t7
	36561	0	**87**	0	0	**38**	8	0	**15**	0	0	**15**	0	
	17557	0	0	**0**	0	0	**39**	0	0	**0**	0	0	**0**	
	NA	**1**	0	0	**4**	0	0	**72**	0	0	**72**	0	0	t8
	36561	0	**87**	0	0	**39**	7	0	**16**	0	0	**16**	0	
	17557	0	0	**0**	0	0	**38**	0	0	**0**	0	0	**0**	
	NA	**1**	0	0	**9**	0	0	**72**	0	0	**72**	0	0	t9
	36561	0	**87**	0	0	**79**	0	0	**16**	0	0	**16**	0	
	17557	0	0	**0**	0	0	**0**	0	0	**0**	0	0	**0**	
		208.65.152.0/22			208.65.153.0/24			208.65.153.0/25			208.65.153.128/25			

Uma boa análise para identificar a qualidade da reprodução do cenário de ataque, é comparar os resultados obtidos com a reprodução realizada anteriormente. Nessa linha, ao confrontar a taxa de erro dos ASs aprendidos como anunciantes dos prefixos IP 208.65.152.0/22, 208.65.153.0/24, 208.65.153.0/25, 208.65.153.128/25, resultante das reproduções nos modos **restrito** e **permissivo** (Figura 26), nota-se que em todas medições o modo **restritivo** teve erro inferior ao **permissivo**, fato que ficou mais evidente a partir do monitoramento t6, quando o AS 36561 anunciou os prefixos com máscara mais restrita de 25 bits.

Além dessa análise geral, é importante observar também a qualidade da reprodução do ambiente a partir das seguintes verificações:

a) Quantidade de ASs que receberam anúncios dos AS origem e `AS_PATH` corretos para cada prefixo IP divulgado. Essa situação representa a reprodução exata do ambiente;

Figura 26 – Taxa de erro dos ASs aprendidos como anunciantes de prefixos IP (MiniSecBGP (modo **permissivo** *versus* **restritivo**))



Fonte: Produzido pelos autores

b) Quantidade de ASs que receberam anúncios com o AS origem correto, mas com `AS_PATH` incorreto. Apesar dessa situação não gerar isolamento do AS afetado, ela pode representar uma forma de interceptação de tráfego;

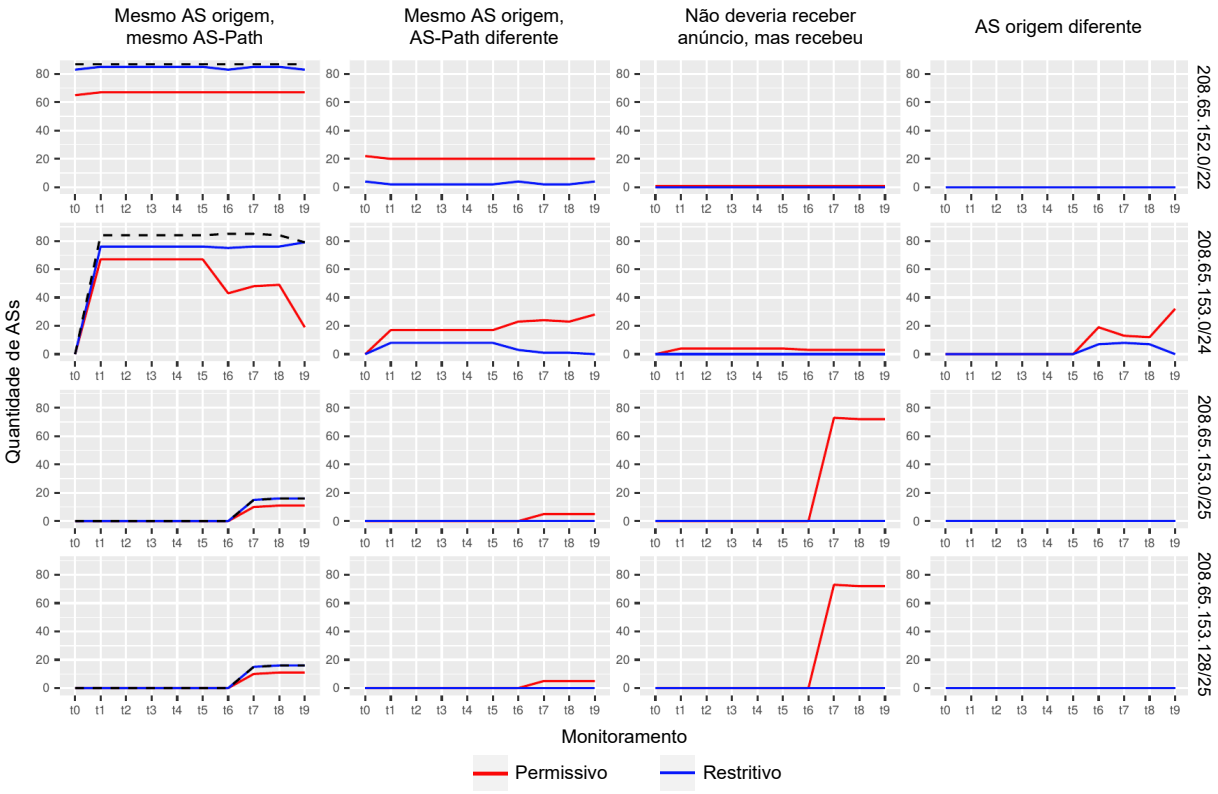
c) ASs que não deveriam ter recebido o anúncio de um prefixo IP, mas receberam.

Nessa linha, a Figura 27 apresenta a relação dos ASs da topologia que aprenderam anúncios com o AS origem e `AS_PATH` corretos; AS origem correto e `AS_PATH` incorreto; ASs que não deveriam ter recebido anúncio do prefixo IP, mas receberam; e os ASs que receberam anúncio com origem incorreta. Os dados estão agrupados por prefixo IP divulgado e correspondem aos resultados obtidos nas reproduções da MiniSecBGP, nos modos **permissivo** e **restritivo**, utilizando como base os dados do RIPEstat.

Quanto aos anúncios com AS origem e `AS_PATH` corretos (primeira coluna mais à esquerda da Figura 27), sua precisão é apontada pela proximidade das linhas que representam os resultados da *testbed* (linhas azul e vermelha) com a linha de referência do RIPEstat (linha tracejada preta). Nesse caso, observa-se que o modo **restritivo** teve maior precisão no anúncio de todos prefixos IP, chegando à 100% de precisão para os anúncios dos prefixos com máscara de 25 bits.

Para os outros parâmetros monitorados, a referência de precisão está na proximidade com o valor 0 no eixo Y dos gráficos, indicando que não foram identificadas ocorrências naquelas situações. Nesse caso, é possível observar que o modo **restritivo** da MiniSecBGP também obteve resultados mais precisos quando comparados ao **permissivo**, pois apresenta menos ocorrências de casos onde os AS origem está correto, mas o `AS_PATH` está incorreto; há nulidade quanto aos ASs que não deveriam ter recebido anúncios de prefixos IP, mas receberam; e a taxa dos ASs origem aprendidos incorretamente também é menor, chegando a ser nula para a maioria dos prefixos IP.

Figura 27 – Relação de ASs da topologia que aprenderam anúncios de origem e AS_PATH corretos; de AS origem correto, mas com AS_PATH incorreto; que não deveriam ter recebido o anúncio do prefixo IP, mas receberam; e dos que receberam anúncio de AS origem incorreto - por prefixo IP anunciado (MiniSecBGP (modos **permissivo** e **restritivo**))



Fonte: Produzido pelos autores

Por fim, para aprofundar a análise dos resultados da reprodução do cenário de ataque no modo **restritivo** da MiniSecBGP, a Tabela 9 apresenta a relação dos ASs que aceitaram anúncios com origem incorreta, apontando as causas da mudança do anunciante em cada ocorrência.

Ao analisar os dados da Tabela 9, fica claro que todas 22 ocorrências de origens incorretas ficaram restritas a apenas 8 ASs. Além disso, todos casos foram motivados pela existência de dois `AS_PATH` com o mesmo tamanho para as duas origens. Nesse caso, a MiniSecBGP contém regras de filtro que permitem receber anúncio das duas origens, mas no caso real, as políticas internas dos próprios ASs afetados, de algum AS pertencente ao caminho ou as próprias métricas do BGP, podem levar a escolha de um ou outro caminho na rede.

Uma alternativa para esses casos é o utilizador da *testbed* intervir manualmente no cenário durante sua execução, através do módulo **Gerador de eventos**, inserindo regras com o auxílio de *ip prefix list*, *ip as-path access-list* e *route-map* para direcionar o tráfego.

Tabela 9 – Relação dos ASs da topologia reproduzida que aprenderam o prefixo IP 208.65.153/24 da origem errada (RIPEstat *versus* MiniSecBGP (modo **restritivo**))

Monitoramento	AS afetado	Causa
t6, t7 e t8	2497	AS-Path com tamanhos iguais para os dois anunciantes 36561: [3549, 36561] e 17557: [3491, 17557]
t6, t7 e t8	6461	AS-Path com tamanhos iguais para os dois anunciantes 36561: [3549, 36561] e 17557: [3491, 17557]
t7 e t8	6667	AS-Path com tamanhos iguais para os dois anunciantes 36561: [3549, 36561] e 17557: [3491, 17557]
t6, t7 e t8	6762	AS-Path com tamanhos iguais para os dois anunciantes 36561: [3549, 36561] e 17557: [3491, 17557]
t6 e t7	16467	AS-Path com tamanhos iguais para os dois anunciantes 36561: [3356, 3549, 36561] e 17557: [2914, 3491, 17557]
t6, t7 e t8	22548	AS-Path com tamanhos iguais para os dois anunciantes 36561: [10429, 12956, 3549, 36561] e 17557: [10429, 12956, 3491, 17557]
t6, t7 e t8	34948	AS-Path com tamanhos iguais para os dois anunciantes 36561: [6453, 3549, 36561] e 17557: [174, 3491, 17557]
t6, t7 e t8	39792	AS-Path com tamanhos iguais para os dois anunciantes 36561: [8359, 3549, 36561] e 17557: [174, 3491, 17557]

Além dos resultados positivos do modo **restritivo** em reproduzir com fidelidade os cenários de ataques reais ao BGP, ele traz consigo a capacidade da MiniSecBGP de reproduzir as políticas internas de roteamento de cada AS da topologia. Esse fato é relevante e deve ser considerado, pois, apesar de não serem informações públicas, essas políticas interferem diretamente no funcionamento de toda Internet.

Capítulo 8

Conclusão

Os ataques ao BGP ainda são uma grande ameaça à integridade e operação da Internet, causando enormes danos. Para enfrentar esse desafio, é necessário o desenvolvimento de métodos e ferramentas de ponta que suportem uma avaliação de risco mais precisa e uma avaliação mais realista das estratégias de mitigação de ataques ao BGP. Para tanto, alguns requisitos devem ser atendidos, como replicar a estrutura dos roteadores dos ASs de forma realista no que diz respeito à sua topologia, a relação de *peering* e as políticas de filtro implementadas entre eles e a reprodução do comportamento do protocolo de roteamento implantado nos roteadores reais.

Em resposta a essas necessidades, este trabalho propôs a MiniSecBGP, uma metodologia e ferramenta para avaliação de risco e estudo de estratégias de mitigação para ataques ao BGP. A MiniSecBGP usa informações sobre ASs, com suas relações de *peering* no BGP, extraídas de bases públicas de dados de rotas, e tecnologias de emulação leves e distribuídas para reproduzir a estrutura de ASs reais da Internet. Conforme demonstrado em nossos experimentos, a MiniSecBGP é capaz de reproduzir fielmente os cenários de ataque com base em dados de eventos reais, permitindo a interação do usuário com o ambiente em tempo de execução para avaliar o risco associado e as estratégias de mitigação. A *testbed* também suporta a modificação dos parâmetros do BGP para especificar e avaliar cenários híbridos, ou seja, com a mistura de dados reais e sintéticos.

8.1 Trabalhos Futuros

Durante o desenvolvimento dessa tese, encontramos aspectos interessantes que poderiam melhorar ainda mais a eficiência da MiniSecBGP. Por mais que tentemos cobrir uma

ampla gama de variáveis, há melhorias que podem ser feitas no ambiente. Destacamos algumas dessas ideias para trabalhos futuros abaixo:

a) Métodos de distribuição da topologia no ambiente distribuído:

   Atualmente a MiniSecBGP conta com a técnica de *Round Robin* para particionamento da topologia entre os nós do *cluster*. Apesar de funcional, este método pode sobrecarregar o *cluster* com o aumento significativo de elementos Mininet criados no ambiente. Como o MaxiNet não suporta conexão direta entre elementos Mininet do tipo **host** em servidores diferentes do *cluster*, são necessários a criação de mais 2 novos elementos do tipo *switch* na topologia a cada conexão de elementos remotos. Nesse caso, é mais indicado a distribuição considerando o número de arestas de cada AS do grafo da topologia no cluster;

b) Integração do Containernet[1]:

   A integração do Containernet na MiniSecBGP representa agregar mais funcionalidades à *testbed*. O Containernet suporta utilização de *containers* Mininet e Docker simultaneamente. Nesse caso, os *containers* Docker podem ser utilizados para inserir funcionalidades de servidores de rede à topologia reproduzida, estendendo a análise do ambiente também às aplicações suportadas na rede;

c) Incluir outras soluções de roteadores BGP:

   Apesar do Quagga ser um roteador amplamente utilizado em ambientes de produção, outras soluções também vêm se destacando, como por exemplo o Bird[2], mas não se limitando a ele. A inclusão dessas soluções possibilita ao usuário reproduzir o ambiente analisado com maior fidelidade tecnológica ao caso real observado;

d) Interface com equipamentos externos:

   Apesar da MiniSecBGP ter como requisito ser um ambiente controlado e funcionar de forma independente aos equipamentos que fazem parte da estrutura da Internet, é importante a possibilidade de conectar os elementos reproduzidos na *testbed* com outros equipamentos externos. Esse tipo de situação mostra-se relevante em casos onde o utilizador deseja analisar o comportamento de um roteador BGP específico, que não é suportado pela MiniSecBGP, por exemplo, em um caso de ataque ao BGP. Nesse caso, é possível agregar esse roteador ao ambiente reproduzido através das interfaces com equipamentos externos;

[1] Containernet. https://containernet.github.io/
[2] Bird Internet Routing Daemon. https://bird.network.cz/

Referências

AKAMAI. **State of the Internet Security - A Year in Review**. [S.l.], 2018. Disponível em: <https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/2018-state-of-the-internet-security-a-year-in-review.pdf>.

AL-MUSAWI, B.; BRANCH, P.; ARMITAGE, G. BGP Anomaly Detection Techniques: A Survey. **IEEE Communications Surveys Tutorials**, v. 19, n. 1, p. 377–396, Firstquarter 2017. ISSN 1553-877X.

_____. Rapid Detection of BGP Anomalies. **44 APNIC Conference, Taichung, Taiwan, 7 - 14 September 2017**, 2017.

ARBOR. **NETSCOUT Arbor's 14th Annual Worldwide Infrastructure Security Report**. [S.l.], 2019. Disponível em: <https://www.netscout.com/report>.

BALLANI, H.; FRANCIS, P.; ZHANG, X. A Study of Prefix Hijacking and Interception in the Internet. In: **Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications**. New York, NY, USA: ACM, 2007. (SIGCOMM '07), p. 265–276. ISBN 978-1-59593-713-1. Disponível em: <http://doi.acm.org/10.1145/1282380.1282411>.

BAREA, E. R. A. et al. Evaluation of lightweight and distributed emulation solutions for network experimentation. In: LATIFI, S. (Ed.). **17th International Conference on Information Technology–New Generations (ITNG 2020)**. Cham: Springer International Publishing, 2020. p. 585–592. ISBN 978-3-030-43020-7.

_____. Avaliação de Soluções de Emulação Leve e Distribuída para Experimentação de Rede. In: **Anais do XVIII Workshop em Desempenho de Sistemas Computacionais e de Comunicação**. Porto Alegre, RS, Brasil: SBC, 2019. ISSN 2595-6167. Disponível em: <https://portaldeconteudo.sbc.org.br/index.php/wperformance/article/view/6461>.

_____. MiniSecBGP: Testbed de Emulação Leve em Segurança BGP. In: **Anais do Workshop de Segurança Cibernética em Dispositivos Conectados (WSCDC - SBRC 2018)**. Porto Alegre, RS, Brasil: SBC, 2018. v. 1. Disponível em: <https://portaldeconteudo.sbc.org.br/index.php/wscdc/article/view/2398>.

BELLOVIN, S. et al. Slowing Routing Table Growth by Filtering Based on Address Allocation Policies. 08 2001.

BRITO, S. H. B. et al. Anatomia do Ecossistema de Pontos de Troca de Tráfego Públicos na Internet do Brasil. In: **Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, SBRC 2015, Vitória, Brasil, maio 18-22, 2015**. [S.l.: s.n.], 2015. p. 67–80.

BROWN, N. **Control groups series**. [S.l.], 2014. Disponível em: <https://lwn.net/Articles/604609/>.

BURKARD, C. **Cluster Edition Prototype**. [S.l.], 2014. Disponível em: <https://github.com/mininet/mininet/wiki/Cluster-Edition-Prototype>.

BUTLER, K. et al. A Survey of BGP Security Issues and Solutions. **Proceedings of the IEEE**, v. 98, n. 1, p. 100–122, Jan 2010.

CALVERT, K. L.; DOAR, M. B.; ZEGURA, E. W. Modeling Internet topology. **IEEE Communications Magazine**, v. 35, n. 6, p. 160–163, June 1997. ISSN 0163-6804.

CANONICAL. **Linux Containers: What's LXC?** 2018? Disponível em: <https://linuxcontainers.org/lxc/>.

_____. **Linux Containers: What's LXD?** 2018? Disponível em: <https://linuxcontainers.org/lxd/>.

CHANDRA, R.; TRAINA, P. **BGP Communities Attribute**. 1996. RFC 1997 (Proposed Standard), Internet Engineering Task Force, updated by RFCs 7606, 8642. [Online]. Disponível em: http://www.ietf.org/rfc/rfc1997.txt.

CHANG, D.-F.; GOVINDAN, R.; HEIDEMANN, J. An empirical study of router response to large bgp routing table load. In: **Proceedings of the 2Nd ACM SIGCOMM Workshop on Internet Measurment**. New York, NY, USA: ACM, 2002. (IMW '02), p. 203–208. ISBN 1-58113-603-X. Disponível em: <http://doi.acm.org/10.1145/637201.637233>.

CHO, S. et al. BGP hijacking classification. In: **2019 Network Traffic Measurement and Analysis Conference (TMA)**. [S.l.: s.n.], 2019. p. 25–32.

CISCO. **Achieve Optimal Routing and Reduce BGP Memory Consumption**. [S.l.], 2008. Disponível em: <https://www.cisco.com/c/en/us/support/docs/ip/border-gateway-protocol-bgp/12512-41.html>.

CLAFFY, K. Border Gateway Protocol (BGP) and Traceroute Data Workshop Report. **SIGCOMM Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 42, n. 3, p. 28–31, jun. 2012. ISSN 0146-4833. Disponível em: <http://doi.acm.org/10.1145/2317307.2317313>.

COHEN, A. et al. Jumpstarting BGP Security with Path-End Validation. In: **Proceedings of the 2016 ACM SIGCOMM Conference**. New York, NY, USA: ACM, 2016. (SIGCOMM '16), p. 342–355. ISBN 978-1-4503-4193-6. Disponível em: <http://doi.acm.org/10.1145/2934872.2934883>.

DANIELS, J. Server Virtualization Architecture and Implementation. **XRDS**, ACM, New York, NY, USA, v. 16, n. 1, p. 8–12, set. 2009. ISSN 1528-4972.

DIMITROPOULOS, X. et al. AS Relationships: Inference and Validation. **SIGCOMM Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 37, n. 1, p. 29–40, jan 2007. ISSN 0146-4833. Disponível em: <http://doi.acm.org/10.1145/1198255.1198259>.

DOCKER. **Welcome to the Docker Cloud docs!** 2018. Disponível em: <https://docs.docker.com/docker-cloud/>.

EDDY, W. **TCP SYN Flooding Attacks and Common Mitigations**. 2007. RFC 4987 (Informational), Internet Engineering Task Force. [Online]. Disponível em: http://www.ietf.org/rfc/rfc4987.txt.

EKPARINYA, P.; GRAMOLI, V.; JOURJON, G. Impact of Man-In-The-Middle Attacks on Ethereum. In: **2018 IEEE 37th Symposium on Reliable Distributed Systems (SRDS)**. [S.l.: s.n.], 2018. p. 11–20. ISSN 2575-8462.

FEAMSTER, N.; JUNG, J.; BALAKRISHNAN, H. An Empirical Study of "Bogon"Route Advertisements. **SIGCOMM Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 35, n. 1, p. 63–70, jan. 2005. ISSN 0146-4833. Disponível em: <http://doi.acm.org/10.1145/1052812.1052826>.

GANESH, P. I. et al. **Workload migration using on demand remote paging**. [S.l.]: Google Patents, 2012. US Patent 8,200,771.

GAO, L. On Inferring Autonomous System Relationships in the Internet. **IEEE/ACM Trans. Netw.**, IEEE Press, Piscataway, NJ, USA, v. 9, n. 6, p. 733–745, dez. 2001. ISSN 1063-6692. Disponível em: <http://dx.doi.org/10.1109/90.974527>.

GILBERT, E. N. Random graphs. **The Annals of Mathematical Statistics**, Institute of Mathematical Statistics, v. 30, n. 4, p. 1141–1144, 1959. ISSN 00034851. Disponível em: <http://www.jstor.org/stable/2237458>.

GIOTSAS, V. et al. Inferring multilateral peering. In: **ACM SIGCOMM Conference on emerging Networking EXperiments and Technologies (CoNEXT)**. [S.l.: s.n.], 2013. p. 247–258.

GOLDBERG, S. et al. How Secure Are Secure Interdomain Routing Protocols. In: **Proceedings of the ACM SIGCOMM 2010 Conference**. New York, NY, USA: ACM, 2010. (SIGCOMM '10), p. 87–98. ISBN 978-1-4503-0201-2. Disponível em: <http://doi.acm.org/10.1145/1851182.1851195>.

GRABER, S. **Network management with LXD (2.3+)**. 2016. Disponível em: <https://stgraber.org/2016/10/27/network-management-with-lxd-2-3/>.

HU, X.; MAO, Z. M. Accurate Real-time Identification of IP Prefix Hijacking. In: **2007 IEEE Symposium on Security and Privacy (SP '07)**. [S.l.: s.n.], 2007. p. 3–17. ISSN 1081-6011.

HUANG, T.-Y. et al. Teaching computer networking with mininet. In: **ACM SIGCOMM**. [S.l.: s.n.], 2014.

HUSTON, G.; ROSSI, M.; ARMITAGE, G. Securing BGP - A Literature Survey. **IEEE Communications Surveys Tutorials**, v. 13, n. 2, p. 199–222, Second 2011.

HYKES, S. **Docker 0.9: Introducing execution drivers and libcontainer**. 2014. Disponível em: <https://blog.docker.com/2014/03/docker-0-9-introducing-execution-drivers-and-libcontainer/>.

JAKMA, P.; LAMPARTER, D. Introduction to the Quagga Routing Suite. **IEEE Network**, v. 28, n. 2, p. 42–48, March 2014. ISSN 0890-8044.

JIN, C.; CHEN, Q.; JAMIN, S. **Inet: Internet Topology Generator**. [S.l.], 2000.

Karlin, J.; Forrest, S.; Rexford, J. Pretty Good BGP: Improving BGP by Cautiously Adopting Routes. In: **Proceedings of the 2006 IEEE International Conference on Network Protocols**. [S.l.: s.n.], 2006. p. 290–299.

KARYPIS, G.; KUMAR, V. **METIS – Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 2.0**. [S.l.], 1995.

KATZ-BASSETT, E. et al. Machiavellian Routing: Improving Internet Availability with BGP Poisoning. In: **Proceedings of the 10th ACM Workshop on Hot Topics in Networks**. New York, NY, USA: ACM, 2011. (HotNets-X), p. 11:1–11:6. ISBN 978-1-4503-1059-8. Disponível em: <http://doi.acm.org/10.1145/2070562.2070573>.

KENT, S.; LYNN, C.; SEO, K. Secure Border Gateway Protocol (S-BGP). **IEEE Journal on Selected Areas in Communications**, v. 18, n. 4, p. 582–592, April 2000.

KERRISK, M. **Namespaces in operation, part 1: namespaces overview**. [S.l.], 2013. Disponível em: <https://lwn.net/Articles/531114/>.

LANTZ, B.; O'CONNOR, B. A Mininet-based Virtual Testbed for Distributed SDN Development. In: **Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication**. New York, NY, USA: ACM, 2015. (SIGCOMM '15), p. 365–366. ISBN 978-1-4503-3542-3.

LEPINSKI, M.; SRIRAM, K. **BGPsec Protocol Specification**. 2017. RFC 8205 (Proposed Standard), Internet Engineering Task Force, updated by RFC 8206. [Online]. Disponível em: http://www.ietf.org/rfc/rfc8205.txt.

LI, J. et al. An Internet Routing Forensics Framework for Discovering Rules of Abnormal BGP Events. **SIGCOMM Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 35, n. 5, p. 55–66, out. 2005. ISSN 0146-4833. Disponível em: <http://doi.acm.org/10.1145/1096536.1096542>.

LI, Y.; LILJENSTAM, M.; LIU, J. Real-Time Security Exercises on a Realistic Interdomain Routing Experiment Platform. In: **Principles of Advanced and Distributed Simulation, 2009. PADS '09. ACM/IEEE/SCS 23rd Workshop on**. [S.l.: s.n.], 2009. p. 54–63.

LUCKIE, M. et al. As relationships, customer cones, and validation. In: **ACM Internet Measurement Conference (IMC)**. [S.l.: s.n.], 2013. p. 243–256.

LYCHEV, R.; GOLDBERG, S.; SCHAPIRA, M. BGP security in partial deployment: Is the juice worth the squeeze? **CoRR**, abs/1307.2690, 2013. Disponível em: <http://arxiv.org/abs/1307.2690>.

MADHYASTHA, H. V. et al. iPlane: An Information Plane for Distributed Services. In: **Proceedings of the 7th Symposium on Operating Systems Design and Implementation**. Berkeley, CA, USA: USENIX Association, 2006. (OSDI '06), p. 367–380. ISBN 1-931971-47-1. Disponível em: <http://dl.acm.org/citation.cfm?id=1298455.1298490>.

MADORY, D. **Large European Routing Leak Sends Traffic Through China Telecom**. [S.l.], 2019. Disponível em: <https://blog.apnic.net/2019/06/07/large-european-routing-leak-sends-traffic-through-china-telecom>.

MAHAJAN, R.; WETHERALL, D.; ANDERSON, T. Understanding BGP Misconfiguration. **SIGCOMM Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 32, n. 4, p. 3–16, ago. 2002. ISSN 0146-4833. Disponível em: <http://doi.acm.org/10.1145/964725.633027>.

MATHURIN, A. **A Regional Look into BGP Incidents in 2020**. [S.l.], 2021. Disponível em: <https://www.manrs.org/2021/03/a-regional-look-into-bgp-incidents-in-2020/>.

MEDINA, A. et al. BRITE: an approach to universal topology generation. In: **Proceedings of the 9th Annual International Symposium on Modeling, Analysis and Simulation on Computer and Telecommunication Systems (MASCOTS'01)**. [S.l.: s.n.], 2001.

MEDINA, A.; MATTA, I.; BYERS, J. On the Origin of Power Laws in Internet Topologies. **SIGCOMM Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 30, n. 2, p. 18–28, apr 2000. ISSN 0146-4833. Disponível em: <http://doi.acm.org/10.1145/505680.505683>.

MERKEL, D. Docker: Lightweight Linux Containers for Consistent Development and Deployment. **Linux J.**, Belltown Media, Houston, TX, v. 2014, n. 239, mar. 2014. ISSN 1075-3583.

MEYER, D.; PATEL, K. **BGP-4 Protocol Analysis**. 2006. RFC 4274 (Informational), Internet Engineering Task Force. [Online]. Disponível em: http://www.ietf.org/rfc/rfc4274.txt.

MITSEVA, A.; PANCHENKO, A.; ENGEL, T. The state of affairs in BGP security: A survey of attacks and defenses. **Computer Communications**, v. 124, p. 45 – 60, 2018. ISSN 0140-3664. Disponível em: <http://www.sciencedirect.com/science/article/pii/S014036641731068X>.

MONTGOMERY, D. C. **Design and Analysis of Experiments**. USA: John Wiley & Sons, Inc., 2006. ISBN 0470088109.

MURPHY, S. **BGP Security Protections**. 2002. Draft-murphy-bgp-protect-01, Network Working Group - Internet Engineering Task Force. [Online]. Disponível em: https://tools.ietf.org/html/draft-murphy-bgp-protect-01.

PILOSOV, A.; KAPELA, T. Stealing the Internet: An Internet-scale man in the middle attack. **NANOG-44, Los Angeles, October**, p. 12–15, 2008.

RAMANATH, A. **A Study of the interaction of BGP/OSPF in Zebra/ZebOS/Quagga**. [S.l.]: August, 2004.

REKHTER, Y.; LI, T.; HARES, S. **A Border Gateway Protocol 4 (BGP-4)**. 2006. RFC 4271 (Draft Standard), Internet Engineering Task Force, updated by RFCs 6286, 6608, 6793, 7606, 7607, 7705, 8212. [Online]. Disponível em: http://www.ietf.org/rfc/rfc4271.txt.

REKHTER, Y.; WATSON, T. J. **BGP Protocol Analysis**. 1991. RFC 1265 (Informational), Internet Engineering Task Force. [Online]. Disponível em: http://www.ietf.org/rfc/rfc1265.txt.

RIPE. **YouTube Hijacking: A RIPE NCC RIS case study**. [S.l.], 2008. Disponível em: <https://www.ripe.net/publications/news/industry-developments/youtube-hijacking-a-ripe-ncc-ris-case-study>.

_____. **RIPEstat Data API**. [S.l.], 2021. Disponível em: <https://stat.ripe.net/docs/data_api>.

SCHLAMP, J. et al. HEAP: Reliable Assessment of BGP Hijacking Attacks. **IEEE Journal on Selected Areas in Communications**, v. 34, n. 6, p. 1849–1861, June 2016. ISSN 0733-8716.

SCHLINKER, B. et al. PEERING: An AS for Us. In: **Proceedings of the 13th ACM Workshop on Hot Topics in Networks**. New York, NY, USA: ACM, 2014. (HotNets-XIII), p. 18:1–18:7. ISBN 978-1-4503-3256-9. Disponível em: <http://doi.acm.org/10.1145/2670518.2673887>.

SERMPEZIS, P. et al. ARTEMIS: Neutralizing BGP Hijacking Within a Minute. **IEEE/ACM Trans. Netw.**, IEEE Press, Piscataway, NJ, USA, v. 26, n. 6, p. 2471–2486, dez. 2018. ISSN 1063-6692. Disponível em: <https://doi.org/10.1109/TNET.2018.2869798>.

SHAVITT, Y.; ZILBERMAN, N. A Structural Approach for PoP Geo-Location. In: **2010 INFOCOM IEEE Conference on Computer Communications Workshops**. [S.l.: s.n.], 2010. p. 1–6.

SHI, X. et al. Detecting Prefix Hijackings in the Internet with Argus. In: **Proceedings of the 2012 ACM Conference on Internet Measurement Conference**. New York, NY, USA: ACM, 2012. (IMC '12), p. 15–28. ISBN 978-1-4503-1705-4. Disponível em: <http://doi.acm.org/10.1145/2398776.2398779>.

SMITH, B. R.; GARCIA-LUNA-ACEVES, J. J. Securing the border gateway routing protocol. In: **Proceedings of GLOBECOM'96. 1996 IEEE Global Telecommunications Conference**. [S.l.: s.n.], 1996. MiniConfInternet, p. 81–85.

SONG, Y.; VENKATARAMANI, A.; GAO, L. Identifying and Addressing Reachability and Policy Attacks in "Secure" BGP. **IEEE/ACM Transactions on Networking**, v. 24, n. 5, p. 2969–2982, October 2016.

SRIRAM, K.; MONTGOMERY, D. **Secure Interdomain Traffic Exchange: BGP Robustness and DDoS Mitigation**. [S.l.], 2018.

VEENMAN, C. Detecting BGP Origin Hijacks: Using a filter-based approach. 2019. Disponível em: https://repository.tudelft.nl/islandora/object/uuid:d6db6c1b-effc-4b69-83ac-e58d96512b08/datastream/OBJ/download.

VERVIER, P.; THONNARD, O. SpamTracer: How stealthy are spammers? In: **2013 Proceedings IEEE INFOCOM**. [S.l.: s.n.], 2013. p. 3477–3482. ISSN 0743-166X.

WäHLISCH, M.; MAENNEL, O.; SCHMIDT, T. C. Towards Detecting BGP Route Hijacking Using the RPKI. In: **Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication**. New York, NY, USA: ACM, 2012. (SIGCOMM '12), p. 103–104. ISBN 978-1-4503-1419-0. Disponível em: <http://doi.acm.org/10.1145/2342356.2342381>.

WANG, A. et al. Reduction-based Analysis of BGP Systems with BGPVerif. **SIGCOMM Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 42, n. 4, p. 89–90, aug 2012. ISSN 0146-4833. Disponível em: <http://doi.acm.org/10.1145/2377677.2377695>.

WATSON, P. A. Slipping in the Window: TCP Reset Attacks. Technical Whitepaper. In: . [S.l.: s.n.], 2003.

WAXMAN, B. M. Routing of multipoint connections. **IEEE Journal on Selected Areas in Communications**, v. 6, n. 9, p. 1617–1622, Dec 1988. ISSN 0733-8716.

WETTE, P. et al. MaxiNet: Distributed emulation of software-defined networks. In: **2014 IFIP Networking Conference**. [S.l.: s.n.], 2014. p. 1–9.

WHITE, B. et al. An Integrated Experimental Environment for Distributed Systems and Networks. **SIGOPS Oper. Syst. Rev.**, ACM, New York, NY, USA, v. 36, n. SI, p. 255–270, dez. 2002. ISSN 0163-5980.

WHITE, R. **Architecture and Deployment Considerations for Secure Origin BGP (soBGP)**. 2006. Draft-white-sobgp-architecture-02, Network Working Group - Internet Engineering Task Force. [Online]. Disponível em: https://tools.ietf.org/html/draft-white-sobgp-architecture-02.

WILLINGER, W. **The science of complex networks and the Internet: Lies, damned lies, and statistics**. Feb, 2010. Disponível em: <https://roughan.info/workshops/willinger_lec2.pdf>.

YAN, L.; MCKEOWN, N. Learning Networking by Reproducing Research Results. **SIGCOMM Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 47, n. 2, p. 19–26, maio 2017. ISSN 0146-4833.

ZHANG, K. et al. Performing BGP Experiments on a semi-Realistic Internet Testbed Environment. In: **25th IEEE International Conference on Distributed Computing Systems Workshops**. [S.l.: s.n.], 2005. p. 130–136. ISSN 1545-0678.

ZHANG, Z. et al. Ispy: Detecting Ip Prefix Hijacking on My Own. **SIGCOMM Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 38, n. 4, p. 327–338, ago. 2008. ISSN 0146-4833. Disponível em: <http://doi.acm.org/10.1145/1402946.1402996>.

ZHAO, X. et al. Detection of invalid routing announcement in the Internet. In: **Dependable Systems and Networks, 2002. DSN 2002. Proceedings. International Conference on**. [S.l.: s.n.], 2002. p. 59–68.

ZHENG, C. et al. A Light-weight Distributed Scheme for Detecting Ip Prefix Hijacks in Real-time. **SIGCOMM Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 37, n. 4, p. 277–288, ago. 2007. ISSN 0146-4833. Disponível em: <http://doi.acm.org/10.1145/1282427.1282412>.