MiniSecBGP: Testbed de Emulação Leve em Segurança BGP

**Emerson R. A. Barea, Denis W. S. Oliveira, Igor M. Floôr,
César A. C. Marcondes, Hermes Senger**

[1]Departamento de Computação – Universidade Federal de São Carlos (UFSCar)
Caixa Postal 676 – 13.565-905 – São Carlos – SP – Brasil

{emerson.barea,igor.floor,denis.oliveira,

cesar.marcondes,hermes.senger}@ufscar.br

Resumo. *O Border Gateway Protocol (BGP) é o protocolo de roteamento responsável pela conexão de toda internet e ataques aos sistemas BGP têm capacidade de causar grande impacto financeiro ou mesmo colocar em risco a soberania de um país. Apesar de sua relevância, não é comum a disponibilização de ambientes que suportem estudos aprofundados acerca do BGP. Nessa linha, este trabalho apresenta o MiniSecBGP, uma testbed que suporta a emulação de parte da topologia da internet de forma realística, por interagir com implementações de BGP amplamente adotadas. O MiniSecBGP possui arquitetura modularizada, tornando-o flexível e expansível. Testes preliminares indicaram uma boa capacidade de escalabilidade e acurácia da solução proposta.*

Abstract. *The Border Gateway Protocol (BGP) is the routing protocol responsible for connecting the entire Internet and attacks on BGP systems have potential to cause major financial impact or even jeopardize the sovereignty of a country. Despite their relevance, it is not common environments that support in-depth studies of BGP. In this line, this work presents the MiniSecBGP, a testbed that supports the emulation of part of the internet topology in a realistic mode, by interacting with widely adopted BGP implementations. MiniSecBGP has modular architecture, making it flexible and expandable. Preliminary tests indicated good scalability and accuracy of the proposed solution.*

1. Introdução

O *Border Gateway Protocol* (BGP) é o protocolo de roteamento que possibilita a comunicação entre as diversas redes que formam a internet, denominadas *Autonomous Systems* (ASes) [Y. et al. 2006]. Por meio dele, são anunciados os blocos de endereçamento IP de cada rede, bem como *loops* são evitados e as políticas de roteamento, que são parte do negócio dos provedores dessas redes, são aplicadas. A segurança do BGP tem sido alvo de novos estudos, entretanto, eventos de falhas de segurança ainda ocorrem. Como exemplo, [Shi et al. 2012] apresentaram um estudo sobre ataques de sequestro de prefixos de *Autonomous Systems* (ASes) mostrando que, em um ano, aproximadamente 220 ataques desse tipo foram detectados. Além disso, ataques de negação de serviço DDoS, Worms, Ransomware, Malwares, Spam, entre outros, são utilizados para explorar diretamente o protocolo e suas implementações [Akamai 2017] e [Arbor 2017].

Apesar de não muito frequente, ataques ao BGP normalmente causam grande impacto, pois, geralmente estão relacionados a indisponibilidade e, até mesmo, comprometimento de dados sensíveis e em grandes áreas geográficas. Tal fato torna a segurança do

BGP assunto de extrema relevância às empresas provedoras de serviços de internet (PSI), seus clientes e até mesmo órgãos responsáveis pela segurança nacional.

Mesmo com a relevância e evolução do tema, ainda não é comum a disponibilização de ambientes computacionais, como *testbeds* – dedicadas, capacitadas, de fácil implementação e baixo consumo de recursos – disponíveis ao estudo aprofundado das vulnerabilidades do BGP e o consequente risco ao ecossistema da internet, sendo assim, diante dessa realidade, é de grande importância a existência de um ambiente que sirva como base para a avaliação profunda dos riscos do BGP e, como resultado, proporcione mecanismos que auxiliem na mitigação dos riscos associados.

Buscando preencher essa lacuna, este trabalho apresenta o MiniSecBGP, uma *testbed* que mimetiza topologias de roteadores BGP com as características necessárias para representar parte da estrutura da internet, ou um conjunto de ASes, com intuito de servir como ambiente base para o estudo aprofundado das características de segurança do roteamento BGP. Entre os destaques e aspectos mais relevantes, este trabalho (*i*) faz uso de virtualização leve como base para *testbed*, (*ii*) utiliza de mecanismos que proporcionam geração de topologias com boa acurácia na representatividade de partes da internet e (*iii*) suporta geração e criação automática de todo ambiente.

O desenvolvimento dessa *testbed* justifica-se pela necessidade do estudo das vulnerabilidades relacionadas à utilização do BGP na internet, porém, a ausência de mecanismos tecnológicos com essa finalidade acaba tornando esse processo complexo e ariscado quando, por exemplo, testes e análises são realizados diretamente na internet [Katz-Bassett et al. 2011]. Além disso, o desenvolvimento de uma *testbed* com essa finalidade proporciona um ambiente capaz de suportar estudos que nos permitam, por exemplo, identificar os efeitos do tráfego da internet na convergência do BGP, avaliar riscos de sequestro de prefixos (*hijacking*), riscos relacionados a novas ameaças, entre outros.

O restante desse trabalho está organizado da seguinte forma: a seção 2 apresenta os requisitos a serem atendidos e os desafios impostos à proposta apresentada. Na seção 3 são apresentadas a arquitetura da *testbed* e o detalhamento do protótipo desenvolvido. A seção 4 apresenta e comenta os resultados dos testes realizados; a seção 5 apresenta trabalhos relacionados; e a seção 6 conclui e apresenta sugestões de trabalhos futuros e seção corresponde aos agradecimentos finais.

2. Requisitos e Desafios

Ao propor o desenvolvimento de uma *testbed* que mimetize a topologia de, pelo menos, uma parte da internet, alguns novos questionamentos surgem e levantam consigo requisitos ainda não contemplados. Por exemplo, segundo dados da Cisco[1], 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. Resultados de testes empíricos realizados junto ao *looking glass* (LG) do *Internet Exchange* (IX) de São Paulo, por exemplo, apontaram um consumo de aproximadamente 28MB de memória para o armazenamento de pouco menos de 87K prefixos IPv4[2].

A princípio esse consumo de memória pode não parecer relevante, porém, quando

[1] http://www.cisco.com/c/en/us/support/docs/ip/border-gateway-protocol-bgp/12512-41.html

[2] Acesso ao LG SP via telnet e captura da tabela de roteamento BGP com o comando `show ip bgp`.

pensamos em uma *testbed* responsável por mimetizar parte 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 sua implementação.

O consumo de recursos inerentes à própria virtualização é outro ponto relevante a ser considerado, por isso, deve-se dar preferência ao uso de tecnologias de emulação leve com suporte a recursos de rede. Um exemplo bastante conhecido de emulador com essas características é o Mininet[3]. O Mininet é um emulador que utiliza o conceito de virtualização baseada em processos de *namespaces* de rede, garantindo emulação leve e isolamento entre instâncias. Suporta a criação de ambientes com hosts, switches, roteadores e links parametrizáveis; possui aplicações Python e outras ferramentas Linux que abstraem em alto nível e facilitam a interação com o ambiente; e suporta a execução de aplicações de rede reais [Team 2018].

Outro requisito relevante é a correta definição da topologia a ser emulada na *testbed*, pois, essa deve ser uma réplica fidedigna e realística da porção real 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, pois, há mais do que simples ligações físicas e meios de comunicação bem definidos que devem ser considerados no ecossistema da internet.

Uma possível abordagem para atender esse requisito é obter a topologia a partir de inferências oriundas dos *traces* de rotas armazenadas da internet no Projeto RouteViews[4] ou RIS RIPE[5], porém, esse processo demanda tempo e poder de processamento que podem degradar a análise a ponto de inviabilizá-la. Uma segunda alternativa é inferir a topologia a partir de análises diretas das tabelas de rotas disponíveis em LGs espalhados pela internet [Brito et al. 2015], porém, esse procedimento também é dispendioso e não contempla a totalidade de caminhos possíveis entre os ASes.

Historicamente alguns modelos vêm sendo utilizados com a finalidade de reproduzir a topologia da internet. Alguns são baseados em pesquisa empírica por *traceroute*, [Madhyastha et al. 2006] e [Shavitt and 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*. Outro modelo bastante utilizado é o baseado em grafos aleatórios, iniciando com Erdos-Renyi [Erdös and Rényi 1959], passando por Waxman [Waxman 1988] e, posteriormente, grafos lei de potência [Medina et al. 2000]. Na linha de geradores de topologias, algumas ferramentas se destacam, como por exemplo o GT-ITM [Calvert et al. 1997] e Inet [Jin et al. 2000], considerados geradores de topologia orientados a modelos, ou seja, utilizam algum dos modelos apresentados acima na geração da topologia; e o 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. Ele lê os parâmetros de geração a partir de um arquivo de configuração que pode ser escrito diretamente pelo usuário, gerado automaticamente em sua GUI, importado de outros geradores de topologias ou a partir de dados topológicos coletados diretamente da internet.

[3]http://mininet.org/

[4]http://archive.routeviews.org/

[5]https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris/ris-raw-data

Um terceiro requisito a ser atendido pela *testbed* é a capacidade de emular topologias extensas e dinâmicas, suportando um grande número de roteadores BGP e links de comunicação que repliquem características reais, como banda suportada, comportamento de atraso e estabilidade. Utilizando como base tecnologias consideradas de baixo custo e sem restrição de uso. Além disso, a automação do *setup* do ambiente também é um requisito, pois, contribui para a agilidade das análises e garante maior acurácia nos resultados.

3. Materiais e Métodos

Nesta seção são detalhados os elementos formadores da arquitetura da *testbed*, bem como apresentados testes e resultados referentes à validação inicial do protótipo MiniSecBGP.

3.1. Arquitetura

A fim de atingir os objetivos propostos, atendendo aos requisitos apresentados e solucionando os desafios impostos, optou-se pela elaboração de uma arquitetura modularizada que suporte todo ciclo de vida de um experimento BGP, guiando o utilizador desde a fase de definição da topologia, passando pela configuração dos detalhes técnicos de endereçamento IP, características dos links e de serviços ofertados pelos ASes, configurando o ambiente de testes e extraindo os resultados. Figura 1.

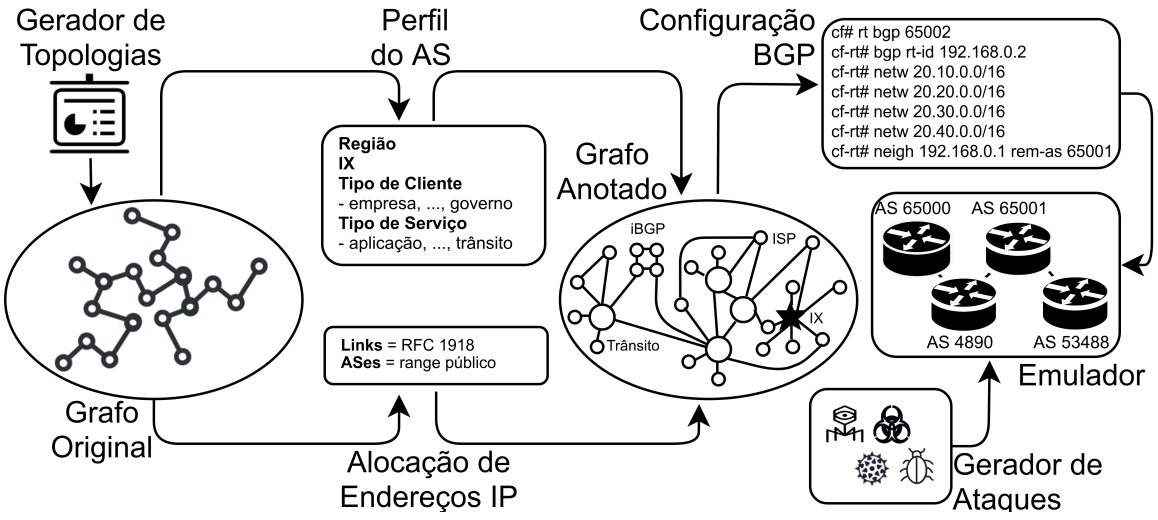


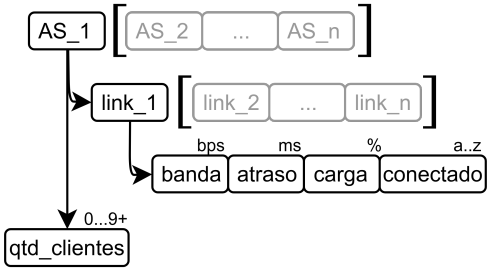
Figura 1. Elementos formadores da arquitetura da *testbed*.

Na Figura 1 é possível observar a relação entre os elementos responsáveis pelo funcionamento de toda *testbed*. Abaixo são descritas suas funcionalidades e a relação entre eles:

Gerador de Topologias: interface responsável pela definição do grafo correspondente a topologia emulada. Suporta a criação de topologias específicas definidas pelo utilizador; baseadas em modelos de grafos já estabelecidos, como apresentado na seção 2; ou importadas já prontas a partir de fontes externas.

A topologia resultante desse módulo tem sua estrutura representada na Figura 2(a). Nela é possível observar que o grafo é composto por um conjunto de ASes e seus atributos. Alguns atributos podem ser repetidos ou não, dependendo do número de ocorrências no

objeto. A Figura 2(b) apresenta um exemplo de objeto JSON de um AS. Nela é possível observar que o atributo `links` é composto por um arranjo que recebe novos elementos sempre quando novos links são identificados em um mesmo AS.



(a) Estrutura do grafo da Topologia

```
{"AS":"AS_1",
 "links": [
     {"descrição": "link_1",
      "banda": 1024,
      "atraso": 35,
      "carga": 70,
      "conectado": "AS_20:link_21"},
     {"descrição": "link_2",
      "banda": 2048,
      "atraso": 10,
      "carga": 55,
      "conectado": "AS_30:link_31"}],
 "qtd_clientes": 1580}
```

(b) Exemplo de objeto JSON de um AS

Figura 2. Representação gráfica e em JSON da estrutura de dados utilizada para representar ASes na *testbed* proposta.

Cada link possui os atributos `descrição`, que é o nome identificador do link; `banda`, representada em kilobits por segundo; `atraso`, em milissegundos; `carga`, que representa a porcentagem de utilização do link para *download*; e `conectado`, que identifica o AS e link ao qual ele está diretamente conectado. O atributo `qtd_clientes` informa o número de clientes conectados em cada AS.

Caso a topologia seja resultado de importação a partir de outra fonte de dados, é necessário que ela siga a estrutura apresentada na Figura 2. Esse módulo suporta a existência de atributos nulos nos objetos JSON, porém, essa ocorrência deve ser devidamente representada atribuindo `null` ao atributo correspondente.

Perfil do AS: define ou valida os valores dos atributos existentes no grafo de topologia e insere novas informações que detalham ainda mais o perfil de cada AS.

Nessa fase, primeiramente o utilizador deve indicar se o grafo gerado representa uma topologia `aleatória` ou `brasileira`. Caso indicado topologia aleatória, todo grafo permanece inalterado assim como recebido do módulo `Gerador de Topologias`. O processamento nesse módulo é então finalizado e o controle sobre o grafo é entregue diretamente ao elemento `Grafo Anotado`.

Se definido que o grafo representa uma topologia brasileira, então são inseridas informações sobre a distribuição territorial de cada AS, sua participação ou não em um IX regional, identificado os perfis dos clientes desse AS e informado os tipos de serviços de internet oferecidos por ele. Para armazenar essas informações, são criados novos atributos em cada objeto de AS no modelo JSON com a respectiva nomenclatura e estrutura:

- `região`: armazena o nome da região da qual o AS é parte;
- `IX`: identifica se o AS possui conexão direta com um IX regional. Recebe os valores `true` ou `false`. Quando pelo menos um AS de uma região participa de um IX, deve ser criado um objeto *Route Server*[6] (RS) no modelo JSON para

[6]http://ix.br/route-server-and-commmunities

representá-lo. Esse objeto possui o atributo `link` que armazena o identificador da conexão do AS ao IX e o atributo recebe novos links a cada novo participante observado no IX.

- `tipo_cliente`: identifica a quantidade de clientes do AS pertencentes a cada um dos grupos `empresa`, `domicílio` e `governo`. O valor 0 (zero) representa inexistência de cliente naquela categoria.
- `tipo_serviço`: composto pelos sub-itens `Aplicação`, `Hospedagem`, `Acesso`, `Conteúdo` e `Trânsito`. Informa os tipos de serviços oferecidos pelo AS a seus clientes. Recebe os valores `true` ou `false`.

Também são configurados novos valores aos atributos de `banda` e `carga` dos links de dados de cada AS, e todos os valores de atributos criados ou alterados nesse processo baseiam-se nos indicadores extraídos de [CGI.br 2016].

De forma resumida [CGI.br 2016] possui uma coleção de dados estatístico que, juntos, permitem identificar as características dos PSIs e medir os serviços oferecidos por eles no Brasil. Seus resultados contemplam a distribuição por região dos indicadores informados anteriormente e tratados neste módulo.

Apesar de somente 70% dos PSIs possuírem um AS, os indicadores da pesquisa são considerados parâmetros válidos nesse ambiente, pois, a conexão de um PSI com a internet passa obrigatoriamente por um AS em algum ponto da rede, portanto, as informações de tráfego, serviços utilizados, tipos de clientes etc, refletem no comportamento dos ASes.

Para melhor entendimento do processo de definição de perfil do AS realizado nesse módulo, o algoritmo 1 apresenta a sequência de tarefas executadas desde o recebimento do grafo da topologia inicial até o preenchimento de todo perfil.

Algoritmo 1: Definição do perfil dos ASes para topologia $brasileira$

Entrada:
$grafo$ ▷ arranjo contendo as informações da topologia recém gerada
$distribuição$ ▷ porcentagens de distribuição dos dados de ASes por região
Saída:
$grafo\_anotado$ ▷ arranjo com todos atributos parametrizados e preenchidos

```
1  início
2      para cada AS ∈ grafo faça
3          AS.região ← distribuição(região que o AS é parte)
4          AS.IX ← distribuição(participa do IX?)
5          AS.tipo_cliente.[emp., ..., gov.] ← distribuição(quantidade de clientes)
6          AS.tipo_serviço.[cont., ..., hosp.] ← distribuição(oferece serviço?)
7          AS.links[descrição, ..., conectado] ← distribuição(atualiza)
8          se AS.IX então
9              se não existir RS regional então
10                 cria RSregional;
11             fim
12             conecta AS com RSregional;
13         fim
14     fim
15 fim
16 retorna grafo_anotado
```

Vale ressaltar que, mesmo não definindo diretamente o comportamento e funcionamento da rede, a inclusão de algumas informações, como `região`, `tipo_cliente` e `tipo_serviço` 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, isto pode representar grande prejuízo financeiro.

As informações de definição do perfil do AS são acrescentadas ao grafo da topologia como novos blocos à estrutura da Figura 2(a), correspondendo a novos atributos no objeto JSON da Figura 2(b).

Alocação de Endereços IP: esse processo ocorre juntamente à definição de perfil dos ASes, tanto para topologias do tipo `aleatória` quanto `brasileira`. Realiza a alocação de endereços IPs nas interfaces dos links de dados e define os prefixos de sub-rede divulgados no BGP. Para isso, o processo descrito abaixo deve ocorrer:

- **links de dados:** recebem endereços IPv4 privados (RFC 1918), com 30 bits na máscara de sub-rede. Cada link do objeto AS recebe um novo atributo `ip` que armazena esse valor. Os endereços são atribuídos sequencialmente, em ordem crescente, através do parâmetro identificador dos links e não há controle de alocação por área geográfica da topologia. Esses prefixos não são divulgados no BGP.
- **prefixos divulgados:** os prefixos divulgados no BGP recebem blocos de endereços aleatórios com o número mínimo de bits na máscara necessário para comportar a quantidade de clientes atribuídos àquele AS durante o processamento do módulo `Perfil do AS`. A alocação não segue nenhuma ordem de distribuição específica, assim como realizado pelos *Regional Internet Registries* (RIR) ou *National Internet Registries* (NIR). Cada objeto AS recebe um novo atributo `neighbor` que recebe mais registros a cada novo prefixo divulgado.

Grafo Anotado: esse módulo faz a fusão do grafo contendo os perfis dos ASes, a estrutura de endereçamento IP e prefixo divulgados no BGP. Também mantém uma representação do grafo de topologia finalizado para consulta e visualização pelo utilizador. Nessa fase, a topologia está finalizada e pronta para ser codificada na tecnologia de emulação utilizada.

Configuração BGP: converte os objetos do modelo JSON em codificação válida no ambiente de emulação. O resultado é o conjunto dos comandos necessários para instanciação de toda *testbed* e sua execução no `Emulador` para ativação do ambiente.

Emulador: tecnologia de emulação leve que suporta toda *testbed*. Nele são criadas as instâncias de cada roteador BGP; configurados os IPs de conectividade e prefixos de rede divulgados no processo de roteamento; configurados os parâmetros de banda e atraso dos links de dados entre os roteadores; e executados os testes a partir de alterações de parâmetros do ambiente ou execução de ataques controlados pelo `Gerador de Ataques`.

Gerador de Ataques: esse módulo possui funções que simulam ataques ao ecossistema BGP previamente cadastrados na *testbed*. Além disso, é responsável por capturar dados que auxiliam na análise dos resultados. Por exemplo: tempo de propagação; área afetada;

banda consumida, processamento e memória dos roteadores, entre outros. Todo escopo de atuação e procedimento de obtenção de resultados do `Gerador de Ataques` deve ser desenvolvido de acordo com as interfaces de acesso à *testbed*, descrito na seção 3.2.

3.2. MiniSecBGP

A fim de validar a arquitetura apresentada anteriormente, foi desenvolvido o MiniSecBGP.

O MiniSecBGP é um protótipo 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, baseia-se na plataforma de emulação Mininet versão 2.2.2 (seção 2) como base dos contêineres dos ASes; utiliza o roteador BGP baseado em software Quagga[7] 0.99.24; o gerador de topologias BRITE versão 2.1b (seção 2) na concepção das topologias definidas pelo utilizador; e um conjunto de outras aplicações responsáveis pelas funções restantes.

Utiliza como base um servidor Dell PowerEdge R210-II, processador Intel Xeon E3-1270 v3 de 3.50GHz e 8 núcleos, com *Hyper-threading* habilitado e 16GB de memória RAM DDR3. O sistema operacional é o Ubuntu 16.04.4 LTS Server.

O acesso ao MiniSecBGP é feito através de conta de usuário válido no sistema operacional e toda interação com a *testbed* acontece através de uma aplicação desenvolvida em Python[8] 3.6.3 que faz a interface entre os módulos da arquitetura e aplicações externas.

Ao iniciar essa aplicação o utilizador tem a opção de escolher a forma como a topologia será gerada: utilizando linha de comando ou a interface gráfica do BRITE, ou importando a topologia de fonte externa. Após a geração da topologia, o grafo é convertido para o modelo JSON (Figura 2(b)) e os módulos `Perfil do AS` e `Alocação de endereços IP` são acionados para continuidade do processo.

Quando finalizados esses dois módulos, o `Grafo Anotado` é então gerado com a inserção de todos atributos no modelo JSON. Após essa fase, o módulo `Configuração BGP` interpreta os atributos dos objetos de ASes e RSes, gera e executa os comandos correspondentes à plataforma de emulação utilizada, no caso o Mininet. Esse processo instancia e configura todos contêineres de ASes e RSes existentes na topologia.

A fim de garantir a emulação do tráfego configurado no atributo `carga` de cada link de um AS, os contêineres recebem instâncias do aplicativo iPerf3[9] devidamente parametrizado para gerar e receber tráfego entre os contêineres num fluxo constante. Ao término desse módulo, toda *testbed* está operacional e pronta para interação do utilizador.

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

[7] https://www.nongnu.org/quagga/
[8] https://www.python.org/
[9] https://iperf.fr/

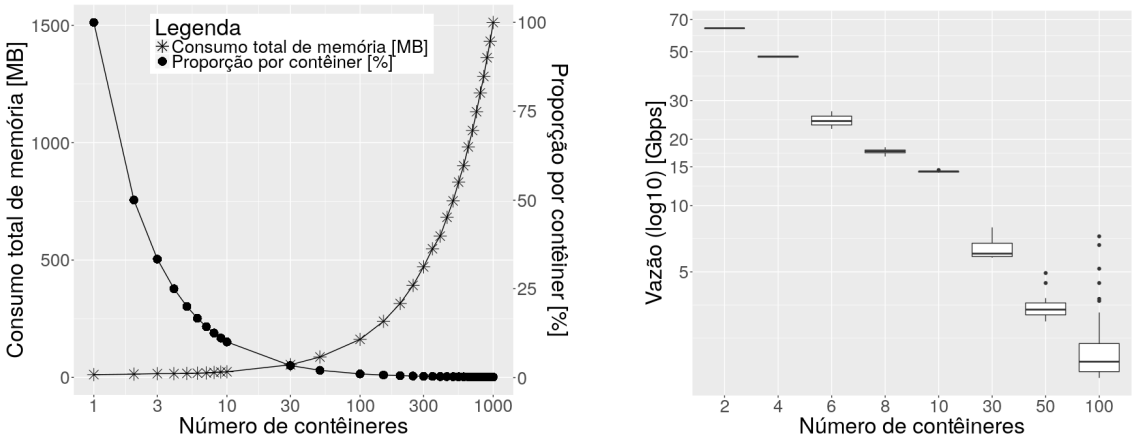
4. Testes de Validação e Discussão de Resultados

Para validação do MiniSecBGP foram propostas duas frentes de testes principais. A primeira analisa o comportamento da *testbed* com relação à escalabilidade do ambiente, enquanto a segunda avalia seu funcionamento em um caso específico de ataque BGP.

Com relação a quantidade de instâncias, o ambiente suportou a ativação de até 8188 contêineres sem modificação dos parâmetros de limites do Kernel, levando aproximadamente 1 segundo a cada grupo de 100 novos contêineres.

A Figura 3(a) apresenta o consumo de memória total (em MegaBytes) observado na ativação de até 1000 contêineres (eixo Y esquerdo). Apresenta também a proporção de memória consumida por cada contêiner (em porcentagem) sobre a memória total utilizada pelo Mininet (eixo Y direito). Apesar de cada instância consumir 1,5 MB de memória em média, notou-se a alocação de aproximadamente 12MB de memória pelo Mininet durante a instanciação do primeiro contêiner. Como parte dessa memória é compartilhada com outras instâncias, a proporção de memória utilizada por contêiner diminui com o aumento do número de instâncias, chegando a 0,1% aproximadamente em 1000 contêineres.

Quanto ao número de interfaces de rede, analisou-se a quantidade máxima suportada por um único contêiner, onde foram ativadas até 20.000 interfaces com sucesso. Constatou-se também que a inclusão de novas interfaces ocasiona pouco aumento no consumo de memória total, pois, o consumo de memória aumenta apenas 0,11% (aproximadamente 10MB) para cada novo grupo de 100 interfaces criadas num mesmo contêiner ou distribuída entre eles.



(a) Consumo de memória (b) Vazão máxima

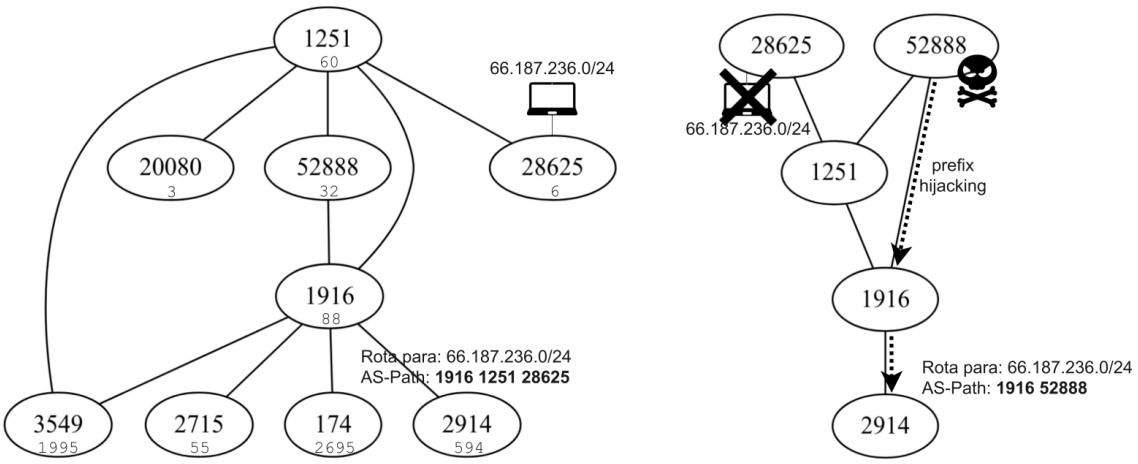
Figura 3. Escalabilidade e consumo de recursos do MiniSecBGP.

Quanto a vazão de rede dos contêineres, inicialmente foram realizados testes utilizando o iPerf3 (seção 3.2) para identificar a sobrecarga gerada pelo Mininet no ambiente. Foram comparados a vazão máxima do Linux no servidor hospedeiro com um par de contêineres Mininet devidamente instanciados e conectados. Foram realizadas 3 sequências de 10 testes cada um, onde constatou-se uma queda de 5,5% de vazão no ambiente Mininet. Além disso, analisou-se a degradação da vazão com o aumento do número de contêineres. Na Figura 3(b) é possível observar que há uma degradação elevada na vazão mesmo com um número não tão elevado de instâncias. O ambiente passa de

uma vazão inicial de 63,9 Gbps, entre um par de contêineres, para uma vazão média de 2,39 Gbps com 100 contêineres 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 testes que baseiam-se em 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 mitigar o erro.

Na segunda fase de testes, foi avaliado o comportamento da *testbed* na emulação de uma topologia que representa um conjunto de ASes contidos na Figura 4(a). A topologia é composta por 9 ASes, com um total de 5528 prefixos divulgados no BGP. Todos ASes estão divulgando e recebendo todas rotas conhecidas (*full routing*).



(a) Grafo da topologia emulada

(b) Exemplo de *hijacking* de prefixo

Figura 4. Representação gráfica de conjunto de ASes e suas conexões, com exemplo de ataque de *hijacking* de prefixo no BGP.

Nesse cenário foi avaliada a integração dos módulos da arquitetura do MiniSecBGP, 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.

Quanto a utilização de memória, somente a instanciação da topologia consumiu 21MB, 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 da *testbed* foi de 96MB.

Se compararmos o total de prefixos mantidos nas tabelas de rotas dos 9 ASes (94.194 rotas) e o consumo de memória correspondente (75MB), com a estimativa apresentada na seção 2, vemos que o ambiente segue o conceito apresentado quanto ao consumo de memória por número de prefixos em tabela de roteamento.

Realizadas as medições, o próximo passo foi a análise funcional do ambiente através da implementação de um ataque de *hijacking* de prefixo pelo BGP. Como apresentado

na Figura 4(b) o AS 52888 passa a divulgar o prefixo `66.187.236.0/24`, originalmente pertencente ao AS 28625, num determinado momento. Como não há mecanismos de filtros implementados nesse ambiente, em pouco tempo o AS 2914 recebe as atualizações iniciadas pelo AS 52888 e atualiza sua tabela de rotas, trocando o caminho para alcançar o prefixo sequestrado.

5. Trabalhos Relacionados

Quanto às abordagens para análise da segurança do BGP, uma parte relevante dos estudos emprega algum grau de simulação para prever os ataques possíveis e propor meios de mitigação. Trabalhos que fazem uso de BGP diretamente para testes na internet, como em [Katz-Bassett et al. 2011], são mais raros, principalmente devido ao alto risco envolvido. Outros trabalhos focam somente em monitoramento, como em [Claffy 2012], análise forense [Li et al. 2005] e [Zhao et al. 2002], validação da configuração [Wang et al. 2012] e visualização dos dados [Shi et al. 2012].

Com relação a *testbeds* para estudo e análise do BGP, [Schlinker et al. 2014] apresentaram o PEERING, uma *testbed* que possibilita testes diretamente conectados à internet. Para isso, tanto o prefixo quanto o AS utilizados são definidos pelos próprios mantenedores da *testbed*, além disso, é imposto controle de taxa de mensagens de atualização do BGP. Apesar das vantagens em realizar testes diretamente conectados à internet, essa *testbed* restringe e controla muitas das ações de seus utilizadores, limitando o escopo da análise.

Na mesma linha de *testbeds*, [Al-Musawi et al. 2017] apresentam um ambiente capaz de identificar anomalias em mensagens de atualização BGP em *traces* de tráfego de dados originados nos projetos RouteViews ou RIS RIPE. Por sua vez, essa *testbed* não suporta outras fontes de informação para definição de topologias, comportamento do BGP ou metodologias para análise das anomalias. Além disso, a *testbed* utiliza o Cisco VIRL[10], limitando sua escalabilidade horizontal e vertical, bem como necessitando de pagamento de licença de uso.

[Li et al. 2009] apresentaram uma plataforma que integra um roteador baseado em software de código aberto, XORP[11], sobre uma plataforma de virtualização leve, o OpenVZ[12], responsável por conduzir exercícios específicos de ataque e defesa BGP. Nessa *testbed* o tráfego gerado é capturado e desviado a um simulador que calcula os atrasos e perdas correspondentes à travessia da rede, com intuito de aumentar o realismo do ambiente. Apesar de empregar conceitos técnicos próximos aos utilizados no MiniSecBGP, essa *testbed* baseou-se em roteadores BGP de uso predominantemente acadêmico, prejudicando a representação fidedigna das tecnologias utilizadas na internet real.

6. Conclusões e Trabalhos Futuros

Este trabalho apresentou o MiniSecBGP, uma *testbed* capaz de mimetizar parte da estrutura de roteadores BGP da internet de forma realística, com a utilização de software comumente encontrado em ambientes de produção.

O MiniSecBGP possui arquitetura modularizada, o que lhe confere flexibilidade e possibilidade de expansão. O protótipo apresentado baseia-se num arcabouço de aplicações

[10]Virtual Internet Routing Lab: https://learningnetworkstore.cisco.com/virlfaq
[11]eXtensible Open Router Platform: http://www.xorp.org
[12]Open Virtuozzo: https://openvz.org

de rede e emulação leve desenvolvidos por terceiros, com acesso livre, somado a um conjunto de funções desenvolvidas especificamente para seu funcionamento.

Testes preliminares demonstraram bom nível de escalabilidade do ambiente quanto ao número de contêineres e interfaces virtuais suportados, bem como possibilitaram acompanhar o comportamento de um ataque de *hijack* de prefixo no BGP.

Como trabalhos futuros, considera-se relevante a realização de testes mais aprofundados para análise do comportamento da emulação de carga dos links no Mininet, assim como as alterações necessárias nos parâmetros de limites do Kernel do Linux para instanciação de um número maior de contêineres na mesma máquina. Outra abordagem considerada relevante é a implementação do MiniSecBGP com o Mininet Cluster[13] ou outra implementação distribuída do Mininet, possibilitando o uso da escalabilidade inerente aos ambientes distribuídos.

Quanto a geração de topologias, considera-se relevante utilizar a análise das tabelas de rotas BGP disponíveis nos LGs dos IXs na montagem de topologias mais precisas da internet brasileira, incluindo essa funcionalidade como mais uma forma de geração de topologia no MiniSecBGP.

7. Agradecimentos

Projeto parcialmente financiado pela FAPESP (Processo nº 2015/24352-9) e pelo Instituto Federal do Tocantins (IFTO). Hermes Senger agradece o apoio da FAPESP (Processos 2015/24461-2 e 2018/00452-2) e CNPQ (Processo 305032/2015-1).

Referências

Akamai (2017). State of the internet - security. Technical Report Volume 4 / Number 3. Q3, Akamai.

Al-Musawi, B., Branch, P., and Armitage, G. (2017). Rapid detection of bgp anomalies. *44 APNIC Conference, Taichung, Taiwan, 7 - 14 September 2017*.

Arbor (2017). Netscout arbor's 13th annual worldwide infrastructure security report. Technical report, Arbor.

Brito, S. H. B., Santos, M. A. S., Fontes, R., Perez, D. A., and Rothenberg, C. E. (2015). 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*, pages 67–80.

Calvert, K. L., Doar, M. B., and Zegura, E. W. (1997). Modeling internet topology. *IEEE Communications Magazine*, 35(6):160–163.

CGI.br, C. G. D. I. N. B. (2016). Pesquisa sobre o uso das tecnologias da informação e da comunicação no brasil - tic provedores 2014. *Disponível em:<http://www.cetic.br/media/docs/publicacoes/2/TIC_Provedores_2014_livro.pdf>*.

Claffy, K. (2012). Border gateway protocol (bgp) and traceroute data workshop report. *SIGCOMM Comput. Commun. Rev.*, 42(3):28–31.

[13]https://github.com/mininet/mininet/wiki/Cluster-Edition-Prototype

Erdös, P. and Rényi, A. (1959). On random graphs i. *Publicationes Mathematicae Debrecen*, 6:290.

Jin, C., Chen, Q., and Jamin, S. (2000). Inet: Internet topology generator. Technical report, Technical Report Research Report CSE-TR-433-00, University of Michigan at Ann Arbor.

Katz-Bassett, E., Choffnes, D. R., Cunha, I., Scott, C., Anderson, T., and Krishnamurthy, A. (2011). Machiavellian routing: Improving internet availability with bgp poisoning. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, HotNets-X, pages 11:1–11:6, New York, NY, USA. ACM.

Li, J., Dou, D., Wu, Z., Kim, S., and Agarwal, V. (2005). An internet routing forensics framework for discovering rules of abnormal bgp events. *SIGCOMM Comput. Commun. Rev.*, 35(5):55–66.

Li, Y., Liljenstam, M., and Liu, J. (2009). 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*, pages 54–63.

Madhyastha, H. V., Isdal, T., Piatek, M., Dixon, C., Anderson, T., Krishnamurthy, A., and Venkataramani, A. (2006). iplane: An information plane for distributed services. In *Proceedings of the 7th Symposium on Operating Systems Design and Implementation*, OSDI '06, pages 367–380, Berkeley, CA, USA. USENIX Association.

Medina, A., Lakhina, A., Matta, I., and Byers, J. (2001). 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)*.

Medina, A., Matta, I., and Byers, J. (2000). On the origin of power laws in internet topologies. *SIGCOMM Comput. Commun. Rev.*, 30(2):18–28.

Schlinker, B., Zarifis, K., Cunha, I., Feamster, N., and Katz-Bassett, E. (2014). Peering: An as for us. In *Proceedings of the 13th ACM Workshop on Hot Topics in Networks*, HotNets-XIII, pages 18:1–18:7, New York, NY, USA. ACM.

Shavitt, Y. and Zilberman, N. (2010). A structural approach for pop geo-location. In *2010 INFOCOM IEEE Conference on Computer Communications Workshops*, pages 1–6.

Shi, X., Xiang, Y., Wang, Z., Yin, X., and Wu, J. (2012). Detecting prefix hijackings in the internet with argus. In *Proceedings of the 2012 ACM Conference on Internet Measurement Conference*, IMC '12, pages 15–28, New York, NY, USA. ACM.

Team, M. (2018). Mininet - an instant virtual network on your laptop (or other pc). *Disponível em:<http://mininet.org/>*.

Wang, A., Gurney, A. J., Han, X., Cao, J., Talcot, C., Loo, B. T., and Scedrov, A. (2012). Reduction-based analysis of bgp systems with bgpverif. *SIGCOMM Comput. Commun. Rev.*, 42(4):89–90.

Waxman, B. M. (1988). Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, 6(9):1617–1622.

Y., R., T., L., and S., H. (2006). A border gateway protocol 4 (bgp-4).

Zhao, X., Pei, D., Wang, L., Massey, D., Mankin, A., Wu, S., and Zhang, L. (2002). Detection of invalid routing announcement in the internet. In *Dependable Systems and Networks, 2002. DSN 2002. Proceedings. International Conference on*, pages 59–68.