Uma abordagem leve para detecção de DDoS a partir de roteadores domésticos

Gabriel Mendonça[1], **Gustavo H. A. Santos**[1],
Edmundo de Souza e Silva[1], **Rosa M.M. Leão**[1], **Daniel S. Menasché**[1]

Universidade Federal do Rio de Janeiro (UFRJ), Rio de Janeiro, RJ.

{gabriel,gustavo,edmundo,rosam,sadoc}@land.ufrj.br

Resumo. *Ataques DDoS são prevalentes. Sua detecção deve ocorrer preferencialmente na borda da rede próximo à sua origem, especificamente nos roteadores domésticos. Entretanto, esses dispositivos tipicamente têm recursos limitados, tornando inadequadas abordagens baseadas em inspeção de pacotes ou análise de fluxos. Propomos um método extremamente leve para detecção de DDoS que usa apenas contadores de bytes de interfaces de rede. Para detectar ataques com tão pouca informação, treinamos modelos de Aprendizado de Máquina com dados reais do tráfego de centenas de usuários domésticos, juntamente com tráfego oriundo de ataques gerados em ambiente controlado. Mostramos que nossos classificadores são muito eficientes na detecção de ataques com diferentes vetores.*

Abstract. *DDoS attacks are prevalent. To mitigate their impact, detection should preferably occur closest to the attack origin, at the network edge, e.g., at home routers. However, these devices typically have limited resources and the use of approaches that resort on packet inspection do not bode well with such devices. We propose an extremely lightweight approach for DDoS detection that employs solely network interface byte counts. To detect attacks with such limited amount of information, our key insight consists in training classifiers making use of real workload data from nearly one thousand home-users augmented with attacks generated in a controlled environment. We show that our classifiers are very efficient in detecting attacks with different vectors.*

1. Introdução

Ataques DDoS (*Distributed Denial of Service*) têm sido motivo de enorme preocupação de governos e empresas. Estima-se que mais de um terço das redes classe C ativas na Internet foram alvo desse tipo de ataque entre 2015 e 2017 [Jonker et al. 2017]. Dispositivos IoT são particularmente vulneráveis, pois são usualmente instalados sem atualizações de *firmware* e com parâmetros *default* mantidos, sendo alvo fácil de *backdoors* e *exploits* para se criar grandes *botnets* [Antonakakis et al. 2017, Kolias et al. 2017]. Mirai, uma enorme *botnet* de IoT que atingiu uma população estável de 200.000-300.000 dispositivos, produziu em 2016 um ataque recorde que atingiu 623Gbps [Akamai 2018].

A recente onda de ataques DDoS cria desafios que exigem novas abordagens para a detecção e mitigação de tais ataques. A detecção deve ser simples o suficiente para escalar e insensível a assinaturas específicas de ataque. O Mirai, por exemplo, evoluiu a partir do *malware* BASHLITE [Antonakakis et al. 2017], destacando a importância da

detecção precoce de novas variantes de *malware*, que surgem diariamente [Akamai 2018, Kolias et al. 2017].

Preferencialmente, a detecção deve ocorrer perto da fonte do ataque para limitar seus efeitos na origem. No caso de *botnets*, ela pode, por exemplo, ser feita nos roteadores domésticos. No entanto, esses dispositivos são tipicamente limitados em termos de memória e poder de processamento. Em particular, o uso de DPI (*deep packet inspection*) ou mesmo de informações coletadas de cabeçalhos de pacotes consome recursos preciosos desses dispositivos. Além disso, o estudo a partir de roteadores domésticos não deve interferir no desempenho dos usuários nem violar a sua privacidade.

Neste artigo, nos propomos a analisar as seguintes questões: (1) Até que ponto é possível detectar ataques DDoS, de maneira extremamente leve, sem depender de informações de cabeçalhos de pacotes? (2) Que tipo de informação pode ser aproveitada para o treinamento de algoritmos de detecção de DDoS quando cabeçalhos não estão disponíveis? Propomos uma nova abordagem *extremamente leve*, usando apenas contadores de *bytes* de interfaces de rede de roteadores domésticos, evitando a análise onerosa de cabeçalhos ou conteúdo de pacotes.

Nossas principais contribuições são resumidas a seguir.
Medições de usuários domésticos na borda Estabelecemos uma parceria com um ISP para fazer a coleta de dados na borda da rede. Para os fins deste trabalho, usamos apenas informações de contagens de *bytes*, coletadas a partir da interface de rede local de 883 roteadores durante um período de 20 dias.
Detecção de DDoS empregando contadores de bytes de roteadores domésticos Executamos experimentos controlados para emular ataques Mirai e BASHLITE considerando diversos vetores de ataque comuns e combinamos nossos *traces* de *bytes* com esses ataques. O *dataset* aumentado é usado para treinar modelos de *Machine Learning* para detectar os ataques. Nossos resultados indicam que é possível detectar a presença de ataques DDoS usando exclusivamente a contagem de *bytes*.
Avaliação do poder de generalização Nossa abordagem se distingue dos trabalhos anteriores concentrando-se numa representação precisa do comportamento normal dos usuários domésticos, fruto de medições na borda da rede, em vez de focar nas especificidades de cada vetor de ataque. Como consequência, nossa proposta é capaz de detectar variantes de vetores de ataque antes de tornarem-se populares.

O restante deste artigo é organizado como segue. A Seção 2 apresenta uma visão geral do estado da arte na detecção de DDoS. Na Seção 3, descrevemos nossa proposta para detecção de DDoS. A Seção 4 relata nossos resultados e as conclusões estão resumidas na Seção 5.

2. Trabalhos Relacionados

Existe uma vasta literatura sobre mecanismos de detecção de DDoS [McDermott et al. 2018, Kuhnert et al. 2018, Dash and Craven 2017, De Carli et al. 2017, Chang et al. 2015, Wang et al. 2015], considerando pontos de detecção no núcleo da rede [Silveira et al. 2011, Liaskos et al. 2016, Mazel et al. 2015, Nevat et al. 2018, Pena et al. 2017], nos *hosts* de rede [Sedjelmaci et al. 2017, Summerville et al. 2015] e na borda [Doshi et al. 2018, Meidan et al. 2018, Ozcelik et al. 2017].

Soluções comuns para detecção e mitigação de DDoS empregam a análise de fluxos de pacotes, cabeçalhos e/ou *payloads* no núcleo da rede [Silveira et al. 2011, Liaskos et al. 2016, Mazel et al. 2015, Nevat et al. 2018, Pena et al. 2017]. Fazendo uso de informações no nível de pacotes, tais soluções são eficientes para detectar assinaturas de ataque previamente conhecidas [Liaskos et al. 2016, Mazel et al. 2015, Nevat et al. 2018, Pena et al. 2017] ou anomalias no comportamento dos fluxos em relação a um equilíbrio [Silveira et al. 2011] ou em relação a propriedades estatísticas dos campos dos cabeçalhos [Lakhina et al. 2005].

Em [Liaskos et al. 2016], os autores propõem uma estratégia de engenharia de tráfego para lidar com ataques de *link-flooding*. Com o redirecionamento contínuo do tráfego, os *bots* são obrigados a ajustar suas estratégias. Tal ajuste, por sua vez, facilita a detecção dos *bots* através de correlações temporais em seu tráfego. Nevat et al. [Nevat et al. 2018] também se aproveitam da natureza temporalmente correlacionada do tráfego gerado por *bots* para detectá-los.

Mazel et al.[Mazel et al. 2015] e Pena et al.[Pena et al. 2017] propõem métodos para detecção de anomalias baseados em aprendizado não-supervisionado que não dependem de assinaturas de ataque ou *labels* de tráfego. Tais métodos têm a vantagem de generalizar para novos vetores de ataque, mas ainda requerem informações no nível dos pacotes.

Embora as soluções implementadas no núcleo da rede sejam adequadas para detectar diferentes tipos de anomalias, elas não funcionam adequadamente para ataques DDoS baseados em *botnets*. Isso porque a detecção, caso seja bem sucedida, geralmente ocorre longe da fonte do ataque, fazendo com que seja difícil minimizar o seu impacto. Além disso, o processamento de grandes volumes de tráfego leva a problemas de escalabilidade, o que pode acarretar ineficiências de detecção.

Em [Sedjelmaci et al. 2017, Summerville et al. 2015], os autores propõem soluções para detectar ataques nos *hosts* da rede. Embora tais métodos sejam eficazes para localizar e isolar *botnets*, sua execução pode não ser viável em dispositivos de IoT. Esses dispositivos geralmente têm severas restrições de recursos, o que limita a complexidade dos algoritmos de detecção. Além disso, esses dispositivos também são limitados com relação a atualizações de *software/firmware*. Quando é possível aplicar *patches*, a atualização dos dispositivos geralmente remove suas vulnerabilidades e *exploits*, impedindo a infecção pelo *malware*. No entanto, essas atualizações são muito irregulares. Por isso é importante buscar por soluções alternativas que não dependam exclusivamente de *patches* nos *hosts* [Wang et al. 2017].

O DDoS baseado em *botnet* é tipicamente dividido em duas fases: varredura e ataque. Detectar uma *botnet* em sua fase de varredura é o ideal, já que a infecção precede o ataque. Ozcelik et al.[Ozcelik et al. 2017] propõem uma abordagem baseada em SDN para a detecção de *hosts* infectados na rede durante a fase de varredura. Para isso, eles se aproveitam de especificidades dos protocolos de controle usados pelas *botnets*. Tais procedimentos de controle e varredura, no entanto, mudam com o tempo e entre variantes de uma mesma *botnet*. A *botnet* Satori, por exemplo, é baseada no Mirai, mas não emprega *logins* forçados ao infectar dispositivos [Akamai 2018]. Há também *botnets* como Hajime que empregam comunicação totalmente distribuída inspirada no protocolo Bit-

Torrent [Kolias et al. 2017]. Por essa razão, neste artigo nos concentramos na detecção de *botnets* na fase de ataque.

Doshi et al. [Doshi et al. 2018] e Meidan et al. [Meidan et al. 2018] visam a classificação de pacotes como normais (gerados pelos usuários) ou maliciosos (gerados por ataques DDoS). Para isso, eles extraem *features* de fluxos de pacotes de dispositivos IoT coletados via `pcap`. Em seguida, modelos de aprendizado de máquina são treinados a partir dos dados coletados. Enquanto em [Doshi et al. 2018] os autores seguem uma abordagem supervisionada e contam com ataques emulados, Meidan et al. [Meidan et al. 2018] empregam uma abordagem baseada em detecção de anomalias usando dados empíricos coletados com os *malwares* Mirai e BASHLITE. O objetivo em [Doshi et al. 2018, Meidan et al. 2018] é classificar pacotes individuais, no entanto, no nosso trabalho o objetivo é detectar a presença de ataques em uma determinada janela de tempo. Essa diferença aparentemente sutil leva a consequências importantes, como discutido no restante deste trabalho. Além disso, nenhum desses trabalhos anteriores contou com dados reais de usuários domésticos como seu *baseline*, e todos exigem informações contidas nos cabeçalhos dos pacotes.

Nosso trabalho. Embora os ISPs tenham acesso aos fluxos de dados no núcleo da rede, detectar ataques na borda da rede é vantajoso para mitigar o impacto das *botnets*. Portanto, neste artigo propomos uma abordagem em que os ISPs coletam de seus usuários uma quantidade mínima de informação: a quantidade de *bytes* que trafegam na rede local dos usuários. Até onde sabemos, este trabalho é o primeiro a considerar tal abordagem que engloba: (1) *features* extremamente leves obtidas através do contador de *bytes* da interface LAN (incluindo tráfego sem fio) do roteador doméstico; (2) um modelo de aprendizado de máquina treinado a partir de dados reais de usuários domésticos; e (3) vetores de ataque gerados pelos códigos-fonte dos *malwares* Mirai e BASHLITE, que são combinados com o tráfego dos usuários domésticos para treinar os algoritmos de detecção.

3. Metodologia

A principal pergunta a ser respondida neste trabalho é: *será possível utilizar padrões de tráfego de usuários domésticos para detectar ataques e reagir rapidamente em tempo real*? Para responder a essa pergunta, desenvolvemos uma metodologia que engloba medições reais realizadas com um ISP e experimentos no laboratório. A partir dos dados coletados, *datasets* são criados para treinar modelos que são então usados para distinguir entre o tráfego normal da casa de um usuário e o tráfego total resultante quando um ataque ocorre.

Os principais requisitos da metodologia proposta são: (a) **simplicidade:** a metodologia deve ser simples o suficiente para escalar; (b) **localidade:** permitir a detecção de problemas próximos da fonte, isto é, no roteador doméstico; (c) **eficiência:** ser capaz de identificar ataques rapidamente, na ordem de minutos a partir do início do ataque; (d) **minimalismo:** coletar uma quantidade mínima de informações nos roteadores domésticos para evitar interferir em seu desempenho ou invadir a privacidade dos usuários.

Para atender aos requisitos apresentados acima, nossa metodologia baseia-se unicamente na observação de séries temporais do *tráfego total de upload* que flui através de roteadores domésticos. As séries temporais consideradas são divididas em *slots*, cuja duração é determinada pelos requisitos de tempo real. Os *slots* devem ser pequenos o

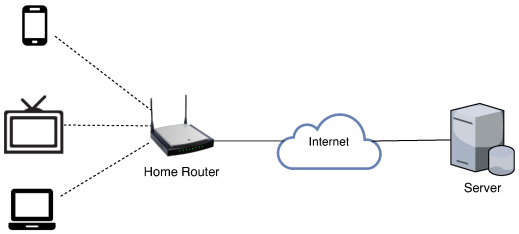

Figura 1. Infraestrutura de Medição

suficiente para permitir a identificação de ataques no horizonte de tempo desejado.

3.1. *Dataset* de tráfego

Introduzimos brevemente nosso *dataset* de tráfego e, em seguida, apresentamos algumas análises das séries temporais para mostrar as características gerais dos usuários domésticos considerados.

Através de uma parceria com um ISP de porte médio, obtivemos amostras do *tráfego total* (*upload* e *download*) fluindo através de roteadores domésticos de quase mil casas. É importante ressaltar que nenhuma informação do cabeçalho ou *payload* dos pacotes foi coletada. Os dados também foram anonimizados para evitar a identificação de residências.

Os dados com os quais trabalhamos foram obtidos de 883 residências em 20 dias, de 5 de julho a 25 de julho 2018. O *dataset* inclui a contagem de *bytes* de *upload* e *download* em cada residência, amostrada em intervalos fixos. Os dados foram coletados a partir de roteadores *wireless* que servem como *gateways* domésticos para o provedor de Internet, como mostrado na Figura 1. Os roteadores executam OpenWrt, usando *software* de código aberto para coletar e enviar informações para um servidor.

Tipos de dispositivos no *dataset*

Não fazemos hipótese alguma a respeito dos dispositivos presentes nas redes domésticas. De fato, antes de obter os dados do ISP, não tínhamos conhecimento prévio sobre o uso da rede doméstica de nenhum usuário e nem dos tipos de dispositivos pertencentes a esses usuários. Além disso, não aplicamos nenhuma pré-filtragem de dispositivos. Nossa metodologia é completamente agnóstica com relação à natureza dos dispositivos, uma vez que não se baseia em especificidades dos mesmos.

Com o objetivo de saber se o tráfego residencial coletado provinha de um conjunto diversificado de dispositivos, e não de um conjunto limitado em número e características, o que não seria adequado ao trabalho proposto, caracterizamos: (a) a distribuição de dispositivos por casa e; (b) a diversidade de dispositivos usados pelos usuários amostrados. A análise longitudinal que segue serve para ilustrar algumas propriedades básicas da população em estudo. Para isso, utilizamos endereços MAC e *hostnames* dos dispositivos que se conectaram pelo menos uma vez durante o período de medição de um subconjunto das residências motitoradas (aproximadamente 20,7% das 883 residências).

Observamos um total de 2.737 dispositivos do subconjunto de residências, com uma média de mais de 14 dispositivos por residência. Concluímos então que as redes domésticas em nosso *dataset* contém um número considerável de dispositivos, e mais de

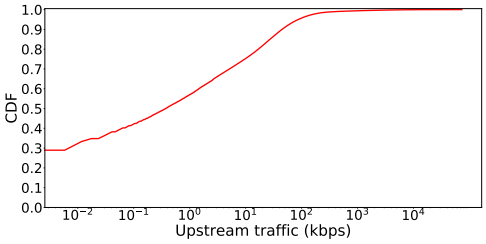

Figura 2. CDF do tráfego *upload*.

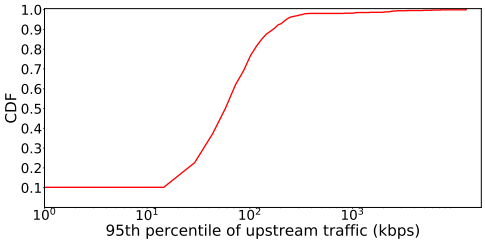

Figura 3. CDF do 95º percentil do tráfego *upload*.

50% das casas contêm pelo menos 11 dispositivos conectados.

Categorizamos os dispositivos observados usando as informações do *hostname* e do fornecedor (*vendor*). Sem nenhuma surpresa, constatamos que a maioria dos dispositivos (quase 50%) usam o sistema operacional *Android*. Nesta categoria pode-se incluir *smartphones*, *tablets* e algumas *smart TVs*. *Smartphones* são os dispositivos mais populares em nosso *dataset* (20%). Entretanto, nosso *dataset* inclui um número bem variado de dispositivos, a saber: *MP3 player*, *receiver* de TV, câmeras, dispositivos de som, *smart watches*, extensores WiFi, impressoras, consoles de *video game*, *smart TV* e dispositivos de *streaming*.

Algumas estatísticas de tráfego do *dataset*

Apesar de coletarmos individualmente o número de *bytes* transmitidos e recebidos por cada roteador doméstico a intervalos fixos, apenas o tráfego de *upload* foi usado para treinar o classificador proposto neste trabalho. Denominamos de *intervalo de amostragem* (IA) o intervalo de tempo entre duas medições. Os IAs foram fixados em um minuto ao longo do experimento. No futuro, planejamos trabalhar com granularidades mais finas, por exemplo, na escala de um segundo. Entretanto, como mostramos na Seção 4, a escala de um minuto foi suficiente para obter excelentes resultados.

Coletamos mais de 11 milhões de amostras (*bytes* transmitidos por IA) dos roteadores residenciais considerados. Cada amostra indica o tráfego total de saída (*upload*) da rede doméstica que passa pelo roteador em um IA. A Figura 2 mostra a distribuição cumulativa (CDF) do tráfego de *upload* por intervalo de amostragem. A Figura 2 indica que cada intervalo de amostragem possui um tráfego muito baixo de *upload*, conforme esperado. Em particular, em 90% dos intervalos a taxa de *upload* foi inferior a 50 kbps.

Para cada roteador doméstico, avaliamos o 95º percentil da taxa de *upload*. A Figura 3 mostra a CDF do 95º percentil para as 883 residências. Como indicado na Figura 3, em 95% dos domicílios, 95% das amostras de tráfego de *upload* são inferiores a 250 kbps. Além disso, em 42,8% das casas o *upload* é superior a 2 Mbps em apenas 10% do tempo.

3.2. Experimentos com ataques de *botnets*

Abaixo descrevemos a metodologia adotada em nossos experimentos controlados para geração de ataques.

Ambiente experimental: Para gerar *traces* de tráfego de ataque, executamos o código fonte de ***malwares* reais** em um RaspberryPi em ambiente controlado em nosso la-

Ataque	Tipo	Código fonte
UDP flood	Volumétrico	Mirai, BASHLITE
UDP-PLAIN flood	Volumétrico	Mirai
TCP SYN flood	Exaustão de Estado	Mirai, BASHLITE
TCP ACK flood	Exaustão de Estado	Mirai, BASHLITE

Tabela 1. Vetores de ataque usados nos experimentos.

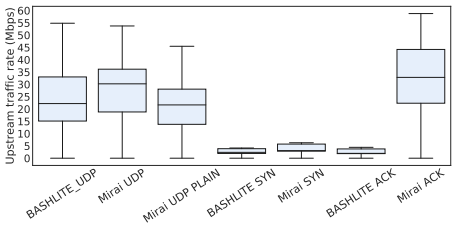

Figura 4. Boxplot do tráfego *upstream* gerado por cada vetor de ataque.

boratório. Conectamos um RaspberryPi via WiFi com um roteador sem fio TP-Link WDR3600 rodando OpenWrt. O RaspberryPi dispara ataques DDoS enquanto o roteador OpenWrt coleta o número de *bytes* recebidos na interface LAN a cada segundo. A partir desses experimentos controlados, obtemos o tráfego de *upload* de um ataque com granularidade de 1 segundo.

Código fonte de *botnets*: Consideramos duas das *botnets* de IoT mais comuns: Mirai e BASHLITE. Obtivemos as funções responsáveis pelos vetores de ataque de interesse de seus códigos fonte disponíveis publicamente, removendo todo código relacionado à varredura de vítimas e à comunicação com o servidor C2 (Comando e Controle). Então desenvolvemos um módulo de software para disparar os ataques DDoS das duas *botnets* para um determinado alvo por um tempo configurável. O software pode lançar ataques volumétricos e de exaustão de estado (ver Tabela 1).

Geração de ataques: A Tabela 1 mostra os quatro tipos de ataques considerados em nossos experimentos e sete vetores de ataque, quatro para Mirai e três para BASHLITE. Em particular, o ataque UDP-PLAIN é um *UDP flood* otimizado implementado apenas pelo Mirai. Para cada vetor de ataque, lançamos 300 ataques. Os ataques DDoS geralmente têm duração relativamente curta, como relatado em [Blenn et al. 2017, Jonker et al. 2017]. Portanto, geramos ataques que seguem uma distribuição Gaussiana com média $\mu = 120$ segundos e desvio padrão $\sigma = 10$ segundos. Como a maioria dos atacantes que usam Mirai e BASHLITE não modificam os parâmetros do *software* [Akamai 2018], adotamos os parâmetros padrão em nossos experimentos.

A Figura 4 compara as taxas de tráfego *upload* para cada um dos vetores de ataque considerados. Vale a pena notar a considerável diferença entre as taxas geradas. Enquanto ataques UDP podem gerar tráfego a taxas elevadas, superiores a 10 Mbps, outros vetores de ataque geram tráfego a taxas menores, 5 Mbps, sendo comparável às taxas de *upload* de usuários comuns. Pela Figura 4 pode-se concluir que a detecção de ataques DDoS não é trivial, exigindo a identificação de padrões sutis no tráfego normal do usuário doméstico e no tráfego de ataque. Em particular, a figura serve como evidência de que uma política simples de limiar, onde o tráfego com taxa superior a um certo valor é considerado como um ataque, não é suficiente para detectar um ataque.

3.3. Modelo de ataque DDoS

Em seguida, descrevemos como o modelo de ataque DDoS proposto combina *traces* de tráfego com os dados do ataque. Seja $\mathcal{H}$ o conjunto de residências utilizadas em nosso estudo. Deste conjunto, escolhemos aleatoriamente $n_B = |\mathcal{B}|$ casas a serem infectadas, onde $\mathcal{B} \subset \mathcal{H}$ é o conjunto de casas infectadas e $|\mathcal{B}|$ é a cardinalidade do conjunto $\mathcal{B}$. Uma *casa infectada* é uma casa que contém pelo menos um *bot* em qualquer dos dispositivos conectados a seu roteador doméstico.

Dividimos o tempo em *slots* de w minutos. Ao longo dos experimentos, escolhemos $w = 5$ minutos, o que equivale a 5.760 *slots* durante um período de 20 dias. Em cada *slot*, temos acesso às contagens de *bytes* de $|\mathcal{H}|$ roteadores domésticos. Seja $s(t, u)$ (resp., $b(t, u)$) o número de amostras (resp., contagem de *bytes*) no *slot* t correspondendo ao usuário u. Note que a maioria dos usuários reúne cinco amostras em cada *slot*. No entanto, a falta de sincronização entre relógios produz alguns pares *usuário-slot* (t, u) com quatro ou seis amostras.

Supomos que os usuários infectados agem como atacantes intermitentes, respondendo a um protocolo de comando e controle. Seja $\mathcal{A}$ o conjunto de *slots* em que os ataques DDoS ocorrem. Selecionamos aleatoriamente com distribuição uniforme um conjunto de $|\mathcal{A}|$ *slots*. Em cada um dos *slots* selecionados, o conjunto de $|\mathcal{B}|$ usuários infectados pré-selecionados contém os atacantes em potencial. Para efeitos de seleção dos pares *usuário-slots* em que um DDoS ocorre, filtramos e negligenciamos todos os pares $(t, u) \in \mathcal{A} \times \mathcal{B}$ tais que $s(t, u) \neq 5$. Nos pares *usuário-slot* restantes, combinamos os *traces* de tráfego real com o tráfego de ataque. Essa filtragem, que tem efeito insignificante sobre a acurácia dos nossos resultados, serve para simplificar o procedimento de combinação dos dados. O procedimento recebe como entrada o conjunto de cinco amostras coletadas em cada *slot* não filtrado e adiciona o tráfego de ataque. Alternativamente, para lidar com falta de dados em *slots* filtrados pode-se considerar o *padding* ou *smoothing* das amostras, com impactos insignificantes nos resultados reportados.

Seja $n_B(j)$ o número de atacantes no *slot* j. Lembramos que a duração de um ataque é escolhida a partir de uma distribuição Gaussiana com média de 120 segundos e variância de 10 segundos. O procedimento que combina *slots* de tráfego dos usuários com o tráfego de ataque garante que os ataques produzidos nunca cruzem as bordas dos *slots*. Devido à filtragem discutida no parágrafo anterior e à discussão acima, segue-se que $n_B(j) \leq |\mathcal{B}|$ se $j \in \mathcal{A}$ e $n_B(j) = 0$, caso contrário.

A Figura 5 ilustra como nosso *dataset* de ataque é construído. Nesse exemplo, $|\mathcal{H}| = 6, |\mathcal{B}| = 3$ e $|\mathcal{A}| = 2$. O tráfego gerado pelo usuário do roteador doméstico é representado por retângulos azuis e o tráfego do ataque é colorido de vermelho. Na figura, os roteadores domésticos HR3, HR5 e HR6 foram definidos como fontes de ataque, e os *slots* 1 e 2 foram escolhidos como *slots* de ataque. Observe que a Figura 5 ilustra o efeito de filtragem que ocorre no *slot* 2 do roteador doméstico HR5, que contém apenas 3 amostras de tráfego. Assim, HR5 é removido do conjunto de atacantes durante o *slot* 2 para simplificar o procedimento de geração do *dataset*.

Como discutido na Seção 3.2, geramos 300 padrões de tráfego de ataque para cada um dos sete vetores de ataque selecionados. Quando um *slot* é sorteado para conter um ataque, um novo padrão é selecionado daquele conjunto de maneira aleatória (uni-

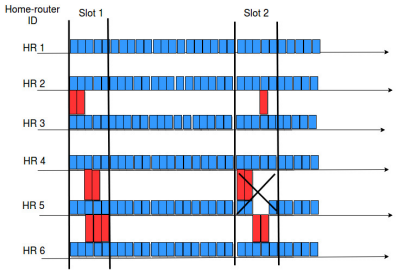

Figura 5. Exemplo de *dataset*.

forme), com reposição. Como a duração dos ataques segue uma distribuição Gaussiana com média dois minutos e um desvio padrão pequeno, a probabilidade de um ataque durar mais de cinco minutos é insignificante. Em particular, nenhum dos 300×7 padrões considerados dura mais de cinco minutos. Seja d_{ut} a duração de um ataque na casa u durante o intervalo t, $(t, u) \in \mathcal{A} \times \mathcal{B}$. Na Figura 5, as casas 3, 5 e 6 geraram ataques com duração $d_{3,1} = d_{5,1} = 2$ e $d_{6,1} = 3$ minutos, respectivamente, durante o *slot* número 1. Seja t_{ij} o instante de tempo em que um ataque do usuário i no *slot* j é iniciado dentro de j. Então, t_{ij} é selecionado de maneira aleatória (uniforme) entre os primeiros $w - d_{ij}$ minutos daquele *slot*. Na Figura 5, $t_{3,1} = 0$ e $t_{5,1} = t_{6,1} = 2$.

O modelo de ataques DDoS apresentado nesta seção se utiliza de nosso *dataset* de tráfego real e dos *traces* de ataque para gerar um *dataset* aumentado contendo ataques de *bots* intermitentes. Usamos esse *dataset* para alimentar classificadores que visam a detecção dos ataques.

3.4. Classificadores para detecção de ataques

A tarefa de nossos classificadores é detectar, para cada usuário doméstico, se um ataque DDoS ocorreu em um determinado *slot* de tempo. Nosso principal objetivo é verificar a viabilidade de se obter um classificador simples e confiável usando apenas contagens de *bytes*. Para tanto, os classificadores considerados fazem uso unicamente de *features* derivadas das amostras de tráfego de *upload* de roteadores durante *slots* de w minutos. Para manter a simplicidade do classificador, os *slots* são tratados de forma independente, sem considerar correlações espaço-temporais em diferentes roteadores domésticos. Na Seção 4.4, também mostramos os ganhos de acurácia obtidos aproveitando correlações espaço-temporais, ou seja, usando os resultados da classificação de roteadores domésticos distintos para tomar decisões em um determinado *slot*.

Lembramos que o tamanho do *slot* w considerado é de 5 minutos. Tamanhos maiores de *slot* favorecem a precisão, englobando mais dados para classificação, em detrimento do aumento do tempo de reação. Um *slot* com 5 minutos serve bem aos nossos propósitos, representando um bom equilíbrio ao lidar com ataques de duração média de 2 minutos permitindo tempos de reação razoáveis.

As *features* usadas pelos nossos classificadores são derivadas dos *traces* de tráfego de *upload* coletados em intervalos de um minuto nos roteadores domésticos, combinados com o tráfego de ataque, conforme descrito na seção anterior. Para cada *slot* de 5 minutos, calculamos, para cada roteador doméstico, as seguintes *features* de seu tráfego de *upload*: média, desvio padrão, mediana, mínimo, máximo e diferença entre máximo e mínimo. Tais *features* são as entradas para nossos classificadores.

Consideramos cinco classificadores diferentes: Regressão Logística, Árvores de Decisão, *Random Forests*, *Gaussian Naive Bayes* e *Multilayer Perceptron*. Todas as *features* foram normalizadas antes do ajuste dos modelos e os hiper-parâmetros foram deixados com os valores padrão fornecidos pelo `scikit-learn`.[1]

4. Resultados Experimentais

Nossos resultados experimentais têm três objetivos principais: (a) Comparar diferentes modelos de *Machine Learning* (ML) para detectar ataques a partir de roteadores domésticos em tempo real (i.e., a partir da borda da rede); (b) Avaliar o poder de generalização do classificador, treinando o classificador com dados de uma *botnet* para detectar ataques de uma *botnet* diferente; (c) Estender nossa análise para considerar correlações espaço-temporais entre os roteadores.

4.1. Conjuntos de treino e teste

Os *datasets* de treinamento e teste do Mirai (resp. BASHLITE) contém uma mistura dos quatro (resp. três) vetores de ataque diferentes considerados (ver Tabela 1). Exceto na Seção 4.3, usamos a mesma *botnet* (Mirai ou BASHLITE) para fins de treinamento e de teste. Usamos validação cruzada *5-fold* para seleção de modelos, treinando nossos modelos com 75% do nosso *dataset*. Nas seções subsequentes, usamos os primeiros 15 dias do *dataset* (75%) para treinar uma *Random Forest*, e os 5 dias restantes (25%) para fins de teste.

De acordo com [Antonakakis et al. 2017], foi medida uma média de 99 ataques DDoS da *botnet* Mirai por dia na Internet, durante um período de 5 meses que termina em 2017. Seguindo essa tendência, distribuímos um total de $|\mathcal{A}| = 1.980$ ataques DDoS durante o período de 20 dias de nosso *dataset*. Assumimos que 10% das residências têm um dispositivo infectado em sua rede local, ou seja, $|\mathcal{B}| = 88$. O desempenho do modelo é pouco sensível a esse parâmetro. Por falta de espaço, omitimos resultados para outros valores de $|\mathcal{B}|$.

4.2. Escolha do modelo e avaliação de desempenho

Escolhemos cinco modelos de ML para testar sua eficácia na detecção de ataques: Regressão Logística, Árvores de Decisão, *Random Forests*, *Gaussian Naive Bayes* e *Multilayer Perceptron*. Esses foram escolhidos favorecendo a simplicidade, ou seja, visando testar se modelos simples podem ser usados para alcançar nossos objetivos. O *F1 score* foi a métrica de desempenho alvo para comparação e uma validação cruzada *5-fold* foi usada no conjunto de treinamento.

Todos os modelos fornecem resultados muito bons no conjunto de treinamento com *F1 scores* maiores do que 0,99. O modelo *Random Forest* foi o melhor para os cenários avaliados. Os resultados expostos a seguir são obtidos usando *Random Forests*.

A Tabela 2 mostra a acurácia, *precision*, *recall* e *F1 score* obtidos pelo modelo *Random Forest* no conjunto de teste (últimos 5 dias) dos *datasets* Mirai e BASHLITE. O modelo é capaz de detectar *slots* de ataque com alta probabilidade (acurácia e *recall* superiores a 0,98) e exibe baixa taxa de alarmes falsos (*precision* superior a 0,99).

[1] `scikit-learn`: `http://scikit-learn.org`.

Botnet	Acurácia	Precision	Recall	F1 Score
Mirai	0.999797	0.995678	0.996146	0.995912
BASHLITE	0.999524	0.992077	0.988720	0.990395

Tabela 2. Resultados da classificação de ataques conhecidos.

Para avaliar a sensibilidade do modelo em relação a diferentes vetores de ataque, dividimos os resultados de desempenho por vetores de ataque, como mostrado na Tabela 3. Para gerar a Tabela 3, o modelo é treinado com uma mistura de todos os vetores de ataque de uma *botnet*, enquanto o conjunto de teste contém apenas um vetor de ataque daquela *botnet*. Os resultados relatados na Tabela 3 indicam que se o modelo for treinado com uma mistura de vetores de ataque, mas o atacante eventualmente usar apenas um único vetor, nosso mecanismo de detecção ainda será capaz de identificar com precisão os ataques.

Botnet	Attack	Acurácia
Mirai	TCP ACK flood	0.998732
Mirai	TCP SYN flood	0.989557
Mirai	UDP flood	1.000000
Mirai	UDP-PLAIN flood	0.996831
Mirai	No attack	0.999890

Botnet	Attack	Acurácia
BASHLITE	TCP ACK flood	0.980234
BASHLITE	TCP SYN flood	0.992432
BASHLITE	UDP flood	0.993898
BASHLITE	No attack	0.999799

Tabela 3. Acurácia do modelo por vetor de ataque.

Em seguida, avaliamos a relevância relativa das seis *features* usadas como entrada para o modelo de *Random Forest*. Para isso, usamos a métrica de importância baseada no *Gini index*. Para o Mirai, as *features* mais importantes foram o desvio padrão (Gini 0.489) e o valor máximo (Gini 0.141), sendo a menos relevante o valor mínimo (Gini 0.007). Já para o BASHLITE, as mais importantes foram a diferença entre máximo e mínimo (Gini 0.371) e o valor máximo (Gini 0.267) e a menos importante a mediana (Gini 0.004). Notamos que a variabilidade do tráfego dentro do *slot* é uma característica importante para diferenciar tráfego usual de tráfego anômalo.

4.3. Detectando ataques inéditos: transferência de conhecimento

Analisamos agora a capacidade de nossos classificadores de detectar novos ataques, treinando nosso modelo com o BASHLITE para detectar ataques do Mirai. A Tabela 4 mostra os resultados considerando todos os vetores de ataque do Mirai e a Tabela 5 detalha o desempenho para cada vetor de ataque. Claramente, o TCP SYN *flood* do Mirai é mais difícil de detectar nesse cenário. Esse fato pode ser explicado pela assinatura de tráfego do TCP SYN *flood* do Mirai, que é muito diferente da gerada pela implementação do BASHLITE. Comparando o código fonte das duas *botnets*, podemos observar que a implementação do ataque varia consideravelmente de um *malware* para o outro. Em particular, o TCP SYN *flood* do Mirai sempre emprega pacotes com *payload* vazio.

Como esperado, detectar um ataque usando modelos treinados com implementações de ataques diferentes não é trivial. Mesmo assim, a *precision* alta relatada na Tabela 4 sugere um número pequeno de falsos positivos, indicando que, ainda que o modelo possa falhar na detecção de alguns *slots* sob ataque, ele raramente classifica tráfego regular como malicioso.

Botnet Treinada	Acurácia	Precision	Recall	F1 Score
BASHLITE	0.996056	0.990786	0.849126	0.914503

Tabela 4. Resultados da classificação de "novos" ataques.

Ataque	UDP flood	UDP-PLAIN flood	TCP ACK flood	TCP SYN flood	No attack
Acurácia	0.999247	0.961972	0.979281	0.479294	0.999799

Tabela 5. Acurácia para os ataques da *botnet* Mirai usando o modelo BASHLITE.

4.4. Uso da correlação espaço-temporal em casas distintas na deteção de ataques

Os ataques DDoS baseados em *botnets* geralmente empregam grande número de dispositivos (ataque sincronizado). Nas seções anteriores mostramos como cada roteador residencial em um ISP, independentemente de outros, sinaliza a existência de um ataque proveniente da residência servida pelo roteador. Os resultados obtidos indicam que é possível identificar ataques com um mínimo de informação. Mesmo no pior cenário, isto é, onde treinamos o modelo com o BASHLITE e esse modelo é usado para detectar o Mirai, 73,44% dos ataques DDoS foram detectados por mais de 90% dos roteadores domésticos com capacidade de medição, enquanto 85,57% dos ataques foram detectados em pelo menos 50% dos domicílios infectados. Além disso, em 99,57% dos horários que não contêm ataques, menos de 1% das casas gerou alarmes falsos.

No que se segue, correlacionamos resultados da classificação obtidos em roteadores residenciais distintos. Para atingir esse objetivo, simplesmente verificamos o número de roteadores que reportaram um ataque a cada intervalo de tempo para decidir se um ataque está ou não realmente acontecendo, e mostramos abaixo que podemos melhorar significativamente a identificação de um ataque.

Podemos formular a correlação espaço-temporal de residências distintas como um problema de decisão usando o critério MAP (probabilidade *a posteriori* máxima) [Gallager 2013, Cap.8], onde D é a hipótese de que há um ataque *DDoS* num determinado *slot* e $\overline{D}$ é a hipótese de que não há ataque. Seja m o número de casas que reportaram um ataque. O problema consiste em encontrar o limiar m_0 para decidirmos pela hipótese D (há um ataque DDoS em curso) se $m \geq m_0$ e, caso contrário, assumirmos que não há um ataque (hipótese $\overline{D}$). As duas *likelihoods* seguem uma distribuição binomial, tendo parâmetros $(|\mathcal{H}|, p_1)$ no caso de ataque e $(|\mathcal{H}|, p_0)$ caso contrário. $|\mathcal{H}|$ (definido na Seção 3.3) é a cardinalidade do conjunto de roteadores residenciais, p_1 é a probabilidade de um roteador detectar um ataque dado que um ataque ocorreu, isto é, que a hipótese D é verdadeira, e p_0 é a probabilidade de um roteador detectar um ataque sem que um ataque tenha ocorrido. Quando não há ataque, p_0 é simplesmente a taxa de falso positivo p_{fp} de nosso classificador. Por outro lado, quando há um ataque, devemos notar que nem todo roteador está infectado. Logo, condicionando-se na probabilidade de uma residência estar infectada ($\frac{n_B}{|\mathcal{H}|}$), $p_1 = p_{rc}\frac{n_B}{|\mathcal{H}|} + p_{fp}(1 - \frac{n_B}{|\mathcal{H}|})$, onde p_{rc} é o *recall* do classificador.

Sejam P_D e $P_{\overline{D}}$ as probabilidades *a priori* de, respectivamente, haver ou não um ataque e $\eta = \frac{P_{\overline{D}}}{P_D}$. Aceitamos a hipótese de que existe um ataque (D) se $P(m|D)/P(m|\overline{D}) \geq \eta$. Não é difícil então mostrar mostrar que

$$m_0 = \frac{\log \eta + |\mathcal{H}| \cdot [\log(1 - p_{fp}) - \log(1 - p_1)]}{\log p_1 - \log(1 - p_1) - \log p_{fp} + \log(1 - p_{fp})}.$$

Escolhendo os parâmetros de acordo com o pior cenário de nosso modelo, teremos $m_0 = 12.86$, ou seja, basta que o ataque seja detectado por 13 residências (dentre as $n_B = 88$ infectadas) para decidir pela hipótese H_1 de que há um ataque DDoS em curso! Ao mesmo tempo, um alarme falso só ocorreria se houvesse mais de 13 falsos positivos em um *slot* sem ataque, e essa probabilidade é extremamente pequena.

5. Conclusão

Neste artigo, propusemos uma nova abordagem para detecção de DDoS que se utiliza de *datasets* reais de contagens de *bytes* de tráfego para caracterizar a atividade de um usuário doméstico. As amostras foram coletadas a partir dos contadores de *bytes* de quase 1000 roteadores domésticos por um período de 20 dias.

Nossos resultados mostram que ataques DDoS podem ser detectados na borda da rede em tempo real. A simplicidade da abordagem é fundamental para atingir os níveis desejados de generalização, permitindo a detecção precisa de variantes de vetores de ataque. Utilizando a correlação espaço-temporal entre casas distintas, podemos aumentar ainda mais o desempenho da abordagem. A alta acurácia do método proposto, combinada à sua baixa taxa de alarmes falsos, indica que essa estratégia é promissora, e não compromete a privacidade dos usuários.

Agradecimento: Este trabalho é parcialmente suportado por projeto de cooperação MCTIC-RNP/NSF, e projetos do CNPq e FAPERJ, além de bolsas CAPES.

Referências

Akamai (2016-2018). State of the Internet: Q3 2016, Summer 2018; Threat Advisory: Mirai Botnet 2016, Satori Mirai Variant Alert 2017. Technical report, Akamai.

Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J. A., Invernizzi, L., Kallitsis, M., et al. (2017). Understanding the mirai botnet. In *USENIX Security Symposium*, pages 1092–1110.

Blenn, N., Ghiëtte, V., and Doerr, C. (2017). Quantifying the spectrum of denial-of-service attacks through internet backscatter. In *Proceedings of the 12th International Conference on Availability, Reliability and Security*, page 21. ACM.

Chang, W., Mohaisen, A., Wang, A., and Chen, S. (2015). Measuring botnets in the wild: Some new trends. In *ACM Symposium on Information, Computer and Communications Security*, pages 645–650. ACM.

Dash, W. and Craven, M. J. (2017). Exploring botnet evolution via multidimensional models and visualisation. In *International Workshop on Security and Trust Management*, pages 72–88. Springer.

De Carli, L., Torres, R., Modelo-Howard, G., Tongaonkar, A., and Jha, S. (2017). Botnet protocol inference in the presence of encrypted traffic. In *INFOCOM*, pages 1–9. IEEE.

Doshi, R., Apthorpe, N., and Feamster, N. (2018). Machine learning ddos detection for consumer internet of things devices. *arXiv:1804.04159*.

Gallager, R. G. (2013). *Stochastic Processes: Theory for Applications*. Cambridge University Press.

Jonker, M., King, A., Krupp, J., Rossow, C., Sperotto, A., and Dainotti, A. (2017). Millions of targets under attack: a macroscopic characterization of the dos ecosystem. In *Proceedings of the 2017 Internet Measurement Conference*, pages 100–113. ACM.

Kolias, C., Kambourakis, G., Stavrou, A., and Voas, J. (2017). DDoS in the IoT: Mirai and other botnets. *Computer*, 50(7):80–84.

Kuhnert, K., Steinberger, J., and Baier, H. (2018). Botnet detection and prevention in anonymous networks. In *Intl. Conf. Autonomous Infrastructure, Management and Security*.

Lakhina, A., Crovella, M., and Diot, C. (2005). Mining anomalies using traffic feature distributions. In *ACM SIGCOMM Computer Communication Review*, volume 35, pages 217–228. ACM.

Liaskos, C., Kotronis, V., and Dimitropoulos, X. (2016). A novel framework for modeling and mitigating distributed link flooding attacks. In *INFOCOM*, pages 1–9. IEEE.

Mazel, J., Casas, P., Fontugne, R., Fukuda, K., and Owezarski, P. (2015). Hunting attacks in the dark: clustering and correlation analysis for unsupervised anomaly detection. *International Journal of Network Management*, 25(5):283–305.

McDermott, C. D., Majdani, F., and Petrovski, A. (2018). Botnet detection in the internet of things using deep learning approaches.

Meidan, Y., Bohadana, M., Mathov, Y., Mirsky, Y., Breitenbacher, D., Shabtai, A., and Elovici, Y. (2018). N-baiot: Network-based detection of iot botnet attacks using deep autoencoders. *arXiv:1805.03409*.

Nevat, I., Divakaran, D. M., Nagarajan, S. G., Zhang, P., Su, L., Ko, L. L., and Thing, V. L. (2018). Anomaly detection and attribution in networks with temporally correlated traffic. *Transactions on Networking*, 26(1):131–144.

Ozcelik, M., Chalabianloo, N., and Gur, G. (2017). Software-Defined Edge Defense Against IoT-Based DDoS. In *2017 IEEE Intl. Conf. Computer and Information Technology (CIT)*, pages 308–313. IEEE.

Pena, E. H., Carvalho, L. F., Barbon Jr, S., Rodrigues, J. J., and Proença Jr, M. L. (2017). Anomaly detection using the correlational paraconsistent machine with digital signatures of network segment. *Information Sciences*, 420:313–328.

Sedjelmaci, H., Senouci, S. M., and Taleb, T. (2017). An accurate security game for low-resource iot devices. *IEEE Transactions on Vehicular Technology*, 66(10):9381–9393.

Silveira, F., Diot, C., Taft, N., and Govindan, R. (2011). Astute: Detecting a different class of traffic anomalies. *ACM SIGCOMM Computer Communication Review*, 41(4):267–278.

Summerville, D. H., Zach, K. M., and Chen, Y. (2015). Ultra-lightweight deep packet anomaly detection for internet of things devices. In *IPCCC*, pages 1–8. IEEE.

Wang, A., Mohaisen, A., Chang, W., and Chen, S. (2015). Delving into internet ddos attacks by botnets: characterization and analysis. In *DSN*, pages 379–390. IEEE.

Wang, B., Li, X., de Aguiar, L. P., Menasche, D. S., and Shafiq, Z. (2017). Characterizing and modeling patching practices of industrial control systems. *POMACS*, 1(1):18.