# Uma Estratégia para Detecção Online de Bots P2P

**Rodrigo M. P. Silva**[1], **Ronaldo M. Salles**[1]

[1]Dept de Engenharia de Computação – Instituto Militar de Engenharia (IME)
Praça Gen Tibúrcio 80 – 22290-270 – Rio de Janeiro – RJ – Brasil

***Abstract.*** *Botnets are the main vehicle for illegal activities on the Internet causing losses of billions of dollars to the world economy and putting at risk the security of nations. Aiming to mitigate this threat several botnet detection schemes have been proposed in the literature, but most of them were designed for centralized architectures and apply methods that perform offline analysis. Such approaches look for patterns in long network traces (past history) often using clustering algorithms, which can take days to get a satisfactory result. This paper proposes an online architecture to detect descentralized botnets divided into two main phases. In the first phase, all nodes that have active P2P applications are sought so that later in the second phase, a differentiation between legitimate P2P applications and P2P bots is achieved. At the end of this work, experimental results are presented for two distinct scenarios to show the efficiency of the proposed architecture.*

***Resumo.*** *Botnets são o principal veículo para atividades ilícitas na internet, o crescimento de suas atividades tem causado prejuízos de bilhões de dólares à economia mundial e colocado em risco a segurança de nações. Com o objetivo de mitigar esta ameaça, muitas propostas de detecção têm sido elaboradas, porém a maioria ainda com seu foco nas arquiteturas centralizadas e empregando métodos que executam análises offline. Essas análises buscam por padrões em longos históricos de rede, muitas vezes utilizando algorítimos de clusterização que podem demorar dias para retornar uma resposta satisfatória. Neste trabalho é proposta uma arquitetura online para a detecção de botnets descentralizadas dividida em duas fases. Na primeira fase são buscados todos os nós que possuem aplicações P2P ativas para que posteriormente, na segunda fase, sejam diferenciadas as aplicações P2P legítimas das bots P2P. Ao final deste trabalho é feita uma validação em dois cenários distintos de forma a comprovar a eficiência da arquitetura na detecção online de bots P2P.*

## 1. Introdução

*Bots* são atualmente a principal ameaça à segurança na internet, este tipo de *malware* dotado de um canal de comando e controle (C&C) é responsável por inúmeras atividades maliciosas. Suas redes, denominadas *botnets*, não possuem fronteiras geográficas para atuação, podendo desta forma possuir membros nos mais diversos países, o que torna a descoberta de seu controlador (*botmaster*) uma tarefa complicada. Não existem ainda garantias de que caso um *botmaster* seja identificado, ocorram sanções legais sobre ele, já que muitos países não possuem uma legislação em relação a crimes cibernéticos.

Atualmente, a principal atividade maliciosa exercida pelas *botnets* é o envio de *spams*. Muitas vezes por serem constituídas por milhares de máquinas, as *botnets*

mostram-se um veículo ideal para esta atividade. Segundo pesquisa realizada pelo Instituto Sophos [sop 2012], o Brasil atualmente ocupa a sexta colocação entre os países que mais enviam *spams*. Esforços com o objetivos de deter a operação destas redes têm sido realizados, diversas técnicas de detecção e mitigação do problema já foram propostas [Spognardi et al. 2005, Coskun et al. 2010, Wang et al. 2009] , entretanto, as *botnets* vêm evoluindo com o intuito de dificultar cada vez mais a sua detecção e de se tornarem mais resilientes.

Elas têm migrado de arquiteturas centralizadas (baseadas em protocolos HTTP e IRC) mais simples, para arquiteturas descentralizadas (baseadas em protocolos P2P) que são capazes de prover maior resiliência as suas redes devido a própria natureza do protocolo em que se fundamentam. Entretanto, muitas técnicas que são propostas ainda nos dias de hoje visam detectar *botnets* com arquiteturas centralizadas, e em sua grande parte não contemplam a dinâmica de nós existente nas rede. Constantemente nós são inseridos e removidos das redes ou mesmo têm seus endereços IPs alterados, o que dificulta o descobrimento de máquinas infectadas pelas metodologias desenvolvidas, caso estas demandem longos períodos de tempos para executar suas análises e retornar uma resposta. Nesses casos as informações sobre nós infectados podem não chegar a tempo de serem úteis.

Neste artigo é proposta uma arquitetura capaz de detectar *bots* descentralizadas (P2P) de forma *online*, ou seja, reduzindo bastante o tempo entre a produção dos dados a serem analisados e os resultados obtidos por esta análise. Ademais, a arquitetura proposta foi dividida em duas fases distintas, detecção de nós com aplicações P2P ativas (na fase 1) e distinção entre aplicações P2P legítimas e *bots* P2P (fase 2), visando assim uma maior otimização do processo. A divisão também é justificada pelo fato de que uma *bot* P2P é essencialmente uma aplicação P2P e muitas vezes utiliza a estrutura dessas redes para seu funcionamento.

Este trabalho apresenta as seguintes contribuições:

- uma arquitetura de detecção online de nós com *bots* P2P ativas;
- módulo para diferenciação entre aplicações P2P legítimas e *bots* P2P;

O trabalho é apresentado da seguinte forma. Na seção 2 são apresentados alguns trabalhos anteriores relacionados ao tema. É detalhada na seção 3 a arquitetura proposta em cada uma de suas fases. Uma validação desta arquitetura é realizada na seção 4, onde são mostrados os resultados obtidos. Na seção 5 é feita a conclusão deste artigo bem como algumas propostas para trabalhos futuros.

## 2. Trabalhos Relacionados

Atualmente as *Botnets* são consideradas uma importante ameaça à segurança da internet. Com isto, diversos pesquisadores têm direcionado seus trabalhos para a detecção e mitigação dessas redes de *malwares*. Neste capítulo relacionamos alguns trabalhos correlatos que foram utilizados como base para a elaboração e execução deste artigo.

Karagiannis et al. em [Karagiannis et al. 2004b] foram um dos primeiros a criticar os métodos baseados em assinaturas de *payload* ou que utilizem valores de portas para a identificação de aplicações. Os autores fizeram uma análise de um conjunto de características de redes, como volume de tráfego, tempo de conexão, número de máquinas

conectados por rede, entre outras, com o intuito de encontrar um padrão adotado pelas aplicações P2P. Através desta caracterização foi possível encontrar protocolos P2P conhecidos e descobrir novos.

Chang e Daniels em [Chang and Daniels 2009] propuseram uma técnica baseada nos padrões do tráfego de rede. Porém, entre as características apontadas para classificação está o valor da porta utilizada, que como mencionado em [Karagiannis et al. 2004a], limita a proposta a detecção de aplicações já conhecidas que não utilizem de escolhas aleatórias para o uso das portas para a comunicação.

Wang et al. em [Wang et al. 2009] estudaram os modelos de propagação e comunicação utilizados por *bots* P2P e fundamentados em técnicas como o *Sybil Attack* conseguiram comprometer o canal de C&C de *botnets* P2P. Apesar de ser eficiente, a abordagem não detecta os membros da rede e não é capaz de garantir que a comunicação não será bem sucedida para todos os nós.

Coskum et al. [Coskun et al. 2010] utilizaram o número de conexões feitas por cada computador para encontrar *peers* membros de uma mesma rede de *bots*. Entretanto, como mencionado pelos próprios autores, redes muito populosas podem não ser detectadas por tal método, pois seus *peers* são confundidos com servidores acessados por diversas máquinas clientes. Outra limitação deste método é a necessidade de se conhecer previamente pelo menos um membro da rede. Ademais, o conhecimento de um *bot*, não permite atestar que todos as máquinas da rede identificadas como livres de *bots*, realmente não estão infectados por alguma outra *botnet*.

Em [Liu et al. 2010], Liu et al. elaboraram uma metodologia em fases semelhante a proposta neste artigo. Inicialmente é desejada a detecção de todos os nós com algum tipo de aplicação P2P ativa, para depois na fase seguinte verificar entre aqueles quais são *bots* P2P. Entretanto, em sua proposta há o emprego de técnicas de clusterização (como *K-means*) que demandam longos períodos de processamento, em alguns casos podendo chegar a dias como mostrado pelos próprios autores, o que torna inviável a detecção em redes dinâmicas. De forma análoga, Zhang et al. em [Zhang et al. 2011] propuseram uma metodologia baseada em fases, porém utilizaram a persistência inerente a *bot* P2P para detectá-la, característica esta que demanda grande quantidade de tempo para ser verificada, não permitindo desta forma uma resposta em um curto período.

## 3. Arquitetura de Detecção de Bots P2P

A arquitetura proposta neste trabalho tem o objetivo de detectar *bots* P2P de maneira *online*. Por *online*, entende-se que não deve haver uma demora entre a produção dos dados a serem analisados e os resultados obtidos por esta análise. A demora ocorrida entre a coleta de dados e posterior análise pode comprometer a usabilidade e veracidade dos resultados.

A arquitetura é fundamentada nas premissas:

- utilização de filtros para a análise do tráfego de rede de pronta execução, ou seja, que não necessitem de históricos de longa duração;
- não utilização de métodos baseados em *Deep Packet Inspection* (DPI) [Chen et al. 2009b], evitando assim o retardo no envio de pacotes pela rede e a sobrecarga com o armazenamento de *payloads*;

- não utilização de métodos baseados em assinaturas, como valores de portas mostrados em [Aviv and Haeberlen 2011], tais métodos limitam-se a encontrar *bots* já conhecidas e não a descobrir novas.

Ademais, para atingir este objetivo, a arquitetura foi dividida em duas fases distintas (como proposto em trabalhos anteriores [Zhang et al. 2011, Liu et al. 2010]): detecção de nós com aplicações P2P ativas (fase 1) e detecção de nós com *bots* P2P ativas (fase 2). Desta forma é obtida uma maior otimização do processo, devido a uma redução do universo a ser analisado na segunda fase. Outro argumento que corrobora a divisão do processo em fases, é o fato de *bots* P2P serem inerentemente aplicações P2P como mostrado em [Wang et al. 2009, Silva et al. 2012], sendo assim suscetíveis a muitas técnicas de detecção destinadas a estas. A Fig. 1 mostra a estrutura da arquitetura proposta.

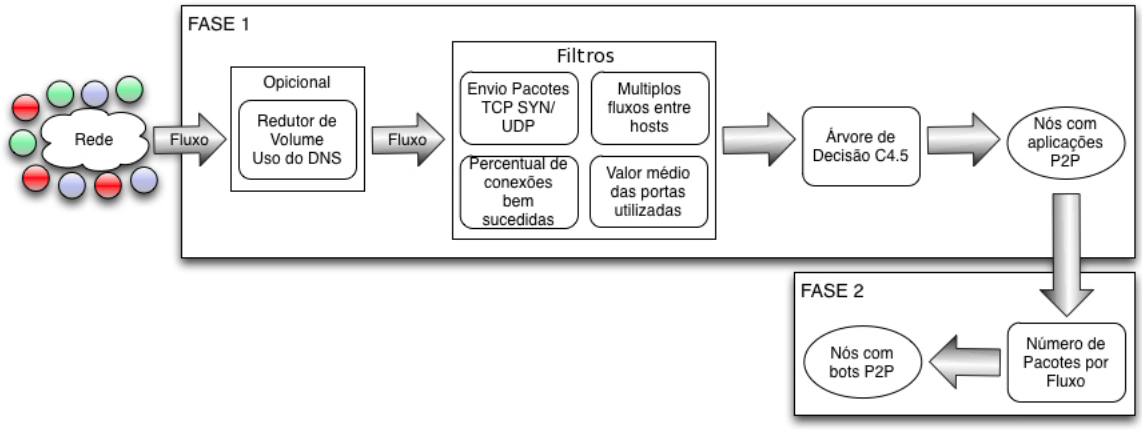


**Figure 1. Arquitetura Proposta para Detecção Online de *Bots* P2P**

### 3.1. Fase 1: Detecção de Aplicações P2P

Nesta fase foram utilizados filtros capazes de detectar características de rede inerentes a aplicações P2P por meio da análise dos fluxos gerados pelos nós monitorados.

Por fluxo de rede entende-se como uma tupla identificada pelo seguinte conjunto de dados: IP de origem, IP de destino, porta de origem, porta de destino e protocolo (equação 1).

$$Fluxo = (IP_{src}, IP_{dst}, Port_{src}, Port_{dst}, Proto) \quad (1)$$

O fluxo é considerado finalizado, quando um pacote TCP RST ou TCP FYN é enviado, ou quando nenhum pacote é enviado por um período de tempo pré-determinado. A utilização dos fluxos visa a otimização no uso do espaço de armazenamento de dados.

O uso de uma arquitetura modular nesta fase é importante devido a flexibilidade gerada, permitindo que novos filtros possam ser adaptados a arquitetura ou vir a substituir os considerados desnecessários ou ineficientes em determinadas redes.

Devido a necessidade de todo o fluxo de dados ser analisado é aconselhável que o agente responsável pela coleta dos dados e produção dos fluxos esteja localizado na entrada / saída da rede monitorada.

### 3.1.1. Filtros de Detecção P2P

Para a compor a arquitetura foi feito um levantamento de diversos filtros empregados e validados em trabalhos anteriores. Ademais, a fim de prover uma exemplificação do comportamentos das redes P2P em relação a esses filtros, traces de duas redes P2P legítimas (*Bittorrent* e *Gnutella*), de uma rede denominada doméstica, livre de aplicações P2P, e de uma máquina contendo a bot P2P Nugache tiveram a característica em estudo mensurada. As redes *Bittorrent* , *Gnutella* e a rede doméstica foram montadas para o experimento deste trabalho, os *traces* da bot Nugache foram retirados do repositório *Open Packet*.

Para a escolha dos filtros adotados nesta fase, foram verificados os seguintes critérios:

- *A compatibilidade do filtro com bots P2P:* como mencionado anteriormente, *bots* P2P são um tipo de aplicação P2P, mas nem todas as características presentes nas redes P2P legítimas estão presentes em *botnets* P2P. Algumas características, como as relacionadas a transferência de grandes volumes de arquivos, muitas vezes não são observadas nessas redes maliciosas.
- *A utilização de filtros que cubram os mais diversos protocolos P2P, bem como bots P2P:* diversas implementações de redes P2P são utilizadas, diferenças variam desde a política de escolha das portas para a comunicação até a estrutura adotada pela rede (redes estruturadas ou não estruturadas). Desta forma, certos módulos que detectam algumas implementações podem não detectar outras. Assim, é importante que esses sejam escolhidos de modo a encontrar os protocolos mais distintos possíveis, ou que se fundamentem em premissas básicas para do funcionamento de qualquer rede P2P.
- *Baixas taxas de falso positivos:* apesar da arquitetura ser dividida em duas fases, o que possibilita que nós identificados erroneamente como detentores de aplicações P2P sejam descartados na segunda fase, baixos índices de falsos positivos são importantes para o desempenho da arquitetura.
- *Valores limites bem definidos:* os valores limites dos filtros que distinguem aplicações P2P das não P2P devem ser bem definidos, valores muito próximos entre os dois grupos podem acarretar em falsos positivos e negativos.

Abaixo são listados os quatro filtros escolhidos para compor a primeira fase da arquitetura proposta (Fig. 1), bem como uma descrição sucinta e os valores obtidos para cada um dos traces utilizados nos testes.

**Taxa de envio de pacotes TCP SYN / UDP:** aplicações P2P necessitam conectar-se a outros *peers* a fim de manter a sua rede funcional. Entretanto, sabe-se que os *peers* em sua grande maioria são computadores normais (e não servidores) que podem possuir endereços IPs dinâmicos ou mesmo podem estar atrás de *firewalls* ou NAT, impedindo assim que a conexão seja efetuada com sucesso. Almejando mitigar tal fato, os nós membros das redes P2P tentam efetuar conexões com diversas máquinas a fim de aumentar o número de sucessos nas conexões e desta forma o número de *peers* a

que estão conectados, o que acarretará em uma maior eficiência na busca e propagação de dados [Chen et al. 2009a]. Este comportamento pode ser medido através da quantidade de fluxos TCP e UDP gerados em um determinado período de tempo. Os traces com aplicações P2P apresentaram um valor médio de 15 e 18 fluxos/seg respectivamente para a aplicação *Bittorrent* e *Gnutella*, porém o trace retirado da rede doméstica livre de aplicações P2P apresentou uma taxa de 2 fluxos/seg. A máquina contendo a bot Nugache apresentou um valor de 13 fluxos/seg com o protocolo UDP.

**Valor médio do número das portas utilizadas:** segundo [Ulliac and Ghita 2010], os protocolos P2P atuais com o intuito de evitar técnicas de detecção baseada em portas conhecidas, utilizam portas aleatórias para efetuar a comunicação entre os *peers*. Como existem 65.536 portas em uma máquina, tem-se uma probabilidade de aproximadamente 0,0015% de uma porta ser escolhida de forma aleatória, por consequência, temos aproximadamente 98.5% de chance de uma porta acima de 1024 (limite máximo das principais portas conhecidas) ser escolhida para uso. O resultado desse procedimento é um valor médio elevado para o número das portas em máquinas possuídoras de aplicações P2P. A medição deste comportamento nos traces *Bittorrent* e *Gnutella* apresentaram um valor médio de 21.027,66 e 6.411,90 respectivamente, enquanto na rede doméstica este valor foi de 753,43 e para a máquina portadora da bot Nugache foi de 20.232.

**Percentual de conexões bem sucedidas:** *peers* segundo [Liu et al. 2009], apesar de possuírem as mesmas atribuições, muitas vezes possuem características distintas (largura de banda, sistema operacional, parâmetros da rede,...), sendo que qualquer alteração em sua configuração pode alterar a conectividade deste com a rede P2P. Ademais, em sua maioria os *peers* não são máquinas servidoras, permanecendo desligados ou com perda de conectividade em alguns momentos. Desta forma, para manter a rede P2P em funcionamento, um *peer* deve tentar se conectar com o máximo de nós possíveis a fim de obter um número mínimo de conexões bem sucedidas. Assim, nós P2P possuem uma taxa de conexões bem sucedidas inferior a de máquinas que só utilizam aplicações com a topologia Cliente / Servidor. As medições feitas nos traces retornaram para a rede *Bittorrent* um percentual de conexões bem sucedidas de 78,3% , para a rede *Gnutella*, 42,38%, para a rede doméstica 89,4% e para a bot 45,9%.

**Múltiplos fluxos entre nós:** operações que envolvam protocolos UDP/TCP possuem a escolha aleatória de pelo menos uma porta (de origem ou de destino), portanto, não é comum que fluxos distintos possuam um mesmo conjunto de atributos identificadores ($IP_{src}$, $IP_{dst}$, $Port_{src}$,$Port_{dst}$, Protocolo). Entretanto, isto acontece com aplicações P2P, já que ambos os *peers* dedicam uma porta para a troca de mensagens e transferência de dados. Esta é a base da heurística proposta em [Marcell Perényi and Molnár 2006]: a existência de fluxos idênticos é um forte indício de comunicação entre aplicações P2P. Ambas as redes P2P e a bot tiveram o comportamento detectado, porém na rede doméstica não houve registro desta característica.

Opcionalmente, pode-se adotar na arquitetura um filtro redutor de volume definido pela análise do uso do DNS (primeiro bloco da Fig. 1). Para redes que trafegam grandes volumes de informação esse filtro pode elevar a eficiência de toda a arquitetura eliminando previamente fluxos de redes não pertencentes a aplicações P2P. Em [Zhang et al. 2011, Silva and Salles 2012] o uso das requisições ao DNS foi utilizado

como um redutor de volume de tráfego a ser analisado, isto é justificado pelo fato de que a comunicação P2P é feita diretamente através do endereço IP dos *peers* membros da rede, contidas nas *peer lists*. O percentual de *peers* com domínios registrados em DNS é menor que 0,5% (normalmente apenas *super peers* ou servidores web que armazenam *peer lists* utilizam registros em domínios), como mostrado em [Silva and Salles 2012]. Através da análise de traces de redes contendo aplicações P2P e aplicações não P2P, foi mostrado que é possível obter uma redução de volume superior a 25% do tráfego a ser analisado. Somente 0,51% das requisições oriundas da rede *Bittorrent* foram direcionadas a um DNS, para as rede *Gnutella* e para a bot Nugache não houve qualquer requisição ao DNS observada e para a rede doméstica este percentual foi de 33,4%.

Após a análise dos fluxos de rede, todos os nós que possuírem características compatíveis com um ou mais filtros terão seus endereços IPs e características armazenadas em banco de dados para posterior análise. O próximo passo, na fase 1 ainda, é o emprego de uma árvore de decisão a fim de gerar como resultado uma lista de nós possuidores de aplicações P2P, essa informação é então passada para a segunda fase da arquitetura (vide Fig. 1). O uso da árvore de decisão é fundamentado no fato de que nem todos os nós que apresentam uma ou mais características buscadas, necessariamente possuem aplicações P2P ativas. Para a execução da árvore de decisão foi utilizada a aplicação WEKA, devenvolvida pela Universidade de Waikato, com a implementação J48 (implementação do algorítimo C4.5 [Quinlan 1993]).

### 3.2. Fase 2: Detecção de Bot P2P

Nesta fase é feita a distinção entre nós com aplicações P2P legítimas e nós com *bots* P2P. A separação fundamenta-se na principal diferença entre elas: a sua funcionalidade.

Aplicações P2P legítimas são utilizadas principalmente para disseminação de conteúdo entre os *peers* membros de suas redes. Por outro lado, *bots* P2P trafegam apenas informações simples como comandos enviados pelo *botmaster* para os membros de sua rede, ou *strings* contendo informações roubadas das máquinas onde estão instaladas (senhas, número de cartões de crédito, entre outras), tais informações limitam-se a poucos bytes de tamanho e muitas vezes conseguem ser transmitidas em poucos pacotes IP.

É nesta diferença que reside a principal forma para distinguir os dois tipos de rede. Devido ao maior tamanho dos dados trafegados nas redes P2P legítimas, essas possuem pacotes com maior tamanho médio, enquanto *bots* P2P tendem a utilizar pacotes com um tamanho médios muito inferior. [Liao and Chang 2010] mensurou em seu trabalho o tamanho dos pacotes enviados por diversos protocolos, encontrando um valor para aplicações P2P superior a 1.300 bytes e para *bots* P2P um tamanho inferior a 300 bytes.

Uma extensão desta característica é proposta neste artigo para a diferenciação das redes: o número médio de pacotes por fluxo de dados. Aplicações P2P legítimas por trafegarem grande volumes de dados tendem a ter uma quantidade muito maior de pacotes por fluxo de dados que as *bots* P2P.

Com o intuito de facilitar a visualização do tamanho médio dos pacotes e quantidade de pacotes por fluxo, foram medidos ambos atributos para redes com diferentes características como mostrado na Fig.2. Os dados usados foram obtidos de traces contidos no repositório *Open Packet* (*www.openpacket.org*) (tráfego HTTP, *Bot Conficker*),

*CRAWDAD* (*crawdad.cs.dartmouth.edu/*) (*Games Online*), ou gerados em experimentos para este trabalho (*Bittorrent, Gnutella* e *Bot Nugache*)

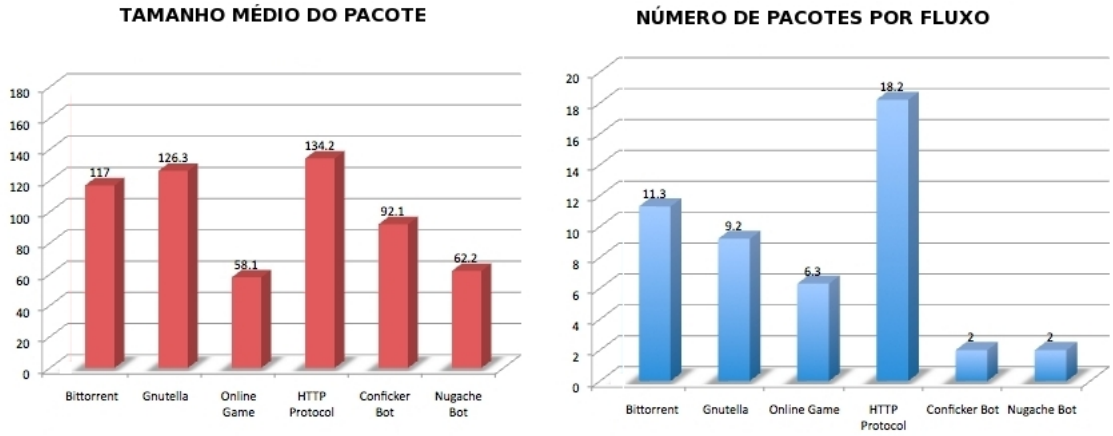


**Figure 2. Tamanho médio de pacotes em bytes (em vermelho) e número médio de pacotes por fluxo (em azul)**

Como observado na Fig. 2, as bots P2P (*Conficker* e *Nugache*) possuem um tamanho de pacote médio inferior ao de outras aplicações, na medição feita para este artigo ambas apresentaram tamanho médio inferior a 100 *bytes*, enquanto aplicações P2P legítimas apresentaram um tamanho médio de pacote superior a 120 *bytes*. Porém, foi constatado que aplicações que envolvem jogos *online*, costumam apresentar tamanho médio de pacote inferior ao das *bots* P2P , como apresentado em [Liao and Chang 2010] e verificado neste artigo. Apesar do comportamento dos jogos *online* serem distintos do comportamento das aplicações P2P, o que os descartaria na primeira fase da arquitetura, uma eventual escolha de nós contendo este tipo de aplicação para serem analisados na fase 2, ocasionaria um aumento dos índices de falsos positivos, caso este parâmetro fosse adotado para a corrente fase.

Desta forma, com o objetivo de diminuir as chances da ocorrência deste tipo de falso positivo, para a fase 2 é proposta uma extensão da característica acima: o número médio de pacotes por fluxo. Como observado na Fig. 2, o número médio de pacotes por fluxo das *bots* P2P é inferior ao de qualquer outra aplicação, inclusive as que envolvem jogos *online*, pois além das *bots* possuírem um tráfego de rede bastante inferior ao das outras aplicações, estas almejam permanecer ao máximo sem serem detectadas nas máquinas hospedeiras, característica denominada *stealthiness*, que os jogos *online* não possuem.

Outro fator que corrobora para a utilização do número médio de pacotes por fluxo em detrimento do tamanho médio dos pacotes é a facilidade para se estipular um valor limite entre aplicações P2P legítimas e as *bots*. Neste caso, a determinação de um valor limite para a distinção de *bots* P2P das demais aplicações se torna uma tarefa mais fácil, podendo ser determinado com uma boa margem de segurança a fim de evitar potenciais erros. Para este artigo é adotado como fator de corte a distância média entre o maior tamanho encontrado para o pacote oriundo de uma bot e o menor tamanho encontrado para pacotes de demais aplicações, ou seja, para o gráfico da Fig. 2 o valor será 4. Vale ainda ressaltar que este valor, aqui adotado, pode ser alterado a depender do tipo de aplicação em uso na rede em que a arquitetura for empregada.

Sendo assim, na fase 2 a distinção é implementada pelo número médio de pacotes por fluxo, onde valores superiores a 4 são considerados de redes P2P legítimas.

Após a análise dos filtros adotados na fase 1 e na fase 2, é possível verificar que o tempo de execução da metodologia é limitado pelo tempo de captura do *trace* da rede a ser analisada, como será verificado durante a validação da arquitetura. Fato este que ratifica o uso do termo *online* para a metodologia.

## 4. Validação da Arquitetura

Para a realização de testes com a arquitetura proposta, dois cenários ($C_1$ e $C_2$) foram montados a partir de traces obtidos dos repositórios CRAWDAD , OpenPacket e DARPA (*www.ll.mit.edu*), além de traces gerados artificialmente pela aplicação *Rubot* [Lee 2009].

O *Rubot* consiste de um *framework* para a emulação e análise de *botnets*. Desenvolvido na linguagem *Ruby*, permite a rápida construção de *botnets* integrando diferentes funcionalidades que se deseja estudar. Sua principal vantagem, diante das demais aplicações voltadas para o estudos desses *malwares*, é a possibilidade de se analisar todos as características da *bot*, desde o seu *payload* até os *traces* de rede gerados por sua comunicação com outros membros.

Para este trabalho foram usados módulos prontos do *Rubot* que emulam a *botnet Nugache*. Utilizando seu *framework* foi construída uma rede contendo quinze membros. Apesar de sua complexidade, o *Rubot* não emula a fase de *bootstrap*, fase em que a bot busca outros *peers* para se conectar a rede, o que inviabiliza o uso de um dos filtros que busca por esta característica. Como a ausência desta fase na *botnet Nugache* é irreal, foi inserido artificialmente em seus *traces* pacotes retirados de *traces* reais da *bot Nugache*, permitindo assim uma maior similaridade entre o *trace* gerado pelo *Rubot* e um *trace* real.

Para a obtenção dos *traces* de rede foi utilizado a aplicação TCPDUMP e para sua posterior conversão em fluxo adotou-se o ARGUS (*Audit Record Generation and Utilization System*).

O cenário $C_1$ contém: 6 nós com a bot P2P *Nugache* gerada pelo *Rubot*, 4 nós executando a aplicação *Bittorrent*, e 39 nós sem nenhum tipo de aplicação semelhante a aplicações P2P retirados da rede DARPA.

O cenário $C_2$ não possui nenhuma bot P2P em seus *traces*, sendo constituído por: *traces* de redes que executam a aplicação P2P *Gnutella*(4 nós), P2P *Bittorrent* (2 nós), *traces* de 39 nós isentos de qualquer aplicação P2P oriundos do DARPA e de uma rede doméstica montada para este experimento contendo 6 nós. Ambos os cenários apresentam uma duração de quarenta minutos.

Antes de inciar o trabalho de validação da arquitetura proposta nos cenários gerados, uma amostra contendo 10 minutos de *trace* foi retirada do cenário $C_1$ para ser utilizada como treinamento para a geração da árvore de decisão da fase 1. Esta árvore foi capaz de classificar corretamente 95,58% dos nós contidos na amostra.

### 4.1. Análise do Cenário $C_1$

O cenário $C_1$ ao ser analisado na fase 1 da arquitetura apresentou uma acurácia de 89,79% e percentual de falso positivo para os nós contendo aplicações P2P e não contendo tais

aplicações de 10,3% e 10%, respectivamente. Em números absolutos é possível observar que entre os 10 nós com aplicações P2P ativas, apenas 1 nó, possuídor de uma *bot Nugache*, foi classificado erroneamente. Entre os 39 nós sem aplicações P2P ativas, 5 nós foram classificados de forma errada (Fig. 3).

Através da análise de cada filtro empregado na fase 1, observa-se que os filtros que apresentaram maior quantidade de nós com aplicações P2P classificados com características adversas a estas aplicações foram os que analisam a taxa de envio de pacotes TCP SYN / UDP e o valor médio das portas utilizadas para a comunicação. Isto é justificado pelo fato de um dos traces da *bot Nugache* utilizados não possuir o procedimento de *bootstrap* e utilizar a porta 8 para sua comunicação, porta esta somente utilizada em implementações antigas desta *bot*.


**Figure 3. Gráficos com a classificação dos nós do cenário $C_1$ em P2P (nós à esquerda) e não P2P (nós à direita).**

Após a análise executada pela fase 1 os nós identificados como aplicações P2P, ou seja, eleitos para serem analisados na fase 2, tiveram o número médio de pacotes por fluxo apurados, conforme mostrado na tabela 1.

| **Nós classificados como possuídores de aplicações P2P Ativas pela fase 1** | | | |
|---|---|---|---|
| | **Bots P2P** | **Outras App P2P** | **App não P2P** |
| **Número de nós** | 5 | 4 | 5 |
| **Número médio de pacotes por fluxo** | 2,1 | 67,17 | 14,13 |
| **Quantidade de nós classificados erroneamente na fase 2** | 0 | 2 | 2 |
| **Número médio de pacotes por fluxo dos nós classificados erroneamente na fase 2** | - | 1 | 2,3 |

**Table 1. Valores obtidos após analise pela fase 2**

Todos os nós possuidores de bots P2P analisados pela fase 2 foram classificados corretamente. Porém, entre os nós com aplicações P2P legítimas, 2 por possuírem número médio de pacotes por fluxo inferior a 4, foram considerados como *bots* P2P. Entre os nós não possuidores de aplicações P2P situação semelhante foi observada, 4 nós foram erroneamente classificados como possuidores de *bots* P2P.

Desta forma, das 6 *bots* P2P existentes no cenário $C_1$ apenas 1 não foi detectada por ter sido erroneamente classificado durante a fase 1, e para os 43 nós não possuidores de *bots* P2P (nós com aplicações P2P legítimas e sem aplicações P2P) apenas 4 nós foram mal classificados (tabela 2). Assim, encontramos para a arquitetura um percentual de falsos positivos de 9,3% e de falsos negativos de 16,7%. O tempo de processamento gasto para as fases 1 e 2 foi de 5,43 segundos.

| RESULTADO PARA O CENÁRIO 1 | | | |
|---|---|---|---|
| | **Quantidade** | **Classificadas Erroneamente** | **Percentual de Erros (%)** |
| **Bots P2P** | 6 | 1 | 16,7 |
| **Outras Aplicações** | 43 | 4 | 9,3 |

**Table 2. Valores obtidos para o Cenário $C_1$**

## 4.2. Análise do Cenário $C_2$

O cenário $C_2$ foi gerado sem que houvesse nenhuma bot P2P presente, apenas aplicações P2P legítimas e nós sem nenhum tipo de aplicação deste gênero. Após a análise pela fase 1 foi obtido um percentual de acerto de 82,35% e uma taxa de falsos positivos para nós com aplicações P2P e nós sem este tipo de aplicação de 20% e 0%, respectivamente. Em números absolutos observa-se que entre os 10 nós com aplicações P2P ativas nenhum foi classificado como não P2P, apesar de 1 nó referente a aplicação *Gnutella* apresentar um valor baixo para a porta de comunicação. E entre os 45 nós livres de aplicações deste gênero, 9 foram classificados erroneamente (Fig. 4). Ao verificar-se o motivo do elevado percentual de erros obtido na classificação dos nós livres de aplicações P2P, observa-se a utilização de portas com valores elevados para a comunicação.


**Figure 4. Gráficos com a classificação dos nós do cenário $C_2$ em P2P (nós a esquerda) e não P2P(nós a direita).**

Após a análise executada pela fase 1 os nós identificados como aplicações P2P tiveram o número médio de pacotes por fluxo extraídos para serem analisados na fase seguinte. A tabela 3 indica os valores obtidos nesta fase.

No cenário $C_2$ não existem *bots* P2P, somente nós com aplicações P2P legítimas (6 nós), dentre os quais 1 foi classificado de forma errada, como bot P2P; e nós sem aplicações P2P (45 nós), dentre os quais 9 foram considerados possuidores de aplicações

| Nós classificados como possuídores de aplicações P2P Ativas pela fase 1 | | | |
|---|---|---|---|
| | **Bots P2P** | **Outras App P2P** | **App não P2P** |
| **Número de nós** | 0 | 6 | 9 |
| **Número médio de pacotes por fluxo** | - | 19,82 | 21 |
| **Quantidade de nós classificados erroneamente na fase 2** | - | 1 | 2 |
| **Número médio de pacotes por fluxo dos nós classificados erroneamente na fase 2** | - | 3,1 | 3,1 |

**Table 3. Valores obtidos após analise pela fase 2**

P2P na primeira fase e 2 foram classificados como bots P2P na segunda. Desta forma, para este cenário foi encontrado um percentual de falsos positivos de 5,9% (tabela 4). O tempo de processamento gasto para a execução das fases 1 e 2 foi de 6,89 segundos.

| RESULTADO PARA O CENÁRIO 2 | | | |
|---|---|---|---|
| | **Quantidade** | **Classificadas Erroneamente** | **Percentual de Erros (%)** |
| **Bots P2P** | 0 | - | - |
| **Outras Aplicações** | 51 | 3 | 5,9 |

**Table 4. Valores obtidos para o Cenário $C_2$**

## 5. Considerações Finais

Como plataforma para crimes cibernéticos, *botnets* são a principal ameaça à segurança na internet. Neste artigo foi apresentada uma proposta de arquitetura para a detecção *online* de *bots* P2P. O modelo proposto é dividido em duas fases buscando assim uma otimização dos recursos. Na primeira fase, filtros de rápida execução, que não necessitam analisar longos históricos de redes, aliados a uma árvore de decisão são utilizados com o intuito de diferenciar aplicações P2P das demais. Posteriormente, em uma segunda fase, é realizada a distinção entre as aplicações P2P legítimas e as *bots* P2P através da análise do número médio de pacotes por fluxo de rede.

Com o objetivo de justificar e implementar a primeira fase da arquitetura, foi feito um estudo do funcionamento dos protocolos P2P e como as bots P2P os utilizam em proveito de suas redes maliciosas. Foi realizado também um levantamento das mais distintas técnicas utilizadas para a detecção de tráfego P2P de modo a empregá-las para detectar não apenas as redes P2P legítimas mas também as *bots*.

Para a implementação da segunda fase, foram observados os fins a que se destinam as redes P2P, legítimas ou não. Através de uma análise empírica foi mostrado que redes P2P legítimas tendem a gerar maior número de pacotes por fluxo que *bots* P2P, pois essas tendem a trafegar grandes quantidades de informação enquanto *bots* P2P limitam-se a simples trocas de mensagens.

Visando uma validação o mais eficiente possível foram elaborados dois cenários através da junção de diversos traces de rede com características distintas. No primeiro

cenário, traces da *bot* P2P *Nugache* foram implantados juntamente com traces de aplicações P2P legítimas. Para esse cenário, índices de falsos positivos e negativos de 9,3% e 16,7% respectivamente foram obtidos. No cenário 2, somente aplicações P2P legitimas foram implantadas, gerando um índice de falsos positivos de 5,9%.

Como visto, um diferencial proposto na arquitetura em relação a outros trabalhos com objetivos semelhantes, é a não utilização de métodos baseados em assinatura e o tempo demandado para a execução das análises (inferior a 7 segundos nos cenários avaliados), fazendo com que o fator limitante de tempo para a execução seja a duração da coleta dos dados a serem analisados e não a análise em si, desta forma podendo ser classificada como *online*. Porém, com o objetivo de diminuir ainda mais este tempo e otimizar os valores de acurácia e falsos positivos encontrados é necessário um estudo da variação da eficiência pelo tamanho do *trace* de rede em análise. Estudos semelhantes já foram feitos, como em [Karagiannis et al. 2004b] para a detecção de aplicações P2P, mas para bots P2P permanecem inéditos. Ademais, uma análise da arquitetura em redes maiores se faz necessário visando verificar sua escalabilidade e obter valores de falsos positivos e negativos mais apurados.

## References

(2012). India becomes the king of the spammers, stealing america's crown. http://nakedsecurity.sophos.com/2012/04/23/india-becomes-the-king-of-the-spammers-stealing-americas-crown/.

Aviv, A. J. and Haeberlen, A. (2011). Challenges in experimenting with botnet detection systems. In *Proceedings of the 4th USENIX Workshop on Cyber Security Experimentation and Test (CSET'11)*.

Chang, S. and Daniels, T. E. (2009). P2P botnet detection using behavior clustering & statistical tests. In *Proceedings of the 2nd ACM workshop on Security and artificial intelligence*, AISec '09, pages 23–30, New York, NY, USA. ACM.

Chen, F., Wang, M., Fu, Y., and Zeng, J. (2009a). New detection of peer-to-peer controlled bots on the host. In *Wireless Communications, Networking and Mobile Computing, 2009. WiCom '09. 5th International Conference on*, pages 1 –4.

Chen, H., Hu, Z., Ye, Z., and Liu, W. (2009b). A new model for p2p traffic identification based on dpi and dfi. In *Information Engineering and Computer Science, 2009. ICIECS 2009. International Conference on*, pages 1 –3.

Coskun, B., Dietrich, S., and Memon, N. (2010). Friends of an enemy: identifying local members of peer-to-peer botnets using mutual contacts. In *Proceedings of the 26th Annual Computer Security Applications Conference*, ACSAC 10, New York, NY, USA. ACM.

Karagiannis, T., Broido, A., Brownlee, N., Claffy, K., and Faloutsos, M. (2004a). Is p2p dying or just hiding? [p2p traffic measurement]. In *Global Telecommunications Conference, 2004. GLOBECOM '04. IEEE*, volume 3, pages 1532 – 1538 Vol.3.

Karagiannis, T., Broido, A., Faloutsos, M., and claffy, K. (2004b). Transport layer identification of p2p traffic. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, IMC '04, pages 121–134, New York, NY, USA. ACM.

Lee, C. P. (2009). *Framework for botnet emulation and analysis*. PhD thesis, Atlanta, GA, USA. AAI3364236.

Liao, W.-H. and Chang, C.-C. (2010). Peer to peer botnet detection using data mining scheme. In *Internet Technology and Applications, 2010 International Conference on*, pages 1 –4.

Liu, D., Li, Y., Hu, Y., and Liang, Z. (2010). A p2p-botnet detection model and algorithms based on network streams analysis. In *Future Information Technology and Management Engineering (FITME), 2010 International Conference on*, volume 1, pages 55 –58.

Liu, F., Li, Z., and Nie, Q. (2009). A new method of p2p traffic identification based on support vector machine at the host level. In *Proceedings of the 2009 International Conference on Information Technology and Computer Science - Volume 02*, ITCS '09, pages 579–582, Washington, DC, USA. IEEE Computer Society.

Marcell Perényi, Trang Dinh Dang, A. G. and Molnár, S. (2006). Identification and analysis of peer-to-peer traffic. In *Journal of Communications*.

Powers, D. M. W. (2007). Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation. Technical Report SIE-07-001, School of Informatics and Engineering, Flinders University, Adelaide, Australia.

Quinlan, J. R. (1993). *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Silva, R. M. P. and Salles, R. M. (2012). Methodology for detection and restraint of P2P applications in the network. volume 7336 of *Lecture Notes in Computer Science*, chapter 24, pages 326–339. Springer Berlin / Heidelberg, Berlin, Heidelberg.

Silva, S. S., Silva, R. M., Pinto, R. C., and Salles, R. M. (2012). Botnets: a survey. *Computer Networks*.

Spognardi, A., Lucarelli, A., and Di Pietro, R. (2005). A methodology for p2p file-sharing traffic detection. In *Hot Topics in Peer-to-Peer Systems, 2005. HOT-P2P 2005. Second International Workshop on*, pages 52 – 61.

Ulliac, A. and Ghita, B. V. (2010). Non-intrusive identification of peer-to-peer traffic. In *Proceedings of the 2010 Third International Conference on Communication Theory, Reliability, and Quality of Service*, CTRQ '10, pages 116–121, Washington, DC, USA. IEEE Computer Society.

Wang, P., Wu, L., Aslam, B., and Zou, C. (2009). A systematic study on Peer-to-Peer botnets. In *Computer Communications and Networks, 2009. ICCCN 2009. Proceedings of 18th Internatonal Conference on*, pages 1 –8.

Zhang, J., Perdisci, R., Lee, W., Sarfraz, U., and Luo, X. (2011). Detecting stealthy p2p botnets using statistical traffic fingerprints. In *Dependable Systems Networks (DSN), 2011 IEEE/IFIP 41st International Conference on*, pages 121 –132.