# Uma arquitetura baseada em botnet para aplicações voltadas a avaliação de aspectos de segurança da informação*

**Matheus Santos[2], Davidson Boccardo[1], Raphael Machado[1], Bruno Guimarães[1], Rafael Soares[1]**

[1]Clavis Segurança da Informação, Rio de Janeiro, Brasil

[2]Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brasil

{davidson,raphael,bruno,rafael}@clavis.com.br, msmartins.ufrj@gmail.com

***Abstract.*** *The rapid growth of the concept of Internet of Things is proportionally linked to problems of security information in data surfing the World Wide Web, rectifying every day more the importance on this subject. Basically, in information security, there are three pillars that guarantee the protection of the information: integrity, confidentiality and availability. The guarantee of these three bases is not trivial, but it is important for various sectors such as industry, services and economy. In this article, we present a distributed architecture, based on the botnet concept that is able to assess these three bases; we describe its main features and functionalities aimed at solving problems in cyber security; and we present a case study, in shapes of Denial of Service tests, using the proposed architecture to characterize the network resistance and the anti-DDoS systems employed on e-commerce Brazilian companies.*

***Resumo.*** *O rápido crescimento do conceito de Internet das Coisas está proporcionalmente atrelado em problemas de segurança da informação de dados que navegam na rede mundial de computadores, retificando a cada dia mais a importância deste assunto. Basicamente, em segurança da informação, existem 3 pilares que garantem a proteção destes dados: integridade, confidencialidade e disponibilidade. A garantia dessas três bases não é trivial apesar de ser importantíssima para diversos setores, tais como indústria, serviços e economia. Neste artigo, apresentamos uma arquitetura distribuída, baseada no conceito de botnet, na qual possibilita o desenvolvimento de aplicações distribuídas capazes de avaliar essas três bases; descrevemos suas principais características e funcionalidades destinadas a resolução de problemas em segurança cibernética. Também apresentamos um estudo de caso, na categoria de testes de indisponibilidade, utilizando a arquitetura proposta em uma ferramenta de simulação de ataques de negação de serviço para a caracterização da resistência de redes e avaliação de sistemas anti-DDoS utilizados por empresas brasileiras de comércio eletrônico.*

## 1. Introdução

Sistemas de informação estão se tornando cada vez mais importantes a medida com que informações críticas e sensíveis estão sendo manipuladas por eles. A complexidade atual desses

* *Trabalho apoiado pelo CNPq (chamada RHAE), FINEP (chamada TI Maior) e Faperj (Tecnova)*

sistemas e o surgimento do conceito de Internet das Coisas, na qual elementos do cotidiano da sociedade estão sendo cada vez mais interconectados com a Internet, faz com que surjam novos desafios relacionados a segurança cibernética [HP 2014]. Estes desafios referem-se a garantir que a criação, manuseio, transferência e armazenamento dessas informações não representem um fator de risco para o detentor delas.

Basicamente, três propriedades são desejáveis para que informações de um sistema não representem risco a uma empresa ou organização. Estas propriedades referem-se aos três pilares da Segurança da Informação: confidencialidade, que representa a garantia de que a informação é acessível somente por pessoas ou serviços autorizados; integridade, que indica a exatidão com que os dados processados são salvos e que não sofreram nenhuma alteração maliciosa ou não autorizada; e a disponibilidade, que é a característica responsável pela garantia do serviço estar disponível a todo momento ou ao momento em que for solicitado.

Dada a vasta quantidade destes sistemas e seu potencial de crescimento devido a emergente Internet das Coisas, torna-se necessário o desenvolvimento de novas metodologias e ferramental a fim de avaliar se os pilares da Segurança da Informação estão sendo atendidos. Uma das formas de realizar a avaliação desses consiste na utilização de aplicações distribuídas, na qual poder de processamento e largura de banda são escaláveis. Uma aplicação distribuída pode ser definida como qualquer tipo de computação paralela realizada por múltiplos sistemas computacionais, conectados a uma rede, com um objetivo de concluir uma tarefa em comum.

No presente trabalho propomos uma a arquitetura para aplicações distribuídas baseada no conceito de *botnet* capaz de realizar diversas tarefas voltadas a segurança cibernética. Este trabalho utiliza dessa arquitetura para prover serviços voltados a avaliação da segurança cibernética, a saber testes de indisponibilidade, varredura e monitoramento contínuo de vulnerabilidades, e processamento paralelo voltado a mineração de dados de segurança. O objetivo dos testes de indisponibilidade é verificar o quão disponível estão as aplicações e serviços oferecidos por uma empresa. A varredura e monitoramento contínuo de vulnerabilidades visam identificar ameaças e vulnerabilidades de uma organização que possam ser exploradas por um atacante. E o processamento paralelo voltado a mineração de dados de segurança permite realizar de forma mais eficiente uma tomada de decisão em sistemas de segurança.

Sendo assim, a arquitetura proposta possibilita o desenvolvimento de aplicações distribuídas que abrangem, de certa forma, os três pilares da segurança da informação com diversas possibilidades de implementação, tratando-se de uma característica muito importante para a área de segurança cibernética. Notadamente, com testes de indisponibilidade, a arquitetura proposta pode avaliar a disponibilidade de aplicações e serviços por meio de geração de um alto tráfego de rede devido a largura de banda escalável. No caso de integridade, podemos utilizar dos chamados ataques de Negação de Serviço Permanente (do inglês, *Permanent Denial of Service*) que podem corromper o *firmware* do sistema-alvo [The Register 2008] comprometendo o sistema inteiro e, consequentemente, causando perda total ou parcial de dados importantes. Já para o caso de confidencialidade, a arquitetura é capaz de simular serviços de testes de invasão e varreduras de vulnerabilidades, que ora encontradas possibilitam o acesso não autorizado de informações por um atacante.

O artigo está organizado da seguinte forma. A seção 2 apresenta o estado da arte dos principais conceitos necessários no que se refere a sistemas baseados em *botnet* e ataques de negação de serviço. A seção 3 contém a proposta da arquitetura distribuída, suas características e implementação. A seção 4 mostra um estudo de caso de teste de indisponibilidade usando a arquitetura proposta em uma aplicação distribuída para simulações de ataques distribuídos de negação de serviço. A seção 5 contém nossas considerações finais.

## 2. Conceitos Gerais

Nesta seção, abordaremos os conceitos gerais que achamos essenciais para o entendimento do nosso trabalho. Descreveremos as principais características de sistemas baseados em *botnet* e definimos os principais conceitos em ataques de negação de serviços, preparando o leitor para a proposta e a simulação realizada.

### 2.1. *Botnet*

Uma *botnet* é uma coleção de máquinas ligadas a uma rede de computadores sendo capazes de se comunicar entre si e realizar tarefas de rede variadas a comando de um controlador. Refere-se a uma coleção de robôs de software, ou *bots*, que executam de forma autônoma e automática. Frequentemente, este termo está associado a fins maliciosos na qual os computadores infectados (*bots*) são chamados de zumbis [WeLiveSecurity 2014]. Notadamente, algumas *botnets* maliciosas podem chegar a ter infectados milhões de usuários [CNET 2013], correspondendo à quantidade de nós que participam atuante.

Uma *botnet* ou um exército de robôs é capaz de prover uma alta largura de banda para explorar serviços de um sistema ou impossibilitar acesso a um determinado serviço de um sistema. Outra capacidade refere-se a possibilidade de lançar tarefas de rede coordenadas em um tempo específico a comando de um controlador. Em geral, as *botnets* podem ser implementadas em servidores IRC [Trac 2015], HTTP [Team Cymru 2008] e, até mesmo, P2P [Dittrich and Dietrich 2008].

Geralmente, as entidades que compõe uma *botnet* são representadas por dois agentes principais: um que será responsável pela transmissão de comandos pelo canal de comunicação, podendo ser chamado de *Master*; e outro que será responsável pela interpretação e execução deste comando, chamado de *Slave* (ou *bot*). Quanto maior a quantidade de *bots* disponíveis para atuação e conectados à *botnet*, maior será a sua largura de banda disponível e poder computacional para realização de tarefas. Este tipo de interação desses agentes é comumente chamada de *Master-Slave*.

### 2.2. Ataques DDoS

Ataques Distribuídos de Negação de Serviço - *DDoS* (do inglês, *Distributed Denial of Service*) são uma classe de ataques cibernéticos que buscam tornar indisponível um sistema computacional, seja este uma aplicação que fornece serviços a usuários legítimos, um *web server* ou, até mesmo, algum outro tipo de sistema computacional que dependa de uma disponibilidade constante. Tais ataques consistem de uma ação coordenada de um grande número de máquinas

controladas por um atacante executando ações rotineiras com objetivo de sobrecarregar um sistema alvo.

A maneira com que um atacante pode atingir seu objetivo é baseada no tipo de ataque que será realizado. A maioria dos ataques são capazes de explorar falhas e fraquezas abusando de especificações de protocolos - podendo não se tratar necessariamente de vulnerabilidades [CWE-400 2014, CWE-410 2014] - causando sobrecarga de requisições que podem levar a indisponibilidade do serviço. Também é importante notar que tais ataques, por não serem baseados em falhas de software, mas sim no uso massivo de recursos do sistema alvo, estão entre os de mais difícil prevenção, detecção e resposta.

Um importante critério de classificação de ataques *DDoS* consiste no protocolo de comunicação e no serviço explorado [Mirkovic and Reiher 2004]. São denominados ataques de infraestrutura aqueles que agem em protocolos de camadas mais baixas, notadamente nas camadas 3 e 4 (rede e transporte). Tais ataques são essencialmente fundamentados no elevadíssimo tráfego, o qual leva à sobrecarga e ao mau-funcionamento de equipamentos de rede (roteadores, *firewalls*, pilha do servidor etc.). Ataques de infraestrutura são predominantes entre os ataques *DDoS*, e a abordagem mais explorada é o *SYN Flood*, seguida por *UDP Flood* e *ICMP Flood* [Radware 2013]. Ataques de aplicação agem em camadas superiores, geralmente na camada de aplicação. Estes ataques são mais sofisticados; por outro lado, demandam menor vazão (volume de dados ou requisições por unidade de tempo) para causar o mesmo efeito. Tais ataques agem diretamente sobre o provedor de serviço (*load balancers*, bancos de dados, servidores de aplicação, etc.). Dentre os ataque *DDoS* em camada de aplicação, os mais frequentes são os que usam o *HTTP GET* [Radware 2013].

## 3. Proposta

Nesta seção, descrevemos a arquitetura de aplicações distribuídas baseada no conceito de *botnet*. Uma *botnet* é estruturada a partir de entidades chamadas de *Master* e *Slave.* O objetivo dessa estruturação é fazer com que um agente *Master* seja capaz de ser o controlador de todos os *Slaves* de forma remota. Esta é a principal característica de uma *botnet*, que podemos denominar de comando e controle unidirecional realizado pelo *Master*.

O *Master* é o agente responsável pela troca de informações com todos os *Slaves* que estiverem conectados a um determinado canal de comunicação. Este canal vai depender da implementação da *botnet,* podendo ser via servidores IRC, HTTP, P2P entre outros. Sua principal função é manter uma hierarquia para envio de comandos para os *Slaves*, com intuito que estes recebam e executem ordens. O *Master* pode enviar comandos para verificar a autenticidade, largura de banda disponível e capacidade de processamento dos *Slaves*; bem como emitir ordens de execução para perpetuar ataques, varredura de vulnerabilidades e ou processamento paralelo de forma coordenada e distribuída.

O *Slave* é o agente responsável pela execução de qualquer tipo de comando enviado pelo *Master*. Sua função é realização de tarefas que demandem alto processamento ou alta largura de banda. A partir do comando emitido pelo *Master*, cada *Slave* irá executar o comando solicitado de maneira distribuída e concorrente, almejando um objetivo em comum. Podemos exemplificar algumas funcionalidades do *Slave*: execução de ataques como por exemplo ataques distribuídos de

negação de serviço; execução de escaneamentos a procura de vulnerabilidades conhecidas; processamento de entradas enviadas pelo *Master* para algum tipo de tomada de decisão como por exemplo, tratamento de tráfego de rede para detecção de anomalias e invasões [Zhao and Li 2011] e etc.

Uma das formas que foram usadas para compor a *botnet* é através de uma proposta na qual clientes podem se registrar neste serviço, obter suas credenciais e "doar" sua largura de banda e ou poder computacional numa forma de comércio entre o contratante e o contratado. Uma vez que os clientes forem contratados para o serviço, ele pode especificar e escolher o tipo de serviço para qual sua máquina participará. Por exemplo, o cliente pode disponibilizar sua máquina para utilização da sua largura de banda em um período de tempo previamente agendado pelo contratante para a participação do mesmo em um teste de *stress* ou para realização de escaneamentos de vulnerabilidades. Essa solução de utilização de clientes com máquinas voluntárias ou contratadas para participação da *botnet* foi extremamente importante, pois é uma solução escalável, uma vez que qualquer usuário da Internet pode estar participando e, menos custosa do que a contratação de serviços terceirizados de virtualização.

Para fins de segurança, é importante notar que esta solução demanda a autenticação das máquinas clientes participantes da *botnet*. A autenticação dos clientes foi desenvolvida utilizando protocolos de criptografia capazes de gerenciar um mecanismo de autenticação para os serviços de segurança. Dessa maneira, adotamos um esquema de autenticação dessas máquinas baseado no protocolo *Kerberos* [MIT 2007]. O objetivo é fazer um *login* de usuários legítimos, que queiram se registrar no serviço, usando uma aplicação web para que estes clientes recebam *tokens* ou *tickets* de validação de usuários.

## 3.1. Implementação

Inicialmente, o *Master* e os *Slaves* estão conectados a um servidor *IRC* (do inglês, *Internet Relay Channel*), afim de usufruírem de um mesmo canal de comunicação para aplicação de comando e controle. Toda a comunicação entre as entidades é feita, estritamente, por tratamento de *strings* dentro do canal do servidor. De certa forma, a razão pela qual foi utilizado o protocolo *IRC*, foi pela facilidade de implementação - mais especificamente pela facilidade de implementação de comandos por *strings* via *chat IRC*. Contudo, a mesma arquitetura da *botnet* poderia ter sido implementada por meio de outros protocolos e, ainda assim, teriam as mesmas especificações, como por exemplo com o protocolo HTTP.

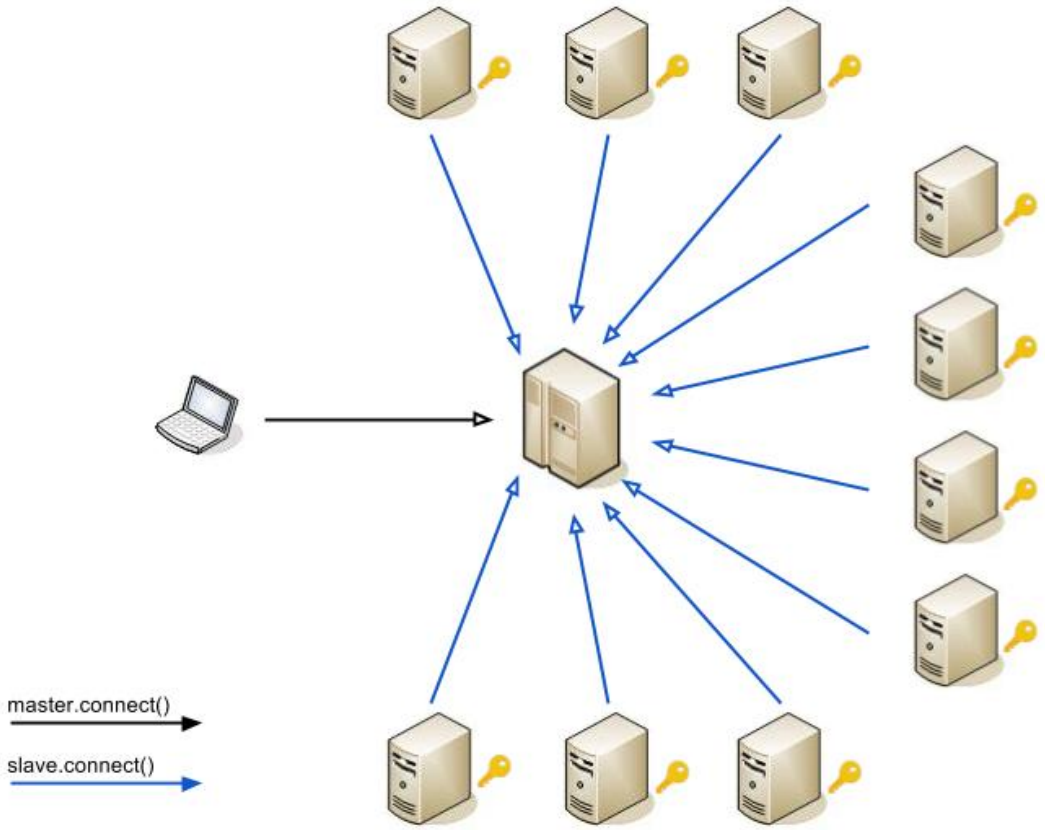


**Figura 1. A arquitetura de uma *botnet.***

Uma vez que um cliente queira participar do serviço, ele precisará se registrar através de um site. Assim, ele poderá ficar a par dos serviços disponíveis que estejam agendados em um determinada data e selecioná-los da maneira que preferir. Se os termos de utilização forem aceitos, a aplicação irá gerar um *ticket* que estará associado a sua máquina através do protocolo *Kerberos*. Dessa maneira, quando a data em que for realizado o serviço chegar, sua máquina - um *Slave* - estará se conectando ao servidor com o seu *ticket* associado. O *Master,* que terá acesso a todos os *tickets* gerados por cada *Slave*, será responsável pela verificação de autenticação de todos os *Slaves* dentro daquele canal. Se a comparação de qualquer *ticket* não for autenticável, será constatada um acesso não autorizado dessa máquina no servidor e as medidas necessárias serão tomadas por ele.

Abaixo, mostramos o seu funcionamento em pseudocódigo.

**Master**
**input** server_ip, port
**1.** master.connect();
**2.** master.startListening();
**3.** **threads#pool** //*pool* de threads responsável pela autenticação de *Slaves*
**4.** wait (ID_1); //espera por um *ticket* a ser autenticado
**5.** **if** (verifyAuth(ticket) == 1)

```
5.1.     post (ID_pool);          //libera Slave do wait
6.     else
6.1      kick (Slave);            //Slave não autenticado
7.     end_thread#pool
8.     thread#2                   //thread responsável pela escrita de comandos
9.     while (master.isListening())
9.1            master.writeComand();    //escreve comandos
10.    end_thread#2

Slave
input  server_ip, port
1.     slave.connect();
2.     slave.startListening();
3.     slave.sendTicket(ID_1,ticket);  //envia seu ticket associado, liberando uma thread da pool
4.     if (wait(ID_pool))              //espera pelo post devido à comparação de tickets
5.            while (slave.isListening())
5.1.       string line = read.line();   //lê strings no chat do canal
5.1.1.     switch(line)                 //executa diferentes tarefas baseado na string lida
5.1.1.1.           case: "X": executeActionX();
5.1.1.2.           case: "Y": executeActionY();
5.1.1.3.           case: "Z": executeActionZ();
6.     else
7.       print("Você não é um usuário legítimo");
```

Na própria autenticação, com a emissão do comando *Master,* é possível separar a atuação dos *Slaves* nos diferentes tipos de serviços a qual foram contratados. Dessa maneira, haveria uma separação de *Slaves* específicos onde somente clientes autenticados possam interpretar determinado comando. Por exemplo, é possível a separação de atuação dos *Slaves* para realização de tarefas variadas em diferentes sites ou domínios da Internet, podendo ainda ser representados por diferentes empresas contratantes de serviço de segurança.

Um dos pontos críticos dessa implementação se refere à segurança do canal de comunicação do servidor IRC. Todas as comunicações via mensagem privada dentro do canal são feitas via túnel criptografado em protocolo *SSL*. Tal mecanismo evitaria, por exemplo, que agentes maliciosos escutem o trafego de rede entre um *Master* e um *Slave* para que este obtenha acesso aos *tickets*, podendo se passar por um usuário legítimo. Há também um preocupação com proteção de software para evitar processos de engenharia reversa que podem representar alterações de código, descoberta de informações críticas ou, até mesmo, descoberta dos *tickets* de autenticação. Formas de proteção baseadas em ofuscação e incorruptibilidade estão sendo incorporadas como formas de proteção [Collberg and Nagra 2010] .

## 4. Avaliação Experimental

Nesta seção do artigo, realizamos uma simulação de testes de indisponibilidade adotando a arquitetura da *botnet* proposta, em um simulador profissional de Ataques Distribuídos de

Negação de Serviço em larga escala denominado SADI (acrônimo para Simulador de Ataques Distribuídos de Indisponibilidade). Listamos as características do simulador, descrevemos o experimento em detalhes e apresentamos seus resultados.

### 4.1. SADI

O SADI é um simulador profissional de Ataques Distribuídos de Negação de Serviço capaz de realizar simulações reais de ataques no que tange a testes de indisponibilidade. A arquitetura do SADI foi primeiramente apresentada [Raphael 2014] como um simulador de ataques para solução de testes de stress e tem sido caracterizado como inovação para indústria . É de se notar a utilização desta ferramenta como um serviço de testes de indisponibilidade para realização de simulações em diversas empresas de *e-commerce* do Brasil, caracterizando-o como um projeto de inovação atuante no mercado brasileiro.

Com a arquitetura distribuída baseada em *botnet* aplicada no SADI e uma ampla biblioteca de ataques, o SADI seria capaz de personalizar cada cenário de acordo com os interesses do usuário, realizando medições precisas que permitem caracterizar completamente a efetividade e resistência de redes perante a essa classe de ataques. Pode-se explorar, ainda, uma série de técnicas que permitem otimizar o uso de recursos de rede, selecionando, a cada simulação, o conjunto de máquinas de ataque que permita alcançar os objetivos desejados com o menor uso de recursos possível.

Descrevemos abaixo um conjunto de funcionalidades e características desejáveis ao SADI que a arquitetura proposta é capaz de cumprir.

- Reunião de uma grande quantidade de máquinas em um canal de comunicação que possam ser coordenadas por um único agente, baseado em um sistema de hierarquia com intuito de ser aplicado comando e controle para realização de ataques distribuídos de negação de serviço.

- A arquitetura implementada garante uma ampla escolha de ataques. Além do tradicional *SYN Flood*, já existem implementações de ataques mais complexos baseados em reflexão de requisições e amplificação de pacotes [Raphael e Matheus 2014]**.**

- Execução de ataques mistos e paralelos. Com a separação de serviços, a arquitetura é capaz de realizar ataques em paralelo em um mesmo alvo - ou em alvos distintos -, podendo ser também ataques de diferentes tipos. Por exemplo, uso do canal#1 para realizar ataques de infraestrutura do tipo *TCP SYN Flood* e o uso do canal#2 com ataques de aplicação do tipo *HTTP GET Flood*.

- Ataques recursivos. Possibilidade de ataques inteligentes capazes de atuar em diversos níveis de um domínio. Por exemplo, através de *DNS Flood* [Raphael e Matheus 2014].

- Medições precisas e configurações salvas. A arquitetura é capaz de gerar relatórios com informações úteis: a simulação foi suficiente para gerar indisponibilidade ao sistema alvo, quantidade total de tráfego gerada, recomendações padrões de mitigação pós teste e etc - podendo também serem usadas no futuro para novos testes.

- Máquinas *Slaves* preferenciais. Os *Slaves* podem atuar como máquinas refletoras e amplificadoras para realização de ataques refletidos e distribuídos de negação de serviço - basicamente atuando como servidores de Internet mal-configurados propositalmente. Ainda, é possível a separação de *Slaves* que possuam alto desempenho rede para maximização da quantidade de tráfego gerada.

### 4.2. Simulação

O experimento realizado foi conduzido em um sistema alvo X. Foram realizadas 2 simulações usando diferentes usando diferentes ferramentas de ataque: *hping3* e *T50*. Na primeira simulação, foram usadas 6 máquinas *Slaves* usando *hping3*. Já para a segunda simulação, o ataque foi realizado com 3 *Slaves* utilizando *T50*. Ambas as simulações duraram em torno de 10 minutos.

O tipo de ataque realizado pelo SADI foi o ataque de infraestrutura *TCP SYN Flood*. Basicamente, este ataque é caracterizado como um ataque de força bruta que explora uma fraqueza no protocolo *TCP*, mais especificamente no *Three Way Handshake* no estabelecimento de uma conexão *TCP* entre Cliente e Servidor. Os *Slaves* começam o *handshake* enviando pacotes *TCP* com a *flag SYN* ativada para o alvo. Após isso, o alvo manda de volta um pacote com a *flag SYN*/*ACK* e espera pela *flag ACK*, fechando o *handshake*. Contudo, o ataque se baseia em não enviar a última *flag ACK*, deixando a conexão aberta. Por padrão, a espera pela *flag ACK* pode demorar 360 segundos, fazendo com que o alvo consuma seus recursos de rede e todas seus *stacks* de conexão disponíveis - que seriam usadas para atender conexões legitimas -, causando assim o status de negação de serviço.

Abaixo alguns resultados do relatório gerado pela arquitetura com os *IPs* omitidos.

| IP | Status | Atacando por | Tráfego gerado |
|---|---|---|---|
| | autenticado | 300 | 206.32 Mbps |
| | autenticado | 353 | 212.98 Mbps |
| | autenticado | 343 | 204.63 Mbps |
| | autenticado | 331 | 41.43 Mbps |
| | autenticado | 320 | 34.97 Mbps |
| | autenticado | 310 | 31.04 Mbps |

Total de tráfego gerado: 731.37 Mbps

**Tabela 1. Mostra informações da primeira simulação.**

| IP | Status | Atacando por | Tráfego gerado |
|---|---|---|---|
| | autenticado | 320 | 445.02 Mbps |
| | autenticado | 310 | 460.04 Mbps |
| | autenticado | 300 | 472.28 Mbps |

Total de tráfego gerado: 1377.34 Mbps

**Tabela 2. Mostra informações da segunda simulação.**

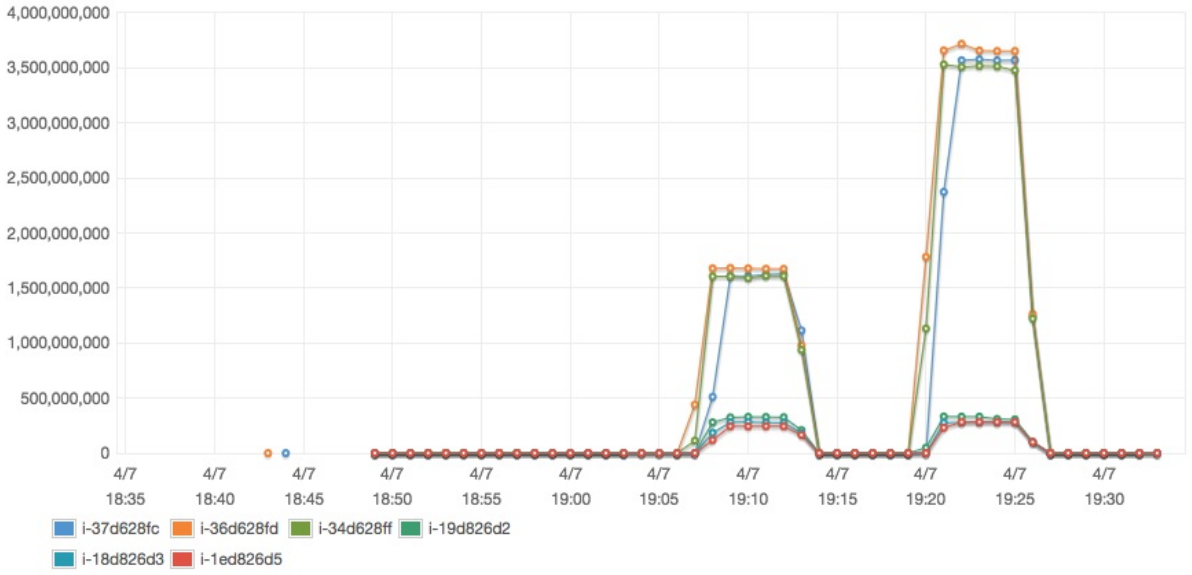


**Figura 2. Exibição em gráfico do resultado das simulações.**

Durante a simulação, a arquitetura se comportou bem. A largura de banda gerada por ambos os testes foi satisfatória para a quantidade de *Slaves* atuantes na simulação. Nota-se que o ataque em si foi melhor executado usando a ferramenta *T50*, gerando picos de aproximadamente 1.3 Gbps. Isso se deve ao fato de termos usado máquinas não otimizadas para desempenho de rede no caso da ferramenta *hping3*, o que mostra, mais uma vez, a importância de comandos de verificação de processamento e largura de banda (pelo *Master*) para a maximização dos resultados. Atualmente, com o mesmo tipo de ataque, pode-se chegar a valores superiores a 100 Gbps usando uma quantidade maior de máquinas que possuam alto desempenho de rede. Isso geralmente representa mais do que uma aplicação normal pode sustentar de trafego de rede.

## 5. Conclusões

A arquitetura proposta, baseada em *botnet*, foi capaz de cumprir seu papel, possibilitando o desenvolvimento de aplicações distribuídas que necessitem de alto poder computacional ou largura de banda. Destacamos que, tais propriedades são de extrema importância para a realização de diversos serviços de avaliação das bases da segurança da informação, e que podemos retificar que hoje a arquitetura é extremamente benéfica para utilização em sistemas que possam distribuir e coordenar tarefas.

Como estudo de caso, a aplicação SADI foi utilizada para a realização de ataques distribuídos e coordenados como forma de avaliar a disponibilidade de um determinado sistema. Sua adoção pelo SADI é um exemplo de sucesso e hoje, o SADI vem sendo usado para testes de indisponibilidade em diversos setores brasileiros (de maneira geral, comércio eletrônico e sistemas bancários). Esta classe de ataques vem causando problemas em diversos setores e vem consolidando como uma das mais letais armas contra tais serviços sensíveis e essenciais. Então torna-se necessário o tratamento dessa situação como um risco de segurança cibernética de alta prioridade, visto que, cada dia mais, dependemos da disponibilidade desses sistemas de informação.

Tudo isso apenas retifica a importância de testes de indisponibilidade para avaliação de sistemas de rede. Por este motivo, soluções anti-DDoS - ou seja, sistemas visando a prevenção, detecção e resposta a ataques DDoS - tendem a ser complexas, fazendo uso de sofisticados equipamentos de rede e avançadas técnicas de reconhecimento de padrões e inteligência computacional. Por estas características, determinar a efetividade das soluções anti-DDoS não é trivial. De fato, apenas por meio da execução de testes que reproduzam um amplo espectro de cenários de ataque é que se torna possível caracterizar a efetividade de uma solução anti-DDoS.

**Trabalhos futuros.** Atualmente, a arquitetura distribuída está em fase de desenvolvimento mas já é funcional e capaz de realizar diversas tarefas de forma distribuída. Descrevemos a seguir alguns pontos a serem trabalhados no futuro.

- *Pendrive* de auto instanciação. *Pendrive* capaz de instanciar uma máquina *Slave* voluntária para a *botnet*, através de PnP (em inglês: *Plug and Play*).

- Desenvolvimento de novos ataques. Notadamente, ataques mais complexos como ataques de negação de serviço permanente e outros ataques refletidos ainda não implementados.

- Algoritmos de Inteligência Artificial. Desenvolvimento e aplicação de algoritmos para detecção e analise distribuída de anomalias e ataques a partir de trafego de rede.

- Ataques *Zero Day*. Estudo de técnicas de evasão de ataques de negação de serviço para maximizar sua eficiência contra sistemas anti-DDoS.

- Medição precisa. Estudo de métodos que permitam estimar com maior precisão a geração de relatórios.

## Referências


HP (2014), " HP Study Reveals 70 Percent of Internet of Things Devices Vulnerable to Attack", http://h30499.www3.hp.com/t5/Fortify-Application-Security/HP-Study-Reveals-70-Percent-of-Internet-of-Things-Devices/ba-p/6556284#.VSMLMvnF9Bm.

The Register (2008), "Phlashing attack thrashes embedded systems", http://www.theregister.co.uk/2008/05/21/phlashing/.

WeLiveSecurity (2014), "Top 5 Scariest Zombie Botnets", http://www.welivesecurity.com/2014/10/23/top-5-scariest-zombie-botnets/.

CNET (2013), "Symantec takes on one of largest botnets in history", http://www.cnet.com/news/symantec-takes-on-one-of-largest-botnets-in-history/.

TRAC (2015), "The Wraith IRC Bot official project site", http://wraith.botpack.net/.

Team Cymru (2008), "A Teste of HTTP Botnets", http://www.team-cymru.com/ReadingRoom/Whitepapers/2008/http-botnets.pdf.

Dittrich D., Dietrich S. (2008), "P2P as botnet command and control: A deeper insight", *Malicious and Unwanted Software, 2008. MALWARE 2008. 3rd International Conference on*, Fairfax, VI, p. 41-48.

Common Weakness Enumeration, Mitre (2014), "CWE-400: Uncontrolled Resource Consumption", https://cwe.mitre.org/data/definitions/400.html.

Common Weakness Enumeration, Mitre (2014), "CWE-410: Insufficient Resource Pool", https://cwe.mitre.org/data/definitions/410.html.

Mirkovic, J., Reiher, P., (2004), "A Taxonomy of DDoS Attack and DDoS Defense Mechanisms", *SIGCOMM Comput. Commun. Rev.*, v. 34, n. 2 (Apr), p. 39-53.

Radware (2013). DDoS Survival Handbook, http://security.radware.com/uploadedFiles/Resources_and_Content/DDoS_Handbook/DDoS_Handbook.pdf.

Ke-nan Zhao, Lei Li (2011), "A New Intrusion Detection System Based on Rough Set Theory and Fuzzy Support Vector Machine", *Intelligent Systems and Applications (ISA), 2011 3rd International Workshop on*, Wuhan, p. 1-5.

Massachusetts Institute of Technology (2007), "Kerberos Project", http://www.kerberos.org/.

Collberg, C. S., Nagra, J. (2010) "Surreptitious Software: Obfuscation, Watermarking, and Tamperproofing for Software Protection" *Addison Wesley.*

Raphael M., Matheus S., Fabio D., Eduardo O., (2014), Henrique S., Rafael S., Bruno G., "Arquitetura de um Simulador de Ataques Distribuídos de Negação de Serviço", XLI Seminário Integrado de Software e Hardware (SEMISH 2014).

Matheus S., Fabio D., Raphael M., (2014) "Arcabouço para Simulação de Ataques Distribuídos Refletidos de Negação de Serviço", I Workshop de Iniciação Científica em Sistemas de Informação (WICSI 2014).