# Detecção de DDoS baseado em Processamento de Eventos Complexo para IoT

**Adeilson M. S. Cardoso[1], Rafael F. Lopes[2], Ariel S. Teles[2]**

[1]Departamento de Informática - Instituto Federal do Tocantins (IFTO)
Araguatins – TO – Brasil
[2]Programa de Pós-Graduação em Ciência da Computação
Universidade Federal Maranhão (UFMA) - São Luís – MA – Brasil

adeilson@ifto.edu.br {ariel,rafaelf}@lsdi.ufma.br

***Abstract.*** *With the expansion of the Internet of things, there has been an increase in the development of services and applications that use context-sensitive devices embedded in intelligent environments. These devices produce large amounts of data and need faster and more accurate defense mechanisms. In this work, we propose a new method that uses Complex Event Processing (CEP) for detecting Distributed Denial of Service (DDoS) attacks and evaluate its performance on the Raspberry Pi platform. The results show that the proposed method can be executed in devices with reduced computational power and that CEP is a viable solution for detection of DDoS attacks in real time in IoT environments.*

***Resumo.*** *Com a expansão da Internet de Coisas, houve um aumento no desenvolvimento de serviços e aplicações que utilizam dispositivos sensíveis ao contexto incorporados em ambientes inteligentes. Esses dispositivos produzem grandes quantidades de dados, e necessitam de mecanismos de defesa mais rápidos e precisos. Neste trabalho propomos um novo método que utiliza Processamento de Eventos Complexos (CEP) para detecção de ataques de Negação de Serviço Distribuído (DDoS), e avaliamos o seu desempenho na plataforma Raspberry Pi. Os resultados mostram que o método proposto pode ser executado em dispositivos com poder computacional reduzido e que o CEP é uma solução viável para detecção de ataques DDoS em tempo real em ambientes de IoT.*

## 1. Introdução

A Internet das Coisas (*Internet of Things* - IoT) é um novo paradigma que combina aspectos e tecnologias provenientes de diferentes abordagens, tais como: computação ubíqua, ciência de contexto, protocolos e tecnologias de comunicação, rede de computadores e dispositivos com sensores e atuadores [Borgia 2014]. Essa integração acontece a partir da mesclagem de milhares de dispositivos endereçáveis e heterogêneos, conhecidos como objetos inteligentes (*smart objects*), que possuem uma representação na Internet e visa a geração de sistemas onde os mundos real e digital se encontram e estão continuamente interagindo, sendo capazes de compartilhar dados de contexto entre si e com diversas aplicações. Com o aumento do número de dispositivos de rede conectados, mais oportunidades para a exploração de vulnerabilidades estarão disponíveis.

Existem muitos métodos de intrusão em redes de computadores, os quais configuram uma variedade de ataques que podem comprometer a segurança dos sistemas e a

disponibilidade dos dados. Os ataques diferem em termos das técnicas utilizadas e dos recursos técnicos que um atacante tem à sua disposição. Dentre as principais técnicas que afetam a disponibilidade dos dados destacam-se os ataques de inundação (i.e., *SYN Flood, UDP Flood, Smurf Attack*). Considerando que os dispositivos de IoT capturam dados sobre eventos de objetos inteligentes e *gateways* de IoT, esses fluxos de eventos precisam ser identificados, processados e ocorrer reação em um curto espaço de tempo [Jun and Chi 2014]. Portanto os mecanismos de segurança devem processar esses fluxos de eventos imediatamente e emitir alertas de segurança.

O Processamento de Eventos Complexos (*Complex Event Processing - CEP*) é uma tecnologia emergente para filtrar e processar eventos em cenários de tempo real [Cugola and Margara 2012]. As informações processadas no CEP são definidas como eventos e os fenômenos ocorridos na rede podem ser modelados como eventos. A tecnologia CEP pode ser usada para detectar relacionamentos complexos significativos entre eventos e responder a eles em tempo real, podendo ser aplicado a um amplo espectro de desafios de sistemas de informação, incluindo automação de processos de negócios, processos de cronograma e controle, monitoramento de rede, previsão de desempenho e detecção de intrusão [Luckham 2008]. Assim, um sistema de detecção de intrusão que utiliza CEP pode ser usado para lidar com padrões de eventos e processar grandes volumes de dados, podendo ser uma ótima solução para melhorar o desempenho e a identificação de ameaças em tempo real em ambientes de IoT.

Este trabalho apresenta um método para detecção de ataques de negação de serviço distribuído que consiste no uso de regras CEP para resolver problemas de segurança em ambientes IoT em tempo real. O método proposto foi projetado para realizar a detecção de três tipos de ataques DDoS: *"SYN Flood, UDP Flood e Smurf Ataque"*, que têm como característica um grande fluxo de dados. Além disso, o método pode ser estendido para detectar outros tipos de ataques (i.e., ARP Spoofing e ataques de roteamento). A avaliação do método foi realizada em Raspberry Pi, um pequeno computador que dispõem de poucos recursos computacionais, através do qual foi possível mensurar a viabilidade da método proposto.

O restante deste trabalho está organizado da seguinte forma. A Seção 2 apresenta uma breve revisão de literatura. A Seção 3 apresenta os trabalhos relacionados. Na Seção 4 é apresentado o método de detecção proposto. A Seção 5 apresenta os resultados. Na seção 6 são apresentadas as discussões e por fim, na Seção 7, a conclusão.

## 2. Revisão de Literatura

Nesta seção são apresentados conceitos relacionados a ataques de negação de serviço distribuído e CEP, os quais são importantes para a compreensão deste trabalho.

### 2.1. Ataques de Negação de Serviço Distribuído

Os mecanismos de segurança destinados a proteger as comunicações em IoT precisam fornecer garantias adequadas em relação a confidencialidade, integridade, autenticidade e disponibilidade dos dados. Os ataques à disponibilidade dos dados são muito comuns e podem causar enormes prejuízos financeiros para as organizações. Dentre os ataques que afetam a disponibilidade dos sistemas, os mais frequentemente utilizados são ataques de negação de serviço distribuído (*Distributed Denial of Service - DDoS*). Um ataque

DDoS tem o objetivo de esgotar os recursos disponíveis para um determinado serviço ou sistema, enviando pacotes desnecessários e, portanto, impede que os usuários legítimos acessem os serviços ou recursos aos quais têm direito [Kumari et al. 2010].

Existem duas classes principais de ataques de Negação de Serviço Distribuído: ataque de redução de largura de banda e ataque de esgotamento de recursos. Um ataque de redução de largura de banda é projetado para inundar a rede com tráfego indesejado que impede o tráfego legítimo e o funcionamento normal da rede. Um ataque de esgotamento de recursos é um ataque projetado para esgotar os recursos de um sistema vítima tornando-o incapaz de processar pedidos legítimos dos serviços [Kumari et al. 2010]. Os principais ataques de esgotamento de recursos fazem uso de exploração de protocolos através do uso indevido de pacotes TCP SYN *(Transfer Control Protocol Synchronize)*. Os ataques de inundação além do protocolo TCP, também podem usar o protocolo UDP (*User Datagram Protocol*) ou ICMP (*Internet Control Message Protocol*). Dentre eles pode ser citado os ataques (*SYN Flood, UDP Flood e Smurf Ataque*).

*SYN Flood* é um dos ataques de negação de serviço em que uma pessoa mal intencionada envia pacotes de sincronização (TCP SYN) a uma taxa maior que 128 pacotes por segundos [Cisco 2018] para um sistema de destino com o objetivo de consumir seus recursos para que o sistema não responda ao tráfego legítimo. O ataque *UDP Flood* é caracterizado pelo grande número de pacotes UDP simultâneos a uma taxa maior que 30 pacotes por segundos [Kasinathan et al. 2013], onde são enviados inúmeros pacotes com o mesmo endereço IP de destino, no entanto para diferentes portas, cujo objetivo é saturar a largura de banda. O *Smurf Attack* também conhecido como ataque de inundação por amplificação é caracterizado pelo envio de requisições falsas para um endereço IP de *broadcast*, fazendo com que todos os endereços IP ativos dentro do intervalo de *broadcast* enviem uma resposta às requisições ao sistema vítima [Somal and Virk 2014].

### 2.2. Processamento de Eventos Complexos - CEP

O CEP tem origem em pesquisas realizadas sobre Simulação de Eventos Discretos, onde o primeiro projeto que utilizou um modelo genérico de linguagem CEP foi desenvolvido na Universidade de Stanford, liderado por David Luckham [Luckham 2008]. O CEP é uma tecnologia emergente que possibilita fazer a correlação de eventos de entrada contínuos e padrões de interesse, onde os resultados podem ser outros eventos complexos, que são derivados dos eventos de entrada. CEP faz o processamento de fluxos de dados sobre eventos que acontecem permitindo inferir uma conclusão a partir deles. Através do processamento de eventos é possível combinar dados de várias fontes para inferir eventos ou padrões que sugerem circunstâncias mais complexas.

Em contraste com os Sistemas de Gerenciamento de Banco de Dados, nos quais os dados são primeiro armazenados para depois serem consultados, o CEP armazena consultas contínuas e executa dados através delas, ou seja, ao invés de armazenar os dados, o CEP se concentra em analisar e processar continuamente os dados que passam, usando as consultas armazenadas [Junior 2017]. Assim CEP é usado para processar e analisar fluxos de dados em tempo real, bem como produzir resultados com baixa latência, mesmo em casos onde o fluxo de eventos é grande. Desse modo o objetivo do processamento de eventos complexos é identificar eventos significativos (como oportunidades ou ameaças) e responder a eles o mais rápido possível [Govindasamy et al. 2014].

CEP utiliza uma linguagem de processamento de eventos adaptada chamada *Event Processing Language - EPL*, que permite expressar condições ricas e correlações entre eventos, possivelmente abrindo janelas de tempo, minimizando assim o esforço de desenvolvimento necessário para configurar um sistema que possa reagir a situações complexas [Luckham 2008]. EPL é similar ao SQL no uso das cláusulas *select e where*. Entretanto, as instruções EPL em vez de tabelas usam fluxos de eventos e conceitos chamados *Views* (Visualizações) e *Event operators* (Operadores de eventos). As visualizações podem representar janelas de tempo sobre um fluxo de eventos, derivar estatísticas de propriedades de eventos ou manipular valores de propriedade de eventos exclusivos. Nos operadores de eventos existem cláusulas do tipo (*select, from, where, group by, order by, insert into, update, having, delete, pattern* e *output*). A seguir é apresentado um exemplo de regra CEP usado para detecção de *Smurf Attack*.

```
"select count(icmp_echo) as smurf_count, src_ip, broadcast from
       ICMPpacketEvent.win:time(1 sec)
            having count(*) > 100 and dst_ip = broadcast"
```

**Código 1. Exemplo de regra CEP para detecção de Smurf Attack**

O Código 1 representa uma regra em CEP utilizada para detecção de *Smurf Attack*. A regra realiza a contagem de todos os pacotes ICMP em uma janela de tempo deslizante de 1 segundo. Em seguida seleciona e retorna o endereço IP de origem e o endereço IP de *broadcast* das conexões com o servidor, verificando se mais de 100 pacotes ICMP foram enviados de um mesmo endereço de origem [Cisco 2018] dentro da janela de tempo especificada, tendo o endereço *broadcast* como endereço de destino. A *view* utilizada foi *win.time* que especifica o intervalo de tempo de chegada dos eventos.

## 3. Trabalhos Relacionados

Dois Sistemas de Detecção de Intrusão - (*Intrusion Detection System - IDS*) populares de código aberto são Snort e Bro [Paxson et al. 2012]. Ambos realizam a captura do trafego da rede através de um *Sniffer* de pacotes e realizam a detecção de ataques a partir de uma grande lista de assinaturas que configuram as políticas para detecção do tráfego malicioso. No entanto, executar uma grande lista de regras pode ser viável para uma rede tradicional utilizando máquinas com alto poder computacional, todavia em pequenas redes, como redes IoT e em dispositivos que dispões de poucos recursos computacionais como um Raspberry PI, essa abordagem pode provocar uma sobrecarga e prejudicar o desempenho do sistema.

Os autores em [Kasinathan et al. 2013] propuseram uma arquitetura de IDS para detecção de ataques de negação de serviços para IoT baseado no *framework* de rede ebbits. Na abordagem proposta, o IDS pode ouvir ou monitorar o tráfego 6LoWPAN (*IPv6 over Low-Power Wireless Personal Area Networks*). O gerenciador de proteção é o componente principal do sistema proposto, o qual gera alertas por meio de informações disponíveis sobre o componente do gerenciador de rede. A arquitetura possui um componente para analisar o tráfego da rede e detectar nós com mau comportamento, buscando identificar ataques de negação de serviço em redes 6LoWPAN. No entanto, o método utilizado pode gerar altas taxas de consumo de recursos e provocar um baixo desempenho.

Os autores em [Jun and Chi 2014] propuseram um IDS baseado em processamento de eventos complexos para resolver problemas de intrusão em IoT. O IDS é baseado

em regras e usa SQL e EPL como referência. Antes de obter informações úteis dos fluxos de eventos, os dados brutos dos *gateways* IoT ou *stream* de servidores são filtrados pelo Filtro de Eventos. Em seguida, os eventos são extraídos e formulados a partir de dados IoT por um modelo de estrutura de dados do evento (i.e., Pojo, XML ou ObjectArray), depois é processado pelo motor CEP para análise de tráfego malicioso. A proposta dos autores foi projetada para detectar apenas o *Land Attack*. Além disso, o IDS proposto pelos autores considera apenas o tamanho dos pacotes no processo de identificação do tráfego malicioso, podendo causar um alto número de falsos positivos.

A proposta dos autores [Kyaw et al. 2015] concentram seus estudos em Raspberry Pi, comparando o desempenho de Snort e Bro. Os autores configuraram o Snort e o Bro para utilizarem apenas os módulos de assinaturas para detecção dos ataques de *Port Scan, SYN Flood e ARP Spoofing*. O experimento é realizado em vários cenários combinando tráfegos em segundo plano com diferentes tamanhos de pacotes. Nossa metodologia de teste também aborda cenários combinando tráfegos em segundo plano, no entanto ela difere deste trabalho a medida que utilizamos uma alta taxa de pacotes por segundo para estressar o sistema.

## 4. Método Proposto

Este trabalho apresenta um método baseado em CEP focado na detecção em tempo real de negação de serviço distribuído em redes IoT. Três tipos diferentes de ataques de redes foram selecionados para estressar e avaliar o sistema proposto (*SYN Flood, UDP Flood e Smurf Ataque*). A arquitetura do método proposto é apresentada na Figura 1.

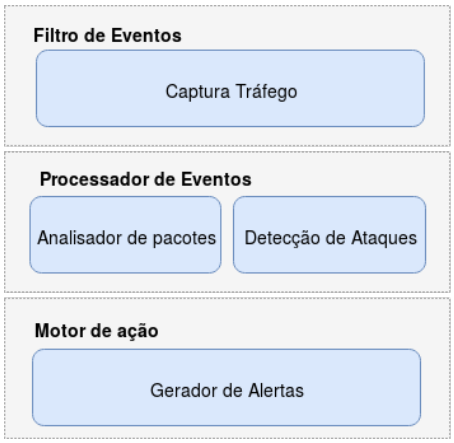


**Figura. 1. Arquitetura Proposta**

A arquitetura do método proposto é composta por três módulos: Filtro de Eventos, Processador de Eventos e Motor de Ação. O sistema verifica fluxos de eventos como entrada a partir de informações coletadas do tráfego da rede, processa, analisa, e produz relatórios de segurança para o motor de ação conforme as regras escritas em CEP.

A módulo **Filtro de Eventos** é usado para monitorar e coletar o tráfego da rede e a ocorrência de eventos. O fluxo de dados é capturado a partir de uma placa de rede do sistema e os dados são encaminhados para o processador de eventos. A módulo **Processador de Eventos** é composto pelas classes *analisador de pacotes* que realiza a análise das características dos pacotes capturados e *detecção de ataques*, que é responsável por

determinar qual tipo de ataque está acontecendo. Os resultados do processamento são gerados automaticamente e continuamente. O módulo **Motor de Ação** é responsável por tratar os eventos que desencadearam as regras CEP, emitir alertas sobre atividades suspeitas de intrusão e bloquear o acesso a serviços para os endereços IPs que ativaram as regras.

Quando o sistema é iniciado, uma rotina de captura de pacotes é realizada pelo módulo capturar tráfego. Após a captura dos pacotes, as etapas seguintes são: (i) extração das características dos pacotes capturados e (ii) análise do fluxo de dados pelo processador de eventos, onde o módulo analisador de pacotes verifica as características extraídas dos pacotes (i.e., endereços IP de origem e destino, número total de pacotes, tamanho médio de pacote, protocolo do pacote, taxa de pacote por segundo) e determina se o tráfego é malicioso ou não. Após a análise ser realizada, é verificado se o resultado caracteriza um ataque. Caso o resultado da análise não detecte nenhuma característica de intrusão, os dados analisados são descartados e o processo de captura do tráfego é reiniciado. Caso contrário, o módulo detecção de ataques irá fazer a identificação do tipo de ataque que está ocorrendo conforme as regras CEP criadas para cada tipo de ameaça. Uma vez que o ataque é identificado, um alerta é enviado de acordo com o tipo de ataque.

### 4.1. Aspectos de Implementação

O método proposto foi desenvolvido usando a linguagem Java abordando a metodologia de Programação Orientada a Objetos. A Jpcap é a biblioteca utilizada para capturar o tráfego de rede e consiste em um conjunto de classes Java e interfaces que implementam chamadas para uma biblioteca do sistema *libpcap*. Jpcap oferece captura de pacotes em tempo real e uma opção para armazenar e ler pacotes capturados em arquivos *offline*, também permite a filtragem de pacotes por tipo de protocolo (IPv4, IPv6, ARP / RARP, TCP, UDP e ICMPv4), que o método abordado usa para detectar ataques DDoS.

O Sistema DoSCEP foi implementado para plataforma Raspberry Pi, um dispositivo com poucos recursos computacionais, que possui 1 gigabyte de memória e um processador ARM quad-core de 1,2 GHz e usa o sistema operacional *Raspbian*. A escolha da plataforma se deu devido as características que o dispositivo oferece, que são: (i) pequeno computador de placa única, com interfaces de rede sem fio que permitem que objetos comuns sejam transformados em objetos inteligentes, com capacidades de detecção e atuação, (ii) ele é portátil, podendo os usuários transportá-lo sem esforços, (iii) é versátil, podendo ser utilizado em qualquer lugar, desde casas inteligentes até universidades e praças.

### 4.2. Regras de Detecção

Para detectar os ataques, foi utilizada uma abordagem baseada na extração de características de pacotes a partir das quais as regras do CEP foram criadas para identificar os ataques. Uma das regras CEP utilizadas para detectar o ataque *SYN Flood* é mostrada no Código 2.

```
"Select count(syn) as synCount, ack, src_ip, dst_ip from
   TCPpacketEvent.win:time(1 sec) having count(syn) > 128 and
    not(ack)"
```

**Código 2. Regra CEP para SYN Flood**

Na regra descrita no Código 2 é emitido um alerta quando a taxa de pacotes *TCP SYN* que chegam no dispositivo for maior que 128 pacotes em uma janela de tempo de 1 segundos [Cisco 2018] e quando a *flag ACK* estiver ausente. A regra no Código 3 é usada para detecção de ataques *UDP Flood*.

```
"Select count(*) as udpEvent, count, src_ip from UDPpacketEvent.
   win:time(1 sec) having count(*) > 30 "
```

**Código 3. Regra CEP para UDP Flood**

Para detecção do *UDP Flood* uma das regras utilizadas foi uma abordagem do IDS Suricata, onde o sistema emite um alerta se os pacotes UDP chegarem a uma taxa maior que 30 pacotes por segundo. Essa abordagem pode ser utilizada para detecção de diferentes ataques de inundação, levando em consideração a quantidade de pacotes em um limite de tempo [Kasinathan et al. 2013]. A regra para detecção do *Smurf Attack* foi apresentada Seção 2.2, Código 1.

## 5. Resultados e Testes

O método proposto foi avaliado em um *Raspberry Pi 3 Model B* com o sistema operacional *Raspbian*. Devido as limitações do poder computacional do Raspberry Pi, três fatores importantes foram considerados para realizar a avaliação do desempenho em tempo real do método e verificar se o uso do mecanismo de processamento de eventos pode atender aos requisitos do ambiente IoT. Os fatores considerados foram: uso da CPU, uso de memória RAM e número de ataques detectados. Foi desenvolvida uma ferramenta para obter informações sobre os usos de CPU e RAM durante a execução dos experimentos e o comando *capstats* foi usado para mostrar o número de pacotes analisados e pacotes perdidos. A Figura 2 mostra um alerta para ataques *SYN Flood*, destacando o endereço do invasor.


**Figura. 2. Alerta SYN Flood**

O sistema utiliza uma abordagem baseada em janelas de tempo, analisando a intensidade do tráfego na rede, o que possibilita um maior número de detecção desses tipo de ataques em comparação aos sistemas que não utilizam essa abordagem.

### 5.1. Cenários de Testes

O tráfico em segundo plano e os ataques para validação do sistema foram realizados com a ferramenta *Hping3*[1]. A Figura 3 exibe o comando usado para executar o ataque *SYN Flood*.

```
# hping3 -V --syn -d -80 -i u999 -a 192.168.2.4 192.168.10.1
```

**Figura. 3. Comando Hping3 em ação**

[1]http://www.hping.org/

Todos os ataques foram realizados com 30% de tráfego de segundo plano e com taxas de 1000 pacotes por segundo. O Hping3 foi configurado para gerar tráfego de segundo plano a partir dos dispositivos Rasp1 e Rasp2 na rede local, conforme ilustrado na topologia de rede na Figura 4. O experimento foi realizado em um ambiente de testes controlado e composto por um *desktop*, um *laptop* e três *Raspberry Pi*. O dispositivo (IP: 192.168.10.1) foi configurado para ser um ponto de acesso *wifi*. Faz parte do cenário para o experimento uma aplicação para medição de temperatura e umidade, no qual foi utilizado um sensor DHT11 para realizar a coleta da temperatura e umidade do ambiente.

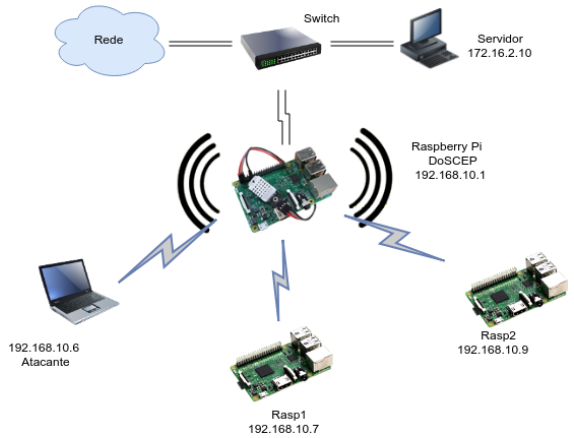


**Figura. 4. Topologia de rede para o experimento**

Os resultados do experimento representam uma média de dez repetições de 120 segundos [Matos et al. 2017] para cada tipo de ataque realizado. Os resultados têm um nível de confiança de 95%. O software usado para as análises estatísticas foi o *PAST*[2] versão 3.20. *PAST* é um software livre para análise de dados científicos.

## 5.2. Uso da CPU e Memória RAM

A Tabela 1 apresenta o resultado médio de utilização da CPU e Memória RAM pela aplicação (método) baseado em CEP durante os testes realizados. A aplicação estava usando 0,5% de CPU e 1,4% de memória RAM antes de iniciar os ataques.

**Tabela. 1. Uso da CPU e RAM durante os testes realizados**

| | SYN Flood | UDP Flood | Smurf Ataque |
|---|---|---|---|
| CPU (%) | 38,15 | 20,49 | 37,06 |
| RAM (%) | 5,65 | 5,56 | 5,52 |

A partir da tabela Tabela 1, é possível observar que o método proposto teve um desempenho satisfatório ao realizar a análise da rede durante as simulações de ataques. Os resultados mostram que a tecnologia CEP pode ser utilizada para melhorar o desempenho em tempo real na detecção de DDoS para IoT em cenário baseado em eventos, uma vez que os consumos máximos atingidos durante os ataques foram de 38.15% e de 5.65% para uso da CPU e memória RAM respectivamente.

[2]https://folk.uio.no/ohammer/past/

### 5.3. Taxa de Detecção

A Tabela 2 resume os resultados experimentais para as taxas de detecção de ataques para verdadeiro positivo (VP) e falso positivo (FP) e precisão para os diferentes tipos de ataques abordados nesta pesquisa. O método proposto alcançou uma boa precisão nos resultados experimentais, tendo a menor precisão de 93,10% para SYN Flood e a maior precisão de 99,24% para *UDP Flood*. Para a taxa de detecção também mostrou um bom desempenho, alcançando 94,09% como a menor taxa de verdadeiros positivos e 6,96% como a maior taxa de falsos positivos.

**Tabela. 2. Taxa de detecção e precisão**

| | SYN Flood | UDP Flood | Smurf Ataque |
|---|---|---|---|
| VP | 94,09 | 98,75 | 96,32 |
| FP | 6,96 | 0,75 | 1,5 |
| Precisão | 93,10 | 99,24 | 98,4 |

### 5.4. Comparações

Nesta seção, faremos um abordagem comparativa do desempenho do método baseado em CEP com o IDS Bro. Bro é um Sistema de Detecção de Intrusão em Redes (NIDS), que foi desenvolvido para sistemas Unix e utiliza o método baseado em anomalias e em assinaturas [Paxson et al. 2012] para monitorar o tráfego de rede. Bro foi configurado para realizar apenas a detecção dos três tipo de ataques abordados nesta pesquisa, sendo avaliado no Raspberry Pi utilizando o mesmo cenário e as mesmas configurações utilizadas durante a avaliação do método baseado em CEP.

**Tabela. 3. Desempenho médio**

| - | **CPU (%)** | **RAM (%)** | **Taxa Detecção (%)** |
|---|---|---|---|
| **IDS Bro** | 80,33 | 5,76 | 82,43 |
| **Método CEP** | 32,08 | 5,57 | 96,38 |

Na Tabela 3 é possível observar que nesse cenário especifico o método proposto teve desempenho satisfatório nos resultados experimentais em comparação com o sistema Bro. O método proposto usou apenas 5,57% de memória RAM, enquanto Bro utilizou 5,76%. No uso da CPU, o método baseado em CEP permanece com melhor desempenho que Bro, utilizando 32,08% de processamento, enquanto Bro utilizou 80,33%. Para taxa de detecção o método proposto obteve 96,38% enquanto o Bro obteve 82,43%.

## 6. Conclusão

A contribuição deste trabalho é um método de detecção de DDoS baseado no mecanismo de processamento de eventos para ambientes de IoT. O método proposto foi avaliado em um Raspberry Pi 3 e os resultado experimentais mostram que ele teve melhor desempenho que Bro para todos os teste realizados. Os resultados mostram que o mecanismo CEP é adequado para aplicações e serviços que precisam processar grandes fluxos de dados, tornando-o uma ferramenta adequada para a detecção de intrusão. O mecanismo de processamento de eventos complexos provou através de avaliações referentes a utilização da

CPU, consumo de memória e taxa de detecção de ataques, ser adequado para aplicações e serviços que precisam processar grandes fluxos de dados gerados a partir de objetos inteligentes em ambientes IoT.

## References

Borgia, E. (2014). The internet of things vision: Key features, applications and open issues. *Computer Communications*, 54:1 – 31.

Cisco, S. (2018). Configuring attack protection - cisco small business isa500. https://www.cisco.com/assets/sol/sb/isa500_emulator/help/guide/ag1359280.html.

Cugola, G. and Margara, A. (2012). Processing flows of information: From data stream to complex event processing. *ACM Comput. Surv.*, 44(3):15:1–15:62.

Govindasamy, V., Ganesh, R., Nivash, G., and Shivaraman, S. (2014). Prediction of events based on complex event processing and probabilistic fuzzy logic. In *2014 International Conference on Computation of Power, Energy, Information and Communication (ICCPEIC)*, pages 494–499.

Jun, C. and Chi, C. (2014). Design of complex event-processing ids in internet of things. In *Proceedings of the 2014 Sixth International Conference on Measuring Technology and Mechatronics Automation*, ICMTMA '14, pages 226–229, Washington, DC, USA. IEEE Computer Society.

Junior, M. P. R. (2017). *DG2CEP: An On-line Algorithm for Real-time Detection of Spatial Clusters from Large Data Streams through Complex Event Processing*. PhD thesis, PhD thesis, Departamento de Informática da Pontifícia Universidade Católica.

Kasinathan, P., Pastrone, C., Spirito, M. A., and Vinkovits, M. (2013). Denial-of-service detection in 6lowpan based internet of things. In *WiMob*, pages 600–607.

Kumari, P., Kumar, M., and Rishi, R. (2010). Study of security in wireless sensor networks. *Proceedings of International Journal of Computer Science and Technology*, 1(5):347–354.

Kyaw, A. K., Chen, Y., and Joseph, J. (2015). Pi-ids: evaluation of open-source intrusion detection systems on raspberry pi 2. In *2015 Second International Conference on Information Security and Cyber Forensics (InfoSec)*, pages 165–170.

Luckham, D. (2008). *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*, pages 3–3. Springer Berlin Heidelberg, Berlin, Heidelberg.

Matos, F. M., de Sousa Araujo, T. E., and Moreira, J. A. (2017). Intrusion detection systems' performance for distributed denial-of-service attack. In *2017 CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*, pages 1–6.

Paxson, V., Sommer, R., Hall, S., Kreibich, C., Barlow, J., Clark, G., Maier, G., Siwek, J., Slagell, A., Thayer, D., et al. (2012). The bro network security monitor.

Somal, L. and Virk, K. (2014). Classification of distributed denial of service attacks–architecture, taxonomy and tools. *Int. J. Adv. Res. Comput. Sci. Technol*, 2(2):118–122.