Integração do TheHive com LLM para Análise Automatizada de Alertas

Éverton Victor Santos Silva

CECyber Capacitação e Soluções de Tecnologias LTDA, São Paulo - São Paulo, evertonvictorpro@gmail.com
Pós - Segurança da Informação e Inteligência Defensiva

Alameda Rio Negro, 503 - Sala 2020, Alphaville Centro Industrial e Empresarial/Alphav
Barueri/SP, Brasil, 06454-000

Resumo – Este trabalho apresenta uma solução de automação na análise de alertas de segurança cibernética utilizando a integração entre a plataforma de gestão de incidentes TheHive, um serviço Webhook desenvolvido em Node.js é uma instância local de uma Large Language Model (LLM). A integração permite a geração automática de recomendações baseadas no conteúdo dos alertas recebidos, acelerando a resposta a incidentes e otimizando o trabalho de analistas de segurança.

Palavras-chave — TheHive, Large Language Models (LLM), Ollama, Llama 3, Segurança da Informação, Resposta a Incidentes, Integração de Sistemas, Webhooks, Análise Automatizada de Alertas, Docker, PowerShell, VMware, Automação em Cibersegurança, Machine Learning, Inteligência Artificial (IA).

I. Introdução

O crescimento exponencial de alertas de segurança em ambientes corporativos têm desafiado as equipes de Segurança da Informação a responder de forma rápida e eficiente a incidentes. Diante desse cenário, soluções de automação e análise inteligente têm se tornado cada vez mais relevantes para apoiar os analistas durante a triagem e o tratamento de incidentes [9][10][11][12].

Este estudo apresenta a implementação de um fluxo de análise automatizada de alertas de segurança utilizando uma integração entre a plataforma de gerenciamento de incidentes TheHive [1][2] e Large Language Model (LLM) local, baseada no modelo LLaMA 3 [3][4]. O objetivo foi validar, em ambiente de laboratório, a capacidade de uma IA local fornecer recomendações contextuais e iniciais de resposta a incidentes cibernéticos [9][10][11].

A abordagem proposta simula a recepção de alertas no TheHive através de requisições HTTP (método POST), imitando a integração com um SIEM [13] como o Wazuh [14] ou qualquer outra fonte de detecção de eventos de segurança [12].

Após o recebimento, cada alerta é processado por uma aplicação Node.js [15], que consulta a LLM [3][4]. local para gerar recomendações automáticas de resposta, as quais são posteriormente anexadas ao incidente dentro do TheHive.

Vale ressaltar que o ambiente de testes foi construído sobre infraestrutura virtualizada, com recursos de hardware limitados, o que impactou o tempo de geração de resposta pela LLM [3][4]. Entretanto, o cenário demonstrou de forma clara o potencial de uso desta arquitetura como um modelo escalável para futuras implementações em ambientes produtivos.

II. Metodologia

Este trabalho foi desenvolvido utilizando um ambiente de laboratório, com infraestrutura virtualizada em VMware [6], para validar o fluxo de integração entre o TheHive e uma LLM (Large Language Model) local.

A. Configuração Inicial do Ambiente

A validação da solução proposta foi conduzida em um ambiente de laboratório, com infraestrutura virtualizada baseada em VMware [6], visando testar o fluxo de integração entre a plataforma de gerenciamento de incidentes TheHive e uma instância local de Large Language Model (LLM), utilizando o modelo LLaMA 3 [3][4].

B. Tecnologias Utilizadas

- TheHive (Plataforma de Gerenciamento de Incidentes) é uma plataforma de código aberto voltada para gestão de alertas, investigações e

resposta a incidentes de segurança cibernética (IR - Incident Response). O sistema foi implementado em containers Docker [5] para facilitar o provisionamento e o gerenciamento dos serviços [1][2].

- Node.js [15] com Express.js (Webhook Listener e Orquestração de Fluxo), responsável por atuar como listener de Webhooks enviados pelo TheHive. Esse serviço tem a função de capturar eventos de criação de alertas e, em seguida, acionar a LLM local com o conteúdo relevante do incidente [15].
- Ollama com LLaMA 3 [3][4] (Implementação da LLM Local), é uma ferramenta que facilita a execução local de modelos de linguagem de código aberto, como o LLaMA 3 [3][4]. A escolha por este modelo considerou fatores como a facilidade de instalação, compatibilidade com ambientes Linux, e a possibilidade de operar localmente sem dependência de serviços externos na nuvem [3][4].
- Terminal, usado durante a fase de testes, a criação de alertas no TheHive [1][2] foi realizada via requisições HTTP POST, utilizando scripts. Essa abordagem permitiu simular o envio de eventos a partir de um SIEM (como o Wazuh [14]), criando cenários realistas de detecção de incidentes [8].
- VMware (Infraestrutura de Virtualização), toda a infraestrutura foi provisionada em ambiente VMware Workstation, proporcionando isolamento entre as máquinas virtuais e possibilitando a alocação de recursos ajustáveis conforme a demanda de processamento dos serviços [6].
- Docker e Docker Compose, usados para conteinerização dos serviços do TheHive [1][2] e seus componentes (PostgreSQL, Elasticsearch, Cortex) [5].

C. Topologia do Ambiente de Testes

A topologia utilizada no laboratório foi composta por dois servidores virtuais ubuntu [7], cada um com funções específicas, comunicando-se por meio de uma rede virtual local (LAN), conforme mostrado na Tabela I.

TABELA I – COMPONENTES DO LABORATÓRIO

Componente	Configuração de Laboratório	Configuração Recomendada para Produção
Servidor TheHive	Ubuntu Server 22.04 LTS 4 GB RAM 2 vCPUs (2x2 núcleos) Execução via Docker Containers	Ubuntu Server 22.04 LTS ou superior Mínimo 8 GB RAM 4 vCPUs Banco de dados dedicado (ex.: Elasticsearch separado)
Servidor LLM (Ollama + LLaMA 3)	Ubuntu Server 22.04 LTS 8 GB RAM 4 vCPUs (2x4 núcleos) Execução via CPU (sem aceleração de hardware)	Ubuntu Server 22.04 LTS ou superior 32 GB RAM ou mais 8 ou mais vCPUs Preferencialmente com GPU (CUDA) para aceleração de inferência
Infraestrutura de Virtualização	VMware Workstation/ESXi Rede virtual LAN	VMware vSphere, Hyper-V ou servidores bare-metal Rede de produção com isolamento por VLANs e políticas de segurança
Recurso de Rede	Rede interna virtualizada com acesso externo	Rede segmentada com controle de acesso, isolamento de VLANs e monitoramento de tráfego

D. Limitações do Ambiente de Teste

Devido às limitações de hardware, especialmente na VM dedicada à LLM, foi identificado um tempo adicional para a geração de respostas por parte do modelo de linguagem. Em um ambiente de produção, recomenda-se o uso de hardware com recursos otimizados de memória e processamento, além de aceleração por GPU, para reduzir a latência na resposta e melhorar a experiência operacional.

E. Considerações para Ambientes Produtivos

Para uso em produção, recomenda-se:

- Uso de GPU para a LLM, visando reduzir o tempo de inferência e aumentar a capacidade de resposta.

- Alta disponibilidade para o TheHive [1][2], especialmente em ambientes com grande volume de alertas, utilizando balanceamento de carga e bancos de dados replicados.
- Monitoramento contínuo, com ferramentas como Prometheus, Grafana ou ELK, para acompanhar desempenho e disponibilidade.
- Comunicação segura, utilizando HTTPS, VPN ou redes isoladas, além de implementar controles de acesso.
- Escalabilidade da LLM, com possibilidade de expansão horizontal para atender a maiores volumes de análise.

III. Configuração da Large Language Model (llm)

A Large Language Model (LLM) utilizada no ambiente de laboratório foi baseada no modelo LLaMA 3 [3][4], executada localmente utilizando a ferramenta Ollama, que facilita o gerenciamento e a inferência de modelos de linguagem em máquinas locais.

A. Preparação do Ambiente

Utilize um servidor ubuntu [7] 22.04 LTS+ limpo e atualizado: *sudo apt update && sudo apt upgrade -y*.

B. Dependências Básicas

Instale o curl, git e outras ferramentas essenciais: *sudo apt install curl git build-essential -y*.

C. Instalação e Configuração do Ollama

O Ollama foi instalado conforme documentação oficial, configurado para rodar o modelo LLaMA 3 [3][4] em modo servidor, expondo uma API REST local na porta 11434, utilizada para integração com o sistema de orquestração via webhook.

A API permite o envio de prompts e a recepção das respostas geradas pela LLM, sendo configurada para não utilizar streaming, simplificando o consumo das respostas no fluxo de automação.

Execute o script oficial para instalação do Ollama no servidor ubuntu [7]: *curl https://ollama.com/install.sh | bash*.

Após instalar o Ollama, faça o download do modelo LLaMA 3 com o comando: *ollama pull llama3*. Esse comando baixa e prepara o modelo localmente para uso.

D. Teste da API Local da LLM

Para garantir que a LLM está funcionando corretamente, execute uma requisição de teste para a API local: *curl http://localhost:11434/api/generate -H "Content-Type: application/json" -d '{"model":"llama3","prompt":"Análise do alerta: ataque brute force SSH detectado. Qual o próximo passo?","stream":false}*'. Se a configuração estiver correta, a resposta incluirá uma recomendação gerada pela LLM [3][4].

IV. Configuração iniciais do thehive

A plataforma TheHive [1][2] foi configurada utilizando Docker Compose [5], possibilitando uma implantação rápida e modular dos serviços essenciais para seu funcionamento, incluindo o banco de dados PostgreSQL, o Elasticsearch e o próprio TheHive [1][2], além do Cortex para análise complementar.

A. Preparação do Ambiente

Utilize um servidor Ubuntu 22.04 LTS+ limpo e atualizado: *sudo apt update && sudo apt upgrade -y*.

B. Instalação do Docker e Docker Compose

Para preparar o ambiente e executar o TheHive [1][2] em containers Docker, siga os passos abaixo para instalar o Docker Engine e o Docker Compose [5] no Ubuntu:

- Instalar dependências necessárias para permitir repositórios: *sudo apt install apt-transport-https ca-certificates curl software-properties-common -y.*
- Adicionar chave GPG oficial do Docker: *curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -*.
- Adicionar repositório estável do Docker para sua versão do Ubuntu: *sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"*.

- Atualizar cache de pacotes após adicionar novo repositório: sudo apt update.
- Instalar Docker Engine e Docker Compose [5]: *sudo apt install docker-ce docker-compose -y*.
- Configurar Docker para iniciar automaticamente na inicialização do sistema: *sudo systemctl enable docker*.
- Iniciar o serviço Docker: *sudo systemctl start docker*.

C. Arquitetura do Docker Compose

O arquivo docker-compose.yml define os seguintes serviços:

- PostgreSQL: Banco de dados relacional usado para armazenar os dados da aplicação TheHive [1][2].
- Elasticsearch: Motor de busca e indexação utilizado para consultas rápidas e pesquisa avançada dos alertas e casos.
- TheHive [1][2]: Plataforma principal de gerenciamento de incidentes, configurada para conectar-se ao PostgreSQL e Elasticsearch.
- Cortex: Serviço auxiliar para análises automatizadas e enriquecimento de alertas.

O ambiente foi provisionado utilizando o seguinte arquivo Docker Compose [5], que define os serviços necessários para a execução do TheHive [1][2] e seus componentes, conforme mostrado na Tabela II.

TABELA II – ESTRUTURA DO DOCKER COMPOSE

Código do Docker Compose

```
version: "3"

services:
  postgres:
    image: postgres:13
    environment:
      POSTGRES_USER: thehive
      POSTGRES_PASSWORD: thehive
      POSTGRES_DB: thehive
    volumes:
      - pgdata:/var/lib/postgresql/data
    networks:
      - thehive_network
    healthcheck:
      test: ["CMD", "pg_isready", "-U", "thehive"]
      interval: 10s
      timeout: 5s
      retries: 5

  elasticsearch:
    image:
docker.elastic.co/elasticsearch/elasticsearch:7.16.3
    environment:
      - discovery.type=single-node
      - xpack.security.enabled=false
      - ES_JAVA_OPTS=-Xms1g -Xmx1g
    ports:
      - "9200:9200"
    networks:
      - thehive_network

  thehive:
    image: strangebee/thehive:5.2.4
    depends_on:
      postgres:
        condition: service_healthy
      elasticsearch:
        condition: service_started
    environment:
      - JAVA_OPTS=-Xms512m -Xmx2G
      - DATABASE_TYPE=postgres
      - DATABASE_HOST=postgres
      - DATABASE_PORT=5432
      - DATABASE_NAME=thehive
      - DATABASE_USER=thehive
      - DATABASE_PASSWORD=thehive
      - INDEX_TYPE=elasticsearch
      - INDEX_BACKEND=elasticsearch
      - ES_HOST=http://elasticsearch:9200
    ports:
      - "9000:9000"
    networks:
      - thehive_network

  cortex:
    image: thehiveproject/cortex:3.1.1
    depends_on:
      elasticsearch:
        condition: service_started
    environment:
      - JOB_DIRECTORY=/opt/cortex/jobs
      -
CORTEX_ELASTICSEARCH=http://elasticsearch:9200
    ports:
      - "9001:9001"
    volumes:
      - cortex_jobs:/opt/cortex/jobs
    networks:
      - thehive_network

volumes:
  pgdata:
```

Código do Docker Compose
cortex_jobs: networks: thehive_network:

D. Execução e Inicialização dos Containers

Após a criação do arquivo docker-compose.yml com a configuração previamente detalhada, os seguintes comandos foram utilizados para a implantação e monitoramento dos serviços:

- Inicialização dos containers em segundo plano (modo detach): *sudo docker-compose up -d*.
- Monitoramento em tempo real dos logs dos serviços: *sudo docker-compose logs -f*.

E. Credenciais Padrão para Acesso Inicial ao TheHive

Após o deploy, o acesso inicial ao painel web do TheHive [1][2] (url de acesso: http://<IP_DO_THEHIVE>:9000) foi realizado com as credenciais padrão:

- Usuário: admin
- Senha: secret

Por questões de segurança, recomenda-se a alteração imediata dessas credenciais após o primeiro login.

V. Configuração do Organização e Geração da Chave de API no TheHive

Para organizar os alertas, investigações e permitir o controle granular de permissões, foi criada uma nova Organização no TheHive [1][2] com as seguintes características:

- Nome da Organização: SOC Lab.
- Descrição: Ambiente de Laboratório.

O processo de criação foi realizado diretamente pela interface web do TheHive [1][2], acessível via navegador no endereço: http://<IP_DO_THEHIVE>:9000.

A. Criação da Organização

O TheHive [1][2] oferece suporte a multi-tenancy, permitindo o gerenciamento de múltiplas organizações de forma segregada dentro da mesma instância. Isso é importante para separar dados de diferentes equipes, departamentos ou clientes. Os passos para criação da organização, são:

- Acesso ao Gerenciamento de Organizações.
- No menu lateral esquerdo, clique em "Administração".
    - Em seguida, selecione a opção "Organizações".
    - Criar Nova Organização:
    - Clique em “Nova Organização” (ou “New Organization”, dependendo do idioma configurado).
    - Preencha os campos obrigatórios
- Clique em "Salvar" (Save).

B. Criação de Conta de Usuário Administrativo na Organização

Após a criação da organização no TheHive, é necessário configurar um usuário com perfil Organization Admin (org-admin) para realizar operações administrativas dentro da organização, como gerenciamento de casos, criação de alertas e geração de API Keys. Os passos para a criação da conta administrativa, são:

- Faça login com a conta de Administrador Global (exemplo: admin/secret).
- Acesse o menu lateral em "Administração", depois em "Organizações".
- Localize a organização recém-criada, por exemplo, SOC Lab.
- Clique em "Gerenciar Usuários" (Manage Users) ou diretamente acesse a aba "Usuários" dentro da organização.
- Clique em "Novo Usuário" (New User).
- Preencha os seguintes campos:
    - Nome de Usuário: soc_admin@teste.com.
    - Nome Completo: SOC Admin.
    - Senha Inicial: soc_admin@teste.com.
    - Perfil: Organization Admin (org-admin).
    - Status: Ativo.
- Confirme e salve a criação do usuário.
- Realize logout da conta de Administrador Global.

- Efetue login com o novo usuário soc_admin@teste.com.

Verifique se o usuário tem acesso aos módulos da organização (alertas, casos, tarefas, etc).

C. Geração da API Key para Integração de Automação

Para permitir que aplicações externas (como o Webhook Node.js [15] e o PowerShell usado nos testes) pudessem interagir com a API REST do TheHive [1][2], foi necessário criar uma API Key associada ao usuário soc_admin@teste.com. Os passos para geração da API Key, são:

- Realizar o login no TheHive [1][2] com a conta soc_admin@teste.com.
- No canto superior direito da interface web, clicar em "Minha Conta".
- Acessar a aba "Chaves da API" (API Keys).
- Clicar em "Nova Chave" (Generate New Key).
- Definir uma descrição para a chave, como por exemplo: Integração Webhook com LLM [3][4].
- Copiar a chave gerada.

Esta chave será utilizada nas requisições HTTP feitas pelo webhook e pelos scripts de simulação de alerta.

VI. Configuração do Listener em Node.Js

Um webhook é um mecanismo de comunicação baseado em eventos que permite que uma aplicação envie dados automaticamente para outra aplicação, assim que um evento específico ocorre. Diferente de uma API tradicional onde o cliente precisa fazer consultas periódicas (polling) para verificar atualizações, o webhook opera de forma reativa: ele "empurra" as informações para um endpoint definido assim que o evento acontece.

No contexto deste projeto, o TheHive [1][2] utiliza um webhook para notificar um serviço externo (desenvolvido em Node.js [15] com Express) sempre que um novo alerta é criado. Esse serviço, por sua vez, faz a análise do alerta utilizando uma Large Language Model (LLM) local e envia de volta um comentário diretamente para o próprio alerta no TheHive [1][2], automatizando parte do processo de análise de incidentes.

A. Requisitos do Ambiente

O Listener desenvolvido em Node.js [15] tem como função receber as requisições HTTP POST enviadas pelo TheHive [1][2] sempre que um novo alerta é criado. Ao receber o alerta, o Listener envia o conteúdo para a LLM local (Ollama + LLaMA 3 [3][4]), aguarda a análise automática e inclui a resposta como comentário no próprio alerta. Antes de executar o Listener, é necessário ter:

- Node.js [15] instalado na máquina que hospeda o Webhook (no servidor da LLM).

Para a instalação do Node.js [15] e do NPM (Node Package Manager) no ubuntu [7], recomenda-se atualizar os repositórios locais e, em seguida, realizar a instalação utilizando o comando: *sudo apt update && sudo apt install nodejs npm -y*.

Este comando garante que o sistema esteja com os pacotes mais recentes e instala tanto o Node.js [15] (runtime necessário para execução de aplicações JavaScript no servidor) quanto o NPM (ferramenta de gerenciamento de dependências para projetos Node.js [15]).

B. Template do Listener

O Listener foi desenvolvido em Node.js [15], utilizando o framework Express.js para escutar requisições HTTP e o pacote Axios para realizar requisições externas. O código abaixo representa o template base do Webhook, conforme mostrado na Tabela III.

Tabela III – Template do Listener

Código do Listener

const express = require('express');
const axios = require('axios');
 	
const app = express();
const PORT = 3000; // Porta onde o webhook ficará escutando
 	
// URL base da API do TheHive e API Key de autenticação
const THEHIVE_BASE_URL = 'http://192.168.17.103:9000/api';
const THEHIVE_API_KEY = 'SUA_API_KEY_AQUI'; // Substitua pela sua chave
 	
app.use(express.json());

Código do Listener

```
app.post('/thehive-alert', async (req, res) => {
  try {
    const alert = req.body;

    console.log('Novo alerta recebido do TheHive:');
    console.log(alert);

    const alertId = alert.objectId || alert.details?._id;
    if (!alertId) {
      throw new Error('ID do alerta não encontrado no corpo da requisição');
    }

    const title = alert.details?.title || alert.object?.title || 'Sem título';
    const description = alert.details?.description || alert.object?.description || 'Sem descrição';

    const prompt = `Analise este alerta do TheHive:\n\nTítulo: ${title}\nDescrição: ${description}\n\nQuais medidas devemos tomar?`;

    // Envia o prompt para a LLM local
    const llmResponse = await axios.post('http://localhost:11434/api/generate', {
      model: 'llama3',
      prompt: prompt,
      stream: false
    });

    const llmText = llmResponse.data.response;

    console.log('Resposta da LLM:');
    console.log(llmText);

    // Adicionar um comentário no alerta no TheHive
    const commentPayload = {
      message: `Análise automatizada via LLM:\n\n${llmText}`
    };

    await axios.post(
      `${THEHIVE_BASE_URL}/v1/alert/${alertId}/comment`,
      commentPayload,
      {
        headers: {
          Authorization: `Bearer ${THEHIVE_API_KEY}`,
          'Content-Type': 'application/json'
        }
      }
    );

    console.log('Comentário adicionado no alerta com sucesso.');

    res.json({
      status: 'success',
      llm_response: llmText
    });

  } catch (error) {
    console.error('Erro durante o processamento:', error.message);
    res.status(500).json({ status: 'error', message: error.message });
  }
});

app.listen(PORT, () => {
  console.log(`Webhook escutando na porta ${PORT}...`);
});
```

C. Instalação das Dependências

Para configurar o ambiente e executar o webhook, siga os passos abaixo:

- Crie um diretório: *mkdir ~/thehive_llm_webhook && cd ~/thehive_llm_webhook.*
- Salve o código apresentado na seção Template do Webhook em um arquivo chamado *thehive_llm_webhook.js*, dentro do diretório do projeto criado. Lembre-se de substituir o valor da variável *THEHIVE_API_KEY* pela chave de API previamente criada no TheHive [1][2], conforme descrito na seção anterior (*Geração da API Key para Integração de Automação*).
- Inicialize o projeto Node.js [15] (criando um package.json): *npm init -y*.
- Instale as dependências necessárias (Express e Axios): *npm install express axios*.
- Execute o Webhook: *node thehive_llm_webhook.js*.

O serviço ficará escutando por padrão na porta 3000, aguardando os eventos de alerta do TheHive [1][2].

VII. Configuração do Webhook no TheHive

Para permitir que o TheHive [1][2] envie informações dos alertas recém-criados para a LLM local, foi necessário configurar um Webhook para disparo automático. Este

Webhook será o responsável por enviar os dados do alerta para a aplicação Node.js [15] que faz a interface com a LLM [3][4].

O objetivo da configuração do Webhook é garantir que, toda vez que um novo alerta seja criado no TheHive [1][2], as informações básicas (como título, descrição e identificador) sejam enviadas para um endpoint HTTP (neste caso, o listener em Node.js [15] na porta 3000).

A. Etapas para Configuração do Webhook

- Acesse o painel de administração do TheHive [1][2] via navegador: http://<IP_do_TheHive>:9000.
- Faça login com a conta administrativa criada anteriormente.
- No canto superior direito, clique no avatar da conta e selecione "Administração" (Administration).
- No menu lateral esquerdo, clique em "Webhooks".
- Clique em "Criar Webhook" (Create Webhook).
- Preencha os campos obrigatórios conforme abaixo:
    - Nome: Webhook LLM.
    - Descrição: Envio de novos alertas para análise via LLM.
    - URL do Listener em Node.js [15]:
        - http://<IP_do_Servidor_LLM>:3000/thehive-alert.
        - Method: POST
        - Content Type: application/json.
    - Trigger on: Alert created (ou seja, apenas quando novos alertas forem criados).

B. Teste do Webhook via Simulação de Alerta

Para garantir o correto funcionamento da integração entre o TheHive e a LLM local, foi desenvolvido um script de simulação que envia um alerta ao TheHive [1][2] utilizando requisições HTTP POST. Essa abordagem permite validar todo o fluxo de processamento, desde a criação do alerta até a análise automatizada via Webhook e a inserção da resposta da LLM como comentário no alerta.

O script simula um cenário realista de alerta, similar ao que seria gerado por ferramentas de monitoramento como Wazuh [14] ou outros sistemas SIEM. Essa simulação facilita a realização de testes controlados e a validação do comportamento do sistema em um ambiente de laboratório, conforme mostrado na Tabela IV.

TABELA IV – SCRIPT PARA SIMULAÇÃO DE ALERTA

Script de Simulação de Alerta

```
$thehiveUrl = "http://192.168.17.103:9000/api/alert"
$apiKey = "THEHIVE_API_KEY"

$body = @{
   title       = "Alerta Crítico: Tentativas de Brute Force
SSH detectadas"
   type        = "external"
   source      = "wazuh"
   sourceRef   = "ssh-bruteforce-007"
   description = @"
Foram detectadas múltiplas tentativas de login via SSH no
servidor 192.168.1.50.
Recomendamos análise imediata dos logs do servidor e
bloqueio do IP atacante.
Possível ataque de força bruta visando comprometer o
sistema.
"@

   severity      = 3  # 0 (informativo) até 4 (crítico)
   severityLabel = "HIGH"
   tlp           = 2  # Traffic Light Protocol: 0 (public) até 3
(restricted)
   tlpLabel      = "AMBER"
   tags          = @("ssh", "bruteforce", "security", "wazuh")

   artifacts = @(
      @{
         dataType = "ip"
         data     = "192.168.1.100"
         message  = "IP suspeito de realizar as tentativas de
acesso"
         tags     = @("attacker", "suspicious")
      },
      @{
         dataType = "hostname"
         data     = "server-192-168-1-50.local"
         message  = "Servidor alvo das tentativas de login"
         tags     = @("target", "ssh")
      }
   )
} | ConvertTo-Json -Depth 6

$utf8Body =
[System.Text.Encoding]::UTF8.GetBytes($body)

$response = Invoke-RestMethod -Uri $thehiveUrl -Method
Post -Headers @{ Authorization = "Bearer $apiKey" }
-Body $utf8Body -ContentType "application/json"

$response | ConvertTo-Json -Depth 5
```

O script PowerShell para envio do alerta pode ser executado diretamente na máquina host que hospeda as VMs, pois as máquinas virtuais (TheHive e LLM) estão configuradas em rede NAT. Dessa forma, a comunicação entre o host e as VMs ocorre normalmente via endereço IP da rede NAT, facilitando os testes e a integração sem necessidade de acesso direto às VMs via rede externa.

A chave de API (API Key) utilizada no script para autenticação nas requisições ao TheHive [1][2] deve ser previamente criada e configurada no sistema. Lembre-se de substituir o valor da variável *THEHIVE_API_KEY* pela chave de API previamente criada no TheHive [1][2], conforme descrito na seção anterior (*Geração da API Key para Integração de Automação*).

VIII. Resultados

Nesta seção, é apresentado o fluxo completo de integração entre o TheHive [1][2] e a Large Language Model (LLM) local, demonstrando a criação de alertas, a ativação do webhook, a análise automática do alerta pela LLM e o retorno da recomendação diretamente no incidente do TheHive [1][2].

A. Geração do Alerta no TheHive

O processo se inicia com a criação de um alerta no TheHive [1][2], que pode ser feito manualmente, via API ou, no caso de ambientes produtivos, por meio de um SIEM (exemplo: Wazuh [14]). Para fins de teste, utilizamos um script PowerShell que simula um alerta crítico relacionado a tentativas de ataque brute force SSH, conforme a Fig. 1.

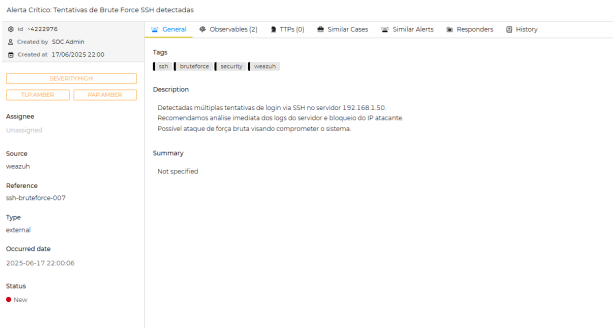

Fig. 1 - Tela do TheHive mostrando o alerta recém-criado

B. Ativação do Webhook e Recebimento do Alerta

Assim que o alerta é criado, o TheHive envia uma notificação para o webhook configurado. Nosso serviço webhook, implementado em Node.js [15], recebe o alerta e extrai informações relevantes, como título, descrição e ID do alerta, conforme a Fig. 2.

Fig. 2 - Logs do webhook mostrando o alerta recebido

C. Chamada para a LLM Local e Geração da Resposta

Com o alerta recebido, o webhook monta um prompt e envia para a LLM local (rodando via Ollama com o modelo LLaMA 3 [3][4]) através da API HTTP. A LLM processa o prompt e retorna uma análise detalhada, contendo recomendações e próximos passos sugeridos para o time de segurança, conforme a Fig. 3.

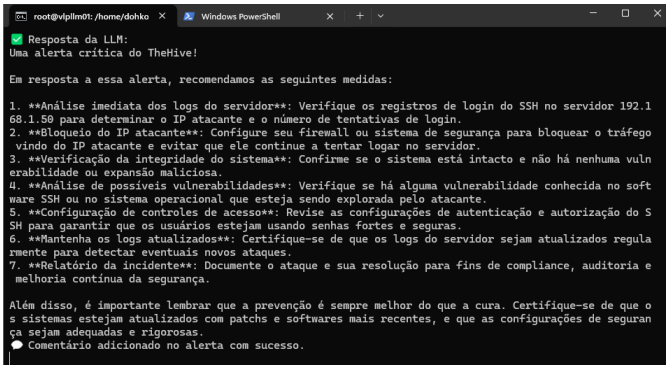

Fig. 3 - Resposta da LLM no terminal do webhook

D. Inserção do Comentário no Alerta do TheHive

Após receber a resposta da LLM, o webhook publica automaticamente essa análise como um comentário no alerta correspondente no TheHive [1][2]. Isso permite que os analistas vejam imediatamente a recomendação gerada pela inteligência artificial, agilizando a resposta ao incidente, conforme a Fig. 4.

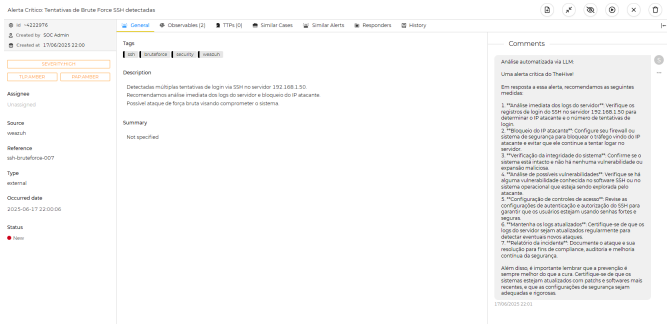

Fig. 4 - Comentário automatizado inserido no alerta no TheHive

E. Observações sobre o Desempenho e Limitações

Durante os testes, notou-se que o tempo para geração das recomendações pela LLM foi um pouco maior que o esperado, devido às limitações de hardware do servidor onde a LLM está instalada (8 GB RAM, CPU 2x4 núcleos, sem GPU). Em ambientes de produção, recomenda-se a utilização de servidores com maior capacidade e aceleração via GPU para reduzir a latência no processamento.

F. Considerações Finais

O fluxo demonstrado comprova a viabilidade de integrar o TheHive [1][2] a uma LLM local [3][4] para automação da análise de alertas. A abordagem pode ser facilmente adaptada para ingestão de alertas via SIEMs [12] como o Wazuh [14], aumentando significativamente a eficiência da resposta a incidentes de segurança.

IX. Conclusão

Este estudo demonstrou, de forma prática, a viabilidade e os benefícios da integração entre uma plataforma de gerenciamento de incidentes (TheHive) e uma Large Language Model (LLM) local, utilizando uma arquitetura de baixo custo baseada em infraestrutura virtualizada.

A implementação de um serviço webhook intermediário, desenvolvido em Node.js, permitiu a automatização da análise inicial de alertas de segurança, com a geração de recomendações contextuais por meio da LLM LLaMA 3. Mesmo em um ambiente de laboratório com recursos computacionais limitados, a solução foi capaz de receber alertas, processá-los com auxílio da inteligência artificial e devolver respostas em formato de comentários diretamente nos registros do TheHive.

Os resultados obtidos evidenciam o potencial de soluções baseadas em LLMs para apoiar analistas de segurança na triagem inicial de eventos, otimizando o tempo de resposta e reduzindo a sobrecarga operacional em SOCs (Security Operations Centers).

Como limitações observadas, destacam-se o tempo de inferência da LLM quando executada apenas com CPU e a necessidade de ajustes de infraestrutura para ambientes com maior volume de alertas.

Para futuras implementações em ambientes produtivos, recomenda-se:

- Execução da LLM com aceleração via GPU para reduzir a latência;
- Uso de banco de dados e armazenamento de logs com alta disponibilidade;
- Adoção de práticas de segurança para proteger a API Key e o endpoint do webhook;
- Integração com SIEMs para ingestão automatizada de alertas em grande escala.

Por fim, este projeto reforça a aplicabilidade de tecnologias de IA generativa no contexto de cibersegurança, abrindo espaço para novas pesquisas e soluções voltadas ao apoio à decisão em processos de resposta a incidentes.

Agradecimentos

Gostaria de expressar meus sinceros agradecimentos a todos os docentes do curso de Pós-Graduação em Segurança da Informação e Inteligência Defensiva da CECyber, cuja orientação e dedicação foram fundamentais para a realização deste trabalho. Em especial, registro minha profunda gratidão ao professor Eduardo R. Sant'Ana Popovici, cuja condução das disciplinas de IA e ML em Cybersecurity e apoio durante todo o curso serviram de base técnica e conceitual para o desenvolvimento deste estudo.

Os conhecimentos adquiridos ao longo das aulas e atividades práticas foram essenciais para a concepção e implementação deste projeto de integração entre o TheHive e a LLM local, reforçando a importância de uma formação sólida e alinhada com os desafios atuais da cibersegurança.

Agradeço também a todos os colegas de turma e aos profissionais da área que, direta ou indiretamente,

contribuíram com ideias, sugestões e discussões que enriqueceram esta pesquisa.

Referências

[1] TheHive Project, “TheHive: Scalable, Open Source and Free Security Incident Response Platform.” [Online]. Disponível em: https://thehive-project.org/. Acesso em: jun. 2025.

[2] StrangeBee, “TheHive 5.x Documentation.” [Online]. Disponível em: https://docs.strangebee.com/thehive/. Acesso em: jun. 2025.

[3] Ollama, “Run open-source large language models locally.” [Online]. Disponível em: https://ollama.com/. Acesso em: jun. 2025.

[4] Meta AI, “Llama 3: Open Foundation and Instruction Models.” [Online]. Disponível em: https://ai.meta.com/llama/. Acesso em: jun. 2025.

[5] Docker Inc., “Docker Documentation.” [Online]. Disponível em: https://docs.docker.com/. Acesso em: jun. 2025.

[6] VMware Inc., “VMware vSphere Documentation.” [Online]. Disponível em: https://docs.vmware.com/en/VMware-vSphere/index.html. Acesso em: jun. 2025.

[7] Canonical Ltd., “Ubuntu Server 22.04 LTS Documentation.” [Online]. Disponível em: https://ubuntu.com/server/docs. Acesso em: jun. 2025.

[8] Microsoft, “PowerShell Documentation - Invoke-RestMethod.” [Online]. Disponível em: https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/invoke-restmethod. Acesso em: jun. 2025.

[9] OpenAI, “ChatGPT: Optimizing Language Models for Dialogue.” [Online]. Disponível em: https://openai.com/blog/chatgpt. Acesso em: jun. 2025.

[10] IBM Cloud Education, “What is Artificial Intelligence (AI)?” [Online]. Disponível em: https://www.ibm.com/cloud/learn/what-is-artificial-intelligence. Acesso em: jun. 2025.

[11] Google Cloud, “What is Machine Learning?” [Online]. Disponível em: https://cloud.google.com/learn/what-is-machine-learning. Acesso em: jun. 2025.

[12] Microsoft News, V. Jakkal, “How AI is Transforming Cybersecurity: Tackling the Surge in Cyber Threats,” News.Microsoft.com: Source Canada. [Online]. Disponível em: https://news.microsoft.com/source/canada/features/ai/how-ai-is-transforming-cybersecurity-tackling-the-surge-in-cyber-threats/. Acesso em: jun. 2025.

[13] P. Kent e M. Souppaya, Guide to Security Information and Event Management (SIEM) Technologies, NIST Special Publication 800-137, 2021. [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-137/final

[14] Wazuh, Wazuh Documentation, 2024. [Online]. Available: https://documentation.wazuh.com/current/

[15] Node.js Foundation, Node.js Documentation, 2024. [Online]. Available: https://nodejs.org/en/docs/