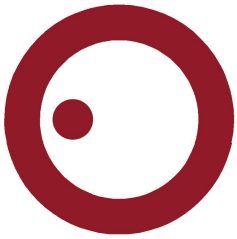


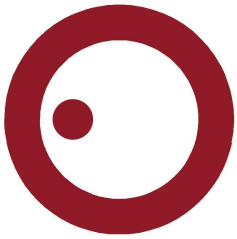
escola superior de tecnologia e gestão
instituto politécnico de leiria

Instituto Politécnico de Leiria
Escola Superior de Tecnologia e Gestão
Departamento de Engenharia Informática
Mestrado em Cibersegurança e Informática Forense

INTEL ANALYSIS - APLICAÇÃO PARA ANÁLISE DE DADOS DE SEGURANÇA

ESTUDANTE MARTINHO JOSÉ FERREIRA GONÇALVES

Leiria, Setembro de 2024



escola superior de tecnologia e gestão
instituto politécnico de leiria

Instituto Politécnico de Leiria
Escola Superior de Tecnologia e Gestão
Departamento de Engenharia Informática
Mestrado em Cibersegurança e Informática Forense

INTEL ANALYSIS - APLICAÇÃO PARA ANÁLISE DE DADOS DE SEGURANÇA

ESTUDANTE MARTINHO JOSÉ FERREIRA GONÇALVES
Número: 2220280

Projeto realizado sob orientação do Professor Especialista Carlos Manuel Gonçalves Antunes (carlos.antunes@ipleiria.pt).

Leiria, Setembro de 2024

AGRADECIMENTOS

Gostava de dedicar este pequeno texto a todos os que me ajudaram no meu percurso académico e pessoal e que de alguma forma contribuíram para o meu sucesso. Sem eles este projeto não seria a mesma coisa.

Agradeço ao meu orientador, professor Carlos Antunes, por todo acompanhamento ao longo de todo o projeto, por ser incansável e por estar disposto a ajudar sempre que foi necessário e principalmente pelos incentivos que foram sem dúvida uma mais valia para a concretização deste projeto.

A todos os professores do Mestrado em Cibersegurança e Informática Forense que me acompanharam nestes dois anos e foram essenciais para a minha formação académica, bem como a todos os meus colegas com os quais tive a oportunidade de partilhar esta experiência e conhecimentos.

Sou muito grato à minha família por todo o apoio demonstrado durante o mestrado, em especial aos meus pais, Arlindo e Adelina, que me incentivaram e me deram força para abraçar este desafio. Sem eles isto não seria possível. Aos meus irmãos, Ana e Ricardo, obrigado por todo o apoio e por estarem sempre ao meu lado.

Por fim, um agradecimento especial à minha namorada Mariana, que me acompanhou diariamente neste desafio, que me deu o seu apoio incondicional mesmo nos momentos mais complicados e que sempre me incentivou e encorajou a fazer melhor. Esteve sempre do meu lado e fez muito por mim no decorrer destes dois anos.

A todos, o meu muito sincero obrigado!

RESUMO

As crescentes necessidades das organizações aplicarem planos e soluções a cibersegurança tem-se revelado um problema difícil de gerir, principalmente para as pequenas e médias organizações. Estas organizações tem pouca capacidade financeira tanto para a aquisição de sistemas adequados para a segurança da organização como para a contratação de mão de obra especializada no tema da cibersegurança. Esta responsabilidade recai nas equipas de informática destas organizações que grande parte das vezes não tem a capacidade, conhecimentos, ferramentas e tempo necessários para dar resposta a este problema.

Detetado este défice nestas organizações e a falta de ferramentas no mercado que sejam capazes de uma forma muito simples e sem grandes requisitos ao nível de conhecimentos da informação importante sobre a cibersegurança da organização, propôs-se o estudo e desenvolvimento de uma solução capaz de mostrar que com pouco esforço é possível obter informação extremamente relevante. A aplicação desenvolvida em formato de demonstrador, tem como principais objetivos facilitar a utilização da ferramenta oferecendo um ambiente gráfico simples e intuitivo de utilizar, permitir que seja colecionados automaticamente dados relevantes externos á organização, permitir recolher dados internos á organização como tráfego de rede e ficheiros de *log* e por fim, cruzar toda a informação de modo a que sejam gerados alertas indicando possíveis anomalias na infraestrutura.

Esta aplicação permite que com poucos recursos computacionais e através da informação recolhida externa e internamente, seja possível gerar alertas capazes de identificar anomalias nas infraestruturas. Estas anomalias são detetadas pela comunicação entre máquinas na rede e endereços IP ou nomes de domínio marcados como possíveis ameaças, despoletando assim alertas para que seja possível uma investigação mais pormenorizada.

Este tipo de aplicações, simples de utilizar e que requeiram poucos conhecimentos para a sua utilização e que acima de tudo sejam capazes de trazer utilidade para as equipas de informática, são sem duvida uma mais valia para as organizações.

ABSTRACT

The growing need for organizations to implement cybersecurity plans and solutions has proven to be a difficult challenge to manage, especially for small and medium-sized organizations. These organizations have limited financial capacity, both for acquiring adequate security systems and for hiring specialized labor in the field of cybersecurity. This responsibility often falls on the IT teams of these organizations, which frequently lack the capacity, knowledge, tools, and time needed to address this issue.

Recognizing this deficit in organizations and the lack of tools in the market that can provide, in a simple way and without extensive knowledge requirements, important information about an organization's cybersecurity, a solution is proposed to show that with minimal effort, highly relevant information can be obtained. The application developed as a demonstrator aims to facilitate the use of the tool by offering a simple and intuitive graphical environment, enabling the automatic collection of relevant external data, allowing for the collection of internal data such as network traffic and log files, and finally, cross-referencing all the information to generate alerts indicating potential anomalies in the infrastructure.

This application allows, with few computational resources and through the collection of both external and internal information, for the generation of alerts capable of identifying anomalies in infrastructures. These anomalies are detected by communication between machines on the network and IP addresses or domain names flagged as potential threats, thereby triggering alerts for more detailed investigation.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

API	Application Programming Interface.
AS	Autonomous System.
C2	Command and Controll.
CIS	Center for Internet Security.
CMS	Content Management System.
CPU	Central Processing Unit.
CRUD	Create, Read, Update and Delete.
CSV	Comma-separated values.
DDoS	Distributed Denial of Service.
DGA	Domain Generation Algorithm.
DNS	Domain Name System.
ENISA	Agência Europeia para a Segurança das Redes e da Informação.
FTP	File Transfer Protocol.
HTTP	Hypertext Transfer Protocol.
IA	Inteligência Artificial.
IDS	Intrusion Detection System.
IoC	Indicator of Compromise.
IP	Internet Protocol.
IPS	Intrusion Protection System.

JSON	JavaScript Object Notaion.
NAT	Network Address Translation.
NLP	Natural Language Processing.
OSI	Open Systems Interconnection.
PID	Process Identifier.
ptr	DNS Pointer Record.
RAM	Random access memory.
SEF	Security Enforcing Function.
SEM	Security Event Management.
SIEM	Security Information and Event Management.
SIM	Security Information Management.
SNMP	Simple Network Management Protocol.
SPA	Single Page Application.
SSH	Secure Shell.
TCP	Transmission Control Protocol.
TLD	Top Level Domain.
UDP	User Datagram Protocol.
URL	Uniform Resource Locator.

1

INTRODUÇÃO

Os sistemas de informação estão em constante desenvolvimento e esta evolução traz vários problemas para as organizações, nomeadamente as necessidades constantes de atualizações de sistemas e de aquisição de conhecimentos e mão de obra qualificada para abraçar os problemas que vão surgindo. Aliado ao crescimento informático está o crescimento contínuo de ações maliciosas que são uma grande ameaça para as organizações. Isto leva à necessidade destas organizações investirem em mão de obra qualificada, em conhecimento e tecnologias capazes de dar resposta às ameaças à segurança.

Este investimento que pode ser feito pelas organizações é, em muitos casos, muito difícil ou até mesmo impossível pois podem ser investimentos de grande valor, tanto a nível de recursos humanos como de tecnologia. Isto demove as organizações com menor poder financeiro de investir na cibersegurança e a deixar a mesma para segundo plano. Isto está a tornar-se um problema em grande escala para o negócio das organizações que, ao ignorarem a cibersegurança, põem em causa o seu negócio e a continuidade da organização no mercado pois um ciberataque pode levar a danos irreversíveis e, consequentemente, ao fecho da organização.

Assim, apesar de já existirem soluções indicadas para a cibersegurança, estas são de extrema dificuldade tanto de implementação como de aprendizagem sendo ainda financeiramente dispendiosas, não sendo viável a sua utilização por pequenas e médias organizações. Existe, assim, falta no mercado de soluções capazes de munir as equipas de informática destas organizações, que têm poucas capacidades, conhecimentos ou até tempo para dispensar ao tema da cibersegurança. Há necessidade de criar uma solução que seja simples, fácil de utilizar e que consiga obter informação útil para a organização oferecendo uma visão de alto nível sobre o seu panorama de cibersegurança.

1.1 OBJETIVOS

Este projeto surgiu com base na falta de soluções no mercado que se adaptem às pequenas e médias organizações, sem o poder financeiro e/ou os recursos humanos adequados às necessidades de cibersegurança dos tempos que correm. O objetivo passa por desenvolver uma aplicação que constrói uma base de informação relevante através de fontes de dados externas e que, através desta informação, gera alertas cruzando informação interna à organização, como tráfego de rede e ficheiros de *log*. Estas fontes de dados externos assentam na partilha da informação relevante para a cibersegurança entre organizações e através desta é possível tomar decisões. Esta aplicação tratar-se-á apenas de um demonstrador de conceito, para poder aferir a robustez do mesmo. A aplicação deverá:

- Ser uma solução *opensource*;
- Ser de fácil utilização e análise;
- Requerer poucos conhecimentos de redes e serviços;
- Recolher informação externa à organização;
- Analisar os dados internos recolhidos com base nos dados recolhidos externamente;
- Gerar alertas.

Dados os objetivos acima definidos, deve desenvolver-se um demonstrador capaz de corresponder a estas necessidades.

1.2 PLANEAMENTO DAS ATIVIDADES

Para desenvolver o projeto acima referido foram desempenhadas várias tarefas ao longo dos últimos 11 meses. Estas tarefas permitiram numa fase inicial perceber o estado da arte no que diz respeito à cibersegurança no geral e as aplicações existentes no mercado através da recolha de informação. Esta recolha teve como objetivo entender o que são ameaças à cibersegurança estudando também a sua taxonomia. A identificação de possíveis fontes internas e externas de dados foi um dos passos cruciais para a implementação do projeto pois este tem nestas fontes os dados cruciais para o funcionamento do mesmo. O estudo de ferramentas já existentes no mercado também foi importante pois permitiu perceber o que já existe e que tipo de tarefas estas oferecem.

Aliadas às ferramentas utilizadas pelos profissionais de cibersegurança, estão também as técnicas por estes utilizadas ou postas em prática através das aplicações. Isto permite perceber que técnicas a utilizar para deteção de anomalias. O passo seguinte, e tendo em conta toda a informação já recolhida, foi abordar a arquitetura da aplicação e como esta deve estar dividida bem como fazer testes às tecnologias a utilizar para perceber se estas se encaixam no que é necessário. Com a arquitetura e tecnologias definidas, e tendo em conta já algum desenvolvimento realizado, segue-se o desenvolvimento tanto do *backend* como do *frontend*. Por último, realizaram-se todos os testes necessários à aplicação. Durante todo o desenvolvimento do projeto foi escrito o presente relatório. A figura 1, apresenta o mapa de *Gant* com o planeamento das tarefas.

Mês / Tarefas	Outubro	Novembro	Dezembro	Janeiro	Fevereiro	Março	Abril	Maio	Junho	Julho	Agosto
Recolha de informação											
Ameaças											
Fontes internas e externas											
Ferramentas											
Técnicas											
Proposta de arquitetura											
Tecnologias utilizadas											
Desenvolvimento - *backend*											
Desenvolvimento - *frontend*											
Testes											
Relatório											

Figura 1: Mapa de *Gant* com o planeamento das tarefas

1.3 ESTRUTURA DO DOCUMENTO

O presente documento está dividido em cinco capítulos diferentes que abordam as tarefas delineadas no ponto anterior. Após a introdução, o segundo capítulo retrata o estado da arte abordando temas como as ameaças, os dados relevantes e as fontes de dados internas e externas. O terceiro capítulo aborda as ferramentas utilizadas por equipas de cibersegurança e quais os objetivos de cada uma. São também abordadas técnicas utilizadas para deteção de anomalias. O quarto capítulo fala sobre a arquitetura da aplicação e está dividido em duas partes. A primeira é a arquitetura lógica da aplicação onde são abordados os pontos principais da aplicação e como está dividida. A segunda parte aborda as ferramentas e técnicas que serão utilizadas para desenvolver a aplicação. O quinto capítulo aborda o desenvolvimento da aplicação, falando sobre as linguagem de programação utilizadas, o método de desenvolvimento bem como as bibliotecas utilizadas. Neste capitulo é apresentado algum código e imagens do protótipo e é explicado como este funciona. De referir que todo o código está disponível no *Github*, mais à frente referenciado. Ainda no

capítulo cinco, são explicados os testes realizados ao demonstrador. Por fim, no capítulo seis, são tiradas as conclusões de todo o trabalho desenvolvido.

2

ESTADO DA ARTE

Este capítulo destina-se a contextualizar o estado atual da cibersegurança ao nível empresarial, abordando temas como a caracterização das ameaças de segurança que as organizações são alvo, a informação relevante que deve ser recolhida e analisada em caso de incidente de cibersegurança e, por fim, os vários tipos de fontes de informação disponíveis para ajudar na deteção e mitigação de possíveis problemas.

2.1 AMEAÇAS DE SEGURANÇA

De forma a poder dar a melhor resposta a um incidente de segurança, ou neste caso, obter a informação mais adequada sobre os ataques, é de elevada importância conhecer as potenciais ameaças de segurança a uma organização. Esta secção destina-se a explicar como pode ser feita a caracterização destas ameaças. É também importante perceber quais as motivações que possam levar um atacante a iniciar uma campanha contra uma organização.

As vulnerabilidades de segurança são causa potencial de incidente indesejável que pode resultar em danos para uma organização ou para qualquer um dos sistemas por ela utilizado, tornando-se ameaça às organizações. Estas ameaças podem ser acidentais ou deliberadas (com dolo) e caracterizam-se por elementos ameaçadores, alvos potenciais e métodos de ataque (Europeu, 2013). Estas ameaças podem ser utilizadas por um agente malicioso de modo a ganhar acesso aos sistemas ou informação de uma dada organização quebrando assim a confidencialidade, disponibilidade e integridade dos dados. A esta prática pode chamar-se de ataque (CNCS, Acedido a 2023-10-06). Estas ameaças de segurança são exploradas através de vários tipos de ataques, que podem ser englobados nos seguintes grupos (M. Uma, 2013):

1. Propósito;
2. Classificação legal;
3. Envolvimento;

4. Âmbito;
5. Tipo de rede.

Os grupos acima referidos englobam um conjunto de tipos de ataque. Podemos desta forma, para cada um dos grupos, enumerar um conjunto de ataques. Os ataques de propósito têm um objetivo simples e claro e podem englobar-se os seguintes tipos:

1. Ataques de reconhecimento;
2. Ataques de acesso;
3. Ataques de negação de serviço.

No que diz respeito à classificação legal, caracterizamos os ataques com base no tipo de crime que um dado ataque infringe. Podemos assim caracterizar da seguinte forma:

1. *Cyber crime*;
2. *Cyber espionage*;
3. *Cyber terrorism*;
4. *Cyberwar*.

Os ataques devem também ser classificados consoante o nível de envolvimento do atacante, isto é, se o atacante interage diretamente com os dados e os altera de alguma foram ou se apenas observa. Neste sentido podem ser classificados como:

1. Ataques passivos;
2. Ataques ativos.

No que diz respeito ao âmbito dos ataques, estes são classificados quanto à intenção do mesmo, isto é:

1. Malicioso
2. Não malicioso

Por fim, podemos classificar os incidentes com base no tipo de rede em que se propagam, ou seja, se são ataques a uma rede *wireless* ou à rede interna de uma organização, podendo ser classificados como:

1. Ataques à rede local (LAN)
2. Ataques à internet (WAN)
3. Ataques à rede *Wi-Fi*

4. Ataques a redes IoT
5. Ataques de engenharia social

Como descrito anteriormente, há muitos ataques/ameaças que se enquadram em vários dos grupos definidos. É importante, então, classificar os ataques mais concretamente. A classificação de ataques e ameaças à segurança das organizações é denominada de taxonomia e tem como objetivo ser uma nomenclatura simples de entender e facilitar a troca de informação entre organizações (Segurança das Redes e da Informação, 2018). Existem vários autores de taxonomia e todos diferem um pouco na sua classificação dos ataques, contudo, englobam a grande parte das ameaças dos dias de hoje. A Agência Europeia para a Segurança das Redes e da Informação (ENISA) disponibiliza um documento onde é feita uma comparação entre as taxonomias existentes, sendo que todas acabam por abranger os mesmos ataques (Segurança das Redes e da Informação, 2018). Desta forma, uma das taxonomias que é bastante utilizada ou referenciada é a do CERT.PT. Esta taxonomia está dividida em duas partes: a classe (um grupo de vários tipos de ataques) e o tipo (que representa uma ameaça concreta à organização). A tabela 1 mostra a taxonomia utilizada pelo CERT.PT que permite perceber as ameaças que uma organização pode ter de enfrentar.

No âmbito deste trabalho, é importante perceber a taxonomia dos ataques para entender quais são as ameaças à segurança dos sistemas e informação que uma organização enfrenta. Conhecer as ameaças que existem é o primeiro passo para uma defesa bem construída pois possibilita à equipa responsável pela segurança de uma organização planear e melhorar as suas defesas da melhor forma. Este conhecimento permite que saibamos o que procurar e onde procurar evidências ou informação relevante que nos pode indicar um possível ataque.

CLASSE	TIPO
Código Malicioso	Sistema Infetado
	Distribuição de *Malware*
	Servidor C2
	Configuração de *Malware*
Disponibilidade	Negação de Serviço
	Negação de Serviço Distribuída
	Configuração Incorreta
	Sabotagem
	Interrupção
Recolha de Informação	*Scanning*
	Sniffing
	Engenharia Social
Intrusão	Compromisso de conta privilegiada
	Compromisso de conta não privilegiada
	Compromisso de aplicação
	Arrombamento
Tentativa de Intrusão	Exploração de vulnerabilidades
	Tentativa de *login*
	Nova assinatura de ataque
Segurança da Informação	Acesso não autorizado
	Modificação não autorizada
	Perda de dados
Fraude	Utilização indevida ou não autorizada de recursos
	Direitos de autor
	Utilização ilegítima de nome de terceiros
	Phishing
Conteúdo Abusivo	*SPAM*
	Discurso Nocivo
	Exploração sexual de menores, racismo e apologia da violência
Vulnerabilidade	Criptografia fraca
	Amplificador DDoS
	Serviços acessíveis potencialmente indesejados
	Revelação de informação
	Sistema vulnerável
Outro	Sem tipo
	Indeterminado

Tabela 1: Taxonomia CERT.PT

2.2 INFORMAÇÃO RELEVANTE

O conhecimento das ameaças de segurança, tal como já referido, é importante para saber como estas funcionam. Cada uma das ameaças acima enumeradas deixa um rasto de informação que pode ser relevante para a prevenção, investigação, mitigação e/ou resolução de um problema. Para isto, as organizações devem estar preparadas para recolher vários tipos de dados nas suas infraestruturas, de modo a poderem ser analisados por especialistas. A informação relevante que pode ser recolhida neste processo pode ser relativa a vários pontos diferentes de uma comunicação. Os dados recolhidos podem ser transformados em Indicator of Compromise (IoC) e desta forma utilizados para tomar ações, como por exemplo monitorizar ou bloquear comunicações. IoC são artefactos forenses usados para identificar potenciais atividades maliciosas num sistema ou rede comprometido. (Ivo Vacas, 2018) Segundo a Kaspersky, estes são alguns exemplos de IoC (Kaspersky, Acedido a 2023-10-09):

1. *Lookups* de DNS;
2. Ficheiros, aplicações e processos suspeitos;
3. Endereços de IPs e nomes de domínios que possam pertencer a *botnets* ou servidores de Command and Controll (C2);
4. Número significante de acesso a um ficheiro;
5. Atividade suspeita em contas privilegiadas ou não privilegiadas;
6. Atualizações de software inesperadas;
7. Transferência de dados por portos pouco utilizados;
8. Comportamento estranhoos num *website* incomum a seres humanos;
9. Uma assinatura de ataque ou *hash* de um ficheiro conhecido como *malware*;
10. Tamanhos incomuns nas respostas HTTP;
11. Mudanças não autorizadas em ficheiros de configurações, registo ou definições de equipamentos;
12. Elevado número de tentativas falhadas de *login*.

A esta lista podemos ainda adicionar os seguintes IoC:

1. Grandes volumes de tráfego;
2. Padrões anómalos de rede;
3. Grandes volumes de exfiltração de dados.

2.3 FONTES DE INFORMAÇÃO

De modo a que seja possível analisar a informação, é importante perceber onde esta poderá estar a ser armazenada ou onde poderá ser obtida/consultada. Existem várias fontes de dados que recolhem informação relevante. Podemos classificar estas fontes em duas grande categorias: fontes internas e fontes externas.

2.3.1 *Fontes internas*

As fontes internas são conjuntos de dados gerados e recolhidos dentro da organização, ou seja, a informação está contida nos seus vários sistemas (equipamentos de rede, de segurança de perímetro ou *endpoints*).

No que diz respeito às fontes internas, podemos encontrar informação através de (Onwubiko, 2018-11-06):

1. Logs/Eventos de segurança - Podem ser de qualquer equipamento de rede (equipamentos de Layer 2 e 3), serviço disponibilizado ou sistema operativo existente;
2. Security Enforcing Function (SEF) - Qualquer equipamento ou sistema de rede cuja função é garantir algum tipo de segurança. Nesta categoria podemos encontrar os Intrusion Detection System (IDS), Intrusion Protection System (IPS), *firewalls*, sistemas de antivírus, entre outros;
3. Fluxos de rede - São um conjunto de pacotes de rede que passam por um ponto de observação de rede num determinado período. Esta é uma fonte bastante importante de informação, pois permite detetar anomalias no tráfego da rede e as resoluções de DNS de uma organização que podem indicar comprometimentos;
4. Sessões - As sessões são o plano de controlo entre uma comunicação TCP, ou seja, entre o cliente e o servidor. As sessões são estabelecidas após o TCP *3-way Handshake* e armazenam informação relevante relativa à comunicação, como a duração ou informação enviada/recebida.

2.3.2 *Fontes externas*

As fontes externas à organização são todos os dados que podem ser recolhidos através de fontes presentes na Internet e que podem ser acedidos por todos, ou seja, são dados que são gerados fora da organização.

A partilha de informação no âmbito da cibersegurança é um fator importante que permite a melhoria da segurança das organizações. Estas fontes contribuem com informação sobre ameaças externas à organização e é possível obter várias informações pertinentes através deste tipo de informação. Relativamente às fontes externas, estas podem ser encontradas através de:

1. *Streams* de dados - *Feeds* de informação ou *Threat Intelligence* que são utilizados na forma de redes sociais, *blogs*, fóruns, comunidades de segurança e fontes livres ou pagas de dados. Esta informação pode ser disponibilizada por governos, entidades de segurança ou empresas de cibersegurança. (**Alves2017**)

Como já referido anteriormente, estas fontes podem ser de acesso público ou pagas e têm várias origens. As fontes externas são utilizadas na sua maioria pelas organizações para *threat hunting*. Desta forma, com a informação partilhada, as organizações têm uma visão mais abrangente do panorama do ciberespaço, como por exemplo, de ameaças ou indicadores de possíveis ameaças. Existe uma panóplia de fontes de segurança que podem ser consultadas e consumidas pelas organizações de forma a aumentar o nível de segurança das mesmas. Nestas fontes de informação encontramos, entre outros, os seguintes dados disponíveis:

1. IPs e nomes de domínio de Command and Controll;
2. URL maliciosos;
3. *Blacklists* de IPs maliciosos;
4. IPs conhecidos por ataques a vários serviços (IMAP, POP3, Servidores Apache, Secure Shell , Postfix, etc..);
5. Informação relativa a certificados digitais.

Existem ferramentas especializadas em recolher e compilar a informação recolhida nestas fontes de dados como é o caso do *IntelMQ*. O *IntelMQ* disponibiliza na sua documentação um conjunto destas fontes (gratuitas e comerciais) que podem ser consultadas. Geralmente, os dados partilhados por estas fontes são em formato JSON, texto ou CSV e ao serem partilhados nestes formatos podem facilmente ser consumidos por uma API, tratados e utilizados.

De seguida, são apresentados dois exemplos de fontes de dados. O primeiro, representado pela listagem 1, é referente à fontes de dados *Feodo Tracker*, que fornece uma lista de Command and Controll *servers* referentes às *botnets* dos *malwares Dridex* e *Emotet* em formato JSON. Na informação disponibilizada podemos encontrar vários dados que permitem identificar concretamente a ameaça e consequentemente atuar sobre a mesma. Nesta fonte é possível obter os seguintes dados:

1. Endereço Internet Protocol (IP) do servidor do C2;
2. Porto utilizada pelo *malware* para comunicação;
3. O estado do servidor;
4. O número do Autonomous System (AS);
5. O nome do AS;
6. O país onde se encontra;
7. A data e hora em que foi identificado;
8. A data da última vez que foi visto *online*;
9. O nome do *malware* a que está associado.

Listagem 1: Exemplo de dados recolhidos da fonte de dados Feodo Tracker

```
[
    {
        "ip_address": "178.128.23.9",
        "port": 4125,
        "status": "online",
        "hostname": null,
        "as_number": 14061,
        "as_name": "DIGITALOCEAN-ASN",
        "country": "SG",
        "first_seen": "2021-05-16 19:49:33",
        "last_online": "2023-11-19",
        "malware": "Dridex"
    },
    {
        "ip_address": "192.99.150.39",
        "port": 7443,
        "status": "online",
        "hostname": "ns509018.ip-192-99-150.net",
        "as_number": 16276,
        "as_name": "OVH",
        "country": "CA",
        "first_seen": "2021-08-27 04:48:12",
        "last_online": "2023-11-19",
        "malware": "Dridex"
    }
]

```

O segundo exemplo, listagem 2, é uma fonte de dados de domínios gerados com recurso a Domain Generation Algorithm (DGA). A *DGA Domains* permite obter um conjunto de informação referente aos domínios, como por exemplo:

1. O nome do domínio;
2. A que *malware* está associado;
3. Um link com mais informação.

Esta fonte de dados disponibiliza informação sobre domínios de DGA conhecidos usados por *malware* no intervalo de tempo de -2 dias e +3 dias da data corrente.

Listagem 2: Extrato de dados recolhidos da fonte de dados DGA Domains

```
############################################################
## Domain feed of known DGA domains from -2 to +3 days
## HIGH CONFIDENCE DOMAINS ONLY
##
## Feed generated at: Mon Nov 18 12:40:01 AM UTC 2024
##
## Feed Provided By: John Bambenek of Bambenek Consulting
## jcb@bambenekconsulting.com // http://bambenekconsulting.com
##
## Use of this feed is governed by the license here:
## http://osint.bambenekconsulting.com/license.txt
## For more information on this feed go to:
## http://osint.bambenekconsulting.com/manual/dga-feed.txt
##
############################################################
wxiclqdekgvcj.com,Domain used by Cryptolocker - Flashback DGA for 16 Nov
↪  2024,2024-11-16
xknnoyjsbdvvi.net,Domain used by Cryptolocker - Flashback DGA for 16 Nov
↪  2024,2024-11-16
xskvphucliehc.biz,Domain used by Cryptolocker - Flashback DGA for 16 Nov
↪  2024,2024-11-16
yfphspbqcfebk.ru,Domain used by Cryptolocker - Flashback DGA for 16 Nov
↪  2024,2024-11-16
ugmvdldinnwex.org,Domain used by Cryptolocker - Flashback DGA for 16 Nov
↪  2024,2024-11-16
vsrhgtjwekwxw.co.uk,Domain used by Cryptolocker - Flashback DGA for 16 Nov
↪  2024,2024-11-16
vbophcugopfjx.info,Domain used by Cryptolocker - Flashback DGA for 16 Nov
↪  2024,2024-11-16
wntbkkbufmfdg.com,Domain used by Cryptolocker - Flashback DGA for 16 Nov
↪  2024,2024-11-16
djvgfcgcmrhee.net,Domain used by Cryptolocker - Flashback DGA for 16 Nov
↪  2024,2024-11-16
```

Como já referido anteriormente, estas fontes de dados são disponibilizadas em formatos que facilitam a integração da informação recolhida com vários sistemas de monitorização, conseguindo desta forma melhorar substancialmente a qualidade dos dados recolhidos internamente quando cruzados com os recolhidos por fontes externas. Para além do referido, a utilização destes dados pode também melhorar substancialmente a segurança de perímetro de uma organização. O vasto âmbito

das fontes externas permite-nos fazer uso delas nas várias fases da resposta a uma ameaça, seja na defesa proativa ou reativa da organização.

A tabela 2 mostra alguns exemplos de fontes de informação que podemos encontrar na Internet. Esta apresenta a descrição do *feed*, os dados que contêm a frequência de atualização do mesmo e qual o licenciamento associado ao *feed* (comercial ou *opensource*). Estes serão os 4 fatores que podem ser importantes na seleção de um *feed* de dados: primeiro é necessário perceber qual a finalidade do *feed*; de seguida é importante conhecer os dados que este divulga e se são relevantes para o caso de uso; em terceiro lugar é importante considerar a frequência de atualização que revela o quão fidedigna é a sua informação e se ainda é relevante à data de recolha (exemplificando, um *feed* de dados que se atualiza uma vez por semana ainda não contém a informação mais relevante); e, por último, o acesso gratuito ou não à informação. Esta escolha foi feita com base nos *feeds* de dados partilhados pela documentação do IntelMQ. (IntelMQ, 2023a)

Nome	**Descrição**	**Dados Principais**	**Tempo de Atualização**	**Licenciamento**
Feodo Tacker Botnet C2 IOCs	Lista de endereços IP de C2 e informação mais detalhada sobre os mesmos	Endereços IP	Diáriamente	Gratuito
Phishtank - Online	Informação sobre ataques de phising	Endereços IP, URLs	1 hora	Pago
DGA Domaisn	Lista de domínios com nomes gerados através de algoritmos	Nomes de domínio	Diáriamente	Sujeito a licenciamento, consoante o tipo de uso
Blocklist.DE *Brute-force Logins*	Lista de endereços IP com tentativas de *brute-force* a serviços de CMS	Endereços IP	30 minutos	Gratuito

Tabela 2: Exemplos de *feeds* de dados

2.4 SÍNTESE

Ao recolher este conjunto de informação e ao fazer a correlação entre os dados recolhidos internamente e externamente, as organizações podem, com a utilização de diversas ferramentas, conseguir identificar precocemente uma ameaça às suas infraestruturas. Perceber em que fonte de dados encontramos os IoC acima referidos

é uma mais-valia pois reduz o tempo necessário para pesquisar as evidências. A tabela 3 mostra qual a fonte mais indicada para obter os IoC, indica onde procurar e que fontes devem ser utilizadas para cada caso especifico.

IoC \ Fontes de Dados	**Logs/Eventos de Segurança**	**Security Enforcing Function**	**Fluxos de Rede**	**Sessões**	**Streams de dados**
***Lookups* de DNS**		X			
Ficheiros, Aplicações e Processos suspeitos		X			X
Endereços de IPs, nomes de domínios e servidores de C2					X
Número de acessos a ficheiros	X			X	
Atividade suspeita em contas de utilizadores	X	X			
Atualizações inesperadas	X				
Transferência de dados em portos pouco utilizados		X	X		
Comportamentos estranhos em *websites*	X	X			
Assinaturas de ataques ou *Hashs* conhecidos		X			X
Tamanhos incomuns nas respostas HTTP	X			X	
Mudanças não autorizadas em ficheiros de configurações, registo ou configurações de equipamentos	X				
Elevado número de tentativas falhas de *login*	X				
Grandes volumes de tráfego		X	X		
Padrões anómalos de rede			X		
Grandes volumes de exfiltração de dados		X	X		

Tabela 3: Correlação entre fonte de dados e IoC

Cada tipo de ataque referenciado na secção 2.1 funciona de forma diferente e deixa vários tipos de evidências. Ao perceber que IoC é criado com cada ataque é possível melhorar a forma como é feita a resposta inicial ao incidente. A tabela que se encontra no apêndice A faz a ligação entre os ataques e os IoC definidos neste trabalho para que possamos ter uma ideia generalizada do que podemos esperar encontrar em cada tipo de ataque.

3

FERRAMENTAS E TÉCNICAS

Existem várias ferramentas que permitem recolher e dar um visão mais alargada dos sistemas informáticos das organizações em tempo real aos técnicos especializados. De forma a completar estas ferramentas, existem técnicas que podem ser utilizadas tendo em conta os dados recolhidos pelas várias ferramentas de monitorização. Esta secção destina-se a dar a conhecer um conjunto de ferramentas que podem ser utilizadas para recolher estes dados e aborda ainda técnicas utilizadas para detetar anomalias.

3.1 FERRAMENTAS

No universo da cibersegurança existem vários tipos de ferramentas que podem ser utilizadas para garantir um bom funcionamento e monitorização dos sistemas. Podemos encontrar ferramentas que se dividem em dois grupos principais: as ferramentas defensivas, mais viradas para as *blue teams*, e as ferramentas ofensivas, que se englobam nas ferramentas das *red teams*. Este projeto foca-se mais na vertente defensiva do espectro da cibersegurança e estuda algumas ferramentas e a sua utilidade para a defesa do ciberespaço de uma organização. No entanto, a utilização de ferramentas ofensivas é parte importante da segurança de uma organização, pois, quando utilizadas para detetar falhas na segurança, permitem que se tenha uma abordagem mais preventiva à segurança de toda a infraestrutura informática, pelo que não devem ser colocadas de parte. De seguida será apresentado um conjunto de tipos de ferramentas que podem ser utilizadas para recolha e análise de informação útil para validação da segurança:

1. Análise de fluxos (exemplo: ntopng);
2. IDS/IPS (exemplo: Snort);
3. Centralizadores de feeds (exemplo: IntelMQ);
4. Sniffers (exemplo: Networkminer);
5. SIEM (exemplo: Wazuh).

Estas são algumas das ferramentas utilizadas por profissionais na área da cibersegurança para melhorar a sua visão da infraestrutura, cobrindo algumas funções de monitorização e recolha de dados dos sistemas. Estas ferramentas focam-se especificamente numa tarefa e devem ser complementadas com outras.

3.1.0.1 *ntopng*

A análise de fluxos de rede é uma mais-valia para qualquer organização. Esta prática possibilita uma visão aprofundada de todas as comunicações que sejam alvo de análise por parte das ferramentas de análise de fluxos de rede. Ao utilizarmos esta categoria de ferramentas podemos detetar anomalias em tempo real. Isto permite-nos, por exemplo, validar a existência de comunicações de e para servidores ou serviços que estejam marcados como possíveis ameaças ou comunicações com demasiado tráfego e tamanho que pode indicar algum tipo de ataque.

O ntopng é uma ferramenta que aplica conceitos de fluxos de rede e que permite fazer uma monitorização ao nível dos pacotes de rede (fluxos) utilizando o protocolo *Netflow*. O protocolo *Netflow* funciona ao nível dos dispositivos de rede, ou seja, para fazer uso deste protocolo é necessário ter um dispositivo que tenha a capacidade de habilitar este protocolo. Pode ser utilizado em *routers*, *switchs* ou *firewalls*. Estes dispositivos armazenam dados do tráfego que por eles passam. A informação é recolhida para uma tabela (*flow cache*) que permite monitorizar os pacotes. Esta ferramenta possibilita uma visão pormenorizada da rede pois tem a capacidade de recolher informação de vários pontos como *mirrors* de tráfego, equipamentos Simple Network Management Protocol, *logs* de *firewalls*, Intrusion Detection System e de *Netflow exporters*. As principais características deste software são (ntopng, 2023):

1. Ordenar tráfego de rede em fluxos utilizando as métricas já descritas anteriormente;
2. Mostrar dados de rede e de equipamentos em tempo real;
3. Produzir relatórios das várias métricas recolhidas das sete camadas do modelo OSI;
4. Dar informação sobre as máquinas que mais tráfego geram;
5. Monitorizar estatísticas de rede;
6. Armazenar os dados permanentemente;
7. Localizar os dispositivos geograficamente;
8. Analisar tráfego IP e organizá-lo de forma adequada;

9. Gerar alertas flexíveis;
10. Focado na cibersegurança;

A utilização de ferramentas como o ntopng permitem recolher informação dos fluxos conseguindo, assim, perceber o tráfego dos equipamentos recolhendo informação como IP origem e destino, porto origem e destino, interface utilizada, número de pacotes de um fluxo e quantidade de dados transmitidos no fluxo. As ferramentas de análise de fluxos dão uma visão pormenorizada do tráfego da rede, facilitando a analise e deteção de anomalias na rede de uma organização.

3.1.0.2 *Snort*

Um Intrusion Detection System (IDS) é uma ferramenta vital na cibersegurança de uma organização, estando focada em identificar atividades anómalas ou potencialmente maliciosas numa rede. Ao analisar padrões de tráfego e comportamento, o IDS alerta os administradores sobre possíveis ameaças, permitindo assim respostas rápidas e eficazes, cruciais no combate a possíveis incidentes de segurança.

Por outro lado temos os Intrusion Protection System (IPS), uma ferramenta mais completa, que atuam automaticamente sobre as ameaças detetadas para bloquear ou prevenir possíveis crises. Estas camadas de defesa ajudam a proteger os sistemas de informação ao fornecer uma vigilância constante, contribuindo para a deteção precoce e a mitigação de potenciais ataques.

O Snort é um Intrusion Protection System *opensource*. Utiliza um conjunto de regras que ajudam a detetar atividades maliciosas na rede. O Snort pode ser utilizado de três principais modos: como um *sniffer* de pacotes, como agregador de pacotes para análise posterior e como um IPS. O Snort pode ser implementado "*in-line*", de forma a poder facilmente detetar os pacotes e impedir o envio dos mesmos. Esta aplicação funciona com base na utilização de dois métodos de deteção diferentes: por assinaturas e por regras. O primeiro método, através de assinaturas, permite detetar ataques com base em características conhecidas. Por exemplo, um certo *exploit* tem características específicas que ao serem relacionadas com as assinaturas de ataques conhecidos, podem despoletar um alerta, ou seja, as assinaturas são desenvolvidas com objetivos específicos para detetar determinado ataque. A utilização de assinaturas para detetar ataques conhecidos é bastante eficaz, contudo, sendo que é necessário previamente existir informação sobre determinado ataque, este método é de proteção limitada e não é compatível com ataques de *zero-day*. Para dar resposta aos ataques de *zero-day*, o Snort permite a criação de

regras, que, por outro lado, já não se focam em características especificas de um dado *exploit* mas sim na deteção de vulnerabilidades. Contudo, a criação de regras requer um elevado conhecimento de como funciona determinada vulnerabilidade (Snort, 2023).

As regras do Snort são constituídas com pelo menos os seguintes dados (Roesch, 2023):

1. Ação - A ação da regra é o que o próprio nome indica, a forma como a regra deve agir. Pode ser dos seguintes tipos:
    a) *Alert* - Gera um alerta e faz *log* do pacote;
    b) *Log* - Faz *log* do pacote;
    c) *Pass* - Ignora o pacote;
    d) *Activate* - Alerta e ativa uma regra dinâmica;
    e) *Dynamic* - Regra que apenas é utilizada quando ativada por uma outra regra e depois faz o *log* do pacote;
2. Protocolo - O protocolo sobre o qual a regra vai atuar;
3. Endereços IP - São definidas as redes/*hosts* que fazem parte da comunicação;
4. Portos - Definem-se os portos utilizados para as comunicações;
5. Direção - Define a direção da comunicação, que pode ser unidirecional ou bidirecional.

As organizações podem melhorar a sua defesa interna ao utilizar ferramentas de IPS, como o Snort, para terem uma visão mais aprofundada sobre o que se passa na sua rede em tempo real, conseguindo assim algum tipo de ação para as ameaças detetadas.

3.1.0.3 *IntelMQ*

Os centralizadores de *feeds* de segurança são uma ferramenta imprescindível para uma organização. Estas ferramentas são responsáveis por agregar e tratar os dados recolhidos de várias fontes de informação externas, sejam elas *opensource* ou pagas. A recolha desta informação permite que as organizações tenham acesso a dados sobre o panorama do ciberespaço conseguindo assim perceber as tendências e as ameaças constantes.

Este trabalho, como mencionado anteriormente, foca-se na utilização de fontes de dados de segurança para fortalecer a cibersegurança das organizações. O IntelMQ,

é uma ferramenta direcionada para as equipas de segurança e permite recolher e processar estas fontes de dados. O principal objetivo desta aplicação é munir as equipas de resposta a incidentes com uma forma fácil de agregar e processar *threat Intelligence* (IntelMQ, 2023b). Esta aplicação tem os principais casos de uso:

1. Tratamento de incidentes automatizado;
2. Consciência situacional;
3. Notificações automáticas;
4. *Data Collector* para outras aplicações.

A ferramenta funciona com base em *Bots*, *Botnets*, *Pipelines*, *Queues* e Mensagens, Eventos e Relatórios. Os *Bots* são um pequeno programa com um único propósito: comunicar com a *pipeline* para receber e enviar mensagens, a informação das fontes de dados. Os *Bots* podem ser de quatro tipos diferentes: *Collectors*, *Parsers*, *Expert* e *Output*. As *botnets*, são um conjunto de *bots*. A *pipeline* é o agente de mensagens que transporta as mensagens entre *bots* sendo esta comunicação definida pelo administrador da aplicação. As *Queues* são as filas de mensagens a receber e a enviar pelos *bots*. Por fim, as mensagens são a informação trocada entre os *bots* e os relatórios são as mensagens não tratadas passadas entre dois tipos de bots diferentes: os *collectors* e os *parsers*. Após tratadas as mensagens são denominadas de eventos e contém um IoC (IntelMQ, 2023c). Resumidamente, o IntelMQ é uma ferramenta que permite criar *bots* (pontos de entrada) para receber e tratar informação que vem de variadas fontes de dados e torna a mesma útil para as equipas de resposta a incidentes.

3.1.0.4 *Networkminer*

Os *sniffers* são ferramentas que permitem analisar o tráfego, à semelhança das ferramentas de análise de fluxos. No entanto, a grande diferença é que estas são responsáveis por fazer a análise pacote a pacote. Os *sniffers*, ao fazerem a análise de uma dada rede, permitem ver em mais detalhe o que as comunicações transportam, podendo dar informações valiosas aos administradores de sistemas.

Sniffers de rede são ferramentas vastamente utilizadas pela indústria da cibersegurança. O Networkminer é uma ferramenta *opensource* que se destina à análise forense da rede, isto é, permite analisar o tráfego capturado, em ficheiros pcap, ao detalhe. As principais funcionalidades desta aplicação são:

1. Análise de PCAP e PCAP-NG;

2. Análise de tráfego em tempo real;
3. Análise de ficheiros .etl (Windows);
4. Extrair ficheiros de vários protocolos diferentes, como por exemplo, FTP, SMB, HTTP, POP3, IMAP;
5. Extrair certificados X.509 de comunicações encriptadas;
6. Desencapsulamento de pacotes de rede, como por exemplo, 802.1Q, PPPoE e OpenFlow.

O Networkminer permite, com recurso à análise de ficheiros pcap ou de tráfego em tempo real, obter ficheiros, imagens, credenciais, pedidos de DNS e máquinas presentes nas comunicações. Para além destas mais-valias, a aplicação permite que sejam executadas pesquisas ao tráfego analisado, conseguindo assim reduzir drasticamente o tempo de procura de artefactos. (Netresec, 2023) Este tipo de ferramentas permite realizar uma análise mais detalhada das comunicações de rede e podem ser utilizadas em conjunto com outras, como por exemplo a já mencionada anteriormente *ntopng*, de modo complementar à mesma.

3.1.0.5 *Wazuh*

Os Security Information and Event Management (SIEM) são ferramentas de agregação, tratamento e consulta de dados de uma infraestrutura informática. Estas ferramentas são a evolução de duas ferramentas bastante utilizadas, Security Information Management (SIM) e Security Event Management (SEM). A primeira destina-se a recolher e a analisar *logs* dos vários equipamentos e sistemas presentes na rede; já a segunda refere-se a uma ferramenta capaz de monitorizar a atividade dos vários sistemas presentes na rede através dos eventos que toda a atividade despoleta. A secção 3.3 destina-se a aprofundar mais sobre a temática dos SIEM.

O *Wazuh* é uma ferramenta que, entre outras funcionalidades, funciona como SIEM. Esta ferramenta é *opensource* e destina-se a agregar e analisar dados de toda a infraestrutura de rede em tempo real, possibilitando assim a deteção de ameaças e a conformidade de todos os sistemas. Permite recolher informação de várias fontes como *endpoints*, dispositivos de rede ou aplicações. Como funcionalidades deste SIEM destacam-se as seguintes (Wazuh, 2023):

1. Análise de *logs*;
2. Deteção de vulnerabilidades;
3. Avaliação da configuração de segurança;

4. Conformidade com a legislação e regulamentos em vigor.

Estas funcionalidades fazem do *Wazuh* um SIEM bastante completo, pois garantem às equipas de segurança um conjunto de possibilidades na segurança da sua infraestrutura como analisar eventos de segurança e identificar anomalias ou IoC, rever configurações de segurança no vários *endpoints* com base nas referências do Center for Internet Security, deteção de vulnerabilidades dos sistemas e das aplicações instaladas nos *endpoints* e, por último, simplifica a forma como as entidades garantem o cumprimentos dos regulamentos em vigor.

3.2 TÉCNICAS

A recolha de informação, só por si, nem sempre é suficiente para garantir uma correta implementação de medidas de segurança numa dada organização. Podem ser aplicadas um conjunto de técnicas à informação recolhida pelas várias ferramentas de monitorização utilizadas. Ao utilizar técnicas de análise de dados na informação recolhida, é possível detetar possíveis IoC que são desencadeados em caso de ataque. Desta forma, aliada à recolha de dados, a utilização destas técnicas é muito relevante no sucesso de uma organização na sua cibersegurança. De seguida, são dados exemplos de algumas técnicas que podem ser implementadas para análise dos dados recolhidos:

1. Análise de semântica e da estrutura do URL;
2. Análise de endereços IP e URL;
3. Reputação do prestador de serviços e de empresas de *hosting*;
4. Assinaturas de antivírus e *hashes*;
5. Lista de portos utilizados nas comunicações;
6. Número de comunicações e tamanho das mesmas;
7. Criação de padrões por utilizadores/IP/equipamento;
8. Análise de *logs*;
9. Monitorização, captura e análise de tráfego de rede.

3.2.1 *Análise de semântica*

Existem formas de colocar estas técnicas em prática, como por exemplo através da utilização de fontes de dados externas com as quais é possível correlacionar os dados recolhidos internamente, como já referido anteriormente. Podem também ser utilizadas técnicas de Inteligência Artificial (IA) ou algoritmos específicos que nos permitam, sem recurso a dados externos, detetar anomalias. O Natural Language Processing (NLP) e o algoritmo da distância de *Levenshtein* são métodos que nos podem ser úteis na análise semântica de nomes de domínio, como por exemplo comparando um domínio real com um detetado em comunicações e aferir se se trata de algum tipo de ataque (comparar os domínios (fictícios) mcif.ipleiria.pt e micf.ipleiria.pt).

3.2.1.1 *Natural Language Processing*

Natural Language Processing é o campo específico da IA que estuda a similaridade semântica entre palavras ou frases, apresentando uma medida quantitativa entre ambas. De uma forma geral, este processo converte palavras ou frases num vetor, que é um representação matemática dessa mesma palavra ou frase. Através da utilização de funções de similaridade, é possível quantificar a igualdade entre os objetos de comparação. As funções de similaridade são utilizadas para medir a distância, ou seja, a semelhança, entre dois vetores. Existem diferentes modelos de comparação que ajudam a determinar a semelhança entre objetos, tais como o comprimento das palavras ou entre diferentes léxicos. De forma a dar um exemplo, num cenário de comparação entre duas frases em que a métrica para comparação, isto é, a função de similaridade, é *cosine* e a saída é um valor entre 0 e 1, comparamos as seguintes frases (Overcash, 2021):

1. Frase 1 - *rivers woods and hills*
2. Frase 2 - *streams forests and mountains*
3. Frase 3 - *deserts sand and shrubs*

Os resultados obtidos forma os seguintes:

1. Comparação entre Frase 1 e Frase 2: 0.84
2. Comparação entre Frase 1 e Frase 3: 0.631
3. Comparação entre Frase 2 e Frase 3: 0.576

Existem outras funções de similaridade que podem ser aplicadas, devendo sempre, de ante-mão, perceber que tipo de dados estamos a comparar de forma a obter uma resposta mais rigorosa.

3.2.1.2 *Distância de Levenshtein*

A distância de *Levenshtein* é um algoritmo que nos diz o quão diferente são duas sequências (*strings*). A distância é medida através do número de vezes que é necessário alterar uma palavra para que esta se transforme noutra, ou seja, cada vez que é alterado ou acrescentado um carácter a distância aumenta. Quer isto dizer que quanto maior a distância de *Levenshtein* maior é a diferença entre as palavras comparadas. (Nam, 2019)

Dando de um exemplo prático, medir a distância de *Levenshtein* entre as seguintes duas palavras:

1. Segurança;
2. Cibersegurança.

A distância de *Levenshtein* entre estas duas palavras é de **6**. Passo a explicar: cada inserção de carácter aumenta 1 valor, ou seja, a inserção das letras 'C', 'i', 'b', 'e', 'r' coloca a distância em 5. Contudo, a alteração do 'S' para um 's', também conta para a distância de *Levenshtein* perfazendo um total de **6**.

3.2.2 *Análise de IPs e URLs*

A análise à reputação de endereços IP é um método que nos permite aferir se dado endereço IP e URL é confiável. Existe um conjunto de fontes de dados externas que podem ser utilizadas para analisar endereços IP e/ou URL e correlacionar com aqueles detetados internamente. Um exemplo de lista de endereços IP é a "*Reputation List*"da AlienVault, que enumera um conjunto de endereços IP maliciosos, bem como a sua localização geográfica. Outro exemplo é a fonte de dados "*BingMURLs via Interflow*"que enumera um conjunto de URL maliciosos detetados pelo Bing da Microsoft. Ao consumir estes dados de fontes externas podem ser tomadas medidas relativas ou preventivas relativamente aos endereços IP e URL listados. Estas listas de endereços IP podem ser construídas de quatro formas diferentes (Holland, 2023):

1. Automaticamente - Algoritmos nos dispositivos/*software* de segurança dos vários vendedores, que ao detetarem atividade suspeita, adicionam o endereço a uma *blocklist*;
2. Manualmente - Exatamente o mesmo processo no entanto, ao contrário de utilizar um algoritmo que adiciona automaticamente os endereços, são os analistas que o fazem manualmente;
3. Baseadas em URL - Os URLs e as páginas *web* associadas a endereços de IP considerados maliciosos são adicionados a uma lista;
4. Com base na reputação - Os domínios e endereços IP que historicamente estão associados a atividades suspeitas têm uma reputação mais fragilizada e são adicionados a este tipo de listas. Não significa automaticamente que sejam maliciosos, mas devem ser monitorizados.

3.2.3 *Reputação das empresas*

A reputação das empresas cai um pouco sobre a mesma da reputação dos endereços IP e URLs. Quer isto dizer que, quanto mais associadas a atividades maliciosas, pior será a sua reputação perante toda a comunidade de cibersegurança. Um exemplo prático seria um dado prestador de um serviço DNS não tomar as devidas precauções quando um dado endereço de DNS é criado, isto é, não validar os dados de quem regista o nome de domínio e não validar a possibilidade de ser um domínio criado com utilização de Domain Generation Algorithm (DGA). Tudo isto são pequenas falhas que ajudam a que atores maliciosos ganhem facilidade na implementação dos seus objetivos.

3.2.4 *Assinaturas de antivírus e hashes*

Uma assinatura de um antivírus não é nada mais do que uma forma de detetar um vírus, ou seja, é uma sequência contínua de *bytes* que são comuns para um certo tipo de *malware*. Esta sequência de *bytes* pode ser encontrada por exemplo dentro de um ficheiro infetado (Malanov, 2023). Quando um novo *malware* é detetado, as empresas de antivírus lançam assinaturas para que estas possam detetar a nova versão do *malware*. Estas assinaturas são na sua maioria utilizadas por antivírus de forma a poderem detetar a existência de vírus nos sistemas. Contudo, as assinaturas têm algumas deficiências no que diz respeito à sua utilização. A primeira é o constante

crescimento da lista de assinaturas, o que torna a sua utilização mais exigente a nível computacional. A segunda é que estas apenas são úteis para *malware* que já é conhecido, não sendo a melhor alternativa para a deteção de *zero-days*. Por outro lado, as *hashes* são uma forma rápida de validarmos se algo é malicioso. Dando um exemplo de um dado ficheiro que é conhecido por ter *malware*, é gerada uma *hash* (neste caso a *hash* funciona como um identificador único para um ficheiro com aquele conteúdo) para este ficheiro e disponibilizada, para que de forma rápida possa ser detetado. A utilização de funções de *hash* possibilita a partilha de informação entre investigadores de uma forma mais eficaz. (Patrol, 2024)

3.2.5 *Lista de portos utilizados*

A utilização de portos fora dos mais conhecidos pode ser uma indicação de algum tipo de incidente, pois estes podem ser utilizados por *malware*. Por exemplo, se uma determinada máquina está a comunicar com um serviço no porto 31338, um porto incomum para correr serviços conhecidos, isto é um indício para analisar. Este porto em específico é conhecido por ser utilizado pelo *malware Net Spay*. A figura 2 mostra uma pequena listagem de um conjunto de portos que são tipicamente utilizados por determinado *malware*. Este tipo de incidentes deve levar a uma análise mais aprofundada pois pode indicar algo mais grave. Desta forma, é importante analisar regularmente as comunicações e ter em consideração os portos mais utilizados.

Port Number	Trojan Name	Port Number	Trojan Name
23432	Asylum	31338	Net Spy
31337	Back Orifice	31339	Net Spy
18006	Back Orifice 2000	139	Nuker
12349	Bionet	44444	Prosiak
6667	Bionet	8012	Ptakks

Figura 2: Portos utilizados por *malware* (Fonte: Trendmicro)

3.2.6 *Número de comunicações e tamanho das mensagens*

No seguimento do ponto anterior, é importante analisar o número de vezes que certas comunicações são executadas bem como o seu tamanho. Demasiadas comunicações e de tamanhos anormais a servidores internos podem indicar vários tipos de problemas, como tentativas de intrusão ou até de negação de serviço.

Em sentido oposto, é importante ver se existem várias comunicações a sair para determinados endereços, indicando que alguma máquina possa estar a fazer parte de uma tentativa de Distributed Denial of Service (DDoS). É importante também ter em atenção o tamanho dos fluxos de dados nas comunicações de saída, isto porque se for detetado um fluxo de dados com tamanhos avultados, por exemplo 30GB, isto pode muito bem indicar uma exfiltração de dados e que um incidente de segurança está a ocorrer. (Kost, 2024)

3.2.7 *Criação de padrões de utilização de rede*

O conhecimento de como a infraestrutura da organização funciona é o primeiro passo para conseguir detetar anomalias. Aliado a este conhecimento, é vantajoso ter um conjunto de padrões de rede, sejam eles dos utilizadores ou das máquinas, para ter forma de comparar um normal funcionamento a uma forma de funcionar que possa estar a ser alterada por motivo de incidente de cibersegurança. Por exemplo, se um servidor cujo funcionamento habitual é apenas receber pedidos Hypertext Transfer Protocol (HTTP) e de repente começa a realizar comunicações anormais para o exterior, isto pode indicar que algo não está a funcionar como esperado.

Outra forma de possível deteção é uma determinada máquina de dada organização, ter periodicamente comunicações com um endereço IP/domínio estranho, o que pode indicar comunicações com um servidor de C2. Esta comunicação é chamada de *"Beaconing"* e é um método utilizado para manter a ligação ao servidor de Command and Controll e/ou receber. Normalmente estas comunicações são feitas em portos comuns como o HTTP:80 e o HTTPS:443, de modo a evitar deteções pelas *firewalls*. Este padrão de comunicações pode ser detetado e analisado. (Kost, 2024)

3.2.8 *Análise de* logs

Não podia faltar numa lista de técnicas a utilizar para melhorar a cibersegurança de uma organização a análise de *logs*. Sem esta análise é extremamente difícil obter mais informações sobre os incidentes que nos possam ser úteis na resolução e/ou mitigação de uma anomalia. É importante analisar regularmente os *logs*, não só quando existem problemas mas também para que estes possam ser detetados antes que causem problemas mais graves.

Existe um conjunto de diferentes *logs* que diferenciam consoante o tipo de sistema operativo utilizado. No caso do Windows, os *logs* ou eventos estão divididos em seis categorias diferentes, sendo elas: (Logic, 2024)

1. *Logs* aplicacionais;
2. *Logs* relacionados com serviços de domínio (*Active Directory*);
3. *Logs* de servidor DNS;
4. *Logs* de replicação de ficheiros;
5. *Logs* de segurança;
6. *Logs* de sistema.

Por outro lado, no que diz respeito aos logs de *Linux*, estes podem ser caracterizados da seguinte forma:

1. *Logs* aplicacionais;
2. *Logs* de eventos;
3. *Logs* de serviços;
4. *Logs* de sistema.

Caracterizando desta forma, podemos assim reduzir o âmbito da procura, focando a análise apenas nos que poderão fazer mais sentido.

3.2.9 *Monitorização, captura e análise de tráfego de rede*

Por último, mas não menos importante, aliado à utilização de ferramentas que permitam a recolha de pacotes de rede e de fluxos de rede, é importante realizar uma monitorização constante do tráfego de rede de uma dada organização. Esta análise diz-nos para onde e o que está a passar na rede em dado momento. Esta

análise, em conjunto com os dados obtidos por fontes externas, podem dar-nos indicadores de que algum incidente possa ter acontecido ou estar a acontecer.

Dando alguns exemplos, se uma máquina na rede da organização está a fazer uma resolução de DNS de um domínio que está numa *blocklist*, é um indicador de que devem ser tomadas medidas sobre esta comunicação, podendo levar até a uma análise mais aprofundada. Outro caso de análise é a monitorização das comunicações para o exterior, garantindo que estas não são feitas para endereços IP ditos maliciosos, ou seja, se estas comunicações são um evento sem precedentes ou se é uma comunicação regular.

3.3 FUNCIONAMENTO GERAL DAS SOLUÇÕES DE SEGURANÇA

A informação que é gerada internamente pelos vários sistemas de segurança de perímetro, equipamentos de rede, *endpoints* e serviços necessita de ser tratada e armazenada. A utilização de repositórios centrais é bastante comum e permite que seja feita uma gestão centralizada de toda a informação. Os Security Information and Event Management (SIEM) são uma ferramenta que permite recolher, armazenar e analisar toda a informação recolhida, normalmente através de protocolos como o *syslog*, permitindo assim que todos os equipamentos possam enviar dados para estes sistemas. Estas ferramentas são muito utilizadas pelas organizações pois, para além de fazerem a gestão centralizada de todos os eventos de segurança, podem prevenir, detetar ou atuar em caso de ciberataque. De uma forma geral estes sistemas colecionam, agregam, armazenam e correlacionam eventos de segurança de uma organização, sendo uma mais-valia para qualquer equipa de segurança. Os SIEM, são geralmente constituídos pelos seguintes blocos (Gustavo González-Granadillo, 2021):

1. Equipamento origem;
2. Logs;
3. *Parsing*;
4. Motor de regras;
5. Armazenamento de logs;
6. Monitorização.

Estas plataformas providenciam às equipas de segurança uma ferramenta para monitorizar os eventos de segurança em tempo real na rede de uma organização,

podendo assim atuar diretamente nos vários alertas que possam surgir, baseados nas regras criadas no SIEM. A imagem 3 (Sandeep Bhatt, 2014) demonstra uma arquitetura tipo de um SIEM. Na imagem podemos ver e relacionar os vários pontos com a lista acima definida. Os primeiros intervenientes são os equipamentos ou serviços (IDS, IPS, Antivírus, equipamentos de rede, *firewalls*, aplicações, etc.) que geram os vários *logs*. Cada um destes é configurado para enviar os seus eventos para um ponto centralizado. A primeira tarefa do SIEM é normalizar os diferentes dados já que estes provêm de vários equipamentos e sistemas com fabricantes diferentes, ou seja, fazer o tratamento (*parsing*) da informação recebida para um formato comum. Esta normalização permite a simplificação do processamento dos dados e da criação de regras. Após normalizados, estes dados podem ser enviados para armazenamento em vários sistemas diferentes, sejam estes para monitorizar ou para investigar.

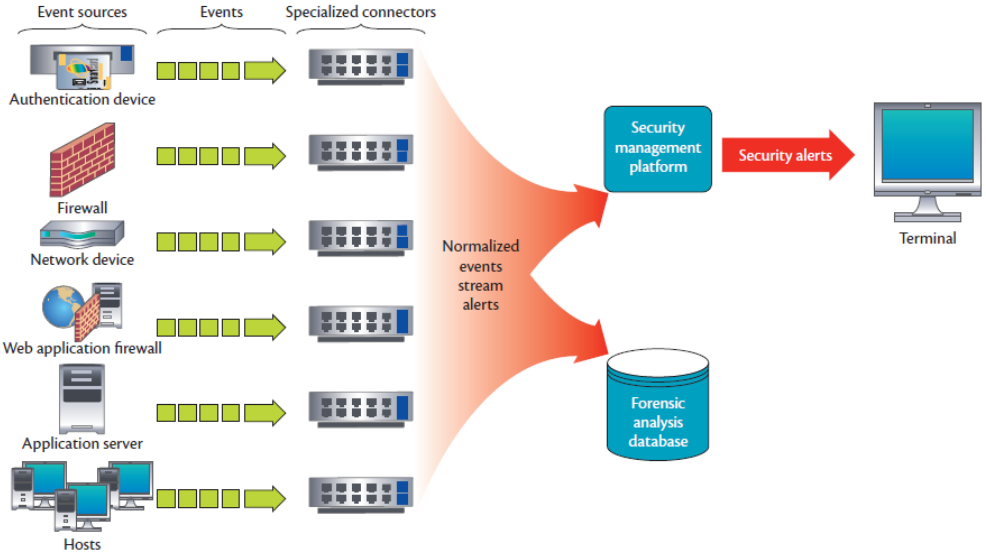


Figura 3: Arquitetura de um SIEM

Como todas as ferramentas e tecnologias, os SIEM apresentam alguns desafios que tornam a sua implementação nas organizações mais difícil. Estes desafios demovem grande parte das empresas de implementar este tipo de soluções, que são uma mais-valia para a cibersegurança. De seguida, enumero alguns dos desafios que podem ser encontrados na implementação e/ou utilização de um SIEM (ATT, 2014; Cybriant, 2023):

1. Poder computacional - Este ponto depende sempre do tamanho da infraestrutura informática que esteja a ser monitorizada na organização. Quanto maior, mais dados são recolhidos e consequentemente mais poder computacional é necessário, o que nos leva ao ponto seguinte.

2. São soluções caras - Existem enumeras soluções no mercado, algumas bastante caras outras que são *opensource*, contudo, não se pode medir apenas pelo custo de licenciamento, mas também o custo do *hardware* e o custo da mão-de-obra especializada que é necessária para manter funcional a ferramenta;
3. Demasiado complexas - Dada a natureza de um SIEM, que se foca na recolha de dados dos vários sistemas de uma organização, é necessário gastar tempo a integrar todos os diferentes sistemas, bem como a garantir que o constante crescimento das infraestruturas possa ser acompanhado pela ferramenta. A criação e utilização de regras de deteção de anomalias é mais um ponto de complexidade. Tudo isto eleva a dificuldade de implementação e de manutenção deste tipo de plataformas;
4. *Know-How* necessário - Como já referido, é necessário ter mão-de-obra qualificada para operar esta ferramenta, dada a complexidade da mesma. Para além do desenho da arquitetura de implementação de um SIEM, a constante melhoria dos dados recolhidos, a expansão do sistema, a utilização de regras, bem como a análise dos dados recolhidos são tudo tarefas que devem ser realizadas por uma equipa multifacetada e com conhecimentos específicos;
5. Complexidade da gestão de alertas - Tendo em conta o volume de dados recolhidos por estas plataformas e a implementação por omissão dos sistemas, os alarmes despoletados acabam por ser demasiados, tirando o foco do que realmente interessa.

3.3.1 *Síntese*

Em suma, existe um conjunto de ferramentas e de técnicas que podem ser aplicadas que podem melhorar consideravelmente a cibersegurança de uma organização. Contudo, implementá-las e aplicá-las são tarefas que podem ser demasiado complexas para grande maioria dos técnicos. Constrangimentos como o custo de implementação, a contratação de mão-de-obra especializada ou o treino contínuo que é necessário nestas equipas são grandes limitações à implementação destas ferramentas.

Detetados estes problemas, surge a necessidade de criar uma ferramenta que ofereça algum tipo de visão sobre o que se passa na rede da organização, sem que seja necessário muito conhecimento para tal. O objetivo é possibilitar que as organizações com menos recursos tenham uma ferramenta que lhes dê *feedback* útil ao invés de não terem nada que as possa ajudar a detetar algum tipo de incidente.

4

PROPOSTA DE ARQUITETURA

Tendo em conta os pressupostos do último capítulo, bem como a análise feita no subcapítulo 3.3.1, propôs-se o desenvolvimento uma solução que vem colmatar algumas das fragilidades (como a difícil implementação e análise dos dados), fornecendo uma forma simples de introduzir e visualizar dados e alertas.

Este capítulo destina-se a dar a conhecer a arquitetura lógica da aplicação. Serão explicadas cada uma das suas funcionalidades entrando no pormenor de como serão implementadas e as tecnologias que serão utilizadas, explicando o porquê da sua escolha. Relativamente à implementação física esta será abordada no capitulo seguinte quando for abordada a implementarão da solução.

4.1 REQUISITOS DA SOLUÇÃO

Tal já referido no subcapítulo 1.1 , a solução a ser desenvolvida terá de respeitar os seguintes requisitos:

- Ser uma solução *opensource* - Ter o código fonte disponível permite que a que a solução possa ter o contributo da comunidade;
- Ser de fácil utilização e análise - Não existir a necessidade ter conhecimentos de como criar regras de deteção ou de mecanismos complexos de análise de dados facilitando a utilização da ferramenta através de uma interface simples e intuitiva;
- Requerer poucos conhecimentos de redes e serviços - Não exigir ao utilizador um profundo conhecimento de serviços e redes de informação para implementar a solução;
- Recolher informação externa à organização - Utilizar fontes de dados externas como a sua grande fonte de dados relevantes à segurança;
- Analisar os dados internos recolhidos com base nos dados recolhidos externamente - Realizar a correlação de todos os dados recolhidos internamente com os recolhidos através de fontes de dados externas;

- Gerar alertas - Criar alertas com base na correlação de dados de modo a que o utilizador possa ter uma forma simples de analisar os dados.

4.2 ARQUITETURA LÓGICA

A arquitetura lógica da aplicação pode ser dividida em 5 grupos diferentes de funcionamento, tal como podemos verificar na figura 4, sendo eles os seguintes:

1. Recolha e tratamento de dados internos;
2. Recolha e tratamento de dados externos;
3. Dados introduzidos manualmente;
4. Processamento dos dados;
5. Alertas.

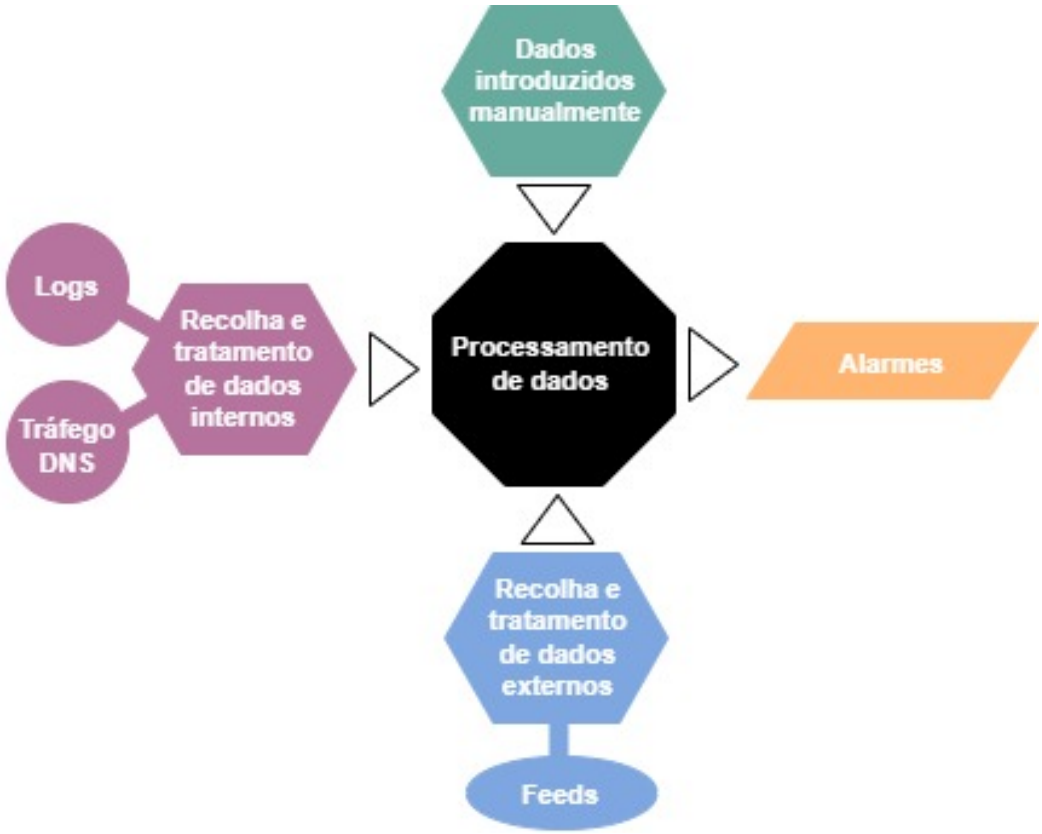


Figura 4: Diagrama da arquitetura

4.2.1 *Recolha e tratamento de dados internos*

Este grupo é responsável pela recolha e tratamento de dados gerados internamente como, por exemplo, os vários tipos de *logs*/eventos de segurança, tráfego de rede, dados de sessões ou dados dos equipamentos de rede da organização como já referido na secção 2.3.1. Neste projeto o foco incidirá sobre os *logs*/eventos de segurança bem como o tráfego de rede associado ao protocolo DNS.

No que diz respeito à recolha dos dados, estes serão recolhidos manualmente, ou seja, o utilizador é responsável por extrair estes dados e enviá-los para que possam ser tratados e para que sejam recolhidas as informações importantes de cada um dos tipos de dados. Quanto ao tratamento dos dados, existirá um mecanismo responsável por identificar os dados relevantes que são enviados para a aplicação para que estes possam ser utilizados. Por fim, os dados que são relevantes recolher neste processo passam por endereços IP no que diz respeito aos *logs*/eventos de segurança, nomes de domínios e endereços de IP traduzidos. Com esta informação recolhida já é possível obter um conjunto de informação que nos permitirá detetar anomalias na infraestrutura.

4.2.2 *Recolha e tratamento de dados externos*

A par do processamento de dados internos, a recolha e tratamento de dados externos é igualmente importante. Esta recolha será feita através da utilização de fontes de dados discutidas anteriormente, na secção 2.3.2. As fontes de dados escolhidas serão aquelas que facultem dados que possam ser cruzados com os dados internos, tal como listas de URLs, endereços IP e nomes de domínio.

A recolha de dados destas fontes externas será feita através de um mecanismo que recolherá periodicamente os dados e armazená-los-á numa base de dados para serem utilizados. Esta base de dados estará em constante atualização, removendo os dados desatualizados e inserindo os novos, sendo possível minimizar a quantidade de falsos positivos que possam aparecer.

4.2.3 *Dados introduzidos manualmente*

É importante que exista uma forma dos utilizadores consigam inserir dados para que a aplicação os possa ter em consideração quando enviar alertas. Tendo por base

o falado na secção 3.2.1, os dados inseridos pelo utilizador serão alvo de análise tendo em conta os dados recolhidos externamente.

O utilizador poderá inserir um conjunto de nomes de domínio e endereços IP para serem analisados. Esta análise terá dois objetivos principais:

1. Validar se os endereços IP inseridos não são encontrados nas listas de endereços maliciosos que serão utilizadas;
2. Validar se os nomes de domínios processados das fontes externas são se assemelham aos domínios introduzidos manualmente;
3. Conjunto de palavras-chave definidas pelo utilizador.

Esta análise irá ajudar o utilizador, primeiro a perceber se os seus endereços IP públicos não são considerados maliciosos e segundo, perceber se existem domínios semelhantes ao ponto de poderem ser utilizados para algum tipo de ataque, mais concretamente ataques de engenharia social.

4.2.4 *Processamento de dados*

O processamento dos dados da aplicação é o cérebro de todo o sistema. É neste ponto que será feito o cruzamento dos dados recolhidos internamente e externamente bem como o processamento dos dados inseridos pelo utilizador. O processamento será responsável por avaliar a severidade das anomalias detetadas, classificando-as de de informativo a critico. Esta forma de classificar irá permitir-nos aferir a importância do alerta. A especificação da classificação a ser executada será aprofundada no subcapítulo 4.3.10.

4.2.5 *Alertas*

Como qualquer aplicação de monitorização, é de extrema importância exista um mecanismo de notificações que permita ao utilizador receber alertas. Todos os alertas gerados na aplicação estarão disponíveis no *dashboard* para que seja possível analisá-los. Os alertas estão divididos em três categorias diferentes:

1. Alertas de tráfego - Os alertas de tráfego são despoletados sempre que são endereços IP e domínios detetados em comunicações para o exterior, ou seja, aqueles que são detetados na análise do tráfego DNS;

2. Alertas de acesso - Os alertas de acesso são despoletados sempre que seja detetado uma comunicação de IPs maliciosos aos serviços disponibilizados pela organização;
3. Alertas de *malware* - Por fim, os alertas de *malware* são despoletados sempre que exista uma comunicação feita para um domínio classificado como malicioso pelas fontes externas de dados.

Esta classificação será feita também com base na origem dos dados externos e o tipo de fonte á qual o dado pertence.

4.3 FERRAMENTAS E TECNOLOGIAS

As ferramentas e tecnologias são um dos pontos cruciais que farão a diferença na forma como este projeto será implementado. Para cada um dos pontos falados anteriormente, no que diz respeito à arquitetura lógica, será feita uma breve introdução às tecnologias utilizadas e o porquê da escolha feita.

A primeira fase, a recolha e tratamento de dados internos, será realizada em dois domínios, como já mencionado anteriormente, através da recolha de logs e da recolha de tráfego referente ao protocolo DNS. Em ambas as recolhas o objetivo é recolher:

1. Endereços IP;
2. Nomes de domínio.

4.3.1 *Recolha de* logs

A recolha e análise de *logs* é um dos pontos cruciais para o sucesso de qualquer sistema de segurança. Tendo este projeto o objetivo de criar uma solução de segurança simplista é imperial que seja implementado algum tipo de análise de *logs*. O objetivo do tratamento de *logs* será recolher informação relevante que possa ser cruzada com as fontes de dados externas que serão consumidas. Os dados a serem recolhidos nos logs são os seguintes:

1. Endereços IP - Relevante para perceber se o endereço de IP é maligno com base nas fontes externas;
2. Nomes de domínio - Relevante para detetar possíveis comunicações para domínios maliciosos;

3. Datas e horas - Para que seja possível, no caso da existência de alertas, identificar a data e hora do incidente.

O foco deste projeto será começar por analisar os ficheiros de *log* e neles recolher a informação enumerada acima. Estes ficheiros podem ser carregados na aplicação de duas formas distintas, manualmente pelo utilizador ou com recurso ao serviço de *rsyslog*. Para testar o processamento dos ficheiros de *log*, foi desenvolvido um *script* (37) que através de *regex* recolhe os endereços de IP e a data e hora associados ao evento despoletado pelo endereço. Neste exemplo, o script valida os *logs* da própria máquina, contudo o objetivo seria que o utilizador da aplicação pudesse colocar os seus ficheiros de *log* para poderem ser tratados, permitindo assim avaliar mais do que uma máquina.

Listagem 3: Código fonte Python para *parsing* de ficheiros para extração de endereços IP e data/hora

```python
import re

def extract_ip_and_datetime(log_file_path):
    """
    Parses the log file and gets all IP address and corresponding datetime
    :param log_path:
    :return: Array with IP address and datetime
    """
    ip_datetime_pairs = []

    ip_pattern = r"\b(?:[0-9]{1,3}\.){3}[0-9]{1,3}\b"
    datetime_pattern = r"^\w{3}\s+\d{1,2} \d{2}:\d{2}:\d{2}"

    with open(log_file_path, 'r') as file:
        for line in file:

            ip_matches = re.findall(ip_pattern, line)
            datetime_match = re.match(datetime_pattern, line)
            if ip_matches and datetime_match:

                datetime_str = datetime_match.group(0)

                for ip in ip_matches:
                    ip_datetime_pairs.append((ip, datetime_match.group(0)))

    return ip_datetime_pairs

def dispalyData(ip_datetime_pairs, log_file) -> None:
    """
    Displays The Array
    :param ip_datetime_pair:
    :param log_file:
    :return: None
    """
    print("IP addresses and corresponding date/time found in the log file
    ↪  {}:".format(log_file))
    for ip, datetime_obj in ip_datetime_pairs:
        print(f"IP: {ip}, Date: {datetime_obj}")

def main() -> None:
    log_file_path = "/var/log/auth.log"
    ip_datetime_pairs = extract_ip_and_datetime(log_file_path)
    dispalyData(ip_datetime_pairs, log_file_path)

    log_file_path = "/var/log/syslog"
    ip_datetime_pairs = extract_ip_and_datetime(log_file_path)
    dispalyData(ip_datetime_pairs, log_file_path)

if __name__ == "__main__":
    main()
```

No que diz respeito aos resultados obtidos através da execução do *script* acima definido, é possível ver, listagem 4, os endereços de IP e a data que estes foram registados. Através desta informação já é possível ter dados internos de *logs* para serem relacionados com as fontes externas de dados e desta forma gerar alertas.

Listagem 4: Resultado do processamento dos ficheiros de *log*

```
martinho@ubuntu3:~$ python3 parseLogFiles.py
IP addresses and corresponding date/time found in the log file
↪  /var/log/auth.log:
IP: 192.168.87.1, Date: Apr 18 20:02:36
IP: 0.0.0.0, Date: Apr 19 21:03:35
IP: 192.168.87.1, Date: Apr 19 21:06:14
IP: 192.168.87.1, Date: Apr 19 21:08:57
IP: 0.0.0.0, Date: Apr 22 18:17:05
IP: 192.168.87.1, Date: Apr 22 18:18:38
IP: 192.168.87.1, Date: Apr 22 18:18:53
IP: 192.168.87.1, Date: Apr 22 18:18:57
IP: 192.168.87.1, Date: Apr 22 18:18:57
IP: 192.168.87.1, Date: Apr 22 18:35:47
IP: 192.168.87.1, Date: Apr 22 18:38:00
IP: 192.168.87.1, Date: Apr 22 18:43:08
IP: 192.168.87.1, Date: Apr 22 18:43:08
IP: 192.168.87.1, Date: Apr 22 19:36:39
IP: 192.168.87.1, Date: Apr 22 20:35:38
IP: 192.168.87.1, Date: Apr 22 20:35:38
IP: 192.168.87.1, Date: Apr 23 19:47:44
IP: 192.168.87.1, Date: Apr 23 20:18:34
IP addresses and corresponding date/time found in the log file /var/log/syslog:
IP: 91.189.91.157, Date: Apr 19 21:03:35
IP: 192.168.87.102, Date: Apr 19 21:03:35
IP: 255.255.255.0, Date: Apr 19 21:03:35
IP: 127.0.0.1, Date: Apr 19 21:03:35
IP: 255.0.0.0, Date: Apr 19 21:03:35
IP: 0.0.0.0, Date: Apr 19 21:03:35
IP: 192.168.87.2, Date: Apr 19 21:03:35
P: 0.0.0.0, Date: Apr 19 21:03:35
IP: 192.168.87.0, Date: Apr 19 21:03:35
IP: 0.0.0.0, Date: Apr 19 21:03:35
IP: 255.255.255.0, Date: Apr 19 21:03:35
IP: 185.125.190.58, Date: Apr 22 18:17:05
IP: 192.168.87.102, Date: Apr 22 18:17:05
IP: 255.255.255.0, Date: Apr 22 18:17:05
IP: 127.0.0.1, Date: Apr 22 18:17:05
IP: 255.0.0.0, Date: Apr 22 18:17:05
IP: 0.0.0.0, Date: Apr 22 18:17:05
IP: 192.168.87.2, Date: Apr 22 18:17:05
IP: 0.0.0.0, Date: Apr 22 18:17:05
IP: 192.168.87.0, Date: Apr 22 18:17:05
IP: 0.0.0.0, Date: Apr 22 18:17:05
IP: 255.255.255.0, Date: Apr 22 18:17:05
```

4.3.2 *Recolha de tráfego*

Existem várias formas pelas quais podemos recolher tráfego para analisar. Nesta secção serão abordados quatro métodos diferentes para realizar a recolha de tráfego. Os métodos são, através de softwares (utilização de IDS e IPS), pela recolha de tráfego manual através de ficheiros ".pcap", pela recolha em tempo real através de *streams* de capturas de tráfego (ex. *tshark*, *tcpdump*) ou por fim, da recolha diretamente na aplicação. Para fazer os testes e selecionar uma das técnicas serão

tidos em conta os seguintes pontos, de modo a conseguir implementar uma solução de fácil desenvolvimento e utilização por parte do utilizador:

1. Facilidade de recolha de dados - Se o utilizador, com pouco esforço consegue enviar dados para a aplicação;
2. Facilidade de tratamento de dados - Se os dados recolhidos são fáceis de tratar e utilizar;
3. Método de envio - Qual o método de mais fácil e viável para enviar os dados para a aplicação;
4. Informação pretendida - Se a informação recebida vai ao encontro daquela que é necessária, neste caso, endereços de IP envolvidos na comunicação e nomes de domínio.

Os testes realizados, serão divididos em duas fases. A primeira, foca-se na recolha de tráfego de (tráfego DNS). Esta segunda fase consiste na definição do método de envio dos dados recolhidos para tratamento, ou seja, investigar qual a forma mais rápida e fácil para enviar os dados.

4.3.3 *Domain Name System*

Primeiro, é necessário perceber como funcionam as comunicações de DNS para entendermos o que é possível obter ao fazer o *sniff* dos pacotes. Existem dois tipos de servidor de DNS, de forma muito simples, os servidores recursivos são responsáveis por encontrar o servidor de DNS onde está alojado o registo. Os servidores autoritários são os servidores que tem neles alojados os registos. Os pedidos de DNS são feitos em 8 passos diferentes: (CloudFlare, 2024)

1. Um utilizador faz um pedido a 'www.ipleiria.pt' é recebida por um servidor DNS recursivo;
2. O servidor recursivo então consulta um servidor raiz DNS (.);
3. O servidor raiz responde ao servidor recursivo com o endereço de um servidor DNS de Top Level Domain (TLD) (como .com ou .pt), que armazena as informações dos seus domínios. Ao procurar por 'www.ipleiria.pt', o pedido é direcionado para o TLD '.pt';
4. O servidor recursivo então faz um pedido ao TLD '.pt';
5. O servidor TLD responde com o endereço IP do servidor de nomes de domínio (*nameserver*), ipleiria.pt;

6. O servidor recursivo envia uma consulta ao *nameserver* do ipleiria.pt;
7. O endereço IP para www.ipleiria.pt é então devolvido ao resolver pelo *nameserver*;
8. O servidor recursivo responde ao cliente com o endereço IP do domínio inicialmente solicitado (www.ipleiria.pt).

Após percebermos como é feita a comunicação no protocolo de DNS, percebemos que ao fazer a recolha do tráfego interno, apenas veremos os pontos 1 e 8, assumindo que o pedido é feito a um servidor de DNS exterior, todo o resto da comunicação é feita fora da rede da organização. Desta forma sabemos que iremos ver os pedidos de DNS internos bem como a resposta.

4.3.4 *Recolha através de software*

Abordando o primeiro método, a recolha foi feita através do uso softwares de IDS e/ou IPS. Neste caso, o teste será feito utilizando o *snort* para fazer a recolha do tráfego, aplicação que foi mencionada anteriormente.

Para realizar os testes com o *snort*, foram utilizadas numa primeira instância, uma máquina virtual com Ubuntu Server 22.04 (jammy) para realizar a recolha de pacotes referentes ao protocolo DNS (udp/53). De forma a atingir este objetivo foi necessário colocar o *snort* em modo de *sniffer* filtrando todos os pacotes que dizem respeito ao porto 53. A listagem 5, mostra-nos o comando utilizado bem como o output obtido.

Listagem 5: Captura de tráfego através do Snort

```
vagrant@ubuntu-jammy:~$ sudo snort -q -i enp@s8 -d udp port 53
↪  03/05-19:01:06.444590 192.168.4.8:44516 -> 192.168.4.254:53 UDP TTL:64 TOS:
↪  0x0 ID: 30471 IpLen:20 DgmLen:53
Len: 25
37 5A 01 00 00 01 ee ee ee ee ee ee 04 73 61 70 7Z..
6F 02 70 74 00 00 01 00 01
.sap
o.pt.
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/05-19:01:06.446703 192.168.4.8:33774 -> 192.168.4.254:53
UDP TTL:64 TOS: 0x0 ID: 9268 IpLen:20 DgmLen:53
Len: 25
35 73 01 00 00 01 ee ee ee ee ee ee 04 73 61 70 5s.... 6F 02 70 74 00 00 1c 00 01
o.pt.
.sap
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/05-19:01:06.462125 192.168.4.254:53 -> 192.168.4.8:44516 UDP TTL:64 TOS: 0x0
↪  ID: 22760 IpLen:20 DgmLen:69 DF
Len: 41
37 5A 81 80 00 01 00 01 00 00 00 00 04 73 61 70 7Z.. 6F 02 70 74 00 00 01 00 01
↪  Ce ec 00 01 00 01 00 o.pt. ee 00 37 00 04 D5 0D 92 8E
..7......
sap
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/05-19:01:06.466173 192.168.4.254:53 -> 192.168.4.8:33774 UDP TTL:64 TOS: 0x0
↪  ID: 22764 IpLen:20 DgmLen:97 DF
Len: 69
35 73 81 80 00 01 00 00 00 01 00 00 04 73 61 70 6F 02 70 74 00 00 1c 00 01 ce ec
↪  ee 06 00 01 00 00 08 01 00 20 02 6 73 C0 0 04 72 6F 6F 74 C0 OC 65 E5 AF 2C
↪  00 00 OE 10 00 00 1 20 00 24 EA 00 00 00 01 20
5s.
sap
o.pt.
.ns...root.
.e...
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
03/05-19:01:06.485480 192.168.4.8:59399 -> 192.168.4.254:53
UDP TTL:64 TOS: 0x0 ID: 3920 IpLen:20 DgmLen:73 Len: 45
83 59 01 ee ee el ee ee ee ee ee ee 03 31 34 32 03 31 34 36 02 31 33 03 32 31 33
↪  07 69 6E 2D 61 64 64 72 04 61 72 70 61 00 00 ec 00 01
.Y........
142
.146.13.213.in-a ddr.arpa..
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Como é possível observar, o comando utilizado "**sudo snort -q -i enp0s8 -d 'udp port 53'**"permite-nos obter os pacotes que dizem respeito ao tráfego UDP do porto 53. De uma forma mais detalhada:

1. -q: *quiet mode*, desta forma o *snort* não mostra o *banner* inicial;
2. -i: especificamos a interface na qual o *snort* vai fazer o *sniff* aos pacotes, neste caso a interface "enp0s8";
3. -d: De modo a podermos observar os dados da camada aplicacional, neste caso, é possível ver que os pedidos DNS dizem respeito ao domínio "sapo.pt";

4. 'udp port 53': O filtro de pacotes aplicado à *query* feita, neste caso, filtramos todos os pacotes que digam respeito ao porto udp 53.

No que diz respeito ao output do comando é possível obter a informação que é necessária, como os endereços de IP dos intervenientes na comunicação, os portos utilizados, o nome de domínio bem como a sua tradução para endereço IP. Podemos ver:

1. Endereços de IP locais: 192.168.4.8 (máquina que fez o pedido) e 192.168.4.254 (máquina que recebeu e tratou o pedido);
2. Portos: 44506 - Porto utilizado para o primeiro pedido; 33774 - Porto utilizado para o segundo pedido; 53 - Porto do serviço de DNS que recebeu o pedido;
3. Domínio: sapo.pt;
4. Endereço IP traduzido: 213.13.146.142

Contudo a forma como esta é disponibilizada não facilita a sua leitura nem tratamento. Como é possível ver na figura, o conteúdo do pacote está em hexadecimal e ASCII, contudo a sua leitura não é direta e exige que sejam aplicados filtros ao *snort* de modo a conseguir ler o conteúdo de forma fácil. Ao exigir isto ao utilizador elevamos o nível de dificuldade da aplicação.

4.3.5 *Recolha através de ficheiros*

A recolha de tráfego pode ser feita através do uso de ficheiros ".pcap"que são gerados e utilizados por *sniffers* de rede como o *Wireshark* ou o *NetworkMinner* que têm ambiente gráfico ou como o TCPDUMP ou o tshak que são as versões de linha de comandos. A utilização destes softwares não exige grandes conhecimentos, tanto na instalação como na sua utilização, tornando-se uma excelente forma para realizar a recolha de todo o tráfego. Contudo, este método para o cenário que aqui é apresentado, necessita de uma forma de processamento diferente para ser possível obter os dados relevantes para análise.

Para dar início aos testes com a recolha de dados através de ficheiros ".pcap", primeiro é necessário perceber o que é um ficheiro ".pcap". Um ficheiro ".pcap"é um tipo de ficheiro que é utilizado para armazenar os pacotes de rede (em cabo ou *wireless*) capturados por determinado *software*. Os pacotes recolhidos nestes ficheiros incluem todos os dados referentes às sete camadas do modelo Open Systems Interconnection. Este ficheiro foi desenvolvido na década de 90 juntamente com o

TCPDUMP e a biblioteca *libcap*. A grande vantagem destes ficheiros é que podem ser tratados utilizando várias ferramentas e métodos distintas. (Endace, 2024)

Para testar a utilização de ficheiros ".pcap"na aplicação foi realizada a recolha de tráfego com recurso ao *Wireshark*, figura 5, e guardado um ficheiro que contivesse o tráfego de rede. Neste rede foram gerados pedidos de DNS de modo a ser possível fazer a recolha dos dados que são relevantes para o teste. O pedido de DNS foi realizado numa máquina Linux com o sistema operativo Ubuntu 22.04LTS.

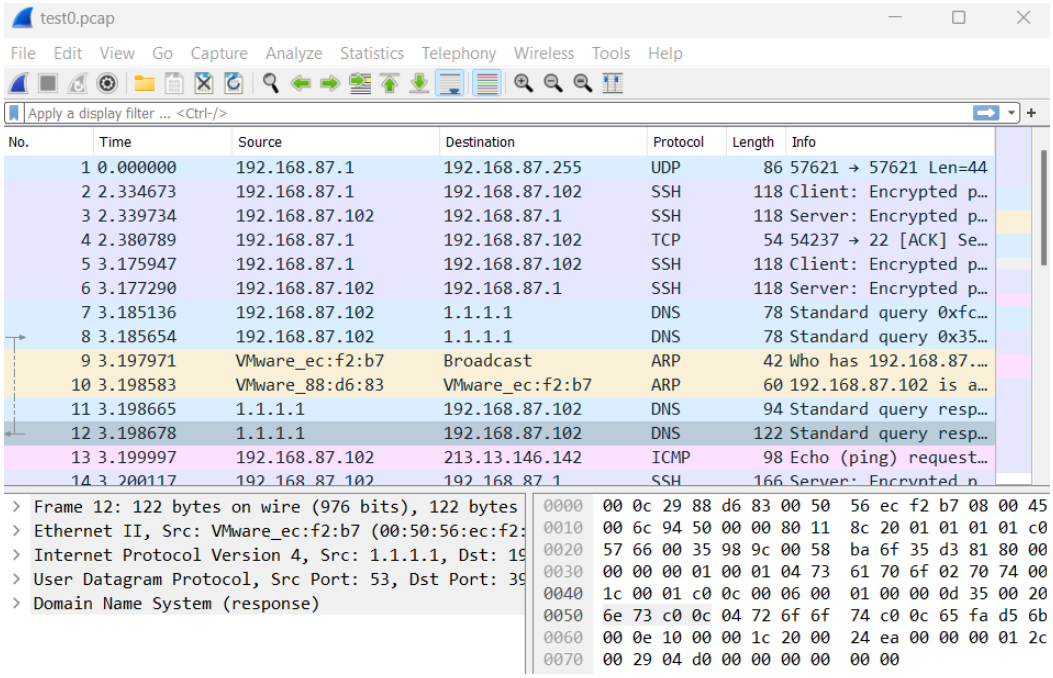


Figura 5: Recolha de tráfego através de Wireshark

Com a recolha de tráfego feita e com o ficheiro ".pcap"guardado foi necessário criar um *script* de prova de conceito, neste caso em *Python*, para fazer o *parsing* dos dados e tentar recolher informação relativa ao tráfego DNS. De forma a ser possível realizar a tarefa, foi instalada a biblioteca *pyshark* (KimiNewt, 2024), uma biblioteca que encapsula o *tshark* e permite, com recurso à linha de comandos e ao *tshark* recolher dados dos pacotes recolhidos. A listagem 6, mostra o código *Python* desenvolvido para atingir esta tarefa. O código foi desenvolvido de forma a:

1. Focar-se no tráfego DNS;
2. Recolher a respostas dos pedidos DNS;
3. Devolver dados como os endereços de IP do servidor e cliente, o pedido realizado e a resposta.

Listagem 6: Código fonte Python para *parsing* do ficheiro PCAP

```
import pyshark

print("Parsing DNS responses from pcap...")

capture = pyshark.FileCapture('test0.pcap')
for packet in capture:
  if hasattr(packet, 'dns'):
    if packet.dns.count_answers > "0":
        print("Query:"+ packet.dns.qry_name)
        print("DNS Server:"+ packet.ip.src) # neste caso é o ip do dns server
        print("Client:"+ packet.ip.dst) # quem fez o pedido
        if hasattr(packet.dns, "a"):
            print("Response: "+ packet.dns.a)
        else:
            print("Response:": packet.dns.ptr_domain_name)
```

Resta agora executar o *script* com os dados recolhidos e ver se é possível obter a informação pretendida. A listagem 7, mostra a informação recolhida e podemos observar que é possível ver os endereços de IP do servidor de DNS e cliente, o pedido bem como a resposta ao pedido. Neste exemplo, vemos um pedido ser feito para o domínio "sapo.pt"e a sua resposta bem como um pedido pelo registo DNS Pointer Record (ptr) ao mesmo domínio.

Listagem 7: Informação obtida através do ficheiro ".pcap"

```
martinho@ubuntu3:~$ python3 parsePCAP.py
Parsing DNS responses from pcap...
Query:sapo.pt
DNS Server:1.1.1.1
Client:192.168.87.101
Response: 213.13.146.142
Query:142.146.13.213.in-addr.arpa
DNS Server:1.1.1.1
Client:192.168.87.101
Response: sapo.pt
```

É importante salientar que este *script* apenas retira informação sobre as respostas aos pedidos DNS.

Visto que é possível obter a informação pretendida facilmente é importante definir como chegam os dados à aplicação. O objetivo com a utilização de ficheiros ".pcap"seria fazer o *parsing* destes ficheiros na própria aplicação, sendo apenas necessário que o utilizador faculte o ficheiro para ser tratado. Desta forma os dados são introduzidos na aplicação manualmente tendo a vantagem de poderem ser capturas de tráfego de qualquer ponto de rede, contudo, existe a desvantagem de não poder ser feito o tratamento dos dados em tempo real.

4.3.6 *Recolha em tempo real*

Um dos pontos cruciais da resposta a possíveis incidentes de segurança é a rápida deteção de anomalias. De forma a conseguir detetar com rapidez é necessário analisar os dados, neste caso o tráfego de rede, o mais rápido possível. Para atingir esse objetivo é necessário tratar os dados em tempo real.

O tratamento de dados em tempo real pode ser obtido tirando partido de *sniffers* de rede, para recolher o tráfego, e a criação de um método em que seja possível o seu envio diretamente para a aplicação para os dados serem tratados. No que diz respeito aos *sniffers*, foram realizados testes com o *tshark*, abordado anteriormente. À semelhança dos testes anteriores, o objetivo é perceber se esta é uma forma viável de recolher informação e envia-la para tratamento. Para isso foram utilizados comandos, do *sniffer* mencionado, para recolher os dados necessários. O cenário de testes, neste caso de uso, consiste em 3 máquinas virtuais. A primeira responsável pela recolha do tráfego, a segunda por gerar tráfego DNS e a terceira que simula o servidor da aplicação para onde é enviada a informação, caso seja possível recolher a informação necessária.

O primeiro teste realizado foi com o *tshark*, com o comando **"sudo tshark -i ens33 -n -T fields -e dns.qry.name -e ip.src -e ip.dst -e dns.a src port 53"** onde foi possível recolher a informação que a listagem 8 mostra. Com este comando, como é possível constatar, recolhemos os dados sobre os endereços de IP envolvidos na comunicação, os pedidos bem como as suas respostas, à exceção do registo ptr que pode ser recolhido através do método anterior.

Listagem 8: Recolha de tráfego DNS através do *tshark*

```
1
2 martinho@ubuntul:~$ sudo tshark -i ens33 -n -T fields -e dns.qry.name -e ip.src
  ↪  -e ip.dst -e dns.a src port 53
3 Running as user "root" and group "root". This could be dangerous. Capturing on
  ↪  'ens33'
4 ** (tshark:1785) 10:33:27.424251 [Main MESSAGE] -- Capture started.
5 ** (tshark:1785) 18:33:27.424543 [Main MESSAGE] -- File:
  ↪  "/tmp/wireshark_ens33L065K2.pcapng"
6 sapo.pt 9.9.9.9 192.168.87.101 213.13.146.142
7 sapo.pt 9.9.9.9 192.168.87.101
8 142.146.13.213.in-addr.arpa
9 9.9.9.9 192.168.87.101
```

O comando utilizado recolhe informação dos pedidos DNS e está dividido da seguinte forma:

1. **-i ens33**: seleciona a interface pela qual é feita a recolha do tráfego;

2. **-n**: não faz a tradução de nomes;
3. **-T fields**: formata a saída do comando para os campos selecionados;
4. **-e <fieldName>**: os campos que pretendemos ver na saída do comando. Neste caso os campos são a *query* feita no pedido, o endereço de IP origem, o endereço IP do servidor de DNS e por fim a resposta ao pedido de tradução de nome;
5. **src port 53**: captura apenas tráfego da porta 53.

Com a fase de recolha de dados feita, é então necessário pensar numa forma para enviar os dados do ponto de captura até ao ponto de tratamento. Para proceder ao envio foram feitos testes utilizando o *netcat*. Esta ferramenta permite criar uma ligação TCP entre as duas máquinas e partilhar desta forma informação. Para proceder ao envio dos dados foram realizados 4 tarefas:

1. Abrir um porto TCP para escuta e guardar os dados recebidos num ficheiro PCAP;
2. Recolher tráfego DNS e enviá-lo para o porto aberto numa máquina remota
3. Gerar tráfego DNS para ser capturado.
4. Validar se o ficheiro PCAP gerado pode ser consumido através do script elaborado anteriormente (6)

A primeira parte foi conseguida através do comando **"nc -lp 32000 > test0.pcap"**, , como podemos ver na listagem 9. Este comando permite colocar o porto 32000 à escuta de ligações TCP e depois envia os dados recolhidos para o ficheiro "test0.pcap", para depois poder ser processado. Toda esta etapa é realizada no servidor onde está alojada a aplicação, neste caso em ambiente simulado.

Listagem 9: Netcat - porto à escuta e envio dos dados para um ficheiro PCAP

```
1 martinho@ubuntu3:~$ nc -lp 32000 > test0.pcap
```

De seguida, é necessário colocar uma máquina a recolher todo o tráfego DNS, e para isso, foi utilizado um comando idêntico ao referido anteriormente, mas como a nuance que desta vez o *output* do comando é redirecionado para o porto 32000 à escuta noutra máquina. O comando utilizado, listagem 10, foi o seguinte: **"sudo tshark -w - -i ens33 -n src port 53 | nc 192.168.87.102 32000"**. À semelhança do comando explicado anteriormente, este permite recolher o tráfego DNS gerado na rede e envia todo o *output* para uma máquina remota no porto 32000. Após

colocar a máquina à escuta de tráfego DNS uma terceira máquina fez um pedido para gerar tráfego para ser capturado e enviado.

Listagem 10: Envio do tráfego recolhido utilizando o netcat

```
martinho@ubuntu1:~$ sudo tshark -w- -i ens33 -n src port 53 | nc 192.168.87.102
↪  32000
Running as user "root" and group "root". This could be dangerous. Capturing on
↪  'ens33'
** (tshark:1331) 20:08:48.677586 [Main MESSAGE] -- Capture started.
** (tshark:1331) 20:08:48.677799 [Main MESSAGE] -- File: "-"
3
```

Após a captura ter sido feita, foi criado um novo ficheiro PCAP com o nome de **"test0.pcap"**. Para garantir que a recolha foi bem feita e que o ficheiro PCAP é utilizável, foi executado o *script* criado anteriormente, 6, e garantir que é possível extrair a informação necessária. A listagem 11 mostra os dados que foram possíveis recolher.

Listagem 11: *Output* da execução do *script* ao PCAP recolhido remotamente

```
martinho@ubuntu3:~$ python3 parsePCAP.py
Parsing DNS responses from pcap...
Query: sapo.pt
DNS Server:1.1.1.1 Client:192.168.87.101
Response: 213.13.146.142 Query:142.146.13.213.in-addr.arpa
DNS Server:1.1.1.1
Client: 192.168.87.101 Response: sapo.pt
```

4.3.7 *Recolha na aplicação*

Fazer a recolha de tráfego diretamente na aplicação permite que, à semelhança do ponto anterior, este possa ser consumido em tempo real e a resposta a possíveis ameaças seja mais rápida. Para além disso, este método tiraria a dificuldade existente de colocar uma máquina a capturar o tráfego de rede e enviá-lo para a aplicação para processamento. No entanto, esta abordagem não permite que a recolha seja feita de qualquer lado da rede da organização, salvo a existência de configurações de rede especificas para permitir que o tráfego seja espelhado para a interface de captura da máquina onde está a aplicação (*Port mirroring*).

Para proceder a este teste, foi desenvolvido um pequeno script, com base no já desenvolvido anteriormente 6 fazendo algumas alterações. Esta alteração permite que a captura seja feita na máquina ao invés de tratar um ficheiro de PCAP. Este

script é apenas uma prova de conceito, para poder validar que este é um método implementável.

Listagem 12: Código fonte Python para captura e tratamento dos pacotes de rede

```
import pyshark

def handles_dns_packets(packet) -> None:
    if packet.dns.count_answers > "0":
        print("Query:"+ packet.dns.qry_name)
        print("DNS Server:"+ packet.ip.src) # neste caso é o ip do dns server
        print("Client:"+ packet.ip.dst) # quem fez o pedido
        if hasattr(packet.dns, "a"):
            print("Response: "+ packet.dns.a)
        else:
            print("Response: "+ packet.dns.ptr_domain_name)


def capture_traffic(*, interface: str, shark_filter: str) -> None:
    # Inicia a captura na interface fornecida com base no filtro facultado
    capture = pyshark.LiveCapture(interface=interface, bpf_filter=shark_filter)

    try:
        for packet in capture.sniff_continuously(packet_count=100):
            if hasattr(packet, 'dns'):
                handles_dns_packets(packet)

    except KeyboardInterrupt:
        print("Capture stopped by user.")

def main() -> None:
    capture_iface = "ens33"
    capture_filter = 'udp port 53'
    capture_traffic(interface=capture_iface, shark_filter=capture_filter)

if __name__ == "__main__":
    main()
```

Como é possível ver no código acima (12), este inicia uma captura numa dada interface de rede com determinado filtro, neste caso com o filtro para a porta 53 UDP e faz o tratamento dos pacotes recolhidos. A listagem 13, mostra o resultado do teste feito ao *script* 12, onde foram feitos dois pedidos de DNS o primeiro a "sapo.pt"e o segundo a "ipleiria.pt". À semelhança do *script* anterior este mostra os dados estruturados da mesma forma, podendo assim obter a informação relevante para ser processada posteriormente.

Listagem 13: Resultado da captura de tráfego na aplicação

```
martinho@ubuntu3:~$ sudo python3 capturePcap.py
Query: sapo.pt
DNS Server:1.1.1.1
Client: 192.168.87.101
Response: 213.13.146.142
Query: 142.146.13.213.in-addr.arpa
DNS Server:1.1.1.1
Client:192.168.87.101
Response: sapo.pt
Query: ipleiria.pt
DNS Server:1.1.1.1
Client: 192.168.87.101
Response: 194.210.216.28
```

4.3.8 *Recolha de dados externos*

Toda a recolha e tratamento dos dados internos da organização acaba por ter pouco significado neste cenário se não for cruzada com dados recolhidos através de fontes externas. Aqui será abordada a forma como os dados serão recolhidos, tratados e guardados.

4.3.8.1 *Base de dados*

O objetivo principal da recolha de dados de fontes externas é poder cruzar os dados com os que são recolhidos internamente. Os dados externos são recolhidos periodicamente e necessitam de ser armazenados para serem utilizados sempre que necessário, quer isto dizer que existe a necessidade de utilizar um sistema de base de dados.

Tendo em conta que a abordagem tomada para a realização deste trabalho poderá levar à recolha de milhares de registos surgiu a dúvida de que tipo de sistema de base de dados deveria ser utilizado. Aqui, e seguindo os sistemas mais complexos existentes no mercado como é o caso dos SIEM, poderíamos pensar em criar uma base de dados não relacional, sendo que estas são as mais indicadas para estes projetos. Contudo, tratando-se isto de uma prova de conceito e não havendo a gritante necessidade de aplicar uma base de dados deste tipo, apontamos para a criação de uma base de dados relacional bastante simples de modo a conseguir armazenar os dados relevantes colecionados e de realizar pesquisas rápidas.

Surge a dúvida de qual será o sistema de base de dados mais adequado para cumprir esta tarefa. As duas opções lógicas para sistemas relacionais são o MySQL e

o PostgreSQL, pois ambos são *opensource*. Estes são dois sistemas de bases de dados muito semelhantes, contudo existe uma grande diferença que tornou a escolha entre os dois evidente. O MySQL apesar de ser bastante rápido em operações de leitura, não é tão eficaz em operações de leitura e escrita de grandes *datasets* (Ravoof, 2023), como é o caso do presente projeto.

Posto isto, a base de dados terá, como tabelas responsáveis para a recolha e armazenamento dos dados, duas tabelas. A primeira onde serão armazenados os *feeds* (fontes de dados externas) para serem consumidas e a segunda tabela será responsável por armazenar todos os dados recolhidos destas fontes. Esta tabela de armazenamento dos dados recolhidos terá os seguintes campos:

1. Tipo - O tipo de valor que é, isto é, endereço IP ou nome de domínio;
2. Descrição - Qual o contexto deste valor, que virá da fonte de dados, e indicará se este é um endereço conhecido por *bruteforce* ou até se é um C2, por exemplo;
3. Valor - O valor recolhido;
4. Valor de validação (*checkValue*)- Valor que irá definir a continuidade deste dado na base de dados e será através deste que será categorizados os níveis de alerta. Este valor inicialmente é igual a metade do número de fontes externas configuradas.

4.3.8.2 *Mecanismo de Recolha*

O mecanismo idealizado para a recolha dos dados externos tem por base a utilização de tarefas calendarizadas, quer isto dizer, que a recolha das várias fontes definidas serão feitas através de *cronjobs*. Estas tarefas serão responsáveis por periodicamente recolherem os dados disponíveis nas fontes externas, tratar esses dados e armazená-los na base de dados definida para serem posteriormente utilizados.

O funcionamento destas tarefas está dividido em três partes:

1. Recolha, onde a aplicação vai a cada fonte externa, através de um URL, buscar os dados;
2. Tratamento dos dados, onde a aplicação trata os dados recolhidos, valida a sua existência e atualiza o valor de validação;
3. Limpeza da base de dados que é responsável por eliminar todos os valores que já não são necessários.

4.3.9 *Dados inseridos pelo utilizador*

Os dados inseridos pelo utilizador, como já referido anteriormente, podem ser bastante importantes para que a aplicação seja levada para o nível seguinte, no que diz respeito às suas funcionalidades. Contudo, e visto que esta primeira implementação da aplicação ser apenas um demonstrador, os dados a serem introduzidos pelo utilizador são apenas de interação com a aplicação para conseguir tirar partido das funcionalidades descritas anteriormente. O utilizador deverá poder interagir com a aplicação da seguinte forma:

1. Upload de ficheiros - O upload de ficheiro serve para o utilizador analisar ficheiros de *log* e recolhas de tráfego;
2. Ativar e desativar o *listener* - Funcionalidade que permite ao utilizador definir um porto para onde são enviadas capturas de tráfego remoto;
3. Ativar e desativar o *sniffer* - Funcionalidade que permite ao utilizador ativar a captura de tráfego diretamente na máquina da aplicação;
4. Ativar e desativar o *rsyslog* - Esta funcionalidade permite ao utilizador configurar, nas máquinas que considere pertinente, o envio de logs através de *rsyslog* para a máquina onde está a aplicação.

4.3.10 *Alertas*

Os alertas são o ponto principal da aplicação, pois é neles que está a grande utilidade e a forma simples como são despoletados. Estes alertas podem ser despoletados tanto através de tráfego de rede como de ficheiros de *log*. Estes dados são introduzidos manualmente pelo utilizador para serem analisados ou podem ser introduzidos de forma automática, tal como referido nos pontos anteriores. Estes alertas são divididos em quatro níveis de severidade (Crítico, Alto, Médio e Baixo) diferente com base num algoritmo bastante simples que tem em conta o valor de validação (*checkValue*) e o número de fontes externas configuradas. Ou seja a classificação dos alertas na escala acima definida tem por base a seguinte fórmula: $x = y/z$, em que x é o nível de alerta, y é o valor de validação para aquele dado recolhido e z é o número de fontes externas configuradas. Quanto maior for o numero de fontes, mais fidedigno será o valor alcançado para o nível de alerta. Desta forma os alertas terão a seguinte classificação:

1. Baixo - Se o nível de alerta for $>$ que 0 e $<$ que 0.24;

2. Médio - Se o nível de alerta for >= a 0.24 e < que 0.4;
3. Alto - Se o nível de alerta for >= a 0.4 e < que 0.9;
4. Crítico - Se o nível de alerta for >= a 0.9.

4.3.11 *Processamento de dados*

Como referido anteriormente, o processamento de dados é o ponto de cruzamento de todos os dados recolhidos. O objetivo principal é criar vários mecanismos que agreguem todas as funcionalidades faladas anteriormente e disponibiliza-las ao utilizar de uma forma simples, e com isto gerar alertas de uma forma simples e eficaz. No processo de processamento de dados serão desempenhadas as seguintes tarefas:

1. Recolha e processamento automático dos dados das fontes externas;
2. Recolha e processamento dos dados introduzidos pelo utilizador;
3. Recolha e processamento de tráfego de rede e de ficheiros *log*;
4. Despoleta de alertas.

5

DESENVOLVIMENTO DA APLICAÇÃO

Este capitulo destina-se a explicar todo o processo de desenvolvimento da aplicação, implementando as funcionalidades descritas no capitulo anterior (4) onde é abordada a proposta de arquitetura a implementar. De salientar que todo o código está disponível neste repositório doo *Github*, (Gonçalves, 2024).

5.1 ARQUITETURA DE DESENVOLVIMENTO

A arquitetura de desenvolvimento desta aplicação está dividida em duas partes distintas, o *backend* responsável pelo processamento dos dados e o *frontend* onde o utilizador interage com a aplicação. No que diz respeito *backend*, este foi desenvolvido com o objetivo de criar uma Application Programming Interface (API) que permite uma fácil integração com outras ferramentas. Quanto ao *frontend*, este foi desenvolvido aplicando o modelo de Single Page Application (SPA). Todo este sistema assenta em cima de uma base de dados relacional em PostgresSQL, tal como já definido anteriormente no capitulo 4, subcapítulo 4.3.8.1.

5.1.1 *Backend*

A escolha da linguagem de programação para o desenvolvimento do *backend* foi o *Python* pois é uma linguagem versátil que permite a integração e utilização de diferentes bibliotecas que possibilitam desenvolver as tarefas definidas no capitulo 4. O *backend* foi desenvolvido utilizando principalmente as seguintes bibliotecas:

1. Django - Para toda a gestão dos modelos e interligação com a base de dados, bem como fazer a gestão e criação das tabelas na base de dados;
2. Django Rest Framework - Para a criação da API;
3. Django Crontab - Utilizado para criar as tarefas calendarizadas;
4. psycopg2 - Responsável pela conexão à base de dados;
5. pyshark - Para realizar a captura de tráfego;

6. psutil - Para a interação com a linha de comandos.

No código desenvolvido para o *backend*, podem ser destacadas três partes cruciais para o seu funcionamento. Os *Models* (Modelos) que representam as classes a ser utilizadas e são responsáveis pela representação das tabelas da base de dados, a API que é responsável pela interação com o *frontend* e as tarefas calendarizadas que executam tarefas automaticamente, sem ser necessária interação do utilizador.

5.1.2 *Modelos*

Como já referido, a API é foi desenvolvida utilizando bibliotecas como o Django e o Django Rest Framework. No desenvolver desta aplicação, e mais especificamente na API, foram criados os seguintes modelos que formam o *core* da aplicação bem como da base de dados:

1. *Alert*
2. *Categoty*
3. *Feed*
4. *Value*
5. *Helper*

Elaborando mais sobre estes modelos, e começando pelo modelo referente aos alertas (*Alert*), que é o modelo responsável por todos os alertas gerados na aplicação, este é definido por quatro campos:

1. *level* - Do tipo *varchar* com até 20 caracteres, é o que define o nível de alerta e pode ter como valores *"Low"*, *"Medium"*, *"High"* e *"Critical"*;
2. *message* - Do tipo *Text* guarda a mensagem de alerta gerado ;
3. *acknowledge* - Do tipo *Boolean*, diz se o alerta já foi ou não visto;
4. *created_at* - Do tipo *timestamp*, serve para indicar o momento em que o alerta foi gerado.

Os próximos dois modelos são referentes à gestão das fontes externas. Nestes são definidas as categorias (*Category*) das fontes, isto é, se são, por exemplo, dados referentes a C2, a máquinas conhecidas por tentativas de *bruteforce* ou até listas endereços IP ou domínios classificados como maliciosos (*malware*). O modelo responsável por definir as fontes de dados externas é o *Feed*, é com base nesta ultima que as tarefas calendarizadas se baseiam e são recolhidos os dados externos

à infraestrutura da organização. O modelo *Category* é definido apenas pelo seguinte campo:

1. *name* - Do tipo *varchar* com até 80 caracteres é o campo onde definimos os tipos de fontes externas que existem. Este foi criado desta forma para no futuro ser possível acrescentar tipos de fontes externas facilmente.

O modelo *Feed* é definido pelos seguintes campos:

1. *category* - Este campo é uma chave estrangeira para o ID da tabela definida pelo modelo *Category*, desta forma podem facilmente ser acrescentados e alteradas as categorias das fontes de dados;
2. *name* - Do tipo *varchar* com até 255 caracteres, tem o fim de dar um nome fácil de reconhecer à fontes externa;
3. *url* - Do tipo *varchar*, é o campo onde é definido o URL para que o mecanismo de recolha da aplicação tenha um *endpoint* para ir recolher os dados;
4. *parser* - Do tipo *varchar* com até 10 caracteres, é neste campo onde é definida a forma como os dados recolhidos são tratados na fase de processamento de dados. Este campo pode ser definido por "json", "txt"e "csv";
5. *duration* - Do tipo *interval*, é utilizado para guardar o tempo em que cada fonte de dados é atualizada. Nesta implementação do demonstrador este campo não é utilizado, pois como será explicado mais à frente a recolha de dados é feita através de tarefas calendarizadas;
6. *delimiter* - Do tipo *varchar* com até 1 carácter é utilizado para definir um delimitador (por exemplo um espaço no caso de um ficheiro "text") para o processamento de dados. É utilizado em conjunto com o campo *parser* e dependendo o tipo de definido é utilizado de forma diferente.
7. *delimiterField* - Do tipo *integer* é utilizado para definir o índice onde está o valor que se pretende processar, ou seja, num ficheiro texto recolhido que tem os vários dados delimitados por um espaço, ao definir o campo *delimiterField* com um 3, este irá buscar o campo que se encontra no índice 3.

Definidos os modelos que compõem as fontes de dados externas é necessário um modelo que guarde toda a informação útil recolhida. É neste sentido que foi criado o modelo *Value* e é definido pelos seguintes campos:

1. *type* - Do tipo *varchar* com até 20 caracteres é utilizado para definir o tipo de dados que está a ser tratado. Estes dados podem ser do tipo "*Domain*"e "*IP Address*";

2. *Description* - Do tipo *varchar* com até 250 caracteres é um descrição para indicar qual o contexto deste valor. Neste demonstrador, este valor está a ser preenchido tendo por base a categoria da fonte externa;
3. *value* - Do tipo *varchar* com até 500 caracteres é o campo responsável por guardar o dado recolhido que será utilizado para lançar alertas;
4. *checkValue* - Do tipo *interger* é utilizado para poder definir o nível de alerta a ser lançado (falado anteriormente na subsecção 4.3.10) e também para definir a continuidade deste registo na base de dados.

Por fim, e não menos importante, o modelo *Helper* foi criado com o objetivo de guardar as interações com o utilizador. Aqui é possível indicar se a recolha de tráfego interno esta ser feita localmente ou através de uma máquina remota que está a enviar tráfego ou se esta recolha de dados internos também está a ser feita através de *rsyslog*. A tabela apenas tem um registo na base de dados. Posto isto, este modelo é composto pelos seguintes campos:

1. *is_listener_running* - Do tipo *boolean* valida se a funcionalidade de receber tráfego remoto está ligada ou desligada;
2. *listener_pid* - Do tipo *integer*, guarda o Process Identifier do processo responsável por executar o *listener*, desta forma este pode ser terminado quando o utilizador entender;
3. *is_sniffer_running* - Do tipo *boolean* valida se a funcionalidade de recolha local de tráfego está ligado ou desligada;
4. *sniffer_id* - Do tipo *integer*, guarda o PID do processo responsável por executar o *sniffer* para que seja possível terminar o processo quando o utilizador entender;
5. *is_using_rsyslog* - Do tipo *boolean*, define se a aplicação também está a receber logs através de *rsyslog* e desta forma os possa tratar os mesmos.

Com a criação de todos os modelos, e tendo em conta a utilização do Django para o desenvolvimento da API, esta biblioteca permite que sejam criadas as migrações necessárias para a criação da base de dados. Estas migrações são criadas através dos modelos aqui definidos. Desta forma é possível recriar facilmente a base de dados tendo acesso ao código fonte desta aplicação.

5.1.3 *Application Programming Interface (API)*

Após desenvolvido o esqueleto da aplicação através da criação dos modelos e consequentemente da estrutura da base de dados, os passos seguintes a serem desenvolvidos foram os *enpoints* da API para interligação com o *frontend*.

Os *endpoits* da API vão desde as funções de Create, Read, Update and Delete dos modelos acima definidos, com algumas exceções, à interação com as ferramentas de tráfego e de tratamento de ficheiros que não estão diretamente ligadas a estes modelos. A tabela 4 mostra todos os *endpoints* disponíveis na aplicação, quais as funções de CRUD disponíveis, os métodos permitidos e uma descrição sobre o *endpoint* que explica a sua funcionalidade.

Nesta explicação, apenas serão abordados em mais detalhe os *endpoints* seguintes, pois estes são o *core* do funcionamento da aplicação sendo que os restantes são apenas funcionalidades de CRUD dos modelos acima definidos.

1. start-listener/ e stop-listener/ - Tal como é possível ver na tabela 4, estes *endpoints* são responsáveis por criar e terminar a tarefa que permite a aplicação recolher tráfego que está a ser capturado remotamente.

   Este *endpoint* ao ser utilizado para iniciar o *listener*, fica à escuta num porto definido pelo utilizador ou por omissão no porto 32000. Após inicializar, todo o tráfego recolhido está a ser gravado num ficheiro que será analisado assim que o utilizador decidir terminar a tarefa. A aplicação inicia um processo com o comando "**nc -l <porto>**"e guarda o PID do processo para ser terminado mais tarde;

2. start-sniffer/ e stop-sniffer/ - Estes *endpoints* são responsáveis por iniciar e terminar o sniffer na aplicação. Utiliza um *script* desenvolvido à parte da aplicação que utiliza as seguintes bibliotecas:

   a) sys - Para utilizar comandos do sistema;

   b) pyshark - Para a captura de tráfego;

   c) psycopg2 - Para a ligação à base de dados

   Em suma, este *script* recebe como argumento uma interface definida pelo utilizador e passada através da API, estabelece uma ligação à base de dados e recolhe tráfego relativo a traduções de DNS. Com a ligação à base de dados, todas as traduções de DNS detetadas são comparadas aos dados recolhidos nas fontes externas. No caso de algum valor corresponder tanto ao endereço IP

ou domínio, é gerado um alerta e guardado diretamente na base de dados, sem que seja necessário interação com a API para agilizar o processo, desta forma o utilizador pode ver no *frontend* os alertas gerados. À semelhança do *endpoint* anterior, o *script* é executado através do comando "**python networksniffer.py <interface>**"(*script*: "*backend/siemapi/networkSniffer.py*") e é guardado o PID do processo para que possa ser terminado posteriormente pelo utilizador;

3. upload-file/ - Este *endpoint*, tal como o nome indica, permite ao utilizador fazer upload de ficheiros. Apenas permite que sejam ficheiros *.log* e *.pcap*, validando nesta fase inicial apenas a extensão do ficheiro. Esta diferença entre tipo de ficheiros dita a forma como estes são tratados.

   No que diz respeito aos ficheiros *.log*, estes são corridos linha a linha, e neles são validadas a existência de endereços de IP e de nomes de domínio. No caso de serem encontrados são validados com os que existem na base de dados e na eventualidade de existir correspondência são gerados alertas para cada um dos dados.

   Quanto aos ficheiros *.pcap*, são validados apenas os pacotes referentes a pedidos de DNS e segue a mesma lógica anterior, no caso de serem encontradas resoluções com nomes de domínio ou endereços de IP com correspondência na base de dados é gerado um alerta para cada um dos destes dados;

4. get-stats/ - Este *endpoint* é apenas informativo, sendo possível obter informações como o número total de alertas, o número total de alertas não lidos, o número total de fontes externas configuradas, o número total de dados recolhidos nas fontes externas bem como uma listagem dos últimos 6 alertas criados;

5. network-interfaces/ - Este *endpoint* devolve todas as interfaces de rede disponíveis na máquina, para que desta forma o utilizador possa definir qual a interface que pretende fazer a recolha local de tráfego (*sniffer*).

Modelo	Enpoint	Create	Read	Update	Delete	Metodos Permitidos	Descrição
Feed	feed/	X	X			GET, POST	Permite listar todas as fontes e criar uma nova fonte
Feed	feed/<int:pk>		X	X	X	GET, PUT, DELETE	Permite obter, atualizar e apagar um feed especifico
Category	category/	X	X			GET, POST	Permite lista todas as categorias e criar uma nova categoria
Category	category/<int:pk>		X	X	X	GET, PUT, DELETE	Permite obter, atualizar e apagar um feed especifico
Value	value/		X			GET	Permite listar todos os valores
Value	value/<int:pk>		X			GET	Permite obter um valor especifíco
Alert	alert/		X			GET	Permite listar todos os alertas
Alert	alert/<int:pk>		X	X		GET, PUT	Permite obter um alerta especifico e atualiza-lo
Helper	helper/<int:pk>		X	X		GET, PUT	Permite obter e atualizar o registo na base de dados
	start-listener/					POST	Permite inicializar o listener
	stop-listener/					POST	Permite terminar o listener
	start-sniffer/					POST	Permite inicializar o sniffer
	stop-sniffer/					POST	Permite terminar o sniffer
	upload-file/					POST	Permite fazer o upload de ficheiros (.log e .pcap)
	get-stats/					GET	Permite obter as contagem de todos os modelos
	network-interfaces/					GET	Permite obter a lista de interfaces de rede presentes na máquina

Tabela 4: *Enpoints* disponíveis

A tabela 5 mostra-nos a forma como devem ser feitas as chamadas aos vários *endpoints* da API, indicando para cada um dos *endpoints* os campos necessários e o seu tipo. Com estas duas tabelas temos a noção do que é possível realizar com cada uma das chamadas à API. Todas as chamadas à API devem ter por base o seguinte esquema: *<protocol>://<hostname>:<port>/api/<endpoint>*, como por exemplo "*http://192.168.87.102:8000/api/start-listener/*".

Endpoint	**Parâmetros**

feed/	{
	"category": <int>,
	"name": <str>,
	"url": <str>,
	"parser": <str>,
	"refresh": <int>,
	"delimeter": <str>,
	"delimeterField": <int>
	}

feed/<int:pk>	{
	"category": <int>,
	"name": <str>,
	"url": <str>,
	"parser": <str>,
	"refresh": <int>,
	"delimeter": <str>,
	"delimeterField": <int>
	}

category/	{ "name": <str> },
category/<int:pk>	{ "name": <str> },
value/	—
value/<int:pk>	—
alert/	—
alert/<int:pk>	—

helper/<int:pk>	{
	is_listener_running: <boolean>,
	is_sniffer_running: <boolean>,
	is_using_rsync: <boolean>
	}

start-listener/	{
	port: <int>
	}

stop-listener/	—

start-sniffer/	{
	interface: <str>
	}

stop-sniffer/	—

upload-file/	{
	file: <file>
	}

get-stats/	—
network-interfaces/	—

Tabela 5: *Endpoints* e parâmetros da API

5.1.4 *Calendarização de tarefas*

Nesta aplicação foram desenvolvidas duas tarefas calendarizadas. A primeira foca-se na recolha de dados em fontes externas que realiza o tratamento e armazenamento da informação recolhida. A segunda tarefa é responsável por fazer o tratamento dos dados internos da organização, neste caso de ficheiros de *logs*, utilizando o serviço de *rsyslog* para envio automático para o servidor onde a aplicação está alojada. Todo o código relativo às tarefas encontra-se no diretório "*backend/siemapi/api/management/cronjobs*"do repositório do *GitHub*

5.1.4.1 *Recolha de dados externos*

Uma das bases essenciais desta aplicação é a inclusão de dados externos às organizações que possam tornar mais simples a deteção de possíveis anomalias. Como já referido anteriormente (5.1.2) as fontes externas são definidas com a criação de *Feeds* e neste demonstrador foram definidos duas fontes externas diferentes. A primeira fonte definida reúne uma listagem de endereços IP que representam ameaças, como por exemplo, endereços que são conhecidos por fazer ataques ao serviço Secure Shell ou ao serviço File Transfer Protocol entre outras (Fail2ban, 2024). A segunda fonte constitui uma lista de domínios gerados através de um algoritmo (DGA) que geralmente são utilizados por atores maliciosos. A tabela 6 mostra como estão definidos na base de dados.

Name	blocklist.de - All	bambenekconsulting
Category	IP Addresses	DGA Domains
URL	https://lists.blocklist.de/lists/all.txt	https://faf.bambenekconsulting.com/feeds/dga-feed.txt
Parser	txt	csv
Refresh	12:00:00	13:00:00
Delimiter		,
DelimiterField	0	0

Tabela 6: Fontes externas definidas

Definidas as fontes externas a serem utilizadas pela aplicação, passamos ao passo de definir uma tarefa que periodicamente vai buscar os dados a estas fontes e populará a base de dados com os mesmos. Para atingir este objetivo foi utilizada uma biblioteca do *Django* a "*django-crontab*". Esta biblioteca permite que sejam criados *cronjobs* na máquina que são executados com uma periodicidade defina. Para o caso deste demonstrador a recolha é realizada todos os dias à 01:00, ignorando o tempo de refrescamento de cada fonte, isto apenas para efeitos de demonstração do conceito. De forma a explicar de forma mais visual o funcionamento da tarefa de

recolha de dados externos a figura 6 representa um fluxograma do modo como a aplicação recolhe os dados externos.

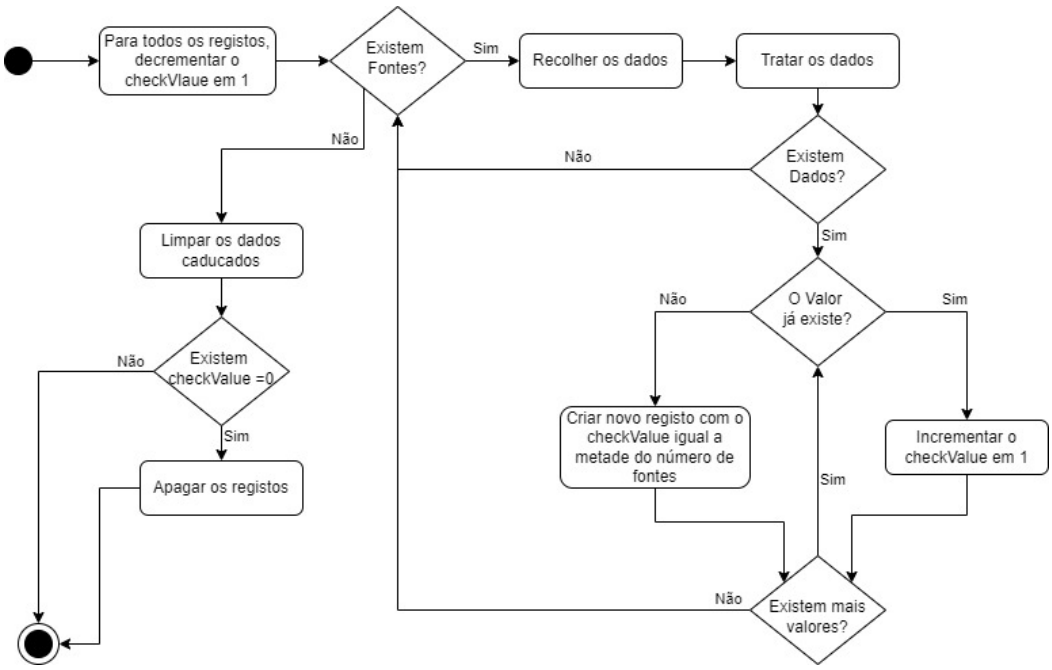


Figura 6: Fluxograma de funcionamento da tarefa de recolha de dados externos

Na figura 6 podemos observar que existem duas partes cruciais para esta tarefa. A primeira é responsável pela recolha e tratamento dos dados, onde tendo em conta o *checkValue* e o valor recolhido o sistema insere na base de dados ou atualiza o mesmo, dando-lhe ou atualizando a ponderação representada pelo *checkValue*. A segunda parte é responsável pela limpeza da base de dados, que mais uma vez tem em conta o *checkValue* para decidir se o valor deve ou não permanecer na base de dados. Este ultimo mecanismo é de extrema importância, pois garante que a existe uma base de dados limpa de registo irrelevantes e torna as pesquisas mais eficazes. Para contextualizar, a integração das duas fontes externas acima referidas geraou mais de 600.000 registos. A utilização de diversas fontes externas irá aumentar exponencialmente este número o que torna este mecanismo de limpeza, apesar de bastante simples, extremamente necessário para tentar atingir uma melhor eficiência.

5.1.4.2 *Recolha de dados internos* - rsyslog

A aplicação aqui apresentada tem várias formas de fazer a recolha e tratamento de dados internos da organização, tal como os já abordados upload de ficheiros (*.log* e

.pcap) ou captura de tráfego. Contudo, para além destas existe a possibilidade de fazer o tratamento de ficheiros logs recebidos na máquina através de *rsyslog*.

Para isso, foi necessário instalar este serviço na máquina onde a aplicação está alojada e configurar o mesmo para permitir receber. A instalação do serviço foi feita através do gestor de pacotes do sistema (*apt*) (Sharma, 2022) e foi alterado o ficheiro de configuração do serviço (*/etc/rsyslog.conf*) para ter as linhas representadas na listagem 14. Foram descomentadas as linhas referentes aos protocolos UDP e TCP bem como à utilização do porto 514, para permitir que este serviço esteja à escuta de ligações no mesmo. De seguida, foi definido um *template* que define o local e a forma como são guardadas as várias mensagens recebidas pelo *rsyslog*. Neste caso as mensagens serão guardadas num diretório com o seu endereço IP e num ficheiro com um nome discriminatório do serviço/programa que representam, dentro do diretório */var/log/remotelogs/*.

Listagem 14: Configuração do ficheiro /etc/rsyslog.conf para o servidor

```
# provides UDP syslog reception
module(load="imudp")
input(type="imudp" port="514")

# provides TCP syslog reception
module(load="imtcp")
input(type="imtcp" port="514")

$template RemInputLogs, "/var/log/remotelogs/%FROMHOST-IP%/%PROGRAMNAME%.log"
*.* ?RemInputLogs
& ~
```

Concluída a configuração do serviço, de forma a receber *logs* remotamente, resta apenas indicar nas máquinas cliente que devem redirecionar os seus *logs* para a máquina correta. Isto é conseguido colocando a linha representada pela listagem 15, em todas as máquinas que se pretende. Esta linha indica que devem ser enviados todo o tipo de *logs* para o endereço 192.168.87.102 no porto 514.

Listagem 15: Configuração do ficheiro /etc/rsyslog.conf para o cliente

```
*.* @192.168.87.102:514
```

Após garantir a correta configuração tanto do serviço como do envio dos *logs* para a máquina, foi necessário no *backend* desenvolver o mecanismo que permite ao utilizador tirar partido desta ferramenta. À semelhança da tarefa anterior, está será executada todos os dias à 01:00, caso o utilizador pretenda utilizar esta funcionalidade que por omissão está desativada. O funcionamento desta tarefa é

bastante simples e passa por correr todos os ficheiros de *log* localizados no diretório "*/var/log/remotelogs*", gerados automaticamente pelo serviço de *rsyslog* sempre que é recebida informação, e validar a existência tanto de endereços IP como de nomes de domínio que sejam correspondência para os que estão presentes na base de dados. No caso de ser detetada alguma correspondência entre o valor presente na base de dados e o valor encontrado no ficheiro de *log* a ser tratado, é despoletado um alerta novo. Após cada execução desta tarefa, todos os ficheiros tratados são eliminados garantindo desta forma que não são gerados alertas de eventos já analisados.

5.1.5 *Frontend*

A existência de uma interface gráfica nesta aplicação é um ponto chave para um dos objetivos que se tenta demonstrar que é a facilidade de utilização. Tal como já referido, o método escolhido para o tipo de aplicação, neste caso o *fronted* é o desenvolvimento de uma Single Page Application. Neste sentido, foi escolhido o *ReactJS* para implementar a interface gráfica da aplicação. A utilização desta biblioteca teve por base a facilidade e conhecimento de desenvolvimento já adquirido, sendo que esta biblioteca permite o rápido desenvolvimento e também escalabilidade da solução aqui apresentada. Como base do projeto do *frontend*, foi utilizado um *template* desenvolvido pelo Creative Tim (Tim, 2024) em *ReactJS* e *Material Tailwind*.

Nesta secção do relatório serão abordadas as quatro páginas que compõem o *frontend* (*Dashboard*, *Tools*, *Alerts* e *Feeds*), sendo feita uma breve explicação de qual o seu objetivo e que tipo de informação apresentam.

5.1.5.1 *Dashboard*

O *Dashboard* é uma página que apresenta um conjunto de estatísticas da aplicação, nomeadamente o número de alertas não vistos, o número total de alertas gerados, o número de fontes externas configuradas, o número total de dados recolhidos através destas fontes e por fim uma pequena listagem dos últimos alertas a serem gerados. A figura 7 mostra o estado atual do *Dashboard*, indicando os dados acima referidos.

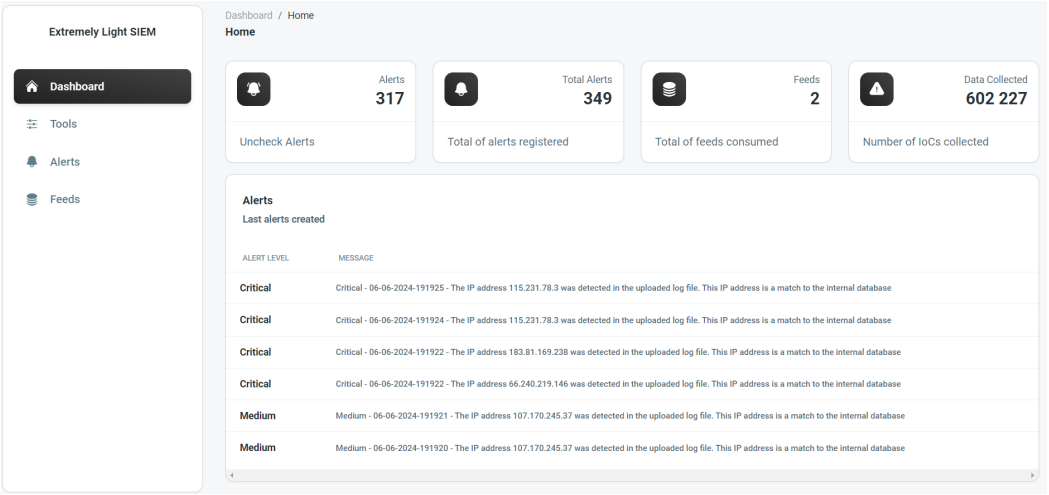


Figura 7: *Dashboard* da aplicação

5.1.5.2 *Tools*

A página referente às ferramentas da aplicação, aqui designada por *Tools*, é a página onde o utilizador pode interagir diretamente com as funcionalidades referidas anteriormente. A figura 8 representa esta página e o utilizador nela pode:

1. Ativar e desativar o *listener* - Para utilizar esta funcionalidade o utilizador, precisa de definir um porto onde a aplicação estará à escuta, que está no porto 32000 por omissão. Após definir o porto basta ativar o *switch* referente a esta funcionalidade e a aplicação recolhe o tráfego enviado, guardando num ficheiro para ser analisado após término desta ação;
2. Ativar e desativar o *sniffer* - A funcionalidade de *sniffing* necessita que seja definida a interface na qual a aplicação irá fazer a recolha de tráfego. É apresentada uma lista de todas as interfaces disponíveis na máquina onde a aplicação está alojada, das quais o utilizador deve escolher uma para poder ativar esta funcionalidade. Visto que esta é uma funcionalidade que utiliza um script externo que se liga diretamente à base de dados, não é possível diretamente perceber quantos alertas novos foram gerados, desta forma a aplicação não esclarece se foram ou não criados novos alertas;
3. Ativar ou desativar o rsyslog - Como referido na subsecção 5.1.4.2, esta é uma funcionalidade que é calendarizada, mas para que esta seja executada deve ser previamente ativada pelo utilizador.

4. *Upload* de ficheiros - Esta é a ultima funcionalidade presente na página das ferramentas. Esta funcionalidade permite ao utilizador fazer upload de ficheiros .log e .pcap para que possam ser analisados pela aplicação.

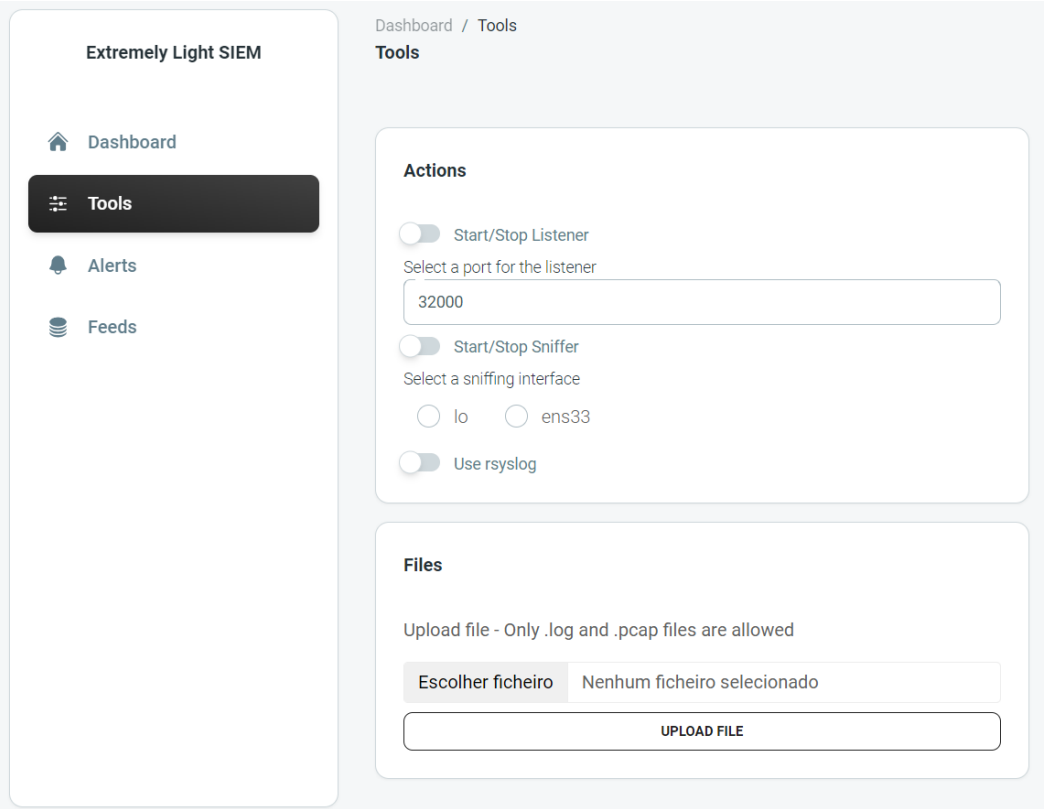


Figura 8: Página das ferramentas da aplicação

5.1.5.3 *Alerts*

Esta página, figura 9, apresenta uma listagem de todos os alertas que foram gerados, listados do mais recente para o mais antigo. É ainda possível ver a mensagem do alerta e marcar o alerta como visto.

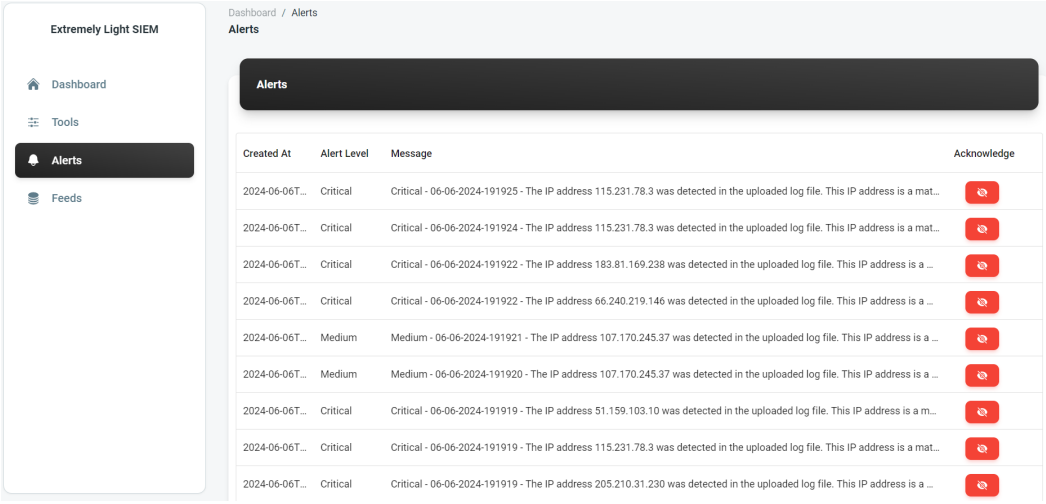


Figura 9: Página dos alertas

Na figura 10, é possível ver que a aplicação permite ao utilizador ver a mensagem completa do alerta gerado. Isto é possível clicando duas vezes em cima do alerta que se pretende obter mais informações.

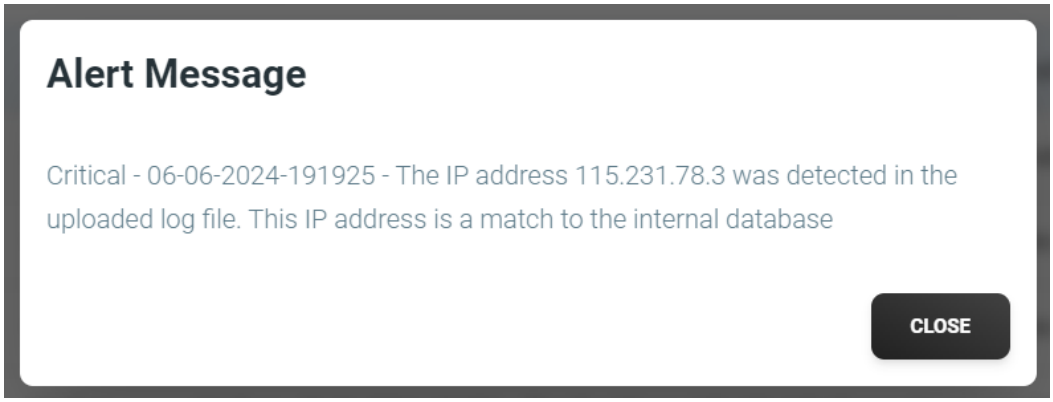


Figura 10: Detalhe dos alertas

A ultima funcionalidade desta página, aqui representada pela figura 11, mais concretamente pelo botão vermelho do lado direito do alerta, permite que o utilizador marque o alerta como visto. Ao fazer isto, tem a indicação visual que o alerta foi visto com o botão a trocar de cor para verde.

Figura 11: *Acknowledge* dos alertas

De referir que a listagem de alertas, está paginada e permite ao utilizador definir o número de alertas a mostrar por página.

5.1.5.4 *Feeds*

À semelhança da página dos alertas, a página referente às fontes, aqui representada pela figura 12, mostra uma listagem de todas as fontes externas configuradas na aplicação. Esta página, de momento não permite que o utilizador insira mais fontes. Tratando-se esta aplicação de um demonstrador de conceito, fez sentido inseri-los manualmente na base de dados e no futuro desenvolver mecanismos mais controlados para a inserção de fontes externas pelo utilizador.

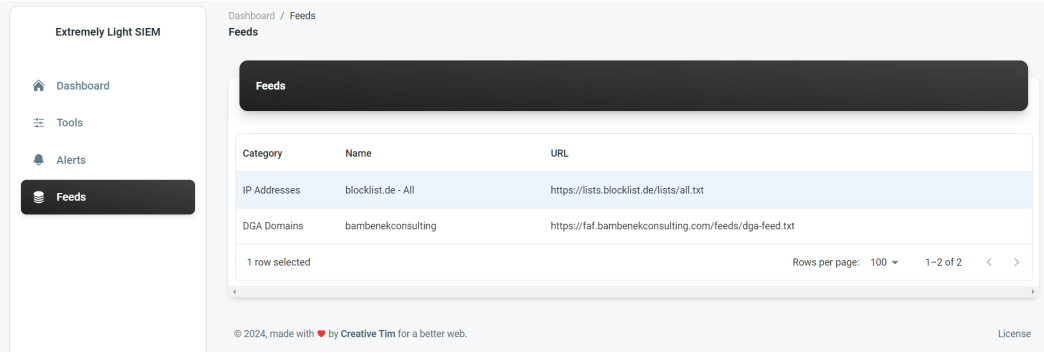


Figura 12: Listagem das fontes externas configuradas

5.1.6 *Síntese*

Neste secção, foi explicado o método de desenvolvimento da aplicação desde o *backend* ao *frontend* e passando pela base de dados. O *backend*, desenvolvido em *Python* com recurso às bibliotecas *Django* e *Django-rest-framework* é responsável pela recolha e tratamento dos dados externos e internos da organização. Utiliza ferramentas que permitem a deteção de endereços IP e nomes de domínio em ficheiros de *log* ou capturas de tráfego. Faz a correspondência a dados existentes na

base de dados para que possam ser gerados alertas. Existem tarefas calendarizadas que facilitam a integração das fontes de dados externas bem como a utilização de *rsyslog*.

Por outro lado temos o *frontend* que foi desenvolvido utilizando *ReactJS* e *Material Tailwind*. Faz a interligação entre o utilizador e a API desenvolvida no *backend* para que este possa usufruir das funcionalidades desenvolvidas. Tem quatro páginas com propósitos diferentes. A primeira é um *dashboard*, onde o utilizador pode obter estatísticas da aplicação, a segunda é uma página de todas a ferramentas disponibilizadas, a terceira página é onde o utilizador pode ver todos os alertas despoletados e interagir com os mesmos e por fim, a última página é onde o utilizador pode consultar todos as fontes externas configuradas.

5.2 CENÁRIO IMPLEMENTADO E TESTES

Esta secção tem como objetivo apresentar o ambiente no qual a aplicação foi desenvolvida e testada, indicando todo o esquema de máquinas virtuais e características das mesmas para que todo o cenário possa ser replicado. Serão também abordados e explicados todos os testes realizados à aplicação.

5.2.1 *Cenário implementado*

O cenário no qual a aplicação foi desenvolvida e testada é composto por 4 componentes principais, o portátil onde estão alojadas as máquinas virtuais, e três máquinas virtuais que servem propósitos diferentes. A figura 13 apresenta o cenário implementado para fins de desenvolvimento e testes da aplicação.

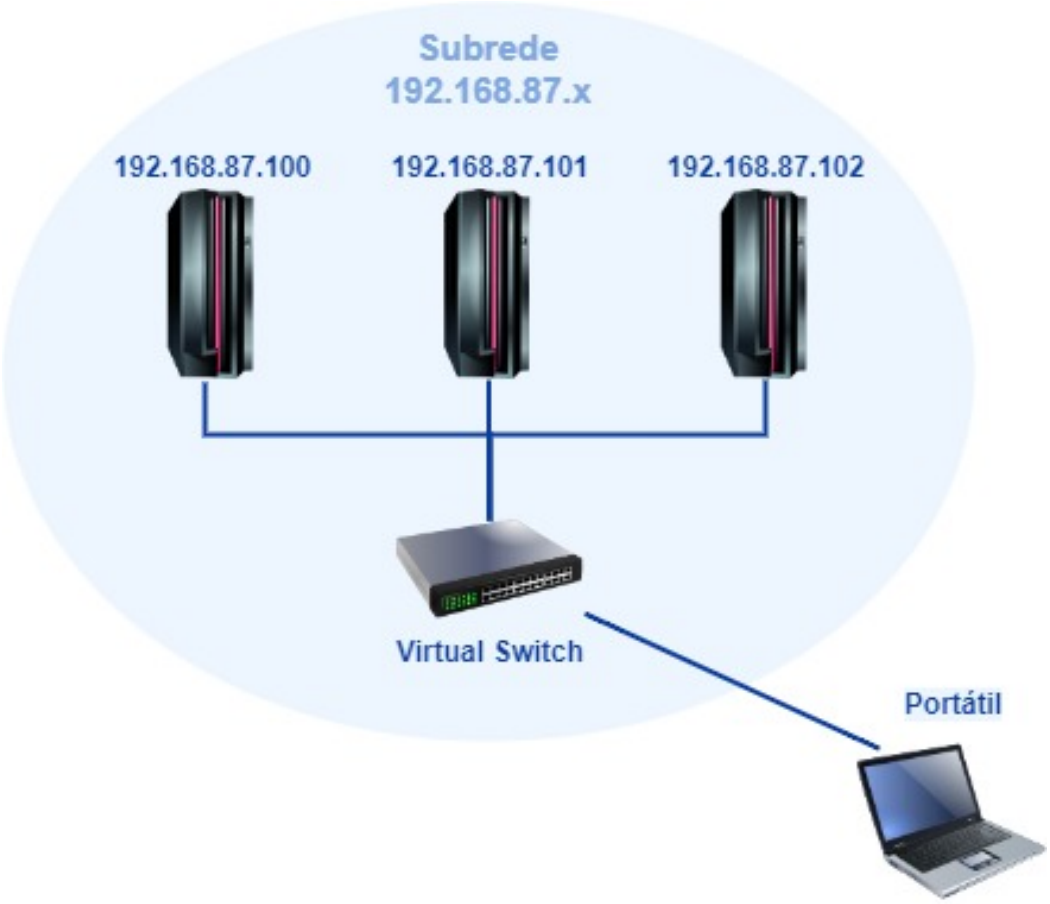


Figura 13: Cenário de desenvolvimento e testes

Aprofundando sobre o cenário representado (figura 13) é composto, como já referido, por 3 máquinas virtuais criadas através do *VmWare Workstation 16 Pro* versão 16.2.5 build-20904516. Todas estas máquinas estão a ser executadas num portátil. As máquinas aqui descritas tem as seguintes características:

1. Máquinas virtuais - Todas as máquinas virtuais tem as seguintes características:
    a) Sistema Operativo: Ubuntu 22.04.4 LTS (Jammy)
    b) Memória RAM: 4 GB
    c) Disco: 30 GB
    d) Processadores: 2
2. Portátil - A máquina *host* das máquinas virtuais tem as seguintes características:
    a) Sistema Operativo: Windows 11 Pro versão 23H2
    b) Memória RAM: 16 GB

c) Processador: 12th Gen Intel(R) Core(TM) i7-12700H 2.70 GHz

Todas as máquinas virtuais estão numa rede em Network Address Translation, permitindo que as máquinas estejam fechadas na sua rede, com acesso à Internet, podendo desta forma controlar o tráfego que passa na rede. Assim é possível ter uma rede sem interferências de todo o tráfego gerado pela máquina *host*.

A existência de três máquinas virtuais diferentes tem os seguintes objetivos. Primeiro, as máquinas representadas na figura 13 com os endereços IP 192.168.87.100 e 192.168.87.101 tem funções semelhantes e o intuito delas é realizar a captura de tráfego e gerar tráfego para que possa ser capturado. Foram criadas duas máquinas na tentativa de emular o máximo possível uma rede comum com vários *endpoints*, podendo estar uma a fazer apenas a captura e o envio para a aplicação e a outra a gerar o tráfego normal de uma rede. Por fim, a máquina representada pelo endereço IP 192.168.87.102 é a máquina onde toda a aplicação foi desenvolvida e se encontra alojada. Responsável por receber as recolhas de tráfego enviadas pelas outras máquinas e também por fazer as próprias capturas de tráfego, como foi descrito anteriormente como uma das funcionalidades.

5.2.2 *Testes e resultados*

Definido o cenário de desenvolvimento e testes, é importante também definir que tipo de testes serão feitos à aplicação e abordar os resultados obtidos de modo a ser possível aferir a viabilidade da aplicação que aqui se demonstra. Nesta fase, foram realizados três tipos de testes na aplicação. Em primeiro lugar e ao mesmo tempo da implementação, foram executados testes preliminares, que validaram o correto desenvolvimento das funcionalidades propostas, sendo que se tratam de testes muito breves e de validação da API. Em segundo lugar foram desenvolvidos testes de validação. O objetivo destes testes foi perceber se as funcionalidades estavam corretamente implementadas e que os objetivos da aplicação foram atingidos, isto é, se a gestão de alertas é funcional quando usadas as ferramentas. Em terceiro lugar foram realizados testes de carga, no sentido de tentar perceber os tempos de execução das tarefas, os tamanhos de ficheiros que são suportados bem como os recursos consumidos. Quanto aos testes de usabilidade, tratando-se de um demonstrador não foram executados, pois a aplicação apresenta as funcionalidades de uma forma que apenas seja apenas necessário validar a eficiência do conceito.

5.2.2.1 *Testes preliminares*

Como referido, estes testes foram realizados no decorrer do desenvolvimento da aplicação. O objetivo destes testes seria apenas de validação da API desenvolvida e da comunicação do *frontend* com a mesma. Não foram tiradas conclusões no decorrer destes testes, sendo que o objetivo foi apenas garantir que a API era funcional. Foram testados todos os *endpoints* representados na tabela 4 em primeiro lugar utilizando o *Postman*, uma aplicação que permite comunicar com a API desenvolvida e em segundo lugar utilizando o *frontend* desenvolvido, para deste modo garantir que este corresponde às expectativas.

No que diz respeito aos testes realizados na API, foram realizados com o objetivo de perceber se cada *endpoint* respeitava os tipos de dados, tabela 5 para cada variável, se era retornado algum erro indicativo e se o objetivo com cada pedido realizado era atingido. Quando se trata de um modelo a própria biblioteca do *Django* ajuda a que estes dados sejam tratados automaticamente, contudo foram validados da mesma forma. Apenas nos *endpoints* que não interagem diretamente com os modelos e que recebem parâmetros nos pedidos (*start-listener, start-sniffer* e *upload-file*) foram tratados os dados para garantir que recebem o tipo de dados corretos. Esta validação foi executada tanto no *backend* como no *frontend*.

Em primeiro lugar, validar o *input* do utilizador para o campo do porto. Como é possível ver pela figura 14, a API valida o porto e retorna uma mensagem de erro sugestiva, indicando o erro e uma possível solução.

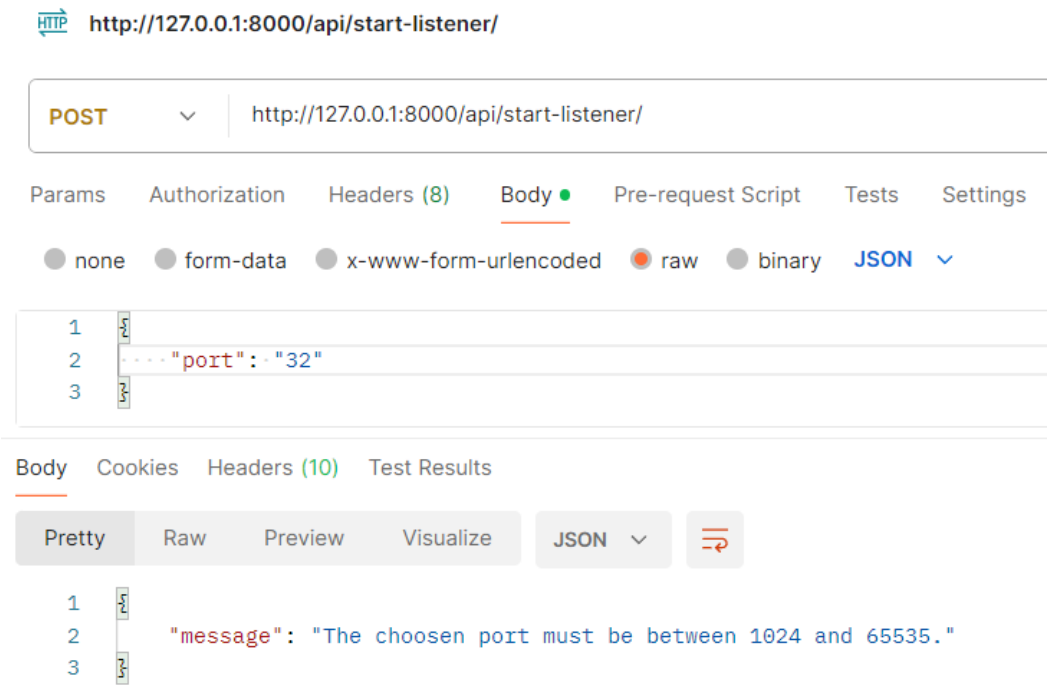


Figura 14: Introdução de um porto inválido no *endpoint start-listener/*

A introdução de um porto válido, ilustrado na figura 15, já permite ao *backend* iniciar o *listener* no porto selecionado.

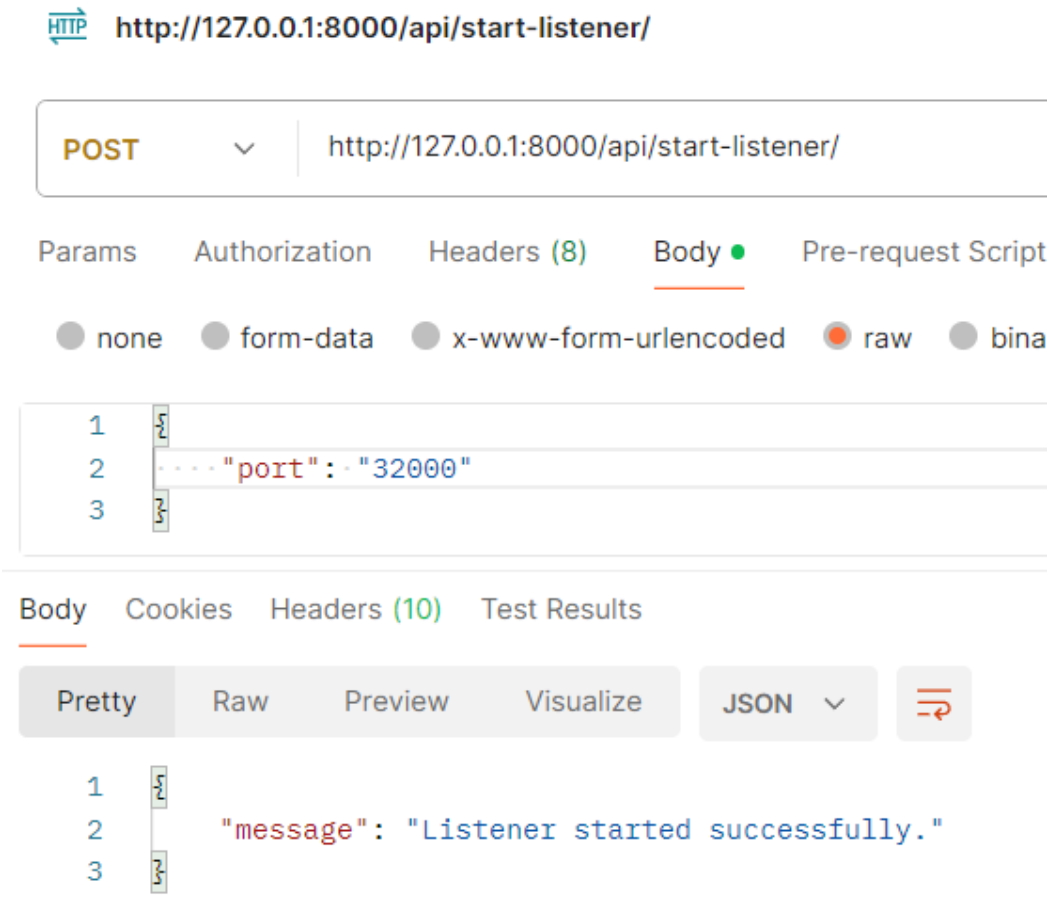


Figura 15: Introdução de um porto válido no *endpoint start-listener/*

Com estes testes foram detetadas algumas anomalias que foram corrigidas. No *endpoint* relativo ao "*start-listener/*", este recebia um inteiro como parâmetro que é convertido para string de modo que o comando possa ser executado corretamente. Este valor teve de ser tratado para garantir que o porto representa um valor válido, isto é: originalmente se o valor enviado fosse por exemplo "32000asd", o pedido era executado corretamente, contudo a execução do comando era terminada com erro. Após corrigida esta questão é garantido que apenas sejam enviados inteiros no pedido, este erro já não acontece. A correção é possível ver na figura 16.

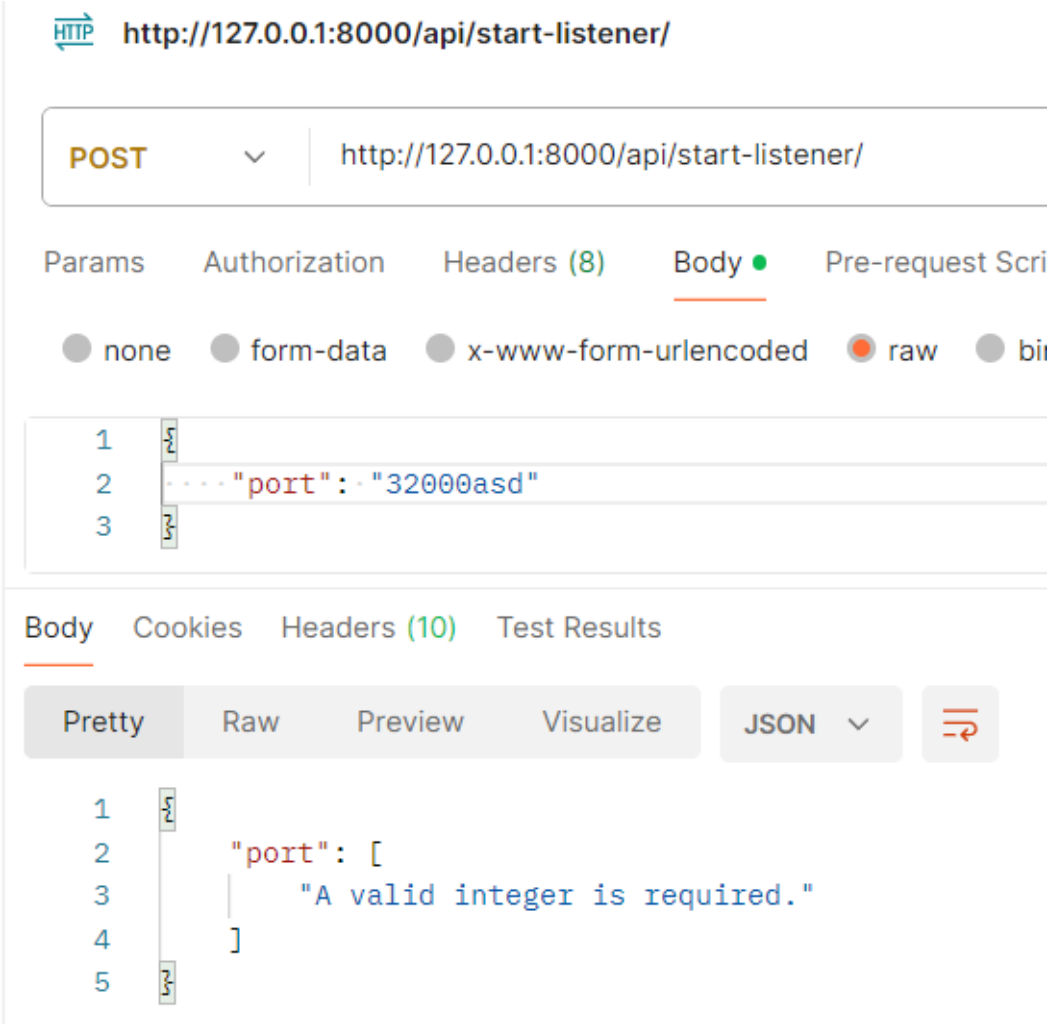


Figura 16: Validação do tipo de dados enviado no *endpoint start-listener/*

O *endpoint* que ativa o *sniffer* (*start-sniffer/*), apresentava um erro relativo à seleção da interface. No *frontend*, o utilizador é obrigado a selecionar uma das interfaces existentes na máquina. Contudo, o *backend* não apresentava validação para isto, permitindo que o *script* de captura de tráfego fosse executado na mesma, no entanto era terminado com erro, e o utilizador teria *feedback* de que teria executado corretamente. Posto isto foi acrescentado um mecanismo que valida-se a interface selecionada, garantindo que é uma existente na máquina. A figura 17, mostra que a validação foi acrescentada, e é então possível validar a existência da interface.

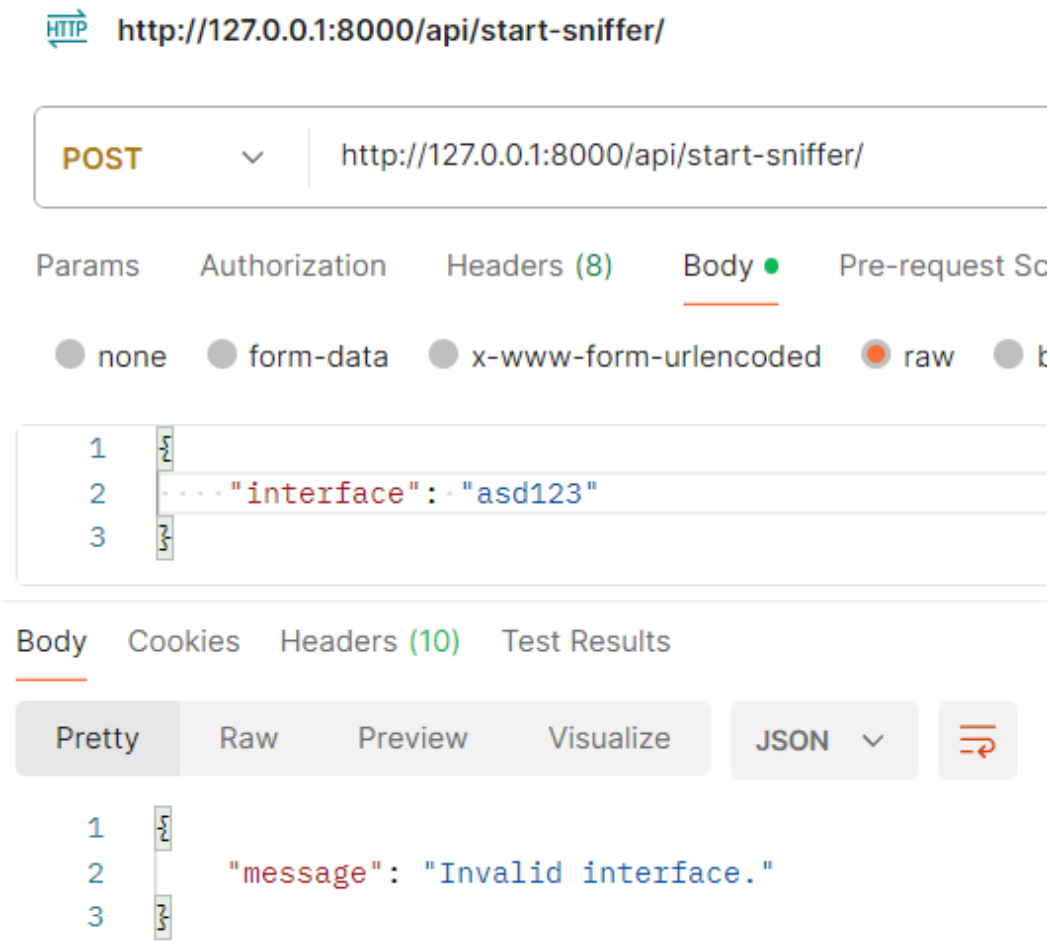


Figura 17: Introdução de uma interface de rede inválida no *endpoint start-sniffer/*

De modo a permitir que o *sniffer* inicie, é necessário introduzir uma das interfaces disponíveis no *frontend*, deste forma já será possível capturar pacotes na máquina. Após a introdução de uma interface válida a figura 18 mostra o *output* esperado nesta situação, que é a inicialização do *sniffer*.

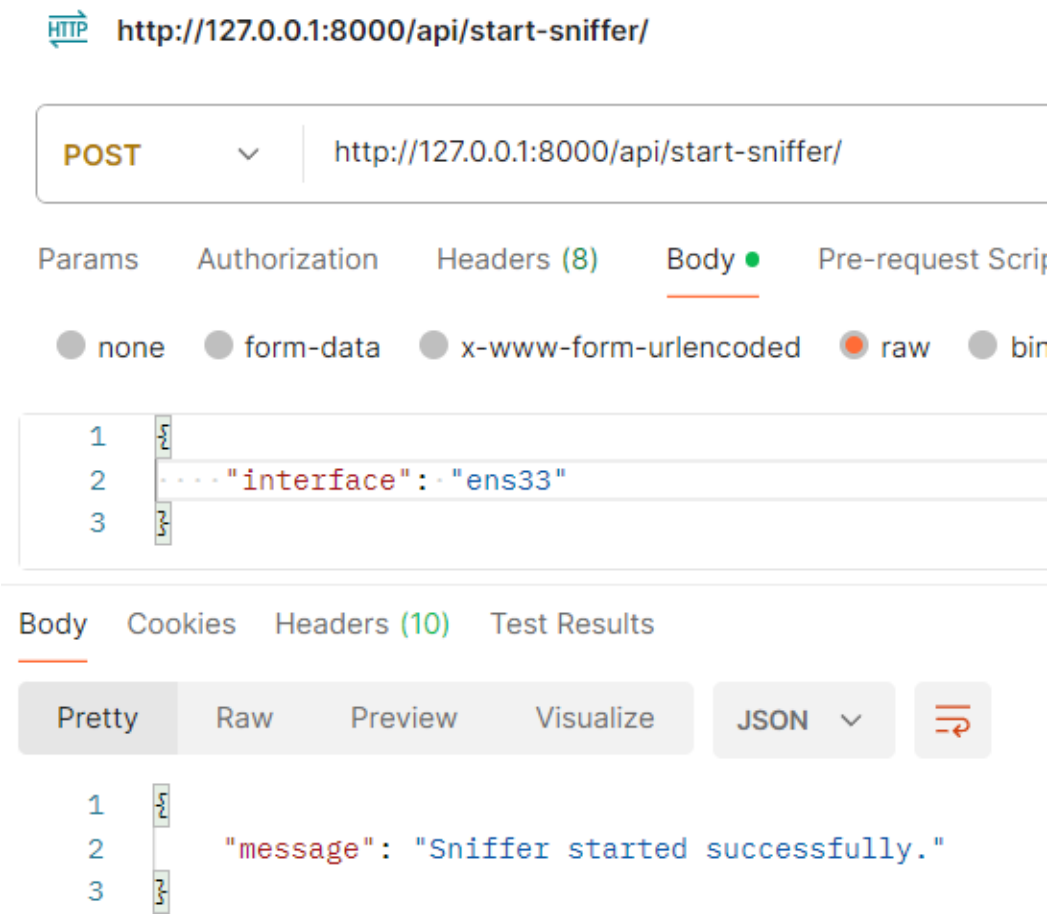


Figura 18: Introdução de uma interface de rede válida no *endpoint start-sniffer/*

Nos testes feitos aos *endpoints* que fazem as funções de CRUD dos modelos, foi detetado que a API trata da mesma forma *strings*, inteiros ou até booleanos que representem o mesmo, isto é, por exemplo, a *string* "0"pode representar *False* quando se trata de um campo booleano ou pode representar 0 quando o campo é um inteiro. Apenas nestes casos, em que as *strings* representem outro valor válido a API não retorna um erro mas o objetivo final é atingido. Como podemos ver na imagem 19, o *endpoint* que tem como objetivo guardar a interações com a aplicação e o estado em que se encontram as funcionalidades (*helper/1/*), é possível ver que ao introduzir um zero (0) ao invés de um booleano (*false*) no campo *is_listener_running*, a aplicação funciona corretamente.

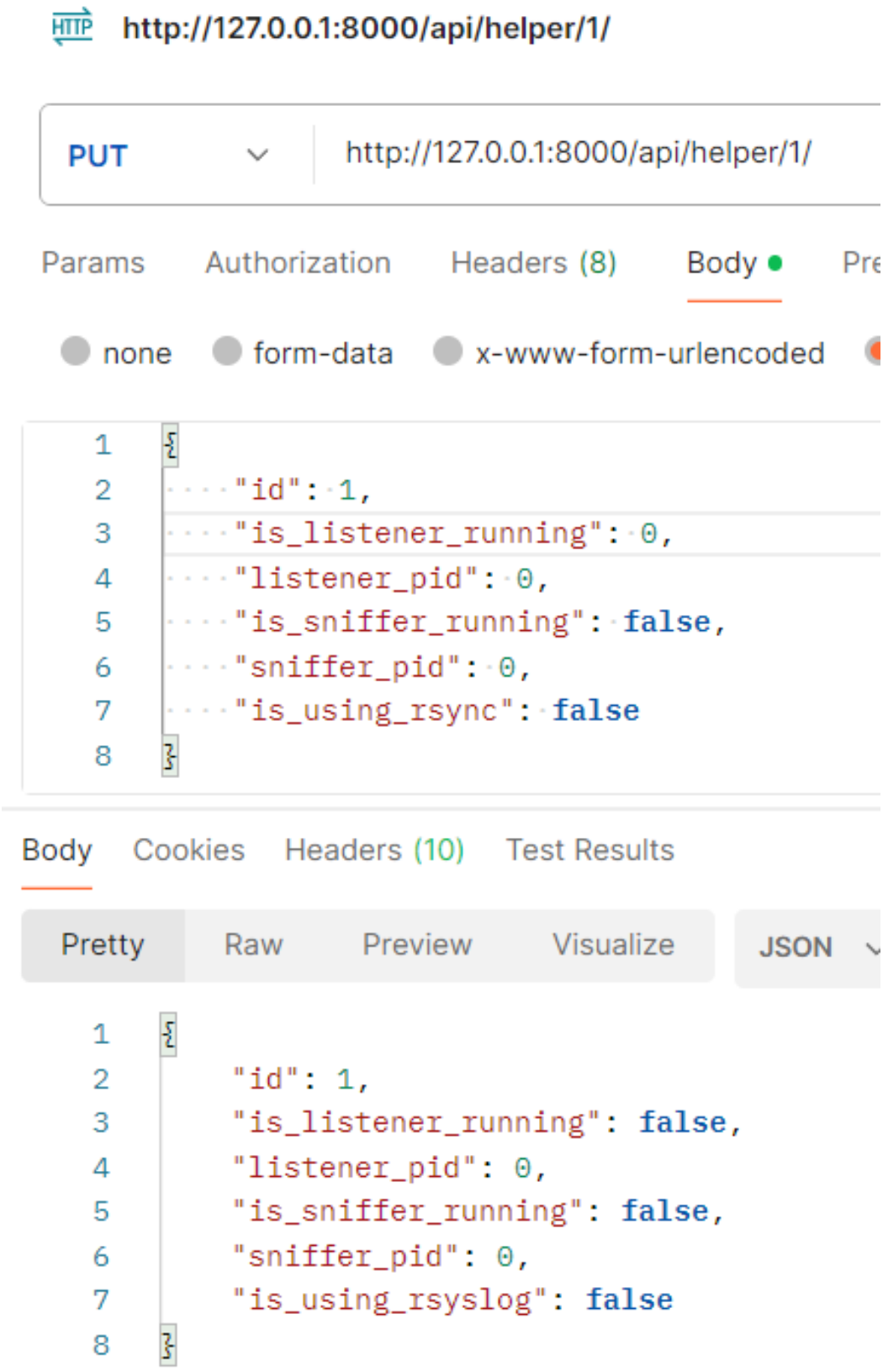


Figura 19: Introdução de inteiro ao invés de um booleano

No caso contrário, ao colocarmos um *false*, no campo destinado ao PID do *listener*, a aplicação retorna um erro, figura 20.

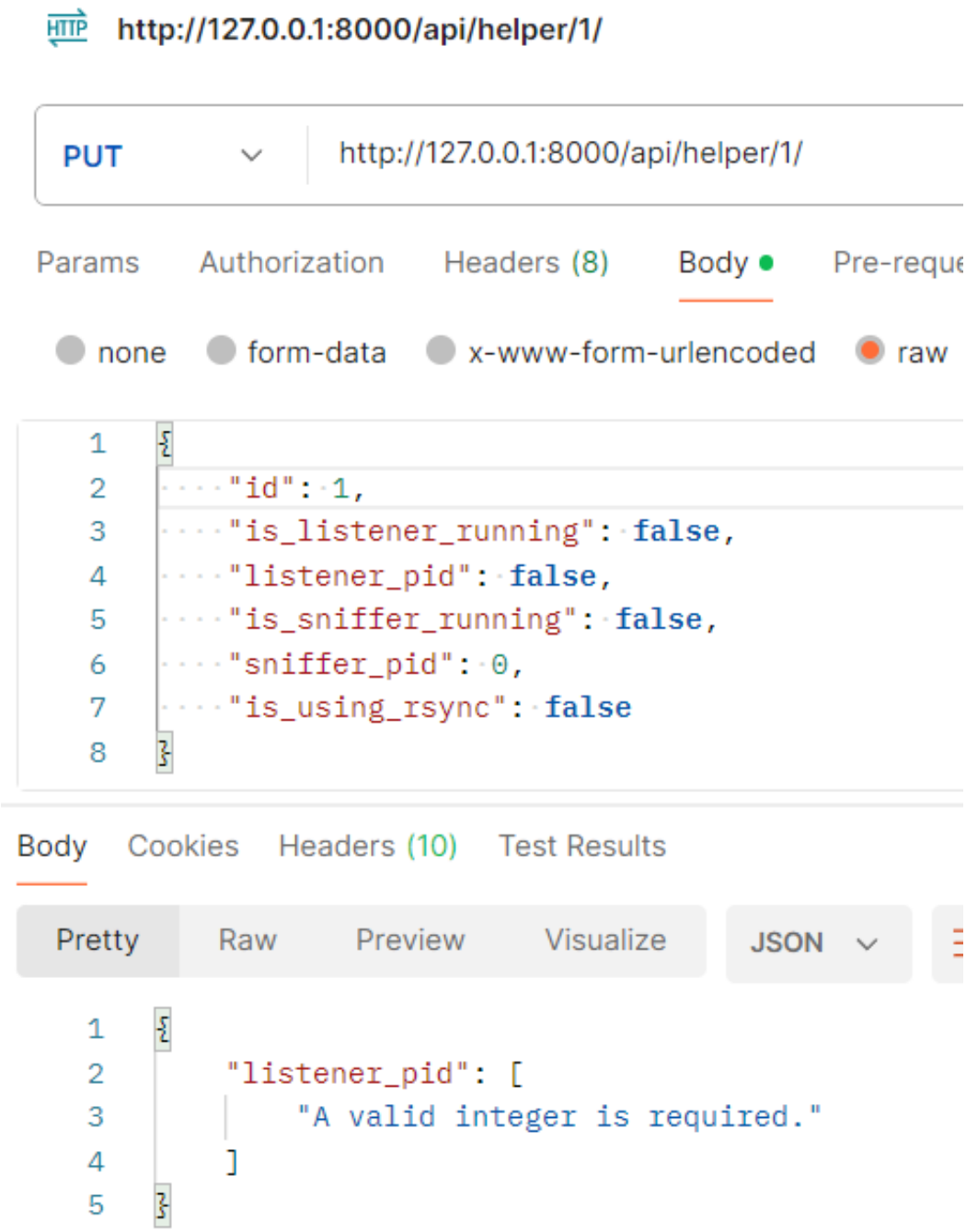


Figura 20: Introdução de booleano ao invés de um inteiro

Por fim, relativamente ao *endpoint upload-file*, foram testados os mecanismos que garantem que apenas é permitido o *upload* de ficheiros *.log* e *.pcap*, garantindo desta forma que a API se encontra funcional. A figura 21 mostra o *output* gerado caso seja introduzido um ficheiro com uma extensão inválida, como é o caso de ficheiros

PDF. É importante referir novamente que esta validação tem em conta apenas a extensão do ficheiro.

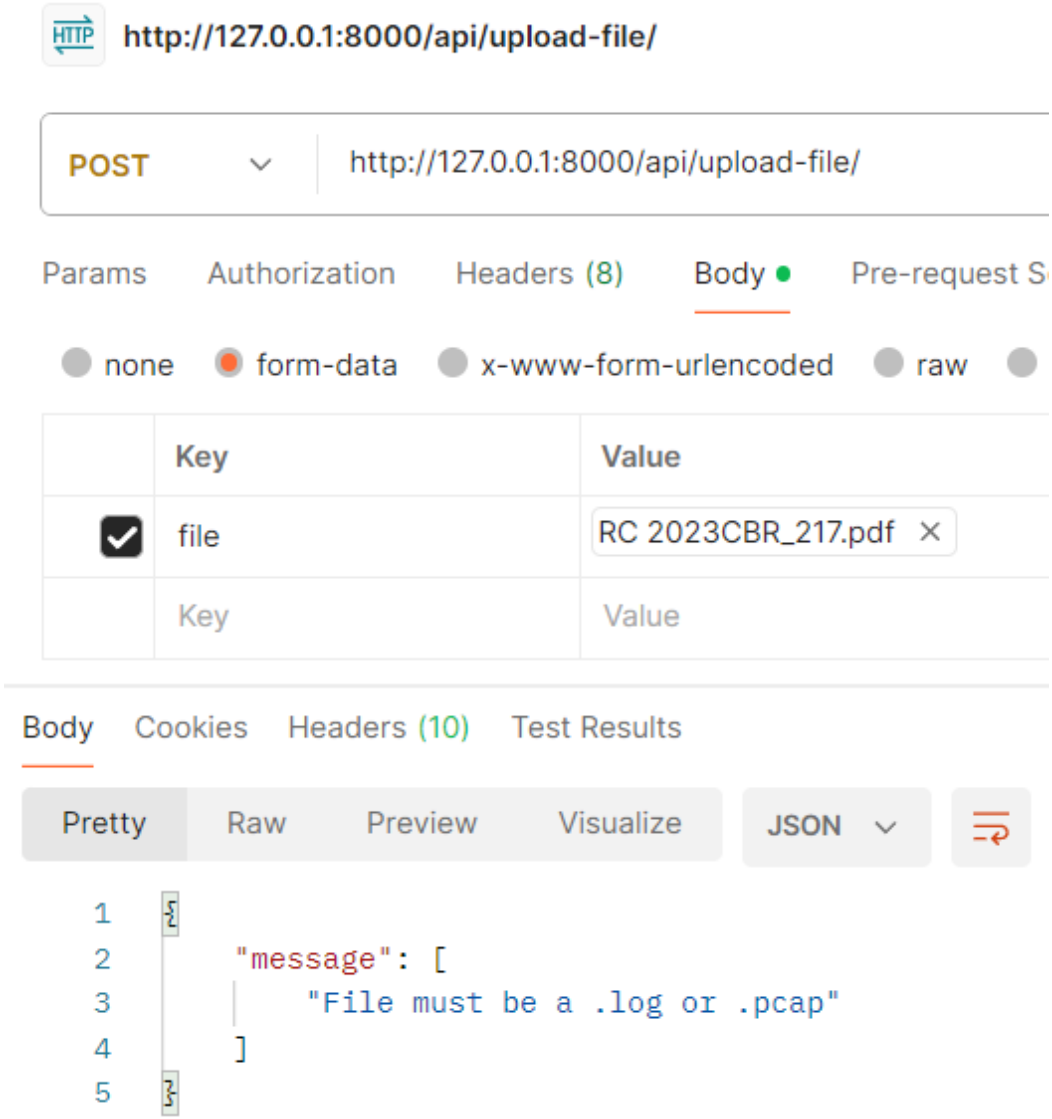


Figura 21: Upload de ficheiro com extensão inválida

Dados os testes realizados, garantiu-se que foram corrigidos os erros encontrados e que todos os *endpoints*, sejam eles testados através do *Postman* ou através o *frontend* desenvolvido, obtêm o objetivo para o qual foram desenvolvidos.

5.2.2.2 *Testes de validação*

O objetivo dos testes de validação, para além de obrigatoriamente validarem o desenvolvimento da API, validam principalmente o correto funcionamento do pro-

cessamento de dados e gestão de alertas. Os testes realizados para validação focam-se no seguinte:

1. Garantir que o mecanismo desenvolvido para obter os dados das fontes externas funciona corretamente;
2. Garantir que o processamento de dados das fontes internas, independentemente do tipo de fontes interna escolhida, é recebido e cruzado com o recolhido das fontes externas e através destes garantir que os alertas são corretamente despoletados.

De modo a testar estas funcionalidades, e tendo em conta as fontes de dados externas configuradas na aplicação, o mecanismo de recolha de dados externos foi testado várias vezes garantindo que os valores são inseridos na base de dados com o *checkValue* correto, bem como os restantes dados associados a cada valor. Quanto ao processamento dos dados internos e cruzamento com os dados externos, no que diz respeito aos dados de capturas locais e remotas de tráfego, foram utilizadas técnicas para forçar a tradução de nomes (presentes na base de dados) para testar os mecanismos de gestão de alertas. No que diz respeito aos ficheiros carregados na aplicação, nomeadamente ficheiros *log*, numa fase inicial foram alterados para conter informação e numa segunda fase foram utilizados ficheiros recolhidos numa *firewall* em produção.

Elaborando um pouco mais sobre os testes, e começando pelo consumo das fontes externas. Esta funcionalidade, como já referida é executada com base numa tarefa calendarizada (*cronjob*). Para os testes, a periodicidade de execução foi alterada em primeiro lugar para 10 minutos, o que rapidamente se mostrou ser demasiado rápido, pois a primeira execução excedeu esse tempo. No segundo teste, foi necessário configurar a tarefa para ser executada a cada hora. Este já foi um espaço temporal mais indicado visto que a primeira execução excedeu os 40 minutos para duas fontes externas configuradas, registando mais de 600.000 registos. Posto isto, e tal como já indicado, esta tarefa foi alterada para ser executada pelo menos 1 vez por dia, garantindo que os dados consumidos são recentes e que com o crescimento do número de fontes configuradas exista tempo suficiente para consumir todas as fontes. No que é possível validar em termos de processamento de dados das fontes externas, os dados são corretamente preenchidos na base de dados. Como é possível ver na figura 22, estão na base de dados 667.492 registos, o que prova que o processamento de fontes externas é funcional.

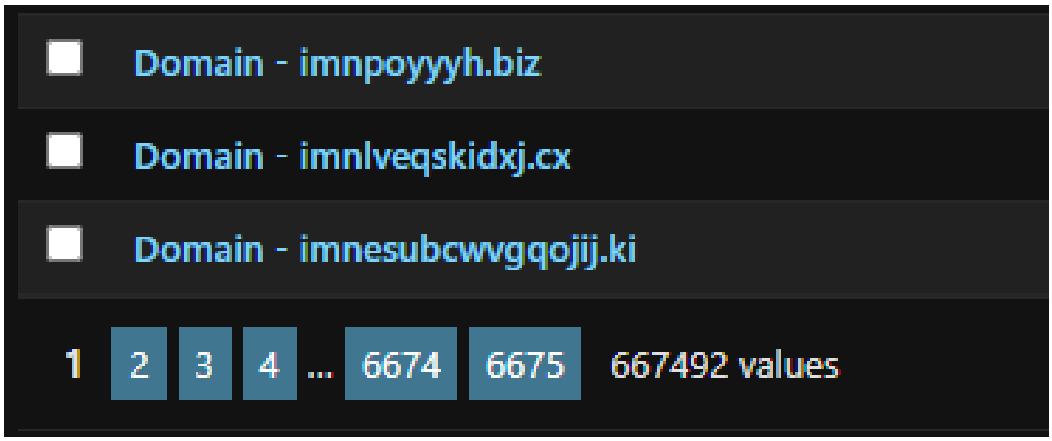


Figura 22: Número de registos na base de dados

Quanto ao segundo teste realizado, que se refere mais especificamente ao processamento de dados internos e cruzamento de dados com as fontes externas para emissão de alertas, foram realizados os seguintes testes:

1. Garantir que o envio de tráfego através da funcionalidade do *listener* é executada corretamente;
2. Garantir que a captura de tráfego através da funcionalidade de *sniffing* é executada corretamente;
3. Garantir que o processamento de dados recebidos através de ficheiros é funcional;
4. Garantir que o processamento de *logs* através do serviço de *rsyslog* é funcional.
5. Garantir que para cada um dos pontos anteriores é possível gerar alertas.

De modo a realizar os testes, foram colocadas em prática os métodos utilizados como prova de conceito descritos na secção 4.3. No que diz respeito ao teste do *listener*, foram todos os seguintes passos:

1. Na base de dados, escolher dos dados recolhidos de fontes externas alguns indicadores para forçar a tradução de nomes;
2. Fazer uma chamada ao *endpoint start-listener* para ativar esta funcionalidade num determinado porto através do *frontend*;
3. Numa das máquinas do cenário acima representado, realizar uma escuta ao tráfego da rede e enviar o mesmo para a máquina da aplicação e porto definido;
4. Terminar o envio de tráfego e terminar também o *listener* na aplicação através do *frontend*

Da base de dados foram escolhidos os seguintes nomes de domino: "**wmail-chat.xyz**"e "**wmail-schnellvpn.xyz**"para fazer os teste de tradução de nomes. Foi iniciado o *listener* através do *frontend* para o porto 18000. O *frontend* tem uma proteção que não permite que o utilizador selecione portas abaixo de 1024 dado serem as que normalmente são utilizadas pelos serviços mais comuns. A figura 23 é a representação no *frontend* da configura do *listener* para o porto 18000 e consequente ativação.

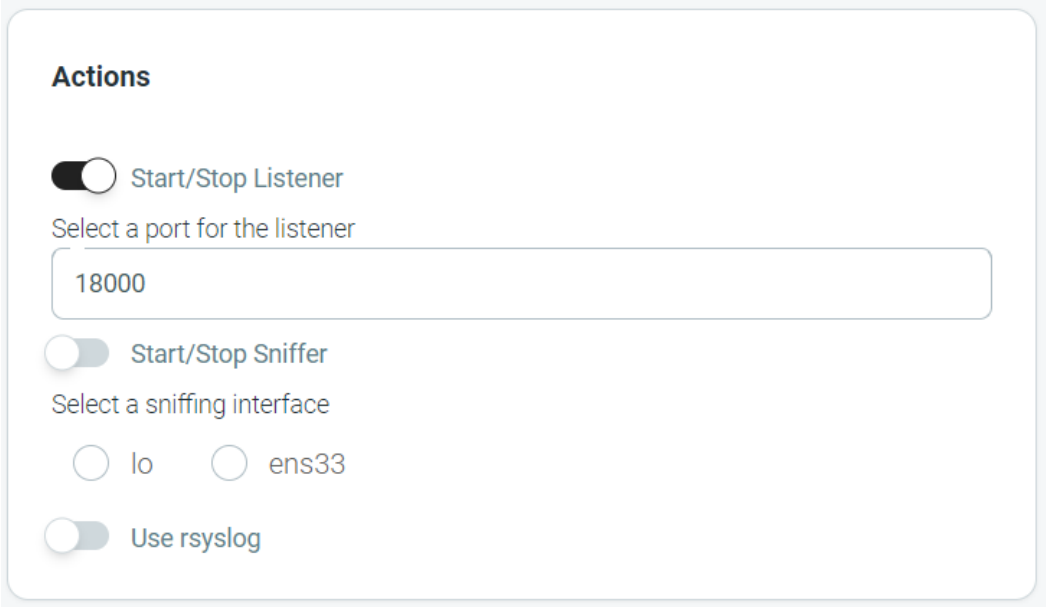


Figura 23: Iniciação do *listener*

De seguida, com recurso ao comando "**sudo tshark -w - -i ens33 -n src port 53 | nc 192.168.87.102 18000**"foi iniciada uma recolha de tráfego DNS (comando explicado anteriormente na secção 4.3) e enviado o tráfego para o porto 18000 da máquina onde está a ser executada a aplicação, como é possível validar na figura 24.

```
martinho@ubuntu1:~$ sudo tshark -w - -i ens33 -n src port 53 | nc 192.168.87.102 18000
[sudo] password for martinho:
Running as user "root" and group "root". This could be dangerous.
Capturing on 'ens33'
 ** (tshark:93610) 15:40:38.126721 [Main MESSAGE] -- Capture started.
 ** (tshark:93610) 15:40:38.127242 [Main MESSAGE] -- File: "-"
35 ^Ctshark: The file to which the capture was being saved
("-") could not be closed: Broken pipe.
Please report this to the Wireshark developers as a bug.
https://gitlab.com/wireshark/wireshark/issues
(This is not a crash; please do not say, in your report, that it is a crash.)
```

Figura 24: Captura e envio de tráfego

Após todo o cenário estar a ser executado resta realizar traduções com os nomes de domínio selecionados, como pode ser validado na figura 25.

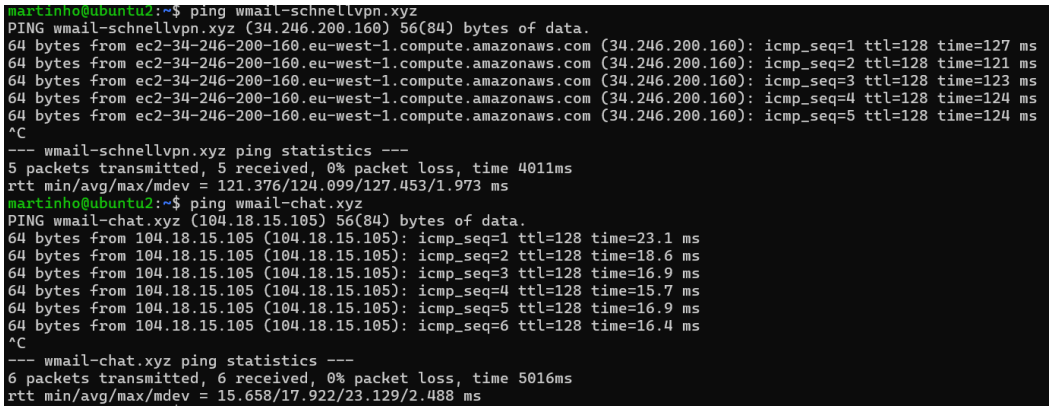


Figura 25: Forçar a tradução de nomes

Por fim, bastou terminar o *listener* e validar se foram criados alertas. Como se pode observar na figura 26 foram gerados três alertas com as traduções executadas. Apesar de apenas estarem presentes na base de dados os dois domínios indicados, foram gerados três alertas, pois a forma como é feita a validação dos pacotes de DNS no backend, pode levar a que um domínio gere dois alertas devido ao DNS Pointer Record. O código referente ao tratamento de pacotes DNS encontra-se no ficheiro */SIEM-Light/backend/siemapi/api/utils.py*

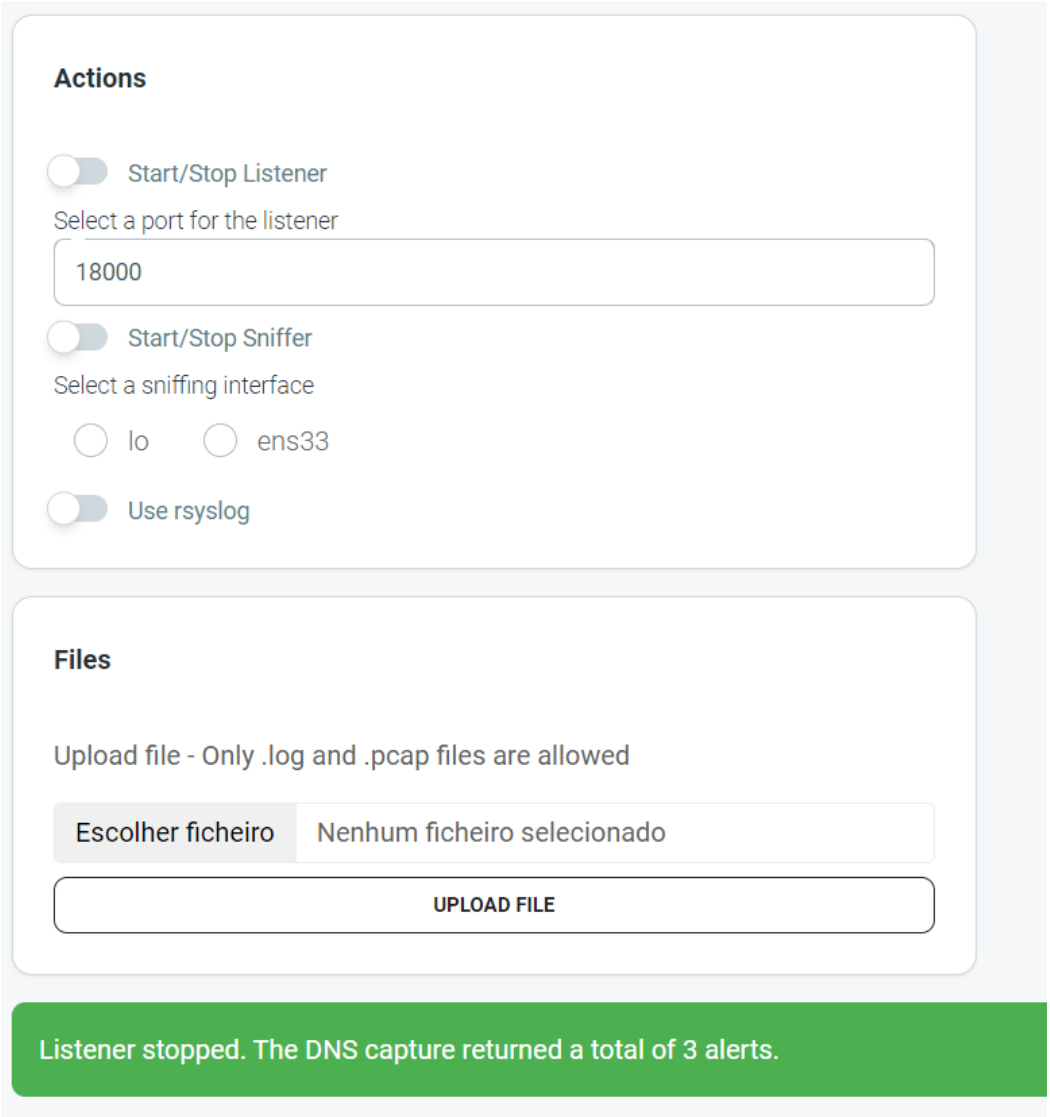


Figura 26: Término do *listener* e alertas criados

Este teste foi executado múltiplas vezes alternando a ordem dos passos acima descritos obtendo as seguintes conclusões:

1. Iniciar a recolha e envio dos dados primeiro - Ao realizar este passo em primeiro lugar o comando termina diretamente pois o porto não está a escuta do outro lado e a ligação não é executada;

2. Terminar o envio na consola e só depois terminar o *listener* - Este ponto poderia causar algum problema, dada a forma que o *backend* termina o processamento, que é através do PID do processo. Ao terminarmos a ligação no nó onde está a ser feita a recolha e envio dos dados antes de terminarmos o *listener* no *frontend*, poderia ser fechada a ligação e o processo responsável pelo *listener* já não existir, podendo não fazer o parse do ficheiro *.pcap* gerado

e dar erro a terminar o processo. Posto isto o mesmo problema não se valida e os alertas são gerados normalmente.

3. A tradução de nomes só pode ser realizada após o *listener* e a captura/envio de dados estarem a funcionar - Isto é normal, pois caso a tradução ocorra antes de estarem os mecanismos a funcionar não é possível detetar a tradução de nomes e consequentemente gerar alertas.

De realçar que os testes foram feitos com nomes de domínio já na base de dados para forçar os alertas. Num funcionamento corrente da aplicação este passo não é necessário. Com os testes realizados validamos que a funcionalidade do *listener* funciona como esperado.

O teste relativo à funcionalidade do *sniffer* é muito idêntico ao teste do *listener*. A grande alteração é que o envio dos dados já não é necessário pois a captura do tráfego é feita diretamente na máquina onde a aplicação esta a ser executada. Desta forma, para testar esta funcionalidade e garantir que tem o comportamento esperado, foram tomados os seguintes passos:

1. Escolher nomes de domínio presentes na base de dados para forçar a tradução de nomes;
2. No *frontend* ativar o *sniffer*, escolhendo a interface desejada para a captura do tráfego;
3. Nas restantes máquinas da rede, utilizando os nomes de domínio escolhidos, realizar a tradução de nomes;
4. No *frontend* terminar o *sniffer* e validar a criação de alertas.

Como anteriormente, os testes foram realizados com base nos seguintes endereços "**wmail-chat.xyz**"e "**wmail-schnellvpn.xyz**". Com os nomes de domínio escolhidos foi necessário ativar o *sniffer* no *frontend*, como é possível ver na figura 27. A aplicação não deixa continuar caso o utilizador não selecione uma das interfaces disponíveis. Este mecanismo de validação de dados introduzidos está validada tanto no *frontend* como no *backend*.

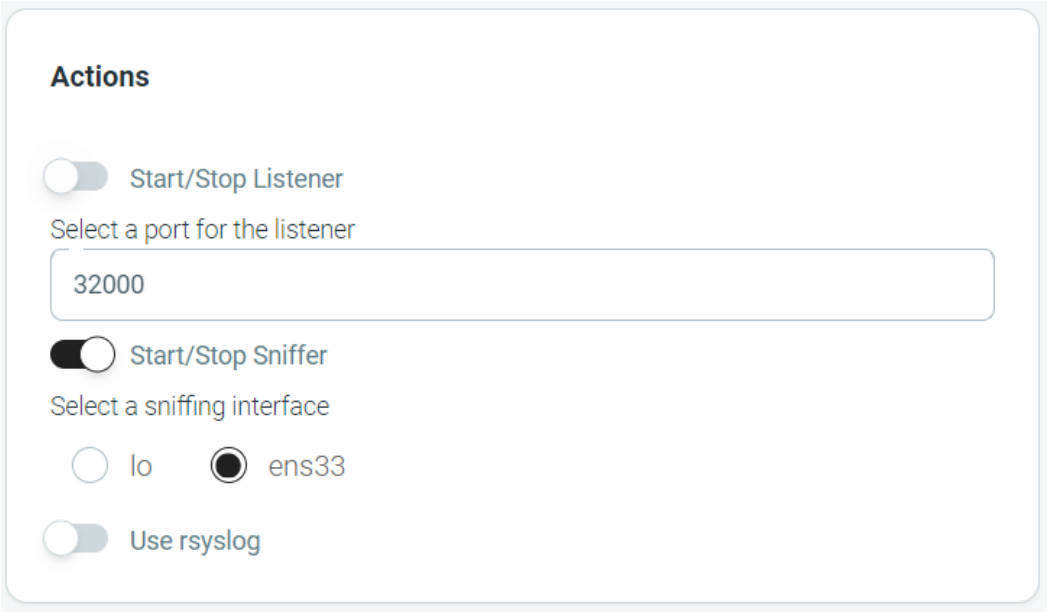


Figura 27: Ativar o *sniffer* com a interface selecionada

Após ativar o *sniffer* e forçada a traduções de nomes, a aplicação apesar de não dar nenhum erro, validou-se que não estavam a ser criados alertas como era suposto. Após validado o código, foi detetado que no *script* a criação de alertas diretamente na base de dados estava a gerar um erro ao inserir os alertas na mesma, pois faltava a coluna "created_at"estar preenchida. Corrigido este erro, é agora possível gerar alertas com a funcionalidade de *sniffing*. De realçar, que a aplicação não dá *feedback* de quantos alertas foram criados, e tem de se validar na página referente aos alertas.

Após garantir que foram gerados os alertas, conclui-se que, feitas as correções necessárias, esta funcionalidade tem o comportamento esperado, isto é, captura tráfego, valida o mesmo e gera alertas caso encontre alguma indicação para alertar.

O próximo teste valida o correto funcionamento da função de *upload* de ficheiros que tem como objetivo, garantir apenas o *upload* de ficheiros *.log* e *.pcap*(também ja validado no *backend*), processar os mesmos e gerar alertas. Os testes realizados nesta fase foram:

1. Garantir que apenas é possível fazer *upload* de ficheiros em *.log* e *.pcap*
2. Fazer upload de ficheiros *.log* e *.pcap* e validar se estas geram alertas.

O primeiro passo foi tentar fazer o *upload* de ficheiros de tipos diferentes do esperado. Neste caso foram testados com ficheiros com as extensões seguintes *.txt*, *.doc* e *.pdf*, obtendo sempre a mesma mensagem de erro, como é possível ver na figura 28.

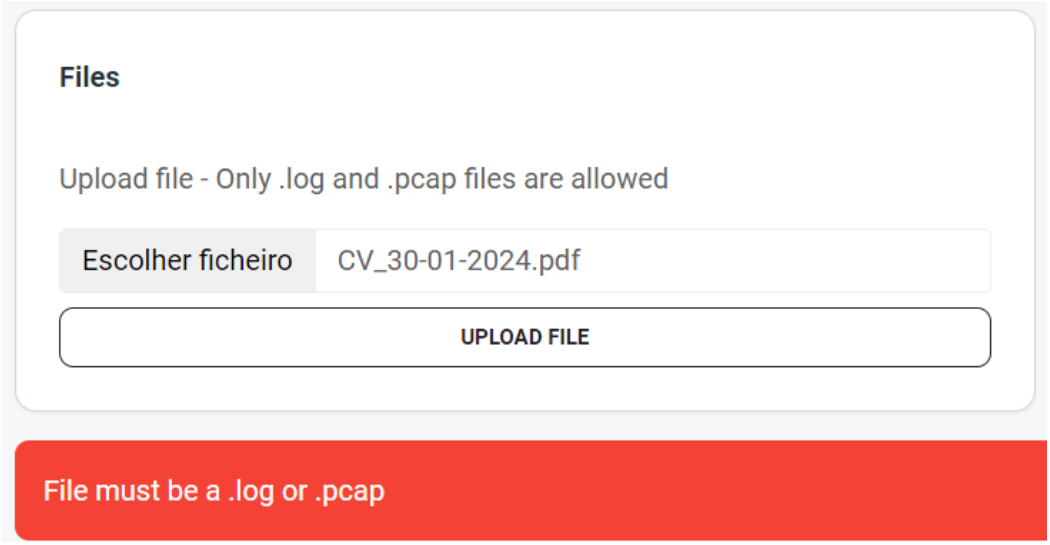


Figura 28: Erro de *upload* de ficheiro com extensão inválida

No que diz respeito ao processamento dos ficheiros carregados na aplicação, foi criada uma captura com a tradução de nomes para os dominios já referidos ("**wmail-chat.xyz**"e "**wmail-schnellvpn.xyz**") e foi utilizado um ficheiro de *log* de uma *firewall* em produção. Para o primeiro caso, o *output* gerado é representado pela figura 29. Como é possível ver, foram criados os 3 alertas esperados, como nos testes anteriores.

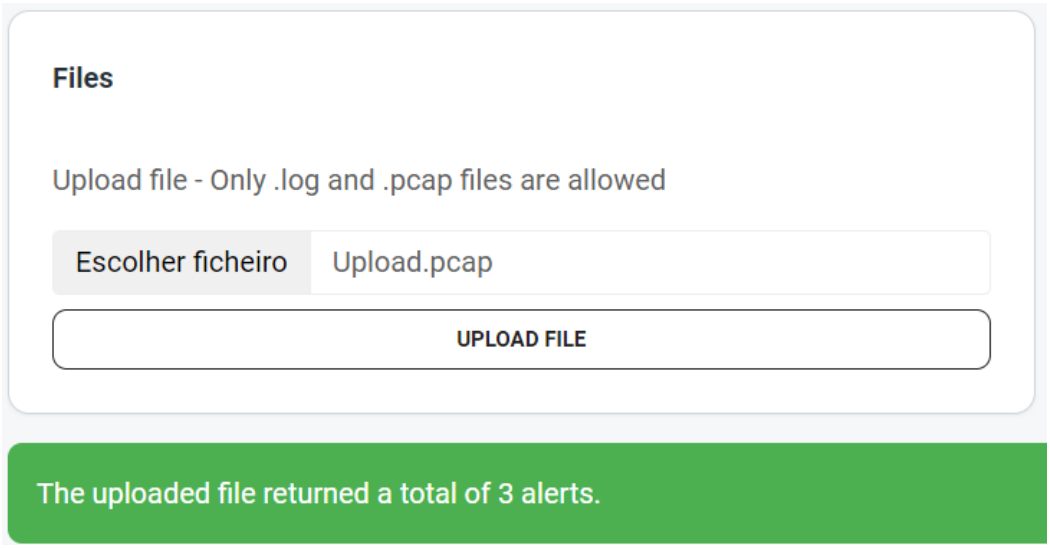


Figura 29: *Upload* de um ficheiro .pcap

Para o segundo caso, o *upload* de um ficheiro log, a figura 30 representa o *output* do ficheiro *log* acima referido. Como é possível constatar, este ficheiro gerou um

número bastante superior de alertas. Podemos ver que no *output* do ficheiro *log* foram gerados 55 alertas novos.

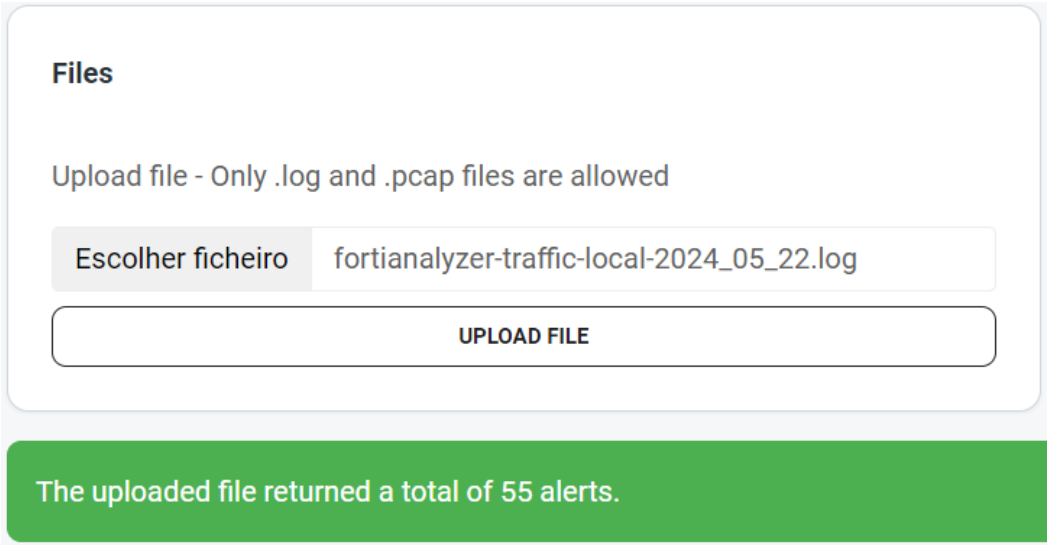


Figura 30: *Upload* de um ficheiro .log

Com estes teste, podemos concluir que a funcionalidade de *upload* de ficheiros esta corretamente implementada, e é possível obter dados bastante relevantes, como podemos constatar pela utilização de ficheiros *log* de uma *firewall* em produção.

Passamos ao ultimo teste de validação de funcionalidade que se foca na tarefa já abordada de sincronização de ficheiros *log* através do serviço de *rsyslog*. Para além de validar se o processamento dos ficheiros funciona como esperado, é importante também validar se é possível ativar e desativar a funcionalidade no *frontend* e se é possível gerar alertas. Para realizar estes testes foram executados os seguintes passos:

1. Validar que existem ficheiros *log* no diretório destinado a *logs* do rsyslog;
2. Num dos ficheiros *log* introduzir um endereço de IP existente na base de dados, para forçar a criação de um alerta;
3. Para efeitos de teste, garantir que a tarefa é executada de 10 em 10 minutos, consequentemente é necessário alterar o *cronjob*;
4. No *frontend* validar que a função de *rsyslog* está desativada;
5. Garantir que a tarefa não é executada;
6. Ativar a funcionalidade de *rsyslog* no *frontend* e garantir que a tarefa é executada;
7. Validar a criação do alerta;

8. Garantir que os ficheiros *log* são eliminados após validação;

Após configuração do serviço *rsyslog*, subsecção 5.1.4.2, foi garantida a correta configuração do serviço e que todas as máquinas configuradas para enviar *logs* deverão ter os logs armazenados na máquina onde está alojada a aplicação. Garantida a configuração do *rsyslog*, foi introduzido no ficheiro de *log* correspondente ao serviço de SSH (sshd.log) o endereço IP "94.74.84.205". Desta forma é possível simular a presença de um endereço de IP teoricamente maligno e gerar um alerta. Após alterar o *conjob* para ser executado a cada 10 minutos, foram executados os testes de validação desta funcionalidade.

Primeiro, é garantir que esta funcionalidade só é executada se o utilizador a ativar. É necessário garantir que a funcionalidade está desativa no frontend, como é pode ser visto na figura 31, é possível validar nos logs da aplicação que a tarefa não é executada, na listagem 16.

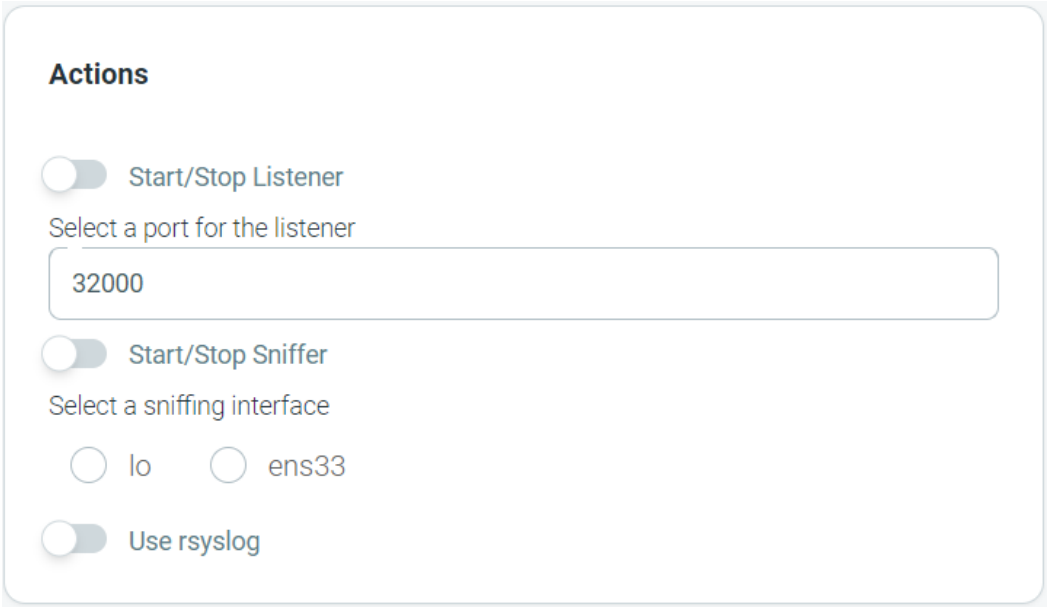


Figura 31: Funcionalidade de *rsyslog* desativada

Listagem 16: *Log* da aplicação a validar que a funcionalidade não está ativa

```
2024-08-02 14:20:03,337 - djangoLogger - INFO - rsyslog parsing is not active
2024-08-02 14:30:03,554 - djangoLogger - INFO - rsyslog parsing is not active
2024-08-02 14:40:03,305 - djangoLogger - INFO - rsyslog parsing is not active
```

De seguida, foi necessário ativar esta funcionalidade e garantir que é gerado um alerta com o endereço de IP acima mencionado. A figura 32, mostra o alerta que foi gerado após a ativação desta funcionalidade. Foi também possível validar que após

a primeira execução todos os ficheiros do diretório foram eliminados, garantindo desta forma que não são gerados alertas repetidos sempre que os *logs* do *rsyslog* são processados.

Figura 32: Alerta gerado pelo processamento do *rsyslog*

Com a criação do alerta e garantindo que os ficheiros de *log* são apagados após a execução, a funcionalidade implementada tem o comportamento esperado.

5.2.2.3 *Testes de carga*

Os testes de carga foram feitos em apenas três funcionalidades, o carregamento de dados de fontes externas na base de dados, o upload e tratamento de ficheiros para a aplicação e por fim o tratamento de ficheiros *log* recebidos por *rsyslog*.

No que diz respeito à recolha de dados externos foi medido o tempo que o processo de recolha demora desde o inicio da tarefa, que começa com o tratamento dos dados na base de dados, até á conclusão da tarefa que termina com o preenchimento da tabela referente aos valores na base de dados (figura 6). A funcionalidade de cronometrar o tempo de execução, foi feito através de uma funcionalidade desenvolvida (/home/martinho/Projects/SIEM-Light/backend/siemapi/api/logger.py) que faz um registo num ficheiro de *log* sempre que necessário, neste caso no inicio e fim da tarefa. Para além de cronometrar a tarefa, foi também registada a utilização de Central Processing Unit (CPU) e Random access memory (RAM) durante a execução destas tarefas.

No que diz respeito ao upload e tratamento de ficheiros, foram feitos os mesmos testes relativos ao tempo de execução bem como ao consumo de recursos. Estes tempos de execução foram contados para ficheiros com as seguintes características:

1. Três ficheiros de captura de tráfego (*pcap*) com tamanhos diferentes. O primeiro com 30MB o segundo com 50MB e o terceiro com 100MB;
2. Ficheiros de *log* recolhidos numa *firewall* em produção referentes a 1 hora e 24 horas de tráfego com ficheiros com aproximadamente os seguintes tamanhos: 59MB, 61MB, 4.2MB e 52MB

A utilização de ficheiros com tamanhos diferentes tem como objetivo validar até que tamanho a aplicação é suporta, o tempo que demora e recursos que utiliza.

Em primeiro lugar foi desenvolvido um pequeno *script* que tira partido da ferramenta "top"para fazer o rastreio do consumo de recursos do sistema. Este *script*, representado pela listagem 17, permite visualizar num dado período definido, neste caso de 15 minutos, a percentagem de utilização de CPU e a quantidade de RAM utilizada. Este scrip guarda ainda o *output* em ficheiros.

Listagem 17: Código fonte utilizado para fazer a monitorização de recursos

```
#!/bin/bash
DURATION=900  # 15 minutos
INTERVAL=10   # 10 segundos

end=$((SECONDS + DURATION))

while [ $SECONDS -lt $end ]; do
    top -b -n 1 | grep "Cpu(s)" >> cpu_usage_log_52_mb.txt
    top -b -n 1 | grep "MiB Mem" >> ram_usage_log_52_mb.txt
    sleep $INTERVAL
done
```

É necessário estabelecer qual é o consumo destes recursos com a máquina em *idle*, de modo a conseguir tirar algumas conclusões. A tabela 7 mostra as percentagens de utilização, no ponto máximo no intervalo de tempo definido, de RAM e CPU, isto é em *idle* o máximo de CPU utilizado foi 21,20% enquanto o máximo utilizado de Random access memory (RAM) foi de 41,81%.

	Idle
Consumo Máximo CPU	21,20%
Consumo Médio CPU	8.59%
Consumo Máximo RAM	41,81%

Tabela 7: Consumos em *idle*

No que diz respeito aos recursos utilizados no processamento de ficheiros que são carregados na aplicação, foram obtidos os resultados presentes na tabela 8. Dos resultados obtidos, conclui-se que o processamento de ficheiros *pcap* é computacio-

nalmente mais pesado do que o processamento de ficheiros *log*, isto deve-se à forma como cada um destes ficheiros é tratado. Podemos constatar que o consumo maior é ao nível do CPU (+30%) comparativamente ao consumo em *idle*, do que em RAM que se mantém muito equivalente aos consumos em *idle*.

	Ficheiros pcap			Ficheiros de log			
Tamanho	30MB	50MB	100MB	59MB	61MB	4,2MB	52MB
Consumo Máximo CPU	57,50%	61,50%	60,50%	51.40%	53,80%	43,20%	57,70%
Consumo Médio CPU	25,19%	29,59%	41,52%	20,03%	24,53%	18,60%	20.62%
Consumo Máximo RAM	42,62%	42,45%	43,38%	41,33%	41,92%	41,92%	42,13%
Duração	00:05:21	00:06:36	00:10:23	00:06:10	00:06:46	00:00:30	00:05:33

Tabela 8: Recursos utilizados no processamento de ficheiros

Nos testes realizados ao processamento de ficheiros vindos do *rsyslog*, apesar de não serem muito conclusivos pois foi feito com uma quantidade pequena ficheiros, 43 ficheiros com tamanho de 648Kb pertencentes à maquina local e a uma máquina remota. A tabela 9 apresenta dos resultados obtidos durante os testes. Nesta tabela podemos ver, que comparativamente ao processamento de ficheiros carregados na aplicação, e muito devido ao tamanho e volume dos ficheiros, esta tarefa não é computacionalmente pesada. No entanto, o consumo de recursos poderá ser consideravelmente superior considerando o número de máquinas que enviam os seus *logs* para a aplicação através do *rsyslog*.

	RSYSLOG
Consumo Máximo CPU	34,20%
Consumo Médio CPU	8,11%
Consumo Máximo RAM	42%
Duração	00:00:02
Tamanho	648K
Número Ficheiros	43

Tabela 9: Recursos utilizados no processamento de ficheiros obtidos através de *rsyslog*

Por fim, os testes aos recursos consumidos pela tarefa de recolha de dados externos. Para isso, foi necessário alterar a periodicidade com a qual a tarefa é executada. A tarefa foi alterada para a uma hora especifica, garantindo desta forma que existe tempo suficiente para ir buscar os dados de todas as fontes externas. Tendo em conta que esta tarefa executa várias operações, deste atualizar a base de dados, processar todas as fontes externas, inserir dados na base de dados e por fim limpar a base de dados, é uma tarefa que computacionalmente é mais exigente do que todas as outras, dai estar previsto apenas ser executada uma vez por dia. No que

diz respeito à duração da tarefa, e tendo em conta que apenas existem duas fontes externas configuradas, é uma tarefa bastante demorada e com o crescimento de fontes externas configuradas pode levar a uma duração ainda superior. A tabela 10 mostra os resultados obtidos.

	Recolha dados externos
Consumo Máximo CPU	62,80%
Consumo Médio CPU	44,04%
Consumo Máximo RAM	33,95%
Duração	00:27:41
Dados processados	73589844

Tabela 10: Recursos utilizados no processamento de fontes externas

Em suma, dadas as características do cenário de implementação, podemos ver que as tarefas executadas tem um maior consumo de CPU comparativamente ao consumo de RAM que se mantém muito idêntico aos consumos em *idle*. Podemos também concluir que a tarefa que é computacionalmente mais pesada é o processamento das fontes externas, com maior consumo de CPU e mais tempo de execução. Para além disso, é possível inferir que o processamento de ficheiros *pcap* é computacionalmente mais pesado do que o processamento de ficheiros *log*. Posto isto, a aplicação não excede muito os 60% de utilização de de CPU e não consome muita memória RAM para além daquela que é consumida em *idle*, pode dizer-se que com algumas otimizações pode ser uma forma muito eficiente de ter uma ferramenta funcional. Se ponderarmos a análise de todas as tarefas em simultâneo concluímos que o com o CPU atual, existe um incremento de 94,9%. Este valor é calculado através da soma das diferenças entre o consumo máximo de CPU e o consumo em *idle* para as tarefas de *rsyslog*, consumo de fontes externas e tratamento de um ficheiro, no caso do pcap de 50MB. Com base neste ultimo teste o dimensionamento das necessidades da máquina a utilizar deve ser com base nestes cálculos de valor de CPU, pois é o recurso com mais relevância. Todos os ficheiros obtidos durante os testes encontram-se no Apêndice B.

6

CONCLUSÕES

A grande maioria das organizações não têm recursos humanos e financeiros capazes de dar resposta á cresce ameaça de ciberataques e de se manterem informados sobre as condições que a sua organização se encontra. Existe uma falta no mercado de aplicações capazes de munir as equipas destas organizações com ferramentas que de uma forma forma simples as possam de indicar a realidade das suas organizações. Apesar de existirem aplicações capazes de dar esta mesma resposta, estas são de elevado grau de implementação, utilização e muita das vezes dispendiosas o que leva às organizações por não optarem por este tipo de soluções.

A criação de uma aplicação que permita a este tipo de organizações, através da recolha de dados internos como tráfego DNS e ficheiros de *log*, obter informação relevante sobre o estado da sua organização foi o principal objetivo deste projeto. Foi criada uma aplicação (demonstrador de conceito) que recolhe dados relevantes sobre possíveis indicadores de compromisso de fontes externas, realiza a recolha de dados interno, como tráfego e *logs* de uma forma manual e automática e gera alertas com base no cruzamento destes dois tipos de informação. Estes alertas, são sem duvida uma forma rápida, apesar de superficial, de avisar as equipas de informática que algo pode estar errado com a sua infraestrutura.

Os testes realizados à aplicação revelam que é uma solução que, no seu formato atual, requer poucos recursos para poder ser executada e é *opensource* removendo na sua grande maioria os encargos financeiros que muitas das outras aplicações acarretam. A sua facilidade de utilização também é um ponto positivo, pois requer poucos conhecimentos na área de cibersegurança para conseguir tirar partido das funcionalidades que a mesma fornece. O mais importante que se retira deste demonstrador desenvolvido durante estes meses foi que ao serem utilizados dados reais podemos sem duvida obter dados muitos relevantes que de outra forma passariam despercebidos, conseguindo assim tomar algum tipo de medida de mitigação ou investigar mais atentamente os alertas. Isto demonstra que a aplicação é útil apesar de ter poucas funcionalidades.

6.1 TRABALHOS FUTUROS

Apesar do principal objetivo ter sido atingido, ou seja a criação de uma aplicação com as características já referidas, a mesma ainda tem bastante margem de progressão e pontos a melhorar. Abordando alguns pontos menos positivos e que ainda tem margem para melhora, enumero os seguintes:

- Não existe autenticação - A não existência de um método de autenticação não permite fazer a gestão de acessos à aplicação o que pode tornar-se um problema;
- Criar diferentes formas de notificação de alertas - O demonstrador desenvolvido apenas permite que os alertas sejam acedidos e analisados através do *dashboard*, um possível forma de trazer valor para a aplicação é criar mecanismos diferentes para a notificação dos alertas para as equipas, tais como, notificação por e-mail, sms, ou por plataformas como o *Teams* ou *Telegram*;
- Criar formas para introdução de dados manuais pelos utilizadores - Permitir ao utilizador de inserir dados para que com bases nestes sejam gerados alertas. Exemplificando, o utilizador introduzir um conjunto dos seus endereços de IP públicos e casos algum destes aparece numa fonte externa seja gerado um alerta. O mesmo pode ser feito para nomes de domínio;
- Melhorias na análise de dados internos - Adicionar técnicas abordadas neste trabalho como a análise de semântica ou a introdução de algoritmos de NLP podendo desta forma aumentar a eficácia da análise;
- Introduzir novos tipos de dados - O presente demonstrador apenas trata endereços IP e nomes de domínio. Uma sugestão de melhoria seria a introdução de dados de outros tipos, como por exemplo, URL;
- Facilidade de implementação - Tratando-se este projeto apenas de um demonstrador, não foi elaborada uma forma simples de implementar a solução, sendo apenas possível através do código fonte e da implementação manual da infraestrutura. A sugestão de melhoria, seria implementar a aplicação com recurso a utilização de *containers*.

Desta forma, ainda existe bastante margem de progressão para este demonstrador, implementando algumas das sugestões enumeradas.

BIBLIOGRAFIA

ATT, Lauren Barraco - (abr. de 2014). *Top 5 Problems with Traditional SIEM*. URL: https://cybersecurity.att.com/blogs/security-essentials/top-5-problems-with-traditional-siem-infographic.

CloudFlare (fev. de 2024). *What is DNS? | How DNS works*. URL: https://www.cloudflare.com/learning/dns/what-is-dns/.

CNCS (Acedido a 2023-10-06). «Glossário Centro Nacional de Cibersegurança». Em: URL: https://www.cncs.gov.pt/pt/glossario/.

Cybriant (dez. de 2023). *SIEM Challanges*. URL: https://cybriant.com/siem-challenges/.

Endace (fev. de 2024). *What is a PCAP file?* URL: https://www.endace.com/learn/what-is-a-pcap-file.

Europeu, Concelho (2013). «DECISÃO DO CONSELHO de 23 de setembro de 2013 relativa às regras de segurança aplicáveis à proteção das informações classificadas da UE (2013/488/UE)». Em: *Jornal Oficial da União Europeia*.

Fail2ban (mar. de 2024). *Blocklist - Fail2ban report service*. URL: https://www.blocklist.de/en/export.html.

Gonçalves, Martinho (2024). *SIEM-Light*. URL: https://github.com/MartinhoGon/SIEM-Light.

Gustavo González-Granadillo Susana González-Zarzosa, Rodrigo Diaz (2021). «Security Information and Event Management (SIEM): Analysis,Trends, and Usage in Critical Infrastructures». Em: *Sensors 2021, 21, 4759*.

Holland, Sam (dez. de 2023). *What is an IP blacklist and how does it work?* URL: https://seon.io/resources/ip-blacklist/.

IntelMQ (dez. de 2023a). *Data Feeds*. URL: https://intelmq.readthedocs.io/en/latest/user/feeds.html#feodo-tracker.

— (dez. de 2023b). *Documentação Oficial*. URL: https://intelmq.readthedocs.io.

— (dez. de 2023c). *Página Oficial do IntelMQ-Tutorial - GitHub*. URL: https://github.com/certtools/intelmq-tutorial.

Ivo Vacas Ibéria Medeiros, Nuno Neves (2018). «Detecting Network Threats using OSINTKnowledge-based IDS». Em: *14th European Dependable Computing Conference*.

Kaspersky (Acedido a 2023-10-09). «Indicator of Compromise (IoC)». Em: URL: https://encyclopedia.kaspersky.com/glossary/indicator-of-compromise-ioc/.

KimiNewt (mar. de 2024). URL: https://github.com/KimiNewt/pyshark.

Kost, Edward (jan. de 2024). *How to Detect to Detect Data Exfiltration*. URL: https://www.upguard.com/blog/how-to-detect-data-exfiltration.

Logic, Sumo (jan. de 2024). *Log file - definition overview*. URL: https://www.sumologic.com/glossary/log-file/.

M. Uma, G. Padmavathi (2013). «A Survey on Various Cyber Attacks and Their Classification». Em: *International Journal of Network Security, Vol.15, No.5, PP.390-396, Sept. 2013*.

Malanov, Alexey (dez. de 2023). *Antivirus fundamentals: Viruses, signatures, disinfection*. URL: https://www.kaspersky.com/blog/signature-virus-disinfection/13233/.

Nam, Ethan (2019). «Understanding the Levenshtein Distance Equation for Beginners». Em: *Medium*. URL: https://medium.com/@ethannam/understanding-the-levenshtein-distance-equation-for-beginners-c4285a5604f0.

Netresec (dez. de 2023). *Networkminer*. URL: https://www.netresec.com/?page=NetworkMiner.

ntopng (nov. de 2023). *High-Speed Web-based Traffic Analysis and Flow Collection*. URL: https://www.ntop.org/products/traffic-analysis/ntop/.

Onwubiko, Cyril (2018-11-06). «CoCoa: An Ontology for Cybersecurity Operations Centre Analysis Process». Em.

Overcash, Tanner (2021). «Semantic Similarity Calculations Using NLP and Python: A Soft Introduction». Em: *Medium*. URL: https://medium.com/@tanner.overcash/semantic-similarity-calculations-using-nlp-and-python-a-soft-introduction-1f31df965e40.

Patrol, Malware (jan. de 2024). *Malware Hashes and Hash Functions*. URL: https://www.linkedin.com/pulse/malware-hashes-hash-functions-malware-patrol/.

Ravoof, Salman (dez. de 2023). *PostgreSQL vs MySQL: Explore Their 12 Critical Differences*. URL: https://kinsta.com/blog/postgresql-vs-mysql/.

Roesch, Martin (nov. de 2023). *Writing Snort Rules - version 1.7*. URL: https://paginas.fe.up.pt/~mgi98020/pgr/writing_snort_rules.htm#rule_actions.

Sandeep Bhatt Pratyusa K. Manadhata, Loai Zomlot (2014). «The Operational Role of Security Informationand Event Management Systems». Em: *Copublished by the IEEE Computer and Reliability Societies*.

Segurança das Redes e da Informação, ENISA -Agência Europeia para a (2018). «ENISA - Taxonomy Good practice guide on how to improve CSIRT capabilities». Em.

Sharma, Akshay (2022). *How to install and set up Rsyslog server - Linux Ubuntu 20.04.1*. URL: https://www.linkedin.com/pulse/how-install-set-up-rsyslog-server-linux-ubuntu-20041-akshay-sharma/.

Snort (nov. de 2023). *Snort - Opensource Intrusion Prevention System*. URL: https://www.snort.org/.

Tim, Creative (jul. de 2024). *Frontend Template*. URL: https://www.creative-tim.com/product/material-tailwind-dashboard-react.

Wazuh (dez. de 2023). *Site oficial do Wazuh*. URL: https://wazuh.com/platform/siem/.

APÊNDICES

A

TABELA DE CORRESPONDÊNCIA ENTRE ATAQUES E IOCS

Classificação	Ataques \ IoCs	Lookups de DNS	Ficheiros Aplicações e Processos Supeitos	Endereços IP,nomes de domínio e servidores C2	Número de acessos a ficheiros	Atividade Suspeita em contas de utilizadores	Atualizações inesperadas	Transferência de dados em portos pouco utilizados	Comportamentos estranhos em *websites*	Assinaturas de ataques ou *hashs* conhecidos	Tamanhos incomuns nas respostas HTTP	Mudanças não autorizadas em ficheiros de configurações, registo ou configuragções de equipamentos	Elevado número de tentativas falhas de login	Grandes volumes de tráfego	Padrões anómalos de rede	Grandes volumes de exfiltração de dados
Código Malicioso	Sistema Infetado		X	X			X		X	X		X			X	
	Distribuição de Malware	X		X											X	
	Servidor C2	X		X						X					X	
	Configuração de Malware		X	X												
Disponibilidade	Negação de Serviço			X									X	X		
	Negação de Serviço Distribuida			X									X	X		
	Configuração Incorreta								X		X	X				
	Sabotagem		X	X				X				X		X		
	Interrupção			X										X		
Recolha de Informação	Scanning			X		X									X	
	Sniffing			X		X									X	
	Engenharia Social		X													
Intrusão	Compromisso de conta previligiada		X			X	X									
	Compromisso de conta previligiada		X			X						X				
	Compromisso de aplicação		X			X						X				
	Arrombamento			X		X							X			
Tentativa de Intrusão	Explocação de Vulnerabilidades					X							X	X		
	Tentativa de logins			X									X	X		
	Nova assinatura de ataque	X		X												
Segurança da informação	Acesso não autorizado				X			X				X				
	Modificação não autorizada				X							X				
	Perda de dados				X			X						X	X	X
Fraude	Utilização indevida ou não autorizada de recursos					X		X	X							
	Direitos de autor			X				X								
	Utilização ilegítima de nome de terceiros					X										
	Phishing			X												
Conteúdo Abusivo	SPAM			X												
	Discursos Nocivo			X												
	Exploração sexual de menores, racismo e apologia à violência	X		X												
Vulnerabilidade	Criptografia fraca								X							
	Amplificador DDoS															
	Serviços acessíveis potêncialmente indesejados															
	Revelação de informação				X			X						X	X	
	Sistema vulnerável		X		X			X	X		X	X				X

B

FICHEIROS DE CONSUMO DE RECURSOS

Listagem 18: Consumo de CPU em *idle*

```
%Cpu(s): 13.5 us, 16.2 sy,  0.0 ni, 70.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  8.7 us, 13.0 sy,  0.0 ni, 78.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 20.0 us, 21.7 sy,  0.0 ni, 56.7 id,  0.0 wa,  0.0 hi,  1.7 si,  0.0 st
%Cpu(s):  9.1 us,  9.1 sy,  0.0 ni, 79.5 id,  0.0 wa,  0.0 hi,  2.3 si,  0.0 st
%Cpu(s):  5.7 us, 20.8 sy,  0.0 ni, 73.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 13.0 us, 15.2 sy,  0.0 ni, 71.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.6 us, 10.3 sy,  0.0 ni, 84.6 id,  0.0 wa,  0.0 hi,  2.6 si,  0.0 st
%Cpu(s):  6.4 us, 14.9 sy,  0.0 ni, 78.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 17.5 us, 12.5 sy,  0.0 ni, 70.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  8.9 us, 11.1 sy,  0.0 ni, 80.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  9.8 us, 21.6 sy,  0.0 ni, 68.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.7 us,  8.1 sy,  0.0 ni, 89.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  9.8 us,  7.3 sy,  0.0 ni, 82.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  8.2 us, 12.2 sy,  0.0 ni, 79.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 20.4 us, 22.2 sy,  0.0 ni, 57.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.2 us, 15.6 sy,  0.0 ni, 82.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.3 us, 15.2 sy,  0.0 ni, 80.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 10.3 us, 15.4 sy,  0.0 ni, 74.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.5 us, 13.6 sy,  0.0 ni, 81.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.7 us,  8.1 sy,  0.0 ni, 89.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 21.2 us, 19.2 sy,  0.0 ni, 59.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.3 us, 17.0 sy,  0.0 ni, 78.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 15.0 us, 12.5 sy,  0.0 ni, 72.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 14.0 us, 11.6 sy,  0.0 ni, 74.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.9 us,  9.8 sy,  0.0 ni, 82.9 id,  0.0 wa,  0.0 hi,  2.4 si,  0.0 st
%Cpu(s):  4.7 us, 11.6 sy,  0.0 ni, 81.4 id,  0.0 wa,  0.0 hi,  2.3 si,  0.0 st
%Cpu(s): 17.0 us, 20.8 sy,  0.0 ni, 62.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.4 us, 12.2 sy,  0.0 ni, 82.9 id,  0.0 wa,  0.0 hi,  2.4 si,  0.0 st
%Cpu(s):  6.5 us, 13.0 sy,  0.0 ni, 80.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 10.5 us, 13.2 sy,  0.0 ni, 76.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.4 us, 13.3 sy,  0.0 ni, 82.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.1 us, 10.3 sy,  0.0 ni, 84.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  6.7 us, 13.3 sy,  0.0 ni, 80.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  9.3 us, 18.5 sy,  0.0 ni, 72.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 20.4 us, 24.1 sy,  0.0 ni, 55.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.1 us,  7.7 sy,  0.0 ni, 87.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.5 us, 13.6 sy,  0.0 ni, 79.5 id,  0.0 wa,  0.0 hi,  2.3 si,  0.0 st
%Cpu(s):  4.4 us, 13.3 sy,  0.0 ni, 82.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.7 us,  7.7 sy,  0.0 ni, 84.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  9.8 us,  7.3 sy,  0.0 ni, 82.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 20.5 us, 13.6 sy,  0.0 ni, 65.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  6.2 us, 14.6 sy,  0.0 ni, 79.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.0 us, 19.3 sy,  0.0 ni, 73.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.3 us,  7.9 sy,  0.0 ni, 86.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  8.0 us, 14.0 sy,  0.0 ni, 78.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  9.1 us,  9.1 sy,  0.0 ni, 81.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.5 us, 12.5 sy,  0.0 ni, 85.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  3.0 us,  3.0 sy,  0.0 ni, 93.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  6.4 us, 12.8 sy,  0.0 ni, 80.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  8.7 us, 10.9 sy,  0.0 ni, 80.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  6.5 us, 13.0 sy,  0.0 ni, 80.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.4 us,  5.4 sy,  0.0 ni, 89.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
```

Listagem 19: Consumo de RAM em *idle*

```
MiB Mem :   3875.9 total,    401.3 free,   1618.3 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    401.3 free,   1618.3 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    401.3 free,   1618.3 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    401.3 free,   1618.3 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    401.3 free,   1618.3 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    401.3 free,   1618.3 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    401.1 free,   1618.5 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    401.1 free,   1618.5 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    401.1 free,   1618.5 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    401.1 free,   1618.5 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    401.1 free,   1618.5 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    401.1 free,   1618.5 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    401.1 free,   1618.5 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    401.1 free,   1618.5 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    401.1 free,   1618.5 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    401.1 free,   1618.5 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    401.1 free,   1618.5 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    401.1 free,   1618.5 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    401.1 free,   1618.5 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    401.1 free,   1618.5 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    401.1 free,   1618.5 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    401.1 free,   1618.5 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    401.1 free,   1618.5 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    401.1 free,   1618.5 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    401.1 free,   1618.5 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    401.1 free,   1618.5 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    401.1 free,   1618.5 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    401.1 free,   1618.5 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    401.1 free,   1618.5 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    400.6 free,   1619.0 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    400.6 free,   1619.0 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    400.4 free,   1619.2 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    400.4 free,   1619.2 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    400.4 free,   1619.2 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    400.4 free,   1619.2 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    400.4 free,   1619.2 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    400.4 free,   1619.2 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    400.4 free,   1619.2 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    400.1 free,   1619.5 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    399.9 free,   1619.7 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    399.9 free,   1619.7 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    399.7 free,   1619.9 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    399.4 free,   1620.2 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    399.4 free,   1620.2 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    399.4 free,   1620.2 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    399.4 free,   1620.2 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    399.2 free,   1620.4 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    399.2 free,   1620.4 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    398.9 free,   1620.7 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    398.9 free,   1620.7 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    398.9 free,   1620.7 used,   1856.3 buff/cache
MiB Mem :   3875.9 total,    398.9 free,   1620.7 used,   1856.3 buff/cache
```

Listagem 20: Consumo de CPU para log de 4MB

```
%Cpu(s): 10.8 us, 16.2 sy,  0.0 ni, 73.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 42.1 us,  7.9 sy,  0.0 ni, 47.4 id,  0.0 wa,  0.0 hi,  2.6 si,  0.0 st
%Cpu(s): 43.2 us,  8.1 sy,  0.0 ni, 43.2 id,  2.7 wa,  0.0 hi,  2.7 si,  0.0 st
%Cpu(s): 42.9 us, 11.9 sy,  0.0 ni, 40.5 id,  0.0 wa,  0.0 hi,  4.8 si,  0.0 st
%Cpu(s):  4.7 us, 11.6 sy,  0.0 ni, 83.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.4 us,  5.4 sy,  0.0 ni, 89.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.0 us, 11.6 sy,  0.0 ni, 81.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 15.6 us, 20.0 sy,  0.0 ni, 64.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.6 us, 16.7 sy,  0.0 ni, 77.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  8.7 us, 10.9 sy,  0.0 ni, 80.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
```

Listagem 21: Consumo de RAM para log de 4MB

```
MiB Mem :   3875.9 total,    399.6 free,   1620.7 used,   1855.6 buff/cache
MiB Mem :   3875.9 total,    388.5 free,   1623.3 used,   1864.2 buff/cache
MiB Mem :   3875.9 total,    388.2 free,   1623.4 used,   1864.2 buff/cache
MiB Mem :   3875.9 total,    387.6 free,   1624.0 used,   1864.4 buff/cache
MiB Mem :   3875.9 total,    395.5 free,   1624.1 used,   1856.2 buff/cache
MiB Mem :   3875.9 total,    395.5 free,   1624.1 used,   1856.2 buff/cache
MiB Mem :   3875.9 total,    395.3 free,   1624.4 used,   1856.2 buff/cache
MiB Mem :   3875.9 total,    395.0 free,   1624.6 used,   1856.2 buff/cache
MiB Mem :   3875.9 total,    395.0 free,   1624.6 used,   1856.2 buff/cache
MiB Mem :   3875.9 total,    394.8 free,   1624.9 used,   1856.2 buff/cache
```

Listagem 22: Consumo de CPU para log de 52MB

```
%Cpu(s):  8.3 us, 14.6 sy,  0.0 ni, 77.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  8.3 us, 10.4 sy,  0.0 ni, 81.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 45.5 us, 13.6 sy,  0.0 ni, 38.6 id,  0.0 wa,  0.0 hi,  2.3 si,  0.0 st
%Cpu(s): 50.0 us,  7.5 sy,  0.0 ni, 37.5 id,  0.0 wa,  0.0 hi,  5.0 si,  0.0 st
%Cpu(s): 46.2 us, 15.4 sy,  0.0 ni, 35.9 id,  0.0 wa,  0.0 hi,  2.6 si,  0.0 st
%Cpu(s): 46.2 us,  7.7 sy,  0.0 ni, 41.0 id,  0.0 wa,  0.0 hi,  5.1 si,  0.0 st
%Cpu(s): 40.9 us, 13.6 sy,  0.0 ni, 38.6 id,  0.0 wa,  0.0 hi,  6.8 si,  0.0 st
%Cpu(s): 37.5 us, 12.5 sy,  0.0 ni, 45.0 id,  0.0 wa,  0.0 hi,  5.0 si,  0.0 st
%Cpu(s): 40.9 us, 18.2 sy,  0.0 ni, 36.4 id,  0.0 wa,  0.0 hi,  4.5 si,  0.0 st
%Cpu(s): 43.9 us, 12.2 sy,  0.0 ni, 39.0 id,  0.0 wa,  0.0 hi,  4.9 si,  0.0 st
%Cpu(s): 42.9 us, 11.4 sy,  0.0 ni, 45.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 47.6 us, 11.9 sy,  0.0 ni, 38.1 id,  0.0 wa,  0.0 hi,  2.4 si,  0.0 st
%Cpu(s): 44.2 us, 16.3 sy,  0.0 ni, 37.2 id,  0.0 wa,  0.0 hi,  2.3 si,  0.0 st
%Cpu(s): 57.7 us, 26.9 sy,  0.0 ni, 13.5 id,  0.0 wa,  0.0 hi,  1.9 si,  0.0 st
%Cpu(s): 47.5 us, 10.0 sy,  0.0 ni, 40.0 id,  0.0 wa,  0.0 hi,  2.5 si,  0.0 st
%Cpu(s): 40.9 us, 13.6 sy,  0.0 ni, 40.9 id,  0.0 wa,  0.0 hi,  4.5 si,  0.0 st
%Cpu(s): 51.3 us, 10.3 sy,  0.0 ni, 35.9 id,  0.0 wa,  0.0 hi,  2.6 si,  0.0 st
%Cpu(s): 40.0 us, 12.5 sy,  0.0 ni, 42.5 id,  0.0 wa,  0.0 hi,  5.0 si,  0.0 st
%Cpu(s): 37.8 us, 18.9 sy,  0.0 ni, 40.5 id,  0.0 wa,  0.0 hi,  2.7 si,  0.0 st
%Cpu(s): 49.1 us, 17.0 sy,  0.0 ni, 30.2 id,  0.0 wa,  0.0 hi,  3.8 si,  0.0 st
%Cpu(s): 45.8 us, 20.8 sy,  0.0 ni, 31.2 id,  2.1 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 47.9 us, 16.7 sy,  0.0 ni, 31.2 id,  0.0 wa,  0.0 hi,  4.2 si,  0.0 st
%Cpu(s): 37.2 us, 11.6 sy,  0.0 ni, 46.5 id,  0.0 wa,  0.0 hi,  4.7 si,  0.0 st
%Cpu(s): 47.5 us, 10.0 sy,  0.0 ni, 37.5 id,  0.0 wa,  0.0 hi,  5.0 si,  0.0 st
%Cpu(s): 46.5 us, 18.6 sy,  0.0 ni, 34.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
```

```
%Cpu(s): 46.7 us, 11.1 sy,  0.0 ni, 37.8 id,  0.0 wa,  0.0 hi,  4.4 si,  0.0 st
%Cpu(s): 43.2 us, 13.6 sy,  0.0 ni, 38.6 id,  0.0 wa,  0.0 hi,  4.5 si,  0.0 st
%Cpu(s): 49.1 us, 15.1 sy,  0.0 ni, 34.0 id,  0.0 wa,  0.0 hi,  1.9 si,  0.0 st
%Cpu(s): 48.8 us, 11.6 sy,  0.0 ni, 34.9 id,  0.0 wa,  0.0 hi,  4.7 si,  0.0 st
%Cpu(s): 42.5 us, 15.0 sy,  0.0 ni, 35.0 id,  0.0 wa,  0.0 hi,  7.5 si,  0.0 st
%Cpu(s): 41.5 us, 12.2 sy,  0.0 ni, 43.9 id,  0.0 wa,  0.0 hi,  2.4 si,  0.0 st
%Cpu(s): 41.5 us,  9.8 sy,  0.0 ni, 43.9 id,  0.0 wa,  0.0 hi,  4.9 si,  0.0 st
%Cpu(s): 39.5 us,  9.3 sy,  0.0 ni, 46.5 id,  0.0 wa,  0.0 hi,  4.7 si,  0.0 st
%Cpu(s): 11.1 us,  8.3 sy,  0.0 ni, 80.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  8.2 us, 14.3 sy,  0.0 ni, 77.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.5 us, 11.4 sy,  0.0 ni, 81.8 id,  0.0 wa,  0.0 hi,  2.3 si,  0.0 st
%Cpu(s):  5.1 us,  7.7 sy,  0.0 ni, 87.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 15.6 us, 13.3 sy,  0.0 ni, 68.9 id,  0.0 wa,  0.0 hi,  2.2 si,  0.0 st
%Cpu(s):  4.7 us,  9.3 sy,  0.0 ni, 83.7 id,  0.0 wa,  0.0 hi,  2.3 si,  0.0 st
%Cpu(s):  2.3 us, 14.0 sy,  0.0 ni, 83.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 10.3 us,  7.7 sy,  0.0 ni, 82.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  8.7 us, 10.9 sy,  0.0 ni, 80.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  6.8 us, 11.4 sy,  0.0 ni, 79.5 id,  0.0 wa,  0.0 hi,  2.3 si,  0.0 st
%Cpu(s):  5.0 us, 10.0 sy,  0.0 ni, 85.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 14.6 us, 14.6 sy,  0.0 ni, 70.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.4 us, 14.3 sy,  0.0 ni, 83.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.1 us,  9.5 sy,  0.0 ni, 83.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 11.5 us, 13.5 sy,  0.0 ni, 75.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.6 us, 10.5 sy,  0.0 ni, 86.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.4 us, 13.3 sy,  0.0 ni, 80.0 id,  0.0 wa,  0.0 hi,  2.2 si,  0.0 st
%Cpu(s):  6.7 us, 13.3 sy,  0.0 ni, 80.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.7 us, 11.6 sy,  0.0 ni, 83.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.3 us, 15.2 sy,  0.0 ni, 80.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  6.5 us, 15.2 sy,  0.0 ni, 78.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 17.5 us, 28.1 sy,  0.0 ni, 52.6 id,  0.0 wa,  0.0 hi,  1.8 si,  0.0 st
%Cpu(s):  0.0 us,  8.6 sy,  0.0 ni, 91.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.3 us,  9.8 sy,  0.0 ni, 82.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.7 us,  8.1 sy,  0.0 ni, 89.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  9.1 us, 20.5 sy,  0.0 ni, 68.2 id,  0.0 wa,  0.0 hi,  2.3 si,  0.0 st
%Cpu(s):  8.7 us, 10.9 sy,  0.0 ni, 80.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.7 us, 11.6 sy,  0.0 ni, 83.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.0 us, 10.0 sy,  0.0 ni, 85.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 17.9 us, 15.4 sy,  0.0 ni, 66.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.5 us,  7.5 sy,  0.0 ni, 85.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.0 us,  9.3 sy,  0.0 ni, 81.4 id,  0.0 wa,  0.0 hi,  2.3 si,  0.0 st
%Cpu(s):  7.7 us,  5.1 sy,  0.0 ni, 87.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.7 us, 11.6 sy,  0.0 ni, 83.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 10.9 us, 17.4 sy,  0.0 ni, 69.6 id,  0.0 wa,  0.0 hi,  2.2 si,  0.0 st
%Cpu(s):  4.5 us, 13.6 sy,  0.0 ni, 81.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.4 us, 14.3 sy,  0.0 ni, 83.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 11.6 us, 18.6 sy,  0.0 ni, 69.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 11.5 us, 13.5 sy,  0.0 ni, 75.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.4 us, 18.5 sy,  0.0 ni, 74.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.1 us, 16.3 sy,  0.0 ni, 79.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.7 us, 14.0 sy,  0.0 ni, 81.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.8 us,  8.3 sy,  0.0 ni, 88.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  9.3 us, 14.8 sy,  0.0 ni, 75.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.3 us, 14.9 sy,  0.0 ni, 80.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.6 us, 12.8 sy,  0.0 ni, 84.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
```

```
%Cpu(s):  4.5 us, 11.4 sy,  0.0 ni, 81.8 id,  0.0 wa,  0.0 hi,  2.3 si,  0.0 st
%Cpu(s): 15.9 us, 20.5 sy,  0.0 ni, 63.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.4 us, 13.3 sy,  0.0 ni, 82.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.6 us,  7.9 sy,  0.0 ni, 89.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 15.8 us, 10.5 sy,  0.0 ni, 73.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 12.0 us, 14.0 sy,  0.0 ni, 74.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.1 us,  9.5 sy,  0.0 ni, 83.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.2 us, 16.7 sy,  0.0 ni, 79.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
```

Listagem 23: Consumo de RAM para log de 52MB

```
MiB Mem :   3875.9 total,    404.4 free,   1615.0 used,   1856.5 buff/cache
MiB Mem :   3875.9 total,    404.4 free,   1615.0 used,   1856.5 buff/cache
MiB Mem :   3875.9 total,    298.6 free,   1617.1 used,   1960.2 buff/cache
MiB Mem :   3875.9 total,    292.8 free,   1622.8 used,   1960.2 buff/cache
MiB Mem :   3875.9 total,    289.1 free,   1626.5 used,   1960.2 buff/cache
MiB Mem :   3875.9 total,    287.7 free,   1627.9 used,   1960.4 buff/cache
MiB Mem :   3875.9 total,    286.4 free,   1629.0 used,   1960.4 buff/cache
MiB Mem :   3875.9 total,    286.4 free,   1629.0 used,   1960.5 buff/cache
MiB Mem :   3875.9 total,    286.4 free,   1628.8 used,   1960.6 buff/cache
MiB Mem :   3875.9 total,    286.4 free,   1628.8 used,   1960.7 buff/cache
MiB Mem :   3875.9 total,    286.4 free,   1628.7 used,   1960.8 buff/cache
MiB Mem :   3875.9 total,    286.4 free,   1628.6 used,   1960.8 buff/cache
MiB Mem :   3875.9 total,    286.4 free,   1628.6 used,   1960.9 buff/cache
MiB Mem :   3875.9 total,    281.8 free,   1633.2 used,   1960.9 buff/cache
MiB Mem :   3875.9 total,    281.8 free,   1633.1 used,   1961.0 buff/cache
MiB Mem :   3875.9 total,    282.3 free,   1632.6 used,   1961.0 buff/cache
MiB Mem :   3875.9 total,    284.2 free,   1630.6 used,   1961.1 buff/cache
MiB Mem :   3875.9 total,    284.5 free,   1630.2 used,   1961.2 buff/cache
MiB Mem :   3875.9 total,    284.5 free,   1630.2 used,   1961.2 buff/cache
MiB Mem :   3875.9 total,    284.5 free,   1630.1 used,   1961.3 buff/cache
MiB Mem :   3875.9 total,    284.5 free,   1630.1 used,   1961.3 buff/cache
MiB Mem :   3875.9 total,    284.5 free,   1630.0 used,   1961.4 buff/cache
MiB Mem :   3875.9 total,    284.5 free,   1629.9 used,   1961.5 buff/cache
MiB Mem :   3875.9 total,    284.5 free,   1629.9 used,   1961.5 buff/cache
MiB Mem :   3875.9 total,    285.3 free,   1629.0 used,   1961.6 buff/cache
MiB Mem :   3875.9 total,    288.7 free,   1625.6 used,   1961.6 buff/cache
MiB Mem :   3875.9 total,    290.2 free,   1624.1 used,   1961.6 buff/cache
MiB Mem :   3875.9 total,    290.2 free,   1624.1 used,   1961.6 buff/cache
MiB Mem :   3875.9 total,    290.2 free,   1624.1 used,   1961.6 buff/cache
MiB Mem :   3875.9 total,    290.7 free,   1623.6 used,   1961.6 buff/cache
MiB Mem :   3875.9 total,    290.5 free,   1623.8 used,   1961.7 buff/cache
MiB Mem :   3875.9 total,    290.0 free,   1624.2 used,   1961.7 buff/cache
MiB Mem :   3875.9 total,    290.0 free,   1624.1 used,   1961.7 buff/cache
MiB Mem :   3875.9 total,    420.8 free,   1596.8 used,   1858.3 buff/cache
MiB Mem :   3875.9 total,    420.3 free,   1597.4 used,   1858.2 buff/cache
MiB Mem :   3875.9 total,    420.3 free,   1597.4 used,   1858.2 buff/cache
MiB Mem :   3875.9 total,    420.3 free,   1597.4 used,   1858.2 buff/cache
MiB Mem :   3875.9 total,    420.3 free,   1597.4 used,   1858.2 buff/cache
MiB Mem :   3875.9 total,    420.3 free,   1597.4 used,   1858.2 buff/cache
MiB Mem :   3875.9 total,    420.0 free,   1597.6 used,   1858.2 buff/cache
```

```
41 MiB Mem :   3875.9 total,    420.0 free,   1597.6 used,   1858.2 buff/cache
42 MiB Mem :   3875.9 total,    420.0 free,   1597.6 used,   1858.2 buff/cache
43 MiB Mem :   3875.9 total,    420.0 free,   1597.6 used,   1858.2 buff/cache
44 MiB Mem :   3875.9 total,    420.0 free,   1597.6 used,   1858.2 buff/cache
45 MiB Mem :   3875.9 total,    419.3 free,   1598.4 used,   1858.2 buff/cache
46 MiB Mem :   3875.9 total,    419.1 free,   1598.6 used,   1858.2 buff/cache
47 MiB Mem :   3875.9 total,    419.1 free,   1598.6 used,   1858.2 buff/cache
48 MiB Mem :   3875.9 total,    418.8 free,   1598.9 used,   1858.2 buff/cache
49 MiB Mem :   3875.9 total,    418.4 free,   1599.3 used,   1858.2 buff/cache
50 MiB Mem :   3875.9 total,    418.4 free,   1599.3 used,   1858.2 buff/cache
51 MiB Mem :   3875.9 total,    418.4 free,   1599.3 used,   1858.2 buff/cache
52 MiB Mem :   3875.9 total,    416.9 free,   1600.8 used,   1858.2 buff/cache
53 MiB Mem :   3875.9 total,    415.9 free,   1601.8 used,   1858.2 buff/cache
54 MiB Mem :   3875.9 total,    414.7 free,   1603.0 used,   1858.2 buff/cache
55 MiB Mem :   3875.9 total,    414.7 free,   1603.0 used,   1858.2 buff/cache
56 MiB Mem :   3875.9 total,    414.7 free,   1603.0 used,   1858.2 buff/cache
57 MiB Mem :   3875.9 total,    414.7 free,   1603.0 used,   1858.2 buff/cache
58 MiB Mem :   3875.9 total,    414.7 free,   1603.0 used,   1858.2 buff/cache
59 MiB Mem :   3875.9 total,    414.7 free,   1603.0 used,   1858.2 buff/cache
60 MiB Mem :   3875.9 total,    414.7 free,   1603.0 used,   1858.2 buff/cache
61 MiB Mem :   3875.9 total,    414.7 free,   1603.0 used,   1858.2 buff/cache
62 MiB Mem :   3875.9 total,    414.7 free,   1603.0 used,   1858.2 buff/cache
63 MiB Mem :   3875.9 total,    414.7 free,   1603.0 used,   1858.3 buff/cache
64 MiB Mem :   3875.9 total,    414.4 free,   1603.2 used,   1858.3 buff/cache
65 MiB Mem :   3875.9 total,    414.2 free,   1603.5 used,   1858.3 buff/cache
66 MiB Mem :   3875.9 total,    414.2 free,   1603.5 used,   1858.3 buff/cache
67 MiB Mem :   3875.9 total,    414.2 free,   1603.5 used,   1858.3 buff/cache
68 MiB Mem :   3875.9 total,    414.2 free,   1603.5 used,   1858.3 buff/cache
69 MiB Mem :   3875.9 total,    414.2 free,   1603.5 used,   1858.3 buff/cache
70 MiB Mem :   3875.9 total,    414.2 free,   1603.5 used,   1858.3 buff/cache
71 MiB Mem :   3875.9 total,    414.2 free,   1603.5 used,   1858.3 buff/cache
72 MiB Mem :   3875.9 total,    414.2 free,   1603.5 used,   1858.3 buff/cache
73 MiB Mem :   3875.9 total,    414.2 free,   1603.5 used,   1858.3 buff/cache
74 MiB Mem :   3875.9 total,    414.2 free,   1603.4 used,   1858.3 buff/cache
75 MiB Mem :   3875.9 total,    414.2 free,   1603.4 used,   1858.3 buff/cache
76 MiB Mem :   3875.9 total,    414.2 free,   1603.4 used,   1858.3 buff/cache
77 MiB Mem :   3875.9 total,    414.2 free,   1603.4 used,   1858.3 buff/cache
78 MiB Mem :   3875.9 total,    414.2 free,   1603.4 used,   1858.3 buff/cache
79 MiB Mem :   3875.9 total,    414.2 free,   1603.4 used,   1858.3 buff/cache
80 MiB Mem :   3875.9 total,    414.2 free,   1603.4 used,   1858.3 buff/cache
81 MiB Mem :   3875.9 total,    414.2 free,   1603.4 used,   1858.3 buff/cache
82 MiB Mem :   3875.9 total,    414.2 free,   1603.4 used,   1858.3 buff/cache
83 MiB Mem :   3875.9 total,    414.2 free,   1603.4 used,   1858.3 buff/cache
84 MiB Mem :   3875.9 total,    414.2 free,   1603.4 used,   1858.3 buff/cache
85 MiB Mem :   3875.9 total,    414.2 free,   1603.4 used,   1858.3 buff/cache
86 MiB Mem :   3875.9 total,    414.2 free,   1603.4 used,   1858.3 buff/cache
87 MiB Mem :   3875.9 total,    525.2 free,   1486.1 used,   1864.6 buff/cache
```

Listagem 24: Consumo de CPU para log de 59MB

```
1 %Cpu(s):  8.5 us, 12.8 sy,  0.0 ni, 78.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
2 %Cpu(s): 44.4 us, 15.6 sy,  0.0 ni, 40.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
```

```
%Cpu(s): 48.7 us, 12.8 sy,  0.0 ni, 33.3 id,  0.0 wa,  0.0 hi,  5.1 si,  0.0 st
%Cpu(s): 40.9 us, 13.6 sy,  0.0 ni, 43.2 id,  0.0 wa,  0.0 hi,  2.3 si,  0.0 st
%Cpu(s): 42.1 us, 10.5 sy,  0.0 ni, 42.1 id,  0.0 wa,  0.0 hi,  5.3 si,  0.0 st
%Cpu(s): 45.2 us, 11.9 sy,  0.0 ni, 40.5 id,  0.0 wa,  0.0 hi,  2.4 si,  0.0 st
%Cpu(s): 51.4 us,  8.1 sy,  0.0 ni, 40.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 51.2 us,  9.8 sy,  0.0 ni, 36.6 id,  0.0 wa,  0.0 hi,  2.4 si,  0.0 st
%Cpu(s): 46.3 us,  9.8 sy,  0.0 ni, 39.0 id,  0.0 wa,  0.0 hi,  4.9 si,  0.0 st
%Cpu(s): 37.2 us, 16.3 sy,  0.0 ni, 41.9 id,  0.0 wa,  0.0 hi,  4.7 si,  0.0 st
%Cpu(s): 45.0 us,  7.5 sy,  0.0 ni, 40.0 id,  0.0 wa,  0.0 hi,  7.5 si,  0.0 st
%Cpu(s): 47.4 us, 18.4 sy,  0.0 ni, 34.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 46.3 us, 19.5 sy,  0.0 ni, 31.7 id,  0.0 wa,  0.0 hi,  2.4 si,  0.0 st
%Cpu(s): 46.8 us, 17.0 sy,  0.0 ni, 31.9 id,  0.0 wa,  0.0 hi,  4.3 si,  0.0 st
%Cpu(s): 50.0 us,  7.5 sy,  0.0 ni, 42.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 50.0 us,  7.5 sy,  0.0 ni, 40.0 id,  0.0 wa,  0.0 hi,  2.5 si,  0.0 st
%Cpu(s): 41.0 us, 23.1 sy,  0.0 ni, 35.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 43.2 us, 11.4 sy,  0.0 ni, 40.9 id,  0.0 wa,  0.0 hi,  4.5 si,  0.0 st
%Cpu(s): 48.9 us, 17.8 sy,  0.0 ni, 28.9 id,  0.0 wa,  0.0 hi,  4.4 si,  0.0 st
%Cpu(s): 40.0 us, 17.5 sy,  0.0 ni, 40.0 id,  0.0 wa,  0.0 hi,  2.5 si,  0.0 st
%Cpu(s): 42.5 us, 15.0 sy,  0.0 ni, 42.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 47.6 us, 14.3 sy,  0.0 ni, 38.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 41.5 us, 14.6 sy,  0.0 ni, 43.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 42.5 us, 10.0 sy,  0.0 ni, 42.5 id,  0.0 wa,  0.0 hi,  5.0 si,  0.0 st
%Cpu(s): 42.1 us, 13.2 sy,  0.0 ni, 39.5 id,  0.0 wa,  0.0 hi,  5.3 si,  0.0 st
%Cpu(s): 42.1 us, 10.5 sy,  0.0 ni, 44.7 id,  0.0 wa,  0.0 hi,  2.6 si,  0.0 st
%Cpu(s): 51.3 us,  5.1 sy,  0.0 ni, 38.5 id,  0.0 wa,  0.0 hi,  5.1 si,  0.0 st
%Cpu(s): 47.6 us, 14.3 sy,  0.0 ni, 35.7 id,  0.0 wa,  0.0 hi,  2.4 si,  0.0 st
%Cpu(s): 45.0 us, 15.0 sy,  0.0 ni, 37.5 id,  0.0 wa,  0.0 hi,  2.5 si,  0.0 st
%Cpu(s): 45.0 us, 12.5 sy,  0.0 ni, 40.0 id,  0.0 wa,  0.0 hi,  2.5 si,  0.0 st
%Cpu(s): 40.4 us, 14.9 sy,  0.0 ni, 40.4 id,  0.0 wa,  0.0 hi,  4.3 si,  0.0 st
%Cpu(s): 48.7 us,  7.7 sy,  0.0 ni, 41.0 id,  0.0 wa,  0.0 hi,  2.6 si,  0.0 st
%Cpu(s): 39.0 us,  9.8 sy,  0.0 ni, 43.9 id,  0.0 wa,  0.0 hi,  7.3 si,  0.0 st
%Cpu(s): 40.5 us, 16.2 sy,  0.0 ni, 37.8 id,  0.0 wa,  0.0 hi,  5.4 si,  0.0 st
%Cpu(s): 48.7 us, 12.8 sy,  0.0 ni, 35.9 id,  0.0 wa,  0.0 hi,  2.6 si,  0.0 st
%Cpu(s): 42.1 us, 23.7 sy,  0.0 ni, 31.6 id,  0.0 wa,  0.0 hi,  2.6 si,  0.0 st
%Cpu(s): 35.9 us, 17.9 sy,  0.0 ni, 43.6 id,  0.0 wa,  0.0 hi,  2.6 si,  0.0 st
%Cpu(s):  6.7 us, 13.3 sy,  0.0 ni, 77.8 id,  0.0 wa,  0.0 hi,  2.2 si,  0.0 st
%Cpu(s):  7.5 us,  7.5 sy,  0.0 ni, 85.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.0 us,  9.3 sy,  0.0 ni, 81.4 id,  0.0 wa,  0.0 hi,  2.3 si,  0.0 st
%Cpu(s):  2.6 us, 10.3 sy,  0.0 ni, 87.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.4 us,  5.4 sy,  0.0 ni, 89.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.4 us,  5.4 sy,  0.0 ni, 89.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.6 us,  7.9 sy,  0.0 ni, 89.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 16.2 us, 10.8 sy,  0.0 ni, 70.3 id,  0.0 wa,  0.0 hi,  2.7 si,  0.0 st
%Cpu(s):  9.8 us,  4.9 sy,  0.0 ni, 85.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.1 us, 14.3 sy,  0.0 ni, 78.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.6 us, 10.5 sy,  0.0 ni, 86.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.5 us, 13.6 sy,  0.0 ni, 81.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.6 us,  2.8 sy,  0.0 ni, 91.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.4 us, 10.8 sy,  0.0 ni, 83.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 22.0 us, 19.5 sy,  0.0 ni, 58.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.6 us, 10.3 sy,  0.0 ni, 87.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  6.8 us, 11.4 sy,  0.0 ni, 79.5 id,  0.0 wa,  0.0 hi,  2.3 si,  0.0 st
%Cpu(s):  7.1 us,  9.5 sy,  0.0 ni, 83.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  8.5 us, 12.8 sy,  0.0 ni, 78.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
```

```
%Cpu(s):  9.8 us,  9.8 sy,  0.0 ni, 80.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  9.3 us, 18.6 sy,  0.0 ni, 72.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.6 us, 10.5 sy,  0.0 ni, 86.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.3 us, 13.6 sy,  0.0 ni, 84.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.2 us, 18.8 sy,  0.0 ni, 77.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 13.0 us,  8.7 sy,  0.0 ni, 76.1 id,  0.0 wa,  0.0 hi,  2.2 si,  0.0 st
%Cpu(s):  2.4 us, 12.2 sy,  0.0 ni, 85.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  9.5 us, 21.4 sy,  0.0 ni, 69.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 12.8 us,  7.7 sy,  0.0 ni, 79.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  8.0 us, 14.0 sy,  0.0 ni, 78.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.8 us, 14.3 sy,  0.0 ni, 81.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.4 us,  5.4 sy,  0.0 ni, 89.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.0 us, 10.0 sy,  0.0 ni, 85.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.7 us,  2.9 sy,  0.0 ni, 91.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.9 us,  9.8 sy,  0.0 ni, 85.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 17.1 us, 14.6 sy,  0.0 ni, 68.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 13.0 us, 19.6 sy,  0.0 ni, 67.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.9 us, 12.2 sy,  0.0 ni, 82.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 11.9 us,  9.5 sy,  0.0 ni, 78.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  6.5 us, 13.0 sy,  0.0 ni, 80.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.7 us, 11.6 sy,  0.0 ni, 81.4 id,  0.0 wa,  0.0 hi,  2.3 si,  0.0 st
%Cpu(s):  4.7 us, 14.0 sy,  0.0 ni, 81.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 10.8 us,  8.1 sy,  0.0 ni, 81.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.2 us, 16.7 sy,  0.0 ni, 79.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.7 us,  2.9 sy,  0.0 ni, 91.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 10.3 us, 12.8 sy,  0.0 ni, 76.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 13.2 us, 10.5 sy,  0.0 ni, 76.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  6.8 us, 11.4 sy,  0.0 ni, 81.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.2 us, 15.6 sy,  0.0 ni, 82.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.6 us, 10.5 sy,  0.0 ni, 86.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  6.2 us, 12.5 sy,  0.0 ni, 79.2 id,  0.0 wa,  0.0 hi,  2.1 si,  0.0 st
%Cpu(s): 10.3 us, 15.4 sy,  0.0 ni, 74.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.0 us,  9.3 sy,  0.0 ni, 83.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  0.0 us, 10.8 sy,  0.0 ni, 89.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  6.5 us, 13.0 sy,  0.0 ni, 78.3 id,  0.0 wa,  0.0 hi,  2.2 si,  0.0 st
%Cpu(s):  0.0 us,  6.1 sy,  0.0 ni, 93.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 10.4 us, 14.6 sy,  0.0 ni, 75.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.9 us,  2.9 sy,  0.0 ni, 94.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.9 us,  2.9 sy,  0.0 ni, 94.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.4 us,  5.4 sy,  0.0 ni, 89.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.9 us,  5.7 sy,  0.0 ni, 91.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.9 us,  2.9 sy,  0.0 ni, 94.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.7 us,  8.1 sy,  0.0 ni, 89.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  8.0 us, 14.0 sy,  0.0 ni, 78.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.0 us, 10.0 sy,  0.0 ni, 85.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 12.2 us, 14.6 sy,  0.0 ni, 73.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  6.8 us,  9.1 sy,  0.0 ni, 84.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
```

Listagem 25: Consumo de RAM para log de 59MB

```
MiB Mem :   3875.9 total,    478.7 free,   1544.5 used,   1852.7 buff/cache
MiB Mem :   3875.9 total,    327.1 free,   1578.4 used,   1970.4 buff/cache
```

```
MiB Mem :   3875.9 total,    311.1 free,   1593.5 used,   1971.3 buff/cache
MiB Mem :   3875.9 total,    307.9 free,   1596.4 used,   1971.5 buff/cache
MiB Mem :   3875.9 total,    306.3 free,   1597.9 used,   1971.7 buff/cache
MiB Mem :   3875.9 total,    306.3 free,   1597.9 used,   1971.8 buff/cache
MiB Mem :   3875.9 total,    305.3 free,   1598.8 used,   1971.9 buff/cache
MiB Mem :   3875.9 total,    305.0 free,   1598.8 used,   1972.0 buff/cache
MiB Mem :   3875.9 total,    306.0 free,   1597.8 used,   1972.1 buff/cache
MiB Mem :   3875.9 total,    304.8 free,   1599.0 used,   1972.1 buff/cache
MiB Mem :   3875.9 total,    304.8 free,   1598.9 used,   1972.2 buff/cache
MiB Mem :   3875.9 total,    304.6 free,   1598.8 used,   1972.5 buff/cache
MiB Mem :   3875.9 total,    303.3 free,   1600.0 used,   1972.5 buff/cache
MiB Mem :   3875.9 total,    303.3 free,   1600.0 used,   1972.6 buff/cache
MiB Mem :   3875.9 total,    303.3 free,   1599.9 used,   1972.6 buff/cache
MiB Mem :   3875.9 total,    301.9 free,   1601.3 used,   1972.7 buff/cache
MiB Mem :   3875.9 total,    302.6 free,   1600.5 used,   1972.8 buff/cache
MiB Mem :   3875.9 total,    302.6 free,   1600.4 used,   1972.8 buff/cache
MiB Mem :   3875.9 total,    302.6 free,   1600.3 used,   1972.9 buff/cache
MiB Mem :   3875.9 total,    302.6 free,   1600.3 used,   1972.9 buff/cache
MiB Mem :   3875.9 total,    303.6 free,   1599.3 used,   1973.0 buff/cache
MiB Mem :   3875.9 total,    302.4 free,   1600.5 used,   1973.0 buff/cache
MiB Mem :   3875.9 total,    302.4 free,   1600.5 used,   1973.0 buff/cache
MiB Mem :   3875.9 total,    302.4 free,   1600.4 used,   1973.1 buff/cache
MiB Mem :   3875.9 total,    302.4 free,   1600.4 used,   1973.1 buff/cache
MiB Mem :   3875.9 total,    302.4 free,   1600.4 used,   1973.1 buff/cache
MiB Mem :   3875.9 total,    302.4 free,   1600.4 used,   1973.1 buff/cache
MiB Mem :   3875.9 total,    302.2 free,   1600.6 used,   1973.1 buff/cache
MiB Mem :   3875.9 total,    302.2 free,   1600.6 used,   1973.1 buff/cache
MiB Mem :   3875.9 total,    300.7 free,   1602.1 used,   1973.1 buff/cache
MiB Mem :   3875.9 total,    300.7 free,   1602.1 used,   1973.1 buff/cache
MiB Mem :   3875.9 total,    301.2 free,   1601.6 used,   1973.1 buff/cache
MiB Mem :   3875.9 total,    300.7 free,   1602.0 used,   1973.2 buff/cache
MiB Mem :   3875.9 total,    300.7 free,   1602.0 used,   1973.2 buff/cache
MiB Mem :   3875.9 total,    300.7 free,   1602.0 used,   1973.2 buff/cache
MiB Mem :   3875.9 total,    300.7 free,   1602.0 used,   1973.2 buff/cache
MiB Mem :   3875.9 total,    432.0 free,   1588.6 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    430.7 free,   1589.8 used,   1855.3 buff/cache
MiB Mem :   3875.9 total,    430.7 free,   1589.8 used,   1855.3 buff/cache
MiB Mem :   3875.9 total,    430.7 free,   1589.8 used,   1855.3 buff/cache
MiB Mem :   3875.9 total,    430.5 free,   1590.1 used,   1855.3 buff/cache
MiB Mem :   3875.9 total,    430.5 free,   1590.1 used,   1855.3 buff/cache
MiB Mem :   3875.9 total,    430.2 free,   1590.3 used,   1855.3 buff/cache
MiB Mem :   3875.9 total,    430.2 free,   1590.3 used,   1855.3 buff/cache
MiB Mem :   3875.9 total,    430.2 free,   1590.3 used,   1855.3 buff/cache
MiB Mem :   3875.9 total,    430.0 free,   1590.6 used,   1855.3 buff/cache
MiB Mem :   3875.9 total,    429.8 free,   1590.8 used,   1855.3 buff/cache
MiB Mem :   3875.9 total,    429.5 free,   1591.1 used,   1855.3 buff/cache
MiB Mem :   3875.9 total,    429.3 free,   1591.3 used,   1855.3 buff/cache
MiB Mem :   3875.9 total,    429.0 free,   1591.5 used,   1855.3 buff/cache
MiB Mem :   3875.9 total,    429.0 free,   1591.5 used,   1855.3 buff/cache
MiB Mem :   3875.9 total,    428.8 free,   1591.8 used,   1855.3 buff/cache
MiB Mem :   3875.9 total,    428.8 free,   1591.8 used,   1855.3 buff/cache
MiB Mem :   3875.9 total,    428.8 free,   1591.8 used,   1855.3 buff/cache
MiB Mem :   3875.9 total,    428.8 free,   1591.8 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    428.3 free,   1592.3 used,   1855.4 buff/cache
```

```
MiB Mem :   3875.9 total,    428.3 free,   1592.3 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    428.0 free,   1592.5 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    428.0 free,   1592.5 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    428.0 free,   1592.5 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    428.0 free,   1592.5 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    428.0 free,   1592.5 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    428.0 free,   1592.5 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    428.0 free,   1592.5 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    428.0 free,   1592.5 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    428.0 free,   1592.5 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    428.0 free,   1592.5 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    428.0 free,   1592.5 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    428.0 free,   1592.5 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    428.0 free,   1592.5 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    428.0 free,   1592.5 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    428.0 free,   1592.5 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    428.0 free,   1592.5 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    428.0 free,   1592.5 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    428.0 free,   1592.5 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    428.0 free,   1592.5 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    428.0 free,   1592.5 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    428.0 free,   1592.5 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    428.0 free,   1592.5 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    428.0 free,   1592.5 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    428.0 free,   1592.5 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    428.0 free,   1592.5 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    428.0 free,   1592.5 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    428.0 free,   1592.5 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    428.0 free,   1592.5 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    428.0 free,   1592.5 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    428.0 free,   1592.5 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    428.0 free,   1592.5 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    428.0 free,   1592.5 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    428.0 free,   1592.5 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    428.0 free,   1592.5 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    428.0 free,   1592.5 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    428.0 free,   1592.5 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    428.0 free,   1592.5 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    427.8 free,   1592.7 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    427.5 free,   1593.0 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    427.3 free,   1593.2 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    427.1 free,   1593.5 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    426.8 free,   1593.7 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    426.3 free,   1594.2 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    426.3 free,   1594.2 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    426.3 free,   1594.2 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    426.3 free,   1594.2 used,   1855.4 buff/cache
```

Listagem 26: Consumo de CPU para log de 61MB

```
%Cpu(s):  6.5 us, 13.0 sy,  0.0 ni, 80.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 53.8 us,  7.7 sy,  0.0 ni, 38.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
```

```
%Cpu(s): 41.9 us, 11.6 sy,  0.0 ni, 44.2 id,  0.0 wa,  0.0 hi,  2.3 si,  0.0 st
%Cpu(s): 50.0 us, 10.5 sy,  0.0 ni, 34.2 id,  0.0 wa,  0.0 hi,  5.3 si,  0.0 st
%Cpu(s): 48.3 us, 22.4 sy,  0.0 ni, 27.6 id,  0.0 wa,  0.0 hi,  1.7 si,  0.0 st
%Cpu(s): 45.7 us, 21.7 sy,  0.0 ni, 30.4 id,  0.0 wa,  0.0 hi,  2.2 si,  0.0 st
%Cpu(s): 43.6 us, 10.3 sy,  0.0 ni, 43.6 id,  0.0 wa,  0.0 hi,  2.6 si,  0.0 st
%Cpu(s): 47.4 us, 18.4 sy,  0.0 ni, 34.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 44.2 us, 16.3 sy,  0.0 ni, 37.2 id,  0.0 wa,  0.0 hi,  2.3 si,  0.0 st
%Cpu(s): 50.0 us, 11.1 sy,  0.0 ni, 38.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 40.0 us, 15.6 sy,  0.0 ni, 40.0 id,  0.0 wa,  0.0 hi,  4.4 si,  0.0 st
%Cpu(s): 38.0 us, 18.0 sy,  0.0 ni, 40.0 id,  0.0 wa,  0.0 hi,  4.0 si,  0.0 st
%Cpu(s): 50.0 us, 13.6 sy,  0.0 ni, 36.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 48.5 us,  6.1 sy,  0.0 ni, 42.4 id,  0.0 wa,  0.0 hi,  3.0 si,  0.0 st
%Cpu(s): 40.5 us, 13.5 sy,  0.0 ni, 40.5 id,  0.0 wa,  0.0 hi,  5.4 si,  0.0 st
%Cpu(s): 38.5 us, 12.8 sy,  0.0 ni, 43.6 id,  0.0 wa,  0.0 hi,  5.1 si,  0.0 st
%Cpu(s): 45.0 us, 12.5 sy,  0.0 ni, 40.0 id,  0.0 wa,  0.0 hi,  2.5 si,  0.0 st
%Cpu(s): 38.1 us, 11.9 sy,  0.0 ni, 45.2 id,  0.0 wa,  0.0 hi,  4.8 si,  0.0 st
%Cpu(s): 43.6 us, 12.8 sy,  0.0 ni, 41.0 id,  0.0 wa,  0.0 hi,  2.6 si,  0.0 st
%Cpu(s): 45.0 us, 17.5 sy,  0.0 ni, 35.0 id,  0.0 wa,  0.0 hi,  2.5 si,  0.0 st
%Cpu(s): 46.9 us, 24.5 sy,  0.0 ni, 28.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 50.0 us, 13.6 sy,  0.0 ni, 34.1 id,  0.0 wa,  0.0 hi,  2.3 si,  0.0 st
%Cpu(s): 46.8 us, 17.0 sy,  0.0 ni, 36.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 39.5 us, 16.3 sy,  0.0 ni, 41.9 id,  0.0 wa,  0.0 hi,  2.3 si,  0.0 st
%Cpu(s): 41.5 us, 14.6 sy,  0.0 ni, 41.5 id,  0.0 wa,  0.0 hi,  2.4 si,  0.0 st
%Cpu(s): 45.0 us, 15.0 sy,  0.0 ni, 37.5 id,  0.0 wa,  0.0 hi,  2.5 si,  0.0 st
%Cpu(s): 48.6 us, 13.5 sy,  0.0 ni, 37.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 47.6 us, 11.9 sy,  0.0 ni, 38.1 id,  0.0 wa,  0.0 hi,  2.4 si,  0.0 st
%Cpu(s): 44.7 us, 13.2 sy,  0.0 ni, 39.5 id,  0.0 wa,  0.0 hi,  2.6 si,  0.0 st
%Cpu(s): 46.3 us, 12.2 sy,  0.0 ni, 39.0 id,  0.0 wa,  0.0 hi,  2.4 si,  0.0 st
%Cpu(s): 38.1 us, 14.3 sy,  0.0 ni, 42.9 id,  0.0 wa,  0.0 hi,  4.8 si,  0.0 st
%Cpu(s): 47.5 us, 10.0 sy,  0.0 ni, 40.0 id,  0.0 wa,  0.0 hi,  2.5 si,  0.0 st
%Cpu(s): 45.9 us, 13.5 sy,  0.0 ni, 37.8 id,  0.0 wa,  0.0 hi,  2.7 si,  0.0 st
%Cpu(s): 50.0 us, 19.6 sy,  0.0 ni, 30.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 37.8 us, 15.6 sy,  0.0 ni, 42.2 id,  0.0 wa,  0.0 hi,  4.4 si,  0.0 st
%Cpu(s): 43.2 us, 15.9 sy,  0.0 ni, 36.4 id,  0.0 wa,  0.0 hi,  4.5 si,  0.0 st
%Cpu(s): 45.5 us, 13.6 sy,  0.0 ni, 38.6 id,  0.0 wa,  0.0 hi,  2.3 si,  0.0 st
%Cpu(s): 43.2 us, 11.4 sy,  0.0 ni, 43.2 id,  0.0 wa,  0.0 hi,  2.3 si,  0.0 st
%Cpu(s): 40.0 us, 11.4 sy,  0.0 ni, 45.7 id,  0.0 wa,  0.0 hi,  2.9 si,  0.0 st
%Cpu(s): 45.2 us, 21.4 sy,  0.0 ni, 33.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  6.8 us, 11.4 sy,  0.0 ni, 81.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.0 us, 10.0 sy,  0.0 ni, 82.5 id,  0.0 wa,  0.0 hi,  2.5 si,  0.0 st
%Cpu(s): 18.9 us, 18.9 sy,  0.0 ni, 62.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.5 us, 10.0 sy,  0.0 ni, 87.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  0.0 us, 10.5 sy,  0.0 ni, 89.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  9.8 us, 14.6 sy,  0.0 ni, 75.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.5 us, 12.5 sy,  0.0 ni, 85.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  8.3 us, 12.5 sy,  0.0 ni, 79.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.6 us, 12.8 sy,  0.0 ni, 84.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  6.5 us, 13.0 sy,  0.0 ni, 80.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  9.8 us, 17.1 sy,  0.0 ni, 73.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 18.4 us, 18.4 sy,  0.0 ni, 63.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.9 us, 12.2 sy,  0.0 ni, 82.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 10.4 us, 10.4 sy,  0.0 ni, 79.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  6.5 us, 15.2 sy,  0.0 ni, 78.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 14.0 us, 20.9 sy,  0.0 ni, 65.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
```

```
%Cpu(s):  5.0 us, 10.0 sy,  0.0 ni, 85.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.6 us, 10.3 sy,  0.0 ni, 87.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  9.1 us, 13.6 sy,  0.0 ni, 77.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 17.5 us, 12.5 sy,  0.0 ni, 67.5 id,  0.0 wa,  0.0 hi,  2.5 si,  0.0 st
%Cpu(s):  4.7 us, 11.6 sy,  0.0 ni, 83.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.8 us, 11.1 sy,  0.0 ni, 86.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  0.0 us,  6.1 sy,  0.0 ni, 93.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  8.0 us, 16.0 sy,  0.0 ni, 74.0 id,  0.0 wa,  0.0 hi,  2.0 si,  0.0 st
%Cpu(s): 14.6 us, 10.4 sy,  0.0 ni, 75.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.3 us,  7.3 sy,  0.0 ni, 85.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 18.5 us, 20.4 sy,  0.0 ni, 59.3 id,  0.0 wa,  0.0 hi,  1.9 si,  0.0 st
%Cpu(s):  2.6 us, 10.3 sy,  0.0 ni, 87.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  8.5 us, 14.9 sy,  0.0 ni, 76.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.3 us,  9.8 sy,  0.0 ni, 82.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.9 us,  2.9 sy,  0.0 ni, 94.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  9.1 us, 11.4 sy,  0.0 ni, 79.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.7 us, 11.6 sy,  0.0 ni, 83.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  8.9 us, 15.6 sy,  0.0 ni, 75.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 14.3 us, 14.3 sy,  0.0 ni, 71.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.6 us,  7.9 sy,  0.0 ni, 89.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  6.5 us, 13.0 sy,  0.0 ni, 80.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  9.5 us,  7.1 sy,  0.0 ni, 83.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.3 us,  5.3 sy,  0.0 ni, 86.8 id,  0.0 wa,  0.0 hi,  2.6 si,  0.0 st
%Cpu(s): 24.4 us, 17.8 sy,  0.0 ni, 55.6 id,  0.0 wa,  0.0 hi,  2.2 si,  0.0 st
%Cpu(s):  2.6 us, 10.3 sy,  0.0 ni, 87.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.0 us,  9.3 sy,  0.0 ni, 83.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.9 us, 14.6 sy,  0.0 ni, 80.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.3 us, 13.2 sy,  0.0 ni, 81.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.9 us, 26.3 sy,  0.0 ni, 65.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  6.4 us, 14.9 sy,  0.0 ni, 78.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
```

Listagem 27: Consumo de RAM para log de 61MB

```
MiB Mem :   3875.9 total,    426.3 free,   1594.2 used,   1855.4 buff/cache
MiB Mem :   3875.9 total,    302.0 free,   1596.4 used,   1977.4 buff/cache
MiB Mem :   3875.9 total,    285.4 free,   1612.9 used,   1977.6 buff/cache
MiB Mem :   3875.9 total,    283.9 free,   1614.4 used,   1977.6 buff/cache
MiB Mem :   3875.9 total,    282.7 free,   1615.7 used,   1977.6 buff/cache
MiB Mem :   3875.9 total,    281.9 free,   1616.4 used,   1977.6 buff/cache
MiB Mem :   3875.9 total,    282.2 free,   1616.1 used,   1977.6 buff/cache
MiB Mem :   3875.9 total,    282.2 free,   1616.1 used,   1977.6 buff/cache
MiB Mem :   3875.9 total,    280.3 free,   1618.1 used,   1977.6 buff/cache
MiB Mem :   3875.9 total,    280.3 free,   1618.1 used,   1977.6 buff/cache
MiB Mem :   3875.9 total,    280.3 free,   1618.1 used,   1977.6 buff/cache
MiB Mem :   3875.9 total,    280.0 free,   1618.3 used,   1977.6 buff/cache
MiB Mem :   3875.9 total,    279.8 free,   1618.5 used,   1977.6 buff/cache
MiB Mem :   3875.9 total,    279.5 free,   1618.8 used,   1977.6 buff/cache
MiB Mem :   3875.9 total,    278.3 free,   1620.0 used,   1977.6 buff/cache
MiB Mem :   3875.9 total,    278.3 free,   1620.0 used,   1977.6 buff/cache
MiB Mem :   3875.9 total,    276.8 free,   1621.4 used,   1977.6 buff/cache
MiB Mem :   3875.9 total,    277.1 free,   1621.2 used,   1977.6 buff/cache
```

```
MiB Mem :   3875.9 total,    277.1 free,   1621.2 used,   1977.6 buff/cache
MiB Mem :   3875.9 total,    277.1 free,   1621.2 used,   1977.6 buff/cache
MiB Mem :   3875.9 total,    276.9 free,   1621.4 used,   1977.6 buff/cache
MiB Mem :   3875.9 total,    276.9 free,   1621.4 used,   1977.6 buff/cache
MiB Mem :   3875.9 total,    276.9 free,   1621.4 used,   1977.6 buff/cache
MiB Mem :   3875.9 total,    276.6 free,   1621.6 used,   1977.6 buff/cache
MiB Mem :   3875.9 total,    276.6 free,   1621.6 used,   1977.6 buff/cache
MiB Mem :   3875.9 total,    275.4 free,   1622.9 used,   1977.6 buff/cache
MiB Mem :   3875.9 total,    275.1 free,   1623.1 used,   1977.6 buff/cache
MiB Mem :   3875.9 total,    274.9 free,   1623.4 used,   1977.6 buff/cache
MiB Mem :   3875.9 total,    274.4 free,   1623.9 used,   1977.6 buff/cache
MiB Mem :   3875.9 total,    274.4 free,   1623.9 used,   1977.6 buff/cache
MiB Mem :   3875.9 total,    274.4 free,   1623.9 used,   1977.6 buff/cache
MiB Mem :   3875.9 total,    274.4 free,   1623.9 used,   1977.6 buff/cache
MiB Mem :   3875.9 total,    274.4 free,   1623.9 used,   1977.6 buff/cache
MiB Mem :   3875.9 total,    274.4 free,   1623.9 used,   1977.6 buff/cache
MiB Mem :   3875.9 total,    274.2 free,   1624.1 used,   1977.6 buff/cache
MiB Mem :   3875.9 total,    274.2 free,   1624.1 used,   1977.6 buff/cache
MiB Mem :   3875.9 total,    274.2 free,   1624.1 used,   1977.6 buff/cache
MiB Mem :   3875.9 total,    273.9 free,   1624.3 used,   1977.6 buff/cache
MiB Mem :   3875.9 total,    274.2 free,   1624.1 used,   1977.6 buff/cache
MiB Mem :   3875.9 total,    273.7 free,   1624.6 used,   1977.6 buff/cache
MiB Mem :   3875.9 total,    403.7 free,   1616.7 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    403.7 free,   1616.7 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    403.5 free,   1616.9 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    403.5 free,   1616.9 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    403.5 free,   1616.9 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    403.5 free,   1616.9 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    403.2 free,   1617.2 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    403.0 free,   1617.4 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    402.7 free,   1617.7 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    402.5 free,   1617.9 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    402.5 free,   1617.9 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    401.3 free,   1619.1 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    401.3 free,   1619.1 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    401.3 free,   1619.1 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    401.0 free,   1619.4 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    401.0 free,   1619.4 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    401.0 free,   1619.4 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    400.8 free,   1619.6 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    400.8 free,   1619.6 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    400.8 free,   1619.6 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    400.8 free,   1619.6 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    400.8 free,   1619.6 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    400.8 free,   1619.6 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    400.8 free,   1619.6 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    400.8 free,   1619.6 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    400.8 free,   1619.6 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    400.8 free,   1619.6 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    400.8 free,   1619.6 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    400.8 free,   1619.6 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    400.8 free,   1619.6 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    400.8 free,   1619.6 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    400.8 free,   1619.6 used,   1855.5 buff/cache
```

```
MiB Mem :   3875.9 total,    400.8 free,   1619.6 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    400.8 free,   1619.6 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    400.8 free,   1619.6 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    400.8 free,   1619.6 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    400.8 free,   1619.6 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    400.8 free,   1619.6 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    400.8 free,   1619.6 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    400.8 free,   1619.6 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    400.8 free,   1619.6 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    400.8 free,   1619.6 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    400.8 free,   1619.6 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    400.5 free,   1619.9 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    400.5 free,   1619.9 used,   1855.5 buff/cache
MiB Mem :   3875.9 total,    400.3 free,   1620.1 used,   1855.5 buff/cache
```

Listagem 28: Consumo de CPU para pcap de 30MB

```
%Cpu(s): 25.0 us, 16.7 sy,  0.0 ni, 58.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  8.5 us, 12.8 sy,  0.0 ni, 78.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.4 us, 15.6 sy,  0.0 ni, 80.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.7 us,  7.7 sy,  0.0 ni, 84.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.3 us,  9.8 sy,  0.0 ni, 82.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  6.7 us, 11.1 sy,  0.0 ni, 82.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  6.5 us, 13.0 sy,  0.0 ni, 80.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  6.8 us, 13.6 sy,  0.0 ni, 79.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  3.9 us, 19.6 sy,  0.0 ni, 76.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 10.2 us, 12.2 sy,  0.0 ni, 77.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.3 us, 14.0 sy,  0.0 ni, 83.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  3.9 us, 17.6 sy,  0.0 ni, 78.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 21.7 us, 19.6 sy,  0.0 ni, 58.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  6.5 us, 15.2 sy,  0.0 ni, 78.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.1 us, 18.4 sy,  0.0 ni, 77.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 10.3 us, 15.4 sy,  0.0 ni, 74.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  8.9 us,  8.9 sy,  0.0 ni, 82.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.3 us, 17.0 sy,  0.0 ni, 78.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  6.1 us, 14.3 sy,  0.0 ni, 77.6 id,  0.0 wa,  0.0 hi,  2.0 si,  0.0 st
%Cpu(s): 26.1 us, 19.6 sy,  0.0 ni, 52.2 id,  0.0 wa,  0.0 hi,  2.2 si,  0.0 st
%Cpu(s):  7.1 us, 11.9 sy,  0.0 ni, 81.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.0 us, 10.0 sy,  0.0 ni, 85.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 53.8 us, 12.8 sy,  0.0 ni, 33.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 56.1 us, 12.2 sy,  0.0 ni, 31.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 53.7 us, 12.2 sy,  0.0 ni, 34.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 50.0 us,  7.9 sy,  0.0 ni, 39.5 id,  0.0 wa,  0.0 hi,  2.6 si,  0.0 st
%Cpu(s): 57.1 us,  9.5 sy,  0.0 ni, 33.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 53.8 us, 12.8 sy,  0.0 ni, 33.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 56.1 us,  9.8 sy,  0.0 ni, 34.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 51.2 us, 12.2 sy,  0.0 ni, 36.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 51.2 us, 12.2 sy,  0.0 ni, 36.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 55.0 us, 10.0 sy,  0.0 ni, 35.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 57.5 us,  7.5 sy,  0.0 ni, 35.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 53.8 us, 10.3 sy,  0.0 ni, 35.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
```

```
%Cpu(s): 52.4 us, 11.9 sy,  0.0 ni, 35.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 55.0 us,  7.5 sy,  0.0 ni, 37.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 54.8 us, 11.9 sy,  0.0 ni, 33.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 48.8 us,  9.8 sy,  0.0 ni, 39.0 id,  0.0 wa,  0.0 hi,  2.4 si,  0.0 st
%Cpu(s): 54.1 us, 10.8 sy,  0.0 ni, 35.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 55.8 us,  9.3 sy,  0.0 ni, 34.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 56.4 us,  7.7 sy,  0.0 ni, 35.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 53.5 us, 16.3 sy,  0.0 ni, 30.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 56.1 us,  9.8 sy,  0.0 ni, 31.7 id,  0.0 wa,  0.0 hi,  2.4 si,  0.0 st
%Cpu(s): 55.0 us, 10.0 sy,  0.0 ni, 35.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 51.2 us, 17.1 sy,  0.0 ni, 31.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 50.0 us, 13.2 sy,  0.0 ni, 36.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 55.0 us, 12.5 sy,  0.0 ni, 32.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 56.8 us,  5.4 sy,  0.0 ni, 37.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 52.8 us,  8.3 sy,  0.0 ni, 38.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 56.1 us,  9.8 sy,  0.0 ni, 34.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 53.8 us, 12.8 sy,  0.0 ni, 33.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.1 us, 10.3 sy,  0.0 ni, 84.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 18.8 us, 14.6 sy,  0.0 ni, 66.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 14.0 us, 23.3 sy,  0.0 ni, 62.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 12.0 us, 10.0 sy,  0.0 ni, 78.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  9.8 us, 13.7 sy,  0.0 ni, 76.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 13.3 us, 11.1 sy,  0.0 ni, 75.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.5 us,  7.5 sy,  0.0 ni, 85.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.5 us, 13.6 sy,  0.0 ni, 81.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.0 us, 11.6 sy,  0.0 ni, 81.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 12.8 us, 17.0 sy,  0.0 ni, 68.1 id,  0.0 wa,  0.0 hi,  2.1 si,  0.0 st
%Cpu(s):  2.3 us, 14.0 sy,  0.0 ni, 83.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.9 us,  2.9 sy,  0.0 ni, 91.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  9.3 us, 18.5 sy,  0.0 ni, 70.4 id,  0.0 wa,  0.0 hi,  1.9 si,  0.0 st
%Cpu(s): 11.5 us, 11.5 sy,  0.0 ni, 76.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.3 us,  7.9 sy,  0.0 ni, 86.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.0 us,  9.3 sy,  0.0 ni, 83.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  9.8 us,  7.3 sy,  0.0 ni, 82.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  9.8 us, 12.2 sy,  0.0 ni, 78.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  8.7 us, 10.9 sy,  0.0 ni, 80.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  6.2 us, 14.6 sy,  0.0 ni, 79.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  6.8 us, 11.4 sy,  0.0 ni, 81.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 10.2 us, 18.4 sy,  0.0 ni, 71.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.4 us, 13.3 sy,  0.0 ni, 82.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.4 us,  5.4 sy,  0.0 ni, 89.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.6 us,  5.6 sy,  0.0 ni, 88.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.0 us, 17.5 sy,  0.0 ni, 73.7 id,  0.0 wa,  0.0 hi,  1.8 si,  0.0 st
%Cpu(s):  7.1 us,  9.5 sy,  0.0 ni, 83.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.5 us, 12.5 sy,  0.0 ni, 85.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
```

Listagem 29: Consumo de RAM para log de 30MB

```
MiB Mem :   3875.9 total,    474.9 free,   1548.7 used,   1852.3 buff/cache
MiB Mem :   3875.9 total,    474.9 free,   1548.7 used,   1852.3 buff/cache
MiB Mem :   3875.9 total,    474.9 free,   1548.7 used,   1852.3 buff/cache
```

```
MiB Mem :   3875.9 total,    474.7 free,    1549.0 used,    1852.3 buff/cache
MiB Mem :   3875.9 total,    474.7 free,    1549.0 used,    1852.3 buff/cache
MiB Mem :   3875.9 total,    474.7 free,    1549.0 used,    1852.3 buff/cache
MiB Mem :   3875.9 total,    474.7 free,    1549.0 used,    1852.3 buff/cache
MiB Mem :   3875.9 total,    474.7 free,    1549.0 used,    1852.3 buff/cache
MiB Mem :   3875.9 total,    474.7 free,    1549.0 used,    1852.3 buff/cache
MiB Mem :   3875.9 total,    474.7 free,    1549.0 used,    1852.3 buff/cache
MiB Mem :   3875.9 total,    474.4 free,    1549.2 used,    1852.3 buff/cache
MiB Mem :   3875.9 total,    474.4 free,    1549.2 used,    1852.3 buff/cache
MiB Mem :   3875.9 total,    474.4 free,    1549.2 used,    1852.3 buff/cache
MiB Mem :   3875.9 total,    474.4 free,    1549.2 used,    1852.3 buff/cache
MiB Mem :   3875.9 total,    474.4 free,    1549.2 used,    1852.3 buff/cache
MiB Mem :   3875.9 total,    474.4 free,    1549.2 used,    1852.3 buff/cache
MiB Mem :   3875.9 total,    474.4 free,    1549.2 used,    1852.3 buff/cache
MiB Mem :   3875.9 total,    474.4 free,    1549.2 used,    1852.3 buff/cache
MiB Mem :   3875.9 total,    474.2 free,    1549.4 used,    1852.3 buff/cache
MiB Mem :   3875.9 total,    474.2 free,    1549.4 used,    1852.3 buff/cache
MiB Mem :   3875.9 total,    474.2 free,    1549.4 used,    1852.3 buff/cache
MiB Mem :   3875.9 total,    474.2 free,    1549.4 used,    1852.3 buff/cache
MiB Mem :   3875.9 total,    339.8 free,    1624.2 used,    1911.9 buff/cache
MiB Mem :   3875.9 total,    344.9 free,    1619.1 used,    1911.9 buff/cache
MiB Mem :   3875.9 total,    340.0 free,    1624.0 used,    1911.9 buff/cache
MiB Mem :   3875.9 total,    338.1 free,    1626.0 used,    1911.9 buff/cache
MiB Mem :   3875.9 total,    334.9 free,    1629.1 used,    1911.9 buff/cache
MiB Mem :   3875.9 total,    335.9 free,    1628.2 used,    1911.9 buff/cache
MiB Mem :   3875.9 total,    335.1 free,    1628.9 used,    1911.9 buff/cache
MiB Mem :   3875.9 total,    335.4 free,    1628.6 used,    1911.9 buff/cache
MiB Mem :   3875.9 total,    332.2 free,    1631.8 used,    1911.9 buff/cache
MiB Mem :   3875.9 total,    331.0 free,    1633.1 used,    1911.9 buff/cache
MiB Mem :   3875.9 total,    329.7 free,    1634.3 used,    1911.9 buff/cache
MiB Mem :   3875.9 total,    329.2 free,    1634.8 used,    1911.9 buff/cache
MiB Mem :   3875.9 total,    327.0 free,    1637.0 used,    1911.9 buff/cache
MiB Mem :   3875.9 total,    325.1 free,    1639.0 used,    1911.9 buff/cache
MiB Mem :   3875.9 total,    323.3 free,    1640.7 used,    1911.9 buff/cache
MiB Mem :   3875.9 total,    323.1 free,    1640.9 used,    1911.9 buff/cache
MiB Mem :   3875.9 total,    321.1 free,    1642.9 used,    1911.9 buff/cache
MiB Mem :   3875.9 total,    321.1 free,    1642.9 used,    1911.9 buff/cache
MiB Mem :   3875.9 total,    321.1 free,    1642.9 used,    1911.9 buff/cache
MiB Mem :   3875.9 total,    320.1 free,    1643.9 used,    1911.9 buff/cache
MiB Mem :   3875.9 total,    319.7 free,    1644.4 used,    1911.9 buff/cache
MiB Mem :   3875.9 total,    319.2 free,    1644.8 used,    1911.9 buff/cache
MiB Mem :   3875.9 total,    319.2 free,    1644.8 used,    1911.9 buff/cache
MiB Mem :   3875.9 total,    318.2 free,    1645.8 used,    1911.9 buff/cache
MiB Mem :   3875.9 total,    315.0 free,    1649.0 used,    1911.9 buff/cache
MiB Mem :   3875.9 total,    314.0 free,    1650.0 used,    1911.9 buff/cache
MiB Mem :   3875.9 total,    314.0 free,    1650.0 used,    1911.9 buff/cache
MiB Mem :   3875.9 total,    312.8 free,    1651.2 used,    1911.9 buff/cache
MiB Mem :   3875.9 total,    312.1 free,    1651.9 used,    1911.9 buff/cache
MiB Mem :   3875.9 total,    454.2 free,    1569.4 used,    1852.3 buff/cache
MiB Mem :   3875.9 total,    454.7 free,    1568.9 used,    1852.3 buff/cache
MiB Mem :   3875.9 total,    454.7 free,    1568.9 used,    1852.3 buff/cache
MiB Mem :   3875.9 total,    454.7 free,    1568.9 used,    1852.3 buff/cache
MiB Mem :   3875.9 total,    454.7 free,    1568.9 used,    1852.3 buff/cache
MiB Mem :   3875.9 total,    454.4 free,    1569.2 used,    1852.3 buff/cache
```

```
MiB Mem :   3875.9 total,    454.4 free,   1569.2 used,   1852.3 buff/cache
MiB Mem :   3875.9 total,    455.2 free,   1568.4 used,   1852.3 buff/cache
MiB Mem :   3875.9 total,    454.9 free,   1568.7 used,   1852.3 buff/cache
MiB Mem :   3875.9 total,    454.7 free,   1568.9 used,   1852.3 buff/cache
MiB Mem :   3875.9 total,    454.7 free,   1568.9 used,   1852.3 buff/cache
MiB Mem :   3875.9 total,    454.0 free,   1569.6 used,   1852.3 buff/cache
MiB Mem :   3875.9 total,    453.7 free,   1569.9 used,   1852.3 buff/cache
MiB Mem :   3875.9 total,    453.7 free,   1569.9 used,   1852.3 buff/cache
MiB Mem :   3875.9 total,    453.5 free,   1570.1 used,   1852.3 buff/cache
MiB Mem :   3875.9 total,    453.0 free,   1570.6 used,   1852.3 buff/cache
MiB Mem :   3875.9 total,    453.0 free,   1570.6 used,   1852.3 buff/cache
MiB Mem :   3875.9 total,    452.7 free,   1570.9 used,   1852.3 buff/cache
MiB Mem :   3875.9 total,    452.7 free,   1570.9 used,   1852.3 buff/cache
MiB Mem :   3875.9 total,    452.7 free,   1570.9 used,   1852.3 buff/cache
MiB Mem :   3875.9 total,    452.7 free,   1570.9 used,   1852.3 buff/cache
MiB Mem :   3875.9 total,    452.5 free,   1571.1 used,   1852.3 buff/cache
MiB Mem :   3875.9 total,    452.2 free,   1571.3 used,   1852.3 buff/cache
MiB Mem :   3875.9 total,    452.0 free,   1571.6 used,   1852.3 buff/cache
MiB Mem :   3875.9 total,    452.0 free,   1571.6 used,   1852.3 buff/cache
MiB Mem :   3875.9 total,    452.0 free,   1571.6 used,   1852.3 buff/cache
MiB Mem :   3875.9 total,    452.0 free,   1571.6 used,   1852.3 buff/cache
MiB Mem :   3875.9 total,    451.8 free,   1571.8 used,   1852.3 buff/cache
```

Listagem 30: Consumo de CPU para pcap de 50MB

```
%Cpu(s): 19.4 us, 11.1 sy,  0.0 ni, 69.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 53.5 us, 14.0 sy,  0.0 ni, 32.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 51.2 us, 12.2 sy,  0.0 ni, 36.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 59.0 us, 10.3 sy,  0.0 ni, 30.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 50.0 us, 16.7 sy,  0.0 ni, 33.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 57.1 us,  9.5 sy,  0.0 ni, 33.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 51.4 us,  8.1 sy,  0.0 ni, 40.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 51.4 us, 10.8 sy,  0.0 ni, 37.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 54.8 us,  9.5 sy,  0.0 ni, 33.3 id,  0.0 wa,  0.0 hi,  2.4 si,  0.0 st
%Cpu(s): 53.7 us, 14.6 sy,  0.0 ni, 31.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 56.1 us,  9.8 sy,  0.0 ni, 34.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 53.7 us,  4.9 sy,  0.0 ni, 39.0 id,  0.0 wa,  0.0 hi,  2.4 si,  0.0 st
%Cpu(s): 47.5 us, 12.5 sy,  0.0 ni, 40.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 55.3 us,  7.9 sy,  0.0 ni, 36.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 58.5 us,  9.8 sy,  0.0 ni, 31.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 58.5 us,  4.9 sy,  0.0 ni, 36.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 55.0 us, 10.0 sy,  0.0 ni, 35.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 59.5 us,  2.7 sy,  0.0 ni, 37.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 56.4 us, 10.3 sy,  0.0 ni, 33.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 55.3 us,  5.3 sy,  0.0 ni, 36.8 id,  0.0 wa,  0.0 hi,  2.6 si,  0.0 st
%Cpu(s): 52.4 us, 11.9 sy,  0.0 ni, 35.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 61.5 us,  5.1 sy,  0.0 ni, 33.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 56.1 us,  9.8 sy,  0.0 ni, 34.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 57.5 us,  5.0 sy,  0.0 ni, 37.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 50.0 us, 15.9 sy,  0.0 ni, 34.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 54.5 us,  9.1 sy,  0.0 ni, 34.1 id,  0.0 wa,  0.0 hi,  2.3 si,  0.0 st
```

```
%Cpu(s): 57.1 us, 11.9 sy,  0.0 ni, 31.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 55.3 us,  7.9 sy,  0.0 ni, 36.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 61.0 us,  7.3 sy,  0.0 ni, 31.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 57.8 us,  8.9 sy,  0.0 ni, 33.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 57.5 us, 10.0 sy,  0.0 ni, 32.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  8.7 us, 17.4 sy,  0.0 ni, 73.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.8 us, 11.9 sy,  0.0 ni, 83.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.7 us,  8.1 sy,  0.0 ni, 89.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 15.0 us, 10.0 sy,  0.0 ni, 75.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.4 us,  5.4 sy,  0.0 ni, 89.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.0 us,  7.5 sy,  0.0 ni, 87.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  6.7 us, 13.3 sy,  0.0 ni, 80.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.5 us, 10.0 sy,  0.0 ni, 85.0 id,  0.0 wa,  0.0 hi,  2.5 si,  0.0 st
%Cpu(s):  9.4 us, 15.1 sy,  0.0 ni, 73.6 id,  0.0 wa,  0.0 hi,  1.9 si,  0.0 st
%Cpu(s):  7.7 us, 10.3 sy,  0.0 ni, 79.5 id,  0.0 wa,  0.0 hi,  2.6 si,  0.0 st
%Cpu(s): 11.1 us, 16.7 sy,  0.0 ni, 72.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  9.8 us, 14.6 sy,  0.0 ni, 75.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  0.0 us,  5.6 sy,  0.0 ni, 94.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.7 us, 11.6 sy,  0.0 ni, 83.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.7 us,  2.9 sy,  0.0 ni, 91.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 15.4 us, 10.3 sy,  0.0 ni, 74.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 12.2 us, 17.1 sy,  0.0 ni, 70.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.3 us, 10.5 sy,  0.0 ni, 84.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.1 us,  9.5 sy,  0.0 ni, 83.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  6.4 us, 12.8 sy,  0.0 ni, 80.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.2 us, 16.7 sy,  0.0 ni, 79.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.9 us,  5.3 sy,  0.0 ni, 86.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.0 us,  9.3 sy,  0.0 ni, 83.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  8.6 us,  2.9 sy,  0.0 ni, 88.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.5 us, 10.0 sy,  0.0 ni, 85.0 id,  0.0 wa,  0.0 hi,  2.5 si,  0.0 st
%Cpu(s):  8.7 us,  8.7 sy,  0.0 ni, 82.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.1 us,  7.7 sy,  0.0 ni, 87.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.1 us, 19.1 sy,  0.0 ni, 78.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.1 us,  7.1 sy,  0.0 ni, 85.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 51.1 us, 13.3 sy,  0.0 ni, 35.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 51.3 us, 15.4 sy,  0.0 ni, 33.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 52.5 us, 15.0 sy,  0.0 ni, 32.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 51.2 us, 14.0 sy,  0.0 ni, 34.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 56.1 us,  7.3 sy,  0.0 ni, 36.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 52.4 us, 11.9 sy,  0.0 ni, 35.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 54.3 us,  8.6 sy,  0.0 ni, 37.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 56.8 us,  8.1 sy,  0.0 ni, 35.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 59.0 us,  7.7 sy,  0.0 ni, 33.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 53.8 us, 10.3 sy,  0.0 ni, 33.3 id,  0.0 wa,  0.0 hi,  2.6 si,  0.0 st
%Cpu(s): 55.0 us, 10.0 sy,  0.0 ni, 35.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 56.1 us, 12.2 sy,  0.0 ni, 31.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 57.5 us,  7.5 sy,  0.0 ni, 35.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 55.8 us, 14.0 sy,  0.0 ni, 30.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 59.1 us,  9.1 sy,  0.0 ni, 31.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 55.0 us, 10.0 sy,  0.0 ni, 35.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 59.5 us,  5.4 sy,  0.0 ni, 32.4 id,  0.0 wa,  0.0 hi,  2.7 si,  0.0 st
%Cpu(s): 48.8 us, 19.5 sy,  0.0 ni, 31.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 48.9 us, 14.9 sy,  0.0 ni, 36.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 58.7 us, 13.0 sy,  0.0 ni, 28.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
```

```
%Cpu(s): 57.9 us,  5.3 sy,  0.0 ni, 34.2 id,  0.0 wa,  0.0 hi,  2.6 si,  0.0 st
%Cpu(s): 53.8 us,  7.7 sy,  0.0 ni, 38.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 55.6 us,  5.6 sy,  0.0 ni, 38.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 53.7 us, 12.2 sy,  0.0 ni, 34.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 58.5 us,  9.8 sy,  0.0 ni, 31.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 54.1 us,  8.1 sy,  0.0 ni, 37.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 50.0 us, 16.7 sy,  0.0 ni, 33.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 50.0 us, 14.6 sy,  0.0 ni, 33.3 id,  0.0 wa,  0.0 hi,  2.1 si,  0.0 st
%Cpu(s): 59.0 us,  7.7 sy,  0.0 ni, 33.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 57.9 us,  5.3 sy,  0.0 ni, 34.2 id,  0.0 wa,  0.0 hi,  2.6 si,  0.0 st
%Cpu(s): 60.0 us, 10.0 sy,  0.0 ni, 30.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 53.5 us, 14.0 sy,  0.0 ni, 32.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 57.1 us, 11.9 sy,  0.0 ni, 31.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 56.8 us,  8.1 sy,  0.0 ni, 35.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 53.7 us,  9.8 sy,  0.0 ni, 36.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 56.5 us, 13.0 sy,  0.0 ni, 30.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 51.2 us, 11.6 sy,  0.0 ni, 37.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 53.7 us,  9.8 sy,  0.0 ni, 36.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 10.9 us, 10.9 sy,  0.0 ni, 78.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 17.1 us, 17.1 sy,  0.0 ni, 65.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  8.9 us, 11.1 sy,  0.0 ni, 77.8 id,  0.0 wa,  0.0 hi,  2.2 si,  0.0 st
%Cpu(s):  7.7 us, 10.3 sy,  0.0 ni, 79.5 id,  0.0 wa,  0.0 hi,  2.6 si,  0.0 st
%Cpu(s):  8.5 us, 17.0 sy,  0.0 ni, 74.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.9 us, 17.1 sy,  0.0 ni, 78.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.6 us,  7.9 sy,  0.0 ni, 89.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  8.8 us,  8.8 sy,  0.0 ni, 82.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.6 us, 10.3 sy,  0.0 ni, 87.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.1 us,  7.7 sy,  0.0 ni, 87.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.6 us, 10.3 sy,  0.0 ni, 87.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  6.2 us, 14.6 sy,  0.0 ni, 79.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  6.8 us, 11.4 sy,  0.0 ni, 81.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 13.3 us, 15.6 sy,  0.0 ni, 71.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.5 us, 10.0 sy,  0.0 ni, 82.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.3 us,  7.3 sy,  0.0 ni, 85.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  9.3 us,  7.0 sy,  0.0 ni, 81.4 id,  0.0 wa,  0.0 hi,  2.3 si,  0.0 st
%Cpu(s): 18.9 us, 13.5 sy,  0.0 ni, 67.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.1 us,  7.7 sy,  0.0 ni, 84.6 id,  0.0 wa,  0.0 hi,  2.6 si,  0.0 st
%Cpu(s):  0.0 us,  5.9 sy,  0.0 ni, 94.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.4 us,  5.4 sy,  0.0 ni, 89.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.9 us, 15.8 sy,  0.0 ni, 76.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.8 us,  8.3 sy,  0.0 ni, 88.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.4 us, 12.2 sy,  0.0 ni, 85.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  6.8 us, 11.4 sy,  0.0 ni, 81.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.6 us, 10.5 sy,  0.0 ni, 86.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 12.0 us, 18.0 sy,  0.0 ni, 70.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.6 us, 10.3 sy,  0.0 ni, 87.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.3 us,  5.3 sy,  0.0 ni, 89.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.5 us, 13.6 sy,  0.0 ni, 79.5 id,  0.0 wa,  0.0 hi,  2.3 si,  0.0 st
%Cpu(s):  5.3 us,  7.9 sy,  0.0 ni, 86.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  9.5 us,  9.5 sy,  0.0 ni, 81.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.1 us, 11.9 sy,  0.0 ni, 81.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.3 us,  7.9 sy,  0.0 ni, 86.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  6.2 us, 14.6 sy,  0.0 ni, 79.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.7 us,  8.1 sy,  0.0 ni, 89.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
```

```
%Cpu(s):  5.3 us,  5.3 sy,  0.0 ni, 89.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  9.8 us, 14.6 sy,  0.0 ni, 75.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 10.0 us, 15.0 sy,  0.0 ni, 75.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.3 us, 13.0 sy,  0.0 ni, 80.4 id,  0.0 wa,  0.0 hi,  2.2 si,  0.0 st
%Cpu(s):  2.4 us, 11.9 sy,  0.0 ni, 85.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  9.5 us, 16.7 sy,  0.0 ni, 73.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.3 us, 15.2 sy,  0.0 ni, 80.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 12.8 us, 14.9 sy,  0.0 ni, 68.1 id,  2.1 wa,  0.0 hi,  2.1 si,  0.0 st
%Cpu(s):  4.9 us,  9.8 sy,  0.0 ni, 85.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 15.4 us,  7.7 sy,  0.0 ni, 76.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.1 us,  9.5 sy,  0.0 ni, 83.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
```

Listagem 31: Consumo de RAM para log de 50MB

```
MiB Mem :   3875.9 total,    472.8 free,   1551.0 used,   1852.0 buff/cache
MiB Mem :   3875.9 total,    353.3 free,   1610.8 used,   1911.7 buff/cache
MiB Mem :   3875.9 total,    352.5 free,   1611.6 used,   1911.7 buff/cache
MiB Mem :   3875.9 total,    350.9 free,   1613.3 used,   1911.7 buff/cache
MiB Mem :   3875.9 total,    347.5 free,   1616.7 used,   1911.7 buff/cache
MiB Mem :   3875.9 total,    345.3 free,   1618.8 used,   1911.7 buff/cache
MiB Mem :   3875.9 total,    339.4 free,   1624.7 used,   1911.7 buff/cache
MiB Mem :   3875.9 total,    338.9 free,   1625.2 used,   1911.7 buff/cache
MiB Mem :   3875.9 total,    337.2 free,   1627.0 used,   1911.7 buff/cache
MiB Mem :   3875.9 total,    335.2 free,   1628.9 used,   1911.7 buff/cache
MiB Mem :   3875.9 total,    333.8 free,   1630.4 used,   1911.7 buff/cache
MiB Mem :   3875.9 total,    331.3 free,   1632.9 used,   1911.7 buff/cache
MiB Mem :   3875.9 total,    331.6 free,   1632.6 used,   1911.7 buff/cache
MiB Mem :   3875.9 total,    329.9 free,   1634.3 used,   1911.7 buff/cache
MiB Mem :   3875.9 total,    327.6 free,   1636.6 used,   1911.7 buff/cache
MiB Mem :   3875.9 total,    327.4 free,   1636.8 used,   1911.7 buff/cache
MiB Mem :   3875.9 total,    327.7 free,   1636.5 used,   1911.7 buff/cache
MiB Mem :   3875.9 total,    327.7 free,   1636.5 used,   1911.7 buff/cache
MiB Mem :   3875.9 total,    326.7 free,   1637.5 used,   1911.7 buff/cache
MiB Mem :   3875.9 total,    323.3 free,   1640.9 used,   1911.7 buff/cache
MiB Mem :   3875.9 total,    326.8 free,   1637.3 used,   1911.7 buff/cache
MiB Mem :   3875.9 total,    327.5 free,   1636.6 used,   1911.7 buff/cache
MiB Mem :   3875.9 total,    328.1 free,   1636.1 used,   1911.7 buff/cache
MiB Mem :   3875.9 total,    327.1 free,   1637.1 used,   1911.7 buff/cache
MiB Mem :   3875.9 total,    326.4 free,   1637.8 used,   1911.7 buff/cache
MiB Mem :   3875.9 total,    326.4 free,   1637.8 used,   1911.7 buff/cache
MiB Mem :   3875.9 total,    324.4 free,   1639.8 used,   1911.7 buff/cache
MiB Mem :   3875.9 total,    324.4 free,   1639.7 used,   1911.7 buff/cache
MiB Mem :   3875.9 total,    323.7 free,   1640.5 used,   1911.7 buff/cache
MiB Mem :   3875.9 total,    322.2 free,   1641.9 used,   1911.7 buff/cache
MiB Mem :   3875.9 total,    322.2 free,   1641.9 used,   1911.7 buff/cache
MiB Mem :   3875.9 total,    481.7 free,   1542.1 used,   1852.1 buff/cache
MiB Mem :   3875.9 total,    480.8 free,   1543.0 used,   1852.1 buff/cache
MiB Mem :   3875.9 total,    480.8 free,   1543.0 used,   1852.1 buff/cache
MiB Mem :   3875.9 total,    480.8 free,   1543.0 used,   1852.1 buff/cache
MiB Mem :   3875.9 total,    480.8 free,   1543.0 used,   1852.1 buff/cache
MiB Mem :   3875.9 total,    480.8 free,   1543.0 used,   1852.1 buff/cache
```

```
MiB Mem :   3875.9 total,    480.8 free,   1543.0 used,   1852.1 buff/cache
MiB Mem :   3875.9 total,    480.8 free,   1543.0 used,   1852.1 buff/cache
MiB Mem :   3875.9 total,    480.5 free,   1543.3 used,   1852.1 buff/cache
MiB Mem :   3875.9 total,    480.3 free,   1543.5 used,   1852.1 buff/cache
MiB Mem :   3875.9 total,    479.6 free,   1544.2 used,   1852.1 buff/cache
MiB Mem :   3875.9 total,    479.6 free,   1544.2 used,   1852.1 buff/cache
MiB Mem :   3875.9 total,    479.6 free,   1544.2 used,   1852.1 buff/cache
MiB Mem :   3875.9 total,    479.6 free,   1544.2 used,   1852.1 buff/cache
MiB Mem :   3875.9 total,    479.6 free,   1544.2 used,   1852.1 buff/cache
MiB Mem :   3875.9 total,    479.6 free,   1544.2 used,   1852.1 buff/cache
MiB Mem :   3875.9 total,    479.6 free,   1544.2 used,   1852.1 buff/cache
MiB Mem :   3875.9 total,    479.6 free,   1544.2 used,   1852.1 buff/cache
MiB Mem :   3875.9 total,    479.6 free,   1544.2 used,   1852.1 buff/cache
MiB Mem :   3875.9 total,    479.6 free,   1544.2 used,   1852.1 buff/cache
MiB Mem :   3875.9 total,    479.6 free,   1544.2 used,   1852.1 buff/cache
MiB Mem :   3875.9 total,    479.3 free,   1544.5 used,   1852.1 buff/cache
MiB Mem :   3875.9 total,    479.3 free,   1544.5 used,   1852.1 buff/cache
MiB Mem :   3875.9 total,    479.3 free,   1544.5 used,   1852.1 buff/cache
MiB Mem :   3875.9 total,    479.3 free,   1544.5 used,   1852.1 buff/cache
MiB Mem :   3875.9 total,    479.3 free,   1544.5 used,   1852.1 buff/cache
MiB Mem :   3875.9 total,    479.3 free,   1544.5 used,   1852.1 buff/cache
MiB Mem :   3875.9 total,    478.4 free,   1545.4 used,   1852.1 buff/cache
MiB Mem :   3875.9 total,    478.4 free,   1545.4 used,   1852.1 buff/cache
MiB Mem :   3875.9 total,    311.6 free,   1612.4 used,   1951.9 buff/cache
MiB Mem :   3875.9 total,    307.9 free,   1616.0 used,   1951.9 buff/cache
MiB Mem :   3875.9 total,    307.5 free,   1616.5 used,   1951.9 buff/cache
MiB Mem :   3875.9 total,    303.8 free,   1620.2 used,   1951.9 buff/cache
MiB Mem :   3875.9 total,    303.8 free,   1620.2 used,   1951.9 buff/cache
MiB Mem :   3875.9 total,    303.3 free,   1620.7 used,   1951.9 buff/cache
MiB Mem :   3875.9 total,    302.8 free,   1621.2 used,   1951.9 buff/cache
MiB Mem :   3875.9 total,    301.8 free,   1622.1 used,   1951.9 buff/cache
MiB Mem :   3875.9 total,    300.1 free,   1623.9 used,   1951.9 buff/cache
MiB Mem :   3875.9 total,    299.9 free,   1624.1 used,   1951.9 buff/cache
MiB Mem :   3875.9 total,    298.4 free,   1625.6 used,   1951.9 buff/cache
MiB Mem :   3875.9 total,    298.9 free,   1625.1 used,   1951.9 buff/cache
MiB Mem :   3875.9 total,    295.9 free,   1628.0 used,   1951.9 buff/cache
MiB Mem :   3875.9 total,    297.8 free,   1626.1 used,   1951.9 buff/cache
MiB Mem :   3875.9 total,    299.3 free,   1624.7 used,   1951.9 buff/cache
MiB Mem :   3875.9 total,    300.6 free,   1623.3 used,   1951.9 buff/cache
MiB Mem :   3875.9 total,    298.9 free,   1625.1 used,   1951.9 buff/cache
MiB Mem :   3875.9 total,    294.2 free,   1629.8 used,   1951.9 buff/cache
MiB Mem :   3875.9 total,    294.0 free,   1630.0 used,   1951.9 buff/cache
MiB Mem :   3875.9 total,    294.2 free,   1629.7 used,   1951.9 buff/cache
MiB Mem :   3875.9 total,    293.7 free,   1630.2 used,   1951.9 buff/cache
MiB Mem :   3875.9 total,    289.5 free,   1634.4 used,   1951.9 buff/cache
MiB Mem :   3875.9 total,    292.0 free,   1631.9 used,   1951.9 buff/cache
MiB Mem :   3875.9 total,    289.6 free,   1634.4 used,   1951.9 buff/cache
MiB Mem :   3875.9 total,    288.9 free,   1635.1 used,   1951.9 buff/cache
MiB Mem :   3875.9 total,    287.6 free,   1636.3 used,   1951.9 buff/cache
MiB Mem :   3875.9 total,    287.4 free,   1636.6 used,   1951.9 buff/cache
MiB Mem :   3875.9 total,    286.4 free,   1637.5 used,   1951.9 buff/cache
MiB Mem :   3875.9 total,    287.6 free,   1636.4 used,   1951.9 buff/cache
MiB Mem :   3875.9 total,    284.0 free,   1640.0 used,   1951.9 buff/cache
MiB Mem :   3875.9 total,    283.5 free,   1640.5 used,   1951.9 buff/cache
```

```
92  MiB Mem :   3875.9 total,    282.7 free,   1641.2 used,   1951.9 buff/cache
93  MiB Mem :   3875.9 total,    282.0 free,   1642.0 used,   1951.9 buff/cache
94  MiB Mem :   3875.9 total,    277.6 free,   1646.4 used,   1951.9 buff/cache
95  MiB Mem :   3875.9 total,    281.0 free,   1642.9 used,   1952.0 buff/cache
96  MiB Mem :   3875.9 total,    280.3 free,   1643.7 used,   1952.0 buff/cache
97  MiB Mem :   3875.9 total,    278.6 free,   1645.4 used,   1952.0 buff/cache
98  MiB Mem :   3875.9 total,    280.2 free,   1643.7 used,   1952.0 buff/cache
99  MiB Mem :   3875.9 total,    479.0 free,   1544.7 used,   1852.2 buff/cache
100 MiB Mem :   3875.9 total,    479.0 free,   1544.7 used,   1852.2 buff/cache
101 MiB Mem :   3875.9 total,    479.0 free,   1544.7 used,   1852.2 buff/cache
102 MiB Mem :   3875.9 total,    479.0 free,   1544.7 used,   1852.2 buff/cache
103 MiB Mem :   3875.9 total,    479.0 free,   1544.7 used,   1852.2 buff/cache
104 MiB Mem :   3875.9 total,    479.0 free,   1544.7 used,   1852.2 buff/cache
105 MiB Mem :   3875.9 total,    479.0 free,   1544.7 used,   1852.2 buff/cache
106 MiB Mem :   3875.9 total,    478.3 free,   1545.3 used,   1852.2 buff/cache
107 MiB Mem :   3875.9 total,    478.3 free,   1545.3 used,   1852.2 buff/cache
108 MiB Mem :   3875.9 total,    478.3 free,   1545.3 used,   1852.2 buff/cache
109 MiB Mem :   3875.9 total,    478.1 free,   1545.6 used,   1852.2 buff/cache
110 MiB Mem :   3875.9 total,    478.1 free,   1545.6 used,   1852.2 buff/cache
111 MiB Mem :   3875.9 total,    478.1 free,   1545.6 used,   1852.2 buff/cache
112 MiB Mem :   3875.9 total,    478.1 free,   1545.6 used,   1852.2 buff/cache
113 MiB Mem :   3875.9 total,    478.1 free,   1545.6 used,   1852.2 buff/cache
114 MiB Mem :   3875.9 total,    478.1 free,   1545.6 used,   1852.2 buff/cache
115 MiB Mem :   3875.9 total,    478.1 free,   1545.6 used,   1852.2 buff/cache
116 MiB Mem :   3875.9 total,    477.8 free,   1545.8 used,   1852.2 buff/cache
117 MiB Mem :   3875.9 total,    477.8 free,   1545.8 used,   1852.2 buff/cache
118 MiB Mem :   3875.9 total,    477.8 free,   1545.8 used,   1852.2 buff/cache
119 MiB Mem :   3875.9 total,    477.8 free,   1545.8 used,   1852.2 buff/cache
120 MiB Mem :   3875.9 total,    477.8 free,   1545.8 used,   1852.2 buff/cache
121 MiB Mem :   3875.9 total,    477.8 free,   1545.8 used,   1852.2 buff/cache
122 MiB Mem :   3875.9 total,    477.8 free,   1545.8 used,   1852.2 buff/cache
123 MiB Mem :   3875.9 total,    477.8 free,   1545.8 used,   1852.2 buff/cache
124 MiB Mem :   3875.9 total,    477.8 free,   1545.8 used,   1852.2 buff/cache
125 MiB Mem :   3875.9 total,    477.8 free,   1545.8 used,   1852.2 buff/cache
126 MiB Mem :   3875.9 total,    477.8 free,   1545.8 used,   1852.2 buff/cache
127 MiB Mem :   3875.9 total,    477.6 free,   1546.1 used,   1852.2 buff/cache
128 MiB Mem :   3875.9 total,    477.6 free,   1546.1 used,   1852.2 buff/cache
129 MiB Mem :   3875.9 total,    477.6 free,   1546.1 used,   1852.2 buff/cache
130 MiB Mem :   3875.9 total,    477.6 free,   1546.1 used,   1852.2 buff/cache
131 MiB Mem :   3875.9 total,    477.6 free,   1546.1 used,   1852.2 buff/cache
132 MiB Mem :   3875.9 total,    477.6 free,   1546.1 used,   1852.2 buff/cache
133 MiB Mem :   3875.9 total,    477.6 free,   1546.1 used,   1852.2 buff/cache
134 MiB Mem :   3875.9 total,    477.6 free,   1546.1 used,   1852.2 buff/cache
135 MiB Mem :   3875.9 total,    477.6 free,   1546.1 used,   1852.2 buff/cache
136 MiB Mem :   3875.9 total,    477.3 free,   1546.3 used,   1852.2 buff/cache
137 MiB Mem :   3875.9 total,    477.1 free,   1546.6 used,   1852.2 buff/cache
138 MiB Mem :   3875.9 total,    477.1 free,   1546.6 used,   1852.2 buff/cache
139 MiB Mem :   3875.9 total,    477.1 free,   1546.6 used,   1852.2 buff/cache
140 MiB Mem :   3875.9 total,    476.9 free,   1546.8 used,   1852.2 buff/cache
141 MiB Mem :   3875.9 total,    476.9 free,   1546.8 used,   1852.2 buff/cache
142 MiB Mem :   3875.9 total,    476.6 free,   1547.0 used,   1852.2 buff/cache
143 MiB Mem :   3875.9 total,    476.6 free,   1547.0 used,   1852.2 buff/cache
```

```
MiB Mem :   3875.9 total,    476.4 free,   1547.3 used,   1852.2 buff/cache
MiB Mem :   3875.9 total,    476.1 free,   1547.5 used,   1852.2 buff/cache
```

Listagem 32: Consumo de CPU para pcap de 100MB

```
%Cpu(s):  7.0 us,  9.3 sy,  0.0 ni, 83.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 34.1 us, 29.5 sy,  0.0 ni, 20.5 id,  0.0 wa,  0.0 hi, 15.9 si,  0.0 st
%Cpu(s): 59.6 us, 15.4 sy,  0.0 ni, 25.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 53.7 us, 12.2 sy,  0.0 ni, 34.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 51.2 us, 12.2 sy,  0.0 ni, 36.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 54.1 us, 16.2 sy,  0.0 ni, 29.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 52.5 us, 12.5 sy,  0.0 ni, 35.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 54.5 us,  6.8 sy,  0.0 ni, 38.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 58.5 us,  9.8 sy,  0.0 ni, 31.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 57.1 us,  7.1 sy,  0.0 ni, 35.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 54.8 us, 11.9 sy,  0.0 ni, 33.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 55.0 us, 10.0 sy,  0.0 ni, 35.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 54.5 us, 15.9 sy,  0.0 ni, 29.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 53.8 us, 10.3 sy,  0.0 ni, 35.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 57.1 us,  5.7 sy,  0.0 ni, 37.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 52.6 us, 10.5 sy,  0.0 ni, 36.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 58.5 us,  4.9 sy,  0.0 ni, 34.1 id,  0.0 wa,  0.0 hi,  2.4 si,  0.0 st
%Cpu(s): 56.4 us,  7.7 sy,  0.0 ni, 35.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 55.0 us,  7.5 sy,  0.0 ni, 35.0 id,  0.0 wa,  0.0 hi,  2.5 si,  0.0 st
%Cpu(s): 55.8 us,  9.3 sy,  0.0 ni, 34.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 56.8 us, 11.4 sy,  0.0 ni, 31.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 55.8 us,  9.3 sy,  0.0 ni, 32.6 id,  0.0 wa,  0.0 hi,  2.3 si,  0.0 st
%Cpu(s): 58.7 us, 13.0 sy,  0.0 ni, 28.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 57.5 us, 10.0 sy,  0.0 ni, 32.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 53.7 us, 14.6 sy,  0.0 ni, 31.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 59.1 us,  9.1 sy,  0.0 ni, 31.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 52.5 us, 12.5 sy,  0.0 ni, 35.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 54.3 us, 13.0 sy,  0.0 ni, 30.4 id,  0.0 wa,  0.0 hi,  2.2 si,  0.0 st
%Cpu(s): 53.8 us, 12.8 sy,  0.0 ni, 30.8 id,  0.0 wa,  0.0 hi,  2.6 si,  0.0 st
%Cpu(s): 53.8 us, 10.3 sy,  0.0 ni, 33.3 id,  0.0 wa,  0.0 hi,  2.6 si,  0.0 st
%Cpu(s): 57.1 us,  7.1 sy,  0.0 ni, 35.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 52.1 us, 16.7 sy,  0.0 ni, 31.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 55.8 us, 14.0 sy,  0.0 ni, 30.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 57.1 us,  7.1 sy,  0.0 ni, 35.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 52.4 us, 11.9 sy,  0.0 ni, 33.3 id,  0.0 wa,  0.0 hi,  2.4 si,  0.0 st
%Cpu(s): 53.5 us, 11.6 sy,  0.0 ni, 34.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 53.7 us, 12.2 sy,  0.0 ni, 31.7 id,  0.0 wa,  0.0 hi,  2.4 si,  0.0 st
%Cpu(s): 59.0 us,  7.7 sy,  0.0 ni, 33.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 57.1 us,  9.5 sy,  0.0 ni, 33.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 53.7 us, 14.6 sy,  0.0 ni, 31.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 52.5 us, 15.0 sy,  0.0 ni, 32.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 55.8 us, 16.3 sy,  0.0 ni, 27.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 47.4 us, 15.8 sy,  0.0 ni, 36.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 56.1 us,  9.8 sy,  0.0 ni, 34.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 52.0 us, 22.0 sy,  0.0 ni, 26.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 57.1 us,  9.5 sy,  0.0 ni, 33.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
```

```
%Cpu(s): 60.5 us,  9.3 sy,  0.0 ni, 30.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 58.3 us, 12.5 sy,  0.0 ni, 29.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 60.5 us,  9.3 sy,  0.0 ni, 30.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 57.5 us, 10.0 sy,  0.0 ni, 32.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 52.8 us,  8.3 sy,  0.0 ni, 38.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 50.0 us, 16.7 sy,  0.0 ni, 31.0 id,  0.0 wa,  0.0 hi,  2.4 si,  0.0 st
%Cpu(s): 49.0 us, 19.6 sy,  0.0 ni, 31.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 54.3 us, 15.2 sy,  0.0 ni, 30.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 55.3 us, 10.5 sy,  0.0 ni, 34.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 53.8 us, 12.8 sy,  0.0 ni, 30.8 id,  0.0 wa,  0.0 hi,  2.6 si,  0.0 st
%Cpu(s): 59.0 us, 10.3 sy,  0.0 ni, 30.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 57.5 us,  5.0 sy,  0.0 ni, 37.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 52.2 us, 15.2 sy,  0.0 ni, 32.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 56.8 us, 11.4 sy,  0.0 ni, 31.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 55.8 us,  9.3 sy,  0.0 ni, 34.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 15.2 us, 15.2 sy,  0.0 ni, 69.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 18.9 us, 18.9 sy,  0.0 ni, 62.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.9 us,  2.6 sy,  0.0 ni, 86.8 id,  0.0 wa,  0.0 hi,  2.6 si,  0.0 st
%Cpu(s): 21.3 us, 27.7 sy,  0.0 ni, 51.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.1 us,  9.5 sy,  0.0 ni, 83.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.8 us, 11.9 sy,  0.0 ni, 83.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.5 us, 13.6 sy,  0.0 ni, 81.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  6.5 us, 13.0 sy,  0.0 ni, 78.3 id,  0.0 wa,  0.0 hi,  2.2 si,  0.0 st
%Cpu(s): 25.0 us, 22.9 sy,  0.0 ni, 52.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.3 us,  7.9 sy,  0.0 ni, 86.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  8.5 us, 10.6 sy,  0.0 ni, 80.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.8 us, 19.6 sy,  0.0 ni, 72.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.7 us, 16.3 sy,  0.0 ni, 79.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.2 us, 16.7 sy,  0.0 ni, 79.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 19.6 us, 15.2 sy,  0.0 ni, 63.0 id,  0.0 wa,  0.0 hi,  2.2 si,  0.0 st
%Cpu(s):  4.0 us, 18.0 sy,  0.0 ni, 78.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 25.0 us, 13.6 sy,  0.0 ni, 61.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  8.1 us,  5.4 sy,  0.0 ni, 86.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 10.2 us, 14.3 sy,  0.0 ni, 75.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.9 us,  5.9 sy,  0.0 ni, 91.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 22.5 us, 15.0 sy,  0.0 ni, 62.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  9.6 us, 11.5 sy,  0.0 ni, 75.0 id,  0.0 wa,  0.0 hi,  3.8 si,  0.0 st
%Cpu(s):  9.6 us, 13.5 sy,  0.0 ni, 75.0 id,  0.0 wa,  0.0 hi,  1.9 si,  0.0 st
%Cpu(s): 17.0 us, 22.6 sy,  0.0 ni, 60.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  3.0 us,  6.1 sy,  0.0 ni, 90.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
```

Listagem 33: Consumo de RAM para log de 100MB

```
MiB Mem :   3875.9 total,    451.8 free,   1571.8 used,   1852.3 buff/cache
MiB Mem :   3875.9 total,    429.0 free,   1562.0 used,   1884.9 buff/cache
MiB Mem :   3875.9 total,    186.0 free,   1637.5 used,   2052.4 buff/cache
MiB Mem :   3875.9 total,    197.6 free,   1625.9 used,   2052.4 buff/cache
MiB Mem :   3875.9 total,    195.8 free,   1627.6 used,   2052.4 buff/cache
MiB Mem :   3875.9 total,    195.6 free,   1627.9 used,   2052.4 buff/cache
MiB Mem :   3875.9 total,    195.4 free,   1628.1 used,   2052.4 buff/cache
MiB Mem :   3875.9 total,    186.9 free,   1636.6 used,   2052.4 buff/cache
```

```
MiB Mem :   3875.9 total,    185.0 free,   1638.4 used,   2052.4 buff/cache
MiB Mem :   3875.9 total,    184.5 free,   1638.9 used,   2052.4 buff/cache
MiB Mem :   3875.9 total,    189.9 free,   1633.6 used,   2052.4 buff/cache
MiB Mem :   3875.9 total,    189.3 free,   1634.1 used,   2052.4 buff/cache
MiB Mem :   3875.9 total,    190.4 free,   1633.0 used,   2052.4 buff/cache
MiB Mem :   3875.9 total,    193.1 free,   1630.3 used,   2052.4 buff/cache
MiB Mem :   3875.9 total,    192.8 free,   1630.6 used,   2052.4 buff/cache
MiB Mem :   3875.9 total,    193.1 free,   1630.3 used,   2052.4 buff/cache
MiB Mem :   3875.9 total,    191.4 free,   1632.0 used,   2052.5 buff/cache
MiB Mem :   3875.9 total,    193.2 free,   1630.2 used,   2052.5 buff/cache
MiB Mem :   3875.9 total,    191.0 free,   1632.5 used,   2052.5 buff/cache
MiB Mem :   3875.9 total,    189.5 free,   1633.9 used,   2052.5 buff/cache
MiB Mem :   3875.9 total,    189.3 free,   1634.1 used,   2052.5 buff/cache
MiB Mem :   3875.9 total,    191.0 free,   1632.4 used,   2052.5 buff/cache
MiB Mem :   3875.9 total,    190.5 free,   1632.9 used,   2052.5 buff/cache
MiB Mem :   3875.9 total,    190.5 free,   1632.9 used,   2052.5 buff/cache
MiB Mem :   3875.9 total,    190.5 free,   1632.9 used,   2052.5 buff/cache
MiB Mem :   3875.9 total,    195.5 free,   1628.0 used,   2052.5 buff/cache
MiB Mem :   3875.9 total,    185.1 free,   1638.3 used,   2052.5 buff/cache
MiB Mem :   3875.9 total,    186.4 free,   1637.1 used,   2052.5 buff/cache
MiB Mem :   3875.9 total,    186.8 free,   1636.6 used,   2052.5 buff/cache
MiB Mem :   3875.9 total,    185.1 free,   1638.3 used,   2052.5 buff/cache
MiB Mem :   3875.9 total,    191.5 free,   1631.9 used,   2052.5 buff/cache
MiB Mem :   3875.9 total,    183.9 free,   1639.5 used,   2052.5 buff/cache
MiB Mem :   3875.9 total,    183.9 free,   1639.6 used,   2052.5 buff/cache
MiB Mem :   3875.9 total,    181.4 free,   1642.0 used,   2052.5 buff/cache
MiB Mem :   3875.9 total,    178.3 free,   1645.1 used,   2052.5 buff/cache
MiB Mem :   3875.9 total,    179.3 free,   1644.1 used,   2052.5 buff/cache
MiB Mem :   3875.9 total,    177.3 free,   1646.0 used,   2052.5 buff/cache
MiB Mem :   3875.9 total,    173.5 free,   1649.8 used,   2052.5 buff/cache
MiB Mem :   3875.9 total,    173.4 free,   1650.0 used,   2052.5 buff/cache
MiB Mem :   3875.9 total,    170.4 free,   1652.9 used,   2052.5 buff/cache
MiB Mem :   3875.9 total,    167.5 free,   1655.9 used,   2052.5 buff/cache
MiB Mem :   3875.9 total,    168.0 free,   1655.4 used,   2052.5 buff/cache
MiB Mem :   3875.9 total,    165.3 free,   1658.1 used,   2052.5 buff/cache
MiB Mem :   3875.9 total,    164.0 free,   1659.3 used,   2052.5 buff/cache
MiB Mem :   3875.9 total,    162.6 free,   1660.8 used,   2052.5 buff/cache
MiB Mem :   3875.9 total,    160.1 free,   1663.2 used,   2052.6 buff/cache
MiB Mem :   3875.9 total,    164.8 free,   1658.6 used,   2052.6 buff/cache
MiB Mem :   3875.9 total,    159.9 free,   1663.5 used,   2052.6 buff/cache
MiB Mem :   3875.9 total,    157.1 free,   1666.3 used,   2052.5 buff/cache
MiB Mem :   3875.9 total,    159.9 free,   1663.5 used,   2052.5 buff/cache
MiB Mem :   3875.9 total,    165.0 free,   1658.3 used,   2052.5 buff/cache
MiB Mem :   3875.9 total,    154.7 free,   1668.7 used,   2052.5 buff/cache
MiB Mem :   3875.9 total,    158.4 free,   1664.9 used,   2052.5 buff/cache
MiB Mem :   3875.9 total,    149.1 free,   1674.3 used,   2052.5 buff/cache
MiB Mem :   3875.9 total,    147.6 free,   1675.8 used,   2052.6 buff/cache
MiB Mem :   3875.9 total,    148.3 free,   1675.0 used,   2052.6 buff/cache
MiB Mem :   3875.9 total,    143.9 free,   1679.4 used,   2052.6 buff/cache
MiB Mem :   3875.9 total,    145.2 free,   1678.2 used,   2052.6 buff/cache
MiB Mem :   3875.9 total,    142.7 free,   1680.6 used,   2052.6 buff/cache
MiB Mem :   3875.9 total,    141.7 free,   1681.6 used,   2052.6 buff/cache
MiB Mem :   3875.9 total,    142.1 free,   1681.2 used,   2052.6 buff/cache
MiB Mem :   3875.9 total,    484.0 free,   1539.2 used,   1852.7 buff/cache
```

```
MiB Mem :   3875.9 total,    484.0 free,   1539.2 used,   1852.7 buff/cache
MiB Mem :   3875.9 total,    484.0 free,   1539.2 used,   1852.7 buff/cache
MiB Mem :   3875.9 total,    484.0 free,   1539.2 used,   1852.7 buff/cache
MiB Mem :   3875.9 total,    484.0 free,   1539.2 used,   1852.7 buff/cache
MiB Mem :   3875.9 total,    484.0 free,   1539.2 used,   1852.7 buff/cache
MiB Mem :   3875.9 total,    484.0 free,   1539.2 used,   1852.7 buff/cache
MiB Mem :   3875.9 total,    484.0 free,   1539.2 used,   1852.7 buff/cache
MiB Mem :   3875.9 total,    483.6 free,   1539.6 used,   1852.7 buff/cache
MiB Mem :   3875.9 total,    483.6 free,   1539.6 used,   1852.7 buff/cache
MiB Mem :   3875.9 total,    483.3 free,   1539.9 used,   1852.7 buff/cache
MiB Mem :   3875.9 total,    483.1 free,   1540.1 used,   1852.7 buff/cache
MiB Mem :   3875.9 total,    482.8 free,   1540.4 used,   1852.7 buff/cache
MiB Mem :   3875.9 total,    482.6 free,   1540.6 used,   1852.7 buff/cache
MiB Mem :   3875.9 total,    482.4 free,   1540.9 used,   1852.7 buff/cache
MiB Mem :   3875.9 total,    482.1 free,   1541.1 used,   1852.7 buff/cache
MiB Mem :   3875.9 total,    482.1 free,   1541.1 used,   1852.7 buff/cache
MiB Mem :   3875.9 total,    481.6 free,   1541.6 used,   1852.7 buff/cache
MiB Mem :   3875.9 total,    480.9 free,   1542.4 used,   1852.7 buff/cache
MiB Mem :   3875.9 total,    480.9 free,   1542.4 used,   1852.7 buff/cache
MiB Mem :   3875.9 total,    480.9 free,   1542.4 used,   1852.7 buff/cache
MiB Mem :   3875.9 total,    480.9 free,   1542.4 used,   1852.7 buff/cache
MiB Mem :   3875.9 total,    479.9 free,   1543.3 used,   1852.7 buff/cache
MiB Mem :   3875.9 total,    479.9 free,   1543.3 used,   1852.7 buff/cache
MiB Mem :   3875.9 total,    479.9 free,   1543.3 used,   1852.7 buff/cache
```

Listagem 34: Consumo de CPU para *rsyslog*

```
%Cpu(s):  4.4 us, 15.6 sy,  0.0 ni, 80.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 22.9 us, 22.9 sy,  0.0 ni, 54.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.4 us, 14.6 sy,  0.0 ni, 82.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 23.7 us, 55.3 sy,  0.0 ni, 21.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 14.0 us, 14.0 sy,  0.0 ni, 72.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.1 us,  7.7 sy,  0.0 ni, 87.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 10.2 us, 12.2 sy,  0.0 ni, 75.5 id,  0.0 wa,  0.0 hi,  2.0 si,  0.0 st
%Cpu(s):  5.6 us,  5.6 sy,  0.0 ni, 88.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  0.0 us,  8.3 sy,  0.0 ni, 91.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 19.1 us, 23.4 sy,  0.0 ni, 57.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  6.7 us, 13.3 sy,  0.0 ni, 80.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.7 us, 11.6 sy,  0.0 ni, 83.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 11.1 us, 15.6 sy,  0.0 ni, 73.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.2 us, 16.7 sy,  0.0 ni, 79.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 16.3 us, 18.6 sy,  0.0 ni, 62.8 id,  0.0 wa,  0.0 hi,  2.3 si,  0.0 st
%Cpu(s):  7.7 us, 17.3 sy,  0.0 ni, 75.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.9 us,  9.8 sy,  0.0 ni, 85.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 21.4 us, 21.4 sy,  0.0 ni, 57.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 10.6 us, 12.8 sy,  0.0 ni, 76.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 15.1 us, 15.1 sy,  0.0 ni, 67.9 id,  0.0 wa,  0.0 hi,  1.9 si,  0.0 st
%Cpu(s):  8.9 us, 11.1 sy,  0.0 ni, 80.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 10.4 us, 10.4 sy,  0.0 ni, 79.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.4 us, 15.6 sy,  0.0 ni, 80.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  6.5 us, 13.0 sy,  0.0 ni, 80.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
```

```
%Cpu(s): 12.5 us, 12.5 sy,  0.0 ni, 75.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  8.9 us, 11.1 sy,  0.0 ni, 77.8 id,  0.0 wa,  0.0 hi,  2.2 si,  0.0 st
%Cpu(s): 23.5 us, 19.6 sy,  0.0 ni, 54.9 id,  0.0 wa,  0.0 hi,  2.0 si,  0.0 st
%Cpu(s):  2.3 us, 14.0 sy,  0.0 ni, 83.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 10.2 us, 12.2 sy,  0.0 ni, 77.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 10.8 us, 10.8 sy,  0.0 ni, 78.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.5 us, 13.6 sy,  0.0 ni, 81.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.9 us, 12.2 sy,  0.0 ni, 82.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 10.9 us, 13.0 sy,  0.0 ni, 76.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.7 us,  5.7 sy,  0.0 ni, 88.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 13.6 us, 13.6 sy,  0.0 ni, 72.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  8.9 us, 13.3 sy,  0.0 ni, 77.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.8 us, 11.9 sy,  0.0 ni, 83.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 23.1 us, 19.2 sy,  0.0 ni, 51.9 id,  0.0 wa,  0.0 hi,  5.8 si,  0.0 st
%Cpu(s):  8.3 us, 12.5 sy,  0.0 ni, 77.1 id,  0.0 wa,  0.0 hi,  2.1 si,  0.0 st
%Cpu(s): 34.2 us, 23.7 sy,  0.0 ni, 42.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 24.0 us, 20.0 sy,  0.0 ni, 54.0 id,  0.0 wa,  0.0 hi,  2.0 si,  0.0 st
%Cpu(s):  9.1 us,  9.1 sy,  0.0 ni, 81.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 17.9 us,  7.7 sy,  0.0 ni, 74.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  8.9 us, 16.1 sy,  0.0 ni, 73.2 id,  0.0 wa,  0.0 hi,  1.8 si,  0.0 st
%Cpu(s): 20.8 us, 22.6 sy,  0.0 ni, 54.7 id,  0.0 wa,  0.0 hi,  1.9 si,  0.0 st
%Cpu(s):  5.9 us, 15.7 sy,  0.0 ni, 76.5 id,  0.0 wa,  0.0 hi,  2.0 si,  0.0 st
%Cpu(s):  6.2 us, 14.6 sy,  0.0 ni, 77.1 id,  0.0 wa,  0.0 hi,  2.1 si,  0.0 st
%Cpu(s):  5.1 us, 10.3 sy,  0.0 ni, 84.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  6.4 us, 12.8 sy,  0.0 ni, 80.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 16.7 us, 16.7 sy,  0.0 ni, 66.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  8.8 us, 17.5 sy,  0.0 ni, 70.2 id,  0.0 wa,  0.0 hi,  3.5 si,  0.0 st
%Cpu(s): 15.2 us, 17.4 sy,  0.0 ni, 65.2 id,  0.0 wa,  0.0 hi,  2.2 si,  0.0 st
%Cpu(s):  3.9 us, 19.6 sy,  0.0 ni, 74.5 id,  0.0 wa,  0.0 hi,  2.0 si,  0.0 st
%Cpu(s):  2.2 us, 17.4 sy,  0.0 ni, 80.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.3 us, 17.1 sy,  0.0 ni, 73.2 id,  0.0 wa,  0.0 hi,  2.4 si,  0.0 st
%Cpu(s):  4.4 us, 13.3 sy,  0.0 ni, 82.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.0 us, 11.6 sy,  0.0 ni, 81.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  8.9 us, 11.1 sy,  0.0 ni, 80.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  6.8 us, 11.4 sy,  0.0 ni, 81.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 13.6 us, 20.3 sy,  0.0 ni, 66.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.4 us, 14.3 sy,  0.0 ni, 83.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 16.7 us, 19.0 sy,  0.0 ni, 61.9 id,  0.0 wa,  0.0 hi,  2.4 si,  0.0 st
%Cpu(s):  8.9 us, 13.3 sy,  0.0 ni, 77.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.0 us, 10.0 sy,  0.0 ni, 85.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 25.5 us, 19.6 sy,  0.0 ni, 54.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  6.5 us, 15.2 sy,  0.0 ni, 78.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 12.2 us, 12.2 sy,  0.0 ni, 75.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 13.2 us, 15.1 sy,  0.0 ni, 69.8 id,  0.0 wa,  0.0 hi,  1.9 si,  0.0 st
%Cpu(s):  4.3 us, 17.0 sy,  0.0 ni, 78.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 12.8 us, 10.3 sy,  0.0 ni, 74.4 id,  0.0 wa,  0.0 hi,  2.6 si,  0.0 st
%Cpu(s):  5.3 us,  7.9 sy,  0.0 ni, 86.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.3 us, 15.2 sy,  0.0 ni, 78.3 id,  0.0 wa,  0.0 hi,  2.2 si,  0.0 st
%Cpu(s):  5.1 us,  7.7 sy,  0.0 ni, 87.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.9 us, 12.2 sy,  0.0 ni, 80.5 id,  0.0 wa,  0.0 hi,  2.4 si,  0.0 st
%Cpu(s): 10.4 us, 16.7 sy,  0.0 ni, 72.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.0 us, 10.0 sy,  0.0 ni, 82.5 id,  0.0 wa,  0.0 hi,  2.5 si,  0.0 st
%Cpu(s):  7.1 us,  9.5 sy,  0.0 ni, 83.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 21.3 us, 14.9 sy,  0.0 ni, 61.7 id,  0.0 wa,  0.0 hi,  2.1 si,  0.0 st
```

```
%Cpu(s):  8.2 us, 12.2 sy,  0.0 ni, 79.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 23.4 us, 14.9 sy,  0.0 ni, 61.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.3 us, 11.6 sy,  0.0 ni, 86.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.7 us, 14.0 sy,  0.0 ni, 81.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  8.1 us, 13.5 sy,  0.0 ni, 78.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 10.6 us, 12.8 sy,  0.0 ni, 74.5 id,  0.0 wa,  0.0 hi,  2.1 si,  0.0 st
%Cpu(s):  8.9 us, 13.3 sy,  0.0 ni, 77.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  6.2 us, 12.5 sy,  0.0 ni, 79.2 id,  0.0 wa,  0.0 hi,  2.1 si,  0.0 st
%Cpu(s):  6.8 us, 13.6 sy,  0.0 ni, 79.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 14.9 us, 17.0 sy,  0.0 ni, 66.0 id,  0.0 wa,  0.0 hi,  2.1 si,  0.0 st
%Cpu(s): 11.3 us, 13.2 sy,  0.0 ni, 75.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 14.3 us, 18.4 sy,  0.0 ni, 67.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.0 us, 11.6 sy,  0.0 ni, 81.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.7 us,  8.1 sy,  0.0 ni, 89.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 11.1 us, 13.0 sy,  0.0 ni, 75.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  9.3 us, 11.6 sy,  0.0 ni, 79.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 22.4 us, 16.3 sy,  0.0 ni, 59.2 id,  0.0 wa,  0.0 hi,  2.0 si,  0.0 st
%Cpu(s):  8.9 us, 11.1 sy,  0.0 ni, 80.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 14.0 us, 16.0 sy,  0.0 ni, 70.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.7 us, 17.3 sy,  0.0 ni, 75.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.3 us,  7.3 sy,  0.0 ni, 85.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 18.6 us, 16.9 sy,  0.0 ni, 62.7 id,  0.0 wa,  0.0 hi,  1.7 si,  0.0 st
%Cpu(s):  0.0 us,  5.9 sy,  0.0 ni, 94.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 10.4 us, 12.5 sy,  0.0 ni, 77.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.1 us, 11.9 sy,  0.0 ni, 81.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.0 us, 11.6 sy,  0.0 ni, 81.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 18.4 us, 14.3 sy,  0.0 ni, 65.3 id,  0.0 wa,  0.0 hi,  2.0 si,  0.0 st
%Cpu(s):  7.1 us,  7.1 sy,  0.0 ni, 85.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.6 us,  7.9 sy,  0.0 ni, 89.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.7 us,  8.1 sy,  0.0 ni, 89.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.3 us,  5.3 sy,  0.0 ni, 89.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 11.4 us, 11.4 sy,  0.0 ni, 77.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.0 us,  9.3 sy,  0.0 ni, 83.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  0.0 us,  6.1 sy,  0.0 ni, 93.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.7 us,  8.6 sy,  0.0 ni, 82.9 id,  0.0 wa,  0.0 hi,  2.9 si,  0.0 st
%Cpu(s):  2.2 us, 15.6 sy,  0.0 ni, 82.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.3 us, 17.0 sy,  0.0 ni, 76.6 id,  2.1 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 13.2 us, 10.5 sy,  0.0 ni, 76.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.0 us, 10.0 sy,  0.0 ni, 85.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.1 us,  7.1 sy,  0.0 ni, 83.3 id,  0.0 wa,  0.0 hi,  2.4 si,  0.0 st
%Cpu(s):  5.1 us,  5.1 sy,  0.0 ni, 87.2 id,  0.0 wa,  0.0 hi,  2.6 si,  0.0 st
%Cpu(s):  2.4 us, 12.2 sy,  0.0 ni, 85.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 16.2 us,  8.1 sy,  0.0 ni, 75.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.7 us,  2.9 sy,  0.0 ni, 91.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.3 us, 15.2 sy,  0.0 ni, 80.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.4 us,  8.1 sy,  0.0 ni, 86.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  8.3 us, 10.4 sy,  0.0 ni, 81.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 10.0 us, 10.0 sy,  0.0 ni, 80.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  9.1 us,  6.8 sy,  0.0 ni, 81.8 id,  0.0 wa,  0.0 hi,  2.3 si,  0.0 st
%Cpu(s):  2.7 us,  8.1 sy,  0.0 ni, 89.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  9.8 us, 17.1 sy,  0.0 ni, 73.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.3 us,  7.9 sy,  0.0 ni, 86.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.5 us, 13.6 sy,  0.0 ni, 81.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.7 us,  8.6 sy,  0.0 ni, 85.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
```

```
%Cpu(s):  6.5 us, 10.9 sy,  0.0 ni, 78.3 id,  2.2 wa,  0.0 hi,  2.2 si,  0.0 st
%Cpu(s):  4.7 us, 11.6 sy,  0.0 ni, 83.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.7 us,  8.1 sy,  0.0 ni, 89.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.4 us, 13.3 sy,  0.0 ni, 82.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 13.2 us, 13.2 sy,  0.0 ni, 71.1 id,  0.0 wa,  0.0 hi,  2.6 si,  0.0 st
%Cpu(s):  7.0 us, 11.6 sy,  0.0 ni, 81.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.7 us,  7.7 sy,  0.0 ni, 84.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 10.3 us, 10.3 sy,  0.0 ni, 76.9 id,  0.0 wa,  0.0 hi,  2.6 si,  0.0 st
%Cpu(s):  0.0 us,  8.3 sy,  0.0 ni, 91.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.3 us,  7.9 sy,  0.0 ni, 86.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.3 us,  7.9 sy,  0.0 ni, 86.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.5 us, 10.0 sy,  0.0 ni, 85.0 id,  0.0 wa,  0.0 hi,  2.5 si,  0.0 st
%Cpu(s):  8.1 us,  8.1 sy,  0.0 ni, 83.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  6.5 us, 13.0 sy,  0.0 ni, 80.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.8 us,  5.6 sy,  0.0 ni, 91.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 18.6 us, 18.6 sy,  0.0 ni, 62.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  9.8 us,  7.3 sy,  0.0 ni, 82.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  0.0 us,  5.9 sy,  0.0 ni, 94.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  0.0 us,  3.1 sy,  0.0 ni, 96.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  0.0 us,  5.9 sy,  0.0 ni, 94.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.3 us,  5.3 sy,  0.0 ni, 89.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 10.9 us, 13.0 sy,  0.0 ni, 73.9 id,  0.0 wa,  0.0 hi,  2.2 si,  0.0 st
%Cpu(s):  0.0 us, 10.5 sy,  0.0 ni, 89.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.6 us, 10.3 sy,  0.0 ni, 87.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 10.9 us,  8.7 sy,  0.0 ni, 80.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.3 us, 14.0 sy,  0.0 ni, 83.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.7 us,  8.1 sy,  0.0 ni, 89.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.9 us,  5.7 sy,  0.0 ni, 91.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.8 us, 11.1 sy,  0.0 ni, 83.3 id,  0.0 wa,  0.0 hi,  2.8 si,  0.0 st
%Cpu(s):  4.5 us, 11.4 sy,  0.0 ni, 84.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.4 us, 12.2 sy,  0.0 ni, 85.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.7 us,  8.1 sy,  0.0 ni, 89.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.3 us,  5.3 sy,  0.0 ni, 89.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.3 us,  5.3 sy,  0.0 ni, 89.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 13.5 us, 10.8 sy,  0.0 ni, 75.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.3 us,  7.3 sy,  0.0 ni, 85.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.8 us,  5.6 sy,  0.0 ni, 91.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 15.0 us, 12.5 sy,  0.0 ni, 70.0 id,  0.0 wa,  0.0 hi,  2.5 si,  0.0 st
%Cpu(s): 10.0 us,  7.5 sy,  0.0 ni, 82.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  8.5 us, 12.8 sy,  0.0 ni, 78.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  9.8 us, 12.2 sy,  0.0 ni, 78.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.1 us,  7.7 sy,  0.0 ni, 87.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.3 us, 15.2 sy,  0.0 ni, 80.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.8 us, 11.9 sy,  0.0 ni, 83.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.5 us,  7.5 sy,  0.0 ni, 85.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 10.3 us, 15.4 sy,  0.0 ni, 74.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  7.5 us, 10.0 sy,  0.0 ni, 82.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.6 us,  7.9 sy,  0.0 ni, 89.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.9 us,  9.8 sy,  0.0 ni, 82.9 id,  0.0 wa,  0.0 hi,  2.4 si,  0.0 st
%Cpu(s):  5.6 us,  2.8 sy,  0.0 ni, 91.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.0 us, 10.0 sy,  0.0 ni, 85.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.8 us, 11.9 sy,  0.0 ni, 81.0 id,  0.0 wa,  0.0 hi,  2.4 si,  0.0 st
%Cpu(s):  6.5 us, 15.2 sy,  0.0 ni, 78.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 10.0 us, 15.0 sy,  0.0 ni, 75.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
```

```
%Cpu(s):  5.1 us, 10.3 sy,  0.0 ni, 84.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.4 us,  2.7 sy,  0.0 ni, 91.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  4.7 us,  9.3 sy,  0.0 ni, 86.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.4 us,  5.4 sy,  0.0 ni, 89.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  0.0 us,  8.1 sy,  0.0 ni, 89.2 id,  0.0 wa,  0.0 hi,  2.7 si,  0.0 st
%Cpu(s):  0.0 us,  8.1 sy,  0.0 ni, 91.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.4 us,  5.4 sy,  0.0 ni, 89.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 11.1 us,  8.9 sy,  0.0 ni, 77.8 id,  0.0 wa,  0.0 hi,  2.2 si,  0.0 st
%Cpu(s):  4.9 us, 12.2 sy,  0.0 ni, 82.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  8.0 us, 12.0 sy,  0.0 ni, 80.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.4 us, 16.2 sy,  0.0 ni, 78.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  2.4 us, 12.2 sy,  0.0 ni, 85.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.9 us, 13.7 sy,  0.0 ni, 76.5 id,  0.0 wa,  0.0 hi,  3.9 si,  0.0 st
%Cpu(s):  2.6 us,  7.9 sy,  0.0 ni, 89.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s):  5.4 us,  5.4 sy,  0.0 ni, 89.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
```

Listagem 35: Consumo de RAM para *rsyslog*

```
MiB Mem :   3875.9 total,    313.8 free,   1591.2 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    313.8 free,   1591.2 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    313.8 free,   1591.2 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    313.3 free,   1591.7 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    313.3 free,   1591.7 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    313.3 free,   1591.7 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    313.3 free,   1591.7 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    313.0 free,   1591.9 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    313.0 free,   1591.9 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    313.0 free,   1591.9 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    312.8 free,   1592.1 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    312.8 free,   1592.1 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    312.8 free,   1592.1 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    312.8 free,   1592.1 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    312.5 free,   1592.4 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    311.5 free,   1593.4 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    311.1 free,   1593.9 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    311.1 free,   1593.9 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    311.1 free,   1593.9 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    311.1 free,   1593.9 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    311.1 free,   1593.9 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    311.1 free,   1593.9 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    311.1 free,   1593.9 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    311.1 free,   1593.9 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    310.8 free,   1594.1 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    310.6 free,   1594.4 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    310.6 free,   1594.4 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    310.1 free,   1594.8 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    309.1 free,   1595.8 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    308.8 free,   1596.1 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    308.8 free,   1596.1 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    308.6 free,   1596.3 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    308.4 free,   1596.6 used,   1971.0 buff/cache
```

```
MiB Mem :   3875.9 total,    308.4 free,   1596.6 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    308.1 free,   1596.8 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    307.9 free,   1597.1 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    307.9 free,   1597.1 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    307.9 free,   1597.1 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    307.6 free,   1597.3 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    277.1 free,   1627.8 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    330.3 free,   1574.6 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    329.9 free,   1575.0 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    329.9 free,   1575.0 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    330.4 free,   1574.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    330.4 free,   1574.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    330.4 free,   1574.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    331.1 free,   1573.8 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    330.9 free,   1574.0 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.1 free,   1572.8 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    331.9 free,   1573.0 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.1 free,   1572.8 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.1 free,   1572.8 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.1 free,   1572.8 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.4 free,   1572.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.6 free,   1572.3 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.4 free,   1572.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.4 free,   1572.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.4 free,   1572.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.9 free,   1572.0 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    333.1 free,   1571.8 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    333.3 free,   1571.6 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    333.4 free,   1571.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    333.1 free,   1571.8 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    333.4 free,   1571.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    333.1 free,   1571.8 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.9 free,   1572.0 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.9 free,   1572.0 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.9 free,   1572.0 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.9 free,   1572.0 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.4 free,   1572.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.4 free,   1572.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.2 free,   1572.7 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.2 free,   1572.7 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    331.9 free,   1573.0 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    331.7 free,   1573.2 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    331.7 free,   1573.2 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    331.4 free,   1573.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    331.2 free,   1573.7 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.1 free,   1572.8 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    331.9 free,   1573.0 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.4 free,   1572.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.4 free,   1572.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.6 free,   1572.3 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.7 free,   1572.2 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.4 free,   1572.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.2 free,   1572.7 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.4 free,   1572.5 used,   1971.0 buff/cache
```

```
MiB Mem :   3875.9 total,    332.7 free,   1572.2 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.9 free,   1572.0 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.9 free,   1572.0 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.7 free,   1572.2 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.7 free,   1572.2 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.7 free,   1572.2 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.4 free,   1572.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.4 free,   1572.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.2 free,   1572.7 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.2 free,   1572.7 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    331.9 free,   1573.0 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    331.7 free,   1573.2 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.9 free,   1572.0 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.7 free,   1572.2 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.7 free,   1572.2 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.4 free,   1572.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.4 free,   1572.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.4 free,   1572.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.4 free,   1572.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.4 free,   1572.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.4 free,   1572.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.4 free,   1572.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.4 free,   1572.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.4 free,   1572.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.4 free,   1572.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.4 free,   1572.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.4 free,   1572.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.7 free,   1572.2 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.4 free,   1572.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.2 free,   1572.7 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.4 free,   1572.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.4 free,   1572.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.7 free,   1572.2 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.7 free,   1572.2 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.7 free,   1572.2 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.9 free,   1572.0 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.9 free,   1572.0 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.9 free,   1572.0 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    333.2 free,   1571.7 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    333.2 free,   1571.7 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    333.2 free,   1571.7 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    333.2 free,   1571.7 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.9 free,   1572.0 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.9 free,   1572.0 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.9 free,   1572.0 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.9 free,   1572.0 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.9 free,   1572.0 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.9 free,   1572.0 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.9 free,   1572.0 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.9 free,   1572.0 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.9 free,   1572.0 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.7 free,   1572.2 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.7 free,   1572.2 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.7 free,   1572.2 used,   1971.0 buff/cache
```

```
MiB Mem :   3875.9 total,    332.4 free,   1572.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.4 free,   1572.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.2 free,   1572.7 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    331.7 free,   1573.2 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    331.7 free,   1573.2 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    331.4 free,   1573.4 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    331.2 free,   1573.7 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    331.2 free,   1573.7 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    331.2 free,   1573.7 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    331.2 free,   1573.7 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    331.2 free,   1573.7 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    331.2 free,   1573.7 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    331.9 free,   1572.9 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.2 free,   1572.7 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.2 free,   1572.7 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.2 free,   1572.7 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.2 free,   1572.7 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.2 free,   1572.7 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.2 free,   1572.7 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    332.2 free,   1572.7 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    333.2 free,   1571.7 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    333.7 free,   1571.2 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    334.4 free,   1570.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    334.4 free,   1570.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    334.4 free,   1570.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    334.7 free,   1570.2 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    335.1 free,   1569.8 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    335.6 free,   1569.3 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    335.9 free,   1569.0 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    336.4 free,   1568.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    336.4 free,   1568.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    336.4 free,   1568.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    336.4 free,   1568.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    336.6 free,   1568.3 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    336.8 free,   1568.0 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    337.1 free,   1567.8 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    336.4 free,   1568.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    337.1 free,   1567.8 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    336.9 free,   1568.0 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    336.9 free,   1568.0 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    336.9 free,   1568.0 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    336.9 free,   1568.0 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    336.9 free,   1568.0 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    336.6 free,   1568.2 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    336.6 free,   1568.2 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    336.3 free,   1568.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    336.4 free,   1568.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    336.6 free,   1568.2 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    336.6 free,   1568.2 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    336.6 free,   1568.2 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    336.4 free,   1568.5 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    336.9 free,   1568.0 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    336.6 free,   1568.2 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    336.6 free,   1568.2 used,   1971.0 buff/cache
```

```
MiB Mem :   3875.9 total,    336.2 free,   1568.7 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    335.9 free,   1569.0 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    335.9 free,   1569.0 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    335.7 free,   1569.2 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    335.7 free,   1569.2 used,   1971.0 buff/cache
MiB Mem :   3875.9 total,    336.2 free,   1568.7 used,   1971.0 buff/cache
```

Listagem 36: Consumo de CPU fontes externas

```
%Cpu(s): 29.3 us, 19.5 sy,  0.0 ni, 36.6 id,  4.9 wa,  0.0 hi,  9.8 si,  0.0 st
%Cpu(s): 41.0 us, 24.4 sy,  0.0 ni, 26.9 id,  2.6 wa,  0.0 hi,  5.1 si,  0.0 st
%Cpu(s): 15.8 us, 10.5 sy,  0.0 ni, 42.1 id, 18.4 wa,  0.0 hi, 13.2 si,  0.0 st
%Cpu(s): 40.0 us, 15.6 sy,  0.0 ni, 33.3 id,  6.7 wa,  0.0 hi,  4.4 si,  0.0 st
%Cpu(s): 41.3 us, 15.2 sy,  0.0 ni, 30.4 id,  6.5 wa,  0.0 hi,  6.5 si,  0.0 st
%Cpu(s): 35.4 us, 22.9 sy,  0.0 ni, 27.1 id,  6.2 wa,  0.0 hi,  8.3 si,  0.0 st
%Cpu(s): 40.0 us, 15.6 sy,  0.0 ni, 40.0 id,  4.4 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 40.9 us, 15.9 sy,  0.0 ni, 34.1 id,  4.5 wa,  0.0 hi,  4.5 si,  0.0 st
%Cpu(s): 40.7 us, 16.7 sy,  0.0 ni, 27.8 id,  9.3 wa,  0.0 hi,  5.6 si,  0.0 st
%Cpu(s): 27.5 us, 12.5 sy,  0.0 ni, 42.5 id, 10.0 wa,  0.0 hi,  7.5 si,  0.0 st
%Cpu(s): 37.8 us, 11.1 sy,  0.0 ni, 40.0 id,  6.7 wa,  0.0 hi,  4.4 si,  0.0 st
%Cpu(s): 46.0 us, 26.0 sy,  0.0 ni, 22.0 id,  6.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 40.7 us, 18.5 sy,  0.0 ni, 29.6 id,  7.4 wa,  0.0 hi,  3.7 si,  0.0 st
%Cpu(s): 33.3 us, 19.6 sy,  0.0 ni, 33.3 id,  5.9 wa,  0.0 hi,  7.8 si,  0.0 st
%Cpu(s): 32.0 us, 24.0 sy,  0.0 ni, 34.0 id,  6.0 wa,  0.0 hi,  4.0 si,  0.0 st
%Cpu(s): 38.1 us, 21.4 sy,  0.0 ni, 26.2 id, 11.9 wa,  0.0 hi,  2.4 si,  0.0 st
%Cpu(s): 36.6 us, 17.1 sy,  0.0 ni, 36.6 id,  7.3 wa,  0.0 hi,  2.4 si,  0.0 st
%Cpu(s): 37.5 us, 23.2 sy,  0.0 ni, 28.6 id,  5.4 wa,  0.0 hi,  5.4 si,  0.0 st
%Cpu(s): 34.8 us, 17.4 sy,  0.0 ni, 34.8 id,  6.5 wa,  0.0 hi,  6.5 si,  0.0 st
%Cpu(s): 31.7 us, 24.4 sy,  0.0 ni, 34.1 id,  4.9 wa,  0.0 hi,  4.9 si,  0.0 st
%Cpu(s): 37.5 us, 15.0 sy,  0.0 ni, 37.5 id,  5.0 wa,  0.0 hi,  5.0 si,  0.0 st
%Cpu(s): 38.2 us, 18.2 sy,  0.0 ni, 32.7 id,  3.6 wa,  0.0 hi,  7.3 si,  0.0 st
%Cpu(s): 34.8 us, 26.1 sy,  0.0 ni, 23.9 id,  6.5 wa,  0.0 hi,  8.7 si,  0.0 st
%Cpu(s): 33.3 us, 21.6 sy,  0.0 ni, 27.5 id,  7.8 wa,  0.0 hi,  9.8 si,  0.0 st
%Cpu(s): 32.6 us, 19.6 sy,  0.0 ni, 37.0 id,  2.2 wa,  0.0 hi,  8.7 si,  0.0 st
%Cpu(s): 28.2 us, 12.8 sy,  0.0 ni, 35.9 id, 10.3 wa,  0.0 hi, 12.8 si,  0.0 st
%Cpu(s): 36.4 us, 18.2 sy,  0.0 ni, 30.9 id,  7.3 wa,  0.0 hi,  7.3 si,  0.0 st
%Cpu(s): 50.0 us, 21.4 sy,  0.0 ni, 25.0 id,  1.8 wa,  0.0 hi,  1.8 si,  0.0 st
%Cpu(s): 36.5 us, 21.2 sy,  0.0 ni, 34.6 id,  3.8 wa,  0.0 hi,  3.8 si,  0.0 st
%Cpu(s): 36.4 us, 18.2 sy,  0.0 ni, 31.8 id,  9.1 wa,  0.0 hi,  4.5 si,  0.0 st
%Cpu(s): 37.0 us, 26.1 sy,  0.0 ni, 26.1 id,  6.5 wa,  0.0 hi,  4.3 si,  0.0 st
%Cpu(s): 34.5 us, 23.6 sy,  0.0 ni, 18.2 id, 12.7 wa,  0.0 hi, 10.9 si,  0.0 st
%Cpu(s): 33.3 us, 10.3 sy,  0.0 ni, 43.6 id,  7.7 wa,  0.0 hi,  5.1 si,  0.0 st
%Cpu(s): 28.9 us, 22.2 sy,  0.0 ni, 42.2 id,  0.0 wa,  0.0 hi,  6.7 si,  0.0 st
%Cpu(s): 33.9 us, 18.6 sy,  0.0 ni, 33.9 id,  5.1 wa,  0.0 hi,  8.5 si,  0.0 st
%Cpu(s): 39.6 us, 18.8 sy,  0.0 ni, 29.2 id,  6.2 wa,  0.0 hi,  6.2 si,  0.0 st
%Cpu(s): 24.4 us, 26.8 sy,  0.0 ni, 43.9 id,  4.9 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 34.7 us, 18.4 sy,  0.0 ni, 32.7 id,  4.1 wa,  0.0 hi, 10.2 si,  0.0 st
%Cpu(s): 33.3 us, 18.8 sy,  0.0 ni, 39.6 id,  4.2 wa,  0.0 hi,  4.2 si,  0.0 st
%Cpu(s): 43.9 us, 19.5 sy,  0.0 ni, 31.7 id,  2.4 wa,  0.0 hi,  2.4 si,  0.0 st
%Cpu(s): 38.3 us, 14.9 sy,  0.0 ni, 34.0 id,  6.4 wa,  0.0 hi,  6.4 si,  0.0 st
%Cpu(s): 52.9 us, 15.7 sy,  0.0 ni, 21.6 id,  5.9 wa,  0.0 hi,  3.9 si,  0.0 st
```

```
%Cpu(s): 27.0 us, 10.8 sy,  0.0 ni, 48.6 id,  5.4 wa,  0.0 hi,  8.1 si,  0.0 st
%Cpu(s): 30.8 us, 15.4 sy,  0.0 ni, 41.0 id,  2.6 wa,  0.0 hi, 10.3 si,  0.0 st
%Cpu(s): 28.6 us, 14.3 sy,  0.0 ni, 47.6 id,  4.8 wa,  0.0 hi,  4.8 si,  0.0 st
%Cpu(s): 37.0 us, 19.6 sy,  0.0 ni, 34.8 id,  4.3 wa,  0.0 hi,  4.3 si,  0.0 st
%Cpu(s): 36.4 us, 18.2 sy,  0.0 ni, 38.6 id,  2.3 wa,  0.0 hi,  4.5 si,  0.0 st
%Cpu(s): 51.9 us, 31.5 sy,  0.0 ni, 11.1 id,  1.9 wa,  0.0 hi,  3.7 si,  0.0 st
%Cpu(s): 45.8 us, 20.3 sy,  0.0 ni, 20.3 id,  6.8 wa,  0.0 hi,  6.8 si,  0.0 st
%Cpu(s): 62.8 us, 27.9 sy,  0.0 ni,  4.7 id,  2.3 wa,  0.0 hi,  2.3 si,  0.0 st
%Cpu(s): 44.7 us, 15.8 sy,  0.0 ni, 34.2 id,  0.0 wa,  0.0 hi,  5.3 si,  0.0 st
%Cpu(s): 44.0 us, 16.0 sy,  0.0 ni, 36.0 id,  0.0 wa,  0.0 hi,  4.0 si,  0.0 st
%Cpu(s): 40.0 us, 44.4 sy,  0.0 ni,  4.4 id,  0.0 wa,  0.0 hi, 11.1 si,  0.0 st
%Cpu(s): 47.9 us, 16.7 sy,  0.0 ni, 25.0 id,  4.2 wa,  0.0 hi,  6.2 si,  0.0 st
%Cpu(s): 41.4 us, 27.6 sy,  0.0 ni, 15.5 id,  5.2 wa,  0.0 hi, 10.3 si,  0.0 st
%Cpu(s): 42.2 us, 20.0 sy,  0.0 ni, 22.2 id,  6.7 wa,  0.0 hi,  8.9 si,  0.0 st
%Cpu(s): 41.7 us, 27.1 sy,  0.0 ni, 14.6 id, 10.4 wa,  0.0 hi,  6.2 si,  0.0 st
%Cpu(s): 40.0 us, 26.7 sy,  0.0 ni, 11.1 id, 13.3 wa,  0.0 hi,  8.9 si,  0.0 st
%Cpu(s): 46.0 us, 20.0 sy,  0.0 ni, 16.0 id,  6.0 wa,  0.0 hi, 12.0 si,  0.0 st
%Cpu(s): 46.3 us, 12.2 sy,  0.0 ni, 17.1 id, 14.6 wa,  0.0 hi,  9.8 si,  0.0 st
%Cpu(s): 53.8 us, 17.3 sy,  0.0 ni, 11.5 id,  7.7 wa,  0.0 hi,  9.6 si,  0.0 st
%Cpu(s): 52.2 us, 17.4 sy,  0.0 ni, 21.7 id,  6.5 wa,  0.0 hi,  2.2 si,  0.0 st
%Cpu(s): 42.9 us, 28.6 sy,  0.0 ni, 18.4 id,  4.1 wa,  0.0 hi,  6.1 si,  0.0 st
%Cpu(s): 42.3 us, 26.9 sy,  0.0 ni, 11.5 id,  7.7 wa,  0.0 hi, 11.5 si,  0.0 st
%Cpu(s): 57.8 us, 22.2 sy,  0.0 ni,  6.7 id,  8.9 wa,  0.0 hi,  4.4 si,  0.0 st
%Cpu(s): 41.5 us, 17.1 sy,  0.0 ni, 24.4 id,  9.8 wa,  0.0 hi,  7.3 si,  0.0 st
%Cpu(s): 38.1 us, 19.0 sy,  0.0 ni, 23.8 id,  9.5 wa,  0.0 hi,  9.5 si,  0.0 st
%Cpu(s): 47.8 us, 21.7 sy,  0.0 ni, 15.2 id,  6.5 wa,  0.0 hi,  8.7 si,  0.0 st
%Cpu(s): 47.2 us, 22.6 sy,  0.0 ni, 13.2 id,  9.4 wa,  0.0 hi,  7.5 si,  0.0 st
%Cpu(s): 53.2 us, 21.3 sy,  0.0 ni, 12.8 id,  8.5 wa,  0.0 hi,  4.3 si,  0.0 st
%Cpu(s): 53.2 us, 19.1 sy,  0.0 ni,  6.4 id, 12.8 wa,  0.0 hi,  8.5 si,  0.0 st
%Cpu(s): 50.0 us, 22.7 sy,  0.0 ni, 11.4 id, 11.4 wa,  0.0 hi,  4.5 si,  0.0 st
%Cpu(s): 42.0 us, 22.0 sy,  0.0 ni, 12.0 id, 12.0 wa,  0.0 hi, 12.0 si,  0.0 st
%Cpu(s): 14.0 us, 18.6 sy,  0.0 ni, 27.9 id, 32.6 wa,  0.0 hi,  7.0 si,  0.0 st
%Cpu(s): 49.0 us, 26.5 sy,  0.0 ni, 14.3 id,  4.1 wa,  0.0 hi,  6.1 si,  0.0 st
%Cpu(s): 43.1 us, 21.6 sy,  0.0 ni, 17.6 id,  5.9 wa,  0.0 hi, 11.8 si,  0.0 st
%Cpu(s): 31.6 us, 18.4 sy,  0.0 ni, 31.6 id,  5.3 wa,  0.0 hi, 13.2 si,  0.0 st
%Cpu(s): 50.0 us, 20.0 sy,  0.0 ni, 12.0 id, 14.0 wa,  0.0 hi,  4.0 si,  0.0 st
%Cpu(s): 40.0 us, 18.0 sy,  0.0 ni, 24.0 id, 10.0 wa,  0.0 hi,  8.0 si,  0.0 st
%Cpu(s): 43.9 us, 19.5 sy,  0.0 ni, 24.4 id,  4.9 wa,  0.0 hi,  7.3 si,  0.0 st
%Cpu(s): 38.3 us, 27.7 sy,  0.0 ni, 25.5 id,  2.1 wa,  0.0 hi,  6.4 si,  0.0 st
%Cpu(s): 38.1 us, 19.0 sy,  0.0 ni, 31.0 id,  4.8 wa,  0.0 hi,  7.1 si,  0.0 st
%Cpu(s): 48.0 us, 22.0 sy,  0.0 ni, 16.0 id,  6.0 wa,  0.0 hi,  8.0 si,  0.0 st
%Cpu(s): 52.8 us, 16.7 sy,  0.0 ni, 16.7 id,  8.3 wa,  0.0 hi,  5.6 si,  0.0 st
%Cpu(s): 43.2 us, 18.2 sy,  0.0 ni, 18.2 id, 13.6 wa,  0.0 hi,  6.8 si,  0.0 st
%Cpu(s): 41.7 us, 25.0 sy,  0.0 ni,  8.3 id, 12.5 wa,  0.0 hi, 12.5 si,  0.0 st
%Cpu(s): 54.3 us, 19.6 sy,  0.0 ni, 13.0 id,  6.5 wa,  0.0 hi,  6.5 si,  0.0 st
%Cpu(s): 50.0 us, 19.6 sy,  0.0 ni, 17.4 id,  8.7 wa,  0.0 hi,  4.3 si,  0.0 st
%Cpu(s): 50.0 us, 19.0 sy,  0.0 ni, 16.7 id,  7.1 wa,  0.0 hi,  7.1 si,  0.0 st
%Cpu(s): 59.1 us, 18.2 sy,  0.0 ni, 13.6 id,  4.5 wa,  0.0 hi,  4.5 si,  0.0 st
%Cpu(s): 47.5 us, 27.1 sy,  0.0 ni, 15.3 id,  5.1 wa,  0.0 hi,  5.1 si,  0.0 st
%Cpu(s): 46.2 us, 23.1 sy,  0.0 ni, 17.3 id,  5.8 wa,  0.0 hi,  7.7 si,  0.0 st
%Cpu(s): 50.0 us, 21.4 sy,  0.0 ni, 12.5 id,  7.1 wa,  0.0 hi,  8.9 si,  0.0 st
%Cpu(s): 45.0 us, 17.5 sy,  0.0 ni, 17.5 id, 10.0 wa,  0.0 hi, 10.0 si,  0.0 st
%Cpu(s): 47.6 us, 16.7 sy,  0.0 ni, 19.0 id,  9.5 wa,  0.0 hi,  7.1 si,  0.0 st
%Cpu(s): 51.1 us, 23.4 sy,  0.0 ni, 17.0 id,  4.3 wa,  0.0 hi,  4.3 si,  0.0 st
```

```
%Cpu(s): 50.0 us, 19.0 sy,  0.0 ni, 19.0 id,  7.1 wa,  0.0 hi,  4.8 si,  0.0 st
%Cpu(s): 36.8 us, 15.8 sy,  0.0 ni, 31.6 id,  5.3 wa,  0.0 hi, 10.5 si,  0.0 st
%Cpu(s): 42.1 us, 24.6 sy,  0.0 ni, 19.3 id,  7.0 wa,  0.0 hi,  7.0 si,  0.0 st
%Cpu(s): 39.5 us, 23.3 sy,  0.0 ni, 27.9 id,  4.7 wa,  0.0 hi,  4.7 si,  0.0 st
%Cpu(s): 43.6 us, 20.5 sy,  0.0 ni, 25.6 id,  5.1 wa,  0.0 hi,  5.1 si,  0.0 st
%Cpu(s): 47.2 us, 22.6 sy,  0.0 ni,  9.4 id, 13.2 wa,  0.0 hi,  7.5 si,  0.0 st
%Cpu(s): 50.0 us, 15.2 sy,  0.0 ni, 21.7 id,  6.5 wa,  0.0 hi,  6.5 si,  0.0 st
%Cpu(s): 35.0 us, 22.5 sy,  0.0 ni, 27.5 id,  7.5 wa,  0.0 hi,  7.5 si,  0.0 st
%Cpu(s): 43.6 us, 25.5 sy,  0.0 ni, 16.4 id,  5.5 wa,  0.0 hi,  9.1 si,  0.0 st
%Cpu(s): 53.1 us, 18.4 sy,  0.0 ni, 12.2 id,  8.2 wa,  0.0 hi,  8.2 si,  0.0 st
%Cpu(s): 56.4 us, 33.3 sy,  0.0 ni,  7.7 id,  2.6 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu(s): 48.8 us, 19.5 sy,  0.0 ni, 19.5 id,  7.3 wa,  0.0 hi,  4.9 si,  0.0 st
%Cpu(s): 47.6 us, 21.4 sy,  0.0 ni, 16.7 id,  4.8 wa,  0.0 hi,  9.5 si,  0.0 st
%Cpu(s): 54.2 us, 18.8 sy,  0.0 ni, 12.5 id,  6.2 wa,  0.0 hi,  8.3 si,  0.0 st
%Cpu(s): 47.8 us, 10.9 sy,  0.0 ni, 17.4 id, 15.2 wa,  0.0 hi,  8.7 si,  0.0 st
%Cpu(s): 55.7 us, 23.0 sy,  0.0 ni, 14.8 id,  3.3 wa,  0.0 hi,  3.3 si,  0.0 st
%Cpu(s): 57.8 us, 20.0 sy,  0.0 ni,  8.9 id,  6.7 wa,  0.0 hi,  6.7 si,  0.0 st
%Cpu(s): 46.0 us, 16.0 sy,  0.0 ni, 16.0 id, 10.0 wa,  0.0 hi, 12.0 si,  0.0 st
%Cpu(s): 43.8 us, 25.0 sy,  0.0 ni, 14.6 id, 10.4 wa,  0.0 hi,  6.2 si,  0.0 st
%Cpu(s): 51.0 us, 22.4 sy,  0.0 ni,  8.2 id,  8.2 wa,  0.0 hi, 10.2 si,  0.0 st
%Cpu(s): 50.0 us, 27.1 sy,  0.0 ni, 10.4 id,  6.2 wa,  0.0 hi,  6.2 si,  0.0 st
%Cpu(s): 45.0 us, 15.0 sy,  0.0 ni, 22.5 id, 10.0 wa,  0.0 hi,  7.5 si,  0.0 st
%Cpu(s): 52.3 us, 22.7 sy,  0.0 ni,  9.1 id,  6.8 wa,  0.0 hi,  9.1 si,  0.0 st
%Cpu(s): 46.7 us, 20.0 sy,  0.0 ni, 17.8 id,  6.7 wa,  0.0 hi,  8.9 si,  0.0 st
%Cpu(s): 45.2 us, 19.0 sy,  0.0 ni, 14.3 id,  9.5 wa,  0.0 hi, 11.9 si,  0.0 st
%Cpu(s): 44.2 us, 18.6 sy,  0.0 ni, 11.6 id, 11.6 wa,  0.0 hi, 14.0 si,  0.0 st
%Cpu(s): 46.3 us, 17.1 sy,  0.0 ni, 17.1 id,  9.8 wa,  0.0 hi,  9.8 si,  0.0 st
%Cpu(s): 34.2 us, 18.4 sy,  0.0 ni, 36.8 id,  2.6 wa,  0.0 hi,  7.9 si,  0.0 st
%Cpu(s): 54.9 us, 21.6 sy,  0.0 ni,  7.8 id,  7.8 wa,  0.0 hi,  7.8 si,  0.0 st
%Cpu(s): 50.0 us, 21.4 sy,  0.0 ni, 11.9 id, 14.3 wa,  0.0 hi,  2.4 si,  0.0 st
%Cpu(s): 54.5 us, 20.5 sy,  0.0 ni,  4.5 id, 13.6 wa,  0.0 hi,  6.8 si,  0.0 st
%Cpu(s): 33.3 us, 11.1 sy,  0.0 ni, 38.9 id, 11.1 wa,  0.0 hi,  5.6 si,  0.0 st
%Cpu(s): 34.1 us, 17.1 sy,  0.0 ni, 34.1 id,  7.3 wa,  0.0 hi,  7.3 si,  0.0 st
%Cpu(s): 51.9 us, 27.8 sy,  0.0 ni,  7.4 id,  5.6 wa,  0.0 hi,  7.4 si,  0.0 st
%Cpu(s): 50.9 us, 24.5 sy,  0.0 ni, 17.0 id,  3.8 wa,  0.0 hi,  3.8 si,  0.0 st
%Cpu(s): 53.1 us, 22.4 sy,  0.0 ni,  6.1 id, 12.2 wa,  0.0 hi,  6.1 si,  0.0 st
%Cpu(s): 52.3 us, 22.7 sy,  0.0 ni, 13.6 id,  4.5 wa,  0.0 hi,  6.8 si,  0.0 st
%Cpu(s): 47.2 us, 11.1 sy,  0.0 ni, 25.0 id,  8.3 wa,  0.0 hi,  8.3 si,  0.0 st
%Cpu(s): 43.5 us, 26.1 sy,  0.0 ni, 21.7 id,  2.2 wa,  0.0 hi,  6.5 si,  0.0 st
%Cpu(s): 48.9 us, 19.1 sy,  0.0 ni, 17.0 id,  4.3 wa,  0.0 hi, 10.6 si,  0.0 st
%Cpu(s): 46.3 us, 19.5 sy,  0.0 ni, 17.1 id,  9.8 wa,  0.0 hi,  7.3 si,  0.0 st
%Cpu(s): 45.6 us, 22.8 sy,  0.0 ni, 12.3 id,  8.8 wa,  0.0 hi, 10.5 si,  0.0 st
%Cpu(s): 46.5 us, 14.0 sy,  0.0 ni, 23.3 id,  7.0 wa,  0.0 hi,  9.3 si,  0.0 st
%Cpu(s): 51.2 us, 16.3 sy,  0.0 ni, 18.6 id,  7.0 wa,  0.0 hi,  7.0 si,  0.0 st
%Cpu(s): 44.2 us, 27.9 sy,  0.0 ni, 14.0 id,  7.0 wa,  0.0 hi,  7.0 si,  0.0 st
%Cpu(s): 47.8 us, 17.4 sy,  0.0 ni, 19.6 id,  6.5 wa,  0.0 hi,  8.7 si,  0.0 st
%Cpu(s): 52.8 us, 26.4 sy,  0.0 ni,  9.4 id,  5.7 wa,  0.0 hi,  5.7 si,  0.0 st
%Cpu(s): 42.9 us, 22.4 sy,  0.0 ni, 22.4 id,  6.1 wa,  0.0 hi,  6.1 si,  0.0 st
%Cpu(s): 52.8 us, 25.0 sy,  0.0 ni,  5.6 id,  8.3 wa,  0.0 hi,  8.3 si,  0.0 st
%Cpu(s): 53.5 us, 23.3 sy,  0.0 ni,  9.3 id,  4.7 wa,  0.0 hi,  9.3 si,  0.0 st
%Cpu(s): 41.2 us, 29.4 sy,  0.0 ni, 17.6 id,  7.8 wa,  0.0 hi,  3.9 si,  0.0 st
%Cpu(s): 42.9 us, 24.5 sy,  0.0 ni, 16.3 id,  8.2 wa,  0.0 hi,  8.2 si,  0.0 st
%Cpu(s): 48.1 us, 20.4 sy,  0.0 ni, 14.8 id,  9.3 wa,  0.0 hi,  7.4 si,  0.0 st
%Cpu(s): 36.4 us, 20.5 sy,  0.0 ni, 29.5 id,  6.8 wa,  0.0 hi,  6.8 si,  0.0 st
```

```
%Cpu(s): 31.0 us, 19.0 sy,  0.0 ni, 31.0 id,  7.1 wa,  0.0 hi, 11.9 si,  0.0 st
%Cpu(s): 45.8 us, 20.8 sy,  0.0 ni, 18.8 id,  8.3 wa,  0.0 hi,  6.2 si,  0.0 st
%Cpu(s): 44.2 us, 25.0 sy,  0.0 ni, 11.5 id, 13.5 wa,  0.0 hi,  5.8 si,  0.0 st
%Cpu(s): 48.3 us, 20.0 sy,  0.0 ni, 10.0 id, 11.7 wa,  0.0 hi, 10.0 si,  0.0 st
%Cpu(s): 54.5 us, 27.3 sy,  0.0 ni,  5.5 id,  3.6 wa,  0.0 hi,  9.1 si,  0.0 st
%Cpu(s): 48.0 us, 22.0 sy,  0.0 ni, 10.0 id,  8.0 wa,  0.0 hi, 12.0 si,  0.0 st
%Cpu(s): 51.1 us, 24.4 sy,  0.0 ni, 15.6 id,  4.4 wa,  0.0 hi,  4.4 si,  0.0 st
%Cpu(s): 41.3 us, 21.7 sy,  0.0 ni, 26.1 id,  2.2 wa,  0.0 hi,  8.7 si,  0.0 st
%Cpu(s): 42.9 us, 26.8 sy,  0.0 ni, 19.6 id,  5.4 wa,  0.0 hi,  5.4 si,  0.0 st
%Cpu(s): 54.8 us, 16.7 sy,  0.0 ni, 14.3 id,  7.1 wa,  0.0 hi,  7.1 si,  0.0 st
%Cpu(s): 53.8 us, 20.5 sy,  0.0 ni, 10.3 id,  5.1 wa,  0.0 hi, 10.3 si,  0.0 st
%Cpu(s): 46.2 us, 12.8 sy,  0.0 ni, 28.2 id,  7.7 wa,  0.0 hi,  5.1 si,  0.0 st
%Cpu(s): 46.5 us, 18.6 sy,  0.0 ni, 27.9 id,  2.3 wa,  0.0 hi,  4.7 si,  0.0 st
%Cpu(s): 46.2 us, 19.2 sy,  0.0 ni, 23.1 id,  3.8 wa,  0.0 hi,  7.7 si,  0.0 st
%Cpu(s): 46.3 us, 14.6 sy,  0.0 ni, 22.0 id,  4.9 wa,  0.0 hi, 12.2 si,  0.0 st
%Cpu(s): 42.5 us, 25.0 sy,  0.0 ni, 20.0 id,  7.5 wa,  0.0 hi,  5.0 si,  0.0 st
%Cpu(s): 44.2 us, 20.9 sy,  0.0 ni, 23.3 id,  4.7 wa,  0.0 hi,  7.0 si,  0.0 st
%Cpu(s): 43.8 us,  9.4 sy,  0.0 ni, 37.5 id,  3.1 wa,  0.0 hi,  6.2 si,  0.0 st
%Cpu(s): 45.2 us, 14.3 sy,  0.0 ni, 26.2 id,  9.5 wa,  0.0 hi,  4.8 si,  0.0 st
%Cpu(s): 44.4 us, 20.0 sy,  0.0 ni, 20.0 id,  6.7 wa,  0.0 hi,  8.9 si,  0.0 st
%Cpu(s): 46.3 us, 12.2 sy,  0.0 ni, 17.1 id, 14.6 wa,  0.0 hi,  9.8 si,  0.0 st
%Cpu(s): 36.6 us, 19.5 sy,  0.0 ni, 24.4 id,  9.8 wa,  0.0 hi,  9.8 si,  0.0 st
%Cpu(s): 53.2 us, 12.8 sy,  0.0 ni, 10.6 id, 12.8 wa,  0.0 hi, 10.6 si,  0.0 st
%Cpu(s): 51.9 us, 25.9 sy,  0.0 ni,  3.7 id, 11.1 wa,  0.0 hi,  7.4 si,  0.0 st
%Cpu(s): 45.8 us, 25.0 sy,  0.0 ni, 10.4 id,  8.3 wa,  0.0 hi, 10.4 si,  0.0 st
%Cpu(s): 37.5 us, 27.1 sy,  0.0 ni, 22.9 id,  4.2 wa,  0.0 hi,  8.3 si,  0.0 st
%Cpu(s): 48.3 us, 20.0 sy,  0.0 ni,  8.3 id, 13.3 wa,  0.0 hi, 10.0 si,  0.0 st
%Cpu(s): 60.0 us, 20.0 sy,  0.0 ni,  7.3 id,  7.3 wa,  0.0 hi,  5.5 si,  0.0 st
%Cpu(s): 49.1 us, 23.6 sy,  0.0 ni, 14.5 id,  5.5 wa,  0.0 hi,  7.3 si,  0.0 st
%Cpu(s): 51.8 us, 21.4 sy,  0.0 ni, 16.1 id,  3.6 wa,  0.0 hi,  7.1 si,  0.0 st
%Cpu(s): 50.0 us, 23.9 sy,  0.0 ni, 10.9 id,  6.5 wa,  0.0 hi,  8.7 si,  0.0 st
%Cpu(s): 55.6 us, 17.8 sy,  0.0 ni, 11.1 id,  6.7 wa,  0.0 hi,  8.9 si,  0.0 st
%Cpu(s): 43.8 us, 22.9 sy,  0.0 ni, 14.6 id, 12.5 wa,  0.0 hi,  6.2 si,  0.0 st
%Cpu(s): 51.0 us, 15.7 sy,  0.0 ni, 13.7 id,  9.8 wa,  0.0 hi,  9.8 si,  0.0 st
%Cpu(s): 31.4 us, 22.9 sy,  0.0 ni, 20.0 id, 11.4 wa,  0.0 hi, 14.3 si,  0.0 st
%Cpu(s): 45.8 us, 23.7 sy,  0.0 ni, 11.9 id, 10.2 wa,  0.0 hi,  8.5 si,  0.0 st
%Cpu(s): 43.8 us, 22.9 sy,  0.0 ni, 16.7 id, 10.4 wa,  0.0 hi,  6.2 si,  0.0 st
%Cpu(s): 44.4 us, 24.1 sy,  0.0 ni, 16.7 id,  5.6 wa,  0.0 hi,  9.3 si,  0.0 st
%Cpu(s): 48.8 us, 29.3 sy,  0.0 ni, 14.6 id,  4.9 wa,  0.0 hi,  2.4 si,  0.0 st
%Cpu(s): 58.3 us, 20.8 sy,  0.0 ni, 10.4 id,  6.2 wa,  0.0 hi,  4.2 si,  0.0 st
%Cpu(s): 40.9 us, 18.2 sy,  0.0 ni, 22.7 id,  9.1 wa,  0.0 hi,  9.1 si,  0.0 st
%Cpu(s): 45.2 us, 16.7 sy,  0.0 ni, 28.6 id,  2.4 wa,  0.0 hi,  7.1 si,  0.0 st
%Cpu(s): 53.2 us, 23.4 sy,  0.0 ni,  4.3 id,  8.5 wa,  0.0 hi, 10.6 si,  0.0 st
%Cpu(s): 54.3 us, 19.6 sy,  0.0 ni, 13.0 id,  4.3 wa,  0.0 hi,  8.7 si,  0.0 st
%Cpu(s): 53.2 us, 23.4 sy,  0.0 ni, 10.6 id,  4.3 wa,  0.0 hi,  8.5 si,  0.0 st
%Cpu(s): 46.8 us, 17.0 sy,  0.0 ni, 19.1 id,  6.4 wa,  0.0 hi, 10.6 si,  0.0 st
%Cpu(s): 44.4 us, 20.0 sy,  0.0 ni, 20.0 id,  6.7 wa,  0.0 hi,  8.9 si,  0.0 st
%Cpu(s): 38.5 us, 25.6 sy,  0.0 ni, 15.4 id, 12.8 wa,  0.0 hi,  7.7 si,  0.0 st
%Cpu(s): 47.6 us, 19.0 sy,  0.0 ni, 19.0 id,  7.1 wa,  0.0 hi,  7.1 si,  0.0 st
%Cpu(s): 56.0 us, 18.0 sy,  0.0 ni,  8.0 id,  6.0 wa,  0.0 hi, 12.0 si,  0.0 st
%Cpu(s): 43.9 us, 17.1 sy,  0.0 ni, 17.1 id,  9.8 wa,  0.0 hi, 12.2 si,  0.0 st
%Cpu(s): 57.5 us, 17.5 sy,  0.0 ni, 10.0 id,  7.5 wa,  0.0 hi,  7.5 si,  0.0 st
%Cpu(s): 52.4 us, 16.7 sy,  0.0 ni, 11.9 id,  9.5 wa,  0.0 hi,  9.5 si,  0.0 st
%Cpu(s): 52.3 us, 20.5 sy,  0.0 ni, 13.6 id,  4.5 wa,  0.0 hi,  9.1 si,  0.0 st
```

```
%Cpu(s): 59.6 us, 19.1 sy,  0.0 ni,  6.4 id,  8.5 wa,  0.0 hi,  6.4 si,  0.0 st
%Cpu(s): 46.7 us, 20.0 sy,  0.0 ni, 13.3 id,  6.7 wa,  0.0 hi, 13.3 si,  0.0 st
%Cpu(s): 42.9 us, 22.4 sy,  0.0 ni, 14.3 id,  8.2 wa,  0.0 hi, 12.2 si,  0.0 st
%Cpu(s): 37.5 us, 20.8 sy,  0.0 ni, 29.2 id,  4.2 wa,  0.0 hi,  8.3 si,  0.0 st
%Cpu(s): 50.0 us, 14.8 sy,  0.0 ni, 24.1 id,  5.6 wa,  0.0 hi,  5.6 si,  0.0 st
%Cpu(s): 43.9 us, 22.0 sy,  0.0 ni, 24.4 id,  4.9 wa,  0.0 hi,  4.9 si,  0.0 st
%Cpu(s): 34.7 us, 16.3 sy,  0.0 ni, 34.7 id,  6.1 wa,  0.0 hi,  8.2 si,  0.0 st
%Cpu(s): 45.8 us, 18.8 sy,  0.0 ni, 29.2 id,  2.1 wa,  0.0 hi,  4.2 si,  0.0 st
%Cpu(s): 33.3 us, 19.6 sy,  0.0 ni, 35.3 id,  5.9 wa,  0.0 hi,  5.9 si,  0.0 st
%Cpu(s): 44.0 us, 16.0 sy,  0.0 ni, 32.0 id,  4.0 wa,  0.0 hi,  4.0 si,  0.0 st
%Cpu(s): 34.1 us, 22.7 sy,  0.0 ni, 29.5 id,  6.8 wa,  0.0 hi,  6.8 si,  0.0 st
%Cpu(s): 33.3 us, 14.6 sy,  0.0 ni, 35.4 id,  8.3 wa,  0.0 hi,  8.3 si,  0.0 st
%Cpu(s): 39.5 us, 13.2 sy,  0.0 ni, 36.8 id,  7.9 wa,  0.0 hi,  2.6 si,  0.0 st
%Cpu(s): 39.6 us, 20.8 sy,  0.0 ni, 31.2 id,  2.1 wa,  0.0 hi,  6.2 si,  0.0 st
%Cpu(s): 28.6 us, 22.4 sy,  0.0 ni, 30.6 id,  6.1 wa,  0.0 hi, 12.2 si,  0.0 st
```

Listagem 37: Consumo de RAM fontes externas

```
MiB Mem :   3875.9 total,   1480.1 free,   1173.2 used,   1222.6 buff/cache
MiB Mem :   3875.9 total,   1481.4 free,   1171.9 used,   1222.6 buff/cache
MiB Mem :   3875.9 total,   1480.9 free,   1172.4 used,   1222.6 buff/cache
MiB Mem :   3875.9 total,   1479.9 free,   1173.3 used,   1222.7 buff/cache
MiB Mem :   3875.9 total,   1479.9 free,   1173.3 used,   1222.7 buff/cache
MiB Mem :   3875.9 total,   1479.9 free,   1173.3 used,   1222.7 buff/cache
MiB Mem :   3875.9 total,   1479.9 free,   1173.3 used,   1222.7 buff/cache
MiB Mem :   3875.9 total,   1479.9 free,   1173.3 used,   1222.7 buff/cache
MiB Mem :   3875.9 total,   1479.6 free,   1173.5 used,   1222.7 buff/cache
MiB Mem :   3875.9 total,   1479.6 free,   1173.5 used,   1222.7 buff/cache
MiB Mem :   3875.9 total,   1479.6 free,   1173.5 used,   1222.8 buff/cache
MiB Mem :   3875.9 total,   1480.9 free,   1172.2 used,   1222.8 buff/cache
MiB Mem :   3875.9 total,   1480.4 free,   1172.7 used,   1222.8 buff/cache
MiB Mem :   3875.9 total,   1480.4 free,   1172.7 used,   1222.8 buff/cache
MiB Mem :   3875.9 total,   1479.9 free,   1173.1 used,   1222.9 buff/cache
MiB Mem :   3875.9 total,   1479.9 free,   1173.0 used,   1223.0 buff/cache
MiB Mem :   3875.9 total,   1474.7 free,   1178.2 used,   1223.0 buff/cache
MiB Mem :   3875.9 total,   1480.1 free,   1172.8 used,   1223.0 buff/cache
MiB Mem :   3875.9 total,   1479.9 free,   1173.0 used,   1223.0 buff/cache
MiB Mem :   3875.9 total,   1480.3 free,   1172.6 used,   1223.0 buff/cache
MiB Mem :   3875.9 total,   1480.6 free,   1172.3 used,   1223.0 buff/cache
MiB Mem :   3875.9 total,   1481.6 free,   1171.2 used,   1223.1 buff/cache
MiB Mem :   3875.9 total,   1481.3 free,   1171.4 used,   1223.1 buff/cache
MiB Mem :   3875.9 total,   1481.3 free,   1171.4 used,   1223.1 buff/cache
MiB Mem :   3875.9 total,   1481.3 free,   1171.4 used,   1223.2 buff/cache
MiB Mem :   3875.9 total,   1481.1 free,   1171.6 used,   1223.2 buff/cache
MiB Mem :   3875.9 total,   1511.9 free,   1173.1 used,   1190.9 buff/cache
MiB Mem :   3875.9 total,   1512.1 free,   1172.8 used,   1190.9 buff/cache
MiB Mem :   3875.9 total,   1512.8 free,   1172.1 used,   1190.9 buff/cache
MiB Mem :   3875.9 total,   1512.6 free,   1172.2 used,   1191.0 buff/cache
MiB Mem :   3875.9 total,   1512.6 free,   1172.2 used,   1191.0 buff/cache
MiB Mem :   3875.9 total,   1512.6 free,   1172.2 used,   1191.1 buff/cache
MiB Mem :   3875.9 total,   1512.9 free,   1172.0 used,   1191.1 buff/cache
```

```
MiB Mem :   3875.9 total,   1512.6 free,   1172.2 used,   1191.1 buff/cache
MiB Mem :   3875.9 total,   1512.4 free,   1172.4 used,   1191.1 buff/cache
MiB Mem :   3875.9 total,   1511.9 free,   1172.9 used,   1191.1 buff/cache
MiB Mem :   3875.9 total,   1511.6 free,   1173.1 used,   1191.1 buff/cache
MiB Mem :   3875.9 total,   1511.4 free,   1173.4 used,   1191.1 buff/cache
MiB Mem :   3875.9 total,   1511.4 free,   1173.4 used,   1191.1 buff/cache
MiB Mem :   3875.9 total,   1511.2 free,   1173.6 used,   1191.1 buff/cache
MiB Mem :   3875.9 total,   1510.9 free,   1173.7 used,   1191.2 buff/cache
MiB Mem :   3875.9 total,   1511.2 free,   1173.5 used,   1191.2 buff/cache
MiB Mem :   3875.9 total,   1511.7 free,   1173.0 used,   1191.2 buff/cache
MiB Mem :   3875.9 total,   1511.7 free,   1172.9 used,   1191.3 buff/cache
MiB Mem :   3875.9 total,   1512.2 free,   1172.4 used,   1191.3 buff/cache
MiB Mem :   3875.9 total,   1512.2 free,   1172.4 used,   1191.3 buff/cache
MiB Mem :   3875.9 total,   1511.9 free,   1172.6 used,   1191.3 buff/cache
MiB Mem :   3875.9 total,   1511.7 free,   1172.9 used,   1191.3 buff/cache
MiB Mem :   3875.9 total,   1511.7 free,   1172.9 used,   1191.3 buff/cache
MiB Mem :   3875.9 total,   1501.4 free,   1183.2 used,   1191.3 buff/cache
MiB Mem :   3875.9 total,   1474.6 free,   1209.8 used,   1191.5 buff/cache
MiB Mem :   3875.9 total,   1474.1 free,   1210.3 used,   1191.5 buff/cache
MiB Mem :   3875.9 total,   1473.9 free,   1210.5 used,   1191.5 buff/cache
MiB Mem :   3875.9 total,   1452.8 free,   1215.6 used,   1207.6 buff/cache
MiB Mem :   3875.9 total,   1451.8 free,   1216.5 used,   1207.6 buff/cache
MiB Mem :   3875.9 total,   1445.6 free,   1222.7 used,   1207.6 buff/cache
MiB Mem :   3875.9 total,   1444.1 free,   1224.1 used,   1207.6 buff/cache
MiB Mem :   3875.9 total,   1440.9 free,   1227.2 used,   1207.8 buff/cache
MiB Mem :   3875.9 total,   1440.9 free,   1227.2 used,   1207.8 buff/cache
MiB Mem :   3875.9 total,   1431.8 free,   1220.2 used,   1223.8 buff/cache
MiB Mem :   3875.9 total,   1430.4 free,   1221.7 used,   1223.8 buff/cache
MiB Mem :   3875.9 total,   1429.9 free,   1222.2 used,   1223.9 buff/cache
MiB Mem :   3875.9 total,   1429.9 free,   1222.2 used,   1223.9 buff/cache
MiB Mem :   3875.9 total,   1428.4 free,   1223.6 used,   1223.9 buff/cache
MiB Mem :   3875.9 total,   1428.1 free,   1223.9 used,   1223.9 buff/cache
MiB Mem :   3875.9 total,   1427.9 free,   1224.1 used,   1223.9 buff/cache
MiB Mem :   3875.9 total,   1427.9 free,   1224.1 used,   1223.9 buff/cache
MiB Mem :   3875.9 total,   1427.9 free,   1224.1 used,   1223.9 buff/cache
MiB Mem :   3875.9 total,   1427.9 free,   1224.1 used,   1223.9 buff/cache
MiB Mem :   3875.9 total,   1427.9 free,   1224.0 used,   1223.9 buff/cache
MiB Mem :   3875.9 total,   1427.9 free,   1224.0 used,   1224.0 buff/cache
MiB Mem :   3875.9 total,   1427.7 free,   1224.3 used,   1224.0 buff/cache
MiB Mem :   3875.9 total,   1428.4 free,   1223.5 used,   1224.0 buff/cache
MiB Mem :   3875.9 total,   1428.9 free,   1222.9 used,   1224.0 buff/cache
MiB Mem :   3875.9 total,   1415.9 free,   1219.6 used,   1240.4 buff/cache
MiB Mem :   3875.9 total,   1415.9 free,   1219.6 used,   1240.4 buff/cache
MiB Mem :   3875.9 total,   1364.5 free,   1270.9 used,   1240.4 buff/cache
MiB Mem :   3875.9 total,   1346.8 free,   1288.7 used,   1240.4 buff/cache
MiB Mem :   3875.9 total,   1346.8 free,   1288.6 used,   1240.5 buff/cache
MiB Mem :   3875.9 total,   1346.3 free,   1289.1 used,   1240.5 buff/cache
MiB Mem :   3875.9 total,   1346.3 free,   1289.1 used,   1240.6 buff/cache
MiB Mem :   3875.9 total,   1346.3 free,   1289.0 used,   1240.6 buff/cache
MiB Mem :   3875.9 total,   1346.3 free,   1289.0 used,   1240.6 buff/cache
MiB Mem :   3875.9 total,   1346.0 free,   1289.3 used,   1240.5 buff/cache
MiB Mem :   3875.9 total,   1345.8 free,   1289.5 used,   1240.6 buff/cache
MiB Mem :   3875.9 total,   1343.1 free,   1284.0 used,   1248.8 buff/cache
MiB Mem :   3875.9 total,   1342.3 free,   1283.5 used,   1250.1 buff/cache
```

```
MiB Mem :   3875.9 total,   1342.3 free,   1283.5 used,   1250.1 buff/cache
MiB Mem :   3875.9 total,   1342.1 free,   1283.7 used,   1250.1 buff/cache
MiB Mem :   3875.9 total,   1336.9 free,   1288.9 used,   1250.1 buff/cache
MiB Mem :   3875.9 total,   1336.9 free,   1288.8 used,   1250.1 buff/cache
MiB Mem :   3875.9 total,   1336.7 free,   1289.1 used,   1250.1 buff/cache
MiB Mem :   3875.9 total,   1336.7 free,   1289.1 used,   1250.2 buff/cache
MiB Mem :   3875.9 total,   1336.2 free,   1289.5 used,   1250.2 buff/cache
MiB Mem :   3875.9 total,   1335.9 free,   1289.8 used,   1250.2 buff/cache
MiB Mem :   3875.9 total,   1335.9 free,   1289.8 used,   1250.2 buff/cache
MiB Mem :   3875.9 total,   1335.9 free,   1289.7 used,   1250.2 buff/cache
MiB Mem :   3875.9 total,   1335.9 free,   1289.7 used,   1250.2 buff/cache
MiB Mem :   3875.9 total,   1335.9 free,   1289.7 used,   1250.2 buff/cache
MiB Mem :   3875.9 total,   1335.9 free,   1289.7 used,   1250.2 buff/cache
MiB Mem :   3875.9 total,   1335.9 free,   1289.7 used,   1250.3 buff/cache
MiB Mem :   3875.9 total,   1335.9 free,   1289.7 used,   1250.3 buff/cache
MiB Mem :   3875.9 total,   1335.9 free,   1289.7 used,   1250.3 buff/cache
MiB Mem :   3875.9 total,   1335.9 free,   1289.6 used,   1250.3 buff/cache
MiB Mem :   3875.9 total,   1335.9 free,   1289.6 used,   1250.3 buff/cache
MiB Mem :   3875.9 total,   1335.9 free,   1289.6 used,   1250.3 buff/cache
MiB Mem :   3875.9 total,   1331.8 free,   1293.6 used,   1250.5 buff/cache
MiB Mem :   3875.9 total,   1331.8 free,   1293.6 used,   1250.5 buff/cache
MiB Mem :   3875.9 total,   1331.3 free,   1294.1 used,   1250.5 buff/cache
MiB Mem :   3875.9 total,   1331.3 free,   1294.1 used,   1250.5 buff/cache
MiB Mem :   3875.9 total,   1331.3 free,   1294.1 used,   1250.5 buff/cache
MiB Mem :   3875.9 total,   1331.1 free,   1294.3 used,   1250.5 buff/cache
MiB Mem :   3875.9 total,   1331.1 free,   1294.3 used,   1250.6 buff/cache
MiB Mem :   3875.9 total,   1331.1 free,   1294.2 used,   1250.6 buff/cache
MiB Mem :   3875.9 total,   1331.1 free,   1294.1 used,   1250.7 buff/cache
MiB Mem :   3875.9 total,   1331.1 free,   1294.1 used,   1250.7 buff/cache
MiB Mem :   3875.9 total,   1331.1 free,   1294.1 used,   1250.7 buff/cache
MiB Mem :   3875.9 total,   1331.1 free,   1294.1 used,   1250.7 buff/cache
MiB Mem :   3875.9 total,   1331.1 free,   1294.1 used,   1250.8 buff/cache
MiB Mem :   3875.9 total,   1331.1 free,   1294.0 used,   1250.8 buff/cache
MiB Mem :   3875.9 total,   1331.1 free,   1294.0 used,   1250.8 buff/cache
MiB Mem :   3875.9 total,   1331.1 free,   1294.0 used,   1250.8 buff/cache
MiB Mem :   3875.9 total,   1331.3 free,   1293.8 used,   1250.8 buff/cache
MiB Mem :   3875.9 total,   1333.5 free,   1291.5 used,   1250.8 buff/cache
MiB Mem :   3875.9 total,   1334.7 free,   1290.3 used,   1250.8 buff/cache
MiB Mem :   3875.9 total,   1334.5 free,   1290.5 used,   1250.8 buff/cache
MiB Mem :   3875.9 total,   1333.3 free,   1291.7 used,   1250.8 buff/cache
MiB Mem :   3875.9 total,   1333.3 free,   1291.7 used,   1250.9 buff/cache
MiB Mem :   3875.9 total,   1333.3 free,   1291.7 used,   1250.9 buff/cache
MiB Mem :   3875.9 total,   1333.3 free,   1291.7 used,   1250.9 buff/cache
MiB Mem :   3875.9 total,   1333.3 free,   1291.7 used,   1250.9 buff/cache
MiB Mem :   3875.9 total,   1333.3 free,   1291.7 used,   1250.9 buff/cache
MiB Mem :   3875.9 total,   1333.3 free,   1291.7 used,   1250.9 buff/cache
MiB Mem :   3875.9 total,   1333.3 free,   1291.7 used,   1250.9 buff/cache
MiB Mem :   3875.9 total,   1333.3 free,   1291.7 used,   1250.9 buff/cache
MiB Mem :   3875.9 total,   1333.1 free,   1291.9 used,   1251.0 buff/cache
MiB Mem :   3875.9 total,   1333.1 free,   1291.9 used,   1251.0 buff/cache
MiB Mem :   3875.9 total,   1332.8 free,   1292.1 used,   1251.0 buff/cache
MiB Mem :   3875.9 total,   1332.8 free,   1292.1 used,   1251.0 buff/cache
MiB Mem :   3875.9 total,   1332.8 free,   1292.1 used,   1251.0 buff/cache
MiB Mem :   3875.9 total,   1332.8 free,   1292.0 used,   1251.0 buff/cache
```

```
MiB Mem :   3875.9 total,   1332.8 free,   1292.0 used,   1251.1 buff/cache
MiB Mem :   3875.9 total,   1332.8 free,   1292.0 used,   1251.1 buff/cache
MiB Mem :   3875.9 total,   1332.8 free,   1291.9 used,   1251.2 buff/cache
MiB Mem :   3875.9 total,   1332.8 free,   1291.9 used,   1251.2 buff/cache
MiB Mem :   3875.9 total,   1332.8 free,   1291.9 used,   1251.2 buff/cache
MiB Mem :   3875.9 total,   1332.8 free,   1291.9 used,   1251.2 buff/cache
MiB Mem :   3875.9 total,   1332.8 free,   1291.9 used,   1251.2 buff/cache
MiB Mem :   3875.9 total,   1332.8 free,   1291.9 used,   1251.2 buff/cache
MiB Mem :   3875.9 total,   1332.8 free,   1291.8 used,   1251.2 buff/cache
MiB Mem :   3875.9 total,   1332.8 free,   1291.8 used,   1251.2 buff/cache
MiB Mem :   3875.9 total,   1332.6 free,   1292.1 used,   1251.2 buff/cache
MiB Mem :   3875.9 total,   1332.3 free,   1292.3 used,   1251.2 buff/cache
MiB Mem :   3875.9 total,   1333.1 free,   1291.5 used,   1251.3 buff/cache
MiB Mem :   3875.9 total,   1334.1 free,   1290.5 used,   1251.3 buff/cache
MiB Mem :   3875.9 total,   1334.3 free,   1290.3 used,   1251.3 buff/cache
MiB Mem :   3875.9 total,   1334.3 free,   1290.3 used,   1251.3 buff/cache
MiB Mem :   3875.9 total,   1329.4 free,   1295.2 used,   1251.3 buff/cache
MiB Mem :   3875.9 total,   1329.4 free,   1295.2 used,   1251.3 buff/cache
MiB Mem :   3875.9 total,   1329.9 free,   1294.6 used,   1251.3 buff/cache
MiB Mem :   3875.9 total,   1329.9 free,   1294.6 used,   1251.4 buff/cache
MiB Mem :   3875.9 total,   1329.7 free,   1294.9 used,   1251.4 buff/cache
MiB Mem :   3875.9 total,   1329.4 free,   1295.1 used,   1251.4 buff/cache
MiB Mem :   3875.9 total,   1309.0 free,   1315.5 used,   1251.4 buff/cache
MiB Mem :   3875.9 total,   1308.8 free,   1315.7 used,   1251.4 buff/cache
MiB Mem :   3875.9 total,   1309.1 free,   1315.4 used,   1251.4 buff/cache
MiB Mem :   3875.9 total,   1308.8 free,   1315.6 used,   1251.4 buff/cache
MiB Mem :   3875.9 total,   1309.6 free,   1314.9 used,   1251.4 buff/cache
MiB Mem :   3875.9 total,   1309.4 free,   1315.1 used,   1251.5 buff/cache
MiB Mem :   3875.9 total,   1309.4 free,   1315.0 used,   1251.5 buff/cache
MiB Mem :   3875.9 total,   1309.4 free,   1315.0 used,   1251.5 buff/cache
MiB Mem :   3875.9 total,   1309.4 free,   1314.9 used,   1251.6 buff/cache
MiB Mem :   3875.9 total,   1309.4 free,   1314.9 used,   1251.6 buff/cache
MiB Mem :   3875.9 total,   1309.4 free,   1314.9 used,   1251.6 buff/cache
MiB Mem :   3875.9 total,   1309.4 free,   1314.9 used,   1251.6 buff/cache
MiB Mem :   3875.9 total,   1309.4 free,   1314.9 used,   1251.6 buff/cache
MiB Mem :   3875.9 total,   1309.4 free,   1314.9 used,   1251.6 buff/cache
MiB Mem :   3875.9 total,   1309.4 free,   1314.9 used,   1251.6 buff/cache
MiB Mem :   3875.9 total,   1309.4 free,   1314.8 used,   1251.7 buff/cache
MiB Mem :   3875.9 total,   1309.4 free,   1314.8 used,   1251.7 buff/cache
MiB Mem :   3875.9 total,   1309.4 free,   1314.8 used,   1251.7 buff/cache
MiB Mem :   3875.9 total,   1309.4 free,   1314.8 used,   1251.7 buff/cache
MiB Mem :   3875.9 total,   1309.1 free,   1315.0 used,   1251.7 buff/cache
MiB Mem :   3875.9 total,   1309.1 free,   1315.0 used,   1251.7 buff/cache
MiB Mem :   3875.9 total,   1309.1 free,   1315.0 used,   1251.7 buff/cache
MiB Mem :   3875.9 total,   1309.1 free,   1315.0 used,   1251.7 buff/cache
MiB Mem :   3875.9 total,   1309.1 free,   1315.0 used,   1251.8 buff/cache
MiB Mem :   3875.9 total,   1309.1 free,   1315.0 used,   1251.8 buff/cache
MiB Mem :   3875.9 total,   1308.9 free,   1315.2 used,   1251.8 buff/cache
MiB Mem :   3875.9 total,   1310.9 free,   1313.2 used,   1251.8 buff/cache
MiB Mem :   3875.9 total,   1311.4 free,   1312.7 used,   1251.8 buff/cache
MiB Mem :   3875.9 total,   1311.6 free,   1312.4 used,   1251.8 buff/cache
MiB Mem :   3875.9 total,   1311.1 free,   1312.8 used,   1251.9 buff/cache
MiB Mem :   3875.9 total,   1311.1 free,   1312.8 used,   1251.9 buff/cache
MiB Mem :   3875.9 total,   1310.9 free,   1313.0 used,   1252.0 buff/cache
```

```
196 MiB Mem :   3875.9 total,   1310.9 free,   1313.0 used,   1252.0 buff/cache
197 MiB Mem :   3875.9 total,   1310.9 free,   1313.0 used,   1252.0 buff/cache
198 MiB Mem :   3875.9 total,   1310.6 free,   1313.2 used,   1252.1 buff/cache
199 MiB Mem :   3875.9 total,   1310.6 free,   1313.2 used,   1252.1 buff/cache
200 MiB Mem :   3875.9 total,   1310.4 free,   1313.4 used,   1252.1 buff/cache
201 MiB Mem :   3875.9 total,   1310.4 free,   1313.3 used,   1252.2 buff/cache
202 MiB Mem :   3875.9 total,   1310.4 free,   1313.3 used,   1252.2 buff/cache
203 MiB Mem :   3875.9 total,   1310.4 free,   1313.3 used,   1252.2 buff/cache
204 MiB Mem :   3875.9 total,   1310.4 free,   1313.3 used,   1252.2 buff/cache
205 MiB Mem :   3875.9 total,   1310.4 free,   1313.3 used,   1252.2 buff/cache
206 MiB Mem :   3875.9 total,   1312.9 free,   1310.8 used,   1252.2 buff/cache
207 MiB Mem :   3875.9 total,   1313.1 free,   1310.5 used,   1252.3 buff/cache
208 MiB Mem :   3875.9 total,   1414.2 free,   1209.5 used,   1252.3 buff/cache
209 MiB Mem :   3875.9 total,   1414.2 free,   1209.4 used,   1252.3 buff/cache
210 MiB Mem :   3875.9 total,   1414.2 free,   1209.4 used,   1252.3 buff/cache
211 MiB Mem :   3875.9 total,   1413.9 free,   1209.6 used,   1252.3 buff/cache
212 MiB Mem :   3875.9 total,   1413.7 free,   1209.9 used,   1252.3 buff/cache
213 MiB Mem :   3875.9 total,   1413.7 free,   1209.8 used,   1252.4 buff/cache
214 MiB Mem :   3875.9 total,   1413.2 free,   1210.3 used,   1252.4 buff/cache
215 MiB Mem :   3875.9 total,   1413.0 free,   1210.6 used,   1252.4 buff/cache
216 MiB Mem :   3875.9 total,   1413.0 free,   1210.5 used,   1252.4 buff/cache
217 MiB Mem :   3875.9 total,   1417.7 free,   1205.8 used,   1252.4 buff/cache
218 MiB Mem :   3875.9 total,   1417.7 free,   1205.8 used,   1252.4 buff/cache
219 MiB Mem :   3875.9 total,   1418.2 free,   1205.3 used,   1252.4 buff/cache
```

DECLARAÇÃO

Declaro, sob compromisso de honra, que o trabalho apresentado nesta dissertação, com o título *"Intel Analysis - Aplicação para análise de dados de segurança"*, é original e foi realizado por Estudante Martinho José Ferreira Gonçalves (2220280) sob orientação de Professor Especialista Carlos Manuel Gonçalves Antunes (carlos.antunes@ipleiria.pt).

Leiria, Setembro de 2024

Estudante Martinho José Ferreira Gonçalves