Apache Metron
Implantação de um Sistema de SOC

Elaborador:	Raphael Gustavo da Silva Boa Nova
Orientador:	Prof. Mestre Edson Roberto Gaseta

FACULDADE DE TECNOLOGIA DE AMERICANA

FICHA CATALOGRÁFICA – Biblioteca Fatec Americana - CEETEPS

Dados Internacionais de Catalogação-na-fonte

N811a NOVA, Raphael Gustavo da Silva Boa

Apache Metron: implantação de um sistema de SOC. / Raphael Gustavo da Silva Boa Nova. – Americana, 2019.

10f.

Relatório técnico (Curso de Tecnologia em Segurança da Informação) -- Faculdade de Tecnologia de Americana – Centro Estadual de Educação Tecnológica Paula Souza

Orientador: Prof. Ms. Edson Roberto Gaseta

1 Segurança em sistemas de informação I. GASETA, Edson Roberto II. Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia de Americana

RAPHAEL GUSTAVO DA SILVA BOA NOVA

APACHE METRON:
IMPLANTAÇÃO DE UM SISTEMA DE SOC

Trabalho de graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Segurança da Informação pelo CEETEPS/Faculdade de Tecnologia – FATEC/ Americana.

Área de concentração: Segurança da Informação

Americana, 10 de junho de 2019.

Banca Examinadora:

Edson Roberto Gaseta (Presidente)
Mestre
Fatec Americana

Rogério Nunes de Freitas (Membro)
Mestre
Fatec Americana

Elton Rafael Maurício da Silva Pereira (Membro)
Mestre
Fatec Americana

FACULDADE DE TECNOLOGIA DE AMERICANA

SUMÁRIO

Lista de figuras

1 Objetivo deste documento

Este documento descreve a implantação de uma instância da plataforma Apache Metron nas instalações da empresa Kryptus S.A.

O Metron é uma solução de código aberto de SIEM (*Security Information and Event Management*) mantida pela Fundação Apache. De acordo com o website da Imperva, um SIEM é “um conjunto de ferramentas e serviços que oferecem uma visão holística da segurança da informação de uma organização”.

Estas ferramentas, usualmente otimizadas para *big data*, coletam e analisam dados de logs reportando ameaças e eventos de segurança, além de monitorar sistemas em tempo real, correlacionar eventos e ameaças e enviar notificações para as partes responsáveis.

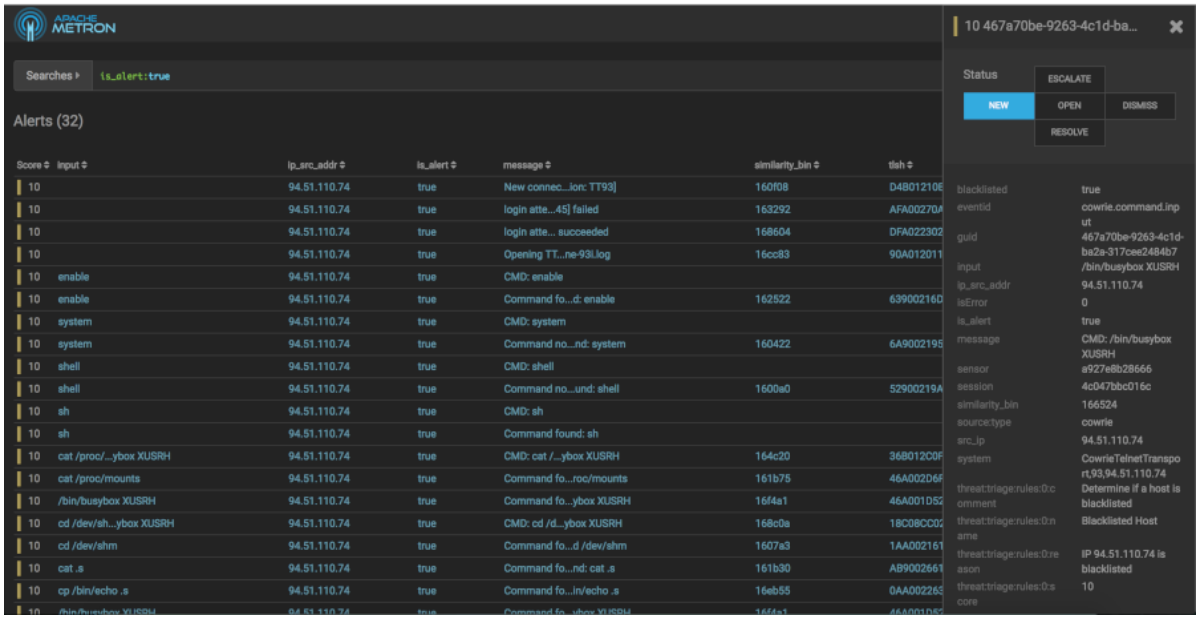
Soluções de SIEM como o Metron se fazem necessárias hoje devido ao grande volume de dados gerados por sistemas computacionais, exigindo a automação do processo de categorização e triagem de eventos para possibilitar a análise de forma apropriada e o foco nos eventos de maior importância. Sem este tipo de ferramenta, um analista de segurança não seria capaz de encontrar os eventos de maior prioridade em meio a milhares de outros eventos e falsos positivos, criando pontos cegos no monitoramento e deixando a infraestrutura vulnerável a ataques.

Outra questão abordada pela plataforma é a facilidade de implantação: devido à quantidade de ferramentas necessárias para este tipo de tarefa, tais plataformas podem se tornar muito complicadas de se implantar.

É preciso configurar toda a instrumentação para as diferentes fases de tratamento dos dados, e cada ferramenta deve estar ajustada precisamente para que possa tratar um grande volume de dados sem esgotar os recursos computacionais disponíveis. Isso exige muito conhecimento dos profissionais responsáveis pela implantação e manutenção do sistema, encarecendo muito o projeto. A proposta do Metron é prover uma solução *out-of-the-box* capaz de competir com as soluções pagas existentes no mercado.

A Figura 1 mostra a interface do Metron onde são exibidos os alertas de segurança após o processamento. Esta interface permite a um analista de segurança visualizar os detalhes de cada alerta, filtrar a lista para exibir apenas os alertas que obedeçam os critérios desejados e fazer a triagem do alerta.

Figura 1: Interface do Metron mostrando alertas de segurança em uma topologia.



Fonte: METRON DOCUMENTATION (2018).

O uso desta plataforma na Kryptus tem como objetivo final criar um centro de operações de segurança, ou SOC (*Security Operations Center*), possibilitando a detecção de ameaças em tempo real e uma rápida resposta à ataques. Esta primeira implantação deve servir de prova de conceito, coletando dados reais da infraestrutura da empresa e dados sintéticos que simulem eventos reais para que os responsáveis pela segurança de TI possam avaliar a plataforma e suas funcionalidades de acordo com as suas necessidades.

2 A necessidade de um SOC

A informação, embora abstrata, é um ativo como qualquer outro para uma organização. E como qualquer outro ativo, ela tem um valor e precisa ser devidamente protegida. A segurança da informação se fundamenta em preservar a confidencialidade, a integridade e a disponibilidade da informação, visando a continuidade de uma organização e de seu negócio. Segundo Sêmola (2014, p. 43), estes três princípios podem ser definidos da seguinte forma:

> **Confidencialidade:** Toda informação deve ser protegida de acordo com o grau de sigilo de seu conteúdo, visando a limitação de seu acesso e uso apenas às pessoas a quem é destinada.
>
> **Integridade:** Toda informação deve ser mantida na mesma condição em que foi disponibilizada pelo seu proprietário, visando protege-la contra alterações indevidas, intencionais ou acidentais.
>
> **Disponibilidade:** Toda informação gerada ou adquirida por um indivíduo ou instituição deve estar disponível aos seus usuários no momento em que eles necessitem delas para qualquer finalidade.

O conceito de segurança da informação precede a existência de computadores digitais e não deve ser confundido como dependente destes; muito antes do advento dos computadores, as organizações já classificavam suas informações armazenadas em pastas e arquivos de acordo com seu grau de confidencialidade, plastificavam os documentos para que se mantivessem íntegros e mantinham cópias extras dessas informações para garantir que estivessem sempre disponíveis. O surgimento dos computadores e da internet apenas possibilitaram um processamento e uma troca de informações de formas mais eficazes.

Esta agilidade no tratamento das informações permite às organizações reagir de forma mais rápida aos acontecimentos, tornando-as mais eficientes. Em contrapartida, um fluxo maior de informações também introduz riscos. Este compartilhamento de informações dificulta o controle sobre a informação que deve ser mantida em sigilo; deve-se cuidar para que a informação tenha por onde fluir sem comprometer o sigilo da informação.

Para garantir a segurança de seus ativos físicos, uma organização pode se dispor de controles para protegê-los das ameaças: cofres, cadeados, portas com controle de acesso e cercas elétricas podem ser usados como empecilhos. Porém, em muitos casos o uso destes empecilhos não é o suficiente; é preciso vigiar os ativos e garantir que nenhum adversário ultrapasse estes obstáculos colocados entre ele e os ativos da organização. Câmeras de vigilância, alarmes de segurança e sensores de presença podem também ser usados para permitir que estes ativos estejam sempre sob a vigilância de alguém capacitado.

De forma semelhante, no plano cibernético muitas vezes não basta apenas implantar controles como firewalls e antivírus para impedir acessos não autorizados à informação. Um adversário pode eventualmente ser capaz de ultrapassar todas estas barreiras e, caso isso venha a acontecer, a organização precisa estar ciente disto imediatamente para que possa reagir de forma adequada. Da mesma forma que uma instalação física pode contar com um time de monitoramento que permita uma resposta rápida à invasões de propriedade, assim também é possível em um ambiente digital. É este o papel de um SOC em uma infraestrutura.

Um SOC é uma central de monitoramento no plano cibernético. Embora muitas ferramentas anti-invasão de rede como firewalls, IDSes (*Intrusion Detection Systems*) e IPSes (*Intrusion Prevention Systems*) sejam capazes de notificar uma falha de segurança a um administrador de redes, estas ferramentas em sua maioria não são integradas entre si e não permitem uma visão holística dos eventos que ocorrem em uma infraestrutura. Alertas distintos podem estar correlacionados com um único ataque, por exemplo, mas o administrador da rede não receberia nenhum indicativo disso.

Outro aspecto das infraestruturas de rede modernas que dificulta o controle sobre a segurança é o grande volume de dados. "Chegamos à era do *big data*, em que volumes maciços de informação são gerados, armazenados, manipulados e compartilhados o tempo todo" (SÊMOLA, 2014 p. 4). Este fluxo intenso de informações, em conjunto com a automatização de escaneamentos de vulnerabilidades e de ataques oriundos da internet, implica na geração de um grande volume de alertas de segurança, o que exige um tratamento de eventos de segurança também automatizado. Em muitos casos é humanamente impossível analisar todos os alertas gerados por firewalls e antivírus sem deixar escapar algum evento importante.

Uma plataforma de SOC deve se propor a priorizar e correlacionar estes eventos de segurança de forma a possibilitar uma triagem e uma resposta eficientes às ameaças. Alertas originados de ferramentas distintas podem ser enriquecidos com dados geográficos e informações vindas de bases de dados de ameaças (*threat intel*), correlacionados, categorizados e priorizados automaticamente, permitindo ao analista de segurança focar nos eventos de maior importância sem se preocupar com falsos positivos.

2.1 Uma visão geral da plataforma

A plataforma Metron integra diversas ferramentas (todas de código aberto), cada uma desempenhando um papel diferente no tratamento de eventos. Dentre estas ferramentas utilizadas pelo Metron, podemos listar:

- **Ambari:** Provisionamento de *clusters* para o processamento de *big data*;
- **Apache NiFi:** Coleta de telemetria / logs;
- **Apache Kafka:** Plataforma distribuída de transmissão de mensagens;
- **Apache Zookeeper:** Configuração e coordenação de sistemas distribuídos;
- **Apache Storm:** Computação distribuída em tempo real;
- **ElasticSearch:** Armazenamento de eventos a curto prazo (*hot storage*);
- **Hadoop:** Armazenamento de eventos a longo prazo (*warm storage*);
- **Kibana:** Painel de visualização de informações.

A plataforma em si é também de código aberto e seu código fonte, escrito em sua maior parte na linguagem de programação Java, está hospedado online no GitHub. A documentação do projeto o caracteriza como sendo “em essência, uma arquitetura Kappa com o Apache Storm como componente de processamento e o Apache Kafka como o barramento de dados unificado” (FUNDAÇÃO APACHE, 2019).

Kappa é uma arquitetura de software comumente usada para o processamento de *big data*, na qual os dados são armazenados em um log imutável (ao contrário das tradicionais bases de dados relacionais e bases de dados NoSQL que permitem a alteração dos dados após a inserção) e a partir deste log os dados são transmitidos e processados através de um sistema computacional e finalmente armazenados em bases de dados auxiliares para o uso.

Esta arquitetura foi descrita pela primeira vez por Jay Kreps, arquiteto-chefe de infraestrutura de dados no LinkedIn, como sendo uma evolução da arquitetura Lambda usualmente adotada por softwares que fazem o uso do Apache Storm. Em seu artigo onde descreve o funcionamento desta arquitetura, Kreps brinca que “talvez poderíamos chamá-la de arquitetura Kappa, embora talvez esta seja uma idéia muito simples para merecer uma letra grega”, fazendo uma alusão ao uso de uma letra grega para descrever a arquitetura Lambda.

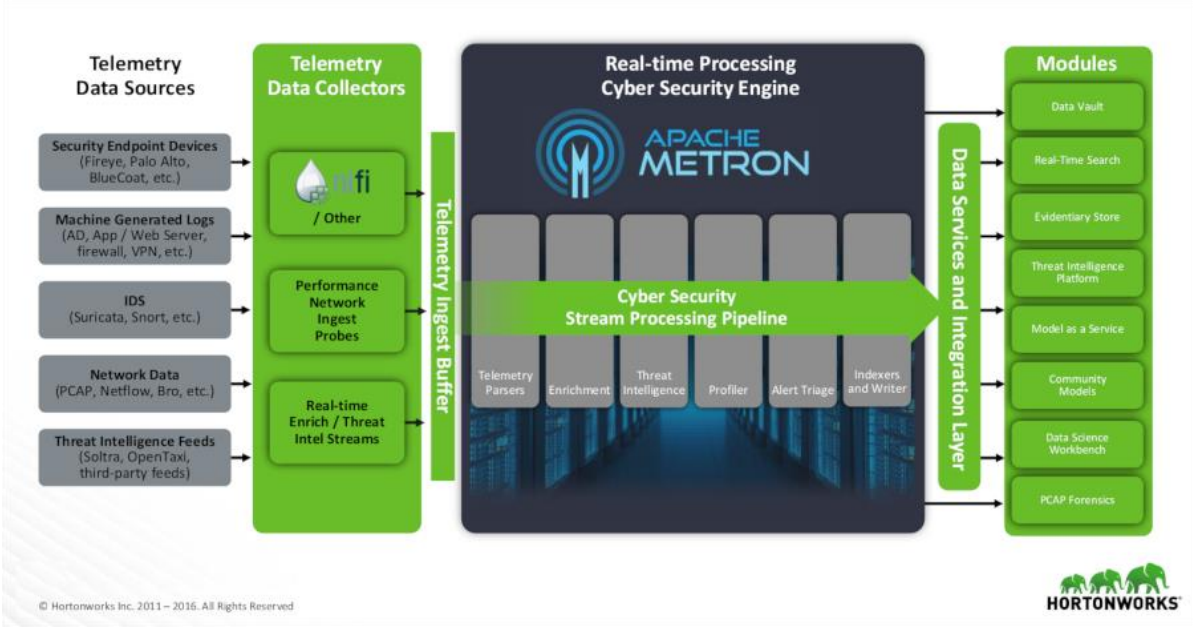
O Metron faz uso do Apache Kafka para o armazenamento de dados brutos provenientes de logs de sistema, firewalls, IDSes, IPSes e outras fontes de informação que são denominadas pelo Metron como **fontes de dados de telemetria**. Estes dados são coletados destas fontes e inseridos em tópicos no Apache Kafka para serem processados. O Metron então lê os dados adicionados nestes tópicos e os processa, para em seguida indexar a informação resultante no ElasticSearch, permitindo que ela seja consumida pela sua interface e/ou por outras ferramentas de análise de dados como o Kibana.

Este processamento é o coração do Metron: é neste passo que os dados são normalizados, enriquecidos com informações provenientes de bases de dados de ameaças cibernéticas e dados geográficos, e então aplicados em um algoritmo de aprendizado de máquina para que estes dados sirvam de insumo para os perfis de uso da rede organizados pela plataforma.

Os elementos desta arquitetura são mostrados de forma simplificada na Figura 2. A origem dos dados em diversas fontes de dados de telemetria é mostrada na primeira coluna, passando pelos coletores de dados de telemetria na coluna seguinte, e finalmente sendo processados pelo Metron e sendo enviado para módulos que farão o uso desta informação resultante.

Esta visão geral auxilia no entendendimento do papel de cada componente nesta arquitetura. Este conhecimento é importante para compreender a implantação da plataforma, descrita adiante.

Figura 2: Visão geral da arquitetura do Metron.

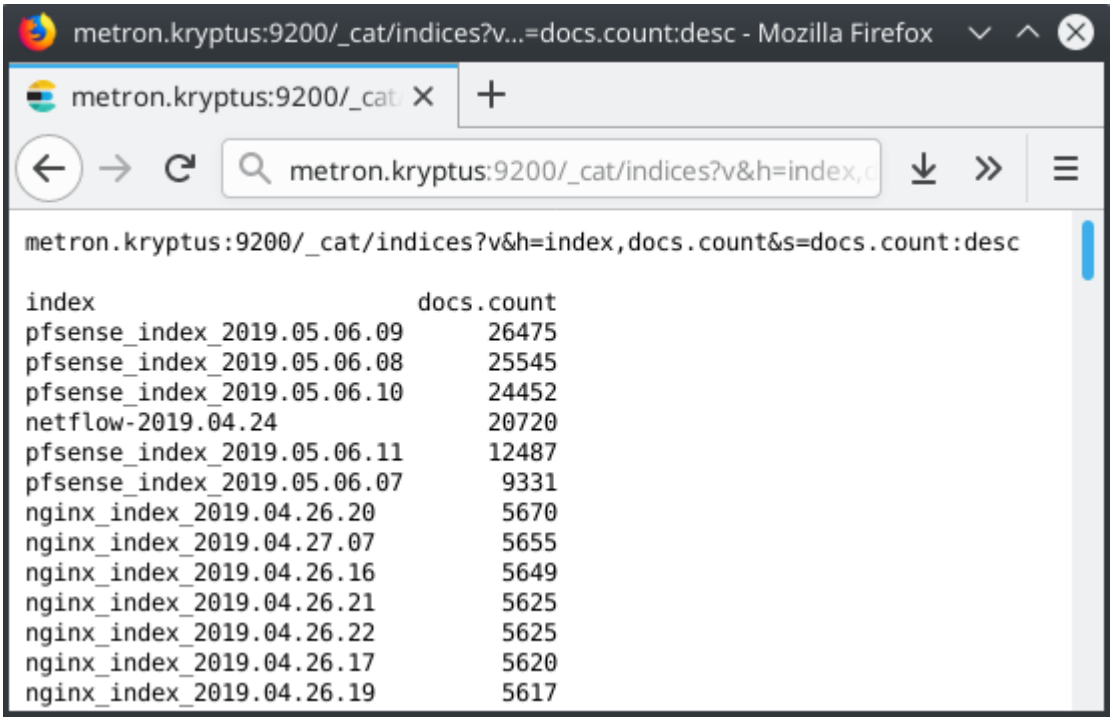


Fonte: DATA PLATFORM CONFERENCE TOKYO 2017.

3 O ambiente na Kryptus

A Figura 3 exemplifica o volume de eventos gerados na topologia de rede da Kryptus. Esta figura mostra os resultados de uma consulta à uma instância do ElasticSearch na qual estes eventos são armazenados, onde cada linha representa um índice contendo os eventos de um período de uma hora originados de uma única fonte de eventos. Por exemplo, podemos ver que o maior índice contém 26.475 documentos, ou eventos, provenientes do pfSense (uma distribuição de firewall/roteador de código aberto baseada no FreeBSD) entre 9 AM e 10 AM do dia 6 de maio de 2019. Estes números ilustram a impossibilidade de um indivíduo monitorar e filtrar todos estes eventos em tempo hábil.

Figura 3: Índices de eventos de segurança armazenados no ElasticSearch.



Fonte: Próprio autor.

Estes eventos não necessariamente indicam atividades maliciosas, mas este alto volume de dados é o resultado de uma decisão tomada pelo administrador de rede que ficou encarregado de configurar a ferramenta de monitoramento: é melhor configurá-la para que envie alertas de todos os eventos (gerando muitos falsos positivos)? Ou é melhor suprimir eventos mais comuns correndo-se o risco de deixar escapar algum alerta importante? Este é um dilema muito comum de se enfrentar em algum momento em uma organização. A organização cresce, a sua topologia de rede se expande e com ela o volume de dados e de eventos, até chegar a um ponto no qual o monitoramento de forma manual não é mais viável. E qual a solução para este dilema? Ser enterrado por falsos positivos ou fechar os olhos para os eventos que acontecem na sua rede?

Esta é uma falsa dicotomia que assume que o trabalho de monitoramento não pode ser automatizado. De fato, quando se trata de monitoramento manual, nenhuma das alternativas é aceitável. Os ativos ficam vulneráveis em ambas as situações. Isso demonstra a necessidade de uma automatização deste tratamento de eventos, e para isso

se faz necessária a geração de muitos eventos que possam ser correlacionados a fim de indicar quais são falsos positivos e quais indicam brechas de segurança reais.

Tomemos como exemplo um ataque no qual um adversário obtém as credenciais de rede de um colaborador (por meio de engenharia social, por exemplo) e consegue efetuar o login em uma das máquinas expostas na rede. Tal evento geraria uma entrada de log no arquivo `/var/log/secure` semelhante à esta:

```
Abr 22 10:22:28 kryptus01 sshd[1427]: Accepted password for alice from 78.110.48.26 port 41066 ssh2
```

Este evento se perderia em meio a inúmeros outros semelhates à ele, e mesmo que um administrador de rede o visse é improvável que ele se destacasse em meio a todos os outros. Uma ferramenta automatizada, porém, seria capaz de geolocalizar o IP registrado nessa linha de log, correlacioná-lo com outros registros semelhantes e identificar que este acesso é, com grande probabilidade, malicioso. Esse endereço de IP é originário da Rússia, e correlacionando este evento com outro evento de login do mesmo usuário agora originado em São Paulo, a ferramenta seria capaz de determinar que é impossível o mesmo indivíduo se deslocar entre estas duas regiões tão distantes uma da outra em tão curto espaço de tempo. De fato, este é um dos casos de uso ilustrados na documentação do Metron.

Embora sua topologia de rede esteja protegida por outras soluções renomadas no mercado, a Kryptus sentiu a necessidade de encontrar uma solução ainda melhor para proteger a sua rede e as de seus clientes. Por questões de confidencialidade, mais detalhes sobre a topologia da organização não podem ser descritos neste relatório, assim como qualquer demonstração de uma vulnerabilidade real. De qualquer forma, este cenário descrito deve servir de exemplo motivador da necessidade de implantação de uma plataforma de SOC como o Apache Metron.

4 Desenvolvimento do projeto

A plataforma oferece diferentes formas de implantação, como por exemplo usando máquinas virtuais, imagens do Docker ou até mesmo uma implantação automatizada no AWS (*Amazon Web Services*, o serviço de computação na núvem da Amazon). Por questões de custo e de disponibilidade de recursos optamos pelo uso do AWS para este projeto, porém sem uma implantação automatizada porque este processo criaria dez instâncias para a plataforma, o que encareceria muito a solução. Esta implantação manual do Metron, atualmente na versão 0.7.0, envolve ajustar as ferramentas utilizadas de acordo com o caso de uso, após a sua instalação.

4.1 Infraestrutura de hardware e sistema operacional

Para esta implantação foram disponibilizadas duas instâncias do Amazon EC2 do tipo a1.4xlarge que contam com 16 processadores, 32Gb de memória ram e 1Tb de armazenamento cada. Isso nos permitiria uma implantação mínima, utilizando uma instância como nó de processamento e a outra como nó de armazenamento de dados. Cada instância foi provisionada com o sistema operacional CentOS7 em uma instalação mínima, sem interface gráfica ou softwares adicionais além do estritamente necessário.

Antes de dar início à instalação das ferramentas, o sistema operacional precisou passar por ajustes finos de acordo com o manual de implantação do Metron. Alguns exemplos destes ajustes finos incluem ampliar os limites de recursos computacionais que podem ser alocados por cada serviço (no arquivo `/etc/security/limits.conf`), e desabilitar no kernel o uso do recurso de *Transparent Huge Pages* que, de acordo com a RedHat, não é recomendado para cargas de trabalho de bancos de dados. Estes ajustes são necessários para melhorar o desempenho das ferramentas, possibilitando um fluxo de dados em maior volume.

Para compilar o Metron a partir de seu código fonte também foi necessário instalar o JDK (*Java Development Kit*, ou Kit de Desenvolvimento Java) e o Maven. Este processo compila o código da plataforma e gera pacotes RPM (*RedHat Package Management*, o formato de pacotes de software pré-compilado usado pelo CentOS) para a instalação usando o gerenciador de pacotes do CentOS.

4.2 Implantação da plataforma

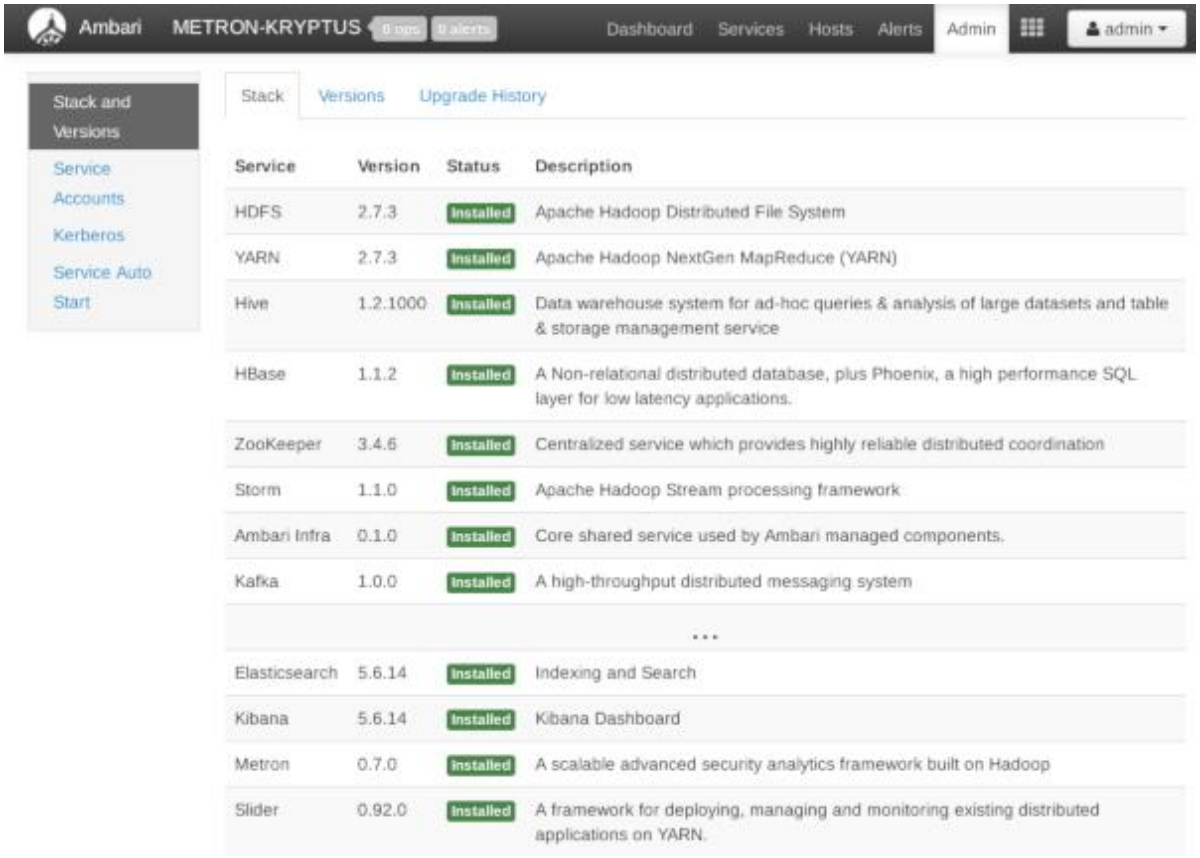
Com o sistema operacional ajustado e o com o Metron compilado, a primeira ferramenta a ser instalada é o Ambari. Esta ferramenta nos possibilita automatizar a instalação das demais e, mais adiante, monitorar a saúde destas ferramentas de forma centralizada.

O Ambari é capaz de fazer o uso de repositórios de software locais para a instalação de software, o que podemos explorar para fazer com que ele instale o Metron a partir dos pacotes RPM gerados. Foi criado um repositório local em cada máquina, o Ambari foi instalado e iniciado e então a instalação pôde ser realizada a partir da interface web desta ferramenta. Ela oferece um assistente de instalação passo-a-passo que permite quais pacotes serão instalados e em quais máquinas isso ocorrerá. O resultado deste processo é exibido na Figura 4.

Ao final deste processo todos os componentes essenciais da plataforma estão instalados e em execução (Kafka, Storm, Metron, Hadoop, ElasticSearch, entre outros), porém o Metron

ainda não está processando dados. Isso porque não existe nenhum processo alimentando o log de dados da plataforma, neste caso o Kafka. Para que este processamento ocorra, os dados provenientes de fontes de telemetria devem ser inseridos no log de dados, e para cada fonte de telemetria diferente o Metron precisa que seja configurado um sensor. Um sensor para o Metron é uma topologia (um conjunto de processos sendo executados no Apache Storm) capaz de ler esta telemetria de um tópico do Kafka, analizá-la (*parse*) de acordo com o formato adotado pela ferramenta de segurança e convertê-la para o formato JSON (*JavaScript Object Notation*), que é o formato usado internamente pelo Metron para o processamento destes dados.

Figura 4: Interface do Ambari mostrando alguns dos serviços que foram instalados.



Fonte: Próprio autor.

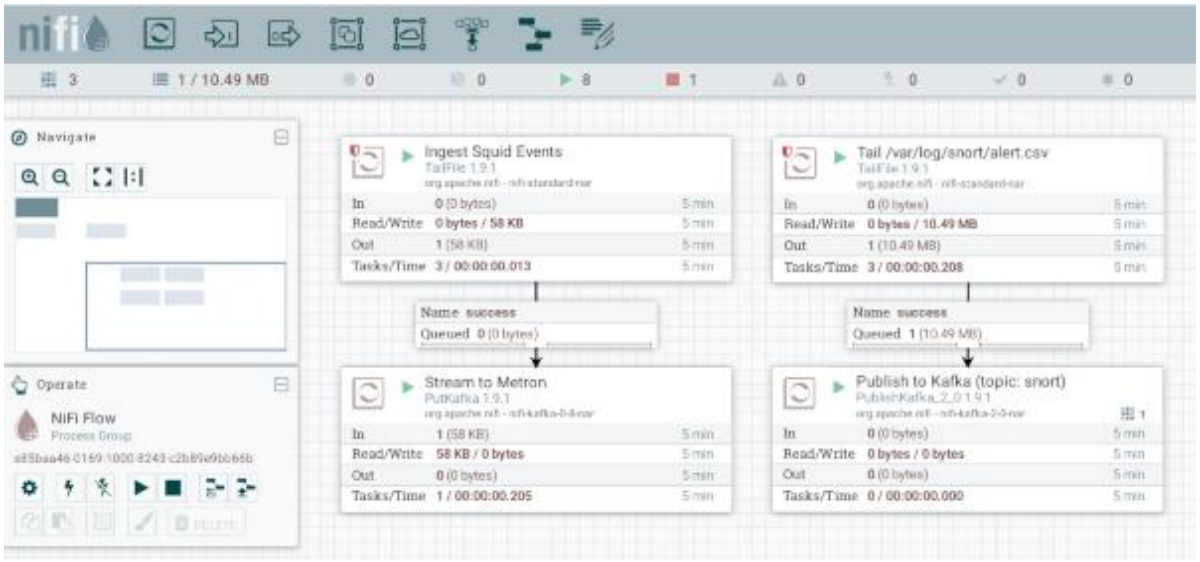
O Metron oferece alguns sensores por padrão, mas para processar dados de qualquer outra fonte de telemetria é necessário configurar um sensor próprio. Isso inclui criar um tópico novo no Kafka para servir de entrada para esses dados, que seja capaz de converter os dados para JSON. A plataforma funciona de forma modular, aceitando que estes novos componentes sejam configurados e implantados sem a necessidade de recompilar a solução por completo.

Essa implantação exigiu a criação de sensores para processar dados provenientes do proxy Squid e dos logs da ferramenta de VPN (*Virtual Private Network*) OVPN, além de sensores distintos para o IDS Snort e para o HIDS (*Host IDS*) OSSEC. Além disso, estas ferramentas não possuem um mecanismo próprio de envio de dados para o Kafka, então foi necessário o uso do Apache NiFi para a coleta destes logs e envio para tópicos

distintos. Esta ferramenta oferece ao usuário uma vasta coleção de componentes que podem ser encadeados para formar uma cadeia de processamento, semelhante à uma plataforma de programação visual, e é capaz de ler dados de diversas fontes diferentes e enviá-los para outros processos usando diversos protocolos de transporte (como TCP e UDP) e de aplicação (como HTTP, SMTP ou neste caso o protocolo binário do Kafka).

O Apache NiFi, mostrado na figura 5, foi instalado em uma das duas máquinas e cadeias de processamento (ou *pipelines*) foram criadas para coletar os dados de logs do Snort, do Squid, do OSSEC e do OVPN, além de receber dados de Syslog e de NetFlow via portas UDP, enviando todos estes dados para o Kafka.

Figura 5: Interface do NiFi mostrando duas pipelines de coleta de telemetria.



Fonte: Próprio autor.

Pela interface web do Metron Config foram criados sensores para estas fontes de telemetria. Quando feito corretamente, este processo termina com a inserção destas informações de eventos salvos no ElasticSearch, podendo ser consultados via REST API.

Muitas vezes, porém, alguma configuração errada resultava na falha do processamento destas informações; a inserção de dados no Kafka ocorria de forma correta, porém não eram criados índices no ElasticSearch. Para diagnosticar esse tipo de problema era necessário acompanhar os dados conforme passavam por toda a arquitetura, checando log por log de cada ferramenta até encontrar o ponto de falha (por exemplo, uma exceção lançada pelo sensor rodando no Apache Storm ou uma réplica do Kafka fora do ar).

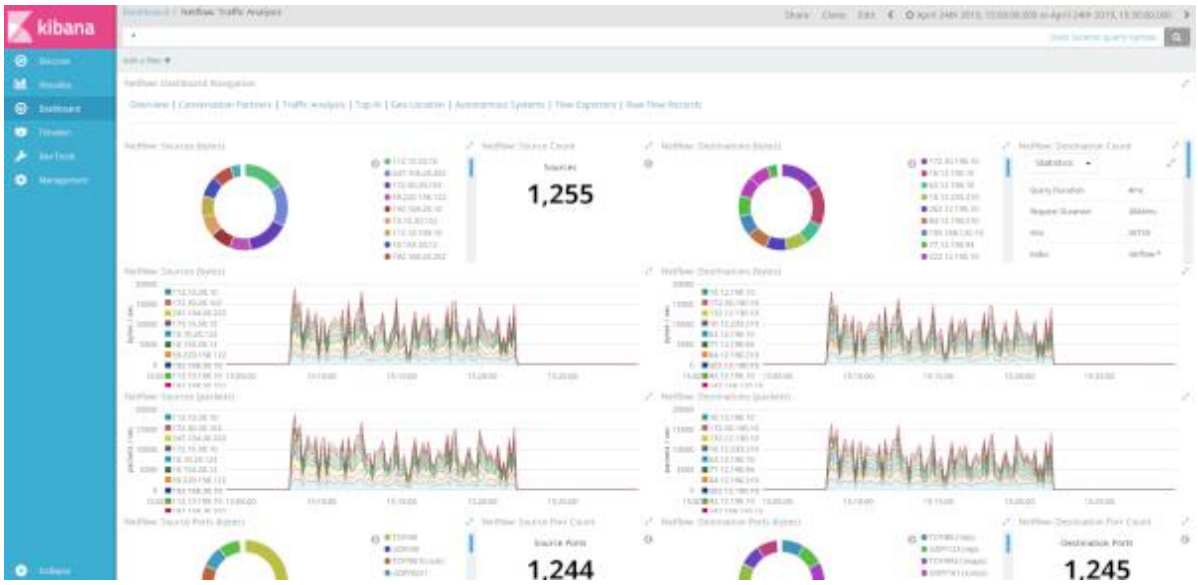
5 Resultados

Depois de concluída a implantação, dados da topologia de rede provenientes de múltiplas fontes de telemetria puderam ser consultados, visualizados e filtrados através da interface do Metron Alerts e do Kibana de forma centralizada, como é demonstrado na Figura 6: gráficos mostrando o volume de tráfego de rede, origens e destinos mais comuns e os protocolos mais utilizados são gerados automaticamente a partir da informação resultante do processamento da plataforma.

Dados sintéticos também foram usados para gerar um maior volume de dados, permitindo a avaliação da plataforma de modo mais próximo da realidade sem que houvesse a necessidade de configurar outras fontes de dados de telemetria extras. Esses dados sintéticos também permitiram a simulação de eventos de ameaças e falhas de segurança sem que a rede fosse exposta à estes elementos.

Com a plataforma funcionando de forma estável, customizá-la exigiu um esforço bem menor do que a implantação, permitindo com que técnicos menos capacitados pudessem adicionar sensores novos e acompanhar a saúde dos componentes pela interface do Ambari.

Figura 6: Interface do Kibana mostrando gráficos de dados sintéticos.



Fonte: Próprio autor.

6 Conclusões e considerações finais

Após essa implantação mínima, foi possível perceber que sem o uso das funcionalidades mais avançadas da plataforma como o enriquecimento de dados com *threat intel* e o sem o *profiler* (componente que usa aprendizado de máquina para identificar perfis de uso da rede), a informação apresentada não oferece uma compreensão mais aprofundada dos eventos que ocorrem na rede. A plataforma é, neste nível de implantação descrito, apenas um agregador e visualizador de eventos de segurança, com pouca capacidade de agir como uma ferramenta de inteligência.

A complexidade da ferramenta e a falta de maturidade do código ainda oferecem algumas barreiras para o seu uso. Existem pontos de falha e o monitoramento da saúde da plataforma exige um bom conhecimento de todos os seus componentes e de como eles interagem. Algumas das funcionalidades sequer estão implementadas por completo, como por exemplo a interface de análise de captura de pacotes de rede que exigiu mudanças no código após a implantação para exibir os resultados na tela de forma correta. A ferramenta exige bastante investimento de trabalho para instalar, manter e usar, dado que seu uso também requer analistas capacitados para acompanhar os eventos de segurança e tomar as medidas adequadas de correção.

Mesmo com essas dificuldades, o projeto continuará em andamento devido à demanda para esse tipo de tecnologia. Uma segunda fase foi iniciada, agora usando servidores físicos dentro do precinto da empresa.

REFERÊNCIAS BIBLIOGRÁFICAS:

DATA PLATFORM CONFERENCE TOKYO 2017. **Hortonworks cybersecurity platform deep dive**. 2017. Disponível em: <http://dataplatform.jp/program/files/A-7.pdf>. Acesso em 05 maio 2019.

FUNDAÇÃO APACHE. **Apache Metron**. 2019. Disponível em: <https://github.com/apache/metron/blob/master/README.md>. Acesso em 08 maio 2019.

IMPERVA. **Security information and event management (SIEM)**. Disponível em: <https://www.imperva.com/learn/application-security/siem>. Acesso em 20 abr. 2019.

KREPS, J. **Questioning the Lambda Architecture**. 2014. Disponível em: <https://www.oreilly.com/ideas/questioning-the-lambda-architecture>. Acesso em 08 maio 2019.

METRON DOCUMENTATION. **Use-cases**: forensic clustering. 2018. Disponível em: <https://metron.apache.org/current-book/use-cases/forensic_clustering>. Acesso em 05 maio 2019.

RED HAT. **Huge pages and transparent huge pages**. Disponível em: <https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/performance_tuning_guide/s-memory-transhuge>. Acesso em 26 abr. 2019.

SÊMOLA, M. **Gestão da segurança da informação**: uma visão executiva. 2. ed. Rio de Janeiro: Elsevier, 2014.

SHIROLKAR, A. **Installation of HDP2.4 with Ambari got stuck**. 2017. Disponível em <https://community.hortonworks.com/questions/127834/installation-of-hdp24-with-ambari-got-stuck.html>. Acesso em 08 maio 2019.