UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

PLATAFORMAS SIEM: IMPLEMENTAÇÃO, CONFIGURAÇÃO E GESTÃO

João Pedro Figueiredo da Cruz

PROJECTO

VERSÃO PÚBLICA

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Arquitectura, Sistemas e Redes de Computadores

2012

UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

PLATAFORMAS SIEM: IMPLEMENTAÇÃO, CONFIGURAÇÃO E GESTÃO

João Pedro Figueiredo da Cruz

PROJECTO

Projecto orientado pela Prof. Doutora Maria Dulce Pedroso Domingos
e co-orientado pelo Mestre Sérgio Valentim Costa de Sá

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Arquitectura, Sistemas e Redes de Computadores

2012

Agradecimentos

Agradeço à minha família por todo o apoio que sempre me deram e por me terem ajudado a alcançar todos os objectivos que tracei.

Aos meus amigos que apareceram no meu caminho na FCUL e que por uma ou outra razão, por mais mínima e insignificante que possa parecer, fizeram com que tudo fosse mais fácil, nomeadamente: Sara, *TAP*, Sá, Marquito, Esteves, *Pimba*, Edgar, André, *Tartaruga*, Andreia, *Bong*, Casais, *Engenheiro*, Bento e *Balsas*.

Agradeço também aos meus colegas da Unisys pelo excelente acolhimento e óptimo ambiente proporcionado e por me terem ajudado a ultrapassar todas as dificuldades que surgiam diariamente, nomeadamente: João Silva, Nelson Silva (ex-Unisys), Gonçalo Martins, Luis Lobo e Jorge Libório.

Quero também agradecer ao meu orientador na Unisys, Sérgio Sá, pela aposta feita em mim e pela disponibilidade mostrada para a elaboração deste documento.

Por último, agradeço à minha orientadora na Faculdade de Ciências, a Professora Dulce Domingos, que se mostrou sempre incansável para aperfeiçoar este relatório, de forma a poder atingir todos os meus objectivos na disciplina de Projecto em Engenharia Informática.

Resumo

As organizações estão cada vez mais dependentes dos meios tecnológicos e, dependendo do ambiente em que se inserem, lidam diariamente com informação sensível: Informações pessoais, dados de cartões de crédito, entre outros. Reter toda esta informação torna-as num alvo primordial a atacantes. Com a sofisticação dos ataques a aumentar, a segurança da periferia da infra-estrutura já não é suficiente. É necessário ter uma visão global sobre o que está a acontecer na rede a todo o momento.

As **plataformas SIEM (Security Information and Event Management)** estão cada vez mais a emergir no mercado actual, surgindo não só como necessidade mas como uma obrigação. Têm como objectivo recolher *logs* dos dispositivos e aplicações, consolidando-os num único repositório e correlacionando a informação relevante de forma a detectar automaticamente comportamentos anómalos para que rapidamente sejam eliminadas as hipóteses de um ataque bem sucedido.

A instalação destas plataformas requer trabalho a vários níveis: Desde a análise de requisitos em conjunto com o cliente para determinar quais os *logs* a recolher e como o fazer, à instalação de agentes que processem esses registos e ao desenvolvimento de agentes que trabalhem sobre *logs* desconhecidos ao SIEM, até à criação de mecanismos na solução que correlacionem esses eventos de forma a identificar ameaças.

Este documento descreve a configuração e personalização de uma solução SIEM numa organização do sector financeiro que necessita também desta ferramenta para demonstrar que cumpre certas normas regulatórias, facto que torna o projecto mais complexo. Como tal, carece de especial atenção dado a rigidez das mesmas. Uma particularidade interessante neste trabalho é o facto de o cliente usar muitas aplicações desenvolvidas internamente cujos *logs* não são compreendidos nativamente pela plataforma. A sua inclusão, não só levanta questões sobre a forma de os processar, mas também como os utilizar para efeitos de monitorização.

Palavras-chave: Segurança; Recolha de *logs*; Correlação de eventos; Monitorização de redes; Resposta a incidentes;

Abstract

Nowadays, organizations depend more and more on their technology and they deal with many confidential information: Personal data, credit card data, and so on. The storage of all this information makes this organizations a desirable target for attackers. These attacks are more powerful today, so periferic security of the core infrastructure is not sufficient anymore. Having a global vision of what's happening at any moment is mandatory.

SIEM solutions (Security Information and Event Management) are more often emerging in today's market, not only as a need but also as an obligation. Their objectives are to collect *logs* from devices and applications, store them in a centralized repository and correlate all the relevant information to automatically detect abnormal behaviors and, therefore, quickly eliminate the possibilities of a well succeeded attack.

Deploying solutions require work at a few levels: Analyzing customer requirements in order to determine which *logs* have to be collected and how they will be collected, configuring the collector agents, developing custom collector agents to process *logs* with an unknown format, and creating resources to monitoring and correlating the data.

This report describes the configuration and customization of a SIEM solution in a financial organization, who needs this platform to be compliant with some standards. The customer have many applications, which were developed internally and, consequently, have a *log* format unknown to the SIEM solution. Including this data sources requires additional work, not only because there are no agents that natively processes them, but also because it's necessary to include them at the correlation level with other known *log* sources.

Keywords: Information Security, Log collection, Event correlation, Network Monitoring, Incident response

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Este capítulo introdutório tem o objectivo de contextualizar as plataformas SIEM e em que ambientes podem e devem ser utilizadas. Apresenta os objectivos do projecto e a estrutura do resto deste documento.

1.1 Motivação

Hoje em dia, qualquer organização, por mais pequena que seja, é dependente da sua infra-estrutura tecnológica. Ataques que ponham em causa a confidencialidade, integridade ou disponibilidade de servidores de Bases de Dados, servidores de autenticação, ou outro elemento crítico, farão com que essa empresa deixe de funcionar e, dependendo dos seus objectivos, os danos poderão ser mais ou menos devastadores. Até há pouco tempo, para se defenderem contra ataques, os esforços estavam dirigidos para fortalecer a segurança periférica: *firewalls* nos pontos de entrada possíveis na infra-estrutura, IDS (Intrusion Detection System) com sensores em locais estrategicamente definidos, etc.

Este tipo de defesa periférica é essencial, mas não é suficiente. Tem-se assistido a um aumento exponencial no número de ataques bem sucedidos às maiores companhias mundiais e, olhando para esses exemplos, as organizações devem perguntar-se se esta segurança periférica é suficiente. *Firewalls*, detectores de intrusões e outros mecanismos fazem o seu papel e registam toda a actividade que executam. Esta informação tem uma importância fundamental para criar uma barreira adicional à já existente na periferia.

Neste contexto, surgem as plataformas **SIEM - Security Information and Event Management**. Centralizando em si esse conjunto de registos de actividade, estas soluções permitem aos seus utilizadores monitorizar a segurança da infra-estrutura em tempo real, bem como automatizar o processo de análise. A detecção de padrões comportamentais indesejáveis em tempo útil é de vital importância para criar uma defesa forte contra eventuais adversários. Com estas ferramentas não só é possível monitorizar os elementos já referidos da periferia da rede, mas também outros elementos internos críticos da organização como servidores de autenticação, servidores Web, *antivírus*, sistemas opera-

tivos, acessos físicos, aplicações, entre outros exemplos. Com esta informação recolhida e consolidada num repositório central é possível correlacionar os registos de actividade dos activos e aplicações da infra-estrutura, detectando atempadamente ataques que ponham em causa a segurança do sistema. Outros elementos, como relatórios de actividade, *dashboards* que permitem visualização gráfica de eventos e notificações podem também ser criados para auxiliar este processo de monitorização.

Globalmente, estas soluções assumem uma arquitectura tripartida que aloja:

1. Os vários activos que geram os registos de actividade,
2. Os agentes que recolhem esses *logs*, e
3. O elemento centralizador que recebe os dados dos agentes, com capacidades de correlacionar eventos mediante um conjunto de regras.

Este relatório apresenta o trabalho realizado para implementar uma solução SIEM numa entidade ligada ao ramo do sector financeiro. O seu objectivo, ao utilizar uma solução SIEM, para além da constante monitorização da infra-estrutura, será o de estar em conformidade segundo uma série de normas regulatórias. A análise de requisitos inicial ditou que teriam de ser monitorizadas mais de 80 dispositivos críticos, entre *firewalls*, IDS, servidores de autenticação e outras aplicações fulcrais para o bom funcionamento da organização. Cada um pode gerar mais do que um tipo diferente de *log*, sendo que uma grande percentagem desses registos estão num formato não suportado nativamente pela plataforma, sendo necessário criar agentes personalizados que processem estes *logs*.

1.2 Objectivos

Ao utilizar a plataforma SIEM, esta organização do ramo do sector financeiro (doravante chamada de "cliente") pretende sobretudo manter uma visão global da sua infra-estrutura, mas também, para efeitos de auditoria externa, mostrar-se cumpridora de um conjunto de regras de segurança que têm como finalidade auxiliar este tipo de empresas a defender-se de ataques externos. Nomeadamente, as normas que terão de ser cumpridas são: PCI DSS (Payment Card Industry Data Security Standard), Visa e Mastercard. Genericamente, estas regras obrigam a que se mostrem evidências de centralização de registos de actividade por um longo período temporal e de monitorização dos *logs* em tempo real, entre outros.

Com este documento pretende-se mostrar o trabalho realizado para instalar e configurar a plataforma SIEM **ArcSight** neste cliente. Este projecto é delineado segundo o conjunto de fases seguintes:

- **Análise de requisitos:** em conjunto com o cliente, reunindo as suas necessidades e expectactivas;

- **Desenho da solução:** arquitectura da plataforma para inclusão da mesma na infra-estrutura já existente;
- **Instalação física:** de todos os elementos da solução, de acordo com o desenho estipulado;
- **Configuração/Desenvolvimento dos agentes colectores:** instalação dos agentes e criação de novos agentes para análise de *logs* com formato proprietário ou desconhecido à plataforma;
- **Criação de conteúdo:** geração de elementos auxiliadores da monitorização, desde relatórios, gráficos de actividade, notificações, etc, dando atenção especial à criação de regras que permitem a correlação entre eventos;
- **Optimização da solução:** fazendo alguns ajustes adicionais para que a plataforma tenha o melhor desempenho possível;

1.3 Estrutura do documento

Os próximos capítulos estão organizados do seguinte modo: O segundo capítulo apresenta trabalho relacionado na área da segurança e monitorização de activos e a arquitectura geral de uma plataforma SIEM. O terceiro capítulo apresenta o desenho da arquitectura ArcSight e sintetiza a informação relativa ao desenvolvimento de agentes personalizados e criação de regras de correlação de eventos; O capítulo quatro apresenta o trabalho realizado no cliente para implementar a solução ArcSight na sua infra-estrutura; O capítulo cinco apresenta a avaliação efectuada à plataforma quanto ao seu desempenho e outras métricas relevantes. Finalmente, o sexto capítulo fecha o documento apresentando as conclusões do projecto e o trabalho futuro que ainda poderá ser desenvolvido.

Capítulo 2

Plataformas SIEM

Este capítulo apresenta as plataformas SIEM, a sua arquitectura e as principais funções associadas. Tem como objectivo introduzir estas soluções de uma forma genérica, sem se comprometer com uma plataforma específica. São também abordados alguns temas relacionados com as soluções SIEM, fazendo também uma breve alusão à origem destas plataformas.

2.1 Conceitos e Trabalho Relacionado

2.1.1 Defesa

Na área da segurança da informação existem três grandes focos de actividade a considerar [1]:

1. Defesa contra catástrofes;
2. Defesa contra faltas/falhas previsíveis;
3. Defesa contra actividades não autorizadas;

2.1.1.1 Defesa contra catástrofes

Esta defesa cobre situações em que um sistema seja afectado fisicamente, tais como catástrofes ambientais (terramotos, incêndios, etc), catástrofes políticas (ataques terroristas, motins, etc) ou catástrofes materiais (perda ou roubo de equipamentos como discos rígidos, portáteis, etc).

Para minimizar o impacto causado por todos estes potenciais cenários, esta defesa pode surgir sob a forma de redundância de *hardware* (discos montados em RAID, por exemplo) ou redundância de equipamentos (Duas ou mais máquinas que oferecem o mesmo serviço mas distantes geograficamente). Outra forma mais básica de defesa pode

ser a simples salvaguarda periódica de informação (*backups*), desde que estas sejam guardadas noutro ponto geográfico, para que a mesma catástrofe não afecte tanto o sistema como as próprias cópias de segurança.

É também desejável que máquinas que ofereçam um serviço crítico estejam localizadas em zonas anti-sísmicas, em abrigos anti-sísmicos ou em zonas elevadas para que se evitem inundações.

2.1.1.2 Defesa contra faltas/falhas previsíveis

Mecanismos de defesa contra estas faltas e falhas minimizam o impacto nos sistemas de situações que ocorrem mais frequentemente do que as anteriores. Exemplos destes cenários previsíveis englobam cortes de energia (que podem ser resolvidos se existirem geradores que oferecem energia eléctrica enquanto não é reposta a fonte principal), bloqueios de sistemas operativos ou na execução de aplicações sobre estes (falhas que podem ser resolvidas com máquinas em *cluster* e sistemas transaccionais que evitem a inconsistência de informação após o bloqueio), falhas de conectividade entre redes (problema que pode deixar de existir se existirem rotas alternativas na rede), etc.

2.1.1.3 Defesa contra actividades não autorizadas

Ao contrário das defesas anteriores, esta visa salvaguardar sistemas de actividades executadas deliberadamente para corromper ou subverter o sistema, podendo ser despoletadas tanto por pessoas na organização ou fora dela. Estas actividades podem ser, por exemplo: Acesso a informação, alteração de informação, utilização excessiva de recursos computacionais, impedimento de prestação de serviço (vulgo DoS ou *Denial of Service*) ou simplesmente vandalismo.

Esta é a vertente mais complexa da segurança, e é onde se podem ver plataformas SIEM a servir de mecanismo de defesa.

2.1.2 Mecanismos de segurança

Esta secção apresenta alguns dos principais mecanismos de segurança que, tipicamente, visam defender sistemas de actividades não autorizadas por parte de atacantes. Todos eles registam informação passível de ser enviada e analisada numa solução SIEM.

2.1.2.1 Mecanismos de controlo de acesso

Entidades que avaliam se um dado utilizador pode executar uma determinada acção sobre um determinado objecto. Estes mecanismos podem ser físicos (leitor de cartões à entrada de um *datacenter* para verificar se um utilizador pode aceder ao mesmo) ou configurações ao nível do sistema operativo (como é o caso das ACL's (Access Control List)).

2.1.2.2 Mecanismos de filtragem

De certa forma, estes mecanismos também executam um controlo de acesso. Estes identificam actividade ou interacções desnecessárias ou não autorizadas entre máquinas, evitando que essas mesmas interacções aconteçam. Um importante mecanismo de filtragem amplamente utilizado são as *firewalls*.

Firewalls

As *firewalls* são um mecanismo importante de defesa de uma infra-estrutura. O seu objectivo é controlar o tráfego de entrada e saída da rede, analisando os pacotes de forma a determinar se este deve ser permitido ou não, segundo um conjunto de regras definidas. Tipicamente protege uma rede interna que se assume como segura de outras redes externas que não se pode assumir como segura.

Existem dois grandes tipos de *firewalls*:

- *packet filter firewall*: Analisa dados como endereços e portas de fonte e destino dos pacotes. Com base nessa informação aceita ou descarta o pacote.
- *application firewall*: Mais granulares que as anteriores. Conseguem examinar o conteúdo de cada pacote e bloquear ou aceitar mediante esse conteúdo. Por exemplo, é possível bloquear certas mensagens HTTP (Hypertext Transfer Protocol) com cabeçalhos anómalos ou certas interacções FTP (File Transfer Protocol). Com as *firewalls* anteriores, era possível apenas bloquear todas as ligações HTTP e FTP. No entanto, são muito mais complexas que as anteriores, tanto a nível de desempenho como de administração.

2.1.2.3 Mecanismos de inspecção

São mecanismos que estão permanentemente a vigiar a rede, em busca de actividades inesperadas ou ilegais, complementando outros mecanismos de segurança que possam ter sido mal configurados. Por exemplo, uma ligação que passou numa *firewall* e não devia pode ser detectada num destes mecanismos. Como exemplos destes elementos, existem os IDS's (Intrusion Detection System).

Intrusion Detection System

O objectivo de um IDS é detectar e contrariar intrusões [1]. Estas intrusões podem ser qualquer acção no sistema que vise degradar a integridade, confidencialidade ou disponibilidade do serviço.

Esta detecção é feita a partir da recolha de informação de diversas actividades e/ou da análise comportamental dos sistemas, tentando depreender se se está perante uma in-

trusão. Dessa detecção podem surgir variadas acções de remediação, desde reforçar a segurança, contra-atacar, recuperação e correcção de falhas, etc.

A arquitectura destes IDS divide-se em quatro componentes (Figura 2.1):

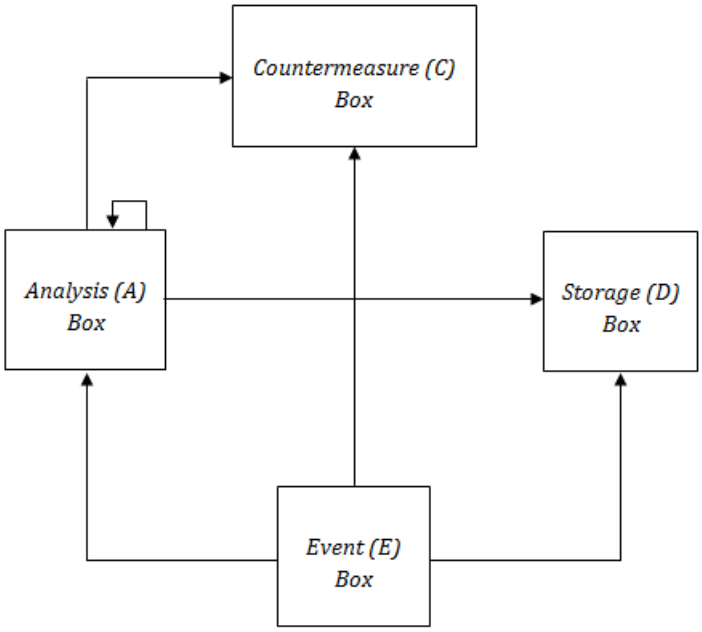


Figura 2.1: Arquitectura de um IDS

- *E boxes:* Sensores que capturam eventos de baixo nível (pacotes, sessões de utilizadores, alterações no sistema de ficheiros, etc);
- *A boxes:* Analisam os eventos recebidos pelas *E boxes*, podendo interpretar vários eventos de diversos sensores; Esta análise e correlação pode gerar eventos de mais alto nível, realimentando as próprias *A boxes*;
- *D boxes:* Armazenam todos os eventos de baixo e alto nível;
- *C boxes:* Elementos que accionam mecanismos de remediação automática;

Estes sistemas podem organizar-se em dois grupos:

- NIDS (Network-based Intrusion Detection System): São implementados em pontos estratégicos da rede de forma a monitorizar o tráfego entre máquinas.
- HIDS (Host-based Intrusion Detection System): São instalados agentes nas máquinas, monitorizando *hardware*, *kernel*, sistema de ficheiros, etc.

Os IDS podem ainda ser baseados em *assinaturas* se comparam a actividade que capturam com uma base de dados que contenha registos de comportamentos conhecidos

como maliciosos, à imagem do que é efectuado por *antivírus*. Por outro lado, podem basear-se em *anomalias*. Neste caso, estabelece-se um padrão comportamental vulgar para a rede que se está a controlar (largura de banda tipicamente utilizada, portas abertas, máquinas normalmente ligadas entre si, etc) e quando algo se desvia deste comportamento, este IDS deve agir em conformidade.

Podem também ser activos ou passivos, caso possuam mecanismos de remediação ou apenas produzam alertas, respectivamente, e podem detectar as intrusões em tempo real ou em diferido.

Não obstante a grande utilidade dos sistemas de detecção de intrusões, é importante denotar três limitações:

- **Adaptabilidade:** A adaptação a ambientes diversificados é complexa e tem, obrigatoriamente, de ser feita de modo a evitar falsos positivos e falsos negativos;
- **Escalabilidade:** Fraca escalabilidade quer a nível de HIDS, pois quanto mais diferentes máquinas existirem na rede, mais difícil será implementar o mesmo IDS e, consequentemente, o mesmo tipo de análise, mas também a nível de NIDS, caso seja uma rede de tráfego elevado;
- **Heterogeneidade:** Não existindo uma taxonomia genérica para caracterizar ataques e intrusões, a interoperabilidade entre IDS diferentes e outros mecanismos de segurança torna-se complexa. Esta normalização tornaria mais fácil a correlação de eventos, sendo este um tema ainda em investigação.

2.1.2.4 Políticas de Segurança

Estas políticas definem o foco da segurança e o que esta deve garantir. Estas políticas são postas em prática, tipicamente, por mecanismos de segurança como os vistos atrás. A definição destas políticas não é simples e requer um enorme cuidado. No entanto, existem padrões que definem um conjunto de boas práticas e que, inclusivé, algumas organizações estão obrigadas a cumprir para conseguirem colocar em prática uma política de segurança adequada. Alguns exemplos destes padrões são o PCI DSS (Payment Card Industry Data Security Standard), SOX (Sarbanes-Oxley), ISO/IEC 27002:2005 (International Organization for Standardization / International Electrotechnical Commission), HIPAA (Health Insurance Portability and Accountability Act), Visa, MasterCard, etc.

2.1.3 Ataques

Nos dias de hoje são cada vez mais os ataques que ocorrem e são cada vez mais sofisticados. Seguem-se alguns dos principais exemplos com que as organizações lidam hoje em dia:

2.1.3.1 Acesso a informação reservada ou confidencial

Acessos não autorizados a ficheiros, bases de dados ou outros objectos podem constituir um grave problema para qualquer organização, sobretudo se essa informação for de cariz confidencial ou pessoal (como dados pessoais, dados de cartões de crédito, *passwords*, etc. Estes ataques podem acontecer derivado de controlos de acesso mal configurados, ou até mesmo a falta deles.

2.1.3.2 Personificação

Estes ataques acontecem quando um adversário consegue subverter os mecanismos de autenticação e faz-se passar por outro utilizador. Os objectivos deste ataque podem passar por usar um sistema em modo privilegiado (se se personificar um administrador, por exemplo), ou simplesmente por efectuar operações em nome de outro utilizador, mascarando por completo a identidade do atacante.

2.1.3.3 Incapacidade de prestação de serviço

Uma ou várias máquinas podem ser incapacitadas de oferecer o seu serviço se forem alvo de ataques de DoS. Estes ataques podem ter inúmeras variantes: *ping flood*, *SYN flood*, *teardrop attack* (envio de pacotes IP com tamanhos maiores do que o suposto) ou, simplesmente, enviar inúmeros pedidos num curto espaço de tempo a uma aplicação que está a ser executada na máquina a atacar.

Quanto maior for a relevância da máquina e do sistema para os clientes que o utilizam, maior será o impacto causado por este ataque.

2.1.3.4 Intercepção, modificação ou reprodução de fluxo de dados

Um ataque de interposição (ou *man-in-the-middle*) sucede quando dois nós comunicantes na rede supõem que estão a comunicar de forma segura entre si, mas na realidade existe um atacante entre eles a personificar ambas as pontas comunicantes. Enquanto o atacante reencaminha os pacotes entre os dois nós para que estes não detectem problemas de comunicação, este pode visualizar o conteúdo dos pacotes trocados, ou até mesmo alterar o seu conteúdo sem que seja detectado.

Esta modificação do fluxo de dados pode ser feita de forma a favorecer o atacante, pois pode levar o sistema a actuar sob o controlo do mesmo, ou simplesmente descontrolá-lo e levando-o a uma situação de negação de serviço.

A reprodução excessiva do fluxo de dados pode também levar à negação da prestação do serviço, pois pode fazer com que o sistema entre em sobrecarga. Dependendo do serviço, a repetição de mensagens pode trazer benefícios ao adversário. Tipicamente, estes ataques são conhecidos como *replay attacks*.

2.2 Origem das plataformas SIEM

O termo "SIEM"é relativamente recente. O interesse por este tema começou a ser definido em 2005, por Mark Nicolett e Amrit Williams, da Gartner[1]. Estes autores descreveram as plataformas SIEM como sistemas que recolhem, analisam e apresentam a informação dos dispositivos de segurança tais como *Firewalls*, IDS (Intrusion Detection System), IPS (Intrusion Prevention System), entre outros, bem como outras informações de segurança de outros activos (como bases de dados, sistemas operativos, etc). Partindo desta análise de eventos, estas plataformas providenciam relatórios de actividade, notificações para pessoas internas à organização de forma a mitigarem um suposto ataque, etc.

A tecnologia SIEM nasce da fusão de duas outras já existentes:

- Plataformas SIM (Security Information Management) - Centralizam *logs* por largos períodos temporais, para posterior análise forense.
- Plataformas SEM (Security Event Management) - Efectuam a monitorização de dados em tempo real, correlação de eventos, notificações, etc.

A Tabela 2.1 sumariza as principais funções de uma plataforma SIEM.

[1] `http://techbuddha.wordpress.com/2007/01/01/the-future-of-siem-%E2%80%93-the-market-will-begin-to-diverge/`

Função	Descrição	Herdado de
Recolha de eventos	Recolha dos eventos gerados por activos ou aplicações	SEM/SIM
Correlação de eventos	Encontrar relações entre eventos para deduzir actividades suspeitas na rede	SEM
Retenção de eventos	Eventos são guardados em bases de dados durante um largo período de tempo, quer para efeitos de correlação de eventos cujo intervalo de tempo possa ser mais alargado, ou de forma a preencher requisitos de normas de segurança	SIM
Geração automática de alertas	Notificações enviadas (por exemplo, mail ou SMS) para pessoal responsável por verificar se algo anormal está a suceder	SEM
Criação de relatórios e outros conteúdos para análise de eventos	A partir dos eventos registados, produzir conteúdo informacional para auxiliar a procura de padrões de actividade anómala, ou identificar outro tipo de actividades que possam infringir políticas de segurança de uma organização	SIM

Tabela 2.1: Principais funções de uma ferramenta SIEM

Qualquer plataforma SIEM tem de ser capaz de oferecer todas as funções referidas. Isto é conseguido através de uma arquitectura seguida, tipicamente, por qualquer solução. Esta mesma arquitectura é detalhada na secção seguinte.

2.3 Arquitectura dos SIEM

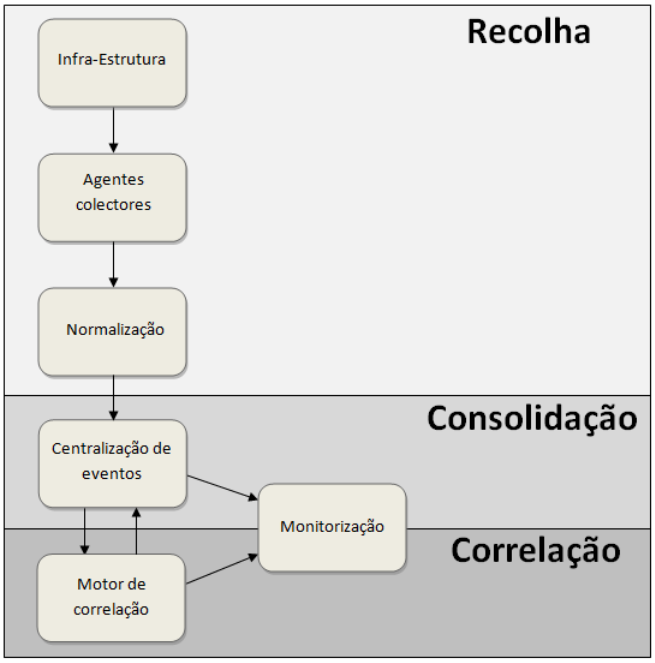


Figura 2.2: Arquitectura SIEM

Apesar de cada solução SIEM poder ter uma arquitectura própria com factores que a diferenciem de outras, existem três pontos que são transversais a todas as plataformas existentes. Estas devem concretizar mecanismos para **recolher**, **consolidar** e **correlacionar** eventos. A Figura 2.2 representa a arquitectura destas plataformas e é detalhada de seguida.

2.3.1 Recolha

Esta camada é responsável por recolher os *logs* dos activos a monitorizar. Tipicamente, esta tarefa é executada por agentes colectores que recolhem e enviam os registos para a camada de consolidação.

2.3.1.1 Infra-estrutura

Do ponto de vista do conceito de uma solução SIEM, os dispositivos que compõem a infra-estrutura de uma organização não são parte integrante da plataforma de monitorização. No entanto, merecem o devido lugar na arquitectura porque sem eles, não existem eventos. São eles que alimentam o sistema para posterior correlação e análise de actividades.

Algumas questões essenciais na definição dos activos a monitorizar são[2]:

- De quais dispositivos deverão ser recolhidos eventos?: Olhando para a realidade da infra-estrutura em questão, há que tirar ilações acerca do índice de risco que cada elemento da rede tem associado. Teoricamente, este risco é calculado pela fórmula:

  Risco = Vulnerabilidade x Ameaça[3]

  Dever-se-á definir o risco associado a cada activo da infra-estrutura (que terá um grau de exposição e ameaça associado) e quanto mais elevado for esse risco, maior importância terá a recolha dos seus eventos.

- Que eventos se devem recolher ou ignorar?: Alguns dispositivos geram uma enorme quantidade de *logs*. Por forma a não sobrecarregar a plataforma, deverá encontrar-se um equilíbrio entre os eventos que podem ser ignorados e os que são realmente relevantes.

- Por quanto tempo guardar os eventos recolhidos?: Pouco tempo de retenção de eventos oferece uma janela temporal curta para análise de incidentes. Porém, quanto mais tempo de retenção for desejado, mais dispendiosa se tornará a plataforma.

2.3.1.2 Agentes colectores

O processo de recolha de registos baseia-se num agente que recolhe *logs* de um ou mais activos, sendo configurado para enviá-los para um servidor central, alimentando assim o sistema SIEM. Este transporte pode ser feito de duas formas:

- Método *push*: Eventos são enviados para o agente.

- Método *pull*: Agente inicia ligações com activos para recolher os eventos.

2.3.1.3 Normalização

Inerente a esta recolha de eventos está uma sub-tarefa de **normalização** que deverá ser executada sobre cada *log* recolhido. Esta normalização é imprescindível na medida em que, por entre todos os dispositivos, sistemas e aplicações que podem ser auditados, existe uma grande disparidade quanto ao formato em que os seus registos de actividade são escritos. O objectivo desta tarefa é homogeneizar os diversos formatos para um formato conhecido pela plataforma de maneira a facilitar a correlação de eventos de diferentes activos. Esta normalização tanto pode ser feita quer nos agentes colectores, quer no repositório que centraliza os eventos, normalizando-os antes de os guardar.

2.3.2 Consolidação

A centralização de eventos registados é vital, não só para ter um registo actualizado do presente, mas também para manter um histórico do passado, caso seja necessário efectuar uma investigação sobre incidentes anteriores.

Dada a importância dos eventos armazenados, esta informação e o seu armazenamento devem serem alvos de auditoria de segurança. Como tal, é aconselhável aplicar sobre estes registos as típicas técnicas que salvaguardam os dados contra as mais diversas catástrofes, como replicação da base de dados, cópias de segurança actualizadas, etc. Sendo informação sensível, devem existir mecanismos que garantam a confidencialidade e a integridade da mesma.

Tipicamente, esta concentração de eventos é feita com recurso a uma base de dados, que pode ser baseada em tecnologias bem conhecidas (MySQL, Oracle, etc) ou até mesmo uma base de dados proprietária.

2.3.3 Correlação

O objectivo do motor de correlação é tentar definir uma relação coerente entre um conjunto de eventos registados, apresentando aos utilizadores da plataforma apenas e só um evento, denominado "evento de correlação". Para isso, são definidas várias regras de correlação de forma a detectar estes padrões comportamentais. Como exemplo tem-se a Tabela 2.2, que sumariza um conjunto de 10 eventos em 10 segundos de tentativas de *login* que se registaram a dada altura numa solução SIEM:

Timestamp	**ID**	**Fonte**	**Destino**	**Evento**
10:10:01 CST	1035	192.168.1.200	10.10.10.25	Login no servidor falhou
10:10:02 CST	1036	192.168.1.90	10.10.10.21	Login no servidor efectuado
10:10:03 CST	1037	192.168.1.200	10.10.10.25	Login no servidor falhou
10:10:04 CST	1038	192.168.1.91	10.10.10.35	Login no servidor falhou
10:10:05 CST	1039	192.168.1.10	10.10.10.2	Login no servidor efectuado
10:10:06 CST	1040	192.168.1.10	10.10.10.3	Login no servidor efectuado
10:10:07 CST	1041	192.168.1.200	10.10.10.25	Login no servidor falhou
10:10:08 CST	1042	192.168.1.201	10.10.10.54	Login no servidor falhou
10:10:09 CST	1043	192.168.1.10	10.10.10.34	Login no servidor falhou
10:10:10 CST	1044	192.168.1.200	10.10.10.25	Login no servidor efectuado

Tabela 2.2: Exemplo de eventos de tentativas de autenticação registados num SIEM

Podendo passar despercebido, existe um sub-conjunto destes eventos de *login* que poderão denotar um ataque de força bruta, visando aceder a um dado servidor. Visualizando

esta tabela com mais atenção, focando os eventos 1035, 1037, 1041 e 1044, constata-se um certo padrão que nos remete para um ataque de força bruta: Um número de tentativas de acesso falhadas (3, no caso) e finalmente um *login* bem sucedido.

Para exemplo em específico, uma possível regra de correlação deveria identificar 3 ou mais tentativas de *login* sem sucesso do mesmo endereço IP de origem, para o mesmo IP de destino e, após esses 3 eventos, um quarto evento que denote um acesso conseguido. Tudo isto numa janela temporal de, suponha-se, 10 segundos. Esquematicamente, esta regra está reflectida na Figura 2.3.

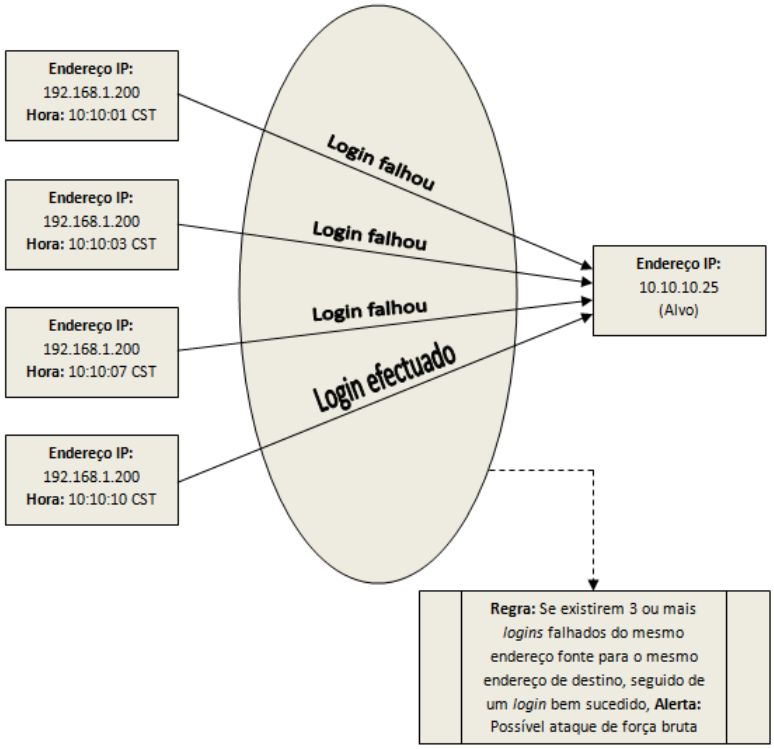


Figura 2.3: Exemplo de regra de correlação - Ataques de força bruta

Finalmente, a fase de **monitorização** tem o objectivo de interagir activamente com os eventos recolhidos e guardados. Para se fazer algo de útil com a informação retida, as soluções SIEM disponibilizam aos seus utilizadores uma consola como interface. Nela poderão criar-se as regras de correlação, efectuar consultas à Base de Dados de eventos, criar conteúdos (tais como relatórios de actividade, *dashboards*, notificações, etc). Tipicamente, esta monitorização é feita por uma ou mais equipas especializadas para o efeito. Seguem-se breves descrições de duas dessas equipas.

CSOC (Cyber Security Operation Center)

O CSOC é um serviço potencialmente utilizador das plataformas SIEM. Analisando os eventos recolhidos 24 horas por dia e 365 dias por ano, a função deste centro passa por assegurar a fiável e rápida detecção de comportamentos anómalos por forma a mitigar esse possível ataque. Para isso, terá ao seu dispor relatórios de actividade, *dashboards* em tempo real, canais de monitorização dos eventos também em tempo real, e é para ele, geralmente, que são enviadas notificações e alertas que eventuais regras poderão ter gerado.

CSIRT (Computer Security Incident Response Team)

Estas equipas, funcionando como um ponto central de coordenação, estão em constante comunicação com o CSOC e respondem aos incidentes de segurança por ele identificados. Tipicamente esta equipa é constituída por um conjunto de pessoas conhecedora da infra-estrutura da organização, suas prováveis fragilidades e especificidade. Torna-se assim a equipa certa para responder rapidamente a cada incidente, pois saberão instantaneamente em que ponto da rede terão de focar o seu esforço.

Na inexistência de um CSOC, o CSIRT poderá também efectuar as suas funções, fundindo-se os dois serviços num só.

2.4 Conclusão do capítulo

Este capítulo introduziu temas que são transversais a qualquer solução SIEM, tais como alguns conceitos básicos e a arquitectura genérica destas plataformas. No capítulo seguinte apresenta-se a plataforma SIEM em que se baseará o projecto.

Capítulo 3

Plataforma ArcSight

Este capítulo tem como objectivo apresentar a solução SIEM da ArcSight, base de todo o trabalho realizado, e os dois pontos de extensão de grande relevância da plataforma SIEM. Para além da arquitectura ArcSight, aborda-se a temática do desenvolvimento de agentes colectores para *logs* cujo formato não é conhecido pela plataforma nativamente. Aprofunda-se, também, o tema de criação de regras de correlação nesta solução, dado que estes são os recursos mais importantes num SIEM.

3.1 Solução SIEM da ArcSight

A ArcSight disponibiliza uma solução que contempla uma vasta gama de produtos que, funcionando como um todo, formam uma solução bastante flexível. Este nível de flexibilidade permite desenhar plataformas SIEM com mais ou menos componentes, tendo em conta os requisitos da organização em questão. O facto de estes componentes não serem dependentes entre si permite também construir sistemas SIEM de acordo com as necessidades requeridas.

3.1.1 Arquitectura

A Figura 3.1 resume a arquitectura da plataforma ArcSight[4] (lado esquerdo), efectuando a respectiva relação com a arquitectura genérica de um SIEM (lado direito), vista atrás na Secção 2.3:

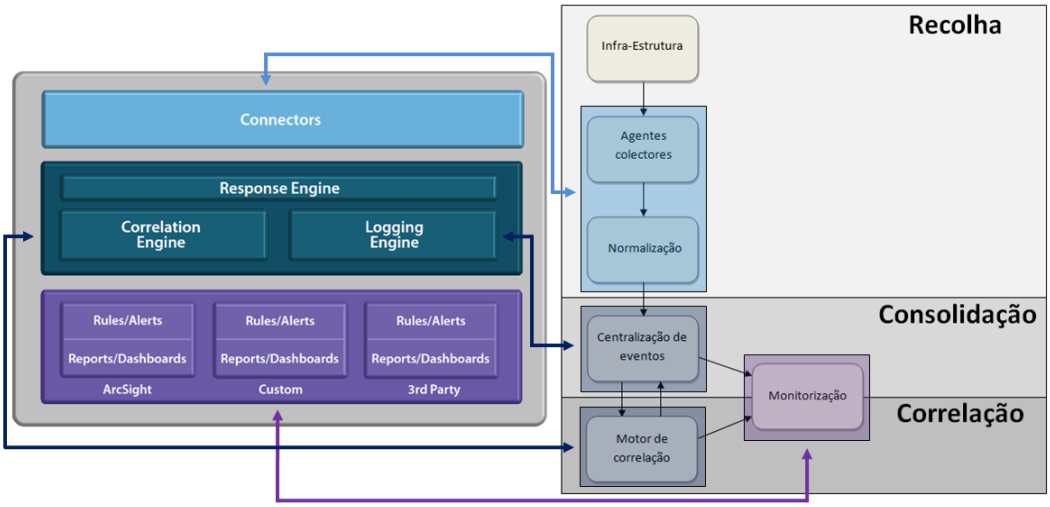


Figura 3.1: Arquitectura da solução SIEM da ArcSight

Da sua análise, depreende-se que:

- **Connectors**: Também chamados de `SmartConnectors`. São os agentes que recolhem e normalizam os eventos. Existem também os **`FlexConnectors`** que executam as mesmas acções, mas são concebidos à medida para um formato de *log* desconhecido; Estes agentes podem ser instalados localmente nas máquinas a monitorizar, remotamente numa qualquer outra máquina (dedicada somente à instalação de `connectors`, por exemplo), sendo que a ArcSight disponibiliza, opcionalmente, um módulo chamado `Connector Appliance` para instalação de `connectors` ou simplesmente para gestão dos mesmos (quer estejam instalados dentro ou fora dela);

- **Correlation Engine**: Módulo que efectua a correlação de eventos. A ArcSight apelida este elemento de `Manager`;

- **Logging Engine**: Responsável pela centralização de eventos. Embora seja possível utilizar o `Manager` para este fim, é bastante comum utilizar um componente denominado `Logger` em paralelo com o `Manager` que, por tipicamente ter mais espaço para armazenamento, consegue centralizar os eventos durante mais tempo do que um `Manager`;

- **Response Engine**: Módulo opcional para resposta automática a incidentes, denominado `NCM/TRM` (Network Configuration Manager/Threat Response Manager);

- **Rules/Alerts/Reports/Dashboards**: Recursos utilizados para efectuar a monitorização da infra-estrutura. Podem ser conteúdos que estão por omissão na plataforma (ArcSight), mas também podem ser criados à imagem da realidade da organização (Custom) e ainda oriundos de pacotes adicionais (3rd Party) para responder a certos

requisitos e normas regulatórias. Estes podem ser criados tanto no `Manager` como no `Logger`, embora a complexidade dos mesmos não seja equivalente em ambos.

A Figura 3.2 ilustra uma configuração típica de uma solução SIEM da ArcSight, englobando os principais componentes descritos atrás. Nela, é possível observar vários dispositivos e aplicações a enviarem os seus *logs* para `connectors` que os processam e reencaminham, paralelamente, para um `Manager` e para um `Logger`. Adicionalmente, existe uma `Connector Appliance` para gerir os `connectors` remotamente.

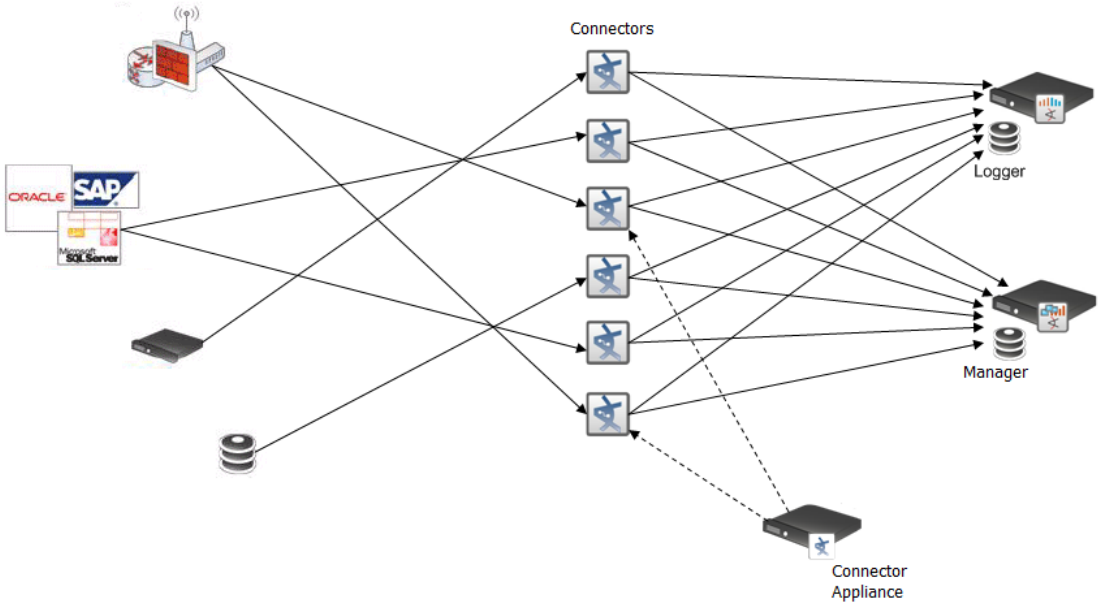


Figura 3.2: Configuração típica de uma solução SIEM da ArcSight

3.2 Desenvolvimento de FlexConnectors

Para os casos em que a ArcSight não desenvolveu um `SmartConnector` que processe *logs* de um determinado produto, é disponibilizado um SDK (Software Development Kit) para concretizar um `connector` adaptado à necessidade em questão. Estes `connectors` especiais, denominados de `FlexConnectors` [5], trabalham exactamente da mesma forma que os `SmartConnectors`, normalizando e reencaminhando os eventos até a um `Manager` ou `Logger`, diferenciando-se apenas no facto de ser preciso escrever previamente um ou mais ficheiros de configuração que indiquem como deve ser feito o processamento dos eventos. Esses ficheiros de configuração terão de ser criados segundo metodologias específicas, dependendo do tipo de `FlexConnector` a desenvolver. Adicionalmente, existe uma série de funções que se podem aplicar sobre os registos para enriquecer ainda mais o processamento.

A maneira como é desenvolvido o `connector` depende da fonte de onde este recolherá os *logs*. Os vários tipos de FlexConnectors existentes são:

- **Log File `FlexConnector`:** Para ler ficheiros de *log* com um formato fixo e bem delimitado;
- **Regular Expression `FlexConnector`:** Para ficheiros com formato variável;
- **Database `FlexConnector`:** Para consultar bases de dados onde estão alojados os registos;
- **Syslog `FlexConnector`:** Para receber *logs* via *Syslog*. Também muito usado;
- **SNMP `FlexConnector`:** Para receber *logs* via SNMP;
- **XML `FlexConnector`:** Para ler ficheiros XML;

3.2.1 Metodologia

Embora a criação de ficheiros de configuração seja transversal a qualquer tipo de `FlexConnector`, a forma de o fazer difere entre eles em certos aspectos. Genericamente, qualquer `FlexConnector` deve:

1. Isolar os atributos do registo de actividade;
2. Mapear os atributos para os campos do formato comum conhecido pela plataforma;
3. Associar ao evento um índice de severidade (importância) do evento;

3.2.1.1 Isolamento de atributos

Este isolamento passa por especificar os fragmentos de um *log* que serão processados. A cada um desses pedaços dá-se o nome de *token*. Cada *token* tem um nome, um tipo e por vezes um formato (como as TimeStamps, por exemplo). Os tipos disponíveis são:

- Integer
- Date
- IPAddress
- IPv6Address
- Long
- MacAddress
- RegexToken
- String

- Time
- TimeStamp

O exemplo seguinte mostra como separar um *log* em 5 fragmentos: Uma *timestamp*, um endereço IP, e três outras *strings*. A definição de cada *token* engloba a sua nomeação (parâmetro *name*), definição de tipo (parâmetro *type*) e, dependendo do tipo, o seu formato (parâmetro *format*).

```
token.count=5

token[0].name=Time_of_the_event
token[0].type=TimeStamp
token[0].format=yyyy-MM-dd HH:mm:ss
token[1].name=ClientIp
token[1].type=IPAddress
token[2].name=Method
token[2].type=String
token[3].name=URL
token[3].type=String
token[4].name=Status
token[4].type=String
```

Desta forma os *tokens* ficam guardados em variáveis diferentes. A definição de como serão repartidos os *logs* é feita mais adiante, sendo que isto é feito de maneira diferente dependendo do tipo de agente a desenvolver.

3.2.1.2 Mapeamento de atributos

Depois de se terem os valores necessários correctamente extraídos do *log* é altura de definir como será feita a normalização desse registo para dar origem a um evento. Isto é feito atribuindo cada *token* a um campo da estrutura de evento conhecida pela plataforma ArcSight.

```
event.deviceReceiptTime=Time_of_the_event
event.sourceAddress=ClientIp
event.deviceSeverity=Status
event.requestUrl=URL
event.requestMethod=Method
```

3.2.1.3 Atribuição de severidade

Opcionalmente é possível definir um mapeamento da severidade de cada evento. Esta distinção é visualizada nas consolas de monitorização, pelo que uma boa atribuição de graus de importância permite distinguir rapidamente eventos vulgares de eventos suspeitos. Esta severidade apresenta-se numa escala de cinco valores: Very High, High, Medium, Low e

Very Low. Um evento sem severidade associada ficará com o grau de Very Low, sendo que um evento grave em termos de segurança terá uma importância "Very High"e os menos importantes serão "Low".

```
severity.map.veryhigh.if.deviceSeverity=404,500
severity.map.medium.if.deviceSeverity=303,302
severity.map.low.if.deviceSeverity=200..204
```

3.2.2 Tipos de FlexConnectors

As três secções anteriores mostram o trabalho a realizar para desenvolver qualquer `FlexConnector`. Resta explicitar a forma como é feita a separação de um *log* em *tokens* para serem recolhidos na fase de isolamento e mapeados para o formato conhecido da plataforma. Isto varia entre os tipos de agentes disponíveis e serão abordadas todas as variantes nas secções seguintes.

3.2.2.1 Log File FlexConnector

Este é o `FlexConnector` mais simples de implementar, embora só possa ser aplicado em ficheiros de *logs* com um formato em que os atributos estão bem delimitados. Seja o caso em que o *log* a processar fosse:

2012-06-16 16:52:23|89.125.23.25|GET|http://www.di.fc.ul.pt|200

Para criar um agente deste tipo bastaria adicionar uma única linha à configuração definida anteriormente:

```
delimiter=|
```

3.2.2.2 Regular Expression FlexConnector / Syslog FlexConnector

Este é o tipo de `FlexConnector` mais comum. Utilizado em cenários em que os *logs* não seguem uma estrutura estática bem delimitada (como no caso anterior), este `FlexConnector` analisa cada linha de *log* ou mensagem *syslog* com o auxílio de uma expressão regular escrita para o formato em questão. Imagine-se, agora, que o *log* a recolher seria o seguinte:

2012-06-16 16:52:03 From:89.125.23.25 Access:GET http://www.di.fc.ul.pt 200

A concepção do `FlexConnector` ficaria completa, adicionando a seguinte propriedade:

```
regex=
(\\d{4}-\\d{2}-\\d{2}\\s\\d{2}:\\d{2}:\\d{2})\\sFrom:(\\d{1,3}\.\\d
    {1,3}\.\\d{1,3}\.\\d{1,3})\\sAccess:([A-Z]+)\\s(\\S+)\\s(\\d{3})
```

Note-se que o isolamento dos *tokens* é feito pelos grupos delimitados pelos parentesis na expressão regular.

3.2.2.3 Database FlexConnector

Dependendo da arquitectura desenhada para a base de dados que recebe os eventos, existem duas formas de criar este `FlexConnector`:

a) A base de dados é *time-based*:

Cada linha representa um evento e a chave primária da tabela de eventos é a *timestamp* do *log*.

É necessário adicionar mais quatro parâmetros adicionais. Seja o seguinte exemplo:

EventID	Time	Origin	Method	AccessedURL	Status
Access-75935	2012-06-16 16:52:03	89.125.23.25	GET	http://www.di.fc.ul.pt	200

Os quatro parâmetros adicionais são:

a 1) Version

```
version.order=1
version.query=SELECT EventID from Access
version.id=12
```

Caso se deseje criar um `FlexConnector` que leia eventos de várias bases de dados de versões diferentes, será necessário preencher estes parâmetros para que o agente se ajuste às várias versões com que possa lidar. Caso hajam vários ficheiros de configuração diferentes, deverá existir um por versão.

- **version.order:** Especifica a ordem pela qual são verificadas as versões da base de dados. Por exemplo, caso existam dois ficheiros de configuração diferentes, a ordem pela qual são utilizados é definida por este parâmetro. Caso a *query* existente no ficheiro com *version.order=1* não produza resultados ou erre inesperadamente, o ficheiro com *version.order=2* é utilizado, e assim sucessivamente.

- **version.query:** Determina o teste a efectuar à base de dados para validar a sua versão. Deve escolher-se uma *query* que não produza o mesmo resultado para duas versões diferentes.

- **version.id:** Caso a *query* anterior produza o resultado esperado, este valor será mapeado para o campo *deviceVersion* (do formato normalizado).

a 2) Query

Query SQL que retornará os eventos da base de dados. Um exemplo pode ser:

```
SELECT Time_of_the_event, ClientIP, Method, URL, Status
FROM Access
WHERE Time_of_the_event >= ?
ORDER BY Time_of_the_event
```

a 3) Timestamp

Para definir o campo que contém a *timestamp*:

```
timestamp.field=Time_of_the_event
```

a 4) UniqueID

Campo(s) a utilizar para distinguir eventos que possam ter *timestamps* iguais:

```
uniqueid.fields=EventID
```

b) A base de dados é *ID-based*:

Cada linha representa um evento e existe um identificador único para cada evento.

Para além de ser necessário definir os parâmetros *version*, *query* e *uniqueid* tal como em agentes para bases de dados *time-based*, são precisos outros dois valores:

b 1) ID

Para definir o campo que contém o identificador:

```
id.field=EventId
```

b 2) MaxID

Para retornar o maior identificador presente na base de dados quando a *query* é executada. Desta forma, essa *query* retornará apenas eventos entre o último ID e este máximo.

```
max.id=SELECT max(EventID) from Access
```

3.2.2.4 SNMP FlexConnector

Não há propriedades a adicionar, bastando apenas definir as propriedades comuns de isolamento, normalização e atribuição de severidade. Apenas é necessário garantir que, quando se definem os *tokens*, estes são descritos na mesma ordem que as variáveis (*var-binds*) constam na *trap*.

3.2.2.5 XML FlexConnector

Caso os *logs* sejam escritos em XML, existe um tipo de `FlexConnector` que se pode desenvolver para ler recursivamente esse formato, com recurso a *queries XPath/XQuery*. Apresente-se o exemplo a seguir:

```
<?xml version="1.0"encoding="utf-8"?>
<apache>
    <accesses>
        <access>
            <datetime>2012-06-16 16:52:23</datetime>
            <srcip>89.125.23.25</srcip>
            <method>GET</method>
            <url>http://www.di.fc.ul.pt</url>
            <status>200</status>
        </access>
    </accesses>
</apache>
```

Para desenvolver um agente que processe *logs* neste formato é necessário instruir o `connector` com a informação do *trigger node*, isto é, o nó que quando é encontrado faz despoletar o processamento de eventos. Neste caso elementar basta adicionar:

```
trigger.node.expression=/apache/accesses/access
```

Para além disto, o isolamento de atributos (*tokenization*) é feito de forma ligeiramente diferente, na medida em que é mandatório que se defina a *query XPath* para retirar os elementos do evento:

```
token.count=5

token[0].name=Time_of_the_event
token[0].type=TimeStamp
token[0].format=yyyy-MM-dd HH:mm:ss
token[0].expression=/apache/accesses/access/datetime
token[1].name=ClientIp
token[1].type=IPAddress
token[1].expression=/apache/accesses/access/srcip
token[2].name=Method
token[2].type=String
token[2].expression=/apache/accesses/access/method
token[3].name=URL
token[3].type=String
token[3].expression=/apache/accesses/access/url
token[4].name=Status
token[4].type=String
token[4].expression=apache/accesses/access/status
```

3.2.3 Operações sobre *tokens*

No momento em que se estão a mapear os atributos na fase de normalização, podem-se executar algumas operações sobre esses *tokens* de forma a enriquecer ou melhorar este mapeamento. Sendo uma espécie de API (Application Programming Interface), estas dezenas de funções permitem trabalhar sobre os *tokens* antes de os mapear para o campo desejado quando for necessário tratar a informação antes desse mapeamento. Seguem-se apenas alguns exemplos das operações mais utilizadas:

- __concatenate(): Concatena todas as *strings* (*tokens* ou literais) passadas como parâmetros;
- __createSafeLocalTimeStamp(date,format): Converte a *string date*, com o formato *format* para TimeStamp;
- __ifGreaterOrEqual(p1,p2,p3,p4): *p1* e *p2* são comparados numericamente. Se $p1 \geq p2$, retorna *p3*. Caso contrário, retorna *p4*;
- __ifThenElse(p1,p2,p3,p4): *p1* e *p2* são comparados. Se *p1=p2*, retorna *p3*. Caso contrário, retorna *p4*;
- __regexToken(string,regex): A expressão *regex* (contendo pelo menos um grupo entre parentesis) é testada sobre *string*. O primeiro grupo que se encontre dentro de *string* é retornado (Exemplo: event.name=__regexToken("foobar","foo(.*)"). event.name ficará com o valor "bar");
- __regexTokenAsAddress(string,token): O mesmo que a anterior, porém o resultado é convertido para um endereço IP;

- __replaceAll(string,regex,rep): Substitui por *rep* todas as ocorrências que coincidam com a expressão *regex* em *string*;
- __replaceFirst(string,regex,exp): O mesmo que a anterior mas só substitui a primeira ocorrência;
- __setYearToCurrentYear(timestamp): Para *timestamps* que não possuam o ano definido, é forçado o uso do ano corrente.
- __simpleMap(maplookup,"key1=value1","key2=value2",...): A *string maplookup* é pesquisada nos restantes parâmetros na forma de pares chave-valor. Caso se encontre *maplookup* no conjunto de chaves, é retornado o respectivo valor.
- __stringConstant(string): Devolve somente o parâmetro *string*

Mapeamentos avançados

Ainda sobre as operações que se podem realizar sobre *tokens*, importa destacar o seguinte sobre a operação *__simpleMap*: Dado que esta apenas permite que seja avaliado um parâmetro (*maplookup*), como fazer mapeamentos mais complexos que permitam a avaliação de vários valores?

Para alcançar este objectivo, é possível definir um ou mais ficheiros de configuração para o efeito. Estes ficheiros, em formato CSV (Comma Separated Values), definem:

- Um conjunto de campos e seus valores possíveis;
- Um conjunto de outros campos e respectivos valores para mapear nos mesmos, caso um dado evento contenha todos os campos atrás com os valores definidos;

Por exemplo, seja o seguinte exemplo em que se um evento vier da máquina *HostA* e o nome do evento for *Port Scan*, deseja observar-se no campo *message* do evento: *Port Scan Detected!*. Se vier do *HostB* e o seu nome for *Brute Force*, o campo *message* deverá conter *Brute Force Attack Detected!*.

```
event.sourceHostName,event.name,set.event.message
HostA,Port Scan,Port Scan Detected!
HostB,Brute Force,Brute Force Attack Detected!
```

3.3 Regras de correlação

Uma regra de correlação é um procedimento que avalia os eventos que chegam à plataforma segundo condições e padrões específicos que, quando um ou mais eventos coincidem com um desses padrões, pode despoletar acções imediatas. Na solução ArcSight, estas regras podem ser de dois tipos: simples ou de conjunção (*join rules*).

3.3.1 Regras simples

Estas regras são accionadas quando eventos coincidem com um conjunto de condições, como por exemplo, eventos que sejam direccionados a uma máquina crítica e que representem acções suspeitas. Se a regra for configurada para agregar múltiplos eventos com determinados campos em comum, a regra só será despoletada após um número de eventos definido e um intervalo de tempo definido. Um exemplo pode ser uma regra que é disparada quando três eventos de uma mesma fonte para um mesmo destino ocorrem num certo intervalo temporal.

3.3.2 Regras de conjunção

Conjugar regras significa que é possível interligar eventos de tipos diferentes, ao contrário das regras simples. São accionadas por eventos que satisfaçam dois ou mais conjuntos de condições. Por exemplo, uma regra de conjunção pode disparar caso surga um evento de intrusão por parte do IDS e uma ligação aceite por parte de uma *firewall*, caso o IP e porto de origem e destino coincidam em ambos os eventos. Também é possível agregar eventos para aguardar que um dado número de eventos ocorra dentro de uma janela temporal definida.

3.3.3 Avaliação de regras

Na plataforma ArcSight, o motor de correlação avalia os eventos que surgem segundo as regras definidas, mantendo em memória (*working memory*) os que satisfaçam as respectivas condições. Estes eventos são passados para um segundo módulo de avaliação (*tracker*) que avalia os eventos posteriores de acordo com definições de agregação ou condições de conjunção. Caso um dado número de eventos satisfaça todas as condições, o motor gerará um evento de correlação que representa todos os eventos que o originaram. Eventuais eventos guardados em memória que não tenham sido suficientes para gerar um evento de correlação são descartados (*garbage collector*). Este processo é descrito na Figura 3.3[6].

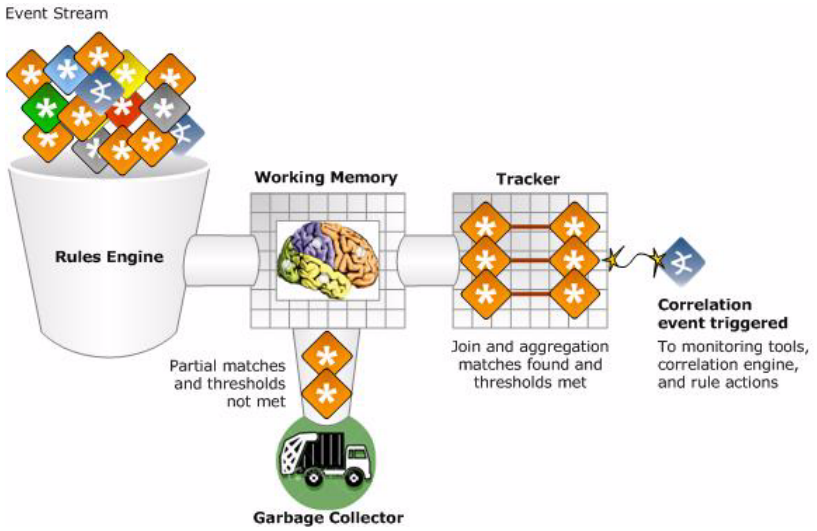


Figura 3.3: Processo de correlação de eventos

3.3.4 Acções a despoletar

Para que algo seja efectuado aquando do disparo da regra, é necessário definir em que condições esta será accionada. Dependendo do objectivo da regra, é possível definir uma ou mais acções para cada um dos seguintes *triggers*:

- ***On first event:*** A accionar na primeira vez que as condições são satisfeitas, independentemente da janela temporal para agregação definida;
- ***On subsequent events:*** A accionar na segunda e subsequentes vezes que as condições são satisfeitas, independentemente da janela temporal para agregação definida;
- ***On every event:*** A accionar todas as vezes que as condições são satisfeitas, independentemente da janela temporal para agregação definida;
- ***On first threshold:*** A despoletar na primeira vez que as condições são satisfeitas e o tempo de espera para agregação termina;
- ***On subsequent thresholds:*** A despoletar na segunda e subsequentes vezes que as condições são satisfeitas e o tempo de espera para agregação termina;
- ***On every threshold:*** A despoletar todas as vezes que as condições são satisfeitas e o tempo de espera para agregação termina;
- ***On time unit:*** A disparar caso as condições sejam satisfeitas e o tempo de espera para agregação termina dentro de uma janela temporal definida[1];

[1]Caso a regra seja definida para accionar se encontrar 2 eventos que a satisfaçam no prazo de 1 minuto, se eventualmente existirem 50 situações num minuto, para que não hajam 50 notificações em tão poucos segundos, define-se a *time unit* para 1 minuto para que, nesse caso, não notifique mais do que uma vez por minuto

- ***On time window expiration:*** A despoletar quando a janela temporal de espera para agregação termina.

Para qualquer um destes *triggers*, é possível definir acções como notificar utilizadores, executar um qualquer *script*, exportá-lo para um sistema externo de *ticketing* para investigação, entre outros.

3.3.5 Criação de regras

O desenvolvimento de regras é um processo que pressupõe a definição de três atributos:

1. **Condição**
2. **Agregação**
3. **Acção**

Veja-se o caso da criação de uma regra de conjunção que é accionada perante uma situação de ataques a autenticação por força bruta. Existe, por exemplo, um IDS que caso detecte várias tentativas de *logins* falhados regista um evento que, depois de normalizado, entra na plataforma ArcSight com os seguintes campos preenchidos:

- **Category Technique:** /Brute Force/Login
- **Category Outcome:** /Attempt

Quanto aos eventos de *login* bem sucedidos, estes eventos têm estes campos preenchidos com:

- **Category Behavior:** /Authentication/Verify
- **Category Outcome:** /Success

Assim, esta regra não é passível de ser concretizada como uma regra simples, pois é necessário conjugar estes dois tipos de eventos. A definição da condição sob a qual a regra será despoletada é feita segundo uma lógica booleana. Definem-se as condições para os eventos de ataques de força bruta e de *login* bem sucedido, e depois é necessário conjugar ambos. Isso é feito com base no operador *Matching Event*, onde se designa as relações que ambos os tipos de eventos devem contrair entre si. No caso, deseja-se que as seguintes premissas sejam satisfeitas (Seja *Brute_Force* o evento que representa um ataque por força bruta e *Login_Success* o que represente um *login* efectuado com sucesso):

- Brute_Force.End Time $\leq$ Login_Success.End Time

- Brute_Force.Target Address = Login_Success.Target Address
- Brute_Force.Attacker Address = Login_Success.Attacker Address

Por outras palavras, estas condições ditam que o evento de ataque por força bruta deverá ocorrer anteriormente ao evento de *login* bem sucedido, e que ambos deverão ser originados do mesmo endereço IP (atacante) para um mesmo endereço IP (alvo). A Figura 3.4 representa a forma como as condições desta regra podem ser definidas no ArcSight.

Figura 3.4: Definição de condições de regras de correlação

De seguida, caso seja requirido, define-se o número de eventos que se devem esperar antes de accionar a regra, e em que condições esta agregação deve ser feita. Deve ser definido o número de eventos que devem coincidir com a condição definida (*# of matches*) num determinado espaço de tempo (*time frame*). Para além disso, esta agregação pode ser feita segundo condições baseadas no conteúdo dos eventos, isto é, pode restrigir-se a agregação para que apenas se faça caso um dado conjunto de campos sejam idênticos ou únicos. A Figura 3.5 retrata estas definições.

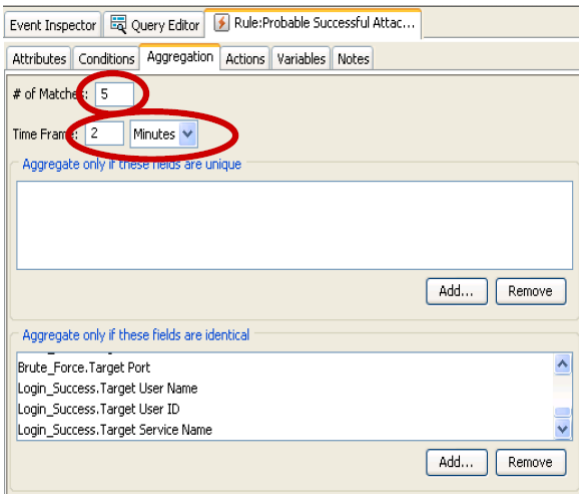


Figura 3.5: Definição de condições de agregação

Finalmente, resta definir as acções que devem ser executadas caso todas as condições definidas sejam satisfeitas, tendo em conta os *triggers* vistos anteriormente. É possível definir uma ou mais acções para cada um, bem como definir múltiplos *triggers* para a mesma regra. A Figura 3.6 ilustra a forma como se adicionam acções para qualquer *trigger*. Neste exemplo, será aberto um caso para investigação e enviada uma notificação para a equipa destacada para emergências, CERT (Cyber Emergency Readiness Team). O evento de correlação gerado terá também os campos *categoryDeviceGroup* e *agentSeverity* preenchidos com os valores "/Security Information Manager"e "High", respectivamente.

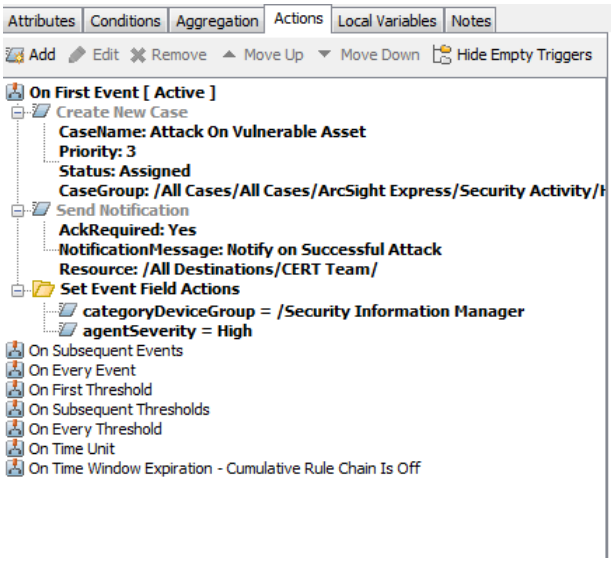


Figura 3.6: Definição de acções em regras de correlação

3.4 Conclusão do capítulo

Este capítulo detalhou a plataforma ArcSight, solução que será implementada no cliente, bem como dois componentes que serão de elevada importância no projecto: O desenvolvimento de `FlexConnectors` e a criação de regras de correlação para auxiliar a monitorização da infra-estrutura. O próximo capítulo apresenta o trabalho que foi realizado para configurar e instalar a solução na infra-estrutura do cliente.

Capítulo 4

Trabalho Realizado

Este capítulo descreve o trabalho efectuado para implementar de raíz a solução SIEM da ArcSight na organização em questão. Detalham-se as fases pelas quais passou esta concretização, iniciando-se na sua análise inicial de requisitos até à implementação na infra-estrutura do cliente, a qual passou, em grande parte, pelo desenvolvimento de `Flex-Connectors`, como se verá mais adiante.

Como objectivo primário, o cliente usará esta solução SIEM como forma de se manter cumpridor de três normas regulatórias essenciais para a sua actividade: **PCI DSS**, **Visa** e **MasterCard**. É a partir desta plataforma que se mostram evidências de que o cliente está a monitorizar constantemente a sua infra-estrutura. Paralelamente às auditorias, a perspectiva futura aponta para a criação de uma equipa que diariamente analisará a actividade centralizada no ArcSight.

4.1 Levantamento de requisitos do cliente

Ao fazer a sua própria análise de necessidades a serem satisfeitas com esta plataforma SIEM, o cliente delineou os seguinte objectivos:

- Retenção de eventos *online*: Três meses para efeitos de correlação; Um ano para armazenamento de longa duração.
- Armazenamento *offline*: Sete anos.
- Solução *agentless*: Os `connectors` não deverão ser instalados na máquina fonte, salvo necessidades do próprio agente.

Quanto aos dispositivos a monitorizar, o cliente definiu concretamente as suas necessidades a curto e médio prazo. Numa primeira fase, a recolha de *logs* é implementada sobre as máquinas que executam operações críticas: Servidores de autenticação (*Domain*

Controllers que concretizam a *Active Directory*[1]), IDS (tanto *host-based* como *network-based*) e o vasto conjunto de máquinas, bases de dados e aplicações proprietárias que trabalhem sobre dados de cartões e transacções bancárias.

Numa segunda fase estão também incluídos os *logs* de *firewalls*, *antivírus*, *routers*, *load balancers*, VPN's (Virtual Private Network), bem como servidores de DHCP (Dynamic Host Configuration Protocol), de comunicações, Servidores Web (actividade do serviço *Apache*), entre outras aplicações e máquinas com variados fins e actividades.

No total, a monitorização é feita sobre 175 máquinas e 243 fontes diferentes de *logs*, tendo em conta que um activo pode enviar mais do que um tipo de *log*.

Após o cliente ter apresentado as necessidades atrás referidas, a sua análise definiu o que está descrito nas próximas secções.

4.1.1 Retenção de eventos

De modo a cumprir o requisito de concretização de dois períodos de retenção *online*, a plataforma é composta por dois módulos que armazenam os eventos de acordo com as configurações temporais desejadas. Assim, a arquitectura apresenta **um `Manager` e um `Logger`**. O `Manager` é o elemento principal da solução, estando a seu cargo as responsabilidades de monitorização e correlação constantes de eventos. As suas capacidades estão dimensionadas para que sejam sempre disponibilizados, no mínimo, todos os eventos dos últimos três meses.

Paralelamente existe também o envio dos eventos para o `Logger`. Estes são desenhados para suportar um maior número de eventos que se poderiam registar num `Manager` e, como tal, é a solução para receber e centralizar os eventos do último ano. Com este espectro temporal mais abrangente é possível efectuar investigações num passado mais longínquo caso seja necessário.

A Tabela 4.1 apresenta as especificações das duas máquinas, desenhadas de acordo com os tempos de retenção respectivos.

	`Manager`	**`Logger`**
CPU	Intel Xeon E5620 Quad Core 2.4 GHz (2x)	Intel Xeon E5620 Quad Core 2.4 GHz (2x)
RAM	36GB	26GB
Armazenamento	6x600GB @ RAID 10	6x1TB @ RAID 1
S.O.	RedHat Enterprise Linux 5.5, x64	RedHat Enterprise Linux 5.5, x64

Tabela 4.1: Especificações técnicas do Manager e Logger

[1]Concretização da Microsoft de directório de utilizadores baseado no protocolo LDAP (Lightweight Directory Access Protocol)

Apesar de estar fora do âmbito deste projecto, será configurado, a partir do `Logger`, o envio dos eventos que estão guardados há mais de um ano para um **arquivo *offline***, que os reterá pelo período definido de sete anos.

4.1.2 Solução *agentless*

De modo a que esta solução seja o menos intrusiva para as máquinas que compõem a infra-estrutura do cliente e que serão auditadas, os `connectors` deverão ficar instalados remotamente. No entanto, instalá-los no próprio `Manager` ou `Logger` acabará por sobrecarregá-los demasiado. Assim, existem máquinas adicionais intermédias para alojar todos os agentes necessários, excepto os que, por questões de concepção do próprio `connector`, têm necessariamente de ser instalados localmente.

Utilizar a `Connector Appliance` para alojar e gerir os agentes poderia ser também uma opção mas dada a heterogeneidade dos sistemas que serão monitorizados pelo ArcSight, sabe-se de antemão que existirão `connectors` em grande número e aglomerá-los num só servidor não só o tornará um ponto único de falha como também requirirá enormes capacidades de processamento.

Assim, existem mais 4 servidores adicionais à solução ArcSight onde estão instalados os `connectors`. Desta forma consegue-se uma distribuição de esforço mais eficiente, não confinando os agentes a uma só máquina, preenchendo também o requisito do cliente. Estas máquinas, que se apelidam de `Server Connector`, são 4 e estão associadas aos pontos geográficos onde se encontram os activos a auditar. Doravante, estes servidores designar-se-ão como `Server Connector` A, `Server Connector` B, `Server Connector` C e `Server Connector` D[2], sendo A, B, C e D os quatros pontos geográficos onde existem activos a monitorizar.

4.1.3 Gestão centralizada dos connectors

Estando os `connectors` dispersos por, pelo menos, quatro servidores, é desejável que qualquer configuração seja feita num local comum a todos os `connectors`. Isto não só é útil pelo facto de ser independente da dispersão dos agentes pelos quatro `Server Connector`'s, como também possibilita alterações em vários `connectors` em simultâneo. Para isso, é utilizada uma `Connector Appliance` que estabelece uma ligação constante até cada `connector`. Este canal serve para alterar configurações nos agentes de forma remota, bem como iniciar, parar ou reiniciar o executável dos mesmos, entre outras tarefas administrativas.

[2]Este projecto não compreende os activos do polo D, dado que a sua monitorização será incluída numa fase posterior à escrita deste documento

4.1.4 Avaliação dos activos a monitorizar

Após o cliente ter definido o conjunto de máquinas cujos *logs* estarão constantemente a ser enviados e centralizados na plataforma, existe uma análise que tem de ser feita com vista a identificar a existência, ou não, de um `connector` já disponibilizado pela ArcSight para a tecnologia respectiva[3].

Analisando a lista de tecnologias e tipos de *logs* que serão monitorizados identificam-se algumas compatibilidades. Nestes casos em que já existe um `SmartConnector` disponível (i.e., um `connector` desenvolvido pela ArcSight capaz de processar registos da tecnologia em questão), a sua inclusão passa simplesmente pela sua instalação e configuração adaptada ao tipo de `connector`.

No entanto, é relevante relembrar que uma nuance importante neste projecto é a sua especificidade quanto à maioria dos dispositivos a monitorizar. Uma análise à tabela 4.2 mostra que serão implementados, no total, 27 `connectors` diferentes, 15 dos quais são desenvolvidos à medida, sendo que 4 deles auditam aplicações desenvolvidas internamente (os últimos 4 itens da tabela). Não obstante a importância inerente aos sistemas cujos registos serão recolhidos por `SmartConnectors`, o maior esforço na implementação deste projecto incidiu na criação dos `FlexConnectors` para conseguir processar os *logs* de formato desconhecido pela ArcSight.

[3] `http://www.arcsight.com/collateral/ArcSight_Supported_Products.pdf`

Tecnologia	`SmartConnector`	`FlexConnector`
Apache	**X**	
DHCP	**X**	
IBM AIX (Audit)	**X**	
Unix (Audit)	**X**	
Windows (Event Log)	**X**	
Cisco ASA	**X**	
Cisco Router	**X**	
Symantec Endpoint Protection	**X**	
Check Point FW	**X**	
Microsoft IAS	**X**	
Microsoft TMG	**X**	
Microsoft SQL Server	**X**	
IBM z/OS		**X**
IBM DB2		**X**
IBM CICS		**X**
IBM Informix		**X**
IBM Tivoli Identity Manager		**X**
Sourcefire NIDS		**X**
OSSEC HIDS		**X**
Cisco ACE		**X**
Cisco TACACS+		**X**
JBoss Application Server		**X**
Avaya Switches		**X**
Aplicação proprietária 1		**X**
Aplicação proprietária 2		**X**
Aplicação proprietária 3		**X**
Aplicação proprietária 4		**X**

Tabela 4.2: SmartConnectors disponíveis *vs.* FlexConnectors a desenvolver

4.2 Desenho

A Figura 4.1 mostra o desenho arquitectural da plataforma ArcSight que será implementada na rede do cliente, fruto da análise de requisitos efectuada. A figura apresenta quatro locais onde existem máquinas a monitorizar. Em cada um deles está instalado um servidor (`Server Connector`) que alojará os agentes a configurar para enviar eventos para as *appliances* ArcSight, sendo que estas últimas estão todas no polo A.

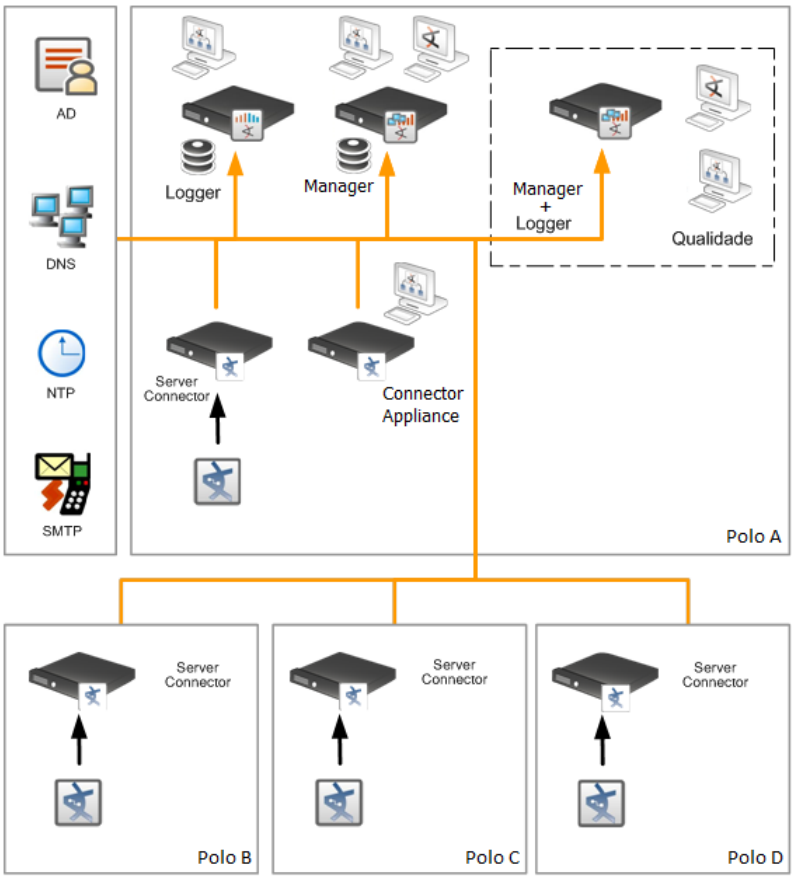


Figura 4.1: Arquitectura a implementar no cliente

Focando agora o interesse nas máquinas ArcSight, é possível verificar que estão montados dois ambientes de correlação e arquivo distintos:

- **Ambiente de Produção:** Ambiente composto por um `Logger` e um `Manager`.
- **Ambiente de Qualidade/Desenvolvimento:** Ambiente composto por um `Manager` e `Logger` numa única *appliance* (*All-in-one*). Este ambiente serve maioritariamente para serem efectuados testes a novos `connectors` ou recursos a criar. Para Produção enviam-se apenas os eventos que tenham sido validados com sucesso em Qualidade (isto é, o processamento está a ser feito correctamente e não existe perda de *logs*).

Adicionalmente, as *appliances* estão ligadas a alguns serviços externos, nomeadamente:

- **Active Directory:** Para mediar as autenticações nas várias consolas de monitorização das *appliances*;
- **DNS:** Para resolução de nomes de eventuais máquinas que constem nos eventos;
- **NTP:** Para sincronização dos relógios das *appliances* e `Server Connector`'s;
- **SMTP:** Para envio de *mails*.

4.3 Pré-configuração

Antes de serem enviadas ao cliente, é necessário efectuar algumas pré-parametrizações em todas as *appliances*. Esta fase preparatória, denominada de ***staging***, tem como objectivo efectuar eventuais *upgrades* necessários às máquinas ArcSight, caso tenham sido disponibilizadas novas versões enquanto o transporte das *appliances* era feito. As parametrizações feitas neste espaço de tempo pré-preparatório prendem-se com a execução dos procedimentos da instalação dos componentes principais de cada uma das máquinas, como o *software* de correlação (`Manager`), a construção das bases de dados do `Manager` e do `Logger`, etc.

Para além dessas configurações, outras se podem fazer para que quando as *appliances* chegarem ao cliente sejam praticamente *plug-and-play*. Nomeadamente, dado que o cliente providenciou antecipadamente as devidas informações, conseguiu-se pré-configurar os endereços IP de cada interface das máquinas, *hostnames*, servidores de DNS (Domain Name System), SMTP (Simple Mail Transfer Protocol), NTP (Network Time Protocol), etc.

Após estas definições preliminares, as máquinas são enviadas ao cliente onde serão instaladas no seu *datacenter* já com endereços e ligações a outros serviços definidos.

4.4 Instalação no cliente

Com as *appliances* já instaladas no cliente e pré-configuradas, o trabalho de implementação pode iniciar-se. Esta instalação compreende inúmeras tarefas que se podem dividir em quatro grandes grupos:

1. **Desenvolvimento de `FlexConnectors`:** Dada a especificidade deste projecto, a maioria dos sistemas que são auditados pela plataforma SIEM não registam a sua actividade num formato por ela conhecida pelas razões já referidas: Ou porque

são dispositivos não suportados nativamente pela ArcSight, ou porque são aplicações desenvolvidas internamente no cliente. Para estes casos torna-se necessário o desenvolvimento de um `FlexConnector` para ler e processar esses registos.

2. **Instalação de `SmartConnectors`:** Para os sistemas cujos *logs* são de um formato conhecido pela ArcSight e já está desenvolvido um agente à sua medida, estes serão instalados seguindo uma metodologia específica dependente do tipo de `SmartConnector` (i.e., se é consumidor de uma base de dados de registos, de ficheiro, de *syslog*, etc). Tanto neste ponto como no anterior, não serão abordados os sistemas do polo D pois, como já foi denotado, não estão ainda no âmbito deste projecto.

3. **Criação de conteúdos:** A partir do momento em que os `connectors` se encontram a reencaminhar os *logs* que processam para o repositório centralizado, pode começar-se a fase de criação de recursos de monitorização da infra-estrutura. Estes recursos compreendem relatórios de actividade, *dashboards*, regras de correlação que denotem um comportamento anómalo e que podem despoletar variadas acções, desde simples notificações a instruções que accionem mecanismos de segurança adicionais de forma proactiva, etc.

4. **Optimização da solução:** Numa fase final, é essencial optimizar o funcionamento dos `connectors` existentes, nomeadamente quanto aos mecanismos de agregação e filtragem de eventos que fazem com que o número de eventos enviados decresça de modo a utilizar-se eficientemente a largura de banda disponível. Outros parâmetros que não devem ser descurados prendem-se com o tamanho da *cache* de cada agente, se faz o envio em tempo-real ou em blocos, etc.

   Existem também outras tarefas que, apesar de não estarem associadas a `connectors`, também são implementadas nesta secção e serão abordadas mais adiante.

4.4.1 Desenvolvimento de FlexConnectors: Cenários práticos

Dado a especificidade deste projecto, o desenvolvimento de agentes torna-se parte central do trabalho. Tal como ficou definido na análise de requisitos, 15 dos 27 agentes a instalar terão de ser desenvolvidos à medida por processarem *logs* de produtos não suportados nativamente pela ArcSight ou porque se tratam de aplicações proprietárias do cliente.

Esta criação de `FlexConnectors` pressupõe, normalmente, dois elementos a fornecer pelo cliente: Amostras dos *logs* a processar e a sua respectiva documentação. As amostras possibilitam que o trabalho de desenvolvimento se possa processar *offline* para que quando se instale o `connector` no seu ambiente, este inicie o processamento dos registos já muito perto da versão final necessitando, eventualmente, de apenas pequenos ajustes. A documentação é essencial para uma leitura correcta dos *logs* e, assim, o

mapeamento para os campos do formato conhecido pela plataforma seja feito de forma coerente.

O primeiro exercício a fazer antes de iniciar propriamente a criação de um FlexConnector é perceber qual o tipo de agente a desenvolver. A Tabela 4.3 reflecte esta análise. Dos 15 `FlexConnectors`, terão que se desenvolver 4 que receberão os eventos via *Syslog*, sendo que os restantes irão ler ficheiros de uma ou mais pastas. Como já foi visto anteriormente, ambos os `connectors` baseiam-se em expressões regulares para avaliar e fragmentar os registos.

Tecnologia	**`FlexConnector`**
IBM z/OS	Ficheiro
IBM DB2	Ficheiro
IBM CICS	Ficheiro
IBM Informix	Ficheiro
IBM Tivoli Identity Manager	Ficheiro
Sourcefire NIDS	*Syslog*
OSSEC HIDS	*Syslog*
Cisco ACE	*Syslog*
Cisco TACACS+	Ficheiro
JBoss Application Server	Ficheiro
Avaya Switches	*Syslog*
Aplicação proprietária 1	Ficheiro
Aplicação proprietária 2	Ficheiro
Aplicação proprietária 3	Ficheiro
Aplicação proprietária 4	Ficheiro

Tabela 4.3: Tipos de FlexConnectors a desenvolver

Todos os `FlexConnectors` desenvolvidos constam no Anexo I deste relatório. Dois deles são detalhados seguidamente para ilustrar o método de criação. O primeiro é o agente desenvolvido para o IBM CICS, que processa ficheiros. Depois, de modo a cobrir alguns aspectos mais avançados, apresenta-se o agente criado para receber via *syslog* os *logs* do detector de intrusões OSSEC.

IBM CICS

Este servidor de transacções regista a actividade das suas aplicações sob o seguinte formato:

```
2012-06-19 08.47.21(+01)AAAAAA BBBBCCCCDDDDEEEEEEEE000XXXXXXXXXXXXXX
   SUCCESSYYYYYYYYYYYYYYYYYYYYYYYY ZZZZZZZZZZZZZZZZZZZZZZZZZ
```

A documentação fornecida para a leitura destes *logs* esclarece o seu formato. Este é caracterizado pelo seguinte:

1. 19 caracteres: Data/Hora (GMT)
2. 5 caracteres: Diferença para hora local
3. 8 caracteres: ID Utilizador
4. 4 caracteres: Aplicação
5. 4 caracteres: Transacção CICS
6. 4 caracteres: Terminal CICS
7. 8 caracteres: Região CICS
8. 3 caracteres: Código de país
9. 20 caracteres: Acção executada
10. 60 caracteres: Entidade envolvida
11. 7 caracteres: Sucesso/Insucesso
12. 70 caracteres: Observações adicionais
13. 120 caracteres: Entidade envolvida (em hexadecimal)

Nestes registos, caso os campos não preencham na totalidade o número de caracteres disponíveis, estes são preenchidos com espaços. Nos exemplos anteriores, por simplificação, estes foram omitidos.

Com amostras de *logs* e respectiva documentação explicativa, é possível iniciar o desenvolvimento deste `FlexConnector`. Inicialmente, o ficheiro de configuração define os 13 *tokens* que se podem extrair do *log*:

```
token.count=13

token[0].name=event_time
token[0].type=String
token[1].name=timezone
token[1].type=String
token[2].name=user_id
token[2].type=String
token[3].name=application
token[3].type=String
token[4].name=cics_trans
token[4].type=String
token[5].name=cics_term
```

```
token[5].type=String
token[6].name=cics_reg
token[6].type=String
token[7].name=country_code
token[7].type=String
token[8].name=action
token[8].type=String
token[9].name=entity
token[9].type=String
token[10].name=succ_fail
token[10].type=String
token[11].name=info
token[11].type=String
token[12].name=hexa_entity
token[12].type=String
```

Após o isolamento dos dados do registo, é feita a normalização para o formato comum da plataforma, atribuindo cada *token* relevante para um campo do referido formato. Importa referir a importância de campos *deviceCustom* e *flex*, utilizados na normalização abaixo. Estes campos são úteis quando se estão a processar *logs* tão específicos cujos atributos não se coadunam com nenhum dos campos comuns do formato da plataforma. Assim, estes campos podem ser preenchidos com qualquer informação, sendo atribuída uma *label* que identifica o tipo de dados desse campo personalizável.

```
event.deviceReceiptTime=__createSafeLocalTimeStamp(__concatenate(
    event_time,"(GMT",timezone,":00)"),"yyyy-MM-dd HH.mm.ss'('z')'")
event.deviceTimeZone=__concatenate(__stringConstant("GMT"),timezone)
event.destinationUserName=user_id
event.requestClientApplication=application
event.deviceCustomString1=info
event.deviceCustomString1Label=__stringConstant("Additional Info")
event.deviceCustomString2=cics_trans
event.deviceCustomString2Label=__stringConstant("CICS Transaction")
event.deviceCustomString3=cics_term
event.deviceCustomString3Label= __stringConstant("CICS Terminal")
event.deviceCustomString4=cics_reg
event.deviceCustomString4Label=__stringConstant("CICS Region")
event.deviceCustomString5=__simpleMap(country_code,"024=AO","616=PL
    ","620=PT","642=RO")
event.deviceCustomString5Label=__stringConstant("Instance")
event.name=action
event.deviceAction=action
event.message=__replaceAll(entity,"[^A-Za-z0-9!+,-./:;=_<>\x22\x27
    ]","?")
event.eventOutcome=succ_fail
event.deviceCustomString6Label=__stringConstant("Environment")
event.flexString1=hexa_entity
event.flexString1Label=__stringConstant("Hexadecimal event.message")

event.deviceSeverity=__simpleMap(action,"OPEN SESSION=0","CLOSE SESSION
    =0","DELETE SESSION=1","CREATE SESSION=1","OPEN TRACE=2","CLOSE
    TRACE=0","VIEW TRACE=2","CHANGE PARAM=2","VIEW FILE=2","START
```

```
    LISTNER=2","STOP LISTNER=1","CONNECT=0","DISCONNECT=0","DELETE MSG
    =1","VIEW MSG=2","EDIT MSG=3","GO FILE=2","DELETE FILE=1","CHANGE
    FILESTATUS=1","ENTER APPL=1","EXIT APPL=0","EXCHANGE KEY=0","START
    TRANS.F.T.=2","GET FILE=2","REFRESH DIRECTORY=0")

event.deviceVendor=__stringConstant("IBM")
event.deviceProduct=__stringConstant("z/OS App")
```

O mapeamento do grau de severidade foi definido da seguinte forma:

```
severity.map.low.if.deviceSeverity=0
severity.map.medium.if.deviceSeverity=1
severity.map.high.if.deviceSeverity=2
severity.map.veryhigh.if.deviceSeverity=3
```

Seguidamente, é apresentada a expressão regular que analisa estes *logs*:

```
regex=(\\d{4}-\\d{2}-\\d{2}\\s\\d{2}\\.\\d{2}\\.\\d{2})\\((.{3})\\)
    (.{8})(.{4})(.{4})(.{4})(.{8})(.{3})(.{20})(.*)(SUCCESS|FAIL.{3})
    (.{70})(.{120})
```

OSSEC HIDS

Este segundo exemplo apresenta a concepção de um `FlexConnector` que recebe *logs* via *syslog*. Em traços gerais, acaba por ser em tudo semelhante a um `FlexConnector` leitor de ficheiros baseado em expressões regulares. No entanto, apesar de existirem algumas diferenças subtis, estes *logs* possuem diferentes formatos para diferentes tipos de mensagem e, por isso, é necessário tratar cada tipo de mensagem de forma diferente para que se consiga extrair toda a informação útil.

Assim, surge neste `FlexConnector` o conceito de **sub-messages**. O objectivo destas sub-mensagens é o de poder tratar diferentes mensagens com diferentes expressões regulares.

Estes *logs* são produzidos no formato apresentados nos dois exemplos seguintes:

```
<132>Jun 20 10:00:01 ossecserv ossec: Alert Level: 5; Rule: 1005 -
    Syslogd restarted.; Location: (host.a) 111.34.123.18->/var/log/
    messages;  Jun 20 10:00:01 host.a syslogd: restart
<132>Jun 20 10:41:49 ossecserv ossec: Alert Level: 7; Rule: 551 -
    Integrity checksum changed again (2nd time).; Location: (host.b)
    121.233.12.194->syscheck;  Integrity checksum changed for: '/etc/
    rsyslog.conf'
```

Estas duas mensagens *syslog* representam dois registos no formato vulgar do OSSEC. No entanto, após análise mais detalhada às amostras oferecidas pelo cliente, constatou-se que existem outros tipos de regras que, quando disparadas, registam mais informação e

por essa razão é necessário tratar algumas dessas mensagens como *sub-messages*. Assim, o ficheiro de configuração criado começa por tratar a informação comum a todas as mensagens, que é até ao identificador da regra disparada (valor inteiro imediatamente a seguir a "Rule:"):

```
regex=(.+):\\s*Alert Level:\\s*(\\d+);\\s*Rule:\\s*(\\d+)\\s*\\-(.*)

token.count=4

token[0].name=module
token[0].type=String
token[1].name=alert_level
token[1].type=String
token[2].name=rule_id
token[2].type=String
token[3].name=remaining
token[3].type=String


event.deviceCustomString1Label=__stringConstant("Module")
event.deviceCustomString1=module
event.deviceCustomString2Label=__stringConstant("Facility")
event.deviceCustomString2=_SYSLOG_FACILITY
event.deviceReceiptTime=_SYSLOG_TIMESTAMP
event.deviceSeverity=alert_level
event.deviceEventClassId=rule_id
event.deviceHostName=_SYSLOG_SENDER
event.deviceProduct=__stringConstant("OSSEC")
event.deviceVendor=__stringConstant("OSSEC")

severity.map.veryhigh.if.deviceSeverity=8..99
severity.map.high.if.deviceSeverity=5,6,7
severity.map.medium.if.deviceSeverity=2,3,4
severity.map.low.if.deviceSeverity=0,1
```

Como já foi descrito, o desenvolvimento de um `FlexConnector` de *Syslog* é em tudo semelhante ao desenvolvimento de um para ler ficheiros. A única diferença a realçar prende-se com quatro *tokens* especiais incorporados neste `FlexConnector`, três dos quais estão a ser utilizados atrás:

- **_SYSLOG_FACILITY:** Campo *facility* que consta no cabeçalho de um pacote *syslog*;
- **_SYSLOG_TIMESTAMP:** *Timestamp* a retirar do cabeçalho *syslog*;
- **_SYSLOG_SENDER:** *Hostname* ou IP a retirar do cabeçalho *syslog*;
- **_SYSLOG_PRIORITY:** Campo *priority* que consta no cabeçalho de um pacote *syslog*;

Processada que está a parte comum a qualquer *log* desta tecnologia, é necessário definir o identificador de cada *submessage* e respectivo fragmento a processar:

```
submessage.messageid.token=rule_id
submessage.token=remaining

submessage.count=7
```

Tal como se pode observar, seguidamente serão definidos 7 *parsers* diferentes para o mesmo número de formatos de mensagens diferente. Uma dessas *submessages* (a primeira que se visualiza de seguida) não tem qualquer *messageid* definido. Isto significa que qualquer mensagem cujo identificador não seja um dos seis apresentados depois, será processada segundo este padrão.

```
submessage[0].pattern.count=1
submessage[0].pattern[0].regex=(.+);\\s*Location:\\s*\\((\\S+)\\)\\s
    *(\\d{1,3}.\\d{1,3}.\\d{1,3}.\\d{1,3})\\->(\\S+);\\s*(.*)
submessage[0].pattern[0].fields=event.name,event.destinationHostName,
    event.destinationAddress,event.fileName,event.message

submessage[1].messageid=502
submessage[1].pattern.count=1
submessage[1].pattern[0].regex=(.+);\\s*Location:\\s*.+\\->(.+);(.+)
submessage[1].pattern[0].fields=event.name,event.destinationProcessName
    ,event.message

submessage[2].messageid=504
submessage[2].pattern.count=1
submessage[2].pattern[0].regex=(.+);\\s*Location:\\s*.+\\->(.+);(.+)
submessage[2].pattern[0].fields=event.name,event.destinationProcessName
    ,event.message

submessage[3].messageid=5402
submessage[3].pattern.count=1
submessage[3].pattern[0].regex=(.+);\\s*Location:\\s*\\((\\S+)\\)\\s
    *(\\d{1,3}.\\d{1,3}.\\d{1,3}.\\d{1,3})\\->(\\S+);\\s*user:\\s*(\\S
    +);(.*)
submessage[3].pattern[0].fields=event.name,event.destinationHostName,
    event.destinationAddress,event.fileName,event.sourceUserName,event.
    message

submessage[4].messageid=5501
submessage[4].pattern.count=1
submessage[4].pattern[0].regex=(.+);\\s*Location:\\s*\\((\\S+)\\)\\s
    *(\\d{1,3}.\\d{1,3}.\\d{1,3}.\\d{1,3})\\->(\\S+);\\s*user:\\s*(\\S
    +);(.*)
submessage[4].pattern[0].fields=event.name,event.destinationHostName,
    event.destinationAddress,event.fileName,event.sourceUserName,event.
    message

submessage[5].messageid=5502
submessage[5].pattern.count=1
```

```
submessage[5].pattern[0].regex=(.+);\\s*Location:\\s*\\((\\S+)\\)\\s
    *(\\d{1,3}.\\d{1,3}.\\d{1,3}.\\d{1,3})\\->(\\S+);\\s*user:(\\s*\\S
    +);(.*)
submessage[5].pattern[0].fields=event.name,event.destinationHostName,
    event.destinationAddress,event.fileName,event.sourceUserName,event.
    message

submessage[6].messageid=5715
submessage[6].pattern.count=1
submessage[6].pattern[0].regex=(.+);\\s*Location:\\s*\\((\\S+)\\)\\s
    *(\\d{1,3}.\\d{1,3}.\\d{1,3}.\\d{1,3})\\->(\\S+);\\s*srcip:\\s*(\\d
    {1,3}.\\d{1,3}.\\d{1,3}.\\d{1,3});\\s*user:\\s*(\\S+);(.*)
submessage[6].pattern[0].fields=event.name,event.destinationHostName,
    event.destinationAddress,event.fileName,event.sourceAddress,event.
    sourceUserName,event.message
```

4.4.2 Instalação de SmartConnector's

Um dos resultados da análise de requisitos revelou o número de agentes já disponibilizados pela ArcSight para recolher e processar os *logs* necessários de alguns dos activos do cliente. Nessa análise ficou estipulado que seriam integrados 9 `SmartConnectors` diferentes, embora possam existir `SmartConnectors` iguais (i.e., para o mesmo tipo de *log*) em `Server Connector`'s distintos. A Tabela 4.4 apresenta os vários agentes instalados nesta fase e a sua respectiva função, bem como o seu método de recolha de registos.

`SmartConnector` para:	**Função**	***Logs* recebidos via:**	**`Server Connector`**
Apache HTTP Server	Recolha de *logs* de erro e de acessos externos ao servidor Web Apache	*Syslog*	A
Check Point	Recolha de *logs* de *firewalls* Check Point	LEA (Log Export API)	B
Cisco Routers/ASA	Recolha de *logs* de *Routers*/*Firewalls* Cisco	*Syslog*	A
IBM AIX (Audit)	Monitorização de acessos a objectos neste sistema operativo por parte de utilizadores	Ficheiro	A
Microsoft DHCP	Recolha de *logs* de actividade DHCP (atribuições de IP, etc)	Ficheiro	Local
Microsoft SQL Server	Recolha de registos de actividade em bases de dados	Ficheiro	A
Microsoft Windows Event Log	Registos de actividade em Sistemas Operativos Windows	*syslog*	A + B + C
Symantec Endpoint Protection	Recolha de *logs* do gestor central de Antivírus	Base de Dados	B
Unix Audit	Registos de actividade em Sistemas Operativos Unix	*Syslog*	A + B + C

Tabela 4.4: SmartConnectors a configurar

Independentemente do `SmartConnector` ou `FlexConnector` a instalar, a sua instalação inicia-se sempre da mesma forma: com um executável que descarrega todos os elementos necessário para que o agente faça o seu trabalho. Por outro lado, as configurações avançadas de cada `connector` diferem bastante entre si. Na próxima secção é possível seguir uma instalação detalhada de um `SmartConnector`, sendo que nas secções subsequentes listam-se somente as configurações específicas para cada `connector`.

4.4.2.1 SmartConnector para Apache HTTP Server

Após serem descompactados os elementos necessários ao `connector` para a sua tarefa de recolha e normalização dos *logs*, o passo seguinte é a escolha de um destinatário para onde serão enviados os eventos (Figura 4.2).

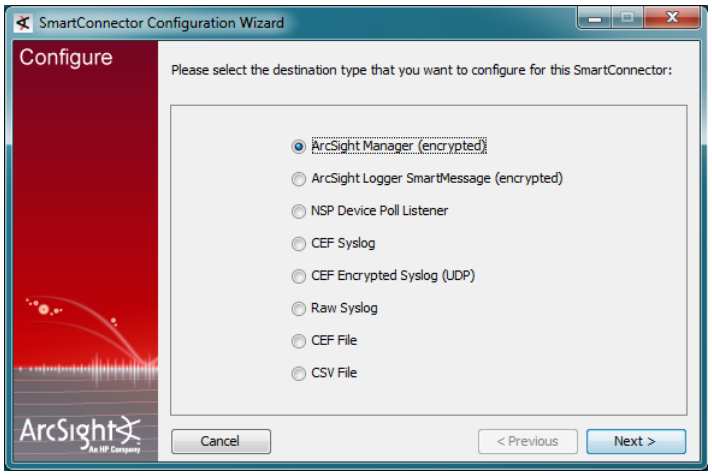


Figura 4.2: Instalação de SmartConnector (1)

O `connector` pode ser configurado para enviar eventos para diversas entidades:

- `ArcSight Manager` (ligação segura)
- `ArcSight Logger` (ligação segura)
- NSP Device Poll Listener[4]
- *Syslog* (formato CEF (Common Event Format) em claro, cifrado ou formato original em claro)
- Ficheiro (formato CEF ou CSV)

Neste exemplo, os eventos serão enviados para um `Manager`. De seguida surge o painel ilustrado na Figura 4.3, onde é necessário indicar o *hostname* do `Manager` de destino, o porto para comunicação e mais duas opções de escolha[7]:

- **AUP Master Destination:** Se for desejável que este `Manager` seja o gestor de ArcSight Update Packs (AUP). Estes ficheiros são normalmente disponibilizados e utilizados pela ArcSight para fazer *upgrades* de versão de `connectors` ou simples *updates* a erros menores. Se estes AUP's forem depositados num `Manager` que seja "AUP Master"ele irá enviar automaticamente estas actualizações para todos os `connectors` que o tomem como AUP Master.
- **Filter Out All Events:** Caso se queira que este `connector` não envie eventos para este destinatário (pressupondo que tem outros destinos configurados). Isto é particularmente útil se se estiver a equacionar a hipótese de utilizar este `Manager` apenas como "AUP Master", ficando assim apenas responsável por eventuais actualizações.

[4]O Network Synergy Platform (NSP) consiste na *appliance* que alberga os módulos opcionais de resposta automática a incidentes

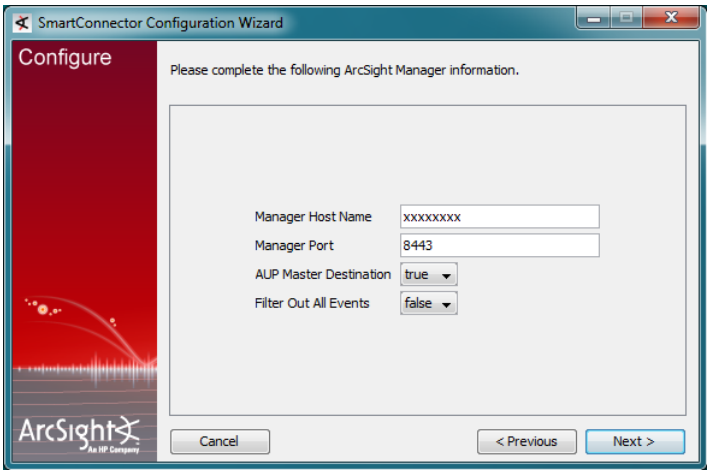


Figura 4.3: Instalação de SmartConnector (2)

Finalmente, na Figura 4.4 é necessário introduzir as credenciais de um utilizador válido na plataforma com privilégios de instalação de agentes.

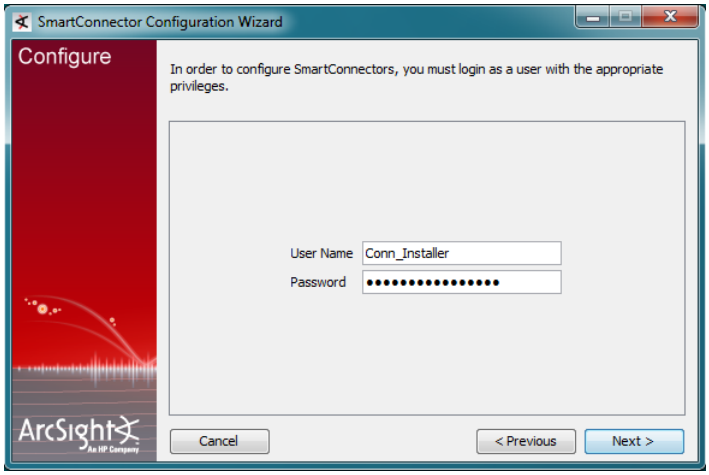


Figura 4.4: Instalação de SmartConnector (3)

Algumas considerações a ter em conta ao nível da segurança da plataforma:

- A introdução de credenciais de um utilizador válido e com privilégios suficientes para a instalação de `connectors` assegura que esta instalação é devidamente autorizada;
- Cada instalação de um `connector` pressupõe que foi copiado manualmente o certificado digital do `Manager` que contém a sua chave pública, usada para assegurar a confidencialidade da comunicação;
- No caso do `Logger`, esta cópia manual de certificado não é necessária pois a comunicação (e prévio estabelecimento de chaves) é feita sobre HTTPS (Hypertext

Transfer Protocol Secure). Aqui não são requiridas credenciais de um utilizador privilegiado mas é necessário saber o nome de um "Receiver"configurado de antemão no `Logger` para receber os eventos;[8]

- Opcionalmente, podem ser aplicados algoritmos de preservação da integridade dos eventos enviados.

Caso o certificado seja válido e a autenticação do utilizador se faça com sucesso, a instalação prossegue com a escolha do tipo de `connector` a instalar (Figura 4.5).

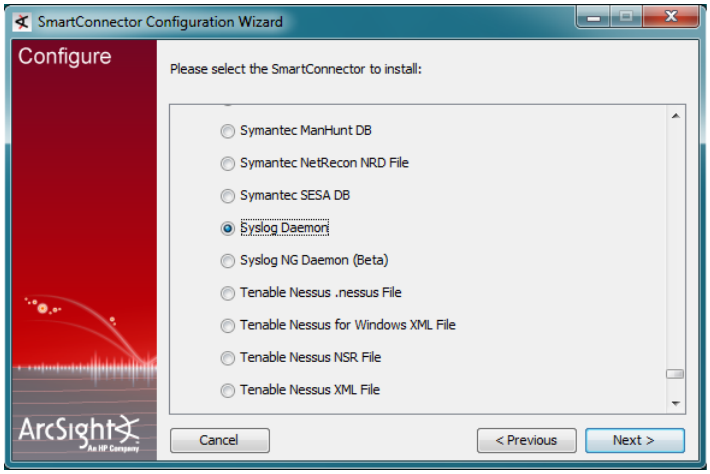


Figura 4.5: Instalação de SmartConnector (4)

Depois de escolher o tipo de `SmartConnector` ou `FlexConnector` desejado, o processo de configuração segue pedindo os parâmetros específicos do tipo de agente a instalar. No caso, e estando a instalar um `connector` para receber os logs de instâncias de servidores Web Apache, este envia *logs* via *Syslog*. Escolhendo o `SmartConnector` "Syslog Daemon"[5], os parâmetros requiridos são o porto de escuta, endereço IP (caso se deseje restringir) e protocolo de transporte (UDP (User Datagram Protocol) ou TCP (Transmission Control Protocol)), tal como é apresentado na Figura 4.6.

[5]O `SmartConnector` de *Syslog* difere um pouco dos restantes pois têm internamente um vasto conjunto de *sub-parsers* para processar *logs* de variados produtos que também transmitam via *Syslog*, daí que este `connector` seja o escolhido em diversas ocasiões.

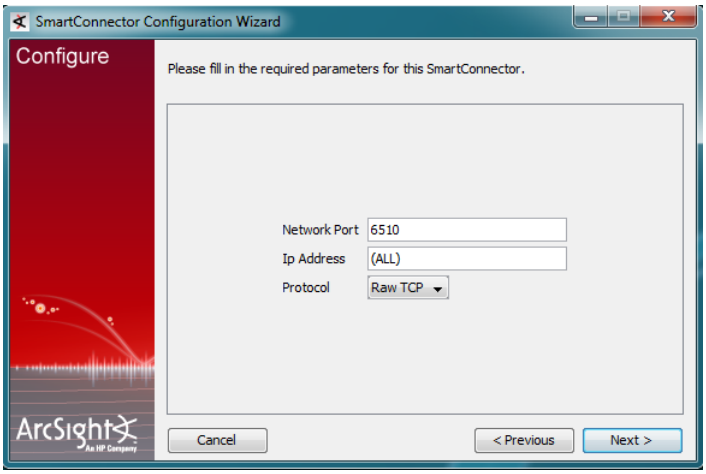


Figura 4.6: Instalação de SmartConnector (5)

No caso, os parâmetros são:

- **Network Port:** 6510
- **Ip Address:** ALL
- **Protocol:** TCP

A instalação termina com a criação do serviço para executar o `connector` ou, caso o serviço não seja desejável, este pode ser executado como uma aplicação normal.

4.4.2.2 SmartConnector para Check Point

Para configurar com sucesso este agente, os parâmetros introduzidos foram:

- **`SmartConnector`:** Check Point OPSEC NG
- **Connection Type:** clear
- **Check Point Server IP Address**[6]**:** xxx.xxx.xxx.xxx
- **Check Point Server Port:** 18184

4.4.2.3 SmartConnector para Cisco (Routers e ASA)

Para processar *logs* de *routers* e *firewalls* Cisco, o `SmartConnector` foi instruído com a seguinte informação:

- **`SmartConnector`:** Syslog Daemon
- **Network Port:** 9056

[6]No contexto da Check Point, esta máquina é a responsável por gerir uma ou mais *firewalls* Check Point

- **Ip Address:** ALL
- **Protocol:** TCP

4.4.2.4 SmartConnector para IBM AIX (Audit)

Para auditar este sistema operativo, o agente específico processa ficheiros de *logs* e foi configurado da seguinte forma:

- **`SmartConnector`:** IBM AIX Audit File
- **Folder:** /opt/ArcSight/log_sources/aixaudit

Embora a transferência dos *logs*, em todos os casos, esteja à responsabilidade do cliente, estes chegam ao `Server Connector` via SFTP (Secure File Transfer Protocol).

4.4.2.5 SmartConnector para Microsoft DHCP

Este é um caso especial por dois motivos: (1) A própria arquitectura do `SmartConnector` dita que este deve ser instalado localmente no servidor de DHCP e (2) nos polos A e B este serviço é suportado por um *cluster* activo-passivo, pelo que nestes casos está instalado um agente em cada uma das máquinas do respectivo *cluster* sendo que à semelhança da arquitectura do *cluster*, um deles estará activo enquanto que o outro está desactivado. Caso a máquina activa mude no *cluster*, o `SmartConnector` inactivo reinicia também a sua actividade, sendo que o outro passa para o estado inactivo.

No total, estão instalados cinco `SmartConnectors` que processam registos de actividade DHCP: Dois no polo A, dois no polo B e um no polo C (onde não existe *cluster*). As respectivas parametrizações foram:

Polo A: Nó 1 e Nó 2

- **`SmartConnector`:** Microsoft DHCP File
- **Log File:** \\xxxxxxxx\DhcpSrvLog-'EEE'.log[7]

Polo B: Nó 1 e Nó 2

- **`SmartConnector`:** Microsoft DHCP File
- **Log File:** \\xxxxxxxx\DhcpSrvLog-'EEE'.log

[7]Dado que os ficheiros de *logs* sofrem rotação diária, 'EEE' denota que o nome do ficheiro alterará consoante o dia da semana. Assim, o `connector` espera ficheiro com nome: DhcpSrvLog-Mon.log, DhcpSrvLog-Tue.log, etc.

Polo C

- **`SmartConnector`:** Microsoft DHCP File
- **Log File:** C:\WINDOWS\system32\dhcp\DhcpSrvLog-'EEE'.log

4.4.2.6 SmartConnector para Microsoft SQL Server

Para este agente, que lê ficheiros contendo *traces* de actividade na Base de Dados a auditar, foram feitas as seguintes configurações:

- **`SmartConnector`:** Microsoft SQL Server Multiple Instance Audit DB
- **JDBC Driver:** com.microsoft.sqlserver.jdbc.SQLServerDriver
- **Trace File Post Processing Mode:** DeleteFile[8]
- **JDBC Database URL:** jdbc:sqlserver://xxx.xxx.xxx.xxx:1433;DatabaseName=xyz
- **Database User/Password**: De um utilizador com permissões de leitura na Base de Dados
- **Audit Type:** GENERAL_AUDIT
- **Trace File Local Folder:** C:\ArcSight
- **Connector Data Folder:** /opt/ArcSight/mssql/winaflshare. Na verdade este caminho e o anterior apontam para uma mesma pasta que conterá os ficheiros de trace a processar. A primeira é utilizada localmente pela Base de Dados para escrever os ficheiros, ao passo que a segunda indica o caminho até eles para serem consumidos. No caso, a pasta winaflshare é um *share* montado localmente no ServerConnector.

4.4.2.7 SmartConnector para Microsoft Windows Event Log

As seguintes configurações foram feitas para instalar um `SmartConnector` de *Syslog*, **em qualquer um dos três `Server Connector`'s**:

- **`SmartConnector`:** Syslog Daemon
- **Network Port:** 6514
- **Ip Address:** ALL
- **Protocol:** UDP

[8] Para que o `connector` apague os ficheiros após o seu processamento

Nativamente a Microsoft não tem forma de enviar os *logs* do seu Event Log via *syslog*. Para isso foi instalado um agente intermediário que lê esse registo de actividade, formata segundo o seu formato próprio e envia via *syslog*. Esse agente denomina-se Snare[9] e o `SmartConnector` de *Syslog* tem embutido um *sub-parser* para o formato do Snare.

4.4.2.8 SmartConnector para Symantec Endpoint Protection

Este `SmartConnector` liga-se à Base de Dados do *manager* da Symantec, extraíndo vários tipos de eventos relacionados com os *antivírus* do cliente. A sua parametrização seguiu os seguintes moldes:

- **`SmartConnector`:** Symantec Endpoint Protection DB
- **JDBC Driver:** com.microsoft.sqlserver.jdbc.SQLServerDriver
- **JDBC Database URL:** jdbc:sqlserver://xxx.xxx.xxx.xxx:50446;DatabaseName=xyz
- **Event Types:** agent-security, agent-traffic, nac-system, nac-traffic, agent-behavior, agent, server, alerts
- **Database User/Password**: De um utilizador com permissões de leitura na Base de Dados

4.4.2.9 SmartConnector para Unix Audit

As seguintes configurações foram feitas para instalar um `SmartConnector` de *Syslog*, **em qualquer um dos três `Server Connector`'s**:

- **`SmartConnector`:** Syslog Daemon
- **Network Port:** 6511
- **Ip Address:** ALL
- **Protocol:** TCP

4.4.3 Criação de conteúdos

Finalizada a instalação de agentes nativos e personalizados, chegou-se a um estado em que já é possível iniciar a monitorização da infra-estrutura. Os elementos de controlo a criar são os comuns em qualquer plataforma SIEM: *Dashboards* que apresentem agregados de informação em tempo real, relatórios mostrando dados de um espaço temporal que, tipicamente, pode ir desde horas até várias semanas e regras de correlação que gerem notificações. Estes conteúdos estão divididos em dois grandes grupos:

[9] `http://www.intersectalliance.com`

- **PCI DSS:** Conteúdos que monitorizam elementos específicos desta norma. Esta descreve que deverão ser auditados todos os eventos que estejam relacionados com dados de cartões de crédito, *antivírus*, acessos à rede, alterações de ficheiros críticos, entre outros.
- **Outros:** Elementos para monitorizar a infra-estrutura sobre diversos temas, tais como actividades de *firewalls*, IDS, servidores Web, Bases de dados, aplicações várias, etc.

4.4.3.1 PCI DSS

Esta norma aplica-se a qualquer entidade que guarde, processe ou transmita dados de cartões de crédito. Agrupa uma série de requisitos que estas organizações devem fazer cumprir, sendo que muitos deles podem ser alcançados com esta solução ArcSight, que permite a geração de relatórios, alertas e outros recursos que visem cobrir os aspectos que a norma trata. São 12 os requisitos base do PCI DSS[9]:

1. Instalar e administrar uma *firewall* que proteja os dados de cartão;
2. Não utilizar *passwords* nem configurações de segurança por omissão nos sistemas utilizados;
3. Proteger dados de cartões armazenados;
4. Cifrar comunicação que contenha dados de cartão;
5. Utilizar *antivírus* e actualizá-los regularmente;
6. Desenvolver e manter aplicações e sistemas seguros;
7. Dar acesso a dados de cartão apenas a utilizadores que necessitem realmente (*need-to-know*);
8. Associar um identificador único para cada utilizador da infra-estrutura;
9. Restringir o acesso físico a dados de cartão;
10. Monitorizar todos os acessos aos recursos da rede e a dados de cartão;
11. Testar regularmente a segurança dos sistemas;
12. Manter uma política que mantenha a segurança da informação;

Com a plataforma SIEM a receber eventos de *firewalls*, antivírus, bases de dados e aplicações que armazenem e utilizem dados de cartão, torna-se simples conceber recursos que suportem a monitorização da actividade, estando tudo centralizado numa única plataforma. Vejam-se exemplos concretos do que está criado para abordar os requisitos mais importantes da norma PCI DSS em que faz sentido utilizar uma plataforma SIEM:

Regras

- **Alteração indevida dos *logs*:** Alertas gerados sempre que exista alteração a ficheiros de *logs* a serem monitorizados pela plataforma;

- **Alteração de ficheiros de sistema:** Alertas gerados sempre que exista alteração de ficheiros de sistema em máquinas monitorizadas pelo OSSEC HIDS;

- **Violações de políticas notadas pelo IDS:** Alertas gerados para notificar a equipa sempre que o Sourcefire (NIDS) gera um evento de *policy*;

Dashboards

- **Alterações indevidas de *logs* a processar:** *Dashboard* para visualizar: Os últimos eventos que denotem alteração de ficheiros de *logs* a monitorizar, e em que máquina isso aconteceu;

- **Eventos OSSEC - Alteração de ficheiros de sistema:** *Dashboard* para visualizar os últimos eventos vindos do OSSEC, bem como um *Top 10* das máquinas que geram mais eventos deste tipo;

- **Alertas Sourcefire:** *Dashboard* que mostra: Os últimos eventos que denotem regras despoletadas neste IDS; *Top 10* de endereços IP e portos como fonte do evento, *Top 10* de endereços IP destino e portos;

Relatórios

- **Todos os acessos a objectos contendo dados de cartão:** Relatório que discrimina acessos de utilizadores a objectos que possam conter dados de cartão, nomeadamente: Acessos a um *file share* específico, todas as tabelas auditadas alojadas no DB2 e algumas tabelas (conhecidas) na base de dados de uma aplicação proprietária; A extrair diariamente;

- **Acções executadas por administradores nas suas máquinas:** Relatório pormenorizado de todas as acções (comandos e processos executados) por parte dos administradores nas suas máquinas; A extrair diariamente;

- **Alterações de ficheiros críticos:** Relatório que disponibiliza uma lista exaustiva de todos os ficheiros de sistema alterados nas máquinas sob a análise do OSSEC; A extrair diariamente.

4.4.3.2 Outros

Porque esta plataforma não está implementada apenas para dar suporte ao cumprimento dos requisitos do PCI DSS, o cliente deseja também cobrir outros aspectos com o intuito de aumentar a segurança da sua infra-estrutura. Outras temáticas como autenticações falhadas, ligações bloqueadas em *firewalls*, acessos desautorizados a ficheiros, entre outras, também são levadas em conta e terão recursos criados adaptados à sua realidade. Exemplos de conteúdos gerados:

Regras

- 3 *logins* sem sucesso na mesma máquina, pelo mesmo utilizador, no espaço de um minuto;
- 15 ligações barradas em *firewalls* do mesmo endereço fonte para o mesmo endereço destino, no espaço de um minuto;
- Vírus detectado mas não removido;
- Número excessivo de ligações a um servidor Web.

Dashboards

- **Para as várias *Firewalls* Check Point:** *Top 10* de: Endereço IP fonte, Endereço IP destino, porto fonte, porto destino. Para ligações negadas: *Top 10* de endereço fonte e destino;
- **Logins falhados:** Máquinas onde são falhados mais *logins*, endereços IP de onde surgem mais *logins* falhados (*logins* remotos);
- **Actividade de antívirus:** Últimos vírus encontrados; Últimos vírus removidos.

Relatórios

- Todas as contas com actividade num período de tempo;
- Criação e remoção de utilizadores;
- Acessos a sistemas por parte de contas supostamente inactivas;
- Alterações feitas a configurações de *Firewalls*.

Um recurso adicional que ainda não foi mencionado são as *listas*. É possível criar listas tabulares para armazenar informação (estática ou dinâmica) por forma a manter qualquer tipo de informação. É possível efectuar *queries* sobre as mesmas, produzir relatórios, etc. Eis alguns exemplos de listas criadas para o cliente:

Listas

- **Lista de utilizadores:** Informação pessoal de todos os utilizadores desta organização, incluindo o seu estado (activo ou inactivo);
- **Máquinas com dados de cartão:** Lista de endereços e hostnames de máquinas que possam lidar com dados de cartão;
- **Administradores de máquinas:** Lista de servidores e respectivos administradores.

Estes e outros recursos foram criados com o constante apoio do cliente, total conhecedor da sua rede. A sua presença nesta fase é de extrema importância para a definição dos tipos de eventos a incluir em cada recurso criado e de que forma devem ser apresentados para análise. Estes conteúdos serviram como base de exemplo para criações futuras de forma autónoma por parte da equipa de monitorização.

4.4.4 Optimização da solução

Na recta final do projecto existem algumas optimizações nos componentes da arquitectura que se devem efectuar, visando um melhor comportamento de cada um deles ao longo do tempo. Estas configurações adicionais podem trazer, dependendo dos casos, um maior nível de segurança nas comunicações, automatização de alguns processos administrativos, diminuição do número de eventos enviados por cada `connector` baixando assim a largura de banda necessária, entre outros. As próximas secções detalham as optimizações mais importantes efectuadas em cada um dos elementos da solução.

4.4.4.1 Manager

Comunicações seguras

Apesar de, nativamente, o `Manager` estar munido de um certificado auto-assinado contendo a sua chave pública, o cliente não deseja utilizá-los. Assim, foi pedido um certificado assinado pela Autoridade de Certificação do cliente para o efeito.

O mesmo foi feito também para a comunicação HTTPS entre o *browser* Web e as interfaces gráficas das várias *appliances* ArcSight.

***Backups* automáticos**

Não existindo formas para criar e arquivar cópias de segurança de configurações e outros elementos de um `Manager`, este mecanismo tem de ser criado separadamente. Estas *backups* são de grande importância pois permitem a recuperação rápida de uma instalação do `Manager` em caso de falha do *hardware*. Posto isto, foi escrito um *script* para o efeito, procedendo da seguinte forma:

1. Extrai os ficheiros de configuração mais importantes, relatórios que eventualmente tenham sido arquivados na máquina do `Manager` e um ficheiro gerado por um outro *script* fornecido pela ArcSight que efectua uma cópia de segurança das tabelas da base de dados de sistema (contendo tabelas de utilizadores, de recursos criados pelos mesmos, entre outros). Esta extracção é feita para um ficheiro *tarball* comprimido (.tar.gz).

2. Efectua uma ligação *sftp* para a máquina remota.

3. Envia o ficheiro resultante de (1), mas não sem antes salvaguardar a *backup* anterior que já residia na máquina remota, para assim se ter a cópia de segurança actual e imediamente anterior.

Este procedimento é efectuado semanalmente e de forma automática, com o auxílio de uma *cron task*[10] criada para o efeito. O `Manager` do ambiente produtivo envia a sua cópia de segurança para o `Manager` do ambiente de qualidade e *vice-versa*. Abaixo pode visualizar-se o *script* que põe em prática este procedimento.

```
#!/bin/sh

backupshost=xxx.xxx.xxx.xxx

backuphome=/home/manbackups
localbackuphome=/opt/arcsight/manager

#export_system_tables
/bin/su - oracle -c "/opt/arcsight/db/bin/arcsight export_system_tables
    arcsight/***********@arcsight"

tar czpf $localbackuphome/manager_prd_backup.tar.gz /opt/arcsight/
   manager/config/jetty/* /opt/arcsight/manager/config/server.
   properties /opt/arcsight/manager/jre/lib/security/cacerts /opt/
   arcsight/manager/i18n/common/* /opt/arcsight/manager/config/
   notification/* /opt/arcsight/manager/reports/* /opt/arcsight/db/
   arcsight.dmp /opt/arcsight/db/system_tables.par /opt/arcsight/
   manager/managerbackup.sh

move="-rm $backuphome/lastbackup/LASTBACKUPmanager_prd_backup.tar.gz\n-
   rename $backuphome/manager_prd_backup.tar.gz $backuphome/lastbackup
   /LASTBACKUPmanager_prd_backup.tar.gz"
echo -e "$move\n-put $localbackuphome/manager_prd_backup.tar.gz
   $backuphome\nbye" >> $localbackuphome/sftpbatch
sftp -b $localbackuphome/sftpbatch manbackups@$backupshost

rm -rf /opt/arcsight/db/arcsight.dmp
rm -rf /opt/arcsight/db/system_tables.par
rm -rf /opt/arcsight/manager/manager_prd_backup.tar.gz
rm -rf /opt/arcsight/manager/sftpbatch
```

[10]Em Unix o utilitário *crontab* permite agendar a execução de *scripts* automaticamente

4.4.4.2 Connectors

Filtragem e Agregação

A largura de banda requirida por um agente é directamente proporcional ao número de eventos que este tem de reencaminhar diariamente. Elevadas taxas de envio podem significar congestionamento na recepção por parte dos destinatários, o que se traduz numa monitorização deficiente já que se estão a receber eventos com atraso. Para uma óptima utilização da largura de banda disponível para tráfego de eventos, os `connectors` têm duas configurações adicionais imprescindíveis nesta fase final de optimização da solução: **Filtragem** e **Agregação**.

Com a primeira opção podem-se ignorar eventos que não sejam úteis para a monitorização da infra-estrutura, significando uma diminuição na taxa de envio de cada `connector`, o que também faz com que a base de dados de eventos dos destinatários não seja sobrecarregada com eventos inúteis.

Com a agregação podemos atingir o mesmo objectivo de forma diferente. Caso existam muitos eventos semelhantes diariamente mas que não sejam passíveis de filtrar, o `connector` pode agregar um número de eventos e enviar apenas um só que represente esse agregado. Por exemplo, seja o caso em que o agente é configurado para agregar eventos com um dado nome se se gerarem 100 desses mesmos eventos em 10 segundos. Caso isso aconteça, o `connector` enviará apenas um e um só evento com a informação de que este agrega 100 eventos semelhantes.

Alguns agentes instalados foram alvo de configuração adicional ao nível da agregação, com níveis mais ou menos elevados dependendo da sua taxa de envio diária. Não foi feito qualquer tipo de filtragem, de acordo com o pedido pelo cliente.

***Backups* automáticos**

Neste caso também não existe mecanismo nativo para efectuar cópias de segurança. Como tal, foi desenvolvido um *script* para o efeito, na mesma filosofia do desenvolvido para as *backups* dos `Manager`'s. Existindo *scripts* em todos os `Server Connector`'s, todas as cópias de segurança são enviadas para o `Server Connector` A, sendo que, por sua vez, este envia para o `Server Connector` B.

Para o efeito, foi criado o seguinte *script*:

```
#!/bin/sh

folders=("aixaudit" "apache" "cisco" "informix" "itim" "mssql" "ossec"
    "snare" "sourcefire" "syslog" "tacacs" "zos" "zosapp")
preprocessors=("/opt/ArcSight/cisco/ciscoppd" "/opt/ArcSight/tacacs/
    tacacspre" "/opt/ArcSight/zos/smfpre")

backupshost="xxx.xxx.xxx.xxx"
connhome="/home/connbackups"
```

```
user=jfcruz

for folder in "${folders[@]}"; do
        name=CONNECTOR$folder\_backup.tar.gz
        filename=/home/$user/$name

        to_tar="/opt/ArcSight/$folder/current"

        if [ "$folder" == "cisco" ]; then
                to_tar="$to_tar ${preprocessors[0]}"
        else
                if [ "$folder" == "tacacs" ]; then
                        to_tar="$to_tar ${preprocessors[1]}"
                else
                        if [ "$folder" == "zos" ]; then
                                to_tar="$to_tar ${preprocessors[2]}"
                        fi
                fi
        fi

        exclusion1="/opt/ArcSight/$folder/current/user/agent/agentdata
           /*"
        exclusion2="/opt/ArcSight/$folder/current/logs/*"
        echo -e "$exclusion1\n$exclusion2" >> exclusions

        sudo tar czpf $filename -X exclusions $to_tar

        move="-rm $connhome/lastbackups/LASTBACKUP$name\n-rename
           $connhome/$name $connhome/lastbackups/LASTBACKUP$name"
        echo -e "$move\n-put $filename $connhome\nbye" >> sftpbatch
        sftp -b sftpbatch connbackups@$backupshost

        rm -rf exclusions
        rm -rf sftpbatch
        rm -rf $filename
done
```

4.4.4.3 Logger

Backups automáticos

Ao contrário do `Manager`, o `Logger` tem já embutida uma opção de efectuar cópias de segurança automaticamente. Basta somente configurar um horário para esta rotina executar e uma máquina remota para se efectuar o envio, sendo que este é feito via SCP (Secure Copy Protocol).

4.4.4.4 Connector Appliance

Backups

Este componente tem apenas mecanismos para efectuar cópias de segurança de forma manual. Foram apenas criadas as condições necessárias na máquina remota para receber as *backups* quando for executado o procedimento.

4.5 Conclusão do capítulo

Este capítulo apresentou o trabalho efectuado no cliente para a implementação da solução SIEM de forma a monitorizar a sua infra-estrutura. Após esta fase de concretização, a plataforma encontra-se a centralizar *logs* de vários tipos: desde registos de sistemas operativos, dispositivos de segurança, *logs* aplicacionais, entre outros. Foram também definidos vários elementos de monitorização com base nestes registos recolhidos, que são usados para efeitos de monitorização. O próximo capítulo apresenta a avaliação a esta plataforma, não só em termos de níveis de recepção de eventos como a nível de desempenho.

Capítulo 5

Avaliação

Neste capítulo são apresentados os resultados da avaliação da solução implementada. Esta apreciação tem em conta dois factores:

- **Recepção de eventos:** utilizando métricas usuais no meio das plataformas SIEM, nomeadamente ao nível de eventos por segundo (EPS), eventos por dia (EPD) e a forma como estes eventos se distribuem pelos vários `connectors`;
- **Desempenho:** mostrando os habituais elementos de análise: consumos de memória RAM e CPU de cada componente da solução;

5.1 Recepção de eventos

A Figura 5.1 apresenta o cenário normal no ambiente implementado durante uma semana, no que concerne ao número de eventos total enviados diariamente. É possível constatar que a plataforma recebe diariamente uma quantidade de eventos na ordem dos dezassete milhões, sendo que muitos destes representam um aglomerado de eventos (de acordo com a política de agregação de cada `connector`), fazendo com que para efeitos de monitorização sejam mais do que estes dezassete milhões de registos. Também é possível aferir que em períodos não laborais (como fins-de-semana e feriados), esta taxa de recepção decresce praticamente em 40%.

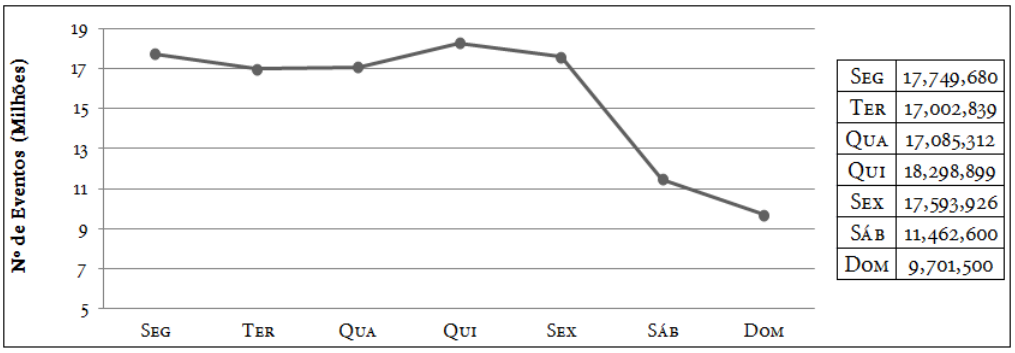


Figura 5.1: Taxa de eventos diária na plataforma

Analisando mais aprofundadamente a distribuição deste número de eventos pelos vários agentes emissores instalados, é facilmente perceptível que esta não é uma distribuição uniforme, havendo `connectors` que enviam constantemente a uma taxa elevada e outros que o fazem muito esporadicamente. A Figura 5.2 reflecte isto mesmo. Nela encontram-se, para cada agente, a taxa de eventos por segundo que lhe é destinada ainda sem qualquer tipo de processamento portanto, ao contrário da figura anterior, aqui (e nas figuras e tabelas seguintes) não são levadas em contas as políticas de agregação de eventos.

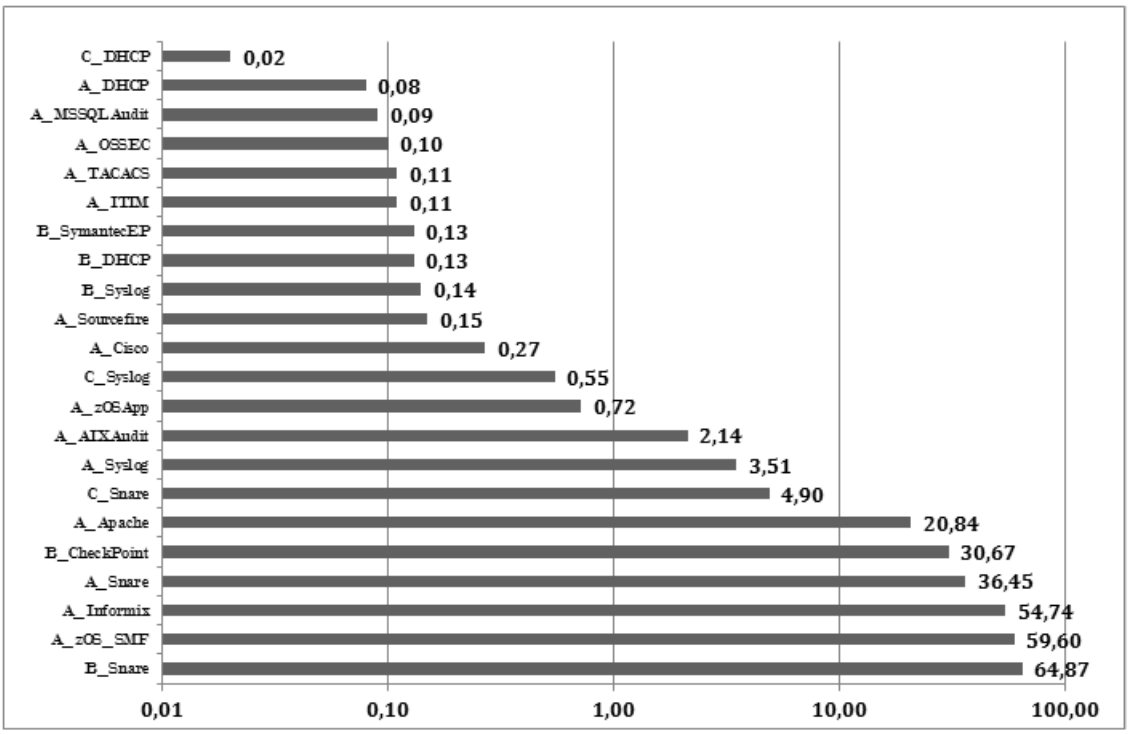


Figura 5.2: Taxa de eventos por segundo (EPS) por connector instalado

Observando a Tabela 5.1 pode-se aferir a distribuição dos activos da infra-estrutura do cliente pelos vários agentes instalados. Analisando-a comparativamente com a Figura 5.2, percebe-se que não existe ligação directa entre o número de máquinas alvo da colecta de *logs* por `connector` e a sua taxa de EPS. Veja-se como exemplo o `connector` para os activos CISCO, que recolhe registos de 95 dispositivos mas tem uma taxa inferior a 0,3 EPS. Por contraposição, no topo da lista dos `connectors` com maior taxa de recepção de eventos estão o `SmartConnector` de Snare (Windows Event Log) que recebe de 10 activos e o `FlexConnector` para IBM z/OS, pouco abaixo e recebendo apenas de 4 *mainframes*.

Assim, para caracterizar convenientemente o ambiente monitorizado pela plataforma ArcSight, a abordagem a seguir não pode ser orientada ao número de fontes por `connector` mas sim orientada à tecnologia, isto é, aos tipos de *logs* que estão a ser centralizados.

Analisando os tipos de *logs* que estão a ser enviados para a solução SIEM, estes podem-se dividir em nove grupos:

1. **Windows:** *Logs* de sistema operativo das máquinas Windows que fazem de Domain Controllers da Active Directory, File Shares, DHCP e outros;
2. **Mainframes:** Máquinas IBM a executar o sistema operativo z/OS (DB2 incluído), onde existem inúmeras bases de dados contendo dados de cartão;
3. **Bases de Dados:** Outras bases de dados (além das que estão nas mainframes sob DB2) contendo dados de cartão, entre outros dados sensíveis;
4. **Security:** *Logs* de *firewalls*, IDS, entre outros activos que preservem a segurança da infra-estrutura;
5. **Web:** *Logs* de Servidores Web;
6. **Unix:** Registos de actividade de variadas máquinas Unix da infra-estrutura;
7. **DHCP:** *Logs* de actividade DHCP;
8. **AV:** *Logs* do gestor de antivírus do cliente;
9. **Outros:** Outros registos variados;

`Connector`	Fontes
A_Cisco	95
A_Syslog	43
A_AIXAudit	21
B_Snare	10
B_CheckPoint	9
A_Informix	8
A_Apache	6
A_Snare	5
A_zOS_SMF	4
C_Snare	3
A_zOSApp	3
A_Sourcefire	3
B_DHCP	2
A_DHCP	2
C_Syslog	1
B_Syslog	1
B_SymantecEP	1
A_ITIM	1
A_TACACS	1
A_OSSEC	1
A_MSSQLAudit	1
C_DHCP	1

Tabela 5.1: Fontes por connector

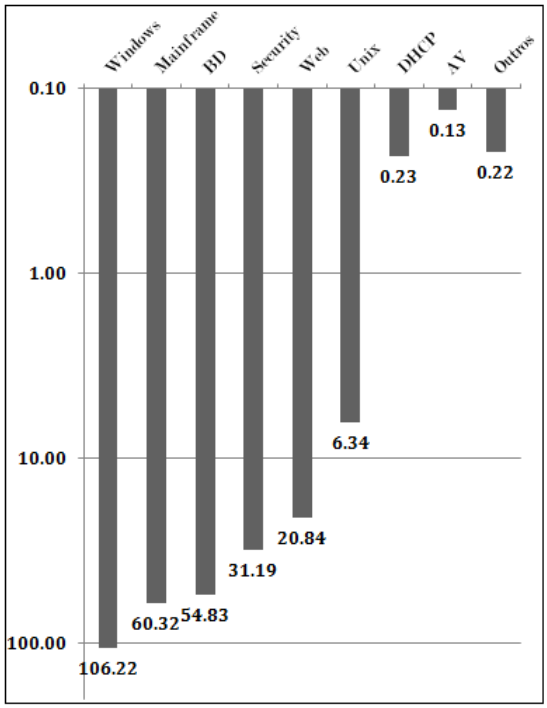


Figura 5.3: EPS por tecnologia

As taxas de recepção de cada um destes tipos de *logs* está visível na Figura 5.3. Maioritariamente, os *logs* recebidos são provenientes das máquinas que actuam como nós integrantes da Active Directory (sendo registados todos os *logins*, *logouts*, entre outros), actividade no sistema operativo z/OS nas mainframes e *logs* de actividade nas bases de dados DB2, outras bases de dados relevantes e, finalmente, activos como *firewalls* ou IDS. Isto deve-se, claramente, ao tipo de cliente em questão e serviços que este presta.

5.2 Desempenho

Quanto ao desempenho da plataforma desenvolvida, existem dois pontos onde é necessário proceder a uma avaliação de forma a perceber se a solução está a responder correctamente perante as taxas de recepção de eventos:

1. **Connectors:** Avaliar o consumo de memória e CPU de cada `connector` em cada um dos três `Server Connector`'s;

2. **`Manager`:** Avaliar o consumo de memória e CPU do `Manager`;

5.2.1 Connectors

Esta avaliação recai sobre todos os `SmartConnectors` e `FlexConnectors` instalados em `Server Connector`'s, excluíndo os 5 que foram instalados localmente (Servidores DHCP). Antes de continuar, importa referir as especificações técnicas de cada `Server Connector`. Todos eles são iguais, e as suas configurações estão ilustradas na Tabela 5.2.

CPU	Intel Xeon E5620 Quad Core 2.4 GHz (1x)
RAM	16GB
Armazenamento	2x146GB + 2x600GB
S.O.	RedHat Enterprise Linux 5.6, x64

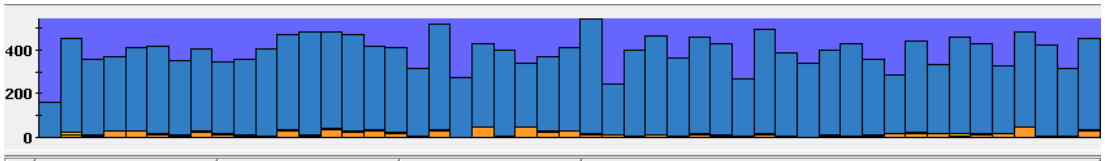
Tabela 5.2: Especificações técnicas de cada Server Connector

Cada `connector` é uma aplicação Java configurada por omissão para ter 256MB de memória RAM dedicada. `Connectors` com taxas de recepção mais elevadas poderão ser configurados para poderem fazer uso de mais memória. O Anexo II contém informação sobre o consumo de memória RAM de cada `connector` num período de duas horas.

Tendo em conta o bom funcionamento da plataforma diariamente (isto é, sem atrasos na comunicação entre `connectors` e `Manager` e `Logger` e sem que os `connectors` sofram de falta de memória RAM), é perceptível que até mesmo o `Server Connector` A (com 13 `connectors`, contra os 4 do servidor do polo B e os 2 do polo C) apresenta um bom consumo de memória, e ainda com margem suficiente para a instalação de mais `connectors`, se necessário. Estando todos os agentes configurados para usufruirem de 256MB de memória excepto um que está configurado para ter 512MB disponíveis, constata-se que existe um consumo máximo de 3584MB, o que representa cerca de 22% da memória disponível no `Server Connector`. Os outros dois servidores estão ainda mais abaixo dessa percentagem de utilização por terem menos `connectors` instalados: 4 agentes instalados no `Server Connector` B - um deles com 512MB de memória dedicada, sendo que os restantes três mantêm-se nos 256MB) perfazendo um total máximo de 1280MB (aproximadamente 8% da memória total da máquina). Já no servidor do polo C, estão 2 `connectors` instalados com 256MB de memória dedicados para cada um, significando apenas um consumo de 512MB no total (aproximadamente 3%).

Estes níveis não só mostram ser bons pela percentagem de memória utilizada face ao disponível como estão a ser suficientes para que todos os eventos sejam reencaminhados o mais rapidamente possível até ao `Manager` e `Logger`. A Figura 5.4 apresenta a diferença não significativa entre o momento em que o evento realmente ocorre (End Time)

vs o momento em que chega ao `connector` (Agent Receipt Time) *vs* o momento em que chega ao `Manager` (Manager Receipt Time). Esta amostra é retirada do `connector` que recolhe os eventos de máquinas Windows do polo B (B_Snare), agente este que é o que mais actividade tem diariamente.



End Time	Agent Receipt Time	Manager Receipt Time 1	Name
2 Ago 2012 15:09:07 BST	2 Ago 2012 15:09:10 BST	2 Ago 2012 15:09:50 BST	The computer attempted to validate the credentials for an account
2 Ago 2012 15:09:07 BST	2 Ago 2012 15:09:10 BST	2 Ago 2012 15:09:50 BST	The computer attempted to validate the credentials for an account
2 Ago 2012 15:09:05 BST	2 Ago 2012 15:09:10 BST	2 Ago 2012 15:09:50 BST	An account was logged off
2 Ago 2012 15:09:05 BST	2 Ago 2012 15:09:44 BST	2 Ago 2012 15:09:50 BST	An account was successfully logged on
2 Ago 2012 15:09:05 BST	2 Ago 2012 15:09:44 BST	2 Ago 2012 15:09:50 BST	A Kerberos service ticket was requested
2 Ago 2012 15:09:35 BST	2 Ago 2012 15:09:44 BST	2 Ago 2012 15:09:50 BST	A Kerberos service ticket was requested
2 Ago 2012 15:09:35 BST	2 Ago 2012 15:09:44 BST	2 Ago 2012 15:09:50 BST	A Kerberos authentication ticket (TGT) was requested
2 Ago 2012 15:09:39 BST	2 Ago 2012 15:09:44 BST	2 Ago 2012 15:09:50 BST	An account was logged off
2 Ago 2012 15:09:01 BST	2 Ago 2012 15:09:09 BST	2 Ago 2012 15:09:50 BST	Scan started on selected drives and folders and all extensions.
2 Ago 2012 15:08:59 BST	2 Ago 2012 15:09:09 BST	2 Ago 2012 15:09:50 BST	An account was logged off
2 Ago 2012 15:08:59 BST	2 Ago 2012 15:09:09 BST	2 Ago 2012 15:09:50 BST	An account was successfully logged on
2 Ago 2012 15:08:58 BST	2 Ago 2012 15:09:09 BST	2 Ago 2012 15:09:50 BST	A Kerberos service ticket was requested
2 Ago 2012 15:08:58 BST	2 Ago 2012 15:09:09 BST	2 Ago 2012 15:09:50 BST	An account was logged off
2 Ago 2012 15:08:58 BST	2 Ago 2012 15:09:09 BST	2 Ago 2012 15:09:50 BST	The computer attempted to validate the credentials for an account
2 Ago 2012 15:08:57 BST	2 Ago 2012 15:09:09 BST	2 Ago 2012 15:09:50 BST	The computer attempted to validate the credentials for an account
2 Ago 2012 15:08:57 BST	2 Ago 2012 15:09:09 BST	2 Ago 2012 15:09:50 BST	An account was logged off
2 Ago 2012 15:08:56 BST	2 Ago 2012 15:09:09 BST	2 Ago 2012 15:09:50 BST	An account was logged off
2 Ago 2012 15:08:56 BST	2 Ago 2012 15:09:09 BST	2 Ago 2012 15:09:50 BST	An account was successfully logged on
2 Ago 2012 15:09:42 BST	2 Ago 2012 15:09:44 BST	2 Ago 2012 15:09:50 BST	An account was logged off
2 Ago 2012 15:09:42 BST	2 Ago 2012 15:09:44 BST	2 Ago 2012 15:09:50 BST	An account was successfully logged on
2 Ago 2012 15:09:42 BST	2 Ago 2012 15:09:44 BST	2 Ago 2012 15:09:50 BST	Special privileges assigned to new logon

Figura 5.4: Tempos de um evento entre os vários componentes

Dado que os `connectors` são concretizados em Java, importa salientar o trabalho efectuado pelo *Garbage Collector* na gestão da memória disponível para cada `connector`. Nos gráficos presentes no anexo é possível visualizar o seu trabalho pelas quedas repentinas constantes dos níveis de utilização de memória, sinal de que objectos já não utilizados foram eliminados e a memória libertada.

Em relação à carga de processamento por parte do CPU de cada `Server Connector`, as Figuras 5.5, 5.6 e 5.7 demonstram que ainda existe uma larga margem para instalação de novos `connectors`, se necessário. Continuando a ser o de A com mais carga de CPU, as figuras demonstram que ainda está muito aquém do seu limite.

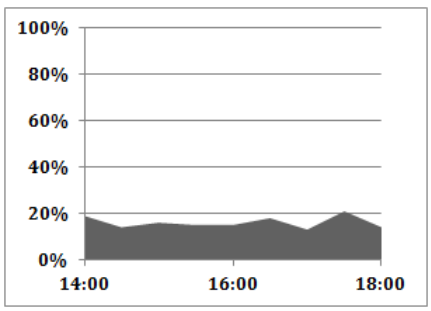


Figura 5.5: Carga de CPU (Server Connector A)

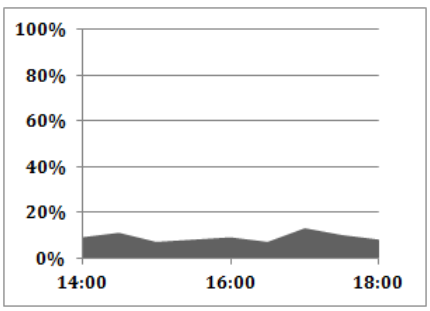


Figura 5.6: Carga de CPU (Server Connector B)

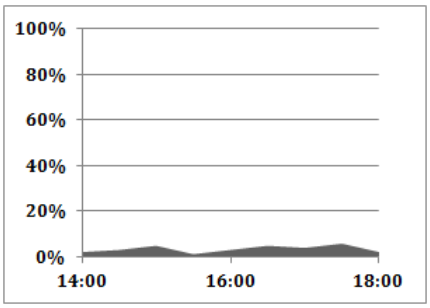


Figura 5.7: Carga de CPU (Server Connector C)

5.2.2 Manager e Logger

Também o `Manager` e o `Logger` apresentam consumos de memória RAM e CPU bons, tendo em conta a taxa de recepção de eventos de cada um e, em particular no caso do `Manager`, todo o trabalho constante que este tem de desempenhar a nível de monitorização e correlação.

As Figuras 5.8, 5.9, 5.10 e 5.11 apresentam os consumos de CPU e memória por parte do `Manager` (em cima) e `Logger` (em baixo). É perceptível que qualquer uma das *appliances* está preparada para a taxa de recepção de eventos actual, podendo essa taxa ser aumentada no futuro sem degradação do serviço.

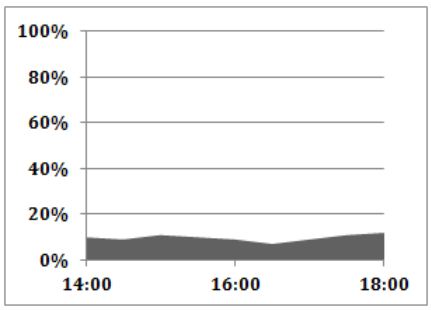


Figura 5.8: Carga de CPU (Manager)

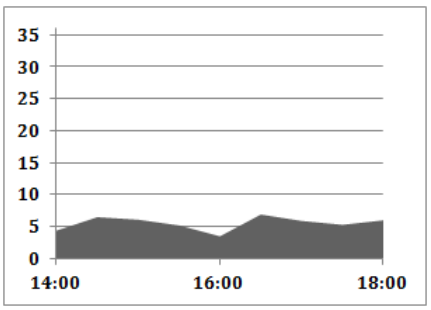


Figura 5.9: Utilização de memória RAM (Manager)

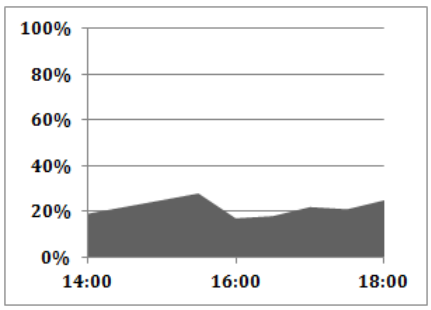


Figura 5.10: Carga de CPU (Logger)

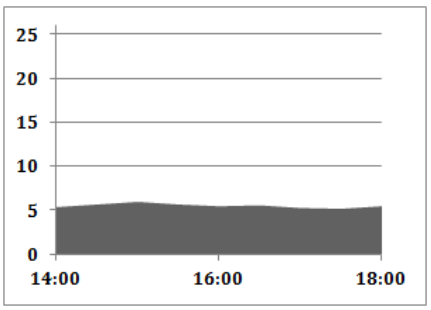


Figura 5.11: Utilização de memória RAM (Logger)

Capítulo 6

Conclusão e Trabalho Futuro

Este relatório documenta o trabalho efectuado para implementar uma plataforma SIEM na infra-estrutura de uma organização. Não obstante este projecto ter sido baseado na solução ArcSight, a implementação de qualquer SIEM numa organização não deverá ser muito diferente do observado aqui: Análise requisitos, instalação das *appliances*, instalação de agentes colectores e desenvolvimento de agentes personalizados, criação de conteúdos e optimização final do sistema.

Apesar da quantidade considerável de eventos que dão entrada na plataforma, esta mostra ser capaz de dar resposta às necessidades do cliente em tempo útil, demonstrando que foi arquitectada uma solução fiável que, caso seja necessário no futuro, ainda pode crescer, não só a nível dos agentes como a nível do número de eventos recebidos.

Hoje em dia, as soluções SIEM surgem cada vez mais como uma necessidade para salvaguardar a segurança da infra-estrutura de qualquer organização. Em particular para estas organizações que lidam diariamente com inúmeros dados sensíveis, uma plataforma deste género tem de ser vista como uma obrigação. No caso específico do cliente em questão, esta plataforma SIEM veio para suprir a falta de monitorização que existia na sua rede e, centralizando e monitorizando estes *logs*, cumprir com alguns dos requisitos do PCI DSS, Visa e Mastercard.

Como trabalho futuro, existem três áreas onde ainda será feito algum trabalho:

- Desenvolvimento de `FlexConnectors`;
- Integração das máquinas do polo D;
- Paralelização com ferramenta externa de *ticketing*;

Na parte de desenvolvimento de `FlexConnectors`, há alguns agentes que ainda terão de ser criados para recolher *logs*, nomeadamente de quatro aplicações proprietárias do cliente. Estas aplicações são de vital importância para o cliente não só pela sua função em si, mas porque algumas delas lidam com dados de cartão. Os agentes para cada uma destas aplicações serão desenvolvidos quando a equipa de desenvolvimento de cada uma

delas terminar o processo de integração de uma estrutura de *logging* que registe a sua actividade.

Terá também de se integrar o sub-sistema o polo D nesta plataforma ArcSight. O processamento de *logs* das máquinas desse local não estava contemplado neste projecto por ser uma sub-organização independente dentro do cliente e ter estado, aquando da execução deste projecto, a ser alvo de auditorias externas por parte da Visa e Mastercard. Como tal, não era desejado fazer integrações de sistemas na infra-estrutura do polo D enquanto esta estava a ser auditada. A instalação física do `Server Connector` deste *site* será feita em breve e a instalação dos `connectors` decorrerá da mesma forma que os já instalados. Não existem `FlexConnectors` a desenvolver para esta sub-organização.

Por fim, há também a possibilidade de existir, paralelamente à arquitectura desenhada, uma ferramenta externa para gestão de *tickets*. A monitorização da infra-estrutura pressupõe que toda e qualquer actividade suspeita detectada por um operador da solução ArcSight registe, num *ticket*, o incidente ocorrido. O objectivo é não só registar estas actividades potencialmente maliciosas mas também o acompanhamento regular destes *tickets* onde o seu responsável deve ir descrevendo todas as conclusões que vai retirando da sua investigação até que, eventualmente, feche este incidente relatando o que foi feito para mitigar a situação.

Abreviaturas

ACL	Access Control List
API	Application Programming Interface
AUP	ArcSight Update Pack
CEF	Common Event Format
CERT	Cyber Emergency Readiness Team
CSIRT	Computer Security Incident Response Team
CSOC	Cyber Security Operation Center
CSV	Comma Separated Values
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DoS	Denial of Service
EPD	Events Per Day
EPS	Events Per Second
FTP	File Transfer Protocol
HIDS	Host-based Intrusion Detection System
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IDS	Intrusion Detection System
IPS	Intrusion Protection System
LDAP	Lightweight Directory Access Protocol
LEA	Log Export API
NCM	Network Configuration Manager
NIDS	Network-based Intrusion Detection System
NSP	Network Synergy Platform
NTP	Network Time Protocol
RAID	Redundant Array of Independent Drives
SCP	Secure Copy Protocol

SDK Software Development Kit
SEM Security Event Management
SFTP Secret File Transfer Protocol
SIEM Security Information and Event Management
SIM Security Information Management
SMS Short Message Service
SNMP Simple Network Management Protocol

TRM Threat Response Manager

VPN Virtual Private Network

Bibliografia

[1] André Zúquete. *Segurança em Redes Informáticas, 3ª edição actualizada e aumentada*. FCA - Editora Informática, 2010.

[2] David R. Miller, Shon Harris, Allen A. Harper, Stephen VanDyke, and Chris Blask. *Security Information and Event Management (SIEM) Implementation*. McGraw-Hill Osborne Media, 2010.

[3] Paulo Veríssimo e Luis Rodrigues. *Distributed Systems for System Architects*. Kluwer Academic Publishers, 2001.

[4] ArcSight. *ArcSight Security Information and Event Management (SIEM) Platform and Integrated Products*.

[5] ArcSight. *FlexConnector Developer's Guide*.

[6] ArcSight. *ArcSight ESM User's Guide: ArcSight ESM v5.0 GA*.

[7] ArcSight. *SmartConnector User's Guide: Topics Applicable to All ArcSight SmartConnectors*.

[8] ArcSight. *ArcSight Logger: Administrator's Guide v5.2*.

[9] LLC PCI Security Standards Council. *PCI DSS Quick Reference Guide, Understanding the Payment Card Industry Data Security Standard version 2.0*. 2010.

Anexos

ANEXO I - FlexConnectors desenvolvidos

CISCO ACE

```
trim.tokens=true

regex=:(\\w{3}\\s*\\d+\\s*\\d{2}:\\d{2}:\\d{2})\\s[A-Z]*:\\s*%ACE-(\\d)
    -(\\d{6}):(.*)

token.count=4

token[0].name=date_time
token[0].type=TimeStamp
token[0].format=MMM dd HH:mm:ss
token[1].name=sev
token[1].type=String
token[2].name=msg_id
token[2].type=String
token[3].name=remaining
token[3].type=String

event.deviceAddress=__regexTokenAsAddress(_SYSLOG_SENDER,"(\\d+\\.\\d
    +\\.\\d+\\.\\d+)")
event.deviceReceiptTime=__useCurrentYear(date_time)
event.deviceSeverity=sev
event.deviceEventClassId=msg_id
event.deviceVendor=__stringConstant("CISCO")
event.deviceProduct=__stringConstant("ACE")

severity.map.low.if.deviceSeverity=6,7
severity.map.medium.if.deviceSeverity=4,5
severity.map.high.if.deviceSeverity=2,3
severity.map.veryhigh.if.deviceSeverity=0,1

submessage.messageid.token=msg_id
submessage.token=remaining

submessage.count=13

submessage[0].pattern.count=1
submessage[0].pattern[0].regex=(.*)
submessage[0].pattern[0].fields=event.name
submessage[0].pattern[0].mappings=$1
```

```
submessage[1].messageid=111008
submessage[1].pattern.count=1
submessage[1].pattern[0].regex=User \\?\\'?(\\S+)\\?\\'? executed the
   \\?\\'?(.+)\\?\\'? command.*
submessage[1].pattern[0].fields=event.sourceUserName,event.fileName,
   event.name
submessage[1].pattern[0].mappings=$1|$2|__stringConstant("User executed
    command")

submessage[2].messageid=199006
submessage[2].pattern.count=1
submessage[2].pattern[0].regex=(Orderly reload started at .+) by (\\S+)
   . Reload reason:(.*)
submessage[2].pattern[0].fields=event.name,event.sourceUserName,event.
   reason
submessage[2].pattern[0].mappings=$1|$2|$3

submessage[3].messageid=251008,251010
submessage[3].pattern.count=1
submessage[3].pattern[0].regex=(Health probe failed) for server (\\d
   +\\.\\d+\\.\\d+\\.\\d+) on port (\\d+),(.*)
submessage[3].pattern[0].fields=event.name,event.destinationAddress,
   event.destinationPort,event.reason
submessage[3].pattern[0].mappings=$1|$2|$3|$4

submessage[4].messageid=442001,442002
submessage[4].pattern.count=1
submessage[4].pattern[0].regex=Health probe (\\S+) detected (\\S+) \\(
   interface (\\S+)\\) in serverfarm (\\S+) changed state to (.*)
submessage[4].pattern[0].fields=event.deviceProcessName,event.
   destinationHostName,event.deviceCustomString1Label,event.
   deviceCustomString1,event.deviceCustomString2Label,event.
   deviceCustomString2,event.name
submessage[4].pattern[0].mappings=$1|$2|__stringConstant("Interface")|
   $3|__stringConstant("Serverfarm")|$4|__concatenate(__stringConstant
   ("Health probe detected a state change to "),$5)

submessage[5].messageid=442004,442005
submessage[5].pattern.count=1
submessage[5].pattern[0].regex=Health probe (\\S+) detected (.+) \\(
   interface (\\S+)\\) changed state to (.*)
submessage[5].pattern[0].fields=event.deviceProcessName,event.
   destinationHostName,event.deviceCustomString1Label,event.
   deviceCustomString1,event.name
submessage[5].pattern[0].mappings=$1|__replaceAll($2," ","-")|
   __stringConstant("Interface")|$3|__concatenate(__stringConstant("
   Health probe detected a state change to "),$4)

submessage[6].messageid=442003
submessage[6].pattern.count=1
submessage[6].pattern[0].regex=Real Server (\\S+) in serverfarm (\\S+)
   changed state to (.*)
submessage[6].pattern[0].fields=event.destinationHostName,event.
   deviceCustomString2Label,event.deviceCustomString2,event.name
submessage[6].pattern[0].mappings=$1|__stringConstant("Serverfarm")|$2|
   __concatenate("A Real Server changed his state to ",$3)
```

```
submessage[7].messageid=442006
submessage[7].pattern.count=1
submessage[7].pattern[0].regex=Real Server (\\S+) changed state to (.*)
submessage[7].pattern[0].fields=event.destinationHostName,event.name
submessage[7].pattern[0].mappings=$1|__concatenate("A Real Server
    changed his state to ",$2)

submessage[8].messageid=442007
submessage[8].pattern.count=1
submessage[8].pattern[0].regex=VIP in class: \\\\'(\\S+)\\\\'(.*)
submessage[8].pattern[0].fields=event.name,event.
    deviceCustomString3Label,event.deviceCustomString3,event.
    eventOutcome
submessage[8].pattern[0].mappings=__stringConstant("VIP state change")|
    __stringConstant("VIP")|$1|$2

submessage[9].messageid=441002
submessage[9].pattern.count=1
submessage[9].pattern[0].regex=Serverfarm \\((\\S+)\\) is now back in
    service in policy_map \\((\\S+)\\) \\W{3} class_map \\((\\S+)\\)
    .(.*)
submessage[9].pattern[0].fields=event.deviceCustomString2Label,event.
    deviceCustomString2,event.deviceCustomString4Label,event.
    deviceCustomString4,event.deviceCustomString5Label,event.
    deviceCustomString5,event.message,event.name
submessage[9].pattern[0].mappings=__stringConstant("Serverfarm")|$1|
    __stringConstant("Policy map")|$2|__stringConstant("Class map")|$3|
    $4|__stringConstant("A Serverfarm is now back in service")

submessage[10].messageid=441003
submessage[10].pattern.count=1
submessage[10].pattern[0].regex=Serverfarm \\((\\S+)\\) failed in
    policy_map \\((\\S+)\\) \\W{3} class_map \\((\\S+)\\) without
    backup.(.*)
submessage[10].pattern[0].fields=event.deviceCustomString2Label,event.
    deviceCustomString2,event.deviceCustomString4Label,event.
    deviceCustomString4,event.deviceCustomString5Label,event.
    deviceCustomString5,event.message,event.name
submessage[10].pattern[0].mappings=__stringConstant("Serverfarm")|$1|
    __stringConstant("Policy map")|$2|__stringConstant("Class map")|$3|
    $4|__stringConstant("A Serverfarm failed without backup")

submessage[11].messageid=251011
submessage[11].pattern.count=1
submessage[11].pattern[0].regex=(ICMP health probe failed) for server
    (\\d+\\.\\d+\\.\\d+\\.\\d+),(.*)
submessage[11].pattern[0].fields=event.name,event.destinationAddress,
    event.message
submessage[11].pattern[0].mappings=$1|$2|$3

submessage[12].messageid=251014
submessage[12].pattern.count=1
submessage[12].pattern[0].regex=(Could not probe server) (\\d+\\.\\d
    +\\.\\d+\\.\\d+) on port (\\d+) for (.*)
```

```
submessage[12].pattern[0].fields=event.name,event.destinationAddress,
    event.destinationPort,event.message
submessage[12].pattern[0].mappings=$1|$2|$3|$4
```

IBM Informix

```
#Flexconnector Informix
#Last Update (08-02-2012)

trim.tokens=true

token.count=9

token[0].name=event_identifier
token[0].type=String
token[1].name=event_time
token[1].type=TimeStamp
token[1].format=yyyy-MM-dd HH:mm:ss.SSS
token[2].name=hostname
token[2].type=String
token[3].name=pid
token[3].type=String
token[4].name=dbservname
token[4].type=String
token[5].name=username
token[5].type=String
token[6].name=errno
token[6].type=String
token[7].name=event_mnem
token[7].type=String
token[8].name=additional_info
token[8].type=String

event.deviceEventCategory=__simpleMap(event_identifier,"ONLN=Dynamic
    Server Event")
event.deviceReceiptTime=event_time
event.sourceHostName=hostname
event.deviceCustomString1Label=__stringConstant("Process ID")
event.deviceCustomString1=pid
event.deviceFacility=dbservname
event.destinationUserName=username
event.eventOutcome=__ifThenElse(errno,__stringConstant("0"),
    __stringConstant("Success"),__concatenate(__stringConstant("Failure
     (ERRNO="),errno,__stringConstant(")")))

event.name=__simpleMap(event_mnem,"ACTB=Access Table","ADCK=Add Chunk
    ","ADLG=Add Transaction Log","ALFR=Alter Fragment","ALIX=Alter
    Index","ALME=Alter Access Method","ALOC=Alter Operator Class","ALOP
    =Alter Optical Cluster","ALTB=Alter Table","BGTX=Begin Transaction
    ","CLDB=Close Database","CMTX=Commit Transaction","CRAG=Create
    Aggregate","CRAM=Create Audit Mask","CRBS=Create Storage Space","
    CRBT=Create Opaque Type","CRCT=Create Cast","CRDB=Create Database
    ","CRDM=Create Domain","CRDS=Create Dbspace","CRDT=Create Distinct
    Type","CRIX=Create Index","CRME=Create Access Method","CROC=Create
    Operator Class","CROP=Create Optical Cluster","CRRL=Create Role","
    CRRT=Create Named Row Type","CRSN=Create Synonym","CRSP=Create SPL
    Routine","CRTB=Create Table","CRTR=Create Trigger","CRVW=Create
    View","DLRW=Delete Row","DNCK=Bring Chunk Off-line","DNDM=Disable
    Disk Mirroring","DRAG=Drop Aggregate","DRAM=Delete Audit Mask","
    DRBS=Drop Storage Space","DRCK=Drop Chunk","DRCT=Drop Cast","DRDB=
```

```
    Drop Database","DRDM=Drop Domain","DRDS=Drop Dbspace","DRIX=Drop
    Index","DRLG=Drop Transaction Log","DRME=Drop Access Method","DROC=
    Drop Operator Class","DROP=Drop Optical Cluster","DRRL=Drop Role","
    DRRT=Drop Named Row Type","DRSN=Drop Synonym","DRSP=Drop SPL
    Routine","DRTB=Drop Table","DRTR=Drop Trigger","DRTY=Drop Type","
    DRVW=Drop View","EXSP=Execute SPL Routine","GRDB=Grant Database
    Access","GRDR=Grant Default Role","GRFR=Grant Fragment Access","
    GRRL=Grant Role","GRTB=Grant Table Access","INRW=Insert Row","LGDB=
    Change Database Log Mode","LKTB=Lock Table","LSAM=List Audit Masks
    ","LSDB=List Databases","MDLG=Modify Transaction Logging","ONAU=
    onaudit","ONBR=onbar","ONCH=oncheck","ONIN=oninit","ONLG=onlog","
    ONLO=onload","ONMN=onmonitor","ONMO=onmode","ONPA=onparams","ONPL=
    onpload","ONSP=onspaces","ONST=onstat","ONTP=ontape","ONUL=onunload
    ","OPDB=Open Database","RDRW=Read Row","RLOP=Release Optical
    Cluster","RLTX=Rollback Transaction","RMCK=Clear Mirrored Chunks","
    RNDB=Rename Database","RNDS=Rename dbspace","RNTC=Rename Table/
    Column","RSOP=Reserve Optical Cluster","RVDB=Revoke Database Access
    ","RVFR=Revoke Fragment Access","RVRL=Revoke Role","RVTB=Revoke
    Table Access","SCSP=SYSTEM Command - SPL Routine","SEOP=SET
    ENVIRONMENT OPTCOMPIND","STCN=Set Constraint","STDF=Set Debug File
    ","STDP=Set Database Password","STDS=Set Dataskip","STEX=Set
    Explain","STIL=Set Isolation Level","STLM=Set Lock Mode","STOM=Set
    Object Mode","STOP=Stop Statement","STPR=Set Pdqpriority","STRL=Set
     Role","STRS=Set Resident","STRT=Start Statement","STSA=Set Session
     Authorization","STSC=Set Statement Cache","STSN=Start New Session
    ","STTX=Set Transaction Mode","TMOP=Time Optical Cluster","ULTB=
    Unlock Table","UPAM=Update Audit Mask","UPCK=Bring Chunk Online","
    UPDM=Enable Disk Mirroring","UPRW=Update Row","USSP=Update
    Statistics - SPL Routine","USTB=Update Statistics - Table")

event.deviceSeverity=__simpleMap(event_mnem,"ACTB=2","ADCK=0","ADLG
    =2","ALFR=1","ALIX=1","ALME=1","ALOC=1","ALOP=1","ALTB=1","BGTX
    =0","CLDB=2","CMTX=1","CRAG=0","CRAM=3","CRBS=0","CRBT=1","CRCT
    =1","CRDB=1","CRDM=1","CRDS=1","CRDT=1","CRIX=1","CRME=1","CROC
    =1","CROP=1","CRRL=2","CRRT=1","CRSN=1","CRSP=1","CRTB=1","CRTR
    =1","CRVW=1","DLRW=1","DNCK=0","DNDM=2","DRAG=1","DRAM=3","DRBS
    =0","DRCK=0","DRCT=0","DRDB=1","DRDM=0","DRDS=0","DRIX=0","DRLG
    =3","DRME=0","DROC=0","DROP=0","DRRL=0","DRRT=0","DRSN=0","DRSP
    =0","DRTB=0","DRTR=0","DRTY=0","DRVW=0","EXSP=1","GRDB=2","GRDR
    =2","GRFR=2","GRRL=2","GRTB=2","INRW=1","LGDB=3","LKTB=0","LSAM
    =0","LSDB=0","MDLG=3","ONAU=2","ONBR=0","ONCH=0","ONIN=2","ONLG
    =2","ONLO=0","ONMN=0","ONMO=0","ONPA=0","ONPL=0","ONSP=0","ONST
    =0","ONTP=0","ONUL=0","OPDB=2","RDRW=0","RLOP=1","RLTX=1","RMCK
    =0","RNDB=0","RNDS=0","RNTC=0","RSOP=0","RVDB=2","RVFR=0","RVRL
    =2","RVTB=2","SCSP=2","SEOP=2","STCN=1","STDF=1","STDP=3","STDS
    =1","STEX=1","STIL=1","STLM=1","STOM=1","STOP=1","STPR=1","STRL
    =2","STRS=1","STRT=2","STSA=1","STSC=1","STSN=2","STTX=1","TMOP
    =0","ULTB=0","UPAM=2","UPCK=0","UPDM=0","UPRW=0","USSP=0","USTB=0")


event.flexString1Label=__stringConstant("Additional info")
event.flexString1=additional_info

event.deviceProduct=__stringConstant("Informix")
event.deviceVendor=__stringConstant("IBM")
```

```
severity.map.low.if.deviceSeverity=0
severity.map.medium.if.deviceSeverity=1
severity.map.high.if.deviceSeverity=2
severity.map.veryhigh.if.deviceSeverity=3

regex=(\\w+)\\|(\\d+-\\d+-\\d+\\s+\\d+:\\d+:\\d+\\.\\d+)\\|(.+)\\|(-?\\
    d+)\\|(.+)\\|(.+)\\|(-?\\d+):(\\w+):?(.*)
```

OSSEC HIDS

```
trim.tokens=true
do.unparsed.events=true

regex=(.+):\\s*Alert Level:\\s*(\\d+);\\s*Rule:\\s*(\\d+)\\s*\\-(.*)

token.count=4

token[0].name=module
token[0].type=String
token[1].name=alert_level
token[1].type=String
token[2].name=rule_id
token[2].type=String
token[3].name=remaining
token[3].type=String


event.deviceCustomString1Label=__stringConstant("Module")
event.deviceCustomString1=module
event.deviceCustomString2Label=__stringConstant("Facility")
event.deviceCustomString2=_SYSLOG_FACILITY
event.deviceReceiptTime=_SYSLOG_TIMESTAMP
event.deviceSeverity=alert_level
event.deviceEventClassId=rule_id
event.deviceHostName=_SYSLOG_SENDER
event.deviceProduct=__stringConstant("OSSEC")
event.deviceVendor=__stringConstant("OSSEC")


severity.map.veryhigh.if.deviceSeverity=8..99
severity.map.high.if.deviceSeverity=5,6,7
severity.map.medium.if.deviceSeverity=2,3,4
severity.map.low.if.deviceSeverity=0,1

submessage.messageid.token=rule_id
submessage.token=remaining

submessage.count=7

submessage[0].pattern.count=1
submessage[0].pattern[0].regex=(.+);\\s*Location:\\s*\\((\\S+)\\)\\s
    *(\\d{1,3}.\\d{1,3}.\\d{1,3}.\\d{1,3})\\->(\\S+);\\s*(.*)
submessage[0].pattern[0].fields=event.name,event.destinationHostName,
    event.destinationAddress,event.fileName,event.message

submessage[1].messageid=502
submessage[1].pattern.count=1
submessage[1].pattern[0].regex=(.+);\\s*Location:\\s*.+\\->(.+);(.+)
submessage[1].pattern[0].fields=event.name,event.destinationProcessName
    ,event.message

submessage[2].messageid=504
submessage[2].pattern.count=1
submessage[2].pattern[0].regex=(.+);\\s*Location:\\s*.+\\->(.+);(.+)
```

```
submessage[2].pattern[0].fields=event.name,event.destinationProcessName
    ,event.message

submessage[3].messageid=5402
submessage[3].pattern.count=1
submessage[3].pattern[0].regex=(.+);\\s*Location:\\s*\\((\\S+)\\)\\s
    *(\\d{1,3}.\\d{1,3}.\\d{1,3}.\\d{1,3})\\->(\\S+);\\s*user:\\s*(\\S
    +);(.*)
submessage[3].pattern[0].fields=event.name,event.destinationHostName,
    event.destinationAddress,event.fileName,event.sourceUserName,event.
    message

submessage[4].messageid=5501
submessage[4].pattern.count=1
submessage[4].pattern[0].regex=(.+);\\s*Location:\\s*\\((\\S+)\\)\\s
    *(\\d{1,3}.\\d{1,3}.\\d{1,3}.\\d{1,3})\\->(\\S+);\\s*user:\\s*(\\S
    +);(.*)
submessage[4].pattern[0].fields=event.name,event.destinationHostName,
    event.destinationAddress,event.fileName,event.sourceUserName,event.
    message

submessage[5].messageid=5502
submessage[5].pattern.count=1
submessage[5].pattern[0].regex=(.+);\\s*Location:\\s*\\((\\S+)\\)\\s
    *(\\d{1,3}.\\d{1,3}.\\d{1,3}.\\d{1,3})\\->(\\S+);\\s*user:(\\s*\\S
    +);(.*)
submessage[5].pattern[0].fields=event.name,event.destinationHostName,
    event.destinationAddress,event.fileName,event.sourceUserName,event.
    message

submessage[6].messageid=5715
submessage[6].pattern.count=1
submessage[6].pattern[0].regex=(.+);\\s*Location:\\s*\\((\\S+)\\)\\s
    *(\\d{1,3}.\\d{1,3}.\\d{1,3}.\\d{1,3})\\->(\\S+);\\s*srcip:\\s*(\\d
    {1,3}.\\d{1,3}.\\d{1,3}.\\d{1,3});\\s*user:\\s*(\\S+);(.*)
submessage[6].pattern[0].fields=event.name,event.destinationHostName,
    event.destinationAddress,event.fileName,event.sourceAddress,event.
    sourceUserName,event.message
```

Sourcefire NIDS

Neste caso foram desenvolvidos quatro *parsers* diferentes, para quatro formatos distintos.

```
trim.tokens=true
do.unparsed.events=true

regex=(.*): \\[IDS\\] (defensecenter|sensor1|sensor2|sensor3):(.*)@(.*)
    ,(.*),(.*)

token.count=6

token[0].name=module
token[0].type=String
token[1].name=facility
token[1].type=String
token[2].name=user
token[2].type=String
token[3].name=from
token[3].type=String
token[4].name=path
token[4].type=String
token[5].name=action
token[5].type=String

event.deviceReceiptTime=_SYSLOG_TIMESTAMP
event.deviceHostName=__ifThenElse(_SYSLOG_SENDER,__stringConstant("
    defensecenter"),__stringConstant("defensecenter.ips"),
    _SYSLOG_SENDER)
event.deviceCustomString1Label=__stringConstant("Module")
event.deviceCustomString1=module
event.deviceCustomString2Label=__stringConstant("Facility")
event.deviceCustomString2=_SYSLOG_FACILITY
event.deviceFacility=facility
event.destinationUserName=user
event.deviceCustomString3Label=__stringConstant("Access From")
event.deviceCustomString3=from
event.name=action
event.message=path

event.deviceEventCategory=__stringConstant("Audit Event")
event.deviceVendor=__stringConstant("Sourcefire")
event.deviceProduct=__stringConstant("Sourcefire")

event.deviceSeverity=__ifThenElse(path,__stringConstant("SEU Install"),
    __stringConstant("2"),__ifThenElse(module,__stringConstant("
    ActionQueueScrape.pl"),__stringConstant("2"),__ifThenElse(module,
    __stringConstant("install-seu.pl"),__stringConstant("2"),
    __simpleMap(action,"Apply=2","Copy=1","Copy All=1","Load=2","Login
    Failed=3","Login Success=1","Logout Success=1","Page View=0","
    Reviewed=1","Reviewed All=1","Save=2","save=2","View=0","Save As
    =2","Form Change=1","Add=2","Edit=2","Cancel=0","Delete=3","Submit
    =2","Update Install=2","Upload=2","Upload Package=2","Change=2","
    Unreview=1","View All=0","Back=0","Rule Edit=2","Update Rule=2","
    Add Event Clause 1=2","Event Property Changed 1 7=2"))))
```

```
severity.map.veryhigh.if.deviceSeverity=3
severity.map.high.if.deviceSeverity=2
severity.map.medium.if.deviceSeverity=1
severity.map.low.if.deviceSeverity=0
```

```
trim.tokens=true
do.unparsed.events=true

regex=(.+):\\s+(.+):\\s+(.+)/(.+) at (\\w{3}\\s+\\w{3}\\s+\\d+\\s+\\d
   {2}:\\d{2}:\\d{2}\\s+\\d{4}\\s+.+):\\s+\\[(\\d+:\\d+:\\d+)\\]\\s
   +(.+)\\s+\\[Impact:(.+)\\]\\s+From\\s"(\\S+)" at (\\w{3}\\s+\\w
   {3}\\s+\\d+\\s+\\d{2}:\\d{2}:\\d{2}\\s+\\d{4}\\s+.+)\\s+\\[
   Classification:(.*)\\]\\s+\\[Priority:\\s+(\\d+)\\]\\s+\\{(\\w+)
   \\}\\s*([0-9\\.]*):?(\\d*)\\W{0,2}([0-9\\.]*):?(\\d*)

token.count=17

token[0].name=module
token[0].type=String
token[1].name=event_type
token[1].type=String
token[2].name=rule_type
token[2].type=String
token[3].name=policy
token[3].type=String
token[4].name=detect_time
token[4].type=TimeStamp
token[4].format=EEE MMM d HH:mm:ss yyyy z
token[5].name=snort_id
token[5].type=String
token[6].name=rule_name
token[6].type=String
token[7].name=impact
token[7].type=String
token[8].name=detection_engine
token[8].type=String
token[9].name=event_time
token[9].type=TimeStamp
token[9].format=EEE MMM d HH:mm:ss yyyy z
token[10].name=classification
token[10].type=String
token[11].name=priority
token[11].type=String
token[12].name=protocol
token[12].type=String
token[13].name=s_ip
token[13].type=IPAddress
token[14].name=s_port
token[14].type=Integer
token[15].name=d_ip
token[15].type=IPAddress
token[16].name=d_port
token[16].type=Integer

event.deviceHostName=__ifThenElse(_SYSLOG_SENDER,__stringConstant("
   defensecenter"),__stringConstant("defensecenter.ips"),
   _SYSLOG_SENDER)
event.deviceCustomString1Label=__stringConstant("Module")
event.deviceCustomString1=module
event.deviceCustomString2Label=__stringConstant("Facility")
```

```
event.deviceCustomString2=_SYSLOG_FACILITY
event.deviceEventCategory=__concatenate(event_type,__stringConstant
    (":"),classification)
event.message=__concatenate(__stringConstant("Rule Type:"),rule_type,
    __stringConstant("; Policy:"),policy)
event.deviceReceiptTime=detect_time
event.deviceEventClassId=snort_id
event.name=rule_name
event.eventOutcome=impact
event.deviceFacility=detection_engine
event.endTime=event_time
event.deviceSeverity=priority
event.transportProtocol=protocol
event.sourceAddress=s_ip
event.sourcePort=s_port
event.destinationAddress=d_ip
event.destinationPort=d_port

event.deviceVendor=__stringConstant("Sourcefire")
event.deviceProduct=__stringConstant("Sourcefire")

severity.map.veryhigh.if.deviceSeverity=0,1
severity.map.high.if.deviceSeverity=2
severity.map.medium.if.deviceSeverity=3
severity.map.low.if.deviceSeverity=4,5,6
```

```
trim.tokens=true
do.unparsed.events=true

regex=(\\w+):\\s+\\W{3}\\s+(.+)\\sFrom\\s+(.+)\\s+at\\s+(\\w{3}\\s+\\w
   {3}\\s+\\d+\\s+\\d{2}:\\d{2}:\\d{2}\\s+\\d{4}\\s+.+)\\s+\\W{3}\\s
   +(.*)

token.count=5

token[0].name=module
token[0].type=String
token[1].name=descr
token[1].type=String
token[2].name=from
token[2].type=String
token[3].name=date
token[3].type=TimeStamp
token[3].format=EEE MMM d HH:mm:ss yyyy z
token[4].name=add_info
token[4].type=String

event.deviceHostName=__ifThenElse(_SYSLOG_SENDER,__stringConstant("
   defensecenter"),__stringConstant("defensecenter.ips"),
   _SYSLOG_SENDER)
event.deviceCustomString1Label=__stringConstant("Module")
event.deviceCustomString1=module
event.deviceCustomString2Label=__stringConstant("Facility")
event.deviceCustomString2=_SYSLOG_FACILITY
event.name=descr
event.deviceFacility=from
event.deviceReceiptTime=date
event.sourceAddress=ip
event.message=add_info
event.deviceVendor=__stringConstant("Sourcefire")
event.deviceProduct=__stringConstant("Sourcefire")

event.deviceEventCategory=__stringConstant("RNA Event")

event.deviceSeverity=__stringConstant("0")

severity.map.low.if.deviceSeverity=0
```

```
trim.tokens=true
do.unparsed.events=true

regex=(\\w+):\\s+HMNOTIFY:\\s+(.+)\\s+\\(Sensor\\s(.+)\\):\\s+Severity
   :\\s+(\\w+):(.*)

token.count=5

token[0].name=module
token[0].type=String
token[1].name=notif
token[1].type=String
token[2].name=facility
token[2].type=String
token[3].name=sev
token[3].type=String
token[4].name=add_info
token[4].type=String

event.deviceHostName=__ifThenElse(_SYSLOG_SENDER,__stringConstant("
   defensecenter"),__stringConstant("defensecenter.ips"),
   _SYSLOG_SENDER)
event.deviceReceiptTime=_SYSLOG_TIMESTAMP
event.deviceCustomString1Label=__stringConstant("Module")
event.deviceCustomString1=module
event.deviceCustomString2Label=__stringConstant("Facility")
event.deviceCustomString2=_SYSLOG_FACILITY
event.name=__stringConstant("Hardware Monitor Notification")
event.message=__concatenate(notif,__stringConstant(":"),add_info)
event.deviceFacility=facility
event.deviceSeverity=sev

event.deviceEventCategory=__stringConstant("Health Monitor Event")

event.deviceVendor=__stringConstant("Sourcefire")
event.deviceProduct=__stringConstant("Sourcefire")

severity.map.low.if.deviceSeverity=normal,recovery
severity.map.medium.if.deviceSeverity=error
severity.map.high.if.deviceSeverity=warning
severity.map.veryhigh.if.deviceSeverity=critical
```

CISCO TACACS+

```
token.count=14

token[0].name=eventid
token[0].type=String

token[1].name=aaa_serv
token[1].type=String

token[2].name=user_name
token[2].type=String

token[3].name=real_name
token[3].type=String

token[4].name=nas_ipaddr
token[4].type=IPAddress

token[5].name=access_dev
token[5].type=String

token[6].name=date_time
token[6].type=TimeStamp
token[6].format=yyyy-MM-dd','HH:mm:ss

token[7].name=acct_flags
token[7].type=String

token[8].name=cmd
token[8].type=String

token[9].name=el_time
token[9].type=Integer

token[10].name=b_in
token[10].type=Integer

token[11].name=b_out
token[11].type=Integer

token[12].name=analyzed
token[12].type=Integer

token[13].name=sev
token[13].type=String

event.externalId=eventid
event.deviceHostName=aaa_serv
event.deviceAction=__ifThenElse(acct_flags,__stringConstant("start"),
    __stringConstant("Login"),__ifThenElse(acct_flags,__stringConstant
    ("reject"),__stringConstant("Denied"),__ifThenElse(acct_flags,
    __stringConstant("stop"),__ifThenElse(cmd,__stringConstant(""),
    __stringConstant("Logout"),__stringConstant("Input")),
    __stringConstant(""))))
event.destinationUserName=user_name
```

```
event.destinationUserId=real_name
event.destinationAddress=nas_ipaddr
event.deviceCustomString1=access_dev
event.deviceCustomString1Label=__stringConstant("Access Device")
event.deviceReceiptTime=date_time
event.deviceCustomString2=acct_flags
event.deviceCustomString2Label=__stringConstant("Acct Flags")
event.name=__ifThenElse(cmd,__stringConstant(""),__ifThenElse(
    acct_flags,__stringConstant("reject"),__stringConstant("FAILED
    LOGIN"),__ifThenElse(acct_flags,__stringConstant("start"),
    __stringConstant("SUCCESSFUL LOGIN"),__ifThenElse(acct_flags,
    __stringConstant("stop"),__stringConstant("LOGOUT"),
    __stringConstant("")))),cmd)
event.deviceCustomNumber1=el_time
event.deviceCustomNumber1Label=__stringConstant("Elapsed Time")
event.bytesIn=b_in
event.bytesOut=b_out
event.deviceCustomNumber2=analyzed
event.deviceCustomNumber2Label=__stringConstant("Analyzed")

event.deviceVendor=__stringConstant("Cisco")
event.deviceProduct=__stringConstant("TACACS+")

event.deviceSeverity=sev


severity.map.low.if.deviceSeverity=0
severity.map.medium.if.deviceSeverity=1
severity.map.high.if.deviceSeverity=2
severity.map.veryhigh.if.deviceSeverity=3

regex=(\\d+),(.*),(.*),(.*),(\\d{1,3}.\\d{1,3}.\\d{1,3}.\\d{1,3}),(.*)
    ,(\\d{4}-\\d{2}-\\d{2},\\d{2}:\\d{2}:\\d{2}),(start|stop|reject)
    ,(.*),(\\d+),(\\d+),(\\d+),(\\d+),(\\d+)
```

IBM z/OS / IBM DB2

```
trim.tokens=true

regex=(\\d+),(\\d+),(\\d+),(.*),(\\d+):(.*),(\\d{2}:\\d{2}:\\d{2}.\\d+)
    \\s+(GMT[+\\-]\\d{2}:\\d{2}),(\\d{4} \\d+),([A-Z]{4}),(.*)

token.count=11
token[0].name=ev_id
token[0].type=String
token[1].name=record_length
token[1].type=Integer
token[2].name=segment_descriptor
token[2].type=Integer
token[3].name=sys_version
token[3].type=String
token[4].name=record_type
token[4].type=String
token[5].name=record_type_descr
token[5].type=String
token[6].name=time
token[6].type=String
token[7].name=timezone
token[7].type=String
token[8].name=date
token[8].type=String
token[9].name=sys_id
token[9].type=String
token[10].name=remaining
token[10].type=String

event.devicePayloadId=ev_id
event.deviceReceiptTime=__createSafeLocalTimeStamp(__concatenate(time,"
     ",timezone,",",date),"HH:mm:ss.SSS z','yyyy' 'D")
event.deviceTimeZone=__concatenate(__stringConstant("GMT"),__regexToken
    (timezone,"(\\D\\d{2}):\\d{2}"))
event.deviceEventClassId=__concatenate(record_type,__stringConstant
    (":"),record_type_descr)
event.deviceVendor=__stringConstant("IBM")
event.deviceProduct=__stringConstant("z/OS")
event.deviceVersion=sys_version
event.deviceHostName=sys_id
event.deviceCustomString5Label=__stringConstant("Instance")
event.deviceCustomString5=__simpleMap(sys_id,"MVSA=PT","MVSB=PT","ROMC=
    PL/RO/AO","ROMB=PL/RO/AO")
event.deviceCustomString6Label=__stringConstant("Environment")
event.deviceCustomString6=__simpleMap(sys_id,"MVSA=DSV/CER/SPP","MVSB=
    PRD","ROMC=SPP","ROMB=PRD")

submessage.messageid.token=record_type
submessage.token=remaining

submessage.count=4

submessage[0].messageid=2
submessage[0].pattern.count=1
```

```
submessage[0].pattern[0].regex=(.*)
submessage[0].pattern[0].fields=event.name,event.deviceSeverity
submessage[0].pattern[0].mappings=$1|__stringConstant("0")

submessage[1].messageid=3
submessage[1].pattern.count=1

submessage[1].pattern[0].regex=(.*)
submessage[1].pattern[0].fields=event.name,event.deviceSeverity
submessage[1].pattern[0].mappings=$1|__stringConstant("0")

submessage[2].messageid=80
submessage[2].pattern.count=1

submessage[2].pattern[0].regex=(.*),(\\d+),(\\d+),(.*),(.*),(\\d+),(\\d
   +),(.{8}),(.{8}),(\\d+),(.{8}),(.*),(.*),(.*,.*),(.*),0x(.{2})
   ,(.{8}),(.*),(.*),(\\d+),(\\d+),(.{8}),0x(.{2}),([0-9a-f]*),(.*)
submessage[2].pattern[0].fields=event.deviceEventCategory,event.
   destinationProcessName,event.destinationUserName,event.
   flexString1Label,event.flexString1,event.reason,event.
   deviceCustomString3Label,event.deviceCustomString3,event.
   deviceCustomNumber1Label,event.deviceCustomNumber1,event.
   deviceCustomNumber2Label,event.deviceCustomNumber2,event.
   deviceCustomString2Label,event.deviceCustomString2,event.
   deviceCustomString4Label,event.deviceCustomString4,event.
   deviceCustomString1Label,event.deviceCustomString1,event.
   deviceCustomDate1Label,event.deviceCustomDate1,event.
   destinationUserId,event.deviceFacility,event.
   destinationUserPrivileges,event.message,event.flexString2Label,
   event.flexString2,event.deviceSeverity
submessage[2].pattern[0].mappings=$1|$13|$4|__stringConstant("
   Hexadecimal section data")|$24|__concatenate($9," ",$17)|
   __stringConstant("Terminal ID")|$12|__stringConstant("Event Code")|
   $2|__stringConstant("Event Code Qualifier")|$3|__stringConstant("
   User Group")|$5|__stringConstant("Authority Used")|__concatenate($8
   ," ",$22)|__stringConstant("Error flag")|$11|__stringConstant("Time
    JOB Recognized")|__createOptionalTimeStampFromString(__ifThenElse(
   $14,__stringConstant(","),null,$14),__stringConstant("HH:mm:ss.SSS
   z',’yyyy' 'D"))|$15|$18|$19|__ifThenElse($24,__regexToken($25
   ,".*,(.*)"),__stringConstant(""),__regexToken($25,".*,(.*)"))|
   __stringConstant("Data Type")|__regexToken($25,"(.*),.*")|
   __simpleMap($1,"Regular=0","User is not defined to RACF=1","Warning
   =2","Violation=3")

submessage[3].messageid=102
submessage[3].pattern.count=2

submessage[3].pattern[0].regex=(\\w{4}),.+,(RMID:\\d+,IFCID:\\d+),.+,
   Correlation Header,.+,AID:(\\w*),CCV:(\\w*),CCN:(\\w*),CPLAN:(\\w*)
   ,COPID:(\\w*),.+,EUID:(\\w*),.+,EUWN:(\\w*),.+\\|(.*)
submessage[3].pattern[0].fields=event.deviceFacility,event.
   deviceEventCategory,event.deviceCustomString4Label,event.
   deviceCustomString4,event.deviceCustomString2Label,event.
   deviceCustomString2,event.deviceCustomString1Label,event.
   deviceCustomString1,event.flexString2Label,event.flexString2,event.
```

```
    destinationUserName,event.destinationUserId,event.
    deviceCustomString3Label,event.deviceCustomString3,event.
    requestContext,event.name,event.deviceSeverity
submessage[3].pattern[0].mappings=$1|$2|__stringConstant("Authorization
     ID")|$3|__stringConstant("Correlation ID")|$4|__stringConstant("
    Connection name")|$5|__stringConstant("Plan name")|$6|$7|$8|
    __stringConstant("Workstation name")|$9|$10|__ifThenElse(
    __simpleMap(__regexToken($2,".*IFCID:(\\d+)"),"23=Utility start
    information","24=Utility object or phase change","25=Utility end
    information","55=Issuance of the set current SQLID command","83=
    Ending of and identify request","105=Association of an internal
    DBID and OBID to the actual data base and table space of index
    referenced","140=Authorization failure","141=Explicit grant or
    revoke","142=Create, Alter or Drop against tables that are audited
    or multilevel security at row level","143=First attempted change (
    write) of an audited object within a unit of recovery","144=First
    attempted access (read) of an audited object within a unit of
    recovery","145=Audit log record in DML Session"),null,
    __stringConstant("DB2 Record with unprocessed IFCID"),__simpleMap(
    __regexToken($2,".*IFCID:(\\d+)"),"23=Utility start information
    ","24=Utility object or phase change","25=Utility end information
    ","55=Issuance of the set current SQLID command","83=Ending of and
    identify request","105=Association of an internal DBID and OBID to
    the actual data base and table space of index referenced","140=
    Authorization failure","141=Explicit grant or revoke","142=Create,
    Alter or Drop against tables that are audited or multilevel
    security at row level","143=First attempted change (write) of an
    audited object within a unit of recovery","144=First attempted
    access (read) of an audited object within a unit of recovery","145=
    Audit log record in DML Session"))|__simpleMap(__regexToken($2,".*
    IFCID:(\\d+)"),"23=0","24=0","25=0","55=2","83=1","105=0",
    "140=3","141=2","142=3","143=1","144=1","145=2")

submessage[3].pattern[1].regex=(\\w{4}),.+,(RMID:\\d+,IFCID:\\d+)
    ,.+\\|(.*)
submessage[3].pattern[1].fields=event.deviceFacility,event.
    deviceEventCategory,event.requestContext,event.name,event.
    deviceSeverity
submessage[3].pattern[1].mappings=$1|$2|$3|__ifThenElse(__simpleMap(
    __regexToken($2,".*IFCID:(\\d+)"),"23=Utility start information
    ","24=Utility object or phase change","25=Utility end information
    ","55=Issuance of the set current SQLID command","83=Ending of and
    identify request","105=Association of an internal DBID and OBID to
    the actual data base and table space of index referenced","140=
    Authorization failure","141=Explicit grant or revoke","142=Create,
    Alter or Drop against tables that are audited or multilevel
    security at row level","143=First attempted change (write) of an
    audited object within a unit of recovery","144=First attempted
    access (read) of an audited object within a unit of recovery","145=
    Audit log record in DML Session"),null,__stringConstant("DB2 Record
     with unprocessed IFCID"),__simpleMap(__regexToken($2,".*IFCID:(\\d
    +)"),"23=Utility start information","24=Utility object or phase
    change","25=Utility end information","55=Issuance of the set
    current SQLID command","83=Ending of and identify request","105=
    Association of an internal DBID and OBID to the actual data base
    and table space of index referenced","140=Authorization failure
```

```
    ","141=Explicit grant or revoke","142=Create, Alter or Drop against
     tables that are audited or multilevel security at row level","143=
    First attempted change (write) of an audited object within a unit
    of recovery","144=First attempted access (read) of an audited
    object within a unit of recovery","145=Audit log record in DML
    Session"))|__simpleMap(__regexToken($2,".*IFCID:(\\d+)")
    ,"23=0","24=0","25=0","55=2","83=1","105=0",
    "140=3","141=2","142=3","143=1","144=1","145=2")

severity.map.veryhigh.if.deviceSeverity=3
severity.map.high.if.deviceSeverity=2
severity.map.medium.if.deviceSeverity=1
severity.map.low.if.deviceSeverity=0
```

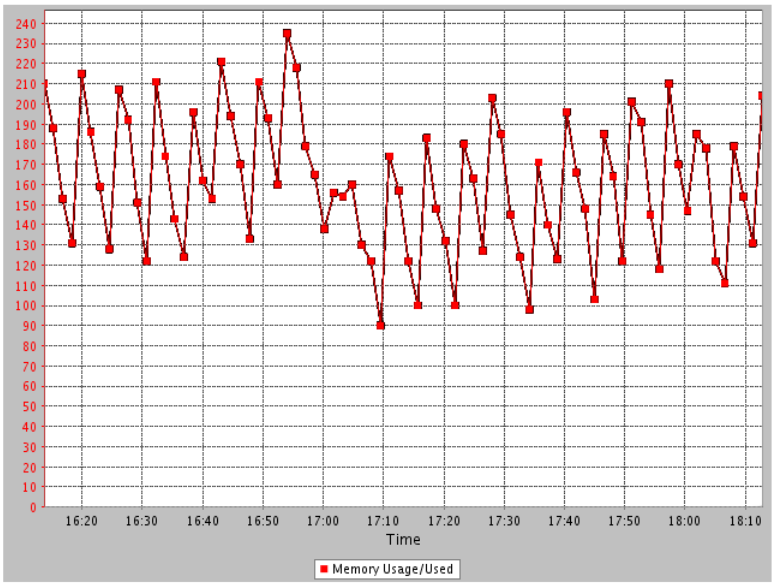
IBM CICS

```
multiline.singleline.nowaiting=true
multiline.starts.regex=\\d{4}-\\d{2}-\\d{2}.*
multiline.ends.regex=\\[0-9A-F\\]*\\s*$
multiline.delimiter=?

trim.tokens=true

token.count=13

token[0].name=event_time
token[0].type=String

token[1].name=timezone
token[1].type=String

token[2].name=user_id
token[2].type=String

token[3].name=application
token[3].type=String

token[4].name=cics_trans
token[4].type=String

token[5].name=cics_term
token[5].type=String

token[6].name=cics_reg
token[6].type=String

token[7].name=country_code
token[7].type=String

token[8].name=action
token[8].type=String

token[9].name=entity
token[9].type=String

token[10].name=succ_fail
token[10].type=String

token[11].name=info
token[11].type=String

token[12].name=hexa_entity
token[12].type=String

event.deviceReceiptTime=__createSafeLocalTimeStamp(__concatenate(
    event_time,"(GMT",timezone,":00)"),"yyyy-MM-dd HH.mm.ss'('z')'")
event.deviceTimeZone=__concatenate(__stringConstant("GMT"),timezone)
event.destinationUserName=user_id
event.requestClientApplication=application
event.deviceCustomString1=info
```

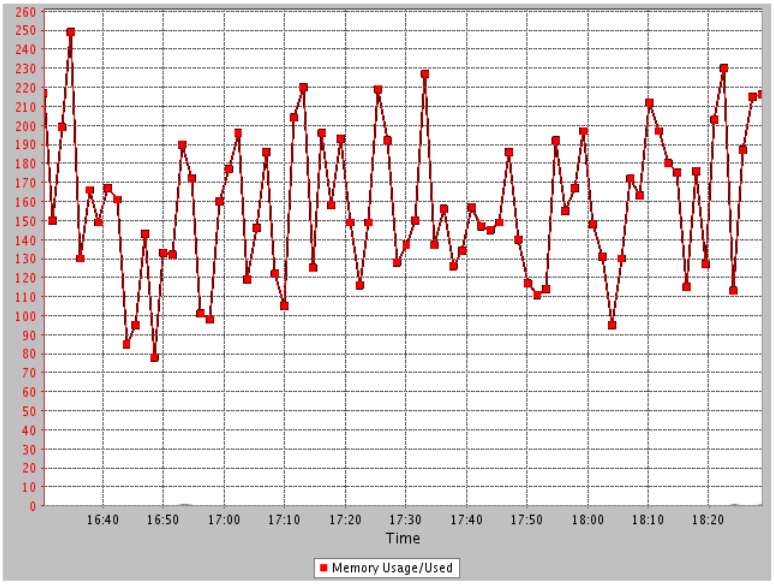
```
event.deviceCustomString1Label=__stringConstant("Additional Info")
event.deviceCustomString2=cics_trans
event.deviceCustomString2Label=__stringConstant("CICS Transaction")
event.deviceCustomString3=cics_term
event.deviceCustomString3Label= __stringConstant("CICS Terminal")
event.deviceCustomString4=cics_reg
event.deviceCustomString4Label=__stringConstant("CICS Region")
event.deviceCustomString5=__simpleMap(country_code,"024=AO","616=PL
    ","620=PT","642=RO")
event.deviceCustomString5Label=__stringConstant("Instance")
event.name=action
event.deviceAction=action
event.message=__replaceAll(entity,"[^A-Za-z0-9!+,-./:;=_<>\x22\x27
    ]","?")
event.eventOutcome=succ_fail
event.deviceCustomString6Label=__stringConstant("Environment")
event.flexString1=hexa_entity
event.flexString1Label=__stringConstant("Hexadecimal event.message")

event.deviceSeverity=__simpleMap(action,"OPEN SESSION=0","CLOSE SESSION
    =0","DELETE SESSION=1","CREATE SESSION=1","OPEN TRACE=2","CLOSE
    TRACE=0","VIEW TRACE=2","CHANGE PARAM=2","VIEW FILE=2","START
    LISTNER=2","STOP LISTNER=1","CONNECT=0","DISCONNECT=0","DELETE MSG
    =1","VIEW MSG=2","EDIT MSG=3","GO FILE=2","DELETE FILE=1","CHANGE
    FILESTATUS=1","ENTER APPL=1","EXIT APPL=0","EXCHANGE KEY=0","START
    TRANS.F.T.=2","GET FILE=2","REFRESH DIRECTORY=0")

event.deviceVendor=__stringConstant("IBM")
event.deviceProduct=__stringConstant("z/OS App")

severity.map.low.if.deviceSeverity=0
severity.map.medium.if.deviceSeverity=1
severity.map.high.if.deviceSeverity=2
severity.map.veryhigh.if.deviceSeverity=3

regex=(\\d{4}-\\d{2}-\\d{2}\\s\\d{2}\\.\\d{2}\\.\\d{2})\\((.{3})\\)
    (.{8})(.{4})(.{4})(.{4})(.{8})(.{3})(.{20})(.*)(SUCCESS|FAIL.{3})
    (.{70})(.{120})
```

ANEXO II - Connectors: Consumo de memória

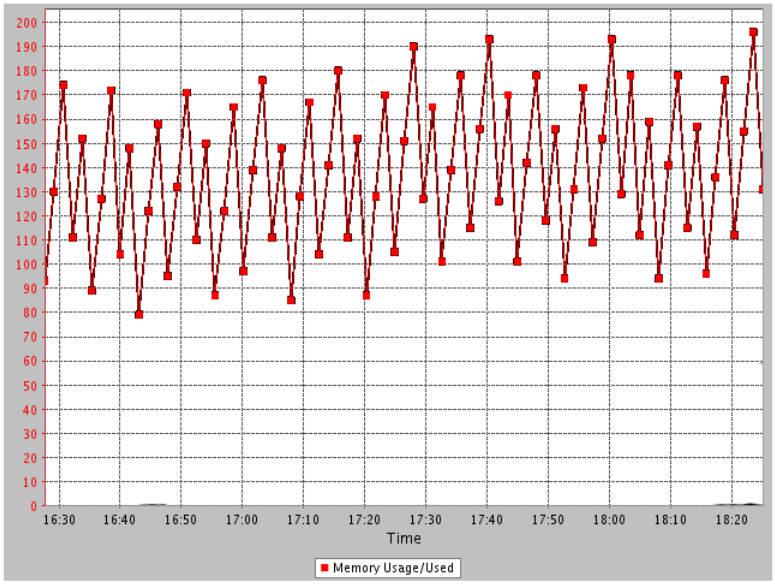
A_AIXAudit



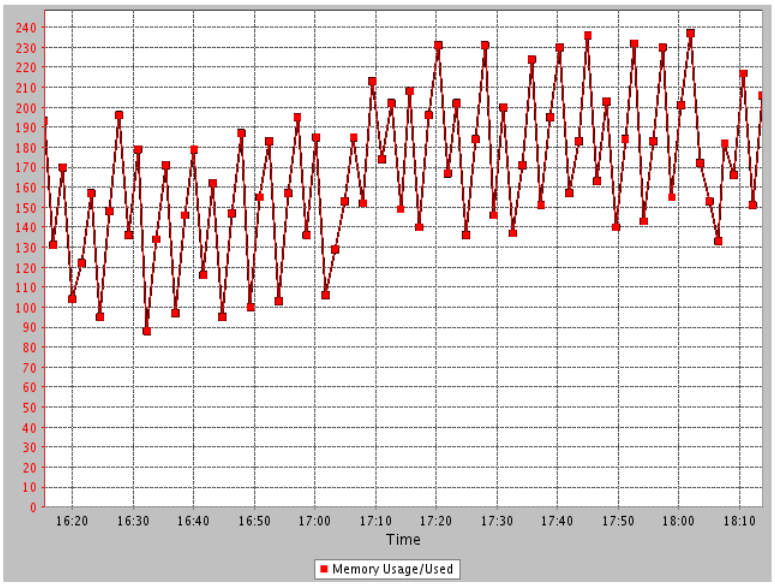
A_Apache



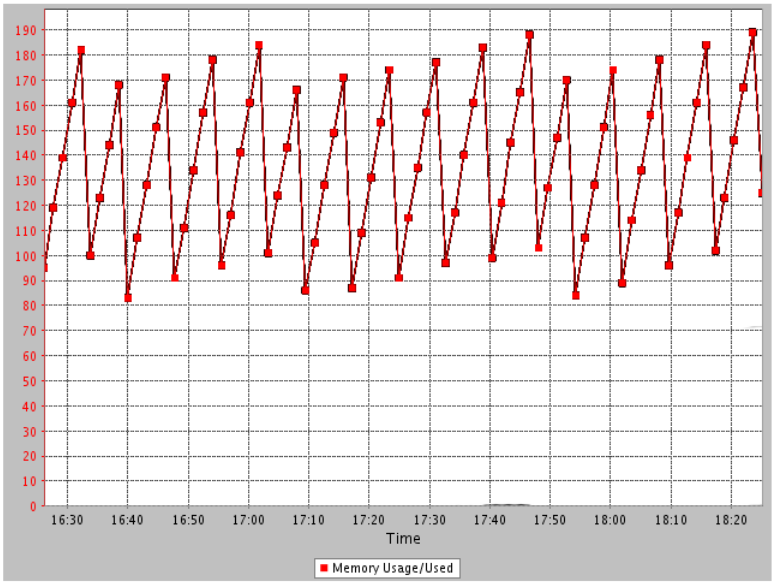
A_Cisco



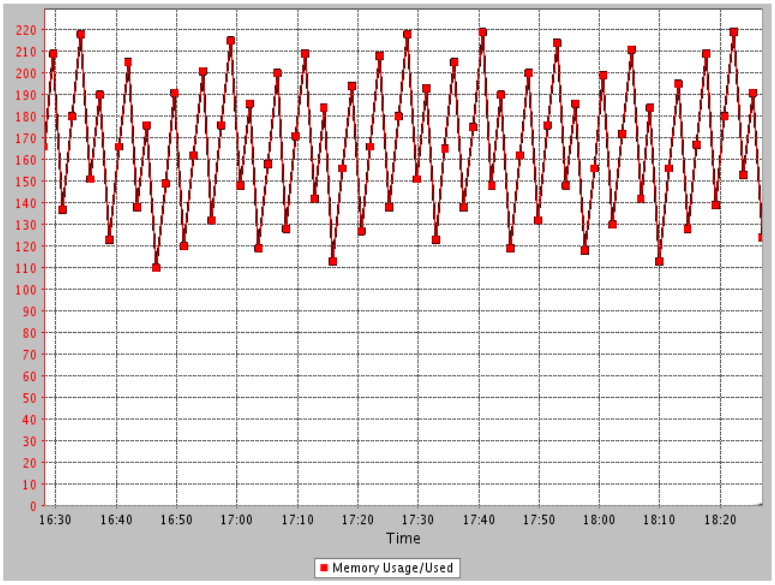
A_Informix



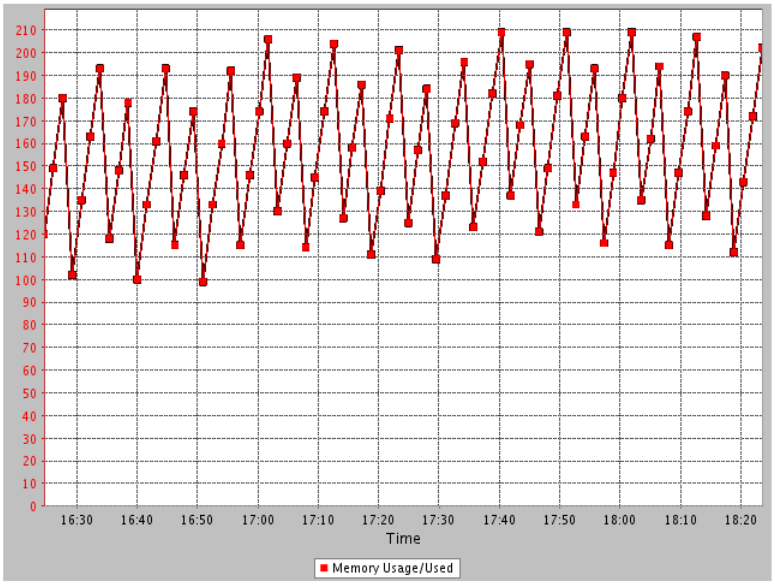
A_ITIM



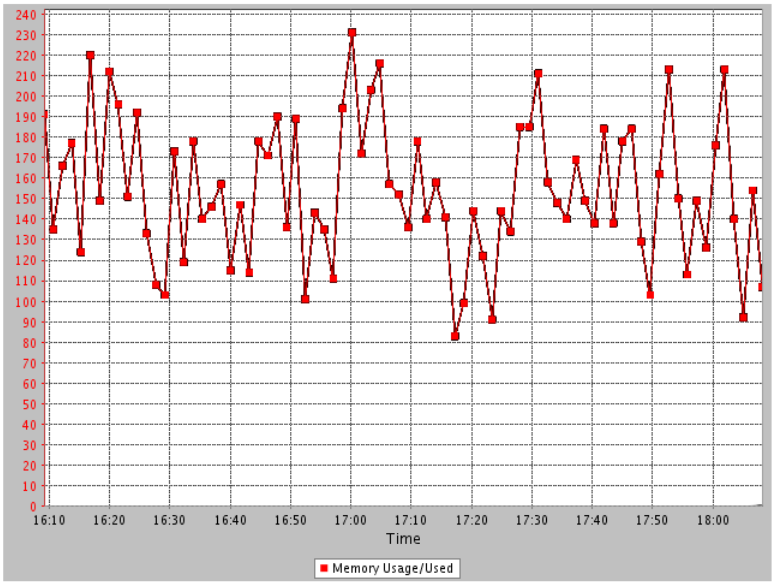
A_MSSQL



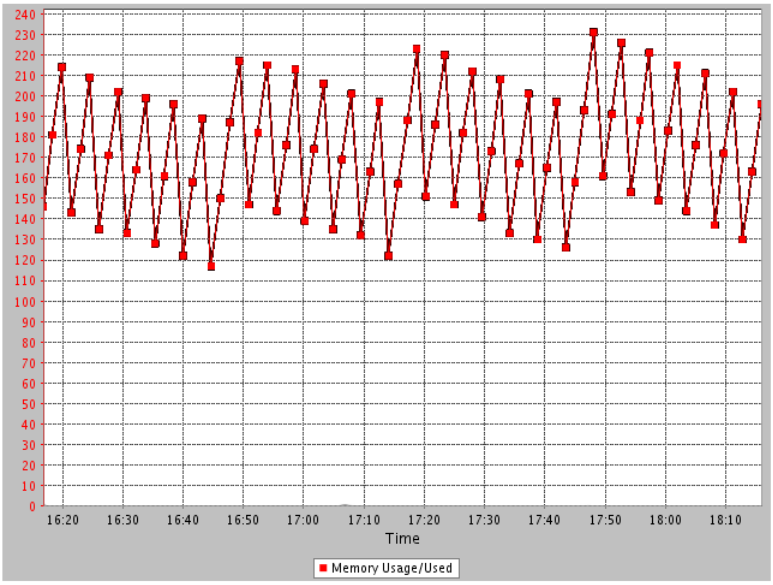
A_OSSEC



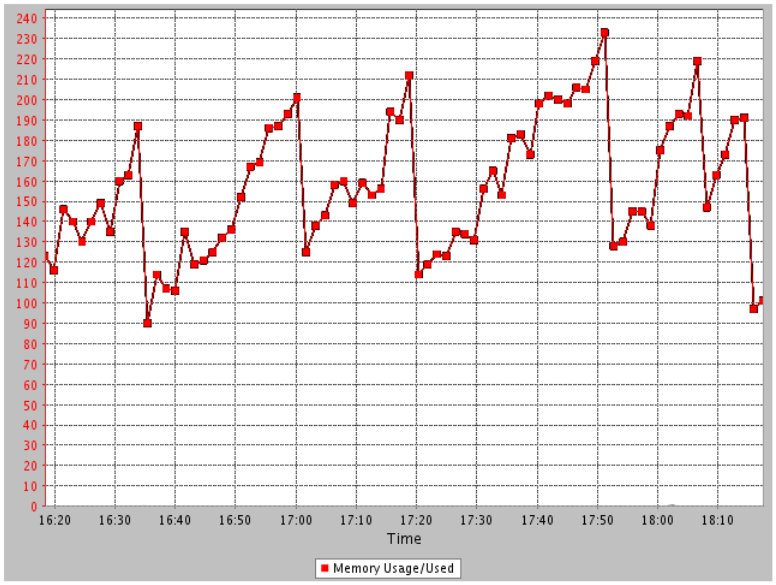
A_Snare



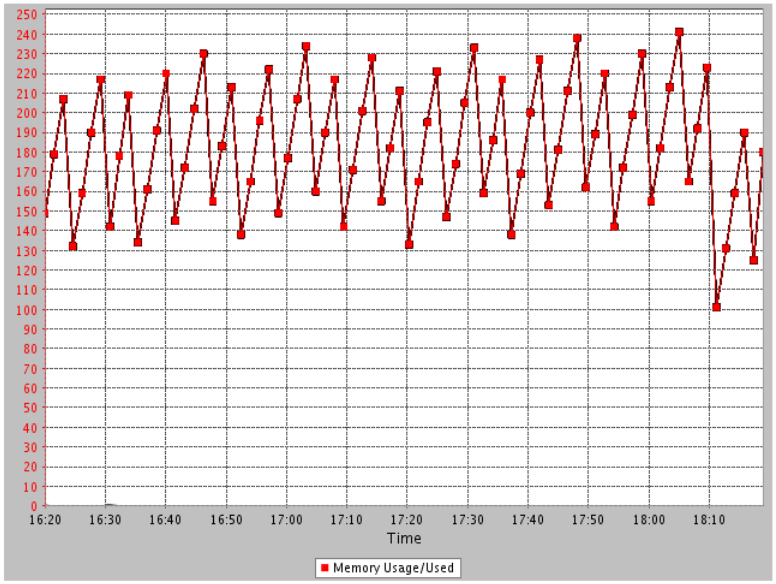
A_Sourcefire



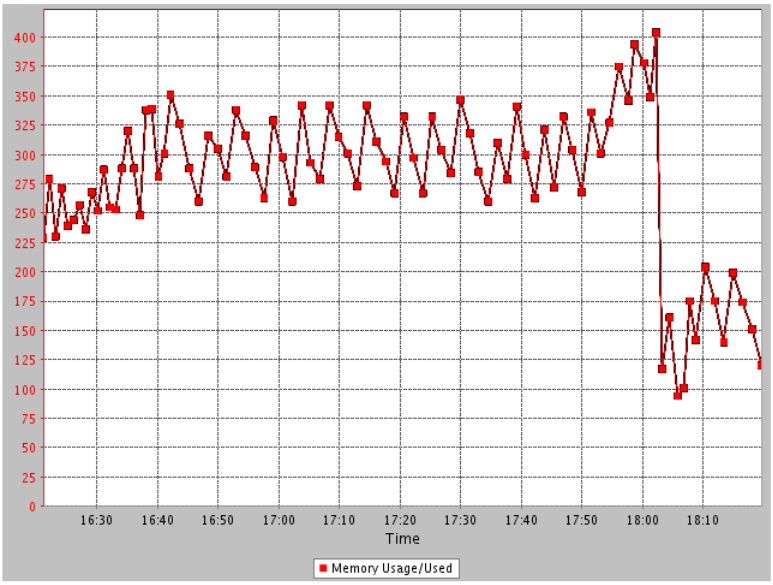
A_Syslog



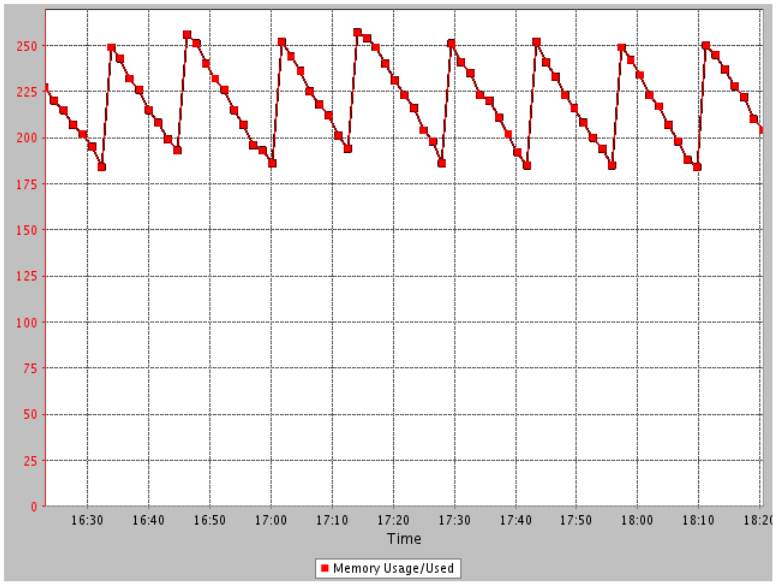
A_TACACS



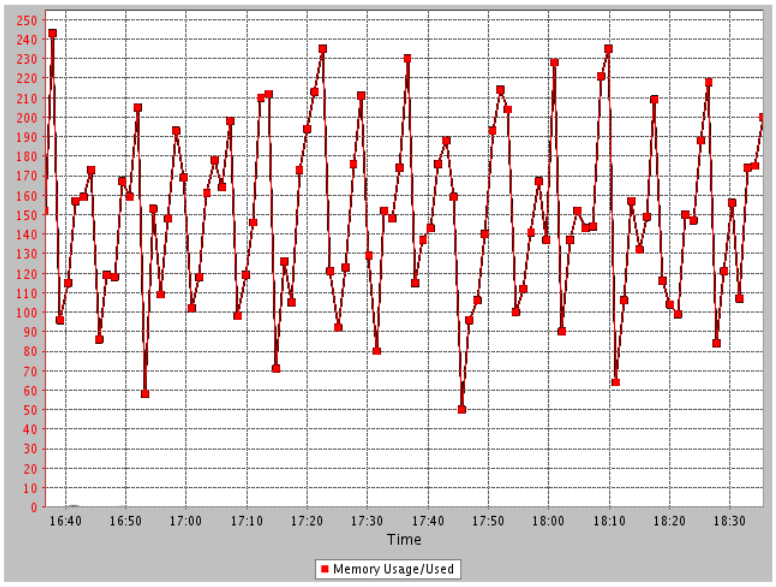
A_zOS_SMF



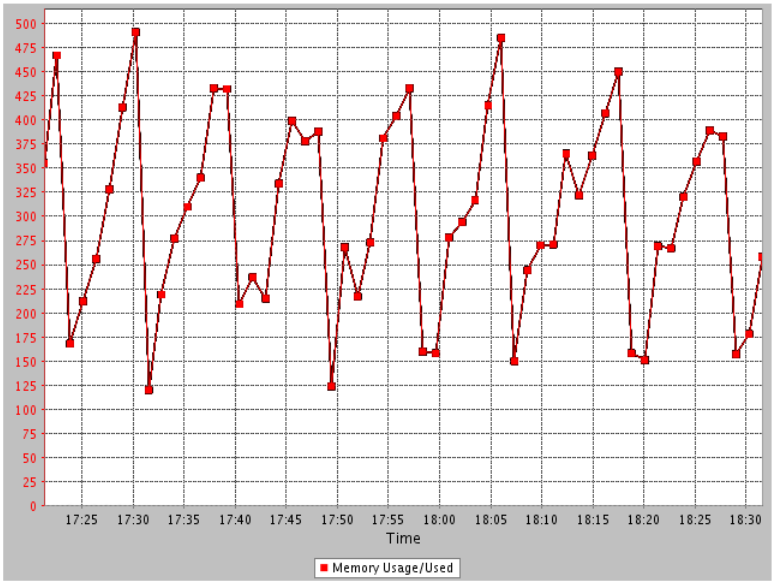
A_zOSApp



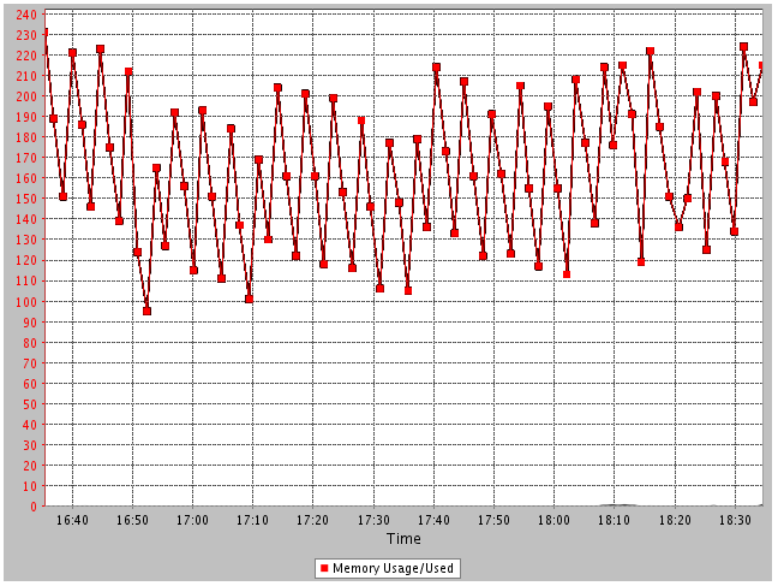
B_CheckPoint



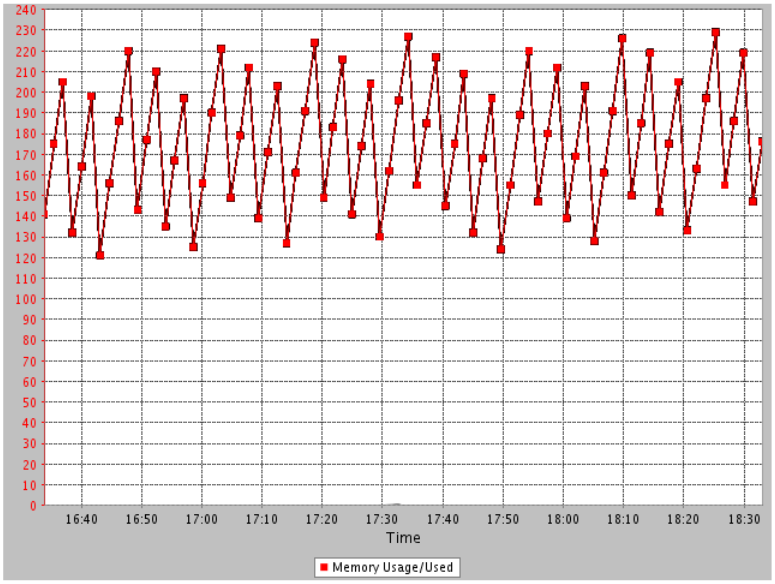
B_Snare



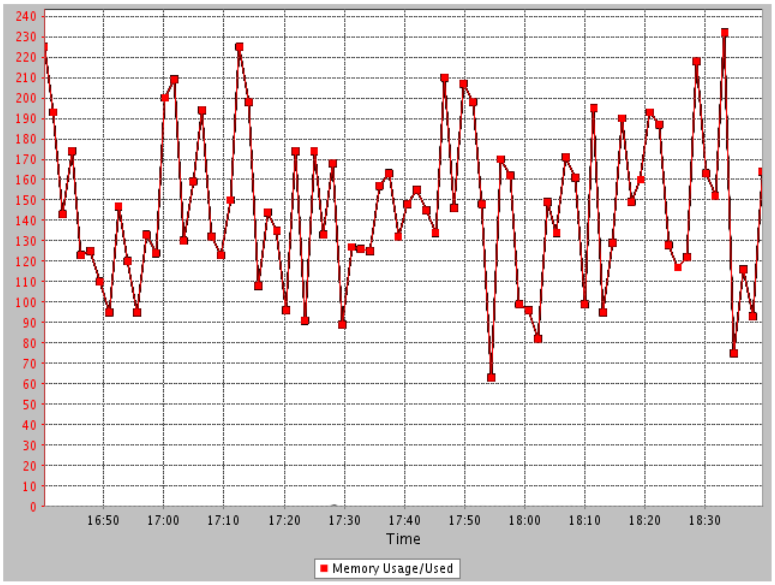
B_SymantecEP



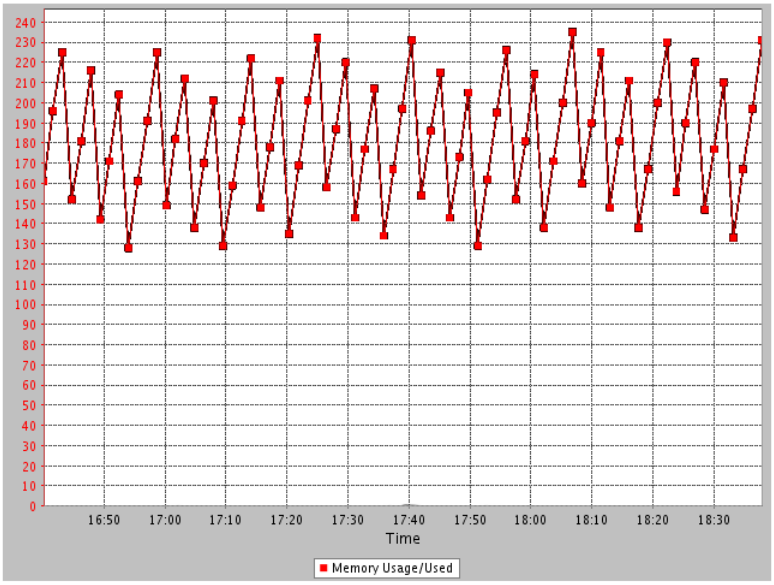
B_Syslog



C_Snare



C_Syslog


240
230
220
210
200
190
180
170
160
150
140
130
120
110
100
90
80
70
60
50
40
30
20
10
0
16:50
17:00
17:10
17:20
17:30
17:40
17:50
18:00
18:10
18:20
18:30
Time
Memory Usage/Used