Observador - Uma Ferramenta Automatizada de Filtragem de Eventos de Auditoria

Samuel Marins de Oliveira e Oliveira*
Instituto Federal da Bahia
Salvador, Bahia
samuelmarins@ifba.edu.br

Resumo

Com a popularização da *internet* e a explosão do acesso à informação, o mundo virtual tornou-se um terreno propício para a ocorrência de crimes virtuais, tais como o de vazamento de informações e o de acesso não autorizado. Investigá-los requer o uso de um conjunto de ferramentas que a computação forense proporciona. Este artigo visa apresentar uma ferramenta denominada Observador, especializada em exibição e filtragem de eventos de auditoria automatizada. A mesma tem por objetivo auxiliar na automatização do processo de ativação da auditoria no sistema operacional *Windows*®, facilitar a aplicação das políticas de auditoria em objetos do sistema (pastas e arquivos) e na filtragem dos *logs* de segurança gerados. As informações contidas nesses *logs* dizem a respeito de atividades desenvolvidas pelos usuários, tais como criação, abertura, modificação e exclusão de arquivos, programas e do registro do *Windows*®. Uma *interface* de apresentação intuitiva do aplicativo proporciona a abstração necessária para que todos os processos descritos sejam transparentes ao usuário.

Palavras-Chave

Auditoria, eventos, computação forense, *logs*, *Windows*®, filtragem.

1. INTRODUÇÃO

"Não há crime sem lei anterior que o defina. Não há pena sem prévia cominação legal que o defina" [2]. Desta forma é iniciado o artigo 1ªdo Código Penal Brasileiro. Crimes fazem parte da humanidade desde que as leis foram produzidas. No entanto, a investigação criminal foi criada para solucionar os delitos e assim sentenciar ou inocentar o(s) acusados(s). Com o avanço da ciência, novas técnicas e conhecimentos científicos foram incorporados à investigação destinada a desvendar os crimes. Tal investigação chama-se perícia forense, que teve seu início no século 18 na necropsia de corpos. A investigação ou perícia forense faz uso de diversas técnicas e instrumentos para alcançar seus objetivos.

Tais técnicas tornam-se mais eficientes ao empregar boas práticas na coleta, restauração, identificação, preservação, documentação, análise de dados periciais, apresentação de vestígios, evidências e provas digitais e interpretação, aplicadas a uma infinidade de crimes, ocorridos também no mundo virtual.

A popularização dos computadores teve o seu surgimento por volta das últimas décadas do século 20, com o início da computação pessoal. Não só revolucionou a sociadade mas também a meneira como interagimos com o mundo. Contudo, foi no advento da *internet* que a realidade tomou um rumo diferente. Os crimes não ficaram restritos ao mundo real, mas chegaram também ao virtual. Gerou-se um novo cenário global, uma vez que a eliminação de fronteiras facilitou e muito a ocorrência de crimes eletrônicos em que a vítima e criminoso podem se encontrar em localizações geográficas remotas ou até mesmo em países distintos [17]. Para um usuário de computador sofrer um incidente de Segurança Computacional, é somente uma questão de tempo. A ameaça tanto poderá ser tratada como um incidente simples ou ser caracterizada como crime [17].

Embora a investigação forense seja de conhecimento público há certa quantidade de tempo, a associação a computação é relativamente nova. Crimes estão ocorrendo no contexto computacional e na própria *Internet*, e geraram demanda para uma nova ramificação da Ciência da Computação , afim de desenvolver inteligência e ferramentas de apoio a investigação e no combate ao crime [17]. De acordo com Gerhel[11], o surgimento da Computação Forense ocorreu pela convergência de dois fatores principal: a) A dependência crescente da criação e aplicação de leis sobre a computação; b) A onipresença de computadores.

Internacionalmente, um vazamento de informações da companhia *Sony*® foi um dos casos de maior repercusão mundial. Os grupos de *crackers e hackers* tais como *Anonymus* e *lizard squad* interrompem o funcionamento normal de *sites* quando assim desejam, causando diversos transtornos.

Para investigar esses e outros crimes virtuais, a computação forense demonstra sua utilidade, já que os investigadores tem a mesma responsabilidade tanto em crimes de cunho comum

*Samuel Marins é estudante do curso Análise e Desenvolvimento de Sistemas no Instituto Federal da Bahia.
E-mail: samuelmarins@ifba.edu.br

quanto em crimes virtuais: obter pistas e coletar rastros. A instalação de programas, criação de arquivos, navegação na *internet* podem deixar ou não referências em repositórios de informações do sistema operacional, tais como: registro, arquivos de texto de *logs* e os *logs* internos do sistema.

Os *logs* do sistemas existem nos principais sistemas operacionais, como nas distribuições *Linux*, nas versões do *Microsoft Windows*® e *MacOS X* e São uma importante ferramenta para manutenção e administração. É através deles que as informações de auditoria, que são relevantes para a coleta de rastros de utilização do usuário, serão obtidas. Por degradar o desempenho do sistema ocasionado pela geração exponencial de *logs*, a auditoria de sistema no *Microsoft Windows*® é desativada por padrão e esta pode não ser uma condição ideal para a computação forense.

Um sistema computacional que sofreu um incidente de segurança ou crime deve ser preservado ao máximo possível. Existem três tipos de análise que podem ser realizadas em um sistema comprometido: 1. Live Analysis. 2. Network Forensic. 3. Mortem Analysis..

Cada uma visa coletar o máximo de informações sem que as mesmas não percam suas características específicas, tais como: poder ser duplicada com exatadião, poder identificar se foi modificada e ser volátil [17]. Então, ferramentas que coletam essas informações não deverão interferir no estado atual sob qualquer condição.

O objetivo deste trabalho é apresentar através de relatórios e gráficos as informações de auditoria de sistemas, relacionadas a ações que o usuário tenha executado no computador, como por exemplo programas executados e arquivos ou pastas excluídos e/ou criados. Contudo, é também possível filtrar e procurar termos específicos e exportar o cojunto de relatórios nos formatos de arquivos mais comuns. Isso possibilita o posterior exame e coleta dessas informações e constituição de prova que possa ser levada a juízo em caso de ocorrência de crimes. E como sendo uma ferramenta especializada em computação forense, a mesma não deve modificar ou acrescentar informações ao sistema a ser analisado.

Este trabalho está organizado em seções e apresentado na seguinte ordem:

- Na seção 2 será explicado o que é auditoria de sistemas, principais riscos, ameaças e como funciona auditoria no *Microsoft Windows*®;
- Na seção 3 será explicada a computação forense, seu objetivo, suas fases de investigação e definição de incidente;
- Na seção 4 será mostrada como é estruturada a auditoria do *Microsoft Windows*®, através dos seguintes tópicos: .Net Framework, Arquivos de Eventos do Windows, *NTFS, AuditPol* e *SetACL*.
- Na seção 5 mostra trabalhos relacionados, como as ferramentas *Windows Event Log Parser*, *Microsoft Log Parser*, *Event Log Explorer* e *Advanced Event Viewer*;
- Na seção 6 é apresentado o modelo proposto, suas funcionalidades, código base utilizado, *APIs* utilizadas, modelagem do sistema, requisitos funcionais e nãofunncionais e casos de uso;
- Na seção 7 são mostrados resultados dos testes e validação realizados;
- Na seção 8 é apresentada a conclusão e possíveis aprimoramentos em trabalhos futuros do Observador.

2. AUDITORIA DE SISTEMAS

Auditoria de sistemas, de informática ou riscos tecnológicos, é a revisão e avaliação dos controles, desenvolvimento de sistemas, procedimentos em TI, infraestrutura, operação, desempenho e segurança da informação que envolve o processamento de informações críticas para a tomada de decisão [16].

Como realiza a função de avaliar os controles, a auditoria verifica quais os constroles necessários para que os sistemas sejam confiáveis e seus níveis de segurança sejam adequados[16]. Isto a torna de fundamental importância para o bom desempenho dos sistemas de informação, provendo informações para administração dos sistemas.

No ambiente virtual, existem riscos e ameaças aos sistemas de computadores que são levados em consideração na auditoria de sistemas. A seguir, exemplos de riscos e ameaças que atingem grandes empresas e que ocorrem com certa frequência são [16]:

Riscos e Ameaças	Consequências
Acesso irrestrito a documentos eletrônicos	Alteração, destruição indevida ou roubo de informações
Hackers, Crackers, Script kiddies (*crackers* inexperientes), Criminosos Profissionais	Custos de *backup* e recuperação de dados
Vírus	Interrupção dos negócios
Espionagem industrial	Vazamento de informações confidenciais para a concorrência

Tabela 1: Relação entre riscos e ameaças à sistemas de informação e suas respectivas consequeências

Em dezembro de 2006, houve o vazamento de documentos confidenciais dos *EUA* para o mundo. Porém, em 2010 ocorreram outros vazamentos que tiveram repercusão internacional e abalaram a confiabilidade da nação considerada mais poderosa do mundo. Todos os casos foram promovidos por uma organização chamada *wikileaks*, criada em 2006 e mantida com diversos domínios ao redor do mundo. Os documentos vazados em 2010 envolviam tanto a guerra promovida pelos *EUA* contra o Iraque, quanto telegramas secretos das embaixadas dos *EUA*.

Houve uma revalação em 2011 pelo jornalista Glenn Greenwald que um orgão da inteligência dos *EUA*, o *NSA*, promovia espionagem em diversos governos, sendo a Presidente Dilma Rousseff um dos alvos. Esses dois casos exemplificam como o vazamento de informações e a espionagem afetam

não somente empresas, mas também governos e como a segurança no espaço cibérnetico é fundamental.

2.1 Auditoria de segurança no Windows

A auditoria de segurança no *Windows*® é dependente do sistema de *log* de eventos, pois é possibilita a publicação de informações, e do sistema de arquivos *NTFS*. O gerenciador de objetos pode gerar eventos de auditoria como resultado de checagem de acesso a objetos. O modo *kernel* é sempre permitido para gerar eventos de auditoria. Para executar uma operação relacionada a auditoria são necessários privilégios específicos. Um privilégio é o direito que uma conta possui para executar operações de sistema, tais como desligar o computador ou ler informações de auditoria [21]. No modo usuário, existem dois tipos de privilégios para gerencia da auditoria do sistema:

- *SeSecurityPrivilege*: É um tipo de privilégio requerido para realizar uma quantidade de ações relacionadas à segurança do sistema, tais como controlar e visualizar mensagens de auditoria. Ele identifica seu titular como sendo operador de segurança [21];
- *SeAuditPrivilege*: É requerido para geração de entradas de *logs* de auditoria.

Desta forma, como é necessário gerenciar *logs* de eventos de segurança, o programa é obrigado a possuir o privilégio SeSecurityPrivilege [22]. Por isso, no *Windows* ® *Vista* ou superiores a ele, o Observador requer privilégios de administrador para ser executado.

Uma importante utilização do mecanismo de auditoria, em especial, na computação forense, é na criação de *logs* de acesso a objetos de segurança, arquivos e pastas em particular [22]. Para que isso ocorra, obrigatoriamente a Política de auditoria de acesso a objeto deve estar ativada, assim como os atributos referentes a auditoria de acesso devem estar habilitados no sistema de listas de controle de acesso do objeto em questão [22].

Sem a prévia ativação da auditoria de acesso a objetos, através de ferramentas tais como *Auditpol*(presente no *Microsoft Windows Vista*® ou superiores) ou via aplicativo "Políticas de segurança local" (*secpol.msc*) para o Console de Gerenciamento da Microsoft® (*MMC*), torna-se mais difícil obter informações referentes a utilização dos arquivos e pastas. Um problema que poderá ocorrer caso seja utilizada uma ferramenta para obter tais informações é o corrompimento do estado da máquina após ocorrer um crime. A instalação poderá manipular a "cena do crime", adicionando elementos que não faziam parte da mesma.

3. COMPUTAÇÃO FORENSE

A computação forense, de acordo com [11], é uma área da computação importante para coleta, análise e validação de provas que podem ser utilizadas em processos criminais aplicados à computação. Ela faz uso de diversas ferramentas e métodos científicos para coleta, preservação, análise, interpretação, documentação e posterior apresentação de evidências que possam ser utilizadas em juízo.

As fases que consiste a investigação forense são:

Figura 1: Ciclo da investigação forense.[1]

Coleta ou aquisição de provas(artefatos a serem analisados) é a primeira fase em uma investigação forense [17]. O sucesso dessa fase dependerá da qualidade do material coletado e da validade dos procedimentos adotados. Os conceitos de Perícia Forense aplicam-se em diversas, portanto, deve haver metodologias e técnicas apropriadas pra manipulação em cada área. Pela diversidade encontrada na computação, o perito em Perícia Forense computacional deve possuir capacidade técnica realizar a perícia em diversos contexto, como sistemas operacionais diferentes, versões de programas e classes e modelos. O objetivo em diferentes sistemas na coleta de informações é o mesmo, porém, a forma como é executado é diferente. Existem algumas fontes de coletas, as que merecem destaque são: exame em mídia de armazenamento computacional, computadores pessoais, servidores e estações em ambiente computacional distribuído, sistemas de rede, equipamento eletrônico programável e sistemas de informação.

A segunda fase da investigação forense é a identificação. Consiste na análise pericional que objetiva organizar os artefatos encontrados, englobando tanto so artefatos identificados no processo, antesd do desligamento abrupto do equipamento denominado Análise *In Vivo*(Live Analysis), como depois do desligamento, denominado Análise *Post Mortem* (Post Mortem Analysis)[17]. Deve ser criada uma documentação clara referente a cada artefato, pois é crucial para elaboração do laudo pericial.

A avaliação é a terceira fase da investigação forense. Consiste na análise de artefatos e da enumeração de eventos em uma linha do tempo. Sua duração varia de acordo com o contexto de como ocorreu o incidente, mesmo utilizando já conhecidas metodologias e ferramentas. Não há equipamentos universais para acesso a dados em todos os tipos de mídia, sendo os resultados derivados exclusivamente das informações extraídas nas mídias.

Ao final, é feita a apresentação dos resultados. Nesta fase é apresentado um relatório de tudo que foi analisado, tentando desta forma desvendar o caso em todos os detalhes. As informações do Perito Forense colaborar na resolução de casos criminosos inocentando ou culpando acusados [17]. De acordo com esse contexto, o Laudo Pericial Forense de Incidente de Segurança pode no seu conteúdo sugerir a origem a categoria do incidente.

Inicialmente utilizada na medicina [11], a investigação forense surgiu como uma ferramenta na autópsia de cadáveres. A medicina forense data do século 18 e evoluiu até dar surgimento a investigação forense criminal através do aprimoramento da examinação de pistas. A preservação, co-

leta, identificação, extração, documentação e interpretação de pistas são processos fundamentais na investigação [11].

Porém, antes de falar a respeito de investigações, é necessário falar sobre incidentes de segurança computacional, que são ocorrências de perda de integridade de dados ou de uma brecha de confidencialidade. Estes incidentes podem consistir de quaisquer atividades de enlace de rede que poderia comprometer a segurança no interior do sistema ou rede [11]. Um incidente também pode ser o acesso não autorizado a informações armazenadas ou a modificação não autorizada de *hardware*, *software* ou rede, que podem ocorrer através de ataques pela rede.

Nos EUA, há uma lei federal que exige que as organizações federais relatem ao *Federal Computer Incident Response Center (FedCIRC)* qualquer violação de um sistema de informação[11]. A *Federal Information Security Management Act (FISMA)*, que faz parte do Departamento de Desesa norte-americano, exige que as organizações realizem as seguintes ações para que seja possível a diminuição de impactos após incidentes de segurança:

- Criar uma política de resposta a incidentes;
- Desenvolver procedimentos para a realização de tratamento de incidentes e relatórios, com base sobre a política de resposta a incidentes; estabeleceram prazos para comunicação com terceiros sobre incidentes;
- Escolher uma estrutura de equipe e com pessoal especializado.

A maioria dos ataques e ameaças a grandes organizações somente são possíveis através da *internet*.[1]

É importante que haja uma maneira de registrar todas as atividades realizadas no computador. A mais usual é através dos arquivos de *log*. *Logs* podem ser definidos como registros detalhados das atividades realizadas dentro de um programa, ou alterações em um banco de dados ou arquivos de computador e mantidos tipicamente em um arquivo [15]. São armazenados em pastas de difícil acesso ao usuários, para evitar que sejam excluídos, dada a sua importância para auditoria de sistemas, computação forense e administração de sistemas. Existem arquivos de *log* de vários tipos, em forma de arquivo texto ou em *XML* ou de eventos de sistema. Por exemplo, quando é instalado um dispositivo no computador usando o *Microsoft Windows*®, é gerado um *log* em forma de texto pelo serviço *plug-and-play* do *Microsoft Windows*®.

Todos os *logs*, principalmente os relacionados ao usuário podem ser revisados, analisados e filtrados posteriormente, tornando-os uma maneira confiável de repositório de informações.

[1]Esses problemas incluem propagação de vírus e *worm*, *spam*, roubo de dados e outras ameaças remotamente exploradas como o ataque denegação de serviço [11].

4. ARQUITETURA DE AUDITORIA DO MICROSOFT WINDOWS

4.1 .Net Framework

A plataforma *.Net Framework*® é estruturada de acordo com os seguintes componentes: CLR (*Common Language Runtime*), *Class Library*, *Windows Forms* e o *ASP. NET (Web Forms e Web Services)*. O principal componente é o *CLR*, pois abstrai o *framework* de liguagem, trabalhando como uma máquina virtual. Este componente possui uma coleção de classes que contém tipos de dados comuns a todas as linguagens. A Figura 2 descreve a arquitetura do *.Net Framework*®.

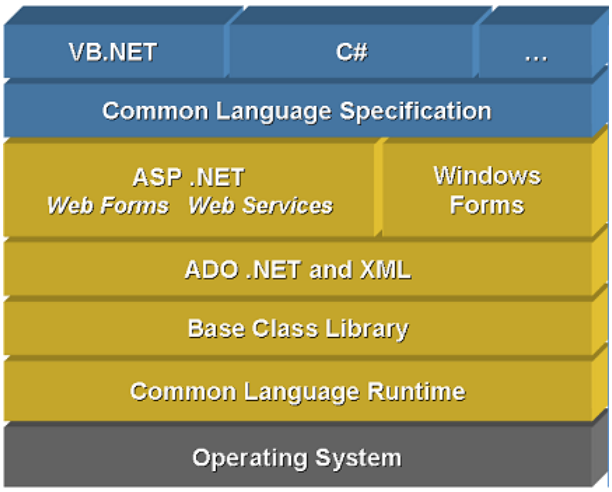


Figura 2: Arquitetura do .Net Framework [12]

O *CLR* é um agente que gerencia o código em tempo de execução, provendo serviços centrais tais como gerenciamento de memória, gerenciamento de *threads*, execução do código, verificação se o código é seguro, compilação e acesso remoto [8]. Este componente está presente no *.Net Framework*® desde sua versão 1.0 [10]. Todas as linguagens são pré-compiladas para a chamada *Common Intermediate Language*, que por sua vez é compilada para linguagem nativa do arquitetura do processador no qual o código é executado. Isso torna o *.Net Framework*® o mais rápido *runtime* comparado com outros, com perda de 5% comparada ao código nativo[10]. Este recurso permitiu por exemplo a adição de uma *API* escrita em *Visual Basic*, enquanto todo o resto do sistema é escrito em *C#*.

O *.Net Framework*® usa os chamados *nameSpaces* para organizar as classes e possuem as seguintes propriedades [9]:

- Organizam o código de grandes projetos;
- São delimitado através do uso de um operador .(ponto);
- A directiva *using* evita o uso do nome do *namespace* para cada classe;
- O *namespace* global ou raiz é é o global::System;

O *.Net Framework*® possui uma coleção de classes que auxiliam o gerenciamento de arquivos de *logs* do *Windows*®.

O *nameSpace* (pacote que encapsula classes) das classes responsáveis pela manipulação de eventos do *Windows*® *no* .Net Framework® é o *System.Diagnostics.Eventing*. As classes que merecem destaque e que foram utilizadas na ferramenta são:

- *EventLogQuery:* utilizada para representar uma consulta de eventos em um *log* e as configurações que definem como a consulta será executada;
- *EventLogReader:* utilizada para habilitar a leitura de eventos de um *log* de eventos baseada na consulta gerada pela classe*EventLogQuery*;
- *EventLogEntry:* utilizada para representar uma entrada de registro no *log* de eventos;
- *EventLog:* utilizada para fornercer interação com *logs* de eventos;
- *EventLogSession:* utilizada para acessar localmente ou em um computador remoto o serviço de *log* de eventos e assim obter os eventos.

Essas classes foram utilizadas na ferramenta para que a consulta em arquivos do tipo .evtx fosse simplificada.

4.2 Arquivos de Eventos do Windows

Os *logs* de eventos dos *Microsoft Windows*® são registros de alertas e notificações do computador. A *Microsoft*® define um evento como "qualquer ocorrência significativa no sistema ou em um programa que requer que os usuários sejam notificados ou uma entrada adicionada a um log. [19].

Existem conjuntos de eventos, que recebem o nome de *channels* e são utilizados para definir categorias de *logs*. Até o *Windows*® *XP*, existiam três categorias (ou *channels*) de arquivos de *log*: *System, Security e Application*. São armazenados em arquivos que levam os mesmos nomes e podem ser encontrados na pasta de sistema: *Windows\System32\Config*. O *Windows*® *Vista*® manteve esses arquivos (ou *channels*) das versões anteriores e também foram introduzidos dezenas de outros novos *channels* específicos de aplicações ou pontos componentes do sistema, tais como *Internet Explorer*® e *Hardware*. Todos os *logs* de eventos podem ser acessados através da ferramenta *Windows Event Viewer*. Para auditoria, o *Windows*® utiliza o *channel security*, que mantém registros de diversos tipos de eventos. Por ser uma ferramenta de auditoria, o padrão de leitura do Observador é o *channel security*. Entretanto, é possível carregar qualquer arquivo .evt ou .evtx de qualquer *channel*.

O sistema operacional *Microsoft Windows*® classifica os eventos pelo tipo. Os tipos de eventos pela sua descrição são os seguintes:

- Um evento de informação descreve o sucesso ao completar uma tarefa, como a instalação de um programa;
- Um evento de aviso notifica o administrador de um problema em potencial, tal como espaço em disco insuficiente;
- Uma mensagem de erro descreve um problema significativo, que pode resultar numa perda de funcionalidade;
- Um evento de sucesso auditoria indica a realização de um evento de segurança auditado, como um usuário final o *logon* com êxito;
- Um evento de falha de auditoria descreve um evento de segurança auditado que não foi concluído com êxito, como um usuário impossibilitado de acessar o sistema, realizando tentativas de senhas incorretas [19].

Até o *Windows*® *XP*® existiam 152 tipos de eventos de auditoria, e houve um salto para 398 a partir do *Windows*® *Vista*® [20].

Os *logs* de eventos possuem as seguintes informações [19]:

- Data: data que ocorreu o evento;
- Hora: hora que ocorreu o evento;
- Usuário: o nome do usuário que estava logado quando o evento ocorreu;
- Computador: o nome do computador;
- Descrição: descrição detalhada do evento;
- Identificador do Evento: o número que o *Windows*® utiliza para identificar o tipo do evento;
- Fonte: o programa ou componente causador do evento;
- Tipo: o tipo do evento (informação, aviso, erro, sucesso ou falha de auditoria de segurança).

Os sistemas operacionais *Windows*® da família *NT*® possuem duas versões para formatar e armazenar dos arquivos de eventos. A primeira versão está presente em todas as edições anteriores ao *Windows*® *Vista*®.

Os arquivos de eventos do Windows são compostos por um cabeçalho [25] com um tamanho fixo de 48 *bytes*, seguido por número variável das gravações dos eventos e o final de arquivo com um tamanho fixo de 40 bytes. O cabeçalho é representado por uma *struct* com nome `ELF_LOGFILE_HEADER`. Essa versão possui a extensão .evt. A interpretação do cabeçalho é feita por uma das classes internas do Observador, para que seja possível o carregamento direto de um arquivo .evt na aplicação.

A versão mais recente dos arquivos de eventos presente no *Windows*® *Vista*® ou superiores é baseado em *XML*. Ele possui a extensão .evtx. Os arquivos são armazenados na pasta do sistema *Windows\System32\WinEvt\Logs*. A estrutura dos arquivos .evtx é composta de: cabeçalho do arquivo (com 4096 *bytes*), blocos de conteúdo e valores vazios [18].

De modo que existem duas versões de arquivos de eventos, a interpretação do conteúdo é feita de forma distinta. Para arquivos os do tipo .evtx, existe uma coleção pronta no *.Net Framework*®, que foi apresentado na seção anterior. O outro modo foi desenvolvido antes da criação de classes de manuseio de eventos e será descrito na seção *Código base*.

4.3 NTFS

O *NTFS* é o sistema de arquivos presente nos sistemas operacionais da família *Windows® NT* [24]. Surgiu em 1993 com a primeira versão do *Windows® NT* 3.1 e foi utilizado em todas versões posteriores, como o *Windows® XP, Windows® Vista, Windows® 7 e Windows® 8*.

Com a intenção de possuir alguma participação do mercado corporativo, que no início dos anos 1990 era dominado pelo sistema operacional *Unix*, a *Microsoft®* decidiu criar um novo sistema operacional, focado em estabilidade, robustez e segurança. Até então, a *Microsoft®* possuía o sistema de arquivos *FAT*, relativamente eficiente para computadores domésticos, mas limitado para o mercado empresarial [24]. Mesmo em sua versão extendida (surgida posteriormente em 1995), o *FAT32*, ele possuia diversas limitações, como as que seguem: baixa ou nenhuma tolerância a falhas, incapacidade de utilização de permissões de arquivos e tamanho limitado a $32GB$ de partições.

Como citado anteriormente, o *NTFS* possui alguns objetivos, entre eles, a segurança. Ela é derivada diretamente do modelo de objeto do *Windows*. Arquivos e pastas são protegidos contra acesso não autorizado [22]. Essa propriedade é disponibilizada por meio do *ACL/ACE* (*Access Control list/Access Control Entry*, respectivamente).

Uma *ACE* é um elemento em uma *ACL*. Cada *ACL* pode conter 0 ou mais *ACE* [3]. Cada *ACE* controla ou monitora o acesso a um objeto por um administrador específico. Existem seis tipos de *ACE*, três com suporte a objetos seguros. Todos os tipos de *ACE* possuem as seguintes informações:

- Um idenfitificador de segurança *SID (Security Identifier)* que identifica o provedor seguro que se aplica o *ACE*;
- Uma máscara de acesso que especifica os direitos de acesso controlados pelo *ACE*;
- Uma *flag* que indica o tipo de *ACE*;
- Um conjunto de indicadores se os objetos filhos do objeto pai que contém o *ACE* podem herdá-lo;

Uma lista de *ACEs* compõe uma *ACL*, com a possibilidade de ter no mínimo zero elementos [4]. Cada *ACE* em uma *ACL* indica um provedor confiável e o tipo se o direito de acesso é permitido, negado ou auditado para esse provedor. Existem dois tipos de *ACL*:

- *DACL (Discretionary Access Control List)* identifica os provedores confiáveis tem o acesso permitido ou negado a um objeto segurado;
- *SACL (System Access Control List)* proporciona aos administradores a possibilidade de registrar em *log* as tentativas de acesso a um objeto seguro.

Cada *ACE* especifica o tipo de tentativa de acesso pelo provedor confiável que a executou, com a função de gerar um registro no *log* de segurança dos eventos do *Windows®*. Um *ACE* pode uma *SACL* pode gerar registros de auditoria para tentativa de acesso, quando esta foi um sucesso, falhou ou para os dois.

As funcionalidades apresentadas pelas *ACLs* e disponibilizadas pelo sistema de arquivos *NTFS* possibilitam de realizar auditoria nos arquivos e pastas do *Windows®*. Desta forma, seria difícil para o Observador realizar os procedimentos que este se propõe a fazer, pois suporte à auditoria deverá ser provido pelo sistema operacional e for utilizado outro sistema de arquivos, ela não estará disponível.

4.4 AuditPol

Ferramenta ou utilitário responsável por exibir informações sobre diretivas de auditoria e executar funções para manipulálas [5]. Está disponível através de linha comando e pode ser utilizada para realizar as seguintes ações:

- Definir e consultar:
  - Diretiva de auditoria do sistema;
  - Diretiva de auditoria por usuário;
  - Opções de auditoria;
  - Descritor de segurança usado para delegar acesso a uma diretiva de auditoria
  - Produzir relatório ou fazer backup de uma diretiva de auditoria para um arquivo de texto delimitado por vírgula (CSV);
  - Carregar uma diretiva de auditoria de um arquivo de texto CSV e configurar SACLs de recursos globais.

O Observador enviará um comando para que o *AudiPol* possa interpretar e assim, ativar ou desativar a auditoria no *Microsoft Windows®*. A versão do *AuditPol* utilizada pelo Observador não é a mesma presente nativamente nos sistemas operacionais *Microsoft Windows® Vista* e superiores. Com o objetivo de ser retrocompatível com o *Windows®* XP e 2003, foi utilizada uma versão que é disponibilizada no *kit* de administração para *Windows®*. Com a utilização do *AuditPol* foi possível tornar as operações de ativar e desetivar a auditoria completamente transparente ao usuário e necessitando somente de um único passo para concretizá-la.

4.5 SetACL

É um programa gratuito para gerenciar permissões, auditoria e informações de propriedade de arquivos do *Microsoft Windows® Vista®* [13]. Ele é capaz de fazer além das ferramentas embutidas no *Microsoft Windows® Vista®*. É capaz de ser automatizada através de *scripts*. Possui duas variantes: um utilitário de linha de comando e uma *COM*, uma *dll* para ser integrada a um projeto em diversas linguagens. SetACL facilita migrações, para reaplicação dos atributos de permissão, pois é um passo essencial. Ele copia permissões entre usuários ou até mesmo domínios. Neste projeto, é responsável por aplicação de permissões de auditoria em arquivos e pastas. Foi escolhida a versão de linha de comando. Sua licença é sobre autorização dos criadores, e foi liberada para o utilização no Observador.

5. TRABALHOS RELACIONADOS

5.1 Windows Event Log Parser(evtwalk)

O *Windows Event Log Parser(evtwalk)* [23] é um utilitário de linha de comando cuja função é a interprertar arquivos de evento do *Microsoft Windows*®, compatível com o *Microsoft Windows XP*® e versões superiores.

O *evtwalk* permite a criação de relatórios de artefatos específicos do *log* de eventos, tais como eventos *plug-and-play* de *USB*, alterações de credenciais de usuários, alterações de senha, eventos de *logon/logoff*, entre outros [23].

Entretanto, este utilitário não é propriamente ligado a auditoria do sistema, reservando-se ao seu propósito inicial. De foma que a auditoria de sistema não é ativada por padrão no Windows, por questões de desempenho, para que os eventos relacionados a isso sejam exibidos, a mesma deverá ser realizada manualmente. Há também uma versão com o mesmo motor de interpretação, porém com *interface* gráfica denominada *evtx_view*.

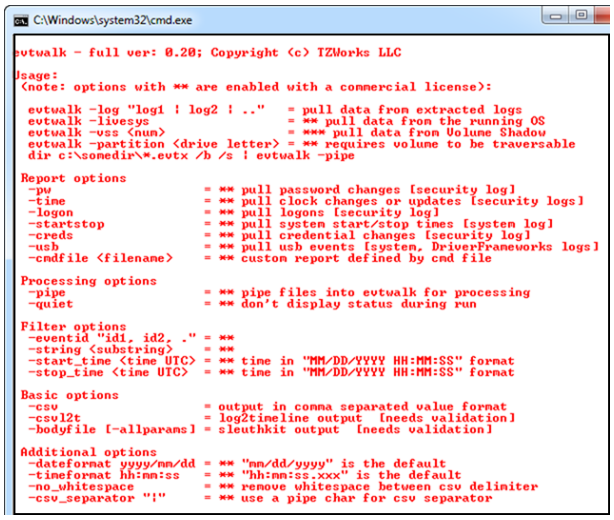


Figura 3: Menu de utilização do *evtwalk*.[23]

5.2 Microsoft Log Parser

Log Parser é uma ferramenta que provém consulta universal a dados baseados em texto, tal como arquivos de *log*, arquivos de *XML* e arquivos *CSV* [7], bem como fontes de dados do sistemas operacional *Windows*® tais como *Log* de eventos, o registro, o sistema de arquivos e o *Active Directory*.

Como as outras ferramentas, esta interpreta os arquivos, porém não ativa auditoria ou tem relação direta com essa premissa.

5.3 Event Log Explorer

O *Event Log Explorer*® é uma ferramenta de visualização de arquivos de evento do *Windows*® compatível com todas as versões atualmente utilizadas. Ele possui algumas funcionalidades que valem ser destacadas, tais como:

- Acessar arquivos de *logs* de eventos *Windows*® no computador local ou em servidor e estações de trabalho remotos;
- Suporte tanto ao formato clássico de *logs* de eventos do *Windows*® NT (.evt) e ao novo formato .evtx;
- Monitoramento e alerta ativo: qualquer problema que ocorrer será informado imadiatamente;
- *Backup* manual e automático dos arquivos de *log* de eventos.

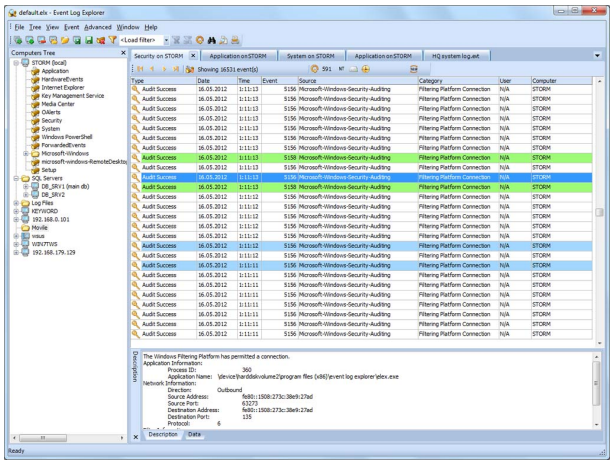


Figura 4: Captura da tela principal do *Event Log Explorer*.[14]

5.4 Advanced Event Viewer

A funcionalidade mais evidente nesta ferramenta é a capacidade de gerenciar e exibir *logs* de eventos de diversos computadores monitorados em uma rede. Ele centraliza todas as informações em um único local em um servidor determinado pelo administrador.

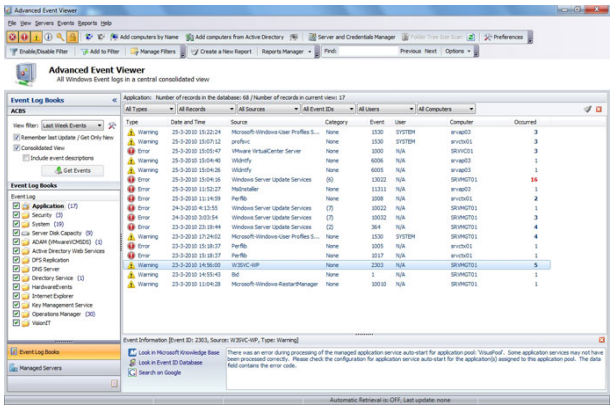


Figura 5: Capture da tela principal do *Advanced Event Viewer*.

É possível observar que nenhum dos programas relacionados ou contidos na tabela comparativa é *Open Source*. Dentre os oito contidos na tabela, seis são pagos na versão com todas as funcionalidades. Todos são de uso generalista, ou seja, destinados a visualização de eventos de todos os tipos. Somente dois possuem a possibilidade de visualização de relatórios em um computador remoto. Dentre os listados,

Nome	Forma de Distribuição	Interface Gráfica	Código Livre	Valor	Compatibilidade	Auditoria	Acesso a Log Remoto
Observador	Gratuito	Sim	Sim	-	XP até 8.1	Sim	Sim
Advanced Event Viewer	Avaliação/pago	Sim	Não	US$199,00(básico)	XP até 8	Não	Sim
Windows Event Log Viewer	Avaliação/pago	Sim	Não	US$240,39	XP até 8.1	Não	Não
MyEventViewer	Gratuito	Sim	Não	-	XP até 8.1	Não	Não
Microsoft® Log Parser	Gratuito	Não	Não	-	XP até 8.1	Não	Não
EventLog Inspector	Avaliação/Pago	Sim	Não	US$ 4,99	XP até 7	Não	Sim
Windows Event Log Parser	Avaliação/Pago	Sim	Não	US$ 228,96	XP até 7	Não	Não
Event Log Explorer	Gratuito/Pago	Sim	Não	Grátis/US$ 49,50	XP até 8	Não	Sim
Event Log Consolidator	Gratuito (menos funcões)/ Pago	Sim	Não	US$ 4.495,00	XP até 8	Não	Sim

Tabela 2: Tabela comparativa entre as ferramentas pesquisadas

somente o produzido pela *Microsoft*®. O Observador é gratuíto, é um *software* livre, possui *interface* gráfica e diferente dos demais, é destinado principalmente a auditoria do sistema operacional *Microsoft Windows* ®. E adicionalmente, possui a capacidade de conectar-se com computadores remotos e obter os relatórios do mesmo. Contudo, possui outras funcionalidades, como gráficos e exportação de documentos, não presentes nas outras ferramentas.

6. OBSERVADOR - UMA FERRAMENTA AUTOMATIZADA DE FILTRAGEM DE EVENTOS DE AUDITORIA

O Observador é uma ferramenta de exibição de relatórios de auditoria de sistemas e filtragem de eventos de auditoria automatizada e geração de gráficos que visa simplificar a tarefa de obter informações sobre as atividades que os usuário realizaram, utilizando a auditoria do sistema *Windows*®. O programa é baseado em algumas funcionalidades nativas do próprio *Windows*®, tais como o log de eventos, suporte a atributos de segurança provido pelo sistemas de arquivos *NTFS* e a interação nativa entre o *Windows*® e o *.Net Framework*®.

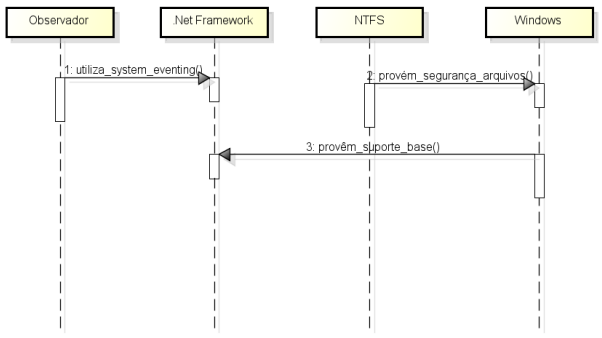


Figura 6: Interação entre os componentes

O ambiente utilizado para o desenvolvimento do Observador é baseado no *.Net Framework*® 4.0. Este *framework* permite a criação de aplicações baseadas na linguagem de programação *C#*, como é o caso do Observador. Por ser uma linguagem fechada, a opção mais plausível de *IDE* é o proprietário da *Microsoft*®, o *Visual Studio* 2010. A *interface* gráfica do Observador é baseada na tecnologia *Windows forms*, de programas desenvolvidos baseados em *C#* e *Visual Basic*.

Para garantir a compatibilidade entre diversas versões do *Microsoft Windows*®, o Observador foi idealizado e desenvolvido tanto no *Microsoft Windows XP*®(x64) quanto no *Microsoft Windows*® 8.1 e testado no *Microsoft Windows*® *XP* (32 *bits*), *Microsoft Windows*® 7 (32 e 64 *bits*).

A implantação do Observador pode ser realizada de duas formas: portátil e instalável. Na instalação portátil, somente haverá uma opção, inclusa na própria *interface* gráfica do Observador, para que o mesmo seja adicionado a pasta de inicialização do *Microsoft Windows*®, para que seja iniciado com o sistema. No modo instalável, haverá um *wizard* para escolha do local de instalação e criação de atalhos. É recomendável utilizar o método portátil pois o mesmo minimiza modificações feitas no sistema. O Observador requer privilégios de administrador em sistemas *Microsoft Windows*® *Vista*® ou superiores.

O desenvolvimento do Observador foi realizado para duas plataformas: x86 e x64. Isso se deu pelo fato de aplicações x86 em sistemas x64 acessarem diretórios diferentes (pastas do sistemas) que uma versão nativa x64. Por exemplo, um aplicativo x86 em um sistema x64 obterá as pastas *Program Files(x86)* e *Syswow64* como sendo *Program Files* e *System32*, respectivamente. Inclusive, o utilitário *SetACL* não poderia atribuir permissões de forma correta se não fosse utilizada uma versão x64 também.

Inicialmente, o Observador realiza a ativação da auditoria de acesso a objetos do *Microsoft Windows*®. Esta etapa é de fundamental importância para o processo de filtragem de eventos, pois obrigatoriamente a auditoria estar ativada para que o sistema operacional inicie o monitoramento dos

objetos. Esta tarefa será realizada pela ferramenta de linha de comando descrita anteriormente, a *Auditpol*.

O Observador, através do seu utilitário *SetACL*, promove a aplicação dos atributos de auditoria de acesso aos objetos que serão escolhidos, através de uma caixa de diálogo de lista de pastas. Esta caixa será ativada logo após a instalação da ferramenta (caso essa seja a opção de implantação) ou em qualquer outro momento que o usuário desejar.

Há dois motores de interpretação de arquivos de *log*, que como citado anteriormente, podem ter duas formatações: evt e evtx. O Observador possui um algoritmo capaz de determinar qual versão do *Windows*® está sendo executada. A partir desta verificação, o motor de interpretação adequado é selecionado.

Todos os *logs* de eventos referentes a auditoria de acesso a objeto são filtrados para serem exibidos na forma que o usuário determinar.

O diagrama que descreve o funcionamento da ferramenta é mostrado na Figura 7.

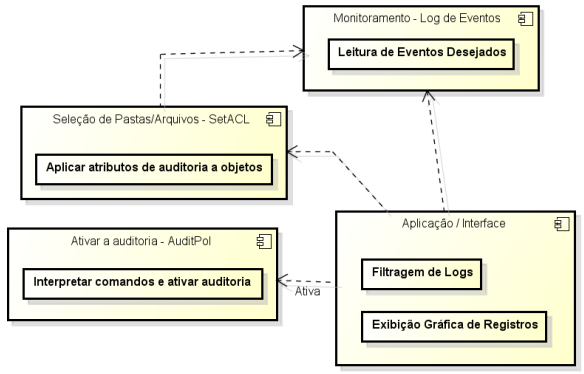


Figura 7: Arquitetura da ferramenta.

6.1 Funcionalidades

Uma das premissas do Observador é possuir uma *interface* gráfica amigável ao usuário. Para cumprir este objetivo, foram projetados dois estilos de *interface* gráfica. O primeiro é baseado no estilo padrão encontrado no *Microsof Office*® 2003. É considerado um estilo clássico e pode ser encontrado trivialmente em diversos outros programas. Já o segundo estilo é mais atual, introduzido com o *Microsoft Office*® 2007. É chamado *Ribbon* e a navegação entre os controles é feita na forma de abas. É possível realizar a troca entre os estilos por meio de botões, localizados a seção de ajuda do programa. Os dois estilo podem ser vistos na Figura 11. Por fim, caixas de diálogos tem um estilo moderno, semelhante ao encontrado em sistemas *Microsoft Windows*® *Vista*®.

A principal funcionalidade do Observador é representar graficamente os *logs* de eventos relacionados a auditoria de sistemas no *Windows*®. Essa representação possui duas vertentes ou maneiras principais: na forma de tabela e na forma de gráficos. O elemento central da *interface* gráfica do Observador é uma tabela, que possui nove colunas, equivalentes ao cabeçalho do arquivo de evento.Todas as funcionalidades são oriundas da representação dos dados nessa tabela, ou seja, ela precisa ser carregada para que todo o resto funcione. Por fim, a representação em gráficos de coordenadas é acessível através do menu "Gráficos"ou aba de mesmo nome, onde é será aberta uma janela que contem uma área para adição de gráficos. É possível adicionar vários gráficos nessa área [continua]. A janela principal dos gráficos é exibida na Figura 8. Para a inclusão de um gráfico é necessário acessar o menu de adição, mostrado na Figura .A tela de adição de gráficos possui opções com quais tipos devem ser escolhidos e em qual área devem ser exibidos. A Figura 9 demonstra essa janela.

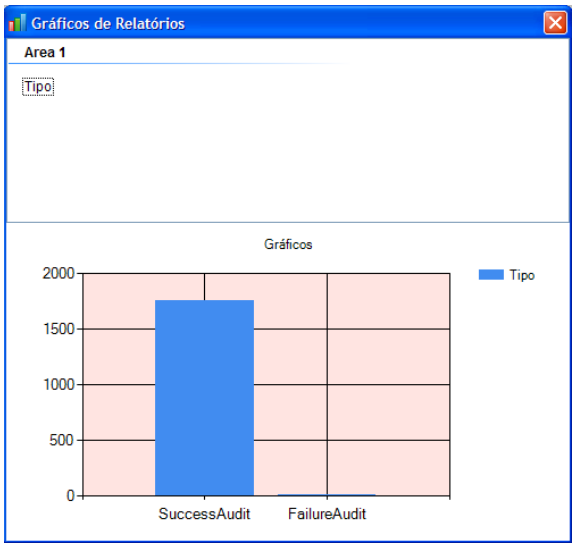


Figura 8: Janela principal de gráficos do Observador.

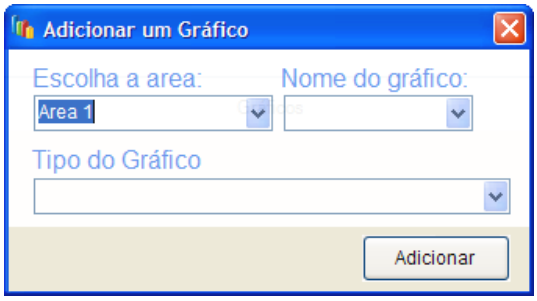


Figura 9: Janela para configuração de gráficos do Observador.

Existem duas formas de obter esses dados em um computador local: através do carregamento de um arquivo de evento (.evt ou .evtx) ou diretamente com o serviço de *logs* de eventos. Para tal, existe um botão com o texto "Exibir relatórios locais", que carrega dados somente de auditoria e somente do computador local, e um item de menu com o texto "Abrir um arquivo de evento", que abre uma caixa de diálogo para selecionar o arquivo e extrair as informações do mesmo.

O Observador ainda conta com a possibilidade de serem visualizados relatórios referentes a computadores remotos.

Uma das premissas do Observador era de ser o máximo compatível com todas as versões do *Microsoft Windows*® disponíveis desde o *Microsoft Windows*® *XP*. Contudo, nem todas as funcionalidades propostas pelo Observador são suportadas no *Microsoft Windows*® *XP e 2003*. Uma delas é a conexão com um computador remoto para obter *logs* de eventos. Então, essa é uma funcinalidade reservada ao *Microsoft Windows*® *Vista* e superiores. Para permitir o funcionamento correto dessa funcionalidade é necessário previamente configurar um usuário para que incluído no grupo de leitores de eventos e o adicionar exceções de gerência de *logs* de eventos às regras de entreada do *firewall*. No caso, as duas condições são feitas no computador de destino.

A *interface* gráfica do Observador possui uma lista de todos os computadores disponíveis no grupo de trabalho ou domínio no qual o computador é pertence. Essa lista é necessária para que haja conexão e inserção de credenciais requeridas para obter os relatórios. Ela é mostrada na Figura 11. A lista omite o computador local onde o Observador está sendo executado.

Há um botão com o texto "Exibir relatórios", que vem desabilitado enquanto nenhum computador é selecionado. Quando algum dos computadores é selecionado, o usuário do Observador é avisado. Após o aviso, o usuário perceberá que o botão ficará disponível, e ao clicar no mesmo, será exibida uma janela para serem inseridos o nome de usuário e senha do mesmo.

Após serem carregados os relatórios, sejam os locais ou referentes a algum computador remoto ou via arquivo, são carregados também os filtros. O Observador possui oito filtros no total, que são equivalente às colunas da tabela excluindo a colula de descrição, pois é individual para cada ocorrência no relatório. Há a possibilidade de quando utilizar um filtro e não quiser posteriormente remove-lo. Para isso, é somente necessário acessar a opção em cada filtro "Todo(a)s os ...". Se a intenção for remover todos os filtros e retornar à visão original do relatório, existe o botão com o texto "Limpar filtros". É possível combinar todos os filtros afim de tornar a tabela de relatórios mais específica possível.

Ainda para aprimorar ainda mais a usabilidade e prover resultados mais refinados, o Observador conta com um campo de pesquisa. Este campo é ativado após o carregamento da tabela. A tabela é filtrada com os resultados enquanto o texto é digitado no campo. Para retornar a visão original, o usuário irá apagar o texto ou clicar no botão de limpeza ao lado do campo.

A qualquer momento após o carregamento do relatório na tabela, é possível atualizar a mesma para verificar se existem novos registros. Existem duas formas de executar essa tarefa: através de um botão com somente um ícone e sem texto, no estilo clássico encontra-se ao lado dos filtros e no estilo *ribbon* encontra-se no topo da janela logo abaixo do título, e simplesmente selecionando através de um clique a tabela e apertando o botão "F5"no teclado. Este procedimento carregará novamente um arquivo ou requisitará as informações novamente do serviço de *log* de eventos, seja local ou remoto.

As informações contidas na tabela de relatórios podem ser vistas detalhadamente de três formas. A primeira é baseada no painel que existe ao lado direito da tabela, com campos correspondentes às colunas da tabela e que são preenchidos ao selecionar alguma linha da mesma. Já na segunda forma, o usuário deverá clicar no *link* com o "Ver propriedades em janela"no painel do caso um. Após o clique, será aberta uma janela contentado as informações e há botões com setas para navegação entre os registros. Existe também um botão para copiar o conteúdo dos campos. Esta janela somente pode ser aberta se os registros já tiverem sido carregados. Se o usuário tentar clicar no *link*, será avisado que a janela está indisponível e que deverá carregar os registros primeiro. O conteúdo dessa janela é exibido na Figura 10.

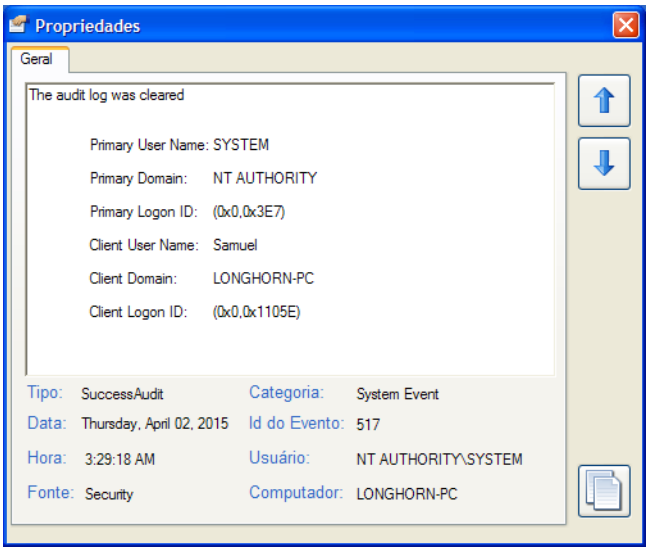
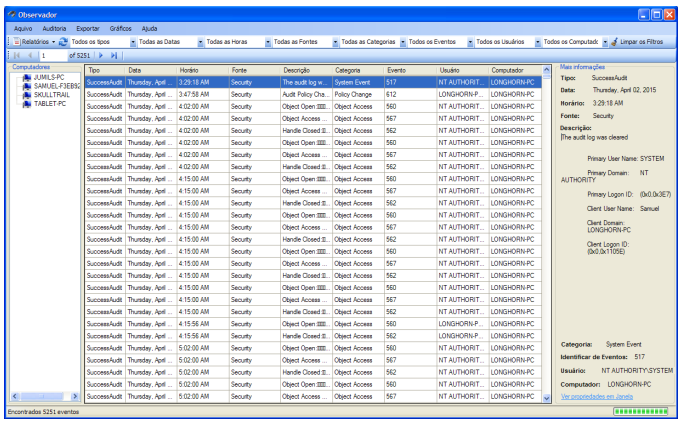


Figura 10: Painel de visualização avançada e *link* de propriedades em janela.

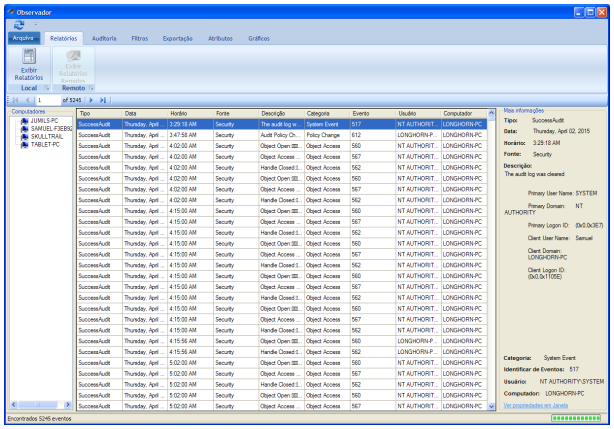
O Observador ainda possui um barra de navegação para a tabela de registros. Essa barra é atualizada com o número de registros encontrados e ao serem filtrados. É possível navegar incrementalmente ou até o último e voltando ao primeiro.

É possível imprimir os registros exibidos na tabela, mesmo que a mesma esteja filtrada. Também é possível visualizar a impressão.

Para auxiliar na criação de documentos com as informações contidas nos relatórios, o Observador conta com a funcionalidade de exportar o conteúdo da tabela em quatro formatos de arquivo: .csv, .xls, .xml e .pdf. Mais uma vez, é possível exportar a tabela com todos os registros ou filtrados. Será salva a visão atual da tabela.



(a) Estilo clássico



(b) Estilo *ribbon*

Figura 11: Elementos gráficos que representam as funcionalidades nos dois estilos do Observador.

6.2 Código Base

Como a formatação dos arquivos de *logs* de eventos foi modificada no *Windows*® *Vista*®, foi necessário utilizar dois algoritmos de interpretação diferentes. Na revisão da literatura foram encontrados dois códigos que serviram de base para os motores de interpreção presentes no Observador.

O primeiro código fonte compõe um protótipo de ferramenta denominado *Event Log Parser*, cuja a função é interpretar arquivos de eventos do *Microsoft Windows*® XP e 2003. Ele foi construído em 2006, baseado no *.Net Framework*® versão 2.0. É restrita a leitura dos arquivos .evt, até por que o *Windows*® *Vista*® não havia sido lançado até aquele ano.

Por se tratar de um protótipo e não uma ferramenta comercial de fato, possui a utilidade de demonstrar o procedimento para interpretação de arquivos .evt utilizando a tecnologia do *.Net Framework*®. Ao contrário do código utilizado para os arquivos *evtx*, a implementação foi de fato desenvolvida pelo criador, em vez de reutilizar classes do *framework*.

O *Event Log Parser* possui três classes: *EventLogParser.cs* (negócios), *MainForm.cs* (*interface* gráfica) e *Program.cs* (principal). A seguir, o trecho do código utilizado para interpretar o cabeçalho dos arquivos *evt*:

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct EventLogRecord
{
    \hspace*{1cm}public uint Length;
    \hspace*{1cm}public uint Reserved;
    \hspace*{1cm}public uint RecordNumber;
    \hspace*{1cm}public uint TimeGenerated;
    \hspace*{1cm}public uint TimeWritten;
    \hspace*{1cm}public uint EventID;
    \hspace*{1cm}public ushort EventType;
    \hspace*{1cm}public ushort NumStrings;
    \hspace*{1cm}public ushort EventCategory;
    \hspace*{1cm}public ushort ReservedFlags;
    \hspace*{1cm}public uint ClosingRecordNumber;
    \hspace*{1cm}public uint StringOffset;
    \hspace*{1cm}public uint UserSidLength;
    \hspace*{1cm}public uint UserSidOffset;
    \hspace*{1cm}public uint DataLength;
    \hspace*{1cm}public uint DataOffset;
}
```

Código 1: Representa o cabeçalho do arquivo de evento do *Microsoft Windows*®

As mudanças no formato do cabeçalho do novo modelo de arquivo *log* de eventos, com extensão .etvx, foram incorporadas nativamente no *.Net Framework*®, a partir da versão 3.5. E para demonstrar como é possível fazer consulta e interpretar arquivos dessa extensão usando o conteúdo do *Microsoft .Net Framework*®, a *Microsoft*® disponibilizou um código de exemplo[6].

Este código possui 6 métodos, incluindo o método principal. Entre esses métodos, 3 são os principais: *QueryActiveLog()*, *QueryExternalFile()* e *QueryRemoteComputer()*. O primeiro é responsável por consultar os eventos por tipo, o segundo por arquivo (carrega um arquivo único) e o último em um computador remoto. A seguir, como exemplo, o método *QueryActiveLog()*:

```
public void QueryActiveLog()
{
 // Query two different event logs using
 //a structured query.
 string queryString =
   "<QueryList>" +
   "  <Query Id=\"0\" Path=\"Application\">" +
   "    <Select Path=\"Application\">" +
   "     *[System[(Level &lt;= 3) and" +
   "     TimeCreated[timediff(@SystemTime)
   "  &lt;= 86400000]]]" +
   "    </Select>" +
   "    <Suppress Path=\"Application\">" +
   "        *[System[(Level = 2)]]" +
   "    </Suppress>" +
   "    <Select Path=\"System\">" +
   "    *[System[(Level=1  or Level=2 or Level=3)
   " and"+
```

```
    "    TimeCreated[timediff(@SystemTime)
    "      lt;= 86400000]]]" +
    "    </Select>" +
    "  </Query>" +
    "</QueryList>";

  EventLogQuery eventsQuery = new EventLogQuery(
  "Application", PathType.LogName, queryString);
  EventLogReader logReader = new EventLogReader(
  eventsQuery);

  // Display event info
  DisplayEventAndLogInformation(logReader);
}
```

Código 2: Realiza a consulta aos eventos do canal "Aplicações" do *Microsoft Windows*®

Analisando este método, é possível observar que a consulta a um *log* de evento é realizada de maneira semelhante a uma *query* em um sistema gerenciado de banco de dados. Na consulta em questão, o canal utilizado é "*Application*", a data de criação Essa facilidade é oriunda da implementação nativa do *.Net Framework*®, na qual é possível realizar tais consultas através de arquivos *XML*.

É ainda necessário ressaltar que foi utilizado um código exemplo para produção do componente de gráficos do Observador. Este código foi disponizado por . Ele utiliza uma *API* nativa do *Net Framework*® chamada *Chart*, disponibilizada a partir da versão 4.0 do *framework*.

6.3 APIs utilizadas

Algumas das funcionalidades presentes no Observador foram obtidas através do uso de *APIs* externas ao código base e o *Net Framework*®, que são distribuídas em forma de .dll. Tais *APIs* são *open source*, sem nenhuma restrição de licença para uso educativo e comercial. As *APIs* utilizadas foram: *DGVPrinter*, *ExcelLibrary*, *itextsharp*, *SecureTextBox*, *System.Windows.Forms.Ribbon35* e *VDialog*.

DGVPrinter foi utilizada para possibilitar a impressão da tabela, seja filtrada ou em seu estado original. É possível imprimir e visualizar a impressão. *ExcelLibrary* por sua vez foi utilizada para criação de documentos do *Microsoft Excel*®, com extensão .xls. Já a *API* itextsharp possibita a criação de arquivos .pdf a partir da tabela. *SecureTextBox* garante a conversão de uma *string* em uma *SecureString*, tipo de dado requerido pra conexões em computadores remotos. O estilo *ribbon* foi provido pela *APISystem.Windows.Forms.Ribbon35*. Por fim, os dialogos ao estilo *Windows*® *Vista*® foram providos pela *API VDialog*.

6.4 Modelagem do Sistema

A arquitetura da ferramenta segue o padrão arquitetural *MVC - Model View Control*. Este padrão garante o isolamento entre os módulos de apresentação, de lógica e de interação com dados. Porém, a arquitetura não pode ser descrita como *MVC* clássico, pois, como há a intenção de interferir no sistema, somente ocorrer a leitura e não escrita de informações. O Observador é composto por 3 classes na camada de controle, 2 classes e uma interface na camada de modelo e 7 classe de *interface* gráfica (*WindowsForms*) com uma classe de apoio. O número reduzido de classes é obtido através do uso do *Net Framework*®.

Todas as classes são representadas na Figura 13. A classe *Utilidades* contém métodos de negócios, necessários para classes de visão acessarem o modelo, entre outras funções. *EventLogParser* é a classe responsável por fazer a leitura e interpretação dos arquivos .evt. Já a classe *EventXLogParser* é responsável pelos arquivos .evtx. A classe responsável por determinado tipo de arquivo será instanciada na classe *Utilidades* após a verificação de qual versão do *Windows*® está sendo executada, através do método *estaExecutandoWinVistaOuSuperior*. O método *obterMotor* requisita a classe adequada para leitura e entrega de informações do fluxo de dados dos arquivos de eventos. Tanto *EventLogParser* quanto *EventXLogParser* implementam a *interface EventLog*.

Por possuir dois estilos de *interface* gráfica, foi-se necessário desenvolver duas classes distintas, mas que possuem funções em comum. Para que não houvesse repetição desnecessária de código, foi criada uma classe *helper*, que recebe o nome de *InterfaceHelper*. As duas classes utilizam métodos da classe *helper*, e toda configuração de caixas de diálogo desenvolvida nela. As classes que pertencem a camada de visão podem ser vistas na Figura 12.

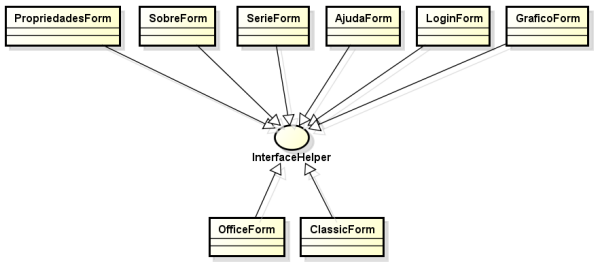


Figura 12: Classes da camada de visão do Observador.

A ferramenta utilizará o repositório de *logs* de eventos obter dados e informações. O Observador suporta a comunicação com esse repositório por duas maneiras: via serviço de *log* de sistemas e por carregamento de arquivo. Para realizar tal função, existem dois métodos: InterpretarLog() e InterpretarLogPorArquivo(). Para a funcionalidade de comunicação de computadores remotos, o aplicativo utiliza uma classe do *Net Framework*® chamada *EventLogSession*. Essa classe permite a comunicação direta com o serviço de *log* do computador remoto. Entretanto, é somente suportada no *Windows*® *Vista*® ou superiores.

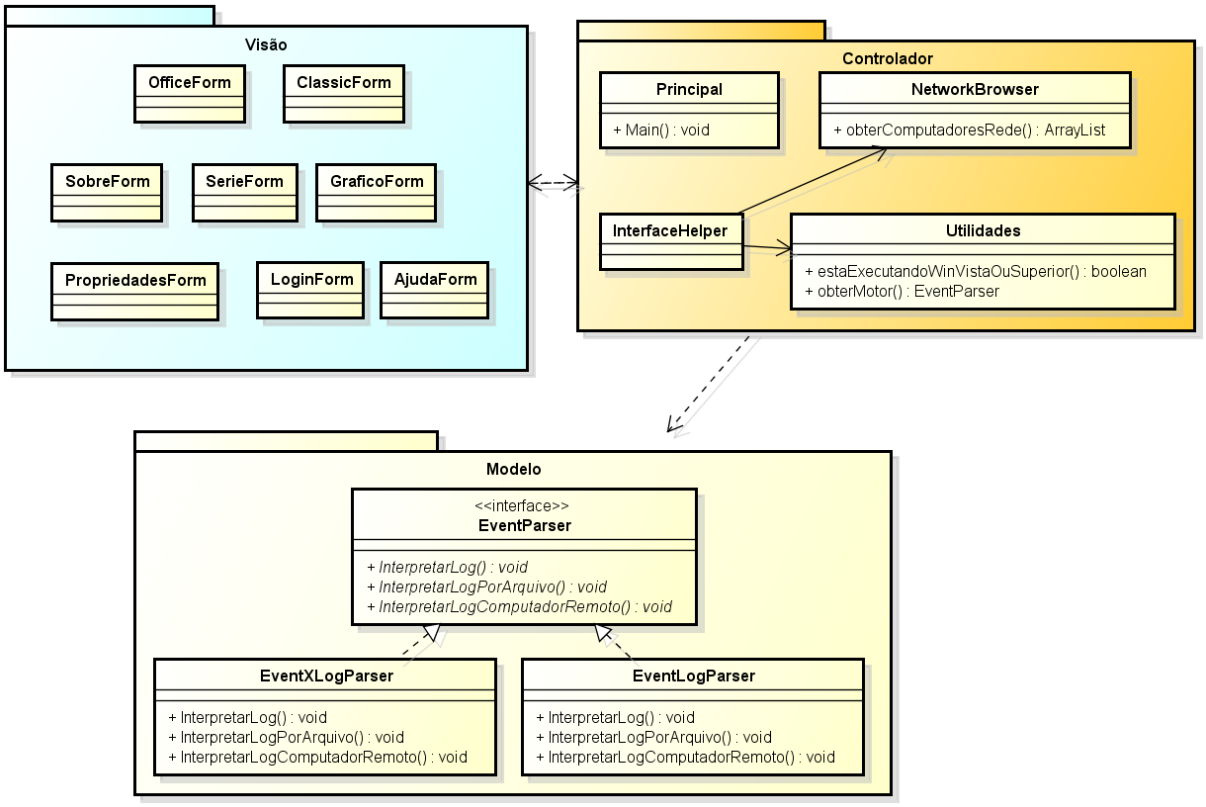


Figura 13: Classes que compõem o Observador.

6.5 Requisitos funcionais e não-funcionais

A fase de levantamento de requisitos faz parte do desenvolvimento de um *software* para que o mesmo funcione da maneira esperada. Desta forma, o desenvolvimento do Observador conteve esta etapa e durante o decorrer da mesma foram encontrados alguns requisitos funcionais e não-funcionais que auxiliaram na modelagem da ferramenta. A seguir, os requisitos divididos em suas categorias.

6.5.1 Requisitos funcionais

Foram encontrados os seguintes requisitos funcionais:

- O Observador deverá interpretar os arquivos de eventos do *Windows*®;
- O Observador deverá gerar relatórios a partir dos arquivos de *log* de eventos do *Windows*®;
- O Observador deverá representar através de gráficos os dados encontrados nos relatórios;
- O Observador não poderá interferir no funcionamento do sistema mais do que o necessário, por ser uma ferramenta de computação forense;
- O Observador deverá ativar a auditoria do sistema operacional *Windows*®;
- O Observador deverá possuir uma *interface* gráfica intuitiva e amigável;
- O Obvervador deverá ser compatível com todas as versões do *Windows*®, a partir da versão XP;
- O Observador deverá possibilitar a aplicação de atributos de auditoria necessários ao processo de auditoria de sistemas;

6.5.2 Requisitos não-funcionais

A seguir, os requisitos não-funcionais encontrados:

- O Observador deverá utilizar o *.Net Framework*® como base para auxiliar no processo de interpretação de arquivos de *log* de eventos;
- Para o desenvolvimento de certas funcionalidades, tais como impressão de tabela, exportação em formatos de documentos populares, o Observador utilizará *APIs* já existentes, e *Open Source*;
- O Observador deverá utilizar a ferramenta *Auditpol* para realizar a ativação da auditoria, pois além de ser de fácil utilização, não interfere além do necessário no sistema;
- O Observador deverá utilizar a ferramenta *Subinacl* para realizar a aplicação de atributos relacionados a auditoria em objetos do sistema (pastas e arquivos);

6.6 Casos de uso

Nesta subseção serão desmonstrados os casos de uso que descrevem as funcionalidades do Observador. Na Figura 15 são representados todos os casos de uso que são possíveis de acontecer.

O Observador será utilizado por um perfil de usuário, que segue a descrição:

- Administradores de sistema: usuários que possuem privilégios elevados e administram sistemas computacionais de uma empresa ou organização. É possível também que seja um administrador de um computador único, porém, deverá ter conhecimentos acima de um usuário comum e também privilégios avaçandos. Pode ser um investigador profissional e da polícia.

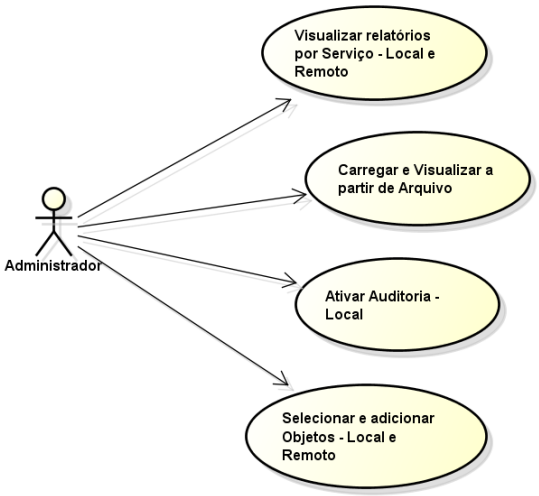


Figura 14: Casos de Uso Primários.

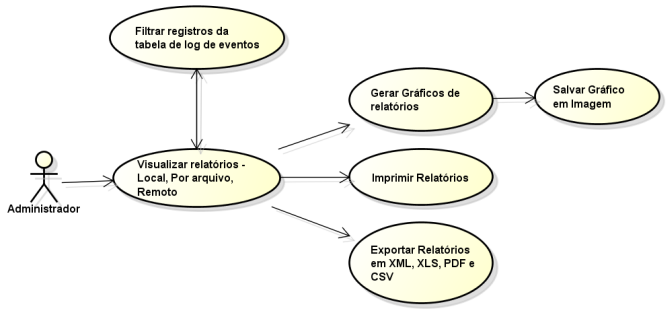


Figura 15: Casos de Uso para Visualização de Registros.

Como mencionado na subseção de funcionalidades, o Observador possui duas *interfaces* gráficas próprias que provêm o acesso as informações dos *logs* eventos. Essas *interfaces* gráficas possuem menus e botões para ativação e desativação da auditoria no sistema operacional *Microsoft Windows*®, abertura de um arquivo específico de *log* de evento ou visualização de *logs* pelo serviço local ou de computador remoto, geração de gráficos dos registros da tabela e gravação em forma de imagem, exportação em formato de documento ou impressão da tabela. Há também filtros para manipular a forma de visualizar dos relatórios gerados a partir dos *logs*. O Observador possui três fluxos que merecem destaque, sendo demonstrados nas Figuras a seguir.

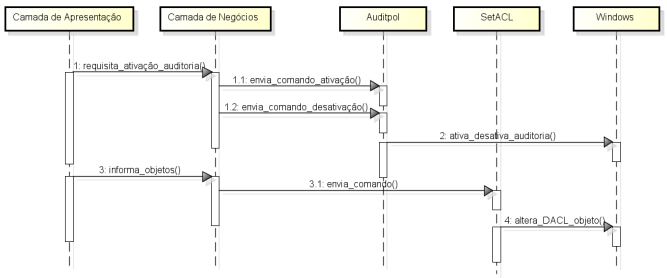


Figura 16: Diagrama de sequência de ativação ou desativação da auditoria de sistema e seleção de objetos

O fluxo para ativar ou desativar a auditoria no *Microsoft Windows*® e aplicar atributos de auditoria a objetos do sistema é descrito a seguir:

1. Após iniciar o programa, o usuário possuirá uma tela com alguns elementos, entre eles, um menu (seja no estilo clássico ou *ribbon*);
2. Uma das opções que esse menu apresenta tem o rótulo de "Auditoria"(uma aba no estilo *ribbon*), que deverá ser selecionada para que seja possível ativar ou desativar a auditoria, e assim, o monitoramento dos *logs* de eventos.
3. Há outra opção no menu do programa, que possui o rótulo de "Selecionar objetos";
4. Se essa opção for selecionada, haverão duas opções, entre seleção de pastas (no qual aplicativo realiza aplicação recursiva de atributos) ou de arquivo, para que sejam monitorados pelo serviço de *log* de eventos.

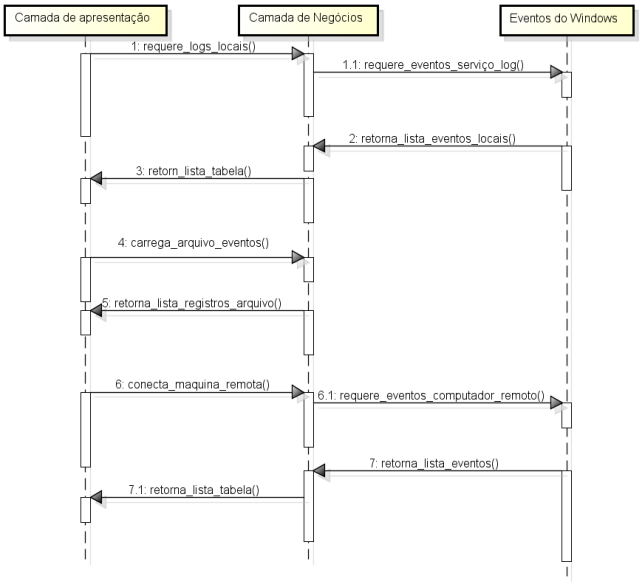


Figura 17: Diagrama de sequência de geração de relatórios.

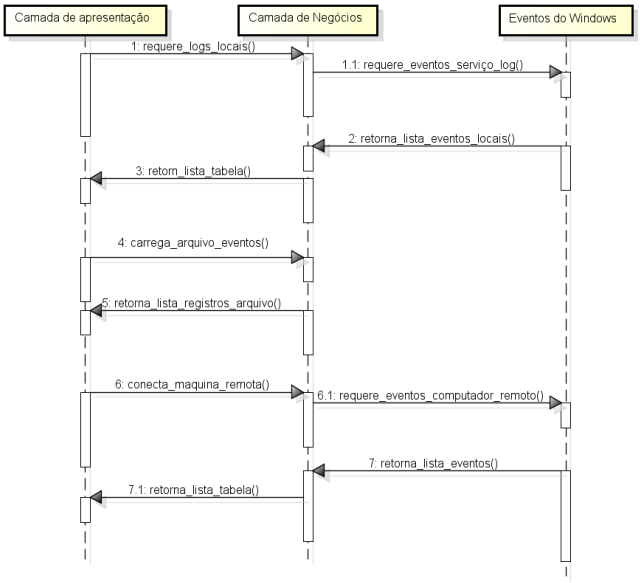


Figura 18: Diagrama de sequência de geração de relatórios.

A figura 17 descreve o fluxo para visualizar de relatórios de *logs* de segurança, para visualizar relatórios a partir de um arquivo ou visualizar relatórios de auditoria de um computador remoto:

- Para que o Observador exiba relatórios relacionados a auditoria, o usuário, na tela principal, efetuará o clique no botão com o rótulo "Relatórios", e em seguida "Exibir relatórios locais". No estilo *ribbon* o procedimento é semelhante, somente sendo diferente o fato de ser acessado por aba;
- O procedimento para acessar relatórios remotos requer mais etapas e o computador local e o remoto devem atender aos requerimentos citados na subseção de funcionalidade. As etapas são: o usuário deverá selecionar um dos computadores listados na janela principal; em seguida, ele receberá um aviso que o computador foi selecionado; o botão de exibir relatórios remotos ficará ativado; ao clicar no botão, será mostrada uma tela de *login*, para validação do usuário no computador remoto; por fim, se os dados estiverem corretos, os resultados começarão a ser carregados;
- No caso de relatórios por arquivo de evento, o usuário deverá acessar o menu com a opção "Arquivo", logo após acessar "Carregar um arquivo", tanto em estilo clássico quanto em *ribbon*. Logo após selecionar o arquivo pela caixa de diálogo que será aberta, os relatórios serão carregados(neste caso, os relatórios dependerão do *channel* dos eventos do arquivo);
- Os três procedimentos possuem métodos distintos, no caso *InterpretarLog(), InterpretarLogRemoto()* e *InterpretarLogPorArquivo*, respectivamente. O método correto será selecionado respeitando os critérios mencionados na subseção de funcionalidades.

O fluxo da Figura 18 descreve como são acessadas funcionalidades avançadas do Observador, e é explanado a seguir:

- É necessário inicialmente exibir relatórios de *logs* em uma das maneiras descritas no fluxo anterior;
- Ao término do carregamento, o usuário poderá fazer um série de ações, descritas na subseção de funcionalidades. Os filtros estarão carregados e poderá ser feita um refinamento da exibição de relatórios;
- Caso seja feita ou não a filtragem dos relatórios, o usuário poderá imprimir todo o conteúdo da tabela acessando o menu "Arquivos"e impressão. Poderá visualizar a impressão acessando o mesmo menu;
- Se a intenção é salvar o conteúdo da tabela para ser anexado a um documento, os relatórios poderão ser salvos dentre os formatos suportados pelo Observador. Esta opção é acessível através do menu "Exportação";
- Para geração de gráficos, o usuário acessará o menu "Gráficos"e posteriormente, clicando no botão "Exibir gráficos". Em seguida, será aberta a janela principal de gráicos. O usuário irá adicionar um gráfico através de um menu de contexto, que abrirá um segunda janela contendo filtros pré-carregados para geração de gráficos simples e cruzados;
- O usuário poderá salvar em imagem o gráfico gerado, acessando o menu de contexto da janela principal de gráficos.

7. TESTES E VALIDAÇÃO

7.1 Testes de desempenho e robutez

É necessário mencionar que ao adicionar muitos objetos para que sejam monitorados pela auditoria de sistema, o número de eventos criados crescerá exponencialmente. Para provar isso, foram realizados com pasta única, pasta com arquivos, arquivo único e conjunto de arquivos. O número de eventos dependerá da quantidade vezes e a forma que o objeto é manipulado. Por exemplo, quando um arquivo é excluído, uma série de eventos anteriores ao evento de exclusão é criada e os eventos são interrelacionados com uma informação chamada *Handle ID*.

A metodologia utilizada dos testes de medição de tempo carregamento de eventos na tabela principal é descrita a seguir: somente o Observador estava sendo executado no computador. Ao término de cada medição, o programa era fechado e aberto novamente. Foram realizados para 5.000, 10.000 e 25.000 eventos aproximadamente e repetidos cinco vezes. E foram realizados para 35.000 eventos e repetidos dez vezes. Os computadores utilizados nos testes possuem as seguintes configurações:

- *Desktop* 1 com processador *Intel Pentium*® *Dual core* G850 de 2.9 *Ghz*, 8 *GB DDR* 3 de 1.333 *Mhz*, 1 *TB* a 7.200 rpm de disco rígido , *Microsoft Windows*® *XP* (x64) e *Microsoft Windows*® 10 (x64) instalados;
- *Desktop* 2 com processador *Intel Pentium*® *Dual core* E2180 de 2.2 *Ghz*, 2.5 *GB DDR* 2 de 667 *Mhz*, 80 *GB* a 7.200 rpm de disco rígido e *Microsoft Windows*® *7* (x86) instalado;
- *Notebook* com processador *Intel Core i3*® M370 *Dual Core* de 2.4 *Ghz*, 4 *GB DDR* 3 de 1.333 *Mhz*, 500 *GB* a 5.400 rpm de disco rígido e *Microsoft Windows*® *8.1* (x64) instalado;
- *Tablet* com processador *Intel Atom*® Z3735G de 1.33 *Quad CoreGhz*, 1 *GB DDR* 3 de 1.333 *Mhz*, 16 *GB* de *SSD* e *Microsoft Windows*® *8.1* (x86) instalado;

O nível de confiança utilizado é de 95%. Os cenários de teste estão descritos abaixo:

Cenário 1: Este cenário retrata a adaptabilidade do Observador. O Observador se mostrou capaz de ser executado em qualquer plataforma *Microsoft Windows*® que atenda aos requisitos mínimos. Ele foi desenvolvido em dois computadores com versões distintas do *Microsoft Windows*®: *Microsoft Windows*® *XP*(x64) e *Microsoft Windows*® 8.1 (x64). Todos os recursos do Observador suportados pela plataforma foram executados com êxito.

Cenário 2: Neste cenário, foram realizados os testes com 35.000 eventos, onde o objetivo é testar a robustez do Observador. Os resultados foram os seguintes:

Computador	Média	Desvio Padrão	Intervalo de Confiança
Desktop 1(Win 10)	28.02s	28.02 +- 0.87	28.02+-0.53
Desktop 1(Win XP)	32.48s	32.48+-0.44	32.48+-0.27
Desktop 2	82.27s	82.27+-1.9	82.27+-1.67
Notebook	41.59s	41.59+-0.37	41.59+-0.32
Tablet	97.78s	97.78+-1.86	97.78+-1.15

Tabela 3: Resultados dos testes de robustez do Observador para 35.000 eventos.

Cenário 3: Neste cenário foi testada a escalabilidade do Observador. As tabelas a seguir retratam os testes para 5.000, 10.000 e 25.000 eventos, respectivamente:

Computador	Média	Desvio Padrão	Intervalo de Confiança
Desktop 1(Win 10)	4.22s	4.22+-0.15	4.22+-0.13
Desktop 1(Win XP)	4,91s	4,91+-0.12	4,91+-0.10
Desktop 2	13.85s	13.85+-1.41	13.85+-1.23
Notebook	6.3s	6.3+-0.37	6.3+-0.32
Tablet	15.02s	15.02+-0.14	32,39+-0.13

Tabela 4: Resultados dos testes de escalabilidade do Observador para 5.000 eventos.

Computador	Média	Desvio Padrão	Intervalo de Confiança
Desktop 1(Win 10)	8.76s	4.22+-0.34	8.76+-0.29
Desktop 1(Win XP)	9.42s	9.42+-0.15	9.42+-0.13
Desktop 2	23,64s	23,64+-0.17	23,64+-0.15
Notebook	11,79s	11,79+-0.17	11,79+-0.15
Tablet	32,39s	32,39+-0.78	32,39+-0.68

Tabela 5: Resultados dos testes de escalabilidade do Observador para 10.000 eventos.

Computador	Média	Desvio Padrão	Intervalo de Confiança
Desktop 1(Win 10)	18,69s	18,69+-0.15	18,69+-0.13
Desktop 1(Win XP)	23,86s	23,86+-0.43	23,86+-0.32
Desktop 2	57,86s	57,86+-1.01	57,86+-0.88
Notebook	28,99s	28,99+-0.24	28,99+-0.21
Tablet	76.07s	76.07+-0.69	76.07+-0.68

Tabela 6: Resultados dos testes de escalabilidade do Observador para 25.000 eventos.

É necessario ressaltar que a funcionalidade de obter relatórios de computador remoto em todos os computadores testados leva um tempo muito grande para ser completada. Um evento leva 1 segundo para ser obtido, ou seja, para quantidades grandiosas é completamente inviável.

Esses testes demonstraram como os objetos a serem monitorados devem ser decididos com clareza e responsabilidade. Pelo que se viu nos testes, é possível entender o porquê da auditoria vir desativa no *Microsoft Windows*®. E também, foi possível enxergar que é necessário limitar o número de páginas para impressão, pois sempre causará problemas se for um número muito grande. Então, recomenda-se filtrar os eventos e assim imprimir os resultados ou salvar em algum formato de arquivo e imprimir por quantidade progressiva de páginas.

7.2 Testes de Opinião e Satisfação

Para realizar a validação do Observador, o programa foi disponibilizado para testes para um grupo de pessoas. Após realizarem testes de funcionalidade, esse grupo de pessoas respondeu um questionário. Infelizmente, nem todas as pessoas contatadas para realização do testes responderam o questionário para que pudessem ser analisadas as respostas. O grupo que participou efetivamente do teste foi composto por cinco pessoas, dentre elas somente uma era leiga na área de computação. Os resultados dos questionários podem ser vistos a seguir:

1. O programa iniciou corretamente? (Sim/Não) - 100% das respostas foram positivas;
2. O programa apresentou algum erro durante a sua execução? (Sim/Não) - 60% das respostas foi positiva;
3. O Observador é fácil de usar? (Sim/Não) - 60% respondeu que sim;
4. O Observador é intuitivo? (Sim/Não) - 80% respondeu que sim;
5. Existe alguma maneira melhor de apresentar os dados? (Sim/Não) Se sim, qual você indicaria? - 100% das pessoas afirmou que não existe uma melhor maneira.

Apesar do grupo não possuir pessoas especialistas na área forense, dentro dos conhecimentos técnicos na computação, algumas comentaram que o trabalho foi desenvolvido de acordo com as especificações. Um dos comentários que merece destaque é o seguinte: "Muito bom mesmo, está de parabéns. Fácil entendimento e útil". É possível observar a importância que uma ferramenta dessa espécie pode ter e como pode ser útil para um verdadeiro especialista na área.

8. CONCLUSÃO E TRABALHOS FUTUROS

A computação forense necessita de ferramentas para tornar possível a obtenção de provas na ocorrência de um crime. E uma ferramenta que possa fazer a auditoria de sistema auxilia nesse objetivo. Existem diveros programas que possibilitam a visualização de informações referentes a eventos, mas nem sempre expõem a informação da melhor maneira. E principalmente, não são destinados primordiamente a auditoria de sistema no *Microsoft Windows*®.

Para o caso específico do *Microsoft Windows*®, que possui o código fechado, onde a auditoria de sistemas vem desabilitada por padrão e os objetos do sistema não são monitorados, a tarefa de obter informações de utilização do computador se torna mais difícil. É sabido que é possível ativar a auditoria e selecionar manualmente quais objetos (pastas e arquivos) serão monitorados, contudo, não é tão simples e um tanto trabalhoso.

Pensando nestas limitações que outras *softwares* apresentam e em auxiliar na obtenção e manipulação de informações, Observador foi criado. Com ele, os processos manuais foram automatizados e tornaram-se transparentes ao usuário. O foco central é a melhor forma de visualizar a informação. E claro, sem modificá-la sob qualquer situação. O Observador contou com várias formas de obter os eventos do *Microsoft Windows*®, com mais uma forma de representá-los (tabela e gráficos), algumas maneiras de filtrar até que se possa achar uma informação específica se for assim desejado e como adicional, poder ter as informações em outros tipos de arquivos ou até impressas em papel.

Apesar do objetivo principal ter sido alcançado, o Observador conta com algumas limitações de robutez e funcionalidades pendentes. Algumas sugestões para aprimoramentos em versões posteriores do Observador podem ser vistas a seguir:

- Possibilitar a conexão e obtenção de relatórios via rede também no *Microsoft Windows*® *XP*;
- Adicionar suporte a exportação em mais formatos de arquivo, como .doc/.docx ou os arquivos de eventos .evt/.evtx (após a realização da filtragem, por exemplo);
- Otimizar o Observador para torná-lo mais robusto para trabalhar com grandes volumes de dados;
- Flexibilizar a inserção de filtros, adicionando filtros personalizados.

9. REFERÊNCIAS

[1] Audio forensics. http://dammahumrecordingstudio.com/services/audio-editing-services-2/audio-forensics/. [Online; acessado 16-Fevereiro-2015].

[2] Poder Legislativo Brasileiro. *Código Penal Brasileiro*. 1940.

[3] Microsoft Corporation. Access control entries. https://msdn.microsoft.com/en-us/library/windows/desktop/aa374868[Online; acessado 22-Fevereiro-2015].

[4] Microsoft Corporation. Access control lists. https://msdn.microsoft.com/en-us/library/windows/desktop/aa374872[Online; acessado 22-Fevereiro-2015].

[5] Microsoft Corporation. Auditpol. https://technet.microsoft.com/pt-br/library/cc731451(v=ws.10).aspx. [Online; acessado 22-Janeiro-2015].

[6] Microsoft Corporation. How to: Query for events. https://msdn.microsoft.com/en-us/library/bb671200(v=vs.90).aspx. [Online; acessado 22-Fevereiro-2015].

[7] Microsoft Corporation. Log parser 2.2. https://technet.microsoft.com/en-us/scriptcenter/dd919274.aspx. [Online; acessado 22-Janeiro-2015].

[8] Microsoft Corporation. Overview of the .net framework. https://msdn.microsoft.com/en-us/library/zw4w595w[Online; acessado 25-Janeiro-2015].

[9] Microsoft Corporation. Namespaces (guia de programação em c#). https://msdn.microsoft.com/pt-br/library/0d941h9d.aspx, 2013. [Online; acessado 25-Janeiro-2015].

[10] Václav Dajbych. Overview of the .net framework. http://www.codeproject.com/Articles/680100/Overview-

of-the-NET-Framework, 2013. [Online; acessado 25-Janeiro-2015].

[11] Phillip D. Dixon. An overview of computer forensics. *Computação*, 2005.

[12] Kaydee Dwarak. Asp.net show - slide show. http://www.asptreeview.com/show/show_slideshow.aspx. [Online; acessado 19-Abril-2015].

[13] Heige kein. Setacl – automate permissions and manage acls. https://helgeklein.com/setacl/, 2015. [Online; acessado 02-Abril-2015].

[14] FSPro Labs. Event log explorer^{TM} for windows event log management. http://www.eventlogxp.com/, 2013. [Online; acessado 24-Janeiro-2015].

[15] Viva Linux. O que é um log? e para que serve? http://www.vivaolinux.com.br/artigo/SysLog-Sistema-de-log-do-Linux, 2005. [Online; acessado 16-Fevereiro-2015].

[16] Alexandro Manotti. *Curso Prático de Auditoria de Sistemas*. Editora Ciência Moderna, 2010.

[17] Sandro Melo. *Computação Forense com Software Livre*. Alta Books, 2009.

[18] Joachim Metz. Windows xml event log (evtx). *Computação*, 2014.

[19] Margaret Rouse. Windows event log. http://searchwindowsserver.techtarget.com/definition/Windows-event-log, 2013. [Online; acessado 22-Janeiro-2015].

[20] Ultimate Windows Security.com. Windows security log events. https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/Default.aspx, 2015. [Online; acessado 02-Abril-2015].

[21] David Solomon and Mark Russinovich. *Windows Internals*, volume 1. Microsoft Press, 6 edition, 2012.

[22] David Solomon and Mark Russinovich. *Windows Internals*, volume 2. Microsoft Press, 6 edition, 2012.

[23] TZWorks. Windows event log parser(evtwalk). https://www.tzworks.net/prototypes.php?proto_id=25, 2015. [Online; acessado 24-Janeiro-2015].

[24] Info Wester. Sistema de arquivos ntfs. http://www.infowester.com/ntfs.php, 2011. [Online; acessado 16-Fevereiro-2015].

[25] Peng Wang Yongjian Lou. Automated event log file recovery based on content characters and internal structure. *Computação*, 2009.