UMA FERRAMENTA PARA MONITORAMENTO DA AÇÃO DE *RANSOMWARES* EM SISTEMA OPERACIONAL WINDOWS.

Ramon Rodrigues Moreira, Daniel Faustino Lacerda de Souza

Resumo: Este trabalho descreve uma ferramenta gráfica para monitoramento e visualização das ações de *ransomwares* no sistema operacional Windows e a configuração de um ambiente de testes seguro para a realização de experimentos com códigos maliciosos. Buscamos apresentar uma ferramenta capaz de trazer maior clareza e facilidade na compreensão de ataques de *ransomwares* observando parâmetros específicos do sistema de arquivos. A ferramenta é composta por três monitores que observam tempo de uso do disco, calculam entropia de arquivos de chamariz e observam diretórios específicos do sistema operacional para acompanhar as alterações realizadas pelo código malicioso. Os resultados obtidos são satisfatórios e motivam a criação de estratégias para o aperfeiçoamento de monitores existentes, bem como a implementação de novos monitores em versões futuras da ferramenta.

Palavras-chave: *ransomware*; monitoramento; código malicioso.

1. INTRODUÇÃO

Com o avanço dos sistemas computacionais surgiram também as preocupações com segurança computacional. Há uma preocupação cada vez maior com a segurança dos dados armazenados e gerenciados por sistemas de computador devido aos riscos de invasões e capturas de informações sigilosas crescentes nos últimos anos. Um tipo de ataque que ganhou notoriedade nos últimos anos foi o de *ransomwares*, códigos maliciosos que capturam informações computacionais criptografando seu conteúdo e exigindo pagamentos para possibilitar o resgate dos arquivos. Os *ransomwares* agem de forma silenciosa, não sendo perceptíveis na maioria das vezes até que todos os arquivos tenham sido criptografados. Perceber a ação de um código malicioso desta natureza é uma tarefa complexa que requer muita atenção a detalhes que podem ser observados no comportamento do sistema operacional. Entender como o programa malicioso funciona é um importante passo para aprender a evitá-lo e combatê-lo. Como observar estes detalhes durante a ação de um *ransomware*?

Os ataques com códigos maliciosos têm crescido ao longo dos anos, tendo grande destaque no uso de *ransomwares*, que tem movimentado grandes quantidades de dinheiro no pagamento do resgate de arquivos, chegando a incríveis somas de US$ 200 milhões por ano [1]. Kharaz [2] aponta dados ainda mais alarmantes: uma única família de *ransomwares* – CryptoWall 3.0 – mostrou-se altamente lucrativa aos criminosos causando danos estimados em US$ 325 milhões. São incríveis somas de dinheiro desperdiçadas com pagamentos a criminosos por órgãos privados e públicos. Relatórios divulgados pela McAfee mostra um preocupante crescimento no número de *ransomwares* desde 2012, mostrando ser um crescente desafio à segurança cibernética, tendo piorado a partir de 2013 e crescendo a cada ano [3].

Diante deste cenário é fundamental conhecer melhor esse tipo de *malware*. É necessário entender seu funcionamento, quais atividades ele executa e como o faz. É preciso entender qual a sua ação no sistema atacado e que procedimentos são realizados para a criptografia e deleção dos arquivos do usuário. Uma boa maneira de entender o funcionamento seria com o código fonte do programa malicioso, mas como na maioria das vezes eles não podem ser localizados é necessário partir para o campo da observação das ações do *malware* em execução. Para tanto, é preciso dispor de ferramentas de observação de características específicas do sistema operacional, que normalmente não estão disponíveis de forma acessível e clara. Somente com conhecimento deste tipo de ação é possível traçar estratégias eficientes para sua proteção, detecção e interrupção de possíveis ataques com o mínimo de danos possível aos arquivos. Um dos ambientes mais propícios para a observação e estudo é o campo acadêmico, entretanto não há ferramentas adequadas conhecidas para a observação e monitoramento eficaz deste tipo de programa, que disponha de um ambiente seguro de execução dos códigos maliciosos e uma interface de observação que permita observar o que acontece no sistema operacional durante um ataque.

A construção de uma ferramenta com este propósito pode trazer grandes benefícios para o desenvolvimento acadêmico e de aprendizado sobre o tema aos alunos de graduação de áreas relacionadas, proporcionando um ambiente seguro e adequado para estudo deste tema. Desta feita, o presente trabalho tem como objetivo geral

desenvolver uma ferramenta para monitorar características do sistema operacional para facilitar a compreensão de ataques de *ransomware*. Os objetivos específicos são desenvolver uma interface de fácil uso, que seja clara e simples de entender e que possa ser utilizada com segurança para propósitos didáticos, para que a ação do código malicioso possa ser estudada com segurança.

O restante deste artigo é composto por fundamentação teórica, trabalhos correlatos, aspectos gerais da ferramenta, desenvolvimento e conclusões.

2. FUNDAMENTAÇÃO TEÓRICA

Os sistemas computacionais têm ganhado cada vez mais notoriedade e destaque em várias áreas de nossa sociedade. Computadores são usados mais do que nunca nos mais diversos cenários, sejam com papéis de gerenciamento e automatização em empresas ou simples entretenimento em casas. Desde nossos computadores até os nossos relógios (no caso dos relógios inteligentes) são dotados de sistemas computacionais responsáveis por nos proporcionar facilidades nunca vistas antes. São televisores, geladeiras, celulares, carros e vários outros produtos além dos computadores domésticos e de trabalho com os quais temos contato todos os dias e que são dotados de componentes tecnológicos.

Com este grande aumento não apenas no número de computadores, mas também nas suas possibilidades de aplicações surgiu a preocupação com a segurança dos sistemas computacionais. Há um grande crescimento no cuidado com a maneira com que os dados gerados por todos esses dispositivos serão armazenados, em como esses dados serão acessados e em quem poderá acessá-los.

Um tipo de ataque que ganhou muito destaque nos últimos anos foi o de *ransomware*. Trata-se de um código malicioso que busca penetrar em nossos dispositivos através de falhas de segurança, restringindo o acesso de seus donos de direito aos seus arquivos e cobrando o pagamento de um resgate para sua devolução. Normalmente não é possível perceber a ação dos *ransomwares* até que a notificação seja dada pelo próprio código malicioso ao usuário, momento em que já não é possível acessar qualquer um dos arquivos. Esta é a última fase de um ataque.

Segundo Brewer [4] podemos dividir a ação de um *ransomware* em cinco fases distintas: infecção, execução, remoção de backups, encriptação dos arquivos e a notificação ao usuário. Estas fases são ilustradas na Figura 1 abaixo.

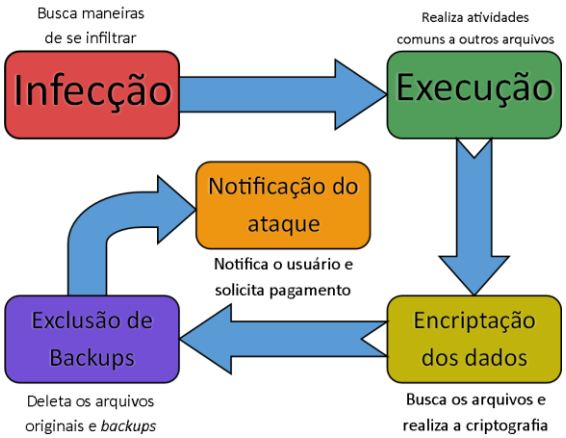

Figura 1. Ilustração do ciclo de vida de um ataque de *ransomware*. (Autoria própria)

Os prejuízos deste tipo de ataque podem ser gigantescos. Em 2017 tivemos um dos maiores ataques deste tipo já registrados, quando o código malicioso conhecido como WannaCry infectou centenas de milhares de sistemas em mais de uma centena de países [5]. A ação afetou o funcionamento de hospitais, empresas de telecomunicações e órgãos governamentais deixando o mundo em estado de alerta. O evento teve grande destaque também em nosso país, tendo notoriedade em telejornais. Em março de 2019 relatórios apontaram o Brasil como segundo país com mais ataques de *ransomwares*, somando cerca de 10% de todas as ocorrências no mundo durante o período observado, ficando atrás apenas dos Estados Unidos [6]. São números preocupantes que nos mostram a importância de aprender a identificar e combater esta atividade criminosa.

O combate aos *ransomwares* tem se mostrado uma atividade desafiadora. Variações dos códigos são desenvolvidas com grande frequência, tornando cada vez mais difícil sua identificação através dos softwares *antimalware*. Uma das técnicas usadas pelos programas de defesa para identificar códigos maliciosos é a sua assinatura. Nesta técnica de detecção, o antivírus observa a assinatura do programa, que é uma sequência de bytes em determinado arquivo do código para definir se é ou não malicioso [7]. Entretanto esta técnica de detecção se mostra ineficiente quando aplicada em *malwares* novos ou capazes de alterar sua forma a cada execução [7], caso de muitos dos novos *malwares*. Observar a execução do código em um ambiente controlado é uma maneira de entender o que o ataque faz e como ele se desenvolve. Detectar um ataque não é uma tarefa trivial, mas certamente as atividades realizadas durante um ataque terão semelhanças perceptíveis através de detalhes que podem ser

observados através do sistema operacional. Ter um ambiente de testes seguro para sua execução, aliado com uma ferramenta capaz de observar detalhes específicos do computador hospedeiro pode ser de grande ajuda para aprender como o ataque se desenvolve e traçar estratégias de detecção e combate.

3. TRABALHOS CORRELATOS

Ao longo dos últimos anos vários trabalhos têm sido desenvolvidos a respeito do desenvolvimento, comportamento, variações e detecção de códigos maliciosos, especialmente *ransomwares*. Esta seção apresenta os trabalhos que tiveram mais relevância para o planejamento e desenvolvimento de nossa ferramenta.

Ahmadian e Shahriari [3] desenvolveram importante trabalho em seu estudo para detecção de *ransomwares* apontando que o acesso a bibliotecas de criptografia é um parâmetro para sua detecção. Durante um ataque várias bibliotecas de sistema responsáveis por rotinas de criptografia de dados podem ser solicitadas pelo código malicioso. Observar os processos que solicitam estas informações pode fornecer dados úteis para a detecção de ataques. Além deste parâmetro, o acesso anormal a arquivos e pastas em um curto espaço de tempo também é um bom indicador. Durante a execução de suas rotinas o *ransomware* realiza a busca e leitura de todos os dados do usuário, realizando vasta varredura em documentos, imagens, músicas e o que mais possa ser útil para garantir que nada seja esquecido. Este processo que garante que nada reste ao usuário sem a devida encriptação também facilita sua identificação. O uso das rotinas de leitura e escrita de disco rígido são drasticamente ampliadas durante o processo.

Scaife et al [8] mostraram que normalmente há um afunilamento de tipos de arquivos durante um ataque de *ransomware*, além de uma grande movimentação no disco causado pelo excesso de leitura, escrita e deleção de dados em um curto espaço de tempo. Também apontam atividades que auxiliam na detecção de códigos maliciosos como a mudança na assinatura dos arquivos. Apenas este não é um sinal de ataque, mas serve como um alerta. O trabalho aponta três classes de atividades comuns durante um ataque: (a) os arquivos são abertos, têm seu conteúdo lido e sobrescrito, (b) eles são movidos para outro diretório (como uma pasta temporária), tem seus dados encriptados no diretório original e depois são deletados ou (c) os dados são lidos, é criado um novo arquivo com as informações encriptadas e depois os originais são deletados ou substituídos pela cópia criptografada. Scaife et al também apontam que mesmo as cópias de segurança do sistema operacional não são suficientes para garantir preservação dos dados, pois durante um ataque elas também são deletadas de forma permanente.

Moore [1] mostra a utilidade de se apropriar de técnicas de *honeypot* para detecções mais breves de um ataque. Manter uma observação profunda e constante sobre os arquivos de um computador não é prático, nem tampouco aconselhável, dados os custos computacionais do processo. A sugestão é a utilização de arquivos de chamariz com nomenclaturas que os qualifiquem como primeiros arquivos localizados durante os ataques. Assim é possível manter uma vigilância muito mais efetiva e detalhada sobre arquivos que podem ser espalhados em pontos estratégicos do sistema de arquivos e ajudam a identificar um possível ataque.

Kharaz [2] sugere a interceptação das requisições geradas pelos processos ao sistema de arquivos para armazenamento dos seus dados como forma de identificar a origem de ações de encriptação de dados, realizando o armazenamento destas informações. Também mostra a utilidade de calcular a entropia dos arquivos que recebem solicitação de leitura de conteúdo e comparar com o valor após sua sobrescrita. Em caso de ataques de *ransomware* a encriptação das informações presentes no arquivo proporciona uma grande mudança no valor da entropia quando comparado ao valor original.

Os trabalhos mencionados trazem importantes observações quanto ao comportamento de ransomwares. O trabalho proposto neste artigo se diferencia dos correlatos em seus objetivos. Enquanto os autores citados trabalham na identificação e detecção de um *ransomware*, nossa proposta é de nos sujeitar propositalmente a um ataque para observar a ação do código malicioso. Para tanto, utilizamos características apontadas pelos trabalhos correlatos.

4. ASPECTOS GERAIS DA FERRAMENTA

A ferramenta foi pensada com o objetivo de auxiliar no estudo de *ransomwares*, buscando apresentar em tempo real dados objetivos e de fácil compreensão sobre a ação do código malicioso em um ambiente controlado. O monitor desenvolvido não tem o objetivo de impedir ataques ou identificá-los, e sim o de acompanhar rotinas específicas do sistema operacional que são frequentemente afetadas por estes ataques. Através destas rotinas específicas, esperamos possibilitar uma maior compreensão da rotina de ataque de diversas famílias de *ransomware*.

Para tanto a arquitetura foi desenhada de maneira que permitisse a ação simultânea de mais de um monitor, funcionando de maneira independente uns dos outros. Também foi pensado para facilitar a adição ou remoção de monitores sem a necessidade de reestruturação completa do projeto. Neste trabalho, três características foram escolhidas para serem observadas: o tempo de uso do disco, um monitor de alterações de diretório e uma rotina de cálculo de entropia. A escolha destas características se deu pela clareza com que estas informações podem ser exibidas aos usuários de nossa ferramenta. Pensando nestas características três monitores foram implementados. São eles o monitor de disco, o monitor *honeypot* e o monitor de entropia. Além dos monitores, existem rotinas

adicionais que compõem o projeto e funcionam como uma espécie de esqueleto. Temos um controlador de monitores responsável por centralizar as definições de cada monitor e realizar sua inicialização. Há também um controlador de notificações que recebe as notificações geradas pelos monitores de *honeypot* e entropia e um controlador de gráficos responsável por interpretar as informações obtidas pelo monitor de disco e exibir as informações através de um gráfico em tempo real. Na Figura 2, apresenta-se um fluxo simplificado da aplicação em suas rotinas de comunicação com o sistema operacional.

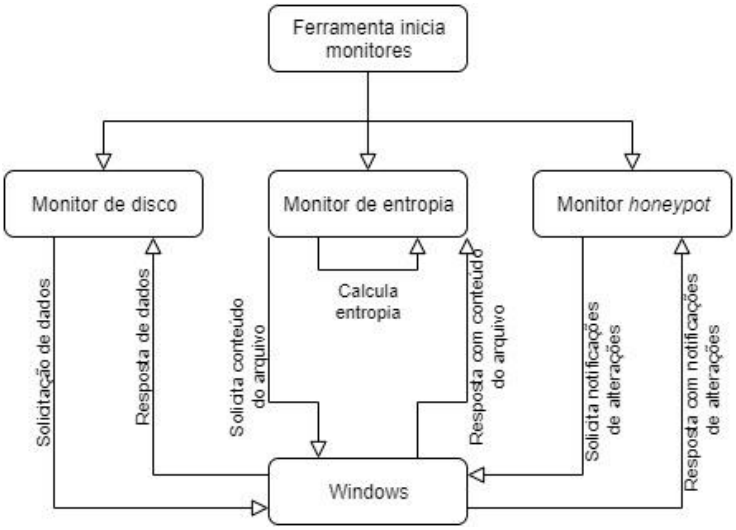

Figura 2. Esquema de comunicação com o Sistema Operacional. (Autoria própria)

O monitor de discos é o responsável por observar o tempo de uso do disco rígido do computador e exibir as informações de maneira gráfica em tempo real. Para isto, faz uso de rotinas do sistema operacional e do controlador de gráficos. O monitor realiza a captura das informações repetidamente, com um intervalo de tempo de um segundo entre as coletas, enquanto a ferramenta estiver em execução. Estes dados são repassados ao controlador de gráficos que, a cada nova recepção de dados, realiza a atualização do gráfico na tela do monitor em forma de gráfico de área para proporcionar fácil visualização das informações coletadas.

Adaptações de técnicas de *honeypot* são utilizadas no monitor homônimo, que funciona em conjunto com o monitor de entropia. A técnica conhecida por *honeypot* ("pote de mel" em tradução livre) consiste em uma espécie de armadilha. Um alvo é deixado desprotegido propositalmente para atrair um ataque em um ambiente observado e controlado. No caso do monitor em questão uma pasta específica criada no sistema de arquivos é monitorada constantemente, gerando notificações a cada interação do sistema de arquivos com ela. Sempre que esta pasta, um de seus arquivos ou subpastas é alterada ou excluída o monitor envia uma notificação ao controlador de notificações que fará a exibição na tela principal da ferramenta. Sempre que um arquivo ou pasta é criada dentro do diretório observado uma notificação também será lançada.

Por fim o monitor de entropia trabalha em conjunto com o monitor *honeypot* utilizando cálculos matemáticos para ajudar a identificar alterações em um arquivo. A entropia é uma maneira de medir o grau de incerteza de uma informação. Na prática, este monitor utiliza este grau de incerteza para identificar alterações em um arquivo de texto. Um arquivo de texto chamariz é mantido dentro da pasta observada pelo monitor *honeypot* e nomeado de forma a ser um dos primeiros arquivos listados pelo sistema de arquivos do sistema operacional. Seu nome é iniciado por uma série de caracteres especiais como a cerquilha. Dentro deste arquivo há um texto que tem sua entropia calculada pelo monitor através da fórmula de Shannon [9], que consiste em calcular o somatório das probabilidades de ocorrência dos símbolos em um arquivo e é dado pela Equação 1 abaixo.

$$-\sum pi * log_2(pi), \text{ onde } pi \text{ é a probabilidade do i-ésimo termo} \quad (1)$$

Para que a entropia possa ser calculada, uma vez que o arquivo é aberto pelo monitor é realizada uma varredura em seu conteúdo para contabilizar a ocorrência de cada caractere e calcular sua probabilidade. Em posse dessas informações, a entropia é calculada. Dentro de intervalos regulares, a probabilidade e a entropia são novamente calculadas e, enquanto não houver alteração da entropia temos a garantia de que o arquivo não foi criptografado. A partir do momento em que o valor de entropia muda sabemos que o conteúdo do arquivo foi alterado.

Quando a primeira entropia é calculada o monitor envia uma notificação ao controlador de notificações com o valor encontrado que é exibido na tela inicial da ferramenta. A cada novo cálculo o valor é comparado com o anterior. Sempre que for encontrado um valor diferente uma nova notificação é lançada. Caso o arquivo não possa ser encontrado também é lançada uma notificação.

A ferramenta é iniciada através de sua interface gráfica, por meio da qual são iniciados os monitores que permanecerão em atividade até que a aplicação seja encerrada pelo usuário. Durante sua atividade as informações

observadas são constantemente exibidas ao usuário através de contêineres dispostos na tela da aplicação, proporcionando fácil leitura e identificação das informações.

5. DESENVOLVIMENTO

A estruturação da ferramenta é feita a partir de sua interface gráfica, já que é através dela que todos os monitores são iniciados. O comando para o início do monitoramento é feito pelo próprio usuário a partir do menu Arquivo da tela da ferramenta, como é possível ver na Figura 3. Quando recebe o comando de início a ferramenta cria o controlador de notificações e inicia os monitores em *threads* (linhas de execução diferentes) simultâneas, de maneira que monitores, controlador de notificações e interface gráfica possam trabalhar em conjunto, mas com a autonomia necessária para que nenhum precise esperar a conclusão da tarefa de outro para que possa ser executado.

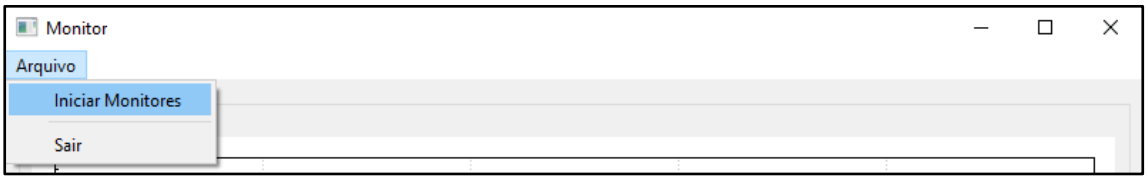

Figura 3. Menu de inicialização dos monitores. (Autoria própria)

Cada monitor funciona de maneira independente dos outros e da própria aplicação, realizando suas próprias rotinas de monitoramento, criação de suas próprias notificações e suas próprias rotinas de comunicação com o sistema operacional. Isso proporciona a ferramenta a modularização necessária para uma fácil revisão, adição e remoção de monitores, sendo preciso realizar apenas as devidas adequações na interface gráfica e desenvolvimento e/ou revisão do(s) monitor(es) desejado(s).

O desenho da ferramenta foi desenvolvido tendo o objetivo de apresentar os dados de forma clara e de fácil leitura. Para tanto optou-se por apresentar todos os dados ao usuário simultaneamente, cada um deles em seu próprio contêiner. Temos então na parte superior da tela um espaço amplo para apresentação contínua dos dados obtidos através do monitor de disco e logo abaixo dois espaços menores reservados para receber as notificações dos monitores *honeypot* e de entropia, como podemos observar na captura de tela apresentada na Figura 4.

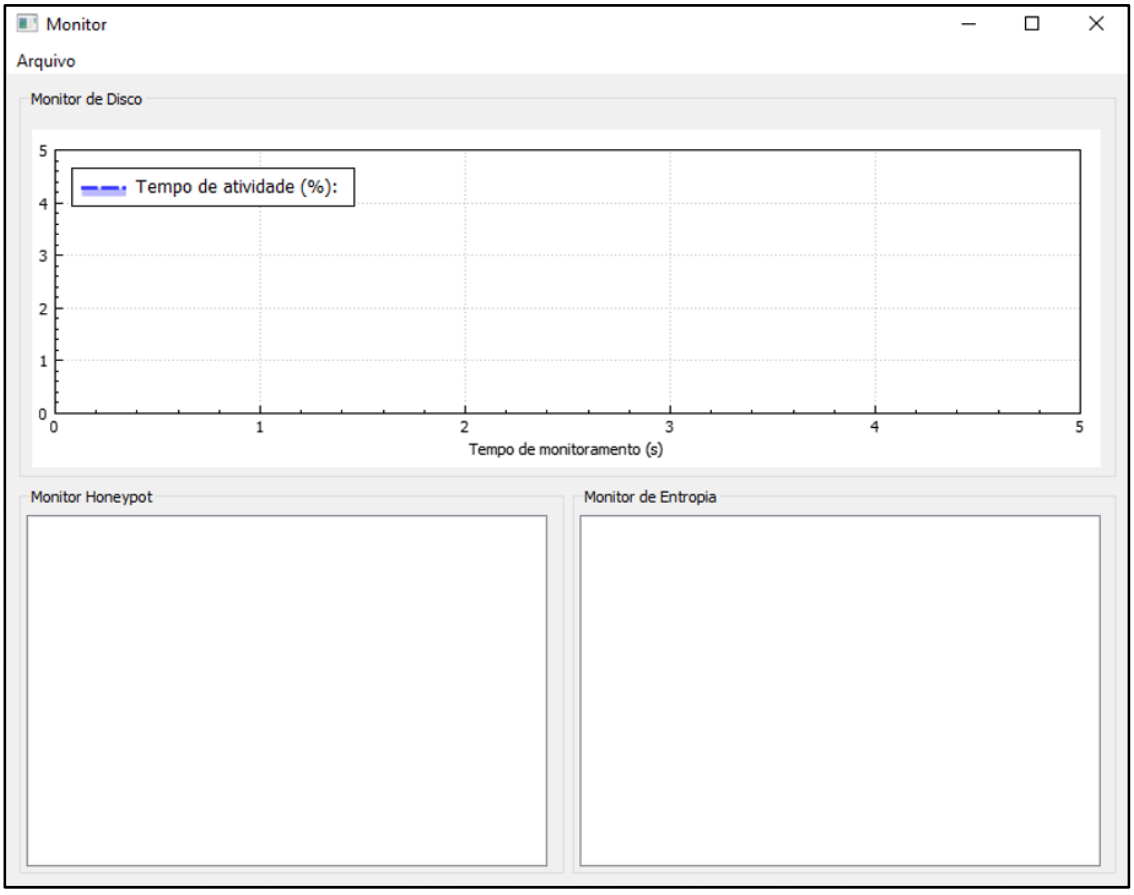

Figura 4. Captura de tela da aplicação. (Autoria própria)

A solução foi construída para utilização no sistema operacional Windows 10 (versão de 64 bits), tendo em

mente ser um dos sistemas operacionais atuais mais comum nos computadores domésticos e profissionais. Desta maneira somos capazes de observar os impactos que um ataque proporciona a um computador comum de nosso cotidiano.

5.1. Ambiente de Testes

A utilização da ferramenta em quaisquer ambientes não oferece nenhum risco aos arquivos pessoais do computador em que ela é executada, entretanto a execução de um código malicioso fora de ambientes de testes controlados certamente oferecerá grandes riscos à segurança de seu utilizador. Por esta razão foi preparado além da ferramenta de monitoramento um ambiente de testes adequado.

Foi utilizado o software Oracle VM VirtualBox 6.0 [10], através do qual foi criada uma máquina virtual. A máquina virtual em questão trata-se de um computador virtual criado a partir da ferramenta e que pode ou não possuir comunicação efetiva com o computador no qual a máquina está sendo executada. Para nosso propósito de executar códigos maliciosos com segurança o ambiente foi configurado incapaz de realizar acesso à internet e qualquer tipo de comunicação com o sistema de arquivos do dispositivo em que é executada, prezando pela segurança de quem quer que o utilize.

Para simular a capacidade de processamento necessária o ambiente foi configurado com 2 GB de memória RAM e 50 GB de espaço disponível em disco. O sistema operacional utilizado é o Windows 10 (versão de 64 bits).

Para suportar a execução do código malicioso foi necessária a instalação da versão de avaliação do WinRAR archiver [11], utilizado para armazenamento do *malware*, além do instalador da aplicação desenvolvida. Nenhuma outra aplicação foi necessária para a realização dos testes de comportamento e monitoramento do sistema. Por ter recebido bastante notoriedade adotamos o *WannaCry* [12] como nosso *malware* de testes.

Para a realização dos testes o sistema recebeu 512 cópias de documento RTF contendo 136 KB de texto Lorem Ipsum [13], 128 cópias de imagens diversas (para simular o arquivamento de fotos de usuários) e 8 cópias de um álbum de músicas disponibilizado gratuitamente pelo artista.

Após a adequada configuração da máquina virtual foi criado um *snapshot* do sistema, uma espécie de "fotografia digital" do estado da máquina que possibilita sua recuperação plena a este estado do sistema, garantindo que os testes possam ser realizadas quantas vezes forem necessárias sem a necessidade de realizar a criação e configuração de novos ambientes a cada execução.

5.2. Aspectos de Implementação

Para que os monitores fossem capazes de possuir comunicação mais rápida e direta com o sistema operacional – que é o fornecedor das informações obtidas pela solução – foi utilizada a linguagem de programação C++, tendo em vista o fato de que esta é a mesma linguagem utilizada para construção das bibliotecas de apoio utilizadas.

O ambiente de desenvolvimento escolhido foi a IDE (*Integrated Development Environment*, Ambiente Integral de Desenvolvimento em tradução livre) Qt Creator [14]. Esta ferramenta proporciona maior facilidade e agilidade no processo de construção de interfaces gráficas a partir da linguagem de programação escolhida oferecendo resultados visualmente mais agradáveis em um menor espaço tempo. Para prover maiores facilidades no desenvolvimento do programa, bem como a compreensão de seu código, a opção escolhida foi desenvolver o projeto a partir do modelo de orientação a objetos.

A ferramenta é composta por sete classes no total. São três classes de monitores (uma para cada um deles), uma classe para modelagem das notificações, uma classe para o controlador de notificações, uma classe para a modelagem do gráfico de disco e uma para gerenciamento da interface gráfica e monitores. A estas classes somam-se os arquivos de cabeçalho das classes e o arquivo de design da interface gráfica.

O gráfico de disco e os dois campos de notificação são objetos da classe principal da ferramenta que faz o gerenciamento da aplicação. Para que seja possível realizar as alterações dos campos de notificação dos monitores a classe de gerenciamento de notificações possui acesso aos campos de notificação e realiza todas as edições necessárias nestes campos. Os monitores de entropia e *honeypot* montam sua notificação (a partir do modelo definido na classe de notificação) e realizam chamada à função *notificar* da classe de gerenciamento de notificações. Um processo semelhante é realizado pelo monitor de discos, com a diferença de que ele apenas encaminha os dados observados do disco, não utilizando o modelo definido de notificação. Através do objeto de notificação recebido a classe de gerenciamento é capaz de observar qual monitor encaminhou a notificação e faz a devida inclusão da notificação ou atualização do gráfico quando é o caso do monitor de discos. A Figura 5 apresenta um diagrama de classes que representa a estrutura de nossa ferramenta.

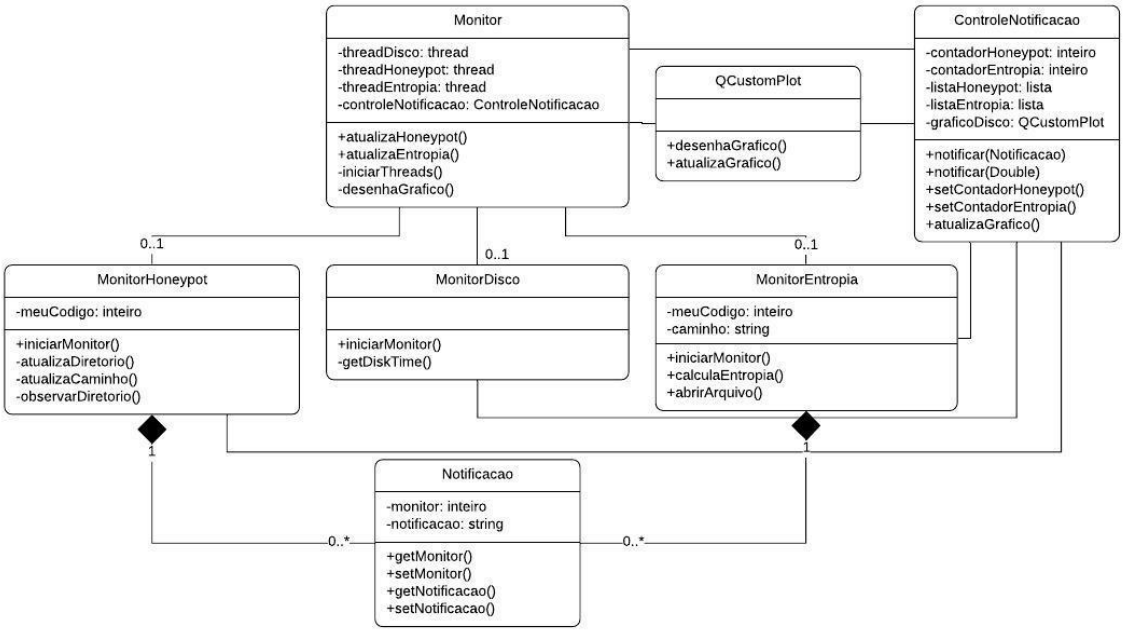

Figura 5. Diagrama de classes da ferramenta. (Autoria própria)

5.3. Monitor de Disco

O monitor de disco, diferente dos outros dois monitores, não apresenta campo de notificações, e sim um gráfico atualizado em tempo real que exibe o tempo de atividade do disco, seja em atividades de leitura ou escrita. Para o desenho do gráfico é utilizada a biblioteca *QCustomPlot 2.0.1* [15] disponibilizada pela IDE adotada no desenvolvimento do trabalho. A cada segundo uma nova consulta de atividade é realizada no disco e incluída no gráfico. O intervalo adotado foi de sessenta segundos, o que significa que apenas as últimas sessenta amostras são exibidas no gráfico. A atualização é realizada através da notificação encaminhada ao controlador de notificações da aplicação, que recebe os dados do monitor de disco e realiza o processo de atualização de gráfico e seu redesenho na tela da aplicação.

Para a realização das consultas o monitor utiliza as funções disponibilizadas pela API (*Application Programming Interface*, Interface de Programação de Aplicativos em tradução livre) do Windows. Através dela é possível acessar dados de contadores de desempenho do sistema operacional a partir da biblioteca *Pdh* [16], realizando sua captura e armazenamento para os fins desejados, que no nosso caso é a apresentação em forma de gráfico. É possível ver como é feita a principal rotina de realização do processo de coleta de dados a partir das listagens 1 e 2. Na Listagem 1, a consulta que será realizada ao sistema operacional através das funções presentes na biblioteca *pdh* é criada e testada. Com a consulta já programada é possível realizar as solicitações ao sistema. A consulta é realizada e retorna o resultado à variável *diskUsage*. A função de notificação do controlador de notificações é invocada com o valor obtido na consulta. A cada segundo (descrito como 1000 milissegundos no código) uma nova consulta é realizada e uma nova notificação é enviada.

Listagem 1. Trecho de código do monitor de disco. (Autoria própria)

```
controleNotificacao = controle;
query = nullptr;
PdhOpenQuery(nullptr, 0, &query);
Status = PdhAddCounter(query, L"\\PhysicalDisk(*)\\% tempo de disco", 0, &counter);
Status = PdhCollectQueryData(query);
if (Status != ERROR_SUCCESS)
{
    wprintf(L"\nPdhCollectQueryData failed with status 0x%x.", Status);
    EXIT_FAILURE;
}

double diskUsage;

while (true) {
    diskUsage = getDiskTime();
    controleNotificacao->notificar(diskUsage);
```

```
        Sleep(1000);
    }
```

Nota: Criação, teste e realização da consulta ao disco.

A coleta dos dados pode ser visualizada na Listagem 2, na qual podemos ver como funciona a função em que a consulta montada anteriormente é efetivamente realizada e retorna os dados para a geração da notificação.

Listagem 2. Trecho de código do monitor de disco. (Autoria própria)

```
double MonitorDisco::getDiskTime() {

   Status = PdhCollectQueryData(query);
   Status = PdhGetFormattedCounterValue(counter, PDH_FMT_DOUBLE, &CounterType,
&fmtCounter);

   double diskUsage = fmtCounter.doubleValue;
   //cout << "Valor na consulta: " << diskUsage << endl;

   return diskUsage;
};
```

Nota: Processo de coleta dos dados do disco.

5.4. Monitor de Entropia

O funcionamento do monitor de entropia pode ser dividido em quatro passos que se repetem em um intervalo de tempo estabelecido. Primeiro é necessário acessar o conteúdo do arquivo de chamariz. Esse processo é realizado através das funções de manipulação de arquivos disponíveis na biblioteca *fstream* [17]. O segundo passo é calcular a probabilidade de cada caractere presente no arquivo. A estratégia aplicada foi a criação de uma estrutura de dados contendo uma variável inteira para armazenar a contagem dos caracteres lidos e um vetor de inteiros com 127 posições para armazenar o número de ocorrências de cada caractere no conteúdo do arquivo. O número 127 foi escolhido por ser o mesmo número de caracteres que podem ser representados através da tabela *ASCII* [18], utilizada para padronizar a representação de letras, números e caracteres especiais nos sistemas computacionais. Isso nos garante que independente do conteúdo do arquivo, o sistema será capaz de realizar uma contagem apropriada de seus símbolos. Ao realizar a abertura do arquivo o monitor lê o seu conteúdo um caractere por vez, identificando seu código na tabela *ASCII* [18] e incrementando em um o valor da posição correspondente ao código da tabela no vetor de inteiros que armazena as ocorrências. Também é incrementando em um o valor do inteiro que guarda a contagem de caracteres do arquivo. Ao fim do processo teremos o número total de caracteres presentes no arquivo e a contagem da ocorrência de cada caractere, disposta no vetor de inteiros tendo a representação do código de cada caractere no índice do vetor. Esta solução nos fornece o que precisamos para calcular a entropia do arquivo. O terceiro passo é o cálculo da entropia que agora pode ser realizado, como mostrado na Listagem 3. A função que adapta a fórmula de Shannon [9] realiza uma varredura na estrutura de dados criada realizando a somatória da probabilidade de cada caractere (dado pela razão entre o número de ocorrências e total de caracteres do arquivo) com frequência diferente de zero e calculando sua entropia. Ao fim do processo a variável *entropia* possui o valor total da entropia do arquivo. O quarto passo é a notificação do valor no campo correspondente da interface gráfica.

Listagem 3. Trecho de código do monitor de entropia. (Autoria própria)

```
float MonitorEntropia::calcula_entropia(vetor_freq vetor) {
    float freq = 0;
    float entropia = 0.0;
    for (int i = 0; i < ASCII; i++) {
        if (vetor.cont_simbolos[i] > 0) {
            freq = (float)vetor.cont_simbolos[i] / (float)vetor.total;
            entropia += ((-1) * freq * log2(freq));
        }
    }

    return entropia;
}
```

Ao realizar o primeiro cálculo de entropia o monitor passa a repetir o processo a cada dois segundos. Considerando que um segundo seria pouco tempo para que fosse realizada uma alteração significativa no arquivo escolhemos o intervalo de dois segundos. Um intervalo de tempo maior resultaria em demora na notificação em

caso de alteração ou deleção do arquivo observado. Se um valor de entropia diferente do anterior for encontrado uma notificação é criada e encaminhada ao controlador de notificações que realizará a atualização no campo correspondente da interface gráfica da aplicação.

5.5. Monitor *Honeypot*

Para a construção do monitor *honeypot* foi utilizado o código de exemplo disponibilizado pela Microsoft [19] em sua documentação para desenvolvedores, nas sessões que tratam sobre o sistema de arquivos do Windows e seu gerenciamento. Adaptações foram realizadas para a adequação das rotinas ao modo de funcionamento da ferramenta, a exemplo da manipulação de tipos de variáveis utilizadas pelas bibliotecas de gerenciamento.

A partir da biblioteca *windows.h* temos acesso a funções que permitem que nossos programas criem estruturas do tipo *handle*, que pode ser descrito como um manipulador. Sozinha, esta estrutura não é útil, mas aliada às funcionalidades presentes na biblioteca *windows.h* é possível criar uma estrutura capaz de manipular funções de observação do sistema operacional, a exemplo da função *FindFirstChangeNotification* utilizada neste trabalho. Através dela é possível programar nossa ferramenta para receber notificações geradas pelo sistema operacional de acordo com as diretrizes desejadas. Para o propósito deste monitor (observação de diretório) são utilizadas as diretrizes *FILE_NOTIFY_CHANGE_FILE_NAME* e *FILE_NOTIFY_CHANGE_DIR_NAME*, responsáveis por observar mudanças em nomes de arquivos e mudanças no nome do diretório, respectivamente. Sua utilização pode ser vista na Listagem 4, que apresenta um trecho de código utilizado.

Listagem 4. Trecho de código do monitor *honeypot*. (Adaptado de [*])

```
// criação de função de notificação de diretório
dwChangeHandles[0] = FindFirstChangeNotification(
    diretorio, // diretório observado
    FALSE, // para não observar subdiretórios
    FILE_NOTIFY_CHANGE_FILE_NAME); // observa mudança no nome de arquivos

// saída de falha da função
if (dwChangeHandles[0] == INVALID_HANDLE_VALUE) {
    cout << endl << "ERRO: Primeiro observador falhou." << endl;
    ExitProcess(GetLastError());
}

// criação da função de notificação de subdiretórios
dwChangeHandles[1] = FindFirstChangeNotification(
    lpDrive, // diretório observado
    TRUE, // para observar subdiretórios
    FILE_NOTIFY_CHANGE_DIR_NAME); // observa mudança de nome de diretórios
```

A mudança no nome de um arquivo não ocorre apenas quando realizamos a operação de renomear. Quando um arquivo é criado ou deletado esta sua propriedade de nome também é alterada. Utilizando a diretriz *FILE_NOTIFY_CHANGE_FILE_NAME* é possível receber informação do próprio Windows sempre que um arquivo ou pasta do diretório especificado tem este atributo alterado. Algo similar acontece quando observamos diretórios. No momento da criação ou exclusão de um diretório o atributo responsável pelo armazenamento do nome também é alterado e notificado pela diretriz *FILE_NOTIFY_CHANGE_DIR_NAME*.

O *handle* da nossa ferramenta verifica constantemente o estado destas diretrizes e sempre que recebe retorno monta sua notificação que será enviada ao controlador de notificações para que as atualizações sejam realizadas no campo destinado ao monitor *honeypot* da nossa aplicação.

Na Listagem 5 é possível ver o processo de criação da notificação (semelhante ao monitor de entropia) em que é criado um fluxo de caracteres que será enviado através de um objeto do tipo *Notificacao* por meio da função *notificar* do controlador de notificações.

Listagem 5. Trecho de código da criação da notificação pelo monitor *honeypot*. (Autoria própria)

```
// criação da mensagem
stringstream ss;
ss << "Uma pasta foi criada ou deletada no diretorio " << converter_lptstr(lpDrive);
string note = ss.str();
// criação da notificação
Notificacao a{ meu_codigo, note };
controleNotificacao->notificar(a);
```

5.6. Resultados e Discussões

A ferramenta desenvolvida apresentou baixo custo operacional, mesmo com as frequentes consultas dos monitores. As operações de leitura têm baixo custo computacional e não impactam de maneira significativa o desempenho do sistema. Isso possibilita que a ferramenta permaneça em execução sem atrapalhar outras atividades do usuário, garantindo que a experiência de um usuário comum não seja impactada.

Na Figura 6 mostra-se uma captura de tela da ferramenta em execução em nossa máquina virtual, operando durante a execução de um *ransomware*. É possível observar que o sistema se comporta bem durante o ataque, não tendo seu funcionamento interrompido mesmo ao fim da fase de encriptação dos dados.

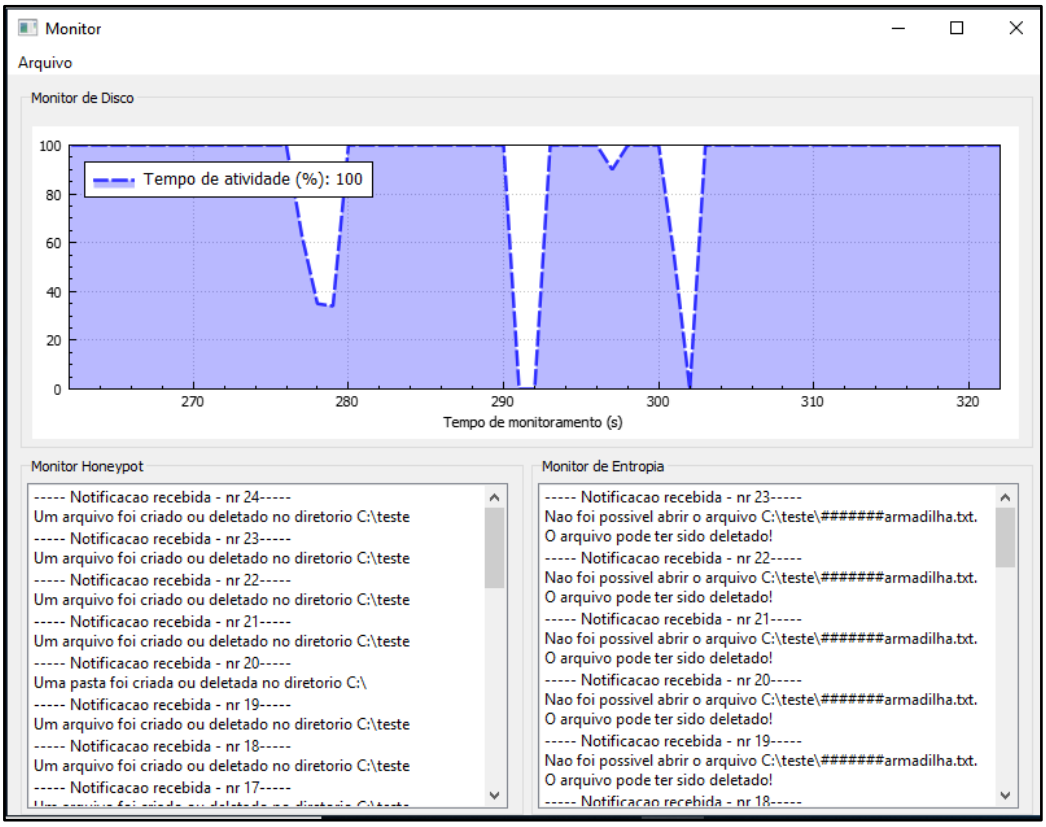

Figura 6. Captura de tela da ferramenta durante ataque. (Autoria própria)

O monitor de disco se mostra bastante eficiente, copiando muito bem o tempo de uso do disco. Este se mostrou um parâmetro significativo de um ataque, visto que nos experimentos realizados os ataques sempre alcançam 100% de uso do disco. É possível perceber rápidas oscilações que logo são superadas pelo total uso do sistema de arquivos para encriptação de dados e deleção dos arquivos originais.

O monitor *honeypot* também se mostrou bastante ativo, gerando um grande número de notificações durante a ação do código malicioso. O monitor capta bem as alterações realizadas no diretório observado do sistema de arquivos e todas as alterações, sejam elas de arquivos, pastas ou subpastas, são notificadas na ferramenta.

O monitor de entropia, por sua vez, não apresentou os resultados esperados. Em testes específicos deste monitor, realizando alterações no conteúdo do arquivo observado ele se mostrou eficiente e rápido, notificando quando a entropia calculada se difere da anterior e sempre que o arquivo não pode ser localizado, como podemos ver na Figura 7. Durante nossos experimentos de ataque em ambiente controlado, com nossa amostra de *ransomware* o cálculo da entropia não se mostrou útil, uma vez que o arquivo observado não chegou a ter seu conteúdo alterado. O procedimento realizado por nossa amostra de código malicioso cria uma cópia encriptada do arquivo e deleta o dado original, não sendo possível localizar o arquivo original após a ação do ataque sobre ele. No entanto, este resultado pode ser diferente quando o experimento for realizado com famílias diferentes de *ransomwares*, dado que podem agir de maneira diferente, realizando alterações de conteúdo no lugar de simples substituição.

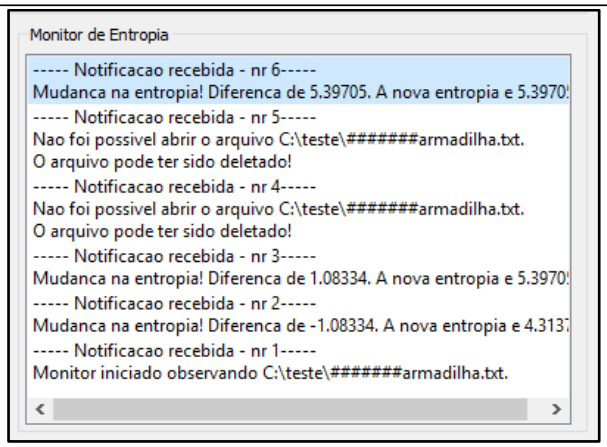

Figura 7. Testes do monitor de entropia. (Autoria própria)

6. CONCLUSÕES

Neste trabalho foi desenvolvida uma ferramenta para monitoramento das ações de *ransomwares* em computadores pessoais ou profissionais utilizando sistema operacional Windows 10 64 bits, sendo composta por interface gráfica, controlador de notificação e três monitores (disco, *honeypot* e entropia). Os monitores se relacionavam com o sistema operacional através das ferramentas disponíveis no Windows para desenvolvedores, com bibliotecas de acesso a dados de desempenho do sistema operacional. Também foi preparado um ambiente de testes seguro no qual a ferramenta poderia ser testada sem os riscos trazidos pela execução de códigos maliciosos, em especial os *ransomwares*. Este ambiente de testes foi preparado através do Oracle VM VirtualBox 6.0 [10], que nos proporcionou a segurança necessária para nossos experimentos.

A ferramenta desenvolvida se mostrou satisfatória nos testes realizados, não apresentando falhas durante a execução ou travamentos, mesmo durante a ação do código malicioso. O desempenho foi um fator bastante positivo. Mesmo realizando consultas frequentes ao sistema de arquivos e cálculos de entropia não foi possível perceber impacto significativo no desempenho do sistema. Foi possível perceber que a demanda por poder computacional da ferramenta é bastante baixa, permitindo sua aplicação em computadores e máquinas virtuais com baixo poder de processamento sem maiores transtornos à utilização natural do sistema operacional.

A ferramenta desenvolvida, ainda que necessitando de ajustes, se mostra promissora em seu objetivo, trazendo um comportamento e interface que favorecem sua utilização em ambientes experimentais. É possível utilizá-la como apoio didático, ajudando a perceber de forma clara a operação de um *ransomware* no sistema operacional, apresentando uma possibilidade maior de entendimento dos riscos e atividades do código malicioso.

Os monitores de disco e *honeypot* se mostraram bastante eficientes assimilando bem as operações realizadas pelo *ransomware* no sistema de arquivos. A observação do tempo de uso do disco apresentou os resultados esperados, exibindo de forma gráfica as variações do uso do disco em tempo real. O monitor *honeypot* também apresentou resultados bastante satisfatórios recebendo notificações das alterações do diretório observado sempre que arquivos e/ou pastas eram criados, alterados ou deletados. O ponto negativo foi o monitor de entropia, que embora tenha apresentado resultados satisfatórios de suas rotinas de cálculo de entropia se mostrou ineficiente durante as execuções de nossos experimentos. Este fato pode estar associado às rotinas desempenhadas pelo código malicioso experimentado (*WannaCry* [12]), que cria novas cópias encriptadas do arquivo e deleta as originais sem, de fato, realizar mudanças de seu conteúdo.

Nesta primeira versão, a ferramenta possui apenas três monitores que mantém suas observações centradas no disco e em seu sistema de arquivos. Para versões futuras, pretende-se ampliar o número de monitores, observando parâmetros de acesso a bibliotecas de criptografia, monitoramento de processos e serviços ocultos no sistema operacional e do *Volume Shadow Copy Service* - *VSS* (“Serviço de cópia de sombra de volume” em tradução livre), tecnologia presente no Windows para realização de *backups* automáticos e que geralmente são destruídos pelos *ransomwares*. Pretende-se também realizar o aperfeiçoamento do monitor *honeypot*, trazendo à tela informações mais específicas de quais arquivos e/ou pastas foram deletados, alterados ou criados.

7. REFERÊNCIAS BIBLIOGRÁFICAS

[1] MOORE, Chris. Detecting Ransomware with Honeypot techniques. In: 2016 Cybersecurity and Cyberforensics Conference, 2016, Amã, Jordânia. Anais... Amã: CCC, 2016. Disponível em: <https://ieeexplore.ieee.org/abstract/document/7600214>. Acesso em: 10 jul. 2019.

[2] KHARAZ, Amin et al. UNVEIL: A Large-Scale, Automated Approach to Detecting Ransomware. In: USENIX Security Symposium, 25., 2016, Austin, Estados Unidos. Anais... Austin: USENIX Security Symposium, 2016. Disponível em: <https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/kharaz>. Acesso em: 01 ago. 2019.

[3] AHMADIAN, M.M.; SHAHRIARI, H.R. 2entFOX: A Framework for High Survivable Ransomwares Detection. In: International ISC Conference on Information Security and Cryptology, 13., 2016, Teerã, Irã. Anais... Irã: Shahid Beheshti University, 2016. Disponível em: <https://ieeexplore.ieee.org/document/7736455>. Acesso em: 03 ago. 2019.
[4] BREWER, Ross. Ransomware attacks: detection, prevention and cure. Network Security, Volume 2016, Issue 9, 2016, p. 5-9. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1353485816300861>. Acesso em: 26 nov. 2019.
[5] RODRIGUES, Renato. Há um ano, WannaCry infectava mais de 200 mil sistemas. **Kaspersky**, 2018. Disponível em: <https://www.kaspersky.com.br/blog/um-ano-wannacry-ransomware/10282/>. Acesso em: 29 nov. 2019.
[6] GARRETT, Filipe. Brasil é o 2°país com mais ransomwares no mundo, revela pesquisa. **TechTudo**, 2019. Disponível em: <https://www.techtudo.com.br/noticias/2019/03/brasil-e-o-2pais-com-mais-ransomwares-no-mundo-revela-pesquisa.ghtml>. Acesso em: 12 nov. 2019.
[7] YEWALE, Abhijit; SINGH, Maninder. Malware detection based on opcode frequency. 2016 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT), Ramanathapuram, Índia 2016, p. 646-649. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7831719&isnumber=7831584>. Acesso em: 29 nov. 2019.
[8] SCAIFE, Nolen et al. CryptoLock (and Drop It): Stopping Ransomware Attacks on User Data. 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS), Nara, Japão, 2016, p. 303-312. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7536529&isnumber=7536347>. Acesso em: 09 ago. 2018.
[9] SHANNON, Claude Elwood. A mathematical theory of communication. The Bell System Technical Journal, vol. 27, no. 3, pp. 379-423, julho. 1948. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6773024&isnumber=6773023>. Acesso em: 04 ago. 2019.
[10] ORACLE VIRTUALBOX. **Downloads**. Disponível em: <https://www.virtualbox.org/wiki/Downloads>. Acesso em: 15 mai. 2019.
[11] WINRAR. **Download**. Disponível em: <https://www.win-rar.com/download.html?L=0>. Acesso em: 30 mai. 2019.
[12] MEDIAFIRE. **WannaCry by Rafael**. Disponível em: <http://www.mediafire.com/file/9f8fds9s3efg7so/WannaCry_by_Rafael.rar/file>. Acesso em 30 mai. 2019.
[13] LOREM IPSUM. **Lorem Ipsum**. Disponível em: <https://www.lipsum.com>. Acesso em 10 jun. 2019.
[14] QT. **Download**. Disponível em: <https://www.qt.io/download>. Acesso em 20 ago. 2019.
[15] QCUSTOMPLOT. **Download**. Disponível em: <https://www.qcustomplot.com/index.php/download>. Acesso em: 01 out. 2019.
[16] MICROSOFT DOCS. **Pdh.h header**. Disponível em: <https://docs.microsoft.com/en-us/windows/win32/api/pdh/>. Acesso em: 22 dez. 2019.
[17] CPLUSPLUS. **Reference**. Disponível em: <http://www.cplusplus.com/reference/fstream/fstream/>. Acesso em: 09 ago. 2019.
[18] ASCIITABLE. **ASCII**. Disponível em: <http://www.asciitable.com>. Acesso em 09 ago. 2019.
[19] MICROSOFT DOCS. **Obtaining Directory Change Notifications**. Disponível em: <https://docs.microsoft.com/pt-br/windows/win32/fileio/obtaining-directory-change-notifications>. Acesso em: 05 out. 2019.