ANÁLISE FORENSE E ENGENHARIA REVERSA APLICADAS À HOMOLOGAÇÃO DE SOFTWARE[1]

Gustavo de Castro Ventura

Resumo: A análise forense, entendida aqui como a observação de vestígios ou rastros sob uma ótica científica, compõe-se de procedimentos e técnicas que podem ser aplicados para descobrir os efeitos que determinado objeto em observação causa no contexto que ocupa. No âmbito da tecnologia da informação, e de posse de determinados conhecimentos, pode-se observar e entender melhor os efeitos causados por um software em um determinado sistema. A proposta deste trabalho é apresentar ferramentas que permitam descobrir possíveis funções ocultas em um software e que por ventura possam apresentar riscos para a segurança da informação. Como ponto de partida, serão apresentados conceitos iniciais sobre análise forense e engenharia reversa, apresentando a sua aplicação na observação de programas em execução. A finalidade é responder com segurança se o programa em análise é um potencial causador de problemas para uma organização.

Palavras-chave: Análise Forense Computacional. Engenharia Reversa. Homologação de Software.

1 INTRODUÇÃO

No ambiente corporativo, falhas, roubo ou fraude de informações podem trazer graves consequências (GEUS e NAKAMURA, 2007, p.49). Neste contexto, nota-se que a utilização indiscriminada de softwares nas estações de trabalho dos empregados pode ser a causadora de falhas que interrompam processos de trabalho e exponham dados

[1] Artigo apresentado como Trabalho de Conclusão do Curso de Especialização em Gestão de Segurança da Informação, da Universidade do Sul de Santa Catarina, como requisito parcial para a obtenção do título de Especialista em Gestão de Segurança da Informação.

confidenciais, entre outros problemas citados por Hoglund e McGraw (2006, p. 25), causando prejuízos à organização.

A utilização indiscriminada de softwares pode ser evitada através de um controle determinando que somente softwares aprovados, testados em ambiente apropriado, possam ser instalados para utilização. Através desse controle, denominado aqui de procedimento de homologação de software, pode-se verificar se determinado programa desconhecido pode representar um risco à segurança das informações da organização. Em outras palavras, se ele é um *malware*[2]. Essa verificação é feita através de uma análise do software, que, conforme um dos cenários descritos por Velho (2016, p. 414) trata de analisar:

> [...] um software já conhecido e utilizado pelos usuários, desenvolvido internamente ou adquirido de terceiros, no qual existe a suspeita de que esse programa execute funções secundárias e maliciosas sem o conhecimento dos administradores.

Dessa forma, observa-se a necessidade se colocar em análise os aplicativos utilizados no ambiente corporativo. A respeito dessa análise, Farmer e Venema (2007, p. 103), apontam que existem diversas maneiras de se estudar o comportamento de um programa, e dividem a análise em estática (quando um programa é estudado sem que seja executado) e dinâmica (onde o programa é analisado à medida em que sua execução ocorre). Os autores ainda apresentam a definição da análise dinâmica de "caixa preta":

> [...] na qual um sistema é analisado sem um conhecimento de seus aspectos internos. Os únicos itens observáveis são as entradas externas, as saídas e seus relacionamentos de tempo. [...] a análise de software do tipo "caixa preta" pode produzir resultados notavelmente úteis apesar de suas limitações aparentes.

Ainda conforme Farmer e Venema (2007), técnicas simples frequentemente podem ser suficientes para revelar o conteúdo malicioso de um software. A

[2] *Malware* é a abreviatura em inglês para *malicious software*, ou software malicioso.

desassemblagem[3] e a *descompilação*[4] são apenas para os mais dedicados (Ibid., p. 104). Como o uso dessas ferramentas pode ser mais demorado, é recomendável "primeiro utilizar ferramentas de nível mais alto para ampliar o código de interesse e então prosseguir a fim de analisar os detalhes mais delicados [...]" (Ibid., p. 120).

No presente trabalho, serão abrangidos conceitos de engenharia reversa que, conforme Venere (2013, p. 1) "pode ser descrita como o processo de analisar, compreender e identificar as funções de um equipamento, software ou dispositivo...". Ainda conforme o citado autor, a engenharia reversa:

> [...] pode envolver diversas técnicas, incluindo as mais simples, como a pura execução do software em um ambiente controlado, para identificar possíveis ações e modificações, até a análise do código de máquina, possibilitando o entendimento das funções do programa no nível mais baixo de código.

As técnicas de análise de *malware* apresentadas anteriormente (estática e dinâmica) são ainda subdivididas em básica e avançada por Honig e Sikorski (2012, p.2). Assim, pode-se apontar as seguintes definições, utilizadas em nosso estudo:

> a) análise estática básica, onde o aplicativo é examinado sem ser executado e sem a visualização de suas instruções;
> b) análise dinâmica básica, na qual o aplicativo é executado e seus comportamentos externos no sistema são observados;
> c) análise estática avançada, que envolve analisar o aplicativo em um *disassembler*, analisando as instruções que o software realiza;
> d) análise dinâmica avançada, onde um *debugger* é utilizado para observar o funcionamento interno do software analisado, em tempo de execução.

Honig e Sikorski (2012) também apresentam seu alerta quanto à necessidade de se configurar um ambiente seguro para a análise dinâmica de softwares desconhecidos, onde sejam diminuídos os riscos de danos em um ambiente de produção.

---

[3] A *desassemblagem*, ou desmontagem, é o procedimento executado por um programa conhecido como *disassembler*, que traduz a linguagem de máquina de um arquivo executável para a linguagem de montagem *Assembly*.

[4] *Descompilação* é, grosso modo, um procedimento que faz o inverso da compilação: traduz a linguagem de máquina de um executável para o código-fonte de alto nível que o gerou.

Nas páginas seguintes, serão apresentados os resultados do estudo desenvolvido tendo como norte esses princípios. Nele procura-se elencar, de maneira lógica, testes e validações que permitam descobrir exatamente o que um software é capaz de fazer em um sistema. A Seção 2 trata de conceitos relacionados à análise de softwares desconhecidos, enquanto que a criação de um ambiente para análise é discutida na Seção 3. A Seção 4, voltada para a coleta de dados, foi subdividida em 6 subseções, cada uma delas tratando de uma técnica específica para análise de software (escaneamento por antivírus, verificação de itens na inicialização do sistema operacional, verificação de funções do software sob análise, visualização de detalhes do processo, captura de eventos do processo e inspeção de tráfego de rede). Na Seção 5 é apresentada a análise dos dados coletados, e as conclusões estão relacionadas na Seção 6.

2 ANÁLISE DE SOFTWARES DESCONHECIDOS

Para que se possa elencar as técnicas de análise a serem adotadas em nosso procedimento de homologação, algumas premissas básicas se fazem necessárias.

Em primeiro lugar, as análises não serão efetuadas em softwares reconhecidamente maliciosos (*malwares*). Os objetos em análise são compostos por softwares que se apresentam como legítimos, cuja intenção não é causar danos e sim disponibilizar alguma função útil. Tal distinção é necessária para que limitemos o escopo das análises realizadas, evitando aprofundamentos desnecessários que possam tirar o caráter prático do procedimento.

Em segundo lugar, e levando em consideração o exposto no parágrafo anterior, os procedimentos apresentados neste trabalho foram ordenados seguindo a lógica proposta por Farmer e Venema (2007, p. 120): utilizar inicialmente as ferramentas de nível mais alto, e ir aprofundando conforme a necessidade. Dessa forma, pode-se logo nas primeiras análises receber resultados que tornem desnecessárias as análises posteriores, ou o contrário, indícios que levem a aplicar as etapas posteriores.

Em terceiro lugar, os procedimentos aqui descritos aplicam-se aos softwares desenvolvidos para o sistema operacional Windows 7 64 bits. Tais softwares podem ser distribuídos pelos desenvolvedores em duas maneiras distintas:

a) Um pacote de instalação, que deve ser então executado para que o software em si possa ser utilizado[5];
b) O próprio software, neste caso não exigindo instalação e já pronto para uso, normalmente distribuído na forma de um pacote compactado[6].

Faz-se necessária a distinção para a distinção, neste trabalho, de três momentos de análise: *análise pré-instalação*, *análise pós-instalação* e *análise de execução*.

a) A análise pré-instalação consiste em técnicas de análise estática básica que verificam ou o instalador do software, ou o software compactado;
b) A análise pós-instalação é composta por técnicas de análise estática básica, a serem utilizadas nos arquivos gerados após a instalação ou descompactação do software;
c) A análise de execução, por fim, é formada por técnicas de análise dinâmica básica, aplicáveis ao software em tempo de execução.

Dessa forma, a presente análise será iniciada "por alto", e serão realizados os aprofundamentos que se fizerem necessários nos momentos oportunos. Oportuno ainda destacar que não serão abordadas, neste trabalho, técnicas de análise estática ou dinâmica avançadas, que por sua complexidade e curva de aprendizagem exigidas (HONIG, 2012) fogem à praticidade requerida.

3 CRIAÇÃO DO AMBIENTE DE ANÁLISE

A análise de um software desconhecido deve ser precedida por alguns cuidados básicos. Conforme Farmer e Venema (2007, p. 104) ao invés de realizá-la em um ambiente onde a execução possa causar danos, é mais seguro executar tanto o programa quanto os procedimentos em uma *sandbox*[7]. Existem duas maneiras de se criar uma sandbox: através da montagem de um ambiente físico, ou da criação de máquinas virtuais (VELHO, 2016, p. 434).

[5] Podemos fazer referência a essa forma de distribuição como *software instalável*, ou ainda *instalador do software*.
[6] Podemos também utilizar o termo *software compactado* para remeter a um programa distribuído dessa maneira.
[7] Uma s*andbox*, ou caixa de areia, neste contexto, é um ambiente controlado e isolado para a execução de softwares.

A montagem de um ambiente físico é um ótimo meio para a análise de *malwares*, uma vez que estes podem possuir técnicas antianálise que detectam a execução em um ambiente virtual e modificam o comportamento (FARMER e VENEMA, 2007, p. 107). Porém, para o propósito do presente trabalho a utilização de uma máquina virtual é considerada medida suficiente, uma vez que os objetos de análise não serão softwares maliciosos em si, e sim softwares desconhecidos que possam vir a apresentar riscos para a organização.

As vantagens na utilização de uma VM (*virtual machine*, ou máquina virtual) envolvem, entre outras, menor custo que a montagem de um ambiente físico e facilidade na manipulação dos discos rígidos virtuais através da utilização de *snapshots*[8] (VELHO, 2016, p. 434). Para o ambiente de análise dos procedimentos descritos neste artigo, foi utilizado para virtualização o software VirtualBox (v. 5.2.8 r121009) contendo uma VM com a seguinte configuração:

a) 2 Processadores;
b) 4096 MB de memória principal;
c) Sistema operacional Windows 7 Professional SP 1 (v. 6.1, compilação 7601, Service Pack 1);
d) Plugin VirtualBox Guest Additions (v 5.2.8);

Demais softwares utilizados durante a análise serão descritos no decorrer do trabalho.

4 COLETA DOS DADOS

Para que os procedimentos pesquisados pudessem ser testados e a coleta de dados realizada, foi escolhido para análise o software Greenshot[9]. A versão utilizada foi a mais recente disponível na data de execução (Greenshot v. 1.2.10.6). O software é distribuído na forma de um pacote de instalação.

A partir de agora, serão descritas as técnicas e procedimentos adotados para análise, bem como os resultados encontrados.

---

[8] Em uma analogia, o *snapshot* é uma fotografia do sistema em um dado instante. Essa "fotografia" nos permite voltar ao mesmo estado em que o sistema se encontrava no momento em que o *snapshot* foi realizado.
[9] Disponível em http://getgreenshot.org/. Acesso em 07 maio 2018.

4.1 ANÁLISE PRÉ-INSTALAÇÃO (ESTÁTICA): ESCANEAMENTO POR ANTIVÍRUS

O escaneamento de um arquivo executável por diversos antivírus é considerado um bom ponto de partida por Honig e Sikorski (2012, p.10). Uma vez que o arquivo sob análise possa ter um código malicioso já identificado em alguma base de dados de assinatura de vírus, a utilização deste recurso pode possibilitar a rápida identificação do *malware*.

O site VirusTotal[10] é uma ferramenta online que realiza a varredura dos arquivos a ele enviados em mais de 60 antivírus mantidos por diferentes empresas. Conforme apontado por Velho (2016, p.422), os múltiplos diagnósticos permitem uma "[...] conclusão mais concreta quanto à qualificação do software, reduzindo casos de falsos positivos ou falsos negativos apresentados por alguns antivírus.".

Figura 1 - Resultado da varredura realizada pelo site VirusTotal no instalador do software Greenshot

Fonte: o autor, 2018.

Na Figura 1 pode-se visualizar os resultados da análise do instalador do software Greenshot. Como pode ser observado, a busca foi realizada por 64 diferentes antivírus, não tendo sido encontrado código malicioso por nenhum deles.

Apesar de a varredura por antivírus não constituir (a princípio) uma metodologia de informática forense ou engenharia reversa, considera-se imprescindível para os propósitos deste artigo, uma vez que permite descartar logo de início um software já reconhecidamente malicioso. Tendo sido afastada a ameaça de *malware* nessa

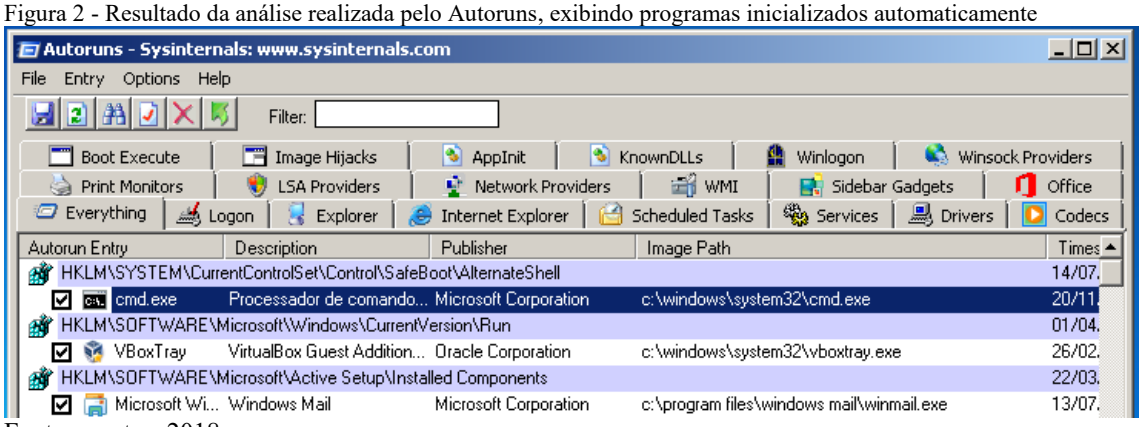
[10] Disponível em https://www.virustotal.com. Acesso em 15 mar 2018.

verificação, pode-se prosseguir com a instalação ou descompactação do software em questão para que sejam realizadas as análises posteriores.

4.2 ANÁLISE PÓS-INSTALAÇÃO (ESTÁTICA): VERIFICANDO A INICIALIZAÇÃO DO WINDOWS

Na inicialização do sistema operacional Windows, diversos locais podem ser utilizados para que outros programas sejam carregados automaticamente (VELHO, 2016, pág. 423). É importante que se verifique se a instalação ou execução de um programa inseriu entradas para carregar automaticamente algum executável, uma vez que o mesmo pode ser um fator de risco para a segurança, sendo executado sem o conhecimento do usuário. Uma ferramenta útil para identificar binários carregados na inicialização é o Autoruns[11], conforme apontado por Velho (Ibid., pág. 423), pois ao ser iniciado carrega todas as referências de inicialização no Windows, conforme pode ser observado pela Figura 2.

Figura 2 - Resultado da análise realizada pelo Autoruns, exibindo programas inicializados automaticamente

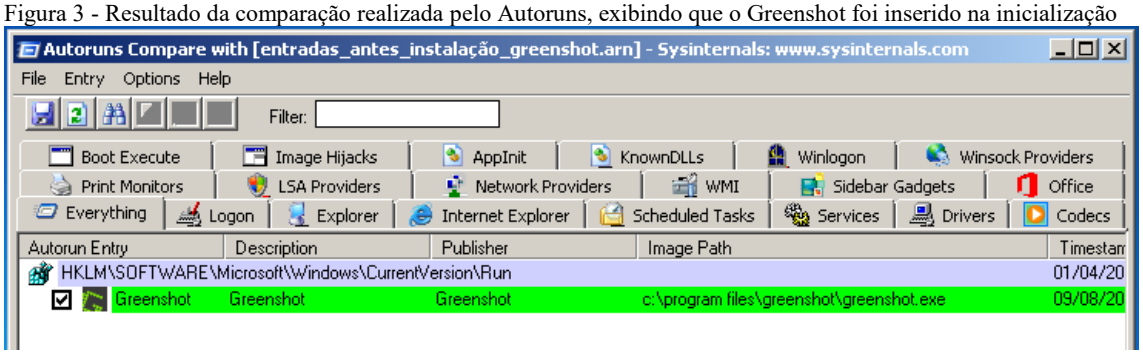


Fonte: o autor, 2018.

O Autoruns apresenta diversas funcionalidades, porém, o seu uso neste artigo se limita a identificar possíveis entradas inseridas após a instalação ou execução do software em homologação. Realiza-se esta tarefa salvando os registros das entradas de inicialização para um arquivo texto antes da instalação ou execução do programa. Tendo esse primeiro registro como linha base, pode-se instalar ou executar o software em questão, executar novamente o Autoruns e comparar os dois resultados encontrados, utilizando uma função do próprio software.

[11] O Autoruns faz parte da suíte de ferramentas *Sysinternals*, homologada pela Microsoft e disponível para download gratuito em https://docs.microsoft.com/pt-br/sysinternals/. Acesso em: 01 abr. 2018.

Figura 3 - Resultado da comparação realizada pelo Autoruns, exibindo que o Greenshot foi inserido na inicialização



Fonte: o autor, 2018.

Conforme o resultado trazido na Figura 3, a instalação do Greenshot acrescentou uma entrada de inicialização do Windows para que o próprio programa seja iniciado durante o carregamento do sistema operacional.

4.3 ANÁLISE PÓS-INSTALAÇÃO (ESTÁTICA): VERIFICANDO FUNÇÕES DO SOFTWARE

Analisando um arquivo executável é possível verificar o que um determinado programa é capaz de realizar no sistema operacional, conforme apontado por Honig e Sikorski (2012, p.18). Para que essa análise seja possível, entretanto, é necessária uma breve introdução à estrutura de um arquivo executável do Windows.

4.3.1 O formato de arquivos executável do Windows

Para que um programa possa ser executado em um sistema operacional, ele deve obedecer a certo padrão, sendo o PE (*Portable Executable*) o utilizado no Windows (VENERE, 2013, p.37). Segundo Velho (2016, p.428), o formato PE:

> [...] é uma estrutura de dados presente em determinados arquivos binários (executáveis, DLLs, *object code*, etc.) que contém informações necessárias para o carregamento e gerenciamento do código executável durante sua inicialização e funcionamento.

Um arquivo PE possui, em sua estrutura, um campo padronizado conhecido como cabeçalho (*PE Header*). O cabeçalho contêm parâmetros que definem como o resto do arquivo deve ser interpretado (as diferentes seções de um arquivo executável), conforme Pietreck (1994).

Para os propósitos deste trabalho, é importante notar que uma dessas seções do arquivo executável armazena informações sobre as bibliotecas e funções importadas e

exportadas pelo programa. Essas bibliotecas são utilizadas por programadores como uma forma de reaproveitamento de código (VELHO, 2016, p. 430). As bibliotecas geralmente estão presentes em arquivos DLL e contém funções que podem ser importadas para utilização por um executável.

O sistema operacional Windows disponibiliza para utilização um grande conjunto de DLLs que podem ser importadas pelos programas. Dessa forma, é realizado o reaproveitamento de código, livrando o programador de desenvolver grande parte das funções, que podem então ser padronizadas pela Microsoft – o que também colabora para manter a uniformidade do sistema operacional. E conforme Honig e Sikorski (2012, p. 18), a Microsoft mantém uma excelente documentação das funções do Windows, disponibilizadas na biblioteca MSDN (*Microsoft Developer Network*)[12].

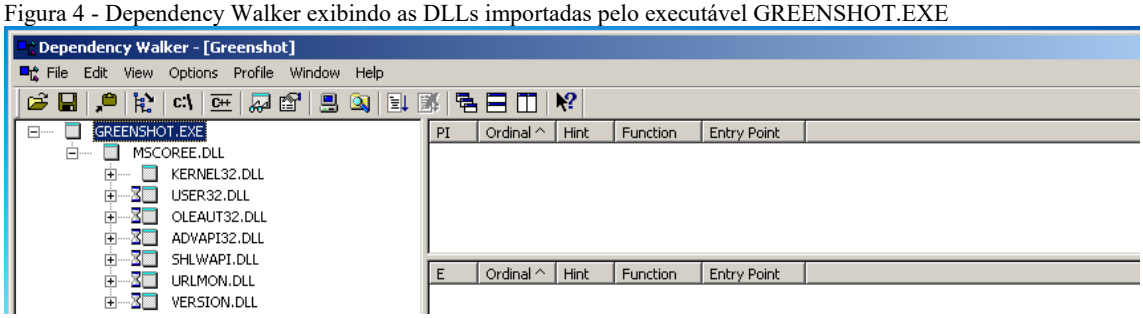
A lista de bibliotecas e funções importadas por um software provê informações valiosas. Baseado nas funções que um software importa do sistema operacional, é possível ter uma ideia das funcionalidades do mesmo sem a necessidade de executá-lo. Por exemplo, se um programa importa a função *URLDownloadToFile*, pode-se supor que em algum momento de sua execução ele tentará se conectar à internet, realizar o download de um arquivo e salvá-lo em disco (HONIG e SIKORSKI, 2012, p. 16).

4.3.2 Utilizando o Dependency Walker para visualizar as importações

O software Dependency Walker[13] é um utilitário capaz de fazer varreduras em arquivos que seguem o formato PE, exibindo as ligações existentes entre o executável analisado e as bibliotecas do qual ele depende. Em outras palavras, ele apresenta todas as funções de biblioteca externas importadas presentes no cabeçalho PE (VELHO, 2016, p. 431).

[12] Informações detalhadas sobre as bibliotecas e funções disponibilizadas pelo Windows podem ser consultadas no portal MSDN pelo endereço https://msdn.microsoft.com/en-us/library/windows/desktop/ff818516%28v=vs.85%29.aspx. Acesso em 02 abr. 2018.
[13] Disponível em: http://www.dependencywalker.com/. Acesso em 02 abr. 2018.

Figura 4 - Dependency Walker exibindo as DLLs importadas pelo executável GREENSHOT.EXE

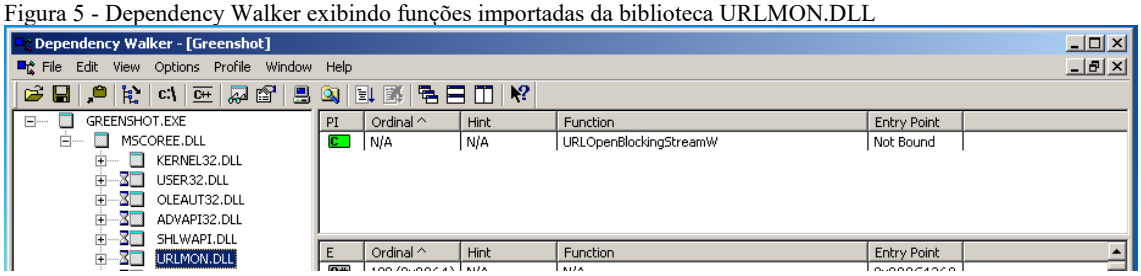


Fonte: o autor, 2018.

Pela Figura 4 observa-se que o executável Greenshot, identificado nas etapas anteriores como um executável que inicializa junto ao Windows, carrega uma série de DLLs do sistema operacional. Com uma consulta ao já citado repositório de informações sobre a API do Windows, pode-se verificar, por exemplo, que a utilização de funções da biblioteca ADVAPI32.DLL indica que o programa irá em algum momento manipular o registro do sistema operacional, ou que a importação da USER32.DLL é um bom indício de que se trata de um programa que utiliza elementos de interface gráfica.

A análise pode ser ainda mais aprofundada ao se clicar nas DLLs exibidas. Por exemplo, ao selecionar URLMON.DLL, obtém-se o resultado trazido pela Figura 5.

Figura 5 - Dependency Walker exibindo funções importadas da biblioteca URLMON.DLL



Fonte: o autor, 2018.

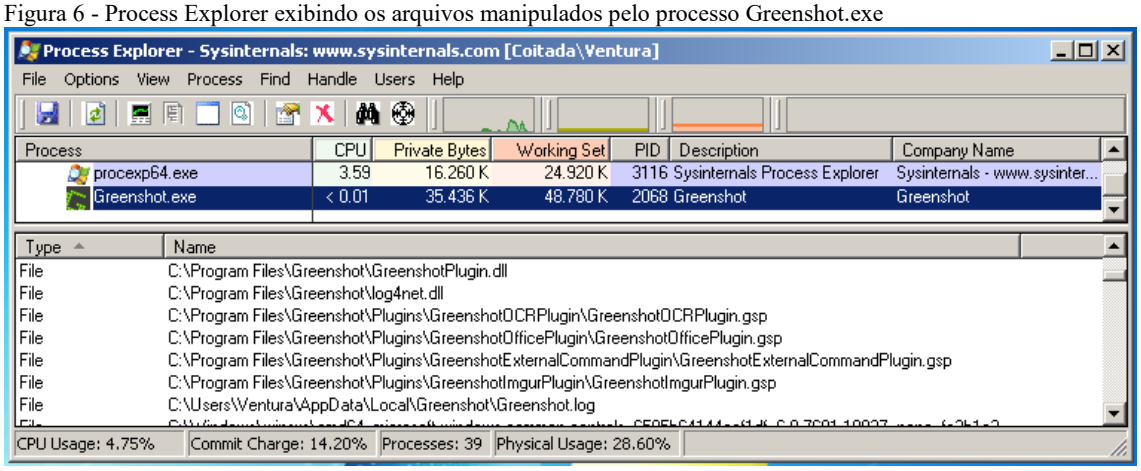
Ao clicar em qualquer uma das bibliotecas, o sistema irá exibir no painel as funções específicas daquela biblioteca que são importadas pelo executável. No resultado exibido na Figura 5, tem-se que o arquivo executável em análise importou para utilização a função *URLOpenBlockingStreamW*, da biblioteca URLMON.DLL.

Consultando a documentação a respeito da função[14], obtém-se a informação de que a mesma é capaz de criar um objeto do tipo *stream*[15] que irá receber o conteúdo baixado de uma determinada URL. A função só retornará quando todo o conteúdo da URL for baixado da internet. Em outras palavras, a presença dessa função nos resultados mostra, sem a necessidade de execução, que em algum momento o software Greenshot tentará se conectar a internet para realizar o download de um arquivo.

4.4 ANÁLISE PÓS-INSTALAÇÃO (DINÂMICA): VISUALIZANDO DETALHES DE PROCESSOS

A partir desse momento, o objetivo das análises será voltado ao levantamento de informações sobre o programa em execução (processo). Inicialmente, serão verificados detalhes sobre o processo e, para esse fim, será utilizado o software Process Explorer, da suíte *Sysinternals*[16]. Ao executá-lo, e selecionando o processo Greenshot.exe, podem-se verificar os *handles*[17] e DLLs carregadas pelo executável, conforme os resultados presentes na Figura 6.

Figura 6 - Process Explorer exibindo os arquivos manipulados pelo processo Greenshot.exe



Fonte: o autor, 2018.

Dessa forma, são identificados diretórios, arquivos, chaves de registro, etc. que foram abertos pelo executável selecionado. Algo que fuja ao esperado para o software

[14] Disponível em: https://docs.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/platform-apis/ms775127(v=vs.85). Acesso em: 04 abr. 2018.
[15] Um objeto do tipo *stream* representa um fluxo de dados ou texto em formato binário. Disponível em: https://msdn.microsoft.com/pt-br/library/office/jj250065.aspx. Acesso em: 04 abr. 2018.
[16] Disponível em: https://docs.microsoft.com/en-us/sysinternals/downloads/process-explorer. Acesso em: 15 abr. 2018.
[17] "Handles são recursos do sistema operacional que estão sendo manipulados por um processo durante sua execução" (VELHO, 2016, p. 436).

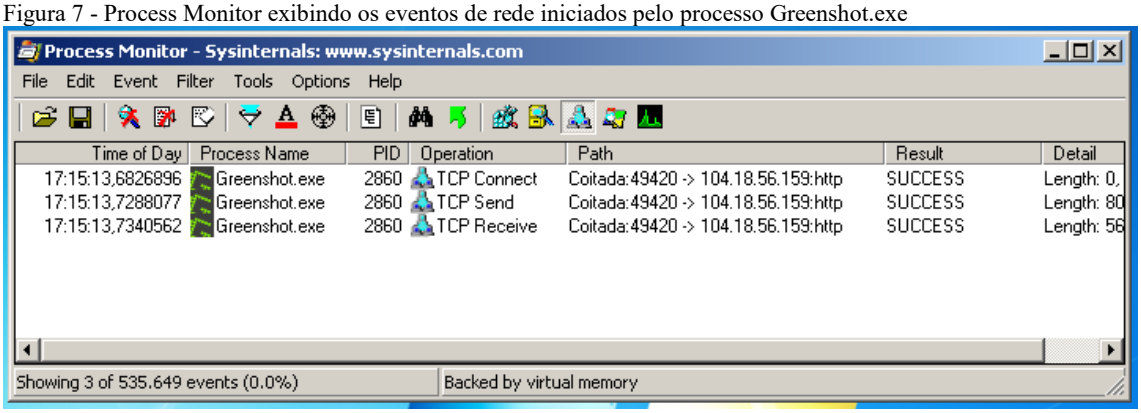
sob análise pode indicar um software suspeito. Além disso, o Process Explorer é capaz de identificar DLLs carregadas pelo processo em tempo de execução, que não são identificadas na análise estática realizada pelo Dependency Walker (VELHO, 2016, p. 436).

4.5 ANÁLISE PÓS-INSTALAÇÃO (DINÂMICA): CAPTURANDO EVENTOS DE PROCESSOS

Para verificar os eventos realizados no sistema operacional pelos processos em um determinado período de tempo, o software *Process Monitor*[18] será utilizado, tendo a capacidade de capturar quatro classes de eventos realizados por um processo no Windows: atividades no registro, atividades no sistema de arquivos, atividades de rede e atividades envolvendo processos e threads (Ibid., p. 438).

Devido à quantidade de eventos que ocorrem em um sistema operacional com diversos processos carregados, é importante que se realize a configuração adequada nos filtros, permitindo, por exemplo, que somente eventos realizados por um determinado processo ou de determinado tipo sejam mostrados.

Figura 7 - Process Monitor exibindo os eventos de rede iniciados pelo processo Greenshot.exe



Fonte: o autor, 2018.

Na Figura 7 pode-se observar resultados onde o Process Monitor foi configurado para exibir somente os eventos de rede relacionados ao PID[19] do processo Greenshot.exe. Nota-se que o processo inicia um processo de conexão TCP para o endereço 104.18.56.159, realizando o envio e recebimento de dados. Somente

[18] Também integra a suíte Sysinternals e está disponível em: https://docs.microsoft.com/en-us/sysinternals/downloads/procmon. Acesso em: 15 abr. 2018.
[19] PID ou Process ID é um número inteiro positivo atribuído pelo sistema Windows para identificar unicamente cada processo que está em execução.

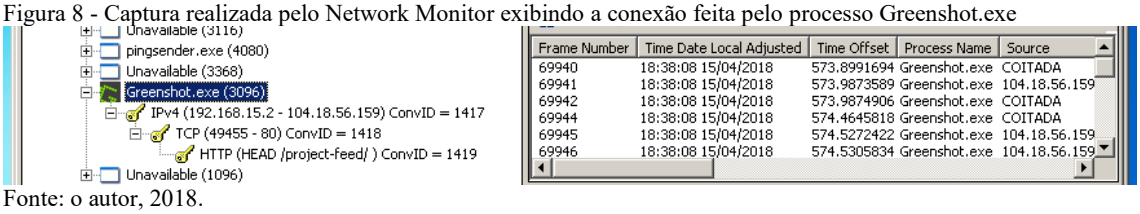
aprofundando nossa análise será possível saber o que foi transmitido ou recebido durante essa conexão.

4.6 ANÁLISE PÓS-INSTALAÇÃO (DINÂMICA): INSPECIONANDO O TRÁFEGO DE REDE

Após os indícios levantados nas Seções 4.3.2 e 4.5, é natural que se avance no processo de análise, buscando identificar os detalhes da operação de rede que foi realizada. Para realizar a captura dos dados transmitidos, é necessário que se faça uso dos chamados *sniffers* de rede. Velho (2016, p. 439) cita que apesar do software Wireshark ser o mais popular, em sistemas Windows é recomendável a utilização de um *sniffer* da própria Microsoft, chamado Network Monitor (NetMon)[20], citando como diferencial do mesmo a simplicidade em realizar a análise de tráfego para um processo específico.

Utilizando o Network Monitor durante a execução do software objeto da análise, pode-se então averiguar em maiores detalhes quais informações são transmitidas ou recebidas. Na Figura 8, vê-se em destaque o tráfego de rede gerado pelo processo Greenshot.exe. A partir dele, percebe-se que a conexão foi iniciada pelo processo na porta 49455, com destino ao servidor 104.18.56.159 na porta 80 (porta HTTP padrão). Por alto, infere-se que o processo se conectou a um servidor externo, utilizando o método HEAD do protocolo HTTP para solicitar informações sobre a página localizada em */project-feed/*.

Figura 8 - Captura realizada pelo Network Monitor exibindo a conexão feita pelo processo Greenshot.exe

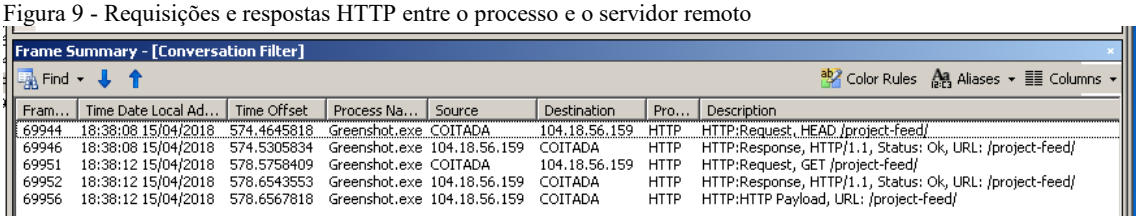


Fonte: o autor, 2018.

Indo mais a fundo, pode-se inspecionar o conteúdo dos pacotes HTTP que foram transmitidos entre processo e servidor. Através da Figura 9 podemos observar as requisições e respostas HTTP que ocorreram entre o processo e o servidor remoto. Pode-se notar que inicialmente o processo solicita os metadados da página */project-feed/*, recebe a resposta e a seguir realiza a requisição do conteúdo completo.

[20] Disponível em: https://www.microsoft.com/en-us/download/details.aspx?id=4865. Acesso em 15 abr. 2018.

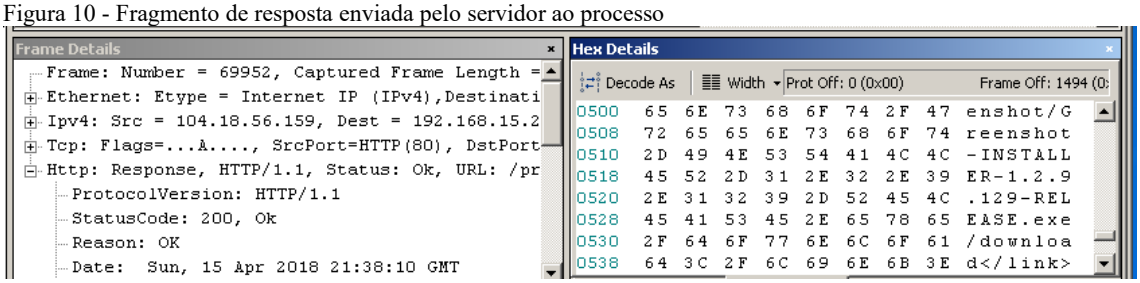
Figura 9 - Requisições e respostas HTTP entre o processo e o servidor remoto



Fonte: o autor, 2018.

Se for exibido o conteúdo da última resposta enviada pelo servidor, verifica-se que se trata de uma página contendo informações sobre versões e links para download do próprio Greenshot, conforme os resultados apresentados na Figura 10.

Figura 10 - Fragmento de resposta enviada pelo servidor ao processo



Fonte: o autor, 2018.

Com isso em mente, e sabendo que certos softwares possuem recursos para verificar e baixar automaticamente novas versões, infere-se que a operação realizada pelo processo provavelmente se trata de uma busca por versões mais recentes. Ele inicialmente verifica se há uma versão mais atual da página e, se houver, faz o download da mesma.

A partir desse ponto, ele pode fazer uma checagem da versão instalada com a versão mais recente disponível e, uma vez que na etapa mostrada na Seção 4.3.2 conseguimos identificar a importação de uma função para download de arquivos, ele provavelmente poderia realizar essa tarefa.

5 ANÁLISE DOS DADOS COLETADOS

A simples utilização de diversas ferramentas para a coleta de evidências não é suficiente para que se possa dizer se determinado software oferece riscos ou não. Os dados coletados exigem uma análise baseada nos conhecimentos de que o analista pode dispor.

Assim, os dados levantados (juntamente com a sua interpretação) poderiam constar, por exemplo, em um laudo expedido com a finalidade de comprovar (ou não) a

idoneidade de determinado software. A título de amostra, a partir dos dados coletados, é possível comprovar que:

a) O instalador do software em questão foi aprovado por 64 mecanismos antivírus;
b) Ao ser instalado, o mesmo insere uma entrada para que seja automaticamente iniciado no momento em que o computador é ligado, ou seja, sem que haja a intervenção do usuário;
c) Entre as bibliotecas do sistema operacional importadas, verifica-se a presença daquelas responsáveis pela manipulação de chaves de registro do Windows, além da utilização de uma biblioteca voltada para a utilização de elementos gráficos (ou seja, o programa provavelmente utiliza uma interface com janelas);
d) Destaca-se também a utilização de uma função específica para download de arquivos, o que poderia causar preocupações em alguns cenários e que estende a análise para a inspeção do tráfego de rede realizado pelo programa;
e) Durante a execução, observa-se que o processo realmente abriu uma conexão com um servidor na internet e, através da inspeção do tráfego de rede, constatou-se que se tratava do download de uma simples página de conteúdo.

6 CONCLUSÕES

A busca por resquícios digitais é uma interessante maneira através da qual podemos descobrir os reais efeitos da execução de um software. Através dos vestígios levantados, podemos analisar e verificar quais deles podem se tornar evidências que corroborem os laudos a serem feitos. No contexto abordado neste trabalho, tais evidências podem indicar que se trata de um software legítimo ou não, dessa forma autorizando ou não sua utilização.

As ferramentas aqui apresentadas possuem diversas funcionalidades, e claro está que não pudemos nos aprofundar na utilização de seus recursos. Além disso, constatamos também que a utilização simples e pura de ferramentas não é suficiente. Sem o conhecimento adequado, até mesmo o mais completo software utilizado em uma análise se torna ineficiente.

Cada um dos programas apresentados possui um grande leque de funções a serem exploradas, porém, seu domínio exige muito mais do que as páginas limitadas de um artigo tem a oferecer. Nossa intenção não foi esgotar as possibilidades de uso dos softwares, e sim apresentá-los para que possam ser estudados com mais profundidade pelos leitores entusiastas.

É importante notarmos também que o conhecimento do próprio analista é parte fundamental nestes testes. Tal conhecimento é composto tanto pelos recursos a que ele pode recorrer, quanto pela expertise profissional do mesmo. A experiência e conhecimentos adquiridos pelo analista ao longo de sua carreira podem facilitar a execução das análises.

Voltando às técnicas utilizadas, tivemos que nos ater principalmente àquelas denominadas como sendo de *caixa-preta*, ou seja, inspecionamos somente os efeitos externos observáveis do software em análise. Conforme apresentado anteriormente, a análise avançada, envolvendo a utilização de *debuggers*, descompiladores ou *disassemblers*, exige a utilização de conhecimentos que ultrapassam a proposta deste artigo.

Considerando a existência de softwares com código fonte aberto, é possível que a análise seja estendida para além de técnicas denominadas de *caixa-preta*. Porém, tal proposta deve ser alvo de cuidadosa consideração, uma vez que a inspeção de códigos pode demandar mais tempo que o disponível. Dado o caráter prático para o qual foi voltado este trabalho, talvez sua aplicação não seja a mais recomendada. Novamente recomendamos a análise *top-down*, ou seja, devemos sempre começar a buscar por evidências em níveis mais altos e realizar os aprofundamentos que se fizerem necessários.

Concluindo, gostaríamos de ressaltar que o trabalho ora proposto pode ser complementado por instruções que orientem a utilização de outras ferramentas de análise, ou até mesmo de apresentação mais completa das ferramentas aqui presentes. Outra possibilidade é estender para além das técnicas básicas aqui descritas, apresentando por exemplo a criação de ambientes físicos voltados para a análise, ou a utilização de *debuggers* para monitorar as instruções de um programa passo a passo

durante a execução de alguma função suspeita. A análise forense nos fornece variadas possibilidades.

REFERÊNCIAS

FARMER, Dan; VENEMA, Wietse. **Perícia Forense Computacional**: Teoria e Prática Aplicada. São Paulo: Pearson Prentice Hall, 2007.

GEUS, Paulo Lício de; NAKAMURA, Emílio Tissato. **Segurança de Redes em Ambientes Cooperativos**. São Paulo: Novatec Editora, 2007.

HOGLUND, Gred; MACGRAW, Gary. **Como Quebrar Códigos**: A Arte de Explorar (e Proteger) Software. São Paulo: Pearson Makron Books, 2006.

HONIG, Andrew; SIKORSKI, Michael. **Practical Malware Analysis**: The Hands-On Guide to Dissecting Malicious Software. San Francisco: No Starch Press, 2012.

PIETREK, Matt. **Peering Inside the PE**: A Tour of the Win32 Portable Executable File Format. 1994. Disponível em: <https://msdn.microsoft.com/en-us/library/ms809762.aspx>. Acesso em: 01 abr. 2018.

VELHO, Jesus Antônio. **Tratado de Computação Forense**. Campinas, SP: Millenium Editora, 2016.

VENERE, Guilherme. **Engenharia Reversa de Código Malicioso**. 1ª Ed. Rio de Janeiro: RNP/ESR, 2013. Disponível em: <https://pt.scribd.com/document/127786652/Engenharia-Reversa-de-Codigo-Malicioso>. Acesso em: 30 out. 2017.