Practical Malware Analysis & Triage

Malware Analysis Report

TCM Security Practical Malware Analysis & Triage Course

Analysis of Malware.stage0.exe

January 2024 | Amy Bathurst | v1.0

Table of Contents

[Executive Summary 3](#_Toc157626070)

[High-Level Technical Summary 3](#_Toc157626071)

[Malware Composition 4](#_Toc157626072)

[Malware.stage0.exe 4](#_Toc157626073)

[werflt.exe 4](#_Toc157626074)

[Basic Static Analysis 4](#_Toc157626075)

[Strings Extracted by Floss 4](#_Toc157626076)

[PEStudio Analysis 5](#_Toc157626077)

[Basic Dynamic Analysis 6](#_Toc157626078)

[Initial Indicators 6](#_Toc157626079)

[Process Monitor Analysis 7](#_Toc157626080)

[TCPView Analysis 9](#_Toc157626081)

[Advanced Static Analysis 10](#_Toc157626082)

[Cutter Analysis of werflt.exe 10](#_Toc157626083)

[Advanced Dynamic Analysis 11](#_Toc157626084)

[Analyzing Malware.stage0.exe in ‘Process Hacker 2’ 11](#_Toc157626085)

[Interpreting the Bytes of Shellcode Written to Memory 12](#_Toc157626086)

[Testing the Reverse Shell 13](#_Toc157626087)

[Indicators of Compromise 14](#_Toc157626088)

[Network Indicators 14](#_Toc157626089)

[Host-based Indicators. 14](#_Toc157626090)

[Appendices 16](#_Toc157626091)

[A. Yara Rule 16](#_Toc157626092)

[B. Decompiled Code Snippets 17](#_Toc157626093)

# Executive Summary

Malware.Stage0 is a loader which drops the malware WerFlt onto the infected host and then launches the process WerFault,exe, the Windows Error Reporting application. WerFlt then injects shell code into process WerFault.exe which opens a reverse shell. This reverse shell connects the infected host to the attacker’s host thereby giving the attacker the ability to remotely execute commands on the infected host. Symptoms of infection include a Windows command line console briefly opening, the file werflt.exe saved to the folder C:\Users\Public, and process WerFault.exe making a network connection to port 8443 of IP address 127.0.0.1.

# High-Level Technical Summary

This malware consists of two components: a loader and a process injector which injects shellcode that opens a reverse shell.

Malware.stage0.exe

Launches WerFault.exe

Drops and launches werflt.exe

werflt.exe

Inject shellcode into WerFault.exe

Shellcode launches a reverse shell

# Malware Composition

This malware consists of the executables Malware.stage0.exe and werflt.exe.

|  |  |
| --- | --- |
| **Malware.stage0.exe** | fca62097b364b2f0338c5e4c5bac86134cedffa4f8ddf27ee9901734128952e3 |
| werflt.exe | 0516009622b951c6c08fd8d81a856eaab70c02e6bc58d066bbdfafe8c6edabea |

## **Malware.stage0.exe**

The executable Malware.stage0.exe is a loader written in the Nim language. It launches process WerFault.exe and drops and launches process werflt.exe.

## werflt.exe

The executable werflt.exe is a process injector written in C++. It injects shellcode into the process WerFault.exe which opens a reverse shell.

# Basic Static Analysis

### Strings Extracted by Floss

The Floss string extraction tool was run on file Malware.stage0.exe.Listed below are the extracted strings of greatest interest. Two significant strings reference executables: werflt.exe and WerFault.exe. The string extraction also revealed which programming languages the malware is written in. The .nim file extension indicates code written in the Nim language and the MinGW compiler commands indicate code written in C/C++.

|  |
| --- |
| @C:\Users\Public\werflt.exe |
| @C:\Windows\SysWOW64\WerFault.exe |
| fatal.nim  io.nim  reams.nim  strutils.nim |
| GNU C99 9.2-win32 20191008 -m32 -mtune=generic -march=pentiumpro -g -O2 -std=gnu99 -fno-PIE  ./mingw-w64-crt/crt/crtexe.c |

### PEStudio Analysis

The executable **Malware.stage0.exe** was evaluated in PEStudio to extract characteristics of the executable. The evaluation showed that Malware.stage0.exe is a 32-bit executable which has a GUI.

A screenshot of a computer

Description automatically generated

The evaluation also showed Malware.stage0.exe imports several Windows API calls of interest. These Windows API calls retrieve information about a process or thread and manipulate processes. In particular, the Windows API calls VirtualProtect and GetProcAddress are often used by malware that performs process injection.

|  |
| --- |
| GetCurrentProcessId |
| GetCurrentProcess |
| GetProcAddress |
| LoadLibraryA |
| TerminateProcess |
| VirtualAlloc |

The executable **werflt.exe** was also evaluated in PEStudio. From the evaluation it was learned that the executable is a 32-bit C++ executable built in Visual Studio 2008.

A screenshot of a computer

Description automatically generated

The executable werflt.exe also imports several Windows API calls that are often used by malware to perform process injection. This is additional evidence that the malware performs process injection.

|  |
| --- |
| CreateRemoteThreat |
| GetCurrentProcessId |
| GetCurrentProcess |
| OpenProcess |
| TerminateProcess |
| WriteProcessMemory |
| VirtualAllocEx |

# Basic Dynamic Analysis

The malware was run in a sandbox so that its behavior could be observed. The basic dynamic analysis showed that Malware.stage0.exe launches the processes WerFault.exe and werflt.exe and that WerFault.exe subsequently creates a network connection.

### Initial Indicators

When started process werflt.ex briefly opens a window. This window is similar in appearance to a Window command line console.

A computer screen with a black screen

Description automatically generated

Process Malware.stage0.exe drops the executable werflt.exe in folder C:\Users\Public.

A screenshot of a computer

Description automatically generated

### Process Monitor Analysis

The malware was analyzed in Process Monitor to view the behavior of process Malware.stage0.exe.

A screenshot of a computer

Description automatically generated

After filtering on process Malware.stage0.exe to obtain its PID, the parent PID filter was set to 9420, the PID of process Malware.stage0.exe, to determine if it launches any processes. Setting this filter revealed that processes WerFault.exe and werflt.exe were launched by Malware.stage0.exe.

A screenshot of a computer

Description automatically generated

The process Wefault.exe launched by Malware.stage0.exe is the Windows Error Reporting application.

A screenshot of a computer

Description automatically generated

The process werflt.exe launched by Malware.stage0.exe is the executable it dropped in folder C:\Users\Public.

A screenshot of a computer

Description automatically generated

### TCPView Analysis

Examining network connections in TCPView revealed that the process WerFault.exe makes a TCP connection to the IP address 127.0.0.1 on port 8443.

A screenshot of a computer

Description automatically generated

# Advanced Static Analysis

## Cutter Analysis of werflt.exe

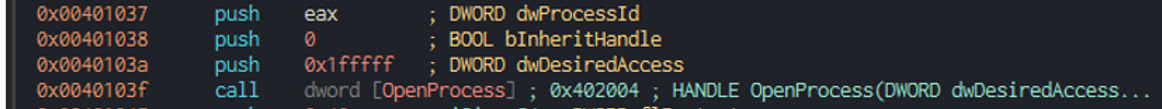
When the code for the executable werflt.exe, was viewed in the disassembler Cutter, a commonly seen pattern of Window API calls used in process injection was found in the executable’s main function. The main function consists of a sequence of Windows API calls used in process injection: OpenProcess, VirtualAllocEx, WriteProcessMemory, and CreateRemoteThread. This code provides evidence that the malware performs process injection.

When werflt.exe is launched, two arguments needed by the process injection code are passed to it. The first argument is the process PID of the process code will be injected into, passed in through the argument lpStartAddress. The second argument is the shell code that that will be written to the process, passed in through the argument lpBuffer.

A black screen with blue text

Description automatically generated

The first Windows API call in the process injection sequence is called. The call to OpenProcess opens the local process object specified by the function argument dwProcessId. In the code, dwProcessId is set to the process PID saved in the variable lpStartAddress. Access rights to the process is set to 0xfffff, for all access rights, through the argument dwDesiredAccess. When the call completes, it returns a handle to the requested process.



The call to VirtualAllocE reserves space in the memory of the process. The handle to the process returned by OpenProcess is passed in to VirtualAllocEx through the argument hProcess. The value assigned to the ftProtect argument specifies that the memory space is given read, write, and execute permissions. If successful, the call returns the base address of the allocated memory space.

A screen shot of a computer code

Description automatically generated

The Windows API call WriteProcessMemory is called. This Windows API call writes to memory allocated in a process. The memory will be written with the shell code stored in the variable lpBuffer. The call is also passed the handle to the process through the argument hProcess and the pointer to the base address of the allocated memory space returned by VirtualAllocEx.

A screen shot of a computer

Description automatically generated

The last Windows API call CreateRemoteThread is called. This call will create a thread inside the allocated address space of the process in which to execute the shell code.

A screen shot of a computer screen

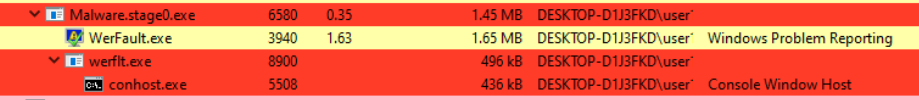
Description automatically generated

# Advanced Dynamic Analysis

Using the application ‘Process Hacker 2’ and the tool scdbg it was confirmed that werflt.exe injects shell into WerFault.exe which opens a reverse shell.

### Analyzing Malware.stage0.exe in ‘Process Hacker 2’

After launching Malware.stage0.exe, both processes can be seen in ‘Process Hacker 2’. In the process tree it can be seen that Malware.stage0.exe launches the processes WerFault.exe and werflt.exe.



Reviewing the memory usage by WerFault.exe shows 4 KB of memory with RWX permissions has been allocated to WerFault.exe. These permissions correspond with the permissions allocated to a memory block by VirtualAllocEx in the program werflt.exe. This memory was saved to a file for further analysis.

A screenshot of a computer

Description automatically generated

### Interpreting the Bytes of Shellcode Written to Memory

The tool scdbg tool was used to interpret the bytes of shell code. The shell code first loads the Winsock library ws2\_32. This library contains functions used to create a network connection. The Winsock functions WSAStartup and WSASocket are called to allocate the necessary network resources and create a socket. After the socket is created, network connection to port 8443 of IP address 127.0.0.1 is made using the connect function. A Windows command line console is opened by calling CreateProcess with the argument “cmd”. A reverse shell has now been created between the host machine and IP address 127.0.0.1. The thread will wait for console input or until the specified time has passed and then get the version of the host’s operating system and exit the process.

A screenshot of a computer program

Description automatically generated

### Testing the Reverse Shell

Using netcat a listener was set up to listen on port 8443. When Malware.stage0.exe started a connection to the listener was made by the infected host. After a brief period, the host’s operating system was obtained, and the connection was closed.

A screenshot of a computer

Description automatically generated

# Indicators of Compromise

## Network Indicators

|  |  |
| --- | --- |
| 127.0.0.1:8443 | The IP address and port the shell code connects to. |

## Host-based Indicators.

|  |  |
| --- | --- |
| C:\Users\Public\werflt.exe | The executable Malware.stage0.exe drops. |
| C:\Windows\SysWOW64\WerFault.exe | The process the shell code is injected into. |

# Appendices

## Yara Rule

rule MalwareStage0 {

meta:

last\_updated = "2024-01-22"

author = "Amy Bathurst"

description = "Yara rule for Malware.stage0"

strings:

// Fill out identifying strings and other criteria

$stringyara1 = "C:\\Users\\Public\\werflt.exe" ascii

$string2 = "C:\\Windows\\SysWOW64\\WerFault.exe" ascii

$string3 = "nim"

$PE\_magic\_byte = "MZ"

condition:

// Fill out the conditions that must be met to identify the binary

$PE\_magic\_byte at 0 and

($string1 and $string2 and $string3)

}

## Decompiled Code Snippets

A screen shot of a computer program

Description automatically generated

A screen shot of a computer

Description automatically generated