Icon

Description automatically generated

Practical Malware Analysis & Triage

Malware Analysis Report

Malware.Cryptlib64.dll

Mar 2024 | Teo Heng Shi | v1.0

# Table of Contents

[Table of Contents 2](#_Toc160510043)

[Executive Summary 3](#_Toc160510044)

[High-Level Technical Summary 4](#_Toc160510045)

[Malware Composition 5](#_Toc160510046)

[embed.vbs 5](#_Toc160510047)

[embed.xml: 5](#_Toc160510048)

[Basic Static Analysis 7](#_Toc160510049)

[Basic Dynamic Analysis 9](#_Toc160510050)

[Advanced Static Analysis 11](#_Toc160510051)

[Advanced Dynamic Analysis 12](#_Toc160510052)

[Indicators of Compromise 15](#_Toc160510053)

[Network Indicators 15](#_Toc160510054)

[Host-based Indicators 15](#_Toc160510055)

[Rules & Signatures 18](#_Toc160510056)

[Appendices 19](#_Toc160510057)

[A. Yara Rules 19](#_Toc160510058)

[B. Callback URLs 19](#_Toc160510059)

# Executive Summary

|  |  |
| --- | --- |
| SHA256 hash | 732f235784cd2a40c82847b4700fb73175221c6ae6c5f7200a3f43f209989387 |

The malware.cryptlib64.dll utilized embed function to execute its malicious activities.

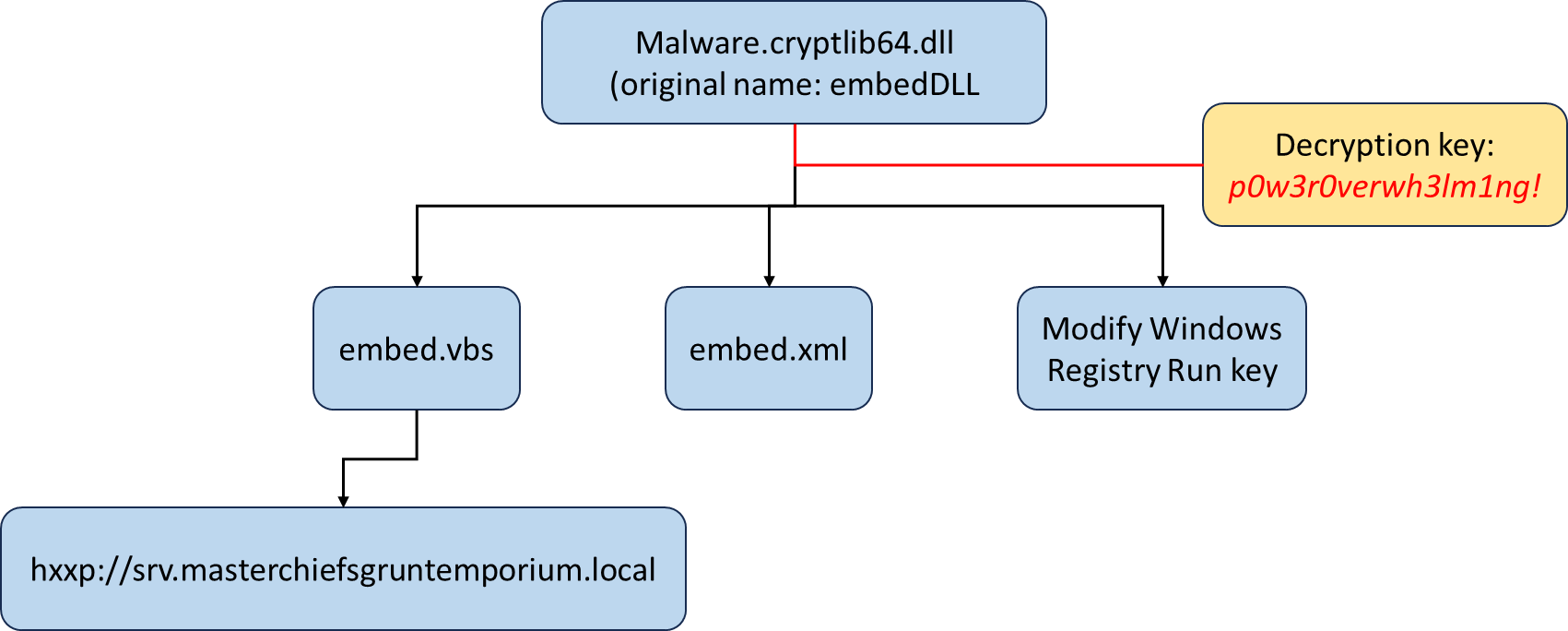
Upon detonated, two (2) files – *embed.xml* and *embed.vbs* have been created and one (1) Windows Registry key modification have been observed. It is noted that the malware persists by saving itself in the registry key, triggering the execution of *embed.vbs* upon user login. Symptoms of infection include beaconing to the URL(s) listed in Appendix B.

YARA signature rules are attached in Appendix A. Malware sample and hashes have been submitted to VirusTotal for further examination.

# High-Level Technical Summary

The malware.cryptlib64.dll generates the files – embed.xml and embed.vbs and saved it into C:\Users\Public\ and C:\Users\Public\Documents respectively.

The VBScript code will invoke MSBuild.exe to process C:\Users\Public\embed.xml. The content of embed.xml contains a sizable block of base64-encoded data. Upon triggered, it is observed that embed.vbs will initiate traffic to hxxp://srv.masterchiefsgruntemporium.local

****

# Malware Composition

DemoWare consists of the following components:

|  |  |
| --- | --- |
| File Name | SHA256 Hash |
| embed.vbs | 66fd543f31545082cf8fcc45a6ab1094bc118c45634f2be450f84f4e5745b291 |
| embed.xml | f1548cd02784606c8abac865abf5ed6220d34eea88c7a5715e0183d7f050f4ab |

## embed.vbs

The script will trigger MSBuild.exe under C:\Windows\Microsoft.Net\Framework\v4.0.30319 and to process embed.xml under C:\Users\Public directory

Set oShell = CreateObject ("Wscript.Shell")

Dim strArgs

strArgs = "C:\Windows\Microsoft.NET\Framework\v4.0.30319\MSBuild.exe C:\Users\Public\embed.xml"

oShell.Run strArgs, 0, false

embed.xml:

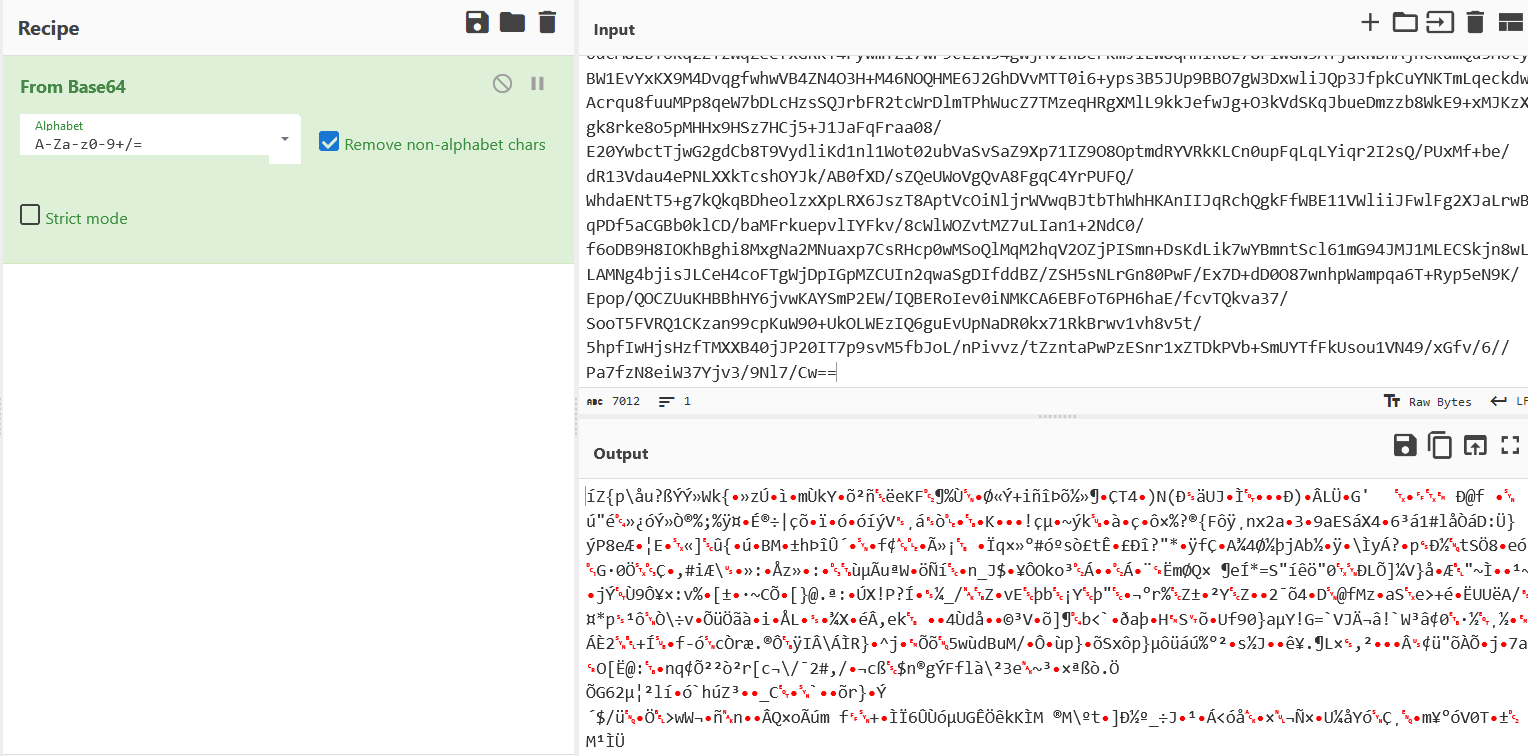
A Base64 encoded payload was observed to be under the embed.xml file; which could not be translated through Cyberchef



=== truncated for brevity==

/xGfv/6//Pa7fzN8eiW37Yjv3/9Nl7/Cw==

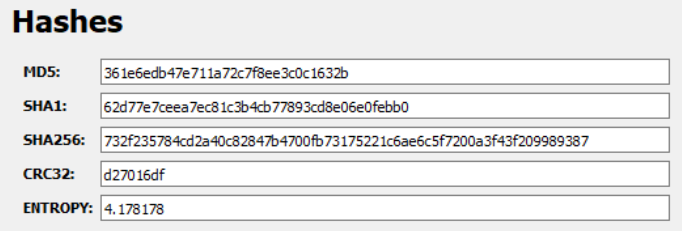
*Fig 1: Base64 encoded payload.*

**

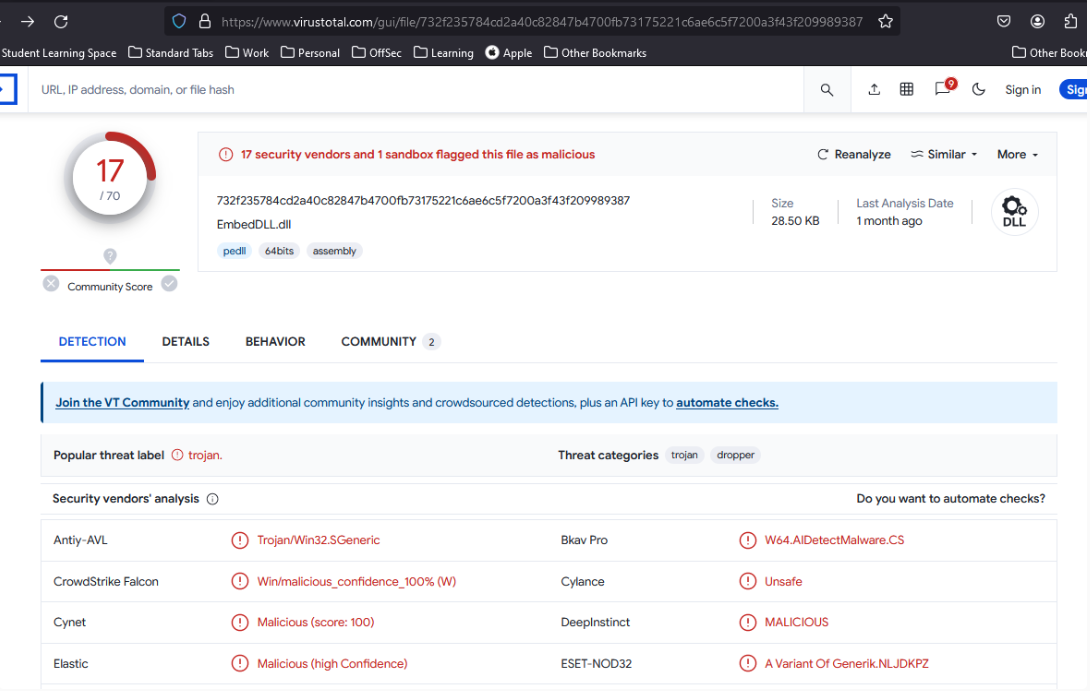
# Basic Static Analysis

{Screenshots and description about basic static artifacts and methods}

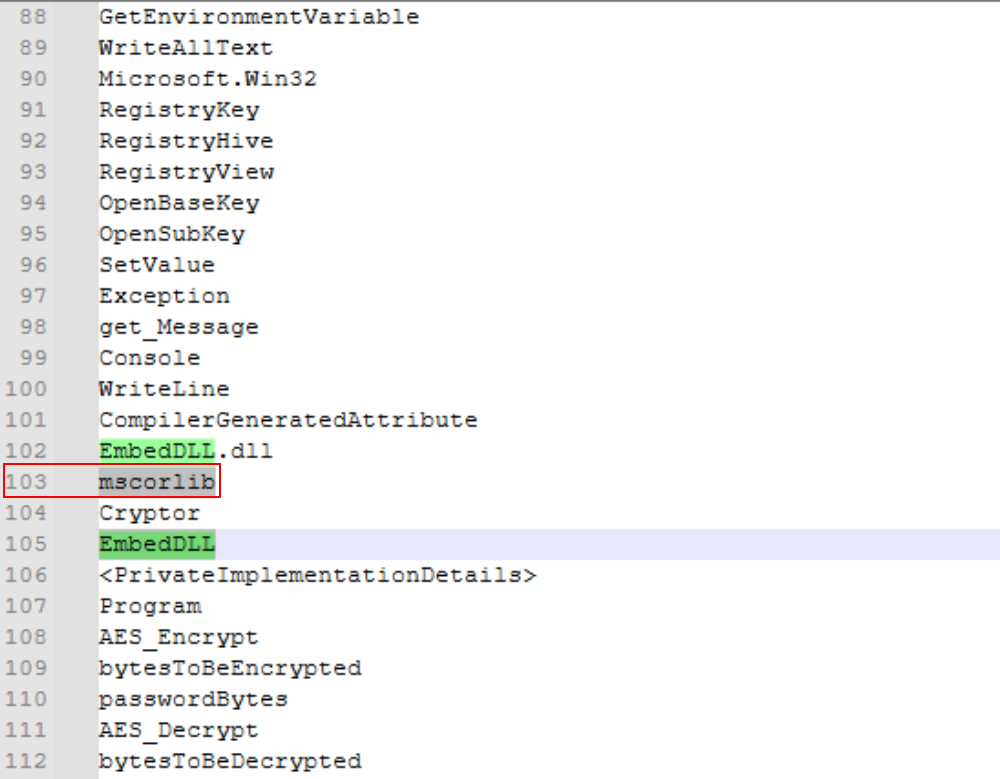
Using Cutter, the SHA256 and MD5sum information can be extracted from the dashboard



With the hash value, we run a check with VirusTotal, where we observed there are 17 security vendors who flagged this as malicious.



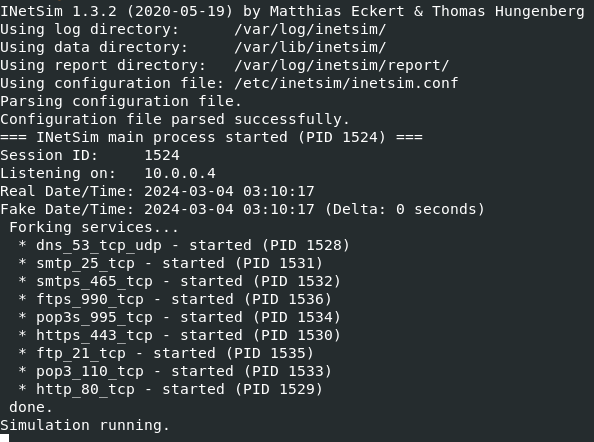
From the floss output, we noted the keywords mscorlib, which is an abbreviation for Multi-language Standard Common Object Runtime Library. This means that the dll is a .Net application, or specifically, a C# (C-Sharp) application.



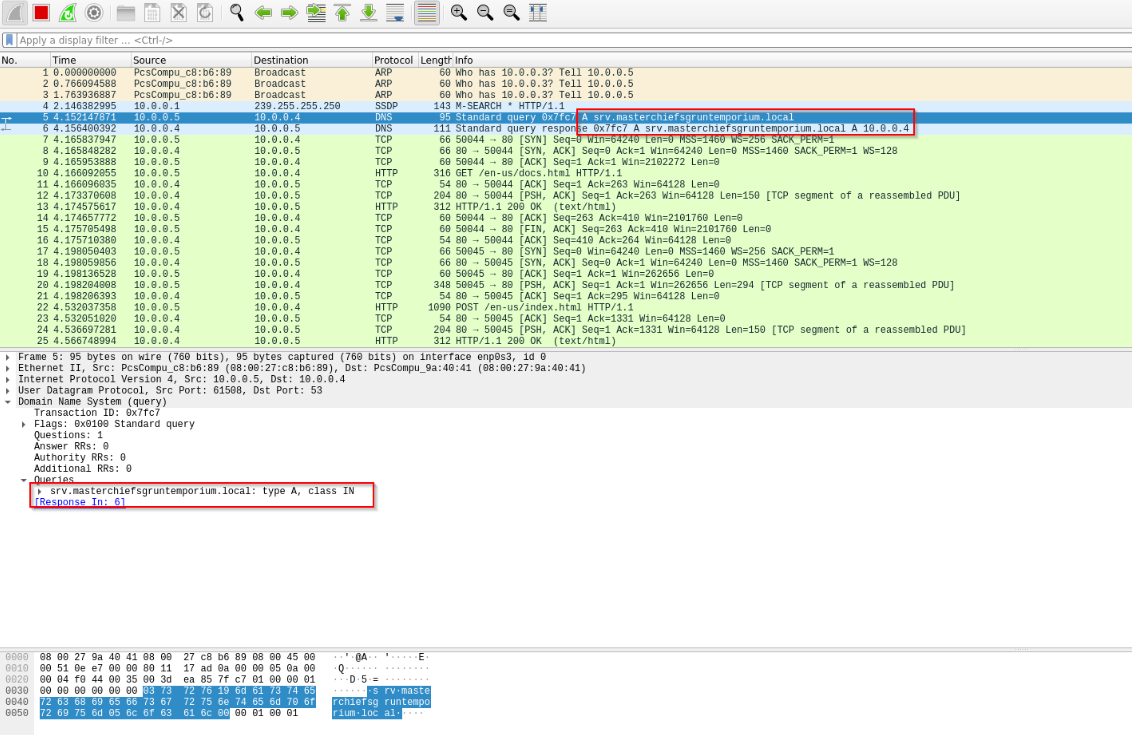
# Basic Dynamic Analysis

{Screenshots and description about basic dynamic artifacts and mUsingethods}

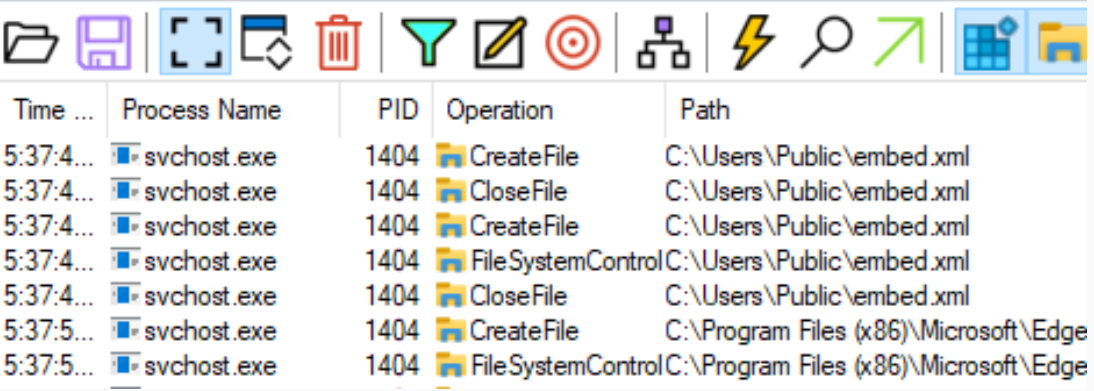
Prior to detonating the malicious dll, setup inetsim to act as a proxy server to serve out the required information and also to monitor the type of information that is triggered from the source malware.



Upon activation, we noted through wireshark that traffic was requested from the source (or affected host) to *srv.masterchiefsgruntemporium.local*



Through Procmon, we also noted that upon detonation, files were created using embed.xml

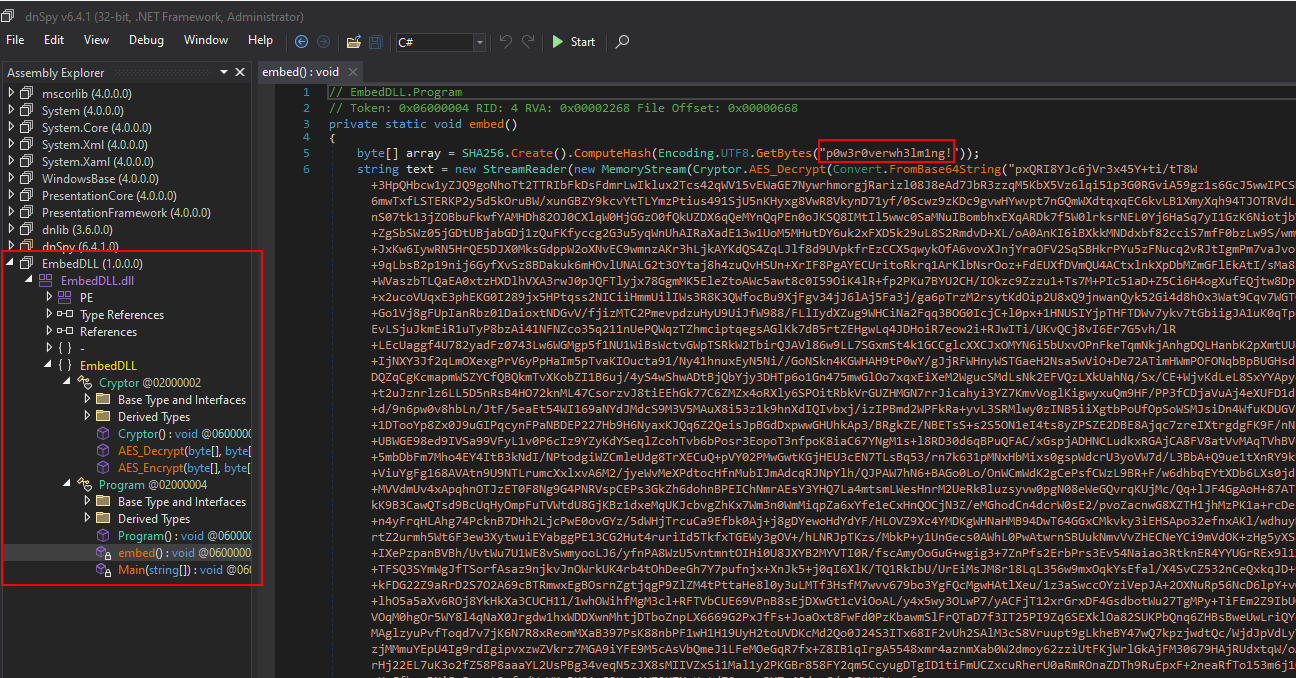


# Advanced Static Analysis

{Screenshots and description about findings during advanced static analysis}

To further analyse the binary and confirm on the source code, dnSpy was used to inspect the code. From the screenshot below, we noted a few things:

1. The binary is named as EmbedDll;
2. The main method is ***embed()***. This is useful when we want to trigger the dll binary;
3. The payload (base64 encoded) is encrypted using the keyword: p0w3r0verwh3lm1ng!;
4. There are 2 components within EmbedDLL, namely Cryptor and Program.

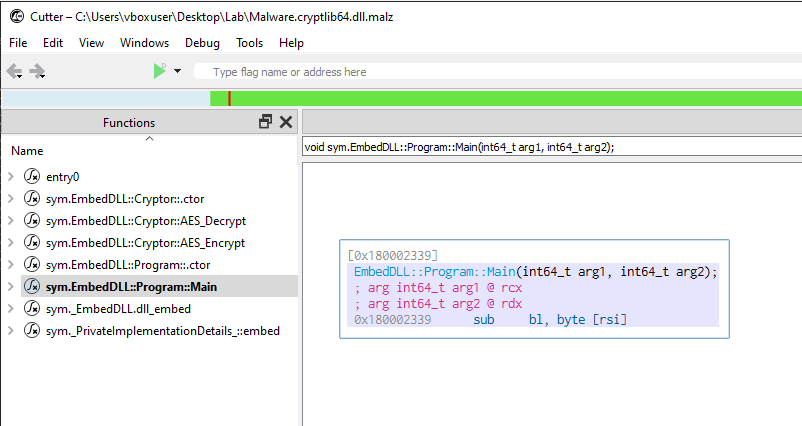


# Advanced Dynamic Analysis

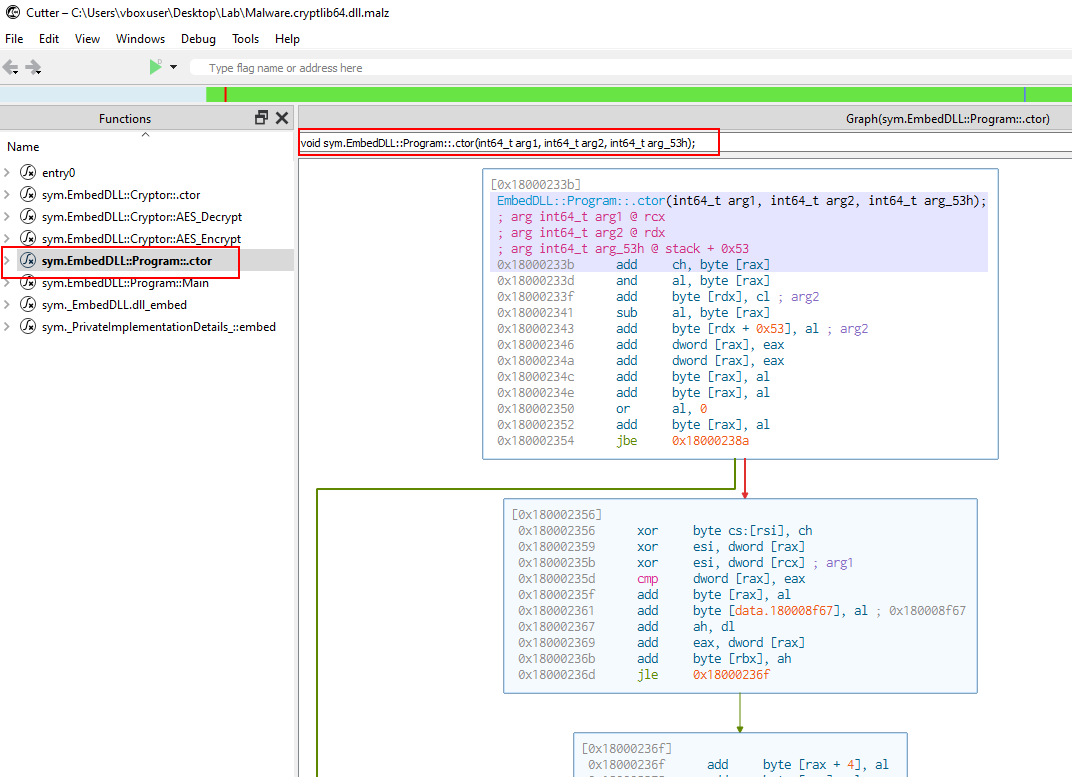
{Screenshots and description about advanced dynamic artifacts and methods}

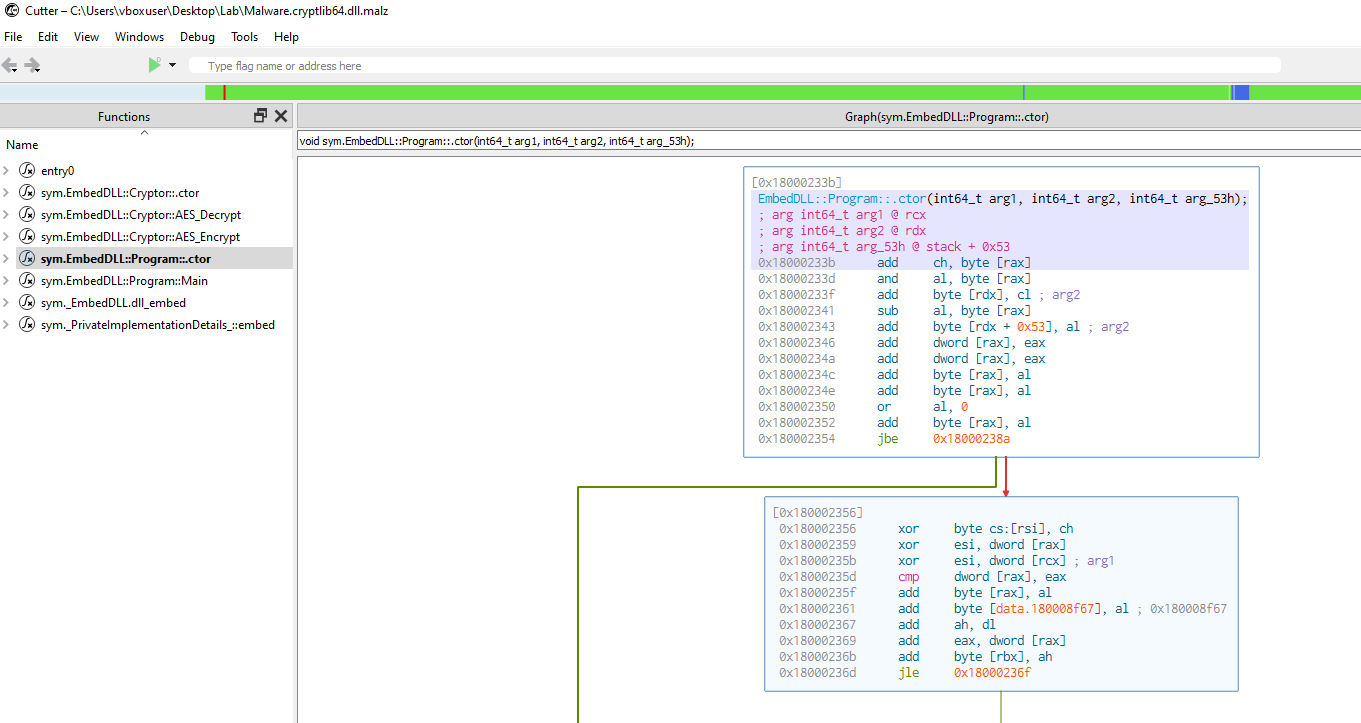
Based on the information retrieved from Advanced Static Analysis, we reviewed the disassembly flow chart under Cutter.

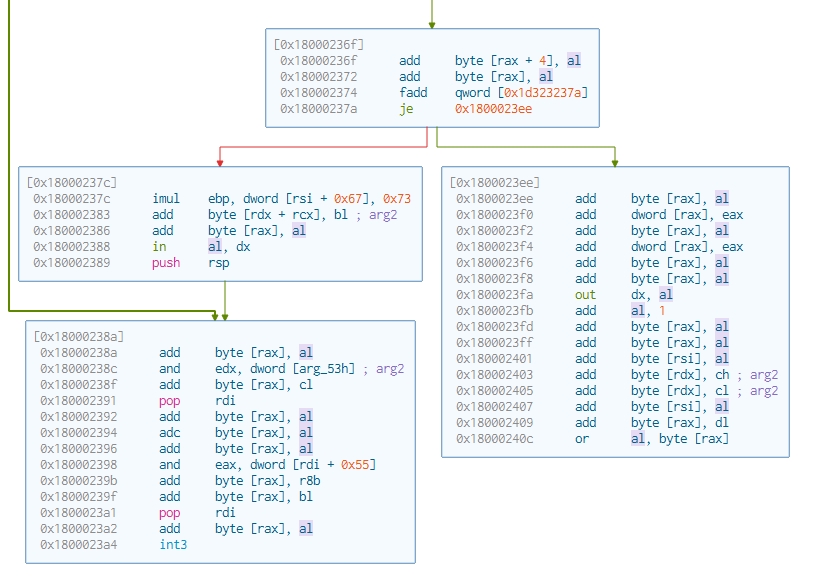
Under Main function



Which basically does not tell us much about the execution flow for embedDLL program. If we explore further into the ctor, we will notice more information





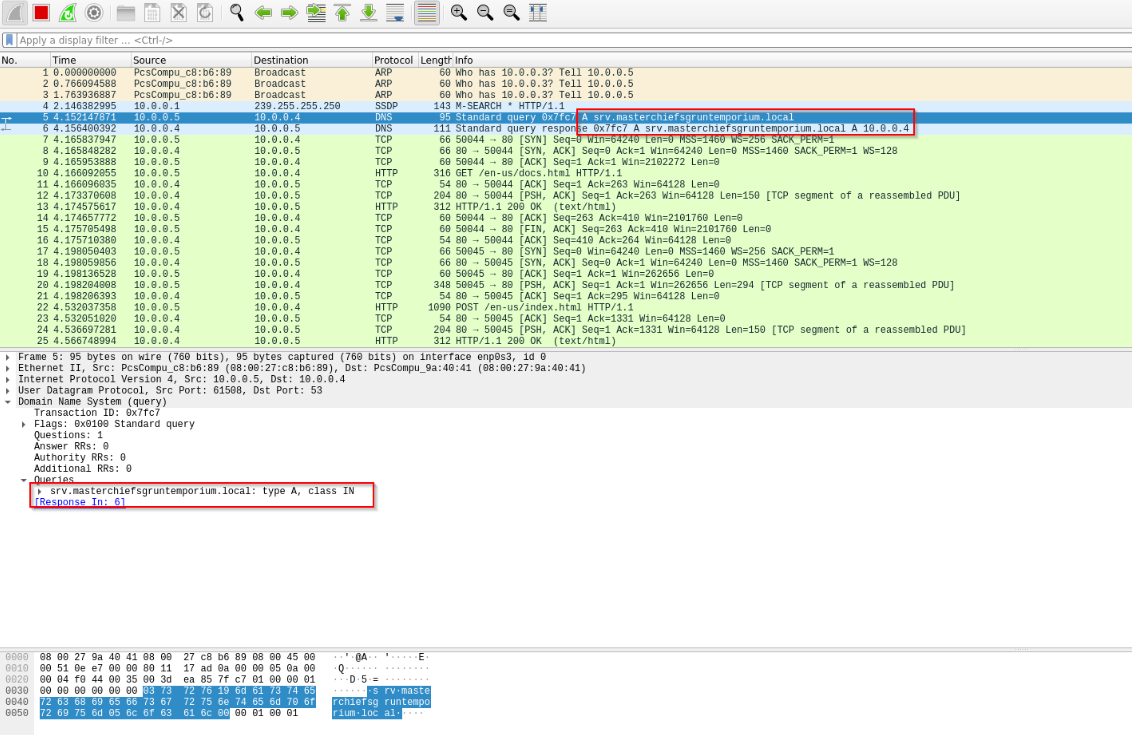


# Indicators of Compromise

The full list of IOCs can be found in the Appendices.

## Network Indicators

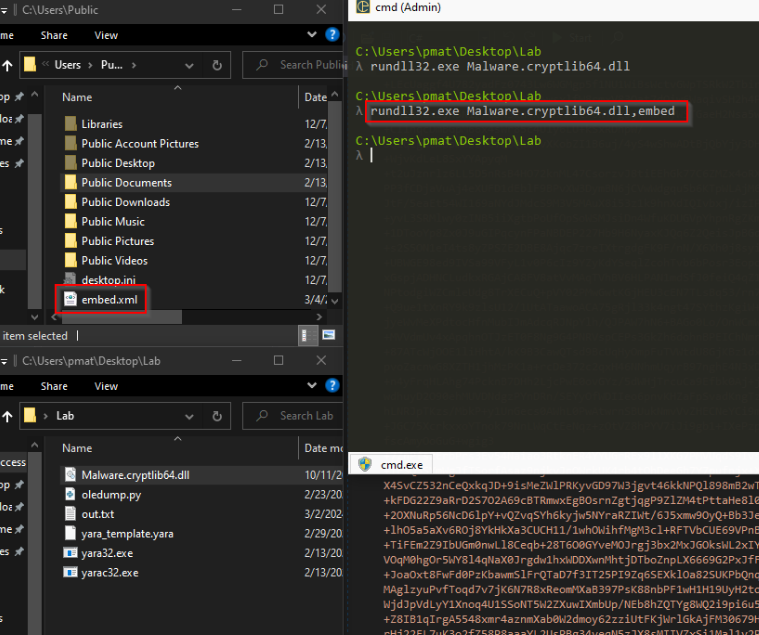
{Description of network indicators}

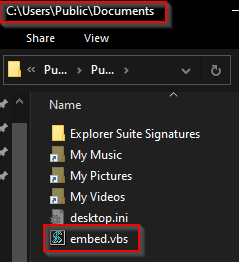


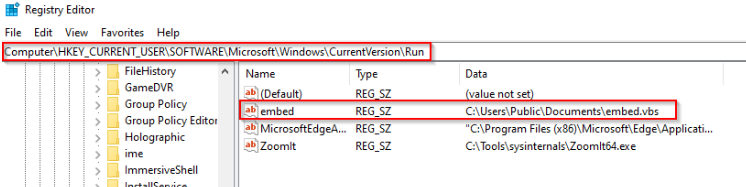
*Fig 3: WireShark Packet Capture of initial beacon check-in*

## Host-based Indicators

{Description of host-based indicators}







# Rules & Signatures

A full set of YARA rules is included in Appendix A.

{Information on specific signatures, i.e. strings, URLs, etc}

# Appendices

## Yara Rules

Full Yara repository located at: http://github.com/te0001hi/pmat\_lab

rule PE\_CSharp {

    meta:

        last\_updated = "2021-10-15"

        author = "PMAT"

        description = "A sample Yara rule for PMAT"

    strings:

        // Fill out identifying strings and other criteria

        $string1 = "p0w3r0verwh3lm1ng" ascii

        $string2 = "mscorlib"

        $PE\_magic\_byte = "MZ"

    condition:

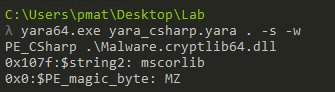
        // Fill out the conditions that must be met to identify the binary

        $PE\_magic\_byte at 0 and

        ($string1 or $string2)

        //any of them

}



## Callback URLs

|  |  |
| --- | --- |
| Domain | Port |
| hxxp:// srv.masterchiefsgruntemporium.local | 80 |