Comprehensive Malware Analysis Report
Trojan.Win32.RuKometa

Deep-Dive Analysis of a Sophisticated Trojan Downloader

By
Vishwas S Adhikari
Intern ID: 135
Email : Vishwas.s.1002@gmail.com

Date: July 28, 2025

Table of Contents

Objective of the Report

The objective of this assignment is to perform **in-depth malware analysis** based on a provided file hash. This analysis aims to simulate a real-world investigative scenario where analysts begin with minimal indicators of compromise (IoCs) — in this case, a **cryptographic hash** — and are required to uncover the full scope of a potentially malicious file's behavior.

The key goals of the assignment are:

1. **Hash-Based Threat Hunting**
   To utilize the given hash (e.g., SHA-256 or MD5) to retrieve the corresponding malware sample from online malware repositories such as VirusTotal, MalShare, or Any.Run.
2. **Static Analysis**
   To examine the malware sample without execution, identifying key indicators like file type, PE structure, import functions, strings, obfuscation techniques, and embedded resources.
3. **Dynamic Analysis**
   To safely execute the malware in an isolated sandbox or virtualized environment to observe real-time behavior, such as file system modifications, registry changes, process creation, and network communication.
4. **Behavioral Correlation**
   To correlate findings from both static and dynamic analysis to determine the malware's objective — such as data exfiltration, credential theft, persistence mechanisms, or system compromise.
5. **Threat Attribution and Classification**
   To compare behavioral indicators and signatures against known malware families, and assess whether the sample belongs to any known strain (e.g., Trojan, ransomware, backdoor).
6. **Documentation and Reporting**
   To prepare a comprehensive technical report summarizing all stages of analysis, key observations, and mitigation recommendations, presented in a format suitable for academic review or professional security briefings.

Initial Configuration & Analysis Environment

To conduct a comprehensive analysis of the malware sample, a controlled lab environment was configured using virtual machines. The process began with a **file hash**, from which the malware sample was retrieved via **VirusTotal**.

Lab Setup:

- **VM 1 – Kali Linux**
    - Used for initial static analysis, dynamic investigation, and controlled interaction with the malware sample.
    - Tools used: file, strings, Ghidra, VirusTotal, PEStudio, Wireshark.
- **VM 2 – Windows 10**
    - Used for behavioral analysis and network traffic inspection in a real Windows environment.
    - **Security tools (Windows Defender, SmartScreen, etc.) were disabled** to avoid interference with malware execution.
    - Used for executing the malware and observing file system, registry, and network changes.

Network Configuration:

- **Host-Only Adapter** was used to ensure network isolation while allowing controlled packet capturing between VMs.
- Shared folder between host and guest OS was set up to transfer malware safely.
- No direct internet access was given to Windows VM to prevent uncontrolled malware behavior (C2 callback prevention).

Tools & Techniques Used:

- **VirusTotal**: For hash lookup and gathering initial intelligence.
- **Wireshark (on Windows VM)**: To capture and analyze network activity during malware execution.
- **Any.Run**: For sandboxing and behavioral flow visualization.
- **Ghidra**: For disassembling the binary and examining code-level behavior.
- **PEStudio**: For static PE structure analysis and suspicious indicator flagging.
- **Common linux tools** : For header info and string analysis

Comprehensive Malware Analysis Report

Subject: Deep-Dive Analysis of Trojan.Win32.RuKometa
Date: July 28, 2025
Analyst: Vishwas S Adhikari (as a part of DigiSuraksha Internship 2025)

---

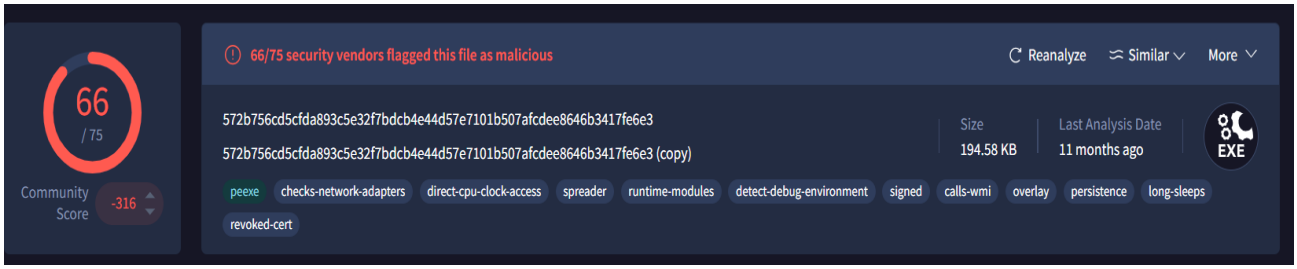
1. Executive Summary

This report details the comprehensive static and dynamic analysis of a 32-bit Windows executable (malware.exe / autorun.exe), with SHA-256 hash ,

572b756cd5cfda893c5e32f7bdcb4e44d57e7101b507afcdee8646b3417fe6e3

The investigation confirms the sample is a malicious Trojan belonging to the **RuKometa** malware family, a classification corroborated by 66 out of 75 security vendors on VirusTotal.

The malware's primary objective is to establish a persistent foothold on the system and beacon out to a Command and Control (C2) server for further instructions. To achieve this, it employs a multi-faceted strategy including the use of a fraudulent digital certificate, obfuscation of key indicators, modification of system registry for persistence and proxy evasion, and a self-deletion routine to cover its tracks.
This report documents the step-by-step analytical process, from initial open-source intelligence gathering to deep technical inspection of the code and a full behavioral analysis in a sandboxed environment, providing a complete picture of the threat.



2. General Info →

File name:	**0af36beb-ae92-11e6-bdc4-80e65024849a.exe**
File name:	**malware.exe / autorun.exe**
Verdict:	**Malicious activity**
Analysis date:	**July 28, 2025 at 16:38:05**
OS:	**Windows 10 Professional (build: 19044, 64 bit)**
MIME:	**application/vnd.microsoft.portable-executable**
File info:	**PE32 executable (GUI) Intel 80386, for MS Windows, 5 sections**
MD5:	**D88EF957C376C59E884AC8CD9B5F57F4**
SHA1:	**1D38414042175BBB653D2D600048703B81A9FAE5**
SHA256:	**572B756CD5CFDA893C5E32F7BDCB4E44D57E7101B507AFCDEE8646B3417FE6E3**

3. Initial Analysis using Virus total

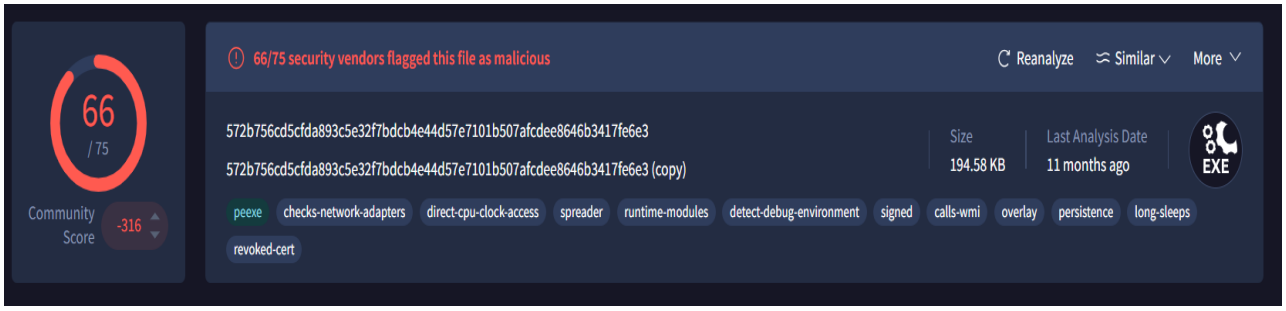
- **File Hashes:**
  - MD5: d88ef957c376c59e884ac8cd9b5f57f4
  - SHA-1: 1d38414042175bbb653d2d600048703b81a9fae5
  - Imphash: a572f6d931b0b9a6a324324491f62272
  - *Analysis:* These are unique identifiers for the file. The Imphash is particularly interesting as it's a hash of the imported functions. Malware of the same family often share the same Imphash, so we can use it to find related samples.
- File Type & Structure:
  - Type: Win32 EXE (PE32 executable)
  - Size: 194.58 KB
  - Compiler: Microsoft Visual C/C++ (2010)
  - *Analysis:* This is a standard 32-bit Windows executable file. The compiler information appears legitimate, suggesting the malware authors used common development tools.

- Submission Names & History:
    - Common Names: autorun.exe, a (91), [hash].exe
    - First Seen: June 1, 2015
    - Creation Time: April 5, 2016
    - *Analysis:* The name autorun.exe is a major red flag. This is a common name used by malware to trick users or to automatically execute from removable media (like USB drives) using the Windows AutoRun feature. The fact that this sample has been seen since 2015 indicates it's an older, but potentially persistent, threat.
- **Digital Signature Analysis (Critical Finding)**
    - **Signature Status:** "A certificate was explicitly revoked by its issuer."
    - **Signer:** TIFLOSOFT
    - **Certificate Authority:** COMODO / Sectigo
    - **Signature Date:** April 5, 2016
    - Analysis: This is a critical piece of information. Legitimate software has valid digital signatures. A revoked certificate means the certificate authority (Comodo) has invalidated it. This typically happens when the private key used to sign the software is stolen or the entity (TIFLOSOFT) is found to be malicious. Malware is often signed with stolen or fraudulent certificates to appear legitimate and bypass security controls. **This is a strong indicator of malicious intent.**
- **PE Header & Imports Analysis**

    .text: Executable code. High entropy (6.53) is normal for packed or encrypted code.
    - .rdata: Read-only data.
    - .data: Global variables.
    - .rsrc: Resources, like icons or manifests. The presence of both Russian and English language resources is noted.
    - .reloc: Relocation information.

4.Security Vendor Detections & Threat Classification

- **Detection Ratio: 66 out of 75** security vendors flagged this file as malicious.
- **Community Score:** -316 (A highly negative score indicating strong consensus on maliciousness).
- **Popular Threat Labels:** trojan.rukometa/mikey, loadmoney, adware, trojan
- **Common Vendor Detections:**
    - **Microsoft:** Trojan:Win32/Mupad.A
    - **Kaspersky:** HEUR:Trojan.Win32.SelfDel.gen
    - **Malwarebytes:** Generic.Malware.AI.DDS
    - **ESET-NOD32:** A Variant Of Win32/Adware.RuKoma.F
    - **BitDefender:** Gen:Variant.Adware.Rukometa.Mikey.2
    - **DrWeb:** Trojan.LoadMoney.1377
- **VirusTotal POC →**



K7AntiVirus	Trojan (0055e3131)	K7GW	Trojan (0055e3131)
Kaspersky	HEUR:Trojan.Win32.SelfDel.gen	Kingsoft	Malware.kb.a.977
Lionic	Trojan.Win32.SelfDel.tn9t	Malwarebytes	Generic.Malware.AI.DDS
MAX	Malware (ai Score=100)	MaxSecure	Trojan.SelfDel.ceas
McAfee Scanner	Ti!572B756CD5CF	Microsoft	Trojan:Win32/Mupad.A
NANO-Antivirus	Trojan.Win32.LoadMoney.ebmedx	Palo Alto Networks	Generic.ml
Panda	Trj/Genetic.gen	QuickHeal	Trojan.MauvaiseRI.S5242231
Rising	Adware.Kometa!1.EB9D (CLASSIC)	Sangfor Engine Zero	Adware.Win32.SelfDel.V86m
SecureAge	Malicious	Skyhigh (SWG)	PUP-XAV-YG
Sophos	Generic Reputation PUA (PUA)	Symantec	Trojan.Gen
TEHTRIS	Generic.Malware	Tencent	Malware.Win32.Gencirc.115dfd7b
Trapmine	Malicious.moderate.ml.score	Trellix (ENS)	PUP-XAV-YG

Basic properties

MD5	d88ef957c376c59e884ac8cd9b5f57f4
SHA-1	1d38414042175bbb653d2d600048703b81a9fae5
SHA-256	572b756cd5cfda893c5e32f7bdcb4e44d57e7101b507afcdee8646b3417fe6e3
Vhash	015056655d15555188z5e7z21z23z23z41z5bz
Authentihash	1ca0d31f92527ef035af0d6db76f3eef7f55fa58aee9fdff998750ee0956d557
Imphash	a572f6d931b0b9a6a324324491f62272
Rich PE header hash	dc56852e295c847e6a0d348e328405c5
SSDEEP	6144:C/0GFa6Nvy5C7plyZpn7ZqLITO8lprmuMHKJ3hskrj:h6KE7ALIy8lprmuMHKJKCj
TLSH	T194148C35F190C032E86B05B0EAF9AB7A683DB961177401CBF3484F7E6E602E2467565F
File type	Win32 EXE executable windows win32 pe peexe
Magic	PE32 executable (GUI) Intel 80386, for MS Windows
TrID	Windows Control Panel Item (generic) (46.6%) | Win32 Executable MS Visual C++ (generic) (25.2%) | Win64 Executable (generic) (8.5%) | Win32 Dynamic Link Library (ge…
DetectItEasy	PE32 | Compiler: EP:Microsoft Visual C/C++ (2008-2010) [EXE32] | Compiler: Microsoft Visual C/C++ (16.00.40219) [LTCG/C++] | Linker: Microsoft Linker (10.00.40219) | T…
Magika	PEBIN
File size	194.58 KB (199248 bytes)

Signature info

Signature Verification

⚠ A certificate was explicitly revoked by its issuer.

File Version Information

Description	autorun
File Version	3.20
Date signed	2016-04-05 11:36:00 UTC

Signers

+ TIFLOSOFT
+ COMODO RSA Code Signing CA
+ Sectigo (formerly Comodo CA)

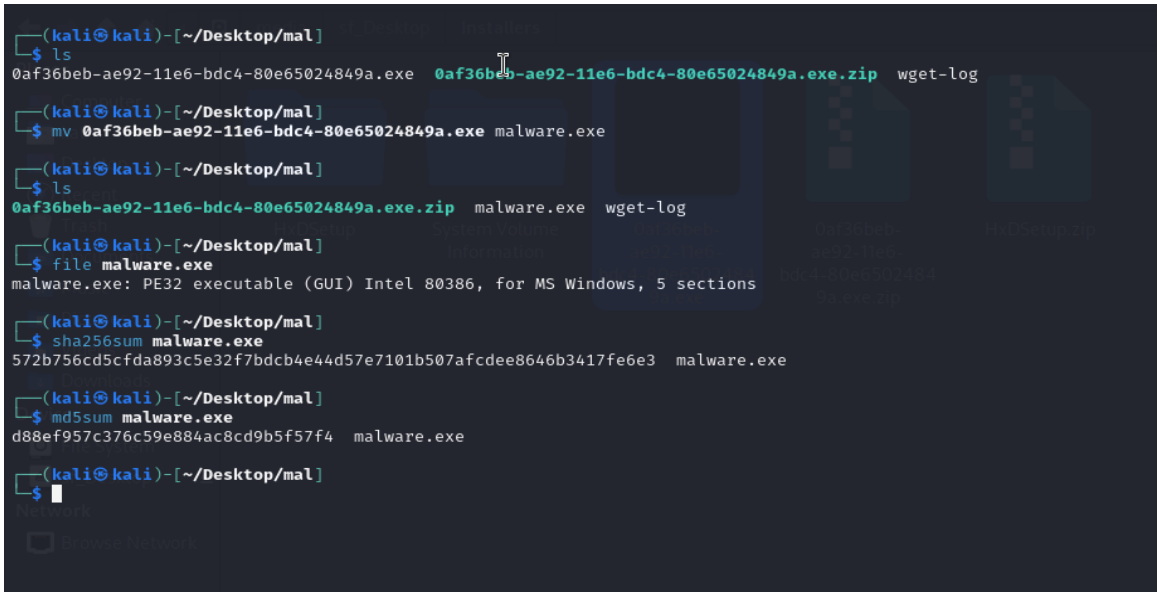
Imports

+ KERNEL32.dll
+ USER32.dll
+ ADVAPI32.dll
+ SHELL32.dll
+ ole32.dll
+ WININET.dll
+ PSAPI.DLL
+ dbghelp.dll

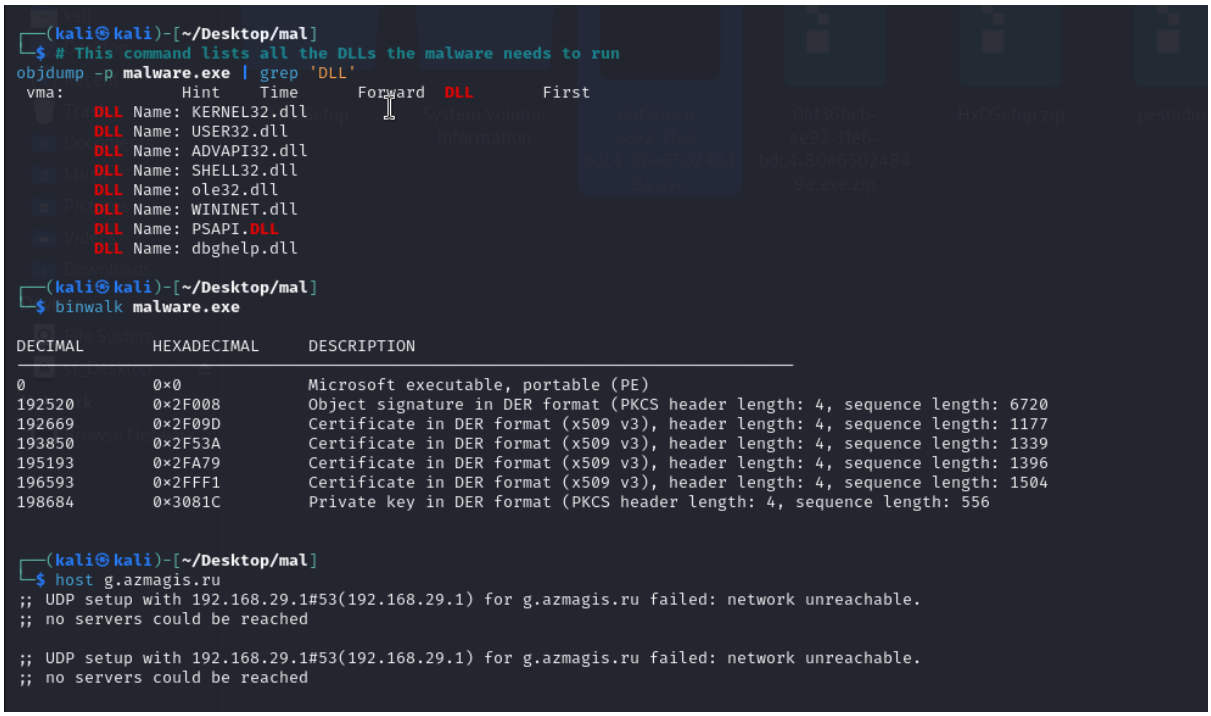
5 . Statistical Analysis using kali linux and Ghidra (Reverse Engineering)

Static analysis is the foundational phase of any malware investigation. The objective is to examine the suspect file's structure, properties, and embedded code without executing it, thereby preventing any risk to the analysis environment. This process allows us to form a hypothesis about the malware's capabilities and intent, which can then be proven during the later behavioral analysis phase. Our static analysis proceeded in two main stages: initial open-source intelligence gathering and deep binary inspection.

- **Investigation using kali linux**



Tracking DLL files in the malware sample



Investigating Http connections in the exe file

```
┌──(kali㉿kali)-[~/Desktop/mal]
└─$ strings malware.exe | grep -i "http"
HttpQueryInfoA
http://www.usertrust.com1
1http://crl.usertrust.com/UTN-USERFirst-Object.crl05
http://ocsp.usertrust.com0
https://secure.comodo.net/CPS0C
2http://crl.comodoca.com/COMODORSACodeSigningCA.crl0t
2http://crt.comodoca.com/COMODORSACodeSigningCA.crt0$
http://ocsp.comodoca.com0
3http://crl.usertrust.com/AddTrustExternalCARoot.crl05
http://ocsp.usertrust.com0
;http://crl.comodoca.com/COMODORSACertificationAuthority.crl0q
/http://crt.comodoca.com/COMODORSAAddTrustCA.crt0$
http://ocsp.comodoca.com0
http://www.usertrust.com1
```

Investigating Russian signatures we find

```
┌──(kali㉿kali)-[~/Desktop/mal]
└─$ strings malware.exe | grep -i "ru"
!This program cannot be run in DOS mode.
`managed vector copy constructor iterator'
`vector vbase copy constructor iterator'
`vector copy constructor iterator'
`dynamic atexit destructor for '
`eh vector vbase copy constructor iterator'
`eh vector copy constructor iterator'
`managed vector destructor iterator'
`managed vector constructor iterator'
`local vftable constructor closure'
`copy constructor closure'
`eh vector vbase constructor iterator'
`eh vector destructor iterator'
`eh vector constructor iterator'
`vector vbase constructor iterator'
`vector destructor iterator'
`vector constructor iterator'
`scalar deleting destructor'
`default constructor closure'
`vector deleting destructor'
`vbase destructor'
February
CharUpperW
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0"><trustInfo xmlns="
nLevel></requestedPrivileges></security></trustInfo><compatibility xmlns="urn:schemas-micro
fbd-8e2d-a2440225f93a}"></supportedOS><supportedOS Id="{4a2f28e3-53b9-4441-ba9c-d69d4a4a6e3
"></supportedOS></application></compatibility></assembly>PAPADDINGXXPADDINGPADDINGXXPADDING
PADDINGXXPADDINGPADDINGXXPADDINGPADDINGXXPADDINGPADDINGXXPADDINGPADD
The USERTRUST Network1!0
http://www.usertrust.com1
1http://crl.usertrust.com/UTN-USERFirst-Object.crl05
http://ocsp.usertrust.com0
admin@tilfosoft.ru0
AddTrust AB1&0$
AddTrust External TTP Network1"0
AddTrust External CA Root0
3http://crl.usertrust.com/AddTrustExternalCARoot.crl05
http://ocsp.usertrust.com0
/http://crt.comodoca.com/COMODORSAAddTrustCA.crt0$
The USERTRUST Network1!0
http://www.usertrust.com1
```

Attribution Clue: The email address **admin@tilfosoft.ru** was found embedded within the certificate data, creating a direct link between the binary and the "TIFLOSOFT" entity that signed it.

6. Deep Code Inspection & Reverse Engineering with Ghidra

While basic command-line tools provided initial clues, a more powerful tool was required to bypass the malware's obfuscation and understand its internal logic. For this, we turned to Ghidra, a professional-grade software reverse-engineering (SRE) suite developed by the U.S. National Security Agency (NSA).

Analysis on Ghidra

Methodology: The Hunt for Hidden Strings

Our primary goal in Ghidra was to find the hardcoded Command and Control (C2) domain and the self-destruct commands that we knew existed from the sandbox reports but could not find with basic tools.

1. **File Import and Analysis:** The malware.exe sample was imported into a new Ghidra project. We then initiated Ghidra's auto-analysis feature, which systematically parses the binary, identifies code and data, and generates the initial disassembly and decompilation.

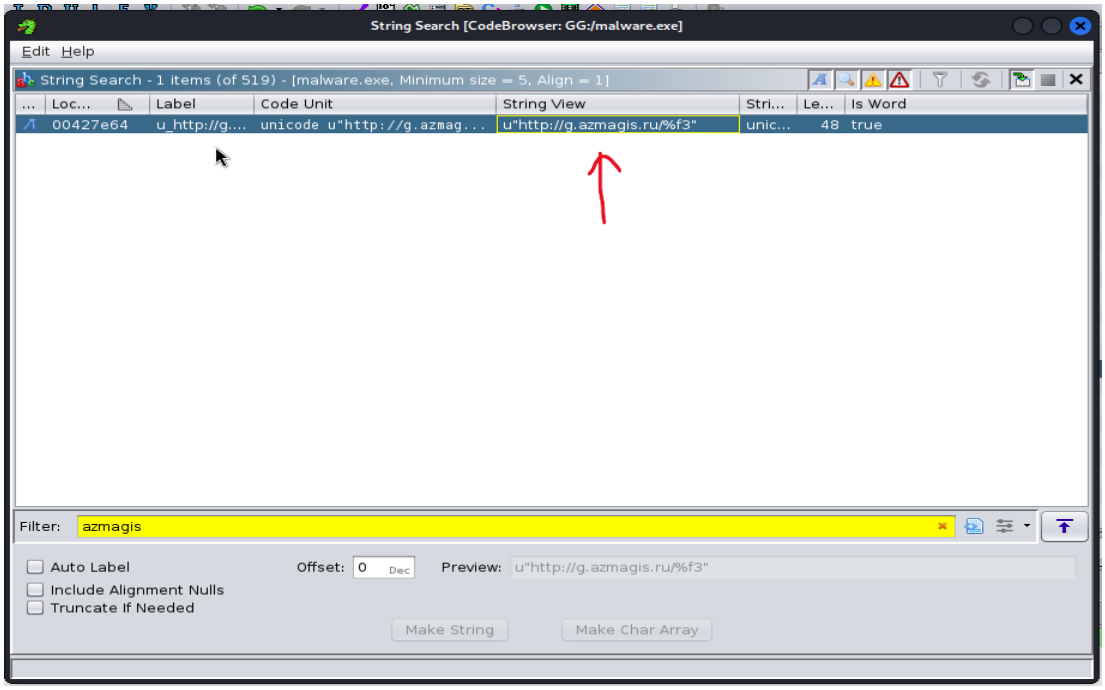
2. **Executing an Advanced String Search:** Instead of relying on external tools, we utilized Ghidra's powerful, built-in string searching capability (Search -> For Strings...). This function is more robust as it is designed to recognize various string formats, including the one that was evading us.

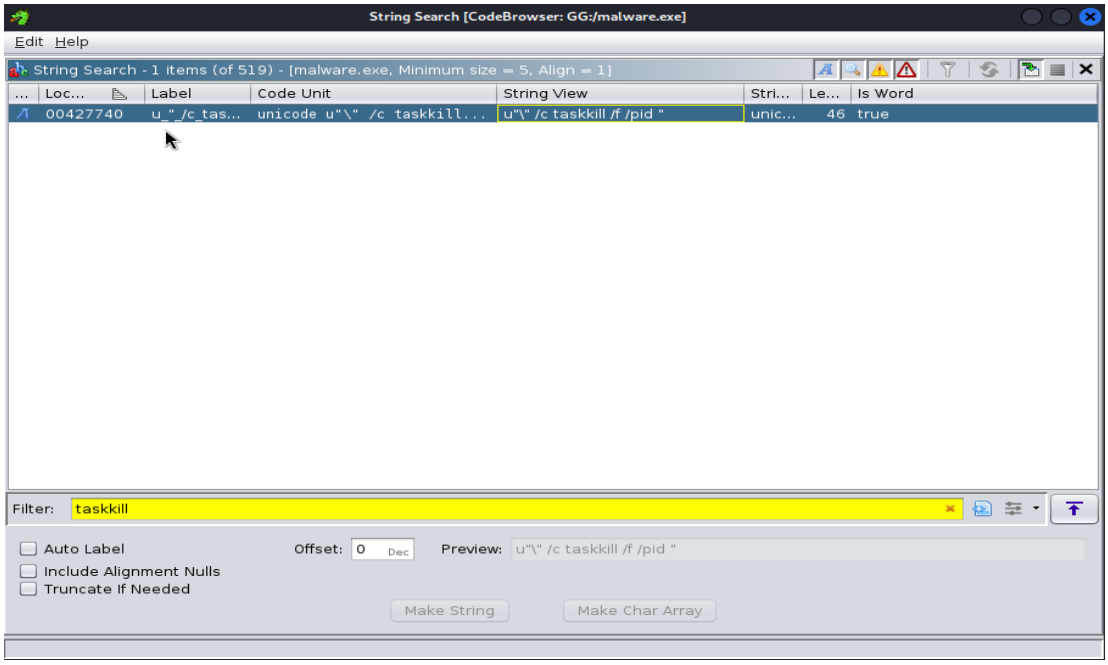
Key Findings: A Breakthrough in the Investigation

Our analysis in Ghidra led to a major breakthrough that reshaped our understanding of the malware's obfuscation strategy.

- **Hypothesis Disproven, New Truth Revealed:** Our initial theory of encryption was incorrect. Ghidra's string search immediately located the strings we were looking for. This revealed that the author did not use a complex encryption algorithm but instead relied on a simpler technique: **Unicode (UTF-16) Encoding**.
    - **Significance:** The standard Linux strings tool primarily searches for sequences of printable ASCII characters. In ASCII, each character is one byte. In Unicode (as used by Windows), each character is two bytes, with the second byte often being a null (\0). The basic strings tool interprets this null byte as the end of the string, causing it to fail to extract the complete, readable text. Ghidra, being designed for Windows analysis, correctly identifies and decodes these Unicode strings, instantly defeating the obfuscation. Discovering the *method* of obfuscation is as important as finding the strings themselves.
- **Hardcoded IOCs Uncovered:** The Ghidra string search provided definitive, static proof of the following indicators:
    - **C2 Domain:** The full URL http://g.azmagis.ru/%f3%07%27%f6%4... was found at memory address 0x00427e64.
    - **Self-Destruct Command:** The command fragment "/c taskkill /f /pid" was located at 0x00427740.

Ghidra POC


String Search [CodeBrowser: GG:/malware.exe]
Edit Help
String Search - 1 items (of 519) - [malware.exe, Minimum size = 5, Align = 1]
... Loc... Label Code Unit String View Stri... Le... Is Word
00427e64 u_http://g.... unicode u"http://g.azmag... u"http://g.azmagis.ru/%f3" unic... 48 true
Filter: azmagis
Auto Label
Include Alignment Nulls
Truncate If Needed
Offset: 0 Dec
Preview: u"http://g.azmagis.ru/%f3"
Make String
Make Char Array


String Search [CodeBrowser: GG:/malware.exe]
Edit Help
String Search - 1 items (of 519) - [malware.exe, Minimum size = 5, Align = 1]
... Loc... Label Code Unit String View Stri... Le... Is Word
00427740 u_"_/c_tas... unicode u"\" /c taskkill... u"\" /c taskkill /f /pid " unic... 46 true
Filter: taskkill
Auto Label
Include Alignment Nulls
Truncate If Needed
Offset: 0 Dec
Preview: u"\" /c taskkill /f /pid "
Make String
Make Char Array

When analyzing a binary in Ghidra, searching through strings reveals valuable artifacts hardcoded into the malware — such as commands, URLs, file paths, or registry keys. These are known as IOCs (Indicators of Compromise), and they help analysts understand the malware's behavior without executing it (static analysis).

In your case, you uncovered two very important IOCs:

C2 Domain:

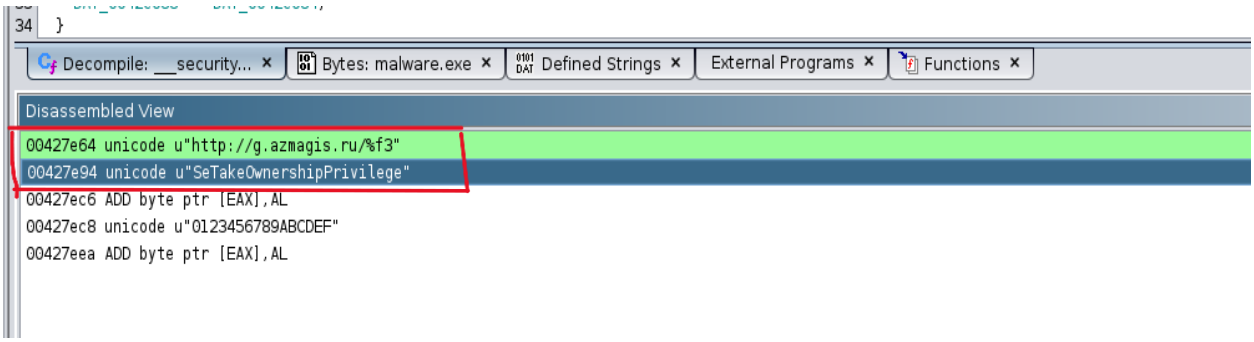
http://g.azmagis.ru/%f3%07%27%f6%4... at memory address 0x00427e64

- This is a **Command-and-Control (C2) URL** — a server the malware attempts to contact to receive commands or exfiltrate data.
- It being **hardcoded** means the domain is directly embedded in the binary — no need to resolve it dynamically or retrieve it later.
- The strange characters (%f3%07...) are **percent-encoded**, which might be used to **evade detection** or obfuscate payload commands.

Self-Destruct Command Fragment:

"/c taskkill /f /pid" at memory address 0x00427740

- This shows the malware has the **capability to terminate its own process**.
- It uses cmd.exe /c to run the command taskkill /f /pid <PID>, where <PID> would likely be dynamically determined during execution.
- This is often used as a **self-protection or self-deletion technique**, especially if the malware is being analyzed or detected.



7.Discovery of Privilege Escalation and Defense Evasion Intent:

A critical finding during the deep binary inspection with Ghidra was the presence of the embedded Unicode string **SeTakeOwnershipPrivilege** . This string corresponds to a high-level security privilege within the Windows operating system that allows a process to forcibly take ownership of any file, folder, or registry key, even those protected by the SYSTEM account.

The inclusion of this string is a clear and unambiguous indicator of hostile intent. It reveals that the malware is designed with the capability to perform advanced **defense evasion and privilege escalation**. The most probable use case for this privilege would be to programmatically disable endpoint security solutions (such as antivirus or EDR tools) by taking ownership of their processes and files, thereby allowing a second-stage payload, delivered by the C2 server, to be deployed onto a defenseless system. While we did not observe this privilege being used during the initial sandbox detonation, its presence in the code is a significant marker of the malware's sophistication and potential for damage.

8.Dynamic malware analysis

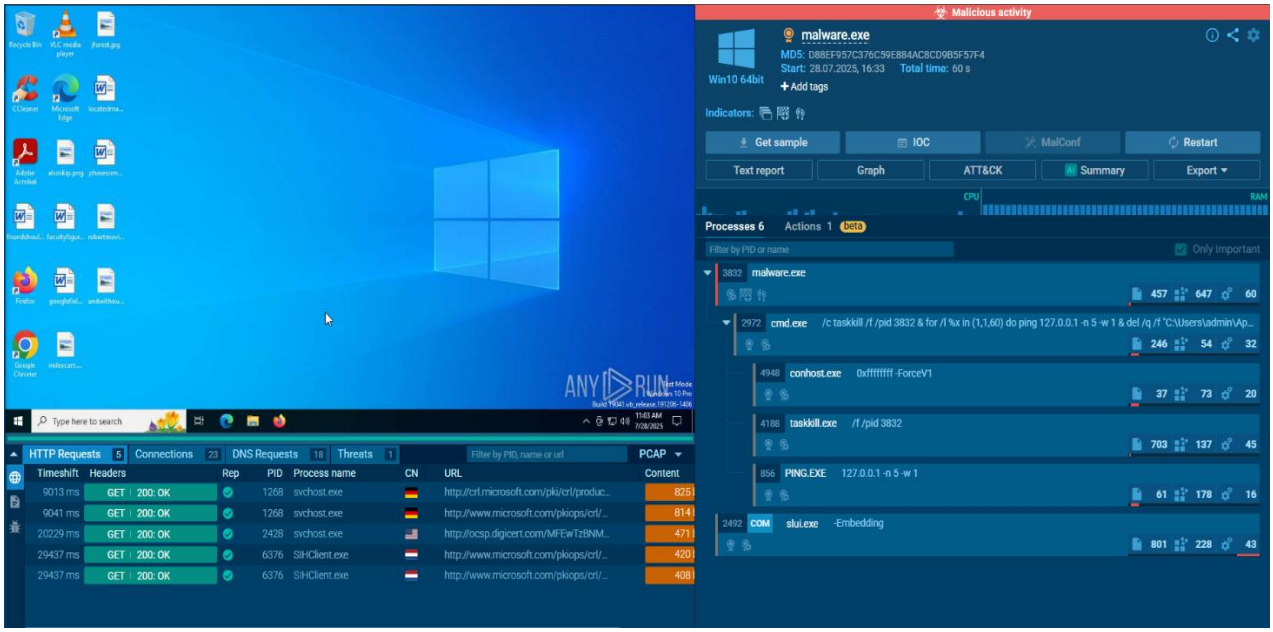
While static analysis reveals the malware's potential capabilities by examining its blueprint, behavioral (or dynamic) analysis provides definitive proof of its actions by observing it in a live, controlled environment. The goal of this phase is to execute the malware and meticulously record its effects on the host system and its network communications. This allows us to confirm the purpose of the Indicators of Compromise (IOCs) discovered statically and to map out the attack's entire execution flow from start to finish
To safely analyze the malware's runtime behavior, we utilized the **Any.Run interactive sandbox**. This platform provides a fully instrumented Windows 10 virtual machine that allows the malware to execute while monitoring every system call, process creation, registry modification

Any.Run is an interactive **malware sandbox** that allows researchers to **observe the behavior** of malicious files in real time within a controlled virtual environment.

Why I Used It:

- To safely **execute and monitor** the malware sample without risking my system.
- To **observe behavioral patterns**, such as:
    - File and registry modifications
    - Network communications
    - Command-line execution
    - Process creation and injection

The sample malware.exe exhibits self-deletion, persistence techniques, registry modification, and system discovery tactics.



malware.exe (PID: 3832)

└─ cmd.exe (PID: 2972)

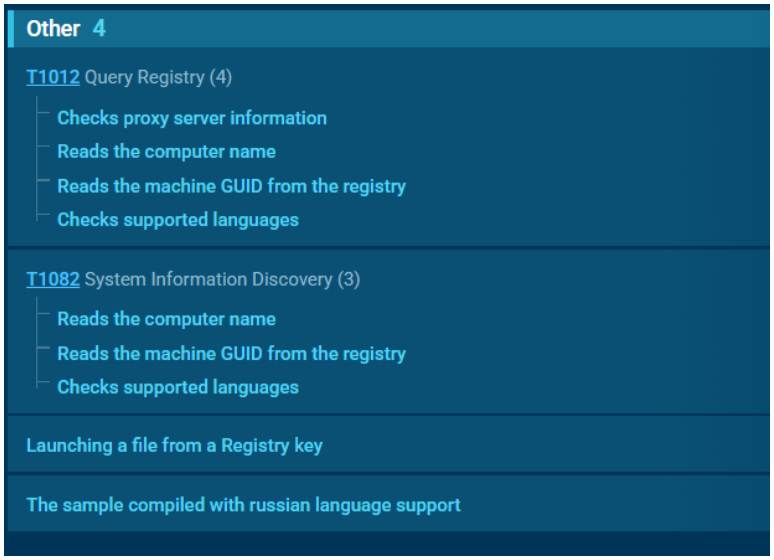
  ├─ conhost.exe (PID: 4948)

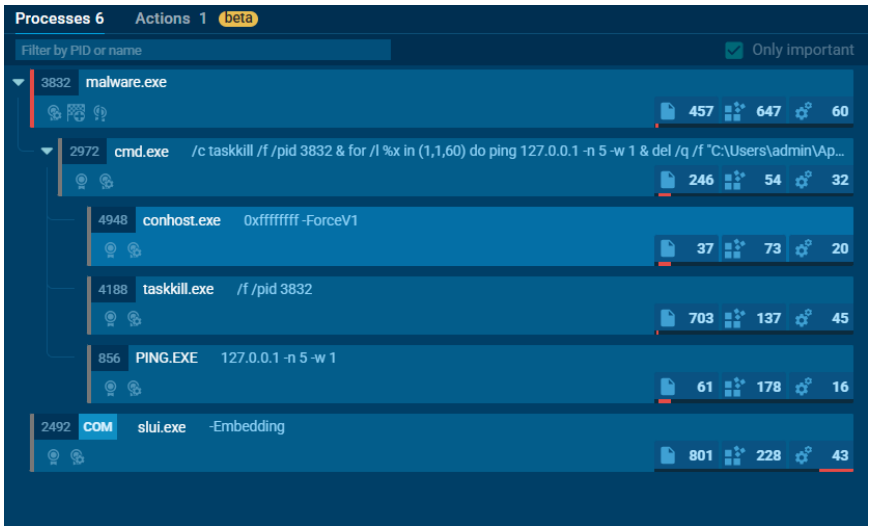
  ├─ taskkill.exe (PID: 4188)

  ├─ ping.exe (PID: 856)

slui.exe (PID: 2492)

Technique ID	**MITRE Technique**	**Description**
T1070.004	File Deletion	Deletes itself using command shell
T1547.001	Registry Run Keys / Startup Folder	Likely achieves persistence via autorun registry entry
T1059.003	Windows Command Shell	Executes commands via cmd.exe
T1012	Query Registry	Gathers system config: IE settings, proxy info
T1082	System Information Discovery	Reads system identifiers and language settings





9. Registry changes caused by malware

Timestamp	Action	Value Name	Registry Key	Data
+58 ms	Write	lecjxegcxm	HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce	C:\Users\admin\AppData\Local\Temp\malware.exe
+89 ms	Delete Value	lecjxegcxm	HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce	C:\Users\admin\AppData\Local\Temp\malware.exe
+136 ms	Write	ProxyBypass	HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap	1
+136 ms	Write	IntranetName	HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap	1
+136 ms	Write	UNCAsIntranet	HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap	1
+136 ms	Write	AutoDetect	HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap	0
+136 ms	Write (again)	ProxyBypass	HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap	1
+136 ms	Write (again)	IntranetName	HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap	1
+136 ms	Write (again)	UNCAsIntranet	HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap	1
+136 ms	Write (again)	AutoDetect	HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap	*No value shown*

Revoked TIFLOSOFT Certificate

10. Behavioural Analysis

MALICIOUS	SUSPICIOUS	INFO
Changes the autorun value in the registry • malware.exe (PID: 3580)	Reads security settings of Internet Explorer • malware.exe (PID: 3580)	Checks supported languages • malware.exe (PID: 3580)
Starts CMD.EXE for self-deleting • malware.exe (PID: 3580)	Starts CMD.EXE for commands execution • malware.exe (PID: 3580)	The sample compiled with russian language support • malware.exe (PID: 3580)
	Uses TASKKILL.EXE to kill process • cmd.exe (PID: 3800)	Reads the computer name • malware.exe (PID: 3580)
	Runs PING.EXE to delay simulation • cmd.exe (PID: 3800)	Launching a file from a Registry key • malware.exe (PID: 3580)
		Reads the machine GUID from the registry • malware.exe (PID: 3580)
		Checks proxy server information • malware.exe (PID: 3580)

11. Network behavioural Analysis – wireshark

To observe the malware's network communications in real-time, we configured our isolated Windows 10 virtual machine for live traffic capture. The goal was to intercept and analyze every packet sent and received by the malware upon execution, providing indisputable evidence of its network-based indicators and communication patterns.

Methodology: A Controlled Experiment

Our procedure was methodical, designed to ensure a clean and accurate capture of the malware's activity and nothing else.

1. Environment Preparation: We started our Malware Present - Pre-Execution snapshot, providing a clean, known state for the Windows 10 VM.

2. Capture Initiation: Before running the malware, we launched Wireshark inside the VM. We selected the active network interface (our "Host-only Adapter") and initiated a live packet capture. At this point, Wireshark was actively listening to all network traffic.
3. Detonation: With the capture running, we navigated to the C:\Analysis folder and executed malware.exe.
4. Observation & Termination: We allowed the malware to run for approximately 60 seconds to ensure its full initial execution sequence could complete. We then stopped the Wireshark capture, freezing the collected data for analysis.

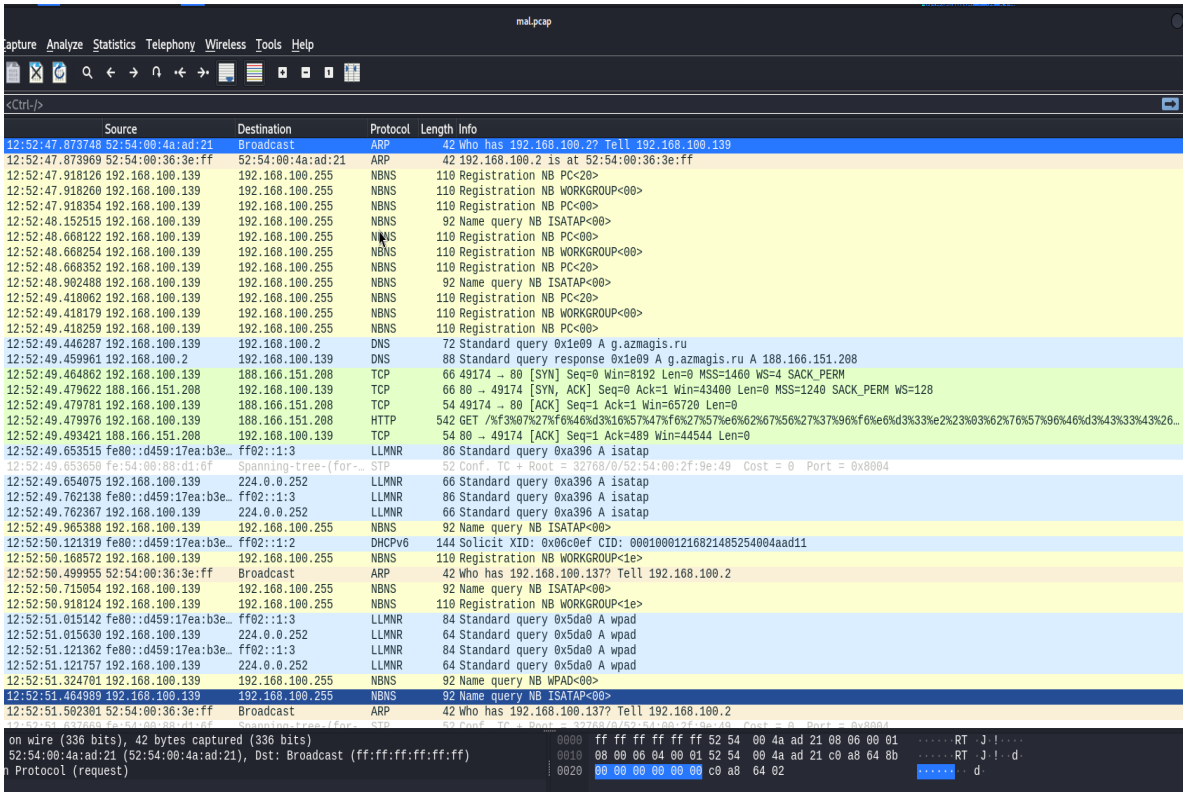
Key Findings: Mapping the C2 Communication Flow

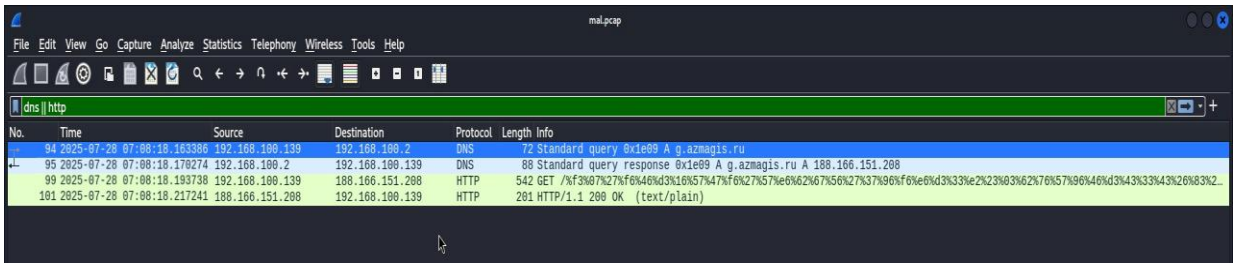
The captured packets, when analyzed in Wireshark, told a clear and concise story of the malware's "phoning home" process.

- **Step 1: DNS Resolution** (Finding the C2 Server)
    - Our Analysis: The first action we looked for was a DNS query. By applying the display filter dns in Wireshark, the background noise vanished, and we immediately isolated an outbound "Standard query A" from our infected host for the domain name **g.azmagis.ru**. This was followed moments later by a "Standard query response" from the network's DNS server, providing the corresponding IP address: **188.166.151.208**.
    - Significance: This was a critical first step. It proved that the domain we found during our static analysis in Ghidra was not a leftover artifact but an active, primary component of the malware's operation. The malware needs to perform this lookup to translate the human-readable domain into the machine-routable IP address of its C2 server.

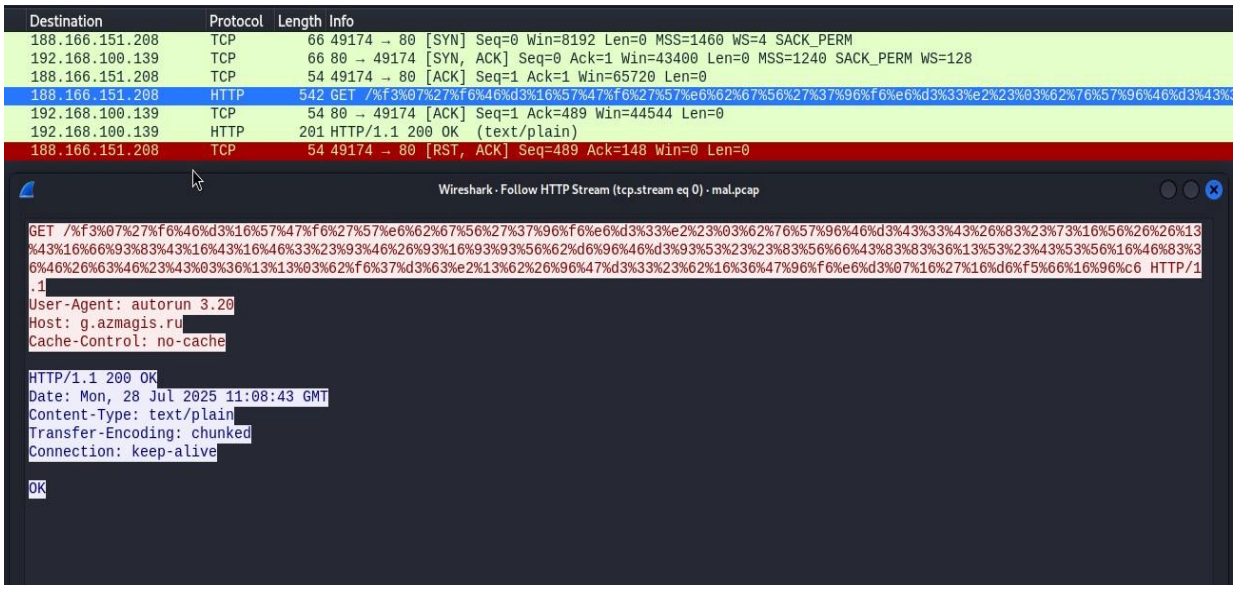
- **Step 2: Establishing a Connection** (The TCP Handshake)
    - Our Analysis: Following the DNS resolution, we traced the traffic to the resolved IP address. We observed the classic three-way TCP handshake sequence, confirming the establishment of a connection:
        1. [SYN] Our host sent a SYN packet to **188.166.151.208** on destination port 80.
        2. [SYN, ACK] The C2 server promptly responded with a SYN, ACK packet.
        3. [ACK] Our host finalized the connection with an ACK packet.

- **Step 3: The Beacon and C2 Response** (The HTTP Conversation)
    - Our Analysis: To reconstruct the conversation, we right-clicked on an HTTP packet within the stream and used Wireshark's powerful "Follow > TCP Stream" feature. This presented the data exchange in a clean, readable format.
        - **The Beacon (Client to Server)**: We observed the malware sending **an GET /%f3%07... HTTP/1.1 request.** This is the malware's beacon, signaling a new infection to the C2 operator. The long, hex-encoded string in the URI is likely a unique identifier for our specific VM.
        - The Orders (Server to Client): The C2 server then replied with a HTTP/1.1 200 OK message. The data contained within the body of this response represents the next-stage instructions being sent *down* to our infected machine.
    - Significance: This was the main thing of our network analysis. We successfully captured the malware not only calling out but also receiving a valid response, proving the C2 server was active and operational.
- Step 4: Rapid Termination (The Cover-Up)
    - Our Analysis: The final packet we observed in this stream was a TCP packet with the [RST, ACK] flags set, sent from our host to the C2 server

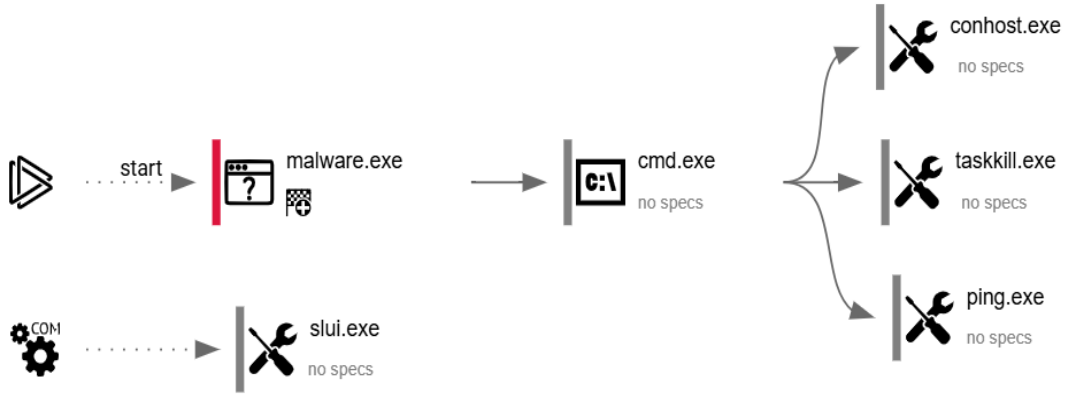




http get request to g.azmagis.ru



12 . Behavioural Analysis graph for Malware execution logic



Process	Role / Function Observed
malware.exe	Main payload. Initiates execution chain, launches cmd.exe with a self-deletion script.
cmd.exe	Executes commands on behalf of malware.exe. Includes process killing, delays, and deletion.
conhost.exe	Console host used by cmd.exe. Typical companion for CMD execution.
taskkill.exe	Forcefully terminates the malware.exe process (PID specified). Part of self-deletion logic.
ping.exe	Simulates delay via loop (sleep function) — often used to bypass sandboxes or time actions.
slui.exe	Executed via COM object. Legitimate Windows binary, possibly abused for stealth or evasion.

The Main Malicious Execution Flow (Top Path)

1 **. start -> malware.exe:** The graph begins with the initial execution of our primary sample, malware.exe. This is the parent process of the entire malicious routine.

2. malware.exe -> cmd.exe: The first action of the malware is to spawn cmd.exe, the legitimate Windows Command Prompt. This is a critical step in its **"Living off the Land"** strategy. Instead of using its own code to perform system actions, it outsources the work to a trusted Microsoft binary, making its behavior appear less suspicious at first glance.

3. cmd.exe -> conhost.exe: The conhost.exe (Console Window Host) process is automatically launched by the system whenever a command-line application like cmd.exe is started. Its presence is normal and expected.

4.cmd.exe -> taskkill.exe: This is the first part of the self-destruct sequence. We see cmd.exe being used to launch taskkill.exe, the Windows utility for terminating processes. The malware uses this to kill its own parent process, malware.exe.

5.cmd.exe -> ping.exe: This is the second part of the sequence. cmd.exe launches ping.exe. As established from our log analysis, this is not for network discovery but is a simple time-delay tactic designed to evade automated analysis environments.

Path 2: The Benign System Process (Bottom Path)

.COM -> slui.exe: This secondary path shows the execution of slui.exe, which is the Windows Activation Client. Its execution appears to be initiated by a COM object. This is likely a normal, background Windows process that happened to run during the sandbox analysis and is not directly related to the malware's primary malicious activity.

13 . MITRE Attack Matrix

14. Summary of the Malware and Conclusion

The Primary Dangers and a Trojan's True Intent

The primary danger of this specific malware sample does not lie in the initial actions we observed. The self-deletion, the ping commands, and even the registry key creation are just the means to an end. They are the "work" the malware does to set up its real attack.

The true danger and the main intent of this malware are centered on one single event:

The successful HTTP beacon to the Command and Control (C2) server.

This malware is a Trojan Downloader or "Dropper". Its entire purpose is to be **the initial point of entry**—the "foot in the door" for a much more severe infection.

What It Mainly Does and How It Affects the System:

1. **It Establishes a Hidden, Persistent Foothold:** The malware's first priority is to infect the machine and ensure it can't be easily removed. It achieves this by writing itself to the RunOnce registry key. This means even if the user reboots their computer, the malware will run again, re-establishing its control. The user will be completely unaware that the infection has survived.
2. **It Creates a Covert Communication Channel:** Its second priority is to open a "backdoor" communication channel to its masters. It uses standard HTTP traffic on port 80, which is allowed by almost every firewall on the planet, making its C2 traffic blend in with normal web browsing.
3. **It Waits for Further Instructions:** The HTTP 200 OK response it received from the C2 server is where the real attack begins. That response contained the *next set of commands*. This is what makes a Trojan so dangerous—its payload is modular and can change at any time based on the attacker's goals.

Summary of Dangers:

- **Initial Access Broker:** This malware acts as a scout. Its successful infection can be sold to other criminals on the dark web. An attacker might sell access to your infected machine to a ransomware gang, a banking trojan operator, or a state-sponsored espionage group.
- **Payload Delivery System:** The attacker can use the C2 channel to command the malware to download and run anything they want. This could be:
    - Ransomware: To encrypt all your files and demand payment.
    - Spyware/Keyloggers: To steal your passwords, banking information, and personal data.
    - Cryptominers: To use your computer's resources to mine cryptocurrency, slowing down your machine and increasing your electricity bill.
- **Platform for Lateral Movement:** Once inside your network, an attacker can use the compromised machine as a launchpad to attack other, more valuable computers on the same network, such as servers or domain controllers.

Conclusion

The comprehensive static and dynamic analysis of the file **malware.exe/autorun.exe/0af36beb-ae92-11e6-bdc4-80e65024849a.exe** (SHA-256: 572b75...) provides conclusive evidence that it is a moderately sophisticated Trojan downloader, consistent with the RuKometa malware family. The investigation successfully mapped its entire initial execution lifecycle, from its blueprint to its live behavior.

Static analysis : Revealed the malware's deceptive nature, highlighted by its use of a revoked "TIFLOSOFT" digital certificate and the obfuscation of critical indicators—such as its C2 domain (g.azmagis.ru) and self-destruct commands—by encoding them in Unicode (UTF-16).

Furthermore, the presence of the **SeTakeOwnershipPrivilege** string indicates a built-in intent for defense evasion and privilege escalation.

Behavioral analysis confirmed these static findings and brought the malware's full operational sequence to light. Upon execution, the malware's primary objectives are systematically achieved:

1. **Persistence:** It first secures its longevity by writing its execution path to the HKCU\...\RunOnce registry key.
2. **C2 Communication:** It then beacons out to its C2 server via an HTTP GET request, confirming a successful infection and opening a channel for receiving further commands.
3. **Evasion:** Finally, it attempts to erase its initial footprint by initiating a self-deletion routine using legitimate Windows tools (cmd.exe, taskkill.exe), a classic "Living off the Land" technique.

In essence, the analysed sample is a purpose-built initial access tool. Its danger is not in its immediate, observable actions, but in its success as a "dropper." The successful C2 beacon represents the opening of a backdoor, through which any number of more severe threats can be deployed onto the now-compromised and persistent host.

Report Compiled By:

Vishwas S Adhikari
Security Researcher
Intern ID : 135
Email : vishwas.s.1002@gmail.com
Phone : 9380590044
Date of Completion: July 28, 2025