XWorm: Technical Analysis of a New Malware Version

By Electron, kinoshi and glebyao

Published: 2023-08-24 · Archived: 2026-04-06 01:07:26 UTC

In this article, we will take a look at the latest version of an XWorm sample — a widespread malicious program that is advertised for sale on underground forums.

We will analyze the functionality of our sample, as well as extract its configuration.

Let’s get started.

What is XWorm Malware?

XWorm is a malware that targets Windows operating systems. It is known for its stealth and persistence, and a wide range of malicious activities, spanning from remote desktop control to ransomware and information theft.

Unfortunately, adversaries employ this threat widely —it’s not uncommon to see it in ANY.RUN’s top 10 most used malware by uploads.

XWorm dynamic sandbox analysis

While searching for new threats, we discovered an interesting sample, uploaded by our users to Public submissions. It was downloaded from the file hosting “Mediafire” in a RAR archive with a password:


A sample with a RAR archive in ANY.RUN

After launching, the threat was identified by Suricata’s network rules as XWorm:


XWorm is identified in ANY.RUN

We decided to check the sample on VT to confirm that it was indeed XWorm, but at the time of writing this article, we were unable to find it there:


VT is missing Xworm sample

The initial analysis, according to the indicators set on process 2784, revealed that the software adds its shortcut to the startup (**MITRE T1547.001**) and uses the task scheduler (**MITRE T1053.005**):

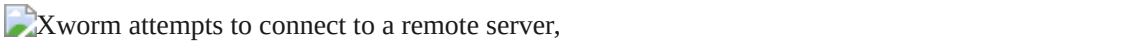

The initial Xworm analysis

The use of the scheduler is necessary to restart the software with elevated privileges, as indicated by the “/RL HIGHEST” parameter.


Restart the software with elevated privilege

According to the file operation data, the software is installed in the Public directory (**MITRE T1074.001**):

Interestingly, the software attempts to connect to a remote server, but no response is received (**MITRE T1571**):



We decided to restart the sample and check for additional activities. Unfortunately, it crashed almost immediately after launch:

We became interested in investigating the cause of the “crash,” and we found that the user-launched sample and the sample restarted by us exhibited different behavior patterns. Specifically, the restarted sample queries a service to determine the external IP address (**MITRE T1590.005**) before crashing. Typically, in addition to the IP address, such services provide the ability to determine whether the software is running on a virtual host:

This is precisely what XWorm does — it attempts to verify whether it’s running on a user’s physical machine or not.

To solve this problem, ANY.RUN has a useful feature called **Residential Proxy** which allows you to hide your actual location and convinces the software that it’s running on a real user’s machine. You can choose any location, in case it’s targeted malware requiring IP addresses from specific countries:

Restarting with the Residential Proxy option enabled was successful, and XWorm exhibited its activity.

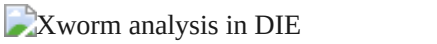
Additionally, we activated the MITM proxy option to find out what data is being transmitted to Telegram (**MITRE T1102**):


Xworm transmitts data to Telegram

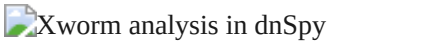
It's evident that the software transmits its version (XWorm V3.1), the machine's username, the operating system version, and likely a hash of a new victim (**MITRE T1082**).

Xworm static analysis

The first step is to place our subject into the DIE — a utility for initial analysis.

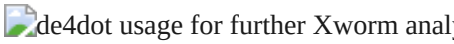

Xworm analysis in DIE

As we can see, we are dealing with a .NET variation, so we promptly opened it in dnSpy.


Xworm analysis in dnSpy

We are immediately met with an unfavorable picture — all the program's members were subjected to obfuscation (**MITRE T1027**). DIE could not recognize the packer even with the "Heuristic scan" being checked.

Our first thought was to try using de4dot to simplify further analysis.


de4dot usage for further Xworm analysis


Xworm analysis

As we can see, not much has changed, so we must continue analyzing what we have.

Reverse engineering: additional anti-evasion techniques and persistence gain

To slow down the analysis and hide from detection systems, the sample employs the following technologies:

1. Virtualization detection using the WMI query "Select * from Win32_ComputerSystem" and checking for operation within VmWare or VirtualBox environments (**MITRE T1047**)


Detection of a virtual machine

2. Debugger detection using the CheckRemoteDebuggerPresent API function


Detection of the debugger using CheckRemoteDebuggerPresent

3. Checking for the loaded dynamic library SbieDll.dll, characteristic of Sandboxie, which is a sandbox-based isolation program.


Detection of Sandboxie

4. A query to check whether the current machine is hosted or located in a data center (this finally clarifies why the sample initially “crashed”)

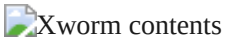

Detection of hosting

The sample also gains a foothold by utilizing the registry and the task scheduler:


Xworm utilizes the registry and the task scheduler

Reverse engineering: Xworm config extraction

After a brief review of the methods’ contents, a constructor was found that bears a striking resemblance to a block containing settings.


Xworm contents

After examining cross-references, we arrive at a method that looks like this:


Xworm reverse-engineering

As we can see, some fields undergo a reassignment stage, after processing by the method “Vc1fSJ4D04O6qGeP2fzA5lFCv8a7buXvJb4sHwuhuifI09pX.” Let’s take a closer look at it.


Xworm reverse-engineering

First, an MD5 hash is computed from the value of the field “hArf0quX6jL4F88ywQTiLn52eBzsJ6HreaOqb0WGSa89u” from the presumed settings section.

Then the obtained value is copied twice into a temporary array (perhaps the malware developer made an off-by-one error when using the Array.Copy method, resulting in the MD5 not being copied entirely twice; the last copied byte after the first copying is overwritten by the subsequent copying, so that the last byte in the resulting array is always zero). The obtained array is used as a key to decrypt the incoming base64 strings using AES in ECB mode.

It’s also interesting that the field used is also a mutex.


Xworm mutex

Now we have all the necessary information for decrypting the settings.


Xworm reverse-engineering

Our final AES key looks like this: “01d31d5e811fce422987107f962c4001d31d5e811fce422987107f962c406600.”


Xworm reverse-engineering

And here we have reached the core of our target’s sample.

The result can be viewed in CyberChef here.

The final config mapping is as follows:

Host	6[.]tcp.eu.ngrok[.]io
Port	13394
AES key	Slaves!-.;!2Swezy999!(xxx
Splitter	Xwormmm
Sleep time	3
USB drop file	USB.exe
Mutex	Lz8qftMH08V7f1rq
Log file	%temp%\\Log.tmp
Telegram token	6674821695:AAExQsr6_hmXk6hz7CN4kMSi9cs9y86daYM
Telegram chat id	5865520781

When the goal isn't to study the malware in-depth but rather to quickly obtain the configuration, this can be efficiently achieved by running the sample in ANY.RUN. This method provides a straightforward way to access the necessary information without the need for extensive analysis, saving potentially hours of work.


Xworm malware configuration in ANY.RUN

See it in action for yourself here.

IOCs

Analyzed files

MD5	F6BB396FD836F66CD9F33CA4B0262DD7
SHA1	BFC7036E32A59AC25DB505D263B5F4CADE24C53C
SHA256	1073FF4689CB536805D2881988B72853B029040F446AF5CED18D1BC08B2266E1
SS	6144:bfIbSc83qUhcX7elbKTua9bfF/H9d9n+:bLc83q3X3u+G

MITRE (ARMATTACK)

Tactic	Technique	Description
TA0003: Persistence	T1547: Registry Run Keys / Startup Folder	Adds a shortcut to the startup folder
TA0003: Persistence	T1053: Scheduled Task	Uses the task scheduler
TA0009: Collection	T1074: Local Data Staging	The malware saves itself in the Public directory
TA0011: Command and Control	T1571: Non-Standard Port	Connects to a remote server
TA0043: Reconnaissance	T1590: IP Addresses	Checks the IP of the running system
TA0011: Command and Control	T1102: Bidirectional Communication	Communicates through Telegram
TA0007: Discovery	T1082: System Information Discovery	Collects information about the victim's computer
TA0005: Defense Evasion	T1027: Command Obfuscation	Obfuscates the executable file
TA0002: Execution	T1047: Windows Management Instrumentation	Gathers system information to detect virtualization
TA0005: Defense Evasion	T1027: Embedded Payloads	Stores information in a mutex

DNS requests

- 6[.]tcp[.]eu[.]ngrok[.]io

More samples for your research

https://app.any.run/tasks/d3858744-f1b2-4a9b-8ef7-deccada2a160/

https://app.any.run/tasks/75f66fd6-d989-4f06-a348-c65e135e8ab4/

https://app.any.run/tasks/5fab7db5-267e-46f6-a374-0f42de1cb328/

https://app.any.run/tasks/b9275944-39fe-42cb-9eae-6b2e05f0892f/

https://app.any.run/tasks/803758bf-387b-42e2-80cc-f20e7140cac4/

Interested in more content like this? Check out our in-depth analysis of the latest .NET variant of LaplasClipper or read a break-down and guide to GuLoader deobfuscation strategies.

A few words about ANY.RUN

ANY.RUN is a cloud malware sandbox that handles the heavy lifting of malware analysis for SOC and DFIR teams. Every day, 300,000 professionals use our platform to investigate incidents and streamline threat analysis.

Request a demo today and enjoy 14 days of free access to our Enterprise plan.

Request demo →

Electron

I'm a malware analyst. I love CTF, reversing, and pwn. Off-screen, I enjoy the simplicity of biking, walking, and hiking.

kinoshi

I'm a dedicated programmer and malware analyst. I derive immense joy from the art of coding and have a deep passion for both low-level and system-level programming. I thoroughly enjoy delving into the intricacies of software and exploring how it operates at a fundamental level. My expertise extends to solving crackme challenges and participating in online CTF competitions, where I tackle complex tasks to enhance my skills.

glebyao

I am a 19-year-old malware analyst, programming in C / C++ / Python. My passion is to reverse-engineer applications of my interest. In my spare time, I participate in CTF events or develop tasks for reverse engineering, pwn, PPC, and other categories.

Electron

Leading malware analyst

I'm a malware analyst. I love CTF, reversing, and pwn. Off-screen, I enjoy the simplicity of biking, walking, and hiking.

kinoshi

Malware analyst at ANY.RUN

I'm a dedicated programmer and malware analyst. I derive immense joy from the art of coding and have a deep passion for both low-level and system-level programming. I thoroughly enjoy delving into the intricacies of

software and exploring how it operates at a fundamental level. My expertise extends to solving crackme challenges and participating in online CTF competitions, where I tackle complex tasks to enhance my skills.

glebyao

glebyao

Malware analyst at ANY.RUN

I am a 19-year-old malware analyst, programming in C / C++ / Python. My passion is to reverse-engineer applications of my interest. In my spare time, I participate in CTF events or develop tasks for reverse engineering, pwn, PPC, and other categories.
