www.trendmicro.com /en_th/research/23/b/investigating-the-plugx-trojan-disguised-as-a-legitimate-windows.html

Investigating the PlugX Trojan Disguised as a Legitimate Windows Debugger Tool

⋮ 2/24/2023

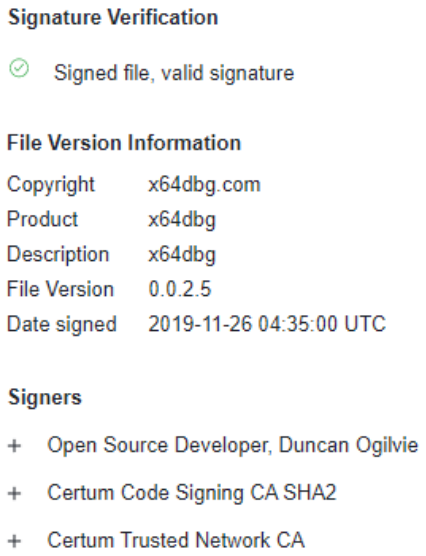


Figure 1. A digitally signed x32dbg.exe (ec5cf913773459da0fd30bb282fb0144b85717aa6ce660e81a0bad24a2f23e15)

Introduction

Trend Micro's Managed Extended Detection and Response (MxDR) team discovered that a file called x32dbg.exe was used (via the DLL Search Order Hijacking or T1574.001 technique) to sideload a malicious DLL we identified as a variant of PlugX (Trojan.Win32.KORPLUG.AJ.enc). This file is a legitimate open-source debugger tool for Windows that is generally used to examine kernel-mode and user-mode code, crash dumps, or CPU registers. Meanwhile, PlugX is a well-known remote access trojan (RAT) that is used to gain remote access to and control over compromised machines. It allows an attacker to obtain unauthorized access to a system, steal sensitive data, and use the compromised machine for malicious purposes. The MxDR team employed a number of advanced security technologies and solutions to gain a comprehensive understanding of the attack, which will be revealed in this report.

Investigating and analyzing the threat with MxDR

Being a legitimate application, x32dbg.exe's valid digital signature can confuse some security tools, enabling threat actors to fly under the radar, maintain persistence, escalate privileges, and bypass file execution restrictions.

The team's attention was first drawn to the command line execution of D:\RECYCLER.BIN\files\x32dbg.exe which was flagged by a VisionOne Workbench alert. Further investigation revealed that this path led to a hidden folder on the USB storage device, which was found to contain a number of threat components.

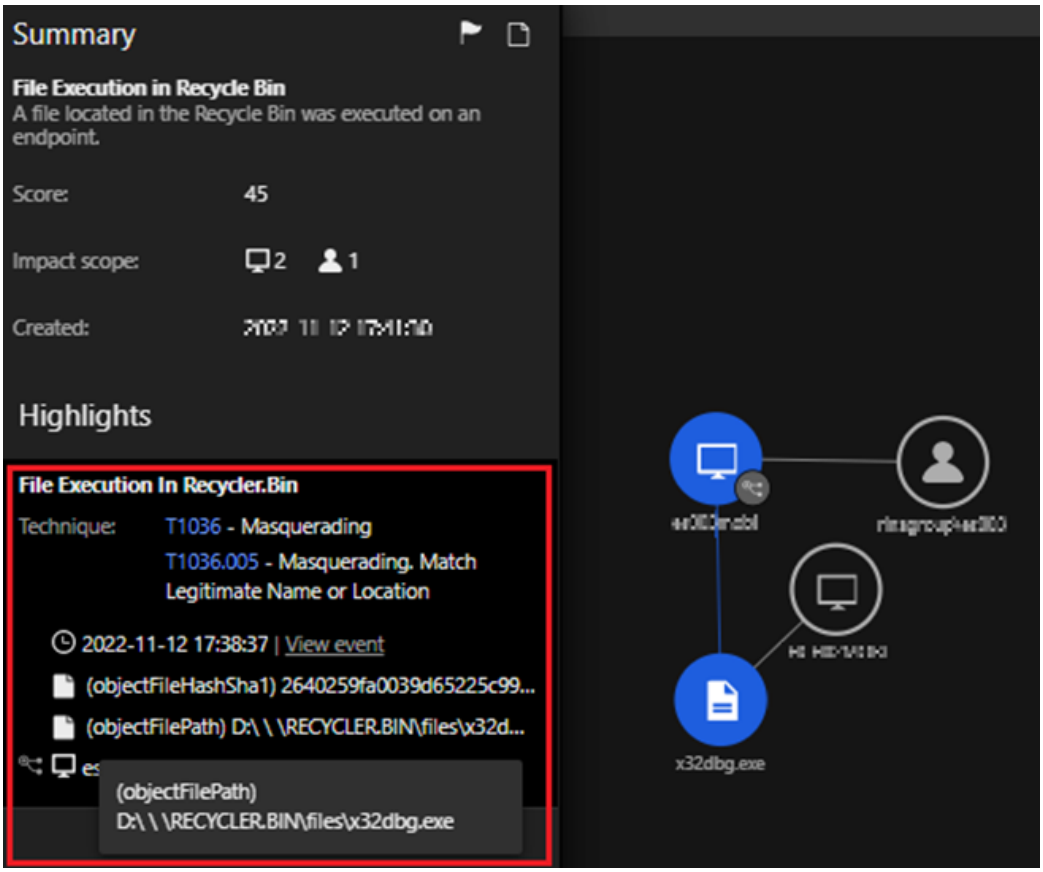


Figure 2. Workbench model triggered by the execution of x32dbg.exe

We uncovered a clear sequence of events that began with a suspicious command line execution launched via cmd.exe. The command line executed the file (ec5cf913773459da0fd30bb282fb0144b85717aa6ce660e81a0bad24a2f23e15) located at D:\RECYCLER.BIN\files\x32dbg.exe. The file was signed by ”OpenSource Developer, Duncan Ogilvie” issued by Certum Code Signing. A visual representation of these events is displayed in Figure 3.

Command Line: "C:\Windows\System32\cmd.exe" /q /c "\ \RECYCLER.BIN\files\x32dbg.exe"

File Path: "D:\ \ \RECYCLER.BIN\files\x32dbg.exe"

SHA256: ec5cf913773459da0fd30bb282fb0144b85717aa6ce660e81a0bad24a2f23e15

Signer: Open-Source Developer, Duncan Ogilvie

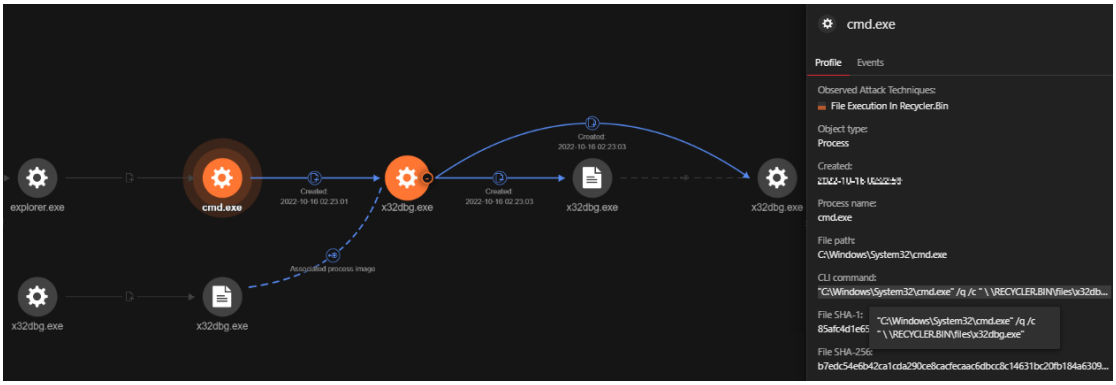


Figure 3. Vision One shows how cmd.exe calls x32dbg.exe from the external/non-system drive

After executing D:\RECYCLER.BIN\files\x32dbg.exe, all of the threat components are copied to the directory C:\ProgramData\UsersDate\Windows_NT\Windows\User\Desktop.

Subsequently, the file C:\ProgramData\UsersDate\Windows_NT\Windows\User\Desktop\x32dbg.exe, a duplicate of the original file, was invoked. The following command line was used to invoke the dropped file:

Command Line: "C:\Windows\System32\cmd.exe" /q /c" C:\ProgramData\UsersDate\Windows_NT\Windows\User\Desktop//x32dbg.exe"

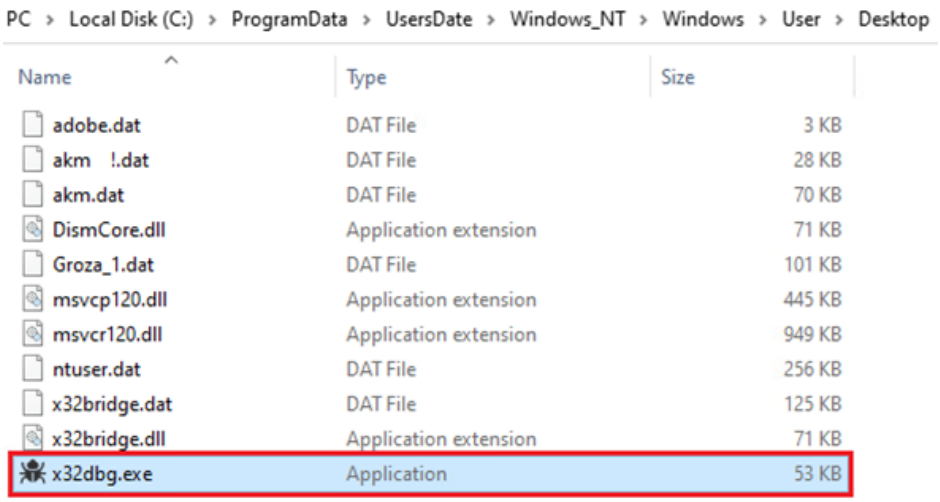


Figure 4. Files created in C:\ProgramData\UsersDate\Windows_NT\Windows\User\Desktop

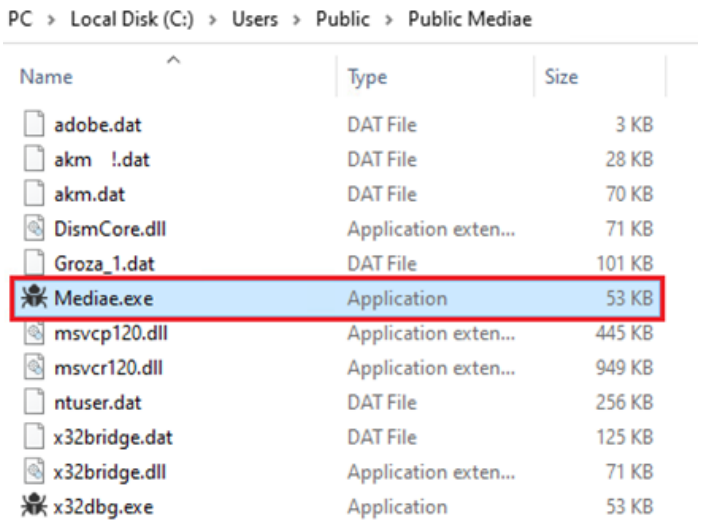


Figure 5. Files created “C:\Users\Public\Public Mediae”

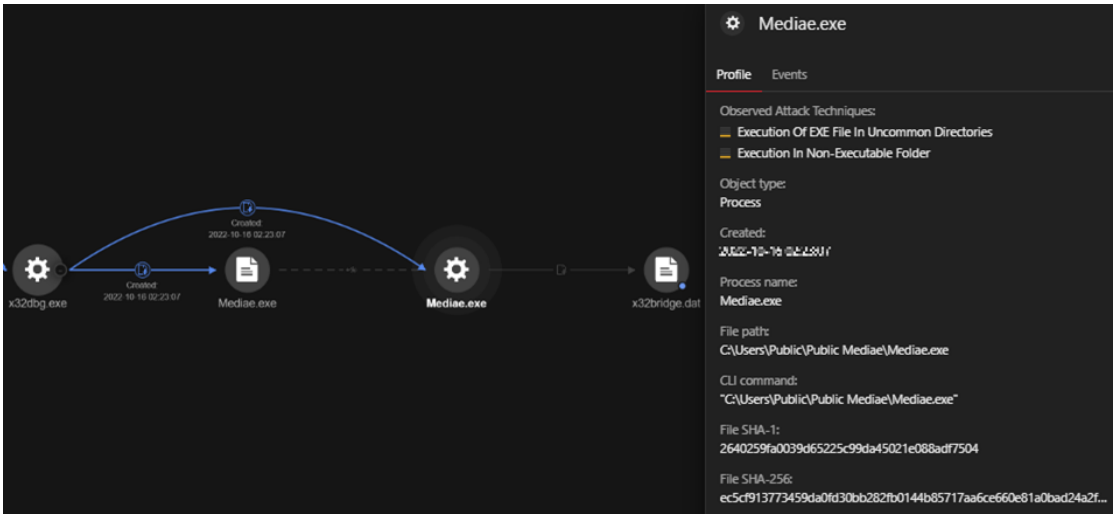


Figure 6. Vision Ones shows how x32dbg.exe copies itself to various directories and renames itself as Mediae.exe

C:\Users\Public\Public Mediae\Mediae.exe followed the same procedure, creating a new directory at C:\Users<username>\Users\ and copying the identical files as shown in Figure 7.

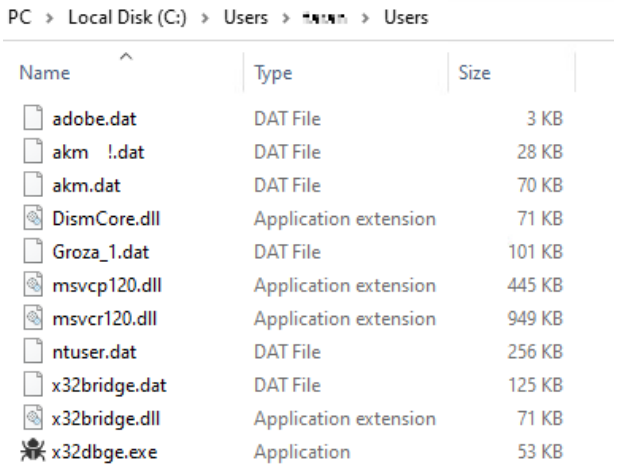


Figure 7. The same set of files were created in C:\Users\<username>\Users\

As a result, a full set of the same files were present in three different directories. This indicated a clear attempt to establish persistence and evade detection by placing copies of the malicious files in multiple locations in the compromised system, specifically:

- C:\ProgramData\UsersDate\Windows_NT\Windows\User\Desktop
- C:\Users\Public\Public Mediae\
- C:\Users\<username>\Users\

Analyzing persistence: how the attacker maintained access

To ensure continued access to the compromised systems, attacker used techniques involving the installation of persistence in the registry, the creation of scheduled tasks to maintain access (even in case of system restarts), the implementation of changes in credentials, and other potential disruptions that could result in lost access.

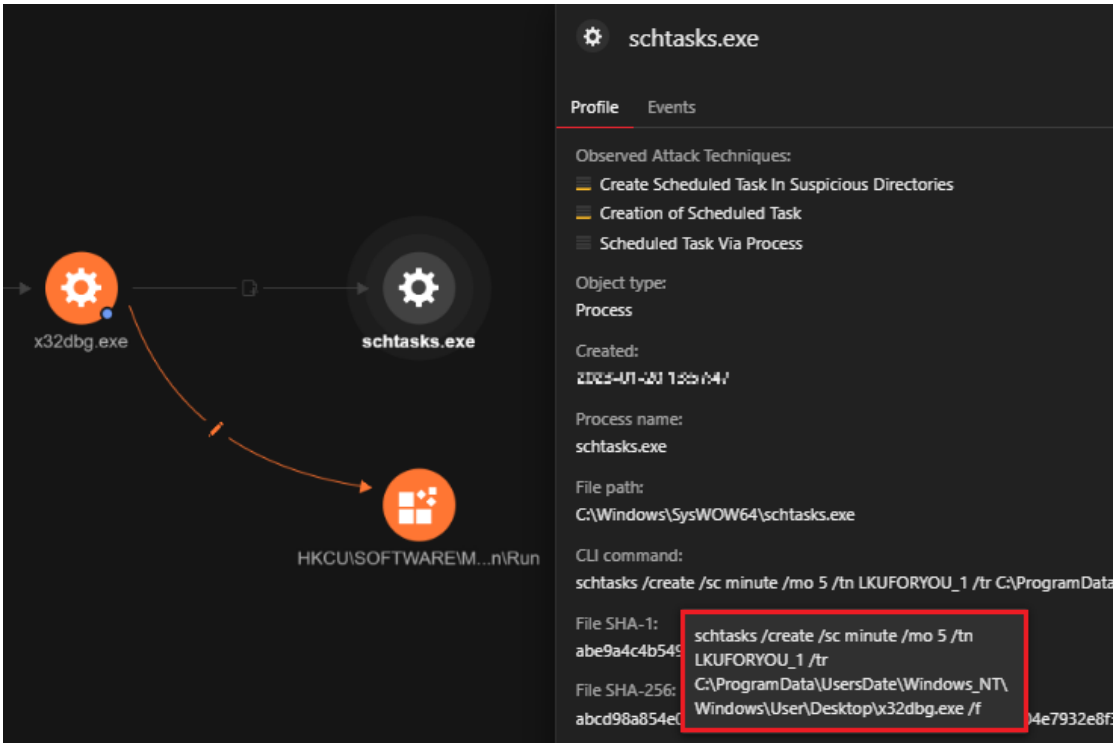


Figure 8. Persistence was created in the scheduled task and run registry

We noticed the creation of a scheduled task via the schtasks command line utility to run a task at a specific time. In this case, the scheduled task is set to execute the x32dbg.exe file, the open source debugger tool that side loads PlugX, every five minutes. The task is disguised under the name "LKUFORYOU_1" to make it more difficult to detect.

Commandline: schtasks /create /sc minute /mo 5 /tn LKUFORYOU_1 /tr C:\ProgramData\UsersDate\Windows_NT\Windows\User\Desktop\x32dbg.exe /f

A brief summary of the parameters used:

- /create: This option instructs the utility to create a new scheduled task.
- /sc minute: This option specifies the frequency at which the task will be executed, which in this case is every five minutes.
- /mo 5: This option sets the duration of the frequency for the scheduled task.
- /tn LKUFORYOU_1: This option sets the name of the task as "LKUFORYOU_1".
- /tr C:\ProgramData\UsersDate\Windows_NT\Windows\User\Desktop\x32dbg.exe: This option specifies the path of the executable that will be executed when the task is triggered.
- /f: This option forces the task to be created without requiring user confirmation.

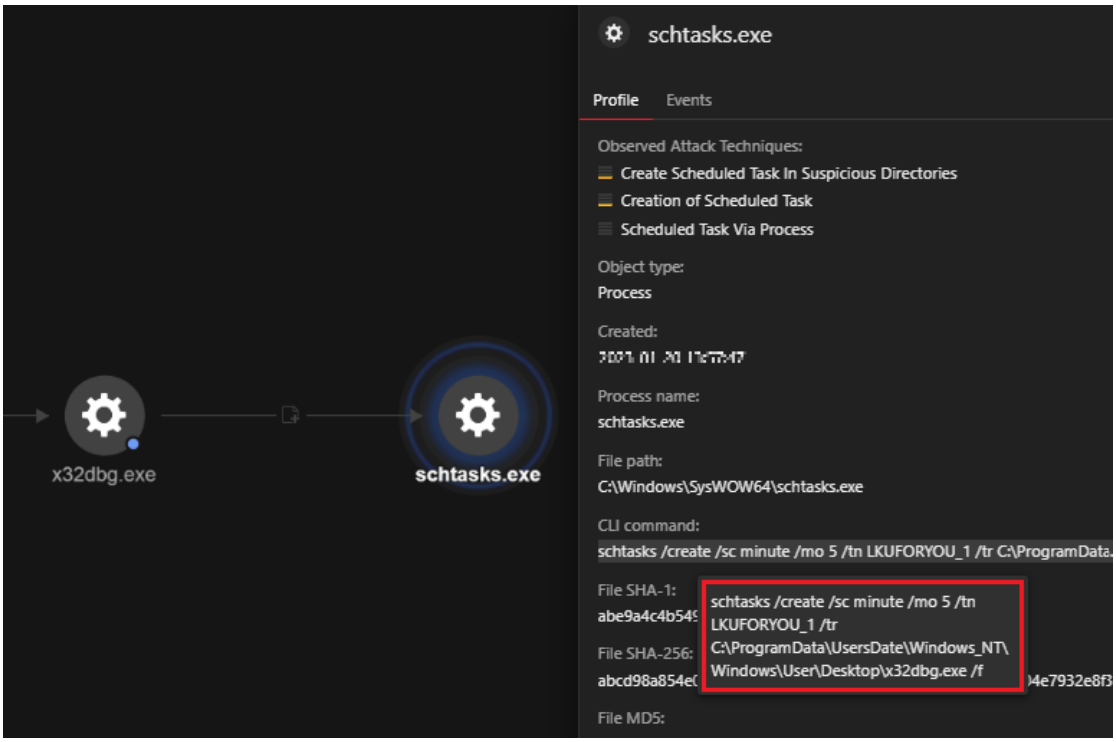
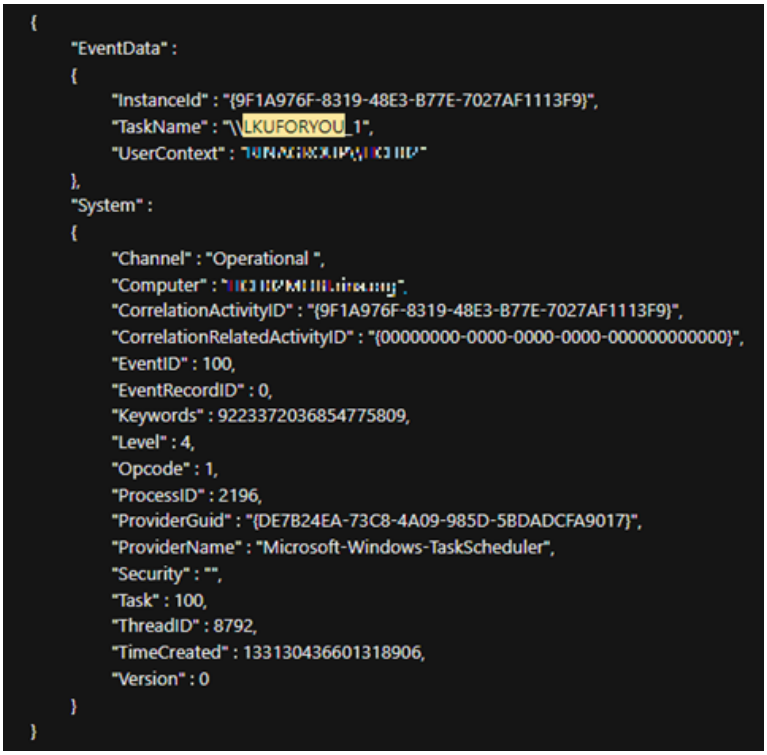


Figure 9. The schtask utility was used to create persistence in the scheduled task

Further evidence supporting the persistence created by the scheduled task was discovered in the event logs via Event ID 100, which clearly showed the successful execution of the file (depicted in Figure 10).



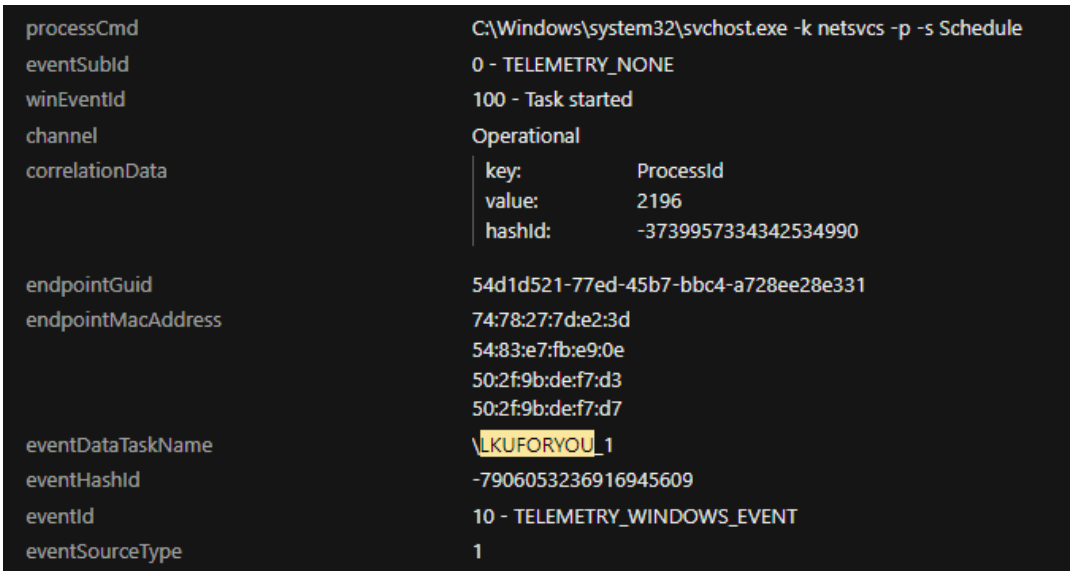


Figure 10. VisionOne Windows event log lelemetry for LKUFORYOU

Figure 11 depicts where run registry keys were installed for persistence, and the data associated with them. These registry keys and values enable the threat to maintain persistence by automatically executing the x32dbg.exe file every time the user logs in.

Registry Key: HKCU\Software\Microsoft\Windows\CurrentVersion\Run
Registry Value Name: x32dbg
Registry Value Data:
C:\ProgramData\UsersDate\Windows_NT\Windows\User\Desktop\x32dbg.exe

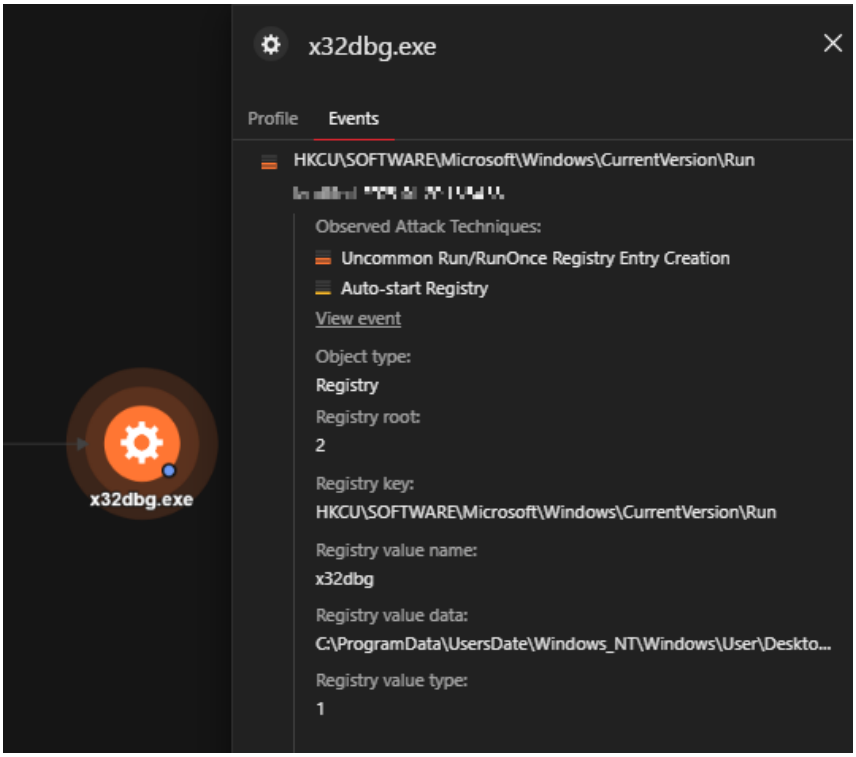


Figure 11. Persistence in the run registry (this image came from ESX testing)

Hiding in plain sight: DLL sideloading with x32dbg.exe

We observed x32dbge.exe being used to sideload the PlugX file x32bridge.dll (0490ceace858ff7949b90ab4acf4867878815d2557089c179c9971b2dd0918b9, detected as Trojan.Win32.KORPLUG.AJ). Sideloading can take advantage of the loader's DLL search order by placing the malicious payload(s) and victim program side by side. This process is likely used by malicious actors as a cover for operations carried out within a trusted, legitimate, and maybe elevated system or software process.

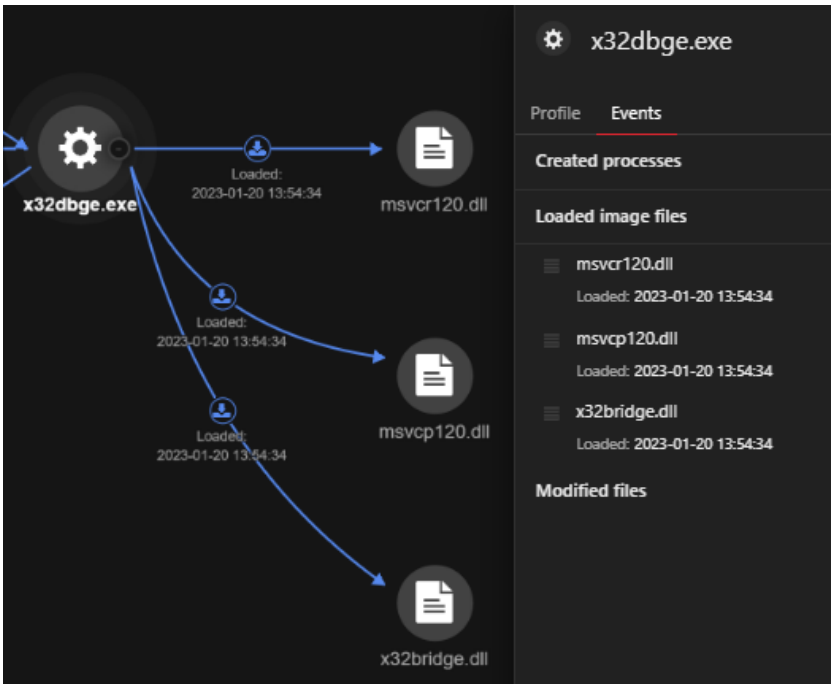


Figure 12. x32dbge.exe sideloaded Plug X file x32bridge.dll (Trojan.Win32.KORPLUG.AJ)

We observed that the file akm.dat (0e9071714a4af0be1f96cffc3b0e58520b827d9e58297cb0e02d97551eca3799, detected as Trojan.Win32.KROPLUG.AJ) was also registered and executed via rundll32, a Windows component which attackers can abuse to facilitate the execution of malicious code. By using rundll32.exe to execute the file, the attackers can prevent security tools from monitoring this activity.

```
rundll32 SHELL32.DLL, ShellExec_RunDLL rundll32
C:\ProgramData\UsersDate\Windows_NT\Windows\User\Desktop\akm.dat,Start
```

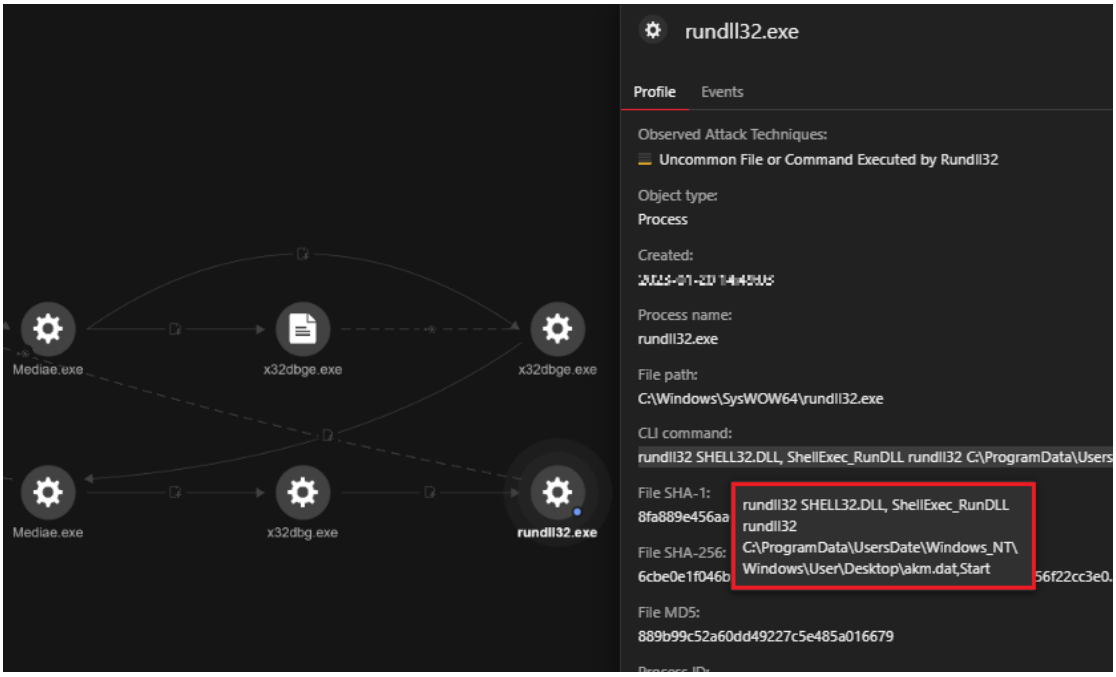


Figure 13. The file akm.dat was executed via rundll32

Unveiling the tactics used: An in-depth analysis of the threat

Through reverse engineering, we were able to gain a deep understanding of how the threat operates. By analyzing the tactics and techniques used by the attacker, we can identify and prevent similar attacks in the future.

Our analysis of this attack in VisionOne revealed that the threat heavily relied on DLL sideloading, which is a typical behavior of PlugX. However, this variant was unique in that it employed several components to perform various functions, including persistence, propagation, and backdoor communication. As a result, we were able to identify and isolate the different files used by the attacker in their routine.

Persistence and propagation: x32dbg.exe (with the components x32bridge.dll and x32bridge.dat)

The file x32dbg.exe is a legitimate executable of a debugging software which, when executed, imports x32bridge.dll and calls on the functions *BridgeStart* and *BridgeInit*. The attackers took advantage of this and replaced the DLL with their own, containing the same export functions but executing entirely different codes:

- BridgeStart – dummy code that does nothing
- BridgeInit – Loads x32bridge.dat, decrypts its contents, then proceeds with the execution of the decrypted code.

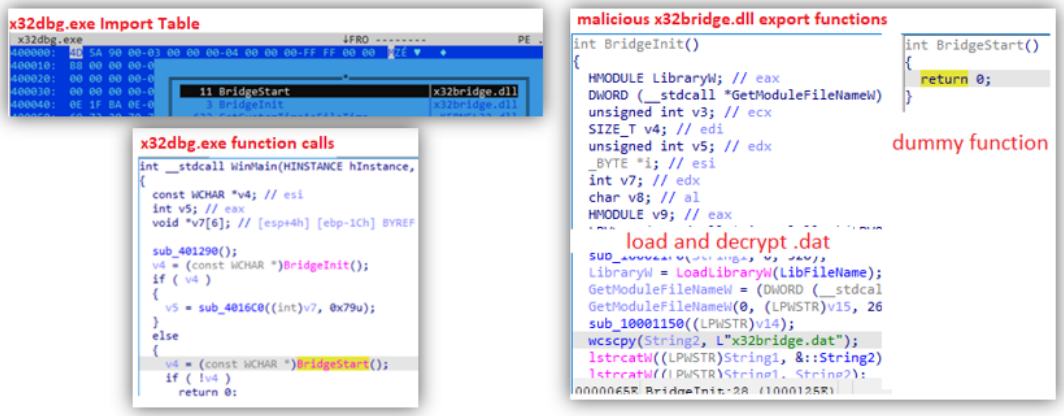


Figure 14. The structure of x32dbg.exe and x32bridge.dll

The hardcoded key “HELLO_USA_PRISIDENT” is used to decode x32bridge.dat, after which execution will continue on the decrypted code.

Figure 15. Decoding x32bridge.dat using the hardcoded key

It will then check for an event named *LKU_Test_0.1* (or creates it if not found). This is followed by the execution of akm.dat found in the same folder.

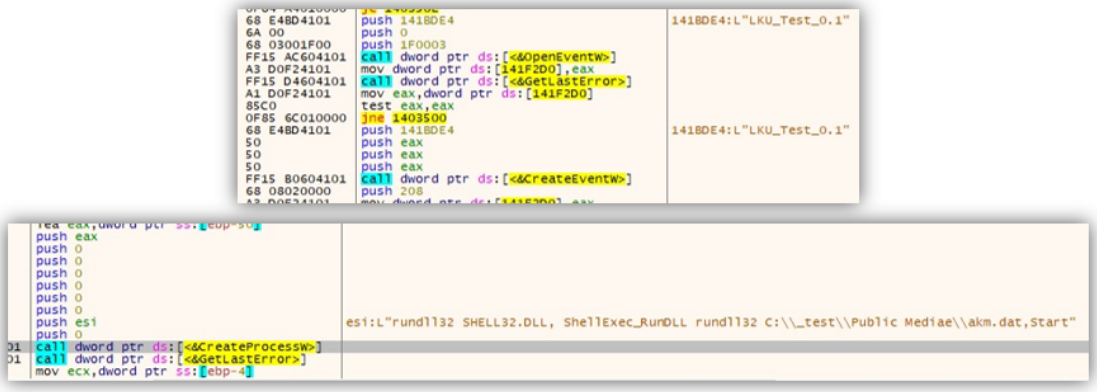


Figure 16. Executing akm.dat

Next, it creates the scheduled task *LKUFORYOU_1* to run x32dbg.exe persistently like what was observed in our VisionOne investigation.

It then enumerates all drives and takes note of removable drives for its propagation routine. When found, it will delete files from any existing RECYCLER.BIN folder before creating a new one. It will copy its components that have the file extensions .exe, .dll, and .dat to the newly created folder and add a desktop.ini file.

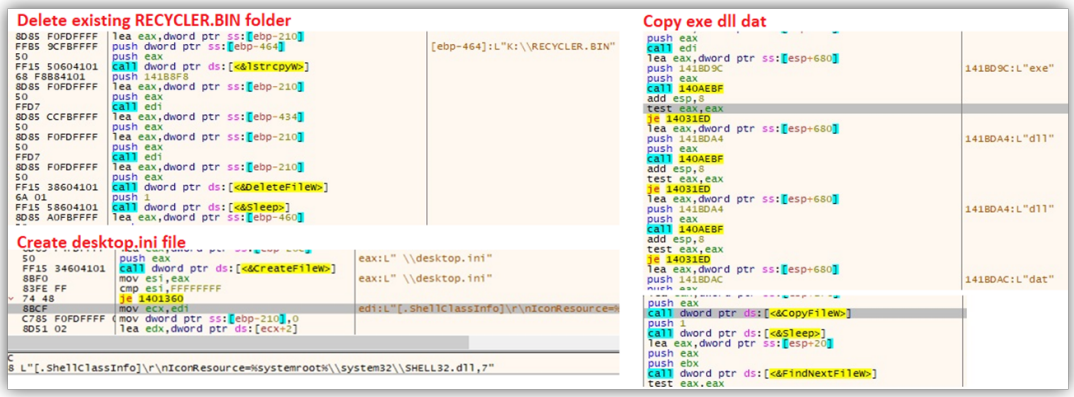


Figure 17. Deleting the existing RECYCLER.BIN folder and creating a new one

Next, it will proceed to its installation routine, where it copies all its components to several folders as listed on the VisionOne analysis.

Figure 18. The installation routine

Once installed, it will run the file Mediae.exe (same file as x32dbg.exe), which will remain in memory, looping through the aforementioned routines.

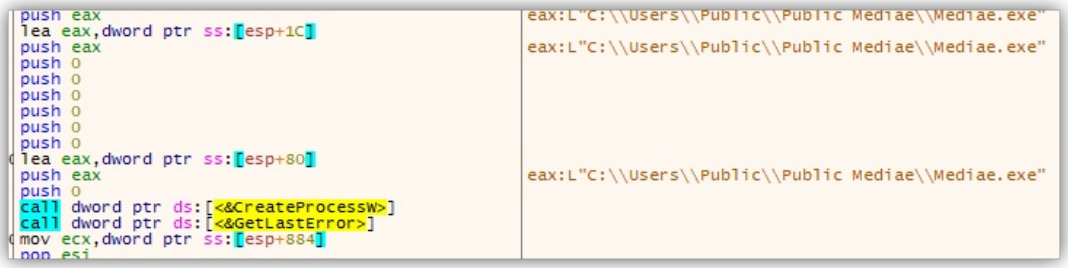


Figure 19. Running Mediae.exe

Mediae.exe also creates the event LKU_Test_0.2, possibly to signal a successful installation.

Figure 20. Creating LKU_Test_0.2

As also seen in the VisionOne analysis, the malware checks if it already has an AutoStart registry key (x32dbg), and creates one if there isn't. Note that the execution path may vary depending on where x32dbg.exe / Mediae.exe was executed.

Next stage loader: akm.dat

The file akm.dat is a DLL with a straightforward function — to execute the next phase of the DLL sideloading routine. Its export function *Start* will execute the file AUG.exe (also included in the previous installation from x32dbg.exe).

Figure 21. The Start function executing AUG.exe

The backdoor UDP Shell: AUG.exe (with the components DismCore.dll and Groza_1.dat)

AUG.exe is a copy of DISM.EXE, a legitimate Microsoft file which is also vulnerable to DLL sideloading. It imports the function DllGetClassObject from DismCore.dll, which will decrypt the contents of Groza_1.dat using the hardcoded key *"Hapenexx is very bad"*.

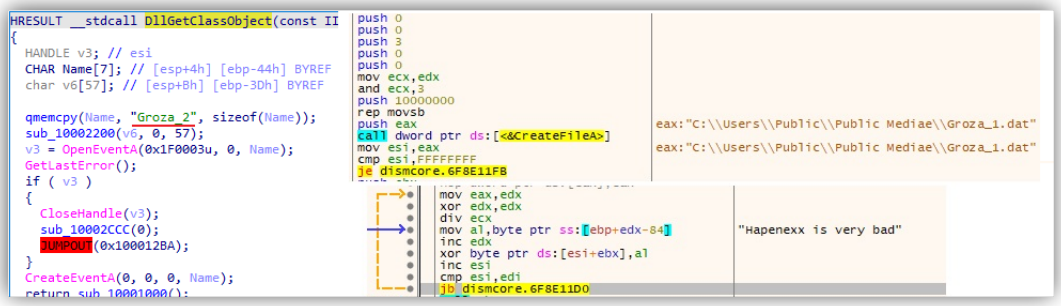


Figure 22. Decrypting Groza_1.dat using the hardcoded key

The execution will continue on the decrypted code, which is a UDP Shell client that does the following:

- Collects host information such as the hostname, IP Address and Mac address and sends it to its command-and-control (C&C) server *160[.]20[.]147[.]254*
- Creates a thread to continuously wait for C&C commands
- Decrypts C&C communication using the hardcoded key *"Happiness is a way station between too much and too little."*

- Hardcoded Debug Info found in file: *C:\Users\guss\Desktop\Recent Work\UDP SHELL\0.7 DLL\UDPDLL\Release\UDPDLL.pdb*

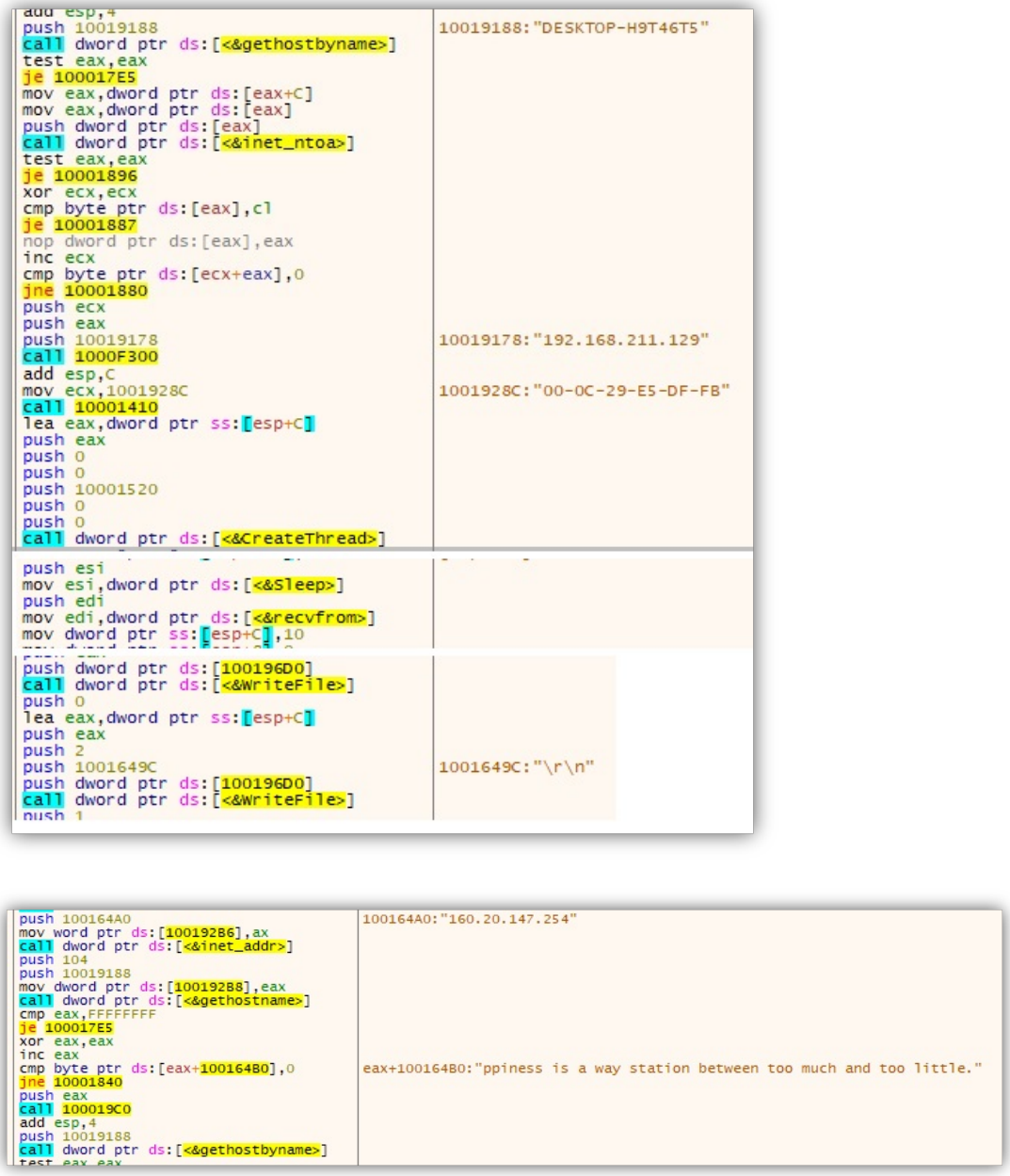


Figure 23. The UDP shell client

Conclusion and Recommendations

The discovery and analysis of the malware attack using the open-source debugger tool x32dbg.exe shows us that DLL side loading is still used by threat actors today because it is an effective way to circumvent security measures and gain control of a target system. Despite advances in security technology, attackers continue to use this technique since it exploits a fundamental trust in legitimate applications. This technique will remain viable for attackers to deliver malware and gain access to sensitive information as long as systems and applications continue to trust and load dynamic libraries.

This incident highlights the importance of having a strong and robust cybersecurity system in place, as threat actors continue to find new ways to exploit vulnerabilities and launch sophisticated attacks. Trend Micro Managed Extended Detection and Response (MxDR) helps in the prevention of DLL sideloading attacks by taking a comprehensive approach to detecting, investigating, and responding to security incidents.

Trend XDR integrates a variety of security technologies, such as endpoint protection, network security, and cloud security, to provide a comprehensive picture of an organization's security posture. This enables MxDR to detect and

prevent DLL sideloading attacks by detecting and blocking malicious activity at various stages of the attack lifecycle before it can cause harm. Furthermore, XDR can perform in-depth analysis and investigation of security incidents, allowing organizations to understand the impact and scope of an attack and respond appropriately.

Here are some recommendations that IT administrators can put into place to prevent DLL side loading attacks:

- **Implement whitelisting:** Allow only known and trusted applications to run on the system while blocking any suspicious or unknown ones.
- **Use signed code:** Ensure that all DLLs are signed with a trusted digital signature to ensure their authenticity and integrity.
- **Monitor and control application execution:** Monitor and control the execution of applications and their dependencies, including DLLs, to detect and prevent malicious activities.
- **Educate end users:** Inform users about the dangers of DLL sideloading attacks and encourage them to exercise caution when installing or running unfamiliar software.
- **Endpoint protection:** Use endpoint protection solutions that offer behavioral analysis and predictive machine learning for better security capabilities
- **Implement effective incident response plans:** Establish a clear and well-defined incident response plan to detect, contain, and respond to security incidents as quickly as possible.

Indicators of Compromise

File name	SHA256	Detection name
x32dbg.exe	ec5cf913773459da0fd30bb282fb0144b85717aa6ce660e81a0bad24a2f23e15	Legitimate Windows debu
x32bridge.dll	0490ceace858ff7949b90ab4acf4867878815d2557089c179c9971b2dd0918b9	Trojan.Win32.KORPLUG.
akm.dat	0e9071714a4af0be1f96cffc3b0e58520b827d9e58297cb0e02d97551eca3799	Trojan.Win32.KORPLUG.
x32bridge.dat	e72e49dc1d95efabc2c12c46df373173f2e20dab715caf58b1be9ca41ec0e172	Trojan.Win32.KORPLUG.
DismCore.dll	b4f1cae6622cd459388294afb418cb0af7a5cb82f367933e57ab8c1fb0a8a8a7	Trojan.Win32.KORPLUG.
Groza_1.dat	553ff37a1eb7e8dc226a83fa143d6aab8a305771bf0cec7b94f4202dcd1f55b2	Trojan.Win32.KORPLUG.

IP address / URL	Description
160[.]20[.]147[.]254	C&C Server