

Updates on OwlProxy malware


























lab52.io/blog/chimera-apt-updates-on-its-owlproxy-malware/

During the analysis of some malicious artifacts collected from an incident, we have recently detected a sample that has caught our attention, the sample was deployed on a server exposed to the Internet and was packed with “VMProtect”. After analyzing this malware sample we could see that it was a recent version of a tool known as OwlProxy, which has been detected on targets hit by the APT group known as Chimera.

The use of “VMProtect” by the actors makes the analysis of their code more time-consuming (although it simplifies its detection since this is already an anomaly by itself). On the other hand, this packer has some weaknesses such as the fact that it does not always protect part of the binary metadata and that once in memory, it does not obfuscate most of the strings, which makes these elements of the executable relatively easy to analyze once in memory.

Especially for this sample it helps for its identification given the amount of quite unique strings it contains. An example of this is the fact that in static you can see how it still contains the string with the original name of the binary (iisdll.dll).

From some of these strings, it can be seen how new samples are being uploaded regularly, which after a more detailed analysis have been identified as different 32-bit and 64-bit versions of the same threat, some of them packaged and others not.

	Detections	Size	First seen	Last seen	Submitters	
<input type="checkbox"/>     16CF6924ECA8208AD909B8470CF8A849995ED4D98105BC706E62AE878EF261BB vnipd.dll pedll overlay	13 / 67	330.79 KB	2021-04-26 15:58:24	2021-04-26 15:58:24	1	
<input type="checkbox"/>     9A9FFD88FC20B6E6E5A10E434F68834E69755FD62157F91884E9A681F0073256 iis-cgi.exe peexe	33 / 70	1.05 MB	2021-04-26 10:30:40	2021-04-26 10:30:40	1	
<input type="checkbox"/>     75E168F87AD0CBE738C4C488C8ECD114871DEF145E21545FA1170503A11D60E vnipd.dll pedll 64bits assembly overlay	42 / 68	260.02 KB	2021-04-19 07:46:16	2021-04-19 07:46:16	1	
<input type="checkbox"/>     DEE3A502AEDCBE109373AA200A690D4E2D1A10FA3CF69A706DB68A78C6ABFC2C Fuscon.bad pedll 64bits assembly overlay	26 / 51	260.02 KB	2021-03-27 05:29:26	2021-03-27 05:29:26	1	
<input type="checkbox"/>     95E7E09468F7DC62DA42E036B6E36CE27DC04C2CA08086A2FED3983CC2BF4A97 Vnipre.dll	31 / 68	3.10 MB	2021-03-16 15:55:42	2021-03-16 15:55:42	1	

Crowdsourced YARA Rules

 Matches rule [win_owlproxy_auto](#) by Felix Bilstein - yara-signator at cocacoding dot com from ruleset win_owlproxy_auto at <https://malpedia.caad.fkie.fraunhofer.de/>
↳ *autogenerated rule brought to you by yara-signator*

Some of them, dating from 2019, largely coincide with the samples analyzed in the following [CyCraft](#)

A summary of its capabilities can be found in this report, which explains that it acts as a proxy between the DMZ of the victim’s infrastructure and its internal network and also allows the execution of commands remotely.

The report shows the metadata of the binary where you can see how the PDB matches with most of the more recent samples that can be found in Virustotal.

File Metadata

md5: cb1f2894cd35b173140690b0a608d4b6
sha1: d744fb9adbd2d79c6016044de4a75e6c4f3fefb0
sha256: 5b3ca2aacfa0996275c7a116bc2b14a03161b264ba4c699a55a5d19b8677969b
family: Owlproxy (family name first designated by TrendMicro)
filetype: PE32+ executable (DLL) (GUI) x86-64, for MS Windows
pdb: F:\project\owl\isapi\x64\Release\iisdll.pdb

Behavior

wmipd.dll uses WinHTTP API to create an HTTP server on port 80 with these endpoints:

```
http[://+:80/servlet/ (General backdoor, execute any commands)  
http[://+:80/servlet/pp/ (HTTP to tcp proxy)
```

IOC

```
file_path: %WINDIR%\System32\Windows Event Manageex.dll  
http[://127.0.0.1]/servlet/  
http[://127.0.0.1]/servlet/pp/  
http[://127.0.0.1]/servlet/pp/$NUMBERS
```

It is also explained in the report that the sample puts its persistence as a Service and highlights the two endpoints it exposes while it is running.

In the case of the samples analyzed, instead of using port 80, they are exposing the port 443 using SSL and they have stopped using the string “servlet” and now they have different strings depending on the sample in question, they have kept the string “pp” for the endpoint that acts as proxy.

Two samples from 02/2019

DEE3A502AEDCBE109373AA200A690D4E2D1A10FA3CF69A7B6DB68A78C6ABFC2C

https://+:443/HelpTheme/ Remote CMD addr

https://+:443/HelpTheme/pp/ Proxy addr

FastUserSwitchingCompatibility Service Name

75E616BF87AD0CBE738C4C48BC0ECD114871DEF145E21545FA11705D3A11D60E

https://+:443/topics/ Remote CMD addr

https://+:443/topics/pp/ Proxy addr

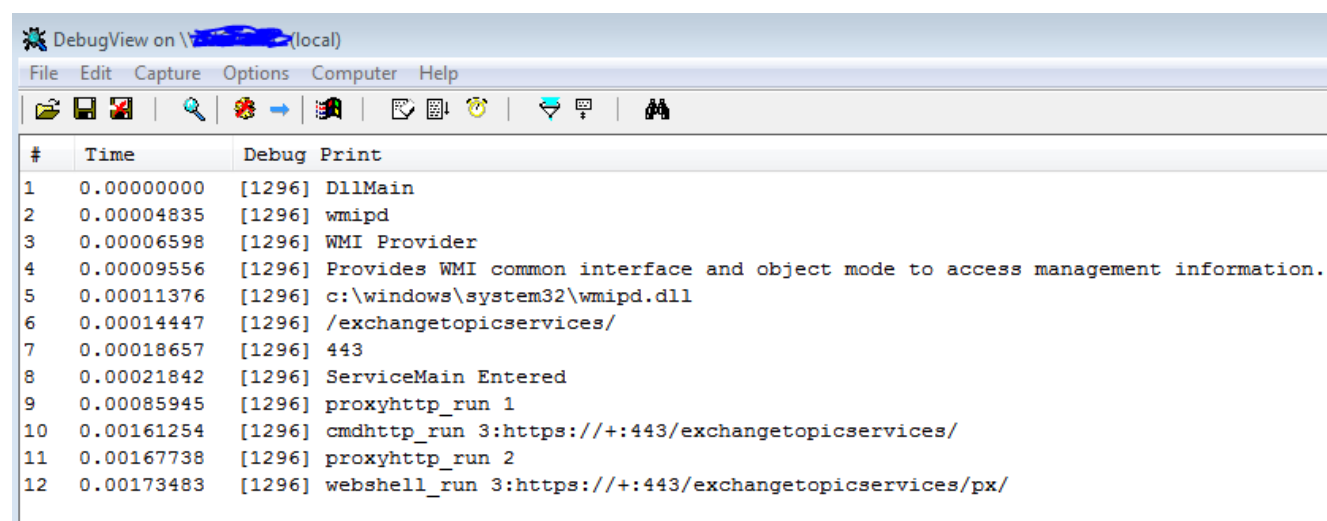
wmipd Service Name

Along with these two samples, there are 3 more recent ones (July 2020). Two of them are newer versions of this same 32-bit and 64-bit tool and the third one is an installer or dropper (9A9FFDB8FC20B6E6E6E5A1DE434F6B834E69755FD62157F91884E9A681F0073256) that when executed, depending on the operating system architecture, extracts from its resources a suitable OwlProxy sample for the architecture in question, installs it in system32 with the name “wmipd.dll”, gives it the same creation and modification date as “calc.exe” and creates a service named wmiipd that runs it at every system startup.

The pdb of this dropper keeps the string “owl” in its path, so we have named it OwlInstaller. “f:\project\owl\isapi\win32\release\iisinstaller.pdb”.

The OwlProxy sample of its resources is split into reverse ordered blocks, so the MZ header, for example, is in the last 2634 bytes of the file.

Once extracted and installed, these more recent samples have many strings and use OutputDebugString in several occasions, which causes that in debugging or using DebugView a few of this strings can be seen as debug information:



Along with the fact that these also use “https” as protocol, and that they have again changed part of the path of their endpoints to use “exchangetopicservices” (probable evolution of the 2019 samples that use the string “topics”), they have added a new endpoint to the tool, as follows:

- https://+:443/exchangetopicservices/ Remote CMD addr

- https://+:443/exchangetopicservices/pp/ Proxy

- https://+:443/exchangetopicservices/px/ Webshell

The path ending in “exchangetopicservices” still works the same way as in previous samples, executing commands in a cmd process through unnamed pipes. The proxy endpoint has also remained the same in terms of capabilities. Finally, thanks to all the strings in the binary, it is easy to spot that they call this endpoint “Webshell_run”, whose URI ends in “/px/”, and consists of a function that adds the following list of commands:

- get_driver
- get_directory

- upload_file
- donwload_file

This area has many more strings than the rest, and it is curious that several of them are misspelled. In fact the first error probably comes in the command “get_driver” since in its internal logic the only thing it does is to list the hard disks installed in the computer:

```

*( _DWORD *)RootPathName = ':\0A';
v24 = 0;
v19 = 1;
v25 = 0;
v26 = 0;
for ( i = GetLogicalDrives(); i; i >>= 1 )
{
    if ( GetDriveTypeW(RootPathName) == 3 )
    {
        sub_10014940(&v15, (int)v1, RootPathName);
        v27 = 1;
        v4 = (void **)sub_10014A40(L"^\\n");
        LOBYTE(v27) = 2;
        if ( v1 != v4 )
        {
            if ( (unsigned int)v1[5] >= 8 )
            {
                j__free(*v1);
                v1[5] = (void *)7;
                v1[4] = 0;
                *( _WORD *)v1 = 0;
                sub_100102B0(v1, v4);
            }
            LOBYTE(v27) = 1;
            if ( v22 >= 8 )
            {
                j__free(v20);
                v22 = 7;
                v21 = 0;
                LOWORD(v20) = 0;
                LOBYTE(v27) = 0;
                if ( v17 >= 8 )
                {
                    j__free(v15);
                    v17 = 7;
                    v16 = 0;
                    LOWORD(v15) = 0;
                }
            }
            ++RootPathName[0];
        }
    }
}

```

Which suggests that probably, this functionality should be called “get_drives” :_)

The next wrong command is “donwload_file”, although in this case it’s easy to guess what it means and it does exactly that, it allows the attacker to download a file from the compromised server.

```

v23 = 13;
v22 = L"donwload_file";
v21 = *(( _DWORD *)v47 + 4);
if ( !sub_10015F60(v47, v12, v21, L"donwload_file", 0xDu) && (v48 - (
{

```

The rest do match their names since “get_directory” can be used to list the contents of a directory and “upload_file” is used to drop a file on the remote server.

The last notable error we have detected is a translation error when a file does not exist in the “get_directory” command.

```

-----
v9 = 0;
v10 = sub_1000E8A0(&v34, L"download_file: ");
LOBYTE(v48) = 4;
v11 = sub_10014270(v9, v10);
v12 = sub_10014270(v11, lpFileName);
v13 = sub_10014F30(v12, L" file isn't exist");
sub_100028E0(v13);
-----

```

The older versions have much fewer strings, so it is likely that in future versions of the threat will remove many of those described in this post, but some of them are used to identify the command executed, so it is possible that they will remain, along with the pdb that they have been leaving since 2019. What is clear is that they are in constant development.

DEE3A502AEDCBE109373AA200A690D4E2D1A10FA3CF69A7B6DB68A78C6ABFC2C	Old OwlProxy x64 version
75E616BF87AD0CBE738C4C48BC0ECD114871DEF145E21545FA11705D3A11D60E	Old OwlProxy x64 version
16CF6924ECA8208AD9D9B8470CF0A849995ED4D981D5BC706E62AE878EF261BB	Recent OwlProxy x86 version
C9B8CFACC859E494984AE9DDF864314AF651FD32E60F33C7F42BFC640620E8CA	Recent OwlProxy x64 version
95E7E09468F7DC62DA42E036B6E36CE27DC04C2CA0BB86A2FED39B3CC2BF4A97	VMProtect OwlProxy x64 version
9A9FFDB8FC20B6E6E5A1DE434F6B834E69755FD62157F91884E9A681F0073256	OwlInstaller