WildPressure targets the macOS platform

By Denis Legezo

Published: 2021-07-07 · Archived: 2026-04-05 14:27:06 UTC

New findings

Our previous story regarding WildPressure was dedicated to their campaign against industrial-related targets in the Middle East. By keeping track of their malware in spring 2021, we were able to find a newer version. It contains the C++ Milum Trojan, a corresponding VBScript variant with the same version (1.6.1) and a set of modules that include an orchestrator and three plugins. This confirms our previous assumption that there are more last-stagers besides the C++ ones, based a field in the C2 communication protocol that contains the “client” programming language.

Another language used by WildPressure is Python. The PyInstaller module for Windows contains a script named “Guard”. Perhaps the most interesting finding here is that this malware was developed for both Windows and macOS operating systems. The coding style, overall design and C2 communication protocol is quite recognizable across all three programming languages used by the authors.

The versioning system shows that the malware used by WildPressure is still under active development. Besides commercial VPS, this time the operators used compromised legitimate WordPress websites. With low confidence this time, we believe their targets to be in the oil and gas industry. If previously the operators used readable “clientids” like “HatLandid3”, the new ones we observed in the Milum samples appear to be randomized like “5CU5EQLOSI” and “C29QoCli33jjxtb”.

Although we couldn’t associate WildPressure’s activity with other threat actors, we did find minor similarities in the TTPs used by BlackShadow, which is also active in the same region. However, we consider that these similarities serve as minor ties and are not enough to make any attribution.

Python multi-OS Trojan

SHA1	872FC1D91E078F0A274CA604785117BEB261B870
File type	PE32 executable (GUI) Intel 80386 (stripped to external PDB), for MS Windows
File size	3.3 MB
File name	svchost.exe

This PyInstaller Windows executable was detected in our telemetry on September 1, 2020, showing version 2.2.1. It contains an archive with all the necessary libraries and a Python Trojan that works both on Windows and macOS. The original name of the script inside this PyInstaller bundle is “Guard”. The malware authors

extensively relied on publicly available third-party code[1] to create it. Near the entry point one can find the first operating system-dependent code, which checks on macOS if another instance of the Trojan is running.

```
@staticmethod
def check_daemon_running():
    terminal = subprocess.Popen(["launchctl" ,"list"] ,  stdout=subprocess.PIPE)
    terminal.wait()
    processList= terminal.communicate()[0].split("\n")
    isScript = CrossPlatformTools.isScript()
    for row in  processList:
        try:
            if(isScript):
                cols=row.split()
                if(len(cols)>1):
                    if(PLIST_SIGN_KEY in row):
                        return True
            else:
                if(PLIST_SIGN_KEY in row):
                    return True
        except Exception as e:
            pass
    return False

@staticmethod
def my_process_is_daemon():
    terminal = subprocess.Popen(["launchctl" ,"list"] ,  stdout=subprocess.PIPE)
    terminal.wait()
    pid = os.getpid()
    processList= terminal.communicate()[0].split("\n")
    isScript = CrossPlatformTools.isScript()
    for row in  processList:
        try:
```

macOS-specific code snippet to check if another Trojan instance is already running

The Guard class constructor contains initial values, such as an XOR key (enc_key field) to decrypt the configuration. In this sample, it is set to decimal 110 and the C2 message type (answer_type_value field) to “Check”. The code that initializes class members for encryption and network communications is OS independent, but persistence methods aren’t.

For macOS, Guard decodes an XML document and creates a plist file using its contents at $HOME/Library/LaunchAgents/com.apple.pyapple.plist to autorun itself; while for Windows, the script creates a RunOnce registry key Software\Microsoft\Windows\CurrentVersion\RunOnce\gd_system. We provide the full list of persistence IoCs at the end of this article.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Label</key>
    <string>apple.scriptzxy.plist</string>
    <key>ProgramArguments</key>
    <array>
        <string>/usr/bin/python</string>
        <string>[pyscript]</string>
    </array>
    <key>KeepAlive</key>
    <true/>
</dict>
</plist>
```

Malware decodes the XML, fills [pyscript] placeholder with its path and drops .plist file for persistence

For fingerprinting Windows and macOS operating systems, Guard uses standard Python libraries. Beacon data for the C2 contains the hostname, machine architecture, OS release name. To fingerprint Windows targets, Guard also uses WQL (WMI Query Language) requests similarly to Milum and WMIC command line utility features. For example, to distinguish the installed security products it executes the following command:

> cmd /c wmic /NAMESPACE:\\\\root\SecurityCenter2 PATH AntiVirusProduct GET displayName,
>
> productUptoDate /Format:List

On macOS, Guard enumerates running processes using the “ls /Applications” command and compares the results against a list of security solutions: [“kaspersky security.app”,”kaspersky anti-virus for mac.app” , “intego”, “sophos anti-virus.app” , “virusbarrier.app”,”mcafee internet security.app”]

The path to the file containing Guard’s configuration data is %APPDATA%\Microsoft\grconf.dat under Windows and $HOME\.appdata\grconf.dat under macOS.

Guard’s configuration data has to start with the string “*grds*”. Below is a comparison between different WildPressure sample parameters, including magic values used to pre- and post-fix the configuration data.

Parameter	**C++ Milum**	**Python Guard**	**VBScript Tandis**
Version	1.0.1 – 1.6.1	2.2.1	1.6.1
Serial	Comparable to “clientid” with values like “HatLandid3”	1——-C29QoCli ————————	1——-Tandis_7 ————————
Relays	List of .php pages hosted on VPS	List of hacked WordPress websites	List of hacked WordPress websites
Encoded configuration	(ws32) (we32)	*grds* *grde*	Configuration embedded inside the script

start\end			

These prefix and suffix values allowed us to decode Mulim and Guard configuration data as well as the self-decrypted Tandis with Bash and Python scripts. Following configuration parsing, the Trojan is ready for its main working cycle. It awaits commands from its C2 that are XML-based and XOR-encrypted with the aforementioned decimal value 110. Among them are typical Trojan functions: downloading files, uploading files, executing commands with the OS command interpreter, updating the Trojan and cleaning up the target.

VBScript self-decrypted variant

SHA1	CD7904E6D59142F209BD248D21242E3740999A0D
File type	Self-decrypting VBScript
File size	51 KB
File name	l2dIIYKCQw.vbs

We named the Tandis Trojan after its “serial” configuration parameter. This VBScript Trojan version is Windows-only and relies much more on WQL queries than Guard. It was first detected in our telemetry on September 1, 2020, showing version 1.6.1. The abilities, parameters and working cycle are quite similar to Guard and other WildPressure malware.

The persistence is again system registry-based (please check the IoCs at the end). The function HexToBin() is in charge of the additional encryption used inside the script for some strings and C2 communication. The basic unhexlify-XOR algorithm is the same as in the initial self-decryption; and to read plain text we used the same aforementioned script with corresponding key (again 110 decimal, stored in a class data member). The C2 communication protocol is “encrypted XML over HTTP” (using Msxml2.XMLHTTP and Msxml2.DOMDocument objects).

Below are the commands that Tandis supports:

Command	Description
1	Wait
2	Silently execute command with interpreter with cmd /c
3	Download file
4	Update the script from server
5	Clean up, remove persistence and the script file
6	Upload file
7	Update wait timings in the configuration

8	Fingerprint the host. In particular, Tandis gathers all the installed security products besides Defender with a WQL query

Plugin-based C++ malware

In addition to the already enumerated scripting implants that WildPressure uses, some findings are related to C++ developments. We discovered several, previously unknown, interconnected modules used to gather data on target hosts in our telemetry. The compilation times seen in this malware precedes our detection date by a large margin, and we therefore consider them to be tampered with.

The plugins we found are rather simplistic. We will therefore focus on the implemented interface between the orchestrator and its plugins.

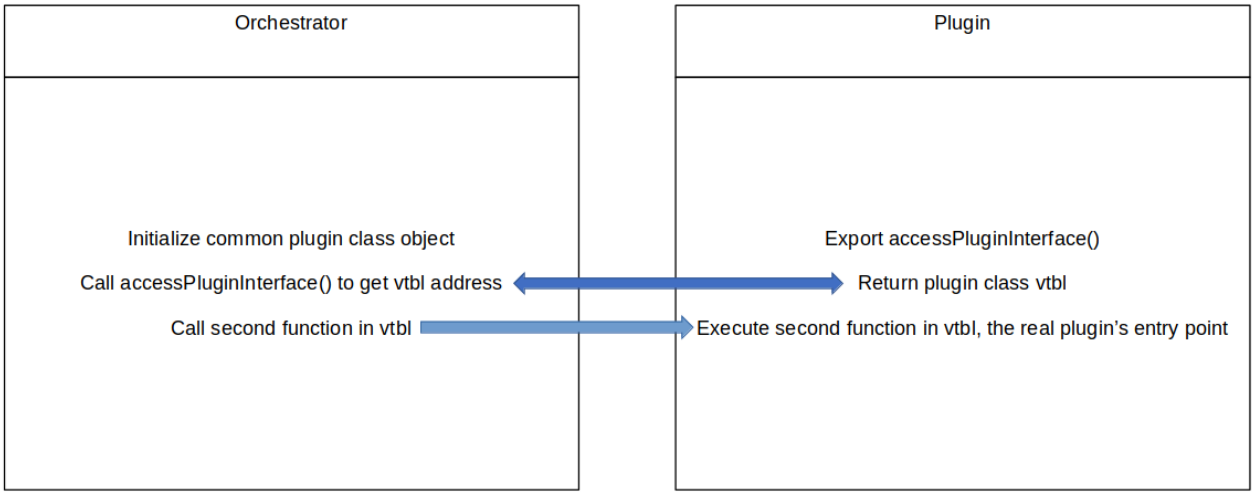
Orchestrator

SHA1	FA50AC04D601BB7961CAE4ED23BE370C985723D6
File type	PE32 executable (console) Intel 80386, for MS Windows
File size	87 KB
File name	winloud.exe

This main module checks for the presence of a configuration file named “thumbnail.dat”. The precise directory of this configuration file varies across Windows versions:

- %ALLUSERSPROFILE%\system\thumbnail.dat
- %ALLUSERSPROFILE%\Application Data\system\Windows\thumbnail.dat

The orchestrator uses a timer function that runs every two minutes and parses the configuration file for the plugin file path, function name, etc., and attempts to execute the corresponding plugin.



The overall communication workflow between orchestrator and the plugins

Plugins come in the form of a DLL that exports a function named accessPluginInterface(), which returns a pointer to a class object to the orchestrator. This main module then runs the second function from the virtual functions table, passing it the pointer to instantiated class objects. The plugins we've seen so far contained RTTI information.

Fingerprinting plugin

SHA1	c34545d89a0882bb16ea6837d7380f2c72be7209
File type	PE32 executable (DLL) (GUI) Intel 80386, for MS Windows
File size	194 KB
File name	GetClientInfo.dll

This plugin gathers really detailed data about the host with WQL queries and creates a JSON with a publicly available library. The data includes OS version and the set of installed hotfixes, BIOS and HDD manufacturers, installed and running software and security products separately, user accounts and network adapters settings, etc. The corresponding executed WQL queries look like this:

```
SELECT Domain, DomainRole, TotalPhysicalMemory, UserName, SystemType FROM
Win32_ComputerSystem
SELECT DHCPServer, DNSDomain, MACAddress, DHCPEnabled, DefaultIPGateway, IPAddress,
IPSubnet FROM Win32_NetworkAdapterConfiguration WHERE IPEnabled ='TRUE'"
```

Keylogging and screenshotting plugins

SHA1	fb7f69834ca10fe31675bbedf9f858ec45c38239
File type	PE32 executable (DLL) (GUI) Intel 80386, for MS Windows
File size	90.5 KB
File name	Keylogger.dll
SHA1	2bb6d37dbba52d79b896352c37763d540038eb25
File type	PE32 executable (DLL) (GUI) Intel 80386, for MS Windows
File size	78 KB
File name	ScreenShot.dll

These plugins are quite straightforward. The keylogger sets a WH_KEYBOARD_LL hook to gather the keystrokes and gets clipboard content and Windows titles. The second takes screenshots by timer and by mouse events, setting a WH_MOUSE_LL hook.

Campaign infrastructure

The actor used both VPS and compromised servers in their infrastructure, most of which were WordPress websites. The legitimate, compromised websites served as Guard relay servers. In our previous 2019 investigation, we were able to sinkhole the Milum C2, upiserversys1212[.]com. During our current investigation we managed to sinkhole another Milum C2, mwieurgbd114kjuvtg[.]com. However, we haven't registered any recent Milum requests sent to these domains with the corresponding main.php or url.php URI.

Domain	IP	First seen	ASN	Malware
N/A	107.158.154[.]66	2021-04-07	62904, EONIX	Milum
	185.177.59[.]234	2021-04-07	44901, BELCLOUD	
	37.59.87[.]172	2014-12-26	16276, OVH	
	80.255.3[.]86	2019-08-28	201011, NETZBETRIEB	
mwieurgbd114kjuvtg[.]com	139.59.250[.]183 (Sinkholed)	2021-04-07 (Sinkholed)	14061, DIGITALOCEAN	

Legitimate, compromised Guard relay servers:

- hxxp://adelice-formation[.]eu
- hxxp://ricktallis[.]com/news
- hxxp://whatismyserver123456[.]com
- hxxp://www.glisru[.]eu
- hxxp://www.mozh[.]org

Who was hit and by whom

We have very limited visibility for the samples described in this report. Based on our telemetry, we suspect that the targets in the same Middle East region were related to the oil and gas industry.

We consider with high confidence that the aforementioned Tandis VBScript, PyInstaller and C++ samples belong to the same authors that we dubbed WildPressure due to the very similar coding style and victim profile. However, another question remains: is WildPressure connected to other threat actors operating in the same region?

Among other actors that we've covered in the region Chafer and Ferocious Kitten are worth mentioning. Technically, there's not much in common with their malware, but we observed some minor similarities with another actor in the region we haven't described publicly so far. Minor similarities with WildPressure are:

- The “pk” parameter in HTTP requests to distinguish the Trojan beacons from, for example, scanners;
- The usage of hacked WordPress websites as relays.

Both tactics aren’t unique enough to come to any attribution conclusion – it’s possible both groups are simply using the same generic techniques and programming approaches.

Learn threat hunting and malware analysis with Denis Legezo and other GReAT experts.

Indicators of Compromise

Milum version 1.6.1
0efd03fb65c3f92d9af87e4caf667f8e

PyInstaller with Guard
92A11F0DCB973D1A58D45C995993D854 (svchost.exe)

Self-decrypting Tandis VBScript
861655D8DCA82391530F9D406C31EEE1 (l2dIIYKCQw.vbs)

Orchestrator
C116B3F75E12AD3555E762C7208F17B8 (winloud.exe)

Plugins
F2F6604EB9100F58E21C449AC4CC4249 (ScreenShot.dll)
D322FAA64F750380DE45F518CA77CA43 (Keylogger.dll)
9F8D77ECE0FF897FDFD8B00042F51A41 (GetClientInfo.dll)

File paths

macOS .plist files
$HOME/Library/LaunchAgents/com.apple.pyapple.plist

$HOME/Library/LaunchAgents/apple.scriptzxy.plist

Config files under Windows

%APPDATA%\Microsoft\grconf.dat

%APPDATA%\Microsoft\vsdb.dat

%ALLUSERSPROFILE%\system\thumbnail.dat

%ALLUSERSPROFILE%\Application Data\system\Windows\thumbnail.dat

Config files under macOS

$HOME/.appdata/grconf.dat

Registry values

Software\Microsoft\Windows\CurrentVersion\RunOnce\gd_system

WQL queries examples

SELECT * FROM Win32_Process WHERE Name = '<all enumerated names here>'

Select * from Win32_ComputerSystem

Select * From AntiVirusProduct

Select * From Win32_Process Where ParentProcessId = '<all enumerated ids here>'

Milum C2

hxxp://107.158.154[.]66/core/main.php

hxxp://185.177.59[.]234/core/main.php

hxxp://37.59.87[.]172/page/view.php

hxxp://80.255.3[.]86/page/view.php

hxxp://www.mwieurgbd114kjuvtg[.]com/core/main.php

[1] E.g. https://gist.github.com/vaab/2ad7051fc193167f15f85ef573e54eb9 and https://code.activestate.com/recipes/65222-run-a-task-every-few-seconds/
