Mind the (air) gap: GoldenJackal gooses government guardrails

ESET Research

ESET Research analyzed two separate toolsets for breaching air-gapped systems, used by a cyberespionage threat actor known as GoldenJackal

Matías Porolli

07 Oct 2024 • , 40 min. read

ESET researchers discovered a series of attacks on a governmental organization in Europe using tools capable of targeting air-gapped systems. The campaign, which we attribute to GoldenJackal, a cyberespionage APT group that targets government and diplomatic entities, took place from May 2022 to March 2024. By analyzing the toolset deployed by the group, we were able to identify an attack GoldenJackal carried out earlier, in 2019, against a South Asian embassy in Belarus that, yet again, targeted the embassy's air-gapped systems with custom tools.

This blogpost introduces previously undocumented tools that we attribute to GoldenJackal based on victimology, code, and functional similarities between the toolsets.

Key points of the blogpost:

- GoldenJackal used a custom toolset to target air-gapped systems at a South Asian embassy in Belarus since at least August 2019. In this blogpost, we describe these tools publicly for the first time.
- This blogpost also features the first public description of a highly modular toolset GoldenJackal deployed on various occasions between May 2022 and March 2024 against a national government organization of a country in the European Union.
- These toolsets provide GoldenJackal a wide set of capabilities for compromising and persisting in targeted networks. Victimized systems are abused to collect interesting information, process the information, exfiltrate files, and distribute files, configurations and commands to other systems.
- The ultimate goal of GoldenJackal seems to be stealing confidential information, especially from high-profile machines that might not be connected to the internet.

GoldenJackal profile

GoldenJackal is an APT group active since at least 2019. It targets government and diplomatic entities in Europe, the Middle East, and South Asia. The group is little known and has only been publicly described in 2023 by Kaspersky. The group's known toolset includes several implants written in C#: JackalControl, JackalSteal, JackalWorm, JackalPerInfo, and JackalScreenWatcher – all of them used for espionage.

Overview

In May 2022, we discovered a toolset that we could not attribute to any APT group. But once the attackers used a tool similar to one of those publicly documented by Kaspersky, we were able to dig deeper and to find a connection between the publicly documented toolset of GoldenJackal and this new one.

Extrapolating from that, we managed to identify an earlier attack where the publicly documented toolset was deployed, as well as an older toolset that also has capabilities to target air-gapped systems. This blogpost shines a light on the technical aspects of the publicly undocumented toolsets, and shares some insights about GoldenJackal's tactics, techniques, and procedures.

Victimology

GoldenJackal has been targeting governmental entities in Europe, the Middle East, and South Asia. We detected GoldenJackal tools at a South Asian embassy in Belarus in August and September 2019, and again in July 2021.

Kaspersky reported a limited number of attacks against government and diplomatic entities in the Middle East and South Asia, starting in 2020.

More recently, according to ESET telemetry, a national government organization of a country in the European Union was repeatedly targeted from May 2022 until March 2024.

Attribution

All the campaigns that we describe in this blogpost deployed, at some point, at least one of the tools attributed to the GoldenJackal APT group by Kaspersky. As was the case in the Kaspersky report, we can't attribute GoldenJackal's activities to any specific nation-state. There is, however, one clue that might point towards the origin of the attacks: in the GoldenHowl malware, the C&C protocol is referred to as transport_http, which is an expression typically used by Turla (see our ComRat v4 report) and MoustachedBouncer. This may indicate that the developers of GoldenHowl are Russian speakers.

Breaching air-gapped systems

In order to minimize the risk of compromise, highly sensitive networks are often air gapped, i.e., isolated from other networks. Usually, organizations will air gap their most valuable systems, such as voting systems and industrial control systems running power grids. These are often precisely the networks that are of most interest to attackers.

As we stated in a previous white paper titled Jumping the air gap: 15 years of nation-state effort, compromising an air-gapped network is much more resource-intensive than breaching an internet-connected system, which means that frameworks designed to attack air-gapped networks have so far been exclusively developed by APT groups. The purpose of such attacks is always espionage, perhaps with a side of sabotage.

With the level of sophistication required, it is quite unusual that in five years, GoldenJackal managed to build and deploy not one, but two separate toolsets designed to compromise air-gapped systems. This speaks to the resourcefulness of the group. The attacks against a South Asian embassy in Belarus made use of custom tools that we have only seen in that specific instance. The campaign used three main components: GoldenDealer to deliver executables to the air-gapped system via USB monitoring; GoldenHowl, a modular backdoor with various functionalities; and GoldenRobo, a file collector and exfiltrator.

In the latest series of attacks against a government organization in Europe, GoldenJackal moved on from the original toolset to a new, highly modular one. This modular approach applied not only to the design of the malicious tools (as was the case with GoldenHowl), but also to their roles: they were used, among other things, to collect and process interesting information, to distribute files, configurations, and commands to other systems, and to exfiltrate files.

Technical analysis

Initial access

So far, we haven't been able to trace back to the initial compromise vector in the campaigns seen in our telemetry. Note that Kaspersky reported in a blogpost that GoldenJackal used trojanized software and malicious documents for this purpose.

The mysterious toolset from 2019

The earliest attack that we have attributed to GoldenJackal, which targeted a South Asian embassy in Belarus, occurred in August 2019. The toolset used in this attack is, to the best of our knowledge, publicly undocumented. We've only observed the following custom tools once, and never again:

- A malicious component that can deliver executables to air-gapped systems via USB drives. We've named this component GoldenDealer.
- A backdoor, which we've named GoldenHowl, with various modules for malicious capabilities.
- A malicious file collector and exfiltrator, which we've named GoldenRobo.

An overview of the attack is shown in Figure 1. The initial attack vector is unknown, so we assume that GoldenDealer and an unknown worm component are already present on a compromised PC that has access to the internet. Whenever a USB drive is inserted, the unknown component copies itself and the GoldenDealer component to the drive. While we didn't observe this unknown component, we have seen components with similar purposes – such as JackalWorm – in other toolsets used in later attacks performed by the group.

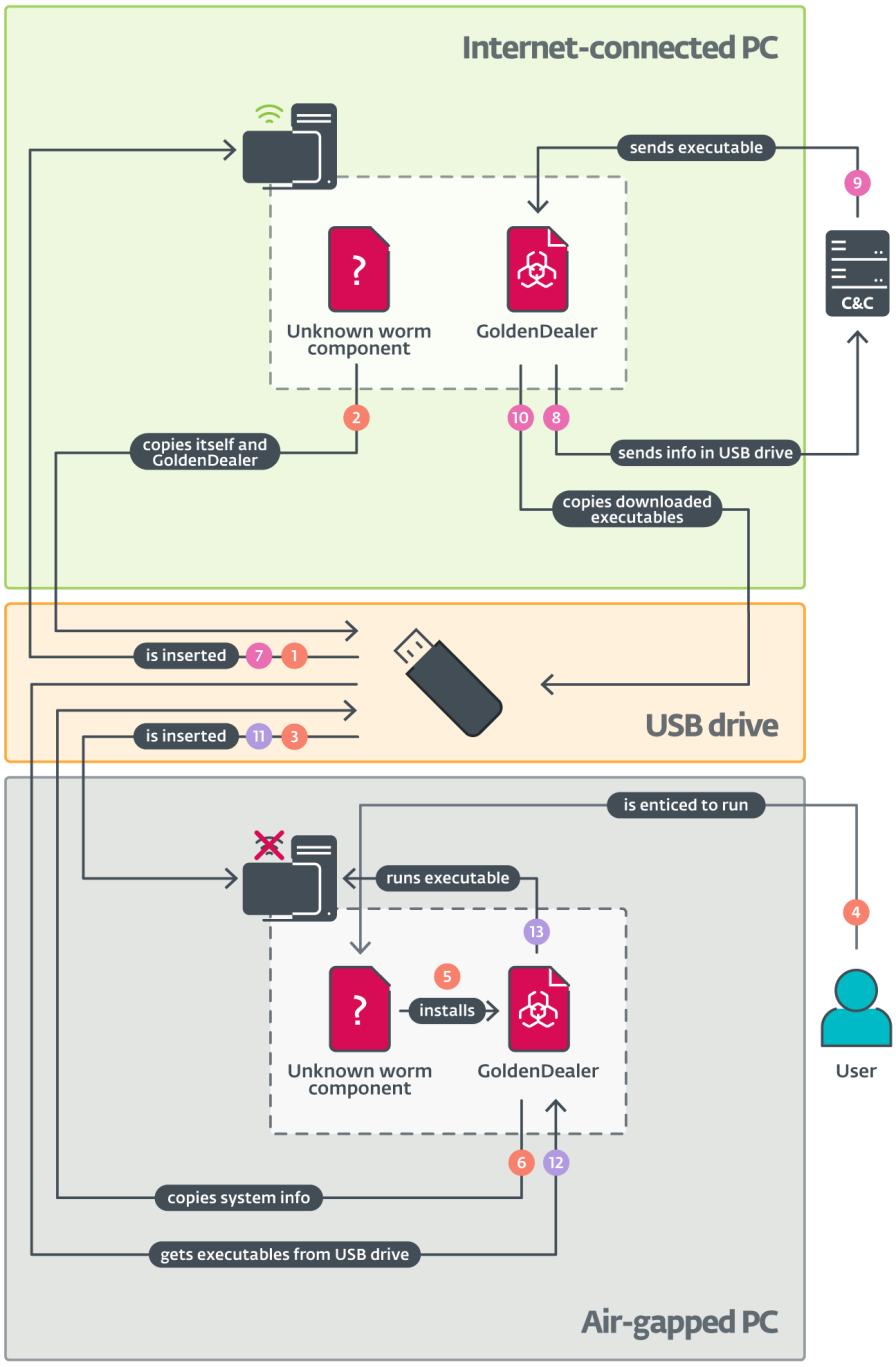

Figure 1. Overview of the initial compromise of an air-gapped system

It is probable that this unknown component finds the last modified directory on the USB drive, hides it, and renames itself with the name of this directory, which is done by JackalWorm. We also believe that the component uses a folder icon, to entice the user to run it when the USB drive is inserted in an air-gapped system, which again is done by JackalWorm.

When the drive is again inserted into the internet-connected PC, GoldenDealer takes the information about the air-gapped PC from the USB drive and sends it to the C&C server. The server replies with one or more executables to be run on the air-gapped PC. Finally, when the drive is again inserted into the air-gapped PC, GoldenDealer takes the executables from the drive and runs them. Note that this time no user interaction is needed, because GoldenDealer is already running.

We have observed GoldenDealer running GoldenHowl on an internet-connected PC. While we didn't observe GoldenDealer directly executing GoldenRobo, we observed the latter also running on the connected PC, used to take

files from the USB drive and exfiltrate them to its C&C server. There must be yet another unknown component that copies files from the air-gapped PC to the USB drive, but we haven't observed it yet.

GoldenDealer

This component monitors the insertion of removable drives on both air-gapped and connected PCs, as well as internet connectivity. Based on the latter, it can download executable files from a C&C server and hide them on removable drives, or retrieve them from these drives and execute them on systems that have no connectivity.

The program can be run with or without arguments. When run with arguments, it takes a path to a file that it moves to a new location and then runs via the CreateProcessW API without creating a window.

To prevent hidden files being shown in Windows Explorer, GoldenDealer creates the ShowSuperHidden value in the HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Advanced registry key, and sets it to zero.

In case GoldenDealer is not running as a service, it creates and starts a service called NetDnsActivatorSharing, then exits. If for any reason the service couldn't be created, persistence is achieved by creating an entry in a Run registry key.

Table 1 shows the list of configuration files used by GoldenDealer. These are located in the directory from which the malware is running: C:\Windows\TAPI in the observed attack. More details about these files is provided in subsequent sections.

Table 1. Configuration files used by GoldenDealer

Filename	Purpose
b8b9-de4d-3b06-9d44	Store status fields.
fb43-138c-2eb0-c651	Store executable files sent by the C&C server.
130d-1154-30ce-be1e	Store information about all compromised PCs in the network.
38c4-abb9-74f5-c4e5	Used as a mutex. If this file is open, it means that an instance of GoldenDealer is already running.

The contents of configuration files are JSON formatted, and stored XOR encrypted on disk. XOR encryption is performed one byte at a time, with a single-byte key that is incremented based on a multiplier.

Network connectivity thread

In order to determine whether a PC is connected to the internet, GoldenDealer sends a GET request to https://1.1.1.1/<user_id> every 15 minutes. If the connection fails, or there's no reply, the PC is assumed to be offline. 1.1.1.1 maps to Cloudflare's DNS resolver, and the expected behavior is to receive a Not Found document and a 404 status code. The <user_id> part is not relevant here, but is used for C&C communication. GoldenDealer generates this user identifier based on:

- The current username as found via the GetUserNameW API.
- The serial number of the first available logical drive in the system. This does not necessarily mean the drive where the OS is installed.

These two strings are separately hashed with the FNV-1a function, and the resulting numbers are XORed together, obtaining a number that identifies the user.

To keep track of network connectivity status, GoldenDealer uses a global variable that can hold any of the following values:

- 0 – Malware started running and connectivity has not been checked.
- 1 – PC doesn't have internet connectivity.
- 2 – PC has internet connectivity.

If the status is 2, a thread is signaled to download executable files from the C&C server, and another thread is signaled to copy the executables to USB drives. A thread to get executables from drives and run them will only be signaled when the status is 1. Whenever the status changes, the configuration file b8b9-de4d-3b06-9d44 is updated with the new value. Fields in this file are:

- wmk – network connectivity status.
- qotwnk – number of seconds without internet. This value is incremented every 15 minutes and reset to zero when there's connectivity. It can be used if the malware is configured to wait a minimum number of seconds before deciding that the PC has no connectivity, but there was no wait in the samples that we observed.
- ltwnk – unknown. This field is not used by the malware.
- rpk – list with hashes of executables downloaded from the C&C server.

Downloader thread

This thread checks the network connectivity status every 30 minutes, and only performs the following actions if the PC is connected to the internet. First, a GET request is sent to https://83.24.9[.]124/<user_id>, just to let the C&C server know that another request is to follow. The reply from the server is not processed. If the request fails, then another request is sent to a secondary server, http://196.29.32[.]210/<user_id>, probably to notify about failure, as the thread doesn't continue to execute in this case. The URLs are hardcoded in the malware and are not configurable in the samples that we observed.

When communication is successful, GoldenDealer sends a request to https://83.24.9[.]124/<user_id>/fc93-10f4-2a68-d548. The server replies with an array of JSON objects with the following fields:

- ek – a base64-encoded string that is an executable file after being decoded,
- tpik – an array of user_ids used to decide whether the executable will be run,
- hek – the FNV-1a hash of ek, and
- apk – date and time when the executable was obtained from the C&C server.

The contents of the last two fields are not relevant, because they are calculated by the downloader thread, replacing original data sent by the C&C server. In both cases, they are stored as decimal numbers.

GoldenDealer will run an executable sent by the server if the corresponding user_id is in the tpik list, and the hek hash is not in the list of hashes stored in the rpk field in the configuration. In other words, connected PCs can download executables and pass them along to other systems via USB drives, but they can also run received executables. When an executable is run, its hash is added to the rpk list, ensuring that it will only be executed once by that victim. Each executable is written in the working directory with the value of <hek> as its filename. All JSON objects with received executables are stored on disk, in the file fb43-138c-2eb0-c651.

As the final step, the downloader thread collects information about the compromised system and sends it to https://83.24.9[.]124/<user_id>/a1e7-4228-df20-1600. The configuration file 130d-1154-30ce-be1e is updated to store this information as well. Figure 2 shows part of the JSON object with the information sent to the C&C server. While all strings are sent as arrays of decimal character codes, for readability we show them as strings in the image. For example, instead of lsass.exe, the value [108, 115, 97, 115, 115, 46, 101, 120, 101] is actually sent.


```
[{
   "iepk": true,
   "pclk": ["ApplicationFrameHost.exe", "CompatTelRunner.exe", "HxTsr.exe",
   "MicrosoftEdgeUpdate.exe", "MoUsoCoreWorker.exe", "MsMpEng.exe", "MusNoti
   "PhoneExperienceHost.exe", "ProcessHacker.exe", "Registry", "RuntimeBroke
   "SearchProtocolHost.exe", "SecHealthUI.exe", "SecurityHealthHost.exe", "Se
   "SgrmBroker.exe", "ShellExperienceHost.exe", "StartMenuExperienceHost.exe
   "TextInputHost.exe", "TiWorker.exe", "TrustedInstaller.exe", "UserOOBEBro
   "backgroundTaskHost.exe", "conhost.exe", "csrss.exe", "ctfmon.exe", "dasH
   "fontdrvhost.exe", "lsass.exe", "msdtc.exe", "msiexec.exe", "notepad++.exe
   "spoolsv.exe", "sppsvc.exe", "svchost.exe", "taskhostw.exe", "vm3dservice
   , "x32dbg.exe"],
   "pglk": ["7-Zip", "Common Files", "HxD", "Internet Explorer", "Microsoft"
   "ModifiableWindowsApps", "Mozilla Firefox", "Mozilla Maintenance Service"
   "RUXIM", "Resource Hacker", "Uninstall Information", "VMware", "WinRAR",
   "Windows Mail", "Windows Media Player", "Windows Multimedia Platform", "W
   "Windows Security", "Windows Sidebar", "WindowsApps", "WindowsPowerShell"
   "pik": 1978909850,
   "sik": {
       "bk": 64,
       "bnk": 9200,
       "idck": false,
       "isk": true,
       "iwk": true,
       "spmjk": 0,
       "spmnk": 0,
       "vlsmjk": 19041,
       "vlsmnk": 2913,
       "vmjk": 6,
       "vmnk": 2,
       "vmsmjk": 10,
       "vmsmnk": 0
   },
   "uck": "SYSTEM",
   "ulk": [{
           "iak": true,
           "nk": "Administrator"
       }, {
```


Figure 2. Part of the information sent by the downloader thread to the C&C server

The members are:

- iepk – a Boolean value that indicates whether the GoldenDealer process is elevated,
- pclk – an array of running processes,
- pglk – an array of installed programs in both Program Files and Program Files (x86) directories,
- pik – the user_id,

- sik – information about the operating system (including version, build, service pack number; architecture; and Boolean values indicating whether the OS is running on a server, domain controller, or workstation),
- uck – the user who is running the GoldenDealer process, and
- ulk – an array of all users, each with a Boolean value indicating whether the user has administrator privileges.

USB monitoring thread on connected PCs

GoldenDealer monitors the insertion of removable drives by creating a window titled defaultWindow with a custom event handler that processes inserted devices and ignores all other system events.

Once a USB drive has been inserted, this thread patches one byte in the master boot record (MBR) of that drive: it searches for the offset where the BOOTMGR string resides, and replaces the first O with 0. Then it checks whether the second O has also been patched to 0, and exits if it hasn't. The USB thread that runs on air-gapped PCs does the opposite: it patches the second O, and checks whether the first has already been patched. This means that the USB drive needs to have been inserted in both a connected PC and an air-gapped PC for the drive to be processed.

If the check is successful, a hidden directory is created on the USB drive, with two files written inside:

- 37b3-ebe5-568e-0676 – this file has the same contents as fb43-138c-2eb0-c651 (all the executables sent by the C&C server). It's used to pass the executables to air-gapped systems, for execution.
- bc41-ac6f-e55e-61a8 – a file with information from air-gapped PCs. It's created empty by this thread, then populated by the USB thread running on air-gapped PCs. The contents of this file are appended to the local file 130d-1154-30ce-be1e (see Table 1), to be sent to the C&C server by the downloader thread.

USB monitoring thread on air-gapped PCs

This is complementary to the thread described in the previous section: it takes the file 37b3-ebe5-568e-0676 with executables on the USB drive and copies its contents to the local file fb43-138c-2eb0-c651. It also takes the local file 130d-1154-30ce-be1e with information about the air-gapped system and adds its contents to the bc41-ac6f-e55e-61a8 file on the USB drive. The code to obtain system information and to run executables is contained in this thread.

GoldenHowl

Another tool from GoldenJackal's 2019 toolset is GoldenHowl, a backdoor written in Python that consists of various modules for malicious functionalities. It is distributed as a self-extracting archive that contains legitimate Python binaries and libraries, as well as malicious scripts. Figure 3 shows the contents of one of these archives. The attackers renamed the Python executable – in version 2.7.15 – as WinAeroModule.exe. This component is intended to be run on PCs with internet connectivity, given its functionalities.

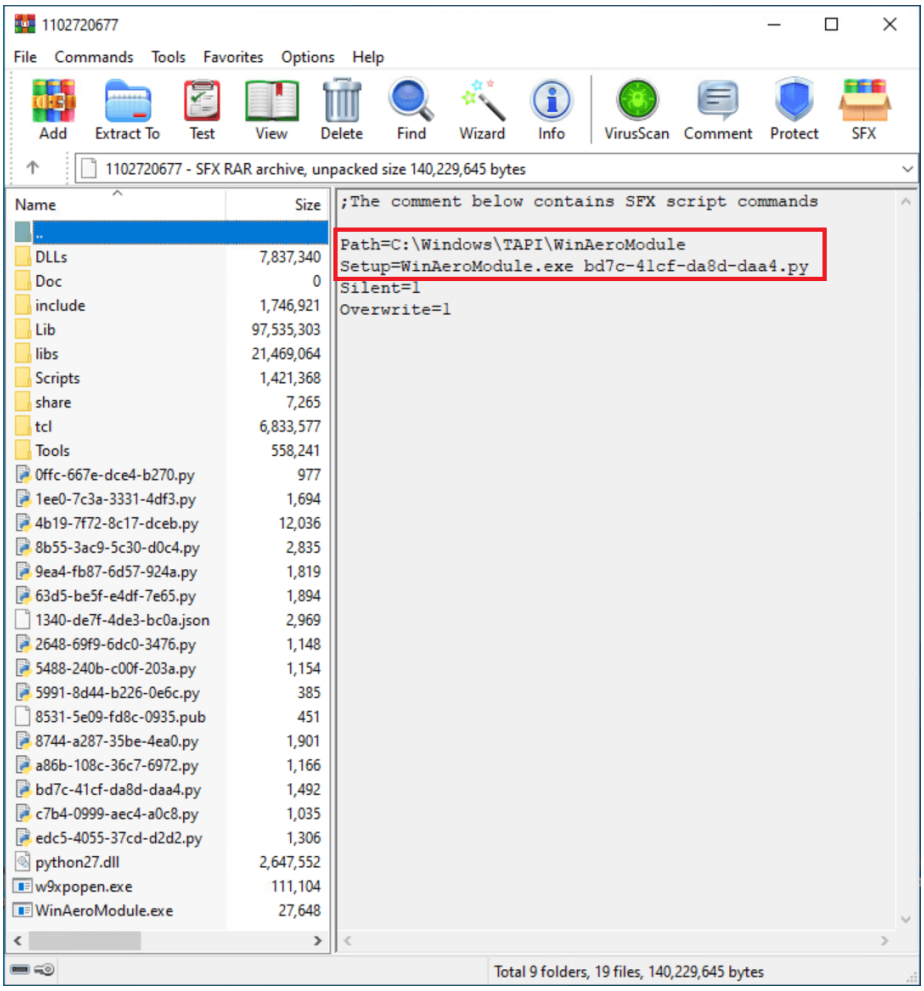

Figure 3. Contents of GoldenHowl's self-extracting archive

The initial script in GoldenHowl, referred to as core_script in the malware's configuration file, performs the following actions:

- decrypts and loads the malware's configuration from a JSON file,
- creates directories used by the malware, and
- starts a thread for each module.

The malware's configuration is decrypted using the Fernet algorithm, with the hardcoded key _ylmUTbqcx6FxMZ5ZvNxDQZYuNh41yxhKcPJLzxgqEY=. Figure 4 shows part of the decrypted configuration.

```
{
  "common" : {
    "use_minipy" : true,
    "files" : {
      "core_script" : "bd7c-41cf-da8d-daa4.py",
      "config_file" : "1340-de7f-4de3-bc0a.json",
      "log_file" : "37d6-006b-2e41-d89b.txt"
    },
    "config_encryption_key" : "_ylmUTbqcx6FxMZ5ZvNxDQZYuNh41yxhKcPJLzxgqEY=",
    "log_need" : false,
    "client_id" : "8102",
    "request_suffix": "0c3a",
    "response_suffix": "c776",
    "dirs" : {
      "temp_dir" : "5bc5-0788-d469-2f3a",
      "data_dir" : "cda2-b818-3403-b564",
      "upload_dir" : "b307-05ea-7ac8-c369",
      "download_dir" : "a700-280c-f067-5a06"
    },
    "data_dir_size_mb" : 5000,
    "upload_dir_size_mb" : 5000,
    "download_dir_size_mb" : 5000,
    "data_dir_delete_new" : false,
    "upload_dir_delete_new" : false,
    "download_dir_delete_new" : false
  },
  "modules" : {
    "persistence_schtasks" : {
      "script" : "5991-8d44-b226-0e6c.py",
      "task_name": "Microsoft\\Windows\\Multimedia\\SystemSoundsService2",
      "is_system" : true,
      "highest" : true
    },
    "files_tree" : {
      "script" : "edc5-4055-37cd-d2d2.py",
      "module_id" : "193a",
      "interval_sec" : 10,
      "cut_files" : true,
      "cut_files_size_mb" : 5
    },
```


Figure 4. Part of GoldenHowl's decrypted configuration

Table 2 shows the Python modules that we've observed – in the order that they appear in the config – along with a description of their functionalities. All modules run indefinitely, except for the persistence_schtasks module, which runs only once.

Table 2. Malicious modules in GoldenHowl

Module name	File on disk	Description
persistence_schtasks	5991-8d44-b226-0e6c.py	Creates the scheduled task Microsoft\Windows\Multimedia\SystemSoundsService2 to persist the execution of core_script.
files_tree	edc5-4055-37cd-d2d2.py	Generates a listing of files and directories by calling Windows' tree command, for a path specified in a request sent by the C&C.
files_stealer	5488-240b-c00f-203a.py	Exfiltrates a single file to the C&C server. The file path is specified in a request sent by the C&C.
data_transform	8744-a287-35be-4ea0.py	Utility module that takes incoming requests from the C&C server and decrypts them, and takes responses from other modules that need to be sent to the C&C and encrypts them. The encryption algorithm is Fernet, and the key is specific to this module: QRqXhd_iB_Y3LpT2wTVK6Dao5uOq2m5KMiVkMnJfgw4=
transport_http	63d5-be5f-e4df-7e65.py	Utility module that uploads and downloads files from the C&C server. See the *C&C communication* section for more information. Note that the word transport is commonly used by Turla and MoustachedBouncer to refer to a type of C&C protocol. Although this might be shared across Russian-speaking developers, this is a low confidence element for attribution.
updater	c7b4-0999-aec4-a0c8.py	Utility module that receives a ZIP archive with updated modules or configuration from the C&C server, extracts the archive, and runs core_script in a new process, terminating the current process.
sshcmd	1ee0-7c3a-3331-4df3.py	Connects to an SSH server specified in a request sent by the C&C. Acts as a reverse shell, executing commands received from the C&C.
ipscanner	a86b-108c-36c7-6972.py	Generates a listing with active IP addresses in an IP range, based on an IP mask specified in a request sent by the C&C server. To do so, it first sends a message to all IP addresses in the range, on port 59173, and then it runs the command arp -a to obtain the ARP cache tables for all interfaces.
portscanner	2648-69f9-6dc0-3476.py	Generates a listing with ports that are accepting connections, based on an IP address and a list of ports specified in a request sent by the C&C server.

Module name	File on disk	Description
sshtunnel	9ea4-fb87-6d57-924a.py	Creates an SSH tunnel with an SSH server, to forward messages going from (and to) a host on a listening port, to a forwarding port on the SSH server. A request from the C&C server specifies: the address and port of the SSH server, username and password for the SSH session, the forwarding port on the SSH server, and the address and port of the listening host.
eternalbluechecker	4b19-7f72-8c17-dceb.py	Checks whether a host, specified in a request sent by the C&C server, is vulnerable to a Windows SMB remote code execution vulnerability. The code for this module is the same as in mysmb.py and checker.py from this public repository. There is no code in this module to exploit vulnerable hosts.
socks_proxy	8b55-3ac9-5c30-d0c4.py	Acts as a proxy server, forwarding packets from a source address to a destination address. The port to listen for incoming connections is specified in a request sent by the C&C server. The code in this module is very similar to that of pysoxy.
text_writer	0ffc-667e-dce4-b270.py	Writes a text file to a given path. The path and text for writing are specified in a request sent by the C&C server.

C&C communication

According to GoldenHowl's configuration, anything that comes from the C&C server is called a request, and files going to the C&C server represent a response. It should be noted that despite this naming convention, GoldenHowl is not a passive implant: it initiates the connections to the C&C server. The transport_http module is responsible for communication with the C&C server, and for writing requests and responses to specific directories. Table 3 shows directories used by GoldenHowl.

Table 3. Directories in GoldenHowl's configuration

Name in configuration	Name on disk	Description
download_dir	a700-280c-f067-5a06	Stores encrypted requests coming from the C&C server.
upload_dir	b307-05ea-7ac8-c369	Stores encrypted responses, with files or output of commands, to be sent to the C&C server.
data_dir	cda2-b818-3403-b564	Stores requests sent by the C&C server, which are taken from download_dir, decrypted, and placed in this directory for modules to process. Also stores output of executed commands (responses), which are taken from this directory, encrypted, and written to upload_dir. These actions are performed by the data_transform module.
temp_dir	5bc5-0788-d469-2f3a	This directory was not used in any observed modules.

Requests and responses have structured filenames:

- Request – <client_id><module_id><request_id><request_suffix>
- Response – <client_id><module_id><request_id><response_suffix>

The fields client_id, request_suffix, and response_suffix are specified in the configuration and are common to all modules (see Figure 4 for examples). The field module_id indicates which module needs to process a request or generate a response, and is defined in the configuration section of each individual module. The field request_id is generated on the C&C server, and ties together requests with responses.

The transport_http module sends GET requests periodically to the C&C server to check for available requests. The configuration fields specific to this module are:

- server_address – address of the C&C server (we observed 83.24.9[.]124, the same address as GoldenDealer's server),
- server_port – the port used to communicate with the server (we observed 443),
- server_use_ssl – indicates whether HTTP or HTTPS will be used for communication,
- base_timeout_sec – the minimum wait time before contacting the C&C server, and
- timeout_filename – the filename of a text file with a number between 0 and 3600, to specify a different number of seconds to wait between communications. This file is not included with GoldenHowl, but it could be placed by the attackers with the text_writer module.

The address used to contact the C&C server is https://<server_address>:<server_port>/<client_id>/.

GoldenRobo

The last of the observed tools from the 2019 toolset, GoldenRobo, is a simple component written in Go that executes the Robocopy utility to stage files and send them to its C&C server. It iterates across all drive letters from A to Z, trying to access each drive. If successful, a Robocopy command is constructed:

robocopy <drive_letter>:\ <malware_folder>\1516-fe89-ad12-8102\<drive_letter>\ *.doc *.docx *.xls *.xlsx *.ppt *.pptx *.pdf *.rtf *.tif *.jpg *.jpeg *.crt *.key *.p12 *.ovpn *.zip *.rar "" /S /R:0 /MAXAGE:7 /XD <malware_folder>\1516-fe89-ad12-8102\\ 1516-fe89-ad12-8102\\ "Application Data" "All Users" "Documents and Settings" "Local Settings"

If we break down the command:

- files are copied from drive_letter,
- to <malware_folder>\1516-fe89-ad12-8102\<drive_letter>\ (for example: C:\Windows\TAPI\1516-fe89-ad12-8102\C\),
- only if the files have extensions .doc, .docx, .xls, .xlsx, .ppt, .pptx, .pdf, .rtf, .tif, .jpg, .jpeg, .crt, .key, .p12, .ovpn, .zip, or .rar,
- including subdirectories (/S),
- not retrying on failed copies (/R:0),
- excluding files older than seven days (/MAXAGE:7), and
- excluding specified directories (/XD).

Copied files from all drives are archived together in a ZIP file _1423-da77-fe86<month>-<day> in the same directory where GoldenRobo is running (with <month> and <day> corresponding to the current date).

The archive is sent base64 encoded to https://83.24.9[.]124/8102/. The last part of the URL is 8102, which is the same as the client_id field in GoldenHowl's configuration. This URL is hardcoded in GoldenRobo, which tells us that the attackers compiled this version of GoldenRobo for this victim exclusively.

The known toolset: Previously documented by Kaspersky

A few weeks after deploying the previous toolset, GoldenJackal started to use other malicious tools on the same compromised computers. In September 2019, we observed the execution of PowerShell scripts to download the JackalControl backdoor. This backdoor was used to execute other PowerShell scripts, to download and run legitimate tools such as Plink and PsExec.

In various attacks, between September 2019 and January 2024, we observed the following tools in GoldenJackal's arsenal:

- JackalControl,
- JackalSteal, a file collector and exfiltrator, and
- JackalWorm, used to propagate other malicious components via USB drives. We observed it propagating the JackalControl backdoor.

As these components have already been documented by Kaspersky, we will not describe them in this blogpost. However, one interesting point to mention is that in early versions of these tools, URLs for C&C servers were hardcoded in the malware binaries. At some point, GoldenJackal modified JackalControl and JackalSteal to receive C&C servers as arguments.

The latest toolset: Keeping a foothold in the network

In May 2022, we observed GoldenJackal using a new toolset while targeting a governmental organization in Europe. Most of these tools are written in Go and provide diverse capabilities, such as collecting files from USB drives, spreading payloads in the network via USB drives, exfiltrating files, and using some PCs in the network as servers to deliver diverse files to other systems. In addition, we have seen the attackers using Impacket to move laterally across the network.

In the observed attacks, GoldenJackal started to use a highly modular approach, using various components to perform different tasks. Some hosts were abused to exfiltrate files, others were used as local servers to receive and distribute staged files or configuration files, and others were deemed interesting for file collection, for espionage purposes. Figure 5 shows a classification of the components that are described over the next sections.

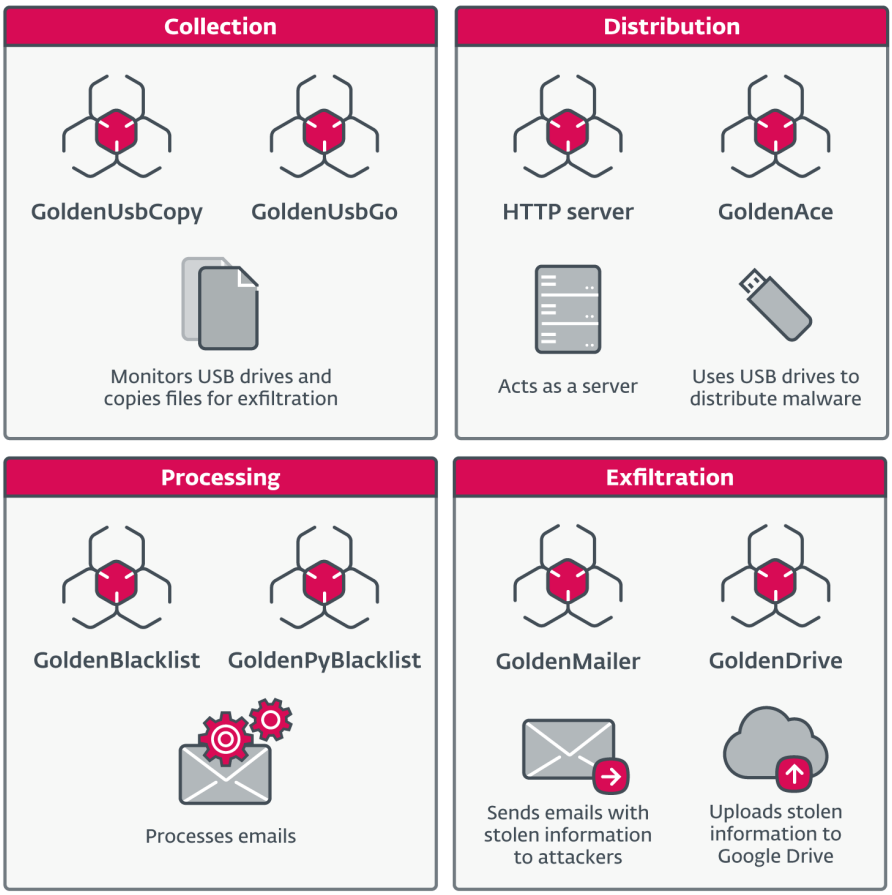

Figure 5. Components in GoldenJackal's latest toolset

Regarding network infrastructure, we didn't observe any external IP addresses in any of the analyzed components. File exfiltrators use publicly available services such as Google Drive or Outlook SMTP servers.

GoldenUsbCopy

GoldenUsbCopy, which we classify as a collection component, monitors the insertion of USB drives, and copies interesting files to an encrypted container that is stored on disk, to be exfiltrated by other components. According to strings found in the binary, the main package for the application is named UsbCopy.

Newly inserted drives are detected by creating a window with name WindowsUpdateManager, to receive system events and process them with a custom handler. If the handler receives a WM_DEVICECHANGE message, with an event type DBT_DEVICEARRIVAL, and the device type is DBT_DEVTYP_VOLUME, this means a new drive is ready to be processed. Figure 6 shows a side-by-side comparison between decompiled code in GoldenUsbCopy and GoldenDealer. Even though each was written in a different programming language, we can see that the code retrieves the letter of the drive to process in the same manner.

```
if ( (_DWORD)wparam != WM_DEVICECHANGE || a5 != DBT_DEVICEARRIVAL
                    || a6->dbcv_devicetype != DBT_DEVTYP_VOLUME )
  return DefWindowProcA(hwnd, uMsg, wparam, lparam);
dbcv_unitmask = a6->dbcv_unitmask;
for ( i = 0; i != 26; ++i )
{
  if ( (dbcv_unitmask & 1) != 0 )
  {
    v10 = i + 'A';
    goto LABEL_9;
  }
  dbcv_unitmask >>= 1;
}
v10 = 91;
LABEL_9:
SetWindowLocalTime = v10;
WindowQueryClassEx = 1;
PostQuitMessage((int)hwnd);
wparam = WM_DEVICECHANGE;
return DefWindowProcA(hwnd, uMsg, wparam, lparam);
```

GoldenUsbCopy

```
if ( a2 == WM_DESTROY )
{
  PostQuitMessage(0);
  return 0;
}
if ( a2 != WM_DEVICECHANGE )
  return DefWindowProcW(hWnd, a2, a3, (LPARAM)a4);
if ( a3 != DBT_DEVICEARRIVAL || a4->dbcv_devicetype != DBT_DEVTYP_VOLUME )
  return 0;
dbcv_unitmask = a4->dbcv_unitmask;
for ( i = 0; i < 26; ++i )
{
  if ( (dbcv_unitmask & 1) != 0 )
    break;
  dbcv_unitmask >>= 1;
}
PostMessageW(hWnd, 1025u, (unsigned __int16)(i + 'A'), 0);
return 1;
```

GoldenDealer

Figure 6. Code comparison between GoldenUsbCopy and GoldenDealer

GoldenUsbCopy determines which files to process from a USB drive based on a configuration that is stored AES encrypted in CFB mode in the file reports.ini. The 32-byte key to decrypt the configuration is hardcoded in the malware. After decryption, the configuration contains the following fields, in JSON format:

- outputCipherFilename – full path to an encrypted archive that acts as a container for other files, such as files that contain listings of filenames from newly inserted drives, and files to be exfiltrated,

- RSAKey – a public key to encrypt AES keys that are used to encrypt files to be exfiltrated,
- lastDate – files that were last modified more than lastDate days ago are not processed,
- registryKey – a key in HKEY_CURRENT_USER that will store SHA-256 hashes of files already processed for exfiltration,
- registryValue – the registry value that stores the list of hashes,
- maxZIPSize – the maximum size in bytes for outputCipherFilename (more details below),
- maxFileSize – files larger than maxFileSize, in bytes, are not exfiltrated, and
- extensionsFile – a list of file extensions for exfiltration (we observed .docx, .pdf, .doc, and .odt).

Once the configuration is decrypted, GoldenUsbCopy waits for a USB drive to be inserted. A listing of all files on the inserted drive is written to a text file, which is then archived in a ZIP file, encrypted with AES, and added to outputCipherFilename. Only the encrypted container is written to disk; intermediate steps, involving text files and archives, are kept in memory.

A similar procedure is done for files on the drive that meet the criteria for exfiltration: these files are archived together preserving their directory structure, encrypted with AES, and added to outputCipherFilename. When selecting files for exfiltration, a list with SHA-256 hashes is retrieved from the registry. If the hash of a file is in that list, the file is not exfiltrated. If the hash isn't in the list, it is added, so that the file won't be exfiltrated again.

Whenever adding files to exfiltrate would exceed the maxZIPSize of outputCipherFilename, the excess files are not added to the archive for exfiltration, but their paths are added to a text file that is archived, encrypted, and added to outputCipherFilename.

Regarding encryption, each individual archive that is added to the encrypted container is encrypted with AES in CFB mode, with a key and an initialization vector (IV) that are randomly generated on the spot. Both the key and IV need to be stored, but only the key is encrypted with RSAKey. Figure 7 shows an example of how these fields are stored in the encrypted container.

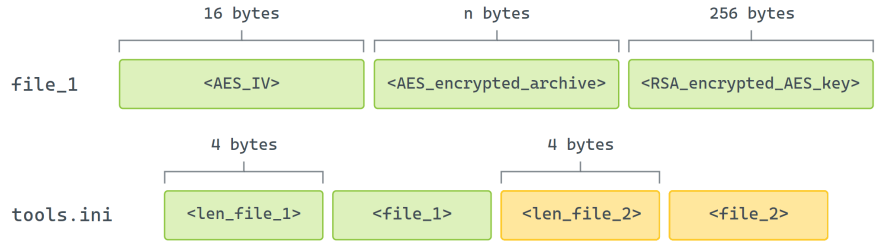

Figure 7. Structure of the encrypted container

GoldenUsbGo

This component is very similar to GoldenUsbCopy and seems to be a later version of it, based on when we observed them in our telemetry and comparing Go versions used to compile them. However, GoldenUsbGo achieves the same functionality with a simpler implementation:

- There is no configuration file. All criteria for file selection are hardcoded in the malware:
  - if filename contains a specific word from a list, process the file regardless of all other criteria (the list contains strings such as pass, login, and key),
  - else, file size must be no bigger than 20 MB,
  - the date the file was last modified must be no more than 14 days ago, and
  - the file extension must be one of .pdf, .doc, .docx, .sh, or .bat.
- Insertion of removable drives is not continuously monitored. A hardcoded list of drive letters is checked periodically to determine if they have an assigned volume of D:, E:, F:, G:, or H:.
- The list of hashes of files that were already processed is kept in memory only.
- There is no size limit for the encrypted container where files are staged for exfiltration.
- Files are not archived but instead are compressed with gzip. Both file contents and filenames are compressed. Figure 8 shows how compressed data is arranged before encryption.

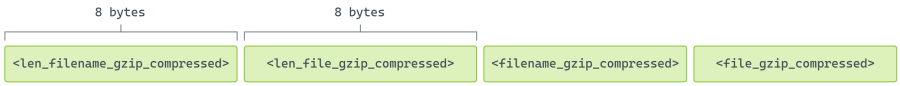

Figure 8. Fields used for gzip-compressed files, before encryption

The path to the encrypted container is hardcoded in the malware:

C:\Users\[redacted]\appdata\local\SquirrelTemp\SquirrelCache.dat

The hardcoded username in the path, redacted above, along with the short list of drives and specific filenames to process, tell us that GoldenUsbGo was compiled and tailored for this particular victim.

Compressed files are encrypted with AES in CFB mode with the hardcoded key Fn$@-fR_*+!13bN5. The structure is the same as in GoldenUsbCopy (shown in Figure 7) but without the AES key. After compressing the files, GoldenUsbGo generates a listing of all files on the inserted drive and adds it to the encrypted container, in the same manner as exfiltrated files. The filename for the listing is formed from the current date and time, replacing : with - (for example, 15 Jan 24 13-21 PST).

GoldenAce

This component, which we classified as a distribution tool in Figure 5, serves to propagate other malicious executables and retrieve staged files via USB drives. While it could be used to target air-gapped systems, it's not specifically built for that, as opposed to GoldenDealer. It works together with a lightweight version of JackalWorm and some other unknown component.

GoldenAce periodically checks drives in the list G:, H:, I:, J:, K:, L:, M:, N:, P:, X:, Y:, and Z:, to find one that is mapped to a volume. Then it checks whether a trash directory exists in the root of that drive. If it doesn't exist, it is created as hidden, and a file called update is copied to that directory, from the same location where GoldenAce is running. The first directory on the drive (in alphabetical order) that is not hidden is set to hidden, and a file called upgrade is copied to the root of the drive and renamed as <name_of_hidden_directory>.exe.

The file upgrade is actually JackalWorm, an executable that uses a folder icon, and whose purpose is to copy and run the update file on another system where the USB drive is inserted. Unlike the version of JackalWorm described by Kaspersky, this one is very limited: it doesn't have code to monitor drive insertions, and it cannot be configured to perform various actions. When executed from the root directory of a removable drive, it opens the hidden folder in Windows Explorer and writes a batch file to execute the payload in update. Contents of this file, update.bat, are shown in Figure 9.

```
@echo off
copy "<drive_letter>:\\trash\\update"
"C:\\Users\\%username%\\AppData\\Local\\update.exe"
"C:\\Users\\%username%\\AppData\\Local\\update.exe" "<drive_letter>:\\trash"
:check1
@tasklist | findstr /i /b "update.exe" >nul
@if %errorlevel%==0 goto check1
@del /f /q /a h "C:\\Users\\%username%\\AppData\\Local\\update.exe"
@del /f /q "C:\\Users\\<username>\\AppData\\Local\\update.bat"
```

Figure 9. Contents of update.bat

We can see that update is run and deleted, along with the batch file, once it's done running. While we didn't observe the contents of the update component, it is likely that it collects files and stages them in the trash directory on the removable drive, since the path to that directory is passed as an argument to update.

When GoldenAce finds that the directory trash already exists on a drive, instead of copying files to the drive, it copies files in the trash directory to C:\ProgramData\Microsoft\Windows\DeviceMetadataCache.

HTTP server

We observed Python's HTTP server, packaged with PyInstaller, being executed via C:\Windows\system32\cmd.exe /K C:\Windows\msahci.cmd. Unfortunately, we didn't observe the contents of the msahci.cmd file, so we don't know the arguments passed for execution, such as the port for the server to listen on.

GoldenBlacklist

As a processing component, GoldenBlacklist downloads an encrypted archive from a local server, and processes email messages contained in it, to keep only those of interest. Then it generates a new archive for some other component to exfiltrate.

The URL to retrieve the initial archive is hardcoded: https://<local_ip_address>/update46.zip. The downloaded file is saved as res.out, and AES decrypted with the hardcoded key k9ksbu9Q34HBKJuzHIuGTfHL9xCzMl53vguheOYA8SiNoh6Jqe62F7APtQ9pE, using a legitimate OpenSSL executable.

The decrypted archive, update46.tar.gz, is extracted in memory, and only those files that match certain criteria are written to a subdirectory tmp, in the directory where the malware is running. Criteria:

- The file does not contain any email on a blocklist of email addresses. This is done to remove email messages that come from senders that usually are not interesting. While we can't include the full list here, it's worth mentioning that many of the email addresses are related to newsletters and press releases. It's important to note that the attackers must have been operating for some time to build a list like this.

- The file contains the string Content-Type: application. This is to keep email messages that have attachments, such as PDF files, Microsoft Office files, and archives, to name a few.

Once the files are selected, GoldenBlacklist archives the tmp directory and encrypts it with openssl.exe, using the same encryption key as the one used to decrypt the initial archive. The resulting file is archive.out. All intermediate files and folders are then deleted, as well as openssl.exe, libssl-3-x64.dll, and libcrypto-3-x64.dll, all located in the malware's directory. This indicates that another component that we didn't observe copied those legitimate binaries there in the first place.

GoldenPyBlacklist

GoldenPyBlacklist is a Python implementation of GoldenBlacklist. It was packaged with PyInstaller and the original name of the script is duplxer_black_list_for_external_use.py. Some differences to the other component are:

- the initial archive is written as ress.out,
- the key for decryption is the same, except for a different first character,
- the decrypted archive is extracted to the C:\Windows\System32\temp directory for processing,
- one additional criterion for file selection is added to process only filenames that end in .msg (these are files created with Microsoft Outlook),
- files that do not meet the criteria are deleted,
- the final archive is created with the 7-Zip archiver, and
- the final encrypted file is named ArcSrvcUI.ter.

GoldenMailer

Classified as an exfiltration component, GoldenMailer exfiltrates files by sending emails with attachments to attacker-controlled accounts. It was written in Python and packaged with PyInstaller, and the original name of the script is send_to_hole.py. GoldenMailer connects to legitimate servers – either smtp-mail.outlook.com or smtp.office365.com – to send email messages, using SMTP on port 587.

The configuration is read from a file, C:\ProgramData\Microsoft\Windows\Caches\cversions.ini, in the same directory where GoldenMailer is running. The configuration consists of the following five lines:

- email address to authenticate to the SMTP server, and to use as both sender and destination address,
- password to authenticate to the SMTP server,
- path to directory with archives to exfiltrate,
- base filename (e.g., press.pdf) used for archives to exfiltrate; these archives use the following naming convention: <base_filename>.<three_digit_sequence_number>, and
- number of files to exfiltrate.

We noticed that this configuration file was copied from another PC in the local network. Given that the configuration file indicates how many archives are available to be exfiltrated, we assume that these archives must also be copied over the network, separating the tasks of collection, distribution, and exfiltration. It is likely that the configuration file is generated by the component in charge of collecting files and creating archives for exfiltration, but we didn't observe that component.

Figure 10 shows an example of an email message sent by GoldenMailer. The subject has a typo: it reads Press realese. The body is very simple and reads: Daily News about Israel-Hamas war. These strings are hardcoded in the malware's binary. Only one attachment is sent per email; if there are many archives to exfiltrate, one email is sent for each.

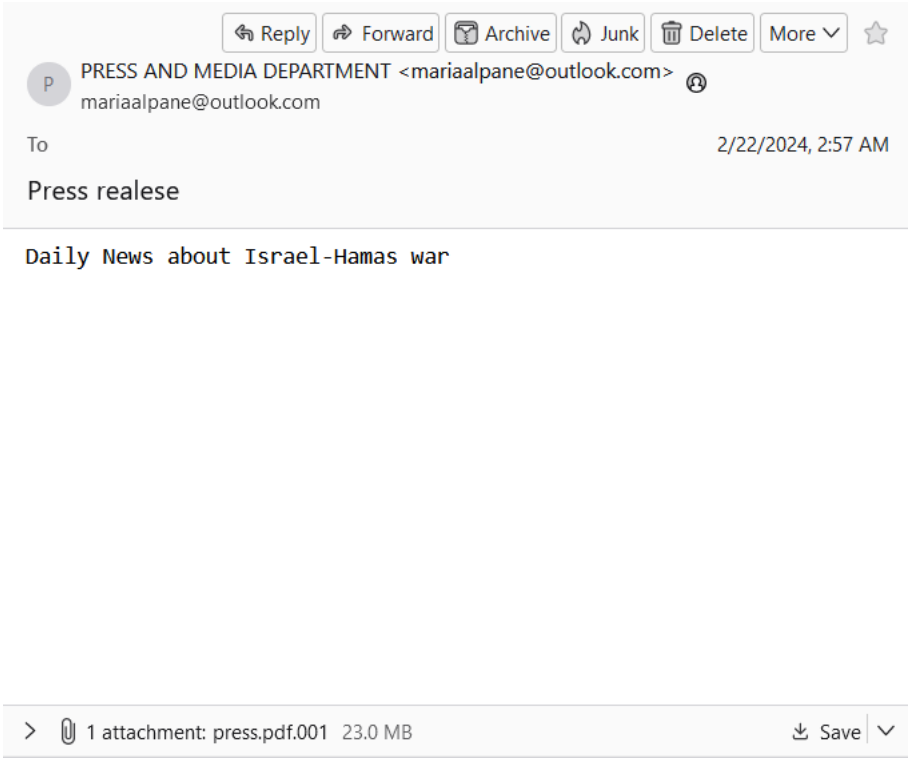

Figure 10. Example of an email message used to exfiltrate files

The configuration files that we observed contained the following email addresses:

- mariaalpane@outlook[.]com
- katemarien087@outlook[.]com
- spanosmitsotakis@outlook[.]com

GoldenDrive

As opposed to GoldenMailer, this component exfiltrates files by uploading them to Google Drive. Necessary credentials are found in two files, which are hardcoded in the malware: credentials.json, which contains fields such as client_id and client_secret, and token.json, with fields such as access_token and refresh_token. A reference to Google Drive's API and some code snippets in the Go programming language can be found here.

Similar to GoldenMailer, this component can upload only one file at a time. GoldenDrive is executed with an argument that provides the full path to the file to upload.

Conclusion

In this blogpost, we revealed two new toolsets used by the GoldenJackal APT group to target air-gapped systems of governmental organizations, including those in Europe. Common functionalities include the use of USB drives to steal confidential documents.

Managing to deploy two separate toolsets for breaching air-gapped networks in only five years shows that GoldenJackal is a sophisticated threat actor aware of network segmentation used by its targets.

A comprehensive list of indicators of compromise (IoCs) can be found in our GitHub repository.

> *For any inquiries about our research published on WeLiveSecurity, please contact us at threatintel@eset.com.*
> *ESET Research offers private APT intelligence reports and data feeds. For any inquiries about this service, visit the ESET Threat Intelligence page.*

IoCs

Files

SHA-1	Filename	Detection	Description
DA9562F5268FA61D19648DFF9C6A57FB8AB7B0D7	winaero.exe	Win32/Agent.AGKQ	GoldenDealer

SHA-1	Filename	Detection	Description
5F12FFD272AABC0D5D611D18812A196A6EA2FAA9	1102720677	Python/Agent.ANA Python/HackTool.Agent.W Python/Riskware.LdapDump.A Python/Riskware.Impacket.C	GoldenHowl.
6DE7894F1971FDC1DF8C4E4C2EDCC4F4489353B6	OfficeAutoComplete.exe	WinGo/Agent.AAO	GoldenRobo.
7CB7C3E98CAB2226F48BA956D3BE79C52AB62140	prinntfy.dll	WinGo/DataStealer.A	GoldenUsbCo
8F722EB29221C6EAEA9A96971D7FB78DAB2AD923	zUpdater.exe	WinGo/Spy.Agent.AH	GoldenUsbGo
24FBCEC23E8B4B40FEA188132B0E4A90C65E3FFB	fc.exe	WinGo/DataStealer.C	GoldenAce.
A87CEB21EF88350707F278063D7701BDE0F8B6B7	upgrade	MSIL/Agent.WPJ	JackalWorm – simpler versio
9CBE8F7079DA75D738302D7DB7E97A92C4DE5B71	fp.exe	WinGo/Spy.Agent.CA	GoldenBlackli
9083431A738F031AC6E33F0E9133B3080F641D90	fp.exe	Python/TrojanDownloader.Agent.YO	GoldenPyBlac
C830EFD843A233C170285B4844C5960BA8381979	cb.exe	Python/Agent.ALE	GoldenMailer.
F7192914E00DD0CE31DF0911C073F522967C6A97	GoogleUpdate.exe	WinGo/Agent.YH	GoldenDrive.
B2BAA5898505B32DF7FE0A7209FC0A8673726509	fp.exe	Python/Agent.ALF	Python HTTP server.

Network

IP	Domain	Hosting provider	First seen	Details
83.24.9[.]124	N/A	Orange Polska Spolka Akcyjna	2019-08-09	Primary C&C server used by GoldenJackal in 2019.
196.29.32[.]210	N/A	UTANDE	2019-08-09	Secondary C&C server used by GoldenJackal in 2019.
N/A	assistance[.]uz	N/A	2019-09-25	Compromised website used to download malware.
N/A	thehistore[.]com	N/A	2019-09-25	Compromised website used as a C&C server.
N/A	xgraphic[.]ro	N/A	2019-09-25	Compromised website used as a C&C server.

Email Addresses

- mariaalpane@outlook[.]com
- katemarien087@outlook[.]com
- spanosmitsotakis@outlook[.]com

MITRE ATT&CK techniques

This table was built using version 15 of the MITRE ATT&CK framework.

Tactic	ID	Name	Description
Resource Development	T1583.003	Acquire Infrastructure: Virtual Private Server	GoldenJackal probably acquired a VPS server to use as a secondary C&C server for the GoldenDealer malware.
	T1583.004	Acquire Infrastructure: Server	GoldenJackal likely acquired a server to use as a primary C&C server for the GoldenDealer malware.
	T1584.006	Compromise Infrastructure: Web Services	GoldenJackal has used compromised WordPress sites for C&C infrastructure, used by the JackalControl and JackalSteal malware.
	T1587.001	Develop Capabilities: Malware	GoldenJackal develops its own custom malware.
	T1585.003	Establish Accounts: Cloud Accounts	GoldenJackal has used Google Drive to store exfiltrated files and legitimate tools.
	T1588.002	Obtain Capabilities: Tool	GoldenJackal uses legitimate tools, such as Plink and PsExec, for post-compromise operations.
Execution	T1059.001	Command and Scripting Interpreter: PowerShell	GoldenJackal executed PowerShell scripts to download the JackalControl malware from a compromised WordPress website.
	T1059.003	Command and Scripting Interpreter: Windows Command Shell	GoldenAce uses cmd.exe to run a batch script to execute other malicious components.
	T1059.006	Command and Scripting Interpreter: Python	GoldenHowl contains various malicious modules that are Python scripts.
	T1106	Native API	GoldenDealer can copy and run an executable file with the CreateProcessW API.
	T1569.002	System Services: Service Execution	GoldenDealer can run as a service.
	T1204.002	User Execution: Malicious File	JackalWorm uses a folder icon to entice a potential victim to launch it.
Persistence	T1543.003	Create or Modify System Process: Windows Service	GoldenDealer creates the service NetDnsActivatorSharing to persist on a compromised system.

Tactic	ID	Name	Description
	T1547.001	Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder	If GoldenDealer fails to create a service for persistence, an entry in a Run registry key is created instead.
	T1053.005	Scheduled Task/Job: Scheduled Task	GoldenHowl creates the scheduled task Microsoft\Windows\Multimedia\ SystemSoundsService2 for persistence.
Defense Evasion	T1564.001	Hide Artifacts: Hidden Files and Directories	GoldenDealer modifies the registry so that hidden files and directories are not shown in Windows Explorer. GoldenDealer, GoldenAce, and Jackal worm create hidden folders on USB drives.
	T1070.004	Indicator Removal: File Deletion	GoldenAce deletes payloads after they are run. GoldenBlacklist and GoldenPyBlacklist delete intermediate files after the final archives are generated.
	T1036.005	Masquerading: Match Legitimate Name or Location	GoldenUsbCopy uses a legitimate Firefox directory C:\Users\ <username>\AppData\Roaming\Mozilla\Firefox\ to stage files.
	T1036.008	Masquerading: Masquerade File Type	JackalWorm uses a folder icon to disguise itself as a non-executable file.
	T1112	Modify Registry	GoldenDealer modifies the registry so that hidden files and directories are not shown in Windows Explorer.
	T1027.013	Obfuscated Files or Information: Encrypted/Encoded File	GoldenJackal uses various encryption algorithms in its toolset, such as XOR, Fernet, and AES, to encrypt configuration files and files to be exfiltrated.
Credential Access	T1552.001	Unsecured Credentials: Credentials In Files	GoldenUsbGo looks for files with filenames that are usually associated with credentials.
	T1552.004	Unsecured Credentials: Private Keys	GoldenUsbGo looks for files that may contain private keys, such as those with filenames that contain id_rsa.
Discovery	T1087.001	Account Discovery: Local Account	GoldenDealer collects information about all user accounts on a compromised system.
	T1083	File and Directory Discovery	GoldenHowl has a module to generate a listing of files and directories on a compromised system. GoldenUsbCopy and GoldenUsbGo generate a listing of files and directories on a USB drive.
	T1046	Network Service Discovery	GoldenHowl can scan a remote system for open ports, and whether the target is vulnerable to EternalBlue malware.
	T1120	Peripheral Device Discovery	GoldenDealer and GoldenUsbCopy monitor the insertion of removable drives. GoldenUsbGo and GoldenAce check for various drive letters, to detect attached removable drives.
	T1057	Process Discovery	GoldenDealer obtains information about running processes on a compromised system.
	T1018	Remote System Discovery	GoldenHowl can scan an IP range to discover other systems.
	T1518	Software Discovery	GoldenDealer obtains information about installed programs on a compromised system.
	T1082	System Information Discovery	GoldenDealer obtains various information about the operating system and user accounts on a compromised system.
	T1016.001	System Network Configuration Discovery: Internet Connection Discovery	GoldenDealer can determine whether a computer is connected to the internet.
	T1135	Network Share Discovery	GoldenAce checks a list of drive letters that can include network shares.
Lateral Movement	T1210	Exploitation of Remote Services	GoldenHowl can check for a Windows SMB remote code execution vulnerability that can then be exploited for lateral movement.
	T1091	Replication Through Removable Media	GoldenDealer copies executables to and from USB drives, to target air-gapped systems. GoldenAce propagates malicious executables via removable drives.
Collection	T1560.002	Archive Collected Data: Archive via Library	GoldenRobo and GoldenUsbCopy archive files to be exfiltrated with the ZIP library.
	T1119	Automated Collection	GoldenUsbCopy and GoldenUsbGo automatically stage files for later exfiltration, when a new removable drive is detected.
	T1005	Data from Local System	Most tools in GoldenJackal's toolset collect information and files from the local system.
	T1025	Data from Removable Media	GoldenUsbCopy and GoldenUsbGo collect interesting files from removable media. GoldenAce can retrieve staged files from a specific directory on a removable drive. GoldenDealer can retrieve information from compromised systems from a specific directory on a removable drive.
	T1074.001	Data Staged: Local Data Staging	Most tools in GoldenJackal's toolset stage files locally for other components to process or exfiltrate them.

Tactic	ID	Name	Description
	T1114.001	Email Collection: Local Email Collection	GoldenBlacklist and GoldenPyBlacklist process email files that were collected by an unknown component in GoldenJackal's toolset.
Command and Control	T1071.001	Application Layer Protocol: Web Protocols	GoldenDealer and GoldenHowl use HTTPS for communication.
	T1092	Communication Through Removable Media	GoldenDealer uses removable media to pass executables to air-gapped systems, and information from those systems back to connected systems.
	T1132.001	Data Encoding: Standard Encoding	Executable files sent from the C&C server to GoldenDealer are base64 encoded.
	T1572	Protocol Tunneling	GoldenHowl can forward messages through an SSH tunnel.
	T1090.001	Proxy: Internal Proxy	GoldenHowl can act as a proxy, forwarding packets.
Exfiltration	T1041	Exfiltration Over C2 Channel	GoldenHowl exfiltrates files via the same channel used as its C&C.
	T1052.001	Exfiltration Over Physical Medium: Exfiltration over USB	GoldenJackal's toolset provides capabilities to copy files from air-gapped systems and move them to connected systems via USB drives, for exfiltration.
	T1567.002	Exfiltration Over Web Service: Exfiltration to Cloud Storage	GoldenDrive exfiltrates files to an attacker-controlled Google Drive account.
	T1048.002	Exfiltration Over Alternative Protocol: Exfiltration Over Asymmetric Encrypted Non-C2 Protocol	GoldenMailer exfiltrates files via SMTP, using STARTTLS on port 587.