EggStreme Malware: Unpacking a New APT Framework Targeting a Philippine Military Company

By Bogdan Zavadovschi

Published: 2025-09-17 · Archived: 2026-04-05 23:44:35 UTC

I'd like to thank my coauthors, Victor Vrabie, Adrian Schipor, and Martin Zugec, for their invaluable contributions to this research.

TL;DR A Chinese APT group compromised a Philippine military company using a new, fileless malware framework called EggStreme. This multi-stage toolset achieves persistent, low-profile espionage by injecting malicious code directly into memory and leveraging DLL sideloading to execute payloads. The core component, EggStremeAgent, is a full-featured backdoor that enables extensive system reconnaissance, lateral movement, and data theft via an injected keylogger.

This report analyzes a sophisticated cyber-attack targeting a military company based in the Philippines, which led to the discovery of a new and advanced malware toolset. Based on the target's strategic value and the geopolitical context of the South China Sea, the attackers' tactics, techniques, and procedures (TTPs) are consistent with those of Chinese APT groups. The attackers' primary focus was to achieve persistent access for long-term espionage and surveillance, highlighting the work of a highly professional threat actor whose objectives align with known national interests.

The core of our findings centers on the EggStreme framework, a tightly integrated set of malicious components. Unlike traditional malware, this framework operates with a clear, multi-stage flow designed to establish a resilient foothold on compromised systems. The attack begins with EggStremeFuel, which deploys EggStremeLoader to set up a persistent service. This loader then executes the EggStremeReflectiveLoader, which in turn launches the main EggStremeAgent.

The EggStremeAgent is the central nervous system of the framework. It operates by monitoring new user sessions and, for every new session detected, it injects the EggStremeKeylogger into the active explorer.exe process to silently collect keystrokes and other sensitive data. This agent is a full-featured backdoor with a broad range of capabilities. Its 58 commands enable the attackers to perform extensive local and network discovery, enumerate system resources, execute arbitrary shellcode, lateral movement, or inject other payloads, most notably the EggStremeWizard backdoor. The attackers use this to launch a legitimate binary that sideloads the malicious DLL, a technique they consistently abuse throughout the attack chain.

What makes this framework difficult to detect is its fileless nature. While encrypted malware components are present on the disk, the decrypted malicious code is executed and resides solely in memory, never touching the file system. This, coupled with the heavy use of DLL sideloading and the sophisticated, multi-stage execution flow, allows the framework to operate with a low profile, making it a significant and persistent threat.

Watch our LinkedIn Live discussion, Ctrl-Alt-DECODE, where we detailed our research on the EggStreme framework and answered live questions from around the world.

Technical Analysis Overview

The first sign of malicious activity that triggered our investigation in early 2024 was the execution of a logon batch script from an SMB share, located at \\<remote samba share>\netlogon\logon.bat. The exact method by which the script was placed on the SMB share is unknown and remains a key area of investigation.

The script's primary function was to deploy two files to the %APPDATA%\Microsoft\Windows\Windows Mail\ directory: a legitimate Windows binary named WinMail.exe and a malicious DLL named mscorsvc.dll. This is a classic example of DLL sideloading (read our explainer), a technique where an attacker places a malicious DLL in a location where a legitimate program will search for it. When the legitimate WinMail.exe is executed, it loads the malicious mscorsvc.dll instead of the system's original version. This allows the attacker to execute their malicious code under the guise of a trusted program, bypassing many security controls.

The malicious mscorsvc.dll is the first stage of the attack chain, referred to as EggStremeFuel. This component serves as a loader and is responsible for setting up the environment for the final payload. EggStremeFuel includes capabilities for system fingerprinting, which allows the attacker to gather information about the compromised machine. Its most critical function is to establish a reverse shell, which it does by invoking cmd.exe and creating a communication channel with the command-and-control (C2) server using read-write pipes. This provides the attacker with a remote command-line interface on the compromised system.

To maintain a persistent presence, the attacker abused several disabled Windows services. This was accomplished by either altering the service's associated registry key to point to a malicious executable or by directly replacing the legitimate service binary with their own. In both scenarios, the attacker configured the service to run with SeDebugPrivilege, a highly elevated right that allows a process to debug and access the memory of other processes on the system.

The malicious binary executed by these services is named EggStremeLoader. This component is responsible for reading a file at %WINDIR%\en-us\ielowutil.exe.mui that contains both the encrypted EggStremeReflectiveLoader and the EggStremeAgent payload. After decrypting the reflective loader, it injects it into a trusted process like winlogon.exe.

The EggStremeReflectiveLoader uses a token from its host process (winlogon.exe) to spawn a new, suspended process—either MsMpEng.exe or explorer.exe—using CreateProcessWithToken(). It then decrypts and injects the final payload, the EggStremeAgent, into this new process.

This final implant, named EggStremeAgent, is a sophisticated backdoor that communicates with the C2 server using the gRPC protocol. gRPC is a modern, high-performance, open-source framework for building remote procedure calls (RPCs).

EggStremeAgent is feature-rich, supporting a total of 58 distinct commands. These commands enable a wide range of capabilities, including:

- **System Fingerprinting:** Gathering detailed host information.
- **Resource Enumeration:** Scanning local and remote network resources.
- **Privilege Escalation:** Gaining higher-level permissions.
- **Command Execution:** Running arbitrary commands on the system.
- **Data Exfiltration:** Stealing sensitive data.
- **File and Directory Manipulation:** Creating, deleting, and modifying files.
- **Process Injection:** Injecting code into other running processes.

On several machines, a secondary, more lightweight backdoor was observed. The attacker used the legitimate xwizard.exe to sideload a malicious DLL named xwizards.dll, which has been named EggStremeWizard. This secondary backdoor provides reverse shell access and file upload/download capabilities. Its design also incorporates a list of multiple C2 servers, enhancing its resilience and ensuring that communication with the attacker can be maintained even if one C2 server is taken offline.

Persistence

After gaining access to the infrastructure through their initially deployed backdoor, EggStremeFuel, the attackers shifted their focus to establishing a stealthy form of persistence. To avoid detection, they leveraged legitimate Windows services that are not enabled by default—those configured with a startup type of Manual or Disabled. This allowed them to blend into normal system operations while maintaining access. Across multiple compromised machines, the following services were observed being abused:

Name	Description
MSiSCSI	Manages iSCSI sessions, enabling the computer to connect to and access remote iSCSI target devices
AppMgmt	Handles the installation, removal, and enumeration of software deployed via Group Policy

SWPRV	Manages software-based volume shadow copies created by the Volume Shadow Copy service

The persistence setup varied across machines, but two main approaches were identified. In the first, upon gaining access to a new machine, the attackers deployed the initial backdoor, EggStremeFuel, and used its reverse shell capabilities to manually execute the required commands. In other cases, tools resembling Impacket were observed being used to run commands manually.

When persistence was set up manually, the attackers altered file permissions, granted SeDebugPrivilege to the targeted service, and then started that service. A conceptual sequence of these commands is provided below:

takeown -f c:\\windows\\system32\\appmgmts.dll

icacls.exe c:\\windows\\system32\\appmgmts.dll /grant administrators:f

sc privs appmgmt SeDebugPrivilege

ren appmgmts.dll appmgmt.dll

ren svchost.dat appmgmts.dll

netsh advfirewall firewall add rule name="Windows Update" dir=in action=allow program="C:\\Windows\\explorer.exe" enable=yes

sc start appmgmt

In this sequence, a temporary malicious file named svchost.dat (which contains the malicious code) is renamed to a legitimate-looking filename, appmgmts.dll, to evade detection.

In other cases, instead of simply replacing the existing DLL file, the attackers modified the ServiceDLL registry value located at HKLM\SYSTEM\CurrentControlSet\Services\<serviceName>\Parameters to load a malicious DLL in place of the legitimate one. For example, with the MSiSCSI service, the path was changed from %systemroot%\system32\iscsiexe.dll to %systemroot%\system32\msiscsi.dll. An even subtler case was observed with the AppMgmt service, where the execution DLL was altered from appmgmts.dll (legitimate) to appmgmt.dll (malicious) - a change that is nearly indistinguishable at first glance.

Infrastructure

Every analyzed configuration file for the EggStremeAgent consistently used the same certificate authority (CA). This CA, identified by its unique Subject Key Identifier 51655e8e97fc7265b1aaa4265d94e2f7cae9c913, acted as the trusted root for the attackers' entire infrastructure. It issued certificates to all the C2 servers, enabling secure, mutual TLS communications. The details for this CA are as follows:

Serial Number: 2019 (0x7e3)
Signature Algorithm: sha256WithRSAEncryption
Issuer: C=, ST=, L=, street=, postalCode=, O=, OU=
Validity

Not Before: Nov 6 05:32:22 2019 GMT
Not After : Jul 11 10:45:22 7498 GMT
Subject: C=, ST=, L=, street=, postalCode=, O=, OU=
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
Public-Key: (2048 bit)
Exponent: 65537 (0x10001)
X509v3 extensions:
X509v3 Key Usage: critical
Digital Signature, Key Encipherment, Certificate Sign
X509v3 Extended Key Usage:
TLS Web Client Authentication, TLS Web Server Authentication
X509v3 Basic Constraints: critical
CA:TRUE
X509v3 Subject Key Identifier:
51:65:5E:8E:97:FC:72:65:B1:AA:A4:26:5D:94:E2:F7:CA:E9:C9:13 Signature Algorithm:
sha256WithRSAEncryption

By using the unique identifier from the certificate authority as a starting point, we were able to find other C2 servers that were also linked to it, ultimately revealing more of the attackers' network.

C2 Servers:

- whosecity[.]org
- webpirat[.]net
- ronaldmooremd[.]net
- kazinovavada[.]com

Our investigation discovered a C2 server at IP 154.90.35.190, which used a certificate for the domain fsstore[.]org. We then found a newer certificate for that same domain on a different IP, 45.115.224.163. The unique Authority Key Identifier (643042DF50CEF080E44851E7D5D6F654F772EBC5) on this new certificate suggests the attackers are actively refreshing their infrastructure. This identifier can be used to uncover other C2 IP addresses tied to the campaign, giving us a clearer view of their updated network.

Malware Analysis

The campaign's success is a direct result of a highly coordinated malware toolkit, not a collection of isolated implants. Each component serves a distinct purpose in the attack chain, from initial execution and persistence to in-memory payload delivery and final remote command and control. A deeper analysis reveals strong ties among the components, suggesting a single, unified development effort. This is evident in the consistent use of shared techniques like DLL sideloading, RC4 and XOR encryption, or fileless execution.

For example, a single file, ielowutil.exe.mui, was found to contain multiple encrypted payloads, including the reflective loader and the core backdoor itself, which are then injected directly into a trusted process to operate

entirely in memory. This section provides a detailed breakdown of each component, analyzing their functionality, communication methods, and role within the overall toolkit.

EggStremeFuel (The Stage 1 Loader)

This malicious DLL is designed to actively communicate with the C2 server. Telemetry shows that it's an initial payload, and it gets executed by being sideloaded by the legitimate binary %APPDATA%\\Microsoft\\Windows\\Windows Mail\\WinMail.exe. Once it's running, it spawns a reverse shell that gives the attackers the ability to run commands remotely.

When the DLL is loaded, a configuration structure is initialized in the memory with hardcoded values, including two C2 servers (a domain and an IP address), a main port (443), and a backup port (5228). A function then checks for an on-disk configuration file at %APPDATA%\\Microsoft\\Windows\\Cookies\\Cookies.dat. If found, it is decrypted with the RC4 key Cookies and the in-memory configuration is updated. If the file doesn't exist, the malware will create it, including the necessary directory structure.

Once the configuration is updated, the backdoor attempts to connect to a C2 server, starting with the domain. The client begins a handshake by sending a 32-byte message containing a 16-byte RC4 key and the first 16 bytes of that key's MD5 hash. Upon receiving a 32-byte response from the server, the client verifies that the last 16 bytes of the response match the hash it sent, confirming the handshake's integrity. If successful, it proceeds to fingerprint the machine by sending an encrypted JSON object with details such as hostname, IP addresses, OS, and MAC addresses:

{

"hostname": "<computerName>\\<username>",

"lip": "<local machine ip>",

"wip": "<result of request to myexternalip.com/raw>",

"os": "<windows version e.g Win7/Win8/Win Vista/>",

"mac": "<a list of all MAC addresses from NICs>",
"time": "<the machine time in format %02d:%02d:%02d>"
}

The JSON object is first serialized into a string, which is then encrypted using RC4. This encrypted data is then prefixed with the command ID, which is also encrypted with RC4, before being sent to the C2 server.

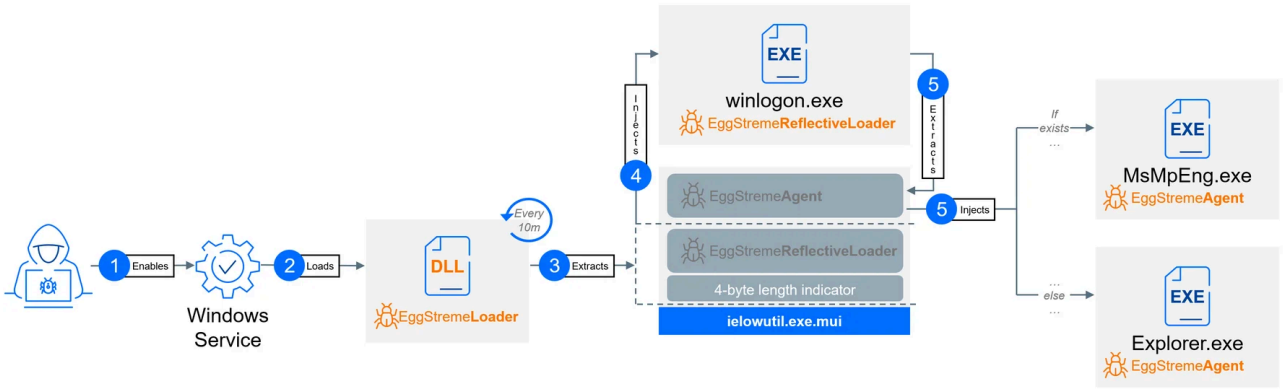
After fingerprinting, the client waits for commands from the server.

Command ID	Description
2|3	Get drive information, along with files and directories from a specified path and their metadata (FileTime, size, Name)

4	Start cmd.exe and establish communication via pipes, redirecting C2 input to the shell and sending the output back
5	Gracefully close all connections and shutdown
6	Read a file from the C2 server and save it to disk
7	Read a local file from a given path and send its content
9	Send the external IP address by making a request to myexternalip[.]com/raw
11	Terminate the socket connection
12	Send the current client configuration
13	Update the in-memory configuration and write it to disk
14	Dump the in-memory configuration to disk

EggStremeLoader and EggStremeReflectiveLoader (Delivery Mechanism)

This chapter details the primary mechanism used to deliver the main payload, EggStremeAgent, and consists of two separate but interconnected components.



EggStremeLoader binary is an advanced loader that is registered as a Windows service. Upon execution, the binary loads nine functions related to Windows service manipulation from advapi32.dll. The names of these functions are hardcoded within the binary and are decrypted at runtime using a hardcoded XOR key (0xFE). The binary dynamically resolves these functions using LoadLibrary() and GetProcAddress().

A clear distinction was found in the malware's hardcoded strings. Those related to file paths, service names, and decryption keys were XORed with the key 0xDD, while strings for function names were XORed with 0xFE.

The binary then reads a file at C:\Windows\en-US\ielowutil.exe.mui, which is composed of three parts:

1. A 4-byte length indicator for the EggStremeReflectiveLoader in big endian
2. The EggStremeReflectiveLoader payload
3. The EggStremeAgent payload

Each of these three parts is individually encrypted with RC4. During the attack, several encryption keys were observed: google, Google, Microsoft, and microsoft. The EggStremeLoader component is responsible for decrypting only the first two parts. It then attempts to inject the extracted EggStremeReflectiveLoader into winlogon.exe by creating a separate page using VirtualProtect() with PAGE_EXECUTE_READWRITE (execute) permissions and adjusting its own privileges to SeDebugPrivilege. This entire process runs in a loop every 10 minutes.

The EggStremeReflectiveLoader serves as an intermediary stage for the final payload. It has an export name of reflective.dll, strongly suggesting it is a reflective loader. A reflective loader is a piece of code that loads a DLL into a running process directly from memory rather than from a file on disk. While a standard Windows loader needs a file path to load a DLL, a reflective loader can work with the DLL's raw byte data. This method is a way to avoid detection, as it bypasses file-based security checks and leaves fewer traces on the system.

The loader opens C:\Windows\en-US\ielowutil.exe.mui, reads the last part (EggStremeAgent) from the file, and decrypts it using RC4. It then checks for the presence of C:\\ProgramData\\Microsoft\\Windows Defender\\Platform\\<>\\<>\\MsMpEng.exe (the Windows Defender service). If the file is not found, it targets explorer.exe for injection instead. The loader duplicates the token from the winlogon.exe process, grants itself SeDebugPrivilege, and uses CreateProcessWithToken() to spawn a new process (MsMpEng.exe or explorer.exe). It then injects the EggStremeAgent payload into this new, suspended process using VirtualAllocEx(), WriteProcessMemory(), and ResumeThread().

EggStremeAgent (Core Backdoor)

The EggStremeAgent is the final and most sophisticated implant responsible for establishing C2 with the attacker. It is a fileless payload, doesn’t touch the disk in a decrypted form, and is injected into memory by EggStremeReflectiveLoader using RC4 encryption with google as the key.

Before any malicious activity begins, the implant starts a new thread to monitor for the WTS_EVENT_LOGON event, which signals a user has logged in. The malware then waits to see if a child process explorer.exe has been spawned under the user's account. Once confirmed, the malware decrypts the EggStremeKeylogger (described later) at C:\Windows\en-US\splwow64.exe.mui using the RC4 key Microsoft and injects it into the user's explorer.exe process.

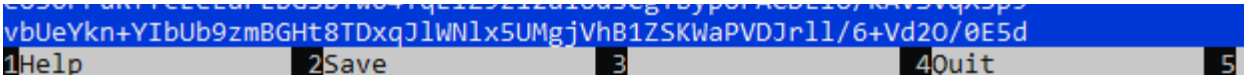
On its main thread, the backdoor decrypts its hardcoded configuration using an XOR key (0xFE). This configuration includes a file path to on-disk configuration (%LOCALAPPDATA%\Microsoft\Vault\Vault.dat), the initial C2 server (sealtribute[.]org), its port (443), and a certificate with a private key. If the on-disk file exists, it's decrypted with RC4 using its file path as the key, and its values override the default ones. If not, a unique ID is calculated based on the computer name and a new encrypted configuration file is created.

The configuration file contains the following encrypted fields:

- id - A unique ID derived from the computer name
- sl (sleep) - The interval in seconds that the agent waits between C2 checks
- rm (remote machine) – The IP address or domain of the C2 server
- rp (remote port) – The port number for C2 communication

- cacrt (CA certificate) – The public key certificate of the trusted root authority
- imcrt (implant certificate) - The certificate for the implant itself, used in mutual TLS authentication
- imkey (implant key) – The private key for the implant certificate


id:F69E63{sl:30{rm:fetraa.com{rp:443{cacrt:-----BEGIN CERTIFICATE-----
MIIDeTCCAmGgAwIBAgICB+MwDQYJKoZIhvcNAQELBQAwTTEJMAcGA1UEBhMAMQkw
BwYDVQQIEwAxCTAHBgNVBAcTADEJMAcGA1UECRMAMQkwBwYDVQQREwAxCTAHBgNV
BAoTADEJMAcGA1UECxMAMCAXDTE5MTEwNjA1MzIyMloYDzc0OTgwNzExMTA0NTIy
WjBNMQkwBwYDVQQGEwAxCTAHBgNVBAgTADEJMAcGA1UEBxMAMQkwBwYDVQQJEwAx
CTAHBgNVBBETADEJMAcGA1UEChMAMQkwBwYDVQQLEwAwggEiMA0GCSqGSIb3DQEB
AQUAA4IBDwAwggEKAoIBAQC/dB1NSWHIBc6gd9nl8cFiJP5kqd5VuTzxI3vj27RN
j5lc/16fHbRUx7m07GPPgPqIny5CJcoj3DthPtyGy4LEpRyiQSNmkoyxQy49GCRV
/YoS2blwO3Ya606SNCXCDCERIFvAk1kHW8F+mMgiE0r0mdjjTC3iTQD0K1DGCBz0
6WFAcrNJxj3hbrth1+YZBAheNgSvphVpjKUHcFLsG6oPTMs8zQpfjsOvsViFVMGa
ktSUhsfeFneS98G28SwPg3ExRWBmhK1Cg+rWJ5GbuGZYPBYSdcegsF52nU7jyNdh
uxKeN9LHJs+9Mz1SC6G2rMaFNfgtiWuQuqVAS96ZQAajAgMBAAGjYTBfMA4GA1Ud
DwEB/wQEAwICpDAdBgNVHSUEFjAUBggrBgEFBQcDAgYIKwYBBQUHAwEwDwYDVR0T
AQH/BAUwAwEB/zAdBgNVHQ4EFgQUUWVejpf8cmWxqqQmXZTi98rpyRMwDQYJKoZI
hvcNAQELBQADggEBAIe4rFhkZk7ZfbUT0zYps2pfXLL1s9QYu+XV6L/BJ6P8qEC+
2kAqCZOma3mj10+95B2JlTkVgZtblmfNeIty9y1DJgOs/VeKhVjSnxMpYo/lQUD6
idpk3Uo3s70MK7lrRhGl2zLCZAf8eiDrzPT/H2PBFzYLFT/+YDyxO5jf/N+vf9cy
dig4bIn/nLTWr/ZGLYIrBkQkYsMN8B2ighNnAz/CaXvHxw4xX0Thu4zO+cDPd1Yc
SiKWNINBdpPADOYgSq0mh8/SEUtjsFixRifP2HkLbcVgANky02tkvGJCPGHjQTHG
sT1b4tLlEOdGufEgWbgtvuSkX5moTj9qFKpMsuY=
-----END CERTIFICATE-----
{imcrt:-----BEGIN CERTIFICATE-----
MIIDmjCCAoKgAwIBAgICBnowDQYJKoZIhvcNAQELBQAwTTEJMAcGA1UEBhMAMQkw
BwYDVQQIEwAxCTAHBgNVBAcTADEJMAcGA1UECRMAMQkwBwYDVQQREwAxCTAHBgNV
BAoTADEJMAcGA1UECxMAMB4XDTIyMDkwOTEzMjYyMFoXDTM0MDExNzEzMTAxNlow
YjEJMAcGA1UEBhMAMQkwBwYDVQQIEwAxCTAHBgNVBAcTADEJMAcGA1UECRMAMQkw
BwYDVQQREwAxCTAHBgNVBAoTADEJMAcGA1UECxMAMRMwEQYDVQQDEwpmZXRyYWEu
Y29tMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAty7xPT4u73b+RxLt
arIjJBSeQWIA4wmdpXhEFGkr14Khju1aluej5y251xPsfI+X/c0rrX0xJAUw347D
05oOSrO5WBYHU+c/T43Prwb8ra9rLXNamhl3WST5p3YjaISsoku6XPTnsJNazNar
7zgmfkItApIDycQzye2HddU/x+GNs+FhX1jumBpKW1yYDiJeOqyplUHw+pCmCiBs
tx1dWhqLh38JF++RFejfaK0sIyj+mPpNIay73ITs2eT3eGQWQz1/kbO4z7WiJ0Gs
bCQER8IB5cgRsUVHKEqaZITor7krgaHPcc+umPGBYnr0VA4jz+jEb7Dd8gQ+CNwU
/1WzSwIDAQABo28wbTAOBgNVHQ8BAf8EBAMCB4AwEwYDVR0lBAwwCgYIKwYBBQUH
AwIwDgYDVR0OBAcEBQECAwQGMB8GA1UdIwQYMBaAFFFlXo6X/HJlsaqkJl2U4vfK
6ckTMBUGA1UdEQQOMAyCCmZldHJhYS5jb20wDQYJKoZIhvcNAQELBQADggEBAFuD
8K7ANMeNyNpeLdy/fxd7rCFOymXOxMfkmZAbUsvKLhqkw4yq+2ZfGfgGlJ8nnN46
+wjo/8pFVSM/pJKVk2YFjeD5HbPRIYVaYWKiEAAKO3YjANS+8ETdBR6+WK/qA1Hs
tbMC6+WeOWvzB8tiz8URpohj9DtIT9BpRnExCoOAqNXaXYq7RAPmz18gTDToyCK+
YFuGGVVbhcb9F3SWc/Lys8hgd3Eml1RvwjN3bj4m247apRvtkzdiciEPkerykyBY
mQUIyfRzoL68lvd/bJos/VLogeNKGtXV8qcK1C+Uwu8SO/EDkdij118cyocJdD1T
gg7jFaBpnXCtOdxWKWA=
-----END CERTIFICATE-----{imkey:-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEAty7xPT4u73b+RxLtarIjJBSeQWIA4wmdpXhEFGkr14Khju1a
luej5y251xPsfI+X/c0rrX0xJAUw347D05oOSrO5WBYHU+c/T43Prwb8ra9rLXNa
mhl3WST5p3YjaISsoku6XPTnsJNazNar7zgmfkItApIDycQzye2HddU/x+GNs+Fh
X1jumBpKW1yYDiJeOqyplUHw+pCmCiBstx1dWhqLh38JF++RFejfaK0sIyj+mPpN
Iay73ITs2eT3eGQWQz1/kbO4z7WiJ0GsbCQER8IB5cgRsUVHKEqaZITor7krgaHP
cc+umPGBYnr0VA4jz+jEb7Dd8gQ+CNwU/1WzSwIDAQABAoIBACMic87FUIdJkXfc
X4EpbagL9FaiD318aKpGKPipO2l4erhJOijbgQpkMNLZ1XAGsqZRZeE+s+AuxrL6
VQEO+VLGqfQFNZ6YKgJ+Nqlhc7F0heAP0f7gh42nk+LUu/HeEKaIU9RkfNPXT4Ee
Y5fcczWgR7XIf8WBRlWio2qsXjFd9EgQ2s/FIIg9Dke+LK1ODAgQsZ6hUXlVtkB0
9jLaMGCB4K/H9oewA6BhrPADtz/dPljiGI09tDpRmFJ5N3FLG73anS1wdcGvAQiB
TrVRhRvTOJdhyB9pbkme7DrOgaaGSV827RdS3T0493/vihXD+vQzQvDHpepf/0TU
WwsZJ0ECgYEAxpmthW4ANLdPZiWGVjznSOAfXqGAxORqL1QyU+KNYR50ba4llgk/
XzY1baTlUryevJcNCGdeBGtFjYhZ6BMKtti/0W8exp33MQRm/pAkC+0HbsK2hwZx
tMf/f4Wb6AutHmtTTtBghhnjZlbU0UnpPKlzYnE4zg0I7n6YbdV5h1ECgYEA7CCT
CB/IPm7AaawnYvFk7bgjmZQY0zyokeAahat0wFfKGWqLn/N34Gbng31NGSeO+nrP
YfydIuSFvneDxVW80aY8xfZQSdiwQKzY0k063BmNEZjl9/en6c5d94Vdz/H37oH0



An example of the configuration file

Using the gRPC C++ library, the malware establishes a secure channel via mutual TLS (mTLS) to communicate with its C2 server. gRPC (Google Remote Procedure Call) is an open-source framework for high-performance communication between services. In this malware's implementation, a total of 58 commands are available, identified by numerical IDs, ranging from 0 to 66, with a few numbers missing.

Command ID	Description
0	**Fingerprinting:** Extracts system details including hostname, username, OS type (via WMIC), MAC address, internal and external IP addresses, sleep time, and a list of installed antivirus products (via WMIC)
17	**Dump Config:** Encrypts the current in-memory configuration using RC4 with the file path as the key and saves it to disk. It also triggers a mini-fingerprint (Command 38)
18	**Send C2 Info:** Sends the remote machine and port (rm & rp) fields from the config to the C2 server in the format <rm>:<rp>
37	**Update Config:** Receives a new remote machine and port from the C2, updates the in-memory configuration, and dumps it to disk
38	**Basic Fingerprinting:** Updates the in-memory sleep time and sends a smaller fingerprint containing the agent ID, hostname, LAN IP, and sleep time to the C2

File and Directory Operations

Command ID	Description
7	**Enumerate Files:** Sends the name, size, type (file/directory), and last write time of all files and directories within a given path
8	**Change Directory:** Changes the current working directory, with support for environment variables
9	**Get Current Directory:** Returns the current working directory
10	**Copy File:** Copies a file
11	**Delete File:** Deletes a file
12	**Create Directory:** Creates a directory

13	**Delete Directory:** Deletes a directory
14	**Screenshot:** Takes a screenshot of the entire screen, saves it as a bitmap, and sends it buffered to the C2 server
26-28	**Write to File:** A three-part command to write data to a file. Command 26 opens a file handle, 27 writes chunks of binary data, and 28 closes the handle
29-30	**Read from File:** A two-part command to read a file. Command 29 opens a handle, calculates and reports the file size. Command 30 uploads the file in 0xA000-byte chunks
57	**Move File:** Moves a file or directory from one location to another
59	**Read Small File:** Reads a binary file and sends it to the server if its size is less than 0x5000 bytes. Otherwise, it reports an error “file over size” or “file is empty”
62	**Timestomping:** Copies basic file metadata (e.g., creation, modification, and access times) from one file to another to evade detection

Local Resource Enumeration

Command ID	Description
15	**List Startup Commands:** Uses a WMIC query SELECT * FROM Win32_StartupCommand to list and send details on all startup commands
16	**List Services:** Uses a WMIC query SELECT * FROM Win32_Service to list and send details on all services, including name, caption, path, and PID if started
19	**List Processes:** Lists all running processes, sending back the PID, parent PID, executable path, username, and architecture
21	**Kill Process:** Kills a specified process by its PID
22	**List Connections:** Lists all TCP connections, including source and destination IPs, state, and the process path that initiated the connection
23	**Network Properties:** Lists network properties in a format similar to ipconfig /all using WinAPI functions
24	**ARP Entries:** Enumerates the Address Resolution Protocol entries
25	**List Drivers:** Lists all drivers, including their name, available and total space
41	**Enumerate SQLite Temporary Files:** Loops over all file objects with a Cookies-journal suffix and sends the file path and the process image of any processes that have a handle opened to them

47	**Get Uptime:** Retrieves the current system uptime
51	**List Registry Keys:** Enumerates all subkeys and values from a registry path received from the C2 server and sends back the collected data, including name, value, and data type
52	**Set Registry Value:** Calls RegSetValueExW() to update or create a specified value in the registry
53	**Delete Registry Value:** Deletes a specified registry value from a given subkey
58	**User Sessions:** Retrieves all user sessions from the machine using WTSEnumerateSessionsA()

Network and Lateral Movement Capabilities

Command ID	Description
40	**RPC Scan:** Connects to a given IP on port 135 (RPC) and sends an RPC bind request with NTLMSSP authentication. The response can reveal the target system's Windows version and hostname
42	**Remote Process:** Remotely connects to a server and creates a new process using WMIC Win32_Process Create() method
43	**IPC$ Reauthentication:** Uses a given IP and credentials to reauthenticate to a remote machine's IPC$ share
44	**Enumerate Network Resources:** Enumerates all network resources on the current machine to discover accessible shares or connected devices
45	**Close IPC$:** Closes an IPC$ share connection
49	**Enumerate Services:** Enumerates all services that are in the SERVICES_ACTIVE_DATABASE on the local or a remote machine
50	**Check/Start Service:** Checks the status of a service on a local or remote machine and attempts to start it if it is stopped
54	**Set Service Type:** Sets the start type for a service on a local or remote machine.
55	**Ping:** Sends ICMP Echo Requests to a single IP or a range of IPs to check connectivity
56	**Port Scan:** Checks if a port is open on a given IP or a range of IPs
60	**Task Scheduler:** Uses COM objects to interact with the Task Scheduler on local or remote machines. Subcommands support listing (0), deleting (1), creating (2), and executing (3) scheduled tasks

63	**Create Remote Service:** Creates and configures a new service on a remote system for persistence by establishing the registry key SYSTEM\\CurrentControlSet\\Services\\%s\\Parameters and assigning the ServiceDll value to a specified DLL path. Additionally, a new service group for svchost.exe is registered by appending the suffix svc to the service name. The service is then registered using OpenSCManagerA() to execute %SystemRoot%\\system32\\svchost.exe -k "<name>svc" with AUTO_START flag

Advanced Execution and Privilege Escalation

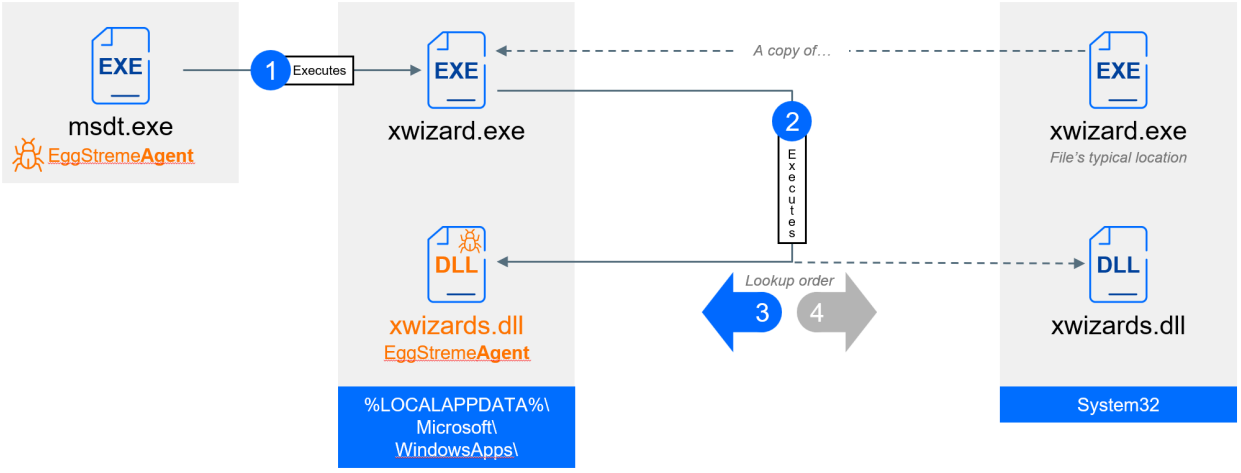
Command ID	Description
20	**Launch Process:** Receives a PID and application path. If the PID is 0, it launches the app with CreateProcess(). If a PID is given, it impersonates the process token to launch the app with CreateProcessWithToken()
32	**Retrieve and Execute Shellcode:** Expects a URL from the C2, retrieves a shellcode via HTTP GET, and executes it on a new thread
33-35	**Interactive Shell:** Implements an interactive shell using cmd.exe with anonymous pipes. Command 33 handles the initialization and process spawning, launching the shell either via CreateProcess() or CreateProcessWithToken() (the token obtained through DuplicateHandle() from a process specified by the C2 server). It also attempts to enable SeDebugPrivilege for elevated interaction with other processes and spawns a separate thread to send all data from the shell's output pipe back to the C2 server. Command 34 sends input, and Command 35 terminates the shell
46	**Terminate Process:** Gracefully closes pipes and terminates a process initiated by the interactive shell command
65	**LSASS Injection:** A complex command likely used to inject code into the LSASS process. It has three subcommands: 0 to check if a process is active; 1 to create pipes, adjust privileges, locate LSASS, duplicate handles, and inject a PE payload into a suspended svchost.exe instance, and then resume its execution; and 2 to forward data to the svchost.exe process via pipes
64 & 67	**Proxy:** Two very similar commands that function as proxy mechanisms. Subcommand 0 initiates a TCP connection and spawns a thread to forward data. Subcommand 1 forwards data from the C2 to the target
48	**Exfiltration:** Compresses a file or all files in a directory into a GZIP archive using Zlib 1.2.13
66	**File Ownership:** Alters file ownership on the target system to BUILTIN\Administrators using Windows API functions SetEntriesInAclA() and SetNamedSecurityInfoA() to bypass access

	restrictions and interfere with detection

EggStremeWizard (Auxiliary Backdoor)

Following the successful execution of EggStremeAgent, the primary command and control implant, the attacker deployed EggStremeWizard on several machines. This component is a secondary lightweight backdoor; its primary role is to ensure redundancy in the event the main agent is detected and removed.

The forensic evidence indicates that a copy of the legitimate xwizard.exe was present in a user-writable directory (%LOCALAPPDATA%\Microsoft\WindowsApps\), a different location than its typical home in %SYSTEM32%. The malicious xwizards.dll was also placed in this same directory. The EggStremeReflectiveLoader, residing in memory within winlogon.exe, initiated the execution of the EggStremeAgent within the msdt.exe binary. The EggStremeAgent then spawned cmd.exe, which in turn launched the relocated xwizard.exe. This last process was used to sideload the malicious library from the same location.



When loaded, this binary first attempts to read a file located at C:\Users\Public\Downloads\ntuser.dat, which contains a sleep interval value in minutes. It then tries to read a second file, C:\Users\Public\Downloads\ntusers.dat, which is encrypted using AES in ECB mode with the key +JBHXU4X*%^Y&(DP and has a maximum size of 0x114 bytes. Note the subtle difference in the filenames (ntuser.dat vs ntusers.dat). The second file is expected to contain a space-separated list of IP addresses and ports. If it is not found, the backdoor defaults to initiating a connection to sinhluc[.]net.

Once the initial setup is complete, the sample initiates communication with the C2 server by sending the plaintext value 0x3E8FB806. It then captures the output of the hostname command, encrypts it with AES in ECB mode, and transmits it. After this handshake, the backdoor begins a handling routine to receive further instructions, with all subsequent communication being encrypted using AES.

Command ID	**Descriptions**
First byte is 1	This command has two subcommands and a default action

sleep	Changes the sleep interval value in the ntuser.dat file
server	Updates the internal list of C2 servers and sends a confirmation message to the active C2: "Tunnel Change Success:\r\nServer : %s:%s\r\n" The Server field is not limited to a single entry
Default	If no subcommands match, the input is treated as a Windows command, executed via cmd.exe /c, and the output is returned to the C2
First byte is 2	Interprets the command as an instruction to download, decrypt, and save a binary file to disk
First byte is 5	Reads a local file from disk, encrypts it, and uploads its contents to the C2 server
First byte is 8	Reinitializes the entire list of C2 servers

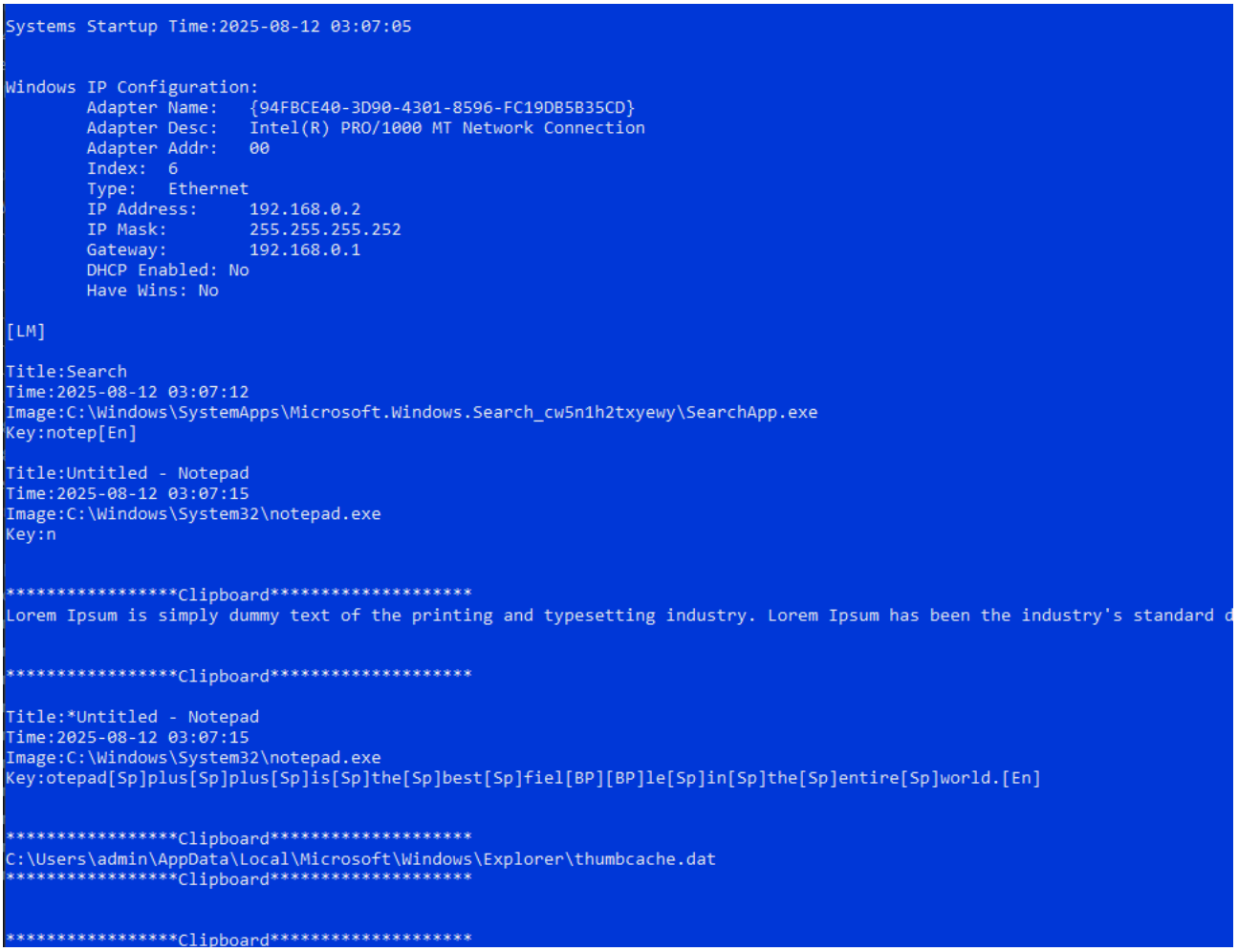
EggStremeKeylogger (Surveillance Module)

As described in the previous EggStremeAgent section, EggStremeKeylogger component is a malicious library that resides at a familiar location on the victim's machine: C:\Windows\en-US\splwow64.exe.mui. It remains encrypted on disk until the EggStremeAgent is ready to use it, at which point it is decrypted in memory using the static RC4 key Microsoft and injected into the user's explorer.exe process.

The EggStremeKeylogger is not a standalone executable; it is a DLL library with a unique loading mechanism. Instead of the standard DllMain() function triggering its malicious behavior, the EggStremeAgent calls an exported function within the keylogger's binary: RegisterWaitChainCOMCallback(). This function serves as a custom reflective loader to load the keylogger's own binary from memory, effectively injecting it back into its own process. This process then triggers the DllMain() function, which acts as the malware's true entry point, allowing the keylogger to initiate its malicious activities.

Once executed, it spawns a hidden window and creates a log file at %LOCALAPPDATA%\Microsoft\Windows\Explorer\thumbcache.dat to store captured data. Each log entry is encrypted in real time using the RC4 algorithm with the key usa1!the8*best9#, and entries are separated by the byte sequence 0xA0D0A0D.

At startup, it first records the system’s startup time and fingerprints the Windows network configuration, enumerating adapter names, descriptions, IP addresses, subnet masks, and gateways. Once initialized, it continuously intercepts keystrokes via GetRawInputData() and MapVirtualKeyA(), logging each sequence alongside the active window title, the current time, and the full process image path. Clipboard monitoring is also persistent, capturing plaintext (GetClipboardData()) and file paths when files are copied (DragQueryFileA()). Additionally, a dedicated thread monitors network adapter changes in real time, automatically collecting and dumping updated configuration details



EggStremeKeylogger example (emulated)

This combination of keystroke logging, clipboard monitoring, file capture, and network fingerprinting provides attackers with comprehensive surveillance over both user activity and system configuration.

Stowaway (The Proxy Tool)

The Stowaway proxy is used by the threat actor to establish an internal network foothold. It operates by creating an exposed proxy on the compromised host, enabling attackers to route traffic and execute commands on other systems within the network.

Forensic analysis indicates that an existing backdoor was used to write the Stowaway binary to disk at %LOCALAPPDATA%\microsoft\windows\burn\burn\burn.conf. The file is a compiled Go binary that functions as a DLL, and its export table contains numerous symbols, including CoStartOutlookExpressW() (typically exported by msoe.dll), CorGetSvc() (mscorsvc.dll), and ServiceMain() (a generic function for services). These symbols indicate the tool was designed for potential sideloading or execution as a Windows service.

The tool's initialization function is custom and hard-coded with specific parameters: a secret (d@rkn3ss) and a listening port (8531). This initialization allows the attacker to connect to the proxy and authenticate with the known secret.

```
  v0 = (utils_AgentOptions *)runtime_gcWriteBarrier(&Stowaway_agent_Args);
else
  Stowaway_agent_Args = (__int64)p_utils_AgentOptions;
v0->Secret.len = 8;
if ( runtime_writeBarrier )
  runtime_gcWriteBarrier(v0);
else
  v0->Secret.ptr = "d@rkn3ss";
v22.tab = (void *)&go_itab__flag_stringValue_flag_Value;
v22.data = v0;
v34.ptr = "s";
v34.len = 1;
flag__ptr_FlagSet_Var(flag_CommandLine, v22, v34, (string)0LL);
v1 = Stowaway_agent_Args;
*(_QWORD *)(Stowaway_agent_Args + 24) = 4;
v2 = (void *)(v1 + 16);
if ( runtime_writeBarrier )
  runtime_gcWriteBarrier(v2);
else
  *(_QWORD *)(v1 + 16) = "8531: p=<%s>";
v23.tab = (void *)&go_itab__flag_stringValue_flag_Value;
v23.data = v2;
v35.ptr = "l";
v35.len = 1;
flag__ptr_FlagSet_Var(flag_CommandLine, v23, v35, (string)0LL);
```

A plausible explanation for the attackers' use of this tool is the requirement to execute remote commands without deploying a full-featured agent on every target. The Go-based binary provides a small footprint and can be executed to establish a temporary proxy. This proxy is then used in conjunction with a framework like Impacket.

On their own machine, the attackers can configure proxychains or a similar tool to route their Impacket commands through the exposed Stowaway proxy. This allows them to effectively target any system on networks accessible by the infected host. The primary threat implication for a security practitioner is that the proxy bypasses network-level segmentation and firewall rules, as the commands originate from within the trusted network segment.

Conclusion and Recommendations

The EggStreme malware family is a highly sophisticated and multi-component threat designed to achieve persistent access, lateral movement, and data exfiltration. The threat actor demonstrates an advanced understanding of modern defensive techniques by employing a variety of tactics to evade detection.

This modular, fileless, and living-off-the-land (LOL) approach highlights a significant shift in adversary tradecraft. The threat is not a collection of individual executables but a dynamic, multi-stage operation that leverages legitimate tools and system behaviors to remain undetected.

To effectively counter threats like EggStreme, security practitioners must adopt a defense-in-depth strategy.

- Proactively Limit LOLBins: Proactively reduce your attack surface by limiting the use of legitimate but high-risk binaries. Implement Proactive Hardening and Attack Surface Reduction (PHASR) to restrict built-in tools like wmic.exe and other LOLBin attacks.
- Adopt Detection and Response Capabilities: A robust security platform like Bitdefender GravityZone with strong EDR/XDR capabilities is essential. These platforms are essential for correlating events across

multiple endpoints to identify complex attack chains and detect behavioral anomalies that bypass prevention layers. This is critical for catching an event like msdt.exe spawning cmd.exe or xwizard.exe running from an unusual directory. You can learn more about EDR/XDR technology on the Bitdefender TechZone website.

- Consider Managed Detection and Response (MDR) for Operational Gaps: For organizations without a dedicated Security Operations Center (SOC) team or operating with a lean security staff, adopting Managed Detection and Response (MDR) services offers an effective solution. MDR effectively acts as an extension of an in-house team, providing 24/7 expert threat hunting, rapid incident response, and continuous monitoring.

By focusing on these areas, organizations can build a more resilient security posture, capable of detecting and responding to even the most covert and persistent adversaries.

IOCs and How to Follow Our Research

For our OEM partners and integrations, access to our threat intelligence data is primarily provided programmatically. We also offer a user interface, **IntelliZone Portal**. This is where partners get more ways to interact with our data, like an operational dashboard of threats targeting their industry. A full breakdown of this research can be found on the platform under ThreatID BDtqkhbtsw :

https://intellizone.bitdefender.com/en/threat-search/threats/BDtqkhbtsw

Beyond our core TI platform, we're also launching three new ways for you to stay current with our research.

Public IOCs on GitHub

We are now hosting all Indicators of Compromise (IOCs) from this and all future research on a public GitHub repository. This will improve accessibility and collaboration for the entire security community. https://github.com/bitdefender/malware-ioc/blob/master/2025_09_10-eggstreme-iocs.csv

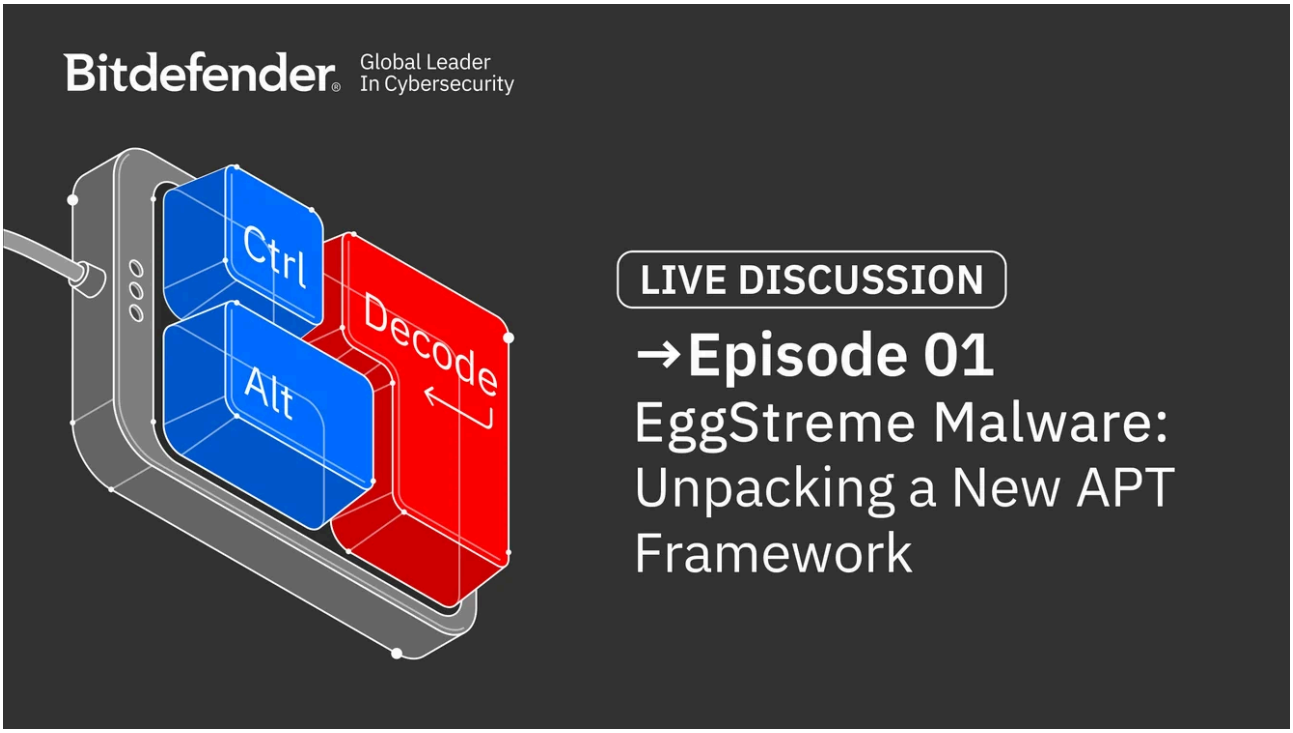
Newsletter: Threat Intel DECODED

We are introducing our LinkedIn newsletter, Threat Intel DECODED, designed to provide you with exclusive threat intelligence, original research, and actionable advisories directly from Bitdefender Labs and MDR.

Subscribe to our newsletter and let us know what you think!
https://www.linkedin.com/newsletters/7371216616015036416/?displayConfirmation=true

Live Discussion: Ctrl-Alt-DECODE

Watch our new discussion series, Ctrl-Alt-DECODE, where we discussed the EggStreme research in-depth and answered live questions. This was our first episode, so thanks for tuning in and helping us get this new series started!



Source: https://businessinsights.bitdefender.com/eggstreme-fileless-malware-cyberattack-apac