Moshen Dragon's Triad-and-Error Approach | Abusing Security Software to Sideload PlugX and ShadowPad

Joey Chen

By Joey Chen and Amitai Ben Shushan Ehrlich

Executive Summary

- SentinelLabs researchers are tracking the activity of a Chinese-aligned cyberespionage threat actor operating in Central-Asia, dubbed 'Moshen Dragon'.
- As the threat actor faced difficulties loading their malware against the SentinelOne agent, we observed an unusual approach of trial-and-error abuse of traditional antivirus products to attempt to sideload malicious DLLs.
- Moshen Dragon deployed five different malware triads in an attempt to use DLL search order hijacking to sideload ShadowPad and PlugX variants.
- Moshen Dragon deploys a variety of additional tools, including an LSA notification package and a passive backdoor known as GUNTERS.

Overview

SentinelLabs recently uncovered a cluster of activity targeting the telecommunication sector in Central Asia, utilizing tools and TTPs commonly associated with Chinese APT actors. The threat actor systematically utilized software distributed by security vendors to sideload ShadowPad and PlugX variants. Some of the activity partially overlaps with threat groups tracked by other vendors as RedFoxtrot and Nomad Panda. We track this cluster of activity as 'Moshen Dragon'.

Usually, good detection has an inverse relationship with visibility of a threat actor's TTPs. When part of an infection chain gets detected, it usually means that we don't get to see what the threat actor intended to deploy or ultimately do. In an unexpected twist, our detection capabilities uncovered an unusual TTP as Moshen Dragon attempted to repeatedly bypass that detection.

Every time the intended payload was blocked, we were able to witness the actor's reliance on a wide variety of legitimate software leveraged to sideload ShadowPad and PlugX variants. Many of these hijacked programs belong to security vendors, including Symantec, TrendMicro, BitDefender, McAfee and Kaspersky.

Rather than criticize any of these products for their abuse by an insistent threat actor, we remind readers that this attack vector reflects an age-old design flaw in the Windows Operating System that allows DLL search order hijacking. Tracking of additional Moshen Dragon loading mechanisms and hijacked software surfaced more payloads uploaded to VirusTotal, some of which were recently published under the name 'Talisman'.

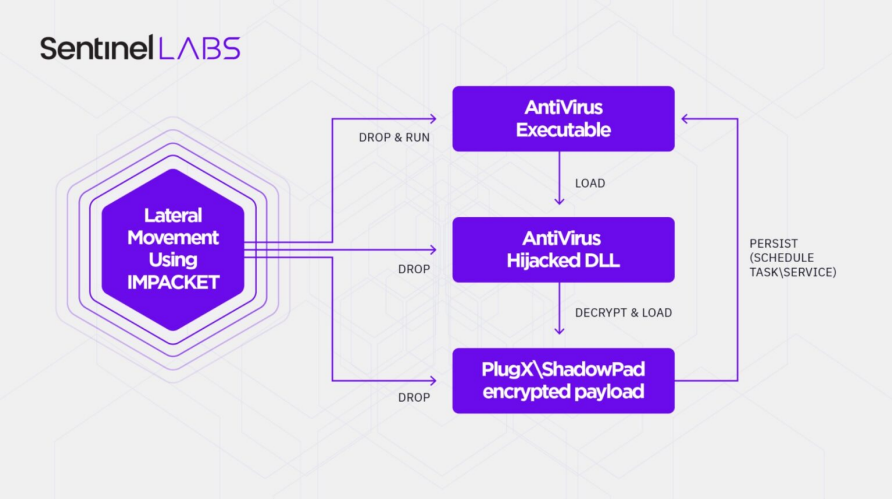
In addition to ShadowPad and the PlugX Talisman variant, the Moshen Dragon deployed a variety of other tools, including an LSA notification package to harvest credentials and a passive loader dubbed GUNTERS by Avast. Despite all of this visibility, we are still unable to determine their main infection vector. Their concerted efforts include the use of known hacking tools, red team scripts, and on-keyboard attempts at lateral movement and data exfiltration.

We will focus on the actor's insistent abuse of different AV products to load malicious payloads in an attempt to 'bruteforce' infection chains that would go undetected by traditional SOC and MDR solutions.

Hijacking Security Products

Moshen Dragon actors systematically abused security software to perform DLL search order hijacking. The hijacked DLL is in turn used to decrypt and load the final payload, stored in a third file residing in the same folder. This combination is recognized as a sideloading triad, a technique commonly associated with Lucky Mouse.

The way the payloads were deployed, as well as other actions within target networks, suggest the threat actor uses IMPACKET for lateral movement. Upon execution, some of the payloads will achieve persistence by either creating a scheduled task or a service.



Execution flow of hijacked software as carried out by Moshen Dragon

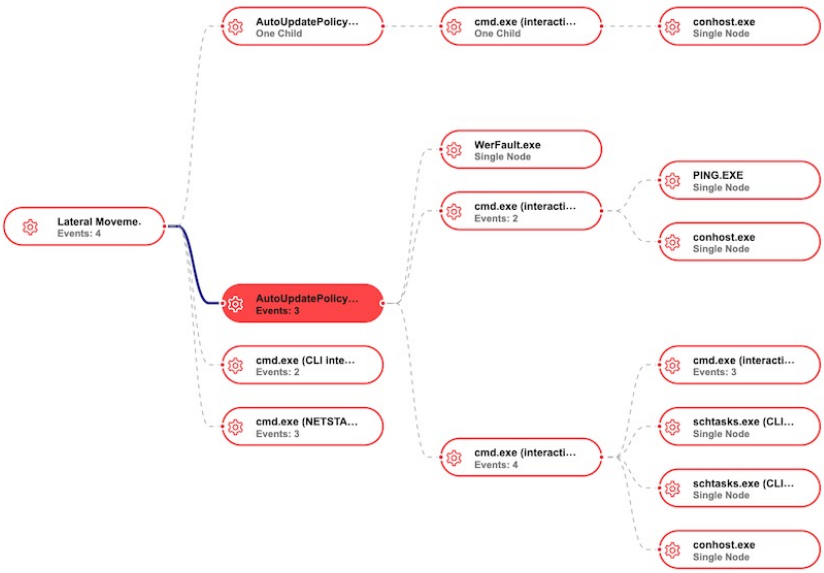
As major portions of the Moshen Dragon activity were identified and blocked, the threat actor consistently deployed new malware, using five different security products to sideload PlugX and ShadowPad variants.

A summary of the hijacked software is presented in the table below:

Product	Path
Symantec SNAC	C:\Windows\AppPatch, C:\ProgramData\SymantecSNAC, C:\ProgramData\Symantec\SNAC, C:\Windows\Temp
TrendMicro Platinum Watch Dog	C:\Windows\ AppPatch
BitDefender SSL Proxy Tool	C:\ProgramData\Microsoft\Windows, C:\ProgramData\Microsoft\Windows\LfSvc, C:\ProgramData\Microsoft\Windows\WinMSIPC, C:\ProgramData\Microsoft\Windows\ClipSVC, C:\ProgramData\Microsoft\Wlansvc, C:\ProgramData\Microsoft\Windows\Ringtones
McAfee Agent	C:\ProgramData\McAfee, C:\ProgramData\Microsoft\WwanSvc, C:\ProgramData\Microsoft\WinMSIPC
Kaspersky Anti-Virus Launcher	C:\ProgramData\Microsoft\XboxLive, C:\programdata\GoogleUpdate

Lateral Movement Using Impacket

Impacket is a collection of Python classes for working with network protocols, commonly utilized by threat actors for lateral movement. One of the favorite tools in the Impacket arsenal is `wmiexec`, which enables remote code execution via WMI. An effective way to identify `wmiexec` execution is searching for the unique command line pattern it creates. Moshen Dragon activities are rife with this pattern.



Lateral Movement utilizing Impacket as identified by the SentinelOne Agent

LSA Notification Package – SecureFilter

When on domain controllers, Moshen Dragon dropped a password filter and loaded it into the lsass process via LSA Notification packages. Impacket is used in the following manner:

```
cmd.exe /Q /c REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v "Notification Packages" /t REG_MULTI_SZ /d "rassfm\0scecli\0SecureFilter" /f 1> \\127.0.0.1\ADMIN$\__[REDACTED] 2>&1
```

The DLL deployed is dropped in the path `C:\Windows\System32\SecureFilter.dll` in order to enable loading using the Notification Package feature. The DLL seems to rely on an open source project named DLLPasswordFilterImplant, effectively writing changed user passwords to the file `C:\Windows\Temp\Filter.log`.

```
bool __fastcall PasswordFilter(__int64 format, __int64 a2, __int64 arg)
{
  HANDLE hFile; // rdi
  int bool_1; // eax
  DWORD NumberOfBytesWritten; // [rsp+60h] [rbp+18h] BYREF

  NumberOfBytesWritten = 0;
  hFile = CreateFileW(L"c:\\windows\\temp\\Filter.log", 4u, 0, 0i64, 4u, 0x80u, 0i64);
  if ( hFile != (HANDLE)-1i64 )
  {
    memset(str, 0, 0x800ui64);
    bool_1 = find(str, L"%wZ :: %wZ\r\n", format, arg);// int vswprintf( wchar_t* ws, size_t len, const wchar_t* format, va_list arg )
    WriteFile(hFile, str, 2 * bool_1, &NumberOfBytesWritten, 0i64);
    CloseHandle(hFile);
  }
  return wcsstr(*(const wchar_t **)(arg + 8), L"no") == 0i64;
}
```

Snippet from SecureFilter.dll

GUNTERS – A Passive Loader

During our analysis of Moshen Dragon's activities, we came across a passive loader previously discussed by Avast as 'GUNTERS'. This backdoor appears to be highly targeted as it performs checks to verify that it is executed on the right machine.

Before execution, the malware calculates the hash of the machine hostname and compares it to a hardcoded value, suggesting that the threat actor generates a different DLL for each target machine.

The loader utilized `WinDivert` to intercept incoming traffic, searching for a magic string to initiate a decrypting process utilizing a custom protocol. Following the decryption process, the malware attempts to load a PE file with an exported function named `SetNtApiFunctions`, which it calls to launch the payload.

```
v1 = 0;
if ( qword_180033140 )
  return 1i64;
strcpy(v5, "NTInit");
strcpy(v8, "NTAccept");
strcpy(v11, "NTAcceptPWD");
strcpy(v6, "NTSend");
strcpy(v9, "NTReceive");
strcpy(v10, "NtIsClosed");
strcpy(v7, "NTClose");
strcpy(v12, "NTGetSrcAddr");
strcpy(v13, "NTGetDscAddr");
strcpy(v14, "NTGetPwdPacket");
qword_180033148 = sub_1800022A0(a1);
if ( qword_180033148 )
{
  qword_180033140 = j__malloc_base(0x50ui64);
  v3 = qword_180033140;
  memset(qword_180033140, 0, 0x50ui64);
  v4 = qword_180033148;
  *v3 = sub_180001C80(qword_180033148, v5);
  v3[1] = sub_180001C80(v4, v8);
  v3[2] = sub_180001C80(v4, v11);
  v3[3] = sub_180001C80(v4, v6);
  v3[4] = sub_180001C80(v4, v9);
```

Exported functions of an internal GUNTERS resource utilized in the loading process

A thorough analysis of the custom protocol and loading mechanism is available here.

Additional Payloads

SentinelLabs came across additional related artifacts overlapping with this threat cluster. It's possible that some of those were utilized by Moshen Dragon or a related actor.

File name	SHA1	C&C
SNAC.log	e9e8c2e720f5179ff1c0ac30ce017224ac0b2f1b	freewula.strangled.net szuunet.strangled.net
SNAC.log	b6c6c292cbd35298a5f055448177bcfd5d0b23bf	final.staticd.dynamic-dns.net
SNAC.log	2294ecbbb065c517bd0e01f3f01aabd0a0402f5a	dhsg123.jkub.com
bdch.tmp	7021a62b68751b7a3a2984b2996139aca8d19fec	greenhugeman.dns04.com

After analyzing these payloads, we found them to be additional PlugX and ShadowPad variants. `SNAC.log` payloads have been identified by other researchers as Talisman, which is known to be another variant of PlugX. In addition, the `bdch.tmp` payload was produced by shellcode with a structure similar to ShadowPad malware but without the initial code obfuscation and decryption logic typically seen in ShadowPad.

Conclusion

PlugX and Shadowpad have a well-established history of use among Chinese-speaking threat actors primarily for espionage activity. Those tools have flexible, modular functionality and are compiled via shellcode to easily bypass traditional endpoint protection products.

Here we focused on Moshen Dragon TTPs observed during an unusual engagement that forced the threat actor to conduct multiple phases of trial-and-error to attempt to deploy their malware. Once the attackers have established a foothold in an organization, they proceed with lateral movement by leveraging Impacket within the network, placing a passive backdoor into the victim environment, harvesting as many credentials as possible to insure unlimited access, and focusing on data exfiltration.

SentinelLabs continue to monitor Moshen Dragon activity as it unfolds.

Indicators of Compromise

Hijacked DLLs

- ef3e558ecb313a74eeafca3f99b7d4e038e11516
- 3c6a51961aa328ba507796153234309a5e83bee3
- fae572ad1beab78e293f756fd53cf71963fdb1bd
- 308ed56dc1fbc98b574f937d4b005190c878416f
- 55e89f458b5f5642300dd7c50b444232e37c3fa7

Payloads

- e9e8c2e720f5179ff1c0ac30ce017224ac0b2f1b
- b6c6c292cbd35298a5f055448177bcfd5d0b23bf
- 2294ecbbb065c517bd0e01f3f01aabd0a0402f5a
- 7021a62b68751b7a3a2984b2996139aca8d19fec

Password Filter

- c4f1177f68676b770934b142f9c3e2c4eff7f164

GUNTERS

- bb68816f324f2ac4f0d4756b66af67d01c8b6e4e
- 4025e14a7f8928753ba06ad155944624069497dc
- f5b8ab4a7d9c723c2b3b842b49f66da2e1697ce0

Infrastructure

- freewula.strangled[.]net
- szuunet.strangled[.]net
- final.staticd.dynamic-dns[.]net
- dhsg123.jkub[.]com
- greenhugeman.dns04[.]com
- gfsg.chickenkiller[.]com
- pic.farisrezky[.]com