Lancefly: Group Uses Custom Backdoor to Target Orgs in Government, Aviation, Other Sectors

By About the Author

Archived: 2026-04-02 10:53:25 UTC

The Lancefly advanced persistent threat (APT) group is using a custom-written backdoor in attacks targeting organizations in South and Southeast Asia, in activity that has been ongoing for several years.

Lancefly may have some links to previously known groups, but these are low confidence, which led researchers at Symantec, by Broadcom Software, to classify this activity under a new group name.

Lancefly's custom malware, which we have dubbed Merdoor, is a powerful backdoor that appears to have existed since 2018. Symantec researchers observed it being used in some activity in 2020 and 2021, as well as this more recent campaign, which continued into the first quarter of 2023. The motivation behind both these campaigns is believed to be intelligence gathering.

The backdoor is used very selectively, appearing on just a handful of networks and a small number of machines over the years, with its use appearing to be highly targeted. The attackers in this campaign also have access to an updated version of the ZXShell rootkit.

The targets in this most recent activity, which began in mid-2022 and continued into 2023, are based in South and Southeast Asia, in sectors including government, aviation, education, and telecoms. Symantec researchers previously saw the Merdoor backdoor used in activity that targeted victims in the same geographies in the government, communications, and technology sectors in 2020 into 2021. Like this recent activity, that activity also appeared to be highly targeted, with only a small number of machines infected.

Merdoor Backdoor

Merdoor is a fully-featured backdoor that appears to have been in existence since 2018.

The backdoor contains the following functionality:

- Installing itself as a service
- Keylogging
- A variety of methods to communicate with its command-and-control (C&C) server (HTTP, HTTPS, DNS, UDP, TCP)
- Ability to listen on a local port for commands

Instances of the Merdoor backdoor are usually identical with the exception of embedded and encrypted configuration, which determines:

- C&C communication method

- Service details
- Installation directory

Typically, the backdoor is injected into the legitimate processes perfhost.exe or svchost.exe.

The Merdoor dropper is a self-extracting RAR (SFX) that contains three files:

- A legitimate and signed binary vulnerable to DLL search-order hijacking
- A malicious loader (Merdoor loader)
- An encrypted file (.pak) containing final payload (Merdoor backdoor)

When opened, the dropper extracts embedded files and executes a legitimate binary in order to load the Merdoor loader.

Merdoor dropper variants have been found that abuse older versions of five different legitimate applications for the purpose of DLL sideloading:

Attack Chain

Evidence from Lancefly's earlier campaign that began in 2020 suggested that in that instance the group may have used a phishing email with a lure based on the 37th ASEAN Summit as an initial infection vector.

In this more recent activity, the initial infection vector was not entirely clear. We saw some indications of what the initial infection vector may have been in two victims, though this was not conclusive.

- In one of the government sector victims, there were indications that the initial infection vector may have been SSH brute forcing. Multiple open-source sources associate one of the IP addresses used by the threat actors in this activity with SSH brute forcing, indicating that the initial infection vector was possibly SSH brute forcing.
- In another victim, a file path (*Csidl_program_files\loadbalancer\ibm\edge\lb\servers\bin*) indicates a load balancer may have been exploited for access, indicating that the initial infection vector may have been an exposed public-facing server.

While evidence for any of these infection vectors is not definitive, it does appear to indicate that Lancefly is adaptable when it comes to the kind of infection vectors it uses.

Credential theft using non-malware techniques

In activity that also aligned with their earlier campaign in 2020/2021, the attackers used a number of non-malware techniques for credential theft on victim machines:

- PowerShell was used to launch rundll32.exe in order to dump the memory of a process using the MiniDump function of comsvcs.dll. This technique is often used to dump LSASS memory.
- Reg.exe was used to dump the SAM and SYSTEM registry hives.
- A legitimate tool by Avast was installed by the attackers and used to dump LSASS memory.

The attackers also used a masqueraded version of the legitimate archiving tool WinRAR to stage and encrypt files before exfiltration.

Notable attack chain tools and TTPs

- **Impacket Atexec:** A dual-use tool that can be used by malicious actors to create and run an immediate scheduled task on a remote target via SMB in order to execute commands on a target system. It is used by Lancefly for lateral movement across victim networks, also possibly for shellcode execution and evasion. It may have been used to delete cmdline output files.
- **Suspicious SMB activity:** Suspicious SMB activity is seen on numerous victim machines. This is likely related to the use of Impacket by the threat actors.
- **WinRAR:** An archive manager that can be used to archive or zip files – for example, prior to exfiltration. It is not clear how the attackers exfiltrate the data from victim machines, but it is most likely via Merdoor.
- **LSSAS Dumper:** Allows the attackers to swiftly steal credentials they can then use to gain further access across victim networks.
- **NBTScan:** Open-source command-line NetBIOS scanner. This can be used to gather information on a network.
- **Blackloader and Prcloader:** Loaders used by the group. These loaders were also both used in earlier Merdoor activity in 2020 and 2021. They have been linked to the delivery of PlugX. Both loaders appear to be sideloaded onto victim machines. It is not clear if these loaders are exclusively used by Lancefly or if their use is shared across multiple groups.

A typical Merdoor attack chain, as seen in one of the victims, appears to be:

- Merdoor injected into either perfhost.exe or svchost.exe.
- Suspicious SMB activity is then normally observed, and the backdoor connects to its C&C server.
- This is often followed by suspicious living-off-the-land activity, such as the execution of commands like mavinject.exe (which can be used for process injection) and createdump.exe (which can be used to dump a process e.g. LSASS).
- A masqueraded WinRAR (wmiprvse.exe) file is then used to stage and encrypt files, presumably prior to exfiltration. We do not actually see the files being exfiltrated from victim networks, but we presume the Merdoor backdoor itself is used to exfiltrate them.

ZXShell Rootkit Technical Details

The ZXShell rootkit was first reported on by Cisco in 2014, but the version of the tool used by Lancefly is updated, indicating that it continues to be actively developed. The source code of this rootkit is publicly available so it may be used by multiple different groups. The new version of the rootkit used by Lancefly appears to be smaller in size, while it also has additional functions and targets additional antivirus software to disable.

Loader

The loader for the rootkit is a 32-bit DLL with the export directory name "FormDll.dll" (SHA256: 1f09d177c99d429ae440393ac9835183d6fd1f1af596089cc01b68021e2e29a7).
It has the following exports:

- "CallDriver"
- "DoRVA"
- "KillAvpProcess"
- "LoadSys"
- "ProtectDllFile"

Export "Loadsys"

Whenever the export "LoadSys" is executed, it drops one of the following files based on the processor architecture:

- "[WindowsDirectory]\system32\drivers\TdiProxy.sys"
- "[WindowsDirectory]\system64\drivers\TdiProxy.sys"

These files are a malicious Windows Kernel driver. This is a variant of a driver that was first documented in an RSA blog several years ago.

It has the PDB filename: "c:\google\objchk_win7_amd64\amd64\Google.pdb"

The sample creates the device: "\Device\TdiProxy0".

It also creates the symbolic link "\DosDevices\TdiProxy0", so that it can be controlled using the pathname "*\\.\TdiProxy0*".

After this, the loader timestamps the dropped file by copying the timestamps from the file "[WindowsDirectory]\system32\drivers\http.sys".

Then it creates a service with the following parameters:

- ServiceName = "TdiProxy0"
- DisplayName = "TdiProxy0" (later replaced with "TdiProxy")
- BinaryPathName = "[WindowsDirectory]\system32\drivers\TdiProxy.sys"

Export "CallDriver"

"CallDriver" opens the following device, which was created by the "*\\.\TdiProxy0*" malicious kernel driver.

It communicates with it using the *DeviceIoControl* API.

The export expects two arguments. The first argument determines the dwIoControlCode parameter to use when calling the *DeviceIoControl* API and it should be one of the following strings:

- "-init",
- "-file",
- "-pack",
- "-port",
- "-removetcpview",
- "-tcpview",

- "-clearall",
- "-clear",
- "-transport",
- "-waitport",
- "-kill",
- "-antiscan",
- "-removeprocessnotify",
- "-setprocessnotify",
- "-antiantigp",
- "-hideproc",
- "-hidekey",
- "-hidefile",
- "-setprotect",

Any other values result in what looks like a buggy dwIoControlCode value. The second argument is a string to pass as an lpInBuffer parameter when calling the *DeviceIoControl* API, after conversion using the *MultiByteToWideChar* API.

Export "DoRVA"

Whenever the export "DoRVA" is executed, it reads the following file:

- "[file_directory_of_the_DLL]\Form.hlp"

The file should start with the magic string "AP32" and contains shellcode to execute in compressed form.

Export "KillAvpProcess"

This enumerates running processes and for selected processes and calls its own export "CallDriver" with the following parameters:

- first parameter: "-kill"
- second parameter: "[ProcessID]"

The export expects a single string parameter to compare with the executable file of running processes for selection.

Export "ProtectDllFile"

This calls its own export "CallDriver" with the following parameters:

- first parameter: "-file"
- second parameter: "[file_path_of_the_DLL]"

Next, it sets the following registry value:

- HKEY_LOCAL_MACHINE\SOFTWARE\Classes\.ptdf\"ptdffile" = "[file_path_of_the_DLL]"

Loadpoint

This is a 32-bit executable with the PDB filename: "M:\Project\database\10.0.18362\Form\Release\Form.pdb". (SHA256: 180970fce4a226de05df6d22339dd4ae03dfd5e451dcf2d464b663e86c824b8e)

Whenever the sample is executed, it loads the following DLL:

- "[file_path_of_the_running_executable]\FormDll.dll"

It also calls its export: "DoRVA".

Installation and Update Utility

The installation and update utility is a 32-bit PE executable (SHA256: a6020794bd6749e0765966cd65ca6d5511581f47cc2b38e41cb1e7fddaa0b221) that shares small but distinctive fragments of code with the Merdoor loader, which is what indicates they are part of the same toolset.

Whenever the sample is executed, it attempts to read and delete the following file containing its configuration data:

- "[file_directory_of_running_executable]\res.ini"

Update functionality

Next, it checks that:

- "\\.\TdiProxy0" device is available, and
- That its own process was started with the command-line parameter "-up".

If both checks pass, the sample attempts to tamper with various antivirus products using the "\\.\TdiProxy0" device. For example, it may terminate the processes "egui.exe", "ekrn.exe", and "msmpeng.exe".

Next, it attempts to rename the file "[FILE_DIRECTORY_OF_RUNNING_EXECUTABLE]\res.dat" as one of the following (depending on the Windows version):

- "[SystemDrive]\Users\All Users\Windows Defender\temp.temp"
- "[WindowsDirectory]\temp.temp".

Based on the structure of the code, the above file should start with the magic string "AP32" and could contain a DLL file in compressed form. The sample then decompresses the renamed file "temp.temp". When decompressing, it may create the temporary file "temp.temp.pack" in the same folder.

Next, the sample appends a certain marker followed by the content of "[FILE_DIRECTORY_OF_RUNNING_EXECUTABLE]\res.ini" (partially transformed using the XOR algorithm with the byte key 0x12) at the end of the decompressed file.

Additionally, it also creates the following registry value:

- HKEY_CLASSES_ROOT\.udf\"BINTYPE" = [content of "[file_directory_of_running_executable]\res.ini" (partially transformed using the XOR algorithm with the byte key 0x12)]

Then the sample checks if the following file exists:

- "[SystemDrive]\Users\All Users\Windows Defender\DefenderSvc.dll"

If so, the sample renames the updated "temp.temp" file to replace it.
Otherwise, it checks the following registry value for the pathname to replace:

- HKEY_LOCAL_MACHINE\SOFTWARE\Classes\.ecdf\"ecdffile"

If that fails, it uses a default from configuration data.

Finally, it checks the following registry value for a service name:

- HKEY_LOCAL_MACHINE\SOFTWARE\Classes\.tudf\"tudffile"

It restarts the referred service.

Installation functionality

The sample attempts to decompress the following file:

- "[FILE_DIRECTORY_OF_RUNNING_EXECUTABLE]\google64.p" (64-bit processor architecture), or
- "[FILE_DIRECTORY_OF_RUNNING_EXECUTABLE]\google32.p" (32-bit processor architecture)

as:

- "[WindowsDirectory]\Microsoft.NET\Framework64\iesockethlp.dll" (64-bit processor architecture), or
- "[WindowsDirectory]\Microsoft.NET\Framework\iesockethlp.dll" (32-bit processor architecture)

Then it may modify one of the following registry values to hijack the corresponding service:

- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\exfat\"ImagePath" = "\??\[PATHNAME_OF_FILE_DECOMPRESSED_ABOVE]", or
- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\RDPWD\"ImagePath" = "\??\[PATHNAME_OF_FILE_DECOMPRESSED_ABOVE]"

Next, it starts the corresponding service and then removes the registry value. It then attempts to tamper with various antivirus products using the "\\.\TdiProxy0" device.

It then creates a service with the following parameters:

- ServiceName: "[PER CONFIGURATION DATA]"
- ImagePath:
    - "%SystemRoot%\System32\svchost.exe -k netsvcs", or

    - "%SystemRoot%\System32\svchost.exe -k ntmssvcs"
- Parameters:
    - ServiceDll:
        - "C:\WINDOWS\Microsoft.NET\Framework64\[PER CONFIGURATION DATA]", or
        - "C:\WINDOWS\Microsoft.NET\Framework\[PER CONFIGURATION DATA]"

Then it creates the following registry value:

- HKEY_LOCAL_MACHINE\SOFTWARE\Classes\.tudf\"tudffile" = [NAME OF CREATED SERVICE]

It then deletes the following registry values:

- HKEY_LOCAL_MACHINE\SOFTWARE\Classes\.ptdf\"ptdffile"
- HKEY_LOCAL_MACHINE\SOFTWARE\Classes\.ecdf\"ecdffile"

Next, it renames the following file:

- "[FILE_DIRECTORY_OF_RUNNING_EXECUTABLE]\res.dat"

as:

- "[WindowsDirectory]\Microsoft.NET\Framework64\[PER CONFIGURATION DATA].back" (64-bit processor architecture), or
- "[WindowsDirectory]\Microsoft.NET\Framework\[PER CONFIGURATION DATA].back" (32-bit processor architecture)

Based on the structure of the code, the above file should start with the magic string "AP32" and could contain a DLL file in compressed form (using aPLib for compression).

The sample then decompresses the renamed "[PER CONFIGURATION DATA].back" as "[PER CONFIGURATION DATA]".

Next, the sample appends a certain marker followed by the content of "[FILE_DIRECTORY_OF_RUNNING_EXECUTABLE]\res.ini" (partially transformed using the XOR algorithm with the byte key 0x12) at the end of the decompressed file.

Additionally, it also creates the following registry value:

- HKEY_CLASSES_ROOT\.udf\"BINTYPE" = [content of "[FILE_DIRECTORY_OF_RUNNING_EXECUTABLE]\res.ini" (partially transformed using the XOR algorithm with the byte key 0x12)]

Finally, when the configuration data includes the option "OneSelfKey", it makes a compressed copy of its own executable as (using aPLib for compression):

- "[WindowsDirectory]\SysWOW64\nethlp.hlp" (64-bit processor architecture), or
- "[WindowsDirectory]\system32\nethlp.hlp" (32-bit processor architecture).

Some samples include an embedded archive with the final payload:

- "Msrpcsvc.dll"

This is a variant of the ZXShell backdoor (SHA256: d5df686bb202279ab56295252650b2c7c24f350d1a87a8a699f6034a8c0dd849).

Possible Links to Other Groups

The ZXShell rootkit used by Lancefly is signed by the certificate "Wemade Entertainment Co. Ltd", which was previously reported to be associated with APT41 (aka Blackfly/Grayfly). However, it is known that Chinese APT groups, such as APT41, often share certificates with other APT groups. The ZXShell backdoor has also previously been used by the HiddenLynx/APT17 group, but as the source code of ZXShell is now publicly available this does not provide a definitive link between these two groups.

Also notable is that the ZXShell rootkit loader component has the name "formdll.dll" and it has the ability to read the file "Form.hlp" and execute its contents as shellcode. Those same files were mentioned as being used in a previous report detailing activity by the Iron Tiger (aka Budworm/APT27) group. In that case, the attackers used these filenames when loading the PlugX backdoor onto victim machines. The prevalence of such files is very low, which may indicate a potential link between that campaign and this more recent activity.

PlugX is also seen being used by Lancefly. PlugX is a remote access Trojan (RAT) with multiple functionalities including backdoor access and data exfiltration. PlugX has existed for well over a decade. It was originally used by Chinese APT groups, but its use is now very widespread, meaning it is difficult to use it as a way of attributing activity.

ShadowPad is also used by these attackers. ShadowPad is a modular RAT believed to be exclusively used by Chinese APT groups. Its capabilities are similar to PlugX, and it is often referred to as a successor to that malware.

While these overlaps and shared tools may indicate some links between Lancefly activity and activity by other APT groups, none of the overlaps are strong enough to attribute this activity and the development of the Merdoor backdoor to an already-known attack group.

Noteworthy Backdoor, Targeted Activity

This recent Lancefly activity is of note due to its use of the Merdoor backdoor, but also the low prevalence of this backdoor and the seemingly highly targeted nature of these attacks. While the Merdoor backdoor appears to have been in existence for several years, it appears to only have been used in a small number of attacks in that time period. This prudent use of the tool may indicate a desire by Lancefly to keep its activity under the radar.

The tools used and sectors targeted all point to the motivations of this attack campaign being intelligence gathering. The similarities between this recent activity and earlier activity by Lancefly indicate that the group perhaps did not realize the earlier activity had been discovered, so it was not concerned about links being made between the two. Whether or not the exposure of this activity will lead to any alteration in how the group carries out its activity remains to be seen.

Protection

For the latest protection updates, please visit the Symantec Protection Bulletin.

Indicators of Compromise (IOCs)

Merdoor Backdoor

SHA256 Filename Description

13df2d19f6d2719beeff3b882df1d3c9131a292cf097b27a0ffca5f45e139581 – a.exe – Merdoor Dropper

8f64c25ba85f8b77cfba3701bebde119f610afef6d9a5965a3ed51a4a4b9dead – chrome_frame_helper.exe – Merdoor Dropper

8e98eed2ec14621feda75e07379650c05ce509113ea8d949b7367ce00fc7cd38 – siteadv.exe – Merdoor Dropper

89e503c2db245a3db713661d491807aab3d7621c6aff00766bc6add892411ddc – siteadv.exe – Merdoor Dropper

c840e3cae2d280ff0b36eec2bf86ad35051906e484904136f0e478aa423d7744 –siteadv.exe –Merdoor Dropper

5f16633dbf4e6ccf0b1d844b8ddfd56258dd6a2d1e4fb4641e2aa508d12a5075 –chrome_frame_helper.dll – Merdoor Loader

ff4c2a91a97859de316b434c8d0cd5a31acb82be8c62b2df6e78c47f85e57740 –chrome_frame_helper.dll – Merdoor Loader

14edb3de511a6dc896181d3a1bc87d1b5c443e6aea9eeae70dbca042a426fcf3 –chrome_frame_helper.dll – Merdoor Loader

db5deded638829654fc1595327400ed2379c4a43e171870cfc0b5f015fad3a03 –chrome_frame_helper.dll – Merdoor Loader

e244d1ef975fcebb529f0590acf4e7a0a91e7958722a9f2f5c5c05a23dda1d2c –chrome_frame_helper.dll – Merdoor Loader

f76e001a7ccf30af0706c9639ad3522fd8344ffbdf324307d8e82c5d52d350f2 –chrome_frame_helper.dll – Merdoor Loader

dc182a0f39c5bb1c3a7ae259f06f338bb3d51a03e5b42903854cdc51d06fced6 – smadhook64c.dll – Merdoor Loader

fa5f32457d0ac4ec0a7e69464b57144c257a55e6367ff9410cf7d77ac5b20949 – SiteAdv.dll, chrome_frame_helper.dll –Merdoor Loader

fe7a6954e18feddeeb6fcdaaa8ac9248c8185703c2505d7f249b03d8d8897104 – siteadv.dll – Merdoor Loader

341d8274cc1c53191458c8bbc746f428856295f86a61ab96c56cd97ee8736200 – siteadv.dll – Merdoor Loader f3478ccd0e417f0dc3ba1d7d448be8725193a1e69f884a36a8c97006bf0aa0f4 – siteadv.dll – Merdoor Loader

750b541a5f43b0332ac32ec04329156157bf920f6a992113a140baab15fa4bd3 – mojo_core.dll – Merdoor Loader

9f00cee1360a2035133e5b4568e890642eb556edd7c2e2f5600cf6e0bdcd5774 – libmupdf.dll – Merdoor Loader

a9051dc5e6c06a8904bd8c82cdd6e6bd300994544af2eed72fe82df5f3336fc0 – chrome_frame_helper.dll – Merdoor Loader

d62596889938442c34f9132c9587d1f35329925e011465c48c94aa4657c056c7 – smadhook64c.dll – Merdoor Loader

f0003e08c34f4f419c3304a2f87f10c514c2ade2c90a830b12fdf31d81b0af57 – SiteAdv.pak – Merdoor encoded payload

139c39e0dc8f8f4eb9b25b20669b4f30ffcbe2197e3a9f69d0043107d06a2cb4 – SiteAdv.pak – Merdoor encoded payload

11bb47cb7e51f5b7c42ce26cbff25c2728fa1163420f308a8b2045103978caf5 – SiteAdv.pak – Merdoor encoded payload

0abc1d12ef612490e37eedb1dd1833450b383349f13ddd3380b45f7aaabc8a75 – SiteAdv.pak – Merdoor encoded payload

eb3b4e82ddfdb118d700a853587c9589c93879f62f576e104a62bdaa5a338d7b –SiteAdv.exe – Legit McAfee executable

1ab4f52ff4e4f3aa992a77d0d36d52e796999d6fc1a109b9ae092a5d7492b7dd – chrome_frame_helper.exe – Legit Google executable

fae713e25b667f1c42ebbea239f7b1e13ba5dc99b225251a82e65608b3710be7 – SmadavProtect64.exe – Legit SmadAV executable

File hashes, simplified list

13df2d19f6d2719beeff3b882df1d3c9131a292cf097b27a0ffca5f45e139581

8f64c25ba85f8b77cfba3701bebde119f610afef6d9a5965a3ed51a4a4b9dead

8e98eed2ec14621feda75e07379650c05ce509113ea8d949b7367ce00fc7cd38

89e503c2db245a3db713661d491807aab3d7621c6aff00766bc6add892411ddc

c840e3cae2d280ff0b36eec2bf86ad35051906e484904136f0e478aa423d7744

5f16633dbf4e6ccf0b1d844b8ddfd56258dd6a2d1e4fb4641e2aa508d12a5075

ff4c2a91a97859de316b434c8d0cd5a31acb82be8c62b2df6e78c47f85e57740

14edb3de511a6dc896181d3a1bc87d1b5c443e6aea9eeae70dbca042a426fcf3 db5deded638829654fc1595327400ed2379c4a43e171870cfc0b5f015fad3a03

e244d1ef975fcebb529f0590acf4e7a0a91e7958722a9f2f5c5c05a23dda1d2c

f76e001a7ccf30af0706c9639ad3522fd8344ffbdf324307d8e82c5d52d350f2

dc182a0f39c5bb1c3a7ae259f06f338bb3d51a03e5b42903854cdc51d06fced6

fa5f32457d0ac4ec0a7e69464b57144c257a55e6367ff9410cf7d77ac5b20949

fe7a6954e18feddeeb6fcdaaa8ac9248c8185703c2505d7f249b03d8d8897104

341d8274cc1c53191458c8bbc746f428856295f86a61ab96c56cd97ee8736200

f3478ccd0e417f0dc3ba1d7d448be8725193a1e69f884a36a8c97006bf0aa0f4

750b541a5f43b0332ac32ec04329156157bf920f6a992113a140baab15fa4bd3

9f00cee1360a2035133e5b4568e890642eb556edd7c2e2f5600cf6e0bdcd5774

a9051dc5e6c06a8904bd8c82cdd6e6bd300994544af2eed72fe82df5f3336fc0

d62596889938442c34f9132c9587d1f35329925e011465c48c94aa4657c056c7

f0003e08c34f4f419c3304a2f87f10c514c2ade2c90a830b12fdf31d81b0af57

139c39e0dc8f8f4eb9b25b20669b4f30ffcbe2197e3a9f69d0043107d06a2cb4

11bb47cb7e51f5b7c42ce26cbff25c2728fa1163420f308a8b2045103978caf5

0abc1d12ef612490e37eedb1dd1833450b383349f13ddd3380b45f7aaabc8a75

eb3b4e82ddfdb118d700a853587c9589c93879f62f576e104a62bdaa5a338d7b

1ab4f52ff4e4f3aa992a77d0d36d52e796999d6fc1a109b9ae092a5d7492b7dd

fae713e25b667f1c42ebbea239f7b1e13ba5dc99b225251a82e65608b3710be7

1f09d177c99d429ae440393ac9835183d6fd1f1af596089cc01b68021e2e29a7

180970fce4a226de05df6d22339dd4ae03dfd5e451dcf2d464b663e86c824b8e

a6020794bd6749e0765966cd65ca6d5511581f47cc2b38e41cb1e7fddaa0b221

592e237925243cf65d30a0c95c91733db593da64c96281b70917a038da9156ae

929b771eabef5aa9e3fba8b6249a8796146a3a4febfd4e992d99327e533f9798

009d8d1594e9c8bc40a95590287f373776a62dad213963662da8c859a10ef3b4

ef08f376128b7afcd7912f67e2a90513626e2081fe9f93146983eb913c50c3a8 ee486e93f091a7ef98ee7e19562838565f3358caeff8f7d99c29a7e8c0286b28

32d837a4a32618cc9fc1386f0f74ecf526b16b6d9ab6c5f90fb5158012fe2f8c

d5df686bb202279ab56295252650b2c7c24f350d1a87a8a699f6034a8c0dd849

a1f9b76ddfdafc47d4a63a04313c577c0c2ffc6202083422b52a00803fd8193d

3ce38a2fc896b75c2f605c135297c4e0cddc9d93fc5b53fe0b92360781b5b94e

210934a2cc59e1f5af39aa5a18aae1d8c5da95d1a8f34c9cfc3ab42ecd37ac92

530c7d705d426ed61c6be85a3b2b49fd7b839e27f3af60eb16c5616827a2a436

5018fe25b7eac7dd7bc30c7747820e3c1649b537f11dbaa9ce6b788b361133bf

efa9e9e5da6fba14cb60cba5dbd3f180cb8f2bd153ca78bbacd03c270aefd894

a5a4dacddfc07ec9051fb7914a19f65c58aad44bbd3740d7b2b995262bd0c09e

10b96290a17511ee7a772fcc254077f62a8045753129d73f0804f3da577d2793

0dcfcdf92e85191de192b4478aba039cb1e1041b1ae7764555307e257aa566a7

415f9dc11fe242b7a548be09a51a42a4b5c0f9bc5c32aeffe7a98940b9c7fc04

947f7355aa6068ae38df876b2847d99a6ca458d67652e3f1486b6233db336088

8d77fe4370c864167c1a712d0cc8fe124b10bd9d157ea59db58b42dea5007b63

d8cc2dc0a96126d71ed1fce73017d5b7c91465ccd4cdcff71712381af788c16d

e94a5bd23da1c6b4b8aec43314d4e5346178abe0584a43fa4a204f4a3f7464b9

5655a2981fa4821fe09c997c84839c16d582d65243c782f45e14c96a977c594e

19ec3f16a42ae58ab6feddc66d7eeecf91d7c61a0ac9cdc231da479088486169

41d174514ed71267aaff578340ff83ef00dbb07cb644d2b1302a18aa1ca5d2d0

67ebc03e4fbf1854a403ea1a3c6d9b19fd9dc2ae24c7048aafbbff76f1bea675

f92cac1121271c2e55b34d4e493cb64cdb0d4626ee30dc77016eb7021bf63414

859e76b6cda203e84a7b234c5cba169a7a02bf028a5b75e2ca8f1a35c4884065

fcdec9d9b195b8ed827fb46f1530502816fe6a04b1f5e740fda2b126df2d9fd5

9584df964369c1141f9fc234c64253d8baeb9d7e3739b157db5f3607292787f2

711a347708e6d94da01e4ee3b6cdb9bcc96ebd8d95f35a14e1b67def2271b2e9 f040a173b954cdeadede3203a2021093b0458ed23727f849fc4c2676c67e25db

90edb2c7c3ba86fecc90e80ac339a42bd89fbaa3f07d96d68835725b2e9de3ba

b0d25b06e59b4cca93e40992fa0c0f36576364fcf1aca99160fd2a1faa5677a2

4c55f48b37f3e4b83b6757109b6ee0a661876b41428345239007882993127397

3e1c8d982b1257471ab1660b40112adf54f762c570091496b8623b0082840e9f

9830f6abec64b276c9f327cf7c6817ad474b66ea61e4adcb8f914b324da46627

79ae300ac4f1bc7636fe44ce2faa7e5556493f7013fc5c0a3863f28df86a2060

Source: https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/lancefly-merdoor-zxshell-custom-backdoor