SafePay ransomware: The fast-rising threat targeting MSPs

Archived: 2026-04-06 00:03:32 UTC

Summary

- In Q1 2025, one ransomware group surged rapidly from obscurity to become one of the most active and dangerous actors on the global threat landscape: SafePay. It has quietly and aggressively built momentum, striking over 200 victims worldwide, including managed service providers (MSPs) and small-to-midsize businesses (SMBs) across industries.
- Acronis Threat Research Unit (TRU) analyzed several SafePay samples and confirmed the group's use of recycled — but highly efficient — tactics, including disabling endpoint protection, deleting shadow copies and clearing logs to suppress detection and response.
- Unlike many ransomware groups that rely on affiliates in a ransomware-as-a-service (RaaS) model, SafePay appears to operate with centralized control, managing its own operations, infrastructure and negotiations.
- The group uses classic but effective techniques: RDP- and VPN-based intrusion, credential theft, privilege escalation and living-off-the-land binaries to quietly move through victim networks, exfiltrate sensitive data and then encrypt files.
- Most recently, SafePay was linked to the ransomware attack that disrupted Ingram Micro, a global distributor serving thousands of partners and MSPs.

SafePay ransomware: A LockBit spinoff?

SafePay first appeared in 2024. In its first year of operations, it claimed more than 20 victims. While it is unknown if SafePay is a new player or a rebrand of an old one, their samples share a lot of similarities with the LockBit ransomware family, a well-known ransomware builder. In 2022, their LockBit 3.0 builder source code was leaked. After the leak, a variety of new ransomware appeared. The 3.0 version of the builder is also known as LockBit Black because it shares a lot of similarities with BlackByte ransomware.

The SafePay ransomware sample analyzed by TRU is PE32 DLL, with a fake compilation timestamp.

The first similarity with LockBit is a dummy function that has several sequential Windows API calls. Writing calls in that way is nonsensical because there are no arguments passed to those functions, which will cause errors. In fact, in both cases, those functions will never be called, as execution will be terminated before the sample can enter this section.

Other similarities with LockBit:

- Requires password for full execution.
- All strings are encoded.
- All WinAPI addresses are resolved during execution.
- Same system languages avoidance.

- CMSTPLUA COM interface abuse for privilege escalation.
- Created threads have ‘ThreadHideFromDebugger’ flag.
- The list of processes and services that must be terminated.

While the sample is not a complete copy of LockBit 3.0 and has some differences, it is common for threat actors to change the source code to make its malware more unique, and more importantly, to add new features and improve detection avoidance.

Delivery and exfiltration

SafePay ransomware was delivered to the victims using RDP connections. While it is unknown how threat actors got credentials, the technique enabled them to disable Windows Defender and upload files to the C2 server before encrypting them. Before exfiltrating files, attackers executed ‘ShareFinder.ps1’ script, which finds all available network shares in the local domain. It was taken from open source project:

https://github.com/darkoperator/Veil-PowerView/blob/master/PowerView/functions/Invoke-ShareFinder.ps1

This script finds network shares on hosts in the local domain. It was previously spotted in Emotet attacks, as well as during the C0015 campaign, which was used to deploy Conti ransomware.

To collect files on the system, the WinRar program was used with following command:

```
WinRAR.exe  a -v5g -ed -r -tn1000d -m0 -mt5 -x*.rar -x*.JPEG -x*.RAW -x*.PSD -x*.TIFF -x*.BMP -x*.GIF -x*.JPG -x*.MOV -x*.pst -x*.FIT -x*.FIL -x*.mp4 -x*.avi -x*.mov -x*.mdb -x*.iso -x*.exe -x*.dll -x*.bak -x*.msg -x*.png -x*.zip -x*.ai -x*.7z -x*.DPM -x*.log -x*.dxf -x*.insp -x*.upd -x*.db -x*.dwg -x*.nc1 -x*.metadata -x*.dg -x*.inp -x*.dat -x*.TIFF -x*.tiger -x*.pcp -x*.rvt -x*.rws -x*.nwc -x*.tif -x*.frx -x*.dyf -x*.rcs -x*.diff C:\[redacted].rar \\[redacted]\C$\Users\
```

After archiving files, a FileZilla client was deployed to exfiltrate files to the C2 server. After the process was done, both WinRar and FileZilla were removed from targets.

Execution

At the start of execution, SafePay decrypts strings. It uses a loop, where it performs XOR operation three times on each byte. Each operation uses a different key. The first uses the current index value. The second uses the first symbol from ‘kernel32.dll,’ which is always ‘M’. The last key is a constant value, which is different for each encrypted string. This decryption routine is not implemented as a separate function but is used to decrypt every string that the sample contains. The sample contains strings only in encrypted format.

The sample also doesn’t contain the import functions table. It decrypts library names and loads them using ‘LoadLibrary’ import, as well as their export function addresses are resolved and saved using ‘GetProcAddress’. Here is the list of library names that SafePay stores in an encrypted format:

```
advapi32.dll, rstrtmgr.dll, ole32.dll, shell32.dll, ntdll.dll, mpr.dll, user32.dll
```

After obtaining additional imports, the SafePay sample gets the current date / time and Windows UI language. Next, it checks if the system language ID number is bigger or smaller than the saved ones. The sample will then jump to a particular section with other numbers comparison.

Using the ‘switch-case’ statement, SafePay can continue execution or jump to the exit function when the obtained value matches one of the next numbers:

Code	Description	BCP 47 code
1049	Russian	ru-RU
1058	Ukrainian	uk-UA
1059	Belarusian	be-BY
1064	Tajik	tg-Cyrl-TJ
1067	Armenian – Armenia	hy-AM
1068	Azerbaijani (Latin)	az-Latn-AZ
1079		

Georgian

ka-GE

Next, the SafePay sample gets command line arguments and decodes additional strings. Those strings are supported arguments:

Argument	Description
-uac	UAC Bypass flag
-network	Network propagation
-selfdelete	Self-deletion after execution
-log	Enable logging
-netdrive	Encrypt network drives
-pass=	Provide password
-path=	Provide a path to encrypt
-enc=	Provide encryption level

To parse arguments, the SafePay sample grabs the command line that was used to execute the sample and stores it as an arguments array, which in the loop will be compared with the saved list. When one element of an array is compared with a saved list, the sample adds ‘1’ to the array index value. When any argument is matched, it sets the appropriate value to ‘1’.

While some arguments are just set flags, others must contain additional information. For example, a password argument must consist of 38 symbols, including the ‘-pass=’ substring. In any other case, it will exit the program.

The password must be 32 bytes in length and used to decode additional information in the code. If the password is unknown, the whole execution process cannot be performed.

The encryption level argument must be six symbols, including an ‘-enc=’ substring. It accepts numbers from ‘1’ to ‘9’ for this argument. This value will be multiplied by 10 and will serve as file encryption percentage. For example, providing value ‘5' will force the sample to encrypt 50% of the file.

After arguments are parsed, the SafePay sample creates a new access control list (ACL) and adds an access-denied access control entry (ACE). This list is used in the ‘SetSecurityInfo’ function.

Next, the sample tries to obtain ‘SeDebugPrivilege’.

After obtaining this privilege, the sample creates a snapshot of all running processes in the system. It then compares their names with its own saved list, which is also stored in encoded format. When the appropriate process is found, it will be terminated.

Here is a list of processes that must be terminated:

sql, oracle, ocssd, dbsnmp, synctime, agntsvc, isqlplussvc, xfssvccon, mydesktopservice, ocautoupds, encsvc, firefox, tbirdconfig, mydesktopqos, ocomm, dbeng50, sqbcoreservice, excel, infopath, msaccess, mspub, far, onenote, outlook, powerpnt, steam, thebat, thunderbird, visio, winword, wordpad, notepad, wuauclt, onedrive, sqlmangr

Besides processes, the sample also terminates some services. First, it opens the service manager and then again starts searching for service names that match its own list. The sample will then terminate services using the ‘ControlService’ function and value ‘1’ in the ‘dwControl’ argument.

The list of services that will be terminated:

vss, sqlsvc, memtas, mepocs, msexchange, Sophos, Veeam, backup, GxVss, GxBlr, GxFWD, GxCVD, GxCIMgr

After emptying the Recycle Bin using the ‘SHEmptyRecycleBinW’ function, the sample loads the ‘*Software\Microsoft\Windows\CurrentVersion\Run*’ registry path and creates a new key with a command that was used to execute the sample, including all arguments. This will cause SafePay to be executed each time Windows starts up.

If the ‘-uac’ flag is passed, the sample abuses the CMSTPLUA COM interface to execute commands with elevated permissions using ‘ShellExecuteW’ API function:

```
/c vssadmin delete shadows /all /quiet

/c wmic shadowcopy delete

/c bcdedit / set{default} bootstatuspolicy ignoreallfailures

/c bcdedit / set{default} recoveryenabled no
```

If ‘-log’ argument is provided, the sample will create a log file: ‘C:\ProgramData\auto.log’. It contains PID (Process ID), time and description of performed operation.

File encryption

Before starting the encryption routine, the SafePay sample opens the handle to the system default cryptographic service provider with RSA_AES type.

Each encryption thread will be created in suspended mode. After thread creation, the sample sets the ‘ThreadHideFromDebugger’ flag to thread to avoid their debugging and sets previously duplicated token information.

To find drives on the system, the sample uses ‘GetVolumePathNamesForVolumeNameW’ and ‘GetLogicalDrives’ imports. Each drive is checked for its type. The SafePay sample will encrypt the drive only if it has type 2 (DRIVE_REMOVABLE) or 3 (DRIVE_FIXED). Additional disk information will be obtained using ‘DeviceIoControl’ with control code ‘IOCTL_DISK_GET_PARTITION_INFO’. If it finds an unmounted drive, the sample mounts it using the ‘SetVolumeMountPoint’ function.

To search files on the system, the sample loads found drive names and uses ‘FindFirstFile’ and ‘FindNextFile’ functions. For each file, it checks ‘dwFileAttributes’ parameter. If the parameter matches 16, which is a directory, the sample will call the same function with the found folder path.

When the sample opens a file using ‘CreateFileW’, it sets ‘dwFlagsAndAttributes’ value to ‘0x04000000’, which is the ‘FILE_FLAG_DELETE_ON_CLOSE’ flag. The sample will then delete the opened file right after the sample closes its handle.

The sample generates 32 random bytes for each file, which is used to be an AES key. Then the AES key will be encrypted using the RSA algorithm.

After encrypting each file in the directory, the sample uses ‘CreateIOCompletionPort’ and ‘PostQueuedCompletionStatus’ APIs to handle multithread encryption safely. Finally, the sample renames the file, appending the '.safepay' extension to it.

Conclusion

SafePay ransomware uses double extortion to ensure that victims pay their ransoms. Exfiltrating files first, attackers deploy malware to encrypt users' files using a strong combination of AES and RSA ciphers. The SafePay sample is a DLL file, which requires ‘regsvr32.exe’ or ‘rundll32.dll’ utilities for execution. The sample accepts multiple arguments but always requires ‘-pass=’ one, as this password is used to decode additional information in code. All saved strings and import names are stored in an encrypted format, making SafePay hard to detect before it is executed.

Detected by Acronis

IoCs

Files

SHA256

a0dc80a37eb7e2716c02a94adc8df9baedec192a77bde31669faed228d9ff526

Network indicators

URL

http://nz4z6ruzcekriti5cjjiiylzvrmysyqwibxztk6voem4trtx7gstpjid.onion

VanessaCooke94@protonmail.com
