



Payload

Technical Analysis — Windows

REVERSE-ENGINEERED REPORT

RansomLook · ransomlook.io

File last modified: 2026-04-28

Analysis date: 2026-04-26

Sample SHA-256: `1ca67af90400ee6cbbd42175293274a0f5dc05315096cb2e214e4bfe12ffb71f`

Payload Ransomware (Windows)

1. Sample Identification

Field	Value
Family	Payload
SHA-256	1ca67af90400ee6cbbd42175293274a0f5dc05315096cb2e214e4bfe12ffb71f
Type	PE32 executable, Intel 80386, console, Windows
Size	394 752 bytes (385 KB)
Language	C++ (MSVC, Concurrency Runtime)
Image base	0x400000
Sections	5: .text , .idata , .rdata , .data , .fptable
Functions	2329 total (22 named, 778 library)
RansomLook	https://www.ransomlook.io/group/Payload

This is the **Windows variant** of the Payload ransomware family, companion to the ESXi/Linux version (`bed8d175...`). Compiled with MSVC, uses the Concurrency Runtime (`Concurrency::` symbols), and makes extensive use of NT Native APIs resolved at runtime. Features AVX2/SSE2-optimized ChaCha20 encryption, Curve25519 key exchange, IOCP-based multi-threaded file processing, ETW patching, and self-deletion via NTFS ADS rename.

Imports (4 DLLs)

DLL	Count	Purpose
KERNEL32	98	Core API: process, file, thread, memory, IOCP
ADVAPI32	7	CryptGenRandom, service control (SCM)
SHELL32	3	ShellExecuteW, SHEmptyRecycleBinA, CommandLineToArgvW
ntdll.dll	14	Resolved at runtime: NtCreateFile, NtReadFile, NtWriteFile, NtQueryDirectoryFile, etc.

2. Infrastructure

Field	Value
DLS Onion	payloadrz5yw227brtbvdqpnlhq3rdcdekdn3rgucbcdeawq2v6vuyd.onion
Chat Onion	payloadynyvabjacbun4uwmx7yvzorycszlzmnleguxjn7glahsvqd.onion
Note filename	RECOVER_payload.txt
Extension	.payload
Mutex	MakeAmericaGreatAgain
Deadline	72h (contact) / 240h (negotiation)
Log file	\\?\C:\payload.log

The infrastructure is shared with the ESXi variant (same onion addresses), but per-victim credentials differ.

3. Ransom Note

Filename

`RECOVER_payload.txt` — written to every traversed directory via `write_ransom_note` @ `0x409EB5` .

Encryption

The note content is stored as a base64-encoded, RC4-encrypted blob in `.rdata` (1896 bytes encrypted). The RC4 key is itself double-base64 encoded: - Outer:

`aFFFUEl0dVRZMmx5dHVmTERwS1FkVmtYV2xvdkMxUVI=` - Inner: `hQEPINuTY2lytufLDpJQdVkrWlovC1QR` (32-byte ASCII key)

Full Ransom Note

Welcome to Payload!

The next 72 hours will determine certain factors in the life of your company: the publication of the file tree, which we have done safely and unnoticed by all of you, and the publication of your company's full name on our luxurious blog. NONE of this will happen if you contact us within this time frame and our negotiations are favorable.

We are giving you 240 hours to:

1. familiarize yourself with our terms and conditions,
2. begin negotiations with us,
3. and successfully conclude them.

The timer may be extended if we deem it necessary (only in the upward direction). Once the timer expires, all your information will be posted on our blog.

ATTENTION!

Contacting authorities, recovery agencies, etc. WILL NOT HELP YOU!

At best, you will waste your money and lose some of your files, which they will carefully take to restore!

You should also NOT turn off, restart, or put your computer to sleep.

In the future, such mistakes can make the situation more expensive and the files will not be restored!

We DO NOT recommend doing anything with the files, as this will make it difficult to recover them later!

When contacting us:

you can request up to 3 files from the file tree,
you can request up to 3 encrypted files up to 15 megabytes
so that we can decrypt them and you understand that we can do it.

First, you should install Tor Browser:

1. Open: <https://www.torproject.org/download>
2. Choose your OS and select it
3. Run installer
4. Enjoy!

In countries where tor is prohibited, we recommend using bridges, which you can take: <https://bridges.torproject.org/>

You can read:

<http://payloadrz5yw227brtbvdqpnlhq3rdcdekdn3rgucbcdeawq2v6vuyd.onion> (Tor)

To start negotiations, go to <http://payloadnyvabjacbun4uwhmxc7yvdzorycslzmnleguxjn7glaahsvqd.onion> and login:

User: [SNIP]

Password: [SNIP]

Your ID to verify: [SNIP]

4. Execution Flow

main @ 0x40A9C7 (0x5A6 bytes)

1. resolve_nt_native_apis() → NtCreateFile, NtReadFile, NtWriteFile, etc.
2. resolve_wow64_apis() → Wow64DisableWow64FsRedirection
3. Parse CLI arguments:
 - log → enable logging to \\?\C:\payload.log
 - p <path> → target specific path
 - algo <name> → force AVX2/SSE2/default
 - threads <n> → override thread count
 - background → run as detached background process
 - m → skip mutex check
 - n → skip ransom note writing
 - d → skip auto-deletion (self-delete)
 - k → skip process/service killing
 - s → skip something (unused?)
 - bypass-etw → patch ETW functions in ntdll
 - l → delete logs
 - i → ignore all file extension filters
4. CreateMutexW("MakeAmericaGreatAgain") → single instance
5. If not --background:
 - a. Re-launch self with --background flag → detached execution
6. If --background:
 - a. If --bypass-etw: patch_etw_functions()
 - b. pre_encryption_setup()
 - c. encryption_orchestrator()
7. ReleaseMutex → exit

pre_encryption_setup @ 0x40A855

1. open_log_file() → \\?\C:\payload.log via NtCreateFile
2. SetProcessShutdownParameters(0, 0) → last to shut down
3. SHEmptyRecycleBinA(NULL, NULL, 7) → empty recycle bin silently
4. If NOT --k:
 - a. delete_shadow_copies() → vssadmin delete shadows
 - b. kill_services() → 44 services stopped
 - c. kill_processes() → 31 processes terminated
5. If NOT --n: write_ransom_note() → RECOVER_payload.txt
6. If NOT --d: self_delete_via_ads_rename() → NTFS ADS rename trick

encryption_orchestrator @ 0x40A22C (0x628 bytes)

1. Decode embedded Curve25519 public key (base64 → 32 bytes)
2. Validate pubkey length >= 32 bytes
3. cpuid_detect_features() → check AVX2/SSE2
4. Select ChaCha20 implementation:
 - AVX2: chacha20_block_avx2 @ 0x402C2E (4619 bytes)
 - SSE2: chacha20_block_sse2 @ 0x401DBD (3596 bytes)
 - Scalar: chacha20_block_scalar @ 0x401992 (1067 bytes)
5. Thread count = CLI --threads or GetLogicalProcessorInformation
6. Queue size = 4 * cpu_cores (min 2)
7. CreateIoCompletionPort()
8. Spawn 2 * cpu_cores worker threads (iocp_worker_thread)
9. If --p: encrypt specific path
Else: enumerate_drives() → recursive_file_traversal() per drive
10. Wait for all workers to complete
11. PostQueuedCompletionStatus(NULL) × workers → signal shutdown
12. Join all threads

5. Encryption System

Architecture

```

encryption_orchestrator
├─ Decode Curve25519 pubkey from base64
├─ CPUID → select ChaCha20 variant (AVX2 / SSE2 / scalar)
├─ CreateIoCompletionPort
├─ Spawn 2*N worker threads
│   └─ iocp_worker_thread @ 0x40A1A2
│       └─ GetQueuedCompletionStatus (infinite wait)
│           └─ encrypt_single_file @ 0x40946F
├─ enumerate_drives (A:-Z: via NT API)
│   └─ recursive_file_traversal @ 0x4099C0
│       └─ Skip "." and ".."
│       └─ Skip directories in exclusion list
│       └─ Skip files matching ransom note name
│       └─ Skip files with excluded extensions
│       └─ Skip files matching whitelisted extensions
│       └─ Rate-limit: SwitchToThread if queue full
│           └─ PostQueuedCompletionStatus (file path)
└─ Wait → signal workers → cleanup

```

encrypt_single_file @ 0x40946F (0x551 bytes)

1. NtCreateFile(path, GENERIC_READ|WRITE|DELETE|SYNCHRONIZE)
2. NtQueryInformationFile → get file size
3. LockFileEx(EXCLUSIVE)
4. CryptGenRandom(32) → ephemeral ChaCha20 key
5. CryptGenRandom(12) → ChaCha20 nonce
6. Clamp key for X25519: key[0] &= 0xF8, key[31] = (key[31] & 0x3F) | 0x40
7. x25519_scalar_mult(ephemeral_key, basepoint_9) → ephemeral public key
8. x25519_scalar_mult(ephemeral_key, master_pubkey) → shared secret
9. Intermittent encryption:
 - Files <= 2 GB: encrypt entirely
 - Files > 2 GB: encrypt 1/5 of file (sparse blocks)
10. Read in 1 MB (0x100000) chunks → ChaCha20 XOR → write back (in-place)
11. Write footer:
 - "payload" marker (7 bytes)
 - RC4-encrypted metadata (56 bytes: ephemeral pubkey + nonce)
12. Rename file with .payload extension
13. UnlockFileEx → NtClose
14. Wipe key material from stack

Crypto Primitives

Primitive	Function	Address	Size
ChaCha20 (AVX2)	chacha20_block_avx2	0x402C2E	4619 B
ChaCha20 (SSE2)	chacha20_block_sse2	0x401DBD	3596 B
ChaCha20 (scalar)	chacha20_block_scalar	0x401992	1067 B
X25519 (Curve25519)	x25519_scalar_mult	0x40598E	142 B
RC4 (KSA)	rc4_key_schedule	0x4093A9	91 B
RC4 (PRGA XOR)	rc4_prga_xor	0x409404	107 B
CSPRNG	CryptGenRandom	ADVAPI32 import	—

Key Material

Key	Value	Purpose
Curve25519 pubkey	aH9Tbdc+qPcQkPwhcLaNYFadhF04GzuGsuRxDbKMRkU =	Master public key for X25519 ECDH
	687f536dd73ea8f71090fc2172568d60569d845d381 b3b86b2e4710db28c4645	(hex)
RC4 key (footer)	FBI (3 bytes)	Encrypts 56-byte metadata appended to each file
RC4 key (note)	hQEPINuTY2lytufLDpJQdVkrWlovC1QR (32 bytes)	Decrypts ransom note content
ChaCha20 constant	expand 32-byte k	Standard ChaCha20 block constant

ChaCha20 Identification

Confirmed by quarter-round rotation constants at `0x401A91` : - ROL 16, ROL 12, ROL 8, ROL 7 →

ChaCha20 (Salsa20 uses 7, 9, 13, 18) - 10 double-rounds (20 rounds total) - 64-byte output blocks - "expand 32-byte k" sigma constant

Encryption Strategy

- **Threshold:** Files > 2 GB get intermittent encryption (1/5 of file)
- **Block size:** 1 MB read/write chunks
- **Footer:** 7-byte marker "payload" + 56-byte RC4-encrypted session data
- **Extension:** `.payload`
- **In-place:** file content overwritten, no temp files

File Metadata Footer (63 bytes)

```
[original file data (partially encrypted)]  
["payload" marker – 7 bytes]  
[RC4-encrypted metadata – 56 bytes: ephemeral X25519 public key (32B) + nonce (12B) + padding (12B)]
```

6. File Targeting

Directory Exclusion List (18 directories)

Directory
AppData
Boot
Windows
windows.old
Tor Browser
Internet Explorer
Google
Opera
Opera Software
Mozilla
Mozilla Firefox
\$Recycle.Bin
ProgramData
All Users
#recycle
\$WinREAgent
.vs
WindowsPowerShell

File Exclusion List (27 filenames)

autorun.inf, boot.ini, bootfont.bin, bootsect.bak, bootmgr, bootmgr.efi, bootmgfw.efi, desktop.ini, iconcache.db, ntldr, ntuser.dat, ntuser.dat.log, ntuser.ini, thumbs.db, ntuser.dat.log1, ntuser.dat.log2, pagefile.sys, hiberfil.sys, swapfile.sys, recovery.ini, boot.sdi, setup.ini, config.sys, system.ini, win.ini, ntbootdd.sys, bootstat.dat

Extension Whitelist (55 extensions skipped)

.exe, .dll, .dll.mui, .sys, .drv, .efi, .cpl, .ocx, .ax, .mui, .msc, .bat, .cmd, .ps1, .ps1w, .psm1, .psd1, .vbs, .js, .jse, .ws, .ms, .blf, .lnk, .url, .library-ms, .search-ms, .searchconnector-ms, .ttf, .otf, .fon, .fnt, .wim, .vhd, .vhdx, .bcd, .sdi, .boot, .cur, .ani

These are system/script/font/boot files — the ransomware avoids encrypting them to keep the OS bootable.

7. Recovery Inhibition

Shadow Copy Deletion — `delete_shadow_copies` @ `0x40A158`

```
ShellExecuteW(NULL, "open", "cmd.exe", "/c vssadmin.exe delete shadows /all /quiet", NULL, SW_HIDE);
```

Uses `Wow64DisableWow64FsRedirection` before and `Wow64RevertWow64FsRedirection` after to ensure the 64-bit `vssadmin.exe` is invoked even from a WoW64 process.

Recycle Bin Emptying

```
SHEmptyRecycleBinA(NULL, NULL, SHERB_NOCONFIRMATION | SHERB_NOPROGRESSUI | SHERB_NOSOUND);
```

8. Services Killed (44)

Service	Target
vss	Volume Shadow Copy
sql	SQL Server
svc\$	SQL-related services
mementas, mepocs	MailEnable
sophos (×2)	Sophos AV
veeam (×2)	Veeam Backup
backup	Generic backup
GxVss, GxBlr, GxFWD, GxCVD, GxCIMgr	Commvault
DefWatch, ccEvtMgr, ccSetMgr, SavRoam, RTVscan	Symantec/ Norton
QBFCService, QBIDPService, Intuit.QuickBooks.FCS, QBCFMonitorService	QuickBooks
YooBackup, YooIT	Yoo Backup
zhudongfangyu	Qihoo 360
stc_raw_agent	StorageCraft
VSNAPVSS	Virtual Snap
VeeamTransportSvc, VeeamDeploymentService, VeeamNFSSvc	Veeam
PDVFSService	PD File Services
BackupExecVSSProvider, BackupExecAgentAccelerator, BackupExecAgentBrowser, BackupExecDiveciMediaService, BackupExecJobEngine, BackupExecManagementService, BackupExecRPCService	Veritas Backup Exec
AcrSch2Svc, AcronisAgent	Acronis
CASADWebSvc, CAARCUdateSvc	Arcserve

Implementation at `kill_services @ 0x40C04D` : opens SCManager with full access, iterates service list, enumerates and stops dependent services first, then stops each service with a 30-second timeout.

9. Processes Killed (31)

Process	Target
sql.exe	SQL Server
oracle.exe	Oracle DB
ocssd.exe	Oracle CSS Daemon
dbsnmp.exe	Oracle SNMP
synctime.exe	Sync utility
agntsvc.exe	SQL Agent
isqlplussvc.exe	Oracle iSQL*Plus
xfssvccon.exe	Oracle XFS
mydesktopservice.exe	MyDesktop
ocautoupds.exe	Oracle Auto Update
encsvc.exe	Encryption service
firefox.exe	Mozilla Firefox
tbirdconfig.exe	Thunderbird Config
mydesktopqos.exe	MyDesktop QoS
ocomm.exe	Oracle Communicator
dbeng50.exe	SQL Anywhere
sqbcoreservice.exe	SQL Backup Core
excel.exe	Microsoft Excel
infopath.exe	Microsoft InfoPath
msaccess.exe	Microsoft Access
mspub.exe	Microsoft Publisher
onenote.exe	Microsoft OneNote
outlook.exe	Microsoft Outlook
powerpnt.exe	Microsoft PowerPoint
steam.exe	Steam
thebat.exe	The Bat!
thunderbird.exe	Mozilla Thunderbird
visio.exe	Microsoft Visio
winword.exe	Microsoft Word
wordpad.exe	Windows WordPad
notepad.exe	Notepad

Implementation at `kill_processes @ 0x40C29C`: `CreateToolhelp32Snapshot` → `Process32FirstW/NextW` → `TerminateProcess(pid, 9)`.

10. Persistence & Evasion

ETW Bypass — `patch_etw_functions @ 0x4092D2`

Patches 4 ETW functions in `ntdll.dll` to disable Event Tracing for Windows: - `EtwEventWrite` - `EtwEventWriteFull` - `EtwEventWriteTransfer` - `EtwRegister`

For each function: 1. `GetModuleHandleA("ntdll.dll")` → `GetProcAddress(func_name)` 2. `VirtualProtect(PAGE_EXECUTE_READWRITE)` 3. Write return stub: - WoW64 (32-bit on 64-bit): `0xC3C3C3C8` (4 bytes) - Native x86: `0x14C2C033` → `xor eax, eax; ret 0x14` 4. `VirtualProtect` (restore original protection) 5. `FlushInstructionCache`

Self-Deletion via ADS Rename — `self_delete_via_ads_rename @ 0x405C7A`

1. `GetModuleFileNameW` → own executable path
2. `CreateFileW(DELETE|SYNCHRONIZE)`
3. `SetFileInformationByHandle(FileRenameInfo)` → rename to `:payload` (NTFS Alternate Data Stream)
4. `CloseHandle`
5. `CreateFileW` again (now named `:payload`)
6. `SetFileInformationByHandle(FileDispositionInfo, DELETE)` → mark for deletion
7. `CloseHandle` → file is deleted on close

This technique bypasses file locks because the rename happens while the process is still running.

NT Native API Usage — `resolve_nt_native_apis @ 0x408D0F`

All file I/O is performed via NT native APIs resolved from `ntdll.dll` at runtime:

API	Purpose
<code>NtCreateFile</code>	Open files for encryption
<code>NtReadFile</code>	Read file content
<code>NtWriteFile</code>	Write encrypted content
<code>NtSetInformationFile</code>	Rename files (add extension)
<code>NtQueryInformationFile</code>	Get file size
<code>NtQueryDirectoryFile</code>	Directory enumeration
<code>NtQuerySystemInformation</code>	System information
<code>NtQueryInformationProcess</code>	Process information
<code>NtOpenSymbolicLinkObject</code>	Drive enumeration
<code>NtQuerySymbolicLinkObject</code>	Resolve drive targets
<code>NtQueryDirectoryObject</code>	Directory objects
<code>NtClose</code>	Close handles
<code>RtlInitUnicodeString</code>	Initialize UNICODE_STRING

This bypasses user-mode API hooks from security products that only hook kernel32/kernelbase.

Mutex

`MakeAmericaGreatAgain` — prevents multiple instances. The `--m` flag disables this check.

Background Re-launch

If not started with `--background`, the ransomware re-launches itself with the `--background` flag for detached execution, then the parent exits.

Shutdown Priority

`SetProcessShutdownParameters(0, 0)` — ensures the process is the last to be shut down during system shutdown.

11. CLI Arguments

Flag	Global	Effect
<code>--log</code>	—	Enable logging to <code>\\?\C:\payload.log</code>
<code>--p <path></code>	—	Target specific path instead of all drives
<code>--algo <name></code>	—	Force crypto variant: <code>avx2</code> , <code>sse2</code> , <code>default</code>
<code>--threads <n></code>	<code>NumberOfConcurrentThreads</code>	Override worker thread count
<code>--background</code>	<code>byte_45F6F1</code>	Run as detached background process
<code>--m</code>	<code>byte_45F6E7</code>	Skip mutex creation (allow multiple instances)
<code>--n</code>	<code>byte_45F6E6</code>	Skip ransom note writing
<code>--d</code>	<code>byte_45F6E5</code>	Skip auto-deletion (keep executable)
<code>--k</code>	<code>byte_45F6F2</code>	Skip process/service killing
<code>--s</code>	<code>byte_45F6F0</code>	Unknown (unused in decompiled flow)
<code>--bypass-etw</code>	<code>byte_45F6F3</code>	Patch ETW functions in ntdll
<code>--l</code>	<code>byte_45F704</code>	Delete event logs
<code>--i</code>	<code>byte_45F6E4</code>	Ignore all file filters (dangerous mode)

12. IOCs

Hashes

Type	Value
SHA-256	<code>1ca67af90400ee6cbbd42175293274a0f5dc05315096cb2e214e4bfe12ffb71f</code>
MD5	<code>e0fd8ff6d39e4c11bdaf860c35fd8dc0</code>

Network

Type	Value
DLS Onion	<code>payloadrz5yw227brtbvdqpnlhq3rdcdekdn3rgucbcdeawq2v6vuyd.onion</code>
Chat Onion	<code>payloadynyvabjacbun4uwmx7yvdzorycsLzmnleguxjn7glahsvqd.onion</code>

Files

Path/Pattern	Description
<code>RECOVER_payload.txt</code>	Ransom note
<code>*.payload</code>	Encrypted files
<code>\\?\C:\payload.log</code>	Debug log file

Mutex

`MakeAmericaGreatAgain`

Curve25519 Public Key

`687f536dd73ea8f71090fc2172568d60569d845d381b3b86b2e4710db28c4645`

Distinctive Strings

- `"MakeAmericaGreatAgain"` (mutex name)
 - `"expand 32-byte k"` (ChaCha20 constant)
 - `"payload"` (file footer marker)
 - `"FBI"` (RC4 key for footer encryption)
 - `"RECOVER_payload.txt"` (note filename)
 - `"SIZE OF PUBKEY IS LOWER THAN NEED (32)"` (pubkey validation)
 - `"[CPU] AVX2 found" / "[CPU] SSE2 found"` (CPU feature logging)
 - `"[Mutex] locker running.."` (instance check)
 - `"bypass-etw"` (CLI flag for ETW patching)
-

13. MITRE ATT&CK Techniques

ID	Technique	Implementation
T1486	Data Encrypted for Impact	Curve25519 + ChaCha20, .payload extension, intermittent >2GB
T1490	Inhibit System Recovery	vssadmin delete shadows, SHEmptyRecycleBinA
T1489	Service Stop	44 services stopped via SCM (backup, AV, database)
T1057	Process Discovery	CreateToolhelp32Snapshot + Process32FirstW/NextW
T1562.001	Impair Defenses: Disable or Modify Tools	ETW bypass: patches EtwEventWrite/Register in ntdll
T1070.004	Indicator Removal: File Deletion	Self-delete via NTFS ADS rename (:payload stream)
T1106	Native API	14 NT native APIs resolved from ntdll.dll at runtime
T1083	File and Directory Discovery	Recursive traversal via NtQueryDirectoryFile
T1082	System Information Discovery	CPUID for AVX2/SSE2, GetLogicalProcessorInformation
T1027	Obfuscated Files or Information	RC4-encrypted strings, double-base64 note key
T1480	Execution Guardrails	Mutex "MakeAmericaGreatAgain", extensive CLI flag system

14. Comparison: Windows vs ESXi Variant

Feature	Windows (this sample)	ESXi (<code>bed8d175...</code>)
File type	PE32 x86 console	ELF x86-64
Size	385 KB	39 KB
Language	C++ (MSVC)	C
File encryption	ChaCha20 (AVX2/SSE2/scalar)	ChaCha20 (AVX2) + Salsa20 (SSE2)
Key exchange	Curve25519	Curve25519
Footer encryption	RC4 (key "FBI")	RC4 (key "FBI")
Note encryption	RC4 (32-byte key, double-base64)	RC4 (32-byte key, base64)
Extension	<code>.payload</code>	<code>.xx0001</code>
Note	<code>RECOVER_payload.txt</code>	<code>welcome.txt</code> (ESXi web UI)
Threshold	2 GB (intermittent 1/5)	5 GB (5-pass partial)
Threading	IOCP completion port	POSIX thread pool
File I/O	NT native APIs (ntdll)	POSIX (fopen/fread/fwrite)
Anti-debug	IsDebuggerPresent (import)	TracerPid (/proc/self/status)
ETW bypass	Yes (patches ntdll)	N/A
Self-deletion	ADS rename trick	No
Services killed	44	N/A
Processes killed	31	N/A
Shadow deletion	vssadmin	N/A
Mutex	MakeAmericaGreatAgain	flock()
Infrastructure	Same onion addresses	Same onion addresses

15. Summary

Payload (Windows) is a sophisticated, MSVC-compiled ransomware targeting Windows workstations and servers. It implements a **Curve25519 + ChaCha20** hybrid encryption scheme with **IOCP-based multi-threading** and **three CPU-optimized cipher paths** (AVX2, SSE2, scalar). The binary uses **NT native APIs** for all file operations to bypass user-mode hooks, features **ETW patching** to blind EDR telemetry, and employs **NTFS ADS rename** for self-deletion. Recovery is inhibited through **shadow copy deletion, recycle bin emptying**, and aggressive **service/process termination** targeting backup, database, and office applications.

The same actor operates both a Windows and ESXi/Linux variant sharing identical infrastructure, encryption architecture (Curve25519 + ChaCha20 + RC4 "FBI"), and ransom note template. The Windows variant is significantly more complex (385 KB vs 39 KB, 2329 vs 180 functions) with additional evasion and anti-forensic capabilities.