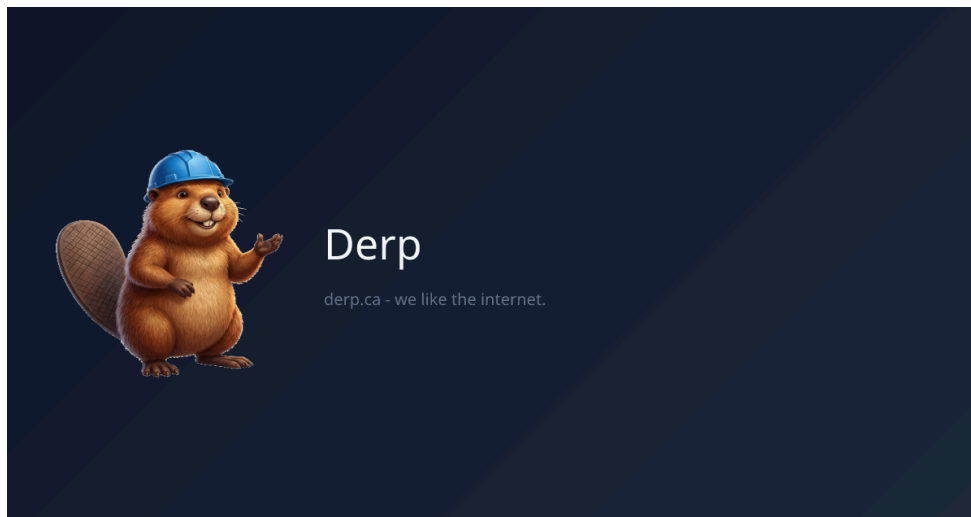


Payload ransomware group: mutex MakeAmericaGreatAgain

 derp.ca/research/payload-ransomware-babuk-derivative

Kirk

March 15, 2026



`malware ransomware babak reverseengineering yara encryption`

On March 15, the Payload ransomware group [claimed a breach of Royal Bahrain Hospital](#), posting 110 GB of allegedly stolen data with a March 23 deadline. The hospital joins 11 other victims on the group's Tor leak site, bringing the total claimed exfiltration to 2,603 GB across 7 countries. The targets have been mid-to-large organisations in real estate, energy, healthcare, telecom, and agriculture, primarily in emerging markets. Payload has been active since at least February 17, 2026. The Windows binary was compiled that same day, and the first victim appeared on the leak site within hours.

We reversed the Windows binary completely. Every code path, every crypto primitive, every command-line switch. The encryption uses Curve25519 key exchange paired with ChaCha20, and the per-file private keys are securely erased from memory after each file is locked. Without the operator's private key, encrypted files are unrecoverable. There is no backdoor in the crypto, no implementation flaw, no shortcut.

One detail stood out during analysis. Each encrypted file has a 56-byte footer that is itself RC4-encrypted. The key for that RC4 encryption is three bytes long: **FBI**. It sits right next to the ChaCha20 constant in the binary's data section, which makes them look like a single string: **expand 32-byte kFBI**. The Linux variant has a similar accident, where **FBI** runs into a thread pool format string to produce **FBIthread-pool-%d**. These are not modified cryptographic constants. They are just neighbouring strings in memory. But they make for distinctive detection signatures.

Seventeen VirusTotal engines detect the binary as Babuk. The encryption scheme, kill lists, and binary structure are consistent with the leaked Babuk builder from September 2021. Some reporting refers to Payload as ransomware-as-a-service, but no public source provides evidence of an affiliate programme or builder panel. We treat that label as unverified. What we can verify is dual-platform coverage across Windows and ESXi, a working double-extortion operation, and a mutex named **MakeAmericaGreatAgain**.

What Payload ransomware does

Payload is offline ransomware. There is no C2 communication. The binary encrypts local and network drives, appends a 56-byte footer to each file, renames files with a **.payload** extension, drops a ransom note, and deletes its own executable. It stops backup and security services, kills processes that hold file locks, and deletes shadow copies. It also empties the recycle bin and optionally wipes all Windows event logs and patches ETW to blind EDR.

The ransom note directs victims to a Tor negotiation portal with per-victim credentials. A separate Tor leak blog publishes stolen data on countdown timers. Payment is negotiated exclusively through the portal. No wallet address appears in the note.

Sample overview

Field	Windows PE	Linux ELF
SHA256	1ca67af90400ee6cbdd42175293274a0f5dc05315096cb2e214e4bfe12ffb71f	bed8d1752a12e5681412efbb8283910857f7c5c431c
MD5	e0fd8ff6d39e4c11bdaf860c35fd8dc0	f91cbdd91e2daab31b715ce3501f5ea0
Type	PE32 (console) Intel 80386, MSVC	ELF 64-bit x86-64, dynamically linked, stripped
Size	394,752 bytes	39,904 bytes
Compiled	2026-02-17 08:39:07 UTC	Stripped (VT first seen 2026-02-17)
Imphash	51da4b9aa541a6fc636a97d44ee265b4	--
Platform	Windows (MSVC, VS2019 16.11)	GNU/Linux 2.6.32 (libxml2, libpthread)
Mutex	MakeAmericaGreatAgain	None
VT detection	57/76	8/76
Sandbox	Triage	Triage

The Windows binary is 10x larger than the Linux variant. Static linking of the MSVC CRT and Concurrency Runtime accounts for most of the difference, along with the service/process kill lists, event log wiper, and ETW patcher, none of which exist in the Linux build.

Encryption scheme

Payload uses Curve25519 ECDH for key agreement and ChaCha20 for file encryption. The Curve25519 implementation is curve25519-donna, identified by the Montgomery ladder pattern and the constant 121665 (0x1DB41) used in fe_mul121666. ChaCha20 is confirmed by the quarter-round rotation constants 16, 12, 8, 7 (Salsa20 uses 7, 9, 13, 18).

The per-file encryption flow:

1. **Generate per-file keypair.** CryptGenRandom produces 32 random bytes for a Curve25519 private key (clamped: low 3 bits cleared, high bit cleared, bit 254 set) and 12 random bytes for a ChaCha20 nonce.
2. **Compute per-file public key.** Scalar multiplication with the Curve25519 basepoint.
3. **ECDH shared secret.** shared_secret = Curve25519(per_file_private, operator_public). The operator's public key is decoded from base64 at startup. The shared secret is used directly as the ChaCha20 key with no hashing step.
4. **Encrypt file.** ChaCha20 in 1 MB chunks: NtReadFile -> encrypt -> seek back -> NtWriteFile. Files over 2 GB get partial encryption: only 20% of the file is encrypted (file_size / 5), with 1 MB chunks evenly spaced. The CPU dispatch selects between a scalar path and an SSE2 4-block parallel path. The binary labels the SSE2 path "AVX2" in its log messages, but the code uses xmm0-xmm7 and movaps with no ymm registers.
5. **Append footer.** 56 bytes, RC4-encrypted with the 3-byte key FBI, appended to the file.
6. **Erase per-file private key.** The private key buffer and public key buffer are zeroed byte-by-byte on the stack immediately after the footer is written. The per-file private key never reaches disk and exists in memory only for the duration of one file's encryption.

The operator's decryptor reads the last 56 bytes of a .payload file, RC4-decrypts with FBI, extracts the per-file public key and nonce, computes the shared secret using the operator's private key, and ChaCha20-decrypts the file body.

Without the operator's Curve25519 private key, the shared secret cannot be derived. Files are unrecoverable.

The "FBI" footer key

The `.rdata` section at file offset `0x4F560` contains a contiguous block of crypto material:

<code>0x4F560: "expand 32-byte k"</code>	ChaCha20 sigma constant (16 bytes)
<code>0x4F570: "FBI\0"</code>	RC4 key for file footer (3 bytes + null)
<code>0x4F574: "aH9Tbdc+qPcQ..."</code>	Operator Curve25519 pubkey (base64, 44 bytes)
<code>0x4F5A0: "\0\0\0\0\0\0\0\0"</code>	Padding (8 bytes)
<code>0x4F5A8: "I1hBex9EMIS7..."</code>	RC4-encrypted ransom note (base64, 2528 bytes)

The footer structure, verified from an encrypted `.payload` file:

Offset	Size	Content
0-31	32 bytes	Per-file Curve25519 public key
32-43	12 bytes	ChaCha20 nonce (from CryptGenRandom)
44-47	4 bytes	Uninitialized stack residue (not used by decryptor)
48-55	8 bytes	<code>payload\0</code> branding marker

The 4 bytes at offset 44-47 are stack data from `[ebp-0xE4]`. The buffer between the nonce and the branding string is never explicitly written. This leaks 4 bytes of stack per encrypted file. It has no cryptographic significance.

A 3-byte RC4 key provides no meaningful protection for the footer. The per-file public key it contains is useless without the operator's Curve25519 private key. The footer encryption obscures metadata, not secrets.

Ransom note decryption

The ransom note is embedded in `.rdata` as an RC4-encrypted, base64-encoded blob. The RC4 key is stored separately at VA `0x452DC8` as a base64 string that decodes to the 32-byte ASCII key `hQEPINuTY2lytufLDpJQdVkrwlovC1QR`. Both the KSA (VA `0x4093A9`) and PRGA (VA `0x409404`) are textbook RC4: 256-byte S-box, standard key scheduling with `div` for modular indexing.

The decrypted note reads:

```
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.
```

It provides two Tor addresses, per-victim login credentials, and a 240-hour total negotiation window. Victims can request up to 3 free file decryptions (15 MB each) as proof of capability.

Field	Value
Negotiation portal	<code>payloadynyvabjacbun4uwmx7yvdzorycs1zmnleguxjn7g1ahsvqd.onion</code>
Leak blog	<code>payloadr25yw227brtbvdqpn1hq3rdcdekdn3rgucbcdeawq2v6vuyd.onion</code>

Both Tor sites were live as of March 15, 2026. The leak blog runs nginx with hardened security headers (CSP, X-Frame-Options DENY, no-referrer). The negotiation portal exposes no Server header. Different backends.

Anti-forensics

ETW patching

The `bypass-etw` flag patches four functions in `ntdll.dll`: `EtwEventWrite`, `EtwEventWriteFull`, `EtwEventWriteTransfer`, and `EtwRegister`. The patch overwrites each function prologue after making the page writable with `VirtualProtect`.

Patch bytes:

- **WoW64 (32-on-64):** `48 33 C0 C3` = `xor rax, rax; ret`
- **Native x86:** `33 C0 C2 14 00` = `xor eax, eax; ret 0x14`

Both patches cause the function to return `STATUS_SUCCESS` immediately, dropping all ETW trace events. EDR that relies on ETW for process-level visibility goes blind.

Event log wipe

The `l` flag triggers a full wipe of every Windows event log channel. The binary loads `wevtapi.dll` at runtime, enumerates all channels via `EvtOpenChannelEnum` / `EvtNextChannelPath`, and calls `EvtClearLog` on each one. Application, Security, System, and every custom operational channel. All cleared. Function pointers are zeroed and the DLL is unloaded after the wipe.

Self-deletion via NTFS ADS

The binary renames its own `$DATA` stream to an Alternate Data Stream named `:payload` (e.g., `malware.exe:payload`), then reopens and marks the file for deletion on close. The ADS rename releases the file lock that Windows holds on running executables, allowing the delete to succeed without spawning `cmd.exe /c del` or writing a temp batch file. No child processes. The `:payload` stream name matches the variant branding.

CLI arguments

Switch	Feature	Default
<code>m</code>	Disable mutex creation	OFF
<code>n</code>	Disable ransom note	OFF
<code>d</code>	Disable self-deletion	OFF
<code>k</code>	Disable process/service termination	OFF
<code>s</code>	Skip network shares	OFF
<code>l</code>	Enable event log deletion	OFF
<code>i</code>	Ignore file/extension filters	OFF
<code>bypass-etw</code>	Patch ETW tracing	OFF
<code>background</code>	Detach console, run in background	OFF
<code>--background</code>	Long-form background flag	OFF
<code>algo</code>	Select encryption path (<code>avx2</code> , <code>sse2</code> , <code>default</code>)	Auto-detect
<code>threads</code>	Encryption thread count	CPU core count

The `i` flag bypasses all file and extension filters. The binary warns internally: "filtering for the note file and files with the `.payload` extension is disabled, which may result in them being processed again."

Kill lists

The binary stops services via the SCM API and kills processes via toolhelp snapshots.

Services stopped (34 total): backup infrastructure (Veeam, Acronis, BackupExec, YooBackup), AV/security (Symantec/Veritas, Sophos, Qihoo 360), and financial software (4 Intuit QuickBooks services).

Processes killed (31 total): database engines (`sql.exe`, `oracle.exe`, `dbsnmp.exe`, `dbeng50.exe`), Microsoft Office suite (`excel.exe`, `winword.exe`, `outlook.exe`, `powerpnt.exe`, `onenote.exe`, `msaccess.exe`, `mspub.exe`, `visio.exe`, `infopath.exe`), email clients (`thunderbird.exe`, `thecat.exe`), browsers (`firefox.exe`), and other file-locking applications (`steam.exe`, `notepad.exe`, `wordpad.exe`).

The binary also empties the recycle bin and deletes shadow copies via `vssadmin.exe delete shadows /all /quiet`.

Linux/ESXi variant

The Linux build is a 39 KB stripped ELF targeting ESXi hypervisors. It links `libxml2.so.2` and parses `/etc/vmware/hostd/vmInventory.xml` using XPath to locate VM disk paths for targeted encryption. The thread pool uses a C-based thpool library with `prctl` thread naming (`thread-pool-%d`). Anti-debugging reads `/proc/self/status` and checks `TracerPid:` for a non-zero value.

The core crypto is identical: Curve25519 + ChaCha20 with the `FBI` RC4 footer key. The operator public keys differ between the Windows and Linux builds, confirming per-campaign or per-affiliate key generation. The adjacent-string artifact `FBIthread-pool-%d` in the Linux `.rodata` mirrors the Windows `expand 32-byte kFBI` artifact.

Missing from the Linux build: no service/process killer, no shadow copy deletion, no event log wiper, no ETW patcher, no mutex, no self-deletion, no verbose logging framework.

Feature	Windows	Linux/ESXi
Encryption	ChaCha20 + Curve25519	ChaCha20 + Curve25519
Footer RC4 key	FBI	FBI
Operator pubkey	aH9Tbdc+qPcQ...	Pmep+UUmeFwx...
VM targeting	--	libxml2 vmInventory.xml parsing
Thread pool	MSVC Concurrency Runtime	C thpool + prctl
RNG	CryptGenRandom	/dev/urandom
Anti-debug	GetTickCount timing	TracerPid check
Self-delete	NTFS ADS rename	--
Size	395 KB	40 KB

IOC summary

Network

Type	Value	Context
Onion	<code>payloadynyvabjacbun4uwvmxc7yvdzorycs1zmnleguxjn7glahsvqd.onion</code>	Negotiation portal
Onion	<code>payloadrz5yw227brtbvdqpnlhq3rdcdekdn3rgucbcdeawq2v6vuyd.onion</code>	Leak blog

Host

Type	Value	Context
Mutex	<code>MakeAmericaGreatAgain</code>	Single-instance lock
Extension	<code>.payload</code>	Encrypted file extension
Filename	<code>RECOVER_payload.txt</code>	Ransom note
Filename	<code>RECOVERY-xx0001.txt</code>	Ransom note (alternate builds)
Log file	<code>C:\payload.log</code>	Execution log
Footer marker	<code>payload\0</code> (8 bytes)	Last 8 bytes of encrypted file footer

Behavioural

Indicator	Description
NTFS ADS rename to <code>:payload</code>	Self-deletion via ADS stream rename
<code>vssadmin.exe delete shadows /all /quiet</code>	Shadow copy deletion
<code>SHEmptyRecycleBinA</code>	Recycle bin emptied
ETW patch: <code>48 33 C0 C3</code> (WoW64) / <code>33 C0 C2 14 00</code> (x86)	ETW function neutering
<code>EvtClearLog</code> on all channels	Full event log wipe
Per-file private key zeroed after use	Secure key erasure

Babuk lineage

We compared the binary against the [leaked Babuk source code](#) (September 2021). The service kill list (41 services) and process kill list (31 processes) are character-identical. The Curve25519-donna implementation uses the same clamping logic and the same constant 121665 (`0x1DB41`). Shadow copy deletion is the same `vssadmin.exe` command. The directory and file exclusion lists share a common core.

The operator made substantial changes on top of this base:

Feature	Babuk (2021)	Payload (2026)
Symmetric cipher	HC-128	ChaCha20
Key derivation	SHA-512 of shared secret	Shared secret used directly
Footer	104 bytes, plaintext, CRC32	56 bytes, RC4(FBI), payload\0 marker
CLI	Named args (debug, shares, paths)	12 switches including single-letter flags
Mutex	DoYouWantToHaveSexWithCuongDong	MakeAmericaGreatAgain
Extension	.babyk	.payload
Note storage	Plaintext, builder-patched	RC4-encrypted, base64-encoded in .rdata
Large file threshold	Multi-tier (20 MB / 5 MB)	Single threshold (2 GB, encrypt 20%)
Self-delete	Not present	NTFS ADS rename to :payload
ETW bypass	Not present	Patches 4 ntdll functions
Event log wipe	Not present	Enumerates and clears all channels

The cipher swap from HC-128 to ChaCha20 may have been borrowed from Babuk’s own NAS variant, which already used ChaCha20 in the leaked source. The anti-forensic additions (ETW, event logs, ADS self-delete) and the granular CLI flags are entirely new.

The ClamAV signature Win.Ransomware.Babuk-10032520-1 matches 154 samples across 8 operations sharing this codebase: RAWorld (143), Babuk original (4), Nitrogen (2), Payload (1), SchoolBoy, Neshta, and Cylan.

YARA rules for both the Windows PE and Linux/ESXi ELF are available at .

Kirk

I like the internet. Want to get in touch? ·