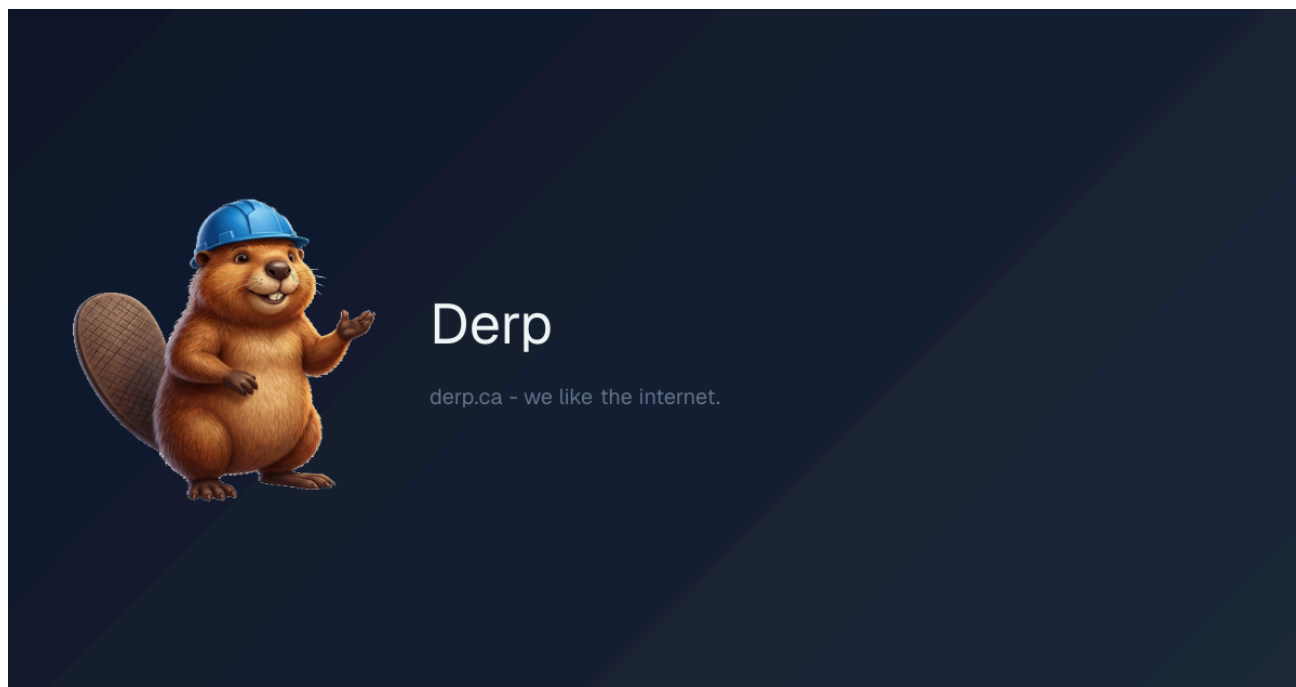


M3rx ransomware: inside a new leak-site actor and Go encryptor

 derp.ca/research/m3rx-ransomware-go-encryptor

Kirk

April 27, 2026



malware ransomware m3rx reverse-engineering encryption go

M3rx is a new ransomware name with a leak site, a Tox contact, and a Windows encryptor that is already doing real work.

[RansomLook](#) lists [M3rx](#) with six public posts as of April 27, 2026. One appeared on April 23. Five more appeared on April 26. [PurpleOps](#) used the spelling [M3RXDLS](#) in its April 26 ransomware activity report, and the victim set lines up with the M3rx leak-site entries. I am treating M3rx, M3RX, and M3RXDLS as labels for the same observed activity unless better attribution appears.

We pulled apart a PE32+ x64 Go sample tied to that contact footprint. The binary was submitted to Triage on 2026-04-25 as task [260425-gayffagw3x](#). It carries an embedded config, writes **RECOVERY_NOTES.TXT**, renames encrypted files to random 16-character names with the **.8hm1sewu** extension, empties the Recycle Bin, and deletes itself through PowerShell at the end of execution.

The crypto and file format are worth the teardown. M3rx uses per-run X25519 key exchange, AES-CTR for file content, AES-GCM to wrap each per-file AES key, and a fixed **0x400**-byte footer. The config is recoverable. The footer is readable. Interrupted files can also expose a different state than completed files.

The short version: M3rx is new, the public trail is still thin, and the encryptor already has a recognizable file format.

If you operate a threat intelligence platform with API access and can provide a researcher account, please reach out to . Additional data sources directly increase the quality and coverage of the threat intel published here.

A new public name

RansomLook's group page showed six M3rx posts as of April 27, 2026. The April 26 batch spans the US, Canada, Australia, the UK, and Switzerland in the actor's own claims.

Date	Claimed victim	Public claim
2026-04-23	rotak.it	no leak size listed on the group table
2026-04-26	rainforestclean.com	259 GB, 77k files
2026-04-26	airdriephysio.com	54 GB, 116k files
2026-04-26	primeproperties.com.au	100 GB, 81k files
2026-04-26	anvilarts.org.uk	480 GB, 299k files
2026-04-26	dmschweiz.ch	120 GB, 100k files

Those numbers are actor-posted claims. They still matter because they show timing and shape: a new name, a short public burst, and contact infrastructure that also appears in the ransomware note.

The contact footprint is also small:

Type	Value
Leak site	4k6plf4h2cm2nco6ae3inrsxnmqgl6l1lmwefydhnlcq4tuhwbj4qpad[.]onion
Chat / client area	pippahtohg6qgioqu3ixrsueefuw7thythmmeanrgwn3eixcuu6jvqd[.]onion
Ransom-note path token	/WAJCTZ6F0JF75KB5XVRYBFERX6

Type	Value
Tox ID	9A1217BEDA4AB77052A25D17CB6FFB34AFA2BE462E607F2FD8E1DF1DDD4CA16A64E18B1A0BF2

The note says files were stolen and encrypted, asks for Bitcoin after negotiation, offers to decrypt three files as proof, and threatens publication plus sharing with competitors. It reads like standard double-extortion copy, but the contact anchors are useful for correlation.

The Windows sample

Field	Value
Sample	b09ece33ffe5efb1903526229595a8c74d983c731505bee09c2a005036c834b8.exe
Triage task	260425-gayffagw3x
Submitted	2026-04-25T05:36:42Z
SHA256	b09ece33ffe5efb1903526229595a8c74d983c731505bee09c2a005036c834b8
SHA1	521b1bd3f30ca50eaae6f74718b97dbe8a49c245
MD5	071e2e0087554d96bba6a4ab73d88cd0
Type	PE32+ Windows x64 Go executable
Size	2,580,480 bytes
Go version	go1.20.14
Build flag	-trimpath=true
Encrypted extension	.8hmlsewu
Ransom note	RECOVERY_NOTES.TXT

Observed host behaviors:

Area	Behavior
File impact	encrypted <code>.8hmlsewu</code> files with random 16-character lowercase alphanumeric basenames
Note dropping	<code>RECOVERY_NOTES.TXT</code> in enumerated <code>Desktop</code> and <code>Documents</code> directories
Recovery disruption	Recycle Bin clearing via <code>SHEmptyRecycleBin</code>
Cleanup	PowerShell self-delete loop unless keep behavior is selected
File access	Restart Manager strings and imports for unlocking held files

Config hidden in plain Go

M3rx stores its operator material in a `0x1000` blob in `.rdata` at virtual address `0x55ecbf`. The blob is not directly gzip on disk. It becomes gzip after prepending:

```
1f 8b 08 00 00 00 00 00 ff
```

After decompression, the payload is Go gob data. The decoded struct shape is `EC{K1, EX, NN, NT}`.

Field	Decoded value
K1	<code>cdbe4aed37c98d67a005ef469e7e0586e0ff8973b91a8d577d320e67cf46b572</code>
EX	<code>.8hmlsewu</code>
NN	<code>RECOVERY_NOTES.TXT</code>
NT	1,361-byte ransom note template

The decode path is short: reconstruct the missing gzip header, decompress the body, then gob-decode the config. The recovered note contains the same Tor client area and Tox ID listed above.

The config is not clever. It is just annoying enough to make strings lie. The Go symbol table makes the rest easier: `main.load_config` at `0x517740` calls the gzip and gob paths directly.

File encryption

M3rx has two layers of key material.

Per process, `main.load_config` generates an X25519 ephemeral keypair. It copies the runtime ephemeral public key to global `0x6c80c0`, loads the operator public key from config field `K1`, performs X25519 ECDH, and copies the resulting 32-byte shared secret to global `0x6c80e0`.

Per file, `main.encode_file` generates a 16-byte IV and a 32-byte AES key. The file body is encrypted with AES-CTR. The per-file AES key is then wrapped with AES-GCM, keyed by the per-run X25519 shared secret.

Scope	Material	Where it appears
Operator	X25519 public key	config field <code>K1</code>
Process	X25519 ephemeral public key	global <code>0x6c80c0</code> , then footer <code>+0x38</code>
Process	X25519 ECDH shared secret	global <code>0x6c80e0</code>
File	AES-CTR IV	footer <code>+0x08</code>
File	AES-256 key	staging footer <code>+0x18</code> , then AES-GCM wrapped in final footer
File	Wrapped AES key	final footer <code>+0x58</code>

The final footer gives the operator enough to recover files with the private side of the X25519 exchange. Without that private side, completed files depend on the live process shared secret.

The 0x400-byte footer

Every encrypted file gets a `0x400`-byte footer. The same record moves through two states.

Offset	Size	Staging role	Final role
<code>0x000</code>	4	magic <code>0x3828ac45</code>	magic <code>0x741fbe88</code>
<code>0x004</code>	4	percentage / mode	percentage / mode

Offset	Size	Staging role	Final role
0x008	16	AES-CTR IV	AES-CTR IV; first 12 bytes reused as GCM nonce
0x018	32	plaintext per-file AES key	overwritten random bytes
0x038	32	zero before finalization	runtime X25519 public key
0x058	48	zero before finalization	AES-GCM wrapped file key
0x088	0x374	filename buffer	filename buffer
0x3fc	4	chunk counter	chunk counter

The staging state matters. If encryption is interrupted while the footer still has magic 0x3828ac45, record+0x18 can still hold the plaintext per-file AES key.

Completed files are different. The final footer magic is 0x741fbe88; record+0x18 has been overwritten; the real file key is wrapped at record+0x58.

Intermittent encryption

The default global percentage is 1. Fast mode sets it to 100.

The nominal per-1 MiB window is:

$$((\text{percentage} * 256) / 100) * 0x1000$$

At the default setting, that is 8192 bytes per 1 MiB window. Write-level capture refined the exact behavior for a 2 MiB test file: the first processed chunk read 8192 bytes, then wrote 7168 encrypted bytes, which is 8192 - 0x400. The footer chunk counter then recorded 1.

The test file makes this concrete:

Field	Value
Source path	C:\Users\michael\Documents\large-archive.zip

Field	Value
Source size	2,097,152 bytes
Source pattern	repeated 0x4d bytes
Encrypted path	c:\users\michael\documents\izzdoda5cbv9yck0.8hmlsewu
Final encrypted SHA256	fa410423b2982a435bc488aa652a96c4fe65dad66313378ca7c14bec23697327
Final footer SHA256	194086c3836c768a871d9998fccbed7ef73fcc5f3fbd541720b52205c774c735
Footer magic	0x741fbe88
Chunk counter	1

This is not victim data. It gives byte-accurate proof of the encryption schedule and footer behavior.

Recovery notes

During analysis, the runtime ECDH shared secret was captured from global 0x6c80e0:

```
ce1a0de9338a3aeb622ebaf27d4b73def4fcdd203e684084b5da8280357c3b4f
```

Using that shared secret, the final footer's AES-GCM field unwrapped the per-file AES key:

```
1c648500122bb140d0857c15e3af92a1a3f3084e9f7247c8c21fc406a384136f
```

The validation script then reversed the observed AES-CTR body span and restored the test file:

Field	Value
Decrypted output	restored large-archive.zip
Output size	2,097,152 bytes

Field	Value
Output SHA256	34af56de4c2b7216ce832be471c791eb350248683cb91924eefdcfc67738f296
Validation	every byte is 0x4d

That result proves the mechanism. It depends on capturing the process secret before it disappears, so it is a narrow recovery condition rather than a general decryptor.

The staging case is simpler. A partial file with footer magic 0x3828ac45 may still contain its plaintext per-file AES key at record+0x18.

Scope

The sample shows the file-impact side of the case: encryption, note dropping, Recycle Bin clearing, Restart Manager file unlocking, and self-delete behavior. The ransom note and leak-site posts supply the extortion claims. I did not find enough public evidence to tie M3rx to a known crew, affiliate toolkit, or specific intrusion path.

Detection Artifacts

Type	Value
Sample SHA256	b09ece33ffe5efb1903526229595a8c74d983c731505bee09c2a005036c834b8
Sample MD5	071e2e0087554d96bba6a4ab73d88cd0
Embedded config SHA256	fc18506bbbbe57fdcecaa424735705501480e6708b634457010a5cf6bdc42525
Operator public key	cdb4aed37c98d67a005ef469e7e0586e0ff8973b91a8d577d320e67cf46b572
Extension	.8hmlsewu
Ransom note	RECOVERY_NOTES.TXT

Type	Value
Staging footer magic	45 ac 28 38
Final footer magic	88 be 1f 74
Onion	pippahtohg6qgioqu3ixrsueefuw7thythmmeanrgwn3eixcuu6jvqd[.]onion
Tox ID	9A1217BEDA4AB77052A25D17CB6FFB34AFA2BE462E607F2FD8E1DF1DDD4CA16A64E18B1A0BF2
Strings	[LOCAL], complited, rstrtmgr.dll, RmShutdown, SELECT * FROM Win32_ShadowCopy

M3rx is still a fresh name. The sample gives it concrete shape: a Go binary, a recoverable gzip+gob config, a `.8hmlsewu` extension, and a final footer marked by `0x741fbe88`.

Kirk

I like the internet. Want to get in touch?