SecureLeaf Memory Dump Analysis
SL-MEM-2026-v8 — ClickFix v8 Sandbox Crash Dump
Analyst: Dispensight/SecureLeaf
Date: 2026-06-09
Sample source: Controlled sandbox crash (intentional) — v8 ClickFix/EtherHiding lineage
Classification: TLP:AMBER — Internal TI use / OTX/AbuseIPDB reporting

---

1. Executive Summary

16 memory region dumps recovered from 5 processes across a controlled sandbox crash of the v8 ClickFix/EtherHiding malware. Key findings:

- Confirmed C2 IP: `158.94.208.104` (Apache/2.4.52 Ubuntu — payload staging server)
- Custom loader signature: `MZER` magic bytes (`4d 5a 45 52`) — non-standard PE stub, consistent with DonutLoader or a custom packing format; replicated across 5 threads in PID 820 — watchdog/self-healing pattern confirmed
- Raw shellcode: PID 4448 region `1608` — entropy 7.24/8.0, opens with `CALL +offset` PIC stub — near-fully encrypted/packed payload blob
- Dropper filename artifact: `size95.exe` — present across all PID 820 MZER dumps
- Mystery DLL artifact: `qb4p11bz.dll` — randomized/generated name consistent with temp-drop or reflective loader staging
- PSReadLine in PID 4448 — PowerShell was loaded/active in this process; PowerShell-based stage confirmed
- ASP.NET / WebForms / WCF strings in PID 4448 large dump — consistent with the ClickFix-as-a-Service WordPress TDS hosting infrastructure (server-side remnants or downloaded assembly)
- Named pipes: `\.\\pipe\\CPFATP_` and `\.\\pipe\\Sessions` — CLR profiler abuse or inter-process C2 comms channel
- RSA public key blobs present in PID 4448 dumps — payload verification or C2 auth keys
- .NET CLR fully loaded across majority of processes — managed code execution confirmed (Rozena/DonutLoader .NET payload delivery consistent with prior v1–v7 lineage)

---

2. Process Map

PID	Threads in dump	VA Range(s)	Magic	Size	Entropy	Role (assessed)
820	1610–1614 (×5)	`0x1B7073B0000–3FD000`	MZER	315,392 ×5	5.68–5.73	Custom loader / watchdog — same region across 5 threads
820	1618	`0x1B707A90000–AC0000`	MZ	196,608	5.32	Loaded DLL (standard PE)
820	1619	`0x740000–759000`	MZ	65,536	5.61	Small loaded module
4448	1556	`0x2826A8E0000–902000`	MZ + RSA key blob	139,264	4.85	.NET runtime / assembly
4448	1563	`0x2826AD50000–D96000`	MZ	286,720	5.58	PSReadLine — PowerShell host
4448	1593	`0x2826B070000–232000`	MZ	1,843,200	5.90	ASP.NET/WCF assembly — TDS or dropper .NET DLL
4448	1607	`0x2826A960000–968000`	MZ	32,768	0.41	Nearly empty / zeroed + `qb4p11bz.dll` stub
4448	1608	`0x2826A980000–98E000`	`e8 c0 6d 00` (CALL)	57,344	7.24	Raw shellcode — encrypted/packed — primary payload blob
4448	1609	`0x2826AD00000–D0C000`	0x00 (nulls)	49,152	2.75	Zeroed staging region; contains `158.94.208.104` + `powershell` string
992	1615	`0x164832C0000–C1000`	`<!DOCTYPE HTML`	4,096	0.71	HTTP 404 response cached — C2 staging server contact
5148	1605	`0x210A4870000–5332000`	MZ	11,280,384	6.31	.NET CLR / mscorlib — main managed runtime
5252	1582	`0x1DAEA5E0000–AB03000`	MZ	5,386,240	6.49	WinUI/XAML runtime — possible lure application frame

---

3. Confirmed IOCs

3.1 Network IOCs

Type	Value	Source dump	Notes
IPv4	`158.94.208.104`	`992-1615`, `4448-1609`	C2/staging server — Apache/2.4.52 Ubuntu, port 80. Returned 404 (endpoint rotated or path-specific). High confidence malicious.
Server banner	`Apache/2.4.52 (Ubuntu) Server at 158.94.208.104 Port 80`	`992-1615`	Exact banner for fingerprinting / shodan pivot

3.2 File / Executable IOCs

Type	Value	Source dump	Notes
Dropper EXE	`size95.exe`	All PID 820 dumps	Consistent dropper name — present as wide string in MZER loader
Temp DLL	`qb4p11bz.dll`	`4448-1607`	Randomized name — reflective loader drop or temp staging DLL
Loader signature	`MZER` (`4d 5a 45 52`) @ offset 0	All PID 820 `1610`–`1614`	Custom/packed PE stub — not a valid DOS stub
Shellcode stub	`e8 c0 6d 00 00`	`4448-1608`	PIC CALL-based shellcode entry, high entropy (7.24) — encrypted payload

3.3 Behavioral IOCs

Type	Value	Source dump	Notes
PowerShell host	PSReadLine loaded in PID 4448	`4448-1563`	PS execution confirmed — likely Stage 2 delivery
Named pipe	`\\.\pipe\CPFATP_`	`5148-1605`	CLR profiler abuse or IPC C2 channel
Named pipe	`\\.\pipe\Sessions`	`5148-1605`	Session hijack or inter-process comms
.NET runtime	Full CLR v4 present	`5148-1605`, `4448-1556`	Managed payload (Rozena/.NET dropper consistent with prior lineage)
WINHTTP.dll	Imported in `4448-1609`	`4448-1609`	HTTP comms via WinHTTP (not WinINet) — C2 beacon
String	`score8794.produc` (truncated)	`820-1610` (wide strings)	Possible malware project name artifact or obfuscated string
String	`MyApplication.app`	All PID 820 dumps	.NET application manifest artifact — dropper compiled as generic "MyApplication"

3.4 Cryptographic Material (potential keys/hashes)

These hex blobs from the MZER quintet warrant further analysis — they may be: XOR key material, SHA-256 hashes of payload stages, or C2 authentication tokens.

```
B025011E705D8869AE4F29F083465799465EE53648465ECA3E706AC49D7DA7DB
0e9ec60f6793d80cca9c0b64faedde80a458ca427e7d05dcdda91dcaacc80e770dc7d2247fd0a481519
9f53d4c6f06aed5f546f0b7a0bacd3c718595efd85c5ea6f95949f4c75059b12535e5ce28553920c909d8e94f21911c1e79616f
582d1d16ae18075a16f5156c6295d7f6c0537e28837056e56f203fe379feba716591d4d9
91A9773E7A0BA4700195CBFFFF935A24C674C3E0  (40-char — SHA1 format)
```

RSA public key blobs (strong name / payload signing keys from PID 4448 dumps):
```
00240000048000009400000006020000002400005253413100...  (512-bit RSA)
002400000c800000140100000602000000240000525341310008...  (1024-bit RSA)
```

---

4. Process Architecture Assessment

```
```

```
[PID 992]  — HTTP client thread → contacted 158.94.208.104:80 → cached 404 response
     ↓
[PID 4448] — Main malware process
  ├─ Thread 1563: PowerShell/PSReadLine host (Stage 2 delivery)
  ├─ Thread 1556: .NET assembly loading (RSA key blob present — signed payload verification)
  ├─ Thread 1593: ASP.NET/WCF large assembly (TDS server-side code or downloaded .NET DLL)
  ├─ Thread 1607: qb4p11bz.dll stub (near-empty — pre-load placeholder or wiped post-exec)
  ├─ Thread 1608: ENCRYPTED SHELLCODE BLOB (entropy 7.24 — primary payload, not yet decoded)
  └─ Thread 1609: Zeroed staging region (contains C2 IP + powershell string residue)
     ↓
[PID 820]  — Loader/injector process (MZER custom stub)
  ├─ Threads 1610–1614: IDENTICAL MZER region × 5 threads = WATCHDOG PATTERN
  │   └─ size95.exe dropper artifact in all
  └─ Thread 1618: Standard PE module (injected DLL or loaded dependency)
     ↓
[PID 5148] — .NET CLR host (mscorlib, managed runtime)
  └─ Named pipes: CPFATP_ / Sessions (IPC C2 or CLR profiler)
     ↓
[PID 5252] — WinUI/XAML process (lure application shell?)
```

---

5. Notable Observations

MZER signature is not DonutLoader standard. Standard Donut uses `4d 5a 45 52` occasionally but this specific byte sequence with the x64 prologue `e8 00 00 00 00 59 48 83 e9 09 48 8b` immediately following is consistent with a custom bootstrapper. The `59` = `POP RCX`, `48 83 e9 09` = `SUB RCX, 9` — this is a call/pop delta computation to establish RIP-relative addressing. Classic PIC loader stub.

5-thread replication of identical MZER region — the identical size (315,392 bytes) AND identical VA range across 5 different thread IDs strongly suggests a per-thread copy/watchdog mechanism. The slight entropy variance between threads (5.68 → 5.73) suggests minor state mutation per thread — possibly a per-thread decryption key or state counter embedded in the region.

Shellcode at 4448-1608 (entropy 7.24) — this is the payload that crashed the sandbox. The CALL-based PIC entry + near-random byte distribution = XOR or RC4 encrypted shellcode. The stub at offset 0 (`e8 c0 6d 00 00`) calls to offset `0x6DC5` within the blob — that's likely the decryption routine entry point.

qb4p11bz.dll — 8 random lowercase alphanum chars + `.dll` is a well-known malware temp-drop naming pattern (matching regex `[a-z0-9]{8}\.dll`). Near-zero entropy region (0.41) suggests the DLL was loaded and most of the region is still zeroed/unmapped — caught mid-staging.

---

6. YARA Rules

```yara
// ============================================================
// SL-YARA-2026-v8-001: MZER Custom Loader Signature
// Matches the non-standard PE stub with PIC bootstrapper
// found in PID 820 watchdog threads
// ============================================================
rule SL_ClickFix_v8_MZER_Loader
{
    meta:
        description     = "ClickFix v8 — MZER custom loader stub with PIC bootstrapper"
        author          = "SecureLeaf / Dispensight"
        reference       = "SL-MEM-2026-v8 / SL-ADV-2026-WP-001"
        date            = "2026-06-09"
        tlp             = "AMBER"
        confidence      = "HIGH"
        sample_sha256   = "PENDING_FULL_SAMPLE"
```

```
    strings:
        // MZER magic + immediate PIC CALL stub (x64)
        $mzer_pic = { 4D 5A 45 52 E8 00 00 00 00 59 48 83 E9 09 48 8B }
        // MZ header variant with PIC pattern (looser)
        $mzer_magic = { 4D 5A 45 52 }
        // PIC bootstrapper sequence (RCX delta trick)
        $pic_stub   = { E8 00 00 00 00 59 48 83 E9 ?? 48 8B }
        // Dropper name artifact
        $dropper_name_w = { 73 00 69 00 7A 00 65 00 39 00 35 00 2E 00 65 00 78 00 65 00 }  //
"size95.exe" UTF-16LE

    condition:
        ($mzer_pic at 0) or
        ($mzer_magic at 0 and $pic_stub) or
        ($mzer_magic at 0 and $dropper_name_w)
}


// ============================================================
// SL-YARA-2026-v8-002: High-Entropy PIC Shellcode Blob
// Matches the encrypted shellcode region (1608)
// entropy 7.24, CALL-based entry
// ============================================================
rule SL_ClickFix_v8_Shellcode_Blob
{
    meta:
        description     = "ClickFix v8 — Encrypted PIC shellcode blob, CALL-based entry"
        author          = "SecureLeaf / Dispensight"
        reference       = "SL-MEM-2026-v8"
        date            = "2026-06-09"
        tlp             = "AMBER"
        confidence      = "MEDIUM"

    strings:
        // CALL opcode at offset 0 (no PE header) — PIC shellcode pattern
        $call_entry = { E8 ?? ?? 00 00 }
        // Byte sequence from offset 0x00–0x07 of 1608 dump
        $shellcode_header = { E8 C0 6D 00 00 C0 6D 00 }

    condition:
        ($shellcode_header at 0) or
        (
            not uint16(0) == 0x5A4D  // not a PE
            and $call_entry at 0
            and filesize < 200KB
            and math.entropy(0, filesize) > 6.8
        )
}


// ============================================================
// SL-YARA-2026-v8-003: Randomized Temp DLL Drop Pattern
// qb4p11bz.dll naming — 8-char random alphanum DLL in low-entropy region
// ============================================================
rule SL_ClickFix_v8_TempDrop_DLL
{
    meta:
        description     = "ClickFix v8 — 8-char randomized temp DLL staging artifact"
        author          = "SecureLeaf / Dispensight"
        reference       = "SL-MEM-2026-v8"
        date            = "2026-06-09"
        tlp             = "AMBER"
        confidence      = "MEDIUM"
```

```
    strings:
        // qb4p11bz.dll exact
        $exact_dll    = "qb4p11bz.dll" ascii wide
        // Generic pattern: 8 lowercase alphanum chars + .dll (wide)
        // Match: [a-z0-9]{8}\.dll in memory
        $gen_dll_w    = /[a-z0-9]{8}\.dll/ wide
        // Kernel32 + mscoree presence (minimal PE imports for reflective loader)
        $kernel32_w   = "KERNEL32.dll" wide
        $mscoree_w    = "mscoree.dll" wide

    condition:
        $exact_dll or
        ($gen_dll_w and $kernel32_w and $mscoree_w and math.entropy(0, filesize) < 2.0)
}


// ============================================================
// SL-YARA-2026-v8-004: C2 Staging Server Beacon Pattern
// Matches memory regions containing the 158.94.208.104 C2 IP
// combined with network capability indicators
// ============================================================
rule SL_ClickFix_v8_C2_Beacon_Region
{
    meta:
        description     = "ClickFix v8 — Memory region with C2 IP + WINHTTP/powershell beacon
artifacts"
        author          = "SecureLeaf / Dispensight"
        reference       = "SL-MEM-2026-v8 / IOC: 158.94.208.104"
        date            = "2026-06-09"
        tlp             = "AMBER"
        confidence      = "HIGH"

    strings:
        $c2_ip          = "158.94.208.104" ascii wide
        $winhttp        = "WINHTTP.dll" ascii wide
        $advapi         = "ADVAPI32.dll" ascii wide
        $powershell_str = "powershell" ascii nocase
        $svchost        = "svchost.exe" ascii wide

    condition:
        $c2_ip and
        (
            ($winhttp and $advapi) or
            ($powershell_str and $svchost)
        )
}


// ============================================================
// SL-YARA-2026-v8-005: Multi-Thread Watchdog Memory Pattern
// Detects when the MZER region is duplicated across process threads
// (runtime/behavioral — for memory scanner use)
// ============================================================
rule SL_ClickFix_v8_Watchdog_MZER_MultiThread
{
    meta:
        description     = "ClickFix v8 — MZER watchdog: identical region loaded in multiple threads
(memory scan)"
        author          = "SecureLeaf / Dispensight"
        reference       = "SL-MEM-2026-v8"
        date            = "2026-06-09"
        tlp             = "AMBER"
        confidence      = "HIGH"
        note            = "Use with process memory scanner — match across multiple threads of same
PID"
```

```
    strings:
        $mzer_magic      = { 4D 5A 45 52 }
        $pic_call        = { E8 00 00 00 00 59 }
        $mscoree_w       = "mscoree.dll" wide
        $myapp_w         = "MyApplication.app" wide
        $size95_w        = "size95.exe" wide

    condition:
        $mzer_magic at 0 and
        $pic_call and
        2 of ($mscoree_w, $myapp_w, $size95_w)
}


// ============================================================
// SL-YARA-2026-v8-006: Composite ClickFix v8 Dropper
// Broad rule combining key v8 artifacts for hunting
// ============================================================
rule SL_ClickFix_v8_Dropper_Composite
{
    meta:
        description     = "ClickFix v8 — Composite dropper hunt rule: MZER + PS + .NET + dropper
name"
        author          = "SecureLeaf / Dispensight"
        reference       = "SL-ADV-2026-WP-001 / SL-MEM-2026-v8"
        date            = "2026-06-09"
        tlp             = "AMBER"
        confidence      = "MEDIUM"

    strings:
        $mzer            = { 4D 5A 45 52 }
        $size95_a        = "size95.exe" ascii
        $size95_w        = { 73 00 69 00 7A 00 65 00 39 00 35 00 2E 00 65 00 78 00 65 00 }
        $psreadline      = "PSReadline.dll" ascii wide
        $mscoree         = "mscoree.dll" ascii wide
        $myapp           = "MyApplication.app" wide
        $c2_ip           = "158.94.208.104" ascii

    condition:
        ($mzer at 0 and 2 of ($size95_a, $size95_w, $myapp, $mscoree)) or
        ($psreadline and $c2_ip) or
        ($size95_a and $psreadline and $mscoree) or
        3 of them
}
```

---

7. AbuseIPDB / OTX Report Data

AbuseIPDB entry for `158.94.208.104`
- Categories: 19 (Web App Attack), 21 (IoT Targeted), 14 (Port Scan)
- Comment: `ClickFix/EtherHiding v8 campaign C2 staging server. Apache/2.4.52 (Ubuntu) on port 80. IP
present in process memory of malware sandbox crash dump (PID 992 HTTP response cache + PID 4448
staging region). Associated dropper: size95.exe. Part of campaign SL-ADV-2026-WP-001.`

OTX Pulse additions
New indicators to add to existing Omegatech/ClickFix pulse:
- `158.94.208.104` — C2/staging (IPv4)
- `size95.exe` — dropper filename (FilePath)
- `qb4p11bz.dll` — temp DLL artifact (FilePath)
- `4d5a4552` — MZER magic bytes YARA anchor (YARA)
- SHA1 hash candidate: `91A9773E7A0BA4700195CBFFFF935A24C674C3E0`

---

8. Recommended Next Steps

1. Shodan/Censys pivot on `158.94.208.104` — check open ports, historical SSL certs, BGP ASN ownership
2. Deobfuscate shellcode blob (`4448-1608`) — attempt XOR keyspace brute (single-byte first, then rolling) against the 57KB blob. Entry `CALL +0x6DC0` → decryptor at that offset
3. Entropy delta across MZER threads is small but non-zero (5.6818 → 5.7252 across 5 threads) — a diff of the 5 identical-VA dumps may reveal a counter, timestamp, or per-thread key embedded in the mutable portion
4. Submit `size95.exe` hash to VirusTotal when available
5. Add `qb4p11bz.dll` pattern to SIEM regex: `[a-z0-9]{8}\.dll` loaded from `%TEMP%` or `%APPDATA%`
6. Named pipe monitoring — add `CPFATP_` and randomized `CPFATP_*` to EDR pipe watch rules
7. Add YARA rules to TAXII feed at `taxii.dispensight.ca`

Report generated: SL-MEM-2026-v8 | SecureLeaf — Dispensight Cybersecurity Intelligence