MQsTTang: Mustang Panda's latest backdoor treads new ground with Qt and MQTT

3/2/2023

Alexandre Côté Cyr
2 Mar 2023 - 11:30AM

ESET researchers tease apart MQsTTang, a new backdoor used by Mustang Panda, which communicates via the MQTT protocol

ESET researchers have analyzed MQsTTang, a new custom backdoor that we attribute to the Mustang Panda APT group. This backdoor is part of an ongoing campaign that we can trace back to early January 2023. Unlike most of the group's malware, MQsTTang doesn't seem to be based on existing families or publicly available projects.

Mustang Panda is known for its customized Korplug variants (also dubbed PlugX) and elaborate loading chains. In a departure from the group's usual tactics, MQsTTang has only a single stage and doesn't use any obfuscation techniques.

Victimology

We have seen unknown entities in Bulgaria and Australia in our telemetry. We also have information indicating that this campaign is targeting a governmental institution in Taiwan. However, due to the nature of the decoy filenames used, we believe that political and governmental organizations in Europe and Asia are also being targeted. This would also be in line with the targeting of the group's other recent campaigns. As documented by fellow researchers at Proofpoint, Mustang Panda has been known to target European governmental entities since at least 2020 and has increased its activity in Europe even further, since Russia's invasion of Ukraine. Figure 1 shows our view of the targeting for this campaign.

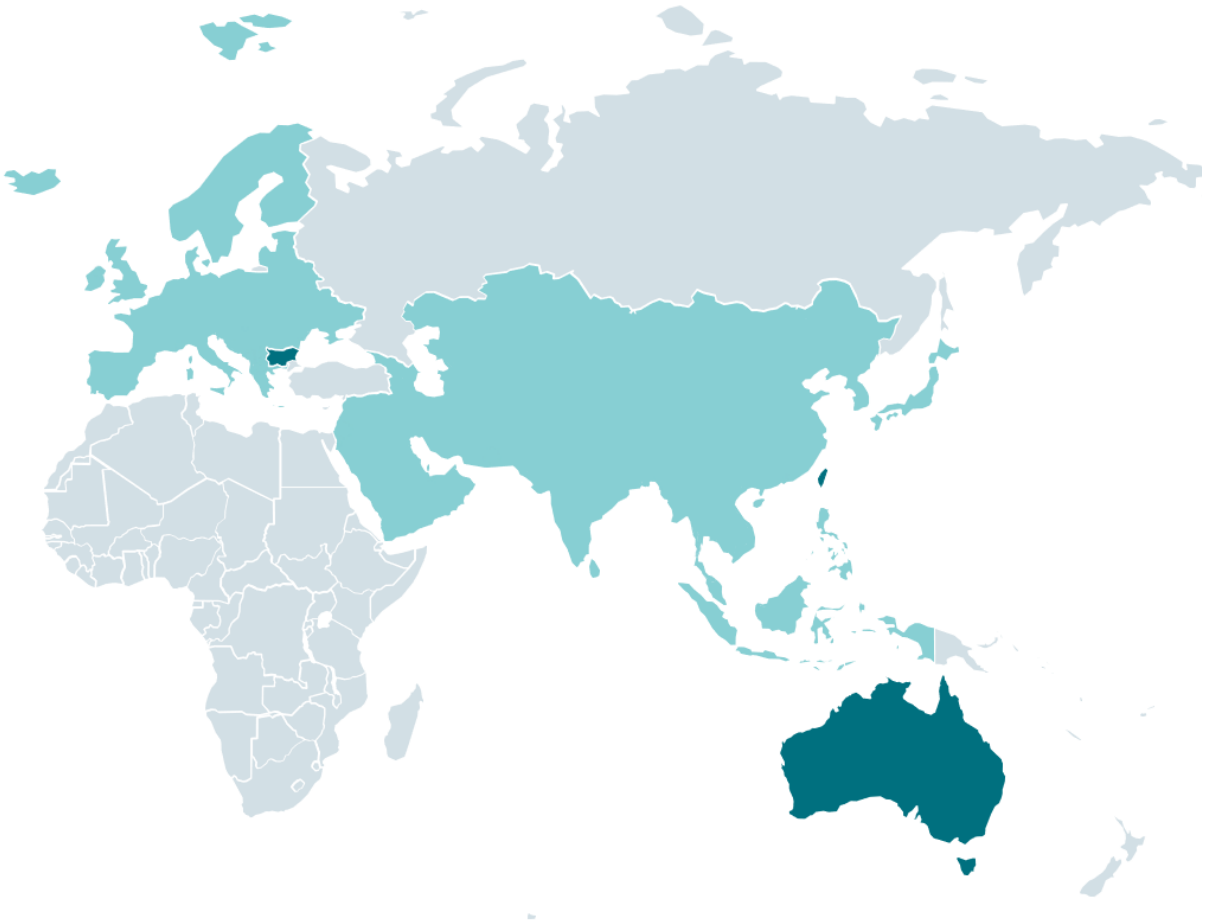

Figure 1. Map showing known and suspected targets of MQsTTang

Attribution

We attribute this new backdoor and the campaign to Mustang Panda with high confidence based on the following indicators.

We found archives containing samples of MQsTTang in two GitHub repositories belonging to the user YanNaingOo0072022. Another GitHub repository of the same user was used in a previous Mustang Panda campaign described by Avast in a December 2022 blogpost.

One of the servers used in the current campaign was running a publicly accessible anonymous FTP server that seems to be used to stage tools and payloads. In the /pub/god directory of this server there are multiple Korplug loaders, archives, and tools that were used in previous Mustang Panda campaigns. This is the same directory that was used by the stager described in the aforementioned Avast blogpost. This server also had a /pub/gd directory, which was another path used in that campaign.

Some of the infrastructure used in this campaign also matches the network fingerprint of previously known Mustang Panda servers.

Technical analysis

MQsTTang is a barebones backdoor that allows the attacker to execute arbitrary commands on a victim's machine and get the output. Even so, it does present some interesting characteristics. Chief among these is its use of the MQTT protocol for C&C communication. MQTT is typically used for communication between IoT devices and controllers, and the protocol hasn't been used in many publicly documented malware families. One such example is Chrysaor, also known as Pegasus for Android. From an attacker's perspective, one of MQTT's benefits is that it hides the rest of their infrastructure behind a broker. Thus, the compromised machine never communicates directly with the C&C server. As seen in Figure 2, this capability is achieved by using the open source QMQTT library. This library depends on the Qt framework, a large part of which is statically linked in the malware. Using the Qt framework for malware development is also fairly uncommon. Lazarus's MagicRAT is one of the rare recently documented examples.

```
; public QMQTT::TimerInterface :
;   public /* offset 0x0 */ QObject
_ZTIN5QMQTT14TimerInterfaceE dd offset _ZTVN10__cxxabiv120__si_class_type_infoE+8
                                        ; DATA XREF: .rdata:0093ADCC↓o
                                        ; .rdata:00941A14↓o
                                        ; reference to RTTI's type class
                dd offset _ZTSN5QMQTT14TimerInterfaceE ; reference to type's name
                dd offset _ZTI7QObject  ; reference to parent's type name
; public QMQTT::SocketInterface :
;   public /* offset 0x0 */ QObject
_ZTIN5QMQTT15SocketInterfaceE dd offset _ZTVN10__cxxabiv120__si_class_type_infoE+8
                                        ; DATA XREF: .rdata:0093ADE4↓o
                                        ; .rdata:00941A64↓o
                                        ; reference to RTTI's type class
                dd offset _ZTSN5QMQTT15SocketInterfaceE ; reference to type's name
                dd offset _ZTI7QObject  ; reference to parent's type name
; public QMQTT::NetworkInterface :
;   public /* offset 0x0 */ QObject
_ZTIN5QMQTT16NetworkInterfaceE dd offset _ZTVN10__cxxabiv120__si_class_type_infoE+8
                                        ; DATA XREF: .rdata:0093ADF0↓o
                                        ; .rdata:00941AB4↓o
                                        ; reference to RTTI's type class
                dd offset _ZTSN5QMQTT16NetworkInterfaceE ; reference to type's name
                dd offset _ZTI7QObject  ; reference to parent's type name
; public QMQTT::Frame
_ZTIN5QMQTT5FrameE dd offset _ZTVN10__cxxabiv117__class_type_infoE+8
```

Figure 2. RTTI showing classes from the QMQTT library

MQsTTang is distributed in RAR archives which only contain a single executable. These executables usually have names related to Diplomacy and passports such as:

- CVs Amb Officer PASSPORT Ministry Of Foreign Affairs.exe
- Documents members of delegation diplomatic from Germany.Exe
- PDF_Passport and CVs of diplomatic members from Tokyo of JAPAN.eXE
- Note No.18-NG-23 from Embassy of Japan.exe

These archives are hosted on a web server with no associated domain name. This fact, along with the filenames, leads us to believe that the malware is spread via spearphishing.

So far, we have only observed a few samples. Besides variations in some constants and hardcoded strings, the samples are remarkably similar. The only notable change is the addition of some anti-analysis techniques in the latest versions. The first of these consists of using the CreateToolhelp32Snapshot Windows API function to iterate through running processes and look for the following known debuggers and monitoring tools.

- cheatengine-x86_64.exe
- ollydbg.exe
- ida.exe
- ida64.exe
- radare2.exe
- x64dbg.exe
- procmon.exe
- procmon64.exe
- procexp.exe
- processhacker.exe
- pestudio.exe
- systracerx32.exe
- fiddler.exe
- tcpview.exe

Note that, while the malware is a 32-bit executable, it only checks for the presence of x64dbg and not its 32-bit counterpart, x32dbg.

The second technique uses the FindWindowW Windows API to look for the following Window Classes and Titles used by known analysis tools:

- PROCMON_WINDOW_CLASS
- OLLYDBG
- WinDbgFrameClass
- OllyDbg – [CPU]
- Immunity Debugger – [CPU]

When executed directly, the malware will launch a copy of itself with 1 as a command line argument. This is repeated by the new process, with the argument being incremented by 1 on every run. When this argument hits specific

values, certain tasks will be executed. Note that the exact values vary between samples; the ones mentioned below correspond to the sample with SHA-1 02D95E0C369B08248BFFAAC8607BBA119D83B95B. However, the tasks themselves and the order in which they are executed is constant.

Figure 3 shows an overview of this behavior along with the tasks that are executed when the malware is first run.

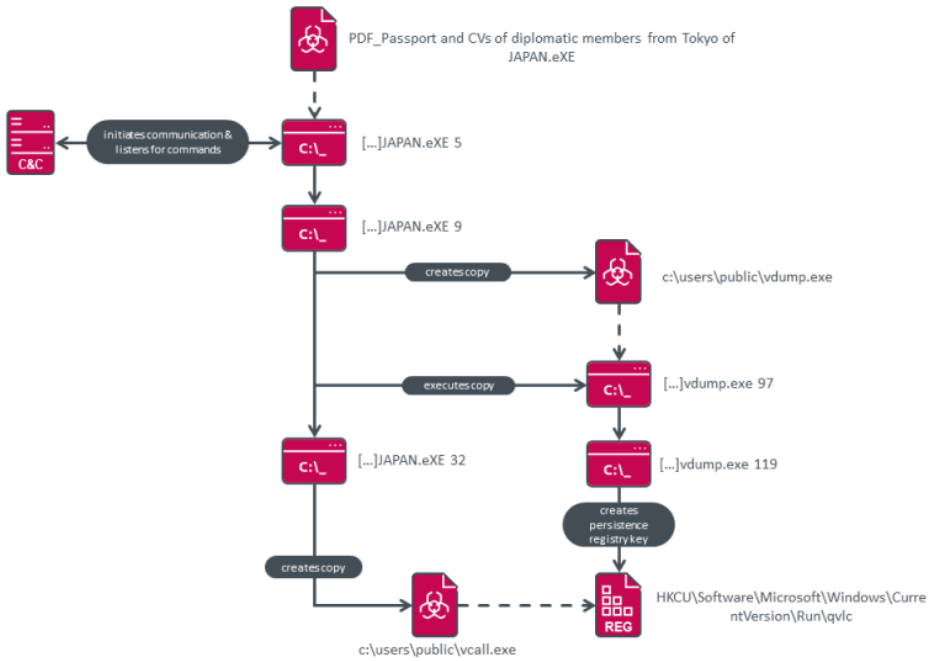

Figure 3. Execution graph showing the subprocesses and executed tasks

Table 1 contains a list of the tasks and the value at which each of them is executed. We will describe them in further detail in the upcoming paragraphs.

Table 1. Tasks executed by the backdoor

Task number	Argument value	Task description
1	**5**	Start C&C communication.
2	**9**	Create copy and launch.
3	**32**	Create persistence copy.
4	**119**	Establish persistence.
5	**148**	Stop recursive execution.

If any analysis tool or debugger is detected using the techniques we described previously, the behavior of task 1 is altered and tasks 2, 3, and 4 are skipped entirely.

Task 1: C&C communication

As was previously mentioned, MQsTTang communicates with its C&C server over the MQTT protocol. All observed samples use 3.228.54.173 as broker. This server is a public broker operated by EMQX, who also happen to be the maintainers of the QMQTT library. This could be a way to make the network traffic seem legitimate and to hide Mustang Panda's own infrastructure. Using this public broker also provides resiliency; the service is unlikely to be taken down because of its many legitimate users and, even if the current C&C servers are banned or taken down, Mustang Panda could spin up new ones and use the same MQTT topics without disrupting MQsTTang's operation.

However, this campaign could also be a test case by Mustang Panda before deciding whether to invest the time and resources to set up their own broker. This is supported by the low number of samples we've observed and the very simple nature of MQsTTang.

As shown in Figure 4, the malware and C&C server use two MQTT topics for their communication. The first one, iot/server2, is used for communication from the client to the server. The second one is used for communication from the server to the client. It follows the format iot/v2/<Unique ID> where <Unique ID> is generated by taking the last 8 bytes, in hex form, of a UUID. If any analysis tool is detected, server2 and v2 are respectively replaced with server0 and v0. This is likely in order to avoid tipping off defenders by entirely aborting the malware's execution early.

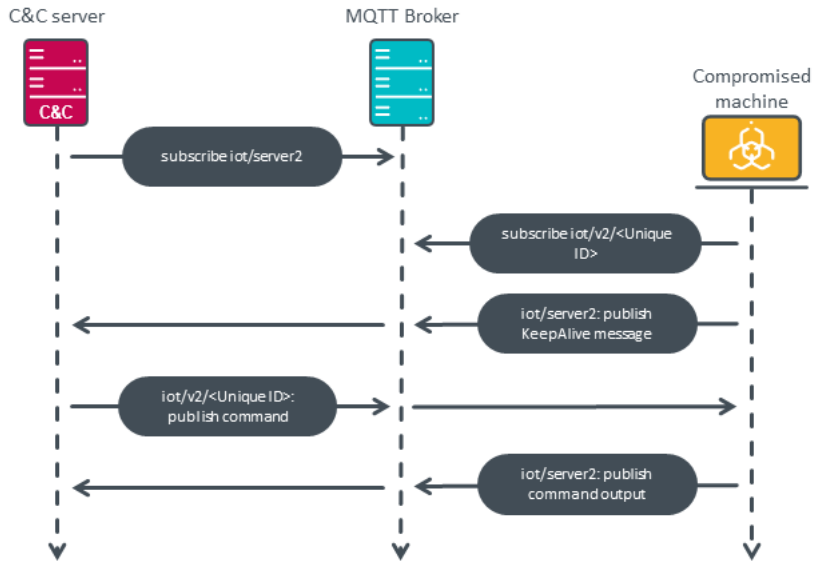

Figure 4. Simplified network graph of the communication between the backdoor and C&C server

All communication between the server and the client uses the same encoding scheme. The MQTT message's payload is a JSON object with a single attribute named msg. To generate the value of this attribute, the actual content is first base64 encoded, then XORed with the hardcoded string nasa, and base64 encoded again. We will describe the exact format of these payloads in the relevant sections.

Upon first connecting to the broker, the malware subscribes to its unique topic. Then, and every 30 seconds thereafter, the client publishes a KeepAlive message to the server's topic. The content of this message is a JSON object with the following format:

```
1 {
2    "Alive": "<malware's uptime in minutes>",
3    "c_topic": "<client's unique topic>"
4 }
```

When the server wants to issue a command, it publishes a message to the client's unique topic. The plaintext content of this message is simply the command to be executed. As shown in Figure 5, the client executes the received command using QProcess::startCommand from the Qt framework. The output, obtained using QProcess::readAllStandardOutput, is then sent back in a JSON object with the following format:

```
1 {
2    "c_topic": "<client's unique topic>",
3    "ret": "<Command output>"
4 }
```

```
QProcess::ctor(&qproc, 0);
QProcess::startCommand(&qproc, command, MEM_READWRITE);
QProcess::waitForStarted(&qproc, 30000);
QProcess::waitForBytesWritten(&qproc, 30000);
QProcess::readAllStandardOutput(&process_output, &qproc);
```

Figure 5. Execution of received commands using the QProcess *class*

Since only the content of standard output is sent back, the server will not receive errors or warnings. From the server's point of view, a failed command is thus indistinguishable from a command that simply produces no output unless some sort of redirection is performed.

Tasks 2 and 3: Copying the malware

The second and third tasks are fairly similar to each other. They copy the malware's executable to a hardcoded path; c:\users\public\vdump.exe and c:\users\public\vcall.exe respectively. The filenames used are different for each sample, but they are always located in the C:\users\public directory.

In the second task, the newly created copy is then launched with the command line argument 97.

Task 4: Establishing persistence

Persistence is established by the fourth task, which creates a new value qvlc set to c:\users\public\vcall.exe under the HKCU\Software\Microsoft\Windows\CurrentVersion\Run registry key. This will cause the malware to be executed on startup.

When MQsTTang is executed on startup as c:\users\public\vcall.exe, only the C&C communication task is executed.

Conclusion

The Mustang Panda campaign described in this article is ongoing as of this writing. The victimology is unclear, but the decoy filenames are in line with the group's other campaigns that target European political entities.

This new MQsTTang backdoor provides a kind of remote shell without any of the bells and whistles associated with the group's other malware families. However, it shows that Mustang Panda is exploring new technology stacks for its tools. It remains to be seen whether this backdoor will become a recurring part of the group's arsenal, but it is one more example of the group's fast development and deployment cycle.

ESET Research offers private APT intelligence reports and data feeds. For any inquiries about this service, visit the ESET Threat Intelligence page.

IoCs

Files

SHA-1	Filename	Detection	Description
A1C660D31518C8AFAA6973714DE30F3D576B68FC	CVs Amb.rar	Win32/Agent.AFBI	RAR archive used to distribute MQsTTang backdoor.
430C2EF474C7710345B410F49DF853BDEAFBDD78	CVs Amb Officer PASSPORT Ministry Of Foreign Affairs.exe	Win32/Agent.AFBI	MQsTTang backdoor.
F1A8BF83A410B99EF0E7FDF7BA02B543B9F0E66C	Documents.rar	Win32/Agent.AFBI	RAR archive used to distribute MQsTTang backdoor.
02D95E0C369B08248BFFAAC8607BBA119D83B95B	PDF_Passport and CVs of diplomatic members from Tokyo of JAPAN.eXE	Win32/Agent.AFBI	MQsTTang backdoor.
0EA5D10399524C189A197A847B8108AA8070F1B1	Documents members of delegation diplomatic from Germany.Exe	Win32/Agent.AFBI	MQsTTang backdoor.
982CCAF1CB84F6E44E9296C7A1DDE2CE6A09D7BB	Documents.rar	Win32/Agent.AFBI	RAR archive used to distribute MQsTTang backdoor.
740C8492DDA786E2231A46BFC422A2720DB0279A	23 from Embassy of Japan.exe	Win32/Agent.AFBI	MQsTTang backdoor.
AB01E099872A094DC779890171A11764DE8B4360	BoomerangLib.dll	Win32/Korplug.TH	Known Mustang Panda Korplug loader.
61A2D34625706F17221C1110D36A435438BC0665	breakpad.dll	Win32/Korplug.UB	Known Mustang Panda Korplug loader.
30277F3284BCEEF0ADC5E9D45B66897FA8828BFD	coreclr.dll	Win32/Agent.ADMW	Known Mustang Panda Korplug loader.
BEE0B741142A9C392E05E0443AAE1FA41EF512D6	HPCustPartUI.dll	Win32/Korplug.UB	Known Mustang Panda Korplug loader.
F6F3343F64536BF98DE7E287A7419352BF94EB93	HPCustPartUI.dll	Win32/Korplug.UB	Known Mustang Panda Korplug loader.
F848C4F3B9D7F3FE1DB3847370F8EEFAA9BF60F1	libcef.dll	Win32/Korplug.TX	Known Mustang Panda Korplug loader.

Network

IP	Domain	Hosting provider	First seen	Details
3.228.54.173	broker.emqx.io	Amazon.com, Inc.	2020-03-26	Legitimate public MQTT broker.
80.85.156[.]151	N/A	Chelyabinsk-Signal LLC	2023-01-05	MQsTTang delivery server.
80.85.157[.]3	N/A	Chelyabinsk-Signal LLC	2023-01-16	MQsTTang delivery server.

IP	Domain	Hosting provider	First seen	Details
185.144.31[.]86	N/A	Abuse-C Role	2023-01-22	MQsTTang delivery server.

Github repositories

- https://raw.githubusercontent[.]com/YanNaingOo0072022/14/main/Documents.rar
- https://raw.githubusercontent[.]com/YanNaingOo0072022/ee/main/CVs Amb.rar

MITRE ATT&CK techniques

This table was built using version 12 of the MITRE ATT&CK framework.

Tactic	ID	Name	Description
Resource Development	T1583.003	Acquire Infrastructure: Virtual Private Server	Some servers used in the campaign are on shared hosting.
	T1583.004	Acquire Infrastructure: Server	Some servers used in the campaign seem to be exclusive to Mustang Panda.
	T1587.001	Develop Capabilities: Malware	MQsTTang is a custom backdoor, probably developed by Mustang Panda.
	T1588.002	Obtain Capabilities: Tool	Multiple legitimate and open- source tools, including psexec, ps, curl, and plink, were found on the staging server.
	T1608.001	Stage Capabilities: Upload Malware	MQsTTang was uploaded to the web server for distribution.
	T1608.002	Stage Capabilities: Upload Tool	Multiple tools were uploaded to an FTP server.
Initial Access	T1566.002	Phishing: Spearphishing Link	MQsTTang is distributed via spearphishing links to a malicious file on an attacker-controlled web server.
Execution	T1106	Native API	MQsTTang uses the QProcess class from the Qt framework to execute commands.
	T1204.002	User Execution: Malicious File	MQsTTang relies on the user to execute the downloaded malicious file.
Persistence	T1547.001	Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder	MQsTTang persists by creating a registry Run key.
Defense Evasion	T1036.004	Masquerading: Masquerade Task or Service	In most samples, the registry key is created with the name qvlc. This matches the name of a legitimate executable used by VLC.
	T1036.005	Masquerading: Match Legitimate Name or Location	When creating copies, MQsTTang uses filenames of legitimate programs.
	T1480	Execution Guardrails	MQsTTang checks the paths it is executed from to determine which tasks to execute.
	T1622	Debugger Evasion	MQsTTang detects running debuggers and alters its behavior if any are found to be present.
Command and Control	T1071	Application Layer Protocol	MQsTTang communicates with its C&C server using the MQTT protocol.
	T1102.002	Web Service: Bidirectional Communication	MQsTTang uses a legitimate public MQTT broker.
	T1132.001	Data Encoding: Standard Encoding	The content of the messages between the malware and server is base64 encoded.
	T1573.001	Encrypted Channel: Symmetric Cryptography	The content of the messages between the malware and server is encrypted using a repeating XOR key.
Exfiltration	T1041	Exfiltration Over C2 Channel	The output of executed commands is sent back to the server using the same protocol.