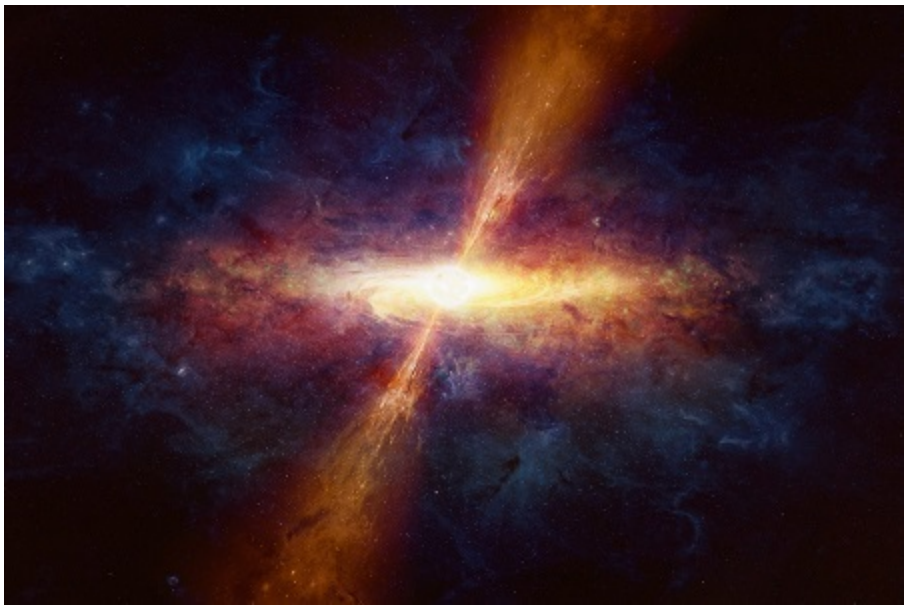
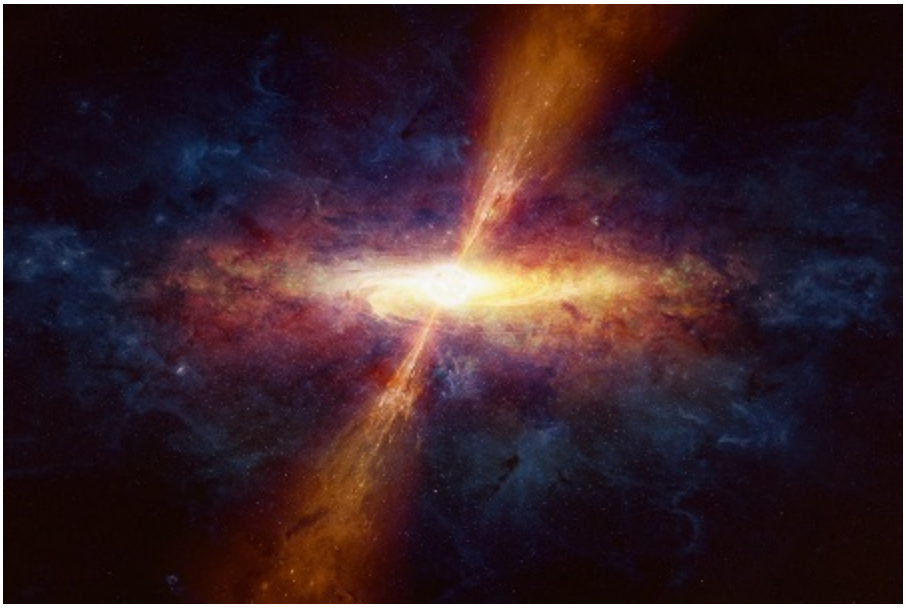
Trapping A Fat Quasar RAT

blog.minerva-labs.com/trapping-quasar-rat





- Tweet
-

How would you go about evading the state-of-the-art sandbox? The most straightforward way would be to test your malware versus the industry's top vendors. A sample we encountered in January, 2021 has taken a different approach. Its developer simply "fattened" it up into a tremendous 624.79 MB file size, thus surpassing the upload limit of most

sandboxing solutions, which is akin to avoiding a security checkup by simply being too big to fit through the door. In this blog we will give a technical overview of this weight-challenged RAT, and how it used its larger size to evade security products.

FatRAT – First Stage Packer:

The sample, named wvQROZu.exe, is a .NET portable executable file, with a highly obfuscated variable and function names. Its initial size of 624 MB shrinks to a measly 806 KB after de-obfuscation by the amazing de4dot:

The first stage of the malware extracts a .NET DLL from a compressed and Base64 hard-coded buffer. The DLL, self-proclaimed MARCUS.dll, is loaded and one of its functions is invoked, passing a resource name and an encryption key as parameters. The resource (which was passed to MARCUS.dll as a parameter) points to a bitmap image in which the

next payload is encoded. The data is stored in the image's RGB pixel values, similarly to the RedLine Stealer sample we covered in a previous post. This time instead of using RC2 stream cipher, the sample uses a XOR based algorithm to decrypt the next payload.

The XOR-based decryption algorithm:

After decryption and Gzip decompression, another assembly is revealed and loaded in-memory.

Second Stage Packer:

The second stage of the malware is yet another obfuscated .NET binary. The binary's developer shamelessly copied a bunch of anti-sandboxing techniques straight from this 2013 post. Specifically, a list of usernames that is compared against the current user and a list of file names are checked against the process' image path.

The blacklisted artifacts:

Usernames	File paths\names
SANDBOX	Virus
Malware	Sandbox
Virus	Sample
Schmidti	C:\file.exe
Currentuser	

It then checks for the existence of a window named “Afx:400000:0” which is indicative of WinJail sandbox.

The last anti-sandboxing check is for the existence of the DLL SbieDll.dll, a file that is loaded in sandboxie’s analyzed process.

The obfuscated SbieDll.dll check:

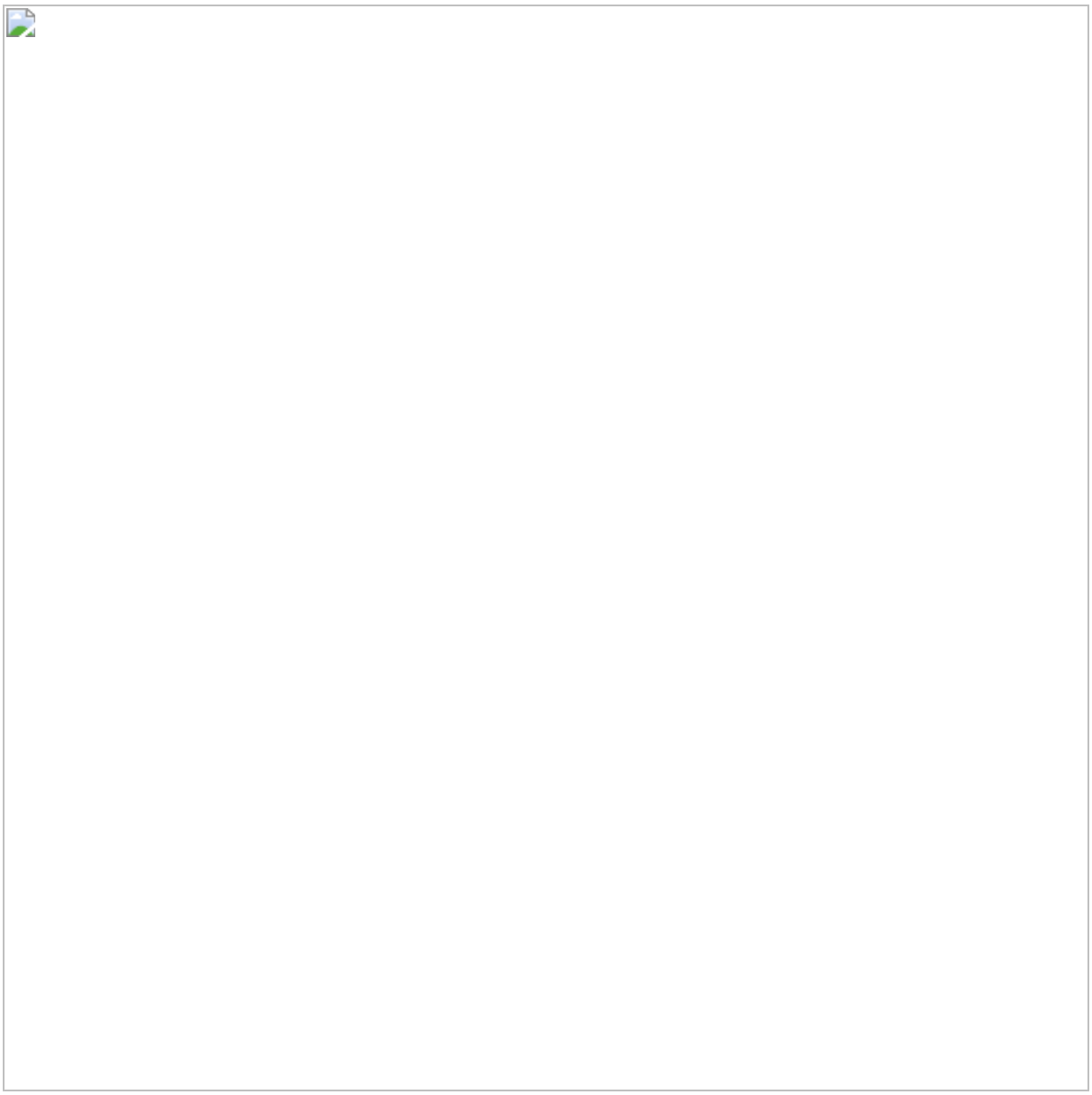
If any of the above artifacts are found, the malware will self-terminate.

After these checks, the malware will check if the file %AppData%\wvQROZu.exe exists, and will copy itself to that location if it is missing.

Persistence wise, the malware creates a schedule task on the infected machine. The scheduled task is created using a unique feature, the malware carries in its resource a schedule task XML file which is modified in runtime to fit the compromised device. The XML file will be dropped to %Temp% directory, then the task will be created using the default Windows binary schtasks.exe and the “/XML” flag. When creating a schedule task from XML, the task creation date can be set to any arbitrary value, and by abusing this feature, the malware’s developers attempt to fool forensics investigators with a fake timestamp.

A snippet from the embedded XML file:

The schedule task command executed by the malware (as seen in Sysinternal’s Procmon) :



After setting up persistence, the final payload is decrypted and then injected using process hollowing into a newly created “Regsvcs.exe” process.

The final payload is revealed to be a Quasar RAT client, which connects to a hardcoded C&C address and enables full control of the device.

Conclusion:

Attackers find new and interesting ways to bypass security products, and while the “fattening” technique used by this malware is not sophisticated, it is nonetheless effective.

Minerva Prevents the threat detailed in this blog, along with other evasive malware using our Hostile Environment Simulation technology.

The malware in question was prevented 3 months before it was uploaded to VirusTotal:

IOCs:

Hashes:

568de2a8e7af6fdda514c4cc7fd37c7ae92182955f2739a75d7c64a6bcc7f46d (wvQROZu.exe)

a58efd253a25cc764d63476931da2ddb305a0328253a810515f6735a6690de1d (Quasar RAT final payload)

82ed88cebe87b50d0ca2f0c452de79b43e66988b0babbadcafce6c07526cb52c (second stage payload)

e83cef9cfa4e2ea30e28843e43c60ebf8f6590fb753ad0aa5ed1123c01280527 (MARCUS.dll)

DNS:

starwort[.]kozow[.]com:1980

Mutex:

QSR_MUTEX_XMISpvJz1tS3wbFVbJ