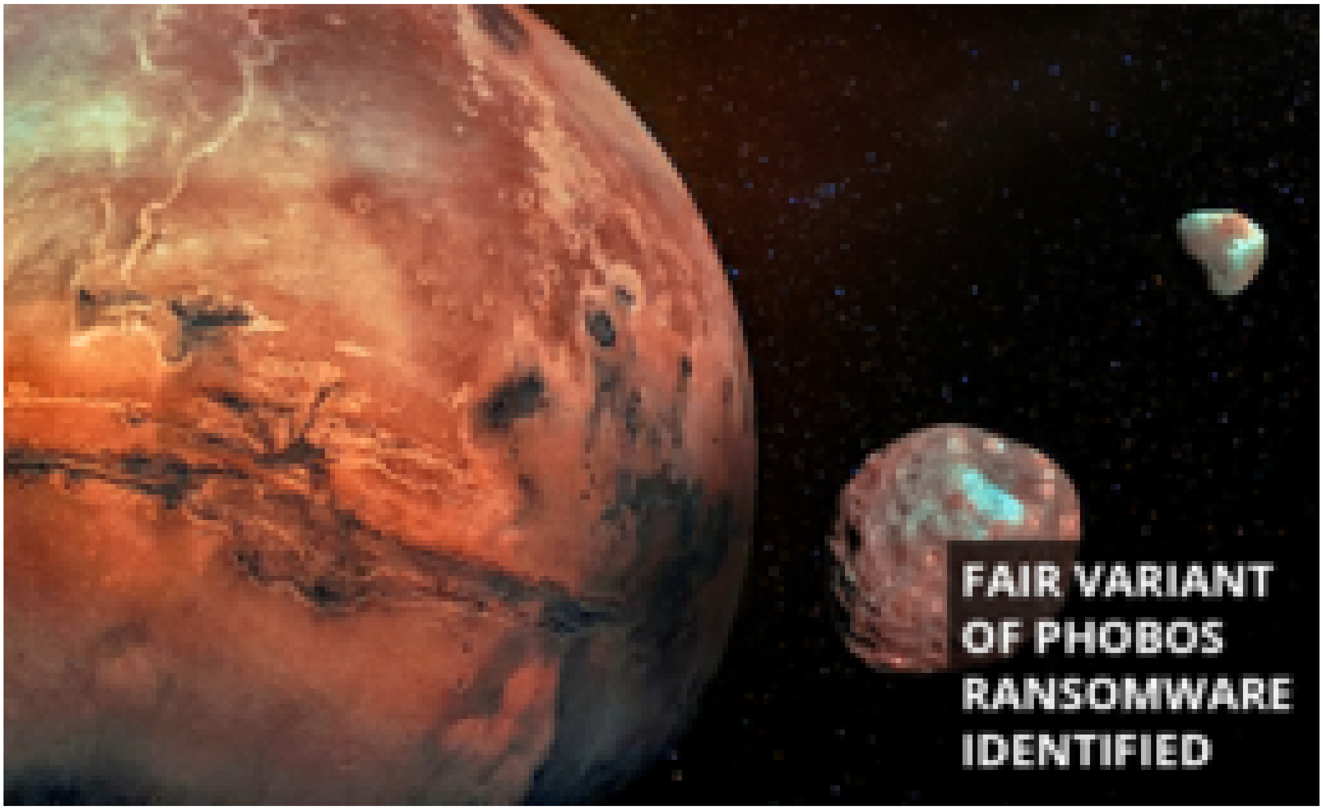
The “Fair” Upgrade Variant of Phobos Ransomware

blog.morphisec.com/the-fair-upgrade-variant-of-phobos-ransomware

Michael Gorelik

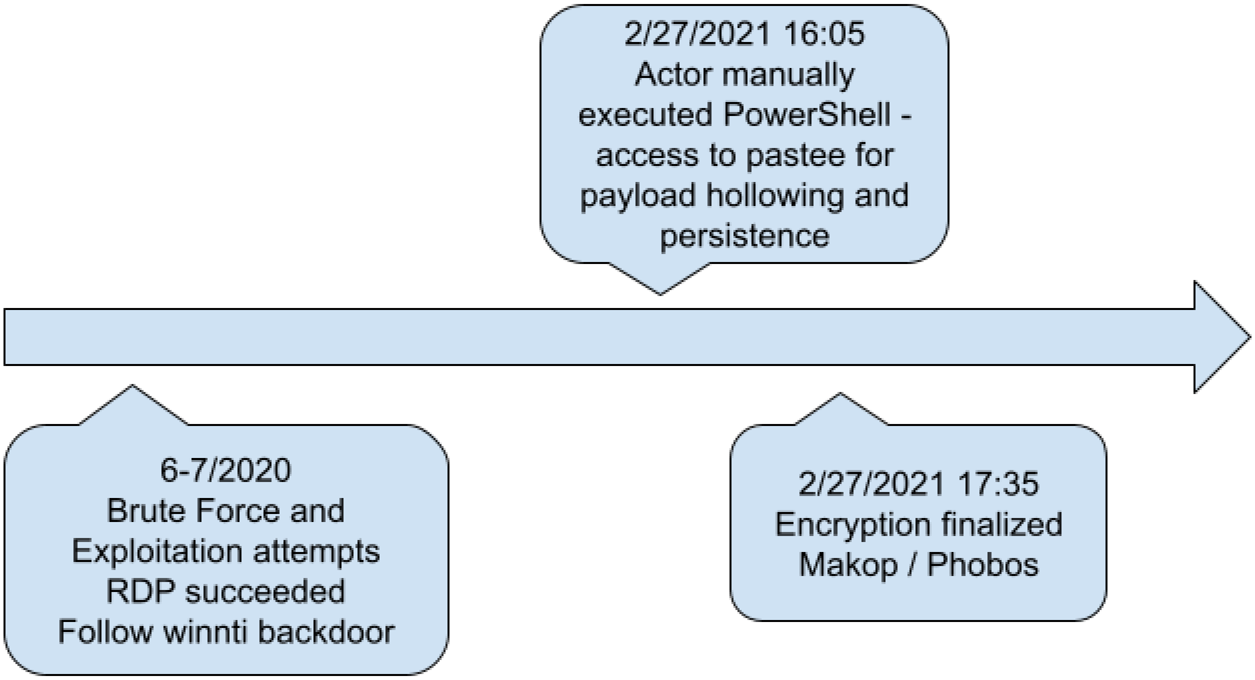
- Tweet
-



The developers of the Phobos ransomware have added new fileless and evasive techniques to their arsenal. Constantly keeping their attack up to date helps them bypass detection technologies through several distinct approaches, the latest of which we detail in this blog.

The following provides details on a new Fair variant of Phobos ransomware. The Morphisec IR team identified this variant during an incident response engagement in early March, provided as part of Morphisec’s new IR services offering. The affected company enlisted our services, and as a result, we identified this newest Phobos variant (compiled in November 2020) in their system. The technical details of the attack follow.

Technical Details


2/27/2021 16:05
Actor manually executed PowerShell - access to pastee for payload hollowing and persistence
6-7/2020
Brute Force and Exploitation attempts
RDP succeeded
Follow winnti backdoor
2/27/2021 17:35
Encryption finalized
Makop / Phobos

In this threat post, we will go into the details of the latest *Fair / Phobos ransomware* delivery and payload as has been identified as part of our routine incident response service procedures. We will present the significant changes that make this ransomware much more relevant than before.

During the first week of March, the Morphisec IR team was enlisted to help with investigating an encrypted backup server. We quickly identified an execution of PowerShell scripts that were delivering the ransomware within memory without a single executable on disk.

We observed the use of paste.ee (a Pastebin alternative) for the delivery of the loader and the ransomware component.

A deeper investigation revealed that the server was compromised for more than 8 months. While all indicators point to a probable brute force theft of the administrator password, backdoor accounts were created to maintain persistent access. Miners and botnets were also installed.

When we dove deeper into the ransomware payload, we identified significant improvements to the process and removal of footprints.

PowerShell:

The original script was base64 encoded. The decoded script is slightly obfuscated using known techniques. As can be seen, the script downloads and evaluates the first component from a *paste[.]ee/r/OwAyf* URL. This component is yet another PowerShell command. Next, the script downloads the ransomware as a string from another paste.ee URL and follows a basic string replacement process which leads to a final stage of hollowing a legitimate MSBuild.exe process.

MSBuild.exe process

MSBuild.exe process 2

Loader:

As was presented above, the first downloaded component is a PowerShell command.

PowerShell command

The PowerShell command

The PowerShell command abuses a VisualBasic CallByName exported function to call the load of a .NET assembly directly in memory.

The loaded assembly is actually a loader obfuscated with an Agile.NET obfuscator. This loader is responsible for the hollowing of a Ransomware payload (the data parameter) into a legitimate .NET assembly of choice.

Phobos loader

phobos execution

Looking through VirusTotal, we identified older versions of the similar loader obfuscated with different obfuscators--such as Vapor--even though the functionality and the activation function were preserved.

Phobos loading

ransomware process

Phobos Ransomware:

phobos

Phobos ransomware is *not* considered one of the more sophisticated ransomware encryptors. Furthermore, it would have a relatively high static detection rate if landed on disk. Nevertheless, it is immediately and constantly updated, and if executed from memory has high chances of success.

The latest FAIR version included some significant upgrades, which we will present here:

One of the more significant changes is a lower scope of encryption. Phobos's developers removed the UAC requirement to maintain *Medium* integrity. This means no encryption of privileged folders, which leads to a lower footprint. While there are fewer files to encrypt, Phobos's developers did not want to compromise on files with open handles, which most probably will have a significant impact on the victims.

1. Files are now opened with *FILE_SHARE_DELETE*. If there are opened handles with similar access rights, then the file open will fail.

FILE SHARE DELETE

2. Upon failure, the ransomware will query the currently opened handles.

currently opened handles

3. Next, it will identify the process that is responsible for the handle by duplicating the handle into the ransomware process, then extracting the filePath it points to (only if file handle) and comparing it to the target filename.

ransomware process

4. Then the ransomware will iterate over the processes with the handles pointing to the file and will attempt to terminate them.

Possibly some of the target files are pointed by the privileged process. In this case, if the same processes have *FILE_SHARE_DELETE* access, the deletion will fail (though it may become pending).

An additional modification is to the list of important files and extensions representing previous versions of Phobos's encrypted files (the intent is to avoid re-encrypting encrypted files). This list also reinforces some previous assumptions about a correlation between LockBit and Phobos ransomware.

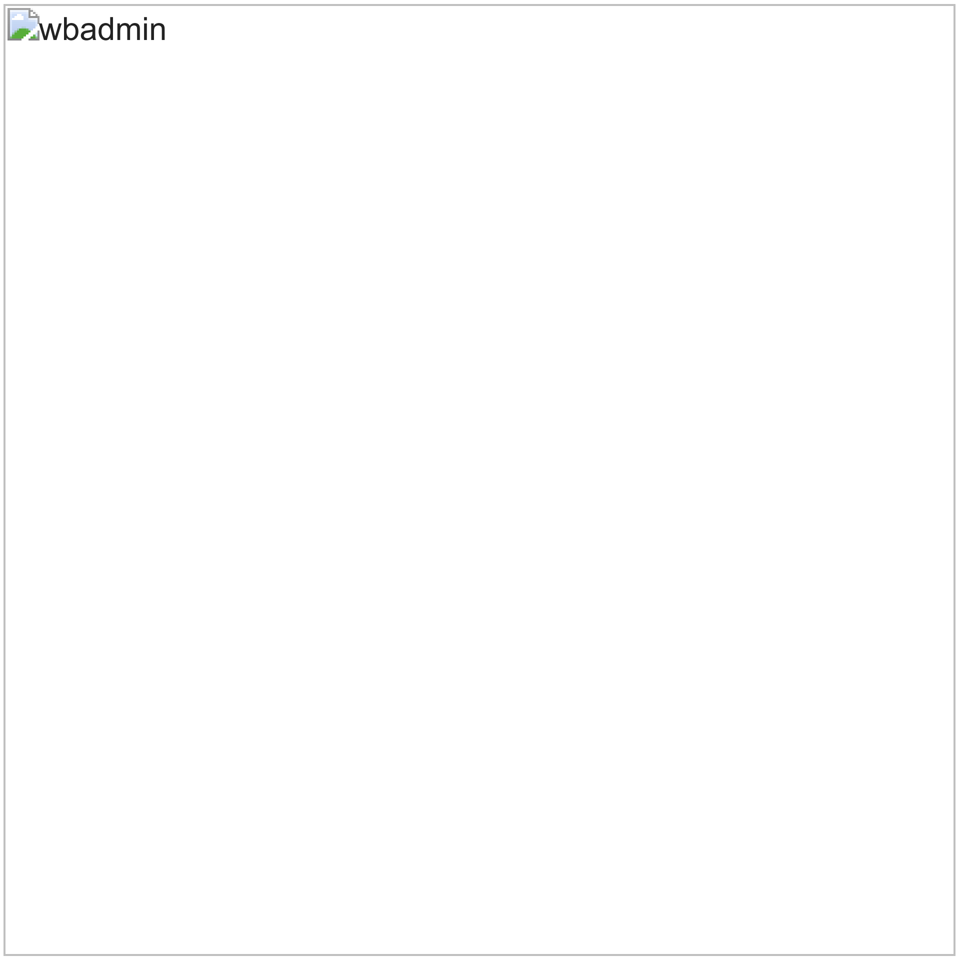
correlation between LockBit and Phobos

We also identified an improved object oriented code style and the removal of redundant parameters such as *nVolumeNameSize,* which is either way ignored. There was also the removal of redundant functions such as *GetFileSize* as the information was already available through *FindFirstFileW,* which populates all the required data. The developers also reduced code obfuscation and increased the use of WinAPI for secure decryption of strings.

An additional improvement to the coverage of files is the support of volumes that span multiple physical drives. The implementation is no longer naive, and in fact, now adheres to the best practices of Windows’ internal development. *DeviceIoControl* is used twice with *IOCTL_VOLUME_GET_VOLUME_DISK_EXTENTS* to extract the size and then the extent of the physical drive path.

Device Io Control

The developers also minimized the interaction with OS native controls due to lower privileges, removed the use of netsh for disabling the firewall, and removed the use of bcdedit. The current version still attempts to delete shadow volumes if enabled and delete backup catalogs on servers (wbadmin will work only on servers).



Conclusions

Based on the research presented here, we can safely conclude that the developers of Phobos are aiming to increase their foothold in the enterprise business. They are gradually moving to fileless delivery and footprint reduction with immediate impact. There are also clear indications for specific emphasis on targeting servers as some of the commands are only relevant to servers.

Obviously, businesses need to look at how to prevent the attack chains much earlier, during the initial access or execution stages as described by MITRE.

Businesses also need to invest in attack surface reduction and zero trust prevention. Morphisec’s runtime zero-trust approach dramatically reduces the risk of initial access and execution exposure.

IoCs

PowerShell script	7f8f8c82fec8acbb0947a192dd5cbe8b95ffdba4e252b582eae127f1c062399b

Ransomware (fileless)	2cadd0ff146e1cdf1270894be4fb1523bfdcc7a31760e0ca5cfd9d8e6b525c21 hxxps://paste[.]ee/r/1q1gD
Ransomware (on disk)	f6b60839de0ac933f0788bc1e12dee859950010f938a05544ad51c424954b9a6
Loader (hollower)	4ff1f8a052addbc5a0388dfa7f32cc493d7947c43dc7096baa070bfc4ae0a14e hxxps://paste[.]ee/r/OwAyf

ADVANCE YOUR WINDOWS 10 PROTECTION
ENHANCE YOUR NATIVE Windows SECURITY CONTROLS TO BLOCK THE MOST SOPHISTICATED ATTACKS.
Single Agent
Single Dashboard
Bulletproof Security
See For Yourself