DPRK Malware Targeting Security Researchers – One Night in Norfolk

Published: 2021-01-26 · Archived: 2026-04-05 23:37:27 UTC

Earlier today, Adam Weidemann from Google's Threat Analysis Group (TAG) published research regarding a threat actor targeting security analysts following a social engineering campaign. Google attributes this activity to DPRK threat actors. This blog has no evidence to corroborate or refute this claim, but considers Google to be a reputable source of information.

According to the published research, the threat actors would engage in a social engineering effort in which they would attempt to collaborate with security analysts on a Visual Studio project, ultimately leading to them delivering a malicious DLL that the researcher would unknowingly launch.

This post examines that DLL and parts of its second-stage workflow.

Technical Analysis

MD5: 56018500f73e3f6cf179d3b853c27912
SHA-1: a3060a3efb9ac3da444ef8abc99143293076fe32
SHA-256: 4c3499f3cc4a4fdc7e67417e055891c78540282dccc57e37a01167dfe351b244

This file is a DLL that expects to be executed under the following conditions to initiate the malicious workflow:

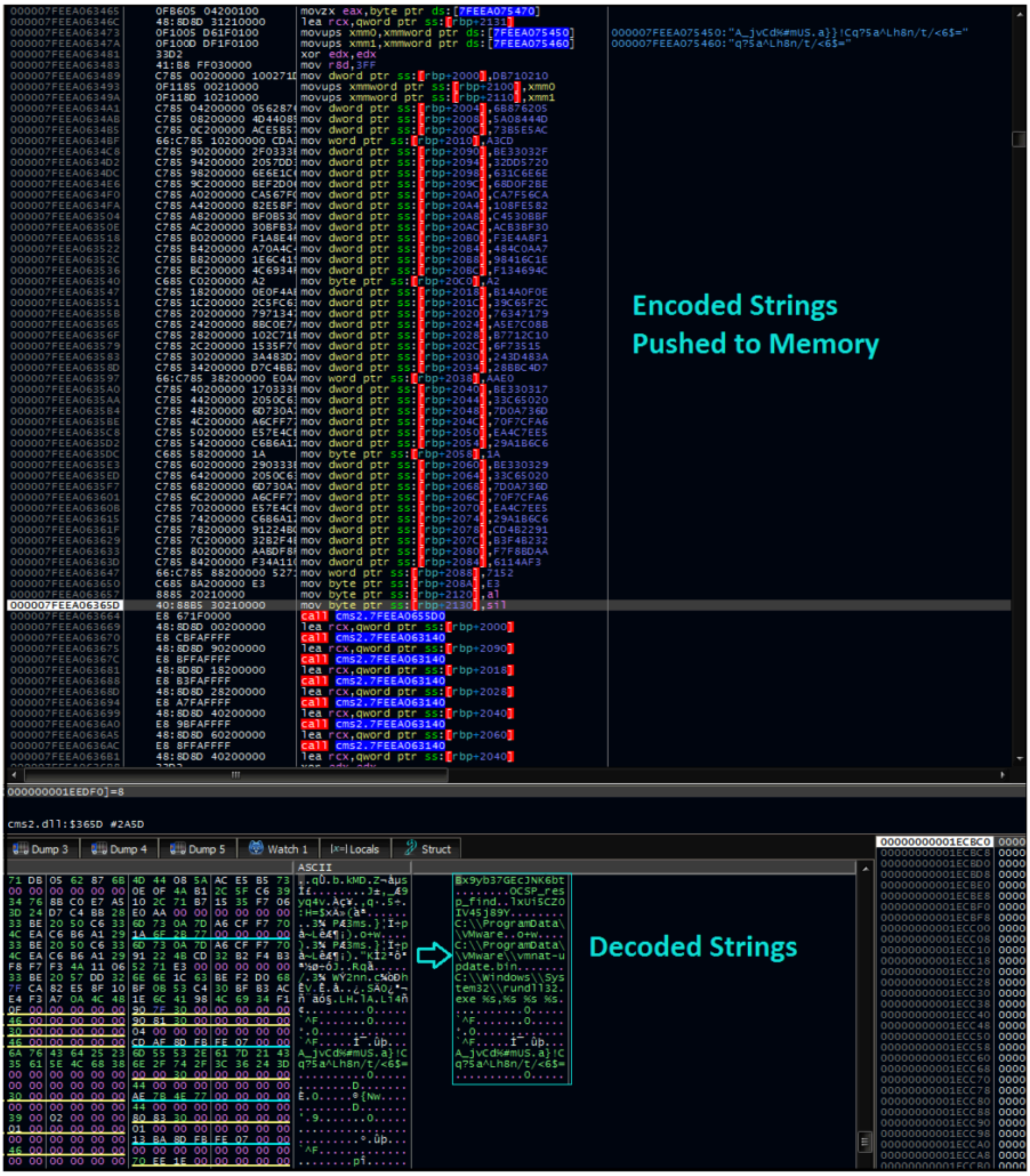
– The operating system must be 64-bit
– The correct export must be called
– Exactly two additional command line arguments must be supplied alongside this export

Although Google provided multiple hashes, the file above was selected as a starting point because its exports matched the export shown in an image in Google's post (CMS_dataFinal). The post used this image how the malware would execute in normal circumstances, which in turn allows us to supply two additional critical parameters to the file:

– Bx9yb37GEcJNK6bt
– 4901

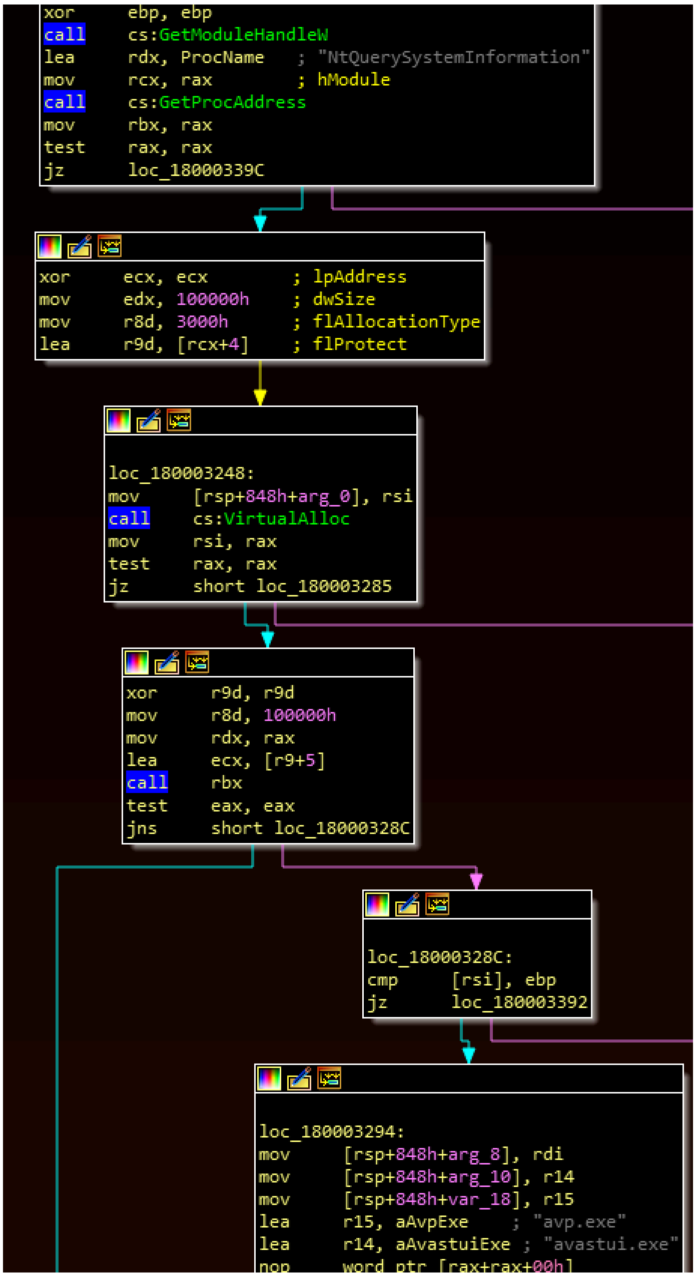
Under the attacker's workflow, these would have been supplied through a PowerShell command initiated through a Visual Studio Build Event, but these can also be supplied through a normal command line.

Once executed under the specified conditions, the malware will move an encoded set of strings into memory and decode them.



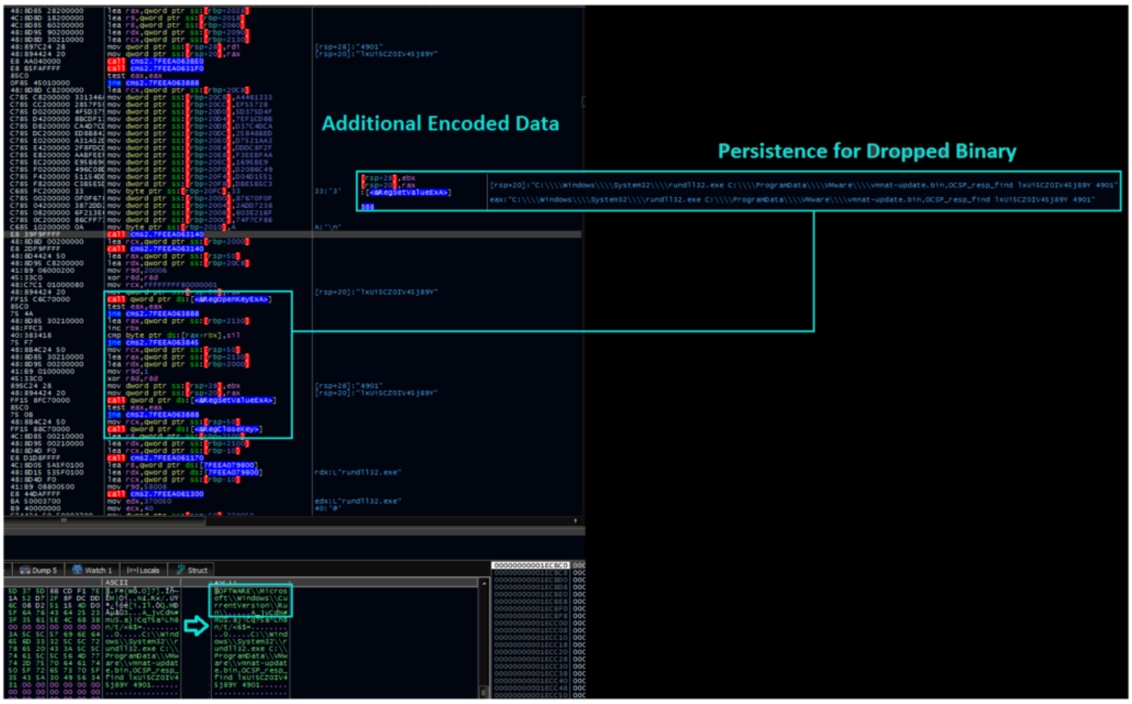
The malware uses these strings to call the CreateDirectory API at C:\ProgramData\VMWare and later to specify a filename at this location for a dropped payload (vmnat-update.bin). These decoded strings also contain an export and an additional parameter for this dropped payload.

After this, the malware enumerates running processes via NtQuerySystemInformation and checks for the presence of avp.exe (Kaspersky) and avastui.exe (Avast). If either of these two processes are found, the malware will gracefully exit. If not, it continues with its workflow.



NtQuerySystemInformation Resolution (top and avp.exe and avastui.exe (bottom)

Following this step, the malware decodes a second set of strings. These are used to create persistence via the CurrentVersion\Run key under an entry named “OneDrive Update.”



Once the malware has created persistence key, it writes a second-stage DLL to the "C:\ProgramData\VMWare\" directory and calls it using the previously decoded parameters.

Second Stage

Analysis of the second-stage payload is in-flight and additional details beyond what is listed below will be provided when available. It is possible (and perhaps likely, due to time constraints) that another researcher will complete this analysis before I do, in which case those details will be corroborated and added below for completeness, along with the appropriate credit.

MD5 – f5475608c0126582081e29927424f338
SHA-1 – 8e88fd82378794a17a4211fbf2ee2506b9636b02
SHA-256 – a75886b016d84c3eaacaf01a3c61e04953a7a3adf38acf77a4a2e3a8f544f855

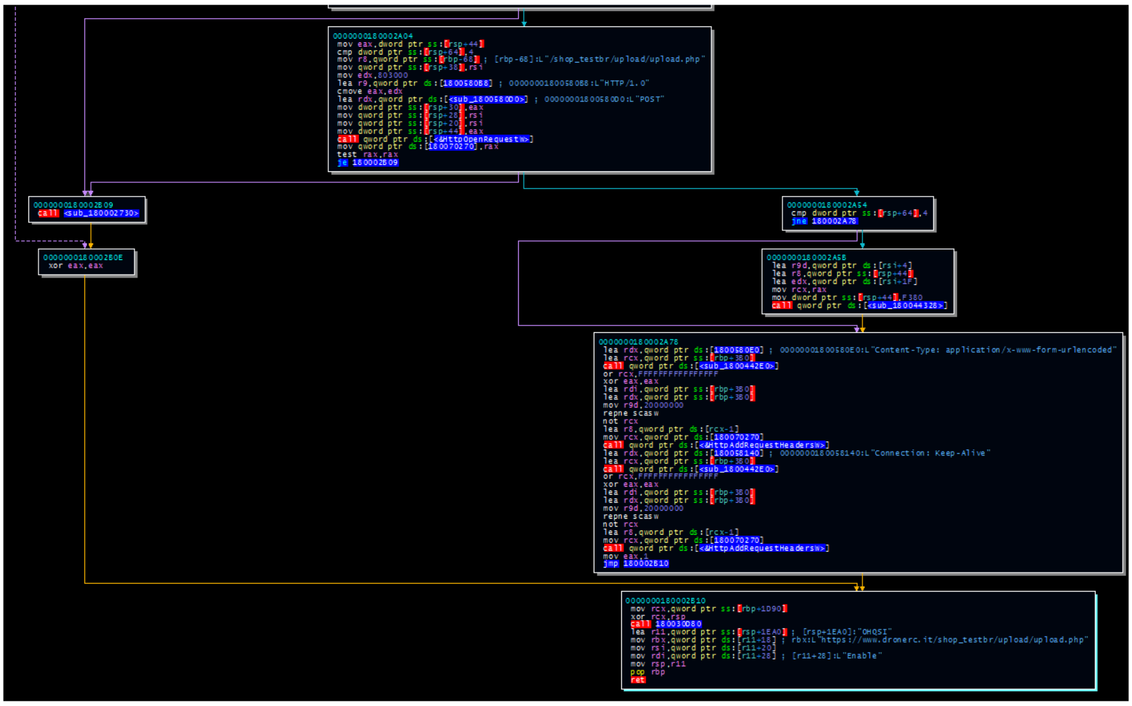
The second-stage malware performs a similar command line check to verify that it is running with two supplied parameters. The first of these parameters, lxUi5CZ0IV45j89Y, is used as to create a mutex to ensure that only one copy of the malware is running at a time. If the mutex already exists, the malware will exit.

The malware then resolves a long list of API calls before jumping in to a section in memory. While this list is extensive, they indicate potential functionality, including C2 operations (HttpOpenRequest, HttpAddRequestHeaders, etc) and host-based operations (GetDesktopWindow, WriteFile).

The sample examined contains multiple C2 domains and URLs, each of which contain a different endpoint for communication.

hxxps:// codevexillium[.]org/image/download/download.asp
hxxps:// www.dronerc[.]it/shop_testbr/upload/upload.php
hxxps:// transplugin[.]io/upload/upload.asp

The malware uses the OpenSSL library and communicates to these endpoints via POST request:



HTTP request (right click and open in new tab to zoom in)

As part of the POST request, the malware transmits the date and time of the malware's execution to the C2 server (encoded in Base64 format). This may be used for additional filtering by the attackers, to ensure that too much time hasn't passed between the malware's execution and communication (which could be indicative of an active debugging/reverse engineering effort).

At this stage, the actions available to the malware as a response to this POST request remain under analysis, although the malware presumably at a minimum provides basic reconnaissance and a channel for command-line execution (either directly or through creation of another payload). These are assumptions based on common malware characteristics, however, and not observed activity.* (Updated below)

***Update (1/26):**

Over the course of the last 24 hours, a lot of great research and analysis came to light from various parties. Most notably, I'd like to direct readers to three posts that offer additional context and demonstrate that the final action after this POST request is to download an additional payload onto the disk:

-
  - 360 Threat Intelligence Center provides additional operational context for these attacks, including social engineering. This may be particularly valuable for threat hunters or threat intelligence practitioners. It also offers more details regarding the POST request and next-stage DLL in similar samples.
  - Qi'anxin Threat Intelligence Center identified similar activity (and malware) from this adversary in September 2020.
  - Anheng Threat Intelligence Center provides additional context regarding the social engineering and Visual Studio stages of this attack.

Post navigation
