

Avast finds compromised Philippine Navy certificate used in remote access tool

By Threat Research TeamThreat Research Team

Archived: 2026-04-05 17:43:42 UTC

Avast Threat Intelligence Team has found a remote access tool (RAT) actively being used in the wild in the Philippines that uses what appears to be a compromised digital certificate belonging to the Philippine Navy. This certificate is now expired but we see evidence it was in use with this malware in June 2020.

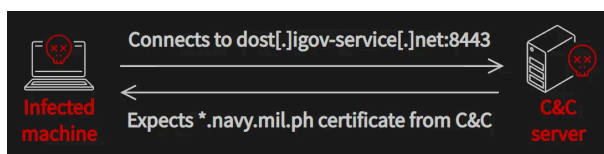
Based on our research, we believe with a high level of confidence that the threat actor had access to the private key belonging to the certificate.

We got in touch with [CERT-PH, the National Computer Emergency Response Team for the Philippines](#) to help us contact the navy. We have shared with them our findings. The navy security team later let us know that the incident has been resolved and no further assistance was necessary from our side.

Because this is being used in active attacks now, we are releasing our findings immediately so organizations can take steps to better protect themselves. We have found that this sample is now [available on VirusTotal](#).

Compromised Expired Philippine Navy Digital Certificate

In our analysis we found the sample connects to `dost[.]igov-service[.]net:8443` using TLS in a statically linked OpenSSL library.



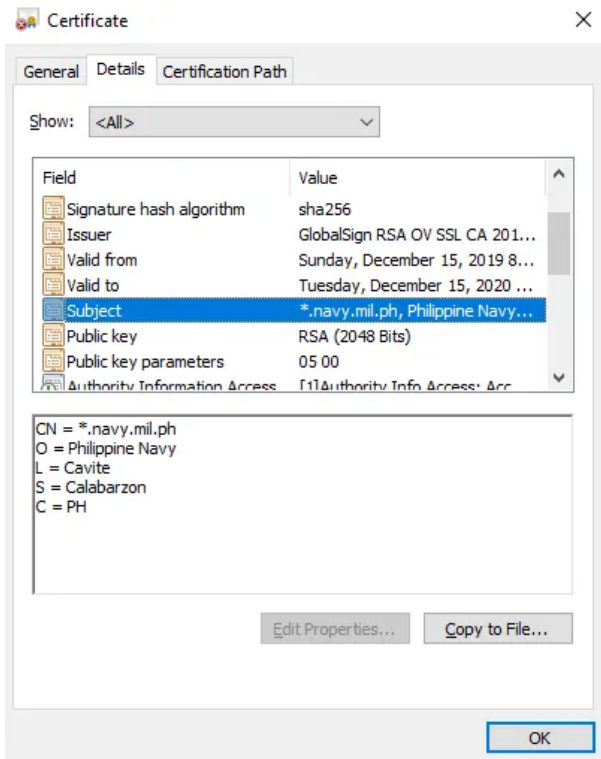
A WHOIS lookup on the C&C domain gave us the following:

Whois Lookup ?

Create date: 2019-12-09
Domain name: igov-service.net
Domain registrar id: 146
Domain registrar url: http://registrar.godaddy.com
Expiry date: 2021-12-09
Name server 1: ns01.domaincontrol.com
Name server 2: ns02.domaincontrol.com
Query time: 2019-12-10 17:42:29
Registrant address: 3267309318f7846c
Registrant city: 3267309318f7846c
Registrant company: 3267309318f7846c
Registrant country: United States
Registrant email: 3267309318f7846cs@
Registrant fax: 3267309318f7846c
Registrant name: 3267309318f7846c
Registrant phone: 3267309318f7846c
Registrant state: 40840a16fcb405fc
Registrant zip: 3267309318f7846c
Update date: 2019-12-09

The digital certificate was pinned so that the malware requires the certificate to communicate.

When we checked the digital certificate used for the TLS channel we found the following information:



Some important things to note:

- The certificate is a valid certificate with a subject of `*.navy.mil.ph`, the Philippine Navy.
- The certificate has recently expired: it was valid for one year, from Sunday December 15, 2019 until Tuesday December 15, 2020.
- Our research shows that [Censys](#) saw [this certificate employed by the actual navy.mil.ph website](#)

Based on our research, we believe with a high level of confidence that the threat actor had access to the private key belonging to the certificate.

While the digital certificate is now expired we see evidence it was in use with this malware in June 2020.

The malicious PE file was found with filename: `C:\Windows\System32\wlbsctrl.dll` and its hash is: `85FA43C3F84B31FBE34BF078AF5A614612D32282D7B14523610A13944AADAACB` .

In analyzing that malicious PE file itself, we found that the compilation timestamp is wrong or was edited. Specifically, the `TimeDateStamp` of the PE file was modified and set to the year 2004 in both the PE header and Debug Directory as shown below:

```

00000000180266710 ; Debug Directory entries ; Sub_18000FF0+0177 ...
00000000180266710 dd 0 ; Characteristics
00000000180266710 dd 40E13AA1h ; TimeDateStamp: Tue Jun 29 09:47:13 2004
00000000180266710 dw 0 ; MajorVersion
00000000180266710A dw 0 ; MinorVersion
00000000180266710C dd 00h ; Type: IMAGE_DEBUG_TYPE_POGO
00000000180266720 dd 3BCh ; SizeOfData
00000000180266724 dd rva aGctl ; AddressOfRawData
00000000180266726 dd 2005h ; PointerToRawData
0000000018026672C dd 0 ; Characteristics
00000000180266734 dd 40E13AA1h ; TimeDateStamp: Tue Jun 29 09:47:13 2004
00000000180266736 dw 0 ; MajorVersion
00000000180266738 dw 0 ; MinorVersion
0000000018026673C dd 00h ; Type: IMAGE_DEBUG_TYPE_ILTCG
00000000180266740 dd 0 ; SizeOfData
00000000180266744 dd 0 ; AddressOfRawData
00000000180266748 dd 0 ; PointerToRawData
00000000180266748 align 10h

00000000180001000 ; Format : Portable executable for AMD64 (PE)
00000000180001000 ; Imagebase : 18000000
00000000180001000 ; TimeStamp : 40E13AA1 (Tue Jun 29 09:47:13 2004)
00000000180001000 ; Section 1: (virtual address 00001000)
00000000180001000 ; Virtual size : 001CA5D1 (1877457.)
00000000180001000 ; Section size in file : 001CA600 (1877504.)
00000000180001000 ; Offset to raw data for section: 00000400
00000000180001000 ; Flags 6000020: Text Executable Readable
00000000180001000 ; Alignment : default
00000000180001000 ; OS type : MS Windows
00000000180001000 ; Application type: DLL
00000000180001000

```

However, we found that the author used OpenSSL 1.1.1g and compiled it on April 21, 2020 as shown below:

```

data:000000001801E6772 db 0
data:000000001801E6773 db 0
data:000000001801E6774 db 0A2h
data:000000001801E6775 db 2
data:000000001801E6776 db 0
data:000000001801E6777 db 0
data:000000001801E6778 a0penssl111g21A db 'OpenSSL 1.1.1g 21 Apr 2020',0
data:000000001801E6794 db 0
data:000000001801E6795 db 0
data:000000001801E6796 db 0
data:000000001801E6797 db 0

```

The username of the author was probably `udste` . This can be seen in the debug information left inside the used OpenSSL library.

```

1E6798 db 0
1E6799 db 0
1E679A db 0
1E679B db 0
1E679C db 0
1E679D db 0
1E679E db 0
1E679F db 0
1E67A0 ; const char aUsersUdsteDes[]
1E67A1 ; const char file[]
1E67A2 ; const char file1[]
1E67A3 ; const char file2[]
1E67A4 ; const char file3[]
1E67A5 ; const char file4[]
1E67A6 ; const char file5[]
1E67A7 ; const char file6[]
1E67A8 ; const char file7[]
1E67A9 ; const char file8[]
1E67AA ; const char file9[]
1E67AB ; const char file10[]
1E67AC ; const char file11[]
1E67AD ; const char file12[]
1E67AE ; const char file13[]
1E67AF ; const char file14[]
1E67B0 ; const char file15[]
1E67B1 ; const char file16[]
1E67B2 ; const char file17[]
1E67B3 ; const char file18[]
1E67B4 ; const char file19[]
1E67B5 ; const char file20[]
1E67B6 ; const char file21[]
1E67B7 ; const char file22[]
1E67B8 ; const char file23[]
1E67B9 ; const char file24[]
1E67BA ; const char file25[]
1E67BB ; const char file26[]
1E67BC ; const char file27[]
1E67BD ; const char file28[]
1E67BE ; const char file29[]
1E67BF ; const char file30[]
1E67C0 ; const char file31[]
1E67C1 ; const char file32[]
1E67C2 ; const char file33[]
1E67C3 ; const char file34[]
1E67C4 ; const char file35[]
1E67C5 ; const char file36[]
1E67C6 ; const char file37[]
1E67C7 ; const char file38[]
1E67C8 ; const char file39[]
1E67C9 ; const char file40[]
1E67CA ; const char file41[]
1E67CB ; const char file42[]
1E67CC ; const char file43[]
1E67CD ; const char file44[]
1E67CE ; const char file45[]
1E67CF ; const char file46[]
1E67D0 ; const char file47[]
1E67D1 ; const char file48[]
1E67D2 ; const char file49[]
1E67D3 ; const char file50[]
1E67D4 ; const char file51[]
1E67D5 ; const char file52[]
1E67D6 ; const char file53[]
1E67D7 ; const char file54[]
1E67D8 ; const char file55[]
1E67D9 ; const char file56[]
1E67DA ; const char file57[]
1E67DB ; const char file58[]
1E67DC ; const char file59[]
1E67DD ; const char file60[]
1E67DE ; const char file61[]
1E67DF ; const char file62[]
1E67E0 ; const char file63[]
1E67E1 ; const char file64[]
1E67E2 ; const char file65[]
1E67E3 ; const char file66[]
1E67E4 ; const char file67[]
1E67E5 ; const char file68[]
1E67E6 ; const char file69[]
1E67E7 ; const char file70[]
1E67E8 ; const char file71[]
1E67E9 ; const char file72[]
1E67EA ; const char file73[]
1E67EB ; const char file74[]
1E67EC ; const char file75[]
1E67ED ; const char file76[]
1E67EE ; const char file77[]
1E67EF ; const char file78[]
1E67F0 ; const char file79[]
1E67F1 ; const char file80[]
1E67F2 ; const char file81[]
1E67F3 ; const char file82[]
1E67F4 ; const char file83[]
1E67F5 ; const char file84[]
1E67F6 ; const char file85[]
1E67F7 ; const char file86[]
1E67F8 ; const char file87[]
1E67F9 ; const char file88[]
1E67FA ; const char file89[]
1E67FB ; const char file90[]
1E67FC ; const char file91[]
1E67FD ; const char file92[]
1E67FE ; const char file93[]
1E67FF ; const char file94[]

```

We found that the malware supported the following commands:

- run shellcode
- read file
- write file
- cancel data transfer
- list drives
- rename a file
- delete a file

- list directory content

```
: enum COMMANDS, mappedto_2257, width 2 bytes
RUN_SHELL_COMMAND = 1
READ_FILE         = 5
WRITE_FILE        = 6
CANCEL_TRANSFER   = 7
GetDrives         = 8
DoPathRename      = 9
DoPathDelete      = 0Ah
GetDirectory      = 0Ch
QUIT              = 1Eh
```

Some additional items of note regarding the malicious PE file:

- All configuration strings in the malware are encrypted using AES-CBC with the exception of the mutex it uses. That mutex is used as-is without decryption: t7As7y9I6EGwJ0QkJz1oRvPUFx1CJTsjzgdIm0CxIa4= .
- When this string is decrypted using the hard-coded key it decrypts to QSR_Mutex_zGkwwAejTD9sDitYcK . We suspect that this is a failed attempt to disguise this malware as the infamous Quasar RAT malware. But this cannot be the case because this sample is written in C++ and the Quasar RAT is written in C#.

Avast customers are protected against this malware.

Indicators of Compromise (IoC)

- Repository: <https://github.com/avast/ioc/tree/master/Philippine-Navy-Certificate>

SHA256	File name
85FA43C3F84B31FBE34BF078AF5A614612D32282D7B14523610A13944AADAACB	C:\Windows\System32\wlbsctrl.dll
Mutex	
t7As7y9I6EGwJ0QkJz1oRvPUFx1CJTsjzgdIm0CxIa4=	
C&C server	
dost[igov-service]net:8443	



A group of elite researchers who like to stay under the radar.

Source: <https://decoded.avast.io/threatintel/avast-finds-compromised-philippine-navy-certificate-used-in-remote-access-tool/>