Static unpacker and decoder for Hello Kitty Packer

By Brenton Morris

Published: 2022-04-25 · Archived: 2026-04-05 17:56:57 UTC

During a recent incident response engagement, the Profero IR team observed a sample of Hello Kitty ransomware. This version of ransomware is intriguing as this sample is packed with a packer written in Go. This packer decrypts the final Hello Kitty payload, which is written in C++, before executing it in memory. The Hello Kitty ransomware is written as a simple tool that an attacker can use to encrypt data on the victim's machine and not as a full-fledged malware with persistence methods of its own. This malware has been covered by previous researchers in-depth, however, there is much less information about the packer used by this ransomware gang.

Due to this fact, we are releasing this report along with a tool that can be used to unpack the payload contained within the Go packer.

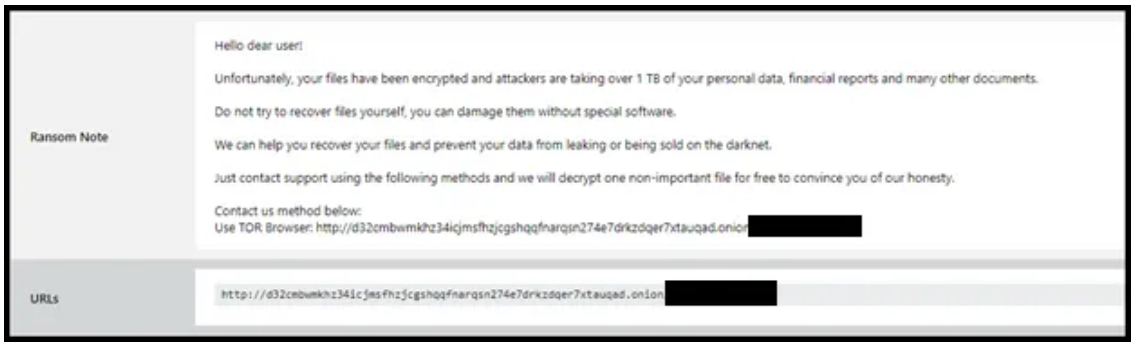
Analysis

Overview

The use of a Go packer makes it hard for reverse engineers to analyze the binary and assists the malware in evading detection by antivirus and other detection systems. This is due to the low detection rate on Go binaries and due to the nature of packers themselves, as they encrypt the final malicious payload to prevent signature-based detections.

Get Brenton Morris's stories in your inbox

Below is an example of the ransom note used by HelloKitty (extracted using the https://hatching.io/ sandbox):



Ransom note used in this attack

When analyzing the unpacked payload, we can see that the ransomware is not written to install on a machine but rather it is written as a command-line tool that attackers will use after gaining access to target machines. The unpacked tool even provides a command-line help message. This can be seen in the screenshot below.

Press enter or click to view image in full size

```
OUTPUT: Registered ctrl handlers
OUTPUT: Help usage:
OUTPUT:            -path | select path; bin.exe -path "c:\temp"
OUTPUT:           -limit | limit file offset; bin.exe -limit 10 (limit first 10 megabytes of file)
OUTPUT:          -blocks | use random blocks encryption; bin.exe -blocks
OUTPUT:         -maxrand | maximum random block range[0..maxrand); bin.exe -maxrand 10
OUTPUT:           -nosum | no use checksum in enc/dec; bin.exe -nosum
OUTPUT:       -norecycle | do not clear recycle; bin.exe -norecycle
OUTPUT:       -noshadows | do not remove shadows; bin.exe -noshadows
OUTPUT:             -log | log output to file format 'log_%pid%.log'; bin.exe -log
OUTPUT:             -key |
OUTPUT:         -visible | do not hide console window
OUTPUT:            -test | run self tests
OUTPUT:           -tests | run self tests
OUTPUT:        -tempfile | set temp file name
OUTPUT:           --help | print this help
OUTPUT:            -help | print this help
```

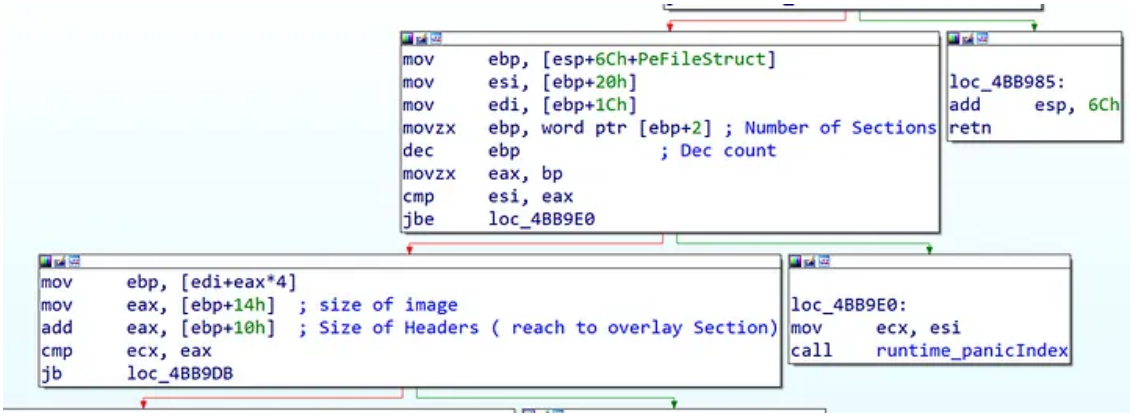
Command help message

Packer Decryption

The packer's decryption process is as follows:

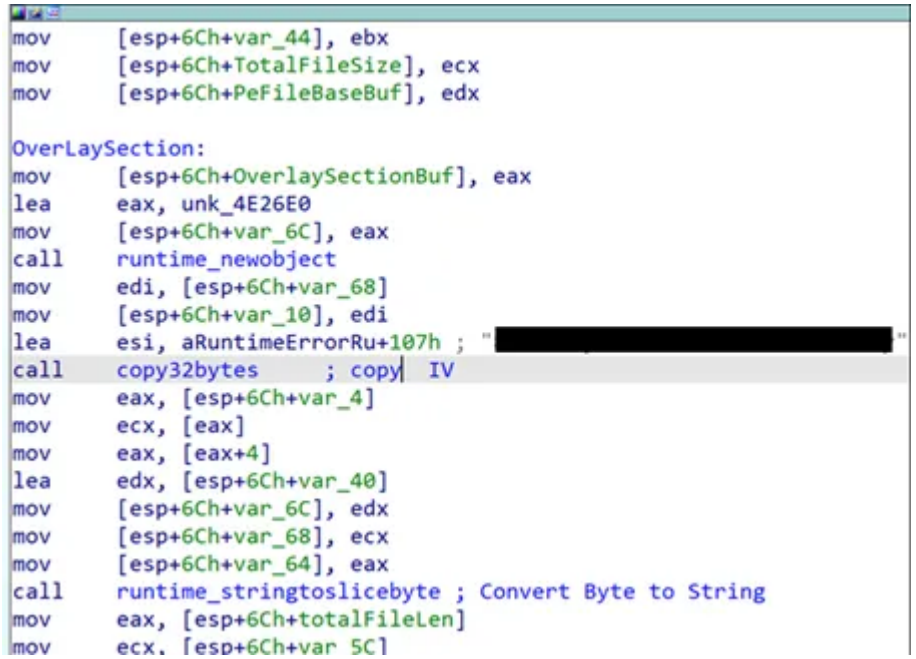
- The packed binary is executed and passed a 16-byte decryption key as a command-line parameter which the packer will use to decrypt the payload
- The encrypted blob is located at the overlay of the packed binary. The location in the file is obtained by parsing the PE header from the binary:

Press enter or click to view image in full size



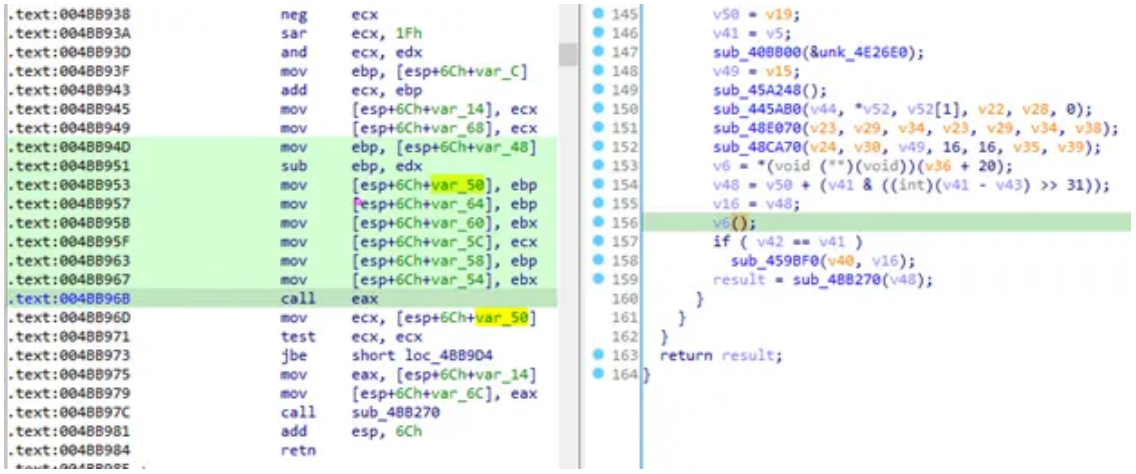
Parsing PE headers

- This encrypted blob is then decrypted using the AES-128-CBC algorithm with IV passed as an embedded string from the binary, this can be seen in the image below:



Locating the IV

- The packer then resolves the entry point of the payload and passes execution to it:



Finding and jumping to the entry point of the packed binary

Unpacking Tool

To assist in the analysis, the Profero team has developed a tool that can be used to unpack the binary and do the following:

- Check if the packed binary is a hello kitty binary
- Extracts the RSA key, C2 server used, IV, etc
- Unpacks the packed file

The tool can be executed as follows:

./HelloKittyUnpacker.exe [input] [key] [dump]

We hope that this tool will assist in speeding up analysis in any future incidents involving this malware. In addition, we wanted to open source the code so it can be used as a blueprint for similar malware.

It can be found on GitHub:

https://github.com/proferosec/HelloKittyUnpacker
