HawkEye Malware Changes Keylogging Technique

By sharon

Published: 2019-08-13 · Archived: 2026-04-05 18:35:17 UTC

Cyberbit Labs have observed HawkEye malware variants changed their keylogging technique. Until now, the most pervasive keylogger malware technique was to register a procedure into the message hook chain of a window using SetWindowsHookExA API. The new variants exploit RegisterRawInputDevices API to register for input from the keyboard. This technique is not a new one. it has been used in Metasploit. but this is the first time it has been observed at use in a HawkEye malware sample.

Hawkeye malware is sold as malware-as-a-service. Over the years, Hawkeye malware has been updated and improved. New capabilities and techniques are added to it from time to time.

Its top targeted industries include software and tech, banking, energy, chemical and automotive.

In this blog post, I will explain and compare the old and the new keylogging techniques used by Hawkeye malware. The device keylogging technique is less common among malware – and was likely adopted by malware authors as an effective way to evade detection by security products that do not intercept all kind of keylogging techniques.

I will also demonstrate how Cyberbit EDR solution is able to successfully detect all keylogging techniques.

Two Hawkeye Malware Keylogging Techniques

Sample 1 – Using SetWindowsHookExA

SHA256: dff7820b2b0b877c46a0cbc6be22b45b11511af7d50e36c7d83fa27f3db277b0

This Hawkeye malware `sample is written in C# and obfuscated. As with most HawkEye samples, it performs process hollowing to its child process – RegAsm.exe – a legitimate Microsoft software used for .NET assembly registration.

The keylogging is done from the hollowed RegAsm.exe


Figure 1 – Entry point of the sample is the Main() of groanwarriorer. Names vary between meaningful words to non-meaningful ones.


Figure 2 – HawkEye spawns a child process, and injects its code into it.

We put a breakpoint on user32.dll!SetWindowsHookExA at the process RegAsm.exe just after user32.dll was loaded. This DLL exports the function SetWindowsHookExA.

This function is used to register a procedure to the message hook chain of a window. After registration, this procedure processes window messages and looks for keyboard messages (using two types of hooks – WH_KEYBOARD or WH_KEYBOARD_LL) – and retrieves from them the key pressed by the user.

We got a breakpoint hit and can see with which parameters SetWindowsHookExA was called:

SetWindowsHookExA(0xD, 0x7D18EA,0,0)

idHook = 0xD = WH_KEYBOARD_LL – Type of hook to install

lpfn = 0x7D17EA – Address of procedure that processes the windows messages

hmod = 0 – NULL since the procedure is within the code of the current process and all the threads running in the same desktop are monitored (see last parameter)

dwThreadId = 0 – Install the hook on all currently existing threads in the same desktop as the calling thread


Figure 3 – Breakpoint hit and the stack on SetWindowsHookExA

The logged keystrokes are later sent to the attacker at a the chosen protocol. Keystrokes can be sent via SMTP (mail), FTP or HTTP, depending on how the sample was configured.

Cyberbit EDR detects this keylogging technique executed from RegAsm.exe

Figure 4 – SetWindowsHookExA keylogger method used by RegAsm.exe


Figure 4 – SetWindowsHookExA keylogger method used by RegAsm.exe

Sample 2 – Using RegisterRawInputDevices

SHA256: a5b2f2fc5b08b09d95302786304f6e4b05b0d326fae8a7fbad6da72ef6e61f25

As in the previous sample, this is an obfuscated C# sample that creates a suspended RegAsm.exe and injects into it. However, this time the breakpoint on user32.dll!SetWindowsHookExA didn't hit.

There are many ways to log user input, one of them is registering to a raw input device – such as a keyboard, mouse, joystick, etc.

The Windows API RegisterRawInputDevices is the one used for registering to raw input devices messages. After registration, the WM_INPUT message should be processed in order to retrieve the key pressed. For an example of how it can be done read Windows Keylogger Part 1

We put a breakpoint on user32.dll!RegisterRawInputDevices at the injected RegAsm.exe and got a hit.


Figure 5 – 1. The breakpoint hit on RegisterRawInputDevices. 2. The stack at the time the breakpoint hits. 3. The structure at 0x54FF70 contains a RAWINPUTDEVICE structure. 4. The last

member of the structure at 0x000C0366 is a handle to a window

This function receives 3 parameters:

- *PCRAWINPUTDEVICE pRawInputDevices*
- *UINT uiNumDevices*
- *UINT cbSize*

pRawInputDevices is an array of RAWINPUTDEVICE structures that represent the devices that supply the raw input.

uiNumDevices is The number of RAWINPUTDEVICE structures pointed to by pRawInputDevices.

cbSize is the size, in bytes, of a RAWINPUTDEVICE structure.

The struct RAWINPUTDEVICE looks like this:

typedef struct tagRAWINPUTDEVICE {

USHORT usUsagePage;

USHORT usUsage;

DWORD dwFlags;

HWND hwndTarget;

*} RAWINPUTDEVICE, *PRAWINPUTDEVICE, *LPRAWINPUTDEVICE;*

usUsagePage specifies the type of device.

usUsage specifices the device within the group of usUsagePage.

dwFlags is a mode flag that specifics how to interpret the information between usUsagePage and usUsage.

hwndTarget is a handle to the target window

Let's look at the parameters passed by the keylogging malware (figure 5):

RegisterRawInputDevices(0x54FF70,1,0xC)

PRawInputDevices = 0x54ff70

At address 0x54FF70 we can see one structure of RAWINPUTDEVICE with the following values:

usUsagePage = 0x0001 – Generic desktop controls

usUsage = 0x0006 – Keyboard

dwFlags = 0x00000100 – RIDEV_INPUTSINK – If set, this enables the caller to receive the input even when the caller is not in the foreground. Note that hwndTarget must be specified.

hwndTarget = 0xc0366

uiNumDevices = 1 – only one structure

cbSize = 0xC – Sum of two USHORT variables, one DWORD and HWND which is also DWORD in size = 2 + 2 + 4 + 4 = 0xC

It is clear now that the malware registered for WM_INPUT messages from the keyboard in order to monitor which keys are pressed.

This keylogging technique is also used by the well-known attack simulation platform Metasploit

Cyberbit EDR also detects this technique, executed from RegAsm.exe:


Hawkeye Malware Keylogging figure 6

Figure 6 – RegisterRawInputDevices keylogger method used by RegAsm.exe

Watch HawkEye Malware Analysis Video:

To Learn More About How to Defend against Hawkeye and other fileless malware – Download Free Whitepaper:

Source: https://www.cyberbit.com/blog/endpoint-security/hawkeye-malware-keylogging-technique/