G Data

Whitepaper 03/2014

Keylogger Protection

System Security Research

G Data. Security Made in Germany.

Motivation

The reason why computer malware is developed and used by attackers has changed greatly in recent years. It started out for the most part as just technical shenanigans and bravado on the part of the malware authors, but today it is almost entirely driven by financial or political motives. The black market that has built up around the creation and distribution of computer malware turns over millions of dollars a year. Furthermore, nations are consciously using malware to spy on and even attack other nations (Zetter, 2013).

In such cases the attackers are concerned with collecting and stealing information, such as email accounts, passwords, social media accounts or bank account and credit card details. The malware programs responsible for this are also called *information stealers*. Recording the victim's keystroke input is a critical part of this. More than three out of four malware programs contain this so-called *keylogger* functionality (Symantec, 2008).

The widespread distribution of keylogger functionality in malware is not surprising when you think about the number of situations in which entire digital identities can be stolen merely by capturing keyboard input. Traditional methods of securing user accounts involve user identification (a user name) and an associated password. This generally applies for online gaming, email accounts, access to social networks, and logins for online payment systems such as PayPal and for online bank accounts. Additional factors for authenticating a user's identity are only required in very critical areas. For example, online banking transfers can usually only be performed with the additional use of a transaction number; actually logging in to the bank account, however, does not generally bear any additional security. A keylogger is all that is needed for stealing all of this information.

How seriously (German) computer users are compromised has been demonstrated in information recently published by the German Federal Office for Information Security (BSI). According to this, 16 million stolen digital identities have been discovered and recovered (BSI, 2014). Keyloggers are thought to have been used in the theft (Golem, 2014). In another case involving keylogger botnets, two million stolen sets of access data were recovered, half of them for Facebook (Trustwave, 2013).

Even manufacturers of computer games have been warning their users for years of the dangers of information stealers, which are used to steal online gaming accounts (Blizzard, 2010). How worthwhile it can be for attackers to steal user accounts is evident in a case of account theft involving a professional online poker player, where the attacker – presumably by using a keylogger – was able to steal over 100,000 US dollars in a single case (OnlinePoker.net, 2012).

According to research, there was a new victim of identity theft every three seconds in 2013. This gave rise to damages of 21 billion US dollars (Javelin, 2013).

How does Windows process keystrokes?

Windows has device drivers for keyboards in the kernel that are able to translate the hardware information sent by a keyboard into keyboard input that the operating system can use. This translated keyboard input is then passed by Windows via the internal drivers to the entire system in a globally readable structure (*global key state*, see diagram below). At the same time the foreground window indicates that a particular key has been pressed by writing the keyboard input to the foreground window's buffer (*key buffer*). The window can read and process this buffer. There are completely legitimate situations in which another process can read the keyboard input in another window. For example, there is speech synthesis software that pronounces every character pressed to make computer use easier for visually impaired users. The *global key state* is entitled to do so as well when, for example, a background window needs to react to specific keyboard input but the information on the key currently being pressed is only contained in the foreground window.

How do attackers use keyloggers?

The technology used by attackers can be split into three types:

- The simplest way is to query the *global key state* cyclically every few milliseconds and hence log each of the keys currently being pressed.
- Another option provided by Windows is to read the *key buffer* of the window concerned. The benefit of doing so for attackers is that they can see specifically which window is receiving which input. The majority of keyloggers uses this method.
- A somewhat newer method, which is not designated by Windows, is frequently used by information stealers in the banking Trojan sector. The best-known examples of this type, *ZeuS* and *SpyEye*, intercept the keyboard input directly in the attacked program's process, which is after the *key buffer* has been read. The technology used for this is called *hooking*. The G Data BankGuard technology has been specifically designed to detect hooking and remove unwanted hooks to prevent such attacks.

What protection is there against keyloggers?

Keylogger Protection addresses the problem of programs being able to read the *global key state* or the actual *key buffer* of a window, which is the first two of the attack methods described above.

It does so by installing a filter driver in the kernel: this receives every keystroke before it is sent to the Windows driver. This enables keystrokes to be filtered out as if they had never occurred. The result is that the keystroke appears in neither the *global key state* nor the *key buffer*, thus preventing malware from intercepting the input data.

However, so that the keystrokes are not simply filtered out, the keys that have been pressed are obviously then added back into the system by sending them directly to the foreground window. This side channel ensures that Windows cannot determine that a particular key has been pressed. Windows simply knows that input has occurred in the foreground window.

Virtual keyboards do not offer sufficient protection

It is repeatedly alleged that a *virtual keyboard* is the only practical solution for stopping keyloggers. However, for various reasons virtual keyboards are no more secure than the current approach using Keylogger Protection. In fact, ultimately they are even easier to get around.

A virtual keyboard is basically a graphic interface representing a keyboard. To simulate keyboard input, the user must use a mouse to click on the corresponding key in the graphic. The key then appears as keyboard input in the foreground window. Because no actual keyboard is used here, there are ostensibly no keystrokes for a keylogger to intercept either. In this regard the protection is considered to be equivalent to Keylogger Protection as described above.

One advantage of virtual keyboards over G Data Keylogger Protection should be acknowledged: if the keylogger has not been installed by the attacker as software but attached as a physical device (commonly called "bug" or "wiretap") between the keyboard and the PC, only a virtual keyboard will provide protection as no actual input occurs on the physical keyboard in this case. However, it can be assumed that an attacker with physical access to a PC is also capable of installing keylogger software, for example, or attaching a camera to record the input on a virtual keyboard.

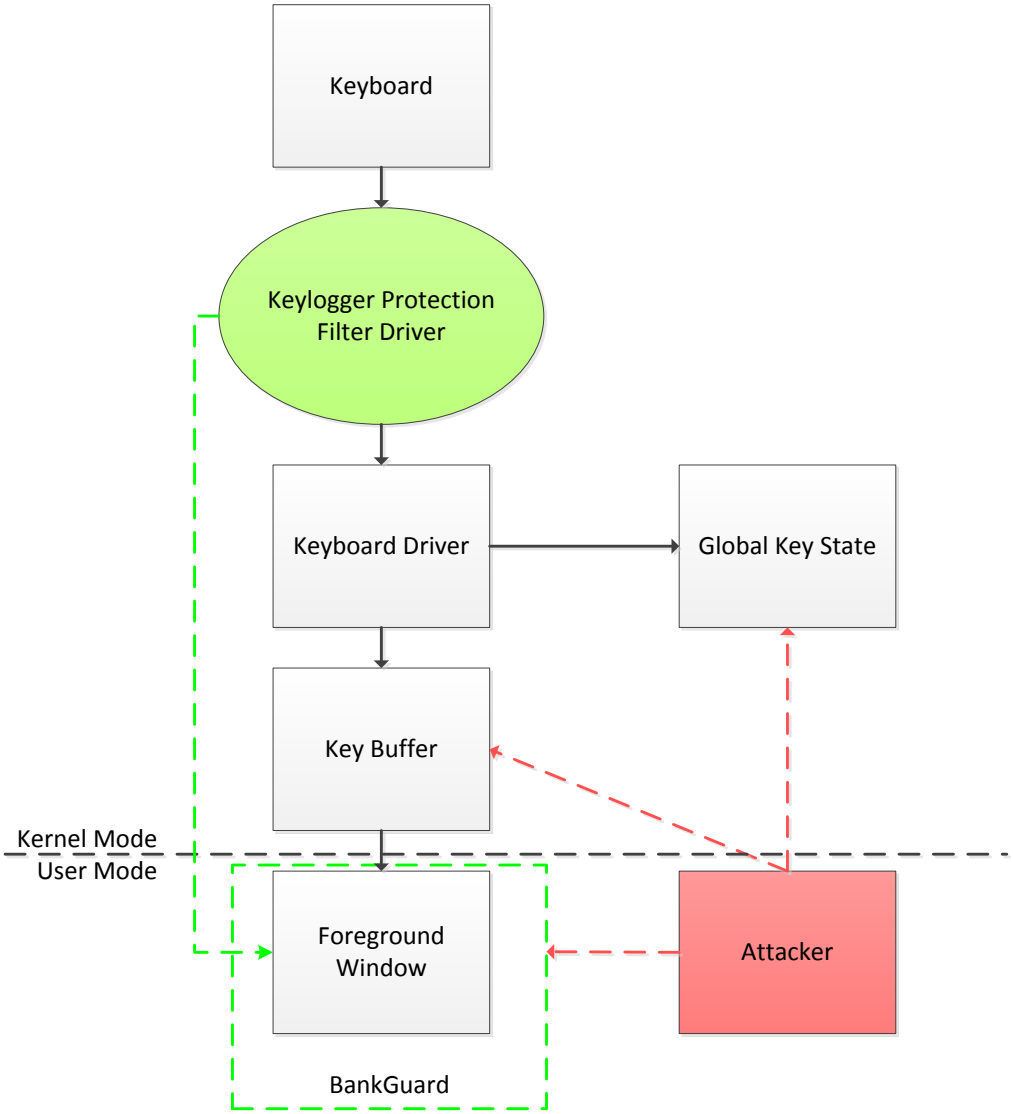
Besides this, a virtual keyboard also offers crucial disadvantages: the very limited user convenience only enables use in specific situations, such as inputting a password. Nobody would want to write an entire document by clicking on a graphic interface. Hence the rest of the normal user input remains unprotected.

If a user forgets to use the virtual keyboard to enter his password just once, installed keyloggers can immediately intercept the passwords. It is also known that current banking Trojans have a screenshot function that snaps the screen at regular intervals when specific websites are being accessed. This makes it easy to photograph the key currently being selected on the virtual keyboard graphic interface (IOActive, 2012).

In addition to this, there is a conceptual problem with using virtual keyboards. If a system is infected, no sensitive data should be entered onto the system until it has been disinfected. If a system is not infected, there is no threat to the user's data and so the method of keyboard input used is of no

consequence. The average computer user cannot judge when his system is compromised and when it is not. Consequently, using a virtual keyboard is just a precautionary measure that gets in the way of user convenience.

For this reason, unlike a virtual keyboard, Keylogger Protection is always active when a browser is being used. The keyboard input is protected without the user's intervention, while user convenience is retained in full.



Processing keyboard input in Windows and the concept behind Keylogger Protection

List of protected processes

As described above, when Keylogger Protection is enabled, Windows cannot detect that a keystroke has been made, just that the foreground process is reporting this. Because Keylogger Protection holds back the user input from large parts of the operating system, this can occasionally lead to unexpected behaviour. For example, 3D applications that communicate with the Windows keyboard driver directly for reasons of speed would lose speed. The keyboard driver never sees the keyboard input, because it is filtered out by the Keylogger Protection filter driver.
To prevent such problems, Keylogger Protection is just temporarily enabled for the browser, not generally – the filter driver is enabled or disabled dynamically, depending on whether there is currently a browser in the foreground or not.

The list of protected processes can be adapted via signature updates.

Current status (17 March 2014):

Browser:

- Microsoft Internet Explorer
- Mozilla Firefox
- Google Chrome

Bibliography

Blizzard. (2010). *us.battle.net.* Downloaded from http://us.battle.net/wow/en/forum/topic/1015381745

BSI. (2014). *BSI-Sicherheitstest.* Downloaded from https://www.sicherheitstest.bsi.de

CrySys. (2011). *Duqu: A Stuxnet-like malware found in the wild.* Downloaded from http://www.crysys.hu/publications/files/bencsathPBF11duqu.pdf

Golem. (2014). *BSI-Warnung - Fragen und Antworten zum Identitätsdiebstahl*. Downloaded from Golem.de: http://www.golem.de/news/bsi-warnung-fragen-und-antworten-zum-identitaetsdiebstahl-1401-104118.html

IOActive. (2012). *Reversal and Analysis of Zeus and SpyEye Banking Trojans.* Downloaded from http://www.ioactive.com/pdfs/ZeusSpyEyeBankingTrojanAnalysis.pdf

Justice, S. B. (2013). Downloaded from http://www.bjs.gov/content/pub/pdf/vit12.pdf

OnlinePoker.net. (2012). Downloaded from http://www.onlinepoker.net/poker-news/general-poker-news/hacker-steals-140k-lock-poker-account/16705

Symantec. (2009). *Security Threat Report.* Downloaded from http://eval.symantec.com/mktginfo/enterprise/white_papers/b-whitepaper_internet_security_threat_report_xiv_04-2009.en-us.pdf

Symantec. (2008). *Internet Security Threat Report XIV.* Downloaded from http://eval.symantec.com/mktginfo/enterprise/white_papers/b-whitepaper_exec_summary_internet_security_threat_report_xiv_04-2009.en-us.pdf

Zetter, K. (2013). *Wired.com.* Downloaded from http://www.wired.com/threatlevel/2013/03/stuxnet-act-of-force/

