In God We Trust All Others We Monitor

[Extended Abstract]

Patrick Stewin and Jean-Pierre Seifert
Security in Telecommunications, Berlin Institute of Technology
Ernst-Reuter-Platz 7
10587 Berlin, Germany
patrickx,jpseifert@sec.t-labs.tu-berlin.de

ABSTRACT

Modern x86 platforms offer stealth capabilities, that are exploited by rootkits to hide malicious code as shown by the *rootkit evolution.* Recently, security researchers discovered a very powerful execution environment for rootkits that is isolated from the actual x86 host platform. According to the capabilities of the isolated environment the researches called it "ring -3". Security mechanisms, such as antivirus software, cannot reveal "ring -3" rootkits, since they are executed in the operating system which makes them unable to access "ring -3".

Agencies could use "ring -3" to host Remote Forensic Investigation Software, that is able to stealthily spy on suspects. This inevitably raises the interesting question if *provable* stealth government software (GovWare) can exist at all.

In this work, we aim to expose the risks that come from that mass technology with regard to privacy concerns. With undetectable GovWare – executed on mass technology like the x86 platform – a government could observe most of their citizens, automatically placing them under general suspicion. We developed a proof-of-concept (PoC) keystroke logger with the aim of identify countermeasures against that threat. Our PoC is able to read the whole host memory from within the "ring -3" environment.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection—*invasive software*; K.4.1 [**Computers and Society**]: Public Policy Issues—*abuse and crime involving computers, Privacy*

General Terms

Security

Keywords

Antivirus; Stealth Federal Trojan; GovWare; iAMT; Northbridge; Remote Forensic Investigation Software (RFIS); Intel x86; Covert Communication Channel

1. INTRODUCTION

Without any doubt, it is clear that the 9/11 terrorist attacks in the U.S. have produced, federal and national investigation tools for covert communication and government surveillance across the globe: Carnivore [4], Magic Lantern [1], Computer and Internet Protocol Address Verifier [5], EnCase Field Intelligence Model [6], etc. Due to the obvious secret nature of such Remote Forensic Investigation Software (RFIS) and the differing laws among different countries very few technical details and capabilities are known. Still, most RFIS are very similar to classical Trojans with strong stealth capabilities.

In this work, we examine the interesting question whether *provable* stealth government Trojans can exist at all. The basis for the examination is our demo of our implemented PoC keystroke logger.

1.1 Motivation

The research done in this work is motivated by the surveillance plans of governements. Government surveillance came up after the 9/11 terrorist attacks in the U.S. The Western civilization – not only the U.S., but especially the German Government increased their surveillance efforts. The discussion about the goals of the German Federal Ministry of the Interior is reminiscent of government surveillance (in an updated form) done by the secret police (Ministerium für Staatssicherheit, colloquial *Stasi*) of the former German Democratic Republic (GDR). The last German Interior Minister aimed to secretly spy on people by using their private computers connected to the Internet. The correspondent RFIS is called *German Federal Trojan* by critics. Critics go as far as to organize demonstrations and protests arguing for users' privacy. In October 2008, up to 70,000 people protested against "surveillance mania" in Berlin [2]. The criticism, discussion and protests against the government surveillance – especially the German Federal Trojan – established a new German political keyword: *Stasi 2.0.*

1.2 Contribution

In this work, we show how existing mass technology can be used to develop nearly perfect GovWare, that arises serious privacy concerns. We address this research issue from an academic point of view. Our objectives are to evaluate to which extent a: **(O1)** *stealth* infiltration of the target platform, **(O2)** *stealth* GovWare Trojan, and **(O3)** *stealth* authentic outbound channel are implementable.

We reveal the risk for the user's privacy by showing: (i) two possibilities how GovWare can be stealthily placed on a computer platform, (ii) a keystroke logger that logs user input, executed in a stealthy high-privileged isolated execution environment that does not affect the performance of the main CPU, and (iii) a very effective covert communication channel realized by means of the stealthy isolated execution environment to transmit data to the observing agency.

We implemented a PoC demo. We want to analyze that implememtation to figure out to what extent stealth GovWare is realizable and to derive countermeasures against such powerful technology. As an example we developed a stealth USB keyboard keystroke logger for the Linux operating system. We are convinced that our PoC implementation is much more stealthily than solutions presented so far. The popularity of the x86 platform makes the impact to users' privacy especially worrying.

2. POC TARGET PLATFORM: X86

On x86 platforms a CPU supports a privilege model called protection mode. Four privilege levels provided by that mode are called rings whereby only ring 0 (kernel space, privileged mode) and ring 3 (user space, unprivileged mode) are used in practice. Code executed in kernel space has more privileges than code executed in user space.

Considering a stealth GovWare PoC suggests itself to have a look at rootkits, since a rootkit is malicious code with certain stealth capabilities. The *rootkit evolution* shows that rootkits moved from user space to kernel space and beyond. With the implementation of hardware support for hypervisors the term ring -1 was introduced, since hypervisors deprivilege the kernel to be able to host several operation systems in parallel. Rootkits found their way into the exection environment of hypervisors, i.e., into ring -1 [10]. Modern x86 platforms offer further execution environments with even more privileges than supported by ring -1. The system management mode (SMM) – a special processor mode – intended to realize efficient energy usage as well as to control system hardware was called "ring -2" by security researchers [13]. Security researches demonstrated that rootkits can be executed in "ring -2".

In 2009, security researches discovered a new and very interesting execution environment for rootkits [12]. According to the properties of that environment the researchers coined the term "ring -3". Common virus scanners are unable to access that isolated execution environment, since they are executed in a less privileged ring.

The "ring -3" environment is actually intended to run Intel's Active Management Technology (iAMT). iAMT is a very powerful technology that supports hardware based security as well as remote management functionality [11]. To realize such features an isolated execution environment consisting of read only memory (ROM), static random access memory (SRAM), DMA hardware to access host memory (cf. [3, 12]) and an additional processor is embedded in the northbridge of x86 platforms. This execution environment uses an additional μ-controller which is called Management Engine (ME). The embedded processor of the ME is an ARCtangent-A4 (ARC4). The isolated execution environment is available regardless of the power state (standby, powered off, etc.). It only requires that the computer be plugged into a power source. During the platform turn-on procedure the iAMT firmware image is loaded into RAM. The firmware runs on the μ-controller's ARC4 processor and uses also some of the system's RAM to store runtime data. That runtime storage is provided by a certain memory area, that is not visible to the main CPU and the operating system. iAMT introduces out-of-band (OOB) communication, i.e., a special network traffic channel for iAMT.

We used that powerful "ring -3" environment to hide our PoC implementation from the actual host.

3. POC: STEALTH KEYSTROKE LOGGER GOVWARE

Infiltration. The first challenge we have to consider is how to stealthily infiltrate the target platform. We see two possibilities: (i) exploit a security vulnerability or (ii) cooperate with the hardware vendor.

Since our target computer is an Internet-connected x86 platform, we can try to infiltrate it using a vulnerability remotely exploitable, i.e., the dropper is an exploit. To find an exploitable vulnerability is not easy, but not impossible as shown in [12].

There is no obvious reason why governments should not cooperate with hard- or software vendors. Examples document that even big companies such as Microsoft (cf. [9]) are willing to cooperate with agencies:

"If it's vital to government, it's mission critical to Microsoft" [9]

Products like the Computer Online Forensic Evidence Extractor are only sold to such agencies. Otherwise, criminals could analyze and circumvent the software (cf. [7]).

Let us consider a government and Intel cooperate. Such a cooperation could be supposable simple: The government develops an ARC4 compliant iAMT firmware and asks Intel to deploy it on x86 platforms during the manufacturing process of its chipsets. Alternatively, Intel can offer that firmware as firmware update on their website. Hence, the GovWare could be provided country-specific fashion.

Data Collection. After infiltrating the target platform the GovWare can start to collect some data. Our goal is to log keystrokes from USB keyboards. To do so, we analzed the USB implementation of the Linux kernel. Linux uses a certain physical address pointing to a DMA based keyboard buffer. We implemented a heuristic to find and monitor that address using the stealth μ-controller based execution environment embedded in the northbridge. Our heuristic is based on similar techniques as described in [8].

The source code of the rootkit provided by [12] is not able to read from host memory. It can just write and therefore it is not suitable to build a keystroke logger. Thus, we set up a similar experiment as presented by [12] to figure out how to read from the host memory and implemented the accordant functionality to find and monitor the DMA address using the μ-controller. The functionality of our keystroke logger is shown in Figure 1. The output of our PoC demo can be seen in Figure 2.

Exfiltration. At the moment one important component of our proof-of-concept implementation is missing. Our keystroke logger must be able to transmit the logged keystrokes to a remote party. To do so, we have to figure out how to use the iAMT network capabilities to send network packets. Furthermore, we have to somehow realize a covert communication channel. Otherwise, the stealth GoveWare could be discovered very easily. Work is currently being done on the realization of the exfiltration described in this Section.

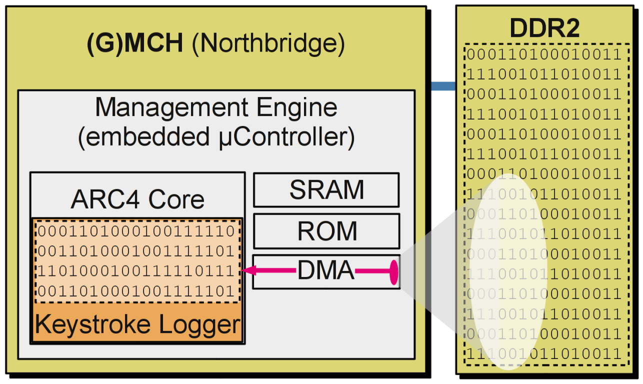
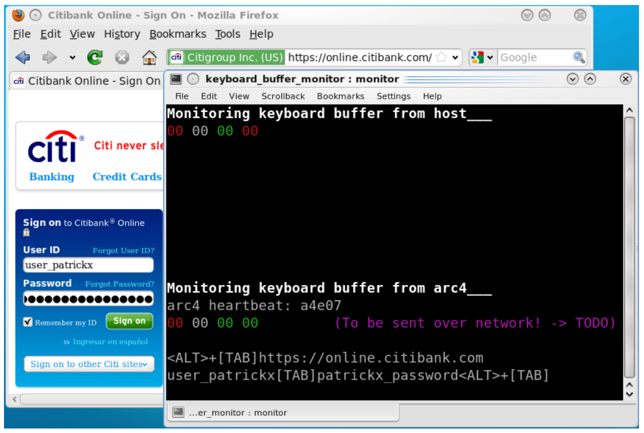


Figure 1: Keystroke Logger executed in Isolated Execution Environment


Citibank Online - Sign On - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Citigroup Inc. (US) https://online.citibank.com/
Google
Citibank Online - Sign On
citi
Citi never sle
Banking
Credit Cards
Sign on to Citibank® Online
User ID
Forgot User ID?
Password
Forgot Password?
Remember my ID
Sign on
» Ingresar en español
Sign on to other Citi sites
keyboard_buffer_monitor : monitor
File Edit View Scrollback Bookmarks Settings Help
...er_monitor : monitor

User ID: user_patrickx

```
Monitoring keyboard buffer from host___
00 00 00 00




Monitoring keyboard buffer from arc4___
arc4 heartbeat: a4e07
00 00 00 00          (To be sent over network! -> TODO)

<ALT>+[TAB]https://online.citibank.com
user_patrickx[TAB]patrickx_password<ALT>+[TAB]
```

Figure 2: Keystroke Logger Demo: Online Banking Sign On

4. CONCLUSION

Our first results show that modern x86 platforms have the potential to be turned into stealth government Trojans under certain circumstances (e.g., when the government cooperates with the hardware vendor or when using exploitable vulnerabilities). However, we have to finish our PoC implementation. A detailed evaluation of the finalized implemetation regarding stealthiness must be done. Dependent on the results of that evaluation, the next step is to develop countermeasures to eliminate the threat of a technology that enables governments to place their citizens under general suspicion.

5. ACKNOWLEDGMENTS

The authors would like to thank Benjamin Michéle, Collin Mulliner, Dmitry Nedospasov, Vassilis Ververis, Matthias Lange, and Steffen Liebergeld for their help and support in this work.

6. REFERENCES

[1] E. Abreu. FBI confirms "Magic Lantern" Project exists. http://www.si.umich.edu/~rfrost/courses/SI110/readings/Privacy/Magic_Lantern.pdf, Dec. 2001. Found at: School of Information – University of Michigan.

[2] R. Bendrath, G. Hornung, and A. Pfitzmann. Surveillance in Germany: Strategies and Counterstrategies. Userpage Ralf Bendrath: http://userpage.fu-berlin.de/~bendrath/Bendrath-Hornung-Pfitzmann_Surveillance-in-Germany_2009.pdf, June 2009.

[3] Y. Bulygin. Chipset based Approach to detect Virtualization Malware. TuCancUnix: http://www.tucancunix.net/ceh/bhusa/BHUSA08/speakers/Bulygin_Detection_of_Rootkits/bh-us-08-bulygin_Chip_Based_Approach_to_Detect_Rootkits.pdf, 2008.

[4] Electronic Privacy Information Center. Carnivore. http://epic.org/privacy/carnivore/default.html, Jan. 2005.

[5] Federal Bureau of Investigation. Declassified FBI CIPAV spyware documents. wired.com, Free Archive: http://www.freearchive.org/o/90a083d1cd08b540693a445854 3d8ac1ca4ca752ed67845986f3476921bf83ef/info, Apr. 2009.

[6] Guidance Software. EnCase Forensic. http://www.guidancesoftware.com/computer-forensics-ediscovery-software-digital-evidence.htm.

[7] K. J. Higgins. Microsoft Forensics Tool For Law Enforcement Leaked Online. DarkReading: http://www.darkreading.com/security/vulnerabilities/showArticle.jhtml?articleID=221600872&cid=ref-true, Nov. 2009.

[8] A. Lineberry. Malicious Code Injection via /dev/mem. Black Hat Europe: http://www.blackhat.com/presentations/bh-europe-09/Lineberry/BlackHat-Europe-2009-Lineberry-code-injection-via-dev-mem.pdf, Mar. 2009.

[9] Microsoft Corporation. Solutions Center for Government: Computer Online Forensic Evidence Extractor (COFEE). http://www.microsoft.com/industry/government/solutions/cofee/default.aspx.

[10] J. Rutkowska. Subverting Vista kernel for fun and profit. Black Hat USA: http://blackhat.com/presentations/bh-usa-06/BH-US-06-Rutkowska.pdf, Aug. 2006.

[11] Y. Saint-Hilaire. Extreme Programming with Intel vPro Technology: Pushing the Limits with Innovative Software. *Intel Technology Journal*, 12(4):335 – 342, Dec. 2008.

[12] A. Tereshkin and R. Wojtczuk. Introducing Ring -3 Rootkits. ITL: http://www.invisiblethingslab.com/itl/Resources.html, July 2009.

[13] R. Wojtczuk and J. Rutkowska. Attacking SMM Memory via Intel CPU Cache Poisoning. ITL: http://invisiblethingslab.com/itl/Resources.html, Mar. 2009.