THE PUBLISHING HOUSE
OF THE ROMANIAN ACADEMY

PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A,
Volume 14, Special Issue 2013, pp. 338–356

TOWARDS SECURE NETWORK COMMUNICATIONS WITH CLIENTS HAVING CRYPTOGRAPHICALLY ATTESTABLE INTEGRITY

Dan Horea LUTAS[*], Sandor LUKACS[*], Raul Vasile TOSA[*], Andrei Vlad LUTAS[*]

[*] BITDEFENDER SRL
1 Cuza Vodă Street, Cluj-Napoca, ROMANIA
Corresponding author: Sándor LUKÁCS, E-mail: slukacs@bitdefender.com

In a client-server communication, the server side must trust the client before releases confidential data or accepts commands received from the client. While industry best practices and considerable amount of research focus on secure credential authentication, we stress that this is at most a deceptive effort, providing only a false sense of security. We underline the sharp contradiction between state-of-the-art industry wide practices and the security that researchers campaign for, with special focus on online banking solutions. We present a brief review of a wide range of cyberattack techniques used today to perform large scale identity theft, financial fraud or espionage. We believe that current approaches, including inside-OS security solutions or relying only on credential authentication are outdated, and an industry wide shift is needed to provide trustable, integrity attested clients. Our solution is created around a type 1 bare-metal hypervisor, relying on hardware-enforced technologies to provide strong isolation between a secure operating environment on the clients and a possibly compromised OS. Blending an easily deployable solution with remote cryptographic identity attestation support, we believe that our proposal carries a significant value, both from security point of view and market applicability. In order to support a proper evaluation of the strength and weaknesses of the solution, we also present a comprehensive review of various remaining attack techniques and strategies.

Key words: trusted communication, hypervisor, VMM, secure client, trusted client, remote attestation, attestable integrity, Intel TXT, malware attack techniques, vulnerabilities.

1. INTRODUCTION

We live in a world where we, our privacy and ultimately the whole global economy are becoming more and more dependent on the Internet [109]. Stock trading and bank transactions are performed online, we pay our taxes online and do a great share of our purchases on the Internet, using credit cards. There are millions of people working remotely [58], accessing confidential business information, governmental or even military data from remote clients. Industrial and military espionage has grown to unprecedented levels [16, 114].

To underline the practicality and impact of our proposal, we shall mention a few issues implying client side malware infections or cyberattacks, comprising two key domains: financial and identity fraud, and economic and military espionage. Financial institutions and citizens using online banking are among the most frequent targets of advanced malware and cyberattacks [83, 45], with a recent United Nations report saying that more than 30% of total global cybercrime is computer related fraud and forgery [118]. There are numerous examples of online banking, identity theft and fraud related, money stealing malware, in many cases with tens of millions of US dollars stolen using a single advanced malware, operated by a few persons [130, 51, 81, 26, 110]. A 2012 study done by the U.K. National Fraud Authority [136] reveals that 9.4% of all adult U.K. citizens had been an identity fraud victim in the last year, only 44.7% of the victims being able to recover their losses. On average these victims lost £481 each. Many banking Trojans have been active in the wilds for 5-8 or more years, producing enormous damage [102, 124]. On the military side, the quite recent Stuxnet, Duqu [56, 8] and Flame [137] APTs were using various client side infection methods to be able to penetrate complex networks in stages. Another recent malware, Gauss [54], was characterized as "*a nation state sponsored banking Trojan which carries a warhead of unknown designation*".

Globally, we are dealing with two major class of cyber-enemies. First of all, there are cybercriminals focusing on stealing our identity, our credentials and ultimately our money. They are behind more than half of all illicit cyberactivities [83], with global cybercrime being well over $100 billion today [107, 41]. Secondly, we are dealing with advanced persistent threats – well organized criminal groups, foreign governments or military having both the intent and the capacity to continuously attack our networks, our systems or even our critical infrastructure.

Today numerous activities and industries are relying more than ever on client-server communications, where, in a broad sense, an attacker could target three key elements: the integrity of the server (*e.g.* directly penetrating a financial institution or a military server), the integrity of the communication channel (*e.g.* intercepting network packets in man-in-the-middle attacks) or the integrity of the client (*e.g.* setup keyloggers or backdoors onto an endpoint device). Analyzing this from a practical point of view, it is important to note, that one can have invulnerable servers, unbreakable cryptography for the communication channel (*e.g.* eavesdropping and brute force attack proof), but once the client side is compromised, so it is our privacy and the security of our critical data.

It is important to note that client systems are in many cases entrance vectors for malware or advanced attacks ultimately targeting critical server systems, as it was the case for Stuxnet family for ex. This is especially true for military grade / espionage, where the ultimate target is almost priceless: there is a long and impressive history of stealing confidential data from critical military projects [16, 88, 114, 79].

In this paper we are focusing on the security and the trustworthiness of the client side. There is one fundamental question the server should ask for in any over-network client-server communication: how does the server side decide if it can trust the client or not? When can confidential information be released to the client over the network?

1.1. Authentication *vs.* integrity attestation

The straightforward answer that comes into one's mind to the previous questions is like: "*the server will ask the client to send valid credentials, comprising user name, password or some kind of token value, and no critical information will be provided, unless the credentials are valid*". Obviously, this scenario assumes that credentials have not been stolen and the communication channel has not been compromised. An immense research has been carried out on the topic. There are countless papers and technologies focused on how to properly validate credentials, how to prevent credentials being stolen, *e.g.* by keyloggers or screen capturing malware. There are whole industries focused on the securing authentication, employing numerous methods, like online banking using hardware token or SMS based two-factor authentication, or employing the state-of-the-art Intel IPT [46, 144, 49] hardware based technologies, but still failing to provide adequate protection against advanced attackers. We will give several reasons for this in Section 2.1.

There is a fundamental and clear difference between *authentication of credentials* and *attestation of integrity*. This is one of the key ideas behind our proposal. In essence, *securing authentication* ensures that no one can steal our credentials or use them to legitimate as being us, *e.g.* to pretend to be us while logging in at an online banking account. On the other hand, *remote integrity attestation* can assure the server side that the client software has not been tampered with, *e.g.* an online banking application is exactly in the same state as the bank delivered it to the client at deployment time. Although the difference is significant between the two cases, to be able to consider the client trustworthy, the client software needs to satisfy another key criteria: it must be in complete control of the client hardware.

1.2. Strengthening the client. Raising the cost of an attack

Before we can talk about highly secure client-server communications, we shall reflect on two more essential aspects. First of all, considering a system where critical data is shared across network by two or more entities, the security of data at maximum equals the security of the weakest link from the whole chain. In most organizations, client systems represent the weakest point.

Secondly, in our vision, there is no perfect security. The strength of the security provided by a given solution is directly proportional to the time and resources needed to be invested in order to break into the protected system. We could quantify the value of a security system by how much does its deployment raise

the cost of the attack needed to successfully break the system. We estimate that our proposal has the value and strength to raise the cost of the attacks by at least a factor of magnitude.

We propose a solution designed around a client oriented, hardware virtualization based type 1 bare-metal hypervisor. As Vogl [116] and Zaharia [131] observe, hypervisors offer numerous advantages, including strong insolation, a significantly reduced codebase – providing a small attack surface and the possibility to analyze or enforce security from outside the operating system. We aim to use the hypervisor to take complete control over hardware, including KVM devices (keyboard, video, mouse) and to create a secure working environment inside a custom Linux trusted virtual machine. The hypervisor will employ all known Intel VT-x/EPT, VT-d and TXT technologies, in order to properly enforce security.

2. ATTACKING THE CLIENTS, STEALING THE DATA

2.1. Advanced malware and attack techniques

One essential property of our day malware is that it tries to remain stealthy and active as long as possible inside a system. The most frequently employed technique to achieve this is the usage of kernel rootkits [105, 69]. Once installed on the target system, widespread kernel-mode rootkits can easily be used to bypass traditional security solutions. According to some reports [100], from a few known rootkit samples 10 year ago, the number of unique rootkits today is likely over the 2.5 million mark. Advanced kernel malware routinely infects millions of computers around the world. In one example, in about a year, one kernel rootkit infected over 16 million PCs [87]. The next version of that malware successfully penetrated at least 46 of the Fortune 500 companies [123].

Another particularly dangerous advanced attacks are the so called DMA (Direct Memory Access) attacks, in which, using DMA controllers from various peripheral devices (*e.g.* network or audio cards), a malware can bypass traditional CPU privilege levels and overwrite any piece of a security software [104, 91, 2]. DMA can also be used to steal critical data (*e.g.* passwords or encryption keys) from different applications, bypassing traditional protection measures. More recently DMA based attacks are integrated directly into exploitation frameworks [11], making them easily deployable in wide scale attacks.

A great number of cyberattacks today relies on poorly written software that contains vulnerabilities, which are routinely exploited to execute malicious code [44, 14]. Landwehr hits the nail on the head [62] saying that "*today, most successful attacks simply exploit flaws that were inadvertently placed in the system during development and were not removed by industrial quality control mechanisms prior to distribution*". Although there are numerous attempts to protect software against those issues (NX bit / DEP, SMEP [50], ASLR [149], sandboxing, hook based protections being the most significant of them), until today none of them resisted for a long time against malware attacks [125, 57]. Even highly elaborated and deeply built-into-OS approaches, like Microsoft's PatchGuard in 64 bit editions of Windows, have been proved to be short of expectations in the fight with advanced malware [99, 36]. Although there are various reasons why those technologies failed to deliver adequate protection, we can observe one common and very significant weakness behind all of them: they are all running at the same privilege level as the malicious code they try to protect from (*e.g.* rootkit). Although today there are more than 50,000 publicly disclosed [14], well-known vulnerabilities, the most alarming fact is that 40-50% of the new ones disclosed each year remains without proper patching [44], such that their exploitation, even at kernel level [125, 5, 13], is both easily at hand and very lucrative for cyber-criminals. We must point out, that in case of advanced attacks, based on kernel exploits or kernel rootkits, there is no way to properly protect any kind of client side application from inside the OS; that is, client side network communication can be easily eavesdropped, encryption keys or credentials stored in memory can be easily stolen.

One particularly important class of malware are Banking Trojans [18], which are behind the so called Man-in-the-Browser (MiB) attacks [80, 70]. They are widely used to infect web browsers, intercepting network communication between the user (client) and bank (server), being able to fundamentally change every aspect of online banking transactions (*e.g.* changing the destination for a transfer or changing also the amount). They usually seek additional user credentials that attackers use to perform fraudulent fund transfers. It is important to note, that MiB attacks essentially mean the corruption of integrity of the client

side application, and thus, one way to protect against them is to be able to attest the integrity of the online banking browser. A critically important aspect of MiB attacks is that they regularly bypass two-way authentication, as all fraudulent activities are carried out after the user legitimately authenticated itself. In other words, credentials do not need to be stolen at all in order for a MiB attack to steal your money. As a prime example, the alarming advance of the ZEUS Trojan triggered the European Network and Information Security Agency (ENISA) to recommend [25] the banks nothing less than to consider all client PCs are being possibly infected. Already in 2011 it was estimated, that ZEUS infected 3.6 million PCs in the U.S. alone, *i.e.* over 1% of all systems [108]. Last year, researchers found a Banking Trojan that is capable to operate completely autonomously, being able to clean the banking account of a victim without requiring user interaction at all [120]. We stress that MiB attacks are not specific only to financial institutions or online banking / shopping. For example, the same techniques can be used for long term industrial espionage also.

The use of digitally signed malware is widespread [129], underlining the weakness of the certification mechanisms. We have examples when government certificates are stolen [33], with advanced attacks using validly signed malware [8, 66] or banking Trojans easily installing on victim systems because they have valid digital signatures [141]. We have today around a million of digitally signed malware, which can silently install kernel rootkits or other advanced malicious codes into the user's machine. This is a solid argument underlying the need to start using integrity attested clients.

The use of password and identity stealing malware is very widespread [103, 27, 72]. They usually employ keyboard, mouse and screen capture software. According to one significant report [68], 100% of data breaches investigated involved some kind of credential theft.

Two-way authentication, based on digital signatures, SMS or hardware tokens is widely used today, especially by the financial industry. However, they are hopelessly broken [92, 59] in case we are dealing with sufficiently skilled attackers. There are numerous reported cases in which criminals successfully intercept both authentication channels (*e.g.* €36 million was stolen recently using mobile malware that intercepted SMS messages from two-way authentication [52]). We must point out, that sophisticated MiB attacks don't even need to bypass two-way authentication or steal the user's credentials to be able to perform fraudulent activities under the client's identity. They can hijack legitimate user sessions then alter the traffic after the user identified itself. More than this, SMS-based two-factor authentication is broken since many years, not only by mobile malware, but by direct GSM/3G traffic eavesdropping also, using relatively cheap and simple equipment available today [28, 82].

We stress that this list is not exhaustive, but only exemplificative. However, at least two more aspects need to be considered here, in order to get a more complete view of the issues we are facing. First of all, advanced cybercriminals (or people behind APT attacks) are widely using techniques to avoid anti-malware detection [34, 17, 97]. Secondly, they use heavy automation and huge computing resources in areas like detection-avoiding malware generation / polymorphization [142], password cracking [85], exploit generation [3] or vulnerability scanning. The results can be devastating, as a recent report [70] described state-of-the-art malware automation used to siphon at least $78 million from bank accounts around the world. According to one research [68], on average, a typical computer network, protected by up-to-date security solutions have been penetrated by advanced attackers 243 days before the security breach is first detected. That is, advanced attackers have an enormous time window to perform their malicious business.

2.2. Significant examples from current solutions and approaches

To better underline the contrast between our proposal and existing solutions, we will present details on two important topics, related to the integrity of clients: online banking and exploit protection solutions.

Current state-of-the-art online banking solutions and approaches, from traditional security software vendors, include the usage of sandboxed evaluation [80] and/or security hardened browsers, with examples like Kaspersky SafeMoney [55] or Bitdefender SafePay [9]. While they do protect against some of the attacks used today, they have no chance against advanced kernel malware or targeted attacks. Many banks explicitly recommend users to have up to date anti-malware and anti-phishing solutions, which again, do indeed protect against some common attacks, as we pointed out in Section 2.1 can be bypassed by advanced attacks. There are several online banking software which include virtual keyboard and screen capture prevention methods (*e.g.* using a kernel driver, thus being completely vulnerable to an advanced kernel

attack running at the same privilege level). Both, the widely used hardware token or SMS based two-factor authentication, considered today an industry best practice and innovative technologies, like Intel IPT also [106, 46], are addressing only the first part of the problem: user authentication. They however do provide effectively no protection at all against advanced banking Trojans, man-in-the-browser attacks (which steal billions of dollars each year), not even mentioning advanced kernel exploit or DMA attacks. To sum it up, current state-of-the-art approaches represent less than half of a solution. We recommend the reader to confront this with best practice recommendations, flyers presenting online banking solutions and FAQ support lists from local banks. Our estimate is, that the usage of 128/256 bit SSL, two-factor authentication based on hardware token or SMS and anti-phishing recommendations will represent the bulk of the list. We still have to see a bank talking about ZEUS grade banking Trojans, offering truly protective solutions.

For protection against exploits a whole set of user-mode and kernel-mode technologies and products have been developed. Until today none of them stands against the attacks, underlying once again the need for a radically new approach: we recommend switching from software-only to hardware-enforced isolation and security solutions. XD (eXecution Disable) / NX bit and related Microsoft DEP technologies are widely used in an attempt to prohibit instruction-fetch operations from protected pages, to prevent code execution. This is however routinely bypasses by various attacks, like return oriented exploits executing VirtualProtect API to disable protection for a given area. ASLR (Address Space Layout Randomization) techniques ensures that after each reboot the various modules will be loaded to a different base-address, thus making exploitation harder (classical exploits tend to rely on hard-coded values – pointers to APIs, to TIB – Thread Information Block, PEB – Process Environment Block etc.). ASLR can be bypassed for example by creating an algorithm that explicitly searches for the needed data structures inside memory (*e.g.* by pattern matching) instead of relying on hard-coded values [50, 139]. Various products (*e.g.* Microsoft Office, Adobe Reader or Google Chrome) employ web or document specific sandboxes and low credentialed processes, by which they do manage to considerably reduce the number of successful exploits. However, many advanced attacks still succeed to bypass such sandboxing [125]. Classical HIPS/HIDS (Host Intrusion Prevention / Detection Systems) technologies, like Comodo, hook certain APIs in order to detect malicious calls to those functions. The bypass method is usually trivial, for ex. by restoring the original code then calling directly the kernel or by calling other undocumented and unhooked APIs to perform the same functionality. Some other exploit shellcode detection solutions are based pattern-matching or statistical analysis of possible shellcode varies. Such protection attempts are bypassed by malware using ASCII-encoded shellcodes, obfuscation or polymorphic/metamorphic shellcode [84]. There are also other attempts to prevent exploit execution or code injection, like per-process instruction-set randomization [57]. However, until now such attempts proved to be theoretical, being very difficult to apply it in actual CPUs.

2.3. Brief survey of other attack techniques. Assumptions

Although not directly linked to client side protection, to have a broader overview of the context clients are working in, we need to consider some other attacks and related assumptions.

Man-in-the-middle attacks (MiM) focus on intercepting the communication between the server and the client. There are two generic ways such attacks are carried out. Many attacks try to brute force the captured network packets or try to break the encryption method using advanced mathematics. We do not cover this topic and assume that 256 bit SSL/AES cryptography to be safe. However, another type of MiM attacks (including most of the known SSL attacks) rely on breaking third-party Certificate Authorities [32, 93, 43]. Here we observe a very disturbing statistics, with Comodo, VeriSign and many other CAs being repeatedly hacked, leading to a point where many consider the usage of third-party CAs to be broken. We think that in any client-server scenarios that permit it (*e.g.* online banking or military), the server side shall issue their own certificates and become their own CA. This is easily doable in scenarios where a physical contact before system setup takes place between client and server side operators. From security point of view, this means, that hacking the CA equals directly hacking the server side. This we think is a strong enough assurance.

We can't talk about securing the client side of a network communication without considering phishing, widely used today to steal credentials [55]. Over 150 million phishing emails are sent daily, and around 80,000 persons become victims of those attacks each day [140]. However, phishing is just one example of well-crafted social engineering, and, while several protective measures can be taken, no one can protect in a

bulletproof way against users that willingly and knowingly give over credentials and their identity. Depending on the exact scenario, many protective approaches can be taken. For example, a dedicated online banking software can use built-in hardcoded URLs (thus avoiding any e-mail based phishing attempt). Other measures include the good practice to avoid shortened URLs or the routing of network traffic through a security proxy server which seeks to filter out all malicious content.

3. HARDWARE VIRTUALIZATION AND SECURITY

3.1. Bounding technology and security concepts

When designing a software-only security solution, in general, or a secure communication software, like an online banking application, in particular, for a typical computing system, one shall note that the OS, the applications running above, the security solution and any potential malware or cyberattack basically share the same computing resources: the same memory, the same CPUs and the same privilege levels (user-mode and kernel-mode). The concepts of different privilege levels or isolation are not new at all, nor is their implementation at hardware level. A notorious example for not employing the principle of isolation could be the fact, that although Intel introduced in 1985 the 80836 microprocessors, supporting four different levels of protection (rings 0, 1, 2 and 3), offering the possibility to isolate the kernel from the drivers, the executive and the applications into different, hardware enforced privilege zones, no widely deployed operating system like Windows or Linux ever used this [50].

Intel released VT-x hardware virtualization support in 2005, followed in 2006 by the release of VT-d and TXT technologies and in 2008 of VT-x/EPT extension [50, 48, 47]. In contrast with this, until this day, mainstream operating systems and security solution are using few to none of the possibilities of those technologies to strengthen security. The reasons behind this are many. Among them was the desire to maintain compatibility with older hardware and software, the lack of know-how and expertise, an industry wide inertia. One notable exception is Linux, where certain flavours support TXT for attestation of the OS kernel, but not to attest security solutions or applications within. Although VT-x, VT-x/EPT and VT-d are primarily focused to provide virtualization capabilities, *e.g.* to be able to run multiple operating systems as guests on a single physical hardware, they also have tremendous potential in the field of security. On the other hand, Intel TXT was specifically designed for security purposes, to allow strong cryptographic load time attestation of system software [35].

We shall also mention other technology elements that are strongly related to providing secure client environments. Second and third generation Intel Core i5/i7 CPUs include [50] built-in AES encryption support (AES-NI). Third generation Intel Core CPUs include built-in hardware random number generators. TPM chips include support for SHA-1 hash based binary image measurements, RSA cryptography and sealed storage [112].

Intel VT-x provides basic CPU virtualization capabilities. VT-x introduces a completely new level (ring -1 or VMX Root Mode) of execution privilege, more privileged than ring 0 used by the operating system kernel. This ring -1 level can also be used by security software – or a secure client side application, if we implement it as a virtual machine monitor (VMM), also known as a hypervisor. This basically provides a security software with the capability to intercept and control the execution of the operating system kernel, with the OS and all applications being moved into a virtual machine (VM), also known as a guest. Intel VT-x/EPT provides the capability to completely virtualize the physical memory space of a guest virtual machine and thus, to be able to control at a page level for each guest the read/write/execute access rights associated with memory [50]. We must also consider the fact that, by the design of the CPUs, there can be only one piece of software running at ring -1 at any time, so, as long as a first software controls the CPU from hypervisor level, no other – possibly malicious – software can get into ring -1.

Intel VT-d provides the capability to virtualize the access to the system memory for extension boards, devices and controllers connected through the system bus. Numerous devices like network cards, storage controllers etc. have built-in DMA controllers that give them direct access to the physical memory of the system using, bypassing any control normally imposed by the CPU. Employing the capabilities offered by Intel VT-d, a security solution is able to provide hardware enforced protection against DMA attacks [48].

Intel TXT is a feature that, based on Intel VT-x/EPT, VT-d and a TPM (Trusted Platform Module) chip [113] provides the system with the capability to load and launch a VMM software in a measured well know state that is cryptographically attested to be tamper-free [35, 47]. The challenges around system software attestation are widely known [65], with several attempts being made to enhance the integrity or attestability of computing platforms and system software (*e.g.* UEFI Secure Boot [117] or Measure Boot for Windows 8 [78]). The need and desire to have some kind of attestations is obvious: one doesn't really want to just know that "*a client side application has been installed on a system*" but would rather like to be ensured at least that "*the client side application that has been installed on the system was started / is running without being tampered with*".

3.2. Hypervisor based security and integrity attestation state-of-the-art

The academia researched heavily both the topic of integrity attestation and that of the employment of hypervisors to enforce security.

Hypervisors provide an excellent way to perform security monitoring from below the OS, running at a different privilege level. Garfinkel *et al.* [31] present an IDS (Intrusion Detection System) built using a hypervisor, Livewire, capable to extract data from the guest, then to reconstruct semantics of those data. Srivastava [98] presents a hypervisor-based architecture that can identify malicious code inside infected systems at process-level granularity. The identification is based on network-level security software, linking each malicious network activity with the process behind. Wang *et al.* introduce HookSafe [121], a hypervisor focused on prevention of kernel rootkit infections by protecting in guest kernel code against hooks. Chen *et al.* present a different approach with Overshadow [15], protecting the integrity of application data against compromised OS. This is achieved by presenting applications with a normal view of its resources, but the OS with an encrypted view. Seshadri *et al.* propose SecVisor [94], a tiny hypervisor aimed to ensure code integrity for commodity OS kernels, protecting them against kernel malware and zero day exploits.

Lampson [61] proposes creating different, well isolated VMs, according to different roles: 'red' VMs for generic applications, possibly compromised by malicious code and 'green' VMs for a critical, trusted applications handling sensitive data. Garfinkel *et al.* propose a similar approach introducing Terra [30], having remotely attestable VMs for specific applications, like DRM media players. Our proposed approach follows the isolation principles presented by Lampson and Garfinkel, respectively.

McCauley introduces a novel way with HyperPass [75] to protect user passwords from being found by malicious software. With HyperPass, the passwords are not stored anymore inside the guest system at all, but instead, are stored inside the hypervisor. HyperPass has complete control over the physical network interface, effectively filtering all network traffic. TLS encryption is handled with a legitimate man-in-the-middle approach. The key idea is, that passwords are filled-in into forms (e.g. at an online banking login) only from inside the hypervisor. This way they achieve a solid credential protection.

Martignoni *et al.* [71] introduce Cloud Terminal, proposing a split approach for security: a thin terminal running on the clients ensuring only communication with a remote, in-cloud rendering engine at the server side. The server side comprises a massive VMM supporting one VM per each client to effectively render application content. Their solution has resemblance with our proposal, as they support on-the-fly loading below a running instance of the host OS and remote attestation.

McCune *et al.* [76] present a novel way to execute security-sensitive code in complete isolation, named Flicker. This solution provides fine-grained attestation of the executed code to a remote party. However they have many limitation, including a severe performance penalty and the requirement to custom compile any sensitive code. McCune later presented TrustVisor [77], featuring much better performance with a considerably bigger codebase. Not covering KVM device virtualization, their approach has limited applicability.

Lee-Thorp [65] provides a great overview of the challenges regarding trusted computing based on TPM chips. He argues that extending the TCG (Trusted Computing Group) architecture and reworking application designs are the most viable routes to making attestation viable in practice. Lyle *et al.* analyze the problems of remote service attestation [67], but from the client's point of view. They argue that critical services, such as healthcare services, electronic voting services, financial services, grid services, all need to be attestable: whenever a user is using one of these services, how can he know if the service is attested (not tampered by a malicious intervention)? Their approach however is vulnerable to MiB attacks. More than this, the integrity

of the client systems in not considered at all. Armknecht presents [1] a pure software-based attestation framework. While a software attestation may increase the security of a system, it is far from providing the same level of security as hardware-assisted integrity attestation does. Huang *et al.* [42] present a practical approach of doing remote-attestation, using hardware-based attestation mechanisms. A protocol for remote attestation is proposed, which has the capability of verifying the integrity of an application with regard to a remote party. The protocol may, however, be the subject of various attacks known in the art.

To be fair in our analysis, we acknowledge that there are also researchers raising concerns regarding hypervisor based security. Heiser *et al.* [39] argue that once the kernel of an operating system can and has been formally verified, new possibilities for security arise. They are against the usage of hypervisors advocating that a verified, and thus really trustworthy, OS kernel can provide better security than hardware-virtualisation based strong isolation. While we admit the exceptional value provided by formally verified code, we strongly believe that the software industry is by far not ready to adopt formal verifications on a wide scale, not affording the increased costs and having neither the required tools nor the expertise.

Heiser *et al.* [145] also argues that there is actually a convergence between hypervisor and microkernel architectures. There are numerous security benefits in hypervisors built using a microkernel-like design approach. Steinberg *et al.* presents Nova [146], introducing a novel way to build a 'microvisor' with a trusted computing base at least of an order of magnitude smaller than common hypervisors. They use microkernel architecture for the hypervisor to achieve lower attack surface and better security.

We close this short review with a significant report. Zaharia *et al.* performed an in-depth analysis of previous academic research about hypervisor supported security [131], concluding: "*the goal* […] *is not to prevent computers from being compromised; the complexity of modern OSes and the gullibility of most users makes compromise too easy to reliably prevent. The goal, instead, is to allow compromised machines to still perform some useful tasks without sacrificing user information* […] *Hypervisors are well equipped for this*". We consider this a truly precise characterization of the situation we are dealing with.

We underline again, the sharp contrast between what has been done in academic and military research versus what has been applied widely in real-life practice. The ideas of creating reliable and secure systems, or to build security based on virtualization are not new at all. Between many other examples [29, 20, 38, 96], there is one very significant one: Karger *et al.* [53] spent a decade long effort between 1981 and 1990 focused on the development of a production-quality VMM security kernel capable of receiving an A1 (highest) rating from U.S. National Computer Society Centre. In spite of this, our feeling is that pretty much the whole history of personal computing, dominated by MS-DOS, Windows (and later Linux or Mac OS) systems, repeatedly and systematically sacrificed security for the sake of performance or compatibility with older software, among other reasons. As a historical example, and before any Linux fan would yell towards us, we would like to gently remind that MINIX was here before Linux, with a much more security supporting microkernel architecture [111]. We are however afraid, that there are still many users, marketing people and corporate leaders, that would sacrifice security for 5-10% performance gain. We found to be very significant M.E. Lesk's depiction [62] "*Right now, most people are still wondering 'what is cybersecurity?' As time goes on, we predict people will start to demand it; then want it cheaper, and then eventually forget it*".

On the field of hardware virtualization, there are few client-oriented type 1 bare metal products available. The most significant of them is Citrix's Xen Client XT [95], a security hardened client virtualization product. It uses all available Intel virtualization oriented technologies (VT-x/EPT, VT-d, TXT) to provide strong security, integrating also several introspection and anti-malware technologies. However, it is not only close sourced, but also restricted as a market only for the U.S. government and federal agencies. We are not aware of Xen Client XT being integrated with remote attestation servers, but the possibility of doing so is certainly in the reach of the vendor. We must also stress, that its deployment is highly restricted because it needs pre-installation, and thus, it is not an option for millions of existing users out there.

Intel / McAffee Deep Defender [73] is the only known product from a traditional anti-virus vendor that incorporates VT-x/EPT technologies, thus provides enhanced protection against rootkits. However, it has several severe limitations and drawbacks, as it does not employ VT-d [74], thus being vulnerable to DMA attacks, and does not employ Intel TXT to assess the solution's integrity.

Qubes OS [86] is likely the most noticeable research project focused to deliver a heavily security oriented environment, based on the open source Xen hypervisor and involving all Intel security features (VT-x/EPT, VT-d and TXT). It is the best example for security-by-isolation, however, involving also some drawbacks. First of all, it does not employ any kind of anti-malware protection, trying just to encapsulate /

isolate any exploit or damage to a specified perimeter (working zone). Secondly, they have no support for remote attestation. We recognise that they could overcome their limitations and we think it would be great to see much more Qubes-like approaches and solutions in widely deployed, real-life applications.

Bromium Microvisor / vSentry [12] aims to provide enhanced security based on per-process virtualization. They do not offer trial versions publicly for detailed evaluation, but according to the available information they focus to secure major applications, including Microsoft Office or Adobe Reader by strong isolation and sandboxing. They do not employ Intel VT-d or Intel TXT and thus their approach is both vulnerable to DMA attacks and lacks integrity attestation support. Another critical aspect is that they run more likely a type 2 hypervisor, based on the environment provided by the host OS and not as a type 1 bare metal hypervisor, thus potentially exposing a much greater attack surface.

4. TOWARDS TRUSTED CLIENTS

4.1. Proposed general architecture

Bearing in mind the failure of today's software-only security solutions, it comes naturally for us to build our proposed design around a security tailored, client oriented, hardware virtualization based type 1 bare-metal hypervisor. We choose this solution because hypervisors numerous advantages, including strong insolation, a significantly reduced codebase – thus a small attack surface and the possibility to analyse or enforce security from outside the operating system. We took this approach considering also the red-green separation principles described by Lampson [61].

We use the hypervisor to take complete control over critical hardware devices and isolate the secure client application inside a trusted VM running heavily customized Linux. The hypervisor employs all known VT-x/EPT, VT-d and TXT security technologies. Our approach, as illustrated in Fig. 1, comprises two different operating environments, isolated by the hypervisor.

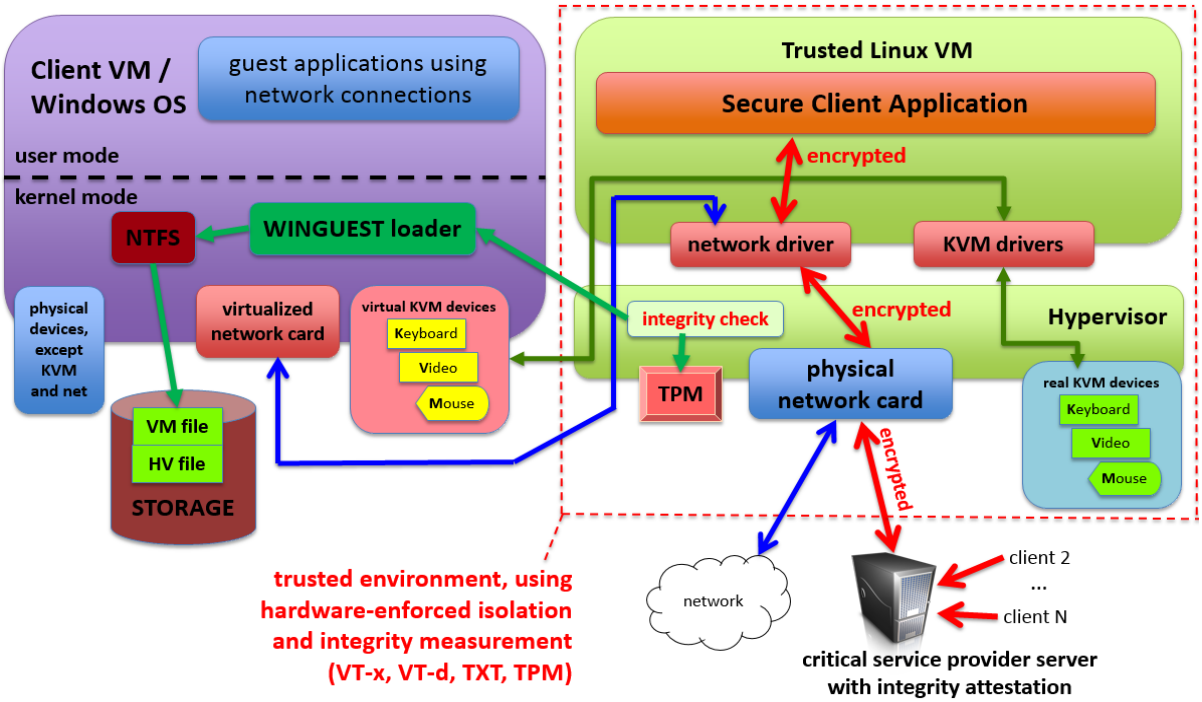

Fig. 1 – High level overview of key components of our proposal. The hypervisor completely takes over the physical KVM devices, thus ensuring any malware targeting our OS cannot interfere with the client application or its network communication. All critical communication is done through encrypted channels. The hypervisor, the Linux VM and the secure client application are hardware-isolated from OS, their integrity is enforced by Intel TXT and TPM. Both the hypervisor and the Linux VM are loaded from monolithic files (not requiring a separate partition), with their integrity being measured load-time, then validated.

The first environment is the existing client Windows operating system, inside which we install a kernel driver (winguest). Its main role is to load and start the trusted environment, for example in response to a hotkey combination hit by the user. It is important to note that we do not assume or require the client VM to be malware-free. For example, it can be infected with advanced malware designed to steal sensitive data, including kernel-mode rootkits. As we will outline in Section 4.4, such a malware can at most perform a DoS (Denial-of-Service) attack against us: it can prevent the loading of the trusted environment, but if the loading succeeds, we are assured that the client is in a cryptographically measured, tamper-free state. Once the trusted environment is loaded, the client VM and operating system will lose direct control of several physical devices, including not only processor, memory and chipset, but also network cards and KVM (keyboard, video and mouse) devices. In order to allow the client VM to fluently continue its execution, the hypervisor will expose to the OS virtualized versions of the devices described above, that will simulate the presence of the physical devices.

The second environment comprises a custom Linux VM running a secure client application, which can be specific to the exact deployment scenario (*e.g.* a security hardened online banking client). The integrity measurements based on Intel TXT and the TPM chip cover not only the hypervisor, but also the trusted VM, the included real device drivers and the secure client application also. The Intel TXT mechanisms ensure us, that if the environment is loaded, then (i) it can completely take over the control of critical CPU, chipset and memory devices, and (ii) the loading was performed in a tamper-free manner. The hypervisor will allow Linux to directly control the physical devices that have been hidden from Windows. This approach will allow the creation of communication channels between the virtualized devices exposed to Windows and the physical devices.

Both operating environments (VMs) are assured shared access to physical processor and to I/O devices in order to continue their execution fluently. The hiding of the physical devices from the client OS is achieved by taking over the complete communication interface between the drivers and the underlying devices, *i.e.* by rendering the device's slot as empty on the PCI bus, then triggering a hardware rescan in the client OS. In a similar way, the simulated hardware appears as newly inserted (hot plugged) devices, for which, in turn, Windows fluently loads new drivers and I/O stacks.

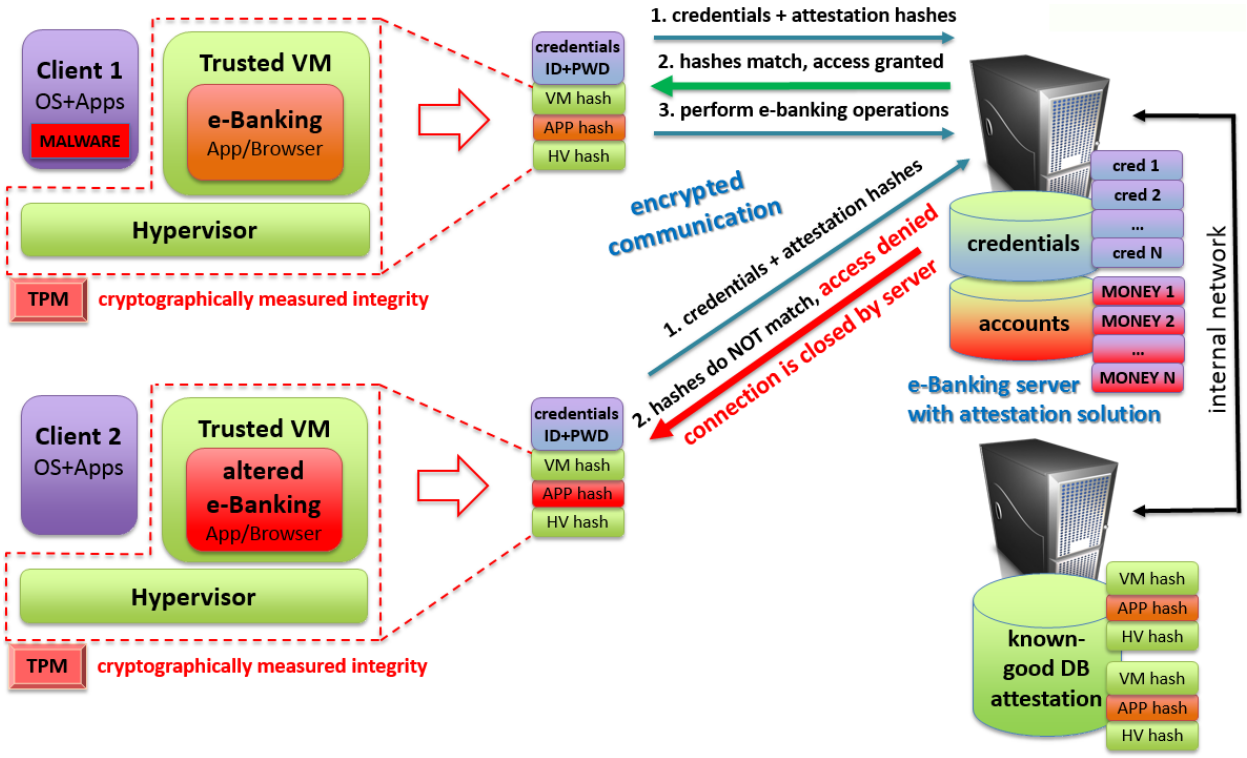

Fig. 2 – One possible application, comprising hypervisor based online banking application. The server side includes an integrity attestation server, holding all known-to-be-good integrity hashes in a database.

To better illustrate the proposed solution, we present one possible application in Fig. 2, depicting two online banking clients trying to connect to a service provider server from the bank. The bank also performs remote integrity attestation of the clients, requiring not only valid login credentials but known-to-be-good client integrity measurement hashes. The first client successfully connects even if the operating system or its applications are altered by a targeted attack or an advanced malware, because the e-Banking VM, the hypervisor and the e-Banking application have tamper-free, cryptographically attested integrity. In contrast with this, the connection of the second client, which is tampered with, is denied, because the e-Banking application's hash does not match the one in the known-good hash database. We expressly choose to present this with external clients having no direct connection at all to the attestation server: this way, in order for an external attack to reach the attestation server, it would require to go through existing vital banking infrastructure. In a way, we can consider the integrity of the known-to-be-good attestation database to be at least as valuable as money itself.

4.2. Particularities and strengths

The proposed approach promises hardware enforced strong protection against various types of attacks and malicious code affecting the client. Our solution allows us to perform secure business with a client, even if the endpoint is infected by advanced malware, including kernel rootkits, password stealers or Trojans.

An advantage of our approach is the inclusion of a full hypervisor and a complete Linux VM in two-three key files (*e.g.* the hypervisor binary, an in-guest driver and a VM image), allowing easy deployment on a client, without the need for end-users to reimage their systems. A related important aspect is, that we can install it under an already existing operating system, which can be already compromised by malware.

Our hypervisor does not need to be started before the operating system, but is started on demand, just like an application, thus greatly improving usability. An important strength is the ability to take over and virtualize runtime, on-the-fly from main operating system key hardware devices, *e.g.* network cards.

One more advantage of our proposal, although relevant for only very specific working scenarios, is the fact that we aim to completely take over the graphics card and assign it to the trusted VM. This way, secure clients can benefit from complete hardware accelerated graphics.

Finally, our proposal integrates well with existing anti-malware solutions, thus a finalized product based on it have the potential to reach tens of millions of client deployment easily.

4.3. Current progress, proof-of-concept results

Since 2011, anti-malware products from Bitdefender include a proprietary thin layer hypervisor as part of Active Virus Control technology. One critical property of it is the support of runtime on-the-fly loading from a single file stored inside the OS, hence it is very easily deployed with standard installation procedures. However, it does not incorporate Intel VT-d or Intel TXT technologies, nor complete interrupt virtualization.

Based on this initial work we have already done exploratory research to prove the feasibility of key ideas from our proposal. We have developed several proof-of-concept modules to demonstrate the feasibility of runtime on-the-fly VT-x/EPT and VT-d initialization, a minimal MLE (Measured Launch Environment) for Intel TXT integrity validations and for on-the-fly takeover of Ethernet network cards. We also demonstrated in-lab also the incorporation of encrypted network communication.

Our current proof-of-concept implementation has been demonstrated on four off-the-shelf, commonly available hardware systems, running Windows 7 and Windows 8 guest operating systems. We have tested several different Ethernet cards to validate multiple drivers from the Linux VM.

While it is too early to provide reliable numbers and statistics, we can mention several preliminary metrics. On-the-fly loading time of our hypervisor is around 3 seconds, including the time needed to bring up the Linux VM. The network virtualization has around 200 Mbit/s throughput without optimizations, deemed to be more than enough for the targeted client scenarios. The total research codebase is around 75 KLOC, excluding the customized Linux kernel.

We have not yet performed KVM device takeover, but considering our experience with Ethernet cards we estimate that to be technically feasible. The Intel TXT based integrity attestation proof-of-concept has been demonstrated on client level, without including any server side part.

4.4. Limitations. Possible advanced attack scenarios

Being focused on applied research, we must consider deployability and usability. Here we find a huge gap between various target audiences: although some of the key technological elements can be the same in home user, corporate and military scenarios, there are significant differences also. Home users would never accept to deploy on a massive scale a technology that needs reimaging from zero their system. They need easy and fast ways for deployment and very simple utilisation. At the other end of the spectrum, military needs maximum security, minimizing and mitigating every conceivable risks and attack vectors. We must be keep in mind those two extremes while analysing the strength and weaknesses of our proposal. We must stress, that most of the following limitations and possible attack scenarios do apply to existing commercial client security solutions and the vast majority of published research work, especially from the academia area. In spite of this, such aspects are usually left out or disconsidered, often generating a biased image.

We identified three important limitations of our proposal. First of all, in the current form we have no data persistence. Users working in the secure environment have no means of saving locally their files and any critical data. As we outline in Section 5.2, we aim to do further exploratory research on this topic. Secondly, the integrity attestation must include hashes covering the measurement of firmware code also, like BIOS / UEFI or SMM code for example. There is an inherent limitation regarding firmware updates, which can render the hypervisor and the trusted VM non-loadable due to hash changes. Thirdly, when switching to the trusted environment, either due to inherent differences between the physical and the emulated devices, or because of loss in the intermediate state of the physical devices and drivers, some existing applications might fail to continue without interruption until the trusted environment is closed, *e.g.* consider a game running on the client, dependent on the graphics adapter, or an application missing several network data packets.

There are numerous further attack techniques and scenarios to consider, while we are evaluating the security of a client. The following list is not exhaustive, but shall nevertheless be a solid starting point for any meticulous evaluation. We skip the ones already mentioned in Section 2.3 and focus only on client side attacks that can still, at least theoretically, affect clients protected by an implementation of our proposal.

Because we use runtime on-the-fly loading for the hypervisor instead of loading it before the operating systems starts, a kernel malware can attack our loader component (winguest). This way, they can perform a denial-of-service attack against us. We stress that such attacks can prevent the starting of our solution, but cannot steal our credentials or access confidential data. The solution is either loaded completely, with its integrity validated, or will fail to load at all.

Client systems can be physically stolen (actually, this is noticeably frequent even for government laptops). In such cases an attacker must break existing security measures (BIOS password, system password, hard disk password or encryption) before can try to launch the trusted environment. If he succeeds, he must brute force our credentials in order to retrieve any TPM sealed-in secrets. On the other hand, if he tries to connect to the server (we don't have local persistence for critical data), the server requires valid credentials, not only attested integrity, having the option to deny access after a few unsuccessful attempts.

Vulnerabilities in the OS kernel software network stack can lead to the compromise of the trusted VM, if an attacker can deliver an exploit, *e.g.* by a malformed network packet. This can be mitigated by isolating further the network drivers and the complete network stack into a third dedicated virtual machine. In this way, a successful attack doesn't end with a compromise of our credentials or critical data, but still allows the attacker to perform MiM or DoS scenarios. This approach is well illustrated by Qubes for example.

Although out of our control, especially in critical applications we shall not disconsider possible bugs and vulnerabilities in the CPU and / or built-in hardware virtualization technologies. Modern processors have numerous known small bugs and glitches. Some of them are fixed by microcode updates, and most of them carry no security impact at all. However, occasionally significant bugs are found that can allow an attacker to perform privilege escalations or to bypass critical security technologies. Wojtczuk *et al.* demonstrated [128] several software attacks against VT-d and against Intel TXT also [126]. There have been disclosed vulnerabilities that allow malware to crash the system [134] or vulnerabilities [138, 23] that allow a local authenticated attacker to perform operating system privilege escalation or a guest-to-host virtual machine escape.

Vulnerabilities in system firmware, like SMM or UEFI code can also lead to compromised clients. This has been demonstrated by Wojtczuk [127] and Duflot [21]. To mitigate such attacks one needs to virtualize also SMM mode, however today this is fully supported on AMD platforms only.

One of the key advantages of a hypervisor is the very low exposed attack surface. This however does not mean that a small attack surface (*e.g.* hypervisor ⇔ VM interface) is necessarily bulletproof. There have been several successful guest-to-host escape vulnerabilities demonstrated on commercial hypervisors [40]. Risks can be minimized by thoroughly reviewing and validating the interface. We also stress again, that this attack surface is significantly smaller than the attack surface exposed by conventional OS and applications.

Vulnerabilities in device firmware (*e.g.* network cards integrating vPRO and Intel AMT, but not only) have been underlined to be highly dangerous by Tereshkin [115] and Duflot [22], allowing remote, silent and practically undetectable compromise of systems. One mitigation of such attacks can be the isolation of the network cards inside a separate network dedicated virtual machine. Another mitigation can be the usage of non-vPRO/AMT network cards along with the proper refresh of the firmware from a trusted source.

Related to firmware based attack, one shall also evaluate the possibility of hardware backdoors, as exposed by Skorobogatov [101] and Brossard [10] among others. While unlikely to be a frequent issue in commercial applications, there are no known generic and efficient mitigation measures.

During ACPI S0⇔S3 (sleep) and S0⇔S4 (hibernate) transitions [135], by the very design of the Intel CPUs and of the TXT architecture, a hypervisor must give over the complete control of the CPU and of the hardware devices (at wakeup, execution will resume in 16 bit legacy real mode). We consider this to be not only a very nasty limitation, but the exposal of a significant attack surface. A secure hypervisor shall either deny completely such sleep state transitions, or completely unload itself and scramble the content of physical memory before entering sleep states.

Cross CPU core cache related side channel attacks have been demonstrated by numerous researchers, including Zhang [132] and Bangerter [6]. Such attacks can be used to either steal encryption keys or to directly access confidential data. Partial mitigations for such attacks is the storage of the encryption keys inside the registers of a CPU core or to completely reserve a CPU core for the trusted VM. Alternatively we can completely suspend the execution of the host OS while the trusted environment is running.

Cold boot attack can be used to retrieve not only encryption keys, but also to directly retrieve critical data from memory, as demonstrated by Halderman [37]. A partial mitigation for this can be the storage of encryption keys in CPU registers only, but decrypted data still can be accessed if system power is cut off before leaving the chance for the hypervisor to scramble the memory (we shall note also, that Intel TXT foresees surprise reboot scenarios using a special in-chipset bit, that requests the firmware to scramble the whole system memory if the last before-reset TXT session was not successfully closed).

At least theoretically, we shall consider a scenario when at install or load time we are running already inside a virtualized environment, *e.g.* a hypervisor level rootkit, supporting nested virtualization above. We consider this extremely hard to be done, as all technological elements, including VT-x/EPT, VT-d, TXT and the TPM chip must be properly virtualized, not only the basic VT-x extensions. Among other, Rutkowska [89] demonstrated hypervisor level rootkits, without however a complete technology emulation. Even the latest commercial Citrix and VMware products are still providing nested virtualization that is not feature complete. There are numerous hypervisor level rootkit detection techniques known in the art, as described by Lawson [64], Lauradoux [63] or Zovi [133].

Care must be taken with device virtualization to clear or scramble all device buffers and caches (*e.g.* network ring buffers, video RAM) while closing a trusted VM, before the client's OS regains control of the hardware devices. Dunn described cases when images and data where successfully recovered [24] after confidential browsing sessions ended.

To avoid disabling Intel TXT or performing TPM takeover from within the operating system (there are system management tools, like Dell Client Configuration Toolkit [19], that allow such operations to be carried out by administrators or a malicious script), systems must be password protected. It is yet to be answered if and how easily could an advanced malware crack such protection in a silent way. Nevertheless, we consider that the inclusion of management features that allow scripts to reconfigure BIOS / UEFI settings from within the OS is wrong by design, being the sacrifice of security for the sake of manageability. Beside this, being at hypervisor level we can deny any illicit BIOS update or reconfiguration.

An advanced Trojan running on the client's system might try to perform meticulous social engineering attacks in order to persuade the user to give away its credentials. Such malware can try to imitate the UI and the user experience of loading and running our hypervisor and the trusted VM. We shall note, that such attacks are limited to credential theft only, as they would be denied access to confidential data by the server

due to the lack of valid attestation hashes. A mitigation for such attacks can be the inclusion at install time of an encrypted, user specific personal image, sealed down using the platform's TPM. Such images can be used to visually confirm the user the authenticity of the exposed interface.

We must also consider the security impact of accessing from the trusted environment of documents or applications containing exploits or malicious code. A client incorporating third-party software or consuming third-party data (*e.g.* opening a malicious PDF or browsing of a website showing exploit-containing advertising images) might be easily compromised. The source of such data can be numerous, including other clients, the server itself, an USB stick (if allowed), the Internet (if open browsing is allowed) and so on. Such attempts can be part of advanced multi-stage attacks. The mitigation of such risks can be various, according to the specific usage of the client and the nature of the data. For some cases, strong hypervisor enforced per-process isolation can be used, as done by Bromium for example [12]. In online banking scenarios we can rely on a single preloaded known-to-be-safe browser and built-in certificates and URLs. For highly critical applications, no more than 'old typewriter style' information reviewing shall be allowed (*e.g.* plain text files in a very simple viewer; similarly, only very basic e-mail systems can be truly safe).

Hardware keyboard loggers, electromagnetic sniffers are much more efficient and easily available than average person would presume. They have been showed by Barisani [7], Vuagnoux [119], Kuhn [60] among many others, to be very effective in stealing not only credentials but critical data also, like secret e-mail messages as we type them into the system. The capturing and reconstruction of video output based on electromagnetic emissions of modern LCD is also practical, as demonstrated by Backes [4] for example. While such attacks and information thefts are easily carried out by a skilled person (or teams specialized on espionage), the mitigation of them is particularly difficult in most scenarios. Some initial mitigation steps can be the usage of so called TEMPEST-proof clients and the avoidance of direct power cable or cooper based network connections, *e.g.* using only interchangeable batteries and optical network cabling. Offering mitigation for some of the related issues, Grawrock argues [35] for the usage of trusted USB peripherals. They include cryptographic processors inside each device and require appropriate key setup to ensure trusted input channels between the peripherals and the drivers running inside the system. Electromagnetic attacks stand as a proof for our strong believe, already stated in Section 1.2, that the strength of a security solution can be measured only relative to the time and resources needed to be invested into an attempt to break it.

There are many other aspects and best-practices that needs to be considered while trying to ensure the security of client system. However, to detail them would exceed the scope of this paper. We also need to acknowledge and consider also that there is a valid criticisms regarding privacy issues and concerns related to TPM [143]. One fundamental issues here is, that clients must trust the TPM chip manufacturer, because the private keys are sealed in at manufacturing time [90]. It would be quite easy for a manufacturer, forced or not by authorities, to retain a list with all private keys and to give over them to third parties. At least for military applications, there is an open possibility to manufacture their own TPMs.

5. CONCLUSIONS

5.1. The value of our proposal. Possible applications. Contributions

We are confident that our proposal can provide much greater security in numerous usage scenarios, avoiding identity or critical business data theft, preventing leakage of credentials and ultimately saving time and money for home users, big enterprises and government consumers also. We strongly believe that our approach can be used on wide variety of desktops and laptops, widely available today. A key advantage of our proposal is the ease of deployment. Being simple to install and integrating well with existing Bitdefender anti-malware solutions, we have the potential to reach tens of millions of clients. What we propose is not unprecedented in term of technology, but our approach can make a paradigm shift across widely deployed security solutions, delivering instead of traditional software-only methods a new, hardware-enforced security solution for the masses. We believe this to be a significant step ahead.

While theoretically not bulletproof, our proposal significantly raises the cost of an attack along with greatly reducing the attack surface, in comparison with nowadays widely deployed and used, mass market approaches. It has the strength to protect clients against an impressive range of advanced malware and attack

techniques widely used by cybercriminals. On top of this, it has the potential to provide guarantee for service providers not only of the identity, but also about the integrity of clients connecting to them over network.

One notable contribution of our paper is the brief technical review of a wide range of real-life attack techniques impacting the security and integrity of client systems, paired with another review of numerous limitations and remaining possible attack methods after our proposal is fully implemented. We feel that a considerable share of published research work has limited real applicability, in part because of the lack of a proper evaluation of state-of-the-art security attacks and their implications. There is a considerable difference between real-life, market and ideal laboratory realities.

We estimate the first complete implementation of our proposal to be a highly secure online banking application, having huge potential impact, including significant direct and indirect cost savings both for clients and financial institutions. According to a 2011 analysis and forecast done by the French National Gendarmerie [122], in this decade we are going to see mass theft of banking, financial, or card information, all exceeding many times in impact and cost classic criminality / burglary. We are already on the verge of it. A 2012 Symantec study [109] estimates, that during 2012 alone 556 million adults worldwide experienced some form of cybercrime. Given this real-life context, our proposal has vast inherent value and a significant market potential.

The possible practical applications of our proposal are by far not limited to secure online banking. A hypervisor and Intel TXT based solution can be the cornerstone for working environments that provide secure e-mail or secure video conferencing for example. Such a setup can be used to create trusted working partitions in BYOD scenarios, or to permit employees to remotely access and control critical infrastructure in a very secure way. We strongly believe that our approach has significant applicability in numerous military and government areas also.

5.2. Future research. Next steps

In the months to come, we aim to focus research on creating a feature complete secure online banking solution prototype. This shall be the first complete and full scale validation of our proposal. To do this, we still need to validate at proof-of-concept level the technical possibility to fluently and safely take over from the OS, on-the-fly, all KVM devices. A related research area we would like to explore comprises the ability to do non-concurrent sharing of hardware resources between a common and a trusted partition, using fast VM switching, similarly with the methods proposed by J. Sun *et al.* [147] and K. Sun *et al.* [148].

We would like to extend our research efforts with the inclusion of some mechanisms that permit limited persistence for user data (*e.g.* emails, documents received by the client in trusted VM). One way to achieve this is to save all data in encrypted form to the untrusted VM, then store for all files a corresponding hash in a meta-file, which in turn is encrypted and sealed into the TPM.

We also plan to do exploratory research and feasibility evaluation for several of the remaining attack strategies presented in Section 4.5, like cold boot or firmware based attacks. It will require a great effort to collect and summarize in all those attack directions the most reliable information and know-how about the latest advancements. However, we feel this is essential in order to maintain an edge over our adversaries.

In the mid future we plan to extend our research efforts to include also cryptographic evaluation and hardening of the communication channel and the integrity attestation mechanisms. For this, we are already in talks with an academic cryptography research team.

ACKNOWLEDGEMENTS

We are grateful to Bogdan Dumitru, Rareş Ştefan and Adrian Coleşa for their support, encouragement and valuable advice.

REFERENCES

1. F. Armknect, A.R. Sadeghi, S. Schulz, C. WAchsmann, *Towards Provably Secure Software Attestation*, Cryptology ePrint Archive, http://eprint.iacr.org/, 2013.
2. D. Aumaitre, C. Delvin, *Subverting Windows 7 x64 Kernel with DMA attacks*, Hack in the Box Conference, 2010.
3. T. Avgerinos, S.K. Cha, B. Lim, T. Hao, D. Brumley, *AEG: Automatic Exploit Generation*, 18th Annual Network & Distributed System Security Symposium, 6 February 2011.

4. M. Backes, M. Durmuth, D. Unruh, *Compromising Reflections – or – How to Read LCD Monitors Around the Corner*, IEEE Symposium on Security and Privacy, pp. 158-169, 18 May 2008.
5. G. Bakas, *Windows Kernel Vulnerability Research and Exploitation*, RUXCON Conference talk, 19 November 2011, http://2011.ruxcon.org.au/2011-talks/windows-kernel-vulnerability-research-and-exploitation/.
6. E. Bangerter, D. Gullasch, S. Krenn, *Cache Games – Bringing Access-Based Cache Attacks on AES to Practice*, Proceedings of the 2011 IEEE Symposium on Security and Privacy, pp.490-505, 2011.
7. A. Barisani, D. Bianco, *Sniffing Keystrokes with Lasers/Voltmeters*, Black Hat USA, 25 July 2009.
8. B. Bencsáth, G. Pék, L. Buttyán, M. Félegyházi, *Duqu: A Stuxnet-like malware found in the wild*, CrySyS Lab Technical Report, 14 October 2011, http://www.crysys.hu/publications/files/bencsathPBF11duqu.pdf.
9. Bitdefender, *Bitdefender Internet Security 2013*, http://www.bitdefender.com/solutions/internet-security.html.
10. J. Bossard, *Hardware backdooring is practical*, Black Hat Briefings and Defcon Conferences, 2012.
11. R. Breauk, A. Spruyt, *Integrating DMA attacks in exploitation frameworks*, 7 February 2012, http://staff.science.uva.nl/~delaat/rp/2011-2012/p14/report.pdf
12. Bromium, *Micro-virtualization vs. Software Sandboxing - A comparison*, Bromium whitepaper, 2013, http://www.bromium.com/sites/default/files/Bromium-Whitepaper-Micro-virtualization-vs-sandboxing.pdf
13. C. Cerrudo, Easy local Windows Kernel exploitation, Black Hat USA, 21 July 2012.
14. U.S. CERT – National Vulnerability Database, http://nvd.nist.gov/
15. X. Chen, T. Garfinkel, E.C. Lewis *et al.*, *Overshadow: A Virtualization-Based Approach to Retrofitting Protection in Commodity Operating Systems*, Proceedings of the 13th international conference on architectural support for programming languages and operating systems, pp. 2-13, 2008.
16. CSIS, *Significant Cyber Incidents Since 2006*, Center for Strategic and International Studies report, February 2013, http://csis.org/files/publication/130318_Significant_Cyber_Incidents_Since_2006.pdf
17. D. Dagon, P. Vixie, *AV Evasion Through Malicious Generative Programs*, https://malfease.oarci.net/help/av_evasion.pdf
18. O. Delgado, A. Fuster-Sabater, J.M. Sierra, *Analysis of new threats to online banking authentication schemes*, Actas de la X RECSI, Salamanca, 2008.
19. DELL, *Dell Client Configuration Toolkit – CCTK*, 2013, http://en.community.dell.com/techcenter/systems-management/w/wiki/1952.dell-client-configuration-toolkit-cctk.aspx
20. J.E. Dobson, B. Randell, *Building reliable secure computing systems out of unreliable insecure components*, Proceedings of the Conference on Security and Privacy, Oakland, USA, pp. 187-193, IEEE, 1986.
21. L. Duflot, O. Levillain, B. Morin, O. Grumelard, *Getting into the SMRAM: SMM Reloaded*, CanSecWest conference presentation, 2009.
22. L. Duflot, Y.A. Perez, B. Morin, *What if you can't trust your network card?*, Proceedings of the 14th international conference on Recent Advances in Intrusion Detection, pp. 378-397, 2011.
23. G. Dunlap, *The Intel SYSRET privilege escalation*, blog, 13 June 2012, http://blog.xen.org/index.php/2012/06/13/the-intel-sysret-privilege-escalation/.
24. A.M. Dunn, M.Z. Lee *et al.*, *Eternal Sunshine of the Spotless Machine: Protecting Privacy with Ephemeral Channels*, Proceedings of the 10th USENIX conference on Operating Systems Design and Implementation, pp. 61-75, 2012.
25. ENISA, *Flash note: EU cyber security agency ENISA; "High Roller" online bank robberies reveal security gaps*, ENISA press release, http://www.enisa.europa.eu/, 5 July 2012.
26. FBI, *Three Alleged International Cyber Criminals Responsible for Creating and Distributing Virus That Infected Over One Million Computers and Caused Tens of Millions of Dollars in Losses Charged in Manhattan Federal Court*, FBI New York Field Office press release, 23 January 2013.
27. D. Florencio, C. Herley, *Is Everything We Know About Password-Stealing Wrong?*, IEEE Security & Privacy, IEEE, Vol. **10**, Issue 6, pp. 63-69, November 2012.
28. E. Gadaix, *GSM and 3G Security*, Black Hat ASIA, April 2001.
29. M. Gasser, *Building a Secure Computing System*, Macmillan of Canada, 1988.
30. T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, D. Boneh, *Terra: A Virtual Machine-Based Platform for Trusted Computing*, Proceedings of the 19th ACM symposium on Operating systems principles, pp. 193-206, December 2003.
31. T. Garfinkel, M. Rosenblum, *Virtual Machine Introspection Based Architecture for Intrusion Detection*, Proceeding of Network and Distributed Systems Security Symposium, 2003.
32. D. Goddin, *How is SSL hopelessly broken? Let us count the ways*, The Register news article, 11 April 2011.
33. D. Goddin, *Certificate stolen from Malaysian Gov. used to sign malware*, The Register news article, 14 November 2011.
34. J. Granneman, *Antivirus evasion techniques show ease in avoiding antivirus detection*, 2013, http://searchsecurity.techtarget.com/feature/Antivirus-evasion-techniques-show-ease-in-avoiding-antivirus-detection.
35. D. Grawrock, *Dynamics of a Trusted Platform*, Intel Press, 1st edition, 2009.
36. D. Gupta, X. Li, *Defeating PatchGuard – Bypassing Kernel Security Patch Protection in Microsoft Windows*, joint McAfee – Intel whitepaper, 2011, http://www.mcafee.com/us/resources/reports/rp-defeating-patchguard.pdf
37. J.A. Halderman, S.D. Schoen, N. Heninger *et al.*, *Lest We Remember: Cold Boot Attacks on Encryption Keys*, Communications of the ACM – Security in the Browser, Magazine, Vol. **52**, Issue 5, pp. 91-98, May 2009.
38. N. Hardy, *KeyKOS architecture*, ACM SIGOPS Operating Systems Review, Vol. **19**, Issue 4, pp. 8-25, October 1985.
39. G. Heiser, L. Ryzhyk, M. Von Tessin, A. Budzynowski, *What If You Could Actually Trust Your Kernel?*, Proceedings of the 13th USENIX conference on Hot topics in operating systems, pp. 27-32, 2011.
40. K.J. Higgins, *Hacking Tool Lets A VM Break Out And Attack Its Host*, http://www.darkreading.com/ article, 4 June 2009.
41. HP, *Cybercrime Costs Rise Nearly 40 Percent, Attack Frequency Doubles*, HP Research press release, 8 October 2012.

42. X. Huang, Y. Peng, *An Effective Approach for Remote Attestation in Trusted Computing* , Proceedings of the 2009 International Symposium on Web Information Systems and Applications, pp. 80-83, May 2009.
43. R. Hurst, *Is SSL Broken?*, blog, 3 February 2013, https://www.globalsign.com/blog/is-ssl-broken.html
44. IBM, *Rising Attacks Focus on Browsers and Social Media Networks*, IBM X-Force 2012 Mid-Year Trend and Risk Report, September 2012.
45. IDC, *IDC Financial Insights Consumer Payments Survey 2012*, July 2012, http://www.idc-fi.com/getdoc.jsp?containerId=prUS23585112
46. INTEL, *Intel Identity Protection Technology (Intel IPT)*, http://www.intel.com/content/www/us/en/architecture-and-technology/identity-protection/identity-protection-technology-general.html
47. INTEL, *Trusted Compute Pools with Intel Trusted Execution Technology (Intel TXT)*, http://www.intel.com/content/www/us/en/architecture-and-technology/trusted-execution-technology/malware-reduction-general-technology.html
48. INTEL, *Intel Virtualization Technology for Directed I/O, Architecture Specification*, http://www.intel.com/content/www/us/en/intelligent-systems/intel-technology/vt-directed-io-spec.html
49. INTEL, *Crimeware Protection: 3rd Generation Intel Core vPro Processors*, Intel Whitepaper, 2012, http://www.intel.com/content/dam/www/public/us/en/documents/white-papers/3rd-gen-core-vpro-security-paper.pdf
50. INTEL, *Intel 64 and IA-32 Architectures Software Developer Manuals*, http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html
51. C. Isidore, *8 charged in $45 million cybertheft bank heist*, CNN Money article, 10 May 2013.
52. E. Kalige, D. Burkey, *A Case Study of Eurograbber: How 36 Million Euros was Stolen via Malware*, joint Verasafe –CheckPoint whitepaper, 2012, http://www.checkpoint.com/products/downloads/whitepapers/Eurograbber_White_Paper.pdf
53. P.A. Karger, M.E. Zurko, D.W. Bonin, A.H. Mason, C.E. Kahn, *A Retrospective on the VAX VMM Security Kernel*, IEEE Transactions on Software Engineering, Vol. **17**, No. 11, pp. 1147-1165, November 1991.
54. KASPERSKY, *Gauss: Nation-state cyber-surveillance meets banking Trojan*, Kaspersky Lab research blog, 9 August 2012, http://www.securelist.com/en/blog/208193767/
55. KASPERSKY, *Protecting your bank account using Safe Money technology*, Kaspersky Whitepaper, 2012. http://www.kaspersky.com/downloads/pdf/kaspersky_lab_whitepaper_safe_money_eng_final.pdf
56. KASPERSKY, *Duqu: Steal Everything*, Kaspersky Lab research, 27 March 2012, http://www.kaspersky.com/about/press/major_malware_outbreaks/duqu
57. G.S. KC, A.D. Keromytis, V. Prevelakis, *Countering Code-Injection Attacks With Instruction-Set Randomization*, Proceedings of the 10th ACM conference on Computer and communications security, pp. 272-280, 2003.
58. Kensigton, *15 remarkable remote working statistics,* blog, 7 June 2012, http://clicksafe.kensington.com/laptop-security-blog/bid/83929/15-remarkable-remote-working-statistics#axzz2VDhGJl49
59. B. Krebs, *Attackers Hit Weak Spots in 2-Factor Authentication*, security blog, 5 June 2012, http://krebsonsecurity.com/2012/06/attackers-target-weak-spots-in-2-factor-authentication/
60. M.G. Kuhn, R.J. Anderson, *Soft Tempest: Hidden Data Transmission Using Electromagnetic Emanations*, Proceedings of 2nd Workshop on Information Hiding, pp. 124-142, 1998.
61. B. Lampson, *Accountability and Freedom*, Microsoft Research, 26 September 2005, http://research.microsoft.com/en-us/um/people/blampson/slides/AccountabilityAndFreedom.pdf
62. C. Landwehr, D. Boneh, John C. Mitchell, S. M. Bellovin, S. Landau, M. E. Lesk, *Privacy and Cybersecurity: The Next 100 Years*, Proceedings of the IEEE, Vol. **100**, pp. 1659-1673, 13 May 2012.
63. C. Lauradoux, *Detecting Virtual Rootkits with Cover Channels*, Annual meeting of the European Institute for Computer Antivirus Research – EICAR, April 2008.
64. N. Lawson, *Don't Tell Joanna, The Virtualization Rootkit is Dead*, Black Hat USA, 28 July 2007.
65. A. Lee-Thorp, *Attestation in Trusted Computing: Challenges and Potential Solutions*, Technical Report RHUL-MA-2010-09, March 2010, https://www.ma.rhul.ac.uk/static/techrep/2010/RHUL-MA-2010-09.pdf
66. M. Lenon, *Microsoft Certificate Was Used to Sign 'Flame' Malware,* news article, 4 June 2012, http://www.securityweek.com/microsoft-unauthorized-certificate-was-used-sign-flame-malware
67. J. Lyle, *Trustworthy Services Through Attestation,* PhD Thesis, Keble College, University of Oxford, 2010.
68. MANDIANT, *APT1 – Exposing One of China's Cyber Espionage Units*, Mandiant Intelligence Center Report, 2013, http://intelreport.mandiant.com/Mandiant_APT1_Report.pdf
69. D. Marcus, T. Sawicki, *The New Reality of Stealth Crimeware*, joint McAfee – Intel whitepaper, 2011, http://www.mcafee.com/us/resources/white-papers/wp-reality-of-stealth-crimeware.pdf
70. D. Marcus, R. Sherstobitoff, *Dissecting Operation High Roller*, joint McAfee – Guardian Analytics whitepaper, 2012, http://www.mcafee.com/us/resources/reports/rp-operation-high-roller.pdf
71. L. Martignoni, P. Poosankam, M. Zaharia *et al.*, *Cloud Terminal: Secure Access to Sensitive Applications from Untrusted Systems*, Proceedings of the 2012 USENIX conference on Annual Technical Conference, pp. 14-14, 2012.
72. MCAFEE, *McAfee Threats Report: Second Quarter 2012*,http://www.mcafee.com/us/resources/reports/rp-quarterly-threat-q2-2012.pdf
73. MCAFEE, *McAfee Deep Defender*, product datasheet, 2012,http://www.mcafee.com/us/resources/data-sheets/ds-deep-defender.pdf
74. MCAFEE, *McAfee Deep Defender Technical Evaluation and Best Practices Guide*, https://kc.mcafee.com/resources/sites/MCAFEE/content/live/PRODUCT_DOCUMENTATION/23000/PD23874/en_US/Deep_Defender_Best_Practices_Guide_Aug_2012.pdf
75. J. McCauley, R. Mittal, *HyperPass: Hypervisor Based Password Security*, project report, University of California, Berkeley, 2012.

76. J. McCune, B. Parno, A. Perring, M. Reiter, H. Isozaki, *Flicker: an execution infrastructure for TCB minimization,* Proceedings of the 3rd ACM SIGOPS/EuroSys Conference on Computer Systems, pp. 315-328, 2008.
77. J. McCune, Y. Li, N. Qu, Z. Zhou, A. Datta, V. Gligor, A. Perrig, *TrustVisor: Efficient TCB Reduction and Attestation*, Proceedings of IEEE Symposium on Security and Privacy, pp. 143-158, 2010.
78. MICROSOFT, *Measured Boot*, MSDN technical documentation, http://msdn.microsoft.com/en-us/library/windows/desktop/hh848050(v=vs.85).aspx
79. E. Nakashima, *Confidential report lists U.S. weapons system designs compromised by Chinese cyberspies,* The Washington Post article, 28 May 2013.
80. Norman, *Security you can bank on*, Norman whitepaper, April 2013, download01.norman.no/whitepapers/shark/Financial-Services-White-Paper-security-You-Can-Bank-On-v1.0-Final.pdf
81. J. O'Dell, *How Much Does Identity Theft Cost?*, infographic, 29 January 2011, http://mashable.com/2011/01/28/identity-theft-infographic/
82. M. Paik, *Stragglers of the Herd Get Eaten: Security Concerns for GSM Mobile Banking Applications*, HotMobile'10 Proceedings of the 11th Workshop on Mobile Computing, Systems and Applications, pp. 54-59, 2010.
83. P. Passeri, *Comprehensive cyberattack reports / 2012 Cyber Attacks Statistics*, http://hackmageddon.com/, http://hackmageddon.com/2012-cyber-attacks-timeline-master-index/
84. M. Polychronakis, K.A. Anagnostakis, E.P. Markatos, *Comprehensive Shellcode Detection using Runtime Heuristics*, Proceedings of the 26th Annual Computer Security Applications Conference, pp. 287-296, 2010.
85. L.Y. Qing, *Cloud a haven for cybercriminals*, ZDNET article, 7 February, 2011.
86. Qubes, *Qubes OS Project homepage*, http://qubes-os.org/Home.html
87. A. Rassokhin, D. Oleksyuk, *TDSS botnet: full disclosure*, Esage Lab security analysis, 13 March 2012, http://nobunkum.ru/analytics/en-tdss-botnet
88. P. Rosenzweig, Significant Cyber Attacks on Federal Systems – 2004-present, blog, 7 May 2012, http://www.lawfareblog.com/2012/05/significant-cyber-attacks-on-federal-systems-2004-present/
89. J. Rutkowska, A. Tereshkin, *Bluepilling the Xen Hypervisor*, Black Hat USA, 7 August 2008.
90. J. Rutkowska, *Trusted Execution In Untrusted Cloud*, blog, 13 December 2011, http://theinvisiblethings.blogspot.ro/2011_12_01_archive.html
91. F. L. Sang, V. Nicomette, Y. Deswarte, *I/O Attacks in Intel-PC Architectures and Countermeasures*, 2011 SysSec Workshop, pp. 19-26, 6 July 2011.
92. B. Schneier, *Hacking Two-Factor Authentication,* blog, September 2009, http://www.schneier.com/blog/archives/2009/09/hacking_two-fac.html
93. B. Schneier, *VeriSign Hacked, Successfully and Repeatedly, in 2010*, blog, February 2012, http://www.schneier.com/blog/archives/2012/02/verisign_hacked.html
94. A. Seshadri, M. Luk, N. Qu, A. Perrig*, SecVisor: A Tiny Hypervisor to Provide Lifetime Kernel Code Integrity for Commodity OSes,* ACM SIG OPS Operating Systems Review, Vol. **41**, Issue 6, pp. 335–350, December 2007.
95. N. Shah, *Meet High-Security Demands with XenClient XT 3.1*, Citrix blog, 25 April 2013, http://blogs.citrix.com/2013/04/25/meet-high-security-demands-with-xenclient-xt-3-1-now-with-a-free-trial/
96. J.S. Shapiro, N. Hardy, *EROS: A Principle-Driven Operating System from the Ground Up*, IEEE Software, pp. 26-33, January/February 2002.
97. H. Shinotsuka, *Malware Authors Using New Techniques to Evade Automated Threat Analysis Systems*, Symantec research blog, 28 October 2012, http://www.symantec.com/connect/blogs/malware-authors-using-new-techniques-evade-automated-threat-analysis-systems
98. A. Shirastava, *Robust and secure monitoring and attribution of malicious behaviors*, PhD Thesis, Georgia Institute of Technology, August 2011.
99. Skape, Skywing, *Bypassing PatchGuard on Windows x64*, security report, 2006, http://uninformed.org/?v=3&a=3&t=sumry
100. J. Skinner, E. Metcalf, *Preventing Stealthy Threats: A Proactive approach from Intel and McAfee*, Intel IT Talk to an Expert Series presentation, 2011, http://www.intel.com/content/dam/www/public/us/en/documents/best-practices/talk-to-an-expert-stealthy-threats-presentation.pdf
101. S. Skorobogatov, C. Woods, *Breakthrough silicon scanning discovers backdoor in military chip*, Cryptographic Hardware and Embedded Systems Workshop (CHES), 9 September 2012.
102. Sophos, *Ukrainian and Russian police arrest banking Trojan masterminds*, news report, 9 April 2013, http://nakedsecurity.sophos.com/2013/04/09/ukrainian-and-russian-police-arrest-banking-trojan-masterminds/
103. M. Ståhlberg, *The Trojan Money Spinner*, F-SECURE, Virus Bulletin Conference, 2007.
104. P. Stewin, I. Bystrov, *Understanding DMA Malware*, DIMVA2012 Proceedings of the 9th Conference on Detection of Intrusions and Malware & Vulnerability Assessment, 26 July 2012.
105. Symantec, *Windows Rootkit Overview*, Symantec Security Response whitepaper, 2005, http://www.symantec.com/avcenter/reference/windows.rootkit.overview.pdf
106. SYMANTEC, *Symantec Introduces Game Changer for Strong Authentication Credentials*, press release, 9 February 2011.
107. SYMANTEC, *Norton Study Calculates Cost of Global Cybercrime: $114 Billion Annually*, press release, 7 September 2011.
108. SYMANTEC, *Banking Trojans*, Symantec white paper, August 2011.
109. SYMANTEC, *Consumer Cybercrime Estimated at $110 Billion Annually*, Symantec press release, 5 September 2012.
110. SYMANTEC, *Symantec Internet Security Threat Report*, Vol. **18**, April 2013.
111. A.S. Tanenbaum, A.S. Woodhull, *Operating Systems: Design and Implementation*, Prentice Hall, 1987.
112. TCG, *TPM Main Specification*, Version 1.2, 2011, http://www.trustedcomputinggroup.org/resources/tpm_main_specification

113. TCG, *Trusted Platform Module Authentication*, technical specifications, http://www.trustedcomputinggroup.org/solutions/authentication
114. R. Tehan, *Cybersecurity: Authoritative Reports and Resources*, U.S. Congressional Research Service report, 24 May 2013.
115. A. Tereshkin, R. Wojtczuk, *Introducing Ring -3 Rootkits*, Black Hat USA, 29 July 2009, 2009.
116. S. VOGL, *Secure Hypervisors*, faculty research report, 2009, http://www.sec.in.tum.de/assets/lehre/ss09/seminar_virtualisierung/Secure_Hypervisors_S-Vogl.pdf
117. UEFI, *UEFI 2.3.1 Specification*, 2012, http://www.uefi.org/specs/
118. UNODC, *Comprehensive Study on Cybercrime*, United Nations Office on Drugs and Crime report, February 2013, http://www.unodc.org/documents/organized-crime/UNODC_CCPCJ_EG.4_2013/CYBERCRIME_STUDY_210213.pdf
119. M. Vuganox, S. Pasini, *Compromising Electromagnetic Emanations of Wired and Wireless Keyboards*, Proceedings of the 18th conference on USENIX security symposium, pp. 1-16, 2009.
120. P. Wagenseil, *Banking Trojan cleans out your account — you won't even see it*, NBC news article, 20 June 2012, www.nbcnews.com/id/47896468/ns/technology_and_science-security/t/banking-trojan-cleans-out-your-account-you-wont-even-see-it/
121. Z. Wang, X. Jiang, W. Cui, P. Ning, *Countering Kernel Rootkits with Lightweight Hook Protection*, Proceedings of the 16th ACM conference on Computer and communications security, pp. 545-554, 2009.
122. M. Watin-Augouard *et al.*, *Prospective Analysis on Trends in Cybercrime from 2011 to 2020*, French National Gendarmerie report, 2012, http://www.mcafee.com/us/resources/white-papers/wp-trends-in-cybercrime-2011-2020.pdf
123. T. Wilson, *New TDSS/TDL4 malware infects 46 of Fortune 500 companies*, news article, 18 September 2012, http://www.darkreading.com/attacks-breaches/new-tdsstdl4-malware-infects-46-of-fortu/240007496
124. C. Wisniewski, *Exposing the Money Behind the Malware*, Sophos Security Trent report, June 2012, http://www.sophos.com/en-us/medialibrary/Gated%20Assets/white%20papers/sophosmoneybehindmalwarewpna.pdf
125. R. Wojtczuk, R. Kashyap, *The Sandbox Roulette: are you ready to gamble?*, Black Hat EU, 12 March 2013.
126. R. Wojtczuk, J. Rutkowska, *Attacking Intel Trusted Execution Technology*, Black Hat USA, 18 February 2009.
127. R. Wojtczuk, J. Rutkowska, *Attacking SMM Memory via Intel CPU Cache Poisoning*, Invisible Things Lab, 2009, http://invisiblethingslab.com/
128. R. Wojtczuk, J. Rutkowska, *Following the White Rabbit: Software attacks against Intel VT-d technology*, Invisible Things Lab, April 2011, http://invisiblethingslab.com/
129. M. Wood, *Want My Autograph? The Use and Abuse of Digital Signatures by Malware*, SPOHOS, VB Conference, 2010.
130. C. Wüeest, *Current state of online Banking Trojans*, ITU-Impact Cybersecurity Forum presentation, 2012.
131. M. Zaharia, S. Katti, C. Grier *et al.*, *Hypervisors as a Foothold for Personal Computer Security: An Agenda for the Research Community*, Technical Report No. UCB/EECS-2012-12, University of California at Berkeley, 13 January 2012.
132. Y. Zhang, A. Juels, M.K. Reiter, T. Ristenpart, *Cross-VM Side Channels and Their Use to Extract Private Keys*, Proceedings of the 19th ACM Conference on Computer and Communications Security, pp. 305-316, 16 October 2012.
133. D.A.D. Zovi, *Hardware Virtualization Rootkits*, Black Hat USA, 29 July 2006.
134. –, *Possible VT-x enabled Intel CPU Crash Vulnerability*, http://seclists.org/fulldisclosure/2010/Mar/550, 31 March 2010.
135. –, *Advanced Configuration & Power Interface Specification*, Rev. 5.0, http://www.acpi.info/spec50.htm, December 2011.
136. –, *Annual Fraud Indicator*, UK National Fraud Authority report, March 2012, https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/118530/annual-fraud-indicator-2012.pdf
137. –, *sKyWIper (a.k.a. Flame a.k.a. Flamer): A complex malware for targeted attacks*, CrySyS Lab Technical Report, 31 May 2012, http://www.crysys.hu/skywiper/skywiper.pdf
138. –, *Vulnerability Note VU#649219: SYSRET 64-bit operating system privilege escalation vulnerability on Intel CPU hardware*, US CERT Vulnerability Database, http://www.kb.cert.org/vuls/id/649219, 12 June 2012.
139. –, *Bypassing Microsoft Windows ASLR with a little help by MS-Help*, 24 August 2012, http://www.greyhathacker.net/?p=585
140. –, *Phishing: How many take the bait?*, infographic, Get Cyber Safe campaign led by Public Safety Canada on behalf of the Government of Canada, 2013, http://www.getcybersafe.gc.ca/cnt/rsrcs/nfgrphcs/nfgrphcs-2012-10-11-eng.aspx
141. –, *Certified online banking Trojan in the wild*, The H Security news, 22 February 2013.
142. –, *Detecting Polymorphic Malware*, LAVASOFT research, http://www.lavasoft.com/mylavasoft/securitycenter/whitepapers/detecting-polymorphic-malware
143. –, *Trusted Computing: Criticism*, Wikipedia, http://en.wikipedia.org/wiki/Trusted_Computing#Criticism
144. –, *Symantec Validation and ID Protection Service*, http://www.symantec.com/verisign/vip-authentication-service
145. G. Heiser, B. Leslie, *The OKL4 Microvisor: Convergence Point of Microkernels and Hypervisors*, Proceedings of the first ACM Asia-pacific workshop on Workshop on systems, APSys '10, pp. 19-24, 2010.
146. U. Steinberg, B. Kauer, *NOVA: a microhypervisor-based secure virtualization architecture*, Proceedings of the 5th European conference on Computer systems, EuroSys '10, pp. 209-222, 2010.
147. J. Sun, D. Zhou, S. Longerbeam, *Supporting Multiple OSes with OS Switching*, Proceedings of USENIX Annual Technical Conference, pp. 357-362, 2007.
148. K. Sun, J. Wangy, F. Zhang, A. Stavrou, *SecureSwitch: BIOS-Assisted Isolation and Switch between Trusted and Untrusted Commodity OSes*, NDSS Symposium 2012.
149. O. Whitehouse, *GS and ASLR in Windows Vista*, Black Hat Washington DC, 2007.

Note: all links have been verified on 7 June 2013.

Received July 17, 2013