Trusted Computing vs. Advanced Persistent Threats: Can a defender win this game?

Nikos Virvilis, Dimitris Gritzalis, Theodoros Apostolopoulos

Information Security and Critical Infrastructure Protection Research Laboratory
Dept. of Informatics, Athens University of Economics & Business (AUEB)
76 Patission Ave., Athens, GR-10434 Greece
{nvir, dgrit, tca}@aueb.gr

Abstract: **As both the number and the complexity of cyber-attacks continuously increase, it is becoming evident that traditional security mechanisms have limited success in detecting sophisticated threats. Stuxnet, Duqu, Flame, Red October and, more recently, Miniduke, have troubled the security community due to their severe complexity and their ability to evade detection - in some cases for several years, while exfiltrating gigabytes of data or sabotaging critical infrastructures. The significant technical and financial resources needed for orchestrating such complex attacks are a clear indication that perpetrators are well organized and, likely, working under a state umbrella. In this paper we perform a technical analysis of these advanced persistent threats, highlighting particular characteristics and identifying common patterns and techniques. We also focus on the issues that enabled the malware to evade detection from a wide range of security solutions and propose technical countermeasures for strengthening our defenses against similar threats.**

Keywords: Advanced Persistent Threat, Duqu, Stuxnet, Flame, Red October, MiniDuke, Trusted Computing

I. INTRODUCTION

In the last two years or so, we have witnessed an impressive change in the complexity of malware. Stuxnet, Duqu, Flame, Red October and MiniDuke are examples of highly sophisticated malware (Advanced Persistent Threats - APT), the development of which required skillful individuals with expertise in multiple technology fields, as well as significant financial resources [1]. The term APT is frequently misused, offering an easy excuse for companies which suffer a security incident, as they can support that although following best practices (which is rarely the case) that the incident was inevitable due to the advanced offensive capabilities of the attacker.

However, we believe that the above mentioned malware has been correctly labeled as APT for two main reasons: a). It was used against sensitive state, military and industrial organizations in targeted attacks and has even been developed for a unique target (e.g. Stuxnet), and b). It makes use of advanced exploitation and evasion techniques, the majority of which were unknown (zero-day) to the security community.

Our contribution in this paper lies with: a). the technical comparison of the malware, based on our own analysis (focusing on behavioral/dynamic analysis of the samples in a controlled environment), as well as on published technical reports, and b). the identification of common attack patterns in the samples, in an effort to identify why current security solutions failed to detect those threats, and propose effective countermeasures for strengthening our defenses against similar threats.

The paper is organized as follows: Section 2 – Related work. Section 3 – Technical analysis of the malware, followed by the presentation of common characteristics in Section 4. In section 5, we describe out test environment and evaluate the effectiveness of specific countermeasures against the malware samples, followed by our conclusions and further work in Section 6.

II. RELATED WORK

In [2], the authors present a high level overview of Stuxnet's architecture. They point on the significant effort required in developing such malware, as well as on the fact that such attack would not be possible to succeed without insider

knowledge and support from a large team of experts. IBM [3], Symantec [4] and ESET [5] presented a detailed technical analysis of the malware and its modules. In [6-7], the authors presented a detailed technical analysis of Duqu malware, highlighting the similarities in terms of architecture and techniques used, with Stuxnet. They also developed a set of tools for detecting Duqu variants, which although effective, are only applicable for the specific malware sample. A small number of technical reports have been published for Flame [8, 9] analyzing its major components. Only preliminary online reports are available for Red October campaign, offering a high level technical view of the malware and its modules [10]. MiniDuke was discovered in February 2013 and early technical analysis revealed it's very unique architecture, a combination of zero-day sandbox evading exploits and pure assembly coding [11-12]. Based on our literature review, no specific models/solutions have been proposed for addressing such threats. Only a few technical countermeasures have been published for the detection of APT [13]. They focus on the identification of fast-flux DNS domains and signature matching. As none of the malware samples were using fast-flux to hide their command and control (C&C) infrastructure and transmitted network traffic was encrypted/-obfuscated, such countermeasures would have limited success discovering those threats. This paper is the sequel of our short paper [14], which includes the technical analysis of Miniduke and the evaluation of the proposed countermeasures.

Trusted Computing has been proposed as a robust solution against widespread malware (worms, trojan horses) [15-17] however, it still faces significant shortcomings which limit its adoption [18]. Furthermore, as of today and to the best of our knowledge there has not been any research on the potential benefits of Trusted Computing against Advanced Persistent Threats.

Finally, in the last few years we have witnessed an impressive rise in the complexity of smartphone malware [19]. The "bring your own device (BYOD)" trend, which has been endorsed by organizations due to the significant cost savings it offers, has turned smartphones into a high value target. Exploitation of such devices can give attackers direct access to the organization IT infrastructure and due to the limited security mechanisms available on such platforms they are an easy target [20-21].

III. TECHNICAL ANALYSIS

A. Stuxnet

Stuxnet was the first of the four highly complex malware that got detected and its earliest sample dates back to June 2009. Stuxnet's speculated purpose was to sabotage the Iranian Nuclear Program and more specifically Natanz uranium enrichment plant, something which has succeeded by causing physical damage to the infrastructure and as a result slowing down the program by four years [22]. Stuxnet interfered with Industrial Control Systems (ICS), which were configured by Programmable Logic Controllers (PLC), and more specifically Windows systems using Siemens Step-7 software. After infecting such systems, Stuxnet would reprogram the PLC to make the centrifuges operate at speeds outside acceptable limits, causing their malfunction and eventually destruction. Stuxnet was completely autonomous - a "fire and forget weapon" in military jargon and as a result, there was very limited window for programming or logical errors from the attacker's side.

In order to develop such a complex threat, the team behind Stuxnet should have at minimum: a) access to a test environment with all relevant ICS, PLC, centrifuges and supporting equipment - ideally an exact replica of the Natanz infrastructure, b) experts for installing, configuring and operating such specialized equipment, and c) a team of highly skilled programmers and security researchers for the development of zero-day exploits used for infection of the targets (or access to a private exploit repository). Based on these requirements, it is logical to assume that the team behind Stuxnet was state-sponsored [2]. An interesting development on Stuxnet's case happened recently, where an older sample was identified, named as Stuxnet 0.5 [23]. Preliminary research revealed that it had been active since 2005, supporting a subset of Stuxnet's functionality and following a different technique for controlling the centrifuges, which was abandoned due to limited success in damaging them.

Initial infection and propagation.

The initial infection method has not been identified, but taking into account that PLC systems are usually not connected to the internet (and most of the time not even connected to a network at all), it could be due to the use of an infected removable drive. This is a realistic assumption, as Stuxnet actually infected removable drives. Stuxnet was able to spread, using multiple propagation methods: Once an infected USB drive was connected to a windows system, Stuxnet would auto-execute requiring no user interaction, making use of a zero-day vulnerability (MS10-046, although older versions of Stuxnet used a modified autorun.inf technique). Also, it would try to exploit any network accessible windows systems using the Windows Server Service (MS08-067) or Print Spooler Zero-Day (MS10-061) vulnerabilities and perform privilege escalation using MS10-073 and MS10 -092.

Apart from the first vulnerability, the rest were unknown to the security community (0-day). This means that Stuxnet's team had either developed those exploits internally, or had access to a private exploit repository. Other propagation methods include copying itself to accessible network shares and the infection of WinCC database server using a hard-coded database password.

Finally, Stuxnet infected Step 7 project files, which - if copied in another system and opened - would infect it.

When Stuxnet managed to gain access to its target systems - a Windows System with Siemens Step-7 PLC - it would re--program the PLC, thus making the centrifuges operate outside their limits and eventually destroy them.

Command and Control Servers.

Stuxnet was mainly a "fire and forget" malware. However three C&C Servers have been identified, where the malware was trying to connect to (assuming there was internet access), to send basic information about the infected system.

Rootkit Functionality.

Stuxnet included rootkit code to hide its binaries on windows systems and also modified PLC code to present "acceptable" values to the monitoring software although the actual systems were working above limits. It also used of two compromised digital certificates to sign its drivers in an additional effort to evade detection [2].

Evasion Techniques.

It would scan for known endpoint security products and based on product name and version it would inject its payload accordingly, to evade detection.

Encryption.

Stuxnet was using XOR encryption with a static key (0xFF) to decrypt parts of its payload and a 32-byte fixed key to encode the data it sent to the C&C server, using again a XOR algorithm.

B. Duqu

Duqu was detected in September 2011. However, it is believed that it has been active since February 2010 [6]. It has significant similarities with Stuxnet, which have led researchers to believe that both threats were developed by the same team, with a different objective [7]: Instead of sabotage, Duqu's objective was espionage. Duqu was a clearly targeted malware and according to estimations infected no more than 50 targets worldwide [9]. After initial infection, Duqu remained active for 36 days before self-destructing, although attackers could command it to persist for as long as needed. It included a key logging component which was used to collect sensitive information, such us passwords, which attackers could use to gain access to other systems on the network. Similarly with Stuxnet, it was a modular malware, which made use of compromised certificates to sign its components and thus be harder to detect.

Initial infection and propagation.

Microsoft Word files which contained the zero day True Type font parsing vulnerability (CVE-2011-3402) were used as the initial attack vector. The malware did not replicate on its own. Nevertheless, attackers could use an infected system as a stepping stone, for manually exploiting and infecting other systems on the same network.

Command and Control Servers.

A small number of C&C Servers running CentOS Linux were identified. The malware connected to these servers over ports 80/TCP and 443/TCP, and used a custom C&C protocol. More specifically, for port 443/TCP a custom encrypted protocol was used. For traffic destined to port 80/TCP, Duqu used steganography, by encoding and attaching the transferred data to JPEG image files.

Rootkit Functionality.

Similarly to Stuxnet, Duqu was using a rootkit module to hide its files.

Evasion Techniques.

Duqu, having a similar list as Stuxnet, would scan for known security products and based on the product and version it would inject its payload accordingly, to evade detection.

Encryption.

Duqu used AES-CBC for the decryption of executable code received from the C&C server. Additionally, it used XOR to encrypt the data captured by the key logger module and to encrypt the configuration file [6].

C. Flame

Flame was first detected in May 2012. However, it is believed that it had been already active for 5-8 years. Flame was incidentally discovered while researching for another malware infection [8]. One of the most interesting aspects of this malware is its size, almost 20 megabytes, including all its modules, which is very uncommon [9]. There are no strong connections between Flame and Stuxnet or Duqu, so it is unlikely that Flame was developed by the same team that developed the other two threats. However, based on the use of the same exploits and common code/functions, it could be the case that these teams were collaborating and had shared source code for at least one module [24]. Flame, like Duqu, was a targeted information stealing malware but significantly more widespread, as it had infected thousands Windows system, mainly in Middle East. It had a key logging module similar to Duqu, took screenshots, intercepted email messages, used the internal microphone of the computer to record conversations and captured information about Bluetooth devices in proximity.

Initial infection and propagation.

Currently, Flame's initial infection point is not clearly known. However, as it infected USB devices, this is a realistic scenario. It did not replicate on its own, but attackers

could command it to infect other hosts using a variety of techniques: Apart from infecting USB devices, it made use of two zero-day vulnerabilities (same as Stuxnet) Print Spooler (MS10-061) and Windows Shell (MS10-046). However the most impressive propagation technique was the impersonation of a Windows Update Server (WSUS). As all software updates are digitally signed, the attackers had to perform a complex cryptanalytic attack (chosen prefix collision attack for the MD5 algorithm) against Microsoft's Terminal Services licensing certificate authority. This attack enabled generation of valid digital signatures for Flame modules. Such advanced cryptanalytic attack required a team of skilled cryptographers. According to estimations, it has cost between 200K and 2M USD [1], another indication that such attacks are likely to be state-funded.

Command and Control Servers.

Flame used more than 80 domains as C&C Servers, mostly Ubuntu Linux Servers. The communication was performed over HTTP, HTTPS or SSH.

Rootkit Functionality.

Flame's rootkit functionality enabled it to hide its network connections.

Evasion Techniques.

Flame included an extended list of more than 100 security products and adopted its strategy accordingly to evade them. Its binaries were using the .ocx extension as it is often not scanned by antivirus engines in real time.

Encryption.

Flame made extensive use of encryption, using substitution ciphers, XOR encryption, and RC4 algorithm to encrypt its configuration, modules and captured data.

D. Red October

Red October was discovered in October 2012. Although it is still under analysis, current findings indicate that it is another targeted malware for information gathering. It is believed that it has been active since May 2007, targeting diplomatic, governmental and scientific agencies [10]. Red October does not seem to have common characteristics with any of the other three malware samples. It followed a minimalistic architecture, with one main component responsible for connecting to the C&C Servers. When commanded to do so by the attackers, it downloaded and executed specific modules (at least 1000 different modules have been identified) enabling it to perform a wide range of tasks [10]. This small footprint was one of the main reasons it managed to evade detection for several years. Some characteristic functionality includes: Stealing of information from Nokia phones and i-Phones, SNMP brute forcing in an effort to gain access to network devices, and recovery of deleted files on removable drives.

Moreover, it included key logging/screen capturing functionality and intercepted outlook's email messages, calendar and contacts list. As a robust persistence mechanism, Red October installed a plugin for Office and Adobe reader applications. This parsed each opened Office or PDF file and tried to identify embedded commands (injected by the attackers) to execute. Using this technique, even if the C&C servers were taken down, the attackers could email specially crafted documents to their victims and regain control of their systems.

Initial infection and propagation.

Targeted emails containing malicious Word and Excel documents which exploited known vulnerabilities (CVE-2009-3129, CVE-2010-3333 and CVE-2012-0158) were used for infecting the targets.

Each malware build was unique for the specific target and each e-mail was also tailor-made to increase the probability of been opened by the victim. Also, a Java exploit (CVE-2011-3544) was used for delivering the malicious payload.

Command and Control Servers.

More than 60 C&C domains were identified. However only three hardcoded domains were included in each custom build of the malware. Additionally, none of these domains were the actual C&C servers, but acted as proxies in order to hide the real C&C infrastructure, which is currently unknown.

Rootkit Functionality.

No rootkit component has been identified.

Evasion Techniques.

The minimalistic architecture of the malware, having a basic component responsible for downloading encrypted modules and executing them in memory, allowed it to remain undetected without having to perform additional evasion techniques.

Encryption.

Red October used a custom packer with XOR encryption. The same algorithm was also used for encrypting exfiltrated data.

D. MiniDuke

MiniDuke was discovered on 27 February 2013, but earlier samples have been identified and date back in June 2011 [11]. It targeted government bodies in 23 countries, mainly in Europe. It had a unique architecture, as it combined mod-

ern exploitation techniques to bypass Adobe's PDF sandbox, and pure assembly coding for its payload [12].

Initial infection and propagation.
MiniDuke spread over email, using malicious, well-crafted PDF files. The PDF files contained code which triggered a vulnerability in Abobe Reader versions 9, 10 and 11, bypassing the sandbox and executing the malware's payload on the victim's system

Command and Control Servers.
The malware used a sophisticated, layered technique for locating the C&C servers. Initially (first stage) it connected to specific Twitter accounts controlled by the attackers, which contained the encrypted URL's of the C&C servers. If Twitter was not accessible (e.g. twitter accounts had been blocked) the malware would use Google Search to find the encrypted C&C servers by searching for specific unique strings. Upon locating the servers, the malware received additional malicious payload (second stage), obfuscated as GIF images, which in turn would download a larger backdoor (third stage) enabling the attackers to control the infected system. A large number of C&C servers have been identified, however all of them have been legitimate web sites, compromised by the attackers.

Rootkit Functionality.
No rootkit functionality has been identified.

Evasion Techniques.
MiniDuke's first stage, written completely in assembly language, contained a list of security related processes (debuggers, disassemblers, file system monitoring software, etc). Upon detection of any of these processes, it would remain at idle state, not performing any malicious actions. Newer samples would also wait for user interaction (e.g. mouse movement) before decrypting and executing the payload to thwart automated malware analysis. Finally, after initial execution the malware would generate a new copy of itself - that was encrypted using a key derived from the computer's hardware configuration - and replace the original executable. As a result, the malware would only be able to decrypt correctly under this particular system, making analysis of the sample on a different system significantly challenging.

Encryption.
MiniDuke was making heavy use of encryption. Each payload was specially built for each victim and was decrypted based on a key generated by the CPU, Drive and Computer name of the victim. Additional layers of XOR and ROL obfuscation were also used.

TABLE I – ADVANCED PERSISTENT THREATS COMPARISON

	Stuxnet	**Duqu**	**Flame**	**Red October**	**Mini Duke**
Active since	June 2009 (2005)	Nov. 2010	May 2012 (2006)	May 2007	June 2011
Detected	June 2010	Sept. 2011	May 2012	Oct. 2012	Feb. 2013
PE Type	DLL		OCX	EXE	EXE
Initial infection	Unknown	MS Word	Unknown	MS Excel / Word, Java	PDF
Replication	Removable drives, network	Manual replication only			
Rootkit module	Yes		No		
Key logging	No	Yes			No
Evasion	Yes			No	Yes
Encryption	XOR	XOR, AES-CBC	XOR, RC4, Substitution	XOR	Unique per victim, XOR, ROL
Target	Sabotage	Information gathering			

IV. COMMON MALWARE CHARACTERISTICS AND POTENTIAL REASONS FOR DETECTION FAILURE

Based upon our technical analysis, we have identified common malware characteristics, such as: similar techniques, attack paths and functionality. Focusing on these characteristics, we highlight in the sequel the potential issues that could have enabled the malware to evade detection by commonly used security technologies:

A. Targeted operating system and architecture.

All samples were targeting 32-bit versions of Windows. Interestingly enough, none of the malware would execute on 64-bit systems. The additional security mechanisms that are

available on the 64-bit versions of windows (especially windows 7 and newer) complicate significantly the exploitation, especially when malware targets kernel components [25]. However, based on the fact that attackers had access to valid certificates (Stuxnet, Duqu) that they could use to sign the 64-bit components of their malware (thus allowed to execute in kernel space), we consider that the main reason that the malware targeted 32-bit systems was, that the majority of the victims were using this architecture.

B. Initial attack vectors.

Duqu and Red October both used malicious Word and Excel documents for infecting their targets, while MiniDuke exploited Adobe's PDF Reader. For Stuxnet and Flame the initial infection method has not been identified, however infection through removable drives or spearfishing attacks, is the most likely scenario. Microsoft since the release of Office 2010 Suite, has introduced "protected view" [26], a sandbox feature for opening office documents received from untrusted sources - such as downloaded from internet or received as an email attachment. By default, all new files are opened in protected mode, thus any potentially malicious payload would have to face the additional barrier of escaping the sandbox.

Further analysis proved that none of the Office vulnerabilities was able to escape the sandbox. Thus, we have to assume that victims who got infected were running outdated versions of MS Office and potentially other third party software (Java, etc.) and/or had disabled the security features offered by newer versions of the software. This was not the case for MiniDuke, which was able to exploit all versions of Adobe Reader and at the same time evade the sandbox.

C. Command execution and escalation of privileges.

All malware made use of exploits for command execution or privileged escalation, the vast majority of which were unknown to the security community (0-day). Exploitation of zero day vulnerabilities against the Operating System is probably the most challenging problem to address. As long as these vulnerabilities remain unknown to the vendor, all affected systems will be vulnerable. Even when security patches addressing these vulnerabilities had been released, still a large number of systems continued to get infected, highlighting the lack of patch management procedures, even in sensitive environments

D. Network Access.

All malware communicated over ports 80/TCP, 443/TCP or 22/TCP, as egress traffic destined to such ports is frequently allowed to pass through network access control mechanisms. The communication protocols were HTTP, HTTPS and SSH. However, additional layers of encryption/obfuscation and compression were also used. Malware's success to communicate back to the C&C infrastructure highlights the fact that most of the victims had very relaxed internet access restrictions in place (if any) - a worrying finding, taking into account the sensitive nature of the targeted organizations.

E. Network IDS and endpoint antivirus products.

Stuxnet, Flame, Duqu and MiniDuke were designed to detect and evade Antivirus Software, using multiple techniques. Moreover, Flame, Duqu, Red October and MiniDuke encrypted or obfuscated their network traffic, to and from the C&C servers, so as to "pass under the radar" of Network Intrusion Detection Systems (NIDS). As a result, the level of protection offered by both Antivirus and NIPS products against advanced threats has received strong criticism, after they failed to identify such threats - even when they had been active for several months/years [27]. The shortcomings of such systems, mainly due to the strong reliance on pattern matching (signature based detection) are well known to the research community for long [28-29], nevertheless, they are most common (if not the only one) set of tools that defenders have.

F. Use of Encryption / Obfuscation.

All samples relied strongly on XOR "encryption" to deter detection and complicate malware analysis (packing), as well as for protecting the configuration file(s) and transmitted traffic. Additionally Duqu and Flame also used AES and RC4 algorithms, respectively. The use of encryption in a program cannot be categorized as malicious on its own, however it is a useful indication during behavioral analysis of potentially malicious files.

G. Exploitation of digital signatures.

Stuxnet and Duqu binaries were digitally signed using compromised digital certificates. Thus, these samples would manage to infect hardened systems, where only digitally signed binaries were allowed to execute. Similarly, Flame exploited the collision resistance of MD5 hash function to perform the same task, and replicated through the WSUS Service.

Based on these findings, allowing execution of binaries based on the existence of valid digital signatures cannot be

considered an effective defense on its own. This is particularly important for Antivirus and HIPS solutions, which tend to avoid real-time analysis of signed binaries for performance reasons.

V. COUNTERMEASURES

Due to the complexity of those threats and the multiple attack paths, a wide range of security countermeasures and hardening procedures are necessary to enable a multi-layered, robust defense. In order to test the effectiveness of each proposed countermeasure in the section, we created the following environment: Each sample was executed in a Windows XP SP3 32bit virtual machine with Office 2007 SP1, while all registry, file system and network connections were monitored. The virtual machine was given a private IP address in the same range where other three additional virtual machines were located (Windows XP SP2 32bit, Windows XP SP3 32bit and Windows XP SP2 64bit), all vanilla installations. The default gateway of all VM's was pointing to a Linux box, which acted as a gateway/proxy and captured all network traffic.

A. Patch Management: Patch management, for both Operating System and third party applications, is the first line of defense. Although it will have limited effect on the mitigation of zero-day vulnerabilities, it will stop further exploitation of new systems when the vulnerabilities are discovered and addressed by the vendor. As expected, all malware samples failed to infect our test system when the corresponding vulnerabilities were patched.

B. Network Segregation: Strong internal network access controls and monitoring are also crucial. In the majority of cases, multiple systems had to be exploited until the objective of an attack was met (e.g. data exfiltration or sabotage). One of the most effective techniques that could severely block the ability of the malware to spread internally is the isolation between the workstations. This could be achieved by the use of network or host based firewalls, configured to allow workstations to only connect to specific server systems, based on their role/business requirements (e.g. Active directory, Mail, Web/Proxy Server).

Although attackers could focus on exploiting the servers, servers tend to be more closely monitored and hardened than workstations, raising the bar for attackers. As Stuxnet was the only sample that would self-propagate, we executed it in the test VM, while monitoring all traffic to the VM network. After a short period of time the two 32bit Windows VM's were also infected. We repeated the experiment (after restoring the VM's in a clean state), but this time we activated the Windows firewall (blocking all ingress ports) on the VMs. As expected, traffic from the infected VM was blocked and thus Stuxnet failed to propagate.

C. Whitelisting: Furthermore, as all malware had to connect back to a C&C server for receiving commands or exfiltrating data, strict internet access policies and granular traffic inspection of both incoming and outgoing data, are required. Instead of following a blacklist approach which would inevitably fail to block all malicious traffic, a relaxed whitelist approach can be a much more secure alternative for use, in sensitive organizations. Depending on the risk appetite of the organization, the whitelist could even include the most common (non-work related) websites that users are visiting. This would completely block any malicious connection attempts to C&C servers, unless attackers were able to exploit any whitelisted services/systems and set their C&C infrastructure there.

For this test we configured our Linux gateway to allow outgoing traffic to the Alexa top 100 web sites [31]. Then we executed each sample on the victim VM, monitoring the network traffic. As the C&C domains have long been taken down, we were not expecting a successful connection to any C&C server, however our test was to prove that all malware specific domains that samples tried to connect to, were blocked by the firewall as there were not in the white list.

D. Dynamic content execution: Focusing on the client side exploitation mechanisms, it is evident that the majority of end-user exploitation techniques (e.g. Malicious Office, PDF documents) required dynamic content execution for triggering the vulnerabilities. Filtering mechanisms at the network ingress points could filter dynamic content in incoming traffic (e.g. Javascript from PDF and html files, macro's from Office files) thus protecting against a wide range of exploitation mechanisms [30].

For the test we installed Microsoft Office 2010 on our test VM (as the protected view feature is supported from version 2010 and later). Then we opened two malicious word files containing Duqu and Red October samples. When the malicious documents were copied locally e.g. copied from a USB device, Protected View was not activated and thus did not provide any protection against the malicious files. In addition as Duqu's exploit was not Office Specific but exploited then Win32k TrueType font parsing engine and thus even with protected view enabled, the system was infected. However Protected View did block the infection

from Red October when the malicious word file exploiting the CVE-2010-3333 vulnerability was executed from a network location.

E. Trusted Computing: A secure computing base by limiting and controlling the software that is allowed to be installed and executed on a system, would significantly reduce the impact of APT attacks. Although the limitations have been extensively discussed [18], we should take into account that the majority of APT attacks target sensitive organizations-/critical infrastructures. We believe that in such environments, the benefits introduced by a TCB significantly overcome the shortcomings.

VI. CONCLUSION

It is evident that use of widely accepted best practices/-countermeasures such as the ones presented in the previous section, would have reduced the impact of such malware. Unfortunately, it seems that even in critical infrastructures, only a small subset of such protection mechanisms was enforced. The required resources, technical expertise and the negative impact on user experience (due to the additional limitations) are blocking their adoption.

Based on the fact that traditional security solutions, have failed repeatedly to address the APT problem and organizations are reluctant to adopt high maintenance solutions/-countermeasures, we need to shift our focus on more robust and transparent solutions. We consider that a Trusted Computing Base can be an invaluable tool in addressing this multi-dimensional problem.

Our future research will take into account, build upon and exploit our earlier publications record on fighting malware [32-36] and will focus on the design of a generic TCB architecture, which could be implemented in security-critical contexts/infrastructures, with a foreseen limited impact on the business operations and user experience.

REFERENCES

[1]. Fisher, D.: Threatpost. [Online] 15 June 2012 [Cited: 01 02 2013] https://threatpost.com/en_us/blogs/what-have-we-learned-flame-malware-061512

[2]. Chen, T., Abu-Nimeh, S.: 2011. Lessons from Stuxnet. Computer 44, 4 (April 2011), 91-93. http://dx.doi.org/10.1109/MC.2011.115

[3]. Larimer, J.: An inside look at Stuxnet, IBM, 2010.

[4]. Falliere, N., Murchu, L., Chien, E., W32.Stuxnet Dossier, Symantec, February 2011.

[5]. Matrosov, A., Rodionov, E., Harley, D., Malcho, J., Stuxnet Under the Microscope, ESET, 2011.

[6]. Symantec, W32.Duqu - The precursor to the next Stuxnet, Symantec, 2011.

[7]. Bencsath, B., Pek, G., Buttyan, L., Felegyhazi, M., "Duqu: Analysis, detection, and lessons learned", in Proc. of the 2nd ACM European Workshop on System Security, 2012.

[8]. Bencsath, B., Pek G., Buttyan L., Felegyhazi M., "The Cousins of Stuxnet: Duqu, Flame, and Gauss", in Proc. of Future Internet, 2012.

[9]. Kaspersky Labs. [Online] 28 May 2012 [Cited: 01 January 2013] http://www.securelist.com/en/blog?weblogid=208193522

[10]. Kaspersky Labs. "Red October" Diplomatic Cyber Attacks Investigation. [Online] 14 01 2013 [Cited: 15 01 2013] http://www.securelist.com/en/analysis/204792262/Red_October_Diplomatic_Cyber_Attacks_Investigation

[11]. Tivadar, M., Balazs, B., Istrate, C., A closer look at Miniduke. [Online] [Cited: 20 05 2013] http://labs. bitdefender.com/wp-content/uploads/downloads/2013/04/ MiniDuke_ Paper_Final.pdf

[12]. Raiu, C., Soumenkov, I., Baumgartner, K., Kamluk V., "The MiniDuke Mystery" [Online] [Cited: 10 05 2013] https://www.securelist.com/en/downloads/vlpdfs/themysteryofthepdf0-dayassemblermicrobackdoor.pdf

[13]. Binde, E., McRee, R., O'Connor, T., "Assessing Outbound Traffic to Uncover Advanced Persistent Threat", SANS Institute. Online] 2011 [Cited: 21 12 2012] http://www.sans.edu/student-files/projects/JWP-Binde-McRee-OConnor.pdf

[14]. Virvilis N., Gritzalis D., "The Big Four - What we did wrong in protecting critical ICT infrastructures from Advanced Persistent Threat detection?", in Proc. of the 8th International Conference on Availability, Reliability & Security, pp. 248-254, IEEE, 2013.

[15]. Gajek, S., “Compartmented security for browsers - or how to thwart a phisher with trusted computing”, in Proc. of the 2nd International Conference on Availability, Reliability and Security, p. 120-127, 2007

[16]. Kuhlmann, D., Landfermann, R., Ramasamy, H., Schunter, M., Ramunno, G., Vernizzi, D., “An open trusted computing architecture - Secure virtual machines enabling user-defined policy enforcement”, Web Page: http://www. opentc.net/activities/otc_HighLevelOverview/OTC, 2006.

[17]. Litty, L., Lie, D., “Manitou: a layer-below approach to fighting malware”, in Proc. of the 1st Workshop on architectural and system support for improving software dependability, pp. 6-11, ACM, 2006.

[18]. Oppliger, R., Rytz, R., “Does trusted computing remedy computer security problems?”, Security & Privacy, IEEE, 3(2), 16-19, 2005.

[19]. Mylonas, A., Tsoumas, B., Dritsas, S., Gritzalis, D., “Smartphone security evaluation: The malware attack case”, in Proc. of the 8th International Conference on Security and Cryptography, pp. 25-36, SciTekPress, 2011.

[20]. Mylonas, A., Dritsas, S., Tsoumas, B., Gritzalis, D., “On the feasibility of malware attacks in smartphone platforms”, Security and Cryptography, pp. 217-232, Springer (CCIS-314), 2012.

[21]. Gritzalis, D., “Embedding privacy in IT applications development”, Information Management & Computer Security, vol. 12, no. 1, pp. 8-26, 2004.

[22]. Lemos, R., Infoworld. [Online] 19 January 2011 [Cited: 02 February 2013] http://www.infoworld.com/t/malware/stuxnet-attack-more-effective-bombs-888

[23]. Symantec, Stuxnet 0.5: The missing link, Symantec, 2013.

[24]. Kaspersky Labs. [Online] 11 June 2012. [Cited: 3 February 2013.] http://www.kaspersky.com/about/news/virus/2012/Resource_207_Kaspersky_Lab_Research_Proves_that_ Stuxnet_and_Flame_Developers_are_Connected

[25]. Field, S., An introduction to kernel patch protection. [Online] 12 August 2006. [Cited: 08 February 2013.] http://blogs.msdn.com/b/

[26]. Microsoft: [Online] [Cited: 11 02 2013.] http://office.microsoft.com/en-001/excel-help/what-is-protected-view-HA010355931.aspx.

[27]. Schneier, B., www.schneier.com. [Online] 19 June 2012. [Cited: 6 February 2013.] http://www.schneier.com /blog/archives/2012/06/the_failure_of_3.html.

[28]. Cohen, F., Computer Viruses, ASP Press, 1986.

[29]. Denault, M., Gritzalis, D., Karagiannis, D., Spirakis, P., “Intrusion detection: Evaluation and performance issues of the SECURENET system”, Computers & Security, vol. 13, no. 6, pp. 495-508, 1994.

[30]. Lagadec P., "Diode réseau et ExeFilter: 2 projects pour des interconnexions sécurisées", in Proc. of SSTIC06, pp. 1-15, 2006.

[31]. Alexa Top Sites - http://www.alexa.com/topsites

[32]. Doumas, A., Mavroudakis, K., Gritzalis, D., Katsikas, S., “Design of a neural network for recognition and classification of computer viruses”, Computers & Security, vol. 14, no. 5, pp. 435-448, October 1995.

[33]. Gritzalis, D., "Enhancing security and improving interoperability in healthcare information systems", Informatics for Health and Social Care, vol. 23, no. 4, pp. 309-324, 1998.

[34]. Kandias, M., Mylonas, A., Virvilis, N., Theoharidou, M., Gritzalis, D., “An Insider Threat Prediction Model”, in Proc. of the 7th International Conference on Trust, Privacy, and Security in Digital Business, pp. 26-37, Springer (LNCS 6264), 2010.

[35]. Katsikas, S., Spyrou, T., Gritzalis, D., Darzentas, J., "Model for network behaviour under viral attack", Computer Communications, vol. 19, no. 2, pp. 124-132, 1996.

[36]. Katsikas, S., Gritzalis, D., Spirakis, P., “Attack Modeling in Open Network Environments”, in Proc. of the 2nd Communications and Multimedia Security Conference, pp. 268-277, Chapman & Hall, 1997.