The Big Four – What we did wrong in Advanced Persistent Threat detection?

Nikos Virvilis, Dimitris Gritzalis

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}@aueb.gr

Abstract: As both the number and the complexity of cyber-attacks continuously increase, it is becoming evident that current security mechanisms have limited success in detecting sophisticated threats. Stuxnet, Duqu, Flame and Red October have troubled the security community due to their severe complexity and their ability to evade detection - in some cases for several years. 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 authors 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, Zero Day, Exploitation, Duqu, Stuxnet, Flame, Red October.

1. Introduction

In the last two years, or so, we have witnessed an impressive change in the complexity of malware. Stuxnet, Duqu, Flame, and Red October 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]. Before the Stuxnet era, Cyber Espionage and Cyber War were terms frequently challenged as marketing slogans, used by security companies for promoting their solutions. Nowadays, the capabilities and effectiveness of such threats cannot be questioned any more: Stuxnet appears to have slowed down the Iran's nuclear program by four years - possibly causing greater damage than a traditional military attack [2]. Duqu, Flame and Red October have managed to operate silently for a long time, exfiltrating vast amounts of data from sensitive environments [3].

Our contribution in this paper is: (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 published technical reports, and (b) the identification of initial common attack patterns, in an effort to identify why current security solutions failed to detect those threats and propose realistic 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 present our proposed countermeasures, followed by our conclusions and further work in Section 6.

2. Related work

In [4], 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. In [5], a detailed technical analysis of the malware and its modules has been presented. In [6], the authors presented a detailed 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. Only limited technical information has been published for Flame [7]. Preliminary online reports are available for Red October campaign, published by Kaspersky, offering a high level technical view of the malware and its modules [8].

Apart from the technical analysis and standalone tools proposed for detection of individual samples, we were unable to find related work focusing on

978-0-7695-5008-4/13 $26.00 © 2013 IEEE
DOI 10.1109/ARES.2013.32

the detection of such threats or the analysis of the reasons that enabled malware authors to evade security solutions for years. A number of technical countermeasures have been published [9], focusing on the detection 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 in the detection of those threats.

3. Technical Analysis

3.1 Stuxnet

Stuxnet was the first of the four highly complex malware that got detected. Its earliest sample dates from June 2009. Stuxnet's speculated purpose was to sabotage the Iranian Nuclear Program, more specifically Natanz uranium enrichment plant. It appears that it has succeeded by causing physical damage to the infrastructure and slowing down the program by four years [2]. 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. As a result, there was very limited window for programing 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 a 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, one may assume that the team behind Stuxnet was state sponsored. Multiple Infections have been reported worldwide with the majority of them in Iran. This can be explained due to the plethora of propagation methods that Stuxnet supported, which could have resulted in spread of the malware outside the target infrastructure. Nevertheless, although Stuxnet infected multiple similar infrastructures, it has only activated its payload (and thus caused physical damage) in controllers at Natanz uranium enrichment plant [10], highlighting the fact that it was indeed targeted to attack that particular infrastructure.

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). Other propagation methods include copying itself to accessible network shares and the infection of WinCC database server using a hardcoded 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 system(s) - a Windows System with Siemens Step-7 PLC, it would reprogram the PLC, making the centrifuges operate outside limits and eventually destroy them.

Command and Control Servers: 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.

Targets Security Products: 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.

3.2 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 [7]. After initial infection, Duqu remained active for 30 days before self-destructing, although attackers could command it to persist for as long as needed. It includeed 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.

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.

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.

Targets Security Products: 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 information stealer (key logger) module, and to encrypt the configuration file [7].

3.3 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 [7]. One of the most interesting aspects of this malware is its size, almost 20MB, including all its modules, which is very uncommon [11]. 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. Flame, like Duqu, was a targeted information stealing malware but significantly more widespread, as it had infected thousands Windows systems, mainly in Middle East. It had a key logging module similar to Duqu, took screenshots, intercepted email messages and used the internal microphone of the computer to record conversations.

Initial infection and propagation: Currently, Flame's initial infection point is not clearly known. It did not replicate on its own. Apart from infecting USB devices (which is likely the initial infection method), 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 may 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 [11].

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

Targets Security Products: Flame included an extended list of more than 100 security products and adopted its strategy accordingly to evade them [11]. 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.

3.4 Red October

Red October was discovered in October 2012. It is believed that it has been active since May 2007, targeting diplomatic, governmental and scientific institutions [8]. 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 [8]. 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 iPhones, SNMP brute forcing in an effort to gain access to network devices, and recovery of deleted files on removable drives. 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 a specific attacker supplied commands to execute.

Initial infection and propagation: Targeted emails containing malicious Word and Excel documents, which exploited known vulnerabilities (CVE-2009-3129, CVE-2010-333 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.

Command and Control Servers: More than 60 C&C domains were identified. However only 3 hardcoded domains were included in each custom build of the malware.

Rootkit Functionality: Currently no rootkit component has been identified. However, and due to the large number of modules, which have not been completely analyzed yet, this is still uncertain.

Targets Security Products: 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 made use of XOR encryption to pack its main executable and for encoding exfiltrated data.

Table 1 –Advanced Persistent Threats comparison

APT's name	Stuxnet	Duqu	Flame	Red October
Active since	June 2009 (2005)	November 2010	May 2012 (2006)	May 2007
Detected	June 2010	September 2011	May 2012	October 2012
PE executable	DLL		OCX	EXE
Initial infection	Unknown	MS Word	Unknown	MS Excel/Word, Java
Self-replication	Removable drives, Over the network	Manual replication only		
Rootkit functionality	Yes		No - Not yet known	
Key logging module	No	Yes		
Targets sec. products	Yes			No
Encryption	XOR	XOR, AES-CBC	XOR, Substitution, RC4	XOR
Target	Sabotage	Information gathering		

4. Common malware characteristics and potential reasons for detection failure

Based on our technical analysis, we identified a number of common malware characteristics.

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.

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 [13]. 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.

Initial attack vectors

Duqu and Red October both used malicious Word and Excel documents for infecting their targets. For Stuxnet and Flame the initial infection method has not been identified, however infection through removable drives or spearfishing attacks, are realistic scenarios. Microsoft, since the release of Office 2010 Suite, has included “protected view” [14], a sandbox feature for opening office documents received from untrusted sources. 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. As a result, we have to assume that victims who got infected were running outdated versions of Microsoft Office.

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). However, the number of infected victims continued to increase, even when security patches addressing these vulnerabilities had been released, highlighting the lack of effective patch management processes.

Network Assess

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. 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.

Network IDS and endpoint antivirus products

Stuxnet, Flame, and Duqu were designed to detect and evade Antivirus Software, by having a list of different evasion techniques. Moreover, Flame, Duqu and Red October 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). 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 [15-16], nevertheless, they are most common (if not the only one) set of tools that defenders have.

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. The use of encryption cannot be categorized as malicious. However, it can be a useful indication during behavioral analysis of potentially malicious files.

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. 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.

5. Defense-in-Depth

Due to the complexity of those threats and the multiple attack paths, no single security product can offer effective protection. Thus, a wide range of security countermeasures and hardening procedures are necessary to enable a multi-layered, robust defense.

In addition, the unique characteristics of each environment need to be taken into account, giving special focus on sensitive organizations/critical infrastructures [17-20].

Technical countermeasures

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. Moreover, new versions of frequently targeted software (e.g. Office Suite, Adobe Acrobat Reader, Flash Player) support additional security features (e.g. sandbox environment), which significantly hardens exploitation.

Strong network access controls and monitoring, on both the internal network and perimeter, 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). As a result, strong network access controls which allow systems/networks to communicate based on specific business requirement, would limit the propagation of the malware on the internal network.

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. Currently there are multiple security technologies supporting stateful application layer inspection and content scanning, use of which would have identified the connections from infected systems to the C&C servers as anomalous, due to the use of custom/obfuscated protocols.

Similarly, anomalies in email attachments, such as the inclusion of shellcode in Office Documents or high entropy content, could also be easily identified. Even if attackers tried to evade detection by conforming to protocol standards (e.g. tunneling all traffic through a regular https connection), protocol aware security solutions can be configured to drop any encrypted traffic unless they are able to intercept and decode it, effectively crippling the malware ability to connect to the C&C server.

Finally, even low cost solutions such as monitoring DNS queries, for the detection of unusual domains, which are periodically accessed by internal systems and monitoring network connections to third countries where there are no business connections, can help administrators to identify anomalies at minimal cost. Honeypots and honeynets [12] are excellent tools for detecting attackers trying to exploit internal systems, the majority of which are open source and require low maintenance cost. Especially in the case of Stuxnet, where the malware spread attacked all vulnerable systems on a network, honeynets would have raised immediate alerts.

Use of Host Based Intrusion Prevention systems (HIPS), would significantly increase the complexity of a successful attack, as the majority of the actions performed by the malware during infection (execution of compiled code from an Office document, Process Injection) would be easily identified as malicious. Controlled use of USB devices: As all samples propagated (or supported propagation) over USB removable drives, limiting access of such devices is an important countermeasure. If there is business need for their use, e.g. in the case of the PLC systems that were not networked, use of special USB devices that can only be connected to authorized workstations and disable of the autorun feature should be used.

6. Conclusion and further work

It is evident that use of widely accepted best practices would have significantly raised the bar for attackers and limited the impact of the malware. However, it seems that even in sensitive environments only a small subset of such protection mechanisms was enforced. Based on the fact that Antivirus and Network Intrusion Detection Products, two of the most widely used security technologies, face serious shortcoming in the detection of APT, we consider that there is need for a radical change in their architecture and detection strategies.

Although it is unrealistic to assume that a purely technical solution will offer complete protection against all threats, we consider that a shift of focus from black-box proprietary security products to a highly integrated architecture, combining multiple security technologies, would significantly enhance the level of protection.

As a result, our future work focuses on proposing a hybrid architecture for the detection of APT, by analyzing low severity events that attackers (or malware) will inevitably generate during the attack's life cycle. Our architecture will perform correlation of semi-real time and historical events from a wide range of sources, supporting holistic view of the infrastructure, enabling defenders to correlate sporadic low-severity events, as a result of an on-going sophisticated attack.

Acknowledgments

The authors would like to thank Yannis Mallios (Carnegie Mellon University) and Bill Tsoumas (European GNSS Agency) for their comments and suggestions.

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

[3]. Schwartz, M.: *Information week*. [Online] 14 January 2013 [Cited: 12 02 2013] http://www.informationweek.com/security/attacks/red-october-espionage-network-rivals-fla/240146215

[4]. Chen, T., Abu-Nimeh, S.: Lessons from Stuxnet. In: *Computer*, Vol. 44, No. 4, pp. 91-93, April 2011 DOI=10.1109/MC.2011.115 http://dx.doi.org/10.1109/MC.2011.115

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

[6]. Bencsath, B., Pek, G., Buttyan, L., Felegyhazi, M.: Duqu: Analysis, detection, and lessons learned. In: *Proc. of ACM European Workshop on System Security* (EuroSec-2012), 2012.

[7]. Bencsath, B., Pek, G., Buttyan, L., Felegyhazi, M.: The Cousins of Stuxnet: Duqu, Flame, and Gauss. In: *Future Internet*, Vol. 4, No. 4, pp. 971-1003, 2012.

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

[9]. 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

[10]. Langner, R.: Stuxnet: Dissecting a Cyberwarfare Weapon. In: *IEEE Security & Privacy*, Vol. 9, No. 3, pp. 49-51, May-June 2011.

[11]. Walter, J.: *"Flame Attacks": Briefing and Indicators.* McAfee Labs, 2012.

[12]. Spitzner, L.: *Honeypots: Tracking Hackers*. Addison-Wesley, USA, 2002.

[13]. Field, S.: *An introduction to kernel patch protection.* [Online] 12 August 2006. [Cited: 08 02 2013.] http://blogs.msdn.com/b/

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

[15]. Cohen, F.: *Computer Viruses*. ASP Press, USA, 1986.

[16]. Denault, M., Gritzalis, D., Karagiannis, D., Spirakis, P.: Intrusion detection: Evaluation and performance issues of the SECURENET system. In: *Computers & Security*, Vol. 13, No. 6, pp. 495-508, October 1994.

[17]. Doumas, A., Mavroudakis, K., Gritzalis, D., Katsikas, S.: Design of a neural network for recognition and classification of computer viruses. In: *Computers & Security*, Vol. 14, No. 5, pp. 435-448, October 1995.

[18]. Lambrinoudakis, C., Gritzalis, D., Tsoumas, V., Karyda, M., Ikonomopoulos, S.: Secure electronic voting: The current landscape. In: *Secure Electronic Voting*, Gritzalis D. (Ed.), pp. 101-122, Kluwer Academic Publishers, USA, 2003.

[19]. Lekkas, D., Gritzalis, D.: Long-term verifiability of healthcare records authenticity. In: *International Journal of Medical Informatics*, Vol. 76, Issue 5-6, pp. 442-448, 2006.

[20]. Iliadis, J., Gritzalis, D., Spinellis, D., Preneel, B., Katsikas, S.: Evaluating certificate status information mechanisms. In: *Proc. of the 7th ACM Computer & Communications Security Conference* (CCS-2000), pp. 1-9, ACM Press, USA, 2000.