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

임 형 진(Hyung Jin, Im)

Email: dlaudwls90@gmail.com
Table of Contents

1. Introduction
2. Related Work
3. Technical Analysis
4. Common Malware Characteristics
5. Countermeasures
6. Conclusion
1. Introduction

Stuxnet, Duqu, Flame, Red October and MiniDuke are examples of highly sophisticated malware (Advanced Persistent Threats – APT)
2. Relation works

• Trusted Computing has been proposed as a robust solution against widespread malware (worms, trojan horses)
• The best of our knowledge there has not been any research on the potential benefits of Trusted Computing against Advanced Persistent Threats
• 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.
• Overview
  – Stuxnet 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.

* Sabotage is a deliberate action aimed at weakening a polity or corporation through subversion, obstruction, disruption, destruction, or underhand tactics
• Initial infection and propagation.
  – taking into account that PLC systems are usually not connected to the internet, it could be due to the use of an infected removable drive.
  – 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)
  – 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(Window Xp Kennel) and MS10-092(Task Schedule).
• **Command and Control Servers**  
  – three C&C Servers have been identified, where the malware was trying to connect, to send basic information about the infected system.

• **Rootkit Functionality.**  
  – 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.
• **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.
• **Overview**
  - Duqu was detected in September 2011.
  - 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.
  - Duqu’s objective was espionage. Duqu was a clearly targeted malware and according to estimations infected no more than 50 targets worldwide.
  - 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.
• **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.
  – 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
• **Overview**
  – 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.
  – One of the most interesting aspects of this malware is its size, almost 20 megabytes, including all its modules, which is very uncommon.
  – 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.

- Flame’s initial infection point is not clearly known. However, as it infected USB devices.
- 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).
- 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 against Microsoft's Terminal Services licensing certificate authority.
• **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.
• OverView
  – Red October was discovered in October 2012.
  – Red October does not seem to have common characteristics with any of the other three malware samples.
  – 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.
• 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.
• Overview
  – MiniDuke was discovered on 27 February 2013, but earlier samples have been identified and date back in June 2011.
  – It targeted government bodies in 23 countries, mainly in Europe. It had a unique architecture, as it combined modern exploitation techniques to bypass Adobe’s PDF sandbox, and pure assembly coding for its payload.

• Initial infection and propagation.
  – MiniDuke spread over email, using malicious, well-crafted PDF files. The PDF files contained code which triggered a vulnerability in Adobe 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 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.
• 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.
  – 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.
## ADVANCED PERSISTENT THREATS COMPARISON

<table>
<thead>
<tr>
<th></th>
<th>Stuxnet</th>
<th>Duqu</th>
<th>Flame</th>
<th>Red October</th>
<th>Mini Duke</th>
</tr>
</thead>
<tbody>
<tr>
<td>PE Type</td>
<td>DLL</td>
<td>OCX</td>
<td>EXE</td>
<td>EXE</td>
<td></td>
</tr>
<tr>
<td>Initial infection</td>
<td>Unknown</td>
<td>MS Word</td>
<td>Unknown</td>
<td>MS Excel / Word, Java</td>
<td>PDF</td>
</tr>
<tr>
<td>Replication</td>
<td>Removable drives, network</td>
<td>Manual replication only</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rootkit module</td>
<td>Yes</td>
<td></td>
<td></td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Key logging</td>
<td>No</td>
<td></td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Evasion</td>
<td></td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Encryption</td>
<td>XOR</td>
<td>XOR, AES-CBC</td>
<td>XOR, RC4, Substitution</td>
<td>XOR</td>
<td></td>
</tr>
<tr>
<td>Target</td>
<td>Sabotage</td>
<td></td>
<td>Information gathering</td>
<td>Unique per victim, XOR, ROL</td>
<td></td>
</tr>
</tbody>
</table>

23
COMMON MALWARE CHARACTERISTICS

• Targeted operating system and architecture.
  – All samples were targeting 32–bit versions of Windows
  – Consider that the main reason that the malware targeted 32–bit systems was, that the majority of the victims were using this architecture.
COMMON MALWARE CHARACTERISTICS

• 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.
  – Infection through removable drives or spearfishing attacks, is the most likely scenario.
COMMON MALWARE CHARACTERISTICS

• Command execution and escalation of privileges.
  – All malware made use of exploits for command execution or privileged escalation
  – Exploitation of zero day vulnerabilities against the Operating System is probably the most challenging problem to address.
  – Even when security patches addressing these vulnerabilities had been released
COMMON MALWARE CHARACTERISTICS

• Network Access.
  – All malware communicated over ports 80/TCP, 443/TCP or 22/TCP
  – 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
COMMON MALWARE CHARACTERISTICS

- Network IDS and endpoint antivirus products.
  - Stuxnet, Flame, Duqu and MiniDuke were designed to detect and evade Antivirus Software, using multiple techniques
  - Intrusion Detection Systems (IDS). As a result, the level of protection offered by both Antivirus and NIPS products against advanced threats has received strong criticism, they failed to identify such threats – even when they had been active for several months/years.
COUNTERMEASURES

• 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.
COUNTERMEASURES

• 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).
COUNTERMEASURES

• 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 tempts to C&C servers, unless attackers were able to exploit any whitelisted services/systems and set their C&C infrastructure there.
COUNTERMEASURES

• **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.
• 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.
Conclusion

• 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.
Q & A
Thank You!