Identifying Vulnerabilities in APT Attacks: A Simulated Approach

Mathew Nicho
College of Technological Innovation
Zayed University
P.O. Box 19282, Dubai, UAE
mathew.nicho@zu.ac.ae

Adelaiye Oluwasegun
Department of Computer Science
Bingham University
Karu, Nigeria
ishayaadelaiye@gmail.com

Faouzi Kamoun
School of Engineering
ESPRIT University
Tunis, Tunisia
faouzi.kammoun@esprit.tn

***Abstract*—This research aims to identify some vulnerabilities of advanced persistent threat (APT) attacks using multiple simulated attacks in a virtualized environment. Our experimental study shows that while updating the antivirus software and the operating system with the latest patches may help in mitigating APTs, APT threat vectors could still infiltrate the strongest defenses. Accordingly, we highlight some critical areas of security concern that need to be addressed.**

***Keywords*—Advanced persistent threats; cyber-attacks, spear phishing; vulnerabilities; mitigation**

I. Introduction

The term 'advanced persistent threat' (APT) refers to an evolving breed of insidious and targeted threats that use multiple attack techniques and vectors and that are conducted by stealth to avoid detection, so that hackers can gain and retain control over the target systems, unnoticed and for prolonged periods of time [1]. Fueled by a growing online networked community, this targeted threat has been drawing increased attention from security practitioners.

Although APT attacks have been associated with 'zero-day exploits' where systems and application vulnerabilities are not yet known, a recent study [2] has shown that only 19% of publicized APT cases involved zero-day vulnerabilities (81% employ readily available malware generation tools), with 70% of the cases involving public vulnerabilities and 11% associated with unknown vulnerabilities.

APTs pose a serious challenge for current anti-virus products and IDS because anomaly-based detection methods depend on predefined attack signatures, while APTs usually exploit unknown as well as known security holes to evade most defense mechanisms. Most of the earlier contributions concerning APTs were based on comprehensive studies of earlier APT attacks to (1) identify the characteristics of these attacks; (2) propose the APT attack stage model; and (3) highlight some generic countermeasures.

This contribution aims to assess the effectiveness of some commonly deployed security mechanisms in thwarting APT attacks. To achieve this goal, we conducted multiple simulated APT attacks in a virtual environment in an attempt to (1) identify potential vulnerabilities against APT attacks, and (2) assess the effectiveness of some security mechanisms in blocking these attacks.

The remainder of this paper is structured as follows: Section 2 highlights the main APT threat vectors. Section 3 outlines the associated experimental setup, along with the attack simulation results. In Section 4, we discuss these results and their practical implications. Finally, in Section 5 we provide a summary of the main findings of the paper.

II. APT Overview

Among the salient features of an APT attack are its ability to: (1) target specific and high-profile organizations and governments; (2) pursue its objectives persistently over an extended period of time; (3) adapt itself to defenders' efforts to resist it; and (4) maintain the level of interaction needed to execute its malicious mission. Simply put, we can define an APT as the persistent attempt by a group of highly organized and skilled attackers to exploit specific targeted organizations using stealthy, adaptive and evasive attack techniques.

An analysis of 89 known APT cases [2] revealed that spear-phishing email attacks and watering hole attacks are among the top APT threat vectors. Other common threat vectors include USB drive virus attacks.

Spear-phishing involves the creation of fraudulent, disguised and trusted emails with customized information targeted at the recipient.

Watering hole attacks are typical examples of indirect delivery mechanisms where a legitimate website that is frequently accessed is compromised to inject malware.

A USB drive is a practical and a common method of manually transferring data among computing devices. The usage of USB devices to transfer data is often considered to be safe. However, a USB has no mechanism for detecting a breach or even properly identifying a spoofed device, thus making it a powerful medium for spreading malware [3].

III. Experimental Setup and Simulated Results

To assess the effectiveness of some common counter-measure techniques in thwarting some selected APT attack vectors, we set up a virtualized experimental environment. We simulated three types of APT attack vectors, namely malicious USB attacks, backdoor attacks and 'drive by download' spear-phishing attacks. In our experiments, we selected three types of exploits per APT attack vector, giving rise to nine APT attack scenarios, as illustrated in Table II. Our choice of APT vulnerabilities is also aligned with the observation of Shanks [4] that the most common problems faced by SMEs come from staff exposing IT systems to malware through opening infected emails, using unsafe websites, or by plugging in external USB sticks.

For each APT threat vector, we considered four countermeasures, namely anti-virus, sandbox, firewall and browser setting approaches. For the selection of the anti-virus software, we opted for a random selection of three popular antivirus solutions from the top ten best antivirus list as reported in [5]. We created virtual machines on a single physical server to simulate the various attack scenarios, including the target system and the attacker machines. The details of our experimental setup are shown in Table I.

TABLE I. EXPERIMENTAL SETUP DETAILS

Entity	Specification
Physical server	*Apple MacBook Pro*
Host OS	*Mac OX El Capitan version 10.11.6*
Virtualization S/W	*VMWare Fusion version 7.1.3*
Guest VM OS	*Windows XP Professional* *Windows 7 Enterprise* *Kali-Linux Debian GNU version 1.0.9*
Windows Firewall	*Windows Defender version 6.1.7600.16385*
USB flash	*Silicon Motion flash drive*
Antivirus software	*Avast version 10.3.2223* *Kaspersky version 15.0.2.360* *AVG version 2016.0.7442*
Sandbox tools	*Sandboxie version 5.06* *Avast Sandbox version 10.3.2223*
Web browser	*Internet Explorer 9 version 9.0.8112.16421*

The results of our experiments are summarized in Table II and are further discussed below.

TABLE II. EFFECTIVENESS OF APT COUNTERMEASURES[1]

APT threat vector	Mitigation/countermeasures			
	Anti-virus	**Sandbox**	**Firewall configuration**	**Browser settings**
USB malware threat				
Veil-Evasion	**X**[1]	✓	N/A	N/A
Metasploit	**X**	✓	N/A	N/A
Shelter	✓	✓	N/A	N/A
Backdoor threat				
SET	✓	**X**	✓	N/A
Backdoor Factory	✓	✓	✓	N/A
Msfvenom	✓	**X**	✓	N/A
Drive-by download threat				
Armitage/BeEF XSS	✓	N/A	✓	**X**
SET	✓	N/A	✓	✓
Websploit	✓	N/A	✓	✓

[1] **X:** Undetected and not prevented, **√:** Detected and prevented, N/A: Not applicable

A. USB Malware Attacks

Hackers have used infected USB devices to lure unsuspicious users into plugging them into their machines. For instance, in some reported cases of data breach [6], hackers have been reported to have deliberately placed USB devices near parked cars belonging to potential targets. USB devices often bypass security infrastructure, as most organizations do not view USB devices as real threat agents [7]. We created malware for USB using three exploits, namely Veil-Evasion, Metasploit and Shellter.

Veil-Evasion: We simulated this attack scenario as follows: First, we initiated the Veil-Evasion tool. Following this, the 32nd shell code (python/shellcode_inject/aes_encrypt) was selected from the list of 46 available codes. Next, we used Veil-Evasion to generate the payload (named 'Update') and converted it into a Windows-executable file using Pyinstaller. The generated payload located in the directory /usr/share/veil-output/compiled/Update.exec was then copied to a USB flash drive. Upon inserting the flash drive and copying the payload into the target machine, the Avast anti-virus software (with up-to-date virus definitions) did not report the presence of any malicious activity. The malware was identified as unknown by the User Account Control (UAC) of the windows machine. On scanning the flash drive content with Avast (with up-to-date virus definitions), no threats were reported.

Upon running update.exe in the sandbox (Sandboxie), the Avast anti-virus software detected the malicious payload and generated an error. We found that the Avast anti-virus runs an analyzer in the background to check the program running in the sandbox for malicious activities, which revealed that the program could not be trusted.

Metasploit: We conducted the second simulation of USB malware attack using the Metasploit msfvenom malware generator that is pre-installed in Kali-Linux. Using reverse TCP connection, we deployed the payload into the target machine using a USB device. Upon installing the Metasploit malware onto the target machine, we observed that neither the AVG 2016 anti-virus nor the Windows firewall was able to detect the malicious application. While running the malicious payload in Sandboxie, we found that the sandbox prevented the executable payload from running.

Shellter: This malicious program runs on Linux, but unlike Metasploit it does not come pre-installed on Kali Linux. However, it can be downloaded from the Internet and then run from the CLI. Shellter does not generate a payload, but modifies an existing application by injecting it with a malicious code. We deployed the infected application onto the target machine using a USB flash drive. Upon deploying the payload, the malware was intercepted and deleted by the latest version of Kaspersky (with up-to-date virus definition) anti-virus. On running the infected application on Sandboxie, the Sandbox generated an error stating that the program has stopped working and that it prevented the malicious application from running.

B. Backdoor Attacks

We simulated three types of APT backdoor attacks, namely SE Toolkit, Backdoor Factory and msfvenom.

SE Toolkit: This is available and preinstalled in the Kali Linux toolkit running on Debian. We activated the SE Toolkit using a Windows shell in reverse TCP. The generated payload was then copied to the target machine where it was opened.

Upon deploying the malware onto the target machine, the antivirus alerted the user to the presence of a malicious program and the malware was moved to the chest. We redeployed the malicious payload onto the target machine after disabling the anti-virus, and allowed it to run in the Avast sandbox. Upon running, the payload successfully established a connection back to the hacker's machine. Several commands were successfully issued to view files and change directories. However, the command to remotely shutdown the target machine was denied. An outbound rule was created in Windows Firewall to block all outgoing traffic apart from the default rules that had already been set to allow some traffic. On activating this rule, the connection between the target and the hacker's machine was dropped, the session was closed, and a host unreachable error was generated.

Backdoor Factory: The Backdoor Factory tool can be downloaded from Github, as it does not come pre-installed on Kali Linux. Once the Backdoor Factory malware was opened, we attached the shellcode to the win32 file and deployed it onto the target machine. Thereafter, we used the CLI to initiate the msfconsole in order to enable the listener.

Upon deploying the infected win32 file, Kaspersky anti-virus (with up-to-date virus definitions) was able to intercept the malicious payload and delete it. We then ran the same infected file on Sandboxie. The Sandboxie application successfully prevented the malware from running in the presumed secure and monitored environment. When running the malicious application with regular inbound filtering of packets, the exploit runs with no obstruction. However, when activating outbound filtering on traffic originating from a host within the network, the application was able to run but the attacker was unable to establish a remote connection to the target machine.

Msfvenom: Msfvenom is a combination of msfpayload and msfencode, and comes pre-installed in Kali Linux. To simulate an APT backdoor attack using msfvenom, we generated the payload and injected it into a Windows executable file with a reverse TCP connection. We then deployed and activated the infected application on the target machine. Upon deploying the infected Windows executable file, AVG anti-virus (with up-to-date virus definitions) was able to intercept the malicious payload and delete it. When we subsequently ran the malicious executable file in the sandbox using Sandboxie, the sandbox tool was not able to detect the application as containing malicious code, and thus did not prevent the malicious software from running. However, when we activated outbound traffic filtering, we found that the attacker was not able to establish a backdoor connection to the target machine.

C. Drive-by Download Spear-phishing Attacks

Drive-by download attacks exploit loopholes in web browsers by exploiting hidden iframes (inline frames) to inject malicious code into vulnerable yet genuine sites. The hidden iframe redirects the browser to another page created by the attacker, which installs the malware onto the victim machine without the user's knowledge. We simulated three types of drive-by-download APT attacks using Armitage integrated with the BeEF XSS, SE Toolkit, and Websploit tools.

Armitage/BeEF XSS attack: Once the BeEF (penetration testing tool) XSS scripts were loaded and initialized, an API key was used to hook the infected URL. We simulated a user visiting the infected webpage to infect the target machine, whereupon we used a meterpreter to create web sessions. Upon activating the Avast antivirus on the victim machine, the BeEF XSS threat was detected by the antivirus and it was blocked. When we disabled the anti-virus and ran the infected webpage on Internet Explorer (with the default security configuration), we observed that the attack was successful. We then changed the security configuration of the web browser to reflect the strictest level. We observed that the exploit was still undetected and ran successfully. We then tested the effectiveness of the firewall in thwarting the attack by creating an outbound rule that blocked outbound traffic on the target machine and observed that the connection to the target machine was successfully blocked.

SET (Social-Engineer Toolkit): We used SET to simulate a drive-by download attack known as the Java applet attack. After initiating SET, we chose the Java applet attack from the 'website attack vectors' option. Thereafter, we used the web templates to deploy the payload with NAT and port forwarding. Generating a self-signed certificate, the Facebook option was selected, with a backdoor executable as the payload. We simulated the user behavior of visiting a fake website to infect the target machine with the malware.

Upon the target machine accessing the infected site, the up-to-date Kaspersky anti-virus software detected the malware. It prevented the site from opening and generated an error message. We then tested the effectiveness of the firewall in thwarting the attack by creating an outbound filtering rule that blocked outbound traffic on the target machine and observed that the connection to the target machine was successfully blocked. Finally, we observed that the attack was successful when the browser configuration was set to its default settings. However, after we changed the security configuration of the web browser to reflect the strictest security level, we observed that the Java applet attack was blocked.

Websploit: On initiating Websploit, we selected the Java applet attack under the exploit module. We then issued the 'run' command on the command line interface to release the infected website. We tested the Java applet attack using up-to-date and fully functional AVG anti-virus software. Upon trying to access the website, the target machine generated an alert highlighting an attempt by malware to gain access to the target machine. We then tested the effectiveness of the firewall in thwarting the attack by creating an outbound filtering rule that blocked outbound traffic on the target machine and observed that the connection to the target machine was successfully blocked. It should also be noted that the regular inbound filtering rule alone was not sufficient to block the attack. Finally, we observed that after disabling the anti-virus software, the attack was successful when the browser configuration was set to its default settings. However, when we changed the security configuration of the web browser to reflect the strictest security level, we observed that the Java applet attack was blocked.

IV. Discussion and Implications

A. *USB Malware Infection*

As illustrated in Table II, antivirus software solutions were unable to detect two of the three USB malware threat vectors tested (namely Veil-Evasion and Metasploit). This can be explained by the fact that these malware vectors use encryption to change form and bypass antivirus software that relies on detecting known signatures. For instance, Veil-Evasion uses AES encryption to evade detection. There are at least three implications of this finding: (1) one should not rely solely on antivirus software to detect and prevent USB malware attacks; (2) antivirus software vendors need to find other means (besides malware signatures) of detecting and preventing encrypted USB malware; and (3) network administrators can block USB drives or configure network management systems (NMS) to generate a trap when a user inserts a USB device into any device connected to the corporate network.

B. *Backdoor Malware Attacks*

As indicated in Table II, up-to-date antivirus solutions were able to detect and block all three simulated backdoor vector attacks. When the firewall was used with its default configuration settings (inbound filtering only), we observed that it failed to detect the backdoor attacks. This issue was remedied when we enabled outbound filtering, which blocked outbound traffic on the target machine. Contrary to the common belief that a sandbox runs applications in a controlled and secure environment, our study revealed that it failed to prevent two of the three simulated backdoor attacks, namely those initiated by the SET toolkit and msfvenom.

There are at least two implications of the above findings: (1) it is important to change the default configuration of the firewall to enable outbound filtering; and (2) running suspicious applications in a sandbox may not be an effective countermeasure for thwarting some backdoor attacks.

C. *Drive-by Download Attacks*

As shown in Table II, each of the four security countermeasures was successful in detecting and preventing the three types of drive-by download vector attacks, with the exception of the browser settings mitigation strategy, which failed to thwart the Armitage/BeEF XSS malware. Our results show that the latest update of Java comes with a security feature that blocks Java applet attacks. Most antivirus software solutions come with plugins for web browsers that can be used to protect against known attacks that try to infiltrate host machines through web browsers. Our experimental results thus highlight the need to change the default firewall configuration settings (inbound filtering only) to enable outbound filtering as well.

V. Conclusion

This research adds to the existing body of knowledge on the adequacy of current mitigation strategies in detecting and thwarting common APT attacks through multiple simulated attacks within a virtualized environment.

We identify five cases where specific countermeasures failed to detect and prevent the APT attack, and discuss the implication of these failures for practitioners.

As with most experimental studies, this contribution has its own limitations, which include the use of a limited number of APT threat vectors and countermeasures, a reliance on a virtualized environment and a lack of use of genuine 'zero day' malware. An extension to this research may involve using a K-means machine learning clustering algorithm for anomaly detection to identify the lateral communications of malware via Apache Spark.

Acknowledgment: This paper has been accepted as a short paper.

References

[1] Tankard, C, "Persistent threats and how to monitor and deter them," Network Security, August 2011, pp. 16-19.

[2] Li, M, Huang, W, Wang, Y, Fan, W, and J. Li, "The study of APT attack stage model", . IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS), 2016.

[3] Jover, R. P, and P. Giura, "How vulnerabilities in wireless networks can enable advanced persistent threats,", International Journal on Information Technology, vol. 1, no. 2, 2013, pp. 145-151.

[4] Shanks, G, Cyber "myths" Putting UK SMEs at Risk, Retrieved from http://www.michelmores.com/news-views/news/cyber-myths-putting-uk-smes-risk

[5] Williams, M, The Best Antivirus Software 2016. Retrieved from http://www.techradar.com/news/top-10-best-antivirus-software-for-2016

[6] Institute for Critical Infrastructure Technology, Hacking Healthcare IT in 2016. Retrieved from http://icitech.org/wp-content/uploads/2016/01/ICIT-Brief-Hacking-Healthcare-IT-in-2016.pdf

[7] Moon, D, Im, H, Lee, J. D, and J.H. Park, "MLDS: multi-layer defense system for preventing advanced persistent threats," Symmetry, vol. 6, no. 4, 2014, pp. 997-1010.