Understanding Advanced Persistent Threats

A Case Study

NED MORAN

Ned Moran has worked in cyber threat intelligence analysis for nearly a decade and a half. He currently serves as the Director of Technical Research at a cyber risk management company. He has also served as an Adjunct Professor of Information Privacy and Security at Georgetown University since 2008. He has been an invited speaker at NATO's Cooperative Cyber Defense Center of Excellence's Conference on Cyber Warfare in Tallinn, Estonia, and at the RSA Conference in San Francisco. He is frequently interviewed by the media concerning the intersection of terrorism and technology.
ned.moran@gmail.com

APT, short for Advanced Persistent Threat, is a commonly used and controversial term bandied about the IT security sector. Many feel that this term is abused and simply used to describe attacks that network defenders failed to prevent—no matter the sophistication of the attack. This article seeks to establish a working definition for APT and to highlight that the sophisticated nature of these attacks lies not within the technology used but, rather, the logistical organization of the adversary. This article will offer an in-depth examination of an APT-style attack as a means of highlighting the operational efficiency of the adversary.

What Is APT?

It is first necessary to establish an accepted definition of APT. Richard Bejtlich, the Chief Security Officer at Mandiant and long-term observer of APT-style intrusions, defines APT as follows [1]:

> **Advanced** means the adversary can operate in the full spectrum of computer intrusion. They can use the most pedestrian publicly available exploit against a well-known vulnerability, or they can elevate their game to research new vulnerabilities and develop custom exploits, depending on the target's posture.
>
> **Persistent** means the adversary is formally tasked to accomplish a mission. They are not opportunistic intruders. Like an intelligence unit, they receive directives and work to satisfy their masters. Persistent does not necessarily mean they need to constantly execute malicious code on victim computers. Rather, they maintain the level of interaction needed to execute their objectives.
>
> **Threat** means the adversary is not a piece of mindless code. This point is crucial. Some people throw around the term "threat" with reference to malware. If malware had no human attached to it (someone to control the victim, read the stolen data, etc.), then most malware would be of little concern (as long as it didn't degrade or deny data). Rather, the adversary here is a threat because it is organized and funded and motivated. Some people speak of multiple "groups" consisting of dedicated "crews" with various missions.

This definition of APT is extremely useful, because it does not focus on the technical sophistication of the adversary or the elegance of the attack code used. Rather, it focuses on the organizational capabilities and intentions of the adversary.

Misunderstanding the A in APT

Thinking about an APT in light of the adversary organization's capabilities and intentions highlights that the "advanced" in APT is less about code and more about techniques, tactics, and procedures. Just as the power of the US military lies largely in its ability to organize and secure a logistics supply chain to its frontline troops, APT actors are able to provide robust support to the frontline intrusion operators—the guys at the keyboard.

This type of organizational structure and efficiency is what truly defines APT. True APT actors are part of a robust organizational infrastructure driven by specific collection requirements that focus these actors onto a set of targets. Further, the ongoing and strategic nature of these collection requirements forces APT actors to develop tools and tradecraft that enable them to fully exploit new collection opportunities—e.g., an infected drone in a targeted organization.

APT actors support their front-line intrusion operators with tools that notify the operators of new infections as well as tools that enable the operators to quickly leverage these initial footholds into a deeper and more resilient presence inside the targeted organization.

What Motivates APT Actors?

APT actors are primarily interested in maintaining reliable access to their targets. The desire for access to sensitive intellectual property drives this need for persistent and reliable access. APT actors are not interested in quick smash and grab attacks. Instead, they want to quietly get inside a target environment and set up a number of redundant listening posts, so that if any one infection is detected the other infections will still provide the adversary with the required access into the targeted organization.

Persistence, the P in APT, also means that the adversary keeps coming back. They will consistently attack the same target over and over again in an effort to maintain a secure foothold within a targeted organization. As infections are discovered and remediated by the victim organization, the adversary will launch a new salvo designed to regain a foothold. The purpose of this foothold is to enable the type of ongoing monitoring required to deliver a complete picture of an organization's internal communications and access to intellectual property.

A Closer Look at an APT-Style Attack

On April 12, 2011, a spear phishing email sent to specific targets was observed in the wild. This observed spear phishing attack provides a good example of an APT-style attack. The targets of this spear phish were specifically chosen by the attackers because the attackers perceived that these targets held information of value.

The spear phishing email contained a Microsoft Word document attachment. This Word document was crafted to appear as a legitimate document and contain specific subject matter of interest to the targeted victim. This type of social engineering is a common tactic in an APT style attack. The adversary conducts detailed pre-attack reconnaissance in an effort to better understand the victims. This reconnaissance enables the adversary to design spear phishing lures that entice the victim into opening the malicious attachment.

Embedded within this document was a malicious Flash file designed to exploit the recently announced Adobe zero-day CVE-2011-0611.

The exploit first makes use of a heap spray to fill memory with 0x11111111 and then loads a second SWF file. It is this secondary SWF that actually triggers the vulnerability. The SWF file makes use of several common obfuscation techniques. The code attempts to confuse disassemblers by setting the size of a group of constants to 0x15 when there really are 0x14 present, causing disassembly to be misaligned with the actual code.

In addition, it does several things which are also fairly usual. For example, streams of instructions which are effectively dead code, conditional branches which can never be taken, jumping around unnecessarily, and blocks of instructions which have no effect on the program itself. All of this isn't really a factor in the exploit itself but is simply obfuscation.

The shellcode executed by this exploit drops a malicious payload with the following properties:

```
File: scvhost.exe
Size: 22016
MD5: 4EC6D3A6B5A5B67D4AB5F04C41BFB752
```

Scvhost.exe is installed and launched with the filename msdtc.exe.

The msdtc.exe payload initiated traffic over port 80 with a command and control server at msejake.7766.org. 7766.org is a dynamic DNS provider that allows domain administrators to quickly and easily point their domain to any IP under their control. During the observed attack the domain resolved to 125.46.42.221. Infected victims were observed sending base64 encoded messages to the control server at msejake.7766.org. Observed traffic was as follows:

```
bG9nb258U1lTVEVNLTEyMzQ1Njc4OXxXaW5kb3dzIFhQfDEwMDcwN3
wzY2I4ZGM5MjI4N2UzZmJmMTA0MmQ2NTRlYzRkY2RhMnw=
YWN0aXZlfA==
```

This traffic decodes to:

```
logon|SYSTEM-DDBLV6BQXN|Windows XP|100707|3cb8dc92287e3fbf1042
d654ec4dcda2|
active|
```

This traffic appears to serve an initial reconnaissance function whereby the infected machine reports back to the control server basic system information including machine name, operating system, and state.

Static analysis of the msdtc.exe reveals a number of strings of interest. These strings include but are not limited to:

```
shell|
filelist|
upload|
```

These are likely commands the trojan is designed to recognize and act upon. "Shell" is likely a command that can be issued by a remote hostile actor to open a shell on the compromised machine. "Filelist" is likely a command that can be issued by a remote hostile actor to enumerate a list of files within a given directory. Finally, "upload" is likely a command that can be issued by a remote hostile actor to exfiltrate information from the victimized machine. For a full list of commands identified via static analysis of the msdtc.exe payload, please see the Appendix.

It is likely that this trojan was designed to enable a remote operator to fully reconnoiter a victim's machine, search for and acquire deeper access within the targeted network, and exfiltrate sensitive information.

In laboratory testing, the intrusion operator was observed establishing active sessions on an infected machine. As noted above, the infected machine beaconed an "active" message to the command and control server—likely informing the controller that the infected machine was available for exploitation. Within 49 minutes of initial exploitation, an intrusion operator initiated multiple sessions on the infected machine. An analysis of the sessions indicates that a live person at a keyboard (as opposed to a scripted service or program) initiated the connection to the back-doored computer.

The malicious operator maintained three different sessions on the infected machine. One served as a command and control session. A second session was used to gather intelligence about the networking infrastructure surrounding the victim. A third session was used to enumerate the hard disk likely in search of sensitive information.

The following table illustrates the commands passed by the operator during these sessions. The first command of "SHELL |" was likely issued to open the back door on the infected computer. The table below shows the number of seconds elapsed between the issued commands and the first "SHELL |" command.

Commands in **bold** were part of a command and control session, and those in *italics* were from a session established to reconnoiter the network of the infected victim. APT actors typically establish a beachhead on the infected machine and then immediately look for additional opportunities to establish additional footholds deep within the penetrated network.

Commands in regular type were part of a session established to scan the hard drive of the infected victim. APT actors also typically scan compromised hosts for sensitive intellectual property.

TIME (in seconds)	**COMMAND**
0	**SHELL |**
2	**SHELL START|**
7	*COMMAND|NET USER*
13	*COMMAND|IP CONFIG /ALL*
62	*COMMAND|NET VIEW/DOMAIN*
70	FILES|
72	FILELIST|C:\*.*
74	FILELIST|C:\DOCUMENTS AND SETTINGS*.*

78	FILELIST|C:\DOCUMENTS AND SETTINGS\ADMINISTRATOR\*.*
70	FILES|
72	FILELIST|C:\*.*
91	*STOP|*
97	FILELIST|C:\DOCUMENTS AND SETTINGS\ADMINISTRATOR\MY DOCUMENTS*.*
101	FILELIST|C:\DOCUMENTS AND SETTINGS\ADMINISTRATOR\DESKTOP\*.*
125	FILELIST|C:\DOCUMENTS AND SETTINGS\ADMINISTRATOR\DESKTOP\SERVICE PACKS*.*
140	STOP|
144	**UNINSTALL**

Figure 1: Decoded commands issued by intrusion operator

These commands were likely issued by the remote attacker via a management dashboard that enables the attacker to select a number of pre-configured scripted actions. Every command issued by the attacker was base64 encoded. The time delays between these commands can be explained by a human operator evaluating the responses from the infected victim prior to deciding which commands to issue next.

This is the type of organizational sophistication that defines the A in APT. The particular trojan used in this attack was not in and of itself sophisticated. No rootkits were used, and a knowledgeable sysadmin likely would have detected the infection via host or network-based analysis.

However, the efficiency of the adversary, as demonstrated by their actions during this observed attack, highlights what the "advanced" in APT is all about. First, the attack was targeted, indicating that the adversary was only interested in penetrating a set of victims they were interested in monitoring.

Second, the speed with which a human operator established a session on the infected host demonstrates the advanced logistics capabilities of the adversary. The adversary established a session within 49 minutes of the initial infection and then proceeded to iterate the infected machine, presumably in search of sensitive information or for connections deeper within the targeted network. This demonstrates that the adversary had "trained" operators on standby ready to exploit newly compromised computers. This type of operation requires resources that only a determined adversary would support.

Third, the operator's decision to quietly uninstall the trojan from the infected machine after two minutes and 22 seconds of reconnaissance indicates that the operator knew exactly what he or she was looking for. This demonstrates that this type of APT attack was driven by a set of requirements, and when the operator of this intrusion was unable to locate data meeting those requirements, he or she decided to quickly retreat and exploit another newly infected computer.

Conclusion

The above example demonstrates that the threat from APT actors is not their embrace of novel exploit code or sophisticated attack tools, but, rather, their organizational efficiency. These actors patiently study their targets, craft enticing spear phishing attacks, and quickly exploit new victims. They will continually seek to maintain secure footholds within their targets, as this is the most efficient way for them to routinely exfiltrate the sensitive data they are tasked with acquiring.

Reference

[1] http://taosecurity.blogspot.com/2010/01/what-is-apt-and-what-does-it-want.html.

Appendix: Strings Pulled from msdtc.exe

```
CmD
shelldata|
shell|
shellstart
command
stop
upload|
upload
files|
driver
driver|
filelist
listerror|
filelist|
delete
run
renamefile
stop
down|
dirlist
direrror|
dirlist|
logon|
shell
active
files
upload
closeos
restart
down
dclose
uclose
uninstall
active|
```