SANS Technology Institute

Assessing Outbound Traffic to Uncover Advanced Persistent Threat

Joint Written Project

Beth E. Binde, Russ McRee, Terrence J. O'Connor
5/22/2011

Table of Contents

Table of Figures

Executive Summary

Advanced Persistent Threat (APT) exhibits discernible attributes or patterns that can be monitored by readily available, open source tools. Tools such as OSSEC, Snort, Splunk, Sguil, and Squert may allow early detection of APT behavior. The assumption is that attackers are regularly attempting to compromise enterprises, from basic service abuse to concerted, stealthy attempts to exfiltrate critical and high value data. However, it is vital to practice heightened operational awareness around critical data and assets, for example, card holder data, source code, and trade secrets. Segment and wrap critical data within the deeper protection of well monitored infrastructure (defense in depth). Small, incremental efforts, targeted at protecting high value data value (typically through smaller and protected network segments), provide far greater gains than broader, less focused efforts on lower value targets. In a similar vein, layered defensive tactics (multiple layers and means of defense) can prevent security breaches and, in addition, buy an organization time to detect and respond to an attack, reducing the consequences of a breach.

Even the best monitoring mindset and methodology may not guarantee discovery of the actual APT attack code. Instead, the power of more comprehensive analysis and correlation can discover behavior indicative of APT-related attacks and data exfiltration. APT examples provided herein include Operation Aurora and the recent attack on RSA. Additional opportunities for discussion include the LizaMoon attacks (SQL injection as an entry vector) as well as analysis specific to how FastFlux traffic might be indicative of deeper malfeasance. These additional considerations are important as they may serve as an APT entry point, or indicate its presence. The defining premise of this paper will be to:

- define and describe advanced persistent threat (APT)
- propose technical approaches to mitigating the threat
- include tools useful in the possible detection of APT

Introduction

In 2006, the United States Air Force (USAF) analysts coined the term **advanced persistent threat** (APT) to facilitate discussion of intrusion activities with their uncleared civilian counterparts. Thus, the military teams could discuss the attack characteristics yet without revealing classified identities. [Bejtlich, 2007]

Bejtlich explains the components of the terminology.

- **Advanced** means the adversary is conversant with computer intrusion tools and techniques and is capable of developing custom exploits.
- **Persistent** means the adversary intends to accomplish a mission. They receive directives and work towards specific goals.
- **Threat** means the adversary is organized, funded and motivated.

Further, objectives may be political, economic (for example, the theft of intellectual property), technical or military (identification of weaknesses). [Bejtlich, 2010]

The Anatomy of an Advanced Persistent Threat [Cutler, 2010] describes the typical APT strategy.

- Attacker gains foothold on victim system via social engineering and malware
- Attacker then opens a shell prompt on victim system to discover if system is mapped to a network drive
- Victim system is connected to the network drive prompting attacker to initiate a port scan from victim system
- Attacker will thereby identify available ports, running services on other systems, and identify network segments
- Network map now in hand attacker moves to targeting VIP victims with high value assets at their disposal

Operation Aurora

Operation Aurora demonstrates several of the key components of an advanced persistent threat (APT) based attack. [McAfee, 2010] This cyber-attack against several companies in the technology, security and defense industries started in mid-2009 and continued through December 2009. [Operation Aurora, 2011] Understanding that the intent of an APT is to gain access to targeted information and maintain a foothold in the environment for future use and control, Aurora is an excellent example to examine. [Daly, 2009] In the case of Aurora, attackers targeted the software-configuration management (SCM) systems that held proprietary information of Google, Adobe and other Fortune 100 companies over several months. The anatomy of the attack categorizes it as a classic APT attack. See Figure 1.

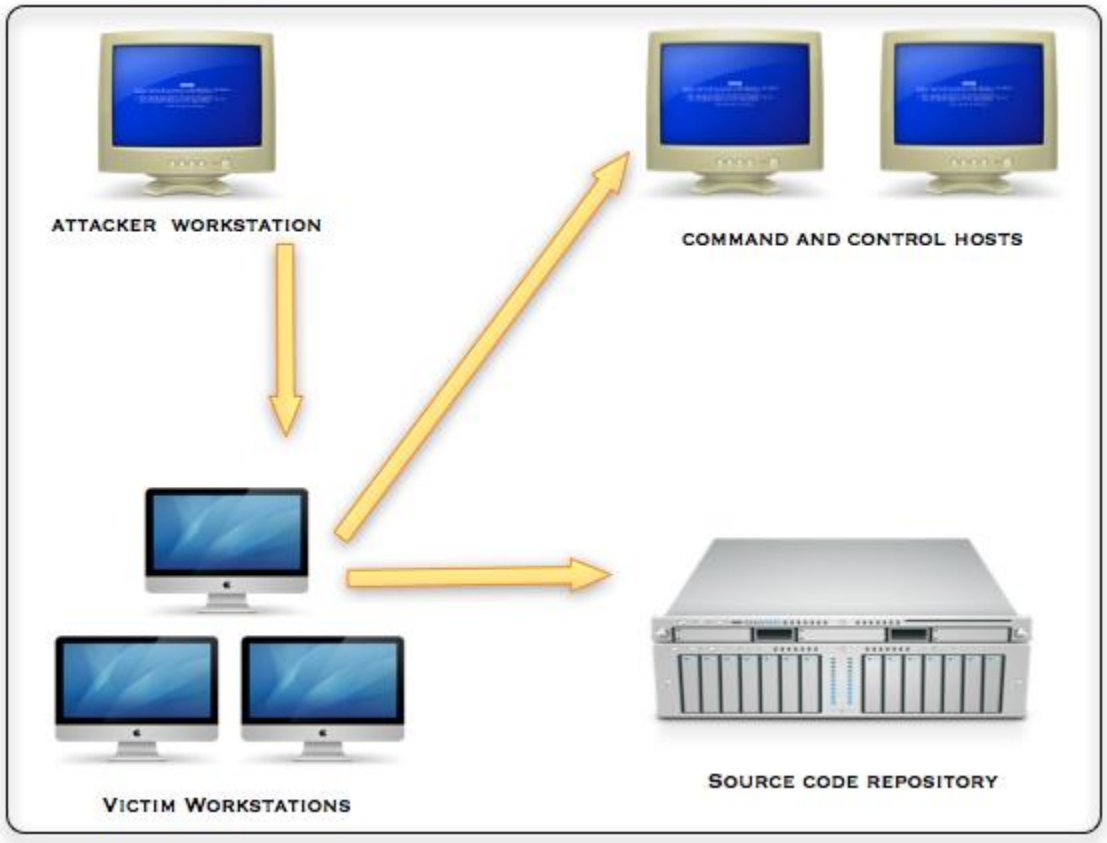


Figure 1: Anatomy of the Operation Aurora Attack

Advanced

In order to gain initial access to the victim's networks, the attackers started with a targeted spear phishing attack against the company. Several employees of the victim companies received an email that appeared to be from someone they trusted. However, the email contained a link to a Taiwanese website that hosted malicious JavaScript. The malware, in turn, exploited an Internet Explorer vulnerability. The exploit triggers when Internet Explorer attempts to access memory that has been partially freed. Below is a program fragment from the Aurora attack. The reason for showing this fragment is to depict the difficulty in discovering this exploit with the lack of a signature or knowledge of the vulnerability.

```
function intiliaize()
{
	obj = new Array();
	event_obj = null;
	for (var i = 0; i < 200; i++ )
		obj[i] = document.createElement("COMMENT");
}

function ev1(evt)
{
	event_obj = document.createEventObject(evt);
	document.getElementById("sp1").innerHTML="";
	window.setInternal(ev2,1);
}

function ev2()
{
	var data,tmp;
	data = "";
	tmp = unescape("%u0a0a%u0a0a");
	for (var i = 0; i < 4; i++)
		data += tmp;
	for (i = 0; i < obj.length; i++) {
		obj[i].data = data;
	}
	event_obj.srcElement;
}
```

Program 1: Program Fragment from Aurora Attack

Although previously known at Microsoft, the vulnerability was unseen "in the wild"---in other words, there was no indication that anyone else knew of the vulnerability--- and no mechanism was in place to detect the attack. Once exploited, the victim system(s) connected to series of

command and control servers using TCP port 443, commonly associated with encrypted traffic and thus difficult to inspect at the network perimeter. The ability to discover novel vulnerabilities and use them in a targeted attack characterizes this classified as advanced.

Persistent

After gaining a foothold in the victim companies, the attackers employed the exploited workstations to compromise other internal resources. Use of a compromised system to attack other systems on the same network, bypassing firewalls and other restrictions, is termed *pivoting*. [Exploit (computer security), 2011] Daly characterizes pivoting as the lateral movement phase of the attack. [Daly, 2009]. In Operation Aurora, the attackers then targeted the SCM systems. The McAfee analysis of Operation Aurora points out that SCM systems are often insecure. [McAfee, 2010]. The attackers then exfiltrated source code to the attacker's command and control servers--- servers with innocuous-sounding domain names such as homelinux.org, ourhobby.com and servebeer.com. [Symantec, 2010] In January, 2010, Google was the first to publicly disclose the loss of intellectual proprietary. [Operation Aurora, 2011]. The attackers accomplished a specific mission meeting the persistence criteria of APT.

Threat

The attacks traced back to two Chinese schools, Shanghai Jiaotong University and the Lanxiang Vocational School. Jiatong hosts one of the top computer science programs in China. In 2010, it beat Stanford University in an international programming competition sponsored by IBM. Lanxiang is a large vocational school established with military support, training some computer scientists for the military. The school is operated by a company with close ties to Baidu, a strong Chinese competitor to Google. Sources within the schools denied organizational involvement in the attacks. [Markoff, 2010] The adversaries, whatever their actual identities, demonstrated high motivation, were adequately funded, and were part of a structured organization. Not only did they have the intent to gain access to sensitive information; they had the capability. [Command Five Pty Ltd, 2011]

RSA Breach

In March 2011, RSA acknowledged a successful attack against the RSA network. In an open letter addressed to customers, Art Coviella, the Executive Chairman, declared that the attack was APT. [Coviella, 2011] This attack further illustrates the key components of an APT attack and some of the methods that could be used to identify a similar attack.

Target

The RSA attack began with a successful phishing campaign. This is a common technique for introducing malware into the target network. In this campaign, the attackers targeted two small

groups of employees, delivering the phishing email over a two-day period and bypassing the installed email filters.

Often APT associated phishing attacks include a Microsoft ® Office exploit, or an exploit for Adobe ® Reader. [Mandiant Corporation, 2011] In this case, a Microsoft Excel spreadsheet contained a "zero-day" Adobe Flash exploit that successfully installed a remote access toolkit on the victim machine. [Rivner, 2011]. A "zero-day" exploit for a vulnerability is created before, or on the same day, as a vendor learns about a particular vulnerability and no patch or remediation is available yet. [Bradley]

After successful exploitation via the infected spreadsheet, the victim workstations installed a remote access toolkit (RAT) known as Poison Ivy; referred to here as PI-RAT (see Figure 2). By default, this remote administration toolkit performs a reverse connection from the client to TCP Port 3460 on the command and control server. [Poison Ivy, 2008] The attacker can search files, explore the Windows ® Registry ®, examine processes and open ports on the infected workstation. An attacker can take remote screenshots, dump the password hashes, relay traffic, and even tap the microphone. Further, the toolkit provides the ability to write third party plug-ins for specific actions on the objective. The PI-RAT toolkit has been extensively used in other attacks, including GhostNet. [Rivner, 2011]

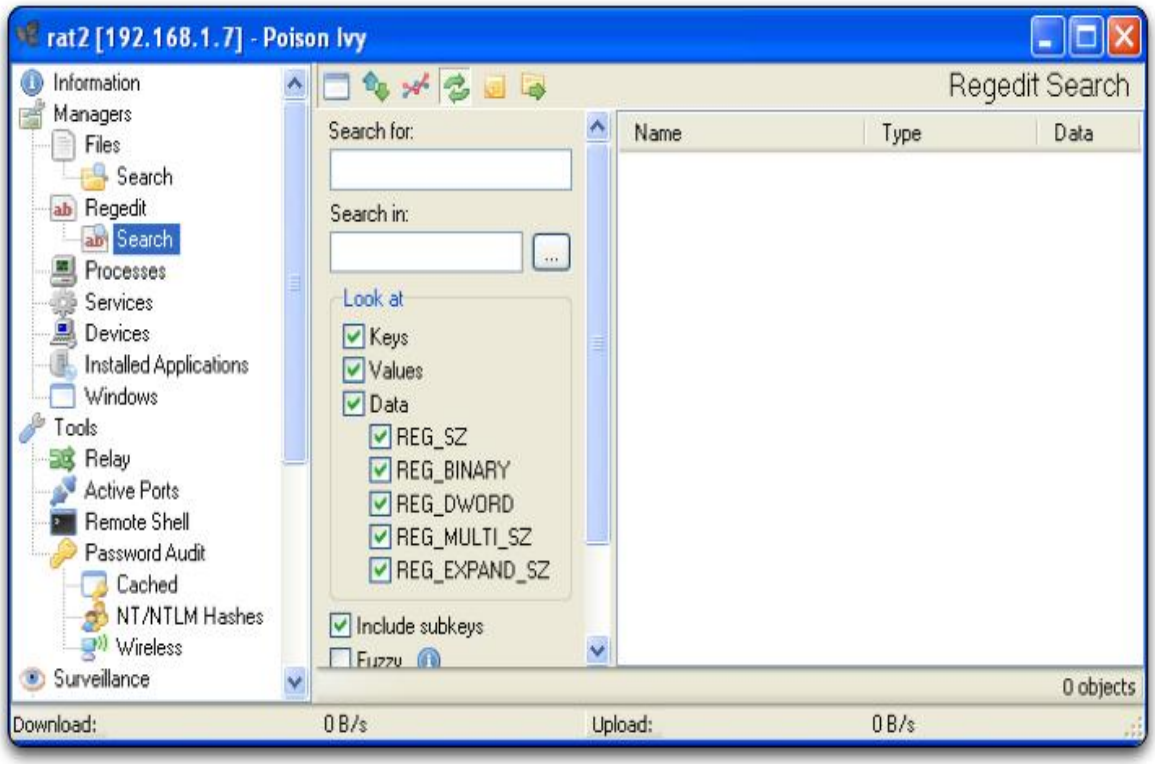


Figure 2: PI-RAT (Poison Ivy Remote Access Toolkit)

Move Laterally Towards Internal Resources to Gather Data

Once the attackers gained access to several victims using PI-RAT, they began moving laterally towards other internal resources on the RSA network. While documentation is scarce, it can be assumed they were interested in gaining access to other machines and users accounts – primarily ones with access to the source code for the RSA product and customer information. As the attackers gathered data to exfiltrate, the data was placed on several internal staging servers and aggregation points internal to RSA networks.

Exfiltrate Data

To further obscure the removal of massive amounts of data, the aggregated data was placed in several compressed and password protected RAR files. RAR is a compressed archive format for files. [Roshal, 2010] They used FTP to transfer the RAR files to an outside staging server (a compromised machine at a hosting provider). Finally, the attackers pulled the files from the external compromised host. [Rivner, 2010]

APT Detection

As APT depends on remote access and control, the network activity associated with remote control can be identified, contained and disrupted through the analysis of outbound network traffic. [Dambala, Inc., 2010] Strategies for detection of APT can be implemented through open source software tools and used to implement methodologies such as:

- Rule Sets
- Statistical and Correlation Methods
- Manual Approaches
- Automatic Blocking of Data Exfiltration

APT detection tools and tactics are available for varied operating system (OS) infrastructure and broad ranges in institutional size. This embraces Microsoft Windows ® , *nix and hybrid installations, encompassing organizations from small to medium businesses (SMB) to global enterprises and Internet Service Providers (ISPs) and the diverse groups in the middle. Here the tools mentioned in the proposed solutions are briefly described.

- Snort is open source network-based intrusion prevention and detection system (IDS/IPS) originally developed by Martin Roesch. Snort employs signature and protocol, as well as anomaly-based inspection. [Sourcefile, 2011]
- Scapy is a packet manipulation program. Scapy can create packets for a wide range of protocols. It can send and receive packets and match requests and replies. It is extensible via Python scripts and can be used for a variety of detective measures. [Biondi, 2011]

- OSSEC is a host-based open source IDS, as opposed to Snort. Its correlation and analysis engine provides log analysis, file integrity checking, Windows registry monitoring, rootkit detection, and time-based alerting as well as active response and can support most operating systems. [TrendMicro, 2011]
- Splunk is a search, monitoring and reporting tool integrating logs and other data from applications, servers and network devices. The data repository is indexed and can be queried to create graphs, reports and alerts. [Splunk.com, 2011]
- Sguil includes an intuitive GUI that provides access to realtime events, session data, and raw packet captures. Sguil facilitates the practice of network security monitoring and event driven analysis. [Visscher, 2008]
- Squert is a web application used to query and view event data stored in a Sguil database. Through the use of metadata, time series representations, weighted and logically grouped result sets it provides additional context to events. [Squert contributors, 2011]
- ModSecurity is a firewall for web applications.[Trustwave, 2010]

The tools fit into the general category of network security monitoring (NSM), as described in several books by Richard Bejtlich. [Bejtlich 2005, 2010, 2011] The challenge is to identify new APT attacks (and variants of previous attacks). The goal is to propose methodology for the detection of APT.

Rule Sets

There are several methods for the possible detection and prevention of APT-like attacks using rule sets or signatures. Commonly used in Intrusion Detection Systems (IDS), signatures match known malicious patterns of system and network behavior. As they relate specifically to the RSA attack, these include:

- identification of phishing campaigns
- recognize and block malicious traffic such as that associated with PI-RAT (Snort)
- monitor the Windows registry for “known bad” entries (OSSEC)

The RSA breach started with a phishing campaign. The Sourcefire Vulnerability Research Team (VRT) maintains a set of rules to identify and block phishing campaigns. This rule base (phishing-spam.rules) can be downloaded from the VRT Certified Rule Packages site and should be enabled in blocking mode. [Sourcefire Vulnerability Research Team, 2011]

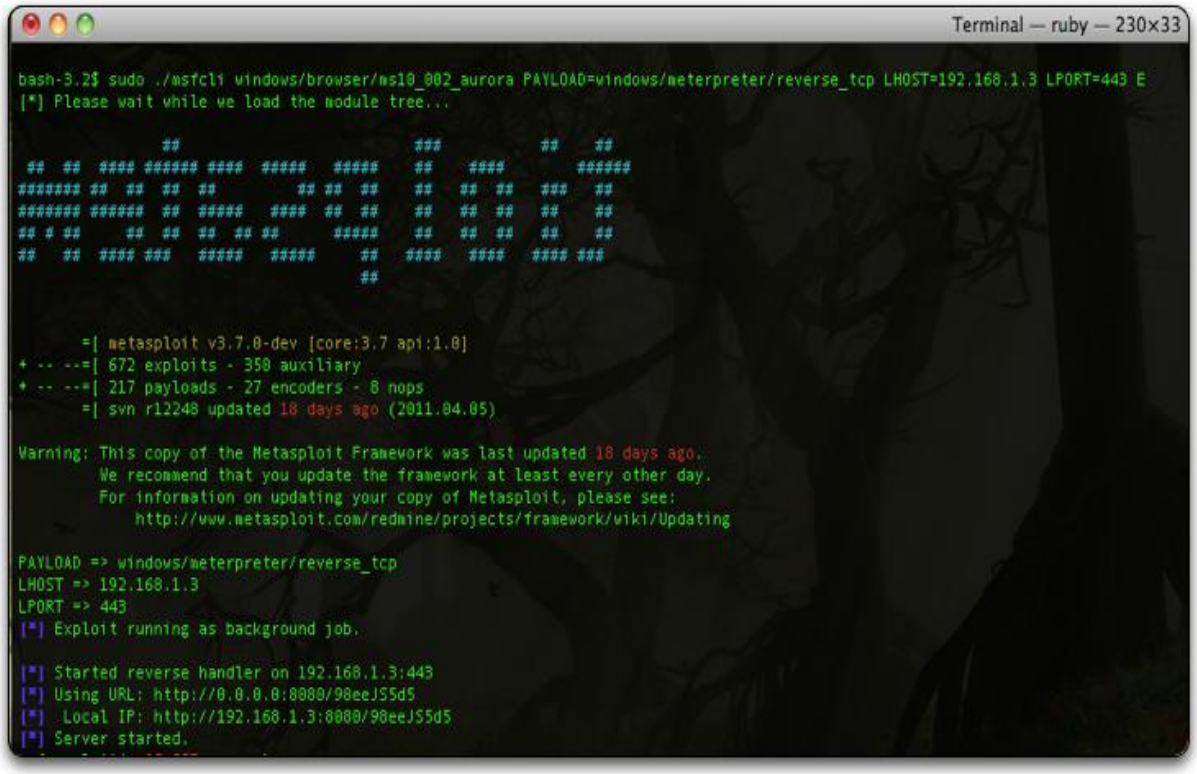


Figure 3: Replicating the Operation Aurora Attack in Metasploit

However, signatures alone have shortcomings. Days after exploit code associated with the attack against Google was made publicly available, the Operation Aurora exploit was integrated into the Metasploit ® framework (see Figure 3). Metasploit ® is a free, open source penetration testing solution for testing exploits. [Metasploit ® contributors, 2010 and 2011].

Shortly thereafter, Avert Labs produced Intrusion Detection System (IDS) signatures to detect the malicious Aurora command and control network traffic. [Blaxo, 2010]. A copy of the signature follows.

```
alert tcp $HOME_NET any -> $EXTERNAL_NET 443 (msg:"ET TROJAN Aurora C&C
Checkin"; flow:established,to_server; content:"|ff ff ff ff ff ff 00 00
fe
ff ff ff ff ff ff ff ff ff 88 ff|"; offset:0; depth:20;
classtype:trojan-activity; reference:url,
www.avertlabs.com/research/blog/index.php/2010/01/18/an-insight-into-
the-aurora-communication-protocol/;
sid:10000000001; rev:1;)
```

```
alert $EXTERNAL_NET 443 -> $HOME_NET any (msg:"ET TROJAN Aurora C&C
Checkin
Response"; flow:established,to_server; content:"|cc cc cc cc cd cc cc
cc cd
cc cc cc cc cc cc cc|"; offset:0; depth:16; classtype:trojan-activity;
reference:url,
www.avertlabs.com/research/blog/index.php/2010/01/18/an-insight-into-
the-aurora-communication-protocol/;
sid:10000000002; rev:1;)
```

While helpful for identifying the January 2010 Operation Aurora attack, the signature lacks the ability to identify completely new attacks or even significant variants of the same attack. This fact points out the importance of multiple detection methods leading to varied but related alerts that can be correlated (see **Statistical and Correlation Methods**)

Returning to the attack on RSA, it was noteworthy for simplicity. The attacker embedded PI-RAT in a Microsoft ® Office document and sent it to select RSA employees.

PI-RAT is described as

> ...an advanced remote administration tool for Windows (the client is reported to run on WINE or other emulators on various Linux/UNIX flavors), written in pure assembly (server), and Delphi (client). [Poison Ivy Contributor, 2008]

It can be used for nefarious purposes as seen in the RSA attack. However, there are some unique indicators for which monitoring can be enabled. Based on runtime analysis, when PI-RAT is installed on a Windows system, it creates the following file and registry keys:

- `C:\Documents and Settings\All Users\Application Data\Microsoft\Network\Connections\Pbk\rasphone.pbk`
- `HKCU\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Network\Location Awareness`
- `HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Network\Location Awareness`

In keeping with PI-RAT analysis, the `win_applications_rcl.txt` file (found in `/var/ossec/etc/shared` on the OSSEC [TrendMicro, 2011] server) was configured with the following entry:

```
[Possible PoisonIvy-RAT] [any] []
f:\Documents and Settings\All Users\Application
Data\Microsoft\Network\Connections\Pbk\rasphone.pbk;
r:HKEY_CURRENT_USER\SOFTWARE\MICROSOFT\WINDOWS
NT\CurrentVersion\Network\Location Awareness;
r:HKEY_LOCAL_MACHINE\SOFTWARE\MICROSOFT\WINDOWS
NT\CurrentVersion\Network\Location Awareness;
```

The following signature ensures that an OSSEC alert will fire if PI-RAT starts on a monitored system.

```
** Alert 1305369454.50708: - ossec,rootcheck,
2011 May 14 06:37:34 (HIOMALVM02.7) 192.168.1.7->rootcheck
Rule: 514 (level 2) -> 'Windows application monitor event.'
Src IP: (none)
User: (none)
Application Found: Possible PoisonIvy-RAT. File: C:\Documents and
Settings\All Users\Application
Data\Microsoft\Network\Connections\Pbk\rasphone.pbk.
```

Assume that PI-RAT performs the reverse connect on its default TCP port 3460. An IDS rule to look for outbound traffic to port 3460 with the string used in the PI-RAT initial communication sequence flags the presence of PI-RAT on the network. A sample IDS rule is below. Additionally, VRT rules detect the several PI-RAT variants. PI-RAT is present in several past APT attacks.

```
alert tcp $HOME_NET -> $EXTERNAL_NET 3460 (msg: "Poison IVY Reverse
Connection"; flow: from_client,established; content:"U|SB EC|P|B8 02 00
00 00 81 C4 04 F0 FF FF|"; depth:15; sid:10000001;)
```

While detecting a zero-day exploit is difficult, detecting the payload is easier. Notice the partial shell code shown in Program 2.

```
# Generated by Poison Ivy 2.3.2
# http://www.poisonivy-rat.com

# Length: 0x000012D3 (bytes)
PIshellcode = \
'\x55\x8B\xEC\x81\xC4\x30\xF0\xFF\xFF\x60\x33\xC0\x8D\xBD\x84\xF0' + \
'\xFF\xFF\xB9\x74\x0F\x00\x00\xF3\xAA\x33\xC0\x8D\xBD\x40\xF0\xFF' + \
'\xFF\xB9\x44\x00\x00\x00\xF3\xAA\xC7\x85\xAD\xF1\xFF\xFF\xE7\x00' + \
'\x00\x00\xE9\x6E\x0D\x00\x00\x55\x8B\xEC\x81\xC4\x30\xFA\xFF\xFF' + \
'\x8B\x75\x08\x8D\x86\xFB\x03\x00\x00\x50\x6A\x00\x6A\x00\xFF\x96' + \
'\x85\x00\x00\x00\x89\x86\xC5\x08\x00\x00\xFF\x96\x89\x00\x00\x00' + \
'\x3D\xB7\x00\x00\x00\x75\x04\xC9\xC2\x04\x00\x56\x8D\x86\x6B\x09' + \
'\x00\x00\x50\x8D\x86\x45\x01\x00\x00\x50\xFF\x96\xFD\x00\x00\x00' + \
```

Program 2: Partial Shell code generated for PI-RAT (Poison Ivy Remote Access Toolkit)

This is shell code, generated by PI-RAT, is to be included as the payload in an exploit. In this particular shell code example, there are 4,817 bytes of shell code and only 6 byes of the shell code are unique to the IP address and port for the callback.

Writing an IDS rule similar to the following detects this shell code within Microsoft ® Office files, various media files, and Portable Document Format (PDF) files. The rule would assist in preventing the zero-day exploits from succeeding.

```
alert tcp any any -> $HOME_NET any (msg: "Poison IVY shellcode";
content: "|55 8B EC 8F C4 30 F0 FF FF 60 33 C0 8D BD 84 F0|"; sid:
10000001; rev: 1;)
```

The pyOleScanner framework [Bonfa, 2011] can examine Microsoft ® Office documents further for suspicious content. Examples include known APIs (application programming interfaces), embedded structures, portable executable content, shellcode or XOR encrypted data. Program 3 shows the test script included with the pyOleScanner package. This can be easily modified to scan the structures, identify suspicious content and prevent its delivery to the end user. [O'Connor, 2011]

```
# Abbreviation of Code From pyOleScanner package
import os, sys
from classOLEScanner import pyOLEScanner
filename=sys.argv[1]
oleScanner = pyOLEScanner(filename)
fole = open(filename,'rb')
mappedOle = fole.read()
fole.close()
shellc = oleScanner.shellcode_scanner()
```

Program 3: Using Python to Detect Shellcode Embedded in Microsoft Office Documents

Statistical and Correlation Methods

The premise for including fast-flux in this discussion of APT is a simple one. Should an attacker wish to make tracking of exfiltration efforts difficult, fast-flux could be quite useful. Identification of such traffic could therefore be of benefit when monitoring egress traffic for patterns that could indicate compromise and possible exfiltation.

The Honeynet Project has documented fast-flux networks and their use at length. Drawing from *Know Your Enemy: Fast-Flux Services Networks* [HoneyNet Project, 2007]

> The goal of fast-flux is for a fully qualified domain name (such as www.hacker.ru) to have multiple (hundreds or even thousands) IP addresses assigned to it. These IP addresses are swapped in and out of flux with extreme frequency, using a combination of round-robin IP addresses and a very short Time-To-Live (TTL) for any given particular DNS Resource Record (RR). Website host names may be associated with a new set of IP addresses as often as every 3 minutes. A browser connecting to the same website every 3 minutes would actually be connecting to a different infected computer each time. In addition, the attackers ensure that the compromised systems they are using to host their scams have the best possible bandwidth and service availability.

As an example, the Conficker worm uses fast-flux to generate its daily list of domain names to check for control servers [Kriegisch, 2009]. A victim client could easily be delivering content to a "mothership" in a manner similar to fast-flux command and control traffic, again as described by the Honeynet Project [2007]:

> Fast-flux "motherships" are the controlling element behind fast-flux service networks, and are similar to the command and control (C&C) systems found in conventional botnets. However, compared to typical botnet IRC servers, fast-flux motherships have many more features. It is the upstream fast-flux mothership node, which is hidden by the front end fast-flux proxy network nodes, that actually delivers content back to the victim client who requests it.

Fast-flux traffic is represented in Figure 4.

No.	Time	Source	Destination	Protocol	Info
1	2011-05-07 06:19:04.913420	192.168.1.121	192.168.1.8	DNS	Standard query response A 192.118.40.97
2	2011-05-07 06:19:25.751097	192.168.1.121	192.168.1.8	DNS	Standard query response A 192.180.99.59
3	2011-05-07 06:19:46.841776	192.168.1.121	192.168.1.8	DNS	Standard query response A 192.106.178.136
4	2011-05-07 06:20:07.852046	192.168.1.121	192.168.1.8	DNS	Standard query response A 192.31.7.237
5	2011-05-07 06:20:28.880871	192.168.1.121	192.168.1.8	DNS	Standard query response A 192.232.239.160
6	2011-05-07 06:20:49.743262	192.168.1.121	192.168.1.8	DNS	Standard query response A 192.114.123.242
7	2011-05-07 06:21:10.860840	192.168.1.121	192.168.1.8	DNS	Standard query response A 192.151.178.111
8	2011-05-07 06:21:31.846147	192.168.1.121	192.168.1.8	DNS	Standard query response A 192.161.62.145
9	2011-05-07 06:21:52.855336	192.168.1.121	192.168.1.8	DNS	Standard query response A 192.10.144.119

Figure 4: Fast Flux Traffic

Note how the standard query response as received from 192.168.1.121 by 192.168.1.8 (infected victim) changes with each response.

Following is a discussion of the generation of fast-flux like traffic, and a Python script for analyzing the output of generated fast-flux network traffic to allow statistical variation from expected norms. ActiveState [Santos, 2006] offers a small but useful Python script to generate fake DNS traffic that closely mirrors that of fast-flux traffic. Within this script, one can define a domain, hacker.ru as an example, as well as manipulate packet count, queries, pointers, response types, and TTL. This fakeDNS script was utilized to generate a PCAP file intended to resemble fast-flux traffic that an infected node would exhibit when querying the "mothership" as described above. A Scapy [Biondi, 2010] script, dnsWatch.py (see Appendix A), was then utilized to analyze the PCAP. From a shell prompt on a host with an available Python interpreter and Scapy module, `dnsWatch.py <foo.pcap>` was executed.
Results follow for two PCAPs generated by fakeDNS.py as parsed with dnsWatch.py

```
$ python dnsWatch.py fast-flux.pcap
[*] DNS Summary for PCAP.
[*] Host: hackers.ru., Unique IP Addresses: 9

$ python dnsWatch.py dns2.pcap
[*] DNS Summary for PCAP.
[*] Host: hackers.ru., Unique IP Addresses: 436
```

The fact that unique IP responses for hackers.ru were excessive (436) is exemplary of a statistical data point for use as part of possible APT detection. Enterprise monitoring teams could configure specifically filtered captures to acquire only DNS queries for a given network segment, perhaps a segment classified as high impact if compromised. Tshark [Combs, 2011] or tcpdump [TCPDUMP contributors, 2011] allow identical syntax for filters.
Something as simple as `tcpdump 'port 53 && net 192.168.1.0/24'` would suffice for a given range. The script dnsWatch.py could then be run on a scheduled interval via a scripted cron job or scheduled task. The resulting log file could then be indexed by Splunk with applicable monitoring and alerting. Included as part of a Splunk dashboard such as slAPT, or fed to a commercial correlation engine, dnsWatch output could provide an additional layer of awareness in the battle against APT.

Figure 5 exhibits an example of the dnsWatch.py output (lower left pane) as populated in the sIAPT Splunk application (discussed in detail further below).

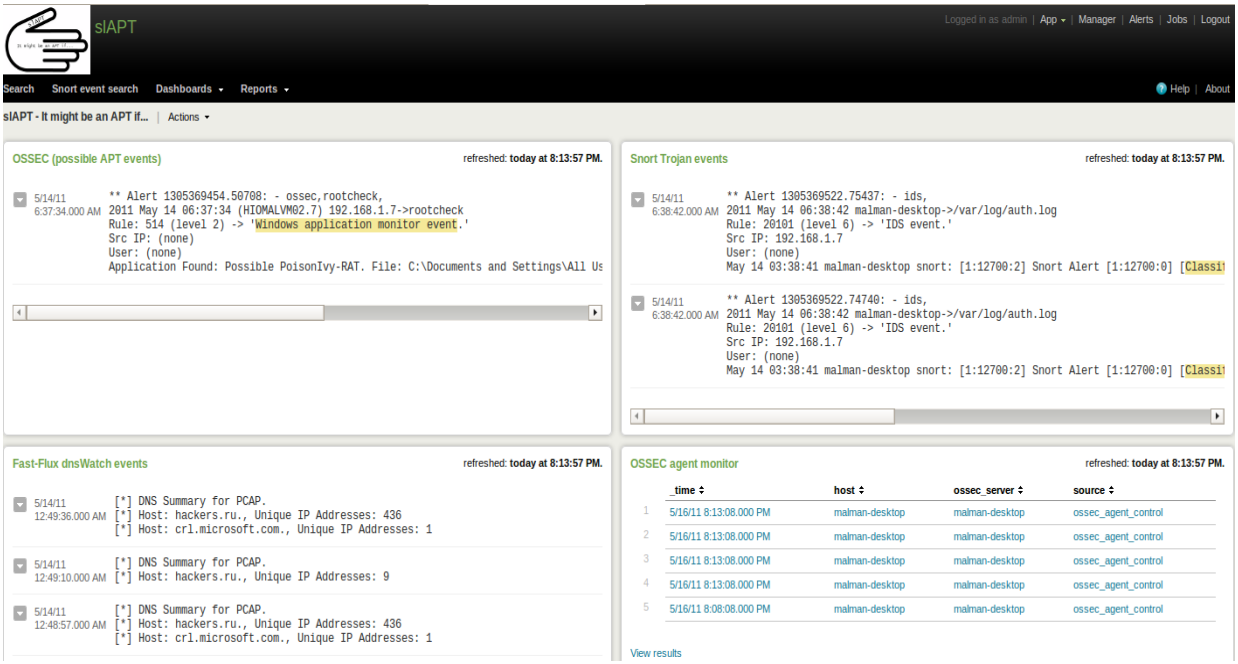


Figure 5: SIAPT Splunk app

Security Onion (SO) [Burks, 2011] is configured for immediate use by default. Ideally, it should be installed on dedicated hardware, but in short term firefights and rapid incident response deployments it can be deployed via virtual machines or commandeered systems booting from optical media. The first step to enhancing SO for optimal correlation functionality is to add a Splunk instance. [Splunk.com, 2011] A commercial license is required for full action-based alerting. However, the limited community release can suffice with a small data footprint and if console-only monitoring.

Splunk installation is remarkably straightforward. Acquire the package appropriate to your system architecture and install via the applicable package manager. On a Debian/Ubuntu system this can be accomplished with the command

```
sudo dpkg -i splunk-4.<current U version and architecture>
```

Start Splunk thereafter with the command

```
sudo /opt/splunk/bin/splunk start
```

Then browse to ***http:// localhost:8000*** (default port, can be re-assigned) or the DNS name for your Splunk server. For advanced Splunk installation tactics and usage methodology refer to the Splunk documentation. Acquire the Splunk for OSSEC application as follows:

`Manager → Apps → Find more apps online → Search for OSSEC → Select Install Free`

Security Onion is configured to use Barnyard2, "…a fork of the original barnyard project, designed specifically for Snort's new unified2 file format. [It] offloads output processing of Snort alert files to a dedicated process, minimizing dropped packets in Snort itself." It also uses similar configuration syntax to that of Snort to simplify deployment and supports all Snort output plugins as well as Sguil.

To that end, after configuring Splunk, enable an additional instance of Barnyard2 for syslog output [Securixlive.com contributors, 2010] on SO with the following:

- Open /etc/nsm/<your SO instance name>/barnyard2.conf via Mousepad
- Comment out `output sguil: sensor_name=malman-desktop-eth0 agent_port=8000`
- Add `output alert_syslog: 127.0.0.1:1514, LOG_AUTH LOG_ALERT` (assumes Splunk and SO on same server)
- Save as `barnyard2-2.conf`
- Start second instance: `sudo barnyard2 -c /etc/nsm/malman-desktop-eth0/barnyard2-2.conf -d /nsm/sensor_data/malman-desktop-eth0 -f snort.unified2 -w /etc/nsm/malman-desktop-eth0/barnyard2.waldo -U`

This allows Snort [Sourcefire, 2011] alerts to report to both Sguil [Visscher, 2008] and Splunk via syslog -- the best of both worlds. The server and sensor can both be restarted if you've further tweaked configurations with

`Applications → NSM → nsm_server_ps-restart` and `nsm_sensor_ps-restart`.

In the spirit of collecting and correlating events from disparate data sources we propose a Splunk app, *sIAPT (It Might Be an APT if...)*, that creates a dashboard converging OSSEC and Snort alerts that when matched per host might indicate or lead to the discovery of an APT attack. This app is built on queries such as

```
index=main sourcetype="Fast-Flux dnsWatch" | head 10000 | rex
"(?i)\\..*?\\., (?P<FIELDNAME>\\w+\\s+\\w+\\s+\\w+:)" | search
FIELDNAME="Unique IP Addresses:"
```

It marries events into a single user interface (UI) and providing the opportunity to create escalation alert logic.

The value of gathering and unifying events is limited, without the capability to generate alerts when certain triggers are met. As with all Splunk capabilities, a relevant search as seen above is the building block for an alert.

Choose `Save search` then check `Schedule search`. Select the interval at the search should run - as frequently as every minute or as infrequently as once a week. An email can be generated each time the search runs or only when certain conditions are met. See Figure 6

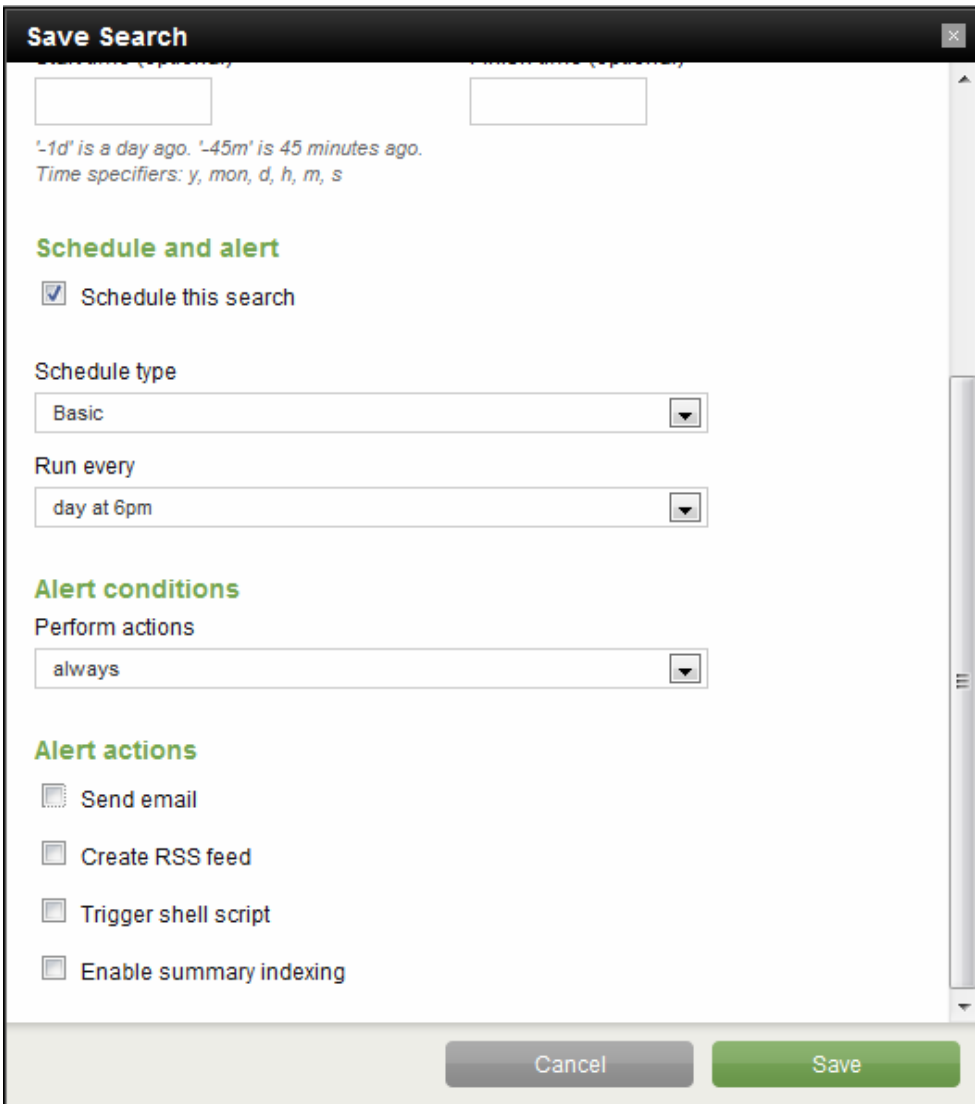


Figure 6: Splunk alert conditions

Assuming mail services or a smarthost is configured properly, an alert is received when conditions are met, inclusive of event details.

Additional actions, including shell script triggers, RSS feed creation, or summary indexing actions can be enabled as well. **Note:** Scheduled searches (Alerts) require a commercial license.

As PI-RAT runtime analysis was conducted, PCAP (packet capture) files were also collected. Using `sudo tcpreplay -i eth0 -t PI-RAT.pcap` at the command line, a Snort alert was immediately noted via Sguil as seen in Figure 7.

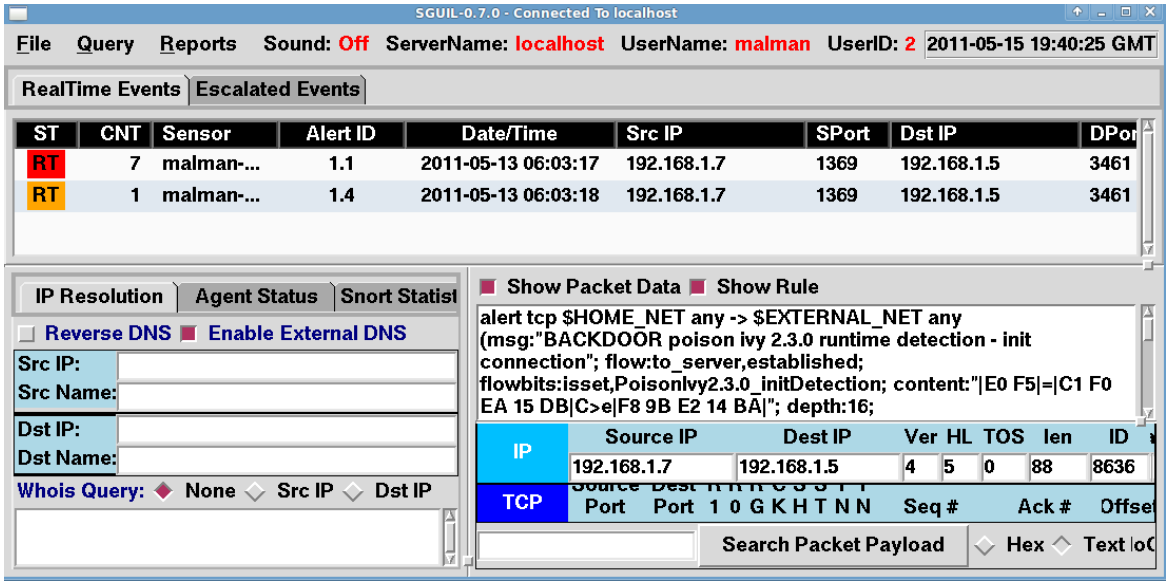


Figure 7: Snort alert noted via Sguil

Correlation options are now presented given two different event sources (OSSEC and Snort) indicating PI-RAT behavior. Sguil is designed to process possible related events. Simply right-click an event in the Sguil UI offers `View Correlated Events`. Utilizing the logic described above with Splunk and the likes of the slAPT app present improved detection opportunities as well. Consider that there are unique elements per alert (OSSEC and Snort) yet matching data that can be queried in concert, creating the impetus for increased escalation and response when correlated.

Referring back to Figure 5, searches can be defined as `eventtypes`, which can then be combined to provide correlation allowing escalated alert logic to be bound to the merged event type. The first event type can be defined as `Possible PoisonIvy-RAT`, the second `Classification: A Network Trojan was detected`. The correlating query would then be formed as:

```
eventtype=”Possible PoisonIvy-RAT” OR eventtype=”Classification:
A Network Trojan was detected”.
```

The above mentioned Splunk alerts conditions (Figure 6) could then be defined specific to the correlating query.

Figure 8 presents the slAPT App showing combined event query results.

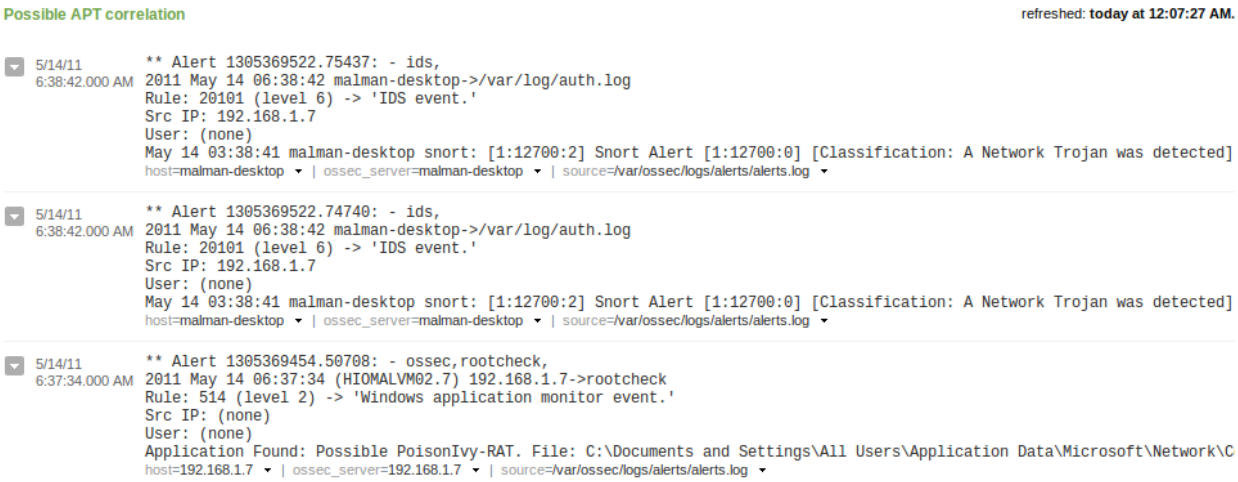


Figure 8: Possible APT correlation

Manual Approaches

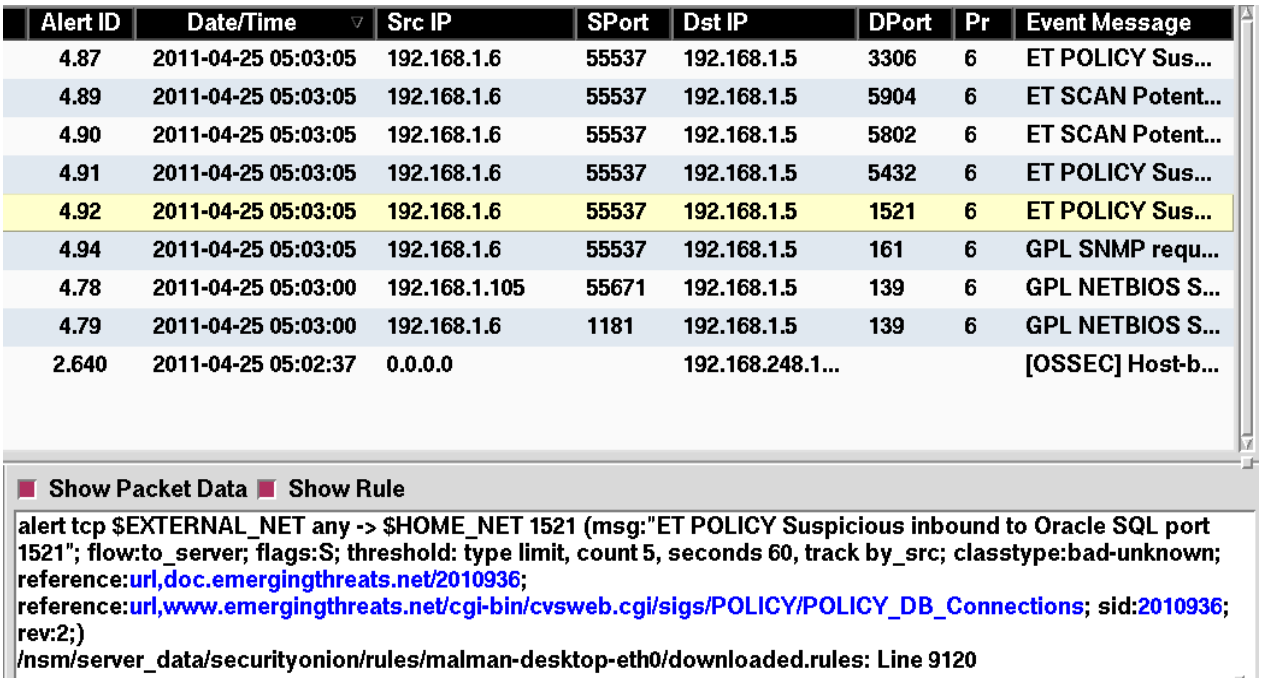
As mentioned earlier, if a zero-day attack is used as part of an APT-like infiltration, it may remain undetected. An attack may be discovered only through other logging and monitoring methods. Rob Lee, in his SANS 408 Computer Forensic Investigations - Windows in Depth course, provides an excellent example of anomalous behavior driving manual detection. Lee exhibits an MS-SQL log visually depicted entries. Simply enough, the SQL statement that is fifty times larger than the others is the obvious entry to investigate and is determined to represent SQL injection attack.

Other examples include:

- odd egress traffic initiated from the target enterprise rather than the attacker source
- DNS logs (i. e., Fast Flux)
- anomalous traffic as compared to known good netflow baselines.

An attack such as Operation Aurora attack might include pivot scanning from an initially compromised host. While the initial host compromise might go undetected, what if network sensors spotted internal network scanning traffic that triggered an investigation in turn discovering a compromised host that otherwise may have gone unnoticed?

The above mentioned behavior should be identified assuming appropriate monitoring and analysis. A typical Sguil view of such port scanning traffic is noted in Figure 9. Note the Emerging Threats rules flagging it as a combination of scan and policy alerts. Consider that in tightly controlled networks (those given the utmost priority due to data value), port scanning from hosts other than those with explicit permission to do so should trigger immediate alerts and escalation.



Alert ID	Date/Time	Src IP	SPort	Dst IP	DPort	Pr	Event Message
4.87	2011-04-25 05:03:05	192.168.1.6	55537	192.168.1.5	3306	6	ET POLICY Sus...
4.89	2011-04-25 05:03:05	192.168.1.6	55537	192.168.1.5	5904	6	ET SCAN Potent...
4.90	2011-04-25 05:03:05	192.168.1.6	55537	192.168.1.5	5802	6	ET SCAN Potent...
4.91	2011-04-25 05:03:05	192.168.1.6	55537	192.168.1.5	5432	6	ET POLICY Sus...
4.92	2011-04-25 05:03:05	192.168.1.6	55537	192.168.1.5	1521	6	ET POLICY Sus...
4.94	2011-04-25 05:03:05	192.168.1.6	55537	192.168.1.5	161	6	GPL SNMP requ...
4.78	2011-04-25 05:03:00	192.168.1.105	55671	192.168.1.5	139	6	GPL NETBIOS S...
4.79	2011-04-25 05:03:00	192.168.1.6	1181	192.168.1.5	139	6	GPL NETBIOS S...
2.640	2011-04-25 05:02:37	0.0.0.0		192.168.248.1...			[OSSEC] Host-b...

Figure 9: Sguil view of pivot host's scan probe

Many commercial security information and event managers (SIEM) offer additional functionality as it pertains to host that may be placed on a watch list, either manually or via a predefined rule set that automatically classifies certain behaviors and increases its scrutiny of the suspect host. While the likes of Security Onion, as good as it is, doesn't provide the same level of functionality one might expect from a commercial product, it still offers certain features inherent to those products. Many commercial vendors are now supplementing detection and alerting with visualization techniques. We contend that certain free and open source software (FOSS) have been meeting the needs of security visualization practitioners for years. Security Onion includes Squert which in turn makes use of AfterGlow and the graphviz libraries to provide on demand

visualizations of captured traffic. Again, making use of the premise of an attacker scanning from a beachhead host (pivoting), related scanning traffic from the pivot host presents itself in a tidy visualization. Note that, as seen in Figure 10, the victim pivot host 192.168.1.6 is probing its network neighbors for well-known services that the attacker can then exploit if vulnerable.

Figure 10: Squert/AfterGlow visualization of pivot host's scan probe

Establishing a baseline of well known, expected behavior from priority-classified network segments allow analysts to fine tune monitoring and alerting as applicable to that segment. Identification of port scanning or egress traffic initiated by an internal host (traffic not in response

to an external request for a service) would then serve as immediate indicators of unexpected and suspicious behavior.

Analysts may additionally seek to conduct geolocation lookups against IP addresses gathered from PCAPs. Appendix B provides a Python script that makes use of the MaxMind GeoIP Python API. Executing `trackByGeo.py <foo.pcap>` will return summary results including destination and source IP addresses by country.

Understandably, the connection between the likes LizaMoon and APT is obscure at first. However, SQL injection attacks are often part of the effort to establish the first foothold in the target environment in a scattershot fashion as seen in mass SQL injection attacks such as LizaMoon. If the mission of an intentional, targeted attack is to gain a beachhead in a victim network, exploiting SQL injection vulnerabilities makes for an easy opportunity. As *Anatomy of an Advanced Persistent Threat* [Shinn, 2009], SQL injection can be utilized to compromise a trusted web application, allowing iframe malware injection that will then victimize intended targets. While SQL injection may not be a root cause for an APT-like compromise, it should be monitored and prevented. A PCAP taken during the above-mentioned LizaMoon compromise provides insight into the value of well implemented NSM as seen in Figure 11.

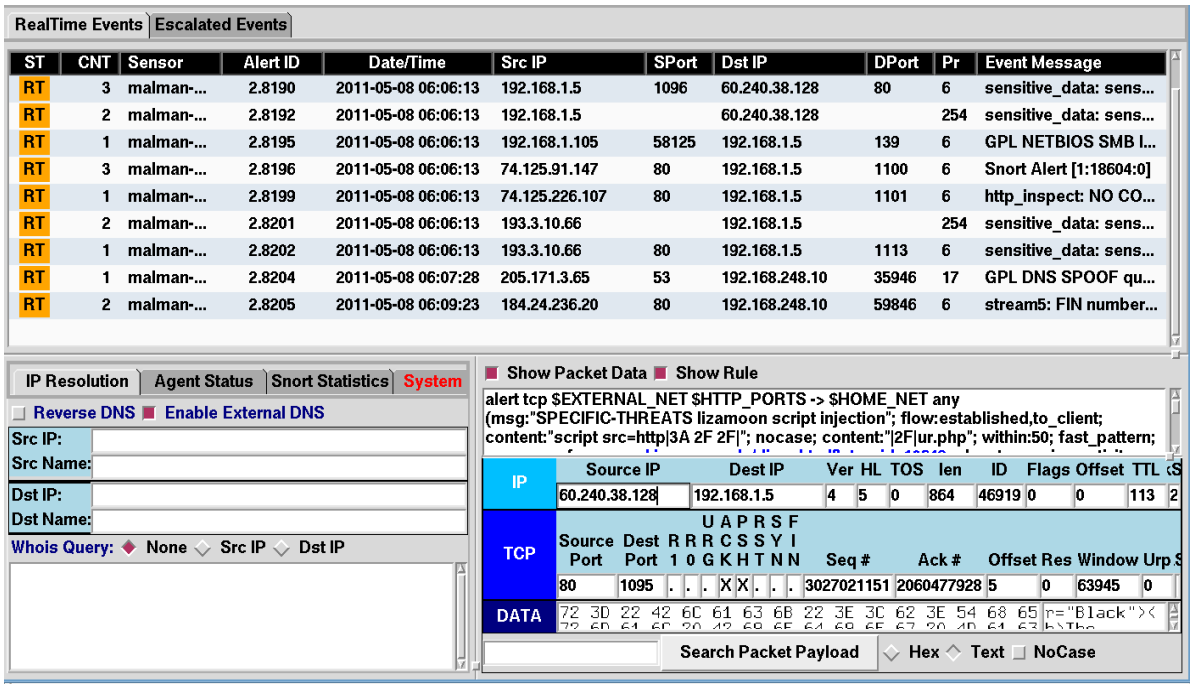


Figure 11: LizaMoon script injection detection

Note that Emerging Threats Snort signatures [Jonkman, 2011] detect LizaMoon traffic. If a would-be victim host contacts a compromised server hosting malicious injected code detection might first occur from the server to the client via SID[1] 18604:

```
alert tcp $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any
(msg:"SPECIFIC-THREATS lizamoon script injection";
flow:established,to_client; content:"script src=http|3A 2F 2F|";
nocase; content:"|2F|ur.php"; within:50; fast_pattern; nocase;
reference:url,isc.sans.edu/diary.html?storyid=10642;
classtype:misc-activity; sid:18604; rev:1;)
```

This alert alone is cause for concern. However, the fidelity of the information is improved when matched to egress traffic indicating compromise possible victim compromise. Alerts fired matching the same victim host but triggered as $HOME_NET to $EXTERNAL_NET allow correlation.

Logging produced by certain web application firewalls (WAF) can be consumed by OSSEC or Splunk. As an example, ModSecurity [Trustwave, 2010] logs can be "watched" by OSSEC. Where a web or proxy server running Apache with ModSecurity generates mod_security audit logs, include the following in ossec.conf (var/ossec/etc):

```
<localfile>
<log_format>syslog</log_format>
<location>/var/log/apache2/modsec_audit.log</location>
</localfile>
```

Alternatively, one could configure Splunk to consume the ModSecurity audit log directly

`/var/log/apache2/modsec_audit.log` as a data input via `Manager → Data inouts → Files & Directories → Add new.`

With tuning, rule optimization and well-built queries, analysts can use alerts from the WAF as stage one of a set of correlation triggers where an event from the WAF correlates to unexpected egress traffic or behavioral alerts from a server that is part of the web service.

Automatic Blocking of Data Exfiltration

IDS signatures can block exfiltration by alerting on the characteristics of outgoing traffic.

- detect and block RAR files
- OSSEC Active Response
- limit outbound access

[1]SID stands for Snort ID

- monitor for precursor attacks

In the RSA breach, the data was obscured by the use of RAR files. The Emerging Threats (formerly Bleeding Snort) group published two useful signatures for detecting RAR files leaving the network. [Emerging Threats, 2008]

```
alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"ET POLICY RAR
File Outbound"; flow: established; content:"|52 61 72 21|";
offset: 0; depth: 4; tag: session; classtype: not-suspicious;
sid: 2001950; rev:3;)
```

Should an enterprise determine that a RAR file should never egress from a given network segment, fwsnort [Rash, 2010] could be utilized on iptables firewall-based systems to block the traffic based on the signature; fwsnort translates Snort rules into iptables rules, then generates a shell script which in turn implements the resulting iptables commands.

The converted example based on the `ET POLICY RAR File Outbound` signature above would be derived as follows. Note SID 2001950 to follow the process logic from the signature above to the fwsnort parsing to the iptables blocking script.
Executing `$ fwsnort --snort-sid 2001950` would result in:

```
[+] Parsing Snort rules files...
[+] Found sid: 2001950 in emerging-all.rules
Successful translation.
[+] Logfile: ./fwsnort.log
[+] iptables script: ./fwsnort.sh
```

The resulting `fwsnort.sh` script would include:

```
$IPTABLES -A FWSNORT_FORWARD_ESTAB -p tcp -m string --hex-string
"|52617221|" --algo bm --from 64 --to 68 -m comment --comment
"sid:2001950; msg:ET POLICY RAR File Outbound; classtype:not-
suspicious; rev:3; FWS:1.1;" -j LOG --log-ip-options --log-tcp-
options --log-prefix "[1] SID2001950 ESTAB "
$IPTABLES -A FWSNORT_INPUT_ESTAB -p tcp -m string --hex-string
"|52617221|" --algo bm --from 64 --to 68 -m comment --comment
"sid:2001950; msg:ET POLICY RAR File Outbound; classtype:not-
suspicious; rev:3; FWS:1.1;" -j LOG --log-ip-options --log-tcp-
options --log-prefix "[1] SID2001950 ESTAB "
```

In particular the fwsnort logic is keying on the hexadecimal content defined in the signature.

Where the signature declares `content:"|52 61 72 21|"`, fwsnort in turn defines `--hex-string "|52617221|"`. As you can imagine, `52 61 72 21` is the hexadecimal file header for a RAR file.

Automated prevention is also achievable with OSSEC Active Response. [TrendMicro, 2010] OSSEC Active Response allows automatic execution of commands or responses when a specific event or a set of events are triggered. OSSEC Active Response can be managed in a scalable fashion that allows preventative command execution on the agent or on the server side.

While active response is highly beneficial, it must be well managed and tuned in order to prevent automated blocking born of a false positive alert. Advantages include fast (time-based) response to attacks where an action can be taken immediately. OSSEC Active Response is "extremely good deterrent against port scans, brute forces and some other types of information gathering attacks" but false positives could lead to blocked legitimate traffic or a denial of service scenario.

OSSEC Active Response is configured twofold:

- commands to execute
- those commands are bound to rules or events

To that end, OSSEC Active Response includes pre-configured shell scripts to provide functionality such as IP blocks via `/etc/hosts, iptables, ifilter, ipfw, ipsec,` and `pf`. Additionally, Windows ® based Active Response allows null routing when enabled in the agent's `ossec.conf` file and configured appropriately on the OSSEC manager.

As an example based on the above-mentioned PI-RAT OSSEC detection (see Rule Sets), `Rule: 514` or `(level 2)` would serve as the impetus to block when a certain rule or level is detected. A defined prevention tool command would be executed based on definition established in the following configuration parameters:

```
<rules_id>Comma separated list of rules id (0-9)</rules_id>
<level>The lower level to execute it (0-9)</level>
```

The `route-null.cmd` would then execute, as defined in `ossec.conf` and drop traffic bound for the suspect system.

In the fight against APT, one method to prevent the attacker from succeeding in the exfiltration of data is to integrate a proxy into the environment. Squid is one such example of a proxy. (Squid, 2011) Squid can support HTTP, HTTPS, and FTP protocols and help as an integral part of a layered defense. Before diving too much into the specific methods for blocking and prevention, consider how Squid couple with a finely tuned firewall can work to prevent the APT persistence.

Consider the following scenario. A user inside the perimeter has Internet Explorer running and is proxying HTTP/S requests through a Squid Proxy before egressing the network. The firewall is enabled to allow only the Squid Proxy to initiate TCP sessions outbound on TCP Port 443. Even if the user is exploited and a remote administration toolkit such as PI-RAT is installed – it will fail to egress the network on TCP Port 443 because only the Proxy is permitted to leave the network on TCP Port 443.

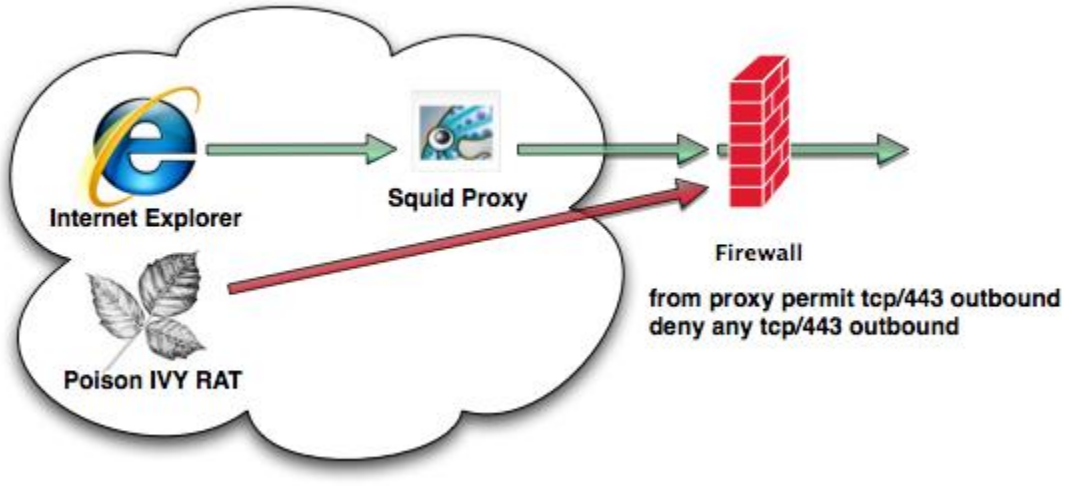


Figure 12: Perimeter Protection By Allowing Only Port 443 Outbound from Proxy

Squid uses a series of access control lists (ACLs) to permit or deny traffic. Let's examine how these ACLs can be useful in the prevention of APT. One interesting statistic about APT is that the attacks frequently occur between 10 pm – 4 am EST when there is limited monitoring of the network and limited use by the intended victims. [Mandiant Corporation, 2011]. If the organization can afford to do it, limiting users' to a period of time when the business is open can certainly assist in the prevention of APT.

```
acl workdays time M T W H F 9:00-17:00
http_access allow workdays
```

While limiting users is certainly a good first step, opportunities will present themselves when a novel attack does enter your network. Squid ACLs can further assist by preventing the attack from succeeding against multiple victims. Consider the Aurora breach, where attackers exploited a use-after-free vulnerability in the JavaScript interpreter in Microsoft ® Internet Explorer ©. An administrator can immediately prevent JavaScript from functioning on vulnerable browsers in his/her organization by adding two lines of code of the Squid configuration as follows:

```
acl javascript rep_mime_type -i ^application/x-javascript
http_access deny javascript
```

Additional rules for specific exploits can even be generated to allow for automatic blocking. In the case of the Liza Moon mass infection, infected sites contained an IFRAME to a PHP script ur.php. To safeguard users from the threat of LIZA MOON accidentally infection, an administrator can simply configure a regular expression to deny access to any ur.php pages from browsers using the Squid Proxy.

```
acl LIZAMOON urlpath_regex ^/ur.php
http_access deny LIZAMOON
```

With the omnipotence of threats to the browser out there, maintaining a contemporary and useful Squid configuration can seem overwhelming. However, an administrator can further automate this process by integrating with already existing tools. DansGuardian is one such example of Squid integrated with the ClamAV virus scanner to dynamically prevent threats to the end users by scanning all files entering the perimeter through the proxy. [Barron, 2009] animalFarm is another example of dynamically rendering active content to neutral, such as removing JavaScript inside of Portable Document Formats (PDFs) as it passes through the proxy. [O'Connor, 2011] Blacklists containing several URL filter and known bad IP addresses and domains can be updated automatically using SquidGuard. [Shalla, 2009] As of May 18th, URL.BlackList contained 3,479,329 entries of sites known to host malicious content or used in an attack. Preventing users from accessing sites that host virus infected files, spyware, phishing, or other attacks can certainly help as another component of a layered defense against the APT threat.

Conclusions

We have endeavored to provide four distinct approaches for consideration as possible countermeasures to the advance persistent threat, a pervasive, worrisome, and maliciously intentional attack against specifically targeted victims and data. These approaches have included well known signature based methodology, manual analytical practices, statistical tactics and correlation concepts, as well as automatic leak prevention.
While there is no silver bullet in the fight against concerted and targeted attacks, a holistic framework that includes varied methodology while embracing layered defensive tactics can prove fruitful. On a battlefield where innocent victims still readily fall prey to social engineering, and enterprises still fail to patch vulnerabilities in a timely manner, the advanced persistent threat will do just that: persist. Yet concepts as described above, implemented with free, open source tools or supported, commercial platforms, coupled with comprehensive and steady analysis, can help turn the tide. While the war may never be won, perhaps some battles can be.

Appendices

Appendix A - dnsWatch.py

[O'Connor, 2011]

```
import scapy
from scapy.all import *
import sys,os

if len(sys.argv) != 2:
    print "[*] Usage: dnsWatch.py <foo.pcap>"
    exit(0)

filename=sys.argv[1]
a=rdpcap(filename)
dnsDict = {}
dnsCount = {}

# Read in a pcap and parse out the DNSRR Layer
for pkt in a:
    if pkt.haslayer(DNSRR):
            rrname=pkt.getlayer(DNSRR).rrname
            rdata=pkt.getlayer(DNSRR).rdata
            # If the RRName is old, append an IP if also new
            if rrname in dnsDict:
                    rlist=dnsDict[rrname]
                    if (rlist):
                            if (rdata not in rlist):
                                    rlist.append(rdata)
                                    dnsDict[rrname]=rlist
            # If the RRName is new, add a dict entry
            else:
                    rlist=[]
                    rlist.append(rdata)
                    dnsDict[rrname]=rlist

# Build a dictionary with the index RRNAME and value=count of IPs
for rrname in dnsDict:
            rlist = dnsDict[rrname]
            if (rlist):
                    dnsCount[rrname]=len(rlist)
            else:
                    dnsCount[rrname]=0

# Sort the dictionary by count for formatting IPs first
items = [(v,k) for k,v in dnsCount.items()]
items.sort()
items.reverse()
items = [(k,v) for v,k in items]

# Print the results
print "[*] DNS Summary for PCAP."
for item in items:
            print "[*] Host: "+str(item[0])+", Unique IP Addresses: "+str(item[1])
```

Appendix B - trackByGeo.py

```
import scapy, GeoIP, sys
from scapy.all import *

gi = GeoIP.new(GeoIP.GEOIP_MEMORY_CACHE)
dstCnt = {}
srcCnt = {}

def displayResults():
    dstItems = [(v,k) for k,v in dstCnt.items()]
    dstItems.sort()
    dstItems.reverse()
    dstItems = [(k,v) for v,k in dstItems]

    srcItems = [(v,k) for k,v in srcCnt.items()]
    srcItems.sort()
    srcItems.reverse()
    srcItems = [(k,v) for v,k in srcItems]

    # Print the results
    print "[*] Destination IP Address Summary for PCAP."
    for item in dstItems:
        print "[*] Destination Country: "+str(item[0])+", Packet Count: "+str(item[1])

    print ""
    print "[*] Source IP Address Summary for PCAP."
    for item in srcItems:
        print "[*] Source Country: "+str(item[0])+", Packet Count: "+str(item[1])
def countPkt(pkt):

    if pkt.haslayer(IP):
   src=pkt.getlayer(IP).src
   dst=pkt.getlayer(IP).dst
   srcCo = gi.country_code_by_addr(src)
           dstCo = gi.country_code_by_addr(dst)


   if srcCo != None :
   if srcCo not in srcCnt: srcCnt[srcCo]=1
   else: srcCnt[srcCo]=srcCnt[srcCo]+1

   if dstCo != None:
   if dstCo not in dstCnt: dstCnt[dstCo]=1
   else: dstCnt[dstCo]=dstCnt[dstCo]+1

# main

if len(sys.argv) != 2:
    print "[*] Usage: trackByGeo.py <foo.pcap>"
    exit(0)

filename=sys.argv[1]
a=rdpcap(filename)
for pkt in a:
    countPkt(pkt)
displayResults()
```

References

Asadoorian, Paul. APT - there.. I said it. 2011 [cited May 1 2011]. Available from http://blog.tenablesecurity.com/2011/03/apt-there-i-said-it.html (accessed 1 May 2011).

Barron, Daniel. 2009. *DansGuardian*. Vol. 2.10.1.1, http://dansguardian.org (accessed 18 May 2011).

Bejtlich, Richard. Understanding the advanced persistent threat. 2010 [cited May 7 2011]. Available from http://searchsecurity.techtarget.com/magazineContent/Understanding-the-advanced-persistent-threat (accessed 7 May 2011).

———. What is APT and what does it want? 2010 [cited May 1 2011]. Available from http://taosecurity.blogspot.com/2010/01/what-is-apt-and-what-does-it-want.html (accessed 1 May 2011).

———. 2005. *Extrusion detection: Security monitoring for internal intrusions* . 1st ed.Addison-Wesley Professional.

Biondi, Philippe. 2010. *Scapy*. Vol. 2.2.0, http://www.secdev.org/projects/scapy/ (accessed 14 May 2011).

Blasco, Jamie. [Emerging-sigs] aurora. 2010 [cited April 25 2011]. Available from http://lists.emergingthreats.net/pipermail/emerging-sigs/2010-January/005567.html (accessed 25 April 2011).

Bleeding Snort Contributors. Bleeding snort. 2011 [cited 14 May 2011]. Available from http://www.bleedingsnort.com/ (accessed 14 May 2011).

Bonfa, Guiseppe. 2011. *pyOLEScanner*. Vol. 1.3, https://github.com/Evilcry/PythonScripts (accessed 18 May 2011).

Bradley, Tony. Zero day exploits: Holy grail of the malicious hacker. in About.com [database online]. [cited May 14 2011]. Available from http://netsecurity.about.com/od/newsandeditorial1/a/aazeroday.htm (accessed 14 May 2011).

Burks, Doug. 2011. *Security onion*. Vol. Security Onion v. 20110404, http://code.google.com/p/security-onion/; http://securityonion.blogspot.com/ (accessed 14 May 2011).

Burks, Doug, and Russ McRee. 2011. Electronic correspondence between McRee and burks. 6 May 2011 (accessed 6 May 2011).

Combs, Gerald, and others. 2011. *Tshark*. http://www.wireshark.org/ ed. Vol. 1.4.6, http://www.wireshark.org/ (accessed 15 May 2011).

Command Five Pty Ltd. Are you being targeted by an advanced persistent threat? 2011 [cited 16 May 2011 2011]. Available from http://www.commandfive.com (accessed 16 May 2011).

Coviello, Arthur W. "Open letter to RSA customers". in RSA [database online]. 2011 [cited May 14 2011]. Available from http://www.rsa.com/node.aspx?id=3872 (accessed 14 May 2011).

Cutler, Terry. The anatomy of an advanced persistent threat. in Wired Business Media [database online]. 2010 [cited May 15 2011]. Available from http://www.securityweek.com/anatomy-advanced-persistent-threat (accessed 15 May 2011).

Daly, Michael. The advanced persistent threat. presentation at 23rd annual large installation system administration conference. baltimore, MD. in USENIX [database online]. 2009 [cited April 23 2011]. Available from http://www.usenix.org/event/lisa09/tech/slides/daly.pdf (accessed 23 April 2011).

Damballa, Inc. What's an APT? A brief definition. 2010 [cited May 16 2011]. Available from http://www.damballa.com/knowledge/advanced-persistent-threats.php (accessed 16 May 2011).

Emerging Threats. ET POLICY RAR file outbound. 2008 [cited May 18 2011]. Available from http://doc.emergingthreats.net/bin/view/Main/2001950 (accessed 18 May 2011).

HoneyNet Project. Know your enemy: Fast-flux service networks. 2007 [cited May 2011 2011]. Available from http://www.honeynet.org/papers/ff (accessed 7 May 2011).

Jonkman, Matt. Emerging threats. 2011 [cited May 15 2011]. Available from http://www.emergingthreats.net/ (accessed 15 May 2011).

Kriegisch, Adi. Detecting Conficker in your Network. 2011 [cited May 21 2011]. Available from http://www.cert.at/static/conficker/TR_Conficker_Detection.pdf (accessed 21 May 2011).

Lemos, Robert. Fast flux foils bot-net takedown
. 2007 [cited May 7 2011]. Available from http://www.securityfocus.com/news/11473 (accessed 7 May 2011).

Mandiant Corporation. 2011. *Mandiant M-trends 2011 report.* Mandiant, (accessed 14 May 2011).

Markoff, John, and David Barboza. 2010. 2 china schools said to be tied to online attacks. *New York Times*, 18 Feb 2010, 2010, sec Technology. http://www.nytimes.com/2010/02/19/technology/19china.html (accessed 14 May 2011).

McAfee Labs, and McAfee Foundstone Professional Services. Protecting your critical assets: Lessons learned from “Operation aurora.”. in Conde Nast [database online]. 2010 [cited April 23 2011]. Available from http://www.wired.com/images_blogs/threatlevel/2010/03/operationaurora_wp_0310_fnl.pdf (accessed 23 April 2011).

McRee, Russ. 2011. Splendid splunk. *ADMIN Network and Security* 1 (0) (June, 2010).

Metasploit (r) Contributors. Metasploit (r) project. 2011 [cited May 14 2011]. Available from http://www.metasploit.com/learn-more/ (accessed 14 May 2011).

Metasploit (r) contributors. 2010. *Reproducing the "aurora" IE exploit*. Metasploit (r) project., http://blog.metasploit.com/2010/01/reproducing-aurora-ie-exploit.html (accessed 14 May 2011).

O'Connor, Terrence J. 2011. *dnsWatch.py*. Vol. 1.0 (accessed 16 May 2011).

O'Connor, Terrence J. 2011. *Animal farm: Protection from client-side attacks by rendering content with python and squid.* SANS Institute, SANS GIAC GCIH, http://www.sans.org/reading_room/whitepapers/intrusion/animal-farm-protection-client-side-attacks-rendering-content-python-squid_33614 (accessed 16 May 2011).

Poison Ivy Contributor. Poison ivy remote administration tool. 2008 [cited May 14 2011]. Available from http://www.poisonivy-rat.com/index.php (accessed 14 May 2011).

Rash, Michael. Fwsnort: Application layer IDS/IPS with iptables. 2010 [cited May 17 2011]. Available from http://cipherdyne.org/fwsnort/docs/ (accessed 17 May 2011).

Rivner, Uri. Anatomy of an attack. in RSA [database online]. 2011 [cited May 2011 2011]. Available from http://blogs.rsa.com/rivner/anatomy-of-an-attack/ (accessed 14 May 2011).

Roshal, Alexander. RAR file format. 2010 [cited May 2011 2011]. Available from http://www.rarlab.com/rar_file.htm (accessed 14 May 2011).

Santos, Francisco. 2006. *Mini fake DNS server*. Vol. 4. ActiveState Code: , http://code.activestate.com/recipes/491264-mini-fake-dns-server/history/4/ (accessed 14 May 2011).

Securixlive.com contributors. 2010. *Barnyard2* . Vol. 1.9, http://www.securixlive.com/barnyard2/ (accessed 14 May 2011).

Shalla Secure Services KG. 2009. *squidGuard*. Vol. 1.4, http://www.squidguard.org/ (accessed 18 May 2011).

Sourcefire. 2011. *Snort (r)*. Vol. 2.9.0.5, http://www.snort.org/ (accessed 14 May 2011).

Sourcefire Vulnerability Research Team (VRT) Contributors. Sourcefire vulnerability research team (VRT). in Sourcefire [database online]. 2011 [cited May 14 2011]. Available from http://www.snort.org/vrt (accessed 14 May 2011).

Splunk.com. 2011. *Splunk*. Vol. 4.2. San Francisco, CA: , http://www.splunk.com/ (accessed 14 May 2011).

Squert contributors. 2011. *Squert*. Vol. 0.8.2, http://www.squertproject.org/ (accessed 14 May 2011).

Symantec Corporation. Trojan.hydraq technical details. in Symantec Corporation [database online]. 2010 [cited April 23 2011]. Available from http://www.symantec.com/security_response/writeup.jsp?docid=2010-011114-1830-99&tabid=2.

TCPDUMP Contributors. 2010. *TCPDUMP*. http://www.tcpdump.org/ ed. Vol. 4.1.1, http://www.tcpdump.org/ (accessed 15 May 2011).

TrendMicro. 2011. *OSSEC (open source host-based intrusion detection system)* . Vol. 2.5.1, http://www.ossec.net (accessed 14 May 2011).

———. Manual: Active responses. 2010 [cited May 16 2011]. Available from http://www.ossec.net/main/manual/manual-active-responses (accessed 16 May 2011).

Trustwave, Inc. Modsecurity. 2010 [cited May 14 2011]. Available from http://www.modsecurity.org/projects/modsecurity/apache/ (accessed 14 May 2011).

Visscher, Bamm. 2008. *Sguil: The analyst console for network security monitoring*. Vol. 0.7.0, http://sguil.sourceforge.net/ (accessed 14 May 2011).

Wikipedia contributors. "Exploit (computer security)". in Wikipedia, The Free Encyclopedia [database online]. 2011 [cited May 2011 2011]. Available from http://en.wikipedia.org/wiki/Exploit_(computer_security) (accessed 14 May 2011).

———. "Operation aurora". in Wikipedia, The Free Encyclopedia [database online]. 2011 [cited May 2011 2011]. Available from http://en.wikipedia.org/wiki/Operation_Aurora (accessed 14 May 2011).

Zeltser, Lenny. Advanced persistent threat (APT): Touchy security topic #1. 2011 [cited May 1 2011]. Available from http://blog.zeltser.com/post/3459353024/touchy-security-topics-apt (accessed 1 May 2011).