Global Information Assurance Certification Paper

Fundamental Honeypotting

GIAC Gold Certification

Author: Justin Mitchell
Advisor: John Ives
December, 2006

© SANS Institute 2008, Author retains full rights.

Table of Contents

1. Introduction

"I spend almost as much time figuring out what's wrong with my computer as I do actually using it." (Stoll, 1995)

Over 20 years ago, a programmer from LBNL (Lawrence Berkeley National Laboratory) by the name of Clifford Stoll started investigating an apparent software glitch that began generating inconsistent debts between computer use charges. After realizing someone was on the other end, he soon used his pioneering surveillance savviness which included several dummy terminals and a "secret" network to fuse this activity to Markus Hess, a German hacker, whom sold ARPANET and MILNET secrets to the KGB. This (possibly the first newage) iconic event solidified the roadmap, importance, and justification as to how honeypots came about and why they exist today (and why they should continue to serve as a valuable tool for tomorrow).

Generally, honeypots accomplish the detection and collection of nefarious activity via emulating a given service or vulnerability then log the corresponding action or download the malicious code accordingly. Among other places, honeypots may reside within the LAN, DMZ, or external network (Internet).

Most cyber-security mechanisms and technologies in use today take a defensive stand (rightfully so) – much like a firewall, router, proxy, or an IPS. However, at times this tends to promote a "blurred" perspective as to what may have taken place if malicious code or an attack was allowed to fulfill its objective. For this reason, honeypots may be one of the more predominate mechanisms to

assess new threats, detect targeted attack vectors, and capture anomalous behavior. These threats, whether it be malware or specific attack channels, both can be analyzed to construct signatures or tighten up security policy and posture of the industry or company (best practices) - benefiting the community as a whole.

2. Abstract

This document is intended to guide an individual through a basic honeypot software install, identify traffic analysis and management techniques, and illustrate how data from a honeypot can be utilized to formulate valuable information.

In section three I will provide a few pertinent definitions to familiarize the reader with any new honeypot terminology. Section four elaborates as to how I came to the conclusion on what software to deploy and some of the more common accompanying features. In regards to Section five, I will discuss the importance and the underlying data management framework needed within all thriving honeypots. Section six describes a few pre-installation steps and walks the user through a basic honeypot and proxypot install. For section seven, I illustrate to the reader a few significant considerations and the backing methodologies when deploying a honeypot or honeynet and how to apply accordingly to the environment. In section eight, analysis results are revealed via graphs, screen shots, network traffic dumps, and there alike. And last but certainly not least, section nine sums up the legal conditions that honeypot users need to aware of. Unless otherwise noted, all commands are preceded by “~#” and comments are preceded with a “#”.

3. Definitions

The reader should be familiar with security terminology, basic networking knowledge, and software consistent within the industry. There are few distinct definitions and technologies that you need to be familiar with before proceeding.

Unarguably the best definition for a honeypot I came across was by Lance Spitzner (security guru whom specifically founded the Honeynet Project - http://www.honeynet.org). A **Honeypot** is "a security resource who's value lies in being probed, attacked or compromised"(Spitzner, 2002). In its simplest form, it's no more than a sponge (single box) in which its sole purpose is to absorb traffic and, if applicable, subsequently respond or react as inconspicuous as possible. In the grand scheme of things, consider them intel boxes (not to be confused with intelligent or the Intel Corporation). As with most terminology in the computer industry, defining a honeypot can be daunting and vague at times in that it encompasses many of the following concepts.

Open-relay/proxy honeypots (also known as **Proxypots**) are just what the name implies. A middleman system dedicated to monitoring activity between two endpoints of a client/server network connection. Typically, they listen on 8080, 1080, or 3128 for client requests and mimic the client and/or servers response. These attempted connections may include but are not limited to; brute-force attacks, anonymity, spam, worm/bot propagation, and reconnaissance.

Note: If you so desire to use an unorthodox port with your proxypot, you should ponder the idea of submitting your IP to a proxy list, possibly

enticing more activity. This may be good or bad depending on your configuration and logic. Personally, I wouldn't, due to legal reasons - some may view this as entrapment.

If you're a do-it-yourself kind of individual, you may find **Homemade honeypots** quite appealing. Simplistic ones that contain no more than a couple lines of code can be readily crafted via the use of a perl/shell script or even a one-liner via netcat. These are particularly effective if you're seeing temporary anomalous activity, say a high volume of probes on an irregular port.

Tarpits are defined as systems that allocate resources to inhibit the proliferation of or hinder network connections. Its main objective is to mitigate the spread of worms or various scanning activity. Despite the potential the community could gain from tarpits, they generally (and usually do) consume vast amounts of resources. Most entities are unable to justify the ROI and simply don't have the bandwidth, time, IP addresses, or hardware to dedicate to this type of honeypot.

Sandnets are used when an individual wishes to analyze malware in a secure environment while mitigating the fear and risk of a production or non-affiliated device becoming affected. A typical environment consists of a physical, logical, or virtually segmented network that is conducive to malware propagation and secure analysis. This makes for a very comfortable locale to create, tune, or test signatures and reverse engineer various code or exploits.

Honeynets may combine all the aforementioned technologies into one network. By far this is the most complex setup which requires the

most knowledge and overhead to effectively maintain. After its conception, the Honeynet Project has made the process readily available and trivial to implement. By nature they (honeynets) have the capability to emulate almost any device, service, or technology which may combine two or more of the following; routers, web servers, email servers, proxy servers, database servers, refrigerators - the possibilities are nearly endless. It boils down to ingenuity and a willingness to explore.

4. Software

I wanted my honeypotting experience and setup to be both realistic and rewarding as possible. Mind you this was my first official attempt at honeypotting so I was undoubtedly divided between a fictitious and an all natural environment. Why not take a base system of Red Hat 7.2 or Windows 2000 and slap it in the DMZ with a default deny ALL outbound and vamoose? This free for all method is more or less how it all started out, yet I also wanted to interlace a few open-source emulation projects within a “real” OS.

Note: The majority of techniques and software discussed within this document are native and/or specific to the Linux environment; however, most are applicable and portable to the Windows environment as well.

Soon into my hunt for honeypot software, I came across honeyd. “It can virtually mimic any device/OS and has been successfully tested emulating 65535 hosts.” (Provos, 2006). Honeyd favors a high interaction environment in that you are able to emulate more than one network and provides many robust configuration options for arbitrary

services and its accompanying OS. Sounded very promising, but it lacked the one main feature I highly sought after - in response to a worm or shellcode it was unable to natively interpret instructions to download a given file (worm, bot, virus, website, etc).

According to the Nepenthes website, “Nepenthes is a versatile tool to collect malware. It acts passively by emulating known vulnerabilities and downloading malware trying to exploit these vulnerabilities.” (nepenthesdev, 2006) In early 2006, two prominent honeypot projects, nepenthes and mwcollect, joined forces in a collective effort to alleviate overlapping and redundant tasks between differentiating (yet very similar) frameworks. Essentially, mwcollect now serves as an information repository for the community whom are interested or involved in collecting malware and nepenthes was named the official software used in collecting the malware itself.

To name a few tools and features for Nepenthes:

- custom modules – core functionality, which entail file upload and download capabilities (particularly useful for efficient analysis by submitting malware to Normans sandbox), shellcode, vulnerability emulation, geo2ip, etc.
- IRC channel/communication logging
- chroot – jailshell simulating a root (/) file system
- bdiffm - a light weight and handy little tool that compares two or more binary files and displays the differences via a percentage matrix.

I also dabbled around with Bubblegum proxypot. A proxy that emulates an open proxy by tangling standard proxy ports and the corresponding spammers' requests, allowing for the interception of (mainly) spam. Without modification, its main objective is to provide evidence against spammers which can be used to notify appropriate authorities if so inclined. Bubblegum is written in Perl, meaning it can easily be manipulated to adapt to almost any related honeynet schema or modified for similar purposes.

Note: While editing this document, the proxypot.org domain has become inaccessible. Please use archive.org if you would like to review or download the software.
(http://web.archive.org/web/20050829230717/www.proxypot.org)

Some bells-n-whistles for Bubblegum include:

- spamstat – a reporting tool which generates an executive summary of emails formatted via html
- parse friendly log file
- SOCKS 4/5, dynamic proxy ports, HTTP CONNECT, etc.
- rate-limit – obviously, this is very useful for keeping connection/traffic flow and system resources to a minimum
- supports mbox or maildir format for message delivery
- emulates everything – faking the SMTP DATA command further solidifying the trust relationship we have with the originating spammer(s)

Should I use VMWare?

With the debut and utter explosion of Virtual Machines (VMs) over the last few years, it's hard to ignore the possibility of having N honeypots or honeynets on one physical system.

After stumbling over Kurt Seifrieds "Honeypotting with VMware – basics"

- *... under windows this will show up in "Add/Remove programs", the Program Files directory and so forth. For UNIX there are Xfree86 patches to improve performance, as well as a complete Xfree86 server optimized for VMware guest operating systems, both of which can be identified by attackers. Much more obvious traces are also left, such as /etc/rc.d/init.d/dualconf, "Copyright (C) 1998-99, VMware Inc." and the /etc/vmware-tools/ directory.*
- *... inability to hide the CPU type effectively, an astute attacker is likely to wonder why a server with 32 megabytes of ram has a 1 gigahertz AMD CPU.*
- *... considering that the BIOS VMware uses is relatively unique it becomes quite easy to check a signature of the BIOS file to see if it is a VMware BIOS.* (Seifried, 2002)

On a Windows system, the preceding information can easily be extracted by malware or an attacker throughout the registry and on a Linux base system, files within the /proc and /etc directories or using a dmesg like command all contain this sensitive information.

and upon reading Joe Stewarts “Behavioral Malware Analysis Using Sandnets” presentation from CodeCon2006 ...

- *... Virtual NIC MAC address*
- *... VMWare I/O “backdoor”*
- *... Location of CPU descriptor tables*
- *... Timing of structured exception handler* (Stewart, 2006)

MAC addresses are unique identifiers containing alpha numeric characters separated by a colon hard coded into all network interfaces. Here’s a MAC address of a box within my network: 08:00:20:D1:65:37. You can visit the following URL for further information on the algorithm used by VMware to generate this virtual address and how to manually modify - http://www.vmware.com/support/esx21/doc/esx21admin_MACaddress.html

The VMware I/O backdoor is merely a method in which specific VMware commands and operations can access the host machine to retrieve various information and if need be, perform a given task. See http://chitchat.at.infoseek.co.jp/vmware/backdoor.html for more information.

CPU descriptor tables are used to segment the memory used by accompanying software running on the system. Generally, only the OS itself has write access but all non-privileged processes can query this information. The process in return can discern at a higher degree of accuracy on what needs to take place given the OS state and residing software (ex. killing the AV software or exploiting VMware). Several binaries that employ this method are publicly available for your testing, such as “Scooby Doo” -

http://www.trapkit.de/research/vmm/scoopydoo/index.html and "The Redpill" - http://invisiblethings.org/papers/redpill.html.

In the end, I emphatically decided to steer away from VMware, opting for solo honeypot deployment using Nepenthes via Debian and a Windows XP box for any malware release. Later on in the The Setup section, I discuss more in depth on how this software is utilized and deployed throughout the network.

5. Data Management

Now that we've established the underlying framework that I'll be utilizing, lets move on to the good stuff. You're only as strong as your weakest link and there are four basic "links" within the chain of a successful honeypot. With the exception of Data Analysis, The Honeypot Alliance provides a more definitive and thorough list of requirements which can be reviewed at the following URL - http://www.honeynet.org/alliance/requirements.html

1. Data Control. There's a good reason why it's listed first, it's important. Containment is an absolute must in order to mitigate risk to the internal and external environments, particularly in regards to new threats and savvy attackers. Your firewall is going to be the first breaking point. The policy or ACL in use will differ depending on your objective and the honeypots configuration but two questions will always remain the same. What needs to come in and what needs to go out? Ex. If you're running a Proxypot via port 8081 and 3128 and only want the XYZ Corporation to connect and not allow your proxy to initiate any

outbound connections, then only allow those IPs/ports from those sources and nothing else. Other factors such as NAT, DMZ, VPNs, bandwidth or connection throttling, and auto denying unruly sources are also good to keep in mind.

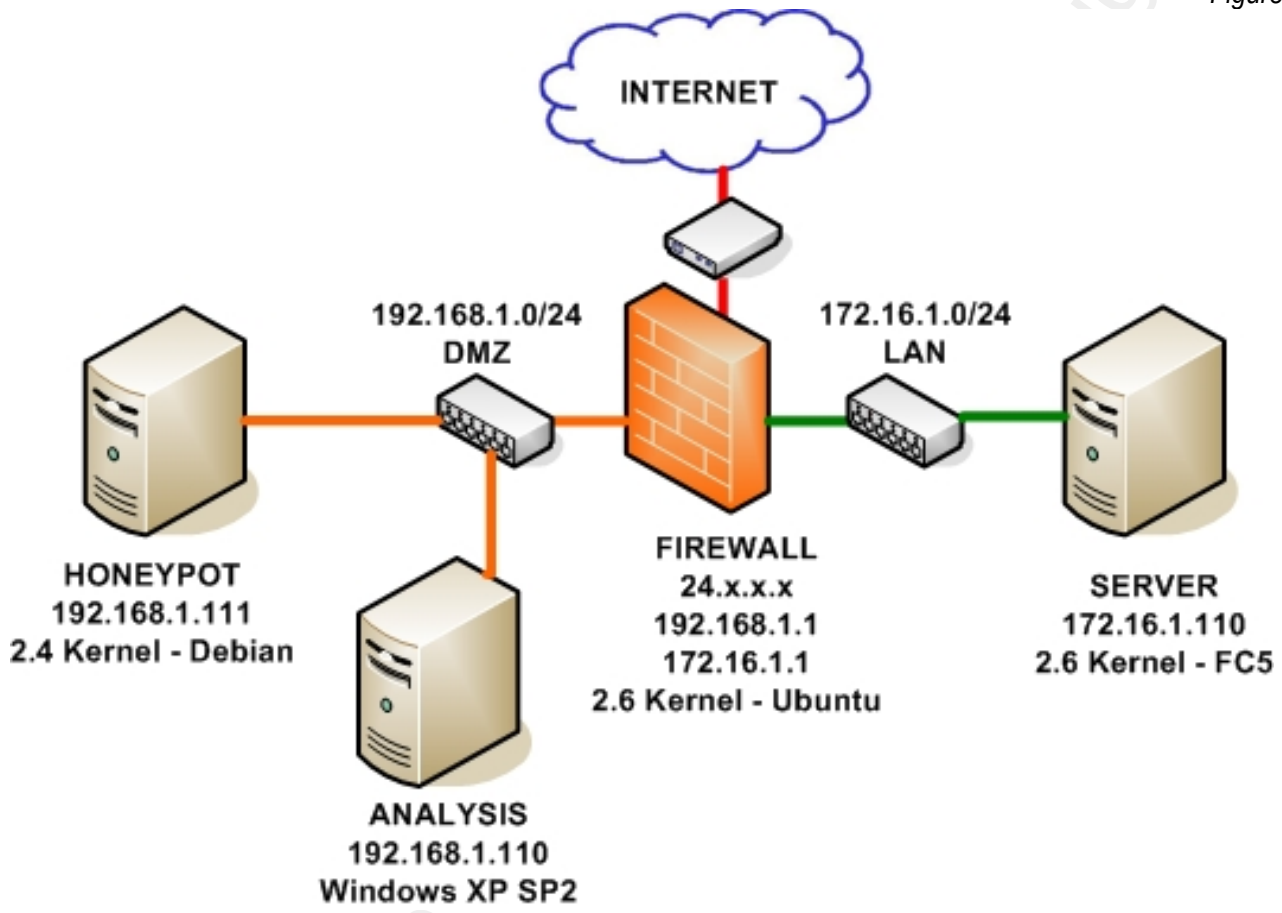
2. Data Capture. Next on our agenda discusses the software and tactics used to obtain all relevant traffic to/from the honeypot and any affiliated devices unbeknownst to the attacker or some other formidable breach. This should include firewall logs, IDS events, packet captures, fingerprints, memory dumps, or local system logs. Initially, I would suggest logging anything and everything (even firewall accepts) and from there forward build a baseline after the first couple days during the initial phase to get a feel for normalcy. Not only that but then you're almost guaranteed to (especially if you're dumping) not miss a beat.

3. Data Collection. Again, like the preceding components, aggregating the data in a secure fashion is imperative if we want to keep our system attractive to intruders and make it to the next process, which is analyzing the data itself. One method I particularly find quite trivial yet very effective is to setup a reverse SSH tunnel or use Stunnel for MySQL and Syslog traffic. This may be a sizable concern when collecting data from multiple sources, honeynet(s), but this will further strengthen covertness and help you sleep better at night. Additionally, mistakes happen and for redundancy, replicate your logs if at all possible. Although intruders are pretty snappy/adept in destroying logs and its accompanying daemon, at least you tried.

4. Data Analysis. And last but certainly not least, we get to attempt to scrutinize what or who we caught (if anything or anyone) and if they or it did anything interesting. Whether you're running MySQL queries, carving out powerful one-liners, reverse engineering a worm, or hashing out a Perl script to graph out the last six months of activity, it doesn't get any better than this. The whole reason why honeypots exists is to aid in sharpening our security posture. Here’s where you get to turn the raw data into useful information and if need be adjust network peripherals and technologies accordingly. Just remember, correlation and timestamps are vital.

6. The Setup

Figure 1



For brevity and to protect the innocent, *Figure 1* only depicts the relevant devices that are being used. Due to the lack of realistic corporate LAN activity and curiosity, I've chose to place my honeypot on the Internet (NAT'd). Externally speaking, my firewall policy essentially says "If you're not from here, there, or over yonder, go to the honeypot."

The following steps provide a basic layout on how to go about installing and configuring the software. Both Nepenthes and Bubblegum proxypot are fairly simple installs (taking no more than an hour to

get them up and operational), assuming both basic compile libraries and software generally already reside on most modern Linux OS's. Adjusting the settings, and code if necessary, are the most time consuming factors in setting up these programs. Given the limits of your OS knowledge and intention(s) the will vary from pot to pot but, but for the most part it starts with configuring the OS itself.

1. First things first, ensure ALL systems have the date and time synchronized via crontab and ntpdate! From the get go this makes it much easier to correlate the who, what, when, and where's. See *Table 1* for further instructions and output.

2. Next, a default install and thorough service audit on the honeypot should be sufficient (disabling/uninstalling the irrelevant or conflicting services). If you choose to do so, configure iptables or Firestarter at the host level for additional protection (aka. defense-in-depth). Debian uses a program by the name of update-rc.d to enable/disable services (simply removing the init and rc*.d scripts should also suffice). Likewise, if you would like to completely remove the software, apt-get should do the trick for most packages. See *Table 2* for the commands on how to do both for Apache.

3. Create two or three "dummy" accounts without shells (honey, c4tchm31fuc4n, proxy). This will allow us to run various services via test accounts for recreational and standard security purposes. If the system does happen to be compromised this will deny the users ability to login. If utilized in a production environment and you're intentions are fixed upon catching an attacker in the act, use something less conspicuous,

such as “tsawyer” or there alike. Reference *Table 3* for command.

4. Install

   A. Install Nepenthes. By far the quickest method is through apt-get. See the online documentation for additional methods, such as installing from source or dpkg - http://nepenthes.mwcollect.org/download. *Table 4* illustrates a nepenthes install by using the apt-get command.

   B. Install Bubblegum proxypot. Download the source from proxypot.org - http://www.proxypot.org/Proxypot-0.7.tar.gz Extract -> perl Makefile.pl -> make -> make install -> done.

5. Configuration. Take a couple minutes to evaluate, make backups, and then modify settings until adequately equipped/content.

   A. Configure Nepenthes. Once installed, by default, all configuration files reside within /etc/nepenthes/ (unless prefixed otherwise). Specifically, nepenthes.conf is the main location for environmental settings and for what its worth, all files are quite trivial to hash-out.

   B. Configure Bubblegum proxypot. Before configuring bubblegum, we'll need to create the configuration files using “proxypot -c” from the command line which appropriately places the files in /etc/proxypot/. Tweak away with the editor of your choice - vi /etc/proxypot/*.

6. Trial and Error. If something shouldn't happen to work as hypothesized, first review the README, located at http://nepenthes.mwcollect.org/documentation:readme, then proceed in taking the logical steps as to why you're not experiencing the expected. For instance, if you haven't seen anything or very little for 2-3 days in your logs, probe your honeypot with a scanner from another device and run a tcpdump on the system and/or check the logs to ensure traffic is actually making it to the honeypot.

7. Considerations

One option I particularly enjoyed using with Nepenthes (among others software and Linux in general) is the ability to place the daemon in a "chrooted" environment. This is accomplished with Nepenthes by using the -r option (a directory structure resembling a Linux file system, mount points, and symbolic links need to be created in order work properly prior to using this feature). See http://en.wikipedia.org/wiki/Chroot or *Table 5* for additional information and links.

1. Logging

In order to get the most out of this step, it's absolutely imperative to capture as much data as possible. Correlating data from each logging point (IDS, firewall, honeypot, etc), is critical when making the best decision – one is bound to leak some info that the other didn't. It's also very important for the data to be held accountable in regards to quality, accuracy, and uncontaminated when

processing and presenting the output, especially if it's going to be used for legal purposes or influence company policy. If need be, as shown in *Table 6*, use gpg and md5 (or sha512 for that matter) to secure things up a bit.

A. Firewall

- Include ALL available logging options at the firewall level. This will make it much easier when the analysis phase comes around for a variety of situations. One example is deducing whether the associated activity is related to return traffic, established, or an initial connection. You'll also find creating prefixes very effective for parsing purposes, respectively demonstrated from the DMZ to the LAN shown in *Table 7*.

B. Honeypot

- Correctly logging the data on/to/from the honeypot is also a necessity if it's going to be transformed into useful information. This may include, setting up an SSH/SSL tunnel or Cryptcat to transfer the data in a secure fashion, installing a kernel-based keystroke logger such as TCLEO, deploying Tripwire to monitor system integrity, and/or creating a filter via Syslog-ng to suppress nonessential events. There is nothing wrong with using the standard Syslog, I just personally prefer Syslog-ng. It's straightforward to configure, yet has some quite advanced options that allow for the most flexibility. Reference the following URLs or *Table 8,*

-
-

Appendix B – cont.

- Table 9, and *Table 10* for additional assistance - http://www.campin.net/syslog-ng/faq.html, http://farm9.org/Cryptcat, http://www.stunnel.org/faq.

NOTE: Take great caution in logging too much data as this can inadvertently, sometimes very quickly; create a DoS against your system(s). Primarily, this is directed towards an abundance of firewall logs. Redundancy is good to a certain extent, however too much tends to fog things up a bit. Remember, if the traffic is directed towards localhost (lo) and the box does happen to become "owned", an intruder can easily still sniff unencrypted UDP/TCP traffic on the local interface. This is true for Cryptcat, SSH tunnels, and Stunnel. Also, prior to utilizing any encrypted data transfers, it's highly advisable to validate with tcpdump the traffic is indeed being encrypted. A software bug (CVE-2002-1653) and/or misconfiguration could easily come back to haunt you if ignored.

2. Persistent Activity

- Inherently, in some honeypot environments, deploying a liberal firewall policy may induce unforeseen consequences. As you might imagine, they're sometimes so forgiving that it begins to unleash havoc on the more prominent resources, which in return, may cause these resources to suffer CIA (confidentiality, integrity, and availability) issues. Luckily, iptables alone has the limited ability to limit or contain unruly activity. Practical options fused together can yield some good

results, such as hosts.deny, denyhosts, Syslog-ng, and Swatch. *Table 11* shows how a combination of two iptables rules will limit incoming connections to port 8080 to no more than 5 attempts in 1 minute. The "—dport" and "-p" options can be replaced to meet just about any known protocol. This is very useful for brute force attempts.

- Another common method to shape traffic flow via the honeypot or firewall is to use iptables in conjunction with tc (Traffic Control). You can specify a multitude of restrictions ranging from burst control, protocol and network/host priorities, and/or bandwidth limitations. tc is by far the most advanced open-source traffic shaping Linux software. Manipulating network traffic takes a somewhat patient and experienced Linux user and is generally not used in solo honeypot deployment. However, Joe Roback has been kind enough to provide a pretty simple "Bandwidth Limiting with IP Masquerade – Howto" which can be read at the following URL - http://roback.cc/howtos/bandwidth.php

- Portscans occur everyday on the external side of production environments and while they pose little threat if certain precautions are taken, I wanted, at very least, to gather any relevant information such as frequency, source IP, and destination ports, etc. In the past, Snort has notoriously had its shortcomings in this area, but recently it has taken great strides in mitigating the false positives via configuration settings (see Snort manual and/or Google for tuning sfportscan). Despite the

improvements, I've found results to be much healthier with scanlogd. See http://www.openwall.com/scanlogd for additional resources.

3. Alerting

- Prioritizing and filtering known suspicious or questionable activity is desired if you plan on staying abreast and responding quickly to any new developments within the honeypot environment. I recommend using a log monitoring software or script that continuously parses or queries the data (from a log file or database) for specific criteria and takes some course of action, say an email or a page to on-call staff. *Table 12* shows how to send an email to honey_admin@server if user/group or account information had been modified on the honeypot.

8. The Analysis

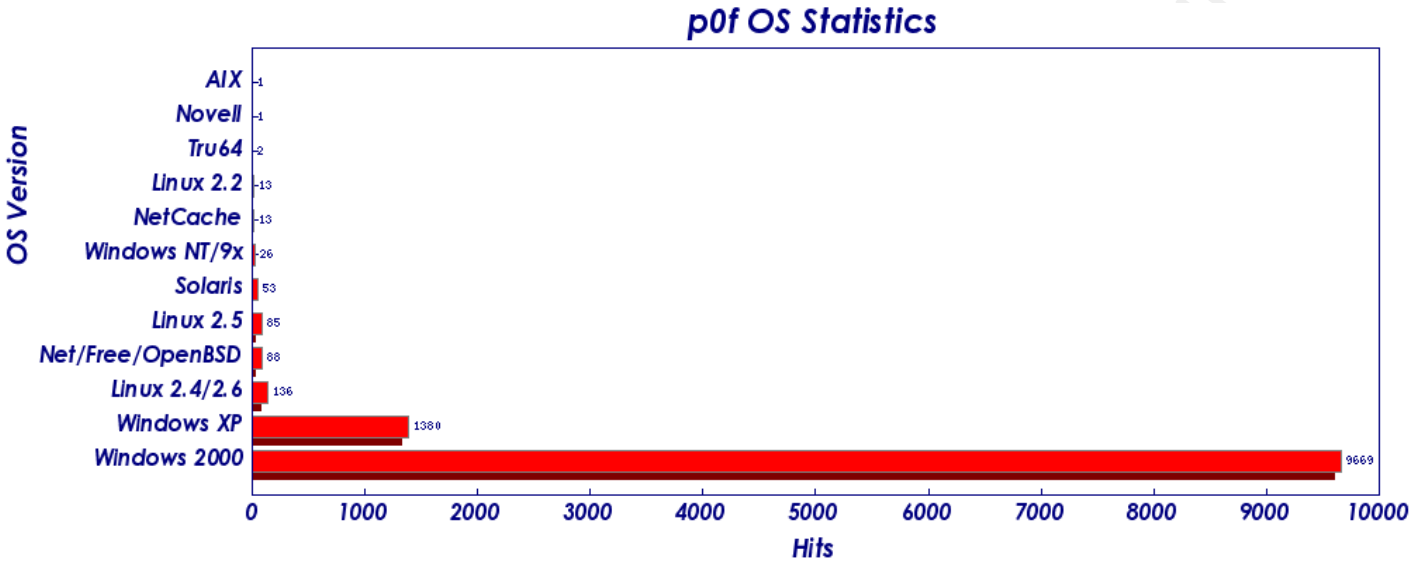
From an analysis perspective, essentially my main concern was any traffic traversing the firewall from the external side to the DMZ and vice versa. I speculate with plausible confidence that my results would greatly differ if I had placed elsewhere - such as, LAN1 -> LAN2, EXT -> DMZ, on commercial as oppose to residential ISP, etc. With that said, prior to exposing your results to your audience, you should probably take into account or at least mention, that your network isn't saturated with an array of malware, an AUP (acceptable use policy) would suppress and mitigate personnel curiosity,

unpredictable or anomalous activity, NAT and Dynamic IP addressing, and various other security related ACLs (physical and logical) would all undoubtedly have some residual impact on the results.

Nevertheless, the following information depicts relevant honeypot activity over the course of roughly 3 months.

p0f is a lightweight yet powerful utility that uses libpcap and a known signature database (TCP/IP options) to passively fingerprint network entities (PIX, PalmOS, Windows, Sega Dreamcast, Linux, etc). It acts much like a lie detector test in that humans are to polygraphs as network devices are top p0f. Predefined characteristics are emitted by both allowing us make rational (and fairly accurate) decisions/assumptions. Unique source IPs from Nepenthes logs were directly correlated with p0f logs as shown in *Figure 2*. Additional information is available at http://lcamtuf.coredump.cx/p0f.shtml. Command line syntax and example output is shown in *Table 13*.

Figure 2



While a "hit" does mean a unique IP did attempt to communicate with the honeypot, some were originating from very similar locations in terms of network vicinity (ex. a.b.c.d and a.b.c.z). Most likely this is the result of either, A. the ISP netblock is rampant with malware or B. the DHCP release time is fairly short at the ISP. Whois data revealed that a little over 75% of these sources originated from high speed residential providers. Often times, as shown later in The Analysis setion, once malware detects a live connection, it further probes for specific information, such as throughput, DNS servers, external netblock. Obviously the malware/exploit will spread much faster if scans its own netblock as opposed to scanning destinations half-way around the world while on a dialup connection.

Another signature based tool that the security field is accustomed to using is Snort. Snort is open source software using predefined signatures to detect network traffic behavior, and alert if applicable. Initially, portscans and scattered ICMP false positives were quite numerous but by tuning the configuration and

ruleset, I was able to selectively toss the majority of junk and any other insignificant alerts. Also, when I say toss, that doesn't necessarily imply I've disabled a signature, instead I chose negation through MySQL select queries. While these high volume alerts were at the top of the list, I honestly didn't mind the incoming traffic as it's quite standard to see this noise given my "accept ALL incoming" firewall policy. Needless to say, they were negated and *Table 14* represents any compelling alerts during the data capture phase.

I'm a stickler for wanting to "see it all" but I personally like the thresholding approach in regards to noisy alerts rather than completely suppressing or removing a rule from the ruleset.

Note: Depending on your network configuration and applications being used, some signatures are very broad and can get quite noisy. If this is the case, look into either modifying the signature itself or removing it from the ruleset as this can fill up a database with garbage rather quickly. If you haven't read it already, I highly advised you to read the Snort documentation at http://www.snort.org/docs/#docs or sign-up on the Snort-sigs list at https://lists.sourceforge.net/lists/listinfo/snort-sigs for additional assistance when tuning your rules and/or configuration.

After reviewing my sheared data, the Slammer worm was at the top with a total of 413 occurrences (71.5 percent of the worms originating from China). Slammer was (and still is) the fastest spreading worm ever released. Most experts conclude this was due to the single 376 byte packet size, UDP, and poor patch management. (CAIDA, 2003) If you've reviewed your IDS alerts since the last US presidential election, this should come as no surprise. Conversely,

if you're startled by this information and haven't updated your system in over three years, the time is now, because as you can tell, some individuals/organizations haven't. Slammer payload can be seen in

Appendix C – cont.

Table 15

A somewhat more recent exploit, released in 2004, still scouring the net lies within various MS Windows authentication services such as Kerberos and NTLMv2. The exploit weighed in at 62.67 percent of the alerts sourcing from the US. I opted to omit the payload given its size and collateral value to my topic. *Table 16* provides the signature that caught the attempts.

Snort alerts were pretty consistent with what other Honeypot users have discovered scouring the net. The Philippine Honeynet Project (Honeypot Alliance member) has hashed out some rather intuitive graphs depicting, among other things, top Snort alerts destined for their honeypots. For instance, in some graphs they've broken down signatures that are affiliated with only certain ports (ex. TCP/80). They're accessible by visiting the following URL - http://www.philippinehoneynet.org/data.php

Towards the end of my research it became increasingly noticeable that I needed a better method to parse network traffic in order to tweak and create signatures based upon raw dump analysis rather than relying on the preexisting Snort dumps or alerts.

Appendix C – cont.

Table 17 illustrates a script that was placed in /etc/cron.daily to aid my parsing efficiency and keep hard drive space somewhat more manageable. Then, if any activity of concern should surface via the Nepenthes, proxypot, messages log files, or there alike, I could further probe the full network dumps for additional information – such as, new shell code destined for the honeypot or a suspicious file download.

Over the course of a few months, the honeypot collected a total of 15 pieces of malware. Most were derivatives or had very similar AV signatures and characteristics of the Agobot family. Embedded throughout the mix were a few IRC bots and Dabber variants as well. *Figure 3* is from a ClamAV (v0.88.4) scan illustrating the corresponding results (with fresh definitions). Another four pieces of malware were caught after the fact.

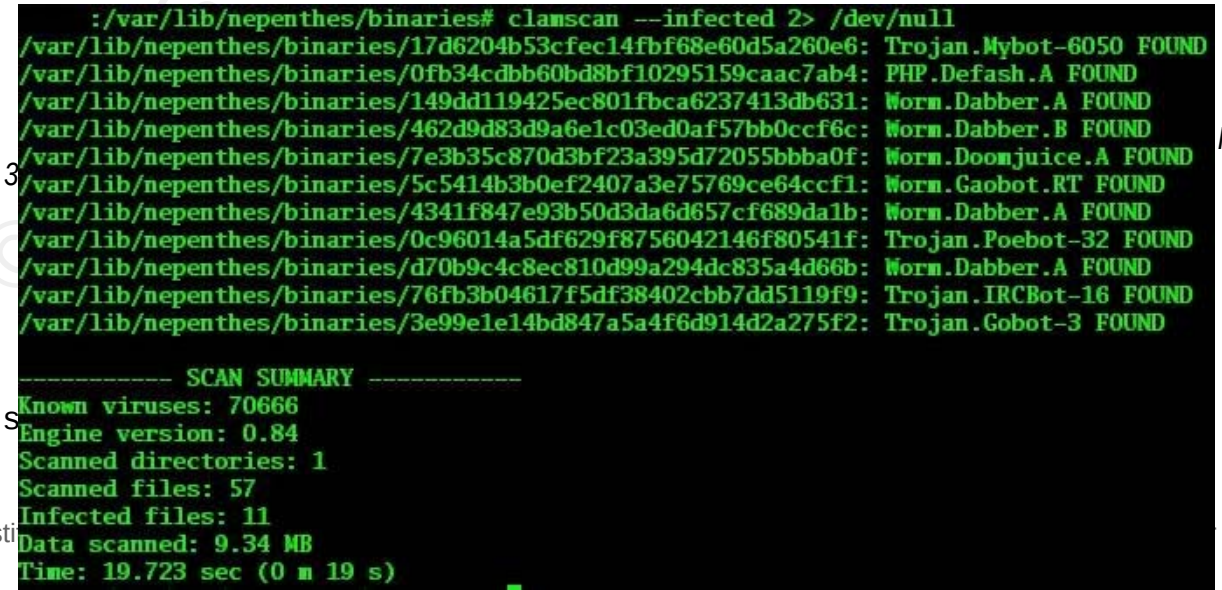


Figure 3

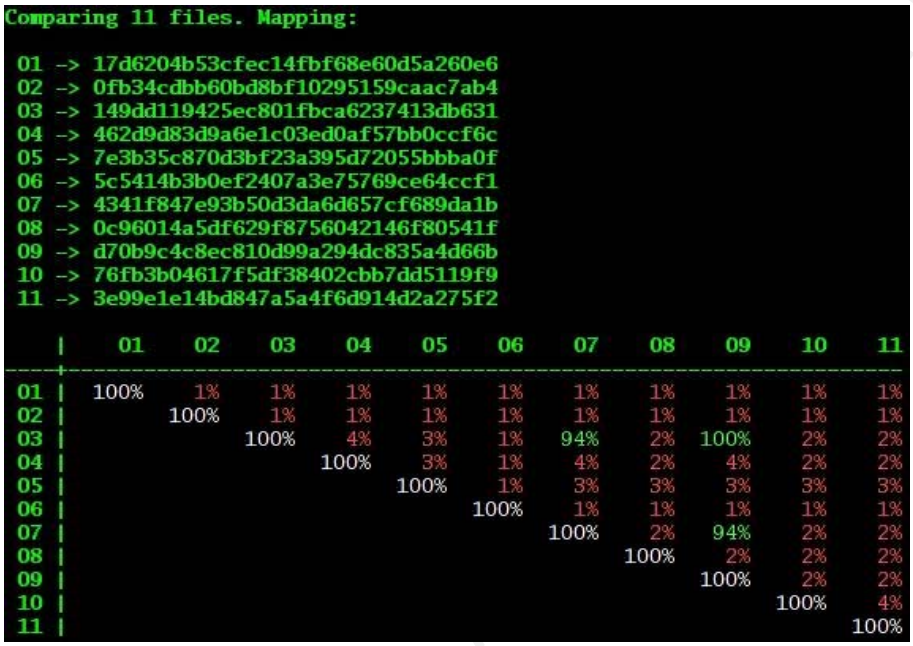
While ClamAV is/was a very useful Linux tool for discovering viruses, it does lack premier support for the most recent definitions. But generally speaking, you have to be mindful of this issue with all vendors. It's a good idea to get a second opinion on anything. That's why I opted to rerun the suspect binaries through a free service known as Virustotal - http://www.virustotal.com, which scans suspicious files with several up-to-date antivirus engines. They provide two submittal options for the community to use, via email or web site, I chose the latter. Like expected, the results were pretty consistent and a few vendors were hit or miss. This solidifies the reasoning that if you're an administrator and you need the purest environment possible (who doesn't), it's ideal (as redundant as it may seem) to have AT LEAST three removal tools at your disposal, at all times. In addition, educating users more often AND deploying a thorough patch management system will also help thwart off those “cleanup weekends”.

Note: Interestingly enough, most engines from Virustotal coined many of the infected files with a different name as their counterpart. Although, many viruses have coding similarities, perform standard system

analysis/modifications, and/or exhibit similar alike network behavior, this still brought up a intriguing question - "Just how accurate are these definitions anyhow and does it matter?". To me, it does matter because if you can't find the issue, you can't find a solution. I'm not scrutinizing the anti-virus companies' competence or software value, but merely stating that there are just way too many variables and dynamic components (variants/coders) that fall into play when it comes to generalizing malware. It seems the end result is, "They found something, you get to re-image the system." Hopefully these naming mishaps will slowly come to a conclusion with the emerging Common Malware Enumeration project gaining recognition. You can read the FAQ at http://cme.mitre.org/about/faqs.html

Bdiffm output as illustrated in *Figure 4* shows all captured malware compared to one another. While the resulting binary differences between each piece of malware was somewhat contradicting (I expected a closer relation given the AV results), I thought it was still sufficient enough to post the outcome. Source code and further information is located at http://nepenthes.mwcollect.org/snippets:bdiffm.

Figure 4



Halvar Flake from the SABRE team compiled a blog titled "*More on automated malware classification and naming*" including some fairly detailed graphs which depict the binary and coding proximity/differences. Article and images can be reviewed by visiting the following URL - http://addxorrol.blogspot.com/2006/04/more-on-automated-malware.html

I reluctantly choose not to venture to far off path into the reverse engineering realm. That subject alone entails many aspects that are beyond the scope of this paper. This field of IT allows you to see the true contrast of what each variants intended functions are by using an overwhelming amount of tools. Some of the more common ones are IDA, OllyDbg, biew, Winalysis, tcpdump, regmon, and tcpview. Additionally, the vast majority of the tools available have an enormous community backing the cause, leading to some pretty robust

techniques and homegrown scripts.

With all this malware in my hands and no manual intrusions (that I know of), curiosity was getting the best of me, so I decided to let one or two loose from the Windows XP system. I arbitrarily picked, 76fb3b04617f5df38402cbb7dd5119f9 and 7d6204b53cfec14fbf68e60d5a260e6. ClamAV respectively deemed the former “Trojan.IRCBot-16” and the latter “Trojan.Mybot-6050”. For the remainder of the paper I'll refer to these Trojans as IRCBot and MyBot.

Upon executing IRCBot

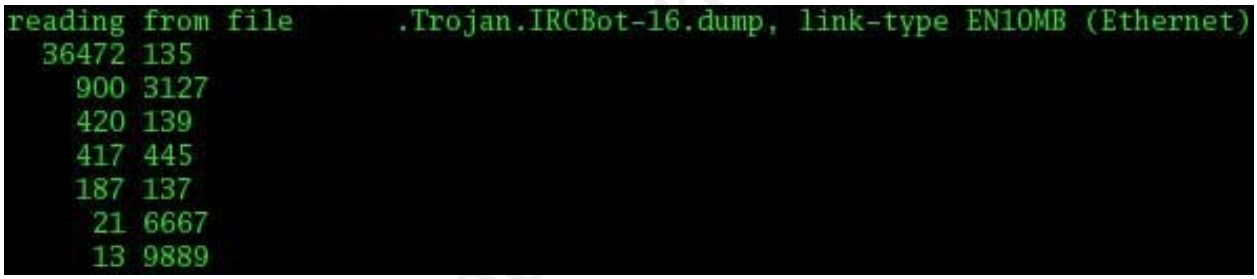
- it began flooding the net with port 135 and 139 SYNs
- less then a second later, attempted connections were being made for port 3127 (how Nepenthes initially picked up the trojan)
- two seconds later after genesis, the bot initiates a DNS lookup for ******.servebeer.com (TLD belongs to a dynamic DNS service). DNS query returned 10.10.10.10. IRCBot attempts to connect to 10.10.10.10 via TCP port 6667 – probable DNS misconfig/mishap given the passive DNS cache results were also pointing to the same IP
- commences outbound port 445 scanning to random hosts
- 5 minutes into the release, me visiting an external website triggered additional port 135 probes on my local ISP network
- with the exception of a few embedded random external NBStat requests, network forensic wise, at this point the bot had exhausted all methods of propagation and the whole process sequentially cycled back to its initial probing routine until the trojan was manually killed
- upon reboot, as shown in *Figure 5*, the malware starts back up as

the random/hidden file name created after the original binary was executed

Figure 5

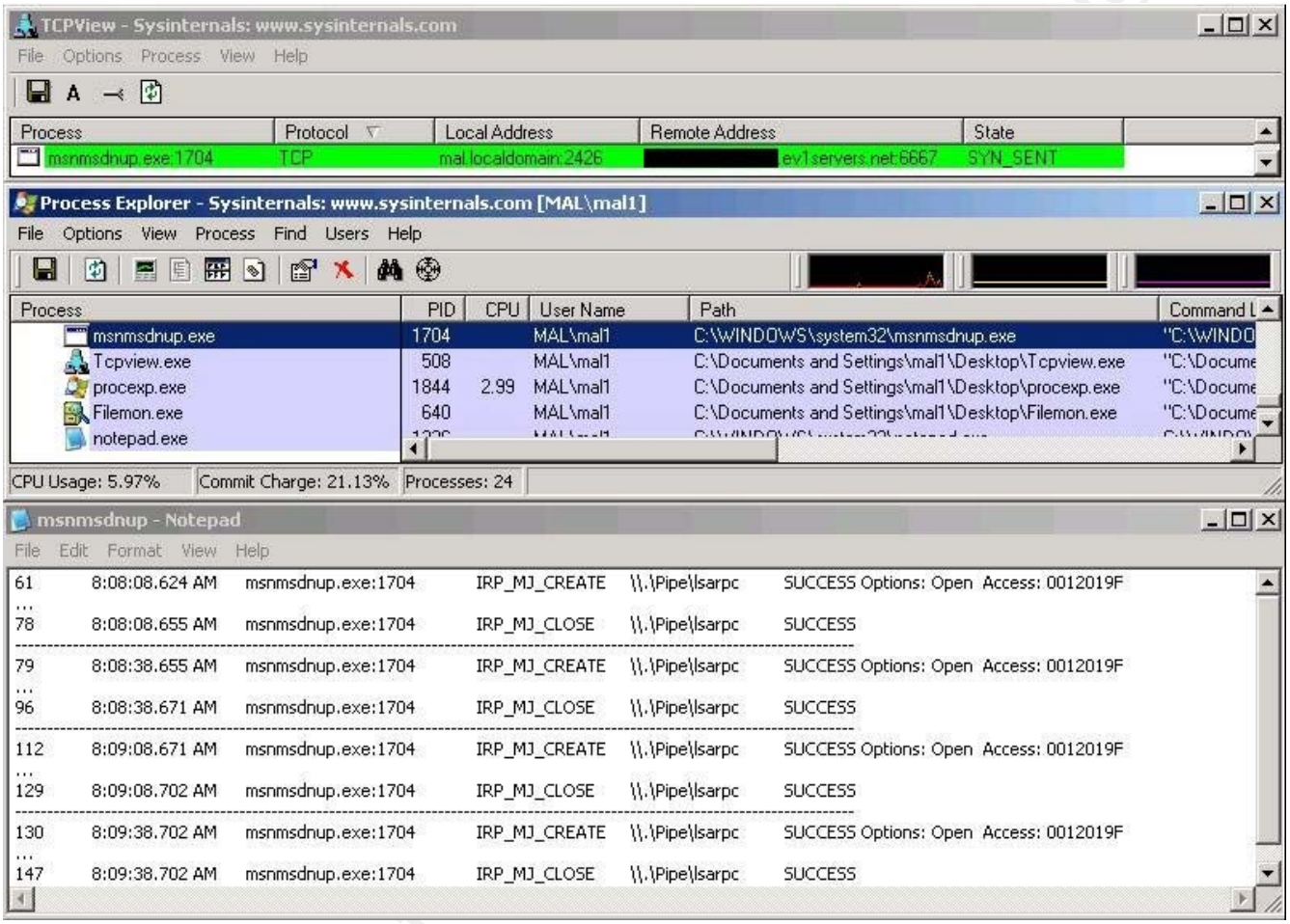
In all, seven destination ports were involved. *Figure 6* shows the corresponding occurrences in the left column and associated port number in the right column. I used tcpdump to capture the network activity while releasing the bot and re-read the capture file back through tcpdump to parse out the destination ports.

Figure 6



The next bot member I chose to toy around with, Mybot,(more commonly known as Sbot or Rbot) in contrast to the previous IRCBot, was relatively quiet in terms of persistence and propagation. Upon execution, the bot started initiating outbound connections via port 6667 to 67.15.78.x every 30 seconds. This activity is quite evident in *Figure 7*.

Figure 7

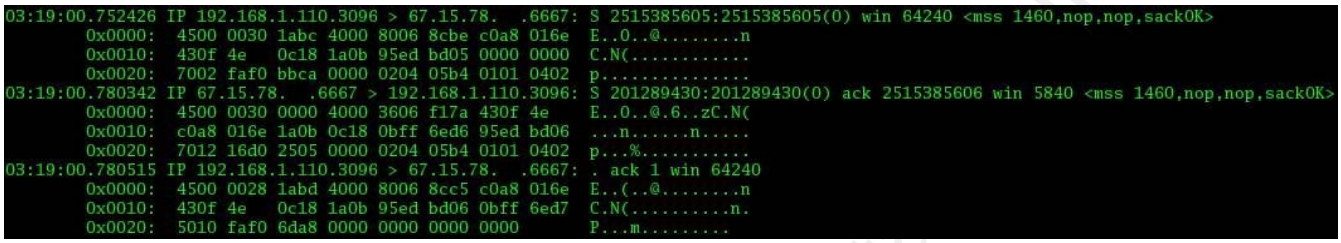


After 15 minutes of firewall drops, I decided to add a single firewall rule to allow for further analysis.

Note: Beware, some C&C (command and control) servers only accept specific commands that aren't readily self-evident. Embedded cryptography engines and algorithms have been used with zombies and malware variants for quite sometime. This amplifies the level of difficulty of an analysis and anonymity. Also, if noticed, probing a C&C server is asking for trouble.

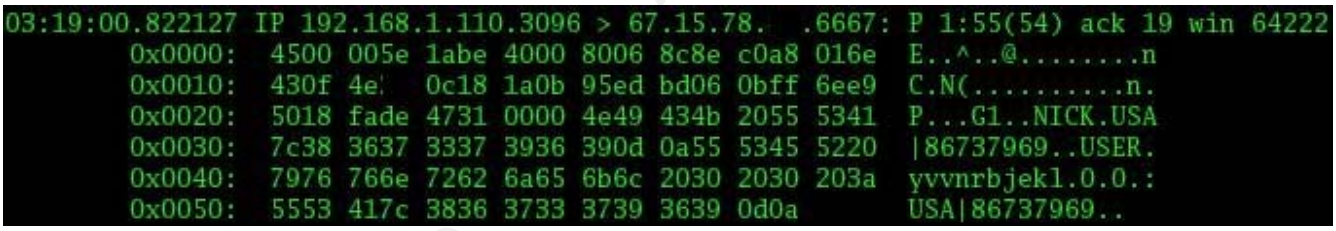
As shown in *Figure 8* the bot started out with your standard TCP three-way handshake.

Figure 8



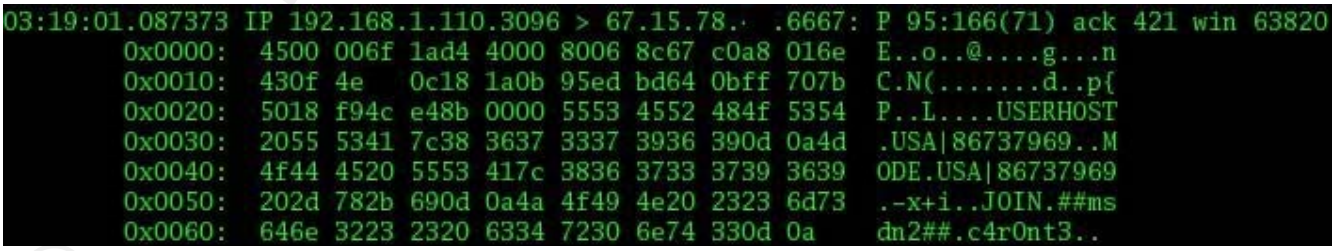
Used a randomized nick, USA|86737969, as seen in *Figure 9*. I was unable to determine the numeric string algorithm due to lack of similar variants but further research does suggest that a country code lookup is likely.

Figure 9



Enabling the invisible (+i) and removes the anonymous attributes (-x), after which, logs into msdn2 channel with c4r0nt3 as the channel key (JOIN ##msdn2## c4r0nt3) as shown in *Figure 10*.

Figure 8



Presumably, the next symbolic event would be an executive task or information retrieval command on behalf of the C&C server destined for the compromised host (possibly both or something completely different).

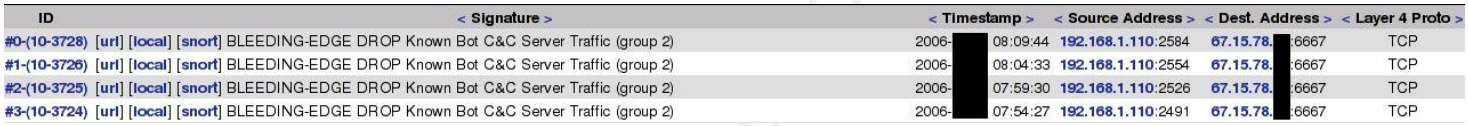
The significance in unraveling traffic patterns of malware is considerably obvious. We can now formulate signatures, based upon these “bad” C&C servers, via our IDS to alert us on systems attempting to contact these boxes and investigate accordingly (with relative confidence and more context prior to “touching” the host). Although most associated botnet ports and domains are pretty murky in nature and the servers may someday (hopefully soon) host legitimate services, dealing with false alarms is a meager hurdle to tackle when considering the potential dividends.

Fortunately for us open source enthusiasts, Snort is a very extensible IDS allowing us to create, share, and push out signatures literally within minutes of a new exploit. After a few minutes of research and constructing my own signatures for these two pieces of malware, I soon stumbled upon semi-newly released signatures made possible by the Shadowserver team (a community dedicated to tracking, infiltrating, disabling, and analyzing botnets). From there, I then appended a few of my extracted botnet IPs into the mix. As you can imagine the list of relevant IP's is quite lengthy. You can download the signatures from the following link - http://www.bleedingsnort.com/bleeding-botcc.rules

While these signatures only observe and track the destination IP addresses, they can be easily chopped up and/or modified to encompass other specifics and mitigation techniques. Examples include; snort

inline drops, DNS queries, automated throttling, or detecting anomalous activity. One definitive upside to these detection methods (specifically, DNS queries and tracking destination IP's) is that an established connection never has to be made from an infected host to the C&C server – which should be the case in a default deny firewall policy. Also, It may be a good idea to tweak the thresholding, by default, the preceding "BLEEDING-EDGE DROP Known Bot C&C Server Traffic" signatures will alert once every hour as shown in *Figure* . I've modified this setting to alert once every five minutes.

Figure 11



ID	< Signature >	< Timestamp >	< Source Address >	< Dest. Address >	< Layer 4 Proto >
#0-(10-3728)	[url] [local] [snort] BLEEDING-EDGE DROP Known Bot C&C Server Traffic (group 2)	2006-[redacted] 08:09:44	192.168.1.110:2584	67.15.78.[redacted]:6667	TCP
#1-(10-3726)	[url] [local] [snort] BLEEDING-EDGE DROP Known Bot C&C Server Traffic (group 2)	2006-[redacted] 08:04:33	192.168.1.110:2554	67.15.78.[redacted]:6667	TCP
#2-(10-3725)	[url] [local] [snort] BLEEDING-EDGE DROP Known Bot C&C Server Traffic (group 2)	2006-[redacted] 07:59:30	192.168.1.110:2526	67.15.78.[redacted]:6667	TCP
#3-(10-3724)	[url] [local] [snort] BLEEDING-EDGE DROP Known Bot C&C Server Traffic (group 2)	2006-[redacted] 07:54:27	192.168.1.110:2491	67.15.78.[redacted]:6667	TCP

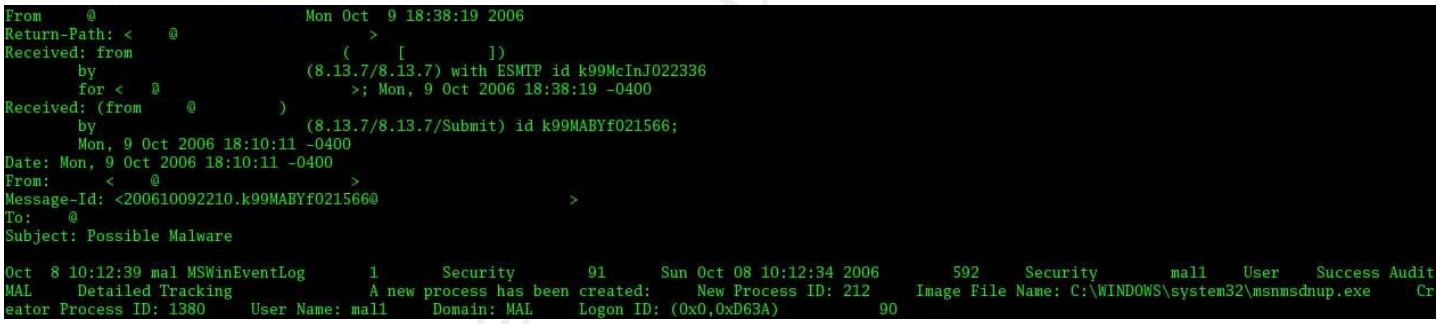
The signature in *Table 18* will detect a DNS lookup to the malicious C&C server at xxxx.dnip.net. Although the payload content option was used, a more viable solution might be to use the pcre for scalability purposes.

With stream4 enabled on snort we can create a signature, shown in *Table 19*, which requires the infected host to have an established TCP connection with the C&C server and detect communication across multiple packets.

Due to these ever changing bot standardizations (if any), possibly the greatest obstacle (particularly for larger more liberal networks, such as a college allowing legitimate IRC) is constructing signatures that produce true positives – as opposed to false positives.

Although, not always feasible, one notable correlation technique is to install Snare (or another system monitoring software) on the host and Syslog this traffic along with firewall logs to a centralized server. In this way, we can directly correlate and compare processes/executable names, firewall drops, system dumps/crashes, and anomalous activity with corresponding IDS data and malicious events. Using swatch, the Windows event seen in *Table 20* can easily being flagged via a regular expression to alert us through email, as illustrated in Figure , if the process msnmsdnup.exe starts up on any given box. I've removed the incriminating specifics.

Figure 12



Here we have the most prevalent file names Nepenthes downloaded and associated protocols used to retrieve these files - listed from top to bottom.

```
~# cat /var/log/nepenthes/logged_downloads | egrep -o '[a-zA-Z]+\.[a-z]+$' | sort -
n | uniq -c | sort -rn
    130 wulogin.exe
     91 UpMsnGraond.exe
     46 msnmsdnup.exe
     43 f.exe
     30 a.txt
     24 cmd.gif
     18 msngrs.exe
     13 hello.all
      7 sched.exe
      3 a.exe
```

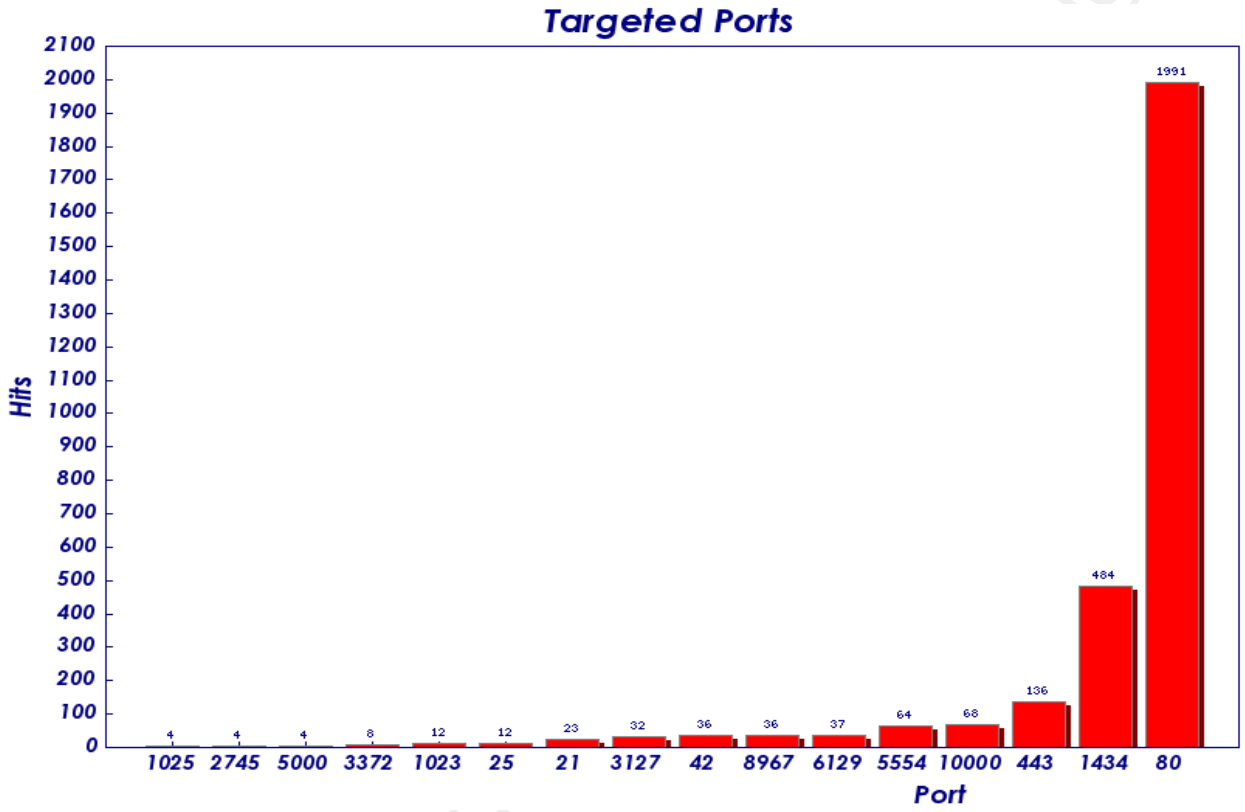
```
      2 svhosl.exe
      2 servic.exe
      2 grun.exe
      2 bootwiz.exe
      1 sendmail.html
      1 bot.exe

~# cat /var/log/nepenthes/logged_downloads | egrep -o '\].*:/' | sed -e 's/\]/ /g'
-e 's/\:\// /g' | sort -n | uniq -c | sort -rn
    156   ftp
     92   http
     59   tftp
      7   link
      1   csend
```

Generally speaking, today's internet is saturated with malware and countless other bot related scripts. In order for the community to interfere with this trendy phenomenon, it's imperative that a default deny firewall policy is enforced throughout the internal network, user education, and, I can't stress this enough, patch management. One method particularly effective to illustrate your point is through graphs.

Not surprisingly, port 80 was relatively noisy. The vast majority of activity was centered around standard PHP based and MS04-007 ASN.1 exploits with a few embedded GET/HEAD/POST requests, but there appears to be nothing out of the ordinary. Some quick research can validate a working (obviously in the wild) exploit for nearly all of the ports in *Figure 13*.

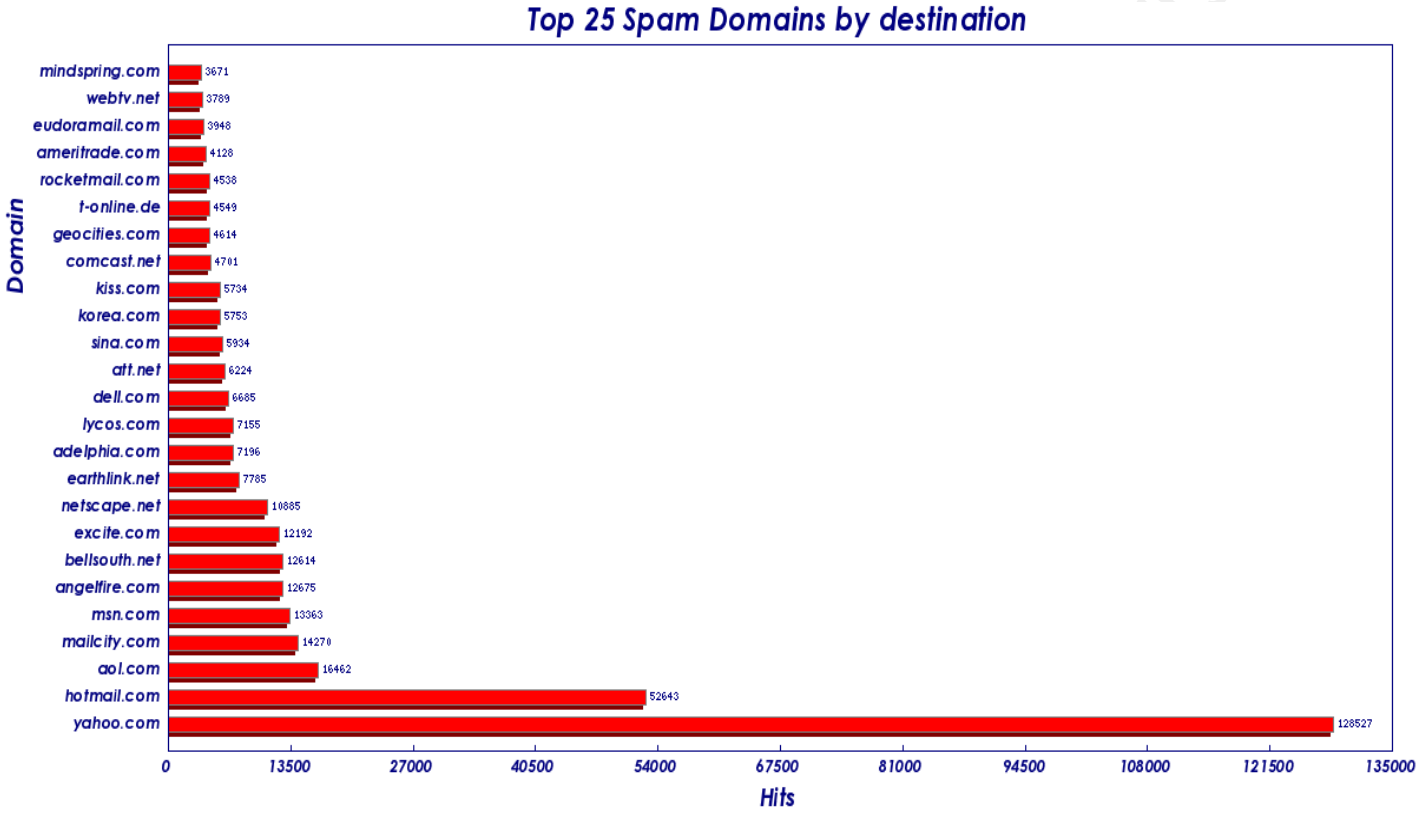
Figure 13



The proceeding graphs were created from Bubblegum proxypot logs.

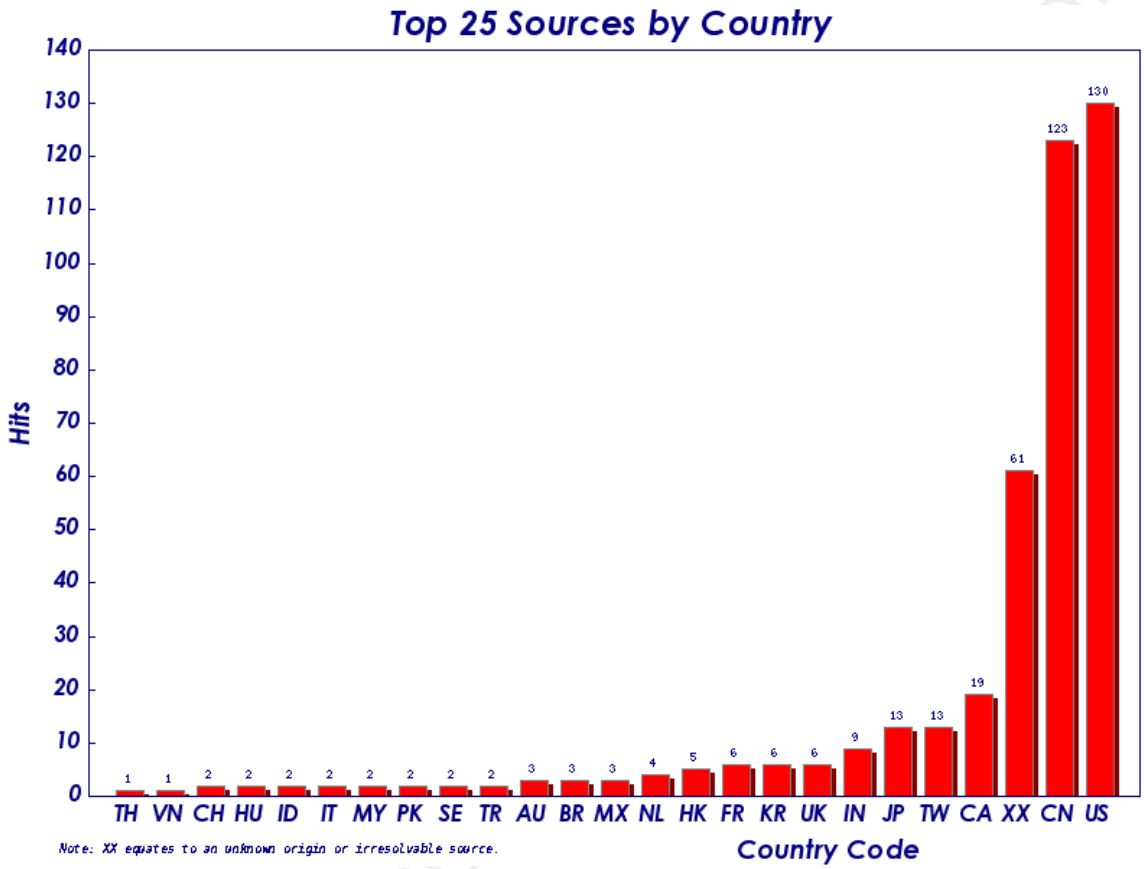
In *Figure 9*, we have the top 25 destination domains extracted from the "To:" email header. For what's worth, this doesn't necessarily mean that Yahoo email accounts are spammed more often than others but rather the spammers that were targeting my proxypot just so happen to chose to target the Yahoo domain more than any other.

Figure 94



In *Figure 15* we have the countries from which all activity originated from. Source IPs were feed into a simple shell script that parsed the country code information via http://www.hostip.info.

Figure 10

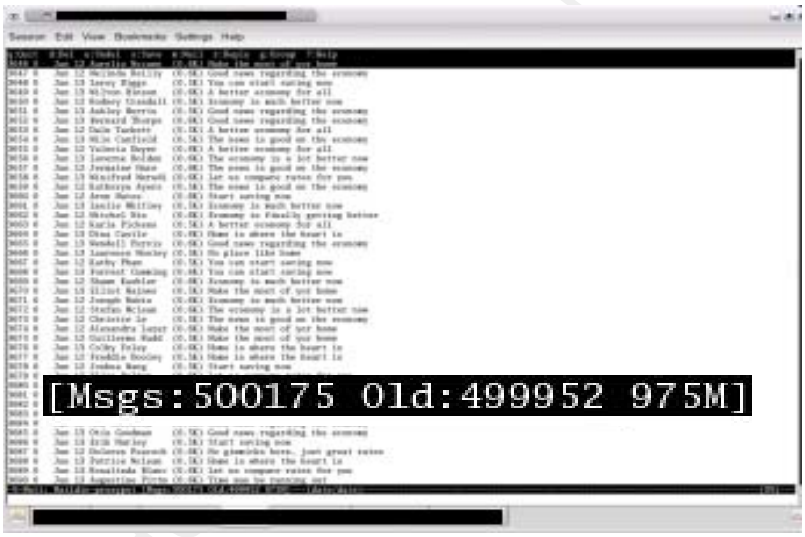


Apparently, when a persistent spammer latches onto an open proxy, they really know how to dish it out. In roughly two weeks I received over 500K emails. It took Mutt nearly 2 hours for the mailbox to come up on an AMD 1 GHz processor with 256 MB of RAM.

Figure is a console screen shot of Mutt struggling to preview the mail

box. I've blown the current vs. old message display up for exhibition.

Figure 16



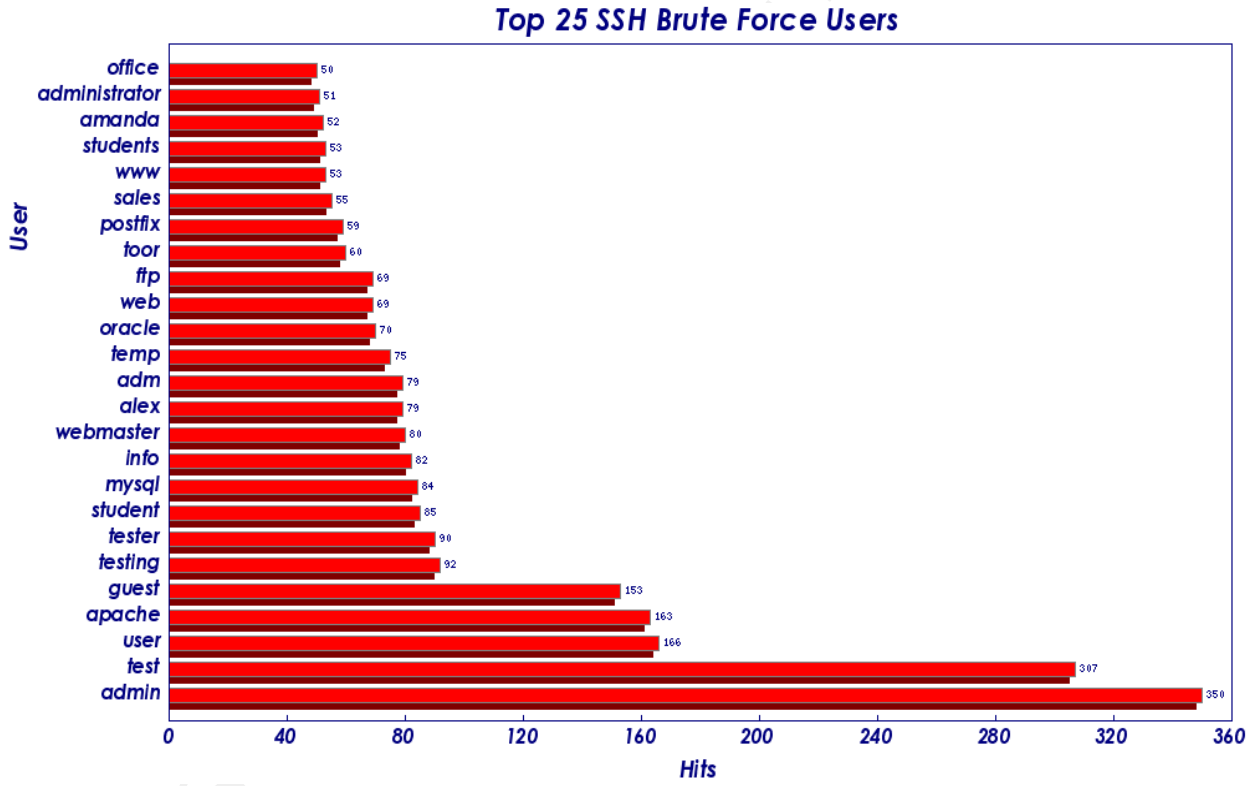
Other interesting tidbits from the honeypot;

A Perl defacing bot was picked up by Nepenthes via HTTP.

You can reference the source code at - http://atashi.net/inu/ja/notes/defacing_b0t.pl.html

A simple shell script was used to extract the most common SSH brute-force (dictionary) users from the /var/log/messages file. As you can see in *Figure* , the attacks were quite numerous. While security through obscurity doesn't work in the traditional sense, it definitely cuts down on the amount of useless data we have to maintain/sift through if we change the default SSH port.

Figure 17



Scanlogd revealed a possible Agobot/Gaobot/Phatbot variant (pretty common) originating from an external host. The table below depicts the accompanying data logged by scanlogd.

```
Aug 17 13:10:46 honeypot scanlogd: 82.96.x.x to 192.168.1.111 ports 80, 3128, 3777,
3802, 6588, 8080, 14441, ..., fSrpauxy, TOS 00, TTL 49 @13:10:46
Aug 22 16:09:49 honeypot scanlogd: 82.96.x.x to 192.168.1.111 ports 80, 3128, 3777,
3802, 6588, 8080, 14441, ..., fSrpauxy, TOS 00, TTL 49 @16:09:49
Aug 25 09:15:34 honeypot scanlogd: 82.96.x.x to 192.168.1.111 ports 80, 3128, 3777,
3802, 6588, 8080, 14441, ..., fSrpauxy, TOS 00, TTL 49 @09:15:34
Sep 11 10:37:17 honeypot scanlogd: 82.96.x.x to 192.168.1.111 ports 3128, 3777,
3802, 6588, 8080, 14441, 65506, ..., fSrpauxy, TOS 00, TTL 49 @10:37:17
Sep 11 13:10:53 honeypot scanlogd: 82.96.x.x to 192.168.1.111 ports 3128, 3777,
3802, 6588, 8080, 14441, 65506, ..., fSrpauxy, TOS 00, TTL 49 @13:10:53
Sep 12 10:41:04 honeypot scanlogd: 82.96.x.x to 192.168.1.111 ports 3128, 3777,
3802, 6588, 8080, 14441, 65506, ..., fSrpauxy, TOS 00, TTL 49 @10:41:04
Sep 13 17:57:59 honeypot scanlogd: 82.96.x.x to 192.168.1.111 ports 3777, 3128,
3802, 6588, 8080, 14441, 65506, ..., fSrpauxy, TOS 00, TTL 49 @17:57:59
Sep 14 10:23:41 honeypot scanlogd: 82.96.x.x to 192.168.1.111 ports 3128, 3777,
3802, 6588, 8080, 14441, 65506, ..., fSrpauxy, TOS 00, TTL 49 @10:23:41
```

9. Legalities

Yes, honeypots are dangerous – for a variety of reasons. Computer Security in its self is still in its infancy. That being said, the fundamental boundaries are being drawn and laws are being made as to what is politically, ethically, and morally correct.

The last thing anyone needs is an attack launched from any system directed towards a legitimate or “valued” target (internal or external), much less a box that was intended to be compromised. Not only is credibility swooped out from under you and/or the company but this also could leave the organization in a major bind, a hefty lawyer fee, or worse, land you in jail.

Some may argue that honeypots are a form of entrapment that entices an entity to break the “CIA” triage. With that in mind, I would proceed with great caution if you decide to post ANY information about specifically running a honeypot and your location to the general public (forums, proxy lists, mailing lists, and there alike). Probably the most profound situation to be aware of is when the honeypot has been comprised then an ensuing attack or other

spurious activity is launched from the system. This may be in the form of an attacker, worm, or spam but whatever the case may be; an abundant amount of research, consulting, and planning is needed prior to deploying the preceding tools/technologies.

Legal Analyst, Richard P. Salgado from Standford Law, favors configurations where a hacker is invisibly rerouted to a honeypot after beginning an attack on a production machine. "The closer the honeypot is to the production server, the less likely that it's going to have some of the legal issues that we're talking about," he said, because the monitoring becomes part of the normal process of protecting the production machine. (Poulsen, 2003)

10. Conclusion

With all these methods (and more) combined, you have a very valuable tool. Whether you're reviewing logs and alerts, embracing topology changes, patch management, or crafting/updating your signatures, all feasible aspects need to be addressed ASAP. Your results may have enough influence to affect all tiers within an organization and there just isn't any room for error when it comes to this industry, the business, or your job.

Honeypots are fun and a great learning tool which not only allows you to orchestrate a wide array of security mechanisms into your policy but as well as brush up on new standards and threats that we are faced with on a day-to-day basis. I imagine individuals that enjoying finding answers and those with a keen sense of curiosity excel in this area as there really isn't a right or wrong way to

deploy such a tool.

All things come with a cost and these pros can often overshadow the double edge sword honeypots bring forth. Attackers as well malicious coders often use this technology to construct evasion techniques by reverse engineering, spawn similar variants, or gain additional devious knowledge about how the code or attack methodology itself operates.

The future of honeypots will most likely entail the versatile and dynamic ability to be plugged right into a port or aimlessly listen via WiFi (hostap) and do its job right from the get go (UPnP if you will) detecting nefarious patterns with minimal user interaction.

Like most computer related topics, unless you stay abreast of the latest and greatest, the value of a honeypot is greatly depreciated each day left untouched or pushed aside and no matter how advanced the software, configuration, and/or devices you may have deployed, don’t forget the basics – common sense and simplicity.

11. References

CAIDA, (2003). The spread of the sapphire/slammer worm. Retrieved Oct, 2006, Web site: http://www.caida.org/publications/papers/2003/sapphire/sapphire.html

nepenthesdev, (2006, June 25). *Nepenthes readme.* Retrieved May, 2006, Web site: http://nepenthes.mwcollect.org/documentation:readme

nepenthesdev, (2006, July 7). *Nepenthes - finest collection.* Retrieved July 23, 2006, Web site: http://nepenthes.mwcollect.org/

Poulsen, K (2003. April 16). *Use a honeypot, go to prison?*. Retrieved October 22, 2006, Web site: http://www.securityfocus.com/news/4004

Provos, N (2006, April 16). *Honeyd development.* Retrieved July 20, 2006, Web site: http://www.honeyd.org

Seifried, K (2002, Febuary 15). *Honeypotting with wmware – basics*. Retrieved June 10, 2006, Web site:

http://www.seifried.org/security/ids/20020107-honeypot-vmware-basics.html

Spitzner, L (2002, May 17). *Honeypots (definitions and value of honeypots)*. Retrieved July 17, 2006, Web site: http://www.governmentsecurity.org/articles/HoneypotsDefinitionsandValueofHoneypots.php

Stwewart, J (2006). *Behavioral malware analysis using sandnets.* Retrieved August, 2006, from http://www.sv-issa.org/sandnets.pdf

Stoll, C (1995). *The quotations page*. Retrieved October 19, 2006, Web site: http://www.quotationspage.com/quotes/Clifford_Stoll/

Appendix A – The Setup

Table 1

```
~# crontab –e

#crontab contents
34 *    * * * /usr/sbin/ntpdate time.nist.gov

#log file
Aug  9 00:34:01 server crond[14216]: (root) CMD (/usr/sbin/ntpdate time.nist.gov)
Aug  9 00:34:01 src@honeypot /USR/SBIN/CRON[26532]: (root) CMD (/usr/sbin/ntpdate
time.nist.gov
```

Table 2

```
~# update-rc.d –f apache
update-rc.d: /etc/init.d/apache exists during rc.d purge (continuing)
 Removing any system startup links for /etc/init.d/apache ...
   /etc/rc0.d/K20apache
   /etc/rc1.d/K20apache
   /etc/rc2.d/S20apache
   /etc/rc3.d/S20apache
   /etc/rc4.d/S20apache
   /etc/rc5.d/S20apache
   /etc/rc6.d/K20apache

~# apt-get remove apache
Reading Package Lists... Done
Building Dependency Tree... Done
The following packages will be REMOVED:
  apache
0 upgraded, 0 newly installed, 1 to remove and 5 not upgraded.
Need to get 0B of archives.
After unpacking 81.9kB disk space will be freed.
Do you want to continue? [Y/n]
```

Table 3

```
~# useradd username -g users -s /bin/false
```

Table 4

```
~# apt-install nepenthes
```

Appendix B – Considerations

Table 5

```
~# nepenthes -r /opt/nepenthes
```

Table 6

```
~# gpg –c /var/log/messages; md5sum /var/log/messages.gpg
```

Table 7

```
#default deny policy on the firewall for DMZ -> LAN. (It's much for feasible to
managed your iptables policy through a script as oppose to individual commands)
~# iptables -N eth2_Out_RULE_1
~# iptables -A OUTPUT  -o eth2  -s 192.168.1.0/24  -d 172.16.1.0/24  -j
eth2_Out_RULE_1
~# iptables -A FORWARD  -o eth2  -s 192.168.1.0/24  -d 172.16.1.0/24  -j
eth2_Out_RULE_1
~# iptables -A eth2_Out_RULE_1 -j LOG --log-level info --log-prefix "RULE 1 – DENY
DMZ_LAN " --log-tcp-sequence  --log-tcp-options  --log-ip-options
~# iptables -A eth2_Out_RULE_1 -j DROP

#corresponding firewall log entry if traffic matches
Jul 26 09:32:27 src@firewall kernel: [4427415.750000] RULE 1 -- DENY DMZ_LAN
IN=eth2 OUT=eth3 SRC=192.168.1.111 DST=172.16.1.110 LEN=60 TOS=0x10 PREC=0x00
TTL=63 ID=12210 DF PROTO=TCP SPT=35863 DPT=22 SEQ=1573038414 ACK=0 WINDOW=5840
RES=0x00 SYN URGP=0 OPT (020405B40402080A00CA86FC0000000001030300)
```

Table 8

```
#honeypot syslog-ng.conf
options { sync (0); log_fifo_size (2048); create_dirs (yes); dir_group (logs); perm
(0640); dir_perm (0750);
};
source src { internal(); unix-stream("/dev/log");file("/proc/kmsg"
log_prefix("kernel: "));
};
#syslog reverse tunnel
destination tunnel {tcp ("localhost" port (65514) );
};
destination messages { file("/var/log/messages");
};
log { source(src); destination(tunnel);
};
log { source(src); destination(messages);
};
```

Appendix B – cont.

Table 9

```
#honeypot using cryptcat to transfer data
~# nc  -l -u -p 65514 | cryptcat 172.16.1.110 9999 &

#syslog server accepting data
~# cryptcat -l -p 9999 | nc -u localhost 65514 &
```

Table 10

```
#syslog reverse tunnel from server residing in /etc/inittab
log1:5:respawn:/usr/bin/ssh -nNTx -R 65514:localhost:65514 user@192.168.1.111
>/dev/null 2>&1

#mysql reverse tunnel for IDS residing in /etc/inittab
log2:5:respawn:/usr/bin/ssh -nNTx -R 63306:localhost:63306 user@192.168.1.111
>/dev/null 2>&1
```

Table 11

```
~# iptables -I INPUT -p tcp --dport 8080 -i eth0 -m state --state NEW -m recent \
--set
~# iptables -I INPUT -p tcp --dport 8080 -i eth0 -m state --state NEW -m recent \
--update --seconds 60 --hitcount 6 -j DROP
```

Table 12

```
#/etc/swatch.conf “account management”
watchfor /groupadd|useradd|adduser|chfn/
        echo
        continue
        mail=honey_admin@server,subject=Account Management

#honeypot messages.log
Aug 11 06:18:47 src@honeypot useradd[9275]: new user: name=new_user, uid=1004,
gid=100, home=/home/new_user, shell=/bin/bash

#server email
Date: Mon, 11 Aug 2006 06:18:48 -0400
From: swatch <swatch@honeypot.mydomain>
Message-Id: <200608141018.k7EAImDA009276@honey.mydomain>
To: user@server.mydomain
Subject: Account Management

Aug 11 06:18:47 src@honeypot useradd[9275]: new user: name=new_user, uid=1004,
gid=100, home=/home/new_user, shell=/bin/bash
```

Appendix C – The Analysis

Table 13

```
#command-line syntax used with p0f
~# p0f -u honey -t -C -iany -o /var/log/p0f.log -d

#p0f.log
<Mon Jul  3 03:22:54 2006> 24.211.x.x:4658 - Windows XP Pro SP1, 2000 SP3 ->
192.168.1.111:1433 (distance 17, link: ethernet/modem)
```

Table 14

```
mysql> select sig_name, count(sig_name) as cnt from acid_event group by
sig_name order by cnt desc;
+-----------------------------------------------------------+-----+
| sig_name                                                  | cnt |
+-----------------------------------------------------------+-----+
| MS-SQL Worm propagation attempt                           | 413 |
| asn1http MS04-007 exploit shellcode                       | 152 |
| WEB-MISC Phorecast remote code execution attempt          |  26 |
| WEB-PHP remote include path                               |  20 |
| PHP remote file include exploit attempt                   |  18 |
| MYSQL 4.0 root login attempt                              |  13 |
| SNMP request udp                                          |  10 |
| EXPLOIT WINS overflow attempt                             |   8 |
| ATTACK-RESPONSES Microsoft cmd.exe banner                 |   8 |
| WEB-CGI calendar access                                   |   7 |
| Sasser FTP exploit attempt                                |   6 |
| SNMP public access udp                                    |   5 |
| MS-SQL xp_cmdshell - program execution                    |   4 |
| EXPLOIT WebDav ntdll.dll (rs_iis)                         |   3 |
| WEB-MISC WebDAV search access                             |   3 |
| WEB-FRONTPAGE Chunked Transfer-Encoding Post (MS03-051)   |   3 |
| WEB-FRONTPAGE rad fp30reg.dll access                      |   3 |
| WEB-MISC Chunked-Encoding transfer attempt                |   3 |
| WEB-MISC http directory traversal                         |   2 |
| MS-SQL Worm propagation attempt OUTBOUND                  |   2 |
| RPC STATD UDP stat mon_name format string exploit attempt |   2 |
| TFTP Get                                                  |   2 |
| WEB-IIS cmd.exe access                                    |   2 |
| Bagle.B-J FTP Download URL                                |   1 |
| WEB-PHP calendar.php access                               |   1 |
| MS-SQL version overflow attempt                           |   1 |
| RPC portmap status request UDP                            |   1 |
| BACKDOOR DoomJuice file upload attempt                    |   1 |
| WEB-MISC bad HTTP/1.1 request, Potentially worm attack    |   1 |
| BACKDOOR mydoom.a backdoor upload/execute attempt         |   1 |
| Bagle.B-J Backdoor Command                                |   1 |
+-----------------------------------------------------------+----
```

Appendix C – cont.

Table 15

```
alert udp $EXTERNAL_NET any -> $HOME_NET 1434 (msg:"MS-SQL Worm propagation attempt";
content:"|04|"; depth:1; content:"|81 F1 03 01 04 9B 81 F1 01|"; content:"sock";
content:"send"; reference:bugtraq,5310; reference:bugtraq,5311; reference:cve,2002-0649;
reference:nessus,11214; reference:url,vil.nai.com/vil/content/v_99992.htm; classtype:misc-
attack; sid:2003; rev:8;)

11:59:41.280888 IP 220.x.x.x.1607 > 192.168.1.111.1434: UDP, length: 376
        0x0000:  0009 5b0a 1851 0050 bfb6 e68a 0800 4500  ..[..Q.P......E.
        0x0010:  0194 8771 0000 6711 677c dcbd c496 c0a8  ...q..g.g|......
        0x0020:  016f 0647 059a 0180 cb98 0401 0101 0101  .o.G............
        0x0030:  0101 0101 0101 0101 0101 0101 0101 0101  ................
        0x0040:  0101 0101 0101 0101 0101 0101 0101 0101  ................
        0x0050:  0101 0101 0101 0101 0101 0101 0101 0101  ................
        0x0060:  0101 0101 0101 0101 0101 0101 0101 0101  ................
        0x0070:  0101 0101 0101 0101 0101 0101 0101 0101  ................
        0x0080:  0101 0101 0101 0101 0101 01dc c9b0 42eb  ..............B.
        0x0090:  0e01 0101 0101 0101 70ae 4201 70ae 4290  ........p.B.p.B.
        0x00a0:  9090 9090 9090 9068 dcc9 b042 b801 0101  .......h...B....
        0x00b0:  0131 c9b1 1850 e2fd 3501 0101 0550 89e5  .1...P..5....P..
        0x00c0:  5168 2e64 6c6c 6865 6c33 3268 6b65 726e  Qh.dllhel32hkern
        0x00d0:  5168 6f75 6e74 6869 636b 4368 4765 7454  QhounthickChGetT
        0x00e0:  66b9 6c6c 5168 3332 2e64 6877 7332 5f66  f.llQh32.dhws2_f
        0x00f0:  b965 7451 6873 6f63 6b66 b974 6f51 6873  .etQhsockf.toQhs
        0x0100:  656e 64be 1810 ae42 8d45 d450 ff16 508d  end....B.E.P..P.
        0x0110:  45e0 508d 45f0 50ff 1650 be10 10ae 428b  E.P.E.P..P....B.
        0x0120:  1e8b 033d 558b ec51 7405 be1c 10ae 42ff  ...=U..Qt.....B.
        0x0130:  16ff d031 c951 5150 81f1 0301 049b 81f1  ...1.QQP........
        0x0140:  0101 0101 518d 45cc 508b 45c0 50ff 166a  ....Q.E.P.E.P..j
        0x0150:  116a 026a 02ff d050 8d45 c450 8b45 c050  .j.j...P.E.P.E.P
        0x0160:  ff16 89c6 09db 81f3 3c61 d9ff 8b45 b48d  ........<a...E..
        0x0170:  0c40 8d14 88c1 e204 01c2 c1e2 0829 c28d  .@...........)..
        0x0180:  0490 01d8 8945 b46a 108d 45b0 5031 c951  .....E.j..E.P1.Q
        0x0190:  6681 f178 0151 8d45 0350 8b45 ac50 ffd6  f..x.Q.E.P.E.P..
        0x01a0:  ebca
```

Table 16

```
alert tcp any any -> any $HTTP_PORTS (msg:"asn1http MS04-07 exploit shellcode";
content:"QUFBQUFBQQMAI"; flowbits:isset,asn1http; classtype:attempted-admin;
reference:url,isc.sans.org/diary.php?date=2005-06-05; sid:1000291; rev:1;)
```

Appendix C – cont.

Table 17

```
#script used to roll up and restart daily capture files
#!/bin/bash
#rotate and compress daily honeypot dump file
logdir=/var/log
now=$(date --utc +%F.%R.%S)
#stop currently running dump
kill -9 `cat /var/run/honeydump.pid`
#compress by date
tar -cvzf $logdir/honey.dump.tar.gz $logdir/honey.dump
mv $logdir/honey.dump.tar.gz $logdir/honey_$now.tar.gz
#start the dump back up and record the process ID (PID)
tcpdump -w $logdir/honey.dump –ieth0 -nnXx -s0 &
echo $! > /var/run/honeydump.pid
```

Table 18

```
alert udp $HOME_NET any -> any 53 (msg:"Possible Rbot/Sdbot DNS Lookup";
content:"|xx xxxx xx04 646e 6970 036e 6574|"; threshold: type limit, track by_src,
count 1, seconds 60; classtype:trojan-activity; sid: 3000011; rev:1;)
```

Table 19

```
alert tcp $HOME_NET any -> any any (msg: "Possible Rbot/Sdbot Infection"; flow:
established; content:"NICK|20|USA"; content:"USER"; content:"JOIN|2023 23|";
content:”|2323|"; sid:3000014;)
```

Table 20

```
Oct  8 10:12:39 mal MSWinEventLog       1       Security        91      Sun Oct 08
10:12:34 2006        592     Security        mal1    User    Success AuditMAL
Detailed Tracking              A new process has been created:     New Process ID:
212     Image File Name: C:\WINDOWS\system32\msnmsdnup.exe     Creator Process ID:
1380     User Name: mal1     Domain: MAL     Logon ID: (0x0,0xD63A)          90
```