SANS | GIAC CERTIFICATIONS

WHITE PAPER

DNS Sinkhole

Guy Bruneau

DNS Sinkhole

GIAC (GCIH) Gold Certification

Author: Guy Bruneau, seeker@whitehats.ca

Advisor: Rick Wanner

Accepted: August 7, 2010

Abstract

This paper describes the architecture and configuration of a complete Domain Name Services (DNS) sinkhole system based on open-source software. The DNS sinkhole can be used to provide detection and prevention of malicious and unwanted activity occurring between organization computer systems and the Internet. The system is inexpensive, effective, scalable and easy to maintain.

1 Introduction

Why use a DNS sinkhole? The Domain Name Service is a core service used to access the Internet, so control of DNS equates to at least partial control of Internet traffic. By intercepting outbound DNS requests attempting to access known malicious domains, such as botnets, spyware, and fake antivirus, an organization can control the response and prevent organization computers from connecting to these domains. This activity prevents unwanted communications and is capable of mitigating known and unknown threats hosted on known malicious or unwanted domains.

A DNS sinkhole works by 'spoofing' the authoritative DNS servers for malicious and unwanted hosts and domains. An administrator configures the DNS forwarder for outbound Internet traffic to return false IP addresses for these known hosts and domains. When a client requests to resolve the address of such a host or domain, the sinkhole returns a non-routable address; or any address *except* the real address. This denies the client a connection to the target host.

When a new domain is added to the list, the domain falls under the direct control of the sinkhole administrator. After this moment, it is no longer possible to access the original host or domain. Using this method, compromised clients can easily be found using sinkhole logs. For real-time notification, a custom IDS signature can be written to detect and immediately alert when a client connects to the host IP provided by the DNS sinkhole.

There are many ways to identify known malicious and unwanted hosts and domains. In order to be effective, the list(s) must be constantly updated by sinkhole administrators. Open-source lists of known malware sites, adware sites, and honeynets, and information from sources like the SANS Internet Storm Center (Ludwig, 2009) can be combined with organization-specific information from DNS Resource Record (RR) queries from infected clients, analysis of malware found on compromised clients, and other sources of information exchange such as government or CERT teams.

The sinkhole can also be used to take control of known hosts and domains that are not specifically malicious, but contravene corporate policy. For example, a network acceptable use policy may stipulate that organization computers are not approved to access social networking sites. Many organizations would enforce this policy using a firewall, web proxy, or network Intrusion Prevention System. However, another option is to create a custom DNS sinkhole list and add desired social networking domains to the sinkhole. When an internal client attempts to connect to a sinkholed domain, it would be redirected to an internal website indicating the site is

blocked in accordance with corporate policy. In this manner, a DNS sinkhole can be part of a defense-in-depth strategy.

2 Problem Statement

It is the golden age of bots. As shown in Figure 1 below, botnet traffic is increasing, exploiting computer systems through a variety of infection vectors and establishing command and control channels for sale or lease to the highest bidder. Targeted attacks are commonly seen in the form of the Advanced Persistent Threat (Damballa, 2010) for espionage, state-sponsored and economic, cyber-crime, and other illegal activity. Bots employ constantly-changing tactics for obfuscating and maintaining the command and control channels that are used to manage botnet clients.

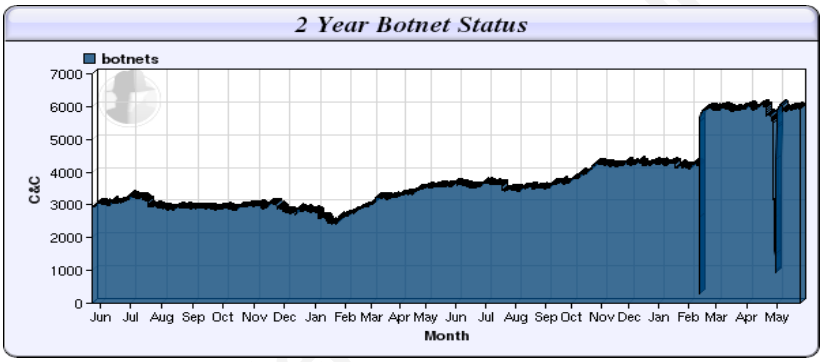

Figure 1: Botnet activity June 2008 - May 2010 (ShadowServer, 2010)

Administrators and security analysts must respond daily to malware that forces a client to unknowingly download suspicious files from sites that they have no legitimate reason to access. Enterprise security architectures typically attempt to mitigate access to such sites through the use of web filtering, firewalls, and/or Network Intrusion Prevention Systems (NIPS). Known vulnerable applications must also be patched against compromise, assuming that a patch is available. Adobe's Acrobat Reader, a common application with millions of installed clients has repeatedly been targeted for attack in the past year with various zero-day attacks.

All of these controls require updates to multiple systems in response to new threats; performing and managing these updates can be an overwhelming task in even a small organization. The management challenges are compounded by the evasion techniques such as fast flux, used by bots to avoid being blocked by constantly changing their IP(s) (Riden, 2007). Bots will also adjust protocols and ports used for communication. These techniques make it almost impossible to control malicious and unwanted sites by patching and black-listing IP addresses alone.

There is a clear opportunity to centralize and simplify the detection and prevention of bots and other unwanted traffic. Anti-malware engineers and security analysts have observed that a malicious website's fully qualified domain name is often hard-coded in malware; this permits the client to download updates or upload the data it collects while taking advantage of evasion techniques such as fast-flux. This is where a DNS sinkhole can be used to prevent access to known malware sites as part of a defense-in-depth strategy.

3 DNS Sinkhole Overview

This section describes key features, benefits and limitations to be considered when designing, testing and deploying a sinkhole.

3.1 Key Features

The DNS sinkhole presented in this paper has several key features that allow it to address many of the issues identified in the problem statement.

First, the system is simple, and for this reason it is effective at detecting and blocking malicious and unwanted traffic attempting to reach the Internet. It is capable of mitigating many different types of threats that use DNS resolution, regardless of the vulnerability that may have allowed a web re-direction or initial compromise/infection.

Administrative effort is very modest for the sinkhole – there will likely be more effort spent handling incidents identified from sinkhole data, then actually maintaining the system. Using well-known DNS zone files, administrators can easily add hosts and domains, individually or by the thousands. There is a large community of security professionals that maintains lists of known malicious and unwanted sites, and this information can be extended to include closed-source domain names if desired; those detected locally.

The DNS sinkhole is based on open-source software and is therefore inexpensive to setup and maintain. The system can be installed on commodity server hardware or on a virtual platform, further reducing capital costs.

The basic DNS sinkhole discussed in the following section provides configurable detection and blocking of malicious and unwanted hosts and domains. This system is simple to setup and maintain and requires very little administrative effort, with the caveat that it does not easily generate information useful for incident response and handling. Additional modular components can be deployed during or following a sinkhole installation to provide real-time notification of sinkhole events, and a packet log of sinkholed host traffic. These features make the sinkhole a key element of an enterprise monitoring strategy.

3.2 Types of Sinkholes

"Sink Holes are the network equivalent of a honey pot." (Green, Barry Raveendran & McPherson, Danny, 2003). ISPs have been using sinkholes for several years to help protect their customers by diverting attacks and detecting anomalous activity (McPherson, 2004). There are several different types of sinkholes based on name resolution or DNS.

The proposed solution (an internal DNS sinkhole server) works by impersonating an authoritative DNS server for malicious and unwanted domains; it basically returns private addresses (the sinkhole) for such host and domain queries. In order for this system to be effective, the list of malicious and unwanted domains must be maintained. A system using an internal DNS sinkhole server can serve an entire enterprise's needs.

Sinkholes for single platforms can be constructed using a simple hosts file. This technique has been widely available for a number of years; it too, requires ongoing administration but it is not well suited to scaling beyond a small number of hosts.

Intrusion Prevention Systems (IPSs) may provide some control over anomalous DNS packets, such as those containing an unusually short DNS time-to-live (TTL). This is somewhat effective at managing fast-flux DNS systems, but it does not work for 'normal' DNS communications with malicious or unwanted systems.

"A darknet is a portion of routed, allocated IP space in which no active services or servers reside" (Team Cymru, 2010). A darknet can be used for collecting data about anomalous network traffic. By definition, allocated, but unused, IP space should not provide listening services, and clients have no reason to communicate with such addresses. This type of traffic is quite often mass scanners, malware looking for new victims, or misconfigured hosts.

Commercial vendors also provide DNS sinkhole services; free consumer-level services are also available. One key point regarding these services is that they do not typically allow customers to view the complete host/domain repository, as this information is a competitive differentiator. For the same reason, clients may be unwillingly or unable to upload closed-source hosts or domains to third party providers (if such an interface were offered).

3.3 Benefits

There are many benefits to deploying an internal DNS sinkhole; a sinkhole is inexpensive, scalable, effective and easy to maintain. It also provides attractive benefits when compared to commercial or cloud-based DNS sinkhole offerings. Some free DNS offerings include Norton DNS

Beta (Norton DNS, 2010), ClearCloud (ClearCloud, 2010) and OpenDNS (OpenDNS, 2010) as a free service and two paid premium services that can whitelist/blacklist between 25 (basic) to 500 (enterprise) domains.

The DNS sinkhole described in this paper uses open-source software and can be implemented on virtually any commodity hardware platform with minimal capital cost.

Ongoing maintenance consists primarily of processing automated updates from one or more open-source lists of known malicious or unwanted hosts or domains. There are no fees to use these lists (unlike commercial services), and additional data sources are frequently added as list servers evolve. The lists can also be viewed in their entirety and edited if needed. Administrators can use lists to verify if hosts or domains will be blocked without active testing.

Organizations can easily integrate their own 'closed-source' sinkhole entries for hosts or domains. This ability to customize data is a significant differentiator from commercial offerings.

3.4 Limitations

Internal DNS sinkholes have a number of limitations and trade-offs that must be understood by systems administrators and analysts. All of the limitations can be managed through the use of standard operating procedures and a defense-in-depth architecture.

The primary limitation with a DNS sinkhole is that it does not actually detect malware, block malware installation, or eradicate malware; it simply detects indicators of malware. These indicators require follow-up investigation by security analysts to determine if a target has in fact been compromised. DNS sinkhole logs can provide some indications of compromise; these are discussed in Use Cases presented later in the paper.

The sinkhole also requires that malware actually use the organization's DNS services; malware using IP addresses or possessing a built-in DNS server will never use the sinkhole. This vulnerability can be mitigated by the well-known best practice of allowing outbound DNS queries only from authorized DNS servers. Perimeter firewalls should be configured to block all other outbound DNS queries (TCP/UDP 53). This will ensure that all DNS traffic is processed by the organization's DNS servers (i.e. the sinkhole). It is also important that DNS sinkholes be protected from external (i.e. Internet-based) queries, so that attackers cannot recursively determine that their command and controls systems have been mitigated by a DNS sinkhole.

Due to the sinkhole's ability to block entire top-level domains, administrators must take care to ensure that false positives (i.e. non-malicious domains) and errors are not introduced to the sinkhole configuration. A simple error, such as entering '.ca' (Canada) instead of '.cc' (Cocos

Islands, a commonly-used domain for malicious code), would result in all communications to top level Canadian domains with a '.ca' being blocked until the error was corrected. The operational impact of this error could be extremely serious. This limitation can be overcome with quality control procedures.

Finally, administrators must remain aware that DNS records have time-to-live (TTL) settings. It may be necessary to configure TTLs with short values (i.e. 10 minutes) in order to be able to rapidly propagate changes, and prevent internal clients from caching sinkhole records for extended periods of time.

4 DNS Sinkhole Architecture

This section describes three DNS reference architectures and sample command-and-control flows to illustrate how a sinkhole works. The architectures include a standard DNS architecture with no sinkhole, a basic sinkhole, and an advanced sinkhole with intrusion detection and network forensic capabilities.

4.1 Normal DNS Flow

The components in a typical DNS flow for Internet-based services are shown in the following figure. This model is typical for most organizations that operate their own DNS server(s).

The figure illustrates the DNS flows that may occur when an attacker compromises a client to establish a botnet. "A botnet is a collection of computers, connected to the internet, that interact to accomplish some distributed task." (ShadowServer, 2010). The term botnet usually refers to systems used for illegal purposes and are assimilated by a bot herder into a network to be used for malicious purposes. The most common botnets are typically used for DDoS, keylogging, click fraud, warez, spam, identity theft, traffic sniffers, and e-mail storms.

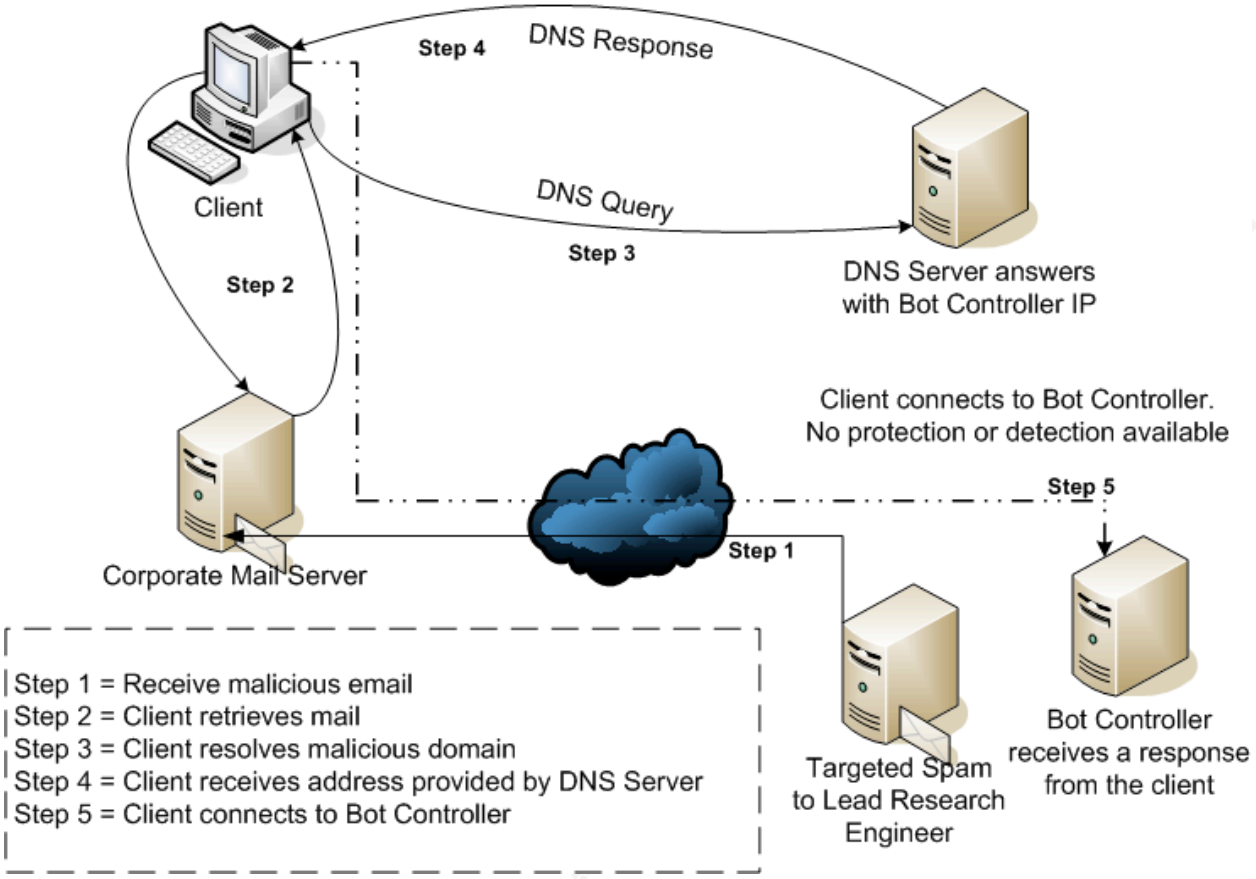

Figure 2: DNS flow without sinkhole

4.2 DNS Sinkhole Flow – Basic

A basic DNS sinkhole deployment includes the following components:

1. Commodity server
2. Linux OS with BIND
3. List(s) of malicious and unwanted hosts, domains and Top Level Domains
4. Custom configuration and update scripts

The figure illustrates the DNS flows that occur when an attacker compromises a client and attempts to contact a botnet. Note that in this situation the compromise *still occurs*.

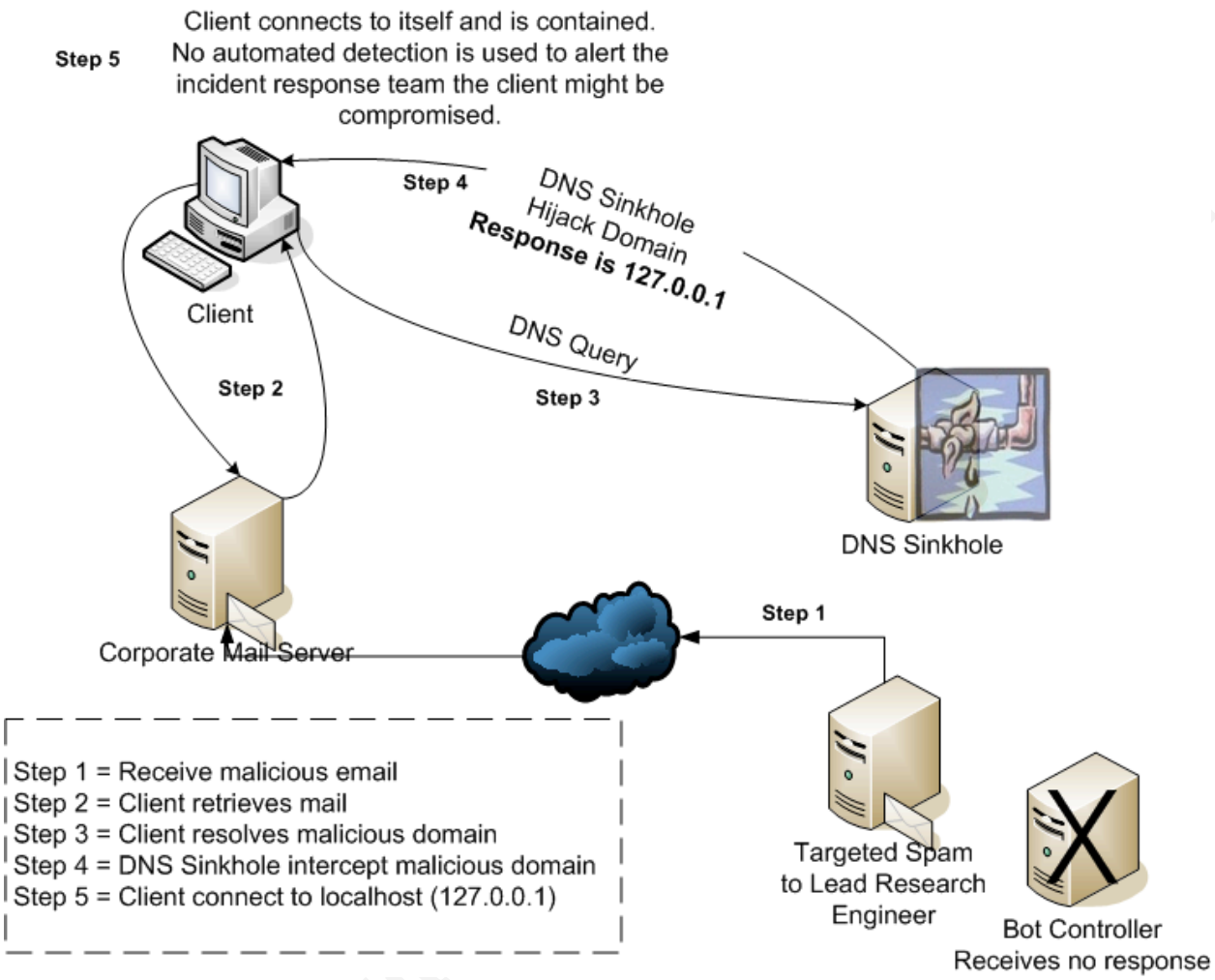

Figure 3: DNS sinkhole with basic functionality

The DNS sinkhole does not allow the domain to be resolved by the domain's authoritative owner. Instead, the DNS sinkhole intercepts the DNS request and responds with an authoritative answer configured by the organization.

With the basic sinkhole functionality, the malware on the compromised client attempts to initiate a connection to a system hosted on 1010101.org, a known malicious domain configured in the DNS sinkhole. The request is sent to the sinkhole which in turn responds with an IP of 127.0.0.1, forcing the client to connect to itself instead of the true IP. The client is unable to contact the malicious site and the command and control connection with the botnet is never established. The bot herder is unaware that the compromise has occurred.

Although the client has been prevented from connecting with the bot server, the incident response process is not yet complete. Preparation, detection, and partial containment have been

performed. Containment is partial because the compromised computer may still attempt to attack internal computers. The organization administrator or Incident Response Team has not been alerted to the compromise, and they must perform additional analysis, eradication, recovery, and lessons learned. The next DNS sinkhole model includes advance detection capabilities to provide real-time notification and additional data for analysis.

4.3 DNS Sinkhole Flow - Advanced

An advanced DNS sinkhole deployment includes the basic sinkhole components, plus the following:

1. Socat service emulating one or more listening services for network forensic analysis
2. Snort IDS with customized sinkhole signatures for real-time alerting
3. Scripts for scheduled reporting

The figure illustrates the DNS flows that occur when an attacker compromises a client and attempts to establish a botnet. Real-time detection and notification are provided to the Incident Response Team.

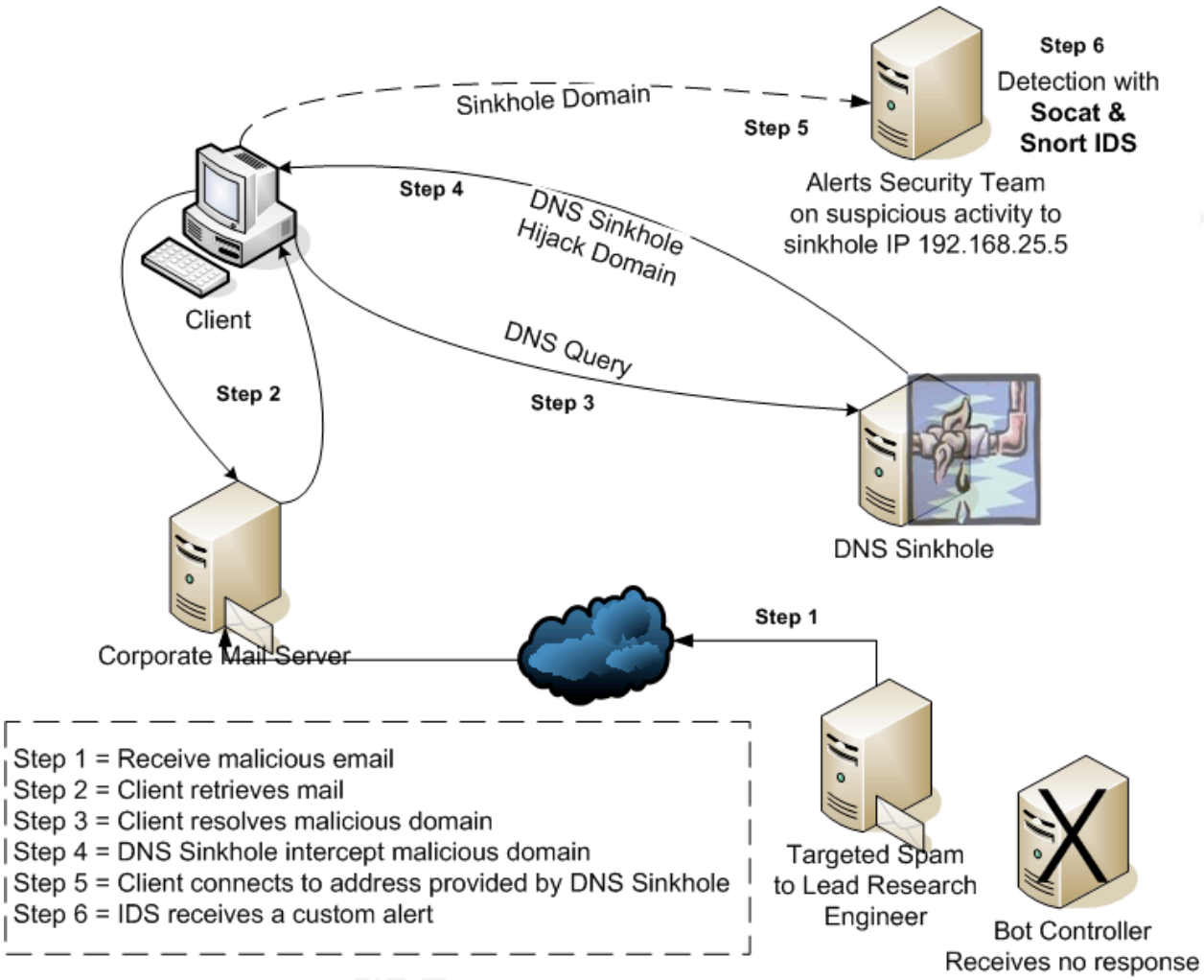

Figure 4: DNS sinkhole with advanced functionality

The first four steps are the same as with basic sinkhole functionality, except that the sinkhole provides the client with a valid internal address (instead of 127.0.0.1) in response to the request for the bot controller. The valid internal address is a re-direct to an administrator-controlled IP address (step 5). Since most outbound command and control traffic is web-based, the listening service is configured on several common HTTP ports.

The most common HTTP ports that should be monitored are port 80, 8000, and 8080 but administrators should tailor the list to include the services allowed through the organizations firewall. HTTPS can also be monitored on port 443 and 8443 by configuring a valid SSL certificate. An advanced configuration could provide a different IP address for each DNS sinkhole group, thereby allowing the administrator to send different types of traffic (i.e. known malicious, known adware, and closed-source hosts) to different IP addresses to speed up analysis and simplify reporting.

An alternative to installing and maintaining an actual webserver is to configure a Linux server using an application called Socat (Rieger, 2010). Socat can be used to listen on multiple ports and capture the initial exchange with a sniffer. Sguil (Visscher, 2008) and tcpdump (tcpdump.org, 2010) work well, or a real-time IDS such as Snort (Roesch, 2010) may be capable of performing packet capture and detection in a single system.

The easiest thing to do is to write an IDS signature for each sinkhole group in use; one IP per group. The IDS will detect each time a client is redirected to one of the sinkhole lists (i.e. known malicious, known adware, and closed-source hosts), to assist in the detection of potentially compromised clients.

Netflow is another real-time detection alternative. However, the name of the target domain will not be provided with this method, which is a considerable drawback. There is also no real-time alerting without the use of other tools. Netflow will build statistics on the number of hosts redirected to the sinkhole over a period of time. Netflow traffic can be generated with a Linux server using softflow (Miller, 2006) and analyzed using a Netflow analyzer such as ManageEngine NetFlowAnalyzer (ManageEngine, 2010).

4.4 Scalability and Performance

BIND DNS is a highly scalable system where the sinkhole can be deployed for organizations with tens of thousands of users. Typical rules of DNS architecture apply, so multiple redundant sinkhole servers can be provisioned, if needed, for high availability and high performance.

A pair of the DNS sinkholes discussed in this paper has supported an organization with more than 15,000 users for nearly a year with no unplanned downtime or performance issues. The systems were tested in various configurations. A single DNS BIND server with 1 GB of RAM performed well loaded with over 20,000 domains.

5 Use Cases

When the client is redirected to the sinkhole, it is for one of two possible reasons: 1) it is already compromised and is attempting to contact a bot controller; or 2) it has been simply redirected (referred) through a website link to the sinkhole. In the second case, the site referred to the sinkhole is a known malicious site or it does not meet corporate policy.

The number of alerts in a large organization might range from a handful to several hundred per day, depending on the number of clients and pattern of activity.

Two different sinkhole use cases are illustrated below: drive-by downloads, and command-and-control channels.

5.1 Blocking Drive-by Downloads

This type of redirect often happens when a client accesses a legitimate website that an attacker has secretly inserted with a malicious hidden link which forces the client to download and execute malicious code without the user's knowledge. This example shows a client redirected from website `http://thecookingcritic.com` to a known malicious site `(zettapetta.com)`. The site `zettapetta.com` is currently in the DNS sinkhole.

The sinkhole's Socat packet log shows the client attempting to get a malicious javascript file, js.php. The client was referred from thecookingcritic.com. The packet log indicates HTTP/1.0 200 (OK) because the Socat server successfully received the request; in fact no file was returned to the client. The malicious download was blocked and no connection was established to zettapetta.com.

```
GET /js.php HTTP/1.1
Accept: */*
Referer: http://thecookingcritic.com/category/by-cookbook/william-
sonoma-pasta-sauces/
Accept-Language: en-ca
UA-CPU: x86
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET
CLR 1.1.4322; .NET CLR 2.0.50727; InfoPath.1)
Host: zettapetta.com
Connection: Keep-Alive

HTTP/1.0 200
Content-Type: text/plain
```

5.2 Blocking C&C Channels

The following example shows a client that was not redirected, but attempted to directly access a C&C bot controller hosted at `hjwbxhqr.com`. It also shows there is no **Referer** indicating a direct connection to this domain. This is a good indicator the client is compromised and the bot is attempting to contact the controller to report in, get updates or upload data.

Since domain `hjwbxhqr.com` is in the sinkhole, the client is unable to reach the bot controller. By preventing access to the site, the Incident Response team has more time to examine the host through host and network forensics. Ultimately the client may need to be rebuilt from clean media.

```
GET /win-
xp/controller.php?action=bot&entity_list=1272385402,1272401299,1272
965869,1272890566,1272974084,1272895110,1272890923,1272966523,12729
84663,1272894972,1272890645,1272891037,1272890722,1272969471,127298
4463,1272967202,1273140516,1275327332&uid=8&first=0&guid=2043777331
&v=15&rnd=11987634 HTTP/1.1
Host: hjwbxhqr.com

HTTP/1.0 200
Content-Type: text/plain
```

6 Deploying and Maintaining a DNS Sinkhole

This section describes activities for deploying and maintaining components in a DNS sinkhole, including Bind, external lists, custom lists, Socat, IDS, and reporting.

6.1 Configure BIND

The primary goal is to automate the process of downloading and parsing already available non-commercial domain lists to populate the sinkhole. To automate the process, a shell script was created to download the information from these lists, aggregate them, and load a single list into the BIND server. See Appendix D.

There are two ways of taking control of a domain: a system host file (in Unix `/etc/hosts`, or in Windows `C:\WINDOWS\system32\drivers\etc\hosts`) and with a DNS server controlled by an organization. This paper uses Unix BIND for DNS. There is an example of how to set up a DNS sinkhole using Windows 2003 available at: http://www.malwaredomains.com/wordpress/?page_id=6.

The key to controlling a site, a domain or a top-level domain (TLD) such as an entire country code, is the configuration files. Examples of a single domain Start of Authority (SOA) file as well as a SOA for an entire domain or TLD are provided.

DNS BIND'S main configuration file is located in the `/etc` directory and is called `named.conf` (see Appendix A). The configuration file tells the ISC BIND named service how to start in the options section, what to log in the logging section, and what zones to load. This file shows a standard configuration for a caching server. This configuration shows the server is also acting as a forwarder with the forwarder option. The forwarder option can contain as many DNS servers as needed.

The named.conf file should be edited to reflect organization-specific site settings. The following settings should be reviewed and adjusted as necessary: allow-transfer, allow-recursion, and forwarders.

Appendixes B and C shows an example of a single and wildcard configuration located in the `/var/named/sinkhole` directory. The file `client.nowhere` and `domain.nowhere` must be configured exactly as shown in the appendixes. The Name Server (NS) and Address (A) records are an important part of the DNS sinkhole configuration (DNS Made Easy, 2010). Both of these are composed of a Name, Time to Live, Type (NS or A) and Data (@ for NS and IP for A record). The A record is the IP used to connect the client to the sinkhole.

6.1.1 BIND Zone File SOA Configuration for a Single Host

In this example, `www.sans.org` is entered as a record in the sinkhole and configured with IP address 192.168.25.5. If a client attempts to connect to `www.sans.org`, the answer the client will get is IP 192.168.25.5 instead of the true address maintained by SANS, 66.35.45.201. This will prevent a corporate client from reaching the actual site. Using methods discussed earlier, the administrator or security analyst can investigate if this was a redirect or the client has been compromised and attempting to contact a C&C server.

Since only a single host has been configured, other hosts and child domains of `sans.org` are not sinkholed. For example, if the client attempted to access `isc.sans.org`, the address would be resolved normally and the connection would be permitted.

```
$TTL  600
@                IN SOA     localhost root.localhost. (
                            1           ; serial
                            3H          ; refresh
                            15M         ; retry
                            1W          ; expiry
                            1D )        ; minimum

                 1H IN NS         @
                 1H IN A          192.168.25.5
```

6.1.2 BIND Zone File SOA Configuration Multiple Domains (Wildcard)

A second method is the addition of a second A record with a wildcard using a star (*). Both A records are needed to ensure the entire domain and sub-domains are included. So, if the domain entered as a record is `sans.org`, this will include subdomains such as but not limited to `www.sans.org`, `isc.sans.org`, and `testing.sans.org`.

Another option is to sinkhole a top-level domain (TLD) like an entire country code (IANA, 2010). If a country code such as `.ca` is entered instead of a sub-domain such as `.sans.org`, any domains ending with `.ca` are redirected to the sinkhole.

Be extremely cautious when wild-carding entire top-level domains. Depending on the domain, the DNS sinkhole could block millions of hosts.

```
$TTL  600
@               IN SOA     localhost root.localhost. (
                           1          ; serial
                           3H         ; refresh
                           15M        ; retry
                           1W         ; expiry
                           1D )       ; minimum

                1H IN NS         @
                1H IN A          192.168.25.5
*               1H IN A          192.168.25.5
```

6.1.3 Master Zone Configuration

Below is an example on how to add a single domain called `our.sink.com` in a sinkhole. The single site sinkhole file is called `site_specific_sinkhole.conf` and contains the zone information for the target domain. This file is located is in the `/var/named` directory.

```
zone "our.sink.com" IN { type master; file
"/var/named/sinkhole/ client.nowhere "; };
```

Below is an example of adding an entire domain called `sans.org` to a sinkhole. By wildcarding the domain `*.sans.org`, any child domain ending with `.sans.org`, e.g. `isc.sans.org` and `rr.sans.org`, would be redirected to the sinkhole. This file is located in the `/var/named` directory. The wildcard sinkhole file is called `entire_domain_sinkhole.conf`.

```
zone "sans.org" IN { type master; file
"/var/named/sinkhole/domain.nowhere"; };
zone "ca" IN { type master; file
"/var/named/sinkhole/domain.nowhere"; };
zone "info" IN { type master; file
"/var/named/sinkhole/domain.nowhere"; };
```

A test against TLD `.ca` using any combination of names will return the following output:

```
guy@gate:$ nslookup testing.ca
Server:        192.168.25.50
Address:       192.168.25.50#53

Name:   testing.ca
Address: 192.168.25.5
```

6.2 Download and Synchronize Lists

The domains listed here must be vetted to ensure they do not include extranets or partners. The Time-to-Live (TTL) assigned to all domains in the sinkhole is very important. In order to quickly remove a domain entered in error, it is a good idea to assign a short TTL (10 minutes to 1 hour) to expire the cache information sooner rather than later. If the site has a large DNS server farm, all the servers will need to expire their cache or have it manually cleared before the true site is accessible. It is important to remember only internal clients attempting to go to the Internet will be affected by the sinkhole. However, Internet users accessing the organization's Internet servers are not.

The following list of sites could be utilized to populate a sinkhole. Some of the lists are reserved for non-commercial use only and have explicit postings stating this restriction (Glosser, 2010):

```
http://pgl.yoyo.org/adservers/serverlist.php?hostformat=;showintro=0
http://support.it-mate.co.uk/downloads/HOSTS.txt
http://www.malwaredomains.com/files/justdomains
http://www.abuse.ch/zeustracker/blocklist.php?download=domainblocklist
http://mtc.sri.com/live_data/malware_dns/
http://www.malwarepatrol.net/cgi/submit?action=list_bind
```

Each of these lists is updated regularly, sometimes several times a day.

While using several lists helps block malicious domains, the script needs to be smart enough to remove duplicate domains and keep a single copy of each domain before adding them to the DNS sinkhole. Since it is possible (and valid) to sinkhole a TLD, the script must ensure that any sub-domains of a listed TLD are removed in the final list.

6.3 Configuring Custom Lists

One of the key features of an internal DNS sinkhole is the ability to create custom lists. Each time a client is discovered accessing a host or domain that is malicious or unwanted because it contravenes corporate policy, it can be added to a custom list to prevent all other network clients from connecting to that host or domain.

The following example shows a client accessing site `ahterbok.com` with the command `GET /web.php?i=1`. A first look would indicate a web.php script is accessed by the client; however, the server is serving a piece of malware code with a filename of `irfvjtg.exe`. Therefore, this

domain `ahterbok.com` should be added to one of the custom lists to prevent other organization clients from potentially being infected.

URL analysis tool	Result
Firefox	Malware site
G-Data	Malware site
Google Safebrowsing	Malware site
Opera	Clean site
ParetoLogic	Clean site
Phishtank	Clean site

Additional information
Normalized URL: http://ahterbok.com/
URL MD5: dbff8f3da7975d10a9abd1739edd124c
Content-Type: text/plain

Figure 5: ahterbok.com VirusTotal URL Analysis

```
GET /web.php?i=1 HTTP/1.1
User-Agent: Mozilla/4.0 (Windows XP 5.1) Java/1.6.0_16
Host: ahterbook.com
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive

HTTP/1.1 200 OK
Server: nginx/0.7.67
Date: Mon, 11 Oct 2010 20:29:47 GMT
Content-Type: application/octet-stream
Connection: keep-alive
X-Powered-By: PHP/5.2.6-1+lenny9
Set-Cookie: PHPSESSID=b0274a42b5cdcd1a9a55d71702d21bba; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0,
pre-check=0
Pragma: no-cache
Content-Length: 26112
Content-Disposition: attachment; filename=irfvjtg.exe

MZ.....@.....x.....
.....!This program cannot be run in DOS mode.
```

6.4 Configuring Socat

There are various ways to capture sinkhole traffic, including web servers, but flexibility and simplicity are the key requirements. Multiple instances of SOcket CAT (Socat) can be set it up to listen to several ports such as TCP 80, 7070, 8000, 8080 on a single IP address. Several of these listeners on multiple IP addresses can be setup to differentiate each sinkhole list as an aid for reporting and for incident response.

Socat can easily be configured to simulate a simple web server or other services if needed. In the case of a web server simulation, Socat performs a 3-way handshake to capture the data the client is attempting to GET or POST. This log data is then used to assist with incident response.

Combined with a sniffer or a custom IDS signature, the sinkhole will provide the ability to alert and capture in real-time the domain a client is attempting to initiate a communications with.

```
socat TCP-LISTEN:80,bind=192.168.25.5,fork,reuseaddr,crlf
SYSTEM:"echo HTTP/1.0 200; echo Content-Type\:text/plain;" &
```

Figure 6: Socat port 80 emulation

6.5 Configuring IDS

The following Snort signature will alert on the sinkhole traffic and capture the domain contacted by the client.

```
alert tcp $HOME_NET any -> 192.168.25.5 any ( msg: "Sinkhole
Site Specific List"; flags: A+; content:"Host:"; classtype:
policy-violation; sid:2010001; rev:1;)
```

This Snort signature combined with a Socat listener will alert each time a client is redirected to the sinkhole based on the A record IP address of 192.168.25.5. The signature includes a combination of TCP flags (Acknowledgement flag plus any others) to capture the GET or POST to be visible in the alert message.

Because this signature is looking for a 3-way handshake, it has limitations. If a client is redirected to a TCP port that Socat is not listening for, no alert will be sent to the incident response team. It is important to configure a suitable range of ports based on the firewall policy for outbound Internet traffic.

Organizations using Sguil as part of the IDS framework, can review the full packet stream along with the alert. This is useful to investigate if a host is infected with malware or was simply redirected from another site.

Beaconing may be indicated by the same event triggered at regular intervals (however not all beaconing occurs at repeatable intervals). Beaconing is a strong indicator the client has been compromised and is trying to contact a Command & Control (C&C) server. It should be immediately investigated and rebuilt from clean media if compromised.

6.6 Configure Reporting

A shell script with httpry can be used to generate a daily summary report of all packets captured that were redirected to Socat or other web listeners. The report shows the number of times each client was sinkholed for each unique website Uniform Resource Locator (URL).

It is important to monitor which clients are connecting to the sinkhole and investigate the root cause of the traffic. Depending on the lists maintained by the organization, the sinkhole reports could contain indicators of malware, adware, spyware, or policy violations. If an organization uses a custom list to monitor previous infections, it may be alerted to infections that were contained but not completely eradicated.

A copy of the shell script used to generate this report is included in Appendix F.

```
        Daily DNS Sinkhole report for the 20100816
Count   Source          Destination     HTTP    Website

1       192.168.25.28   192.168.25.6    GET  static.xvideos.com
1       192.168.25.28   192.168.25.6    GET  www.xvideos.com
2       192.168.25.26   192.168.25.6    GET  ad.adriver.ru
5       192.168.25.25   192.168.25.6    GET  finderwid.org
6       192.168.25.28   192.168.25.6    GET  bar.hub.cc
```

Figure 7: Sinkhole Summary Report

7 Conclusion

This paper illustrates how to use BIND to setup a simple, yet very effective, DNS sinkhole which provides detection and prevention of malicious and unwanted activity, occurring between organization computer systems and the Internet. Combining a DNS sinkhole with other defenses increases the ability to detect and prevent compromised assets from communicating with malicious or unwanted hosts or domains on the Internet. The sinkhole is OS and protocol-independent which can be configured to provide scheduled reporting and real-time notification for improved incident response.

A full capture and analysis of all organizational DNS records could help identify threats that might be deeply entrenched in the organization. For further information, see the techniques illustrated in Andrew Hunt's paper "Visualizing the Hosting Patterns of Modern Cybercriminals" (Hunt, 2010).

As part of this research project, anyone who wishes to test and install a ready-to-use DNS Sinkhole, the author has made available a hardened DNS Sinkhole ISO freely available at: `http://www.whitehats.ca`.

8 Appendix A: BIND Configuration

This is a modified version of `/etc/named.conf` with the addition of forwarders, logging and custom sinkhole zone files.

named.conf

```
options {
       directory "/var/named";
        // version statement - inhibited for security
        version "my own";
        // optional - disables all transfers
        // slaves allowed in zone clauses
        allow-transfer {"none";};
        allow-recursion {192.168.1.0/24; localhost;};
        forwarders { 192.168.20.5; 4.2.2.1; };

      /*
       * If there is a firewall between you and nameservers you want
       * to talk to, you might need to uncomment the query-source
       * directive below.  Previous versions of BIND always asked
       * questions using port 53, but BIND 8.1 uses an unprivileged
       * port by default.
       */

      // query-source address * port 53;
};

//
// log to /var/log/named/example.log all events from
// info UP in severity (no debug)
// defaults to use 3 files in rotation
// BIND 9.x parses the whole file before using the log
// failure messages up to this point are in (syslog)
// typically /var/log/messages
//

logging {

      channel default_syslog {
            // Send most of the named messages to syslog.
            syslog local2;
            severity debug;
      };

      channel audit_log {
            // Send the security related messages to a separate file.
            file "/var/log/named/named.log";
            severity debug;
```

```
            print-time yes;
      };

      channel query_log {
            // Send the security related messages to a separate file.
            file "/var/log/named/query.log";
            severity debug;
            print-time yes;
      };

      category default { default_syslog; };
      category general { default_syslog; };
      category security { audit_log; default_syslog; };
      category config { default_syslog; };
      category resolver { audit_log; };
      category xfer-in { audit_log; };
      category xfer-out { audit_log; };
      category notify { audit_log; };
      category client { audit_log; };
      category network { audit_log; };
      category update { audit_log; };
      category queries { query_log; };
      category lame-servers { audit_log; };

};
//
// a caching only nameserver config
//
zone "." IN {
      type hint;
      file "caching-example/named.root";
};

include "/var/named/site_specific_sinkhole.conf";
include "/var/named/entire_domain_sinkhole.conf";
include "/var/named/malware_zone.dns";

zone "localhost" IN {
      type master;
      file "caching-example/localhost.zone";
      allow-update { none; };
};

zone "0.0.127.in-addr.arpa" IN {
      type master;
      file "caching-example/named.local";
      allow-update { none; };
};
```

9 Appendix B: Zone File to Sinkhole a Single Domain

This zone file is used to assign the IP address and the Time-to-Live (TTL) to a single domain. This configuration uses a 1 hour (1H) for the TTL and responds with IP 192.168.25.5.

```
client.nowhere

$TTL    600
                IN SOA          localhost root.localhost. (
                                42          ; serial
                                3H          ; refresh
                                15M         ; retry
                                1W          ; expiry
                                1D )        ; minimum

                1H IN NS        @
                1H IN A         192.168.25.5
```

10 Appendix C: Zone file to Sinkhole Multiple Domains (Wildcard)

This zone file is used to assign the IP address and the Time-to-Live (TTL) to a domain and all sub-domains. This configuration uses a 1 hour (1H) for the TTL and responds with IP 192.168.25.5.

```
domain.nowhere

$TTL  600
@               IN SOA     localhost root.localhost. (
                           1          ; serial
                           3H         ; refresh
                           15M        ; retry
                           1W         ; expiry
                           1D )       ; minimum

                1H IN NS         @
                1H IN A          192.168.2.5
*               1H IN A          192.168.25.5
```

11 Appendix D: Download and Parser Script

This is the main script to download, parse and build a functional list to be loaded in BIND.

sinkhole_parser.sh

```
#!/bin/sh

# Guy Bruneau, seeker@whitehats.ca
# Version 1.0
# 15 Jan 2010
#
################################################################################
## This script is used to process malware site lists to create a single master
## list to use for a DNS sinkhole.
## Using this script will combine various malware domains into a single file
## name malware_zone.dns in the /var/named directory.
## The script is run at the command line and has built-in check to make sure
## there are no errors before the file is reloaded into the DNS server.
################################################################################

# Today's date

TODAYDATE="`/bin/date '+%Y-%m-%d_%T'`"

fin=n
while [ $fin=n ]
do

   clear
   echo
   echo "                          DNS Sinkhole Menu"
   echo
   echo "                     A. Manually add single domain to sinkhole"
   echo "                     D. Download sinkhole udpates"
   echo "                     T. Testing new zone file for errors"
   echo "                     F. Empty PowerDNS database of all its records"
   echo "                     R. Zone check failed, restore and exit"
   echo "                     B. Zone file is good, load it in BIND and exit"
   echo "                     P. Zone file is good, load PowerDNS and exit"
   echo "                     E. Exit script"
   echo
   echo "                     What is your choice?"
   read option


      case "$option" in
      a|A) clear

      echo "This script is used to enter a single domain in the sinkhole"
      echo "If using BIND DNS for your sinkhole, make sure these lists"
      echo "are in /var/named.conf before proceeding"
      echo
```

```
      echo "Add /var/named/custom_wildcard_sinkhole.conf and"
      echo "/var/named/custom_single_sinkhole.conf to /var/named.conf"
      echo
      echo "Enter the domain you need to be entered in the sinkhole?"
      echo
      read Domain

      echo "Enter whether you want the domain to be wildcard (Y) or not (N)?"
      read Wildcard

      # Transfer record into a file for processing

      echo $Domain > custom
         if [ $Wildcard = Y ]; then

            echo "Adding to custom wildcard sinkhole list..."
              /root/scripts/domain_dns_nowhere.pl < custom >>
/var/named/custom_wildcard_sinkhole.conf

         else
            echo "Adding to custom single sinkhole list..."
              /root/scripts/client_dns_nowhere.pl < custom >>
/var/named/custom_single_sinkhole.conf
         fi

      /bin/rm domain

        read;;

      d|D) clear

################################################################################
# http://pgl.yoyo.org
#
# The ad banners that you see all over the web are stored on servers. Stopping your
# computer communicating with another computer can be quite simple. So, if you have
# a list of the servers used for banners, it's easy to stop ad banners even getting
# to your browser.
#
# Note: This list is not malware but adware and could be turned off
#
################################################################################

# This is for the Adserver List

# wget http://pgl.yoyo.org/adservers/serverlist.php?hostformat=;showintro=0

# cat serverlist.php\?hostformat\= |tail +51 | sed '/<\//d; /^$/d' >> final.sorted

# /bin/rm serverlist.php?hostformat=

################################################################################
# http://hosts-file.net
#
# HPHosts - This service is free to use, however, any and ALL automated use is
# strictly forbidden without express permission from ourselves. This script does
```

```
# not query the database. The script only download the host.txt file.
#
# Note: This list is not enabled by default. It contains ~125K sites
#
###############################################################################
# Download HPHosts hosts.txt file list
#wget http://support.it-mate.co.uk/downloads/HOSTS.txt
#grep "127.0.0.1" HOSTS.txt |grep -v localhost | awk '{ print $2 }' | sed 's/\"//g'
| sed '/^$/d' >> final.sorted
#/bin/rm HOSTS.txt
###############################################################################
#  http://www.malwaredomains.com
#
#  WARNING: All domains on this website should be considered dangerous. If you do
#  not know what you are doing here, it is recommended you leave right away. This
#  website is a resource for security professionals and enthusiasts.
#
###############################################################################

# Downloading site list from Malware Domains
wget http://www.malwaredomains.com/files/justdomains
cat justdomains | tail +2 > final.sorted
/bin/rm justdomains

###############################################################################
#  https://zeustracker.abuse.ch
#
#  With the ZeuS Tracker you are able the generate a IP- and domain-blocklist which
#  contains all ips / domains which are currently used as Command&Control server
#  (C&C) by the ZeuS crimeware.  Both blocklists will be generated in text format.
#  This allows you to import the blocklist into a firewall or corporate webproxy
#  to block all traffic to the malicious ZeuS C&C servers. You can find a short
#  description of the two blocklist above.
#
###############################################################################

# This is for the Zeus Tracker list

#wget --no-check-certificate
https://zeustracker.abuse.ch/blocklist.php?download=domainblocklist
wget http://www.abuse.ch/zeustracker/blocklist.php?download=domainblocklist

# Parsing the data

cat blocklist.php\?download\=domainblocklist | tail +7 >> final.sorted

/bin/rm blocklist.php?download=domainblocklist*

###############################################################################
# http://mtc.sri.com
#
#  Download a list of the most observed malware DNS names that we have seen looked
#  up during malware infections or embedded within malware binaries.
#
###############################################################################
```

```
# Download SRI list Most Observed Malware-Related DNS Names

wget http://mtc.sri.com/live_data/malware_dns/

grep "<td>" index.html | sed '/mtc\.sri\.com/d; /.*\..*/!d;' | sed 's/<td>//g;
s/<\/td>//g' >> final.sorted

/bin/rm index.html


################################################################################
# http://www.malwarepatrol.net
#
# Malware Patrol is a free, automated and user contributed system for checking URLs
# for Viruses, Trojans, Worms, or any other software considered bad.
#
################################################################################

# Download Malware Patrol list Most Observed Malware-Related DNS Names

wget http://www.malwarepatrol.net/cgi/submit?action=list_bind

cat submit?action=list_bind | awk '{ print $2 }' | sed 's/\"//g' | sed '/^$/d' >>
final.sorted

/bin/rm submit?action\=list_bind

################################################################################


# Backing up working file to /tmp

mv /var/named/site_specific_sinkhole.conf /tmp/site_specific_sinkhole.conf

# Clean up the file from random characters

cat final.sorted | sed '/^$/d' | tr -d '\r' | sed 's/\.$//g' | sort -u > domainlist

# Remove all domains matching site wildcard lists

/bin/cat /var/named/entire_domain_sinkhole.conf | awk '{ print $2 }' | sed
's/\"//g' | sed '/^$/d' > tempwildcarddomains
/bin/cat /var/named/*wildcard_sinkhole.conf | awk '{ print $2 }' | sed 's/\"//g' |
sed '/^$/d' >> tempwildcarddomains

# Add $ to each line for a perfect domain match

/bin/cat tempwildcarddomains | sed 's/$/$/g' > wildcarddomains
/bin/rm tempwildcarddomains

# Build final wildcard list of domains and remove temp files

/bin/grep -hvf wildcarddomains domainlist > malwaredomains
/bin/rm wildcarddomains
/bin/rm domainlist
```

```
# Parsing the files to /var/named

/root/scripts/client_dns_nowhere.pl < malwaredomains >
/var/named/site_specific_sinkhole.conf

#-------------------------------------------------------------------------

# Second Option - List of Sites that should never be Sinkhole

# This option enable the script to create a list of site you do not want to ever
# see in the sinkhole. To enable this list, comment out the 3 commands above and
# uncomment all the options between the 2 dotted lines. Add a list in the
# /root/scripts and name it checked_sites and add all the domains you never want
# sinkholed. When the script runs it will delete the list from going into the
sinkhole

# Backing up working file to /tmp

#mv /var/named/site_specific_sinkhole.conf /tmp/site_specific_sinkhole.conf

# Clean up the file from random characters

      #cat final.sorted | sed '/^$/d' | tr -d '\r' | sed 's/\.$//g' | sort -u >
domainlist

# Remove all domains matching site wildcard lists

#/bin/cat /var/named/entire_domain_sinkhole.conf | awk '{ print $2 }' | sed
's/\"//g' | sed '/^$/d' > tempwildcarddomains
#/bin/cat /var/named/*wildcard_sinkhole.conf | awk '{ print $2 }' | sed 's/\"//g' |
sed '/^$/d' >> tempwildcarddomains

      # Add $ to each line for a perfect domain match

      #/bin/cat tempwildcarddomains | sed 's/$/$/g' > wildcarddomains
      #/bin/rm tempwildcarddomains

# Build final wildcard list of domains and remove temp files

#/bin/grep -hvf wildcarddomains domainlist > /tmp/cleanup_malwaredomains
#/bin/rm wildcarddomains
#/bin/rm domainlist

# Remove local checked list from web list

#/bin/grep -hvf /root/scripts/checked_sites /tmp/cleanup_malwaredomains >
malwaredomains

#/bin/rm /tmp/cleanup_malwaredomains

# Parsing the files to /var/named

#/root/scripts/client_dns_nowhere.pl < malwaredomains >
/var/named/site_specific_sinkhole.conf

#     -------------------------------------------------------------------------
```

```
      # Eliminate duplicates contained in custom lists against the
site_specific_sinkhole.conf file

      /usr/sbin/named-checkconf | awk ' {print $3 }' | tr -d \': > /tmp/list
      /bin/mv /var/named/site_specific_sinkhole.conf
/var/named/site_specific_sinkhole.tmp
      /bin/grep -hvf /tmp/list /var/named/site_specific_sinkhole.tmp >
/var/named/site_specific_sinkhole.conf
      /bin/rm /tmp/list
      /bin/rm /var/named/site_specific_sinkhole.tmp

      echo "Press Enter to continue..."

        read;;

      t|T) clear
        echo "Checking all DNS BIND zone files for errors..."
        echo "Running BIND named-checkconf to test zones files for errors."
      echo "If after this test is completed no errors are reported, the press Enter
to continue..."
        /usr/sbin/named-checkconf

        echo
      echo "Test completed..."
        echo "If you see any errors on this screen, press r to reload backup.
Otherwise press e to exit ..."

        read;;


      f|F) clear
      clear

      echo "WARNING!!!"
      echo
      echo "If the database get flushed clean all previous sinkhole"
      echo "records will not be accessible by your clients until"
      echo "the sinkhole records are reloaded into the PowerDNS database"
      echo "Do you still want to proceed? (Y)"
      read INPUT
           if [ $INPUT = Y ]; then

      echo "Flushing PowerDNS database..."
      /root/scripts/pdns.sh
      /etc/rc.d/rc.pdns restart
      /etc/rc.d/rc.pdns_recursor restart
      echo "Press enter to continue..."
      else
            echo "Terminating script..."
            exit
      fi
        read;;

      r|R) clear
         echo "Zone checked failed and restoring backup zone file"
```

```
         /bin/mv /tmp/site_specific_sinkhole.conf
/var/name/site_specific_sinkhole.conf

         echo "File malware_zone.dns has been restored..."
         echo "Reloading updated zones..."

         /usr/sbin/rndc reload
         /bin/rm final.sorted
         /bin/rm malwaredomains

         exit;;

      b|B) clear

        echo "Reloading BIND updated zones..."

        /usr/sbin/rndc reload
        /bin/rm final.sorted
        /bin/rm malwaredomains
        /bin/rm /tmp/site_specific_sinkhole.conf

        echo "Done DNS Malware list zone updates..."
        echo

        /usr/sbin/rndc status
        echo

        echo "Done reloading BIND zones..."
        echo "Press ENTER to exit ..."

      read

        exit;;

      p|P) clear

        echo "PowerDNS Zone update..."


      # Need to get complete list of current DNS Sinkhole
      cat /var/named/*.conf |awk '{ print $2 }' | sed 's/\"//g' | sed '/^$/d' |sort
-u > New_malware.list
      # Dumping the domains.txt records for sorting

      /usr/bin/mysqldump pdns domains --tab --fields-terminated-by=' ' -T/tmp

      # Extracting list of sites in the PowerDNS Database
      cat /tmp/domains.txt  | awk ' {print $2 }' | sort -u > PowerDNS.lst
      comm -1 -3 PowerDNS.lst New_malware.list > PowerDNS_import.lst
      comm -2 -3 PowerDNS.lst New_malware.list > PowerDNS_remove.lst
      /bin/rm PowerDNS.lst
      /bin/rm New_malware.list

      # This script builds the list of records to remove from the PowerDNS database
```

```
      /root/scripts/remove_powerdns.pl < PowerDNS_remove.lst > /tmp/sinkhole.remove
      /bin/rm PowerDNS_remove.lst
      REMOVE=`/bin/grep -c records /tmp/sinkhole.remove`

      # Remove old records from PowerDNS Database

      echo
      echo "Deleting $REMOVE old records from PowerDNS database..."
        /usr/bin/mysql -upowerdns -ppassword -D pdns -h 127.0.0.1 <
/tmp/sinkhole.remove
        /usr/bin/mysql -upowerdns -ppassword -D pdns -h 127.0.0.1 <
/root/scripts/optimize.dns

      echo "Backup of deleted domains in /root/backup ..."

      # Making backup directory if it doesn't already exist

      /bin/mkdir -p /root/backup

      /bin/mv /tmp/sinkhole.remove /root/backup/sinkhole_remove.$TODAYDATE

      # This routine prepare the list of domains to import

      cat PowerDNS_import.lst | sed 's/^/"/g' > temp.lst
      mv temp.lst PowerDNS_import.lst

      # This routine builds the list of records to add to the PowerDNS database

      /bin/grep -hFf PowerDNS_import.lst /var/named/*.conf > /tmp/sinkhole.add

      /bin/rm PowerDNS_import.lst

      # Convert DNS addresses to import into PowerDNS Database

      /usr/bin/zone2sql --gmysql --named-conf=/root/scripts/named.conf
/tmp/sinkhole.add > /tmp/powerdns_zones.sql

      echo "Backup of new sinkhole additions in /root/backup ..."

      /bin/mv /tmp/sinkhole.add /root/backup/sinkhole_add.$TODAYDATE

      echo
      echo "Loading DNS sinkhole database with current/updated address list..."
        /usr/bin/mysql -upowerdns -ppassword -D pdns -h 127.0.0.1 <
/tmp/powerdns_zones.sql
      /usr/bin/pdns_control purge
      /usr/bin/pdns_control rediscover

      # Cleanup temporary files
      #
      echo
      echo "Cleaning up and removing temporary files..."
      echo

        /bin/rm final.sorted
        /bin/rm malwaredomains
```

```
        /bin/rm /tmp/site_specific_sinkhole.conf
      /bin/rm /tmp/powerdns_zones.sql
      /bin/rm /tmp/domains.*

        echo "Done PowerDNS Malware list zone updates..."
        echo

        echo "Press ENTER to exit ..."

      read

        exit;;

      e|E|q|Q) echo "                    Done with Sinkhole parser"
      exit;;
      *) echo "                   Invalid entry, try again"
         sleep 2;;
   esac
done
echo "done"
```

12 Appendix E: Creates Integrated Sinkhole Single List

This Perl script is used by sinkhole_parser.sh script to build a sinkhole list based on Fully Qualified Domain Names (FQDN) using data downloaded from websites listed in this document.

```
client_dns_nowhere.pl

#!/usr/bin/perl -X

# Guy Bruneau, seeker@whitehats.ca
#
# Date: 11 January 2010
# Version: 1.0

# This script is used to create a sinkhole list from the downloaded
# sinkhole addresses available from the Internet.

use IO::Handle;
use Getopt::Long;

my $dns = " ";
my $doublequote = "\"";
my $zonemalware = " IN { type master\; notify no\; file
\"/var/named/sinkhole/client.nowhere\"\; }\;";

# This subroutine process the logs into a format to be sorted in the
next routine

while (<>) {

        @DNSList = split;

     DomainID();

}

sub DomainID {

        $dns = $DNSList[0];
    $dns = lc ($dns);
        print  "zone " . $doublequote . $dns . $doublequote .
$zonemalware . "\n";

}
```

13 Appendix F: Daily Report Script

This script uses httpry to create a daily report using packets captured by the DNS sinkhole.

httpry_daily.sh

```
#!/bin/sh
#
# httpry DNS Sinkhole daily report
#
# Written by Guy Bruneau (seeker@whitehats.ca)
# August 2010

# The purpose of this report is to create a summary list of
# all the clients that were blocked by the DNS Sinkhole

# Global HTTPRY options
FIELDS="source-ip,dest-ip,method,host"

# Get yesterday's date

DATEDIR="`/bin/date --date "-1 days" +"%Y-%m-%d"`"
DATE="`/bin/date --date "-1 days" +"%Y%m%d"`"

# This provides the date of the previous day to process the logs
# used in the bothunter rules

        # This loop process hours 0 to 9

        for ((i=0; i <=9; i+=1)); do

           /usr/local/bin/httpry -f $FIELDS -r
/LOG/sinkhole/dailylogs/$DATEDIR/$DATE"0"$i > /tmp/httpry.log

         done

         # This loop process hours 10 to 23

         for ((i=10; i <=23; i+=1)); do

           /usr/local/bin/httpry -f $FIELDS -r
/LOG/sinkhole/dailylogs/$DATEDIR/$DATE$i >> /tmp/httpry.log

         done

# Adding a header with the date to the report
# Adding a description header for each field
#
```

```
/bin/echo "        Daily DNS Sinkhole report for the $DATE" >
/usr/local/webmin/reports/DNS/$DATE.txt
/bin/echo " " >> /usr/local/webmin/reports/DNS/$DATE.txt
/bin/echo "Count   Source          Destination     HTTP    Website "
>> /usr/local/webmin/reports/DNS/$DATE.txt
/bin/echo " " >> /usr/local/webmin/reports/DNS/$DATE.txt
/bin/echo " " >> /usr/local/webmin/reports/DNS/$DATE.txt

# Sorting the sinkhole events with a count

/bin/cat /tmp/httpry.log | sort | uniq -c |sort | sed 's/^[ ]*//' |
sed 's/[ ]/\t/' >> /usr/local/webmin/reports/DNS/$DATE.txt
cd  /usr/local/webmin/reports/DNS

# Turning the report into an HTML report and posting it in
# Webmin -> Servers -> DNS Reports
#
/usr/local/webmin/reports/Htmlify.pl $DATE.txt
/usr/local/webmin/reports/ExplorerIndex.pl
/bin/rm /usr/local/webmin/reports/DNS/$DATE.txt
/bin/rm /tmp/httpry.log
```

14 Appendix G: References

Damballa. (2010, August 1). *Advanced persistent threats (apt)*. Retrieved from http://www.damballa.com/knowledge/advanced-persistent-threats.php

ClearCloud. (2010, October 1). *Set up Clearcloud the Easy Way*. Retrieved from http://clearclouddns.com/Setup/Windows/

DNS Made Easy, Initials. (2010). *Records 101*. Retrieved October 15, 2010, from http://www.dnsmadeeasy.com/s0306/res/recs.html

IANA. (2010, October 12). *Root Zone Database*. Retrieved from http://www.iana.org/domains/root/db/

Glosser, D. (2010, July 23). *164 New Domains (Zeus, Gumblar, iframe, etc)*. Retrieved from http://www.malwaredomains.com/wordpress/?p=1129

Green, Barry Raveendran & McPherson, Danny. (2003, June). *A Swiss Army Knife Security Tool*. Retrieved August 15, 2010, from Arbor Network Web site: http://www.arbornetworks.com/dmdocuments/Sinkhole_Tutorial_June03.pdf

Hunt, Andrew. (2010, September 12). *Visualizing the Hosting Patterns of Modern Cybercriminals*. Retrieved from http://www.sans.org/reading_room/whitepapers/dns/visualizing-hosting-patterns-modern-cybercriminals_33498

Ludwig, Andre. (2009, July 6). *IE 0day exploit domains (constantly updated)*. Retrieved from http://isc.sans.edu/diary.html?storyid=6739

ManageEngine. (2010, May 6). *Netflow analyzer*. Retrieved from http://www.manageengine.com/products/netflow/

Norton DNS. (2010, October 1). *Norton DNS Beta*. Retrieved from http://www.nortondns.com

McPherson, Danny. (2004, February 23). *ISP security: deploying and using sinkholes. Proceedings of the APRICOT 2004 - KUALA LUMPUR, MY*, http://www.tcb.net/Apricot-2004/Apricot-2004-Sinkholes.ppt

Miller, Damien. (2006, November 2). *Softflowd*. Retrieved from http://www.mindrot.org/projects/softflowd/

OpenDNS. (2010, October 1). *OpenDNS*. Retrieved from http://www.opendns.com/start/

Riden, Jamie. (2007, July 13). *Know your enemy: fast-flux service networks*. Retrieved from http://www.honeynet.org/papers/ff/

Rieger, Gerhard. (2010, January). *Socat - Multipurpose Relay*. Retrieved from http://www.dest-unreach.org/socat/

Roesch, Martin. (2010, October 04). *Snort Downloads*. Retrieved from http://www.snort.org/snort-downloads

ShadowServer. (2010, May). *2 Year Botnet Status*. Retrieved from http://www.shadowserver.org/wiki/pmwiki.php/Stats/BotnetCharts

ShadowServer. (2010, October). *What is a Botnet?*. Retrieved from http://www.shadowserver.org/wiki/pmwiki.php/Information/Botnets

tcpdump.org. (2010, April 5). *Tcpdump & Libpcap*. Retrieved from http://www.tcpdump.org/

Team Cymru. (2010, August). *The Darknet Project*. Retrieved from http://www.team-cymru.org/Services/darknets.html

Trojan.Bredolab. Symantec, c. Web. (2010, September 12). Retrieved from http://www.symantec.com/security_response/writeup.jsp?docid=2009-052907-2436-99

Visscher, Bamm. (2008, March 26). *Sguil: The Analyst Console for Network Security Monitoring*. Retrieved from http://sguil.sourceforge.net/