Global Information Assurance Certification Paper

Implementation and use of DNS RPZ in malware and phishing defence

GIAC (GSEC) Gold Certification

Author: Alex Lomas, a.lomas [at] imperial.ac.uk

Advisor: Lee W. Peterson

Accepted: March 10, 2014

Abstract

There has been growing interest in the use of DNS RPZ (domain name system response policy zones) as a mechanism to defend against malware on the web. This paper will examine the history of DNS RPZ, its applications (including malware and phishing) and experience of its deployment instead of other layer 7 based filtering systems. This paper will provide a step-by-step process for configuring DNS RPZ in BIND, as well as ancillary services for logging and manual manipulation of the RPZ, and examines the need for user education and response to 'false positives'.

1. Introduction

Many organisations, large and small, have a need for outbound content filtering. This can be to meet regulatory needs (*e.g.* prohibiting unauthorised communications mechanisms in the financial services sector), safeguarding requirements (*e.g.* in schools), and for other operational needs. Even in institutions that have a more relaxed web usage policy, IT departments often desire a mechanism to limit access to harmful sites involved in malware. This paper particularly examines the latter case, although the techniques it discusses could be extended to filter any kind of content with the relevant set of data.

Websites hosting malware (perhaps inadvertently) are increasingly prevalent (Rains, 2013) and exploit 'drive-by' attacks on surfers that are using out-of-date browsers and plugins. Many browser vendors offer URL screening systems, *e.g.* IE's SmartScreen and Chrome's Safe Browsing; however, this won't necessarily protect against network activity not initiated in the browser or in devices such as smartphones.

Organisations are also under sustained attack from phishing. Credential theft can either be part of a wider campaign of infiltration or can simply be the 'mining' of accounts for later resale (Krebs, 2012). A single account compromise can also be used to perpetuate phishing internally with much more plausible source addresses (Betts, 2013).

One approach is to manage blacklists on endpoint devices, for example with Active Directory Group Policy or with enterprise antivirus products such as those from Symantec or Malwarebytes. This requires the whole environment to be tightly controlled and managed and may therefore cause issues where bring-your-own-devices (BYOD) are permitted.

Network-level filtering is therefore required to ensure full coverage across all devices. This can be achieved through router access control lists (ACLs), proxies, content-aware firewalls and DNS RPZ.

Author Name, email@addressa.lomas [at] imperial.ac.uk

1.1. Router ACLs

Border routers can be configured to black-hole destination addresses, however this can require a high degree of manual management, and lead to the accidental blocking of innocent sites hosted on the same network or server (A. Ferguson, 2013).

Many organisations maintain an ingress IP filter to exclude bogons, unroutable RFC1918 private addresses, and spoofed packets as per BCP38 (P. Ferguson & Senie, 2000). The management of some kind of IP ACL on an external router can be complex and has traditionally been used to defend against ingress rather than egress, but the chances are that most organisations have such a mechanism in place already.

To extend IP ACL filtering to egress traffic would require a translation from malware site full qualified domain name (FQDN) to IP, and the ongoing maintenance of that lookup, because malware command and control (C&C) servers often move hosts as they are discovered. For a small set of destinations this is possible, but this technique can quickly become unmanageable.

Costs: Low

Complexity: Moderate

Overhead: High

1.2. Web proxy filter

A web proxy can be set up using site categorisation, and by forcing all systems to send web traffic via the proxy. Harmful sites and other content deemed restricted by the organisation can then be screened out.

Many organisations, particularly those in regulated sectors such as healthcare and finance, already pass web content through a proxy to enforce organisational policy. Typically proxy appliances download a vendor-maintained categorisation list and the organisation then chooses what to block or permit in broad categories; although individual URL exceptions can be made. Dependent on the vendor and appliance, the

underlying inspection engine can be extended to screen for threats, such as Websense ACE (Websense, 2014).

Assuming the deployment is capable of it, extending the proxy to screen for threats should be relatively straightforward for organisations that already enforce web access via such a proxy. Using proxies *does* however require an appropriately-sized infrastructure which - for many organisations - can be large and costly. In addition, it incurs ongoing subscription and maintenance fees. Being reliant on site (mis)categorisation by a third party can also cause problems (Burrell, 2013).

Costs: Moderate

Complexity: Low-Moderate (depending on existing deployment)

Overhead: Low

1.3. Content-aware firewall

Modern layer 7 aware firewalls can be configured to screen web sites and to block malicious activity. Whilst some of the terminology around 'next gen' and 'content aware' firewalls could be interpreted as marketing, in this context the difference between the web proxy in 1.2 and the content aware firewall is that the firewall can transparently inspect traffic flows inline.

Web proxies have traditionally been setup to one side (with appropriate firewall rules permitting web egress) thus requiring client devices to be configured manually to send their web traffic through them. Firewalls, by their nature, are already inline and inspecting traffic, so can be leveraged to inspect content at layers 4 & 7.

Many organisations, especially SMEs, may already have border internet routers and firewalls that have these features (*e.g.* Juniper SRX, Draytek Vigor) and therefore the cost of implementing them can be low. For larger enterprises full layer 7 inspection of all traffic on the default route may well require unfeasibly expensive equipment.

Costs: Low-Moderate

Complexity: Moderate

Overhead: Low

1.4. DNS RPZ

DNS RPZ is an attempt to bridge the gap between the need to cover the entire corporate network and the expense involved in purchasing and maintaining deep packet inspection capabilities.

DNS RPZ was first proposed by Paul Vixie (Vixie & Schryver, 2010) as a mechanism to grade the trustworthiness of domains involved in spam mail filtering. It provides an extension for BIND (the DNS application) that allows an organisation to create non-standard responses for specific domains: for example to respond that a domain or individual entry does not exist (NXDOMAIN); or to respond to the client with a redirect to another server entirely.

A drawback to DNS RPZ is that - to ensure client compliance - outbound DNS requests would need to be blocked. However, in general it is reasonable to assume that the majority of client devices will be using the organisation's DNS servers especially if DHCP is widely used.

Driving an organisation's DNS RPZ from an external reputation list, akin to existing real-time blackhole lists (RBLs), therefore provides a way to manage "bad" domains across an organisation irrespective of device and at enterprise scale.

Costs: Low

Complexity: Low

Overhead: Low

2. Implementing DNS RPZ

RPZ was created as an open standard by the Internet Systems Consortium (ISC) (Vixie & Schryver, 2010) and is intended to be used by any vendor. At present BIND is the only DNS server platform that supports and complies with this standard. RPZ works by holding policy zones (which are in fact regular zone files) on recursive DNS servers with instructions on what to do in case of a match (pass through, log, redirect, *etc.*). As the policy zones are formatted as regular zone files they can be imported easily from an outside supplier as a zone transfer, which supports automatic updates through DNS NOTIFY.

This paper assumes that you are using BIND on Linux as your organisation's DNS server and that you are comfortable configuring it along with separate servers for Apache and logging.

As shown in figure 1, RPZ does not significantly increase the load on your DNS; however you should ensure your existing deployment's performance is satisfactory for your environment with some headroom.

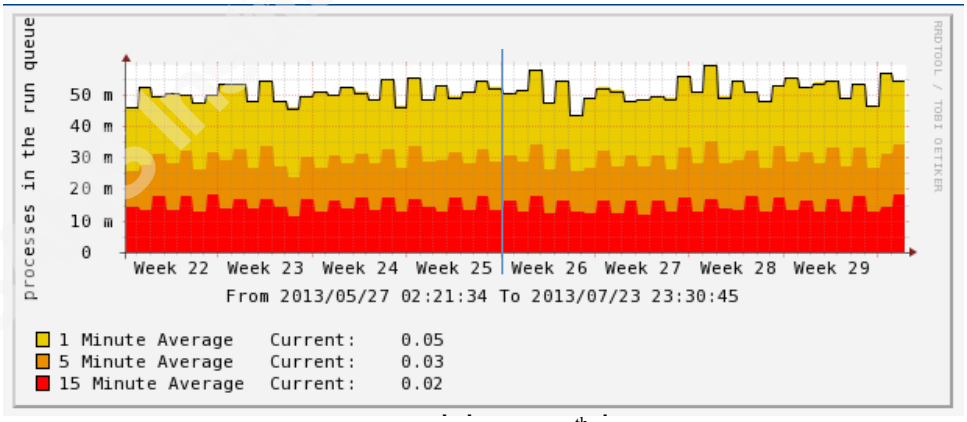

Figure 1. RPZ was implemented on this DNS server on 27th June 2013; there was no change in load average

At the end of this section you will have an automated mechanism for blocking access to malware domains based on an external reputation list. You will also have an internal web server to redirect requests to, for education and logging.

2.1. Risks

Deployment of DNS RPZ carries a degree of risk. You should test in a lab environment first, and you should follow any change management processes before promoting it into production.

DNS is an organisation-critical piece of infrastructure. Outages or problems with it will cause major disruption to your clients and users. Whilst previous deployments of RPZ may have had no noticeable impact, this does not mean this will invariably be the case.

Using externally-curated blacklists for RPZ hands over partial control of your infrastructure to a third party, and you become somewhat reliant on their accuracy. You should extensively test to validate the data received from them. You must be comfortable with the service level agreement (SLA) they provide.

As shown in 2.3, configuring RPZ in BIND is relatively straightforward and therefore so is reverting it in an incident should this be required. Ensure this knowledge and any local procedures (*e.g.* config version control) are well documented so that others can take action if required.

Formulate policies about what will and will not be blocked: for example, extending malware-blocking technology for censorship is unlikely to be accepted. Communicate these to your users together with details of any impending changes to DNS behaviour.

2.2. RPZ source data

You will firstly need to determine the set of source data that defines the malware-containing domains you wish to block. Adoption of DNS RPZ as a mechanism for the distribution of this data is still relatively new technology. However there is already a number of providers available (Internet Identity, 2013; Spamhaus, 2011; SURBL, 2013). Spamhaus, in particular, are keen to increase the uptake of DNS RPZ, and they provided access for experimentation as part of this paper. Some providers may charge for access to DNS RPZ data and - whilst it is possible to generate your own list via freely available

malware datasets (The DNS-BH Project, 2013) - this will require you to maintain the zones, removing one of the key advantages of DNS RPZ: automated zone transfers.

2.3. Configuring BIND

Pulling in an external RPZ takes very little modification to an existing *named.conf* (Connery, 2013); however, you should ensure you are running an up-to-date version of BIND with support for DNS RPZ (*i.e.* 9.8.0 and above). You must also ensure your data provider allows zone transfers from your source IP.

In *named.conf,* under the *options* section, add the following to redirect clients to your internal web server (see 2.4). For specific overrides see 2.5.

```
response-policy {
  # Company-specific local and overrides, references name file
  zone "rpz.companyname.com";
  # spamhaus name-based - redirect to badware
  zone "rpz.spamhaus.org" policy cname badware.companyname.com;
};
```

See the BIND administrators' reference manual for further configuration options (ISC, 2013).

2.4. Configuring malware redirection server

Instead of returning NXDOMAIN for a malware site, the site can be aliased (CNAMEd) to an internal web server that displays the organisation's logo and information about the site the client attempted to visit, see figure 2.

Figure 2: Example of what a client would see in the browser on an attempt to access a harmful site.

Any web server can be used for this purpose; however, in larger environments it is useful to be able to track individual access attempts in case they need to be followed

up. In this instance a simple Python script is used to generate unique IDs which are displayed to the client and can be tied back to the log.

2.4.1. Sample Apache vhost configuration

You will need to edit the relevant paths, IPs and names.

```
# SELinux errors otherwise...
WSGISocketPrefix run/mod_wsgi

# enable a unique ID
LoadModule unique_id_module modules/mod_unique_id.so

# enable a custom log format for bad site access attempts
LogFormat "%h %t %{UNIQUE_ID}e %{Host}i \"%r\" %>s %b \"%{Referer}i\" \"%{User-
Agent}i\"" badware

# enable name-based hosts on this IP
NameVirtualHost 1.2.3.4:80

# 1st match if no "ServerName" match
<VirtualHost 1.2.3.4:80>
        ServerName              *
        DocumentRoot            "/var/www/badware/html"

        # A request to a random HTTP server; redirect to badware.mydomain.com
with some URL args
        RewriteEngine           on
        RewriteRule             (.*)
http://badware.mydomain.com/python/badware?host=%{HTTP_HOST}&rqid=%
{ENV:UNIQUE_ID} [R,L]
        # Also do some custom logging to keep the unique ID, original host/url
and user agent
        CustomLog logs/badsite_access_log badware
        ErrorLog logs/badsite_error_log
</VirtualHost>


<VirtualHost 1.2.3.4:80>
        ServerName              badware.mydomain.com

        DocumentRoot            "/var/www/badware/html"

        # WSGI python script that prints the dynamic page
        WSGIDaemonProcess badware user=badware group=badware processes=6
threads=1 display-name=%{GROUP}
        WSGIProcessGroup badware
        WSGIScriptAlias /python/badware /var/www/badware/app.wsgi

        CustomLog logs/badware_access_log common
        ErrorLog logs/badware_error_log
</VirtualHost>
```

2.4.2. Sample WSGI Python script

This generates a more informative page to the client, including the original domain they were trying to reach.

```
#!/usr/bin/python

import urllib
import urlparse

TEMPLATE = """<html><body>
<!-- generated by mod_wsgi -->
<div>
<img src="/logo.png" alt="Company logo">
<h1>The website you are attempting to visit has been marked as harmful and
access to it is blocked from the company network</h2>
<pre>
        %(badhost)s
</pre>
<p><a href="http://www.mycompany.com/link">Please click here for more
information</a></p>
<p>If you believe this site has been blocked in error please click
<a href="mailto:ict@mycompany.com?subject=%(subject)s&body=%(body)s">this
link</a> to report it to ICT.</p>
</div></body></html>"""

EMAIL_TEMPLATE = '''The site %(host)s has been blocked by RPZ when browsing
the web.

<Explanation of why you think the site is legitimate>.

The block reference was %(uid)s'''

def redirect(environ, start_response, url):
        hdrs = [('Location', url),]
        start_response('302 Moved', hdrs)
        return []

def application(environ, start_response):
        qs = urlparse.parse_qs(environ['QUERY_STRING'])
        hs = qs.get('host')
        uid = qs.get('rqid')
        if not hs or not uid:
                return redirect(environ, start_response, '/index.html')
        uid = uid[0]
        host = hs[0]
        env = {
                'badhost': host,
                'uid': uid,
                'subject': urllib.quote('RPZ site '+host+' blocked'),
                'body': urllib.quote(EMAIL_TEMPLATE % locals())
        }
        resp = TEMPLATE % env
        hdrs = [('Content-Type', 'text/html'), ('Content-Length',
str(len(resp)))]
        start_response('200 ok', hdrs)
        return [resp]
```

2.5. Testing

Firstly, test your RPZ setup in log-only mode. In log-only mode, BIND will log events as if a match had occurred but will not rewrite the domain. Use the policy option DISABLED to do this (ISC, 2013) and monitor your logs carefully.

When you are happy, you can then set your policy to respond as you wish, either via a CNAME or NXDOMAIN.

2.6. Whitelists and blacklists

Irrespective of your RPZ data provider, you will need an override mechanism to either permit sites inadvertently blocked that you don't deem a threat, or to block sites not covered by the external data that are targeting you, *e.g.* via phishing.

Most organisations have an automated mechanism and web interface to manage their BIND zone files (see figure 3) that can be adapted to provide a front-end to manage RPZ overrides. However this can also easily be accomplished by manually editing the zone file.

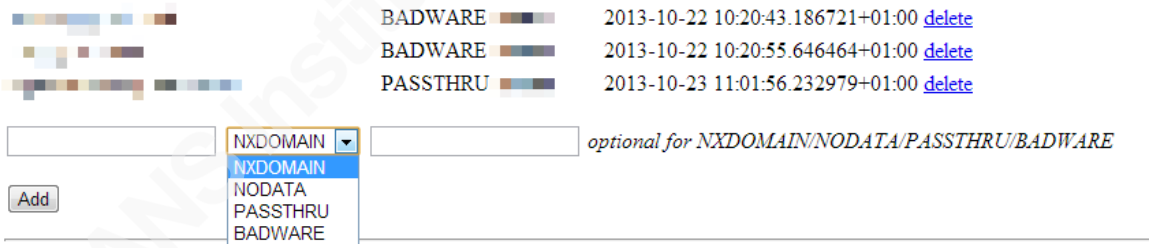

Figure 3: Example of web interface for managing RPZ overrides.

This internal RPZ zone file should be placed above any external RPZ source in *named.conf* so that the internal data takes precedence. Your zone file should look similar to the example below:

```
rpz.mydomain.com.     86400 IN SOA       dns0.mydomain.com.
       hostmaster.mydomain.com. 2011122175 2700 1800 3600000 86400
rpz.mydomain.com.           86400 IN NS   dns1.mydomain.com.
rpz.mydomain.com.           86400 IN NS   dns2.mydomain.com.

www.badsite.com.rpz.mydomain.com. 86400 IN CNAME     badware.mydomain.com.
falsepositive.rpz.mydomain.com.   86400 IN CNAME     rpz-passthru.
```

2.7. Logging

BIND will log actions taken against RPZ directives, which is helpful for analysis. Logged events will look similar to the following:

```
Oct 24 14:17:58 dns1.mycompany.com named[20814]: rpz: info: client
1.2.3.4#65153 (www.baddomain.com): view main: rpz QNAME Local-Data rewrite
www.baddomain.com via www.baddomain.com.rpz.mycompany.com
```

Analysis of these logs against specific domains provides rudimentary intrusion detection systems (IDS) capabilities and can be set to provide alerts for specific viral infections, especially those calling home to C&C servers *e.g.* Zeus, ZeroAccess and Shylock.

2.8. User education

Changing the behaviour of your DNS and redirecting end-user devices to an internal malware page will inevitably lead to some confusion for users. This may be because of incorrect site classification (false positives) or because of an actual malware infection (true positives). It is therefore important to provide information and education to your users about upcoming changes, and to provide examples of what they might see. You should make it easy for your users to report false positives by providing a link containing the relevant information for your service desk or security team to assess.

Assessing the legitimacy - or otherwise - of a site can be tricky; however, the following could be used for an initial assessment.

- Google safe browse diagnostic.
  Interrogates Google's database of potentially dangerous sites. This database is used by Chrome and in search results.
  http://www.google.com/safebrowsing/diagnostic?site=www.example.org
- Malware domains.
  Community project that maintains a listing of domains known to propagate malware and spyware.
  http://www.malwaredomains.com

- Anubis

  Automated service that visits a site then analyses changes to a remote virtual PC

  http://anubis.iseclab.org/
- Cuckoo Sandbox

  Similar to Anubis, this is a disposable virtual machine you run that allows the safe investigation of a site.

  http://www.cuckoosandbox.org

3. Analysis

On our large network, which has many thousands of networked devices, we have had approximately 100 reports regarding sites blocked by RPZ over the last 12 months. Of these, the vast majority have transpired to have been legitimately blocked.

DNS RPZ provides a useful mechanism for tracking malware infections on a network. Many Trojans report back to a known set of C&C servers, and whilst DNS RPZ will be effectively blocking this communication, analysing your logs will allow detection and clean-up of the original infection.

For example, one of the many C&C hosts used by Shylock/Caphaw is at wguards [dot] cc (BAE Systems Detica, 2013). Although your RPZ implementation should be preventing infected devices from calling home, Shylock can still be an exceptionally chatty beast. Most of the peaks in figure 4 are down to individual machines contracting Shylock and repeatedly trying to reach their C&C server.

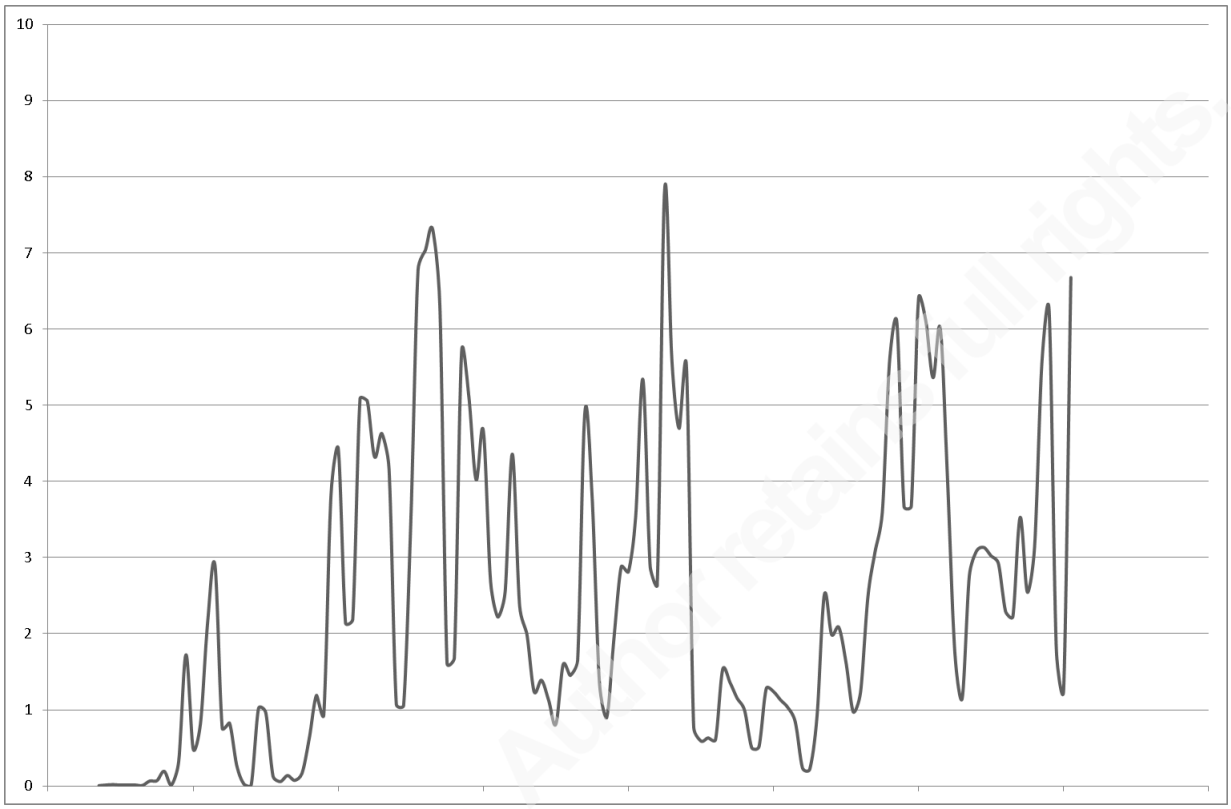

Figure 4: Number of RPZ malware hits over a four month period in 2013; absolute numbers have been obfuscated.

RPZ log analysis can also be useful in analysing attempted response rates following phishing attacks which can then be used for tailored user-education.

4. Conclusion

DNS RPZ provides a scalable, relatively low cost and effective mechanism for blocking malware and phishing on a corporate network where no existing layer 7 filtering mechanism is currently deployed. Implemented correctly it results in a low rate of false positives and little extra overhead for IT security teams.

5. Acknowledgements

Phil Mayers for all his assistance in setting up and configuring DNS RPZ.

Steve Cook for proof reading.

Author Name, email@addressa.lomas [at] imperial.ac.uk

6. References

BAE Systems Detica. (2013). Pray before you buy with shylock. Retrieved 3/10, 2013, from http://baesystemsdetica.blogspot.co.uk/2013/03/pray-before-you-buy-with-shylock.html

Betts, A. (2013). A sobering day. Retrieved 3/10, 2013, from http://labs.ft.com/2013/05/a-sobering-day/

Burrell, I. (2013). O2 changes porn filter after charity sites blocked. Retrieved 01/16, 2014, from http://www.independent.co.uk/life-style/gadgets-and-tech/news/o2-changes-porn-filterafter-charity-sites-blocked-9023209.html

Connery, H. (2013). Response policy zone history, usage and research. Retrieved 3/10, 2013, from http://www.spamhaus.org/whitepapers/RPZ-History-Usage-Research.pdf

Ferguson, A. (2013). Blocks of EZTV have unintended consequences. Retrieved 3/10, 2013, from http://www.thinkbroadband.com/news/5988-blocks-of-eztv-have-unintended-consequences.html

Ferguson, P., & Senie, D. (2000). *RFC 2827 network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing.* ().IETF.

Internet Identity. (2013). Retrieved 10/18, 2013, from http://www.internetidentity.com/dns-security-solutions/

ISC. (2013). *BIND Administrator's Reference,* (9.9.4), 84.

Author Name, email@addressa.lomas [at] imperial.ac.uk

Krebs, B. (2012). Exploring the market for stolen passwords. Retrieved 02/27, 2014, from http://krebsonsecurity.com/2012/12/exploring-the-market-for-stolen-passwords/

Rains, T. (2013). On the origins of malware: Are malware hosting sites in your state or region? Retrieved 3/10, 2013, from http://blogs.technet.com/b/security/archive/2013/03/06/on-the-origins-of-malware-the-global-distribution-of-malware-hosting-sites.aspx

Spamhaus. (2011). Spamhaus' DBL as a response policy zone (RPZ). Retrieved 10/18, 2013, from http://www.spamhaus.org/news/article/669/

SURBL. (2013). Introducing SURBL URI reputation data. Retrieved 10/18, 2013, from http://www.surbl.org/

The DNS-BH Project. (2013). Retrieved 10/18, 2013, from http://www.malwaredomains.com/

Vixie, P., & Schryver, V. (2010). *ISC-TN-2010-1-B3 DNS response policy zones (DNS RPZ)* (3rd ed.) ISC.

Websense. (2014). ACE (advanced classification enginer). Retrieved 01/16, 2014, from http://www.websense.com/content/websense-advanced-classification-engine.aspx