Response Policy Zone
History, Usage and Research

Hugo M. Connery
Technical University of Denmark

©
License: CC-BY-SA 3.0 [1]

May 16, 2013
List of Tables

1. Single Client RPZ Timing Analysis ........................................ 12
2. Single Client RPZ Bucket Analysis ....................................... 16
1 Introduction

Response Policy Zone (RPZ [4]) is an extension for recursive DNS [2] resolvers (rDNS) [3], which has been implemented in recent BIND [6] name server software.

A brief overview of RPZ, and its origins are presented, followed by descriptions of two case studies of its use in a production environment. A potential for using RPZ to identify infected systems using data generated through the use of RPZ is considered.

The use of RPZ for phishing attack defence is presented.

2 Background

2.1 rDNS: Normal Operation

Under normal operation (without RPZ) an rDNS is tasked with accepting queries and responding with either the authoritative answer, or that an error occurred. An authoritative answer may either by an answer with data, or that the domain does not exist.

Within normal operation, there is no flexibility given to an rDNS. It must just do its best to find an authoritative answer and provide it, or report some error.

2.2 Reputation Data

In the fight against Unsolicited Bulk Email (UBE) [7], also known as spam, an opportunity for the internet community to assist other members of the community was developed by Paul Vixie [11]. The project was MAPS [5] (SPAM spelt backwards: Mail Abuse Prevention System), allowing community members to share information about known internet systems (email servers) that delivered only spam. By sharing and combining their reputation knowledge, the community created a improved defence.

There were legal repercussions. Spam was not illegal (certainly not in all jurisdictions) and some businesses based their livelihood on delivering spam.

Despite legal challenges, the mechanism proved valuable and an industry was born in which organisations could pay other organisations to receive reputation data to protect their personnel’s mailboxes receiving unwanted and/or malicious email.

2.3 Malware

Technically skilled and creative people have been devising ways to induce computers to perform unrequested actions since the emergence of computers themselves. With the growth of the internet, its number of connected systems, and the basis of most early (and still many) internet protocols not taking system security as a fundamental concern, the ability for these people to cause unexpected behaviour on computers increased dramatically.

The purpose of these pieces of software turned from advertising the skill of the author, to practical jokes, and continued to the enslavement of the computer which had performed the ‘unrequested action’.

Mechanisms for the delivery of these unrequested softwares have also evolved, often using internet protocols (FTP, SMTP, etc.), and also using transportable digital storage devices (Floppy disks, USB storage devices etc.).
The control of thousands, or millions, of enslaved computers creates a powerful resource for use in modern societies that use the internet as an augmentation of, or replacement for, many previously non-computer assisted actions (communications, banking etc.). Thus, the collective enslavement of large numbers of computers becomes a potentially profitable tool for use in spamming, fraud, blackmail and other activities.

Current terminology for an enslaved computer is a bot \[8\] (contraction of robot) or zombie. A collection of bots is known as a botnet \[9\] (as the collective control is enabled by a network, very commonly the Internet).

Various groups, often with malicious purpose, compete amongst themselves for the identification and infection of computers that are vulnerable to become enslaved, so as to increase the size of their botnet(s).

2.4 Conficker

A botnet, named Conficker \[10\], was identified in late 2008. It was a well crafted and virulent, self propagating piece of malware (a worm) to enslave computers. The actions of infected systems could be controlled by a central authority (thus a botnet). It grew to several million infected computers.

During remedial efforts to control the spread of Conficker, Paul Vixie was requested to assist neutralising the threat of Conficker (to sinkhole the botnet). At that time, Conficker was known to exist, but not known to have performed any malicious action. It was a ticking bomb of several million internet connected computers.

Analysis of Conficker showed that it was programmed to use dates and times to form domain names which a bot would then attempt to contact to receive instruction. These were what are known as command and control domains.

To sinkhole the botnet, these domains were registered ahead of time to be in the control of the 'defenders' rather than others who may have been behind Conficker. Initial work required the registration of 500 domain names per day, across some generic top level domains (TLD).

Thereafter, the authors/controllers of Conficker, unable to use DNS to control the bots, re-used the initial attack vector (the owners of the compromised systems has still not protected their systems from the threat which first saw them infected) to re-infect the systems with an update of the Conficker worm software. This used 50 000 domain names per day, that were spread across not just generic top level domains, but also many different country code TLD’s (ccTLD).

Countering this 'misuse' of the DNS by the Conficker malware called for a new strategy, as forward registering 50 000 domains per day across many TLD’s and ccTLD’s is expensive, and often unfeasable due to language barriers and variations in registration mechanisms.

2.5 Generalisation and Domain Name Reputation

As DNS Registrars are a business, and that those businesses exist globally, and that the number of domains required to be 'forward registered' or to be 'not allowed to be registered' was essentially limitless, the only available solution to the problem was within the DNS system itself, rather than with the registrars. Paul Vixie, a founder of Internet Systems Consortium \[14\], the maintainers of BIND (the most common name
server software used on the internet), chose to describe a proposal [13] for the use of reputation data for domain names themselves, and to have that implemented in BIND.

Due to the publication of the specification for RPZ, the mechanism can be implemented by any provider of DNS software, should they wish.

Again, an industry of providing reputation data, this time of domains, rather than of email servers, has been created.

3 RPZ

RPZ has been available in BIND at least from version 9.8.1 (available in late 2010).

3.1 Zones and Zone Files

BIND uses zone files to store information (resource records) which describe what it knows about DNS data. aDNS servers describe the domains about which they are authoritative in zone files. rDNS usually only have zone files for fixed standardized domains (e.g. localhost). An rDNS will use the DNS itself for determining answers to queries from clients.

RPZ uses zone files to define domains which are to be filtered.

An rDNS configured to use RPZ may use locally defined zones and/or zones that are provided by external reputation data providers.

When external zones are used, incremental zone transfer (IXFR) is commonly used to reduce network traffic. Zones of reputation data are often updated frequently (i.e. every few minutes) and often define hundreds of thousands of domains, resulting in zone files that contain over a million lines of text. IXFR enables updates to transfer only differences in zone file data, greatly improving performance.

An rDNS, using RPZ, has access to the following data:

- answers to questions which it has already performed (cache)
- zone files loaded into memory for which it is authoritative

The loaded zones are the standardized (e.g. localhost) and the RPZ zones (either locally defined or obtained from external providers).

3.2 Policy

An rDNS has some defined zones. RPZ allows the description of policy on these zones, if they are to be used for RPZ.

Each zone file can define individual records to indicate which type of policy should be applied to that record. Individual records can be specified to indicate various policy choices, including NO-OP.

The response policy specification can provide a policy for all records in the zone. Individual records that specify NO-OP cannot be overridden by the zone level policy specification.

Generally used zone level policy specifications are:

- NXDOMAIN (the default): indicate to the client that the domain does not exist
• CNAME xxx.yyy.zzz: tell the client to visit the xxx.yyy.zzz domain instead (redirect)
• given: use whatever is specified at the record level

Other options are available. Please consult the RPZ specification [13] for details. If one uses a redirect (CNAME), this is known in RPZ parlance as a 'walled garden'.

3.3 Operation
The normal operation of an rDNS upon receiving a query from a client is:

1. if the answer is in cache and still valid, provide that
2. perform a full recursive operation querying aDNS' and provide that answer

RPZ inserts a new operation between steps 1 and 2; check your zones. If the query is found in a zone, apply whatever policy is specified and return that response.

3.4 Example Config
The following configs are redacted from real configs that have been in use since 2012-09-16.

The named.conf uses both a locally defined RPZ zone (provided) and Spamhaus' RPZ data (redacted).

This config is provided as an example, not a recommendation.

3.4.1 named.conf

```plaintext
// the people that we will serve
acl clients
{
    // IPv4
    127.0.0.0/8;
    192.168.0.0/16;
    // routable IPv4 networks redacted
    // IPv6
    ::1;
    // routable IPv6 networks redacted
};

// redact other acls and logging definitions

options{
    allow-recursion { clients; };
    allow-query-cache { clients; };
    allow-query    { clients; };

    listen-on port 53 { any; };
}
```
listen-on-v6 port 53 { any; }

directory     "/var/named";
dump-file     "/var/named/data/cache_dump.db";
statistics-file "/var/named/data/named_stats.txt";
memstatistics-file "/var/named/data/named_mem_stats.txt";

version "None of your business";

response-policy
{
    zone "local.rpz" policy CNAME warning.domain.org;
    zone "rpz.spamhaus.org" policy CNAME warning.domain.org;
}

// The locally defined rpz zone
zone "local.rpz" in
{
    type master;
    file "local.rpz.zone";
    allow-query   { clients; }
}

// The locally defined rpz zone
zone "rpz.spamhaus.org" in
{
    type slave;
    masters {
        // redacted: spamhaus rpz DNS'
    }
    file "rpz.spamhaus.org.zone";
    allow-query   { clients; }
}

3.4.2 local.rpz

This is real data of phishing attack defence. No redaction.

$TTL 86400
@ IN SOA @ root ( 20130050701 ; serial 3H ; refresh 15M ; retry 1W ; expiry 1D ) ; minimum

IN NS LOCALHOST.
; Our locally configured nasty domains
; The first record is a local test case. A non-existent domain to
; be used for testing RPZ.
nastynasty.com IN CNAME .
*.nastynasty.com IN CNAME .

; Phishing domains
;
; phishing scam (2012-10-29)
webmail-danskebank.com IN CNAME .
*.webmail-danskebank.com IN CNAME .
; phishing scam (2012-10-31)
ws21.tijdelijke-url.nl IN CNAME .
*.ws21.tijdelijke-url.nl IN CNAME .
; phishing scam (2012-12-06)
edryecc.com.au IN CNAME .
; phishing scam (2012-12-11)
shilohstreet.com IN CNAME .
*.shilohstreet.com IN CNAME .
; more phishing (2012-12-19)
cobern-gaming.nl IN CNAME .
*.cobern-gaming.nl IN CNAME .
; more phishing (2013-01-16)
naddeoporte.it IN CNAME .
*.naddeoporte.it IN CNAME .
; more phishing (2013-01-24)
alfakitap.com IN CNAME .
*.alfakitap.com IN CNAME .
; more phishing (2013-04-16)
lancapris.com IN CNAME .
*.lancapris.com IN CNAME .
onlinewebshop.net IN CNAME .
*.onlinewebshop.net IN CNAME .
; more phishing (2013-05-07)
ecdomonline.com IN CNAME .
*.ecdomonline.com IN CNAME .

4 Implementing RPZ: A Trial

The Department of Environmental Engineering (ENV) at the Technical University of Denmark (DTU) decided to trail the use of RPZ in late 2012. Spamhaus graciously provided their RPZ data feed gratis.

A report of the trial has been published by Spamhaus [15].
4.1 Goals
The goal of trialing RPZ was to:

- increase client system security by preventing access to nasty domains
- increase client system security by informing the community of the risk of visiting nasty domains
- increase general security by raising the awareness amongst the department’s IT staff of nasty domains

4.2 Architecture

4.2.1 Resolvers
ENV had two in house rDNS’ which were used by all internet connected systems. A third was established to subscribe to the Spamhaus data feed. Additionally, a local rpz zone (as shown above) was created, to describe a single fictitious domain to be used in testing.

Once the new third resolver was operational, the two primary rDNS’ were set to subscribe to the first as the master for both the Spamhaus data feed (zone) and the locally defined zone.

This architecture was chosen as it reduces the load on Spamhaus, and increases the speed of zone update transfers (1 Gbps local area network).

In accordance with recommendations from ISC, the two local resolvers were given an additional 1 GB of RAM, and the new third resolver was commissioned with 2 GB (all have 2 GB).

The resolvers rely on keeping their zone data in memory for speed. CPU cycles are not the issue, RAM is.

4.2.2 Policy
The default policy for RPZ is NXDOMAIN. This provides the defence that one wishes for (clients cannot visit nasty domains), but does not provide the end user with any manner to differentiate filtering from non-existence.

ENV has an educated community (university) and wished to engage the community in understanding why this was being done, and more importantly to let them complain if they felt that they were being inappropriately filtered, or that this technology was impacting their productivity.

Thus, a constant CNAME redirection was used. A local website was established, with the departmental logo and information about why people had ended up there, and encouragement to complain if they felt that this was inappropriate.

Thus, a person attempting to visit a filtered domain would end up at the ‘warning’ web site instead of just being unable to connect to the dangerous domain.
4.3 Politics

An open lunchtime seminar was held to engage the community in the filtering and the goals of the trial. Specifically, the warning web site page was displayed so that should a community member end up visiting it unintentionally, they would already be aware of it and its purpose. Discussions allowed community concerns to be aired and addressed.

4.4 Results

During the 4 week trial just under 5 000 attempts to contact nasty domains were prevented. 75 client systems were defended.

One report of 'inappropriate filtering' was received. It was a false false positive: it was an email harvesting website.

No reports of 'loss of productivity' were received.

Additionally, analysis of the log data (from both the resolvers and the warning site) enabled the identification of malware installed on two systems that had professional anti-virus software installed, with latest updates.

4.4.1 Summary

- 75 systems were protected from contacting nasty domains
- no loss of productivity reported
- no 'inappropriate filtering' reported
- two infected systems identified
- increased awareness in both the general and IT community was achieved

Due to this entirely positive result, ENV decided to approve the continued use of RPZ and Spamhaus’ data. It is still in use 8 months after the trial.

5 Phishing Attack Defence

The university (DTU) came under sustained phishing attacks during and after the RPZ trial. Indeed, the Danish Computer Emergency Response Team (DK*CERT [16]) published on 2013-05-13 that the month of April 2013 had the all time highest number of reported phishing attacks against Danish organisations.

The local.rpz.zone file was used to provide defence against phishing attacks. The community was asked to forward to IT any email which contained high quality phishing attacks. Given the level of security awareness in the community, there was no need to filter poor quality attacks, just the good ones.

Since the use of RPZ to protect against phishing attacks 68 attempts (as of 2013-05-14 19:12 CEST) to contact phishing sites have been prevented.
6 Response Policy Zone Log Analysis: RPZLA

Once using RPZ, one has the opportunity to use the log data generated by BIND, and a web server (if one is using a 'walled garden' (CNAME redirect)).

The availability of this data allows various heuristics to be used on the data to assist in the identification of infected systems.

To allow this analysis the author built a toolset known as RPZLA [17].

RPZLA is a FOSS project. Take it and do what you will.

The software is in 'alpha'. Bugs need fixing. It is a query interface and does not update the data it displays. Thus, any bugs affect the display interface, but not the data.

6.1 Overview

RPZLA comprises a collection of log scrapers, which read BIND recursive resolver logs and/or Apache (walled garden) logs and ship them to a PostgreSQL database, and an Analysis Website which displays the shipped data.

Once established RPZLA builds and ever growing collection of data obtained from the recursive resolvers and 'walled garden' which can be used to identify systems that may be compromised.

Using DNS RPZ for Security Analysis

The original diagram, with better resolution, is also viewable [18].

6.2 Notes on Observed Data

The architectural decision to use a walled garden (CNAME) approach was largely based on community awareness, but also on the idea that one could then differentiate between
human behaviour (person clicking on a link in a phishing email) and general malware
behaviour.
This assumption was poor. Malware sometimes behaves exactly like a browser
(HTTP protocol and following DNS redirects).
Indeed, much of the data indicates that community members are visiting news (or
similar) web sites that are including malicious advertisements. Thus, DNS records an
RPZ hit as the page is loading, but the person is then not clicking on the advertisement
and thus no log entry appears in the walled garden. Often one has numerous RPZ hits
from numerous nasty advertisements, at the same time on the same client.
Thus, differentiating between nasty advertisements, at the same time on the same client.

6.3 Timing Analysis
The natural distinction between a human and malware is that the malware is automated,
and will thus often display regular behaviour in time. This is somewhat obfuscated by
browser add-ons which regularly contact nasty domains, or by JavaScript which cause
regular page reloads. Browser add-ons which regularly contact nasty domains are malware
embedded in a browser. Some may consider JavaScript which regularly reloads a page
as malware.
Humans do not visit domains with chronologically tight regularity.
The data of a timing analysis is presented in Appendix [A].

7 Future Directions
7.1 Phishing Attack Defence
The current process of the community forwarding phishing attack emails to the IT group
to create a defence is poor. Its better than nothing, but it relies on the awareness of the
IT group and their manual effort.
A better strategy may be to build a more automated mechanism.
A delivery point for the community can be created, to which they forward phishing
attacks. The delivery point scans the delivered email and removes a known whitelist
of domains and then forwards the redacted email to Spamhaus’ engine. Thus, attacks
are rapidly redacted and forwarded to Spamhaus so that we can receive their updated
domain list after their engine analyses the emails.

7.2 RPZLA
RPZLA is currently a data display engine targeted at engineers.
To use it to identify potentially compromised systems involves human effort (quantity
of queries, timing differences etc.)
RPZLA could become a more ‘where should I place effort’ focussed user interface
that identifies potentially infected systems via a collection of heuristics, and highlights
suspicious systems.
References

[1] Creative Commons, *Creative Commons Attribution-ShareAlike 3.0 Unported*
http://creativecommons.org/licenses/by-sa/3.0/legalcode

http://en.wikipedia.org/wiki/Domain_Name_System

http://en.wikipedia.org/wiki/Domain_Name_System#Recursive_and_caching_name_server

http://en.wikipedia.org/wiki/Domain_Name_System

http://en.wikipedia.org/wiki-Mail_Abuse_Prevention_System

https://www.isc.org/software/bind/documentation

http://www.spamhaus.org/consumer/definition/

http://en.wikipedia.org/wiki/Internet_bot

http://en.wikipedia.org/wiki/Botnet


http://en.wikipedia.org/wiki/Paul_Vixie

http://www.youtube.com/watch?v=k5DbNgEXDHo
Appendices

A Timing Analysis

A.1 Raw Data

Here are the RPZ ‘hits’ of a single client to a single domain with date/time stamps, and the differences in time between successive events, rounded down to 10 second intervals.

<table>
<thead>
<tr>
<th>Date/Time</th>
<th>Difference</th>
<th>Date/Time</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>2013-05-03 15:00:31.816</td>
<td>900</td>
<td>2013-05-05 00:08:42.770</td>
<td>950</td>
</tr>
<tr>
<td>2013-05-03 15:30:36.445</td>
<td>1800</td>
<td>2013-05-06 00:15:33.463</td>
<td>410</td>
</tr>
<tr>
<td>2013-05-03 16:30:49.315</td>
<td>900</td>
<td>2013-05-06 00:30:37.384</td>
<td>350</td>
</tr>
<tr>
<td>2013-05-03 16:45:52.233</td>
<td>900</td>
<td>2013-05-06 00:40:35.358</td>
<td>590</td>
</tr>
<tr>
<td>2013-05-03 17:31:04.298</td>
<td>2710</td>
<td>2013-05-06 01:30:59.335</td>
<td>3020</td>
</tr>
<tr>
<td>2013-05-03 18:01:11.023</td>
<td>1800</td>
<td>2013-05-06 01:44:25.020</td>
<td>800</td>
</tr>
</tbody>
</table>
A.2 Difference Analysis

If one summarises the above data by grouping differences into 10 second 'slots', and disregarding slots with only one or two entries in them, one gets the following picture:
Table 2: Single Client RPZ Bucket Analysis

<table>
<thead>
<tr>
<th>Rounded Difference</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>190</td>
<td>3</td>
</tr>
<tr>
<td>260</td>
<td>3</td>
</tr>
<tr>
<td>280</td>
<td>3</td>
</tr>
<tr>
<td>740</td>
<td>3</td>
</tr>
<tr>
<td>900</td>
<td>37</td>
</tr>
<tr>
<td>950</td>
<td>79</td>
</tr>
<tr>
<td>960</td>
<td>5</td>
</tr>
<tr>
<td>1800</td>
<td>17</td>
</tr>
<tr>
<td>1910</td>
<td>16</td>
</tr>
<tr>
<td>2700</td>
<td>4</td>
</tr>
<tr>
<td>2860</td>
<td>3</td>
</tr>
<tr>
<td>2870</td>
<td>7</td>
</tr>
<tr>
<td>2880</td>
<td>3</td>
</tr>
<tr>
<td>7430</td>
<td>3</td>
</tr>
</tbody>
</table>

General statistics are:

- total readings: 287
- number of (disregarded) single or double value time slots: 85

The peaks at 900 and 950 seconds are instructive. They are mirrored by peaks at double their values (1800 and 1910 seconds).

This is almost certainly not human behaviour. It seems likely that, the system is infected with malware, or a browser is visiting a page with nasty advertising and is regularly reloading the page, amongst other possibilities.