

SPAMHAUS
KNOW HOW
<email_filtering/>

Spamhaus Know How eBook

Getting the most from email filtering



Contents



Foreword

Page 3-4



Busting blocklist myths

Page 5-10



Understanding the source code of a malicious emails

Page 11-16



What blocklists to apply to email elements

Page 17-18



DNSBLs and email filtering – how to get it right & flow chart

Page 19-25



Foreword

Written by Simon Forster



After being immersed for years in the world of Spamhaus, with its deep technical focus, it's easy to forget life before it became filled with reputation data, DNS blocklists, Response Policy Zones, and Passive DNS data.

I'll readily admit that I had nerd-like tendencies before joining Spamhaus. Witness the (badly written) code, long since consigned to the digital dustbin.

However, when I talk with peers or have discussions with organizations who are struggling to achieve high catch rates, I am reminded of how niche our area of the industry is. The Spamhaus team's expertise is considerable, and we sometimes take it for granted, expecting all those who use our data to have the same level of understanding as we do.

“ The aim is simple - to make it easier to understand how to use IP, domain, and content reputation data to protect your email stream, your users, and your network. ”

With this in mind, we are sharing some nuggets of DNSBL wisdom. Firstly we're addressing the various myths we regularly hear bandied around, relating to blocklists.

Secondly, we have broken down the complexities of using our blocklists in the email filtering process into an easy to follow best practice.

The aim is simple - to make it easier to understand how to use IP, domain, and content reputation data to protect your email stream, your users, and your network. It's a long read but definitely worth it.

Happy reading.

Simon Forster, CEO



Busting blocklist myths

Written by The Spamhaus Team



There are various comments bandied about with some regularity relating to Domain Name System Blocklists (DNSBLs). So, we thought it was about time to provide some truths when it comes to blocklists and dispel these myths.

I Blocklists are old technology

The delivery mechanism for DNSBLs has been around for years. However, the threat intelligence within blocklists is continually evolving to reflect current dangers, as are the means by how this data is researched and published.

Tried and tested

There is a vast difference between “old and dated” and “tried and tested.” One can’t deny that blacklist data’s delivery mechanism, i.e. via [DNS zone](#), is decades old. In fact, the first DNSBL was created in 1997 by [Eric Ziegast, who was working at MAPS with its founder, Paul Vixie](#).

But, and this is a big BUT, DNSBLs are a light, robust, and convenient real time delivery mechanism for reputation data relating to IPs, domains, and content. They are continually developing to counter nefarious online activity.

New research technologies

The early days of blacklist creation involved hours of tireless manual research and the use of heuristics.

“ By leveraging new tools and technologies, we continue to identify bad actors and bad behavior while limiting the number of false positives. ”

Today, machine learning enables us to process an ever-increasing amount of data, with our researchers still undertaking manual investigations and applying heuristics.

The type of datasets that are produced is also advancing. In addition to IP and domain reputation data, [hash blocklists](#) are compiled, enabling specific content such as email addresses, malware files, and cryptowallet addresses to be listed.

By leveraging new tools and technologies, we continue to identify bad actors and bad behavior while limiting the number of false positives. We spend considerable resources advancing our threat hunting capabilities, allowing our customers to place their focus elsewhere.

Busting blacklist myths (continued)

Written by The Spamhaus Team

So, while DNSBLs' delivery mechanism might be considered "old," the actual data within is carefully researched using up to the minute technology and techniques to deliver IP, domain, and content reputation data that protect against current threats.

2 Blocklists only stop spam

Er, no. DNSBLs can filter all manner of email-borne threats, including phishing links, malware files, bots, and URLs to malware dropper sites. Spam is just the tip of the iceberg.

Clarification

Firstly, let's define spam.... Obviously, we don't mean the trademark name of the famous processed meat product. We are talking about unsolicited emails that are sent in bulk to a large number of recipients.

Most people consider spam to be the umbrella term for the plethora of scummy emails distributed, selling pills, sex, and lists, among other undesirable things.

Email, though, is also a vector for more sinister issues like malware files ([94% of which are delivered via email](#)), ransomware, phishing sites... and, so the list goes on. Blocklists can mitigate the risks from these and more.

Blocking malware files

Emails that have malware files attached can be blocked with the malware component of the afore mentioned Hash Blocklist. This lists cryptographic hashes of malware files. The Spamhaus researchers can list malware files within 20 seconds of initially observing them.

Blocking domains

[Domain blocklists \(DBL\)](#) contain lists of domains that are not only associated with sending spam but also those associated with phishing, malware, and botnet command and controllers (C&Cs).

“ DNSBLs can filter all manner of emailborne threats, including phishing links, malware files, bots, and URLs to malware dropper sites. ”



Blocking compromised machines

Our [eXploits Blocklist \(XBL\)](#) only lists IP addresses (approximately 13 million of them) that are showing signs of compromise, i.e., machines infected with malware, worms, and Trojans, or third-party exploits such as open proxies or devices controlled by botnets.

You'll note that there isn't a single mention of the word 'spam' in that litany of internet baddies. Where a machine is compromised, it's highly likely that further criminal activity will follow, including spewing out emails with malware attachments to all available contacts to further propagate and increase its reach or the download of its log-in and financial credentials.

Enough said?

We could continue, but we figure that we've made the point that blocklists filter out a whole lot more than just spam!

3 By the time entries are in a blocklist; it's too late

The belief that blocklists are purely reactive, i.e., an IP, domain, or hash is only listed once observed to be behaving maliciously, is simply not accurate.

Spamhaus spends a significant amount of time investigating internet environments and resources. Our researchers are continually looking for indicators that suggest a property is about to participate in malicious activity. Here are some examples of where blocklists are proactive in their listings:

Hijacked IP ranges

Where we [observe a hijacked IP range](#), it is (almost) a dead cert that nefarious activity involving that range will follow. Therefore, we will list the respective IP range(s) to block any potential attack vectors they may emit.

Poor domain reputation

Spamhaus is provided with lists of domains, as and when they are registered, and we proactively investigate their various proprieties. From this analysis, our researchers can score the domain's reputation, evaluating its likelihood of being used with malicious intent. These are then listed before the domains are used in the wild.

Newly registered domains

Our [Zero Reputation Blocklist \(ZRD\)](#) lists all newly registered domains for 24 hours. When a new domain is initially registered, it's improbable that emails will be sent from it immediately, unless it's being used by bad actors, who register and burn through hundreds and thousands of domains.

IP ranges that shouldn't be sending email

The Policy Blocklist (PBL) that Spamhaus supports enables Internet Service Providers (ISPs) to list ranges of IP addresses that should never be sending email directly to the internet. This is an entirely pro-active blocklist.

Reactive intelligence can still protect

Spamhaus has reduced latency, so when attacks occur, it takes less than 20 seconds from detection for an IP, domain, or hash to be listed. For large, vicious campaigns, our data may not protect users hit at the initial start of the attack, but in the case of a hailstorm campaign, which runs for up to 5 minutes, 90% of those affected will have protection.



4 All blocklists are created equal

This is far from true in many ways. Having read this far, you will quickly be realizing (if you hadn't already) that no blocklist is the same. Each has separate listing policies, i.e., the criteria used to decide as to whether something should be listed or not. Similarly, the organizations that research and compile blocklists are not the same. While starting a blocklist is relatively easy, maintaining its consistency is difficult (and crucial).

Different threats and different places to apply the data

Given that every blocklist has a different focus, they need to be applied to the right part of an email transaction. For example, using the PBL in deep header parsing would be a recipe for disaster. Follow our [best practice](#) to ensure you are implementing them correctly.

Different risk profiles call for different blocklists

Always make sure you understand the functionality of each blocklist you use. Spamhaus, for example, has an "Abused legit" section to the DBL. Here hacked legitimate resources are listed. You can choose whether to ignore or act on the query response relating to this part of the DBL. Be aware that it can potentially increase your risk of false positives. You need to weigh-up your organization's appetite for risk versus the operational issues relating to an increased number of false positives.

Consistency is king

When it comes to blocklists, providers need to be consistent. Each listing needs to adhere to its particular policies, day in, and day out, which isn't easy to do.

5 Blocklists should be used only during the SMTP transaction

Incorrect! Suppose you use blocklists only during the SMTP transaction. In that case, you miss the opportunity to scrutinize many elements, such as URLs, Reply-To email addresses, files sent as attachments, etc., against reputation data.

While it is true that the SMTP stage alone can block 95% of the spam or more, a fraction of nasty messages carrying threats could still pass through.

6 Blocklists should be used only during content analysis

No! If you are only using blocklists during content analysis, you are missing a huge opportunity to immediately drop a massive percentage (see that 95% figure above?) of unwanted email at the initial IP connection and during the SMTP transaction before transferring data.

Dropping malicious email before it reaches the expensive content-filtering stage enables you to save on bandwidth, significantly reduce the number of open connections, conserve storage, and benefit from a general reduction in memory and CPU requirements.

7 Only IP blocklists should be used to filter email during the SMTP transaction

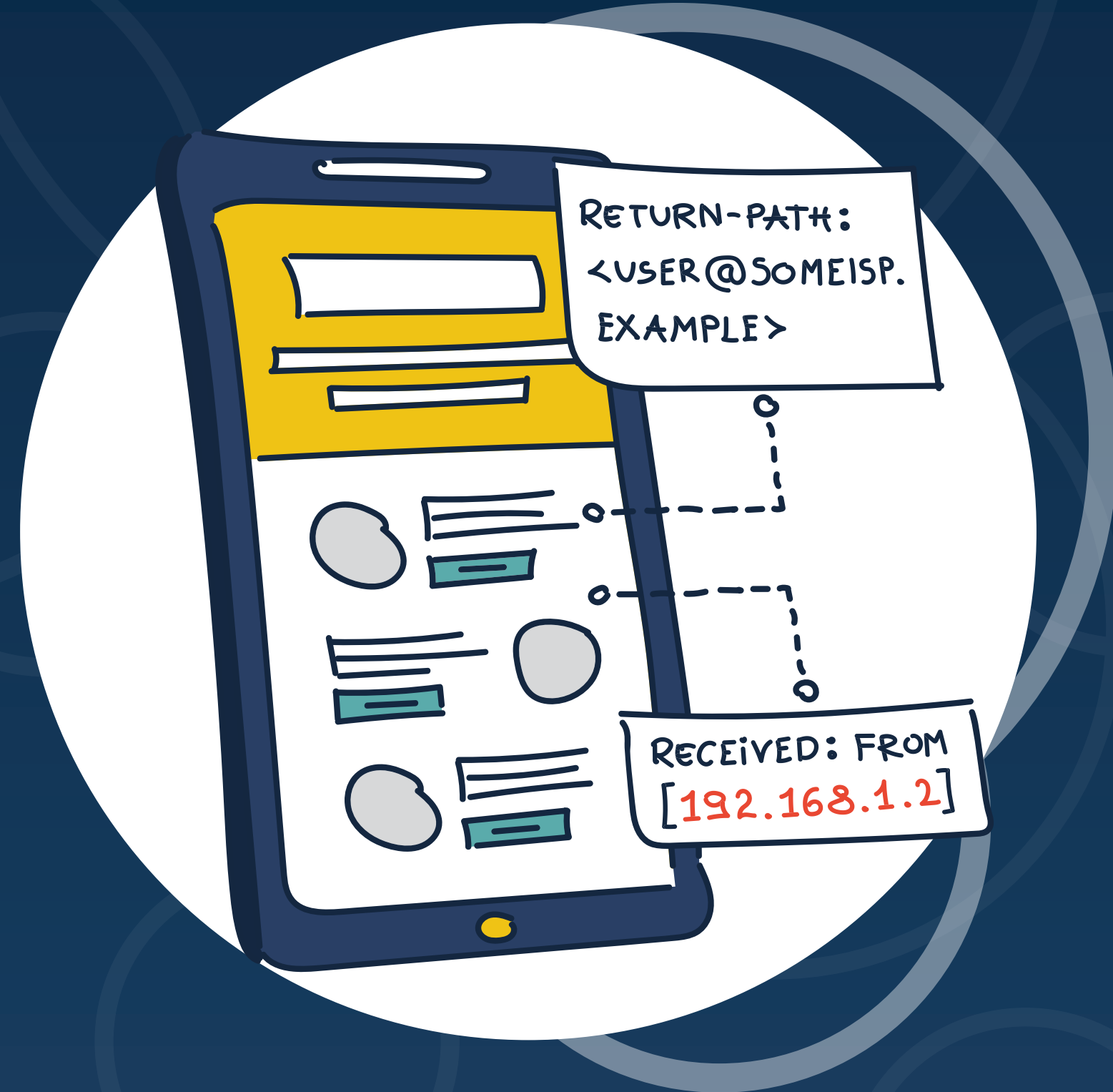
Absolutely not! As discussed, Spamhaus lists a number of domains before they're seen in the wild. Therefore, querying the DBL and ZRD at the appropriate stages of the SMTP transaction provides additional proactive protection.

What myth would you like busted?

That's enough myth-busting for one day. If you've any other myths relating to blocklists and Spamhaus, let us know via Twitter or LinkedIn - we'd be delighted to bust those too.

Or, if you'd like to trial our DNS Blocklists, then [sign up](#) to the data, free for 30 days. We'd love to know how you get on.





Understanding the source code of a malicious email

Written by Riccardo Alfieri

Understanding the source code of a malicious email

Written by Riccardo Alfieri



When an end-user views an email, it isn't always apparent if it's malicious or not. However, if you look at the email's source code, a wealth of information is revealed. Here's the anatomy of an email...

An email as it looks to an end-user

Let's put some context around the difference between the information that an end-user sees when viewing a malicious email and the actual data contained within the source code. Firstly, take a look at the email below:

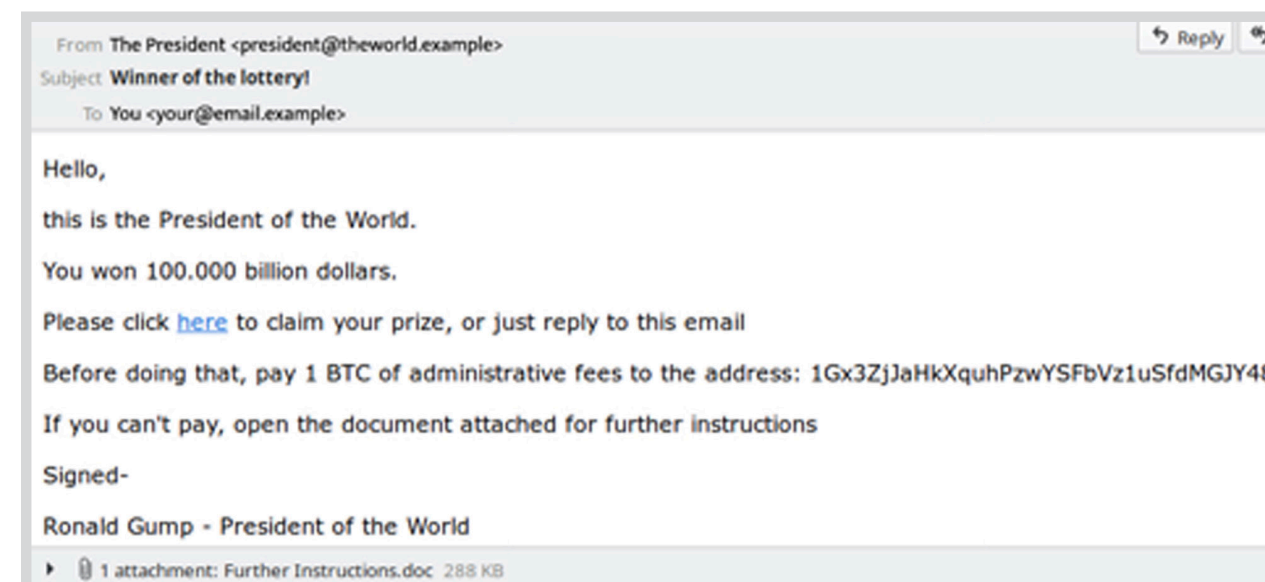


Figure 1 | End-user view of an email

Above is a typical example of how a spam email displays via end-user client software, such as Apple Mail or Outlook. In the rendering of this email, some elements of the email content may start to ring some warning bells, i.e., an HTML link, a bitcoin address and an attachment. Unfortunately, to a casual user, these elements will probably seem benign.

An example of email source code

On [page 13](#) is the same email as pictured to the left, only in "raw" format. It illustrates what information is seen and recorded by internet servers as they send and deliver email messages.



Understanding the source code of a malicious email (continued)

Written by Riccardo Alfieri

```
Envelope from Return-Path: <user@someisp.example>
The received chain
1 B Received: from smtp.someisp.example (smtp.someisp.example [192.0.3.1]) by mta-out.someisp.example (Postfix) with ESMTPTPSA id 4579820702 for <your@email.example>; Mon, 18 May 2020 10:18:00 +0200 (CEST)
2 C Received: from mta-out.someisp.example (HELO mta-out.someisp.example) (192.0.4.1) by mx.email.example (qpsmtpd/0.80) with (AES256-SHA encrypted) ESMTPTPS; Mon, 18 May 2020 08:19:25 +0000
3
4
5
6 Reply to address Reply-to: The Prez <prezident@gmail.example>
7 Sender-address From: The President <president@theworld.example>
Friendly-From

Subject: Winner of the lottery!
To: You <your@email.example>
MIME-Version: 1.0
Content-Type: multipart/mixed;
Content-Language: en-US
This is a multi-part message in MIME format.
-----4FBA580188D90BB4BF89E5DD
Content-Type: multipart/alternative;
boundary="-----7F511474A638B241A042D452"
-----7F511474A638B241A042D452
Content-Type: text/html; charset=utf-8
Content-Transfer-Encoding: 7bit
```

Email body

```
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
</head>
<body>
<p><font face="Verdana">Hello,</font></p>
<p><font face="Verdana">this is the President of the World.</font></p>
<p><font face="Verdana">You won 100.000 billion dollars.</font></p>
URL <p><font face="Verdana">Please click <a moz-do-not-send="true" href="http://www.worldlotterywinner.example/guhPzxYSFb">here</a> to claim your prize, or just reply to this email</font></p>
8
9
Bitcoin Wallet <p><font face="Verdana">Before doing that, pay 0.01 BTC of administrative fees to the address: <code>1Gx3ZjJaHkXquhPzwYSFbVz1uSfdMGJY48</code></font></p>
<p><font face="Verdana">If you can't pay, open the document attached for further instructions</font></p>
<p><font face="Verdana">Signed-</font></p>
<p><font face="Verdana">Ronald Gump - President of the World</font><br>
</p>
</body>
</html>
-----7F511474A638B241A042D452--
-----4FBA580188D90BB4BF89E5DD
Attachment Content-Type: application/msword;
name="Further Instructions.doc"
Content-Transfer-Encoding: base64
Content-Disposition: attachment;
filename="Further Instructions.doc"
...
```

Email source code elements explained

The Received Chain

Think of the “Received” chain as a paper trail of all the servers that have handled your email, including your mail server. The reading order is “final connection/most recent” at the top, and the preceding connections are listed in reverse order, culminating with the “originating computer/oldest” at the bottom.

These are vital elements because they include all the IP addresses of the email servers that have handled your email and information like the HELO string and reverse DNS (rDNS[1]) of the IPs.

What is the “HELO“?

This is a required element in a Simple Mail Transfer Protocol (SMTP) transaction. It is similar to the greeting that a postman might give to a colleague when handling over postal mail: “Hello, I am John from The Postal Service, and I have mail for someone on your route.”

Received Chain C – The most recent received line:

This is the most recent and most trusted “Received” line because the recipient’s mail server has registered it; therefore, it’s guaranteed not to have been tampered with.

Things to watch out for:

In an attempt to confuse anti-spam systems, miscreants will sometimes try to scramble the “Received” line chain. With this in mind, you should pay particular attention to the subsequent “Received Lines” following on from this most trusted one.

```
Envelope from Return-Path: <user@someisp.example>
The received chain
C Received: from mta-out.someisp.example (HELO mta-out.someisp.example) (192.0.4.1) by mx.email.example (qpsmtpd/0.80) with (AES256-SHA encrypted) ESMTPS; Mon, 18 May 2020 08:19:25 +0000
B Received: from smtp.someisp.example (smtp.someisp.example [192.0.3.1]) by mta-out.someisp.example (Postfix) with ESMTPE id 4579820702 for <your@email.example>; Mon, 18 May 2020 10:19:19 +0200 (CEST)
A Received: from [192.168.1.2] (unknown [192.0.2.1]) by smtp.someisp.example (Postfix) with ESMTPE id 323AB5EF4D for <your@email.example>; Mon, 18 May 2020 10:19:19 +0200 (CEST)
Reply to address Reply-to: The Prez <prezident@gmail.example>
Sender-address From: The President <president@theworld.example>
/ Friendly-From Subject: Winner of the lottery!
```

The key elements in Line C are:

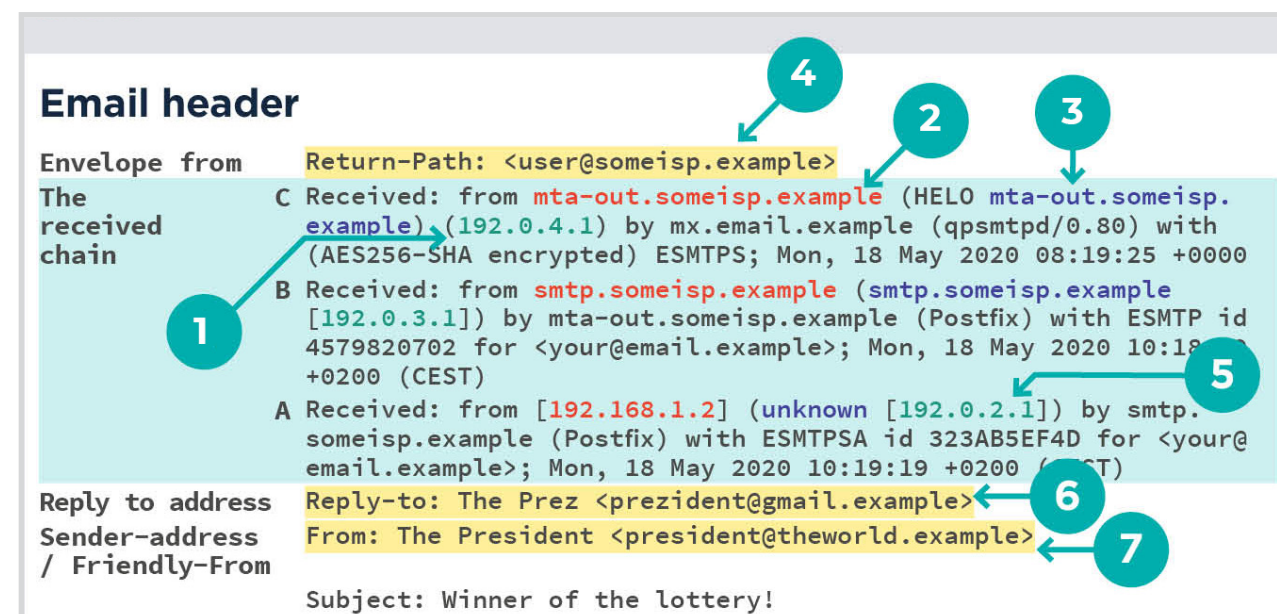
- Connecting IP: 192.0.4.1 (#1)
- rDNS of connecting IP: mta-out.someisp.example (#2)
- HELO string: mta-out.someisp.example (#3)

Explanation: the sending mail server “mta-out.someisp.example” (#3) makes a connection to the recipient’s mail server “mx.email.example”, announcing itself via HELO with the IP “192.0.4.1” (#1) and the rDNS of “mta-out.someisp.example” (#2). This is the final connection – if it is accepted (and in this case it was), then the email addressed to <your@email.example> is delivered.

Received Chain B:

This is where “smtp.someisp.example” acknowledges receipt of the email for the destination email address <your@email.example>.

Explanation: the mail server “smtp.someisp.example” connects to the outbound (Postfix) mail server “mta-out.someisp.example” for that ISP, telling the outbound that it has mail to be sent to <your@email.example>.



Received Chain A - the original transaction

This line represents the original email transaction that generated the email. Since this is the initial transaction, the pertinent detail here is the connecting IP.

The key elements in Line A are:

- Originating local machine : [192.168.1.2]
- rDNS of connecting IP: unknown
- Connecting IP: 192.0.2.1 - (#5)

Explanation: the originating local machine, “192.168.1.2”, makes an outgoing connection to “smtp.someisp.example” and announces itself as “192.0.2.1”. Upon connection, the server “smtp.someisp.example” tries to look up the rDNS of the connecting IP, “192.0.2.1” (#5), and fails, returning in an “unknown” result.

Return-Path (#4)

The “Return-Path” is also known as the “Envelope-From” or “Mail-From.” This is the “actual” email address that sent the email.

In terms of old-fashioned snail-mail, the “Return-Path” is similar to a local postal service stamp, i.e., it is an accurate indicator of the letter’s origin. However, when you’re reading the actual contents of the letter, you can’t see the postal service stamp on the envelope.

In the case of email, you can’t see the sender’s real email address in the visual rendering of an email, as illustrated in figure 1.

Reply-To address (#6)

Upon clicking “Reply” to an email, this field serves to instruct your email client software regarding what email address to insert in the “To” field. Usually, with legitimate emails, the “Reply-to” and “From” in the original email will be the same.

Our email example shows that the apparent sender is “president@theworld.example,” but if you were to reply to this email, the address would populate as “prezident@gmail.example”... and the bad actors are hoping you won’t notice that.

Sender-Address (From) (#7)

This is also commonly referred to as the “Friendly From.” The sender-address is comparable to the signature of the person who signs a letter you’ve received through the post, i.e., they can sign it from anyone.

In our example, the “Return-Path/Envelope-From” (#1) is different from the “Sender-Address (From)” (#7), aka “Friendly-From.” The Friendly-From name, which is visible to an end-user, is how the sender wants to appear to the recipient. Typically, it includes the sender’s name and email address.

Remember! In most cases, email client software only shows the Friendly-From. It's like opening a letter and throwing away the envelope before handing it to someone to read; they only see the writing on the actual letter.

Why do miscreants use different Return-Path and Friendly-From email addresses?

That's simple – to trick people! Bad actors want the recipient of the email to believe it's from someone credible, important, or someone who is known to them. They are trying to add legitimacy and believability.

```
URL      <p><font face="Verdana">Please click <a moz-do-not-send="true"
href="http://www.worldlotterywinner.example/guhPzxYSFb">here</a>
to claim your prize, or just reply to this email</font></p>
Bitcoin  <p><font face="Verdana">Before doing that, pay 0.01 BTC of
Wallet   administrative fees to the address
        1Gx3ZjJaHkXquhPzwYSFbVz1uSfdMGJY48</font></p>
        <p><font face="Verdana">If you can't pay, open the document
        attached
        for further instructions</font></p>
        <p><font face="Verdana">Signed-</font></p>
        <p><font face="Verdana">Ronald Gump - President of the World</
font><br>
        </p>
        </body>
        </html>
Attachment  -----7F511474A638B241A042D452--
-----4FBA580188D90BB4BF89E5DD
Content-Type: application/msword;
name="Further Instructions.doc"
Content-Transfer-Encoding: base64
Content-Disposition: attachment;
filename="Further Instructions.doc"
...
<snip>
```

Things to watch out for:

In many cases, the URLs included in malicious emails will be directing you to a “spamvertised” product, a phishing website, or a malware dropper site.

URL (#8)

The end-user has no visibility of the URL (website address) in the email they are viewing. This is because the sender has concealed the URL with a hyperlink associated with “click here.” The actual website URL is “www.worldlotterywinner.example/guhPzxYSFb”.

To see where a link will take you, hover over the link, and the details of the URL will display.

Bitcoin wallet (#9)

A Bitcoin “wallet” is a crypto-address controlled by cybercriminals, often used as an avenue for payment in sextortion spam campaigns.

Attachment (#10)

In this example, the attachment is an Office document called “Further instructions.doc.” More than likely, it will be ‘weaponized’ in some shape or form.

What do we mean by “weaponized”

If you were to download and open the document, malware would infect your computer. The cybercriminal can then use your computer for whatever purpose they want. This can include sending more spam, [exfiltrating](#) your personal data/passwords, or utilizing your computer in a [distributed denial-of-service \(DDoS\)](#) attack.

See [page 18](#) to see an easy to refer to guide as to what blocklists you should apply to these email elements.



What blocklists to apply to email elements

Crafted by The Spamhaus Team

What blocklists to apply to email elements

Crafted by The Spamhaus Team

Element	Example	SMTP Phase	IP Blocklists (ZEN)				Content Blocklists		
			Spamhaus Blocklist (SLB)	Exploits Blocklist (XBL)	Policy Blocklist (SLB)	Auth BL	Domain Blocklist (DBL)	Zero Reputation Domain Blocklist (ZRD)	Hash Blocklist (HBL)
1	Received Chain - Line C (Connecting IP)	192.0.41	Initial Connection	✓	✓	✓			
2	RDNS of connecting IP	mta-out.someisp.example	Pre-data Phase (SMTP Transaction)				✓	✓	
3	Helo string	mta-out.someisp.example		✓*			✓	✓	
4	Mail-from/Envelope-from/Return-path	user@someisp.example		✓*			✓	✓	✓
5	Received Chain - Line A (Original email)	Originating IP 192.02.2.1	Post-data Phase (Content Inspection)	✓			✓	✓	
6	Reply-to address	prezident@gmail.example							✓
7	Sender-address (From)	The President <president@theworld.theworld.example>		✓*			✓	✓	✓
8	URL or email address	http://www.worldlotterywinner.example/guhPzxYSFb		✓*			✓	✓	✓***
9	Bitcoin wallet	1Gx3ZjJaHkXquhPzwYSFbVz1uSfdMGJY48							✓
10	Attachment	Further Instructions.doc						✓	

*Forward DNS lookup of the domain required to check against SBL

**Reverse DNS lookup required to check against DBL/ZRD

***Entire URL checked against HBL



DNSBLs and email filtering

Written by The Spamhaus Engineers

DNSBLs and email filtering – how to get it right

Written by The Spamhaus Engineers



Domain Name System Blocklists (DNSBLs) are a low-cost and effective way of stopping the torrent of inbound email traffic associated with spam and other malicious emails.

There are multiple benefits to using blocklists: reducing infrastructure costs, and [workforce hours](#) to [increasing catch rates](#). However, to get the most from DNSBLs, it's vital to use them at the right points in your email filtering process.

Efficient and effective filtering

There's a wide range of email security options for your infrastructure, from content filtering and scanning to network-based tools that use distributed and collaborative methods.

Unfortunately, the majority of the above are resource-heavy and expensive. The key to email filtering is to remove the bulk of unwanted emails right at the beginning of the process before they reach the resource-intensive content inspection stage.

A DNSBL check is the simplest, quickest, and most economical method you have at your disposal....so why not use it?

Stages to utilize DNSBLs

When you accept and process an email, there are three areas that you should use blocklists:

- 1 At the initial connection – against the connecting IP
- 2 Throughout the pre-data phase of an email, i.e., the SMTP transaction
- 3 Once the email data has been accepted i.e., during content inspection

Benefits of using DNSBLs before content filtering

- **Saves on bandwidth** – you significantly reduce the number of messages that require downloading.
- **Saves on open connections** – many connections are not needlessly held open.
- **Saves on storage activity** – rejection before content filtering doesn't use any storage resources (with the exception of a few logging lines).
- **General reduction in memory, storage, and CPU requirements** – you have a far lower volume of emails that require expensive content scanning.

1 Using DNS blocklists at the initial email connection

At the very beginning of an email transaction, a remote server attempts to initiate a connection with your server. At this point, a quick lookup (query) to a blocklist can instantly (well, almost instantly) advise if the connecting IP address is associated with malicious behavior or not.

If the IP address is listed, you can choose to drop the connection immediately. Remember, this query is undertaken even before the Simple Mail Transfer Protocol (SMTP) handshake occurs.

What blocklists to use:

- Spamhaus Blocklist (SBL)
- eXploits Blocklist (XBL)
- Policy Blocklist (PBL)



2 Using DNS blocklists during the pre-data phase

What is the pre-data phase?

For all email commands that occur after the initial email connection set-up (as detailed above) and before the DATA command during the SMTP connection, we term as the pre-data phase.

Check the reverse DNS of the connecting IP

Once the connection is open, you (the receiver) should do a [reverse DNS](#) lookup of the connecting IP address.

Does the result of the rDNS check match the HELO string?

The result of the above lookup should be queried against the domain contained in the HELO from the sender, e.g. (HELO mta-out.someisp.example).

DNSBLs and email filtering – how to get it right

Written by The Spamhaus Engineers

Query the result of the rDNS against blocklists

The domain returned should be queried against the DBL and the ZRD.

Query the domain contained in the HELO against blocklists

The domain contained in the HELO should be queried against the DBL and the ZRD, alongside running a forward DNS lookup against the SBL.

Query the MAIL FROM (Return-path) domain

Subsequent to the HELO, during the SMTP connection, is the MAIL FROM command. This is often referred to as the “Envelope-from” or the “Return-path.” You should query the domain included in this string, e.g., <user@someisp.example>.

Whenever a query is made against the DBL and ZRD and a positive response is returned, receivers can reject the email or tag it for further inspection during the content filtering process.


The transfer of the email header and content


Once the RCPT TO command has been completed, the DATA command follows, and it is this that initiates the transfer of data, both the email header and content. After the receiver confirms this has been successfully executed, the sender transmits a QUIT command, either before or after content inspection, and the connection is closed.

Is the connection kept open at content inspection?


This comes down to both your implementation decisions and software choices. As with all options there are pros and cons for both.

Some implementations choose to keep the connection open, until they have the result of the content inspection.

 **Pros:** Senders will always be notified of a rejection, which is the best strategy should false positives occur.

 **Cons:** This is resource heavy and your MTA must be up to the task, even during peaks. Where resources are not sufficient delays may occur.

The majority of implementations close the connection before content inspection.

 **Pros:** Lowers costs due to lower resource requirements.

 **Cons:** Where false positives have occurred, the sender will not be notified.

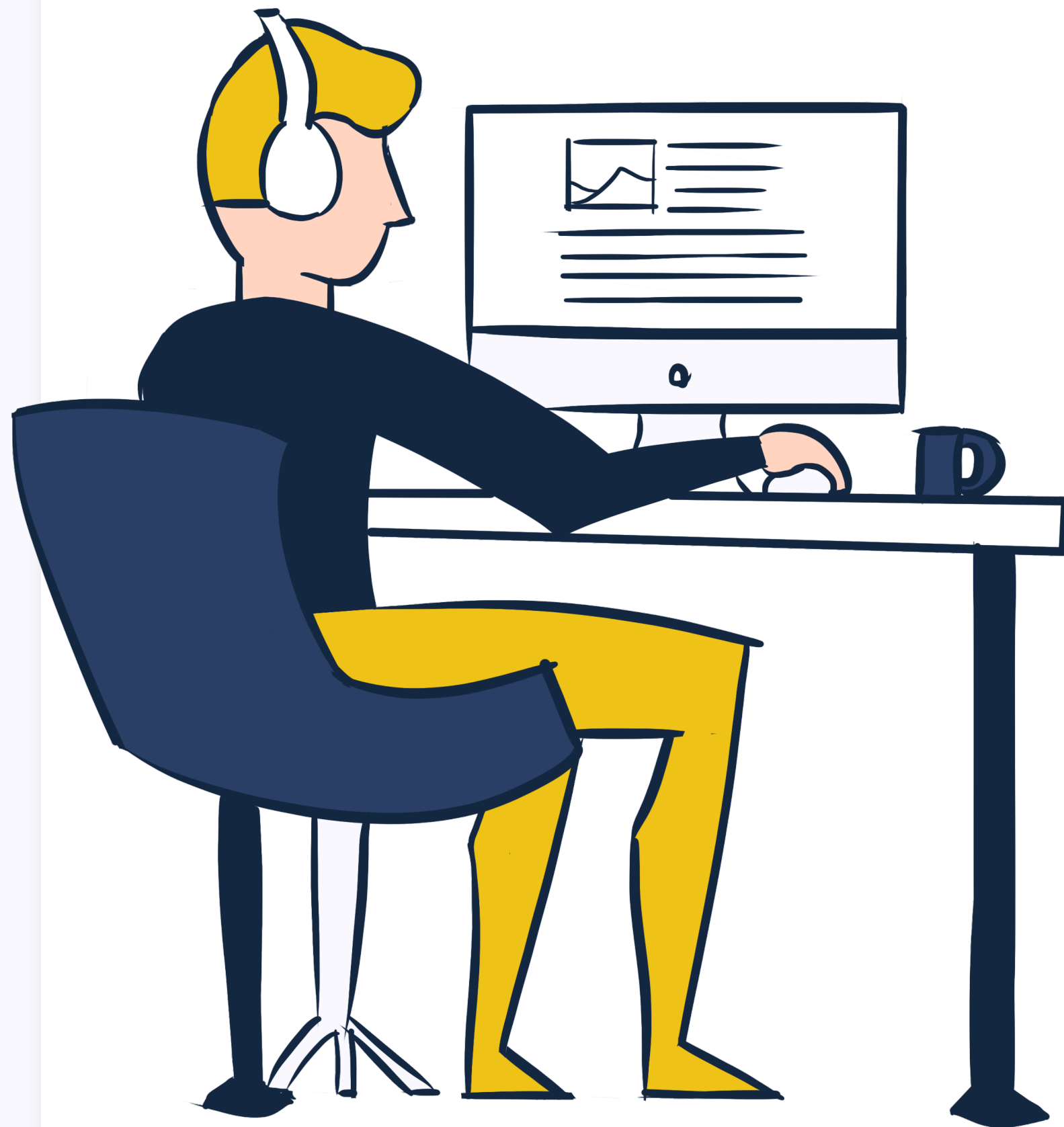
A deeper dive into the benefits of using DNSBLs before content filtering

Reducing the load placed on your resources

As previously mentioned, content filtering and scanning are resource heavy. Reducing the number of emails that reach this point of the email filtering process saves on memory, reduces processing power and reduces latency.

DNSBLs and email filtering - how to get it right

Written by The Spamhaus Engineers



Infrastructure protection against spam campaigns

Consider a situation where you receive hundreds of messages per second during a sustained or escalating spam campaign. If you are reliant on content filtering or scanning to mitigate the attack, your email infrastructure has the potential to collapse under the load put on it by the sheer volume of inbound emails.

If you're lucky enough for this not to happen, you will still incur high costs, as every email needs to be downloaded and scanned.

Senders can debug mail relay issues

When an email is rejected at the initial connection or SMTP handshake, a Delivery Status Notification (DSN) is returned in real-time to the sender, advising them of the failed delivery. From a sender's perspective, this can help with identifying and resolving mail relay problems.

Emails aren't left to linger in junk folders

As a result of emails being rejected early on in the email transaction, potentially spammy emails aren't downloaded and left, forgotten in a junk folder. This stops end-users from hunting through their junk folder to find one legitimate email out of hundreds, if not thousands of spam ones.

Innocent 3rd party infrastructures aren't flooded with return messages

If the sender address is forged, the innocent third party whose email was used in the forged sender address will be flooded with return messages.

Increased protection from hailstorm and snowshoe spam campaigns

Using domain blocklists as well as IP blocklists provides you with extra protection against hailstorm and snowshoe spam, in addition to other threats. For a deeper understanding of this subject matter, we suggest you read [MTA developers: allow use of domain DNSBLs at the SMTP level](#).

It's worth remembering that our domain blocklist will contain many domains before they're seen in the wild!

3 Using DNS blocklists during the post-data phase

Once the DATA command has run and the email header and content have transmitted to the receiver, content inspection commences, alongside running SPF, DKIM, and DMARC checks.

Email headers contain structured data making them far less resource-intensive to parse compared to that of the body of an email. To get a deeper understanding of an email header and body components, read [Understanding the source code of a malicious email](#).

Here's where blocklists can be used during the header inspection:

Query the originating IP address:

This is the IP address contained within the original transaction's received chain, i.e., what initially generated the email. This lookup can be done against the SBL and AUTH BL. An rDNS lookup can also be done against the DBL & ZRD.

Query the Reply-to address:

The full email address in this field should be queried against the Hash Blocklist (HBL), which contains cryptographic hashes of email addresses.

Query the Sender-address:

Commonly referred to as the "Friendly From" address, it should be queried against the HBL, and the domain contained in it should be queried against the DBL & ZRD, with a forward DNS lookup against the SBL.

Finally, we're onto the inspection of the content (scroll up to think of all the opportunities you have had to reject email before it reaches this more expensive part of the email filtering process!).

Domains that are included in the email body

Website URLs or emails in the body content should all be queried against the DBL, ZRD, with a forward DNS lookup against the SBL. Emails contained within the email body can also be queried against the HBL.

Bitcoin addresses that are included in the email body

Query these against the HBL, which has a cryptowallet section containing hashes of bitcoin and other cryptocurrency addresses.

Email attachments

All attachments should be queried against the HBL, which lists malware files, in addition to email addresses and cryptocurrency addresses.

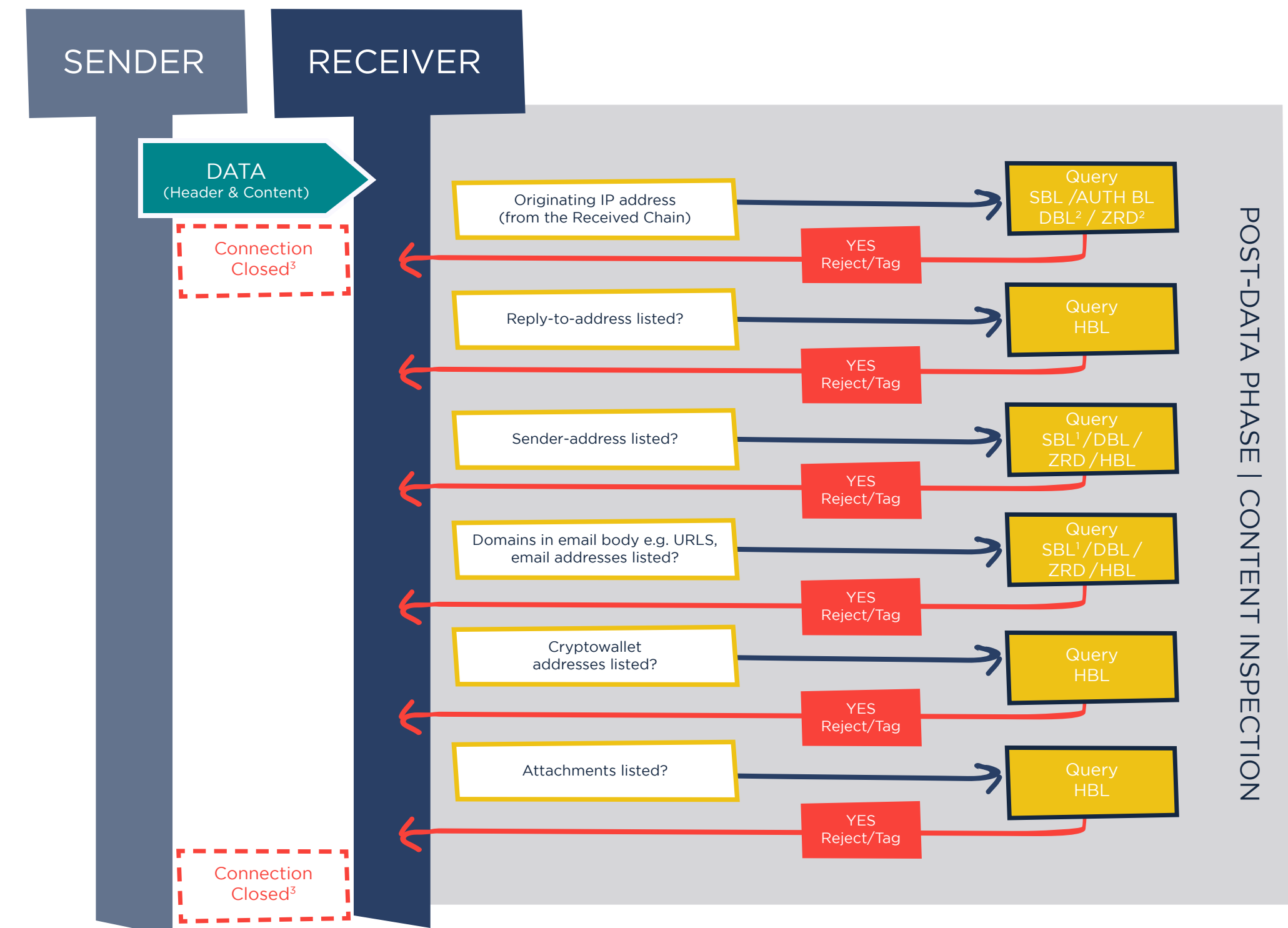
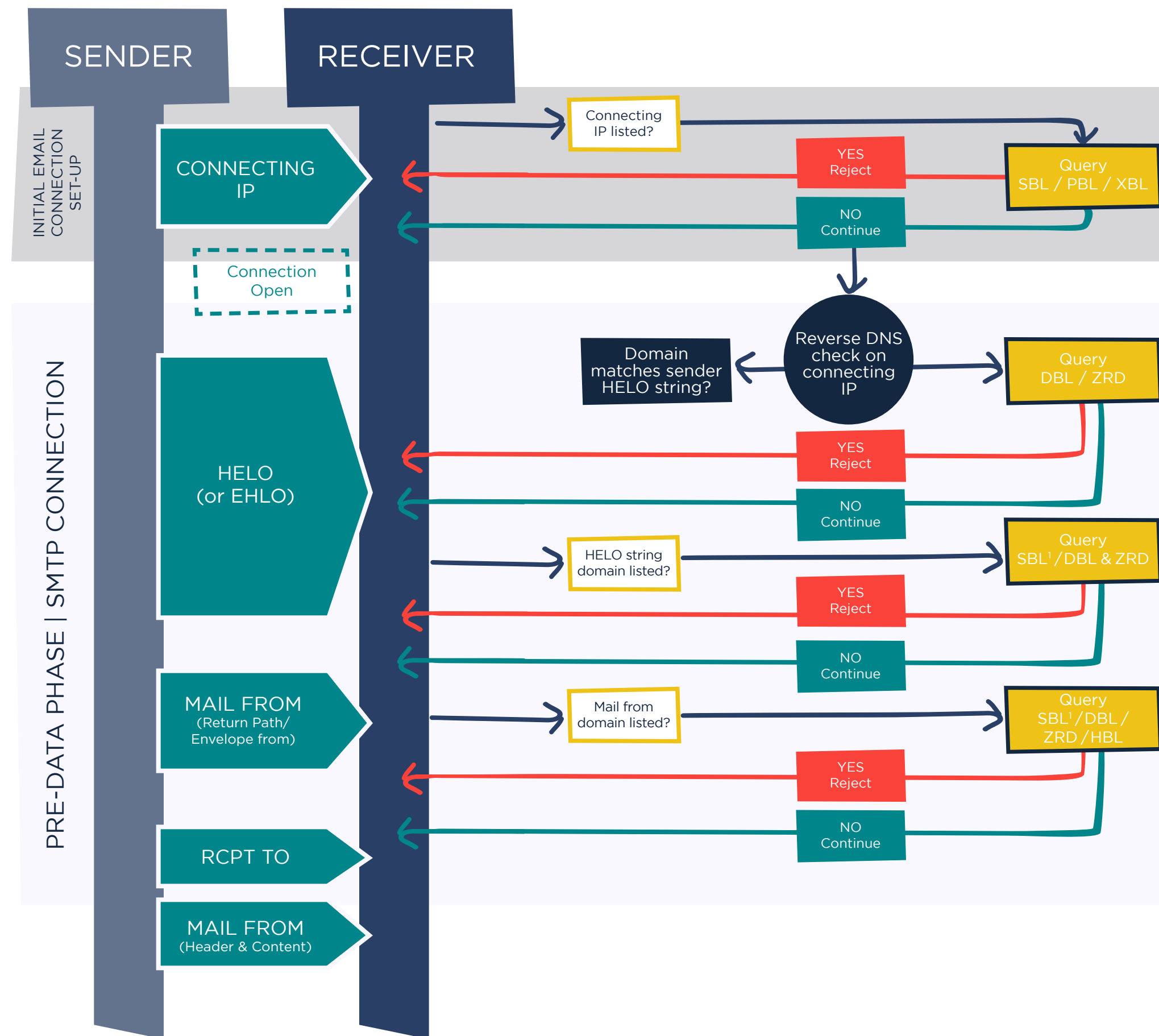
How to implement all of this?

Sadly, with so many variances it's not possible to cover all the specific operational details for different mail systems. Where some of our checks can't be implemented, we recommend reaching out to the support desk of the product you are using.

The good news is that we do have two plugins for opensource filters, [SpamAssassin](#) and [Rspamd](#), for both pre-data (SMTP) filtering and content analysis. Also here's the full outline of [what blocklists to apply where](#).

DNSBLs and email filtering - flow chart

Written by The Spamhaus Engineers




1. Forward DNS lookup of the domain required to check against SBL
2. Reverse DNS lookup required to check against DBL/ZRD
3. The SMTP connection can be closed either before or after content inspection - this is an implementation choice.



Click here to trial for free

Test out our DNSBLs, free for 30 days.

 [www.spamhaus.com/free-trial/
free-trial-for-data-query-service/](https://www.spamhaus.com/free-trial/free-trial-for-data-query-service/)