IP Addresses in Email Clients

Joshua Goodman[1]

Microsoft Research, Redmond, WA 98052

Abstract. IP addresses are an important tool for fighting spam, used for safe lists, blackhole lists, anti-spoofing and related purposes. While it is trivial to find the sender's IP address in most email server software, it turns out to be surprisingly difficult to do so in email client software: we explain why. This implies that either alternative approaches are needed (typically signature-based) or that new standards for communicating between clients and servers are needed. We suggest several forms such standards might take.

1 Introduction

IP addresses are a key part of many anti-spam efforts. As we discuss later, the sender's IP address is more-or-less the only part of an email message that cannot be faked. It is thus a key tool for most anti-spam efforts involving identity. IP addresses are a key way to identify bad senders, including blacklists like the MAPS RBL or violators of the rules of Habeas' Sender Warranted Program. They can also be used to identify known good senders, including Ironport's Bonded Sender program. Habeas' Sender Warranted Program has also recently been extended to identify good senders by IP address [2]. IP addresses are the key component of many anti-spoofing proposals, such as SPF (Sender-Permitted-From), Sender-ID [4], and other proposals.

The Received lines in email headers contain the (alleged) list of IP addresses that email has flowed through, as it is passed from one server to the next. As we will discuss below, many of these lines can be faked (all those added by servers external to the recipient's organization), while all of those added by trusted servers (those internal to the recipient's organization) can be believed. The key line is the first one internal to the recipient's organization, which gives the IP address of the outside machine that delivered the message across the internet to the recipient's organization. This line can be trusted. We will refer to this line as the "first internal line." Identifying this line in clients is the goal of this paper.

Note that our objective here is to identify the first internal line, not the "true" sender. So, for instance, in the case of mail that is automatically forwarded (e.g. through a .forward file), or sent through a mailing list server, our goal is to identify the server that forwarded, or the mailing list server, respectively, *not* the actual initial sender of the mail, which in general there is simply no way to reliably do (because spammers can always make it look like a machine is a mailing list server or forwarder even when it is not.)

In email *server* software, it is typically trivial to identify this first internal line. For instance, the server at the gateway which actually writes this line is often also the place where spam software runs, and thus trivially knows which line to use. In more complex environments, where internal servers need this information, email administrators can configure server software appropriately. For instance, the administrator could let a machine know that it is 3 hops from the gateway server, and thus should look 3 lines down to find the relevant line.

For email client software, end users cannot be assumed to know the configuration of their email servers. For instance, they do not know the precise number of steps from the gateway computer to the mail-store computer, so a line counting technique will not work. One can imagine a number of alternative algorithms. Later, we will describe reasons that each of these is problematic, especially for large organizations, and especially in the face of spammer attacks. These approaches include looking for the highest line containing an IP address with an MX record for the recipient's organization; manually configuring client software with the number of hops; or using HELO information to find the last machine internal to the recipient's organization. We will show that all of these have problems for large, complex email systems, where a single set of servers may host multiple domains, complex load-balancing routers may confuse IP addresses, and where many computers, especially those that are internal only, may not have DNS entries. In addition, algorithms that work today may be vulnerable to spammer attacks. While these attacks do not occur in spam today, we can assume that spammers will use them if a given vulnerable algorithm were widely deployed.

In the rest of this paper, we will first give a quick review of received headers. Next, we describe in more detail each algorithm that does not work, and then give an example where the algorithm fails. Then, we will describe an algorithm that works, in the sense that it knows when it does not know: it either returns the correct IP address, or returns nothing. Unfortunately, there are many cases in which returning nothing is the best it can do. We will describe potential standards changes that could remedy this problem, as well as various alternative approaches, including abandoning IP addresses and using signing-based approaches. We conclude that something must be done: we must either create a new standard or abandon the idea of using IP addresses in client software.

2 Review of Received Headers

In this section, we quickly review Received headers. See RFCs 2821 and 2822 for details.

Email is normally passed on the internet from an originating client to a server to another server to yet another server and so on, until it is eventually stored in a mailbox, where it can be accessed from the recipient's client. SMTP specifies that each server should prepend (place at the beginning) a "Received" line. An example line might look like the following:

```
Received: from mlb.com ([172.20.120.74]) by
    mail02.lists.mlb.com (Postfix) with ESMTP id 8D944F8244 for
    <exampleusername@hotmail.com>; Fri, 12 Apr 2002 20:01:51 -0400 (EDT)
```

This example line says that the machine mail02.lists.mlb.com received the message from the machine mlb.com. Note that mlb.com is not necessarily the machine that really communicated this mail – this portion comes from the HELO SMTP command and a dishonest sender may insert any information at all here. The sender's IP address was 172.20.120.74, and that information cannot be faked without extreme effort. By tracing through the list of received from lines, a human can find the path that mail has actually followed.

The typical path for mail to follow is from an email client through one or more servers internal to the sending organization, and then, across the internet, to the recipient's first external server. This external server typically relays the mail across the firewall to an internal server, where it then passes through one or more other internal servers (e.g. virus checking) before ending up in a mailbox. As the message is passed from one server to the next, received lines are prepended.

It is sometimes said that it is easy to fake the source of mail. Indeed, it is easy for a mail sender to add any received headers he wants, and to delete or modify received headers lower in the list (earlier in the path.) However, assuming a recipient can trust his own servers, all received headers created by machines in the recipient's organization can be trusted. Most of these are of no interest from the point of view of fighting spam: the fact that my virus checking machine received mail from my gateway machine is boring. The only header that both can be trusted and is interesting is the one added by this first internal server. That header contains the IP address of the last external server in the sender's organization. If that last external server is known to be bad, e.g. is a known open proxy or a known sender of spam, I probably want to reject the mail. If that last external server is known to be good, e.g. is on Bonded Sender's list of good senders, I probably want to accept the mail. This is also the IP address used by anti-spoofing standards such as Sender-ID. Thus, finding this first internal line is key.

Note that there is some confusion about the difficulty of faking an IP address over TCP/IP. TCP packets contain a 32 bit sequence number communicated from the recipient to the sender, which the sender must then use. In order to fake an IP address, the sender must somehow guess this sequence number (one in 4 billion.) Many older systems used poor methods of randomizing this sequence number, and spoofing was possible. Modern systems correctly randomize the sequence number, making this attack extremely difficult. There are also other attacks on IP addresses, such as injecting incorrect routing information into the internet, so that the sequence number information from the recipient arrives at the forger's machine instead of at the apparent sender's machine. This attack has been successfully used for otherwise unused IP addresses but we are not aware of its use for IP addresses that are in use. A spammer sitting between the alleged sender and the recipient can also eavesdrop/intercept the relevant sequence information. This means that an ISP or backbone provider can easily forge data appearing to come from any of their customers. Fortunately, we are not aware of spammers using this attack, which seems difficult to carry out.

3 Algorithms that do not work

In this section, we describe a number of algorithms for finding the first internal line in email clients. We then explain why each of these algorithms, unfortunately, does not work all of the time, and especially not in the face of spammer attacks.

3.1 Find the MX Record

The most obvious algorithm is to look through the list of IP addresses until one is found that matches the recipient's domain's MX record. The MX record contains the list of inbound mail servers for a given domain. That is, imagine I have an account at example.com and example.com's MX record lists three hosts, a.example.com, b.example.com and c.example.com, with IP adresses 1.2.3.4, 1.2.3.5 and 1.2.3.6 respectively. If I have a list of received from lines such as

```
Received: from y.example.com ([1.2.3.10]) by mailbox.example.com
Received: from x.example.com ([1.2.3.9]) by y.example.com
Received: from a.example.com ([1.2.3.4]) by x.example.com
Received: from spammer.sender.com ([90.91.92.93]) by a.example.com
Received: from faked.microsoft.com ([80.81.82.83) by spammer.sender.com
```

then I can simply scan through this list until I find, in this case,

```
Received: from a.example.com ([1.2.3.4]) by x.example.com
```

I assume that all lines before the gateway (including this one) are internal and can be trusted, and that this line, containing a machine listed in the MX records, must be the one in which my gateway machine passed the message to an internal machine. This means that the following line

```
Received: from spammer.sender.com ([90.91.92.93]) by a.example.com
```

must have come from the true last external machine of the sender, and that 90.91.92.93 is the sender's IP address.

Unfortunately, this algorithm fails in some important cases. Large recipients of email often place a load balancing router in front of their gateway (first internal) machines. The load balancing router has a different IP address than the actual gateway machines. For instance, imagine that example.com switches to this strategy to balance their load across their machines a, b, and c. They create a new pseudo machine mail.example.com, with IP address 1.2.3.20. This IP address goes to a load balancing router, which routes the traffic round robin to a, b, or c. Load balancing routers are not servers, so there is no line for mail.example.com in the received list. a.example.com does not know or care about the routers that mail passes through, and so inserts

```
Received: from spammer.sender.com ([90.91.92.93]) by a.example.com
```

just as before. However, when we now try our "MX record" algorithm, it fails. The only MX record now is for mail.example.com (1.2.3.20) which does not appear anywhere in the received from list.

This failure may not sound so bad – at least we know that we have failed. Unfortunately, an evil spammer can now manipulate us. He inserts received from lines so that the final mail looks like this:

```
Received: from y.example.com ([1.2.3.10]) by mailbox.example.com
Received: from x.example.com ([1.2.3.9]) by y.example.com
Received: from a.example.com ([1.2.3.4]) by x.example.com
Received: from spammer.sender.com ([90.91.92.93]) by a.example.com
Received: from mail.example.com ([1.2.3.20]) by spammer.sender.com
Received: from faked.microsoft.com ([80.81.82.83) by mail.example.com
```

(The last two lines are fake lines inserted by the spammer.) Now when we use our MX record algorithm, we find the MX record line, and looking at the line after it, we think that this mail originated at Microsoft. Not only has our MX record algorithm failed, it has made it easy for spammers to trick us.

3.2 Manually configure clients

One option is to actually give clients all the information they need. For instance, we could tell them that our servers at example.com are 4 deep, and that they should always look 4 lines down to find the true sender. Or we could tell them the true list of gateway machines (a.example.com, b.example.com, c.example.com) rather than having them rely on the MX record. Or we could tell them that the range of IP addresses (1.2.3.*) corresponds to example.com's domain. These are fine solutions for email server software, run by technically savvy administrators, but relying on end users to configure complex information like this in clients is unworkable for any mass market software. Also, server information can easily change, and there needs to be a way to easily update the client software.

This idea is not completely hopeless: standards changes could allow this information to be communicated to clients automatically, and later, in the section on standards changes, we will consider related ideas.

3.3 Use HELO information

For a human looking at the preceding lines, it is very obvious that spammer.sender.com is not internal to example.com. We could use an alternate algorithm of going through the received lines until the first HELO that does not match the recipient's organization, in this case spammer.sender.com. Unfortunately, this fails for multiple reasons.

The first reason this fails is that spammers can easily send whatever information they want in the HELO data. In our example, our spammer's machine could have said "HELO internal.example.com." One can imagine performing forward and/or reverse IP lookups on the various names found in the headers to verify that nothing is forged. Unfortunately, these may detect forgeries or problems where none exist. For instance, we have seen at Microsoft that machines like virus scanners that are not accessible externally are not registered with all DNS servers.

In addition, and perhaps more seriously, the names of machines used in an organization do not necessarily match the recipient's domain name. In our example, all internal domain names were in the example.com domain. Unfortunately, in many large installations, the same machines are used to host multiple domains. For instance, microsoft.com machines handle all email for xbox.com.

4 An Algorithm That Knows When it Works

In this section, we describe a conservative algorithm for finding the IP address that when it works, knows that it worked, and when it didn't work, knows that it didn't. Unfortunately, in many interesting cases (many of the examples above) the algorithm simply concludes that it cannot tell which line is the correct one.

The basic idea is that in several cases, we can be sure that we are still internal to the recipient's domain. We keep searching down (backwards in time) until we find a machine in the MX record, immediately followed by an address that appears to be external to the recipient's domain.

There are several cases where we can be sure we are in the recipient's domain: if the IP address belongs to a machine listed in the MX record for the domain, we can be sure. If the IP address belongs to a range used for dynamic IP address assignment, and we are sure that we are still in the domain, then we can be sure. If the domain name in the HELO is internal to the domain, and a DNS lookup verifies that the HELO matches the IP address, we can be sure.

We can keep looking through the list until we reach a machine listed in the MX record. If the machine listed in the MX record received from an external sender, we can be very confident that we have found the sender.

```
bool fFoundMXRecord = FALSE;
for each received from line of the form Received from a.b.c [i.j.k.l] {
    if i.j.k.l in MX records of receiver domain {
        fFoundMXRecord = TRUE;
        continue;
    }
    if i.j.k.l is of form
        10.x.y.z or
        172.16.y.z to 172.31.y.z or
        192.168.0.z to 192.168.255.z
    {
        continue;         # Must be internal
    }
    if DNS lookup of a.b.c yields i.j.k.l and b.c is receiver domain {
        continue; # Must be internal
    }
    if fFoundMXRecord {
        Output sender's alleged domain a.b.c and sender's
        actual IP address i.j.k.l
    } else {
        Error: unable to identify sender's IP address
    }
}
If we reach here, then Error: unable to identify sender's IP address
```

Note that this algorithm is not 100% foolproof. Consider a bizarre configuration for example.com's server. Imagine that after being received by mx1.example.com, the mail follows the following path:

```
mx1.example.com
confusing.com
mx2.example.com
mailbox.example.com
```

A machine called mx1.example.com is listed in the MX records for example.com. mx1.example.com passes the mail to confusing.com – a machine in a different domain. This machine in a different domain passes the mail back to mx2.example.com – a machine that is also listed in the MX records for example.com which only then passes the mail to mailbox.example.com. It's hard to picture such a configuration existing.[1] Note that the following configuration does not confuse this algorithm:

```
mx1.example.com
mx2.example.com
mailbox.example.com
```

The following scenario (which is an odd unlikely one) will confuse the algorithm if internal.example.com does not have a DNS record, or has a misconfigured HELO.

```
mx1.example.com
internal.example.com
mx2.example.com
mailbox.example.com
```

Thus we believe that this algorithm is safe to use in practice.

5 New Standards

Unfortunately, in too many interesting cases, this algorithm fails to find an answer. In particular, it fails to find an answer when the gateway machine sits behind a load balancing router, as it often does at large organizations. It also fails when the same organization hosts multiple domains. While this algorithm fails safely, "knowing that it doesn t know", that is insufficient.

We thus believe that some sort of new standard is needed for communicating between servers and clients. In this section, we review several possibilities, rejecting some, but at this time not endorsing any, pending further investigation.

[1] One could imagine this particular sequence in a forwarding scenario, such as a mailing list at example.com that sends to a user at confusing who forwards to a user at example. But as mentioned in the introduction, in the case of mailing lists or forwards, we consider the sender to be the forwarder or mailing list.

One naive approach is having servers simply add a header, but spammers could add the header as well. While new servers would strip out this header, old servers would not, and new clients matched with old servers would look for the headers and be deceived by spammers.

Another approach is to have any IP address related work done in the servers, which then inserts the results of its work for the client. For instance, the server could check if the sender is on the Bonded Sender list, and add a header that says "BondedSender: TRUE" or "BondedSender: FALSE" This does not solve the problem, but only shifts it, since we still must solve the problem of how to communicate this information reliably from server to client; there is nothing to prevent spammers from inserting these headers, again confusing new clients used with old servers.

A variation on this approach is to let servers not just use the additional information, but also take any needed actions. For instance, using a blacklist, the server can delete the email. Unfortunately, this approach also has problems: if client and server need to cooperate, they cannot. For instance, many per-user safelists exist on clients only, so a server using a blacklist like the RBL would not be able to make exceptions for an individual user. This also forces people to upgrade servers for any new functionality, eliminating the ability of individuals to install new functionality in their clients.

Another approach, suggested by Bob Atkinson, involves adding information to the recipient's DNS entries. Various information could be added here. In our favorite variation, just as approaches like Sender Permitted From, Sender-ID and RMX have proposed listing the valid outbound IP addresses for senders, we could add yet another DNS entry that listed the gateway servers for a domain that were visible internally (e.g. after load balancing routers, etc.) Unlike other uses of DNS entries, these are meant to be used *internally.* (Atkinson suggests a slightly different approach than this, but we prefer this variation.)

Yet another approach is for the last recipient server to modify the top received from line, adding, for instance, a specially formatted comment to that line that contains the appropriate IP address. (RFC 2821 allows comments in received from lines.) Only an internal server is capable of adding a comment in the top (most recent) received from line, so, if present, such a comment can always be trusted.

These two approaches – more DNS entries and modified received lines – should be carefully evaluated in real systems. Do most email clients have access to DNS servers, or do they sit behind corporate firewalls that blocks such access? Will adding comments cause any unforseen problems? If the DNS approach works, it seems preferable to us, since it does not require any server modifications.

6 Alternative approaches

This paper has focused on finding the relevant IP address as a general problem. However, it is not always clear that this is the right answer at all. For *negative* use of IP addresses, such as blacklists, there are other options. In this section, we examine a "shot gun" approach for negative information, in which we examine all or nearly all IP addresses. For positive information, we briefly consider signature based approaches as an alternative to IP-based approaches.

This paper has focused on finding the relevant IP address as a general problem. For "positive" uses of IP addresses, such as Bonded Sender or Sender-ID, this is important. For negative uses of IP addresses, such as blacklists, a different technique is possible: one can scan the headers for any bad IP addresses. If any address in the headers is bad, then the sender is assumed bad. One can first remove any addresses that are obviously internal, e.g. those that are obviously in the same domain as the recipient. This approach is relatively robust and has only one problem: if the recipient's domain is listed as bad, the recipient's mail is always penalized. For instance, if the recipient's mail servers are listed on a blacklist, the recipient might have all of his inbound mail deleted. While morally, one might or might not think it fine to delete all mail sent *to* a domain that spam is sent *from*, this is not fiscally wise: such an action is sure to result in expensive support calls or worse.

One alternative to IP address based schemes for positive information is signature-based approaches. Instead of basing identity on the IP address, we force senders to crytpographically sign their mail in some way. This can be used for anti-spoofing proposals as suggested for, e.g. Yahoo's DomainKeys proposal, as well as for safe sender lists, e.g. ePrivacy group's Trusted Sender program. These signing-based approaches have the advantage that they work equally well in both clients and servers. It is important to realize, however, that while signature-based approaches avoid one problem (the client IP problem) they introduce many other problems. The most important is that any senders wishing to adopt them must modify their software. IP-based approaches do not require any modifications by senders, speeding broad adoption. Signature-based proposals have other problems as well. For instance, some servers may make surprisingly large changes to the body of email messages, breaking some overly simplistic signing approaches. The goal of this section is not to recommend either IP address approaches or signing approaches, but simply to make it clear that in both cases there are difficult issues that must be solved.

7 Conclusion

Most people find it very surprising that it is so difficult to find the right received line in an email client. IP address features have long been used in servers (particularly blacklists) and any human being examining headers can determine the right line. Thus, both industrial practice and personal experience give the impression that this is an easy task. Also, because there have been no widespread, automated, client-side uses of IP information, spammers have not yet had much incentive to defeat some of the faulty algorithms described in this paper, and thus the attacks are not well known: they must be predicted, rather than being observed.

IP address information is becoming increasingly important. IP addresses are already widely used in blacklists, but these are primarily server or router-based. Aside from blacklists, IP addresses are not yet widely used. However, it looks like at least some form of anti-spoofing proposal will soon gain acceptance and safe sender lists (like Bonded Sender) are also growing in popularity. As the importance of IP addresses increases, especially IP addresses used for positive information, the importance of making them available in clients also increases.

We have proposed two methods for solving this problem. Our favored proposal (due to Bob Atkinson) is to expose the relevant IP addresses in DNS entries, in a manner similar to Sender-ID. Alternatives, such as modifying the top received header, also merit investigation. And of course, this being a new problem, it is quite possible there are other, even better solutions. In any event, some new standard must be adopted if IP addresses are to be useful in clients.

References

1. Habeas. Sender warranted program. See `http://www.www.habeas.com`.
2. Habeas. Habeas releases patches for SpamAssassin, April 6 2004. See `http://www.www.habeas.com/configurationPages/spamassassin.htm`.
3. Ironport. Bonded sender, 2004. See `http://www.bondedsender.com`.
4. Jim Lyon and Meng Weng Wong. MTA authentication records in DNS, 2004. Available from `http://www.microsoft.com/mscorp/twc/privacy/spam%5Fsenderid.mspx`.
5. MAPS. Mail abuse prevention system realtime blackhole list, 2003. See `http://mail-abuse.org/rbl/`.
6. SPF. Sender Permitted From. See `http://spf.pobox.com`.