SMTP Information gathering

Lluis Mora, Neutralbit
llmora@neutralbit.com

Abstract

The SMTP protocol, used in the transport and delivery of e-mail messages, includes control headers along with the body of messages which, as opposed to other protocols, are not stripped after the message is delivered, leaving a detailed record of e-mail transactions in the recipient mailbox.

Detailed analysis of SMTP headers can be used to map the networks traversed by messages, including information on the messaging software of clients and gateways. Furthermore, analysis of messages over time can reveal organization patching policies and trends in user location and movements – making headers a very valuable resource during the target selection phase of targeted attacks.

Keywords

Information gathering, targeted attack, SMTP security

SMTP INFORMATION GATHERING

Introduction

SMTP headers record extensive information about the sender and the path a message takes through the network; some of these headers are actually used in the communication but others are just intended to ease troubleshooting when problems arise and are superfluous otherwise, leaking precious information about the sender and the communication - it is like having debug information built into the protocol.

The SMTP header analysis techniques presented here can be used to gather information on a user or group of users, and are specifically useful in target selection (vulnerable software and patching patterns) and tracking user location and movements.

Control information

What makes SMTP messages extremely interesting for information gathering purposes is the fact that control information is embedded as part of the communication; each message includes the content (the data being transferred) and the control information (envelope and headers) needed to get it from sender to destination, along with information on each of the hops it traverses as it travels through the network.

Some control information is essential and an integral part of the communication: the sender and recipient information, the subject and the date the message was sent are all stored as message headers. Other information, such as the “Received” headers, are useful to the MTA but are no longer used once the message is delivered, while other headers (such as the ones identifying the e-mail application) are not used at all.

The mandatory presence of some of these headers usually means headers are not stripped on delivery, and they end-up being stored in the user mailbox along with the communication content.

SMTP messages are usually only fully available to the recipient, to analyse communications an attacker needs access to the recipient storage. Luckily enough the concept of "public recipients" (a.k.a. mailing lists) is widespread; they become an easy way to access SMTP messages and control information that tell a tale of each sender. Mailing lists often keep an archive in *mbox* format, a storage of data and control information ready to be analyzed.

The following sections will study how some of this control information can be used to draw conclusions on the senders and their organizations.

Network topology mapping

Each time an SMTP message is relayed through the network, the recipient gateway records the transaction by adding a "Received" header to the incoming message. This header, whose prepending is mandatory as per RFC2821 (section 3.8.2), is designed to avoid mail loops and to facilitate troubleshooting.

A received header looks like this:

```
Received: from relay.example.com (201.20.51.192)
        by neutralbit.com (Postfix) with ESMTP id 35B83500EC
        for <llmora@neutralbit.com>; Mon, 15 May 2006 20:26:52 +0000 (UTC)
```

As each gateway adds its own details, when the message reaches the final recipient it includes a complete trace of the path it has taken through the SMTP network. Each "Received" header includes, at least, the following information:

- IP address of sending gateway
- FQDN of receiving gateway
- Transfer protocol: most of the times SMTP, although sometimes it is HTTP (webmail), POP3 (fetchmail) or even ASMTP (Authenticated SMTP)
- MTA server software

- Timestamp when the message was relayed, including time zone

A “Received” header includes information on the current hop, including the IP address that connected to do the delivery. By matching this IP address with the hop FQDN from the previous “Received” header an attacker can get a good understanding of the path the message followed.

Obviously this is no *traceroute* as only the systems that participate in SMTP transactions (e.g. the sender and any mail gateways the message goes through) will show up in trace headers. The advantage in front of an IP level *traceroute* is that routing information is added to the message and forwarded to the next hop; as such internal SMTP gateway addresses are usually found in "Received" headers, allowing the mapping of network topology behind NAT gateways.

In a way “Received” headers act like the “Record Route” IP option, with the difference that SMTP headers are not limited in the number of hops they can record and, most importantly, they are rarely disabled.

From the SMTP path defined by received headers an observer can extract interesting information, such as:

- Corporate IP subnetting: internal IP addressing scheme, static or dynamic connection to the Internet on corporate locations, etc.
- Corporate SMTP gateway policies: whether various corporate locations use the same gateway or one gateway per location and identify those "border" SMTP servers
- Server location: timestamps include the time zone which can help in narrowing down gateway location
- Server software and versions deployed in the organization
- Whether relay links are being encrypted, and what encryption is being used
- Whether mail tracing (e.g. ReadNotify[1]) is being used, suddenly e-mail from a contact goes through totally different servers

The information gathered can then be used by an attacker willing to compromise e-mail communications to do target selection with a greater success rate:

1. Plot a SMTP connectivity graph
2. Find out which nodes are common to most of the traffic
3. Weight down which server software and releases are easier to compromise / sniff

An attacker could construct a graphical representation of the information to aid in target identification; this is an example of an SMTP path graph obtained using Graphviz[2]:

[1] Readnotify, http://www.readnotify.com/
[2] Graphviz, http://www.graphviz.org/

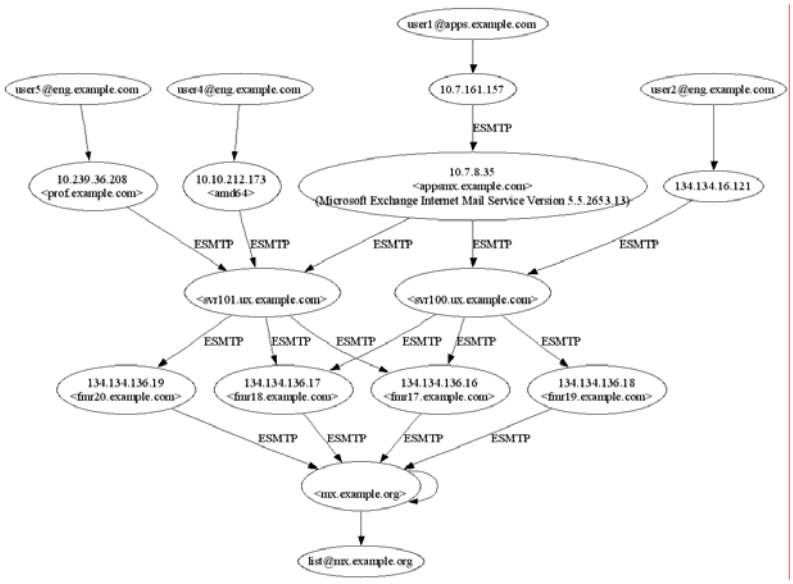

Figure 1. Trace of a SMTP path

Exploiting SMTP topology knowledge

Although SMTP paths are not required to be symmetric, in practice they usually are. That means that the path a message takes through the internal network to reach a recipient is usually followed in reverse order when the recipient replies.

An attacker willing to compromise a SMTP relay needs only to identify a SMTP path that traverses the gateway and send a message to a user behind it - with a bit of luck the path will be symmetrical and the message will go through the server.

Needless to say, this only works when the vulnerability exists in the handling of messages, for instance in the previous SMTP path scenario, an attacker will probably identify *svr100*, *svr101* and *appsmx* as servers that handle most of the messages, weight down the server software they are running and decide on the one using "Exchange 5.5 SP4" (v5.5.2653.13) which is vulnerable to the TNEF decoding vulnerability. The attacker will send a message crafted with the exploit to *user1@apps.example.com* which will hopefully lead the message through the vulnerable server - exploiting it along the way.

Sender mailer fingerprinting

As has been previously discussed, server software can be extracted from the “Received” header added by relays; a different set of headers can be used to determine the mail client used by the sender of the message.

Most e-mail applications add an “X-Mailer” or equivalent header describing the application and version used to send the message. These headers (which have no practical use) tend to provide extensive detail on the software used to create the message and its version, down to the build number. A few examples:

```
X-Mailer: Microsoft Office Outlook, Build 11.0.5510
User-Agent: Thunderbird 1.5.0.7 (Windows/20060909)
X-Mailer: ColdFusion MX Application Server
X-MimeOLE: Produced By Microsoft MimeOLE V6.00.2900.2962
X-Mailer: Evolution 2.2.3 (2.2.3-4.fc4)
X-Mailer: iPlanet Messenger Express 5.2 Patch 2 (built Jul 14 2004)
X-Mailer: Lotus Notes Release 5.0.6a  January 17, 2001
User-Agent: SquirrelMail/1.4.3a
```

This level of detail offers an attacker an excellent description of the messaging application the user is using; the only task left is to find an appropriate vulnerability and exploit it by sending a message to the victim.

Client application usage

For attacks targeting a specific organization a stealth approach is critical - the attacker needs to increase as much as possible the chances of success with as little messages as possible; target selection is in these cases as important as it is the use of an undisclosed vulnerability.

When the attacker has access to a mailbox spanning a wide period of time, he can match sender software information extracted from the previously discussed headers with the date the message was sent, effectively tracking user updates to the messaging application. For instance, take into account the following headers found in various e-mail messages from the same sender, sorted chronologically:

```
Date: Fri, 28 Apr 2006 11:42:04 +0200
User-Agent: Thunderbird 1.4 (Windows/20050908)

Date: Wed, 22 Jul 2006 17:30:21 +0200
User-Agent: Thunderbird 1.5.0.2 (Windows/20060308)

Date: Wed, 13 Sep 2006 13:43:13 +0200
User-Agent: Thunderbird 1.5.0.5 (Windows/20060719)

Date: Fri, 29 Sep 2006 16:34:56 +0200
User-Agent: Thunderbird 1.5.0.7 (Windows/20060909)
```

What information can be extracted from these headers? The user has certainly updated its mail client over time, but what has been the gap between product release and installation? Let's add release information to the mix:

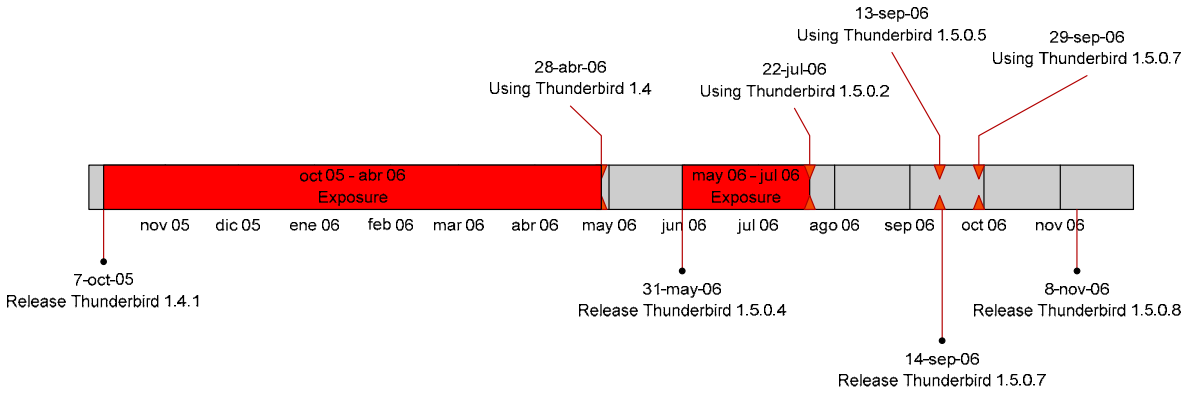

Figure 2. User mailer usage against release dates

As can be noted the user has not really been keeping up to date with releases, if these releases were due to published vulnerabilities the gaps become really scary. An attacker had over 6 months to come up with an exploit and perform an attack, certainly not very 0-day. The data shows that more recently the gap between update release and installation has been reduced significatively - maybe the user has been made aware of the importance of patching or (more likely) the organization has deployed a patching policy.

With enough messages from users in an organization, it is easy to establish which patching policy (if any) the organization has deployed, take for instance the following chart using mailer version distribution amongst a group of users of the same organization, extracted from contributions to a public mailing list:

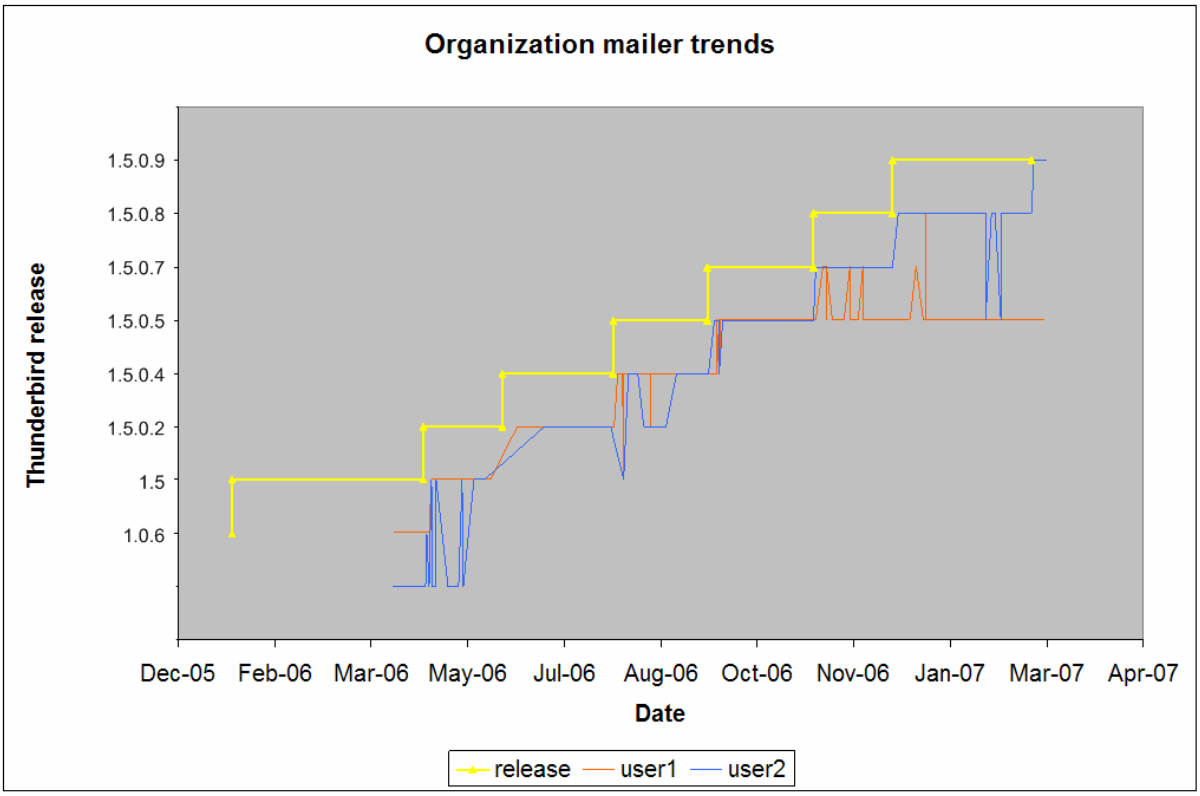

Figure 3. Organization mailer usage against release dates

Patching policies can be established by analysing mailer evolution over time; an attacker can then easily get answers to questions such as:

- Is there a defined patching policy in the organization?
- Are there any policies on the usage of particular mailer software?
- What is the mean time between update releases and update installation
- Are there any exceptions to the policy?
- Who is the easier prey?

Definitively a lot of information that is being given away for no reason: these headers are never used during message transport and delivery.

Using the same analysis technique server software patching policies can be easily gathered by plotting the software versions extracted from “Received” headers against release dates.

Mail usage trends

Changes over time are interesting not just to establish software usage and patching policies but also to discover user behaviour changes that might be significative for a targeted attack. A few of these scenarios are described in the following paragraphs.

The observation of a user sending e-mails with alternating mailers (different software or versions) might indicate that the user is using that e-mail address from multiple locations, such as using a mail client from home and another one from work. If the attacker is after compromising the corporation that user belongs to, maybe the user has a VPN to the office when working from home and the attacker should focus on attacking the less protected home network?

On the other hand, a user sending e-mails with the same mailer but from alternating locations might indicate that the user is using the same computer from different locations – maybe he takes the laptop home? Protection there may be more relaxed than at the office making it an easier target for compromise.

E-mails with different domains originating from the same location often mean that the user has several e-mail accounts configured and that he is receiving and replying to non-corporate messages while at work (or the other way around). In these cases even though the corporate infrastructure might have filters in place to prevent attacks perhaps by sending the message to the personal address it ends up in the same user desktop, avoiding any message filters.

Timestamps as used in SMTP always include a time zone indication; messages from the same sender but with different time zones might point to a user which is usually on the go, perhaps connecting from not-so-secure places.

Extracting information from other headers

We have covered two different classes of headers that are of little practical use to the SMTP protocol and that (with some care) can be easily stripped off outgoing messages. However, even if they are stripped the information they provide might be indirectly derived from other headers which are not so easy to remove. This section covers some of those a priori uninteresting headers that, sometimes, can provide an attacker with valuable information.

User agent headers are not the only way to identify an application, much in the same way as "Server" headers in HTTP are not the most reliable way to identify a server. Differences in the way they implement standards, ordering of the headers, how replies are quoted, information embedded in the message content, etc. This and other information can be used to fingerprint the e-mail application even after the User-Agent header has been stripped.

The "Message-ID" header is created by the sending application and is only required to be unique to the host that generates it, so each implementation decides what its format will be. Most implementations include the host address that generated the message, although this is not a requirement. Other implementations decide to add some other meaningful information that an attacker certainly appreciates, these are some illustrative examples:

- Application, platform, version, host, sending PID and timestamp:

```
Message-ID: <Pine.LNX.4.21.0611280421440.26304-100000@example.org>
```

- Application, client IP, host and timestamp:

```
Message-ID: <1103.203.41.53.196.1128283359.squirrel@mail.example.com>
```

- Application, host, sending user and timestamp:

```
Message-ID: <11363603.1154544476739.JavaMail.root@appserver.example.net>
```

The varied formats of Message-ID make it a very interesting header for e-mail client application fingerprinting; other interesting pieces of information include:

- Dates and timestamps in SMTP messages (appearing in various headers) can be used to determine if machines within an organization are somehow synchronized.
- MIME multipart boundaries might include mailer information

```
Content-Type: multipart/mixed; boundary=Apple-Mail-1--944594902
```

- Even if "Received" headers are removed, some mailers encode the client address on a variety of headers:

```
X-Originating-IP: [195.16.101.21]
```

```
X-Client-IP: 192.166.19.63
```

- Antivirus / Antispam information can point to additional attack vectors on the network:

```
X-StarScan-Version: 5.5.10.7; banners=-,-,-
X-Spam-Checker-Version: SpamAssassin 3.1.7 on fw.example.com
```

TNEF: An unsuspecting source of information

Although not part of the SMTP specification, TNEF (Transport Neutral Encapsulation Format) is a clear example of the kind of leaks that can happen even when control information is being scrubbed.

If you use Outlook or Exchange you have probably used TNEF, even if you are not aware of it. TNEF is used by default when you want to resend a message by drag-and-dropping it in a new message or if you use the "Rich Text" format in e-mails, etc.

A mail message containing TNEF information is represented as a MIME multipart message including a body with a content type "application/ms-tnef", where the actual TNEF encoded information is stored. This non-standard extension manifests itself in non-Microsoft e-mail clients by looking like an attachment with the name of "winmail.dat".

TNEF is a proprietary encapsulation format, although there are some tools out there to decode the "winmail.dat" attachment. Amongst the information that we usually find inside the encoding are the complete path to the user PST file and his Windows logon username, see for instance some of the strings that can be extracted from one such TNEF attachment:

```
$ strings winmail.dat
IPM.Microsoft Mail.Note
PCDFEB09
C:\Documents and Settings\llmora\Configuración local\Datos de
programa\Microsoft\Outlook\Outlook.pst
```

CONCLUSIONS

"Loose lips sink ships": strip unneeded information whenever possible - there is already enough information being leaked in ways that can not be prevented such as implementation differences, information encoded in unsuspecting places, leaks in the message content, etc.

Although it is not a complete solution –some information can still be indirectly gathered- it drastically reduces exposure. Of course limiting exposure will not get rid of the messages already out there; studying what information has leaked and what are the weakest points that can be gathered from its analysis will point us in the direction of what needs to be fixed in the first place.

A final warning note: most of the discussed analysis techniques work on the delivered message, relying on information provided by the client which is easily faked (such as adding extra hops, or modifying user agents), so take collected data with caution.

REFERENCES

- Klensin, J., Editor, "Simple Mail Transfer Protocol", RFC 2821, March 2001.

- Heasman J., Litchfield M., "Vulnerability affecting Microsoft Exchange", NGSSoftware, January 2006.

- Resnick, P., Ed., "Internet Message Format", RFC 2822, April 2001.