Towards a Tor-safe Mozilla Thunderbird

Reducing Application-Level Privacy Leaks in Thunderbird

3072D/234E33B6

July 2011

Abstract

The potential privacy issues involved when using mail clients with Tor are not well understood. This work contains a privacy analysis of the popular mail client Mozilla Thunderbird in a setting with Tor. We focus on finding the answer to the following questions: What are the technical privacy risks when using Mozilla Thunderbird with Tor and how can these risks be reduced? To answer these questions we analyze relevant protocol and mail header specifications and their usage in Thunderbird. We investigate if the few currently available Thunderbird configuration steps for anonymizer are enough to prevent privacy leaks and deanonymization. Based on our findings, we propose a set of configuration parameters and modifications to Thunderbird that a potential "Tor Mail Bundle" may include.

Contents

1 Introduction

With our work we aim to prepare the creation of a privacy-enhanced version of the email client Mozilla Thunderbird to give email users more privacy when sending and receiving mails. We analyze Mozilla Thunderbird in a setting with Tor [54], to locate privacy issues that should not be present in a privacy-enhanced version of Thunderbird. The privacy-enhancement might be an easily installable extension or a change to Thunderbird.

Section 1 shortly describes related work, our motivation, use cases and functional requirements. Section 2 discusses our threat model and its players. Section 3 contains our analysis and provides easy solutions to minor problems. Section 4 depicts a few attack possibilities and discusses our proposed defense to more complex issues. In section 5 we summarize our results and section 6 lists a few open questions.

1.1 Goals and Non-Goals

Preventing anonymity set reductions and deanonymization is our goal. In particular the following leaks:

- "leaking requests"
- IP address disclosure at application-level
- timezone disclosure
- system language disclosure
- OS disclosure
- Thunderbird version disclosure
- protocol features that leak privacy relevant information
- active proxy circumvention attacks
- general fingerprinting possibilities

The result should contain a list of possible privacy problems which occur during the use of Thunderbird with Tor and it should also be shown how to defend against these problems, which in turn would be useful for someone actually implementing a "Tor Mail Bundle".

Non-goals:

- hide the fact that we are using Thunderbird
- hide the fact that we are using Tor
- defeat author identification based on linguistic analyzes
- detect identifying information in the mail body or attachments
- perform an analysis at source code level (Thunderbird's source code was consulted to answer a few specific questions only)
- conceal the addressees of mails from the mail provider(s)

1.2 Tor

Tor primarily provides network-level client anonymity. Tor's design clearly separates the network-level from the application-level. This allows Tor to provide a generic interface that theoretically can be used by any TCP application. Tor does not aim to do any *"protocol cleaning"* [54].

> *"Tor [...] relies on the filtering features of privacy-enhancing application-level proxies"* [54]
>
> *"Tor by itself is NOT all you need to maintain your anonymity."* [34]

With respect to the above-mentioned it is obvious that Tor is one important "building block" to be used if a specific Internet application is to provide anonymity. Network-level anonymity gained through the use of Tor can easily be destroyed at the application-level. Another "building block" needs to provide protocol cleaning and it is either a proxy that sits between Tor and the application, or the application is made "Tor-aware" and avoids and prevents privacy problems itself.

1.3 Use Cases

This section describes two different use cases, showing why one would use Tor to send and receive emails. Both use cases list a few examples.

1.3.1 Location Privacy

Although there is a common misconception that it does not make sense to use Tor with real identities, it makes totally sense to do that. Users that do not mind disclosing their identity, but do not want to disclose their location might use Tor to make it harder for an adversary to determine their location. A few examples:

- an employee does not want to disclose their holiday location to their employer by sending an email
- someone sending emails to public mailing lists does not want to publish their location to an unknown subscriber list
- someone does not want their email provider to track their movements

Location privacy can be an absolute or limited requirement to a user. A user might not mind disclosing his location while he is at home, but when being at work he might does (or vice versa).

1.3.2 Pseudonymity

Pseudonymous email communication usually requires a stronger level of protection, because someone using a pseudonym usually wants to protect his real identity. A pseudonym is defined by Pfitzmann and Hausen as follows:

> *"A pseudonym is an identifier of a subject other than one of the subject's real names."* [8]

There are public pseudonyms where the link between the real name an the pseudonym is publicly known. We assume non-public pseudonyms are used. With increasing use of a pseudonym the risk of linkability to the real identity is

rising because the amount of information known about a pseudonym is increasing with its usage. An identity may generate multiple pseudonyms to use different pseudonyms with different communication partners. This reduces the amount of information associated with a particular pseudonym.

For instance, someone might use a specific pseudonym only for a specific mailing list and when joining a different one, a new pseudonym is generated to prevent the collection of interests of an identity. The collection of interests and the shown knowledge form a profile and may allow the linking of a pseudonym with the real identity behind it.

A pseudonym holder should be aware of the risks associated with a long term pseudonym. We cover only the prevention of linkability at a technical level. Using pseudonyms allows two-way communication, something that is not possible when the sender remains anonymous.
Examples for sending emails via Tor for stronger pseudonymity:

- an employee criticizing their company does not want to loose their job
- an informant that wants to stay unidentified
- an undercover agent does not want the receiver to know from where he is sending emails from

An anonymous usage scenario is not covered because we require two-way communication – which is not possible if the sender remains completely anonymous. An additional benefit that might come with the use of Tor is censorship circumvention. This would allow users to freely choose their email provider in case they are blocked.

1.4 Motivation

1.4.1 Why an Email Client?

Currently the Tor Browser Bundle already offers a way to send and receive mail pseudonymously or for location privacy via webmail. Why should one then bother using an email client? An email client offers a variety of improvements over webmail.

User Experience. Using Tor to route the traffic for Internet services introduces an additional non-trivial amount of latency on top of the usual latency of the Internet connection. High latency has an impact on the user experience when using webmail because every action performed needs to be transported to the server and the response needs to reach the browser. The latency is also present when fetching mails with a mail client, but the mail client retrieves a mail usually only once. Webmail performs actions on remote data, email clients can perform actions on local data after retrieving them once – which omits the latency and improves the user experience.

Confidentiality. Email clients give the user the possibility to use end-to-end encryption. Most of today's email clients come with integrated S/MIME support or are easily extensible to support OpenPGP. End-to-end encryption is hardly feasible when using webmail. Webmail providers that offer PGP in their webinterface do not provide real end-to-end encryption. It is possible to achieve end-to-end encryption while using webmail, by copying the encrypted text block directly into the webinterface or by using browser extensions (FireGPG) but the user experience suffers compared to mail clients. Email clients can store contact details offline and do not need to disclose them to the email provider. Mails that are deleted before they are sent are never disclosed to the mail provider.

Privacy. If an email is stored locally (mail client) the user can access, copy and manipulate it without the email provider even noticing. Text is typed locally and the server is not able to fingerprint typing behavior [43].

1.4.2 Why Mozilla Thunderbird?

The previous section discussed the benefits of mail clients. This section provides arguments why we choose Mozilla Thunderbird. Thunderbird has a few important properties that make him a good candidate to be used as opposed to other mail clients:

- cross-platform

  Thunderbird runs on multiple platforms and operating systems. It is important to have all users, regardless of OS, in one anonymity set, and it is also important to gather many users.

- multi-language support

  Thunderbird is available in over 50 different languages.

- free software

  Thunderbird is tri-licensed under the Mozilla Public License, the GNU General Public License and the GNU Lesser General Public License. These are free software licenses.

- free of charge

  Thunderbird does not come with license costs.

Thunderbird is probably not the only mail user agent (MUA) that has the above mentioned properties, but it is probably the most popular amongst those with these properties. Users are unlikely to change their mail client, therefore privacy and anonymity needs to be integrated in popular software, because the anonymity of a system also depends on its user base. A small user base represents a small anonymity set. Small anonymity sets provide weak anonymity.

1.5 Functional Requirements

The efforts to restrict information leakage for privacy reasons should not restrict Thunderbird's functionality and usefulness. Restricting any of the following features would drastically reduce the potential user base, which is not acceptable. Restricting the potential user base reduces the size of the anonymity set and is counter productive. These are our functional requirements:

- two-way communication
- UTF-8 support in body and email header (multi-language support)
- threading support
- support for attachments (it might be all right to restrict it to certain content-types)

- the submission and retrieval should not be significantly delayed compared to "normal" mail (<10min)

1.6 Related Work

1.6.1 Remailer

David Chaum introduced untraceable electronic mail using cascades of mixes over 30 years ago [22]. Since then different types of anonymous remailers have been implemented and used. A remailer aims to protect a sender by removing or replacing sender identifiable header data. Anon.penet.fi was a popular remailer in the 90's. Its design, however, was weak because everyone had to trust one central host to keep the link between a real name and a pseudonym secret. Eventually, the operator was forced to reveal certain identities in court and the remailer was shut down. The cypherpunk remailer (type I remailer) – created by Hughes and Finney - has a distributed trust model and uses public key cryptography. The type I design is no longer dependent on a single server and uses a network of servers. To link an identity to a pseudonym the entire chain of used servers must be compromised. Type I remailer did not support replies from the beginning. The ability to send replies was added with the reply block, but it is vulnerable to replay attacks [26, 27]. Type I remailers are not safe against traffic analysis. Someone monitoring the network links can link a sender to their recipients without compromising a server in the network. The use of type I remailers is discouraged by the successors, Mixmaster and Mixminon.

> *"Use of Type I remailers for any purpose is discouraged."* [28]

> *"[...] we can finally retire the Type I remailer network ."* [24]

Mixmaster's (type II remailer) first version was designed in 1994 by Lance Cottrell to safeguard against traffic analysis by reordering, padding and delaying messages [26]. Even a powerful adversary that is able to monitor all network links should not be able to link the sender to a recipient. Mixminion designed by Danezies, Dingledine and Mathewson is a type III remailer. Mixminion features link encryption with ephemeral keys to provide forward secrecy and the possibility to send replies using reply blocks that are not vulnerable to replay attacks (single-use). The Mixminion remailer was started by the Tor designers Dingledine and Mathewson but it is not actively being developed anymore. Roger Dingledine stated on the mixminon-dev mailing list in May 2010:

> *"[...] while high-latency mix systems can provide theoretically stronger anonymity than low-latency systems like Tor, without actual users this theoretical anonymity is not actually better in practice."* [25]

The Pynchon Gate [29] was designed by Sassaman, Cohen and Mathewson. It uses type II and III remailer in combination with bittorent as its building blocks to prevent the linking of a recipient with their pseudonym.

State of the art remailer designs can protect more information (who sends mail to whom) against more powerful adversaries than Tor can, but they have a smaller user base and network and are not as actively being developed and used as Tor. Tor - like all low-latency anonymity networks - is not able to provide protection against a global adversary that is able to monitor all communication links.

Remailers may introduce delays of hours to days. Even though email is asynchronous by nature, today's users expect email to arrive within minutes. Delaying of mails over a number of minutes to hours is likely to be unacceptable by the majority of mail users. Using email clients with Tor will not introduce additional latency that is above a few minutes, but the main issue with remailers currently is probably usability.

1.6.2 Torbutton

Torbutton is an addon for the web browser Mozilla Firefox. Torbutton protects against many application- level privacy leaks and attacks. Torbutton enables the use of anonymous browsing on top of network-level anonymity provided by Tor. It features a toggle model where the user can easily switch between Tor and Non-Tor activity. Due to security concerns against the safe use and the high effort to keep the toggle model (state separation) safe, the developer of Torbutton, Mike Perry, announced the end of the toggle model and the end of Torbutton on the tor-talk mailing list in April 2011 [30, 31]. Torbutton has been removed from the Mozilla Addons web site with the release of Torbutton 1.4.0 [32] but is still available on the Tor Project website [33]. Torbutton's privacy protecting functionality is still available in the Tor Browser Bundle. The Tor Browser Bundle includes Tor, Vidalia (a graphical user interface for Tor), a modified Firefox and a few default addons like Torbutton, HTTPS-Everywhere, NoScript and BetterPrivacy. The incorporated HTTP proxy is no longer required.

Torbutton provides Firefox with protection that mail users would need for Thunderbird, but Thunderbird is not a browser and therefore Torbutton is not suitable for Thunderbird – even though the initial versions of it were compatible with it.

1.6.3 Known Anonymizer Configurations for Thunderbird

Currently, there is little information available that addresses the use of Mozilla Thunderbird in combination with Tor or anonymizer in general. The JonDonym website, has a setup guide for Thunderbird [35] in combination with JAP [36]. In May 2011 a developer of the Tor live CD tails [37] added a few configuration steps for Thunderbird to the Tor wiki [13, 14] that are more extensive than [35] and are marked as "experimental".

2 Threat Model

2.1 Assumptions

We assume that SSL/TLS [60] is reasonable secure to protect confidentiality, authenticity and integrity against active and passive attacks. Nevertheless, the near past has shown that this is not always the case [52]. We acknowledge that this assumption might be to strong and the trust in public key infrastructure as we know it today needs to be revisited [57]. New protocols are in preparation [53] to provide new ways to verify authenticity of a public key or certifi-

cate. New standards may or may not replace the current certificate authorities as trust anchor [58].

We assume that the protocols SMTP, POP3 and IMAP are used over a secure channel (SSL/TLS or STARTTLS).

2.2 Players

We consider the following adversaries:

- email service provider
- communication partner
- malicious exit node

The most powerful adversary is representing all three roles at the same time. He runs the exit node, he is our email provider and also recipient. If we are safe against him, we are safe against all three adversaries individually too. We do not consider an adversary with physical access. Our adversaries are not powerful enough to monitor the Internet uplink of the user. As we use Tor we do not aim to protect against traffic confirmation attacks. We do not protect against a global passive adversary. The adversaries have similar goals but a different set of possibilities.

An adversary reached his goal if he can link a pseudonym to its holder or if he can link multiple pseudonyms. In the case of location privacy the adversary reached his goal if he can determine the location of the author at the time of the email submission.

Linkability is defined by Pfitzmann and Hansen as follows:

> *"Linkability of two or more items of interest (IOIs, e.g., subjects, messages, actions, ...) from an attacker's perspective means that within the system (comprising these and possibly other items), the attacker can sufficiently distinguish whether these IOIs are related or not. "* [8]

2.2.1 User

The user wants to send and receive emails while preserving his privacy. Depending on the use case, privacy has a specific meaning to the user. For the use case defined in 1.3.1 (location privacy) it means that an adversary is unable to link the user to its location (at a specific time). For the use case defined in section 1.3.2 (pseudonymity) it means the unlinkability of a pseudonym with its holder.

The user accepts the linkability of his email address (real identity or pseudonym) with his communication partner (email providers and others can see the connection). Tor's design goal is to make linkability of communication partners harder *"Tor seeks to frustrate attackers from linking communication partners"* [54], but in our setting Tor is used to prevent the linkability of the email client with its email provider and not to prevent the linkability of the sending email address with the receiving email address. Email communication is a level above.

The summary of contacts or groups of a person might already be enough to reveal his identity [55, 56]. To avoid linkability through communication partners, the user might create multiple pseudonyms at different service providers. Using multiple pseudonyms concurrently might make them linkable. The user may protect the mail content with end-to-end encryption mechanisms and may require signed or encrypted email from his communication partners.

2.2.2 Email Service Provider

The email service provider has the ability to directly communicate with the user (via Tor) using the protocols in today's email landscape: SMTP, POP3, IMAP. The communication is always initiated by the client. The email service provider may respect or disrespect the protocol specifications (RFCs). An email server may mangle email headers and their content before delivery. An email server may insert any identifying information in outgoing emails before delivering them to the recipient. The email provider may cooperate with other email providers to reveal the identity or location of a subscriber or to link multiple mailboxes.

The email service provider is the most powerful adversary because he is the only adversary that has the ability to directly communicate with the user and is able to see with whom a pseudonym is communicating at what times. The mail provider is not necessarily able to read the mail body or remain undetected when manipulating or spoofing the body of mails.

2.2.3 Malicious Exit Node

An exit node is more powerful than a router after and before him. Using the Tor circuit ID, he is able to associate multiple TCP connections (streams) from the user (for a limited amount of time).

This adversary is able to actively manipulate the traffic routed through him. The exit node likes clear text protocols and long circuit live times. The exit node may cooperate with the email service provider. He should not be able to link circuits or streams with email addresses without cooperating with the email service provider or email recipient.

2.2.4 Communication Partner

A communication partner has far fewer possibilities when compared with the email provider. The communication partner can only communicate via the email server with the user and has never a direct connection to the user. The communication partner has, compared to the email server only one protocol to indirectly interact with the user: SMTP. The communication partner has no means to interact with the user via POP3 or IMAP. The communication partner must adhere to the SMTP and other related standards to successfully deliver a mail to the user's mailbox. The communication partner has only one capability that the mail provider does not have. The communication partner can decrypt emails addressed to him and might send emails to the user with a trusted email body signature.

2.3 Unavoidable and accepted Information Disclosures

When communicating via email certain information disclosures are almost unavoidable and need to be accepted for communication to function. Someone writing an English text discloses the fact that he most likely knows English. If emails are usually answered within a certain time frame after they were sent, the recipient can derive the interval in which emails are fetched from the server. And furthermore it is obvious that a user discloses the frequency and point

in time when he checks his mailbox to the email provider. This can be useful for the mail provider when trying to link multiple mailboxes to one holder.

Using Tor to send mails discloses communication partners and timestamps to the email service provider(s) and potentially others.

We agree to disclose the fact that we use Thunderbird, because it is available for many platforms and languages. We do not accept to disclose of the fact that we use Enigmail, because we do not want to separate Enigmail users from non-Enigmail users.

One might argue that revealing the used MUA is already a risk because it makes targeted attacks easier. This is a valid point. We do not actively reveal our MUA and the proposed changes reduce the identifying information considerably, but this does not stop an attacker from detecting Thunderbird, because Thunderbird most likely has a unique MUA fingerprint in terms of supported protocol and header features.

3 Analysis

3.1 Scope

It is important to define a scope of an analysis to give an idea of how much of the given area has been analyzed, what has not been analyzed and what should probably get some more attention in the future to reach an acceptable coverage. The personal threat model influences the required coverage. The scope of this analysis has included a series of RFCs, a traffic analysis and an inspection of Thunderbird's email headers in order to reveal privacy relevant issues and information leaks. The scope has also been influenced by the server side support for protocols and protocol extensions to avoid analysis of protocol features that are supported by Thunderbird but rarely supported by freemail providers as a result of which they are not frequently used.

The following list shows which RFCs have been inspected. The list does not imply that Thunderbird (v3.1.x) supports/implements these RFCs.

- RFC5321 - SMTP
- RFC4954 - SMTP Service Extension for Authentication
- RFC5322 - Internet Message Format
- RFC1939 - POP3
- RFC2449 - POP3 Extension Mechanism
- RFC3501 - IMAP4rev1
- RFC4409 - Message Submission for Mail
- RFC2045 - MIME Part One
- RFC2046 - MIME Part Two
- RFC2017 - Definition of the URL MIME External-Body Access-Type
- RFC2397 - The data URL scheme
- RFC2047 - MIME Part Three
- RFC3798 - Message Disposition Notification
- RFC2646 - The Text/Plain Format Parameter
- RFC2183 - The Content-Disposition Header Field
- RFC2920 - SMTP PIPELINING
- RFC3207 - SMTP STARTTLS

Besides the classic email protocols like SMTP, POP3 and IMAP, Thunderbird has also non-email features which have, however, not been included in the scope of this analysis:

- NNTP
- RSS
- LDAP

The scope has further not included Thunderbird addons except for the Enigmail extension (v1.1.2) that provides OpenPGP functionality to Thunderbird users.

Thunderbird's usage of HTTP for autoconfiguration, update management, PGP key retrieval and certificate status checks (OCSP) have been covered in this analysis.

3.2 Test Environment

The test environment consisted of different Microsoft Windows and GNU/Linux virtual machines (VMs). They have been installed with different timezones and different language settings. VMs: Windows XP Professional, Windows Server 2008, Windows Vista, Fedora 14, RHEL 6 Beta, Debian Wheezy, Linuxmint 10, Ubuntu 10.10, Mandriva 2010.2, OpenSUSE 11.4; A minimal Debian VM provided SocksPort, DNSPort and TransPort to the internal network (192.168.12.0/24). Every TCP traffic that was not explicitly send to Tor was redirected to the TransPort (by the Debian VM).

This analysis used primarily Thunderbird version 3.1.x (3.1.9 - 3.1.11) but to answer some specific questions older and newer versions have also been involved: 1.5.0.14, 2.0.0.9, 5.0b1, 5.0b2.

3.3 Methodology

We have performed the analysis in three individual steps.

1. Protocol and Header Specifications Analysis (RFCs)

   The protocol and specifications analysis has the goal to find protocol features that are relevant for this analysis because of their privacy implications. Findings in this part may be also relevant to other MUAs because they are not Thunderbird specific. The knowledge about header specifications is important when removing header fields to prevent information disclosure. The specifications define which header fields can safely removed and which are mandatory.

2. Analysing Thunderbirds Mail Header

   This part of the analysis covers mail header specifically to Thunderbirds implementation. Even if the specification did not indicate privacy problems the specific implementation can still be problematic.

3. Network Traffic Analysis

   This is a typical analysis to see if the given application – in our case Thunderbird – is safe to use with Tor from a traffic point of view. This part of the analysis aims to reveal problems where Thunderbird or Enigmail sends request without using the specified proxy or uses cleartext protocols.

Easy solutions to problems (configuration changes) are to be found within sections 3.5 and 3.6. Proposed solutions to more complex problems are discussed in section 4.2 (Defense).

3.4 Protocol and Header Specifications Analysis

The different specifications have been analyzed to spot protocol features that might have an impact on the anonymity of the user when used by the user or actively by an adversary to reveal otherwise undisclosed information.

3.4.1 SMTP

SMTP is used by mail user agents (MUAs) when submitting an email to the mail server. SMTP is a human readable protocol. The most relevant command for this analysis is the EHLO command because it is intended to identify the client with the server. The EHLO SMTP command is required to submit an email:

> *"In any event, a client MUST issue HELO or EHLO before starting a mail transaction."* [1]

Hence simply omitting the entire command is not possible. Usually the argument to the EHLO command is the fully qualified domain name (FQDN) of the client. If the client has no FQDN, it should provide the IPv4 address in brackets or its IPv6 address. The RFC specifies the following restrictions for FQDNs:

> *"A domain name that is not in FQDN form is no more than a local alias. Local aliases MUST NOT appear in any SMTP transaction. [...] The domain name given in the EHLO command MUST be either a primary host name (a domain name that resolves to an address RR) or, if the host has no name, an address literal"* [1]

RFC5321 does not specify any requirements for the IP address, hence non-public IP addresses are not forbidden.

Despite the requirements in RFC5321, today's mail servers do accept EHLO arguments like "localhost" or "localhost.localdomain" and are not uncommon. Gmail's SMTP servers even accept EHLO commands without any argument.

In reply to the EHLO command the SMTP server tells the client which SMTP extensions it supports. The MUA proceeds with specifying his email address (MAIL) and one ore more recipients (RCPT) and finally it will issue the DATA command. After the server acknowledges the DATA command with status code 354 it accepts multiple lines until it receives a line with a single dot followed immediately by carriage return line feed (CRLF). The actual text submitted in those lines is subject to a different RFC (RFC5322), but SMTP servers modify the submitted text by adding a trace line at the top every time a message is relayed by them. The trace entry is required and contains, among other things, the EHLO argument given by the SMTP client, its public IP address and the DNS reverse lookup result.

RFC5321 requires the trace entry in the email header:

> *"When an SMTP server receives a message for delivery or further processing, it MUST insert trace ('time stamp' or 'Received') information at the beginning of the message content"* [1]

Some providers do not add the trace entry for the MUA to MTA connection to protect the customers' privacy. Some companies do the same to prevent the disclosure of internal IP addressing and naming schemes. The RFC explicitly mentions this information disclosure in an own subsection:

> *"In some circumstances, such as when mail originates from within a LAN whose hosts are not directly on the public Internet, trace ('Received') header fields produced in conformance with this specification may disclose host names and similar information that would not normally be available. This ordinarily does not pose a problem, but sites with special concerns about name disclosure should be aware of it."* [1]

3.4.2 Internet Message Format (RFC5322)

The RFC5322 specifies the rules for the text that is transmitted with the SMTP DATA command. The RFC does not require any specific order of header fields, only the trace header field must not be reordered, others should not be reordered. The header is typically one of the places where most of the involuntary day to day information leaking takes place. A typical email header contains a variety of different standardized and non-standardized header fields. Usually only a few of them are actually required or useful to the user. Most of the header fields are added by MTAs as the mail traverses them, and are not under the control of the MUA. For our analysis only the header fields that are influenced or generated by the MUA are of interest. Header fields generated by the MTA independently from the MUA are not seen as sensitive because pseudonymous users should not use their own MTA. Therefore MTA information does not belong directly to the MUA and the user.

The important first *Received* header field is described in chapter 3.5.1. A few header fields generated by the MUA:

- *User-Agent*

  While it usually does not provide any useful functionality for the end user, it discloses a great deal about the MUA and the operating system.

- *MIME-Version*

  MIME is required when the user wants to send more than just US-ASCII emails without attachments. This header field discloses little to nothing and provides useful functionality. Header fields beginning with "Content-" are MIME header fields and are not part of RFC5322. MIME is discussed in the next section.

- *Message-ID*

  The *Message-ID* header field must be a globally unique identifier for an email and is useful because it allows MUAs - in connection with other header fields - to associate messages to a thread. This is especially useful in terms of mailing lists. Thunderbird specifics related to this header field are discussed in section 3.5.4, dealing with the mail header analysis.

- *Date*

  The *Date* header field specifies the sender's local time including the timezone (UTC offset) when the user completed the mail. In cases where the MUA was offline this timestamp does not has to match the timestamp in the *Received* header field. The timezone is sensitive because it allows to reduce the anonymity set and discloses the approximate location of the user. The timestamp in the *Date* header field is usually automatically converted to the receivers local timezone when displayed by the receiver's MUA. It is possible to specify a valid *Date* header field without disclosing the timezone. The UTC offset "+0000" is used for the UTC timezone. "-0000" is short for *"the date-time contains no information about the local time zone."* [2].

When trying to reduce unnecessary information disclosure it is important to know which header fields are mandatory. According to RFC5322 only *Date* and *From* are mandatory fields.

Limiting the header fields to the mandatory fields to prevent information disclosure is not desirable, because it seriously restricts the feature set, including some of our functional requirements defined in section 1.5 and would not be used by a broad user base.

Even with mandatory header fields only, some fingerprinting possibilities would still persist. Further discussion about email header fields specific to Thunderbird and what we propose to prevent unnecessary information disclosure is discussed in section 3.5 and section 4.2.

3.4.3 Multipurpose Internet Mail Extensions (MIME)

MIME has been designed to allow additional non-US-ASCII characters and attachments including binary data.

Content-Type Header Field

This field describes the data contained in the body of the mail (media type). A *Content-Type* is divided into top level type and subtype. While there are only a few top level types that rarely are extended (text, application , audio , example , image , message , model , multipart , video) there are hundreds of subtypes. IANA assigns and lists content-types at their website [45].

The *Content-Type* header field is typically used by MIME aware MUAs to choose the appropriate steps to present the data to the user when parsing a message. In case of attachments the content-types are typically associated with external programs that understand the data. External applications are dangerous from a privacy point of view, because as soon as an external program starts it can cause network connections that do not adhere to the MUA's proxy settings and will likely deanonymize the user. External programs are generally unsafe.

The top level type multipart has a mandatory parameter *boundary*. The *boundary* parameter is used as a delimiter between the body parts. The delimiter consists of a random string. Thunderbird specifics for this string are discussed in section 3.5.6.

The *charset* parameter is used for text media types and specifies the used character set. The character set might be specific to a language and region and is potentially sensitive. While analysing the initial set of content types specified in RFC2046, one seemed to be particularly interesting: *message/external-body*.

The external-body subtype basically includes only a reference to the body but not the body itself and tells the MUA how to fetch the body. RFC2046 defines a few ways to access the referenced body via so called access-types (ftp, anon-ftp, tftp, local-file, mail-server, url (RFC2017)). IANA lists all defined access-types on their website [41].

Thunderbird aims to support these access-types, including the url access-type, but practical tests have not confirmed that. This features might become problematic in future if the implementation becomes functional. Thunderbird does not implement MIME completely.

Content-Transfer-Encoding Header Field

SMTP is a text based protocol. To transport arbitrary data (i.e. binary files) over SMTP the data needs to be properly encoded (represented) in the available US-ASCII character set. Possible values for this header field are: 7bit, 8bit, quoted-printable, base64 and binary.

3.4.4 IMAP

IMAP4rev1 is specified in RFC3501. IMAP offers the user a mailbox that is kept on the server and can be managed via a MUA. IMAP allows the user to manage one central mailbox from different locations and devices with multiple MUAs. When being online all of them have basically the same view of the mailbox. States of emails - like "unread" - are also kept on the server via flags. These properties make IMAP the first choice when both mailbox access protocols (IMAP and POP3) are available.

IMAP features have some serious downsides from a privacy perspective. Actions within the MUA cause IMAP commands that are sent to the server. This allows the email provider to determine which content (folder, mails) the user might cares more about. The usage behavior data generated by the user when using his IMAP client, can be used by the provider for profiling.

For example, if a user clicks on a new message, the server knows about it by receiving a command from the MUA that asks him to add the "\Seen" flag:

```
uid store 301749458 +Flags (\Seen)
```

Listing 1: IMAP command: mail gets the \Seen flag

If a user clicks on the "Trash" folder for the first time during an IMAP session, the server knows about it by receiving the following command:

```
select "Trash"
```

Listing 2: IMAP command: User selects the “Trash” folder

Besides the general problem that IMAP usage brings along, there might arise privacy issues by using the following IMAP commands:

- The SEARCH command includes the search criteria and an optional charset. The charset in use might be specific to a country or region and could potentially disclose the approximate location of the user. The search string is also relevant.
- An IMAP client does not only perform commands that result in data flowing from the server to the client. The IMAP protocol is also used to upload data to the IMAP server. When a user composes a new message, it is usually automatically submitted to the IMAP server (autosave feature of the MUA). When storing the composed messages on the IMAP server all mail headers are included, even those that might have only local meaning (e.g. *X-Identity-Key*). The APPEND command is used to upload the entire mail (including the header part).

Most of the actions in the client result in an IMAP command but that does not necessarily imply that the server knows what the user did in the client – the server knows that the user, or the MUA itself, probably did something.

An example would be the user clicking on an already fetched email in the client, this results in an NOOP command being sent to the server. The NOOP command does nothing and has no command argument, it is just used to poll the server for new emails.

3.4.5 POP3

POP3 is a simple protocol specified in RFC1939. RFC2449 specifies a way to extent POP3 functionality and includes some extensions. POP3 basically transfers data from the server to the client only. This one-way property of POP3 is great from a privacy point of view because it limits the potential information disclosure already on a protocol design level. The POP3 protocol allows extensions to be specified in new RFCs but any additional complexity and features that go into the direction of IMAP are discouraged by RFC2449:

> *“Future extensions to POP3 are in general discouraged, as POP3’s usefulness lies in its simplicity. POP3 is intended as a download-and-delete protocol; mail access capabilities are available in IMAP”* [6]

This gives some confidence that POP3’s command set remains (quite) static and makes it unlikely that the new protocol extension would have an impact on privacy.

Depending on the server configuration, it is possible to keep emails on the server after retrieving them with the RETR command. The command to mark a message for deletion (DELE) is optional. A server might not allow to keep emails on the server. A POP3 server may restricts the size of a mailbox or the retention time of mails. Using the unique ID (UIDL command) POP3 clients can emulate something similar to IMAP (without server side flags), but with a growing number of mails in the mailbox this becomes quickly inefficient. Keeping mails in the mailbox after retrieval has also security implications. Removing them from the server after retrieval limits the impact of server or account compromises to future mails while past emails remain unaffected if the user chooses to remove mails after retrieval.

Two potential privacy attacks covering POP3 are described in section 4.1.2 and 4.1.3.

3.4.6 POP3 vs. IMAP

Whenever a new email account is configured in an MUA a user needs to decide whether to use either the POP3 or IMAP protocol to retrieve emails from the server. The previous two sections described briefly IMAP and POP3, including their privacy implications. In terms of privacy and security, POP3 is preferred to IMAP, because:

- the POP3 protocol design drastically limits the amount of potentially disclosed information to the server (one-way)
- folder structure and names in POP3 accounts are not disclosed to the server
- sent mails and drafts are not disclosed to the POP3 server
- user actions (selecting/reading mails, choosing folder, creating folder, searching, ...) do not result in protocol commands and are not disclosed to the POP3 server – this significantly limits fingerprinting and user behavior capabilities
- POP3 causes fewer network traffic than IMAP – this might makes certain attacks harder
- regarding latency, POP3 user experience will be better in comparision to IMAP (POP3 keeps data locally)
- using POP3 (incl. DELE command) reduces the impact of server/account compromises by third-parties and does not affect past mails. This is not a valid argument when considering the mail server as a malicious party, but nonetheless it is considered an improvement over mail storage on always online systems.
- POP3 protocol is simpler than IMAP and therefore has a smaller attack surface

3.5 Thunderbird Mail (Header) Analysis

This section contains an analysis of Thunderbirds mail header information disclosures.

3.5.1 Received Header Field

```
Received: from [192.168.12.18] (server104800.
    santrex.net [193.106.172.85])
        by mx.example.com with ESMTPS id
            o17sm777717fal.1.2011.04.06.11.52.48
```

```
        (version=SSLv3 cipher=OTHER);
        Wed, 06 Apr 2011 11:52:49 -0700 (PDT)
```

Listing 3: First *Received* header field containing EHLO command argument

The first *Received* header field contains the SMTP EHLO argument ("[192.168.12.18]") - which is the IP address of the Internet facing network card from the sender host. In this example Thunderbird was running on a host which was connected to the Internet via a router that performed network address translation (NAT). The EHLO argument is therefore a private IP address (RFC1918) which has limited meaning, but it is still useful for fingerprinting and linkability. In situations where the host is directly connected to the Internet, it would be the public IP address of the host – regardless of whether Tor is used or not.

The hostname and IP address that follows in the parenthesis are determined by the SMTP server from the TCP connection. Using Tor will resolve the issue with the latter IP address because that one is determined by the SMTP server by looking at the TCP connection's endpoint – which is the Tor exit node.

More attention should be paid to the EHLO argument, because it happens at the application level within the SMTP session. Thunderbird offers a preference to statically set the EHLO argument value for all SMTP sessions. The preference name is *mail.smtpserver.default.hello_argument*. Its value is commonly set to "localhost" [13] despite the FQDN requirement in SMTP (see section 3.4.1). We propose to use "[127.0.0.1]".

"ESMTPS" implies that the SMTP session took place over a secure channel. "ESMTPA" would imply that the SMTP session was authenticated (trailing "A"). The timestamp in the *Received* header field is indirectly influenced by the user. The user chooses when he wants to send a message.

3.5.2 User-Agent Header Field

This header field discloses detailed information about the system:

- OS and version
- language
- MUA and version

The *User-Agent* header field can be omitted without problems, because it is not required by RFC5322 and it does not provide any functionality to the user. Creating an empty Thunderbird preference `general.useragent.override` [13] omits this header field entirely - not only in the email header but also in HTTP requests and MDN replies (section 4.1.1).

Omitting vs. Faking

We prefer omitting to faking the *User-Agent* string, because

- users do not need to pretend to use a specific language they do not actually use
- there is no need to upgrade to newer *User-Agent* strings in the future
- no UA might makes it easier for other MUAs to join the same header fingerprint in the future

Torbutton fakes the *User-Agent*. Entirely omitting the *User-Agent* string on the web is not feasible because it would have a serious impact on the user experience. For instance, the online encyclopedia wikipedia is not usable with an empty *User-Agent* string because of script detection mechanisms.

In the context of mail, the *User-Agent* might be used as one of many input parameters of a spam detection algorithm, but the absence does not immediately result in a rejected mail.

3.5.3 Date Header Field

The *Date* header field includes location specific information. The timezone is given as an offset to UTC timezone. Even though there are parts of English words like "Sat" and "Jun" included in the header field value, they do not disclose the language of the system because they are in English in any case.

```
Sat, 11 Jun 2011 13:27:55 -0500
```

Listing 4: Typical Date Header Field Value

The timezone can be manipulated at start time by using the environment variable TZ, but the timestamp itself brings another problem along. The timestamp is specific to the local system. The time difference between the *Date* header value and the first *Received* header value can be used to fingerprint a client – especially if the clients time settings are wrong. The offset can be used to link multiple email accounts to the same origin (in combination with other inputs). The time offset in the header excerpt shown in the listing 5 is -21690 seconds (6h 90s). The client has probably a wrong timezone or an incorrect clock. Offsets near 0 (<100s) are less useful for an adversary as probability of their occurrence is less unique.

```
Received: from [192.168.12.18] (server104800.
    santrex.net [193.106.172.85])
        by mx.example.com with ESMTPS id
            o17xm777717fal.1.2011.04.06.11.52.48
        (version=SSLv3 cipher=OTHER);
        Wed, 06 Apr 2011 11:52:49 -0700 (PDT)
Message-ID: <4D9C61C7.40206@example.com>
Date: Wed, 06 Apr 2011 20:51:19 +0800
```

Listing 5: Mail header excerpt from a sender with wrong time

The *Date* header field is one of two mandatory header fields. Potential ways to solve the problem are discussed in section 4.2.3

3.5.4 Message-ID Header Field

Thunderbird *Message-ID*s consist of two parts, the Unix timestamp in hexadecimal format (matching the time in the *Date* header field) and a random number followed by an "@" and the domain.

```
Message-ID: <4D9C61C7.40206@example.com>
```

Listing 6: *Message-ID* with timestamp in hexadecimal form

4D9C61C7 => 1302094279 => Wed, 06 Apr 2011 12:51:19 GMT

The length of the second part ("40206") varies because leading zeros are not shown. If there is any difference between Thunderbird versions in the algorithm that generates the *Message-ID*, then the generated string would result in information disclosure and could be used to determine if someone might use Thunderbird version X or not. A brief inspection of the source code of Thunderbird version 1.5.0.14, 2.0.0.9 and 3.1.9 showed that the code to generate the *Message-ID* was not changed in these versions. The code in version 5.0b1 was slightly changed but the changes do not seem to result in a change of the *Message-ID* format of Thunderbird. With the currently available Thunderbird version the *Message-ID* cannot be used to build a Thunderbird version detection. Every future version that changes the format introduces a possibility to detect the Thunderbird version.

3.5.5 From Header Field

```
From: user <user@example.com>
To: user@example.com
```

Listing 7: *From* and *To* header fields with and without display name

The part before <> is the so called display name and is optional. The display name should be omitted to reduce unnecessary data and the possibility to create different anonymity sets. Whenever a single email account is configured on multiple hosts, the *From* header field value should match (case sensitive) to avoid the detection of multiple clients.

3.5.6 Content-Type Header Field

Certain MIME content types require a boundary parameter to separate body parts. In case of Thunderbird it is a random number with preceding dashes. The way to generate the number was not changed since 1.5.0.14 (and probably earlier). The boundary string cannot be used for a Thunderbird version detection.

Typical boundary string:

```
Content-Type: multipart/mixed;
 boundary="------------070804020700050606060907"
```

Listing 8: MIME: Typical Thunderbird *boundary* parameter

The function that generates the *boundary* string accepts a prefix string that is inserted between the dashes and the numbers. The function is usually called with an empty string (""), but in some cases the prefix is non-empty:

- S/MIME signed messages have an "ms" prefix ("ms" stands probably for *multipart/signed*)
- Message delivery notifications have an "mdn" prefix

These prefixes are not problematic because they do not reveal OS or version information as long as they are used in all Thunderbird versions in the same situations. Another prefix – the "ad" prefix - seems to be used for appledouble files – which are MAC OS X specific. Appledouble files use content type *multipart/appledouble* [10]. The content type *multipart/applefile* seems also to be MAC OS X specific. The presence of the boundary part shown in listing 9 can be used to confirm the use of MAC OS X, but not all mails from a MAC OS X use this prefix – only appledouble files.

```
------------ad
```

Listing 9: Beginning of an appledouble *boundary* delimiter

The relevant code can be found in the Thunderbird source code [11] file:
`mailnews/compose/src/nsMsgAttachmentHandler.cpp`.
Our test environment did not include a MAC OS X. We did not verify this observation. It might be the case that boundary strings are generated without using the function `mime_make_separator()`.

Enigmail – a Thunderbird extension that provides PGP support - can be used in PGP-inline or in PGP/MIME mode. PGP/MIME is specified in RFC3156 [9]. Enigmail recommends to use PGP-inline for interoperability reasons. Enigmail defaults to PGP-inline. When Enigmail is used in PGP/MIME mode the boundary string is Enigmail specific and it always starts with the string shown in listing 10.

```
------------enig
```

Listing 10: Beginning of an Enigmail *boundary* delimiter

By inspecting the *boundary* parameter one can detect the use of Enigmail. Currently Enigmail is the only available extension that provides PGP support for Thunderbird, but one could also send a PGP mail by copying a valid PGP block into a plaintext mail. The Enigmail specific string can be omitted by using PGP-inline. The prefix can not be changed without changing Enigmail's source code [12]. The `MakeBoundary()` function is always called with the "enig" parameter – which is the prefix.

Character Set

The *charset* parameter specifies the used character set for the *text/plain* content type and is sensitive because it might disclose language and region of the system or at least reduces the anonymity set. This *Content-Type* header field parameter is covered in section 4.2.4.

3.5.7 Enigmail and GPG Version Disclosure

Enigmail and GPG add a few new header fields that disclose version information:

```
[...]
MIME-Version: 1.0
To: user@example.com
Subject: NM: WinXP 64 Prof Enigmail
X-Enigmail-Version:  1.1.1
Content-Type: text/plain; charset=ISO-8859-1
Content-Transfer-Encoding: 7bit

-----BEGIN PGP SIGNED MESSAGE-----
Hash:  SHA1

body
-----BEGIN PGP SIGNATURE-----
Version:  GnuPG v2.0.17 (MingW32)
Comment:  Using GnuPG with Mozilla -
    http://enigmail.mozdev.org/

ixIcBAEB [...] W8QhBxID
=zJwi
-----END PGP SIGNATURE-----
```

Listing 11: Enigmail and GPG Information Disclosure

The lines taken into consideration are highlighted in bold. All of them can be omitted with configuration changes. The *X-Enigmail-Version* header field can be removed by changing Thunderbird's boolean preference `extensions.enigmail.addHeaders` to false. GPG version is removed via the GPG command option `--no-emit-version`. The comment line can be removed by setting `extensions.enigmail.useDefaultComment` to true. The used hash algorithm (SHA1) is selected with the preference `extensions.enigmail.mimeHashAlgorithm` and is covered in section 4.2.4.

3.5.8 Reply Header Type

When replying to a mail, Thunderbird automatically inserts the so called `reply_header_type` before the actual quote of the original text. Despite its name - this is not a header field, it is located in the body of the message. In cases where the transmitted email contains a quoted block - this is usually the case in replies. Listing 12 shows a typical reply header.

```
On 27/04/11 21:16, Bob wrote:
```

Listing 12: Typical Reply Header Type within Mail Body on an English System

The timestamp from the original email is automatically converted to localtime. The recipient of the reply can determine the timezone of the replier, because Bob knows when he sent the email (15:16:14 -0400) - this means his communication partner is likely in UTC +0200. The very same string also discloses Thunderbird's language. The following preferences remove everything but the name:

```
mailnews.reply_header_type = 1
mailnews.reply_header_authorwrote = %s
```

The above example would look like:

```
Bob:
```

Listing 13: Reply Header Type Style with new Settings

3.5.9 Mozilla Specific Header Fields

Thunderbird uses non-standardized header fields for internal and local use. These header fields normally do not appear in outgoing mails. In earlier versions of Thunderbird (2.x) these header fields were disclosed when forwarding mails. Since Thunderbird has changed its forwarding mode of emails to inline this is no longer an issue [59].

An incomplete list of Mozilla header fields:

- X-Mozilla-Status
- X-Mozilla-Status2

  A detailed description for status header fields can be found at [40].
- X-Identity-Key
- X-Account-Key
- X-Mozilla-Keys
- X-Mozilla-Altered
- X-Mozilla-External-Attachment-URL
- X-UIDL
- X-Mozilla-Draft-Info
- X-jsemitter-part-path
- X-Mozilla-IMAP-Part
- X-Priority
- X-Mozilla-PartURL
- X-Mozilla-PartSize
- X-Mozilla-PartDownloaded
- X-Template

These header fields and their functions are either poorly or not at all documented.

Particularly interesting is the *X-Mozilla-External-Attachment-URL* header field. It seems to implement a similar function that is provided by the MIME content-type *message/external-body* (section 3.4.3). Using this header field, we were able to open a browser without the user's explicit consent. The attack is described in section 4.1.4.

3.6 Network Traffic Analysis

The traffic analysis focuses on discovering leaks that are not caused by transmitting information over the SMTP, POP3 or IMAP but are related to the traffic generated by the use of Thunderbird itself and the use of Thunderbird with proxies. When using an application with Tor it is crucial that all traffic caused by the application is routed through the specified proxy. Requests that are not send to the proxy are commonly referred to as "leaking request". If an application leaks requests a destination may be able to associate non-Tor traffic with Tor traffic which results in the deanonymization of the user. Any leaking traffic regardless of content and destination is unacceptable. This section should answer the following questions:

- Is any network traffic from Thunderbird and Enigmail leaking and send without using the configured proxy?
- Are leaks avoidable with specific settings or are code changes required?
- Do Thunderbird or Enigmail generate traffic that is not destined to the SMTP and POP3/IMAP server:port destination?
- Questions specifically regarding traffic that is not destined to the email provider:
  - Which traffic involves cleartext protocols and what is transmitted in those requests?
  - What is disclosed in encrypted connections and to whom?

If not explicitly stated otherwise - requests adhere to the proxy settings and do not leak DNS requests. For the network traffic analysis Thunderbird was used in version 3.1.10 and 3.1.11. For the cleartext detection Thunderbird was used without any proxies. To detect leaking requests the following "Manual proxy configuration" was used:
HTTP Proxy: 192.168.12.1:8118
("*Use this proxy server for all protocols*" is not ticked.)
SSL Proxy: 192.168.12.1:8118
SOCKS Host: 192.168.12.1:9050
192.168.12.1:8118 provides a polipo proxy.
192.168.12.1:9050 provides a Tor SocksPort. 192.168.12.1 is a separate virtual machine (Debian VM). The Thunderbird preference `network.proxy.socks_remote_dns` was set to true [13]. This preference decides whether the client makes DNS requests (false) or the SOCKS proxy (true). Thunderbird does not support the CONNECT method. That would allow arbitrary connections through an HTTP proxy [42].

3.6.1 Thunderbird Start Page

Every time Thunderbird is started, it sends an HTTP request to live.mozillamessaging.com containing specific Thunderbird version, OS, locale and buildid information. According to the WHOIS entry live.mozillamessaging.com belongs to the Mozilla Corporation.

```
GET /thunderbird/start?locale=en-US&version
    =3.1.10&os=WINNT&buildid=20110414114714 HTTP
    /1.1
Host: live.mozillamessaging.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT
    5.2; en-US; rv:1.9.2.17) Gecko/20110414
    Thunderbird/3.1.10
Accept: text/html,application/xhtml+xml,
    application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
```

Listing 14: Cleartext start page request to live.mozillamessaging.com (1)

The server responds with an HTTP redirection (302) which results in the second request:

```
GET /en-US/thunderbird/3.1/start/?uri=/
    thunderbird/start&locale=en-US&version
    =3.1.10&os=WINNT&buildid=20110414114714 HTTP
    /1.1
Host: www.mozillamessaging.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT
    5.2; en-US; rv:1.9.2.17) Gecko/20110414
    Thunderbird/3.1.10
Accept: text/html,application/xhtml+xml,
    application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
```

Listing 15: Cleartext start page request to live.mozillamessaging.com (2)

These requests adhere to Thunderbird's proxy settings and do not leak DNS requests, but they are in cleartext and unnecessary. However, they can easily be omitted by setting the following preference to false:
`mailnews.start_page.enabled`
This preference corresponds to the "*When Thunderbird launches, show the Start Page in the message area*" option in the graphical user interface.

This setting does not disable the following request. When Thunderbird runs for the first time after an update, it requests the "whatsnew" page containing similar URL parameters:

```
GET /thunderbird/whatsnew?locale=en-US&version
    =3.1.11&os=WINNT&buildid=20110616145146 HTTP
    /1.1
Host: live.mozillamessaging.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT
    5.2; en-US; rv:1.9.2.18) Gecko/20110616
    Thunderbird/3.1.11
Accept: text/html,application/xhtml+xml,
    application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
```

Listing 16: Cleartext "whatsnew" request to live.mozillamessaging.com after Thunderbird update (1)

The server responds with an HTTP redirection (302) which results in the following request:

```
GET /en-US/thunderbird/3.1/whatsnew/?uri=/
    thunderbird/whatsnew&locale=en-US&version
    =3.1.11&os=WINNT&buildid=20110616145146 HTTP
    /1.1
Host: www.mozillamessaging.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT
    5.2; en-US; rv:1.9.2.18) Gecko/20110616
    Thunderbird/3.1.11
Accept: text/html,application/xhtml+xml,
    application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
```

Listing 17: Cleartext "whatsnew" request to live.mozillamessaging.com after Thunderbird update (2)

This is also an unnecessary disclosure of information to the exit node, Mozilla and everyone in between. The disclosure to Mozilla may not be a problem because of the Tor usage, but the exit node can associate this cleartext information with other TCP connections on the same Tor circuit (same Tor client).

3.6.2 Mozilla Update Mechanism

Thunderbird automatically checks if there are new Thunderbird versions or updates for installed extensions. Thunderbird distributed in GNU/Linux distributions usually does not check for a new Thunderbird version because the package manager handles updates. The interval in which these

requests to Mozilla server occur is defined by the preference `app.update.interval` and is by default 86400 seconds (1 day). An HTTPS request from the Windows XP virtual machine, checking for new Thunderbird versions:

```
GET /update/3/Thunderbird/3.1.10/20110414114714/
    WINNT_x86-msvc/en-US/release/Windows_NT
    %205.2/default/default/update.xml?force=1
    HTTP/1.1
Host: aus2.mozillamessaging.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT
    5.2; en-US; rv:1.9.2.17) Gecko/20110414
    Thunderbird/3.1.10
Accept: text/html,application/xhtml+xml,
    application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
```

Listing 18: Update check: HTTPS request to aus2.mozillamessaging.com

The HTTPS response body contains the URL to fetch the update and a SHA512 hash of the file.

```
[...]<patch type="partial" URL="http://download.
    mozilla.org/?product=thunderbird-3.1.11-
    partial-3.1.10&amp;os=win&amp;lang=en-US&amp;
    force=1" hashFunction="SHA512" hashValue="
    c92be1f9349d562a9ac6c88f4d9dd8325f932fcc8f
    378be94f573b20d5e47a4c84fcf13b8bf6753d88
    25954
    bb88c1a638d1b6572ff6c9a5d1c2eb6e8136e3deb"
    size="881064"/>
[...]
```

Listing 19: Update check: excerpt of the response body containing URL and file hash

Thunderbird parses the XML file and requests the partial file:

```
http://download.mozilla.org/?product=thunderbird
    -3.1.11-partial-3.1.10&os=win&lang=en-US&
    force=1
```

After a redirection, it fetches the file from a mirror:

```
http://napoleon.acc.umu.se/pub/mozilla.org//
    thunderbird/releases/3.1.11/update/win32/en-
    US/thunderbird-3.1.10-3.1.11.partial.mar
```

Note: The version check occurs via HTTPS and the actual download is taking place via HTTP. The HTTPS response to the version check contains the size and SHA512 hash of the file. If Thunderbird checks the SHA512 hash before applying the downloaded update, then the update mechanism can be considered reasonable safe, but the procedure leaks the Thunderbird version, OS and language to the exit node and everyone after the exit.

The procedure for addon updates is similar, every addon is checked separately. Below there is an HTTPS request for the "CompactHeader" addon on the Fedora 14 virtual machine:

```
GET /update/VersionCheck.php?reqVersion=2&id={58
    D4392A-842E-11DE-B51A-C7B855D89593}&version
    =1.4.0&maxAppVersion=5.*&status=userEnabled&
    appID={3550f703-e582-4d05-9a08-453d09bdfdc6}&
    appVersion=3.1.11&appOS=Linux&appABI=x86_64-
    gcc3&locale=en-US&currentAppVersion=3.1.11&
    updateType=97 HTTP/1.1
Host: versioncheck.addons.mozilla.org
User-Agent: Mozilla/5.0 (X11; U; Linux x86_64; en
    -US; rv:1.9.2.18) Gecko/20110621 Fedora
    /3.1.11-1.fc14 Thunderbird/3.1.11
Accept: text/html,application/xhtml+xml,
    application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache
```

Listing 20: Extension version check: HTTPS request to versioncheck.addons.mozilla.org

Response body contains the download URL and the SHA256 hash of the XPI file:

```
[...]
<em:updateLink>http://releases.mozilla.org/pub/
    mozilla.org/addons/13564/compactheader-1.4.0-
    tb-linux.xpi</em:updateLink>
<em:updateInfoURL>https://addons.mozilla.org/
    versions/updateInfo/1247685/%APP_LOCALE%/</em
    :updateInfoURL>
<em:updateHash>sha256:4
    e28fae49121f646c8bc5cfd40703ecf1257e477bb5
    d421d95dd9bbd912705f0</em:updateHash>
[...]
```

Listing 21: Extension version check: response body containing download URL and file hash

Thunderbird would fetch the update from the following URL (but it has already installed the latest version).

```
http://releases.mozilla.org/pub/mozilla.org/
    addons/13564/compactheader-1.4.0-tb-linux.xpi
```

Not all addon updates are fetched via HTTP, only some of them, depending on the addon. These version checks tell Mozilla what extensions in what versions on which operating system one is using, but Mozilla does not see the email address or email server in update check requests. Everytime a new update is available and downloaded, the used addon (and OS) is also disclosed to the exit node.

3.6.3 Mozilla Blocklist

Mozilla maintains a blocklist for extensions to be able to centrally block frequently crashing or insecure extensions or plugins. Most of the extensions are listed because of a "*high crash volume*" [70], some are blocked for security reasons. When requesting the blocklist Thunderbird submits information about the operation system to Mozilla. The example below was taken from the Fedora 14 virtual machine and contains the Linux Kernel and GTK version. The HTTPS request also includes a long-term cookie.

```
GET /blocklist/3/%7B3550f703-e582-4d05-9a08-453
    d09bdfdc6%7D/3.1.11/Thunderbird
    /20110623003135/Linux_x86_64-gcc3/en-US/
    default/Linux%202.6.35.13-92.fc14.x86_64%20(
    GTK%202.22.0)/default/default/ HTTP/1.1
Host: addons.mozilla.org
```

```
User-Agent: Mozilla/5.0 (X11; U; Linux x86_64; en
    -US; rv:1.9.2.18) Gecko/20110621 Fedora
    /3.1.11-1.fc14 Thunderbird/3.1.11
Accept: text/html,application/xhtml+xml,
    application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
Cache-Control: no-cache
Cookie: BLOCKLIST_v3
    =80.237.226.76.1308782793.7804
```

Listing 22: Mozilla blocklist: HTTPS request to addons.mozilla.org including long-term cookie.

3.6.4 Mozilla Cookies

The Mozilla webserver aus2.mozillamessaging.com sets a cookie via HTTPS when Thunderbird performs the update check to see if there is a new Thunderbird version. The cookie has no secure flag set and it is valid for five years. The following request was generated by a manually installed Thunderbird on the Fedora 14 VM:

```
GET /update/3/Thunderbird/3.1.11/20110623003135/
    Linux_x86_64-gcc3/en-US/default/Linux
    %202.6.35.13-92.fc14.x86_64%20(GTK%202.22.0)/
    default/default/update.xml?force=1 HTTP/1.1
User-Agent: Mozilla/5.0 (X11; U; Linux x86_64; en
    -US; rv:1.9.2.18) Gecko/20110623 Lanikai
    /3.1.11
Host: aus2.mozillamessaging.com
Accept: text/html,application/xhtml+xml,
    application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive Cache-Control: no-cache
```

Response (including Set-Cookie):

```
HTTP/1.1 200 OK
Server: Apache/2.2.3 (CentOS)
X-Powered-By: PHP/5.1.6
Set-Cookie: aus2=78.31.70.182.1308912797.1249;
    expires=Thu, 23-Jun-2016 15:57:07 GMT; path
    =/; domain=aus2.mozillamessaging.com
Cache-Control: max-age=900
Expires: Fri, 24 Jun 2011 11:08:17 GMT
Content-Type: text/xml;
Content-Length: 42
X-Varnish-IP: 10.200.74.52
Date: Fri, 24 Jun 2011 10:53:17 GMT
X-Varnish: 2383121211
Age: 0
Via: 1.1 varnish
Connection: keep-alive
```

Listing 23: aus2.mozillamessaging.com setting an identifying cookie without secure flag and five years validity.

The cookie *aus2* consists of three parts:

- public IP address
- Unix Timestamp (the time when the Set-Cookie was generated)
- the last part is probably a random number between 1 (or 0) and 9999 (sometimes with leading zeros sometimes without)

Due to this properties the cookie is assumed to be a unique. Should Thunderbird – for whatever reason - go online without using Tor, then this cookie would effectively deanonymize the user because it links Tor activity with non-Tor activity. The same issue occurs if a user configures Thunderbird for Tor and the cookie is set before that configuration takes place. addons.mozilla.org also sets a cookie via HTTPS with a different name (BLOCKLIST_v3) but basically consists of the same content and properties. Cookies can easily be disabled by setting `network.cookie.cookieBehavior` to "2".

3.6.5 Account Autoconfiguration

New mail accounts are set up with the "Mail Account Setup" wizard. The wizard tries to automatically configure the account settings. The user does not have to know the mail server settings. The end user provides his name and email address only. This wizard does not completely adhere to the proxy settings and fetches settings over an insecure channel.

The wizard performs multiple steps to provide the user with a working email client configuration without much user intervention [46]:

1. contact the mail service provider via HTTP
2. contact Mozilla via HTTPS
3. configure by probing

The wizard was tested with "johnrandom@lavabit.com" as email address (lavabit.com is the email provider). In the first step Thunderbird tries to fetch the appropriate settings from "autoconfig.lavabit.com" via HTTP. This hostname is always built by combining "autoconfig" with the email providers domain. The HTTP GET parameter that contains the email address and User-Agent:

```
GET http://autoconfig.lavabit.com/mail/config-v1
    .1.xml?emailaddress=johnrandom%40lavabit.com
    HTTP/1.1
Host: autoconfig.lavabit.com
User-Agent: Mozilla/5.0 (X11; U; Linux x86_64; en
    -US; rv:1.9.2.17) Gecko/20110428 Fedora
    /3.1.10-1.fc14 Thunderbird/3.1.10
Accept: text/html,application/xhtml+xml,
    application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Proxy-Connection: keep-alive
```

Listing 24: Cleartext request to the autoconfig subdomain of the mail provider including email address in GET parameter.

Thunderbird proceeds with the next step if this attempt fails – for example because the autoconfig subdomain does not exist at all.

```
GET http://lavabit.com/.well-known/autoconfig/
    mail/config-v1.1.xml?emailaddress=johnrandom
    %40lavabit.com HTTP/1.1
Host: lavabit.com
User-Agent: Mozilla/5.0 (X11; U; Linux x86_64; en
    -US; rv:1.9.2.17) Gecko/20110428 Fedora
    /3.1.10-1.fc14 Thunderbird/3.1.10
Accept: text/html,application/xhtml+xml,
    application/xml;q=0.9,*/*;q=0.8
```

```
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
```

Listing 25: Cleartext request to mail provider including email address in GET parameter.

The first request actually does not leave the proxy (DNS lookup only) because the hostname autoconfig.lavabit.com does not exist. Besides disclosing the email address in cleartext, the used protocol (plain HTTP) allows an attacker to configure Thunderbird via a man-in-the-middle attack. This could be the malicious exit node adversary telling Thunderbird to use no connection security and plaintext passwords when connecting to the mailserver (IMAP, POP3, SMTP) or using a server controlled by the attacker with connection security. To see if an attack that manipulates the reply actually works, we performed an MITM attack on the reply and replaced the "404 – Not found" with our configuration. If Thunderbird receives an insecure configuration (no connection security), it shows a red warning page, to avoid that warning an attacker needs an SSL certificate for his server – which is not a big effort today anymore. Thunderbird's autoconfiguration mechanism has had a security review by Mozilla but the review does not match the actual implementation. The review states:

> *"To retrieve configuration parameters, Thunderbird will use HTTP on a transport of SSL or on a transport of TLS. Thunderbird will abort the retrieval if the server presents an improper certificate. We chose this protocol over DNS, our choice being more secure than DNS is."* [17]

The mismatch between review and implementation was also mentioned by Brian Smith in a discussion in Mozilla's Bug-tracking System [18].

If the email provider does not offer their settings via autoconfig locations, Thunderbird asks a Mozilla server if there is an entry in the ISPDB for the given email provider. The following request sends the email provider of the user to Mozilla via HTTPS and requests the configuration for it.

```
GET https://live.mozillamessaging.com/autoconfig/
    v1.1/lavabit.com HTTP/1.1
Host: live.mozillamessaging.com
User-Agent: Mozilla/5.0 (X11; U; Linux x86_64; en
    -US; rv:1.9.2.17) Gecko/20110428 Fedora
    /3.1.10-1.fc14 Thunderbird/3.1.10
Accept: text/html,application/xhtml+xml,
    application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
```

Listing 26: ISPDB: HTTPS request to live.mozillamessaging.com containing mail provider domain.

If the provider is not present in Mozilla's ISPDB, Thunderbird asks Mozilla for the MX record.

```
GET https://live.mozillamessaging.com/dns/mx/
    lavabit.com HTTP/1.1
Host: live.mozillamessaging.com
User-Agent: Mozilla/5.0 (X11; U; Linux x86_64; en
    -US; rv:1.9.2.18) Gecko/20110621 Fedora
    /3.1.11-1.fc14 Thunderbird/3.1.11
Accept: text/html,application/xhtml+xml,
    application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
```

Listing 27: ISPDB: HTTPS request to live.mozillamessaging.com requesting the MX record

These requests are sent through the configured proxy.

If all queries fail, Thunderbird starts probing, and during this phase Thunderbird does not respect the proxy settings. After querying the DNS server for several typical

Protocol	Info
DNS	Standard query A mail.lavabit.com
DNS	Standard query AAAA mail.lavabit.com
DNS	Standard query A lavabit.com
DNS	Standard query AAAA lavabit.com
DNS	Standard query A pop3.lavabit.com
DNS	Standard query A pop.lavabit.com
DNS	Standard query AAAA pop.lavabit.com
DNS	Standard query AAAA pop3.lavabit.com
DNS	Standard query A imap.lavabit.com
DNS	Standard query AAAA imap.lavabit.com
DNS	Standard query A smtp.lavabit.com
DNS	Standard query AAAA smtp.lavabit.com

Figure 1: Autoconfiguration probing: leaking DNS requests

hostnames (figure 1), it tries to connect to SMTP, SMTP over SSL, submission, POP3, POP3 over SSL, IMAP and IMAP over SSL ports (figure 2). Some of the sessions are

Source	Destination	Info
192.168.12.10	72.249.41.52	55193 > imap [SYN]
72.249.41.52	192.168.12.10	imap > 55193 [SYN,
192.168.12.10	72.249.41.52	55193 > imap [ACK]
192.168.12.10	72.249.41.52	53451 > imaps [SYN]
72.249.41.52	192.168.12.10	imaps > 53451 [SYN,
192.168.12.10	72.249.41.52	53451 > imaps [ACK]
192.168.12.10	72.249.41.52	55195 > imap [SYN]
72.249.41.52	192.168.12.10	imap > 55195 [SYN,
192.168.12.10	72.249.41.52	55195 > imap [ACK]
192.168.12.10	72.249.41.52	53453 > imaps [SYN]
72.249.41.52	192.168.12.10	imaps > 53453 [SYN,
192.168.12.10	72.249.41.52	53453 > imaps [ACK]
192.168.12.10	72.249.41.52	Client Hello
72.249.41.52	192.168.12.10	imaps > 53451 [ACK]
192.168.12.10	72.249.41.52	Client Hello
72.249.41.52	192.168.12.10	imaps > 53453 [ACK]
192.168.12.10	72.249.41.52	58702 > pop3 [SYN]
72.249.41.52	192.168.12.10	pop3 > 58702 [SYN,
192.168.12.10	72.249.41.52	58702 > pop3 [ACK]
192.168.12.10	72.249.41.52	38870 > pop3s [SYN]
72.249.41.52	192.168.12.10	pop3s > 38870 [SYN,
192.168.12.10	72.249.41.52	38870 > pop3s [ACK]
192.168.12.10	72.249.41.52	Client Hello

Figure 2: Autoconfiguration probing: leaking TCP connections

cleartext, these sessions do not disclose the email address

or the password, they are only used to probe the capabilities of the server. No configuration settings where found to prevent these requests from leaking.

The entire autoconfiguration is insecure due to the missing connection security. Even without the cleartext problem it requires a lot of trust in Mozilla's configuration server. Mozilla – or someone compromising the autoconfiguration server – can redirect users to their own server and collect login credentials and emails.

An alternative method to provide clients with the required configuration uses DNS SRV records. This method is specified in RFC6186. In cases where the login name does not match the email address (or the first part of it), the user needs to enter the login name manually. RFC6186 is not supported by Thunderbird [20], but it would remove the current central point of failure in Thunderbird's autoconfiguration mechanism. In combination with DNSSEC it would be reasonable secure. DNS SRV requests in combination with Tor are a problem.

The current autoconfiguration mechanism can be easily exploited by our malicious exit node adversary. Not using the autoconfiguration mechanism would prevent the leaking requests, but currently there is no option to opt-out. The user can manually set up the account only after the autoconfiguration took place. No configuration option was found to opt-out of autoconfiguration.

3.6.6 Enigmail

The Enigmail extension (v1.1.2) also generates HTTP requests when searching for public keys or when fetching them from keyservers. The keyserver can be chosen by the user. A search using Enigmail's key management interface (search string "LEAKING?") while using pgp.mit.edu as keyserver results in the following cleartext HTTP request.

```
GET /pks/lookup?op=index&options=mr&search=
    LEAKING%3F HTTP/1.1
Host: pgp.mit.edu:11371
Accept: */*
Pragma: no-cache
Cache-Control: no-cache
```

Listing 28: HKP: Cleartext request to pgp.mit.edu:11371 containing search parameter

Usually the search parameter would contain the key ID. Whenever a user chooses to import a missing key the search parameter is automatically completed by Enigmail with the appropriate key ID.

When Thunderbird is configured with HTTP, HTTPS and SOCKS proxies, all network traffic should be handled by them. Enigmail's primary network activity is HTTP/HKP traffic to fetch public keys and therefore it uses the HTTP proxy if configured – entering only the SOCKS proxy is not enough. Usually Enigmail does not leak DNS requests, but when using the "*Refresh All Public Keys*" feature Enigmail leaks a SRV DNS request (see figure 3). The same leak occurs when choosing to publish a public key by uploading it to a keyserver. Enigmail is a graphical user interface that invokes GnuPG commands. Listing 29 shows the gpg command executed when searching for keys, which does not leak DNS requests.

```
DNS     Standard query SRV _pgpkey-http._tcp.pgp.mit.edu
DNS     Standard query response
TCP     45246 > privoxy [SYN] Seq=0 Win=5840 Len=0 MSS=1460
```

Figure 3: Enigmail: "Refresh All Public Keys" - DNS SRV request leak

```
/usr/bin/gpg --charset utf8 --command-fd 0 --no-
    tty --batch --fixed-list --with-colons --
    keyserver-options http-proxy=http
    ://192.168.12.1:8118 --keyserver hkp://pgp.
    mit.edu:11371 --search-keys LEAKING?
```

Listing 29: Enigmail: GPG command - searching for keys (no DNS leak)

The command shown in listing 30 is executed when using the "*Refresh All Public Keys*" feature – this results in a DNS leak.

```
/usr/bin/gpg --charset utf8 --batch --no-tty --
    status-fd 2 --keyserver-options honor-http-
    proxy --keyserver pgp.mit.edu --refresh-keys
```

Listing 30: Enigmail: GPG command - refreshing keys (DNS leak)

One can easily spot the difference in the `--keyserver-options` and `--keyserver` arguments. HKP does not standardize a TCP port for the protocol, port 11371 is just a convention. If a user does not specify the port, the client should query the DNS SRV record for the correct port – according to an HKP draft [15]. The leak occurs because the second gpg invocation does not specify a port – that triggers the DNS SRV request. After contacting the Enigmail Developer, the parameters to the `--keyserver-options` switch were changed in revision 1300 [16] but that did not address the problem – which is associated with the `--keyserver` parameter. An end user can prevent the DNS leak via a configuration change. Changing the keyserver parameter to include the port makes the DNS SRV query obsolete and therefore fixes the DNS leak (figure 4).

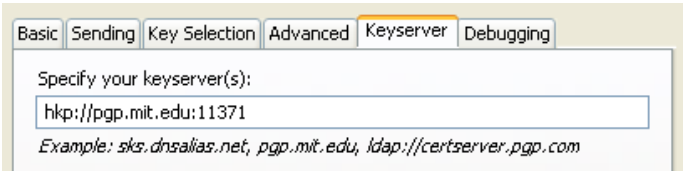

Figure 4: Enigmail: keyserver configuration to prevent DNS SRV leak

Apart from the DNS leak, the "*Refresh All Public Keys*" feature has a more severe side effect. Refreshing the keys means that gpg requests every single key in the keyring from the keyserver via plain HTTP. This means the exit node sees every key ID in the user's keyring, including the key ID(s) of the user's own keys. Depending on how much gpg is actually used, it might reveal the entire contact list to the exit node. The solution is HKP over SSL [64] but HKP servers tend to be plaintext only. Indymedia runs HKP over SSL (untrusted CA cert certificate) and a Tor hidden server (not reachable at the time of this writing) [65].

3.6.7 Online Certificate Status Protocol (OCSP)

When Thunderbird connects to servers using transport security the certificate is verified via the Online Certificate Status Protocol which uses HTTP. OCSP tells the client if a particular certificate is valid. With the content of the OCSP request, the OCSP server (usually operated by a Certificate Authority) knows what host someone might use at the time. A OCSP request does not contain a static identifier for a client. The OCSP server should not be able to link multiple requests to one particular client.

3.6.8 HTTP Strict Transport Security (HSTS)

HTTP Strict Transport Security (HSTS) [48] is a mechanism to prevent certain active MITM attacks [51]. Thunderbird supports HSTS since version 3.3 [49]. The security benefit of HSTS comes not without privacy implications [50]. With HSTS, a webserver can instruct Thunderbird to use HTTPS instead of HTTP for future requests. This information is stored within Thunderbird for a server specified amount of time. A server can misuse HSTS to store an identifying amount of bits using multiple subdomains. We do not communicate over HTTP with our mail provider.

We propose hardcoded HSTS entries for Mozilla's servers. All other entries should be cleared when Thunderbird is closed. We do not expect HTTP traffic to non-Mozilla destinations besides OCSP and HKP requests.

3.6.9 TLS vs. STARTTLS and STLS

There are two different ways to achieve connection security when using SMTP, POP3 or IMAP. Either by using TLS [60] on a specific TCP port or with a protocol command (STARTTLS/STLS). Depending on the used method more or less information disclosure occurs.

The listings 31 and 32 show the information sent and received in cleartext before the secure channel is established when using the protocol commands method (STARTTLS, STLS). The disclosed information is not sensitive because it does not contain client specific data. Listing 33 shows that in case of SMTP it is sensitive because the EHLO argument is disclosed. Commands send by the MUA are shown in bold.

```
* OK lavabit.com IMAP4rev1 lavad server ready.
1 capability
* CAPABILITY IMAP4rev1 STARTTLS LITERAL+
1 OK Completed.
2 STARTTLS
2 OK Begin TLS negotiation now.
<encryption starting>
```

Listing 31: Start of an IMAP session with STARTTLS

```
+OK lavad
CAPA
+OK Capabilities follow.
TOP
USER
UIDL
STLS
PIPELINING
EXPIRE NEVER
LOGIN-DELAY 0
RESP-CODES
AUTH-RESP-CODE
IMPLEMENTATION lavad 4.3.0
.
STLS
+OK Ready to start TLS negotiation.
<encryption starting>
```

Listing 32: Start of a POP3 session with STLS

```
220 lavabit.com ESMTP lavad
EHLO localhost
250-lavabit.com
250-8BITMIME
250-STARTTLS
250-PIPELINING
250-SIZE 1073741824
250-AUTH LOGIN PLAIN
250-AUTH=LOGIN PLAIN
250 Okay.
STARTTLS
220 Ready to start TLS negotiation.
<encryption starting>
```

Listing 33: Start of an SMTP session with STARTTLS (EHLO argument disclosure)

Not every email provider offers both possibilities to its users. We prefer TLS to STARTTLS and STLS, but MSA's on TCP port 587 do not provide TLS (related to section 4.2.3).

An MITM attacker could also remove the STARTTLS and STLS capabilities from the server response, but Thunderbird does not proceed if the server does not offer these capabilities if configured to do so.

4 Attack and Defense

4.1 Attacks

Most of the described application level privacy leaks occur without active involvement of an adversary. This section describes methods to gather information that involve an active adversary.

4.1.1 Message Delivery Notification (MDN)

Message delivery notification (RFC3798 [62]) is a (semi) automated way to ask the receiver of a message for a delivery notification. Thunderbird calls this feature "return receipt".

Besides the fact that a MDN discloses the fact that the email was read and when, there are some more severe disclosures. The MDN reply (that is automatically generated if the user accepts it) contains:

- System specific language in the subject
- A *Reporting-UA* header. This is the same string as the *User-Agent* header field (see section 3.5.2)
- The entire header of the received email. This includes headers added locally after the email was received. Listing 34 shows a few of them.

```
X-Account-Key: account3
X-UIDL: GmailId3348fc8ffe3facce
X-Mozilla-Keys:
[...]
```

Listing 34: Mozilla specific local header fields disclosed to the requester of an MDN.

Using the header fields shown in listing 34 the attacker can determine:

- the user has multiple accounts configured (at least 3)
- the user downloads mail using POP3

The rest of the header (not shown in listing 34) discloses the path the email took, this might disclose email aliases.

We do not propose a solution, we recommend to silently ignore MDNs.

4.1.2 POP3: Counting Clients via UIDL

The UIDL-feature (described in section 3.4.5) can be used by the email service provider to detect if multiple clients are retrieving emails from a single mailbox. If multiple POP3 clients are configured to fetch mails from the same mailbox and every client leaves the mails on the server, then the server can detect how many clients are configured to fetch emails from a given mailbox by counting the times a specific mail – identified by its UIDL - is downloaded. A given client would not fetch a certain email more than ones.

This reveals the fact that the user has multiple configured clients and it might also be useful for an adversary to determine if an email was written from location/client A or B. The locations are not revealed, but the existence of multiple clients might be already useful for an adversary trying to gather information about a certain account holder – especially if a group of people is using a certain account as a shared account.

4.1.3 POP3: Confirming Access via LOGIN-DELAY

One of the extensions specified in RFC2449 are the POP3 capabilities LOGIN-DELAY and RESP-CODES. The LOGIN-DELAY capability gives the POP3 server a defined method to tell the client the shortest allowed login interval. This feature should give the server the possibility to reduce the load caused by clients asking for new emails with a high frequency. The information is transported to the client in response to the CAPA command. A client uses the CAPA command to ask the server for the supported capabilities/extensions. It is up to the server if the delay is enforced or not. POP3 without any extensions has only a very basic response mechanism. If a command succeeds, the server responds with "+OK". If it fails, the response is "-ERR". The capability RESP-CODES extends the response mechanism to include more information in the response code. One of the extended response codes is "LOGIN-DELAY" to indicate that a client should come back later because his login interval is to short. There are two commands where the server may respond with that response code: USER and PASS. Should the server implementation issue the response code in response to the USER command (which is strongly discouraged) then this results in a privacy issue. If all of the following conditions are met, an arbitrary attacker can periodically query the POP3 server to determine if a given user checked his inbox within the last X seconds (X is defined by LOGIN-DELAY returned in response to the CAPA command):

- the POP3 server supports the LOGIN-DELAY and RESP-CODES capabilities
- the server imposes a LOGIN-DELAY greater 0
- the server enforces the LOGIN-DELAY
- the server returns the LOGIN-DELAY response code in response to the USER command rather than to the successful PASS command (resp. code is sent to the unauth. user)

Gmail supports LOGIN-DELAY and RESP-CODES but does not enforce the 300 seconds delay.

```
+OK Gpop ready for requests from 72.46.129.46
    n27pf1777777faa.95
CAPA
+OK Capability list follows
RESP-CODES
LOGIN-DELAY 300
[...]
```

Listing 35: Gmail POP3 server capabilities

It is improbable to see a server where all preconditions are met. The "IN-USE" response code is not an issue because it is only returned after a successful authentication.

4.1.4 "X-Mozilla-External-Attachment-URL" Attack

As discussed in section 3.5.9, Mozilla introduced a few non-standardized mail header fields for Thunderbird operations. If a user chooses to detach an attachment from an existing email, the attachment is no longer stored in the mailbox and it is replaced with a reference. The reference is represented with a *X-Mozilla-External-Attachment-URL* header field containing the path to the file. In case of IMAP this results in the disclosure of local paths - which usually contains local usernames - to the IMAP server. This reference is similar to the standardized MIME content-type *message/external-body* (section 3.4.3).

To explore the handling of the *X-Mozilla-External-Attachment-URL* header field we experimented with a few test mails to see if this header is also parsed if it is received in an email. Usually this header is inserted by Thunderbird itself after the email was received.

Our goal was to cause a request to a remote location without the user's explicit consent. This would disclose the fact that the email was read and when. Listing 36 shows our constructed mail.

```
Date: Mon, 13 Jun 2011 19:46:30 +0000
From: user@example.com
MIME-Version: 1.0
To: user@example.com
Subject: PoC1
Content-Transfer-Encoding: 8bit
Content-Type: multipart/mixed;
 boundary="------------12347"

--------------12347
Content-Type: image/jpg; name="tb.png"
X-Mozilla-External-Attachment-URL: http://upload.
    wikimedia.org/wikipedia/en/f/f7/
    Mozilla_Thunderbird_logo.png
Content-Disposition: attachment; filename="tb.png
    "
```

```
--------------12347--
```

Listing 36: Email with *X-Mozilla-External-Attachment-URL* attachment

Thunderbird will display it as usual attachment in the mail view (figure 5) even though it contains a reference only. The user has no means to see the difference without looking at the mail source. If the user chooses to save it, the file is

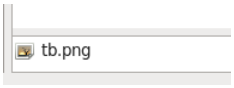

Figure 5: *X-Mozilla-External-Attachment-URL* attachment (1)

downloaded before the user confirms the dialog shown in figure 6. The dialog clearly shows the remote location, but at that point in time the request reached already the remote server. The file is fetched by Thunderbird and via the con-

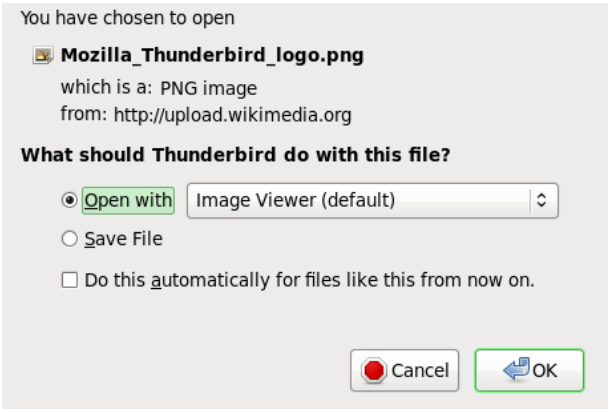

Figure 6: *X-Mozilla-External-Attachment-URL* attachment (2)

figured proxy – no leaking requests.

There is a special case that has far more serious consequences. If the webserver replies with a HTTP redirect, Thunderbird starts automatically the default browser with the new location without even showing the dialog box (figure 6). This effectively deanonymizes the user - the browser fetches the new location without using the Tor. This behavior is independent of content-type and action performed by the user ("Save as..." or "Open") and works across platforms. This issue was reported to Mozilla in June 2011.

4.1.5 Detecting Enigmail

Enigmail automatically parses mails when they are selected. PGP-inline header are automatically removed when the message is displayed. When replying to a signed message the PGP header of the original message is automatically removed and is not quoted. This behavior can be used to determine if a user has Enigmail installed (explicitly enabling OpenPGP is not required). Before performing the following procedure the adversary has already determined that the user uses Thunderbird.

1. The adversary sends a PGP-inline signed message to the user
   - If the reply does not contain any of his PGP lines this might indicate the presence of Enigmail. To further test the adversary might proceed with step 2.
   - If the reply does contain PGP lines step 2 is not necessary
2. The adversary sends a second PGP-inline message but introduces an error in the header:

   `-----BEGIN PGP SIGNED MESSAGE-----`
   becomes:
   `-----BEGIN PGP SIGNED MESAGE-----`

   If the reply does contain the PGP parts, this is a second indication for the presence of Enigmail, but these lines are also present if Enigmail is not installed. This is only useful in combination with the fist test.

These indicators are not reliable. A user can remove headers manually. It is easier to detect the absence of Enigmail, because it is unlikely that a user adds the PGP lines manually.

4.1.6 Thunderbird Version Detection

As specified in section 2.3 the header information reduction does not aim to hide the fact that Thunderbird is the used MUA. Someone able to detect Thunderbird's approximate version can use it to reduce the anonymity set of a user. We believe that after taking all mentioned steps to prevent information disclosure discussed in this document there are limited possibilities to determine Thunderbirds exact or approximate version by inspecting mail header at the receiver side. Another much more powerful way to determine Thunderbirds version is by inspecting and probing its protocol behavior. What is the standard sequence of commands in a POP3 session? What are the used authentication schemes? What protocol extensions are used? How does Thunderbird react to protocol misbehavior? By wading trough Thunderbirds changelogs one might find potential candidates that are detectable by the server. An example for Thunderbird version 3 beta 3:

- *"Fixed: 476260 - support for IMAP XLIST"* [47]
- *"Fixed: 401673 - Add support for IMAP COMPRESS Extension – rfc4978"* [47]

These type of detection mechanisms are not limited to the mail provider. The communication partner - that is not able to directly inspect IMAP, POP3 or SMTP sessions of the user - might be able to detect an approximate Thunderbird version too. This could be the case if Thunderbird adds support for a new MIME content types. We believe that there will always remain a possibility for version detection at this level, but using POP3 instead of IMAP makes it harder for an attacker (IMAP is more frequently extended).

4.2 Defense

4.2.1 Prevent the Execution of External Applications

External applications are unsafe because their proxy settings are not related to Thunderbird's proxy settings. If an

external application causes traffic it probably will not be routed through the Tor network. Traffic that is not routed through Tor will deanonymize the user. To prevent the automatic start of the browser for http and https URLs the following preferences should be set.
`network.protocol-handler.warn-external.http`
`network.protocol-handler.warn-external.https`
This will intercept the call to the browser and the user must confirm before proceeding, but the problem is not limited to http and https URLs. Every attachment that is opened with an external program has the same potential issue. The dialogbox to open files should be replaced with a warning like Torbutton displays when trying to open a file.

4.2.2 Avoid Linkability of Multiple Pseudonyms through the Tor Circuit

Creating Tor circuits is resource intensive and introduces latency. To reduce resource requirements and latency a Tor circuit transports multiple TCP connections (streams). When using multiple pseudonyms with a single Tor instance, a Tor circuit might become the linking element because traffic for both pseudonyms may be routed through the same Tor circuit. This allows linking of pseudonyms by the exit node. The exit node is not able to see the used email address (we assume transport security) but he might collaborate with the email provider – or the email provider is the exit node relay. This would allow the exit node to link multiple pseudonyms. To prevent the linking of multiple pseudonyms (or the linkage of a pseudonym with a real name) through the used Tor circuit, we propose to use a separate Tor client instance per pseudonym/email account.

The Tor proposal 171 [21] was written to address stream separation in future Tor versions (0.2.3.x). Thunderbird does not offer the possibility to set proxies per email account. To separate the TCP connections of different accounts, multiple Thunderbird instances (profiles) are required. Every Thunderbird connects to a different Tor instance. This should ensure separation.

Note: Separating traffic of different mail accounts to different circuits via multiple Tor instances does not ensure that different exit nodes are used. Both Tor instances may chose the same exit node, but the traffic of two different accounts will not leave same exit node through the same circuit. The exit node can not use the circuit as the linking element any more.

After the implementation of Tor proposal 171, one Tor instance will be enough to achieve the separation of streams. One Tor instance will provide multiple ports to connect to and every Thunderbird instance will connect to a different port. To avoid multiple Thunderbird instances one could modify Thunderbird to support proxy settings per account. Also a receiver could link multiple accounts based on used exit node.

4.2.3 Prevent time-offset based Fingerprinting

Date and *Message-ID* header fields disclose the machine specific time (see section 3.5.3). The offset of the machine specific time to the mail servers time can be used as fingerprinting input. The mail servers time is assumed to be more or less constant. It is not required to be correct. There are a few options to tackle this issue:

1. assume the user has already a reasonable accurate time which is unlikely to be useful for fingerprinting (NTP)
2. introduce randomness or reduce the granularity of the timestamp
3. remove the leaking header fields and submit an "incomplete" mail header to a message submission agent (MSA) for completion and forwarding.

Note: Tor will not start if the system time is out of certain boundaries, but we do not assume that Tor is running locally in every case. Making less assumptions will result in a setting that is applicable in more situations, hence we drop option 1. The second option would only help in scenarios where the user has a reasonable correct time, but users with a huge offsets are still at risk. We chose option 3. This option needs some introduction regarding message submission agents. In contrast to MTAs MSAs provide functionality only to MUAs but do also use the SMTP protocol to accept messages. Message submission for mail is specified in RFC4409 [7]. MSAs require authentication before accepting mail. They usually listen on port 587 but may also listen on port 25. RFC4409 mentions so called "incomplete" messages. MSAs must complete or reject incomplete messages before forwarding them to MTAs:

> *"In addition to authentication and authorization issues, messages being submitted are in some cases finished (complete) messages, and in other cases are unfinished (incomplete) in one or more aspects. Unfinished messages may need to be completed to ensure they conform to [RFC2822], and later requirements. For example, the message may lack a proper 'Date' header field, and domains might not be fully qualified. In some cases, the MUA may be unable to generate finished messages (e.g., it might not know its time zone)."* [7]

and further:

> *"Any message forwarded or delivered by the MSA MUST conform to the requirements of RFC821 and RFC822/RFC2822."* [7]

The latter gives a certain confidence that sending incomplete messages to an MSA will not introduce incomplete messages into the MTA relay network. The first MTA will receive a message that is compliant to RFC2822/RFC5322. The RFC is mentioning exactly our use case *"In some cases, the MUA may be unable to generate finished messages (e.g., it might not know its time zone)."* [7].

Thunderbird does not offer an option to send messages without *Date* and *Message-ID* header fields. To see the real world handling of "incomplete" messages we modified Thunderbird to omit the *Date* and *Message-ID* header fields

and inspected the message at the receiver. The test was performed with Gmail's MSA on TCP port 587. The recipient received the mail shown in listing 37.

```
[...]
Received: from localhost ([180.149.96.69])
        by mx.example.com with ESMTPS id 74
            sm1777777eet.19.2011.06.23.02.35.29
        (version=TLSv1/SSLv3 cipher=OTHER);
        Thu, 23 Jun 2011 02:35:36 -0700 (PDT)
Message-ID:
    <4e0308e8.e2090e0a.7b21.25d7@mx.example.com>
Date:  Thu, 23 Jun 2011 02:35:36 -0700 (PDT)
From: user@example.com
MIME-Version: 1.0
To: user@example.com
Subject: M: ua,ha,ce,mid,date / F14
Content-Type: text/plain; charset=UTF-8; format=
    flowed
Content-Transfer-Encoding: 7bit

body line 1
```

Listing 37: Mail with server side generated Date and Message-ID header fields

The lines in bold were inserted by the email provider, hence do not contain anything client specific. The time in the *Date* header field matches the time in the *Received* header field because both lines were generated on the same system. A Thunderbird modified in such a way does not disclose machine specific time information in *Date* and *Message-ID* header fields any more. The receiver still receives RFC5322 compliant messages. By changing Thunderbird's *Message-ID* format one could avoid the time information in that header field without omitting it. The same test was repeated with other service ports to see how realistic this approach would be:

- gmail.com (port: 25)
- gmail.com (port: 465)
- lavabit.com (port: 587)
- lavabit.com (port: 25)
- lavabit.com (port: 465)

The results were the same – header fields were inserted. It is likely that an operator is running the same software behind all client facing ports anyway. This approach is experimental.

Note: It is not recommended to send emails to a destination port 25 over Tor because the few Tor exit nodes that allow exiting on that port are likely blacklisted.

Thunderbird uses the *Message-ID* also internally. Entirely removing the *Message-ID* is likely harmful in someway for Thunderbird's internal operations. Changing the format (removing the timestamp information) is therefore necessary. We would propose a *Message-ID* that is based on a hash of the email and generated as follows:

1. perform all steps that lead to a message ready for submission (attachment encoding, S/MIME or PGP signing/encrypting, ...)
2. remove all *Message-ID* and *Date* header fields from the mail header (if present)
3. arrange the header fields (received from step 2) in alphabetical order and append the email body (with the usual empty line between header and body)
4. calculate the hash of the entire email (taken from step 3)
   mail-hash = sha512(email)
5. generate a random string with a fixed length of 8 characters (character set: a-z0-9)
6. build the message-id
   message-id = mail-hash|random-string@domain (domain is the last part of the sender address)
7. insert the *Message-ID* as the last line of the header and submit the email to the mail server (without any *Date* header field)

Even after these steps were taken to prevent a system time disclosure via the email header other vectors still leak the same information. When negotiating a secure connection the client submits a timestamp. When signing mails a timestamp is included. When validating certificates the client needs to have a roughly correct time to detect expired certificates.

We believe a complete time independence can not be achieved, but the above method at least reduces the information disclosure in the mail header and can be useful in case of unsigned mails.

4.2.4 Establishing a Common Fingerprint

Establishing a common fingerprint has two goals:

- keep the anonymity set as big as possible
- make linking of multiple pseudonyms harder

while avoiding to reduce or restrict functionality or lose flexibility. A certain reduction of flexibility is unavoidable. To prevent linkability caused by unusual configurations one part of the defense strategy is to establish a common fingerprint. This is a common technique to build large anonymity sets and prevent unique fingerprints [71]. A common fingerprint makes linkability actually easier if there are only a few user having it. We acknowledge that the introduction of a new fingerprint results in a more unique fingerprint than faking an existing, but we aim for a fingerprint that should not change with new product versions or even across different products. The resulting fingerprint should remain static and the anonymity set will likely grow with its age.

We propose the following settings to Thunderbird and Enigmail to establish a common fingerprint:

- always use SSL/TLS if available (instead of STARTTLS)
- disallow cookies
  `network.cookie.cookieBehavior = 2`
- EHLO SMTP argument
  `mail.smtpserver.default.hello_argument = [127.0.0.1]`

- DNS via SOCKS
  `network.proxy.socks_remote_dns = true`
- no *User-Agent* header field
  `general.useragent.override =`
- send email in text/plain content-type
  `...compose_html = false`
  (There is one such setting per account.)
- charset UTF-8
  `mailnews.send_default_charset = UTF-8`
- do not send flowed text
  `mailnews.send_plaintext_flowed = false`
- default wraplength
  `mailnews.wraplength = 72`
- reduced reply header
  `mailnews.reply_header_type = 1`
  `mailnews.reply_header_authorwrote = %s`
- do not load the start page
  `mailnews.start_page.enabled = false`
- Enigmail/GPG specifics: (section 3.5.7)
- PGP-inline (no PGP/MIME)
- When sending encrypted emails with attachments: "Encrypt/sign each attachment separately and send the message using inline PGP" (second option)
- no *X-Enigmail* header field
  `extensions.enigmail.addHeaders = false`
- no *OpenPGP* header field
- no *Comment* header field
  `extensions.enigmail.useDefaultComment = true`
- hash algorithm SHA512
  `extensions.enigmail.mimeHashAlgorithm = 5`
- no GPG version disclosure
  `extensions.enigmail.agentAdditionalParam = --no-emit-version`
- HKPS server
- RFC5322 does not specify any requirements regarding the order of header fields, nevertheless a common fingerprint should have a recommended header field order to prevent fingerprinting based on header field order (arrange alphabetically - see section 4.2.3)
- *Date* header field: Until the proposed solution in section 4.2.3 is reviewed and extensively tested on a broad range of freemail services we assume NTP is in place.
- *Message-ID* based on a SHA512 hash (4.2.3)

4.2.5 Defense in Depth

Software contains errors. As such one can not assume that Thunderbird will never fail. Defense in depth tries to reduce the impact of a partial failure of the system. In our setting this could mean, that the user is still safe although Thunderbird is leaking requests. This can be achieved by transparently routing traffic into Tor. The routing decision is no longer dependent on the correct functioning of the application. The downside of re-routing without specific knowledge about content is the problem discussed in section 4.2.2. A router might be unable to determine if two streams should be separated or not. For this reason the transparent re-routing should be used as an additional safeguard but not as a replacement for application proxy settings. Tails [37] is a live CD that ensures traffic is routed through Tor to prevent denanymization via leaking traffic. An even higher failure tolerance could be achieved if the routing decision is enforced by a separate host.

5 Conclusion

We analyzed Thunderbird and Enigmail and found a series of privacy and deanonymization issues including security issues which are also relevant for non-Tor user. Our results might also be useful to ordinary Thunderbird users to increase the efforts required for profiling and targeted attacks. The currently available configuration guidelines for Mozilla Thunderbird in combination with Tor (section 1.6.3) are not safe. Many identified problems are not directly related to the sending and receiving of mail.

Some of the problems identified in the course of our analysis:

- Thunderbird discloses machine specific clock information (section 3.5.3 and 3.5.4)
- Thunderbird may discloses the underlying OS (section 3.5.6)
- Thunderbird requests a website in cleartext at startup, disclosing its version, OS, language and buildid (section 3.6.1)
- Thunderbird stores and submits an identifying HTTP cookie with a lifetime of five years to Mozilla (section 3.6.4)
- Thunderbird discloses the email address in cleartext to the exit node and the network (section 3.6.5)
- Thunderbird retrieves mail server configurations from a remote server over an insecure channel (section 3.6.5)
- Thunderbird leaks DNS requests and TCP connections (section 3.6.5)
- Enigmail leaks DNS requests (section 3.6.6)
- Enigmail may disclose all keyIDs in the keyring to the exit node and the network (section 3.6.6)
- Thunderbird may disclose local paths to the IMAP server (section 4.1.4)

- Thunderbird insecurely handles certain mail header and opens the browser without the user's consent (section 4.1.4)
- an attacker might be able to determine Thunderbird's approximate version (section 4.1.6)

(The list does not include problems that are already solved with currently available configuration steps - section 1.6.3).

Based on our findings we provided configuration settings for easy solutions (3.5, 3.6) and proposed changes for more complex issues (4.2). Some of the issues are software bugs within Thunderbird, which eventually will be addressed by Mozilla (3.6.5, 4.1.4). We do not propose solutions to all problems, either because they are hard to prevent and are low risk (4.1.6) or because it is a protocol/server problem that can not be solved by the client (4.1.3). At some point we accept minor information disclosure for a major security benefit (3.6.2). We deem it acceptable because the benefits outweigh the risks.

Most identified problems are solved easily with configuration changes, only a few are not solvable with configuration changes (3.5.6, 3.6.5, 3.6.8, 4.1.4, 4.2.1, 4.2.3) and need code changes. We propose the following modifications to Thunderbird to solve these problems:

Modifications

- remove possibility to load website at startup including "whatsnew" page (3.6.1)
- replace autoconfiguration with manual configuration (3.6.5)
- boundary string without prefix in any case (3.5.6)
- stripe Mozilla specific header from incoming mails (4.1.4)
- intercept any call to open an external application (4.2.1)
- implement per account proxy settings including a global proxy for non-mail traffic (4.2.2)
- implement time-offset fingerprint prevention (4.2.3)

Configuration options

All options are enabled by default.

- "make 'connection security' mandatory"
- "use common fingerprint" (4.2.4)
- "HSTS state reset on shutdown"
- "ignore MDN requests silently"
- "disable HTML parsing"
- "do not display attachments inline"
- "make URLs in emails not clickable"
- "display warning when setting up an IMAP account"
- "display warning if multiple accounts share the same proxy"
- "display warning when adding attachment"

Location privacy and pseudonymity can be achieved when sending and receiving emails through Tor, however, Mozilla Thunderbird is not safe yet to be used with Tor. We believe that our results will help to reduce application-level privacy leaks in Thunderbird and eventually provide end users with better privacy.

6 Future Work

Our proposed solutions should be extensively tested and scrutinized including operating systems that were not available for our tests. Specific parts of Thunderbird require a review at source code level.

POP3 and SMTP PIPELINING [6, 68], deployed by the majority of freemailers, but not implemented by Thunderbird, would improve performance and user experience especially for Tor users.

The use of a single pseudonym in combination with a high number of communication partners is risky (2.2.1). The use of a high number of pseudonyms might not be manageable. The security and privacy properties of state of the art remailers [24] are appealing. Remailers might make multiple pseudonyms obsolete. The usability of remailers needs to be improved.

Anonymously signing up for an email account is becoming harder. Gmail started to require SMS verification in many cases.

Users might hesitate to use Tor to send their emails, fearing their emails might get marked as spam. To what degree are emails send via Tor more susceptible to be marked as spam? Does our proposed fingerprint influence the outcome of spam detection algorithms? Does DKIM [69] reliable prevent emails from being marked as spam?
How could Tor's circuit creation automatically avoid exit nodes that are listed in RBLs?
What are the implications of using S/MIME and OpenPGP with Tor?
Which of our proposed settings (common fingerprint) could be pushed to the upstream Thunderbird?
These questions should be addressed to determine if email submission through Tor is as reliable as without using Tor and to bring privacy to a broader user-base.

References

[1] J. Klensin, "Simple Mail Transfer Protocol", RFC5321, October 2008.
[2] P. Resnick, "Internet Message Format", RFC5322, October 2008.
[3] N. Freed and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC2045, November 1996.
[4] M. Crispin, "INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1", RFC3501, March 2003.
[5] J. Myers and M. Rose, "Post Office Protocol - Version 3", RFC1939, May 1996.
[6] R. Gellens, C. Newman and L. Lundblade, "POP3 Extension Mechanism", RFC2449, November 1998.
[7] R. Gellens and J. Klensin, "Message Submission for Mail", RFC4409, April 2006.
[8] A. Pfitzmann and M. Hansen, "A terminology for talking about privacy by data minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management", August 2010, <http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf>.
[9] M. Elkins, D. Del Torto, R. Levien and T. Roessler, "MIME Security with OpenPGP", RFC3156, August 2001.
[10] P. Faeltstroem, "MacMIME - How to send Macintosh files with MIME", March 1993, <http://www.iana.org/assignments/media-types/multipart/appledouble>.
[11] Mozilla Foundation, "Mozilla Thunderbird v5.0", June 2011, <ftp://ftp.mozilla.org/pub/thunderbird/releases/5.0/source/thunderbird-5.0.source.tar.bz2>.
[12] P. Brunschwig, "Enigmail v1.1.2", June 2010, <http://www.mozilla-enigmail.org/download/source/enigmail-1.1.2.tar.gz>.
[13] "anonym", "doc/TorifyHOWTO/EMail", May 2011, <https://trac.torproject.org/projects/tor/wiki/doc/TorifyHOWTO/EMail?action=diff&version=34&old_version=33>.
[14] "anonym", "Tails - Icedove implementation", May 2011, <http://tails.boum.org/forum/Icedove_implementation/#comment-db2c1b0c2e1ebf869a8d89c3ca976199>.
[15] D. Shaw, "The OpenPGP HTTP Keyserver Protocol (HKP)", draft-shaw-openpgp-hkp-00, March 2003.
[16] P. Brunschwig, "enigmail.js rev1.300", June 2011, <http://www.mozdev.org/source/browse/enigmail/src/package/enigmail.js#rev1.300>.
[17] Mozilla Foundation, "Security review FetchConfigFromISP", June 2009, <https://wiki.mozilla.org/Thunderbird:Autoconfiguration:Security_review_FetchConfigFromISP>.
[18] Mozilla Foundation, "Improve privacy and security of Thunderbird account auto-configuration", July 2011, <https://bugzilla.mozilla.org/show_bug.cgi?id=664633>.
[19] C. Daboo, "Use of SRV Records for Locating Email Submission/Access Services", RFC6186, March 2011.
[20] Mozilla Foundation, "mail account auto-configuration via DNS SRV", March 2011, <https://bugzilla.mozilla.org/show_bug.cgi?id=342242>.
[21] R. Hogan, J. Appelbaum, D. McCoy and N. Mathewson, "proposal 171 - separate streams", December 2010, <https://gitweb.torproject.org/torspec.git/blob/HEAD:/proposals/171-separate-streams.txt>.
[22] D. Chaum, "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms", February 1981, <http://www.freehaven.net/anonbib/cache/chaum-mix.pdf>.
[23] U. Moeller, L. Cottrell, P. Palfrader and L. Sassaman, "Mixmaster Protocol Version 2", July 2003, <http://www.abditum.com/mixmaster-spec.txt>.
[24] G. Danezis, R. Dingledine and N. Mathewson. "Mixminion: Design of a Type III Anonymous Remailer Protocol", May 2003, <http://mixminion.net/minion-design.pdf>.
[25] R. Dingledine, "Re: Status?", May 2010, <http://archives.seul.org/mixminion/dev/May-2010/msg00001.html>.
[26] S. Parekh, "Prospects for Remailers", August 1996, <http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/476/397>.
[27] A. Engelfriet, "Creating encrypted reply blocks", January 1998, <http://www.iusmentis.com/technology/remailers/reply-blocks.html>.
[28] U. Moeller, J. Jagars, P. Palfrader, L. Sassaman, C. Tuckley and S. Crook, "Mixmaster v3.0", March 2008, <http://sourceforge.net/projects/mixmaster/files/Mixmaster/3.0/mixmaster-3.0.tar.gz/download>.
[29] L. Sassaman, B. Cohen and N. Mathewson, "The Pynchon Gate: A Secure Method of Pseudonymous Mail Retrieval", 2005, <http://www.abditum.com/pynchon/sassaman-wpes2005.pdf>.
[30] M. Perry, "To Toggle, or not to Toggle: The End of Torbutton", April 2011, <https://lists.torproject.org/pipermail/tor-talk/2011-April/020077.html>.
[31] M. Perry, "To Toggle, or not to Toggle: The End of Torbutton", May 2011, <https://blog.torproject.org/blog/toggle-or-not-toggle-end-torbutton>.
[32] M. Perry, "Torbutton 1.4.0 released", July 2011, <https://lists.torproject.org/pipermail/tor-talk/2011-July/020774.html>.
[33] The Tor Project, "Tor Project: Torbutton", July 2011, <https://www.torproject.org/torbutton/>.
[34] The Tor Project, "Tor Project: Anonymity Online", July 2011, <https://www.torproject.org/>.
[35] The JAP-Team and JonDos GmbH, "JonDonym Help: Anonymous email accounts", (n.d.), <http://anonymous-proxy-servers.

net/en/help/thunderbird.html>.
[36] The JAP-Team, "JAP", 2006, <http://anon.inf.tu-dresden.de/index_en.html>.
[37] Tails, "Tails - Privacy for anyone anywhere", July 2011, <http://tails.boum.org/>.
[38] L. Mora, "SMTP Information gathering", 2007, <http://www.blackhat.com/presentations/bh-europe-07/Mora/Whitepaper/bh-eu-07-mora-WP.pdf>.
[39] V. Carvalho, R. Balasubramanyan and W. Cohen, "Information Leaks and Suggestions: A Case Study using Mozilla Thunderbird", 2009, <http://www.cs.cmu.edu/~vitor/papers/ceas09.pdf>.
[40] C. Eyrich. "X-Mozilla-Status explained", December 2010, <http://www.eyrich-net.org/mozilla/X-Mozilla-Status.html>.
[41] IANA. Access Types, April 2010, <http://www.iana.org/assignments/access-types/access-types.xml>.
[42] Mozilla Foundation, "support HTTP proxy connect feature for proxying mail/news protocols", March 2011, <https://bugzilla.mozilla.org/show_bug.cgi?id=264981>.
[43] P. Chairunnanda, N. Pham and U. Hengartner, "Privacy: Gone with the Typing!", 2011, <http://petsymposium.org/2011/papers/hotpets11-final8Chairunnanda.pdf>.
[44] B. Schneier, "Crypto-Gram Newsletter", June 2011 <https://www.schneier.com/crypto-gram.html>.
[45] IANA, "MIME Media Types", March 2007, <http://www.iana.org/assignments/media-types/index.html>.
[46] Mozilla Foundation, "Autoconfiguration in Thunderbird", May 2011, <https://developer.mozilla.org/en/Thunderbird/Autoconfiguration>.
[47] G. Kwong, "Thunderbird 3 Beta 3 Released", September 2009, <http://www.rumblingedge.com/2009/07/21/thunderbird-3-beta-3-released/>.
[48] J. Hodges, C. Jackson, A. Barth, "HTTP Strict Transport Security (HSTS)", draft-ietf-websec-strict-transport-sec-01, March 2011.
[49] Mozilla Foundation, "HTTP Strict Transport Security", July 2007, <https://developer.mozilla.org/en/Security/HTTP_Strict_Transport_Security>.
[50] M. Davidov, "The Double-Edged Sword of HSTS Persistence and Privacy", March 2011, <http://haxsys.net/2011/04/the-double-edged-sword-of-hsts-persistence-and-privacy/>.
[51] M. Marlinspike, "sslstrip", May 2011, <http://www.thoughtcrime.org/software/sslstrip/>.
[52] J. Appelbaum, "Detecting Certificate Authority compromises and web browser collusion", March 2011, <https://blog.torproject.org/blog/detecting-certificate-authority-compromises-and-web-browser-collusion>.
[53] P. Hoffman and J. Schlyter, "Using Secure DNS to Associate Certificates with Domain Names For TLS", draft-ietf-dane-protocol-08, July 2011.
[54] R. Dingledine, N. Mathewson and P. Syverson, "Tor: The Second-Generation Onion Router", August 2004, <https://svn.torproject.org/svn/projects/design-paper/tor-design.pdf>.
[55] G. Wondracek, T. Holz, E. Kirda, C. Kruegel, "A Practical Attack to De-Anonymize Social Network Users", May 2010, <http://www.iseclab.org/papers/sonda-TR.pdf>.
[56] A. Narayanan and V. Shmatikov, "De-anonymizing Social Networks", March 2009, <http://www.cs.utexas.edu/~shmat/shmat_oak09.pdf>.
[57] C. Soghoian and S. Stamm, "Certified Lies: Detecting and Defeating Government Interception Attacks Against SSL", March 2011, <http://files.cloudprivacy.net/ssl-mitm.pdf>.
[58] M. Marlinspike, "SSL And The Future Of Authenticity", April 2011, <http://blog.thoughtcrime.org/ssl-and-the-future-of-authenticity>.
[59] MozillaZine, "Thunderbird 3.0 - New Features and Changes", June 2011, <http://kb.mozillazine.org/Thunderbird_3.0_-_New_Features_and_Changes#Forwarding_and_Message_Attachments>.
[60] T. Dierks and C. Allen, "The TLS Protocol Version 1.0", RFC2246, January 1999.
[61] EFF, "How Tor works", November 2011, <https://media.torproject.org/image/webimages/htw2.png>.
[62] T. Schweyer, "I2P Anonymous Network", June 2011, <http://www.i2p2.de/>.
[63] T. Hansen and G. Vaudreuil, "Message Disposition Notification", RFC3798, May 2004.
[64] D. Shaw, "HKP keyservers over SSL", March 2009, <http://lists.gnupg.org/pipermail/gnupg-devel/2009-March/024861.html>.
[65] The Independent Media Center, "Indymedia Keyservers", (n.d.), <http://keys.indymedia.org/>.
[66] E. Hughes, "A Cypherpunk's Manifesto", March 1993, <https://w2.eff.org/Privacy/Crypto/Crypto_misc/cypherpunk.manifesto>.
[67] L. Cottrell, "Are you Vulnerable?", June 2010, <http://www.youtube.com/watch?v=JOJVbTxosIk>.
[68] N. Freed, "SMTP Service Extension for Command Pipelining", RFC2920, September 2000.
[69] E. Allman, J. Callas, M. Delany, M. Libbey, J. Fenton and M. Thomas, "DomainKeys Identified Mail (DKIM) Signatures", RFC4871, May 2007.
[70] Mozilla Foundation, "Blocklisting", April 2011, <https://wiki.mozilla.org/Blocklisting>.
[71] P. Eckersley, "How Unique Is Your Web Browser?", 2010, <https://panopticlick.eff.org/browser-uniqueness.pdf>.