e-mail via .onion addresses

Johannes Berg <johannes@{sipsolutions.net,mtx4fufynowi6kvp.onion}>

February 17, 2006

Abstract

Tor allows hidden services that look like DNS names. In this paper, we address problems with using such hidden services for the exchange of electronic mail, and show that this is indeed possible while maintaining a maximum of user convenience.

1 Introduction

Tor (The Onion Router) allows for hidden services, names of the form *some-id.onion*, that are stable over time, cannot be faked, yet are location-hidden in the sense that only the provider knows where the service is located, and the service can move between physical hosts without major disruption.

This suggests that it should be possible to form e-mail addresses with .onion 'domains', of the form *localpart@some-id.onion*. This paper sets out to explore implementation details of such email addresses, and we show that it is indeed feasible to do with current infrastructure, if configured correctly.

Note that in the following we may use example names of the form *example.onion* (which obviously isn't a valid *.onion* name), and that everything here is applicable to *.tor* names as well.

This can be used to have email addresses permanently linked to a person, but only that person itself knows that she owns the email address. Other people can only 'get to know' the person via the email address handle, but due to the hidden service model in Tor it is guaranteed that the handle cannot be stolen.

2 Problems

Current email infrastructure (usually) revolves around a model where clients give all their outgoing email to a single mail server (the smarthost) which delivers it to them. Our aim is to keep this model (other models have been proposed[1]) for ease of use. We will address this in the next chapter.

Secondly, mail servers must not identify themselves with their real identity if configured as a hidden service. This will also be addressed later.

Thirdly, upon sending mail from *.onion* addresses, all information that might give away the sender must be stripped out, or not be added. `Received` lines in the header would give away the IP address of the sender which is exactly what should not happen.

Another problem is the fact that Tor → normal email delivery is hard. Since the Tor default exit policy does not allow connections to mail servers, it is not easy to guarantee privacy for such email. We hope that this problem will not be a major problem, but it has to be observed. We also expect that such usage will be low, for most email delivery will occur completely within the Tor network (Tor → Tor).

[more?]

3 Proposed Solution for smarthosts

A proof-of-concept implementation has been done for the popular exim4 mail server. This mail server does not allow using outgoing socks proxies for connections, so a different solution must be implemented[2]. Our proof-of-concept implementation routes mail to *.onion/.tor* addresses to a local pseudo-SMTP server which is offered on a different port via xinetd, and is just a small proxy program that uses Tor as a socks proxy to open a connection to the destination mail server, and then lets exim talk to it. Exim must be configured to route mail to *.onion/.tor* addresses to this special mail

[1]One suggestion was to have clients directly deliver mail to the destination mail server via Tor, but this has the drawback that the user must either deliver all their mail via Tor (which is not useful when they use a smarthost), or choose (manually or via a program) which email to deliver via Tor — this destinction must be made on target address. Having a program for this complicates installations since this program must be installed on each client, and must be able to route mixed-emails as well.

[2] We could have implemented socks support in exim, but this seemed more complicated and less portable to other mail servers.

server, and the actual destination is passed to the proxy smtp server via the HELO/EHLO command. Please see the FAQ for more information.

4 Proposed Solutions for hidden SMTP servers

The second problem was hiding the service completely. One option is to run a completely stand-alone server (possibly on a different port) just for the hidden service, but this is sometimes not feasible and we try to have just a different port for the hidden service and configure the mail server properly to not give away itself. We need to observe the following ways it can give away itself:

- the initial greeting

  exim's default initial greeting is

  ```
  220 hostname.example ESMTP Exim <version> <date>
  ```

  where `hostname.example` is the primary domain name of the server. This gives away the domain name (possibly the 'real' domain the server is running in, not the *.onion* name), and the server's date and time including the timezone, which is a clue to its location.

  Additionally, this may be used in a correlation attack: If an attacker suspects that say one of a dozen servers is the backend service for the .onion domain, it can check which announces the same MTA version and time.

- error messages may include hostname information/IP addresses (if the Tor server is running locally (it should!) then the IP will most likely be 127.0.0.1 which doesn't give away anything)

- accepting mail for *host.example* when connected to via Tor: adversaries that have a suspicion that *xxzz.onion* is really *host.example* might try to test this by connecting via the hidden service *xxzz.onion* but sending mail to *postmaster@host.example.*

- TLS certificates and keys must not give away information.

- Even temporary RSA or Diffie-Hellman parameters used for encrypted connections might give away information! With the current exim4 code base it isn't possible to specify another file for these when exim is compiled with gnutls as opposed to OpenSSL (gnutls is default on Debian).

- possibly more?

So far we have not fully configured exim for these points, please see the section on hiding exim.

5 Stripping information

When a mail server that acts as a smarthost for a *.onion* domain is asked to deliver mail on behalf of a user (the user should use TLS and authenticate to the server, she need not connect via Tor since the server does this for her, obviously she has to trust her smarthost), it should make sure that it adds no information that might give away the sender. This includes Received: lines as well as information on the mailer, for example `X-Mailer` lines or the message ID. We recommend that the server create an own message ID based in its *.onion* domain, and strip all `Received`, `X-Mailer`, `User-Agent` and similar lines, and maybe even replace the date with the current date in the neutral UTC timezone.

Obviously users have to make sure they do not add information that might give them away to their emails, like documents containing names (cf. the amount of information embedded in Microsoft Word documents), digital signatures with their own key, or mail signatures they commonly use on other accounts as well.

6 Hiding exim

6.1 Hiding the hostname

Hiding the hostname was achieved by using another new port for the hidden service *mtx4fufynowi6kvp.onion*. We used port 26 in this example, and configured the hidden service as such:

```
HiddenServiceDir /var/lib/tor/hidden/smtp/
HiddenServicePort 25 127.0.0.1:26
```

Then, we told exim to listen on that port:

```
local_interfaces = <; :: ;\
                       127.0.0.1.26
```

and created a file named `/etc/exim4/port2hostname` with the following content:

```
# this file lists ports and the respective hostnames.
# this file serves to change the smtp_active_hostname

# port 26: hidden service
26: mtx4fufynowi6kvp.onion
```

and then told exim to use the active hostname depending on the port:

```
# if nothing is found in the file, default is $primary_hostname
smtp_active_hostname = \
  ${lookup{$interface_port}lsearch{/etc/exim4/port2hostname}}
# hide version/date against correlation attacks
smtp_banner = $smtp_active_hostname
```

This changes all exim messages (except custom ones) that use the hostname because they use this variable.

6.2 TLS certificate and key

[hopefully works as with the hostname, note that the key isn't needed since we cannot do TLS on outgoing connections, which sort of means we shouldn't need to accept TLS on incoming ones either... well maybe we can work around the outgoing TLS issue some time]

6.3 Hiding other domains

If your server is serving multiple domains, like ours is serving *sipsolutions.net* and *mtx4fufynowi6kvp.onion*, then you need to hide the fact that it is doing so. If an adversary might suspect that *sipsolutions.net* is also handling *mtx4fufynowi6kvp.onion*, he could connect to *mtx4fufynowi6kvp.onion* and try sending an email to a known address in the *sipsolutions.net* domain to confirm his suspicion. We address this as follows:

First, we removed the local addresses 127.0.0.1 and ::1 from the relay_from_domains list the debian default installation uses. This is required because if Tor is running on the same machine (recommended!) connections from the Tor network come over the local interface, just on a different port.

Secondly, we instead added a way to distinguish between connections to the acl_smtp_connect acl:

```
acl_check_connect:
  warn
    set acl_c7  = local
  warn
    !hosts      = :
    set acl_c7  = onion
  warn
    condition   = ${if eq {$interface_port}{25}}
    set acl_c7  = smtp
  warn
    condition   = ${if eq {$interface_port}{25}}
    hosts       = 127.0.0.1 : ::::1
    set acl_c7  = local
  warn
    condition   = \
      ${if or {{eq {$interface_port}{587}}{eq{$interface_port}{465}}}}
    set acl_c7  = submit

  # always accept submit connections, we later allow
  # mail on these connections only if the user authenticated
  accept
    condition   = ${if eq {$acl_c7}{submit}}

  # no point doing any DNS stuff...
  accept
    condition   = ${if eq {$acl_c7}{onion}}
```

after this we only have some DNS blacklist checks that result in the sender being stalled.

If you ever need to check for local, do it this way then:

```
  # Accept if the source is local
  accept
    condition   = ${if eq {$acl_c7}{local}}
```

With the acl_smtp_rcpt ACL we then check if the rcpt to can be accepted over this connection. We do this by first accepting any authenticated connections, and then inserting the following statements. Make sure you insert them **before** any statements that accept mail to postmaster, since otherwise one can still do the correlation attack with the postmaster address!

Note that since a *.onion* name may also have one or multiple *.tor* names associated, we need to handle that too. To do this, we create a directory `/etc/exim4/onion-aliases/` and a file named `xxx.onion` for each *.onion* domain we handle, into this file we put all aliases (usually just one *.tor* name) per line, below ACL reads information in this scheme. Note that the directory has to exist even if you put nothing in it!

```
deny
  !condition  = ${if eq {$acl_c7}{onion}}
  condition   = ${if match {$domain}{\N.*\.(onion|tor)$\N}}
  message     = relay not permitted

deny
  condition   = ${if eq {$acl_c7}{onion}}
  !condition  = ${if eq {$domain}{$smtp_active_hostname}}
  !condition  = \
    ${if and {\
      { \
        eq {${lookup{$smtp_active_hostname}\
              dsearch{/etc/exim4/onion-aliases/}}} \
           {$smtp_active_hostname} \
      } \
      { \
        eq {${lookup{$domain}\
              lsearch\
                {/etc/exim4/onion-aliases/$smtp_active_hostname}\
                {yes}{no}\
             }\
           }\
           {yes} \
      } \
    } }
  message     = relay not permitted
```

Note that it is wise to delay there if you also delay on regular ‘relay not permitted’ messages.

This is almost all you need. Now, sending mail to *johannes@sipsolutions.net* no longer works via the *mtx4fufynowi6kvp.onion* name and vice versa, but if I also had the email address *foo@sipsolutions.net* *foo@mtx4fufynowi6kvp.onion* would also automatically work. We solved this by implementing a virtual user scheme which is beyond the scope of this document.

7 Conclusion

We have shown that it is indeed possible to implement a hidden mail server network with hidden services inside the Tor network, and have implemented this on a few servers. If you want to test the hidden network you can reach Johannes Berg as johannes@mtx4fufynowi6kvp.onion. Note that by publishing the link here I loose the privacy of the *mtx4fufynowi6kvp.onion* name, but that is intentional since it was created for testing purposes only.

We hope that other people will follow and implement at least the outgoing part on their mail servers allowing them to send mail to *.onion* addresses. We suggest that if you do, you also run Torque on the server so that *.tor* addresses may be used as well. We also hope that this may help some people who need to stay anonymous yet want to participate in public communities.

Due to the many errors that can happen we also suggest that people using this use a special email client for sending mail with their *.onion* address so that they do not accidentally give away information from their regular email accounts, and that they test what information this email client sends and strip on the server accordingly.

A References

Tor http://tor.eff.org

The Exim mail server http://exim.org

Torque http://balrqba4x57ofa6s.onion/torque.php