Requirements to send unobservable messages accross the internet

Martin Gwerder

Abstract—In this paper We elaborate all criteria required to create a system capable of transporting messages unobserved through public networks. This is done by collecting all requirements based on the assumed capabilities of the participants. Sending unobservable messages is much harder as generally assumed. It involves a lot of parameters such as ”acceptance” or ”trust” which are commonly overlooked because of the technical approach most authors prefer. Using a holistic view to the topic we collect all requirements for such a system and categorise them. We furthermore describe why certain systems may not be considered unobservable at any time by definition.

Index Terms—Data privacy, Message systems, Security

1 Introduction

We do send messages across the internet every day. Most of them seem to be considered as ”not valuable”. However – secret services, advertising companies and researchers have found means to extract data from these messages and create profiles of all participating parties. These profiles often contain very valuable and personal information. The information collected contains often intimate details about all aspects of a person. Even worse: These details might be used against a sender, a recipient or even another peer-partner if they are considered a problem. To give a drastic example: This may happen if someone is a peer-partner to more than one person that is suspected to be a terrorist.

As economic aspects are in favour of this kind of data harvesting we can not expect this problem to solve itself. We do need the possibility to send information across public networks which do not disclose personal informations about any of the involved parties. A naive approach would be to encrypt all messages. While this might be sufficient to hide the content of a message it still leaks a lot of information. Even in a perfectly encrypted message the system still needs routing information (such as sender and recipient) to pass on the message. Using this and the technical information we have we may extract many valuable details such as sender, recipient, movement patterns of them, frequency of sending messages between them, average message length, classification of use scheme, preferred services, used devices and clients, and so on.

This does not solely apply to personal information. On a similar level it does disclose partnerships between companies, and reveal informations such as research, negotiations, and legal affairs.

There are lots of works [1] [2] [3] [**?**] [**?**] that relate to anonymous message transfer. However – none of these works (with exception to TOR [1]) have been widely adopted in the internet. The reason for this is usually that research tends to concentrate on the method to transport the message and fail at the same time completely to take the real world and its problems into account.

In this paper we analyse what is really required to build a service sending unobservable messages. I categorize them and try to build a foundation for future research on this topic.

gwm
Mai 27, 2015

2 Methods

I assume that we do have a *system* which may or may not require an *infrastructure*. An

Fig. 1. sending unobservable messages is not easy (Replace or remove image)

infrastructure may contain one or more participating nodes. These *nodes* serve one or more purposes, do have an *owner* that is capable of administering it, and may have *users* that use them. Users are either message *sender* or *recipients*.

On the thread side we do have an *observer* that is capable of monitoring all traffic in the public network and wants to collect data. He may cloak himself as *user* or *owner* and introduce malicious *nodes* into any part of the *system*.

For a more verbose definition see Appendix A. The *nodes*'s are connected through a packet based network (for ease of argumentation an IP based network is assumed). For definitions of terms such as ”unobservable”, ”anonymity”, ”psudonymity”, ”unlinkability” please refer to [**?**].

3 Results

Requirements for an unobservable system may be categorised into three groups:

- Acceptance requirements
- Protocol requirements
- Infrastructure requirements

3.1 Acceptance requirements

In order to establish means to send unobservable messages people must be willing and ready to use it. Acceptance of such a system is therefore an absolute key factor. Acceptance needs to be established at all levels (sender, recipient and owner). However – The exact acceptance criteria and the weighting may differ from group to group.

- Easy (acc.1)
  A system must be easy to use. The possibilities should be similar to common elaborated systems and the usage should be alike or the same. This offers a steep learning curve to the user.
  If ignored, only the users heavily concerned about their privacy would be willing to use the system. All others would ignore it as they are not ready to invest efforts into a system that offers them not sufficient benefits but new limitations.
- Fast (acc.2)
  In today's world we already adapted to fast moving messages. It is quite common that people talk to each other and send at the same time additional informations by chat or mail. They do expect that this information propagates fast through public networks. For some messages even an almost instant reply of the recipient is expected by the sender. Therefore any system must allow a fast transport of messages from the sender to the recipient.
- Reliable (acc.3)
  Messages are expected to arrive at the recipient's device. Today there are numerous common systems such as email, chat, sms and mms offering reliable transfers. Any system not sending reliably will not be used due to the limitations given by an unreliable system.
  Another part of reliability is the protection. The message protection must be unbreakable (within reasonable bounds). If the system can be attacked easily then it offers ”no value” for ”additional effort”. For most users this would be a reason to discard such a solution.

- Not abuseable (acc.4)
  Any system may be abused. The willingness of using a system if it is to easily abusable is very limited. A user will not be using a system which increases UBM (unsolicited bulk messages) or enables someone to blackmail him easily.

3.2 Protocol requirements

- Unidentifiable (pro.1)
  If a message or a participating node is identifiable then it is easy for an observer to block some or all parts of the system. This makes the system unreliable and may force users to use specific nodes (such as nodes which are under the control of the observer) and therefore compromise the overall security. Only a service that is able to hide its messages in legitimate network traffic is not subject to selective blocking.
- Untagable (pro.2)
  If messages going through the system are tagable by any of the participants (nodes) then an observer might tag messages and then follow them while they are propagating the network. If information is appended to a message it must be cloaked with the same reliability as the original message itself.
- Unreplayable (pro.3)
  If an observer can replay any part of the message (send it multiple times), he can identify the traffic generated by those messages by statistical means. This would enable him to identify traffic which is caused by a specific message and thus narrow down the possible final recipients.
- Monolithic messages (pro.4)
  Messages should not depend on external content (such as images). If a message is not self-contained then "bugging" is an easy way to identify the message on its way up until they reach the recipient or the recipient itself.

3.3 Infrastructure requirements

- Unknown endpoints (inf.1)
  Every endpoint should behave the same as an intermediate routing point. They should receive and send messages so that they are not identifiable as endpoints. Identifiable endpoints simplify analysis.
- No relations between single hops (inf.2)
  Messages transferred from server to server must be unrelated. Server identifiable to send messages due to received messages are potential targets for analysis.
- Untrusted infrastructure (inf.3)
  Unlike in a company owned net, in a public network trusting an infrastructure is not sensible. It is very often not clear who owns a server and who else does have access to it. The motivation of an infrastructure owner is often not clear and his intentions may or may not be sincere. So an unobservable system may not build its unobservability based on behaviour of the transporting infrastructure.
- No central infrastructure (inf.4)
  Central infrastructure may be attacked or shut down. They are easier to monitor than an unknown number of participants. Furthermore a central infrastructure may be used to compromise security of messages or nodes. It enables an observer to identify nodes by monitoring the traffic of a central infrastructure.
- No direct communication between endpoints (inf.5)
  If sender and receiver communicate directly then they are easily identified. So – all communications between endpoints should normally be done via intermediate nodes.

4 Discussion

Taking all criteria listed above together we get an unexpected set of results. First of all: it seems that unobservable messages is very likely to be asynchronous. It is far harder to hide a multi hop communication if we do not have the possibility to blend into other traffic. Blending into other traffic can be done on a high load node or when collecting up data before sending it on.

We identified clusters of criteria which should not be looked at isolated. These clusters are:

- Unidentifiable (pro.1), unknown endpoints (inf.1), no relations between single hops (inf.2), and no direct communication between endpoints (inf.5).
- Untrusted infrastructure (inf.3), no central infrastructure (inf.4), and unidentifiable (pro.1).
- Reliable(acc.3), and untrusted infrastructure (inf.3) .

Looking at the first cluster we see that the users interaction of the system should be equal to the one between nodes. This brings the conclusion that a users node is a specialized message router. Furthermore the infrastructure has to be an established one which has a common legitimate use. This denies the use of a dedicated service. It furthermore leads us to the conclusion, that the chosen service is asynchrnous (or synchronous with message clustering) and allows the service to send and receive messages (e.g. SMTP, or XMMP; but unlike HTTP).

Since trust is limited to the users any additional measures to hide the message (e.g. such as dummy traffic) have to be user controlled. Sending one big message containing all messages and dummy traffic will result in a big message that decreases in size over time. This is easily detectable. Therefore, the protocol needs a capability to recombine parts of the message without the knowledge of its content.

While using an existing layer of transport we still need to send commonly looking messages. This is only possible by applying techniques of steganography. Unfortunately there are very little elaborated technologies such as F5 [**?**] are available and there are very good approaches to detect their presence [**?**]. Even if we hide the message we have to be careful not to run into the dead parrot problem [**?**].

Looking at the second cluster brings up other conclusions. If we can not trust an infrastructure and the nodes and protocol should be unidentifiable then it means we have to use a common established infrastructure as a transport media. Since no such infrastructure has features mentioned above we need to add them on top of it. As we cannot trust an infrastructure or its admin it means that users must be capable of modifying the behaviour of the infrastructure to match the required results. This has to be done without any privileged access to the server.

The third cluster is not important from a protocol and infrastructure perspective. It is very hard to make a service error prone (acc.3) that includes the possibility to send error messages and react to it considering that we are not able to trust the infrastructure (inf.3). If we cannot trust it we are unable to provide a readable error reply address as it would disclose the senders identity. We do therefore need the possibility to provide a identity or routing token which provides the mean to create a fully unobservable way to reply. Naturally, this does not only apply to the senders identity but to the whole message.

If we have no trust into an infrastructure then we have to establish a kind of discardable identity. If not we do introduce this we do create a kind of pseudonymity enabling an observer to match same identities. Another approach would be that messages are sent to multiple destinations (could be done using a DC algorithm [**?**] or an improved version of it such as [**?**], or Xor-Trees [3]) without knowing the recipient. It could even be used to recombine multiple messages into one without the knowledge of any node that it has been done.

Another important criteria is ”not-abuseable” (acc.4). Having a system which sends unobservable messages makes it easy to blackmail someone. It is therefore vitally important to add unobservable routing information to a message to make sure that a legitimate recipient is capable of identifying the senders identity. Naturally this could be done by a signature. The problem with a signature approach is that as of today we do not have a possibility to find a trustworthy real world identity to a digital identity. We could enforce the use of certificates which contain real world information but by doing so we must trust third party infrastructure (violates inf.3) and add considerable complexity in the use of such a system (may violate acc.1).

5 Conclusion

As outlined in the previous sections being unobservable is tricky by itself. The main problem however is to make unobservable messages acceptable for every days use. To achieve this we need to take into account that people are not really aware of the extend being observed. And their readiness to invest efforts into this field is very limited.

Summing up the findings based on the requirements it looks as follows:

- Has to build on one or more established message transmission systems. Criteria are:
  - High load or asynchronous behaviour.
  - Symmetric behaviour in terms of routing.
  - Usable by steganography.
- Based on well known clients or having a custom one to behave similarly.
- Able to recombine message parts to one big message without knowing its content.
- Ability to send error messages to the unknown sender.
- Ability to modify routing behaviour without modifying the underlying infrastructure.
- Ability to hide next hop recipient or sender either by ...
  - introduce discardable identities
  - sending to multiple targets without knowing the true recipient.
- A routing token which allows to be attached to a message of unknown content. This token hides the recipient and the message from an observer.

Appendix A
Definitions

A.1 Definition of "observer"

As an observer of the system we assume the following attributes:

- Available founding is huge.
- Can have own mailer infrastructure.
- Is able to read, write or modify network data freely at any point of the net.

His intensions are:

- Discover message flows
- Discover message contents
- Identify users of the system

A.2 Definition of "user"

The assumed user of the system is:

- Does care about privacy.
- Does or does not have support from a mail server admin.
- Has no special computer knowhow.
- Has the ability to install a program or plugin on his personal computer.
- Has no cryptographic knowhow.
- Is using a device with enough calculation power to solve cryptographic tasks.

His intensions are:

- Send personal or confidential Information securely to another user

His expectations are:

- System should be easy to configure and maintain (in an ideal world: Zero touch).
- System should be fast.
- System should be reliable.
- System should work on any client he is using.
- System should not be a legal problem to him or any of his peers.

A.2.1 Subclasses of users

Users may be split into two groups.

- There is always one sender of a message.
- There is one or more recipient of a message.

There may be certain members of these groups which do not care about privacy may not be using an unobservable system .

A.3 Definition of "node"

The assumed node of the system is:

- Server publicly reachable.
- Server participating in the whole system.
- serves one or more defined purposes.
- does hove users participating in the unobservable system and other users.

A.4 Definition of "owner"

The assumed infrastructure owner of the system is:

- Does care about privacy.
- Has considerable computer knowhow.
- Has the ability to install programs or plugins.
- Has possibly no cryptographic knowhow.
- Does know his own infrastructure.
- Is using an Infrastructure with enough calculation power to solve cryptographic tasks.

His intensions are:

- Support his users in sending personal or confidential information securely to another user

His expectations are:

- System should be easy to configure and maintain (in an ideal world: Zero touch).
- System should be fast.
- System should be reliable.
- System should work on any client he is using.
- System should not be a legal problem for him or his company.
- System should still allow him to do regulatory tasks such as virus scanning or backup.

ACKNOWLEDGMENTS

The authors would like to thank their families for being so patient with them, Dr. Craig Hamilton and Dr. Manolis Sifalakis for their thoughts on the paper.

REFERENCES

[1] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *Proceedings of the 13th USENIX Security Symposium*, Aug. 2004. [Online]. Available: http://www.dtic.mil/cgi-bin/GetTRDoc?Location=U2&doc=GetTRDoc.pdf&AD=ADA465464

[2] U. Möller, L. Cottrell, P. Palfrader, and L. Sassaman, "Mixmaster Protocol — Version 2," IETF Internet Draft, Jul. 2003. [Online]. Available: http://tools.ietf.org/pdf/draft-sassaman-mixmaster-03.pdf

[3] S. Dolev and R. Ostrobsky, "Xor-trees for eicient anonymous multicast and reception," *ACM Trans. Inf. Syst. Secur*, vol. 3, no. 2, pp. 63–84, 2000. [Online]. Available: ftp://ftp.cs.bgu.ac.il/pub/people/dolev/31.ps

Martin Gwerder

Martin Gwerder was born 20. July 1972 in Glarus, Switzerland. He is currently a PhD student at the University of Basel.

After having concluded his studies at the polytechnic at Brugg in 1997, he did a postgraduate education as a master of business and engineering. Following that, he changed to the university track doing an MSc in Informatics at FernUniversität in Hagen.

While doing this, he steadily broadened his horizon by working for industry, banking, and government as an engineer and architect in security-related positions.

He currently holds a lecturer position for cloud and security at the University of Applied Sciences Northwestern Switzerland. His primary expertise is in the field of networking-related problems dealing with data protection, distribution, confidentiality, and anonymity.