Scalability & Paranoia in a Decentralized Social Network

Carlo v. Loesch[*] Gabor Toth Mathias Baumann

July 2011
(updated in October 2011)

1 Abstract

There's a lot of buzz out there about "replacing" Facebook with a privacy-enhanced, decentralized, ideally open source something. In this talk we'll focus on how much privacy we should plan for (specifically about how we cannot entrust our privacy to modern virtual machine technology) and the often underestimated problem of getting such a monster network to function properly. These issues can be considered together or separately: Even if you're not as concerned about privacy as we are, the scalability problem still persists.

2 More Than Just "Privacy"

One of the aims of a new social network infrastructure should be to maximize privacy, not just by encrypting exchanged data, but also to hide who is talking to whom, who is friends with whom, and more aspects of this kind.

The need for this degree of privacy may sound excessive at first, but "the history of cryptography is an arms race between cryptographers and cryptanalysts."[1] Therefore what is a theoretical fear today is a real threat tomorrow.[2] [3] In order to achieve long-lasting improvements over the current status quo it's a good idea to aim for the highest degree of protection currently feasible, right from the start, even if the end-users may not be aware and may not be asking for it as yet. We want to provide several features with absolute privacy:

1. updates, comments, postings, messages, files and chat are only visible to the intended recipients (not the administrators of any servers or routers)
2. the type and content of a message cannot be guessed at by looking at its size

[*] symlynX

[1] Jaeger, PSU: Introduction to Computer and Network Security, Lecture 5 - Cryptography (2007)

[2] Certificate Patrol implements pinning and monitoring of X.509 certificates. At first considered overkill, now recommended by W3C's Web Security Context WG.

[3] SSH. In the 90's, Unix systems shipped with telnet, rsh and rlogin. It was unimaginable at the time, that a freeware software would soon replace them for good.

3. communication between parties cannot be measured as they may have none to several routing hops in-between. an observer never knows if a communication came where it came from and ends where it is going to.
4. automatic responses and forwarded messages can intentionally be delayed so that an observer cannot tell two communications are related
5. communications cannot be decrypted weeks later, just because the attacker gained access to one of the involved private keys (forward secrecy)
6. even if an attacker gains access to a cleartext log, there is no proof the material was actually ever transmitted by anyone (for a case in court mere data would not suffice, you need actual testimonies)
7. the list of contacts is never managed on potentially unsafe servers, it is only visible to those it should be visible to
8. the infrastructure is robust and resilient against attacks

There is currently no technology we are aware of that actually fulfills all of these requirements. Even before looking at protocols, PGP as a tool fulfills 1 but fails at 5 and 6. Among protocols, OTR fulfills 1 and 5, but fails at 2, 3 and general reliability issues (you can't be sure a message was delivered and successfully decrypted). SILC provides for 1 and 7 but presumably fails at the rest. TLS only fulfills point 5 thanks to ephemeral DH handshakes and fails at the reliability of the certification infrastructure X.509.

Most web-based tools including home user web server technologies rely on TLS. Diaspora in particular considers status updates *low privacy*.[4] Unfortunately all web-based offerings cannot provide end-to-end encryption because web browsers simply do not provide it. In order to achieve that degree of privacy you will have to install an add-on or a software onto each involved device, anyway. Luckily there are P2P technologies that are getting closer to fulfill the list of requirements above. We'll get to them later.

2.1 Virtual Machines Considered Unsafe

"What is wrong with things on a server", you may ask, "if it is my own server?" Not many individuals own a server, but if they do so, in the coming years it will most likely be a virtual machine, not a physical one – simply because it is cheaper and easier to manage in many ways.

The problem with that is, cryptography in a virtual machine is very susceptible to attacks.[5] [6] By supplying it with a false random number generator, it can be lead to produce vulnerable encryptions.[7] [8] Also, the outer system running the VM can observe its memory and actions

[4]Notes on Diaspora.

[5]Scott Yilek, University of St. Thomas, Virtual Machine Reset Vulnerabilities and Hedging Deployed Cryptography (Dec 2010)

[6]Andy Greenberg, "Why Cloud Computing Needs More Chaos" (July 2009)

[7]T. Ristenpart, S. Yilek, UCSD, "When Good Randomness Goes Bad: Virtual Machine Reset Vulnerabilities and Hedging Deployed Cryptography"

[8]Michael Cobb, Cloud computing risks: Secure encryption key management on virtual machines (2011)

without being noticeable from inside.[9] This is a big change from traditional servers that would just run and do what they do. Anyone with access to the outer shells of hosting servers can monitor and collect data of all VMs running on each machine.[10] In the "Cleartext Passwords in Linux Memory" paper Sherri Davidoff shows how the extraction of many cleartext passwords in memory can be automated.[11] Even physical servers are affected using either a cold boot attack[12] or simply by attaching a snooping device to an external bus port like Firewire or E-SATA.

When dealing with virtual machines any other well structured data such as private keys or the social graph can presumably be extracted if a read access to the memory image is available while it runs. Xen's "xm dump-core" command for example lets the administrator inspect a VM's memory without stopping it. If implemented on a large scale, this allows to watch over many installations of privacy oriented services. An observing government can demand access to the outer controlling system of virtual machines and thus eavesdrop on private social networking activity. Legislation could come, that requires all hosters to provide access for government agencies to all servers, which would allow for data mining of an entire federated social web, if it is based on such servers. In fact it's so easy to provide for such a backdoor, it's improbable government agencies will not consider this option soon.

Thus we cannot entrust servers with the care-taking of our privacy. We must do it on the user's personal devices, be it a laptop or smartphone, or on a device which is maintained safely in a user's private home, such as a home server, a NAS or Internet router box. What we can do on servers however is to manage routing and storage of messages while users are offline. Routers will not be able to decipher the messages, they might even not know who they are from, they just have some things waiting for somebody, and only that somebody should be able to make sense out of it.

This may sound quite complicated technologically, but we should still be able to produce a Facebook-like user experience, at a degree of privacy that was to some extent hitherto enjoyed only by mad doctors, paranoid hackers and file sharers.

3 One Too Many: Multicasting for Scalability

It is generally underestimated how different the challenge for a social network is compared to the web or a messaging technology. Both the web and messaging are fundamentally one-to-one technologies whereas social usually requires one-to-many or many-to-many communications. The concept of the status update, its comments, or anything else you do in public is something that is intended to be broadcast to all of your friends, or rather contacts or peers, since users add a lot of people they aren't *really* friends with.

Some chat and messaging systems have a certain degree of experience with this challenge. 70%

[9] M. van Dijk, A. Juels: On the Impossibility of Cryptography Alone for Privacy-Preserving Cloud Computing (2010)

[10] Eric Siebert, "How to steal a virtual machine and its data in 3 easy steps" – as mentioned by High Cloud Security

[11] S. Davidoff, MIT: Cleartext Passwords in Linux Memory, 2008

[12] Ryan Naraine and Dancho Danchev, Cryogenically frozen RAM bypasses all disk encryption methods, 2008

of inter-server XMPP traffic is presence updates, the sort that needs to go from the sender to all subscribers.[13] Not surprisingly some parts of the XMPP backbone occasionally shake under the load - it is hard to imagine that this will improve once all the social traffic and status update comment chatter is added on top of that.

Now also consider a realistic number of peers: Today the average person on Facebook is said to have more than one hundred contacts.[14] Since people add contacts over a lifetime and hardly ever remove any, you should consider a thousand peers by 2020, if we seriously consider replacing Facebook.

XMPP provides simple distribution strategies: Each time a message needs to be sent to a person's peers the generating server will connect all receiving servers and send at least one copy of the message to each one of them, if not more than one. Multicast has been proposed and considered several times in the past, but rejected because of an unsolved trust issue towards the nodes that are expected to relay information. Since XMPP isn't encrypted from end to end, they could forge or censor messages. If it were encrypted end to end, it would be very inefficient because of the base64 encoding for embedded binary data. What remains is the possibility of out-of-band data exchange which means XMPP would only be used to signal the existence of messages and the actual traffic would be run over a custom tree of network connections using a separate binary-capable protocol.[15] In that case there isn't much left for XMPP to do, since most exchanged messages necessitate a smarter distribution and probably also deserve a more private treatment. There apparently is no satisfying solution to this dilemma in sight.

HTTP as a transport medium has advantages and disadvantages over XMPP, but as far as we know there is no out of the box solution to the distribution problem, either. So far it has worked by throwing processing power and bandwidth at the problem, especially between the involved data centers in cloudy setups. Therefore it isn't entirely surprising that corporations have a rather firm grip on seriously sized social networks: They can deploy proprietary solutions for achieving efficient distribution while openly federated systems risk running into scalability issues just when a technology actually reaches a certain degree of popularity. If it is hard to make it work for a single proprietary setup, it is unlikely to work in the heterogeneous Internet without a suitable new distribution technology.

IRC is worth looking at. It creates an application-level multicast tree for a more efficient distribution of chat messages than attempting to send a copy of a message to each recipient. Unfortunately IRC requires all users to be known to the entire distribution network which makes for a serious amount of overhead traffic and increases the effects of a denial of service attack on key router servers.

Telecomix seem to be looking at Usenet's NNTP. That could actually work as NNTP spans multicast trees, too.[16]

[13] Issues in XMPP

[14] https://www.facebook.com/press/info.php?statistics

[15] Statement confirmed in person by XMPP Standards Foundation (XSF) council member Ralph Meijer at the FSW2011 conference.

[16] Telecomix

4 Solutions?

Both scalability & privacy requirements for a truly worthwhile decentralized social network can be addressed, but it takes more effort than the federated social web has been planning for. What we probably need is a distributed application-level private multicast backbone.

Network level "IP Multicast" doesn't scale for a large number of multicast contexts (each user's status updates being at least one). It should work for a few thousand television channels, but not for millions of personal subscription channels.[17]

Since we have to rethink our architecture from scratch it isn't very helpful to stick to suboptimal protocols with legacy requirements and implications, we can however leverage existing deployments and social structures by upgrading the underlying technology.

In our project group we have been researching suitable technologies. We have found GNUnet to be the most promising, followed by Maidsafe[18], A3[19] and Tonika[20]. GNUnet researchers at Munich University of Technology have worked on the issues raised in this paper in detail[21], based on previous work by projects such as Freenet[22], Tor[23] and I2P[24].

All of them would need to be extended with trust[25] and social network modeling plus, in the long run, with multicast routing options. An ultra private file exchange and friend-sharing tool is likely to come out as a side effect, since that is frequently the original aim of these projects.

We have decided against some alternatives, most notably I2P and Diaspora (as they currently stand), since we think the core routing technology should be in a natively compilable language, suitable for running on embedded machines and, by default, as a background daemon on end user operating systems.

5 Conclusion

For "the web" to become seriously social, it needs a native, open and free backbone technology that enables communication beyond the one-to-one scenario. In order to protect privacy the best, such a technology should also fulfill several security requirements.

[17]Notes on IP Multicast

[18]Maidsafe, maidsafe-dht

[19]http://a3anonymity.com/

[20]http://5ttt.org/

[21]https://gnunet.org/

[22]https://freenetproject.org/

[23]https://www.torproject.org/

[24]https://www.i2p2.de/

[25]Papers like 'Community-based trust mechanism in P2P social networks' and 'PeerSoN: P2P social networking - early experiences and insights' underline the importance of a trust model to counter malicious attacks in P2P architectures.