PRIVACY AND SECURITY - FRIENDS OR ENEMIES?

Michael Sonntag

Institute of Networks and Security
Johannes Kepler University Linz
michael.sonntag@ins.jku.at

Keywords

Cyber security, privacy, interdependence, IoT

Abstract

Privacy and security seem to be natural enemies: an attacker cannot be identified if he is anonymous, and if we want to remain incognito, man-in-the-middle attacks become possible. But this is not always the case, as privacy and security might need each other. For instance anonymous remailers do not work without encryption, and police investigations often require anonymity. But while privacy obviously needs security, the other connection is not so obvious. This article discusses such interdependencies. Specific topics include storing IP addresses in webserver logs (legality is based on whether technical needs exists outweighing privacy risks) and whistleblowing platforms (strong security needed to absolutely guarantee anonymity, including how to obtain trust in a platform where the actual users will never recommend them to remain anonymous).

1. Introduction

Privacy and Security are often seen as antagonists: only criminals want privacy as everyone else has nothing to hide, and if privacy is granted, security deteriorates and terrorists rejoice. However, this need not be the case. Consider the following examples: if a security problem/incident can be disclosed anonymously, an employee is much more likely to publish it, as she need not fear repercussions from her employer. And publishing a vulnerability a company knows that it exists, but does not acknowledge or patch, can be a significant permanent increase of security – after a short time of decrease. Or imagine disclosing that all customer login passwords (clear text or weakly hashed) had been stolen: users can then take countermeasures (changing password) or precautions (paying attention for signs of identity theft). Companies like to keep such incidents secret, even if they occurred on a massive scale, which was made evident last year e.g. by Yahoo, when it admitted to have lost 500 million user accounts in 2014 and 1 billion in 2013 (Goel/Perlroth 2016). But privacy can directly improve security too: encrypting communication (using HTTP*S* instead of HTTP) protects against eavesdropping and content manipulations, but is also useful to verify the identity of the webserver, i.e. to ensure you are really contacting your bank and not a phishing website. And note that while your bank has every right to identify you before granting access to your account, everyone else should not even know which bank you are contacting (highly desired for phishing to show you an exact replica of *your* bank, and not just *some* bank) and for what (could influence your creditworthiness). Strong anonymity, e.g. surfing the web via the Tor anonymization network, improves user's security (or safety) significantly – in some countries this might actually save your life.

This paper is therefore concerned with the interdependence of features: explicit functional ones (what privacy implications security features have and vice versa), but also indirect ones (like how to acquire trust into anonymous servers in the so-called darknet). Another issue is feature interaction in the security domain: two features might be perfectly secure and fine, but their combination is perhaps not. Consider the following scenario: your flat is equipped with one of the new loudspeaker-assistants (Amazon Echo, Google Home etc; they require a specific word to activate as a "security" feature) as well as a door intercom (securing physical access). What if someone rings at your door, you answer it (thereby activating the loudspeaker of the intercom), and the "attacker" than quickly plays a high-volume command, which is immediately executed by the system before the owner can react? This could be a command to open the door, download some malware, or access pornography. Or consider a fridge with an audio interface (for telling you the current content/weather/…) or a smart mirror (reading to you the news or your new E-Mails). If these devices get hacked, a similar indirect attack becomes possible. Increased privacy, like physical interaction before performing commands or training to specific voices could reduce this security issue. That such IoT devices can often be hacked easily is shown by the Mirai botnet (Krebs 2016), which consisted at one time of approx. 3 million devices. Apart from being used in attacks on third parties, for instance hacked cameras are definitely a privacy problem too.

What this paper leaves open is finding the exact balance between security and privacy – which is less a technical discussion and more an aspect of society and to be framed in laws by politics.

2. Interdependence of privacy and security

Privacy needs security. If data is not encrypted and secured against modification, privacy does not really exist. Because if data remains unencrypted, everyone with access to it will be able to see the content/any metadata (documents stored somewhere) or know who is the sender/recipient (communication). Additionally a lack of security allows impersonation, thereby obtaining more personal data from the other party. This is not a law of nature. If for instance in the offline world the source has been completely anonymized (cutting letters from newspapers, gluing them on an empty sheet with gloves etc), it can be stored or sent anonymously. This is possible, but impractical. Unfortunately with IT systems this is impossible, as communication needs source information for replies (technical, not in the sense of a message), and stored data is always associated with metadata by various entities like the operating system, a webserver, anti-virus logs (so in the analogy no gloves exists). Also, practically all activities on a computer leave traces – these can only be prevented by security measures.

A big disadvantage of IT security measures improving privacy is that they mostly depend on third parties. To identify the other partner (whether we can trust him/her) we need a certification authority or lots of other persons (web of trust). If we want to communicate anonymously, we need trustworthy Tor node operators or anonymous remailers. If we buy digital goods online with Bitcoins, we have to hope that the operators do not store our IP or E-Mail address etc. Therefore most systems contain some sort of "chaining"; they perform anonymity measures several times after each other identically and ensure that any single trustworthy entity suffices. In this way the lack of trust into an individual is replaced by a trust, that at least one of a group will be trustworthy.

Security needs privacy. As seen recently, security researchers (Krebs's website under the Mirai botnet attack) or security measures (regular DDoS attacks of increasing bandwidth against providers) can come under attack easily. The reason is, that everyone knows who and where they are. Attack-

ing someone anonymous (see the similarly named collective) is much more difficult, enhancing their security. And while security by obscurity is no solution, it often renders attacks much more difficult (so use it, but do not rely on it). Another example is Google. Many persons depend on its search engine, especially that they are directed towards valid information (not fake news) and no websites containing malware. But attackers urgently desire to be included at the top of result lists. Therefore they present different content on their websites to the Googlebot than to other (human) visitors. If everyone is perfectly identified by the webserver, countermeasures against this would be extremely difficult, as the owners could easily differentiate whether to include malware in their response or not. Because of this, Google visits webpages anonymously too: they impersonate a "normal" visitor with a normal browser and compare the results. This includes not only faking the browser header but also using varying IP addresses that are not directly associated with Google. So they are using privacy features to verify web visits for search results, enhancing end-user security. Consider also Uber (Isaac 2017), who showed false information to persons suspected as government officials to avoid being detected operating illegally. Here privacy would have been very helpful to obtain evidence.

But most of the time, security can work without privacy, and sometimes this might even cause problems. See e.g. mail server lists against spam (open relays, weak anti-spam policies…): mail servers should not remain anonymous; otherwise this countermeasure will not work anymore. Therefore many exit nodes in the Tor network prohibit port 25 - the SMTP protocol – which is used for sending E-Mails.

2.1. Storing IP addresses in webserver logs?

Whether storing (dynamic) IP addresses of visitors in the logfile of a website is legal or not was at the center of a recent court proceeding in Germany (BGH VI ZR 135/13 – not yet decided), which also involved the ECJ (6.12.2016, C-582/14). The question was, whether these IP addresses are personal data or not. If yes, then they may not be stored and logs may only be kept without this information (or individual consent by each user is necessary – including for the first request, which is technically challenging). This is obviously a privacy issue, but could also involve security implications. The ECJ decided that IP addresses are personal data, as long as the person storing them has the legal possibility of obtaining the identity of the person behind the IP address. This is possible for instance in case of cyberattacks, when the web server operator can ask (through a court) the internet service provider owning this IP address to provide the identity of their customer it was assigned to at that point in time. As IP addresses are now definitively personal data, storing them is generally prohibited. In Germany this was even more strict, as no usage at all, except for providing the service and charging the user, was allowed by law. Therefore theoretically no web server is allowed to store any IP address in Germany anymore (but see below). This is not limited to webservers (only these proceedings are), but applies e.g. to firewalls or other security devices too.

While this is advantageous for privacy, it could result in a security issue. However, the ECJ simultaneously decided that the absolute prohibition of storing such data in German law contradicts EU law. The option of the person storing personal data to explain that because of overriding interests such storage and use is legal (as laid down in Art 7 lit f of the EU privacy directive 95/46) must remain possible. Then the provider has to argue why storing IP addresses is necessary, and for how long this reasoning applies, to allow the current practice to continue. This is however seen as problematic, as an expertise in this proceedings (Köpsell 2011) explained that for security reasons it is never necessary to store IP addresses, at least in direct (=unencrypted or unhashed) form. However, this expertise is in my opinion at least partially flawed or misleading. Consider e.g. the following

case: an attacker tries to hack a webserver and fails initially, but the next attack succeeds. The only trace to this attacker might be his IP address from the first attempt, and while it is correct that proxies or Tor anonymization servers can be used to hide the IP address of the real attacker, not every attacker will be using these, sometimes forget about using them, or mess up in a different way. Moreover, it is well established that encryption is very useful to improve privacy, but does not in any way remove the legal classification of data as "personal" as long as the key is not under control of a different person without a disclosure obligation. The same probably applies to hashing in case of IPv4: there are only 4 billion IP addresses, so hashing them all for a comparison is trivial. Because of this, encryption does not change anything legally, and from the security point of view only is helpful in the event of an attacker obtaining access to the log files, but not the decryption key (so asymmetric cryptography is needed or the symmetric key would presumably be obtained at the same time as the logs). Legally it is difficult to argue that a measure that would at least in some cases help to identify an attacker is more harmful than briefly storing an IP address, which in itself (and note: even for almost all attackers!) cannot legally, and almost always practically too, be associated with a computer - which is moreover one additional step removed from a person.

Additionally it should not be forgotten that storing IP addresses also works the other way round, e.g. for companies storing the IP addresses of their employees when accessing their own webserver or (e.g. on a web proxy) external servers. In this case there is no question that this is personal data, but additionally there can be no doubt that this storage is useful to pinpoint e.g. data extrusion (by the employee or because that computer is infected by malware) or illicit behaviour.

In my opinion the storage of IP addresses is a good example that privacy and security can coexist friendly. If this data is stored, there is little danger for the person involved: unless an attack occurs, there is little incentive for the company to attribute this IP address to an individual. Simultaneously, it would be very hard (and often illegal) for it as well. Any attacker would have the same problem as the company, so unless reidentification is possible, e.g. based on other information available to the attacker or inherently from the logged data, this is neither a very lucrative target nor a danger. The main driver behind this peaceful coexistence is the separation of the identifier (IP address) and the identifying information (who this IP address belongs to). Therefore any "integration", e.g. by secret services etc, is much more problematic. This assessment therefore only applies if the IP addresses are stored (securely!) and are used only for investigations because of actual attacks. This principle has been extended in the new privacy regulation as "pseudonymity". However, the regulation does not require different data owners or the existence of legal restrictions; only keeping it separate and technical and organisational measures to prevent combining the pieces are needed. This applies to the company-internal example above too, as typically the IP assignment is separated from the log files and must be kept in this way. It should be noted that additional security precautions are advisable to ensure that this separation remains in place unless there is a legitimate reason for identifying someone – and to prevent both parts stolen simultaneously, or combined without any traces like logfiles or instruction from supervisors.

2.2. Whistleblowing systems

Whistleblowing is becoming a commonplace occurrence in some areas, e.g. regarding secret services. Even if such disclosures are not necessarily seen as positive by all involved parties, much smaller occurrences are important as well. E.g. in many companies such a system is helpful to discover internal fraud or bribery. While in this context the need for privacy is obvious, special circumstances apply. For instance, often a channel for bidirectional communication is desirable, to e.g. allow questions regarding details or clarifications, preventing many common forms of anonymous

communication. Additionally security gets important in both directions: preventing an anonymous report with an attachment (e.g. ostensibly a document or financial data) from infecting company computers used for investigating the report, but also similar attachments of a question to whistle-blowers from disclosing their identity.

A new functionality hitherto not implemented by whistleblowing systems could be a secure confirmation of sending a report. The idea is, that you send a report about some shady dealings within your company, and if the problem becomes public, you can show the confirmation and explain, that you did report it and nothing more could have been expected from you. If the management did not act on his information, at least the whistle-blower will not go to jail. Here both security and privacy must be closely intertwined and work together to guarantee anonymity for the whistle-blower, but allow the whistle-blower – and only him/her! – to reveal her/his identity and prove, that a specific report was submitted at a provable point in time. How could such functionality be implemented?

First, the confirmation would have to include the specific content of the report. Otherwise a useless triviality is reported and the confirmation used for exculpation in a much more serious case. This content need not necessarily be the full report, but could e.g. be a hash value only. However, as this is to be used potentially "against" the company, the whistle-blower would have to be able to recreate the exact report or it would be useless as the company could provide subtly different files as "officially reported" – or none at all ("No report was received."). It is therefore recommended to simply include the whole message with all attachments. Only then a verification is possible and the report itself can be evaluated whether it did contain "enough" information for legal privilege. Simultaneously it must be ensured that the report has really been sent (and received), moreover at a specific point in time, as mere production and local storage of a report cannot be enough. This can be ensured with a signature received when submitting a report. Again it must be ensured that the company cannot later somehow invalidate this signature, e.g. by deleting a key or certificate. Therefore the whole signature chain needs to be stored. As the evidence might have to remain valid for a long time, a very secure signature is recommended; respectively the whistle-blower has to obtain confirmation timestamps/updated signatures over time to retain evidentiary value.

It is necessary to take into account that the company officially "signs" a document they do not yet know the specific content of. From the security point of view it is therefore important to use separate keys/certificates from other functionality and ensure that it is impossible to obtain an arbitrary signature. Therefore the system should not sign the report as it is, but rather a "confirmation document" containing a hash of the report, together with a timestamp and some random (=unpredictable) data. The whistle-blower can then obtain a signed timestamp from a third party for this document to prove that this report was received before that point in time.

A remaining problem is that the whistle-blower has to store not only the confirmation but also the report. This is evidence against him or her - if found in his/her possession. Secure storage is therefore important to protect privacy; e.g. by storing it in an encrypted container or as a (re-digitizable) printout in a physically secure location. If in doubt that the company would decline issuing a confirmation, a third party could be involved (which could be e.g. an independent operator of the whistle-blowing platform, or be implemented using blind signatures or fair exchange protocols with some other TTP). In this way simultaneous disclosure (report and confirmation) becomes possible.

Another potential issue is a fraudulent confirmation, e.g. after the content of the report is discovered and published, a top manager creates a report he claims to have submitted a year before, but which had been ignored, as an excuse for himself. As he might obtain access to the technical infrastructure (including keys etc) of the company, the only solution to this attack is an external timestamp. There-

fore the timestamp above for a successfully handed-in report, while optional for an honest whistle-blower, must be mandatory so nobody can create a report after the fact.

This example clearly shows the intimate relationship between security and privacy, where security is needed to guarantee privacy, while privacy is needed to obtain information improving inter alia security.

2.3. Trust in darknet servers

Servers in the darknet remain anonymous in the sense, that their physical location (and typically also their operator) are hidden. While this is not a problem for e.g. accessing Facebook via the Tor network (typically to avoid censorship), other services, for instance whistle-blowing platforms, find this problematic. How does someone know he really is at the platform to disclose information to *the* New-York Times and not just *someone/-thing* called "New York Times"? And where is this server physically (e.g. regarding fears of government interference)? He can only compare the onion URL (nytimes2tsqtnxek.onion) to a known value (of a hopefully trustworthy source), and potentially (if present) check a certificate (in this case issued only to "The New York Times Company" – which could be correct or not). While it would be very hard to modify the directory of hidden services, adding a new hidden service is trivial to do; then you only have to convince the victim to connect to your site instead of the real one.

Additionally there is a second level of trust: can we trust the real New York Times? This institution might be well known, but they are subject to USA law. And for shops offering illegal goods typically no one is known as the operator (respectively only as a pseudonym). To reduce this problem third parties holding the payment until confirmation of receipt are sometimes used. But whether these can be trusted is similarly problematic, as every now and then one of these escrow sites folds and runs away with the currently entrusted bitcoins. This closely resembles e.g. e-Bay: can you trust the seller to actually send the product (and not a stone inside a package), or the buyer to not claim having not received the product at all (or an empty package)? And while official postal confirmation is easy for legal sales, sending e.g. drugs through mail with registered sender and recipient could be problematic regarding privacy (the recipient must always be known or no product can be sent; false identities or stolen accounts for mail drop systems are needed then to retain anonymity).

So how is trust possible when the other side is technically anonymous – everyone can claim to be someone else easily? Here security comes in, similar to the web of trust. If many other persons trust a website/shop/whistle-blowing platform, then I can trust them too. Unfortunately, lots of positive reviews are easily written by a single person. Security should ensure that these persons are real and different (but potentially unknown) persons. Electronic signatures do not help there, as a key is as easily generated as registering a free E-Mail address is. So only if this anonymous identity is tied to multiple and extensive other elements (old blog posts, known from other sites etc) believability exists. But confirmation by other persons is only useful, if a chain to non-anonymous and trusted persons can be built, i.e. if the persons are not fully anonymous, but merely for you (=pseudonymous). While this can be falsified too, it is a lot more effort and especially it needs time to prepare. So logically "long existence" is equated with "trust". This therefore only works against short-term fraud, but not one prepared for a long time in advance; and identities providing confirmation for one fraud can be reused for the next one. Note that such trust is very difficult to bootstrap, as no previous record exists. The first user must blindly trust the service, and only later users might determine – based on his reports – that this is a/the real website.

Therefore this is especially problematic regarding whistleblowing platforms, as the actual users definitely do not want any traceability to their real identity. Confirmation can therefore only stem from non-users, i.e. the operators of the platform. As these are probably unknown as persons to potential whistle-blowers, another confirmation is necessary. Mere mention of the Onion-URL on a associated normal website is only useful, if the integrity and identity of this website is assured. E.g. if this website is not accessed via HTTPS, any system on the transmission path could easily change the URL. Another possibility would be a signed statement including the Onion-URL on the submission or the public website. Then trust in this statement's signers is necessary. But many persons can sign it so the probability of a trust path is much higher. Also, these need not be limited to the organization operating the whistle-blowing website, but can be any third person.

3. Feature interaction in the security/privacy domain

Feature interaction is concerned with two (or more features), which are perfectly fine if considered solitary, but may lead to undesirable behaviour if combined in a specific way. This can also be seen as emergent behaviour or an aspect of holism: combining elements can amount to more than the mere "addition" of these elements. This is not directly applicable to security, as is is not a functional feature which is "added" to a product. Still, security consists of very specific individual functions, like methods of user identification (password, fingerprints, facial recognition etc), storing data (various methods of encrypting data and organizational measure for storing the keys) or ensuring anonymity (like partitioning data into directly identifying information – kept securely locked away – and the rest of the data used under a pseudonym). Unfortunately, these elements might still be somewhat distributed (e.g. permission checks), so some standard approaches to prevent undesirable feature interaction, like architectural decomposition or serialisation of activities, are not applicable.

For privacy feature interaction is an ubiquitous problem, but typically seen under the name of "reidentification". Assuming we store various information about a person in file A, then this might be perfectly anonymous (or at least pseudonymous, according to the new data protection regulation; Example: student number + birth date). Simultaneously other information about the same person is stored at B (student number and home address), which is again anonymous or pseudonymous. But if both files are combined, the identity of the person can be identified, as birth date + home address are very likely unique, while either of this information is not. Any other data in these files is then attributable to this person.

What could be done to prevent, or at least simplify detection of, unwanted interactions of security or privacy features between themselves or with other elements? Examples are:

- Clear separation of security functionality: If security is intermingled with other functions, changing and updating it becomes difficult. Additionally it becomes more difficult to identify undesirable interactions, as it can be difficult to localize a problem, if it can stem from a security denial or a functional problem, but occurs at the same location. In the context of a web application for example a definition of access permission to pages should be centralized, e.g. to a configuration file, while manually checking permissions in the code should be avoided (easy to forget, varying error messages etc).
- Unified and centralized storage definition: Similar as security logging and data storage should be centralized. This means that the configuration which elements to log/store and where should have logically a single central configuration, while the physical location of the elements can be distributed widely and in whatever manner is useful (and which might de-

pend e.g. on privacy considerations too). This also allows to determine from a sample of the stored data whether their combination is problematic for privacy or not.

- User input must be treated cautiously: Every user input may be a security issue, as it might stem from a malicious user. In web applications this is obvious, as a website is typically accessible from the whole world. But this applies to all sensor input: it might have been hacked or its input (or output) modified, which could lead to undesirable behaviour in further processing. Any user input also has an originator, whose privacy might be important to protect – or not. Because of this, all input needs considerations whether it is attributed to a single person (can this be removed, should it be stored etc.), or whether – together with other data – it might allow identification.
- Improve authentication: See e.g. the audio examples from the introduction: while it is nice not to have to train the audio recognition, an unknown voice should be limited to "harmless" commands (which in itself can be difficult to decide upon). Speaking the name of the recognized person in the answer could be used as a simple feedback. This feature would simultaneously prevent loops between devices similar to the whistle-blowing example above: do not sign (or repeat) anything provided as a report (voice input), but create a new output merely based on the input received. While announcing the name seems to be contrary to privacy, it allows filtering. Only general information
- Interaction between security and privacy: Smart homes typically have a problem: either no internal permissions exist (anybody can issue any command; security problem), or authentication is necessary for each sensitive command (cumbersome, potentially problematic for devices lacking a user interface; privacy issue). If the identity is verified only once and then the user tracked across the home, no separate identification is necessary. This could be implemented in a way that any individual device only receives the information that the person issuing a command is authorized, but not who this person is, or only a pseudonymous token. This approach becomes especially important if these devices are not merely local elements, like a TV (age restrictions for watching), but networked and information is exported to external providers (like networked thermostats).

And while it seems obvious, a significant improvement for feature interaction in the security as well as privacy domain would be testing in actual environments by simulated attackers. The focus here lies on "actual environments" – as an individual element they may be perfectly safe, but in a *typical* (hopefully) and (ideally) *any possible* environment other elements could be important. As an example see the Mirai botnet mentioned above: alone e.g. a camera may be reasonably secure and not pose a danger. But when considering that it is connected to the Internet and is both reachable by anyone and can connect everywhere, they are insecure.

Centralisation seems a good approach to solve feature interaction (see e.g. the case for a single central smart home system in the cloud, monitoring and controlling everything), but this is in my opinion doubtful regarding feature interaction. A central instance may have the possibility to check all interactions, but is this actually going to happen? And how? The more elements are combined (and a single central instance would have to encompass by definition everything), the more interactions may and will occur, with a theoretically (in practice not everything will interact with every other device) exponential increase. And would it be in the interest of this company? Imagine that several devices together might disclose the identity of visitors in the home (e.g. doorbell/presence sensors + personal calendars of the persons living there) – for advertisers or social networks this would be very valuable and useful information and not an instance of unwanted, but rather desirable feature interaction. The visitors might assess this differently. Similarly criminals might see security prob-

lems as an advantage, but also would intelligence agencies or the police. Feature interaction, especially in security and privacy, must therefore also be appraised from the point of view of various stakeholders. Some might see this interaction as positive, others as negative.

Full separation of all devices is definitely a solution to feature interaction, but simultaneously ends all interaction and is therefore no practical and desirable solution. So a middle ground is necessary. How could this be found? Criteria for deciding on the degree of centralisation could be:

- Compartmentalization: Centralisation, but only for limited and small areas. E.g. centralisation for heating in a single household, but not citywide. The scope of potentially unwanted interaction is limited in this way, and easier to test too (see above). At the same time spreading out personal information is limited too.
- Hierarchy: Less interaction, and on a more general level the larger the scope is. E.g. regarding the heating example, a shared "average outside temperature estimation" could be citywide communicated to all devices. This also means disclosing only that information (outside temperature), which is really needed at higher levels. In this way privacy (e.g. inside temperature and its adjustment) is protected best.
- Independence: Systems should contain all the capabilities they need and not depend on other systems as far as possible. For instance speech recognition could take place locally instead of at a central instance, while e.g. the weather forecast will remain centralized. This might not be as efficient, but it automatically creates redundancy. This will hinder some business models and increase prices (more computing power needed locally), but retain at least some functionality in case the central instance fails, communication is not possible, updates are no longer available, the central API changes etc.

As an example for (lack of) independence serves a recent investigation (Lauinger et al 2017) of the security of JavaScript libraries, i.e. whether a website uses an outdated and known to be vulnerable version of a library: when a library is directly included in a website, then it is less likely to be outdated than when it is included indirectly from an intermediate script or within a frame. The inclusion of a script by ad, widget, or tracker code is in some areas almost double as likely to be outdated than direct inclusions. This could e.g. be reduced by directly including all elements into the own website (=local copy on webserver), which would also solve or at least reduce the problem of double inclusion, perhaps even in different versions, of the same library (also investigated in that paper). When not including scripts from external websites, these external parties also loose the possibility of tracking users, e.g. through cookies, thereby improving privacy. As a bonus, legal certainty increases too, as linking to foreign data is now becoming increasingly dangerous (legally and within the EU), a local copy and publication poses much less difficulties in assessing its legality.

4. Conclusions

Privacy is subordinate to security in ICT: security can exist without privacy, but not vice versa. But in many cases security does need privacy or is improved through it. This interdependence was explored through a few examples specifically and a discussion of feature interaction more generally. It was also identified that more attention is needed for the stakeholders: who is privacy and security intended for? In many cases these might be intended for "externals" only, but not between the end-user and a service provider (especially privacy). Security might also be intended to merely fulfil

legal obligations and secure one party – whether the other party is adequately covered is less an issue. For future development therefore the following aspects seem paramount:

- Independence: The more elements from other providers are included, the more fragile the whole system becomes, the more avenues for attacks exist, and the more persons can identify and track the user.
- Include embedding in a system: Similar to ordinary products, software must consider not only whether security and privacy are guaranteed in itself, but also for both normal use as well as foreseeable misuse. In the area of product security law this is already the basic standard for physical products. Software should adhere to the same standard.
- Interdependence in design: When implementing security measures, investigate the privacy implications and vice versa. Depending on the circumstances, one aspect might be more important than the other, several options might exists, or small additions can improve the other element.

5. References

Goel, Vindu, Perlroth, Nicole, Yahoo Says 1 Billion User Accounts Were Hacked, NY Times 14.12.2016, https://www.nytimes.com/2016/12/14/technology/yahoo-hack.html

Isaaa, Mike, How Uber Deceives the Authorities Worldwide, NY Time 3.3.2017, https://www.nytimes.com/2017/03/03/technology/uber-greyball-program-evade-authorities.html?action=Click&contentCollection=BreakingNews&contentID=64982515&pgtype=Homepage

Köpsell, Stefan, Sachverständigengutachten zu 57 S 87/08, http://www.daten-speicherung.de/wp-content/uploads/Surfprotokollierung_2011-07-29_Sachverst_an_LG.pdf

Kreb 2016, Krebs on Security: Mirai, https://krebsonsecurity.com/?s=mirai&x=0&y=0

Lauinger, Tobias, Chaabane, Abdelberi, Arshad, Sajjad, Robertson, William, Wilson, Christo and Kirda, Engin, NDSS' 17, http://www.ccs.neu.edu/home/arshad/publications/ndss2017jslibs.pdf