Secure Email - A Usability Study

Projet Fin d'Études (PFE)
CryptogrAphie, Sécurité, et vie Privée dans les Applications et Réseaux (CASPAR)
Supervisor: Karima Boudaoud
Polytech Nice Sophia Antipolis

Adrian Reuter
Technische Universität München
Email: adrian.reuter@tum.de

Ahmed Abdelmaksoud
Polytech Nice Sophia Antipolis
Email:

Wadie Lemrazzeq
Polytech Nice Sophia Antipolis
Email: wadie.lemrazzeq@etu.univ-cotedazur.fr

ABSTRACT

Keywords

Source Routing, Source Packet Routing, Segment Routing, MPLS, SPRING, RPL, DSR, RH0

1. INTRODUCTION

asdfasdfa

2. RELATED WORK

@Wadie

3. ANALYSIS OF END-TO-END ENCRYPTION TECHNOLOGIES FOR EMAILS

3.1 Pretty Good Privacy (PGP)

Developed by Phil Zimmermann in 1991, Pretty Good Privacy (PGP) is an encryption program that provides cryptographic privacy and authentication for data communication. PGP is used for signing, encrypting, and decrypting texts, e-mails, files, directories, and whole disk partitions and to increase the security of e-mail communications. In this study we will focus only on using PGP for e-mail security (e-mail encryption). It follows the OpenPGP standard for encrypting and decrypting data. Many e-mail clients provide OpenPGP-compliant e-mail security as described in RFC 3156 [1]. The current specification is RFC 4880 (November 2007) [2]. PGP encryption uses a serial combination of hashing algorithms (SHA-1, SHA-224 / 256 / 384 / 512), data compression algorithms (zip, zlib, and bzip2), symmetric encryption algorithms (3DES, AES-128 / 192 / 256, CAST5, IDEA) and finally asymmetric encryption algorithms (ElGamal, RSA (MUST NOT <1024 bits)). Symmetric-key cryptography involves using the same key to both encrypt and decrypt data.

In PGP, one-off key is generated randomly, which is known as the session key. The session key encrypts the message, which is the bulk of the data that needs to be sent. This type of encryption is relatively efficient, but it has a problem of sharing the session key with your recipient because without the key your recipient will only see the ciphertext. PGP solves this problem with public-key cryptography, also known as asymmetric cryptography. In this kind of encryption there are two keys: a public key and a private one. The public key of your potential correspondent can be found by searching through key servers or by asking the person directly. Moreover, each public key is bound to an e-mail address and has a unique fingerprint which can be used to get the right corresponding public key [?]. In PGP, public-key encryption isnâĂŹt used to encrypt the message, just the one-off session key that was generated to encrypt it. It would take too long and use a larger amount of computational resources. Since the body of the message usually contains the bulk of the data, PGP uses the more economical symmetric-key encryption for this. It reserves the lumbering public-key encryption for the session key, making the whole process more efficient. Our written signatures are frequently used to verify that we are who we say we are. They are far from foolproof, but they are still a useful way of preventing fraud.
Digital signatures are similar, using public-key cryptography to authenticate that the data comes from the source it claims to and that it has not been tampered with. Digital signatures work by using an algorithm to combine the senderâĂŹs private key with the data that they are authenticating. The plaintext of the message is fed through a hash function, which is an algorithm that transforms inputs into a fixed-size block of data, called a message digest. The message digest is then encrypted with the senderâĂŹs private key. This encrypted message digestis what is known as the digital signature [?]. In [?], the digital signature is sent alongside the message body (which can either be encrypted or in plaintext). When someone receives a digitally signed electronic mail (e-mail), they can check its authenticity and integrity by using the public key of the sender. First, a hash function is used on the message that was received and this gives the message digest of the email in its current form [?]. The next step is to calculate the original message digest from the digital signature that was sent. The senderâĂŹs public key is used to decrypt the digital signature, and this gives the message digest exactly as it was when it was signed by the sender. The final step is to compare the message digest

from the email they received to the message digest that they derived from the digital signature. If the message has been altered, then the message digests will be completely different, and the recipient will know that there is a problem with the message. If the two message digests are not identical, there are three likely culprits [?]:

- Inhalt...

3.2 Secure Multipurpose Internet Mail Extension (S/MIME)

3.3 Pretty Easy Privacy

Pretty Easy Privacy (pEp) is an initiative which aims to simplify end-to-end communication security. It is developed by two entities:

1. pEp Foundation, a non-profit organization based in Switzerland, owning trademarks and the pEp engine
2. pEp Security AG, a company based in Switzerland and Luxembourg, developing commercial pEp app and plugin implementations

The principle design goal of pEp is being easy-to-install and easy-to-use. pEp is strongly based on OpenPGP and its message format, but introduces several enhancements and new concepts in favor of usability and ease-to-use. The major conceptual improvement consists of depending less on user interaction and automatizing procedures, as well as moving forward to a security by default instead of a security by opt-in philosophy. Up to point of this paper, there exist three major implementations of pEp that can already be purchased and downloaded for usage: a Microsoft Outlook Plugin, a Mozilla Thunderbird plugin and a Google Android mobile app. Furthermore there is an Apple iOS mobile app announced to be in development and released soon [?].

Automatic Key Management
To achieve its primary design goal, pEp is designed to work without prior configuration by the user and to provide encryption by default and without user interaction [?]. The own PGP key pair is automatically generated in background upon first usage of pEp, or imported automatically if the user previously used PGP and already has a key pair on his system. Key pairs generated by pEp are RSA 4096 key pairs by default. If in the latter case an existing PGP key pair has less than 2048 bit length, a new key pair is generated instead of using the old key, such that the pEp security level does not fall below the commonly recommended RSA key length of 2048 bit for todays usage [?]. The own public key is always attached as file *pEpkey.asc* to each outgoing email, and consequently the public keys of other pEp users are received respectively by incoming mails [?]. Those public keys extracted from incoming mails are imported automatically into pEp. Using this key management approach, pEp does not depend on any centralized infrastructure such as key servers (as for PGP) or a PKI based on external certification authorities (as for S/MIME). As consequence, the manual key import by the user as it is known for PGP is circumvented and key management is fully automated. Encryption is applied by default once received the communication partner's public key, i.e. the user does not have to opt-in encryption manually.

pEp Handshake
A careful reader will recognize that the automatic key import mechanism explained above is vulnerable to a potential man-in-the-middle (MitM) attacker: He could replace the legitimate key in the mail attachment with his own key file. This fraudulent key would then be imported automatically without further verification, and the attacker could decrypt all mails sent to the victim whose key was replaced.
To mitigate this risk, pEp let's its users do a so-called pEp handshake: Once both pEp users received an email from the respective other user and thus having received its public key, pEp offers an option to verify the fingerprint of the received key. This is done by comparing so-called trustwords, which are shown on the screens of both users. These trustwords have to be compared via an out-of-band process, e.g. a phone call or any other secure instant messenger. Trustwords are 16-bit mappings between natural language words (e.g. english language) and the bitwise XOR of the own public key and the one of the communication partner [?]. If the trustwords shown to both users are equivalent, the received key is verified and can be trusted, with no MitM-attacker being present. The users can select to either trust or mistrust the other party's public key according to the result of the trustword comparison. To support compatibility with PGP users, the key fingerprint is also shown next to the trustwords, so that pEp users can always fall back to directly comparing fingerprints with non-pEp users [?].

Enhanced Security Status Transparency
pEp strives to make the security status of each individual communication channel (i.e. an email exchange with a certain recipient) more transparent and easy to assess for users. It does so by using a standardized color scheme which represents the security status of a channel. The status of no cryptographical end-to-end protection being applied is represented by grey colored icon, displayed by a visually dominant icon in the graphical interface when writing a new mail. This status corresponds to email communication with a recipient that do not use pEp, or to the very first mail sent to another pEp user [?]. Once received the public key of the recipient as e-mail attachment, future e-mails to that recipient will be sent encrypted and integrity protected, represented by a yellow colored icon. After two users successfully performed a handshake and chose to trust the other party's public key, this status is represented by a green icon [?]. In case there is a trustword mismatch (a potential MitM-attacker in place) and the users choose to mistrust the received key, this status is represented by a red colored icon. It is important to mention that the step of mistrusting a public key is non-reversible and can't no be undone other than deinstalling the pEp instance – which evidently would also delete any positive handshake results [?].

4. METHODOLOGY

4.1 Online survey

4.2 Live observations

5. RESULTS

5.1 Online Survey

5.2 Live observations

6. DISCUSSION

7. CONCLUSION

8. REFERENCES

[1] M. Elkins, D. D. Torto, R. Levien, and T. Roessler, “MIME Security with OpenPGP.” RFC 3156 (Proposed Standard), Aug. 2001.

[2] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer, “OpenPGP Message Format.” RFC 4880 (Proposed Standard), Nov. 2007. Updated by RFC 5581.