Underappreciated Security Mechanisms

Peter Gutmann

University of Auckland

Introduction

Standards for secure email and network communication have been around for years

Everyone has heard of S/MIME, SSL, IPsec

Why isn't the Internet secure yet?

- Existing mechanisms are too hard to use
- Existing mechanisms solve the wrong problem

Lesser-known (but very useful) security mechanisms get almost no coverage

What this talk will cover

User identification

Opportunistic encryption

Key continuity management

User Identification /Authentication

Allow users to sign up for online information (mailing lists, web sites)

- Fraudsters sign up in other people's names
  - Used for DoS, not just pure fraud
- Bots sign up large numbers of addresses to obtain accounts for spam purposes

Email-based Identification

Use the ability to receive mail as a form of (weak) authentication

- Sign up using an email address
- Server sends an authenticator to the given address
- Address owner responds with the authenticator to confirm the subscription
- Sometimes known as double opt-in

Widely used for password resets, mailing list subscriptions

- Good enough unless the opponent is the ISP

Email-based Identification (ctd)

Self-auditing via email confirmation

- Attempting to use the account results in the legitimate owner being notified
- Changing the email address should result in a notification being sent to the original address

Outlook

- More of the same
- Low-value authentication, but relatively difficult to defeat

Identifying Humans

Prevent bots from signing up for online accounts

- Reverse Turing test
  - Turing test: Can't distinguish between human and machine
  - Reverse Turing test: Distinguishes between human and machine
  - Also known as a Human Interactive Proof (HIP)

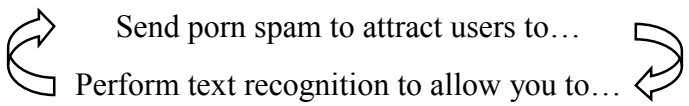
Reverse Turing Test example

- Display distorted/noisy picture of a word
- User has to enter the actual word
- Text recognition in the presence of noise is extraordinarily difficult for computers

Identifying Humans (ctd)

Harness humans to defeat it

- Pay people in third-world-country Internet cafes to perform the text recognition
- Grant access to porn sites in exchange for performing text recognition



Outlook

- RTT will at least slow the flood
- Not a perfect solution, since you can always use real humans to defeat it

Opportunistic Encryption

After 10-15 years effort, S/MIME and PGP use is lost in the noise floor

- Most mail clients include S/MIME support
- Many (OSS) clients include PGP support
- Usage is virtually nonexistent
  - It's too much bother for most people

The vast majority of users detest anything they must configure and tweak. Any really mass-appeal tool must allow an essentially transparent functionality as default behaviour; anything else will necessarily have limited adoption

— Bo Leuf, "Peer to Peer: Collaboration and Sharing over the Internet"

Opportunistic Encryption (ctd)

Encrypt data using keys managed via key continuity (see next section)

- Completely transparent to end users
- Requires no extra effort to use
- Effectively free (except for the slight CPU overhead)

Most commonly encountered in SMTP/POP/IMAP

- Protects mail in transit
- Authenticates sender/prevents unauthorised relaying/spamming

STARTTLS/STLS/AUTH TLS

What is it?

- Opportunistic encryption for SMTP/POP/IMAP/FTP

```
220 mail.foo.com ESMTP server ready
EHLO server.bar.com
250-STARTTLS
STARTTLS
220 Ready to start TLS
<encrypted transfer>
```

- Totally transparent, (almost) idiot-proof, etc
- Protects more mail than all other email encryption protocols combined

STARTTLS/STLS/AUTH TLS (ctd)

Outlook

- A year after appearing, STARTTLS was protecting more email than all other email encryption protocols combined, despite their 10-15 year lead
- Just as SSH has displaced telnet, so STARTTLS may displace (or augment) straight SMTP
  - Auckland Uni turned off unencrypted mail to local servers after STARTTLS appeared, just as they turned off telnet after SSH appeared
- Not perfect, but boxes attackers into narrower and narrower channels

Key Continuity Management

Where's the PKI?

It's too…

- Expensive
- Complex
- Difficult to deploy
- Doesn't meet any real business need
- etc etc etc

Key Continuity Management (ctd)

The only visible use of PKI is SSL

- This is certificate manufacturing, not PKI
- Once a year, exchange a credit card number for a pile of bits
- Three quarters of *all* SSL server certs are invalid (SecuritySpace survey, December 2003)
- No-one notices…

Assurance through Continuity

Continuity = knowing that what you're getting now is what you've had before/what you were expecting

- McDonalds food is the same no matter which country you're in
- Coke is Coke no matter what shape bottle (or can) it's in, or what language the label is in

Continuity is more important than third-party attestation

- Equivalent to brand loyalty in the real world
- Businesses place more trust in established, repeat customers

Use continuity for key management

- Verify that the current key is the same as the one you got previously

Key Continuity in SSH

First app to standardise its key management this way

- On first connect, client software asks the user to verify the key
  - Done via the key fingerprint, a hash of the key components
  - Standard feature for PGP, X.509, …
- On subsequent connects, client software verifies that the current server key matches the initial one
  - Warn user if it changes

Concept was formalised in the resurrecting duckling security model

- Device imprints on the first item it sees
- Device trusts that item for future exchanges

Key Continuity in SIP

Same general model as SSH

- First connect exchanges self-signed certificates
- Connection is authenticated via voice recognition

Same principle has been used in several secure IP-phone protocols

- Users read a hash of the session key over the link

Key Continuity in STARTTLS et al

SMTP/POP/IMAP servers are usually configured by sysadmins unconcerned about browser warning dialogs

- Remember the initial certificate, warn if it changes
- Using self-signed certificate avoids having to pay a CA

The ideal key continuity solution

- Automatically generate self-signed certificate on install
- Use key continuity to warn if the certificate changes

Key Continuity in STARTTLS et al (ctd)

Currently still somewhat haphazard

- Many open-source implementations support it fully
- Some still require tedious manual operations for certificate management
- Commercial implementations often require CA-issued certificates, an even more tedious (and expensive) manual operation

Key Continuity in S/MIME

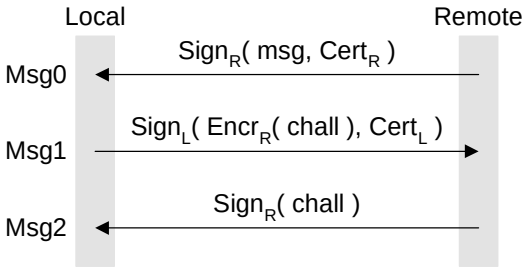
S/MIME has a built-in mechanism to address the lack of a PKI

- Include all signing certificates in every message you send
- Lazy-update PKI distributes certificates on an on-demand basis

S/MIME gateways add two further stages

- Auto-generate certificates for new users
- Perform challenge-response for new certificates they encounter

Key Continuity in S/MIME (ctd)



Msg0 gets remote certificates to local server (as standard S/MIME message)

Msg1 gets local certificates to remote user

Msg2 proves possession of remote server keys/certificates

(Variants, e.g remote server sends challenge in Msg3)

Key Continuity in S/MIME (ctd)

- Provides mutual proof of possession of keys and certificates to both sides
  - In practice has a few extra tricks to avoid various attacks
- Both parties now have verified keys for the other side
- Fully automatic, no human intervention required

Outlook

- Invented/reinvented as needed by implementors
  - Not specified in any formal standard
    - Standards groups are still waiting for PKI to start working
  - Present in many apps, but needs standardisation to unify approaches
- IETF BCP draft in progress

Conclusion

Simple human-in-the-loop solutions can be remarkably effective against large-scale automated harvesting attacks

Opportunistic encryption has achieved more penetration in one year than traditional methods did in 10