Ralph Durkee
Independent Consultant
www.rd1.net

Security Consulting,
Security Training,
Systems Administration,
and Software Development

PGP and GnuPG Rochester OWASP

Agenda:

- Generic Public Key Encryption
- PGP Encryption
- Digital Signatures
- PGP Keys
- Key Management
- Example Application usage
- Discussion

Public Key a.k.a. Asymmetric Encryption

- 2 Keys used
  - 1 Public Key – Made publicly known
  - 1 Private Key – Carefully Protected
- May Encrypt with either key (Public or Private)
- Decrypt with the opposite key used for encryption

Symmetric Encryption a.k.a. Shared Secret

- Encrypt and Decrypt with same key
- Problems
  - Have to securely pre-negotiate a shared secret
  - Sender can also decrypt messages.
  - Security depends on the strength of the secret
  - Need a unique secret for each communication path
- Benefits
  - Symmetric Encryption is about 100 - 1000 times faster than Asymmetric Encryption

The Best of Both Worlds Hybrid A/Symmetric Encrypt.

Combine the Best of Public Key with the Best of Symmetric Encryption

2. Start off with PK to avoid a shared Secret
3. Use PK to authenticate parties (May be either Party or Both)
4. Generate a one time random session key
5. Share the session key via PK
6. Switch to Symmetric Encryption for better performance, using the shared session key

Hybrid A/Symmteric Encryption

- Common Reoccurring Design Pattern Used in
  - SSL
  - IPSec
  - SSH
  - S/MIME

And …

  - PGP

PGP E-mail Encryption

1. Document is compressed
2. Random Session Key generated
3. Document is encrypted (symmetric) with the session key
4. Session Key is encrypted with the public key of every recipient.
5. Encrypted Key(s) and Document converted to ASCII Armor (RFC 2015)

Digital Signatures

- Digital signature is applied prior to encryption
  1. Generate a Secure Hash of the entire document.
  2. Secure Hash is kind of like a check-sum, but much better as it is not easily calculated in reverse.
  3. The Secure Hash is encrypted with the signers private key.
  4. Anyone with public key can decrypt the hash
  5. Decrypted Hash is compared against recalculated hash to verify that the document hasn't been modified.
  6. Provides non-repudiation as well as integrity.

Algorithm Details

- PGP like IPSec and SSL is really a collection or suite of Algorithms
- Choices of Algorithms (RSA & DH/DSS)
- Choices of Symmetric Ciphers
- Choices of key lengths
- Added Info included in the message
- Also included with the keys

PGP Keys -- What's in a key

- **ID** – Unique Identifier (0xDB93230C)
- **Type** – Algorithm Used (RSA or DH/DSS)
- **Size** – Key size in bits (usually 1024 and up)
- **Created** – Date Key was created.
- **Expire** – Date the Key expires
- **Cipher** – Symmetric Algorithm (CAST, IDEA, 3DES, AES-128 & AES-256)
- **Finger Print** – Secure Hash of the key (Good for validating key)
- **Validity** – Valid only if signed
- **Trust** – Implies auto-trust of other keys signed

Key Management
Getting Keys

Getting a key is easy -- Via:

- E-mail
- Key Servers e.g. **wwwkeys.us.pgp.net**
- Off from a web site
- PDA transfer
- Most any way a file can be received

Key Management Trusting Keys

- Methods of Validation
  - Face to Face with Photo ID (best verification)
  - Phone Call and read the Key Finger Print (Good if you know the person)
  - Get key from multiple sources and verify (Good for verifying signatures from on downloaded files.)
- Signing the key ensure its integrity

PGP File Formats

- **ASCII Armor -** for encrypted e-mail and preferred for key exchanges.
- **Clear Text Signed** – Text with in-line PGP signature
- **8 bit Binary** - default for file encryption for both GnuPG and pgp.com
- **Detached Signed** – Signature store in separate file (may be in ASCII or binary format).
- All formats have good interoperability across platforms. Even key rings are surprisingly portable

Sample Key
ASCII Armor Format

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: PGP 8.0

mQGiBDrP1xURBADFhi5Aw9XQ2ULmCZne7aY4VlV2Iqvw8lkDZvVe+TbSE1sHu8qB
1IBqQETXCM4H3c/jdJyqR+He4J7R3rfy/f/uAJhKyI5oklr+r43ardotsvvXoFXb
7qaQMw5H8lPINAHzUey86kNem1ewwSnrkicdRWOTuAokZJhGIc1U3cEUrwCg/9yJ
imMOduXdVTkO1Jy8BE6Lt9sEAMSHLGAdWz4sW3Mdc9KkVxDrWErY3SKTcLxpJ8B1
YTvHr7S+n66VK4mEzIrycTL4vGOWGpeKI+ga17fe/swYNz3TPhlJhwX3cvPI/37d
2Gqdie55qbYNyOgAO1oSLGWRqUc4nGkuwxA83uEwLHNRAIr/S+2gPRKWhxdCPN1T
3NR/A/934ervK95h+HtJwMmDopiPK1im3q4TuOoJSp/lOkhBGUIrXHQJoKUXCDNb

 . . .
iEYEGBECAAYFAjrP1xUACgkQLNgyONuTIwxlzgCgOfO31ddKPROkmZPGnq+etlia
sgMAnOIQlK4flhVqi29OMw9jIeDo+FDR
=abqO
-----END PGP PUBLIC KEY BLOCK-----
```

PGP E-Mail Formats

- PGP In-line – ASCII Armor or clear text signed
  - Recommended for e-mail without an attachment
  - Simplest format, just another message
  - Most compatible with MS Windows Mail Clients
- PGP/MIME - RFC 1847, 2015, 2440 & 3156
  - Encapsulates the message and all attachments into single verifiable MIME.
  - Most compatible with Unix Mail Clients
  - Most Reliable due to encapsulation

Sample
Clear Text Signed Format

```
Subject: SANS NewsBites Vol. 6 Num. 14
To: Ralph Durkee (SD1168) <rd@rd1.net>

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

**********************************************************************
SANS NewsBites                 April 7, 2004           Vol. 6, Num. 14
**********************************************************************
TOP OF THE NEWS
. . .
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.2.3 (Darwin)

iD8DBQFAc/CG+LUG5KFpTkYRApmaAJ4vVBY62P9Xv9AweD2rrJvH8qjnoQCdEO3x
NKG7+JbxZQ4OGe62FLBupRI=
=kXtD
-----END PGP SIGNATURE-----
```

Sample PGP/MIME

```
--=--
Content-Type: application/pgp-encrypted

Version: 1

--=--
Content-Type: application/octet-stream

-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.0.6 (FreeBSD)
Comment: For info see http://www.gnupg.org

qANQR1DBwU4D9+nmRMAIsHMQCAC6ne2TmKp0fX2fKDMGO0cO7aAvzLZYRUcrQgHg
. . .
F82Z0qFvv3HKHMifQB0XNPM=
=/4GH
-----END PGP MESSAGE-----

--=----
```

PGP More E-Mail Formats

- **Third Format --** Separate encrypted Attachments
  - Each encrypted separately
  - Message body may be encrypted in-line and/or as an attachment.
  - Doesn't prevent tampering by removing or adding attachments.
- MS Outlook always sends PGP w/ attachments separately, body is in-line and as an rtf attachment.
- Eudora prompts with warning that PGP/MIME is not compatible with some email clients.

Default File extensions by Platform

File Format	GnuPG	PGP
ASCII Armored	.asc	.asc
Binary Encrypted	.gpg	.pgp
Detached Signatures	.sig	.sig

PGP enabled Unix Mail Clients

Client	PGP/In-line	PGP/MIME
Mutt	Yes	Yes
Ximian Evolution	Yes	Yes
Sylpheed	Yes	Yes
Mac GPGMail	Yes	Yes
Mozilla Thundbird with Enigmail	Yes	Yes
Kmail	Yes	Yes

PGP enabled MS Windows Mail Clients

Client	PGP/In-line	PGP/MIME
MS Outlook	Yes	No
Eudora	Yes	Yes

PGP usages in Applications

- **Application to User**
  - Typically just encrypted or signed e-mail
  - Example generated and e-mailed SANS newsletter
- **Application to Application**
  - Typically Encrypted and signed
  - **FTP, HTTP** – Encrypted files generated and transferred for data processing
  - **Web Services –** encryption and signing for specific sensitive content.

PGP usages in Applications

- **User to Application**
  - **E-mail example**: PGP encrypted & signed e-mail opens access to an SSH port.
  - **HTTPS to E-mail example**: forms receives information over SSL, PGP encrypts and e-mails to user or to an application

Benefits of PGP for Application usage

- Sending Application only has public key for encryption
- Once original clear-text information is securely removed, information can not be obtained from sending application.
- Works best for message based application
- May not be suitable for where high volume of information is exchanged and high performance is required.

GnuPG Made Easy (GPGME)

- New GnuPG library for application usage.
- Intended to make Application integration easier.
- Mostly used by Mail User Agents currently
- Expected to help in future development of PGP-enabled-application

HTPS-PGP
Example of PGP integration

- Form receives sensitive information over HTTPS
- Rather than stored, The posted the information is PGP encrypt and e-mail to an application (or a user)
- Doesn't provide end-to-end encryption since it is re-encrypted
- Advantages, simple to implement and provides for distributed application.

Example Implementation

- Perl web form piped to gpg (GnuPG) command line piped to mail.
- **GnuPG options useful for applications**
  - **-e or –encrypt**
  - **-s or –sign**
  - **-r or –recipient**
  - **-a or –armour** (ASCII rather than binary)
  - **--batch** (don't use interactive questions)
  - **--no-tty** (don't output to the terminal)
  - **--quiet** (suppress messages)

PGP usage for Example

- PGP Public key of recipient application must be readable by web server user id.
- Public key and key-ring should not be writable.
- Private key for recipient application should not exist on the front-end web application.
- Clear-text private key is needed for signing by the web server. Helps recipient validate message.
- From address for e-mail should be web server, not the user posting the message.

Threats, Risk Security Controls

- May be appropriate to allow unauthenticated access to https form
- Be prepared for malicious abuse of the web form either way
- Use a Security hardened platform
- Follow Operating System and Web Server security best practice guidelines such as Center for Internet Security (www.CISecurity.org)
- Consider OWASP top 10 vulnerabilities and guidelines

Carefully Validate All Input

- Check for unexpected input field names
- Check maximum length for field values
- Check for non-ASCII values in input fields
- Check for unexpected ASCII values in input fields.
- Limit number of field and total post length
- Is the server vulnerable to HTTP header be manipulated
- preset key environment variables (like PATH) to expected values.

Other Risks

- **Access controls**
  - Is form accessible via http as well?
  - Can the HTTP get method be used?
  - If authentication is required, can it be by-passed?
- **XSS** – May or may not be an issue depending on back-end procesing
- **Injection flaws**
  - Ensure input can not affect the processing by perl, shell or gpg
  - Input should only be processed as data piped to gpg.

Discussion

- Any remaining Questions?
- Thought questions:
  - What are the weakness of PGP?
  - What attacks are likely if someone wanted to decipher PGP encrypted e-mail?
  - Is a Man-in-the-middle attack possible?
  - If not why not, or if so how could it be prevented?

References

- **GNU Privacy Guard** http://www.gnupg.org/
- **International PGP** http://www.pgpi.org
- **PGP Corporation** - http://www.pgp.com
- **mutt patch: pgp-menu-traditional** http://www.woolridge.org/mutt/pgp-menu-traditional.html
- **RFC 3156** - MIME Security with OpenPGP (updates 2015)
- **RFC 2015** - MIME Security with Pretty Good Privacy (PGP)
- **RFC 2440** - OpenPGP Message Format
- Key Servers http://pgp.mit.edu/ http://www.us.pgp.net