Protecting resources in an open and trusted peer-to-peer network

J.-F. Lalande, D. Rodriguez

Ensi de Bourges, France

METHOD 2012 - July 16, 2012
Peer-to-peer technologies are widely used:

- Open source software (e.g. Linux distributions)
- Commercial software
  - e.g. Skype
- Private networks (encrypted tunnels, authenticated users)

Not so much used for:

- Content delivery
- Business exchanges
Main issues with peer-to-peer technologies

- Contradictory with copyright laws
- The distribution process is uncontrolled
- The security guarantees are mainly for users
Main issues with peer-to-peer technologies

- Contradictory with copyright laws
- The distribution process is uncontrolled
- The security guarantees are mainly for users

The protocols mainly focus on safety:

- Anonymity of users (GAP, Freenet) [6, 3, 1]
- Survivability/Availability of resources [4]
- Access control?
  - ECRS [2] → sort of confidentiality and integrity by obfuscating and checking the content that is exchanged
- Protection of resources?
- Expressing security properties for resources?
Our goal: solve this conflict:

Express and enforce security properties

and

Keep the peer-to-peer network open

Open ?
- Keep the exchange protocol open
- Keep the client source code open and free
- Let the user define the policies

Security properties ?
- What can be expressed ?
- How to enforce them ?
What we do not want...

Change the peer-to-peer protocol:
- Authenticate users
- Use cryptology mechanisms to protect data

Change the peer-to-peer software:
- Use a closed source peer-to-peer client
- Rely on a trusted OS

Change the nature of the peer-to-peer network:
- Centralize the security checks
- Control the security policies of peers
Notion of domains

A domain is:
- a named group of resources
- associated to a set of security properties

The user is in charge of:
- create domains
- define the policy

User A

Domain A

Resource 1
Resource 2
Property 1
Property 2

Domain B

Resource 3

J.-F. Lalande, D. Rodriguez
Protecting resources in an open and trusted P2P network
Exchanges between domains

User A
- Domain A
  - Resource 1
  - Resource 2
  - Property 1
  - Property 2
- Domain B
  - Resource 3

User B
- Domain A
- Property 2
- Domain C
  - Resource 3
Protecting domains

- **integrity(sensitive_data_domain):** the resources of the sensitive_data_domain domain must not be modified.
Protecting domains

- **integrity(sensitive_data_domain):** the resources of the sensitive_data_domain domain must not be modified.

- **confidentiality(secret_domain):** the resources of the secret_domain domain must stay in this domain.
Protecting domains

- **integrity(sensitive_data_domain):** the resources of the sensitive_data_domain domain must not be modified.
- **confidentiality(secret_domain):** the resources of the secret_domain domain must stay in this domain.
- **spread(diffusion_domain):** the resources of the diffusion_domain domain must be available as much as possible for all peers and can freely change of domain.
Protecting domains

- **integrity**(sensitive_data_domain): the resources of the sensitive_data_domain domain must not be modified.

- **confidentiality**(secret_domain): the resources of the secret_domain domain must stay in this domain.

- **spread**(diffusion_domain): the resources of the diffusion_domain domain must be available as much as possible for all peers and can freely change of domain.

- **nopublication**(fee_paying): no new resources can be added in the fee_paying domain.
Protecting domains

- **integrity**(sensitive_data_domain): the resources of the sensitive_data_domain domain must not be modified.
- **confidentiality**(secret_domain): the resources of the secret_domain domain must stay in this domain.
- **spread**(diffusion_domain): the resources of the diffusion_domain domain must be available as much as possible for all peers and can freely change of domain.
- **nopublication**(fee_paying): no new resources can be added in the fee_paying domain.
- **noshare**(confined_domain): the files of the confined_domain should not be shared with another peer.
Protecting domains

- **integrity(sensitive_data_domain):** the resources of the sensitive_data_domain domain must not be modified.
- **confidentiality(secret_domain):** the resources of the secret_domain domain must stay in this domain.
- **spread(diffusion_domain):** the resources of the diffusion_domain domain must be available as much as possible for all peers and can freely change of domain.
- **nopublication(fee_paying):** no new resources can be added in the fee_paying domain.
- **noshare(confined_domain):** the files of the confined_domain should not be shared with another peer.
- **cooperation(priv_A, priv_B):** the peer should help the exchange of resources between priv_A and priv_B.
### Conflicting properties

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Conf. Integ. Spread !Pub !Share Coop.</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

**Conflicting properties**

For example:

- confidentiality conflicts with spread
Example of scenario

peer "director"

peer "employee A"

peer "unit director"
Example of scenario

peer "director"

domain company_info

memo.pdf

peer "employee A"

domain company_info

peer "unit director"

domain company_info
Example of scenario

- **peer "director"**
  - domain company_info
    - memo.pdf
  - spread(company_all)
  - integrity(company_all)

- **peer "employee A"**
  - domain company_info
    - spread(company_all)
    - integrity(company_all)

- **peer "unit director"**
  - domain company_info
    - spread(company_all)
    - integrity(company_all)
Example of scenario

peer "director"

| domain company_info | memo.pdf | spread(company_all) | integrity(company_all) |

peer "employee A"

| domain company_info | memo.pdf | spread(company_all) | integrity(company_all) |

peer "unit director"

| domain company_info | memo.pdf | spread(company_all) | integrity(company_all) |
Example of scenario

peer "director"

domain company_info

memo.pdf

spread(company_all)

integrity(company_all)

domain company_all

phone_numbers.doc

spread(company_all)

peer "employee A"

domain company_info

memo.pdf

spread(company_all)

integrity(company_all)

domain company_all

spread(company_all)

peer "unit director"

domain company_info

memo.pdf

spread(company_all)

integrity(company_all)

domain company_all

spread(company_all)
Example of scenario

peer "director"

domain company_info

memo.pdf

spread(company_all)

integrity(company_all)

domain company_all

phone_numbers.doc

spread(company_all)

peer "employee A"

domain company_info

memo.pdf

phone_numbers.doc

spread(company_all)

integrity(company_all)

domain company_all

peer "unit director"

domain company_info

memo.pdf

spread(company_all)

integrity(company_all)

domain company_all

phone_numbers.doc

spread(company_all)
Example of scenario

Peer "director"
- Domain: company_info
  - MEMO.pdf
  - Spread(company_all)
  - Integrity(company_all)

Peer "employee A"
- Domain: company_info
  - MEMO.pdf
  - Phone_numbers.doc
  - Spread(company_all)
  - Integrity(company_all)

Peer "unit director"
- Domain: company_info
  - MEMO.pdf
  - Phone_numbers.doc
  - Spread(company_all)
Example of scenario

peer "director"

- domain company_info
  - memo.pdf
- spread(company_all)
- integrity(company_all)

peer "employee A"

- domain company_info
  - memo.pdf
- spread(company_all)
- integrity(company_all)

peer "unit director"

- domain company_info
  - memo.pdf
- spread(company_all)
- integrity(company_all)
Example of scenario

peer "director"

domain company_info

- memo.pdf

spread(company_all)

integrity(company_all)

domain company_dir.

- memo_directors.pdf

integrity(company_dir)

confid.(company_dir.)

peer "employee A"

domain company_info

- memo.pdf

spread(company_all)

integrity(company_all)

domain my_domain

- my_report.doc

peer "unit director"

domain company_info

- memo.pdf

spread(company_all)

integrity(company_all)

domain company_dir.

integrity(company_dir)

confid.(company_dir.)

---

J.-F. Lalande, D. Rodriguez

Protecting resources in an open and trusted P2P network
Example of scenario

peer "director"

- domain company_info
  - memo.pdf
  - spread(company_all)
  - integrity(company_all)
- domain company_dir.
  - memo_directors.pdf
  - integrity(company_dir)
  - confid.(company_dir.)

peer "employee A"

- domain company_info
  - memo.pdf
  - spread(company_all)
  - integrity(company_all)
- domain my_domain
  - my_report.doc
  - confid.(company_dir.)

peer "unit director"

- domain company_info
  - memo.pdf
  - spread(company_all)
  - integrity(company_all)
- domain company_dir.
  - memo_directors.pdf
  - integrity(company_dir)
  - confid.(company_dir.)
Monitoring agent I

The security mechanisms are delegated to a Monitoring Agent:

- Manage the policies
- Checks policies when resources are exchanged
- Negotiate policies of domains when an exchange occurs
- Computes the trust of other peers
- Enforces policies locally
- Controls the peer-to-peer client
Monitoring agent II

- Peer client A
  - patch
  - controls
  - logs
  - P2P transactions
  - Security policy checks

- Peer client B
  - patch
  - controls
  - logs

J.-F. Lalande, D. Rodriguez
Protecting resources in an open and trusted P2P network
Monitoring agent III

Peer client A

controls
logs

Monitoring agent

integrity("companyFoo")
confidentiality("companyFoo")

P2P transactions

Security policy checks

Peer client B

controls
logs

Monitoring agent

confidentiality("companyFoo")

J.-F. Lalande, D. Rodriguez
Protecting resources in an open and trusted P2P network
An exchange, step by step

Peer client A
patch

Monitoring agent

integrity("companyFoo")
confidentiality("companyFoo")

Peer client B
patch

Monitoring agent

confidentiality("companyFoo")
An exchange, step by step

J.-F. Lalande, D. Rodriguez
Protecting resources in an open and trusted P2P network
An exchange, step by step

- **Peer client A**
  - patch
  - request

- **Peer client B**
  - patch
  - upload

- **Monitoring agent**
  - integrity("companyFoo")
  - confidentiality("companyFoo")

- **Logs**
An exchange, step by step

Peer client A

Patch

Logs

Monitoring agent

integrity("companyFoo")

confidentiality("companyFoo")

Peer client B

Patch

Logs

Monitoring agent

Policy of target domain?

J.-F. Lalande, D. Rodriguez

Protecting resources in an open and trusted P2P network
An exchange, step by step

Peer client A
patch

Monitoring agent
integrity("companyFoo")
confidentiality("companyFoo")

Sends policy of domain companyFoo

Peer client B
patch

Monitoring agent
confidentiality("companyFoo")

request
upload

J.-F. Lalande, D. Rodriguez
Protecting resources in an open and trusted P2P network
An exchange, step by step

Peer client A
patch

Peer client B
patch

Monitoring agent

integrity("companyFoo")
confidentiality("companyFoo")

checks policies...

request upload

J.-F. Lalande, D. Rodriguez
Protecting resources in an open and trusted P2P network
An exchange, step by step

1. Peer client A requests a patch.
2. Monitoring agent validates the request and logs it.
3. Peer client A uploads the patch.
4. Monitoring agent verifies the upload and logs the integrity and confidentiality of the patch.

Trust principles:
- 

Integrity("companyFoo")
- Confidentiality("companyFoo")

J.-F. Lalande, D. Rodriguez

Protecting resources in an open and trusted P2P network
Policy checks

Policy checks that should deny a request:
- target policy (peer A) is inconsistent:
  - confidentiality(companyFoo), spread(companyFoo)
- conflicts between target policy and source policy:
  - source (B): confidentiality(companyFoo)
  - target (A): spread(companyFoo)

If some checks fails:
- the peer-to-peer client download is stopped
- or the peer-to-peer client is killed
Advantages

For the implementation:

- a small modification of the peer-to-peer is needed
- any open source peer-to-peer client can be supported

For the peer-to-peer network:

- a peer A can participate without the monitoring agent
  - peer B will only upload for domain without properties
- policies are outside the peer-to-peer client
- policies can evolve to reflect new needs
Malicious peers

Peer A can be supposed to be a malicious node:
- What happens if A tries to guess source policy?
- What happens if A announces a fake policy?
- Is there any security enforcement in A?

For example, case 1:
- peer A knows that a file memo_directors.pdf exists
- peer A floods the peer-to-peer networks of requests
- For each request:
  - he tries a new domain name (to guess it)
  - he tries a new security policy (to be compatible)

⇒ evaluate the trust to put in a peer
Malicious peers

Peer A can be supposed to be a malicious node:

- What happens if A tries to guess source policy?
- What happens if A announces a fake policy?
- Is there any security enforcement in A?

For example, case 2:

- peer A announces the policy "confidentiality(company_directors)"
- peer A uploads files from company_directors for any request

⇒ evaluate the trust to put in a peer
Protecting resources in an open and trusted P2P network
Trust
Trust

file request (1)

Policy check (2)

peer A

peer B

P1

P2

P3

J.-F. Lalande, D. Rodriguez

Protecting resources in an open and trusted P2P network
Trust

File request (1)

Policy check (2)

Log evaluation (3)

peer A

peer B

P1

P2

P3

J.-F. Lalande, D. Rodriguez

Protecting resources in an open and trusted P2P network
Trust
Trust

File request (1) -> peer A
Log evaluation (3) -> peer A
Challenge Requests (4) -> peer A
Policy check (2) -> peer B
Remote Procedure Tests (5) -> peer A

J.-F. Lalande, D. Rodriguez
Protecting resources in an open and trusted P2P network
Trust

![Diagram of trust process](image-url)

- Introduction
- Proposed architecture
- Exchange principles
- Benefits and threats
- Trust

Trust vector computation (6)

File request (1)

Log evaluation (3)

Policy check (2)

Remote Procedure Tests (5)

Challenge Requests (4)
Protecting resources in an open and trusted P2P network

J.-F. Lalande, D. Rodriguez
Trust

The trust evaluation of $A$ is a combination of:
- the policy checks
- the reputation of $A$
- the evaluation of logs of $A$
- the evaluation of challenges sent to $A$

$\Rightarrow$ evaluates the trust $B$ can put in $A$
Prototype

Local enforcement of policies: FUSE module
- is configured by the monitoring agent
- protects resources from other processes
- informs the monitoring agent of accesses
Simulation hypothesis

Simulation with 100 peers:

- Discrete event simulator for peer-to-peer protocols [5]
- At each update each peer has:
  - 5% of chance to add a new file
  - 1% of chance to delete a file
  - 30% of chance to download a file chosen randomly
- 95% regular peers, 5% of malicious peers

For policies, history of transactions:

- static random consistent policies
- evaluation of history of transactions:
  - considered good for regular peers
  - considered bad for malicious peers
Simulation results I

Evolution of trust for regular and malicious peers

Feedback indicator: $F(p_B)$

Feedback average

Feedback average of malicious nodes

J.-F. Lalande, D. Rodriguez
Protecting resources in an open and trusted P2P network
Simulation results II

Peer 99 becomes malicious after 500s of simulation
## Conclusion and perspectives I

### Security properties associated to domains

- managed by a monitoring agent
- compatible with open peer-to-peer clients and protocols
- defined by the user (can evolve)
- enforced (eventually) locally
- enforced by evaluating trust of peers

### Difficulties for evaluating simulations:

- difficult to automatically simulate users
  - how to simulate domains?
  - how to simulate download requests?
  - how to simulate policy evolving?
Conclusion and perspectives II

Our other works related to this one

- open distributed crisis management tool
  - e.g. ensure confidentiality of some information
- security properties for cloud computing resources
- self protection of Android applications

All these systems have open frameworks!

- Users need security guarantees
- The system/network cannot be trusted or modified

How to bring more security to these systems?
Questions
References I

Tom Chothia.
Analysing the mute anonymous file-sharing system using the pi-calculus.

An encoding for censorship-resistant sharing-eecs.
Technical report, University of Purdue (USA), University of Denver (USA), University of Helsinky (FINLAND), 2003.
References II

Ian Clarke, Scott G. Miller, Theodore W. Hong, Oskar Sandberg, and Brandon Wiley.
Protecting free expression online with freenet.

Theodore W. Hong and Ian Clarke.

Aleksandra Kovačević, Sebastian Kaune, Nicolas Liebau, Ralf Steinmetz, and Patrick Mukherjee.
Benchmarking Platform for Peer-to-Peer Systems.

Bennett Krista and Grothoff Christian.
Gap - practical anonymous networking.
Technical report, Departement of Computer Sciences, University of Purdue (USA), 2002.