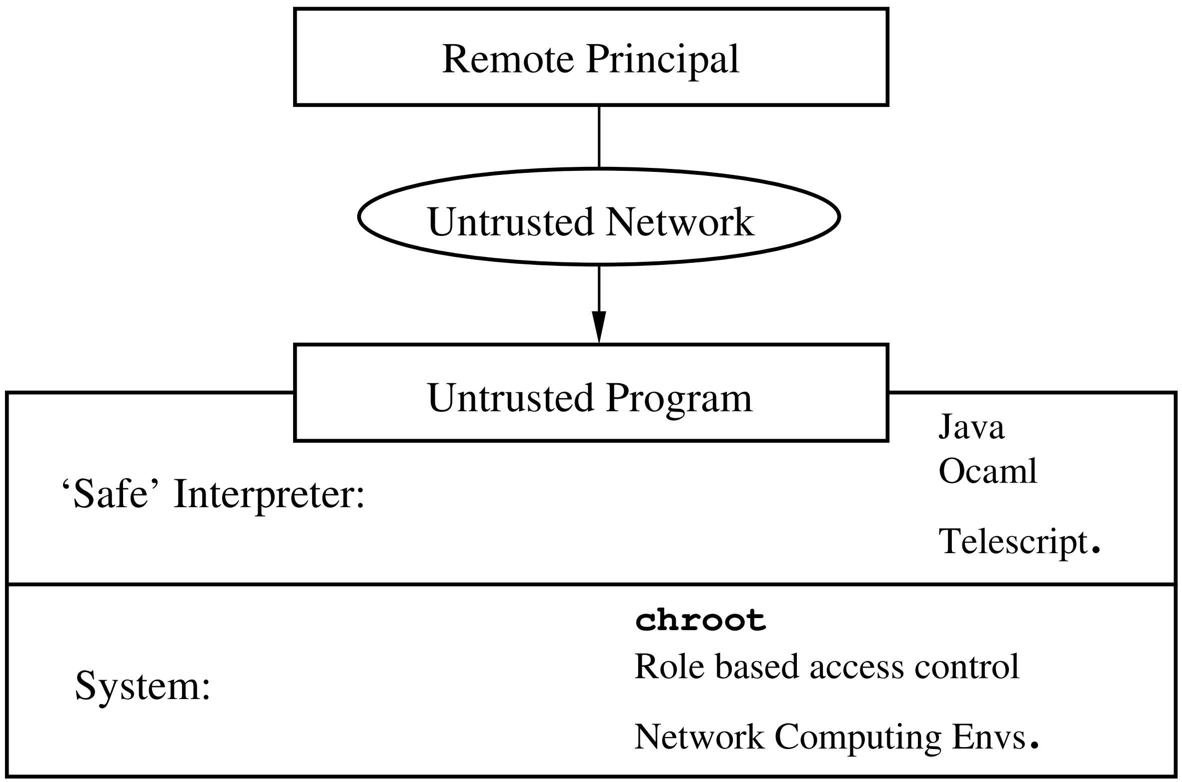
Building Systems That Flexibly Control Downloaded Executable Content

Michael Hicks

CIS 640

Mobile Program Architecture



Mobile Program Security

- Protect the *downloading principal* from actions performed by the *downloaded executable content* obtained from a *remote principal*
- Do so at least expense to application's *expressiveness* and *performance*

Approaches

- Limit content language's expressiveness (i.e. use a 'safe' interpreter)
- Restrict 'unsafe' operations
  - globally (all mobile programs)
  - selectively (per user, per program, etc.)
- Combine these two

Discretionary Access Control

- Specification of *access rights*
  - per system object
    - files
    - processes
    - network descriptors ...
  - per application object

Discretionary Access Control

- Enforcement of access rights
  - per remote principal
  - per application
  - per content-type
- Alteration of rights at runtime

Discretionary Access Control

- Advantages
  - More granular flexibility
- Disadvantages
  - Greater complexity
  - Reduced performance

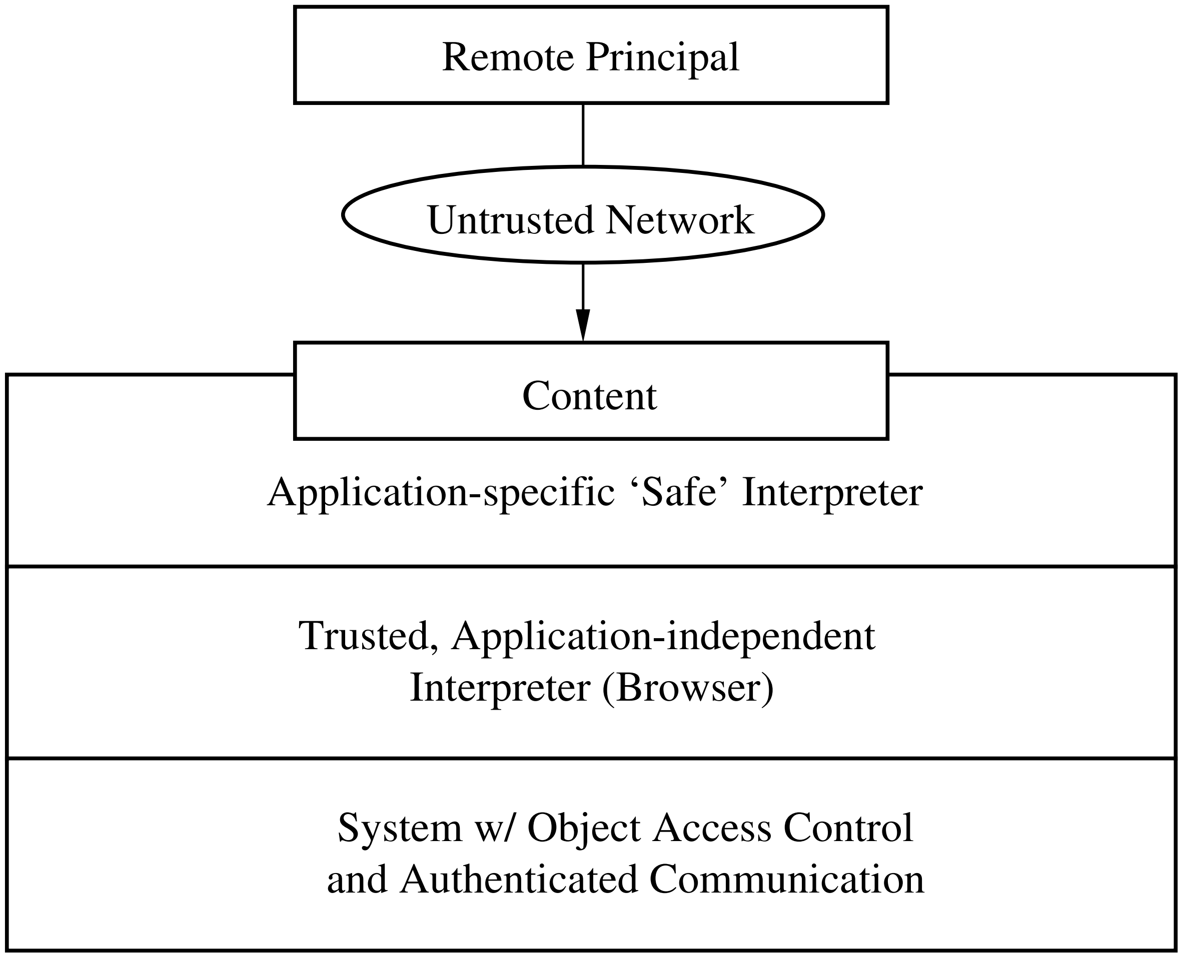
Terminology

- Principal
- Object
- Operation
- Domain

Expanded Architecture

- Application-specific interpreter
  - executes downloaded content
- Browser
  - controls access to system objects from interpreters
- System
  - control access outside of interpreter (I.e. spawned processes)

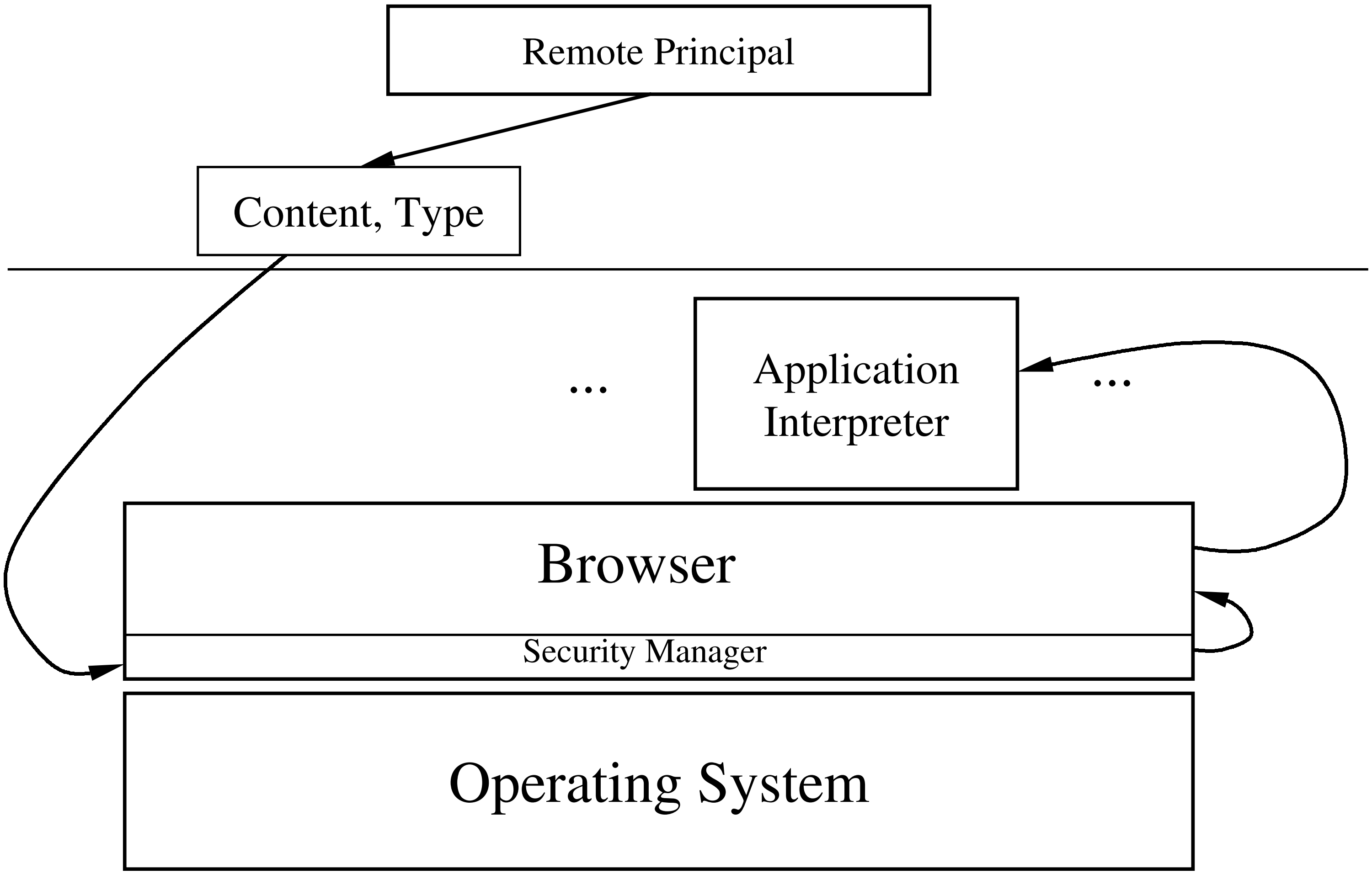
Expanded Architecture



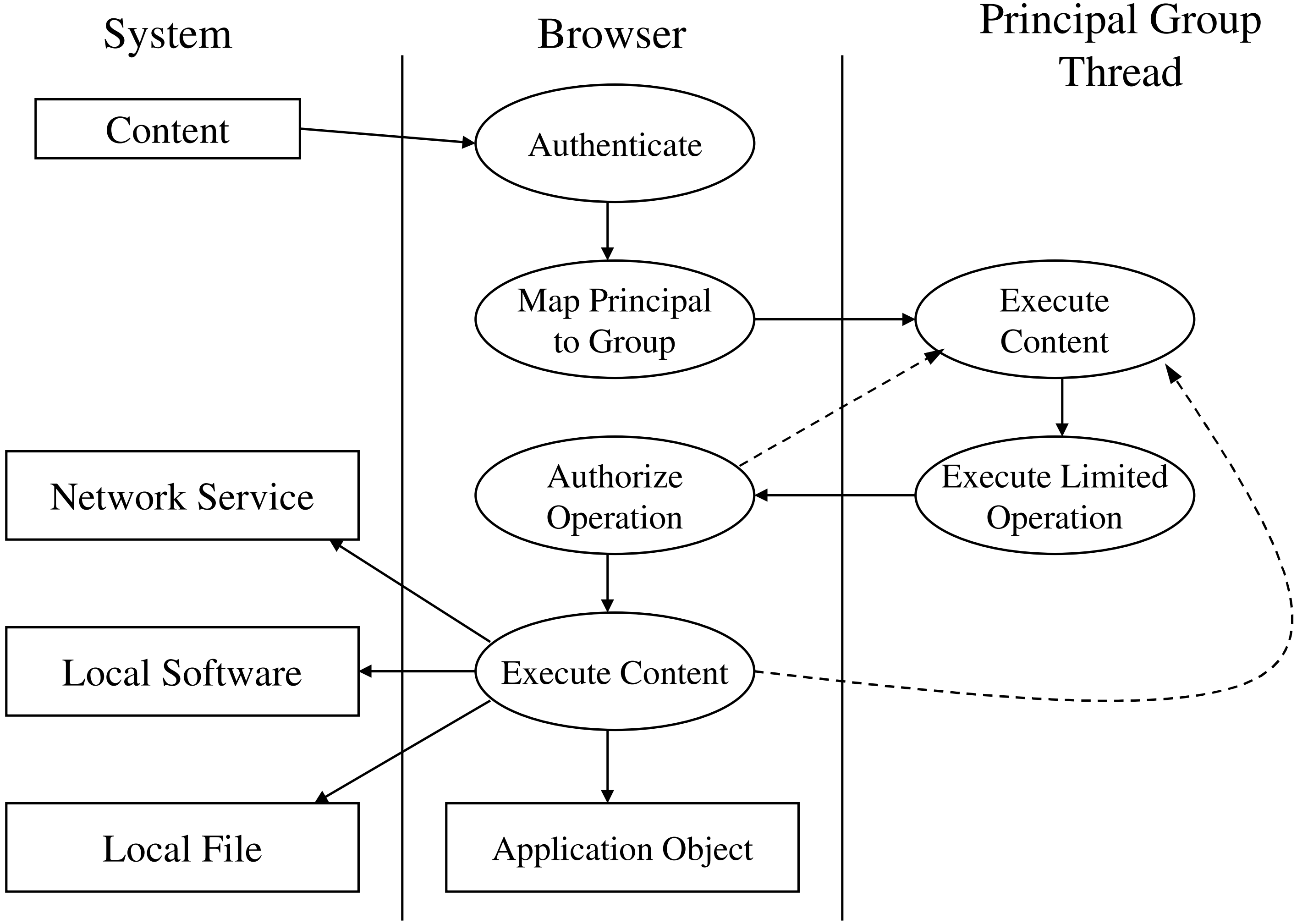
Executing Downloaded Content

- Download and verify content
- Determine access rights
- Execute content while enforcing the access rights

Executing Downloaded Content



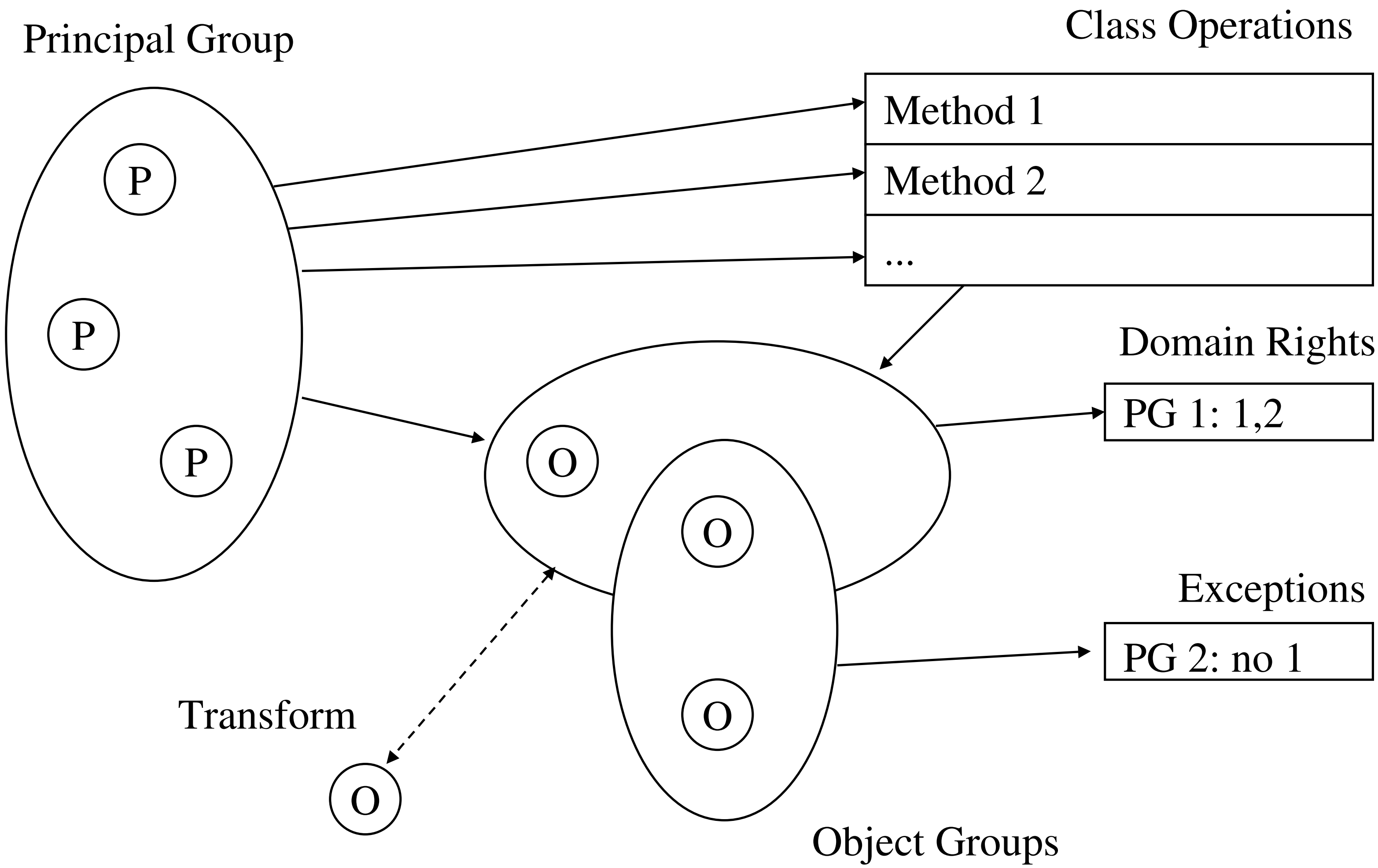
Executing Downloaded Content



Access Control Model

- Principal group
- Object group
- Access rights
  - Domain Rights
  - Exceptions
- Class Operation
- Transform

Example


Principal Group
Class Operations
P
P
P
Method 1
Method 2
...
Domain Rights
PG 1: 1,2
O
O
O
Exceptions
PG 2: no 1
Transform
O
Object Groups

Implementation

- Authentication mechanisms etc.)
- Tcl 7.5 interpreter
  – application-specific interpreter
  – browser
- DCE (based on Kerberos)
  – system enforcement

Implementation

- Authentication mechanisms etc.)
- Tcl 7.5 interpreter
  - application-specific interpreter
  - browser
- DCE (based on Kerberos)
  - system enforcement

Implementation

- Authentication mechanisms
  - public and secret key
- Tcl 7.5 interpreter
  - application-specific interpreter
  - browser
- DCE (based on Kerberos)
  - system enforcement

Downloading Content

- Content message (r,c,t,a,s):

  r = identity of remote principal

  c = content

  t = content type

  a = content's authentication information

    (n,i) = nonce, integrity info

  s = session identifier (optional)

Downloading Content

- Receive content message *cm*
- Authenticate *cm*
- If *r* is allowed to execute content *c's* interpreter *t* then
  - set up/update the session *s*
  - invoke *t* with *c*

Determining Access Rights

- Determine rights for interpreter t
- Determine rights for principal groups in t
  - static, and alterable by transforms
- Map remote principal to principal group

Enforcing Access Rights

- To system objects
  - ‘safe’ operation calls browser
  - authorization and execution occurs within browser
- To system software
  - authorize within browser
  - specify access rights to DCE
  - hand control to DCE (hand-waving here)

Criticisms

- Access descriptions are complicated
  - How to determine reasonable rights?
  - How to know which rights to change (where to put transforms in code)?
- Implementation details not concrete
  - How to use a non-TCL content interpreter?
  - How are class operations specified?
  - How are DCE access rights specified?

Criticisms

- No mention of performance
  - Cost of delegation
  - Cost of authentication/authorization
  - Cost of DCE/Kerberos

Conclusions

- Mechanisms developed for fine-grained access control:
  - per user, per application
  - at the application and system level
- Obviates need for arbitrary restrictions (no file I/O, etc.)
- Questions of performance and policy remain