Horton’s *Who Done It*?

Communicating Authority with Responsibility Tracking

Mark S. Miller        Google Research\(^1\)
Jed Donnelley        LBNL/NERSC
Alan H. Karp        HP Labs

Usenix HotSec Workshop,  August 7, 2007

\(^1\)Work done while at HP Labs
Communicating Object Access with Delegation

Initial Conditions:

Alice has: 1. A capability to send to Bob and
2. A capability to a document with chapters.
Capability Communication of the Document Reference

Alice sends a message to Bob containing a reference to the document.
Horton Magic: Bob Receives a Delegated Capability

Alice can’t act with Bob’s responsibility
Bob can’t act with Alice’s responsibility
Delegating Least Authority
Delegating Least Authority
Delegating Least Authority

A → foo(●) → B

C
Delegating Least Authority
Delegating Least Authority

-.Msgs are *only* means to cause effects
-Refs control authority
- Leverage OO patterns
Delegating Least Authority

-_msgs are only means to cause effects
-Refs control authority
-Leverage OO patterns
-Anonymous
Two styles, relative strengths

Program decisions
- Fine-grained
- Built for safety
- Least authority
- Virus resistant
- Authorization-based
  - Object-capabilities (ocaps)

Human decisions
- Large-grained
- Built for damage control
- Most responsibility
- Spam resistant
- Identity-based
  - ACLs
Two styles, relative strengths

Program decisions
- Fine-grained
- Built for safety
- Least authority
- Virus resistant
- Authorization-based

Human decisions
- Large-grained
- Built for damage control
- Most responsibility
- Spam resistant
- Identity-based

Object-capabilities (ocaps)

ACLs

Polaris, Plash
Bitfrost?
Two styles, relative strengths

Program decisions  Human decisions
Fine-grained        Large-grained
Built for safety    Built for damage control
Least authority     Most responsibility
Virus resistant     Spam resistant
Authorization-based Identity-based

Object-capabilities (ocaps) "Hybrid" Cap Systems (SCAP, Sys/38) ACLs
Two styles, relative strengths

Program decisions
Fine-grained
Built for safety
Least authority
Virus resistant
Authorization-based

Object-capabilities (ocaps)

Human decisions
Large-grained
Built for damage control
Most responsibility
Spam resistant
Identity-based

ACLs
Two styles, relative strengths

Program decisions  Human decisions
Fine-grained  Large-grained
Built for safety  Built for damage control
Least authority  Most responsibility
Virus resistant  Spam resistant
Authorization-based  Identity-based

Object-capabilities (ocaps)  ACLs

Horton
Can’t vet code or actions of each object.
Can’t vet code or actions of each object.
Can’t vet code or actions of each object.
Can’t vet code or actions of each object.
Can’t vet code or actions of each object.
Can’t vet code or actions of each object.
Can’t vet code or actions of each object.
Can’t vet code or actions of each object.
Can’t vet code or actions of each object.
Can’t vet code or actions of each object.

Aggregate into long-lived responsible identity.
Story Needs Four Characters

Alice & Bob
  • Old patterns for identity-based control: *identity tunnel*

Alice introduces Bob & Carol
  • Builds new relationships from old

Carol also hears of Bob from Dave
  • Corroborates Bob’s independence from Alice
Two-party intermediation

A message travels through an identity tunnel
Do I still use Bob’s services?
Bob, deliver foo() to B.
deliver("foo", [])
Do I still honor Alice’s requests?

foo()
Deliver `foo()` to B for Alice
Three-party intermediation

Build new relationships from old
b.foo(c)
A

Alice

A

foo( )

Bob

B

C

Carol
Carol, please provide Bob access to C
Alice needs tunnel for Bob

Carol, please provide Bob access to C

foo( )
Alice

A

foo(

Bob

B

Carol, please provide Bob access to C

Carol

Gift wrap it for Bob

C
Alice

A

foo()→

Bob

B

Carol, please provide Bob access to C

To Bob
From Carol

Carol

Gift wrap it for Bob

C

Intro(
Alice

```python
foo()  # Call to function

intro()  # Function call

Carol, please provide Bob access to C

To Bob From Carol
```

Bob

```python
return Bob's gift
```

Carol
Bob, deliver \texttt{foo()} to B with Carol's ( )
deliver("foo", [[], []])
Unwrap Carol’s gift from Alice

Carol

deliver("foo", [[], []])

To Bob
From Carol
Unwrap Carol’s gift from Alice.
Is Bob a pseudonym for Alice?
Four party intermediation

Only corroborating introductions let Alice shed blame
Better Identities than ACLs

Fully decentralized
  • No global administrator or name server

Track bilateral responsibility
  • For requests and for service
  • Also tracks delegation chain

Sybil resistant aggregation strategy
Corroboration-driven disaggregation
Conclusions

Delegate authority, bound to responsibility for using that authority.

Fine-grain least authority for safety.
Large-grain identities for damage control.

Reference implementations in Java & E: http://erights.org/download/horton/
Three-party intermediation

The details
Rights Amplification

- Inspired by PK
- Simple oo pattern
- No explicit crypto
- Can represent responsible identity
b.foo(c)
Carol, please provide Bob access to C
Carol, please provide Bob access to C.
Bob, please use Carol’s C
Alice
BeAlice
Bob
P1
deliver("foo", [])
tracked(···)
P2
introl(···)
S2
C
Alice
BeCarol
Bob
S1
Carol
BeBob
Bob
Alice
BeAlice
Make a stub for Bob’s use
Make a stub for Bob’s use.
wrap(s3, whoBob, beCarol)

deliver("foo", [])
tracked(•)

A
P1
P2

Bob

Alice

BeAlice

BeBob

Alice

Bob

S1

S2

S3

C

Bob

Carol

Carol
wrap(s3, whoBob, beCarol)
wrap(s3, whoBob, beCarol)
deliver("foo", [])
tracked(

pr

seal(●)
wrap(s3, whoBob, beCarol)
return gift
deliver("foo", [])
tracked(\ldots)

A
P1
P2

Bob
C
C
pr
S2
S3

Alice
BeAlice

Bob
BeBob

Alice
BeBob

Bob

Carol

Alice
BeAlice

Carol

B
unwrap(. , whoCarol, beBob)
unwrap(, whoCarol, beBob)

unseal( )
unwrap(, whoCarol, beBob)
unwrap( , whoCarol, beBob)

seal( )
unwrap(, whoCarol, beBob)
unwrap( , whoCarol, beBob)
unwrap(  , whoCarol, beBob)
unwrap( , whoCarol, beBob)
unwrap( , whoCarol, beBob)
makeProxy(..)
CapWiki with attribution
The Web: Good, Bad, and Ugly:

1. Good: Internet hypertext, wonderful!

2. Bad: Username/passwords for every site that has any sort of access control.

3. Ugly: Hard to share limited access to network objects. Hard to combine network objects with access restrictions.
Alice’s Domain

Sends:
- BobSend
- EveSend
- IvanSend
Alice’s Domain

Sends:
BobSend
EveSend
IvanSend

CapWiki:
CapWiki Stuff:
Concepts
Finances
Other

CapWiki Finances:
Investor
Market
Alice’s Domain

Sends:
- BobSend
- EveSend
- IvanSend

CapWiki:
- CapWiki Stuff:
  - Concepts
  - Finances
  - Other

Bob’s Domain

Receives:
- AliceReceive

Sends:
- AliceSend
- DaveSend

Here are the CapWiki:
- Finances
  - Dave

Daves’s Domain

Receives:
- * BobReceive

CapWiki

Finances:
- Investor
- Market
Here are the CapWiki: Finances Dave

Receives: BobReceive

Sends: AliceSend DaveSend

Receives: AliceReceive

Sends: BobSend EveSend IvanSend
Here are the CapWiki: Finances
Bob's Domain

Sends: AliceSend DaveSend

Receives: *AliceReceive

Investor Market

CapWiki: CapWiki Stuff: Concepts Finances Other

Alice's Domain

Sends: BobSend EveSend IvanSend

Daves's Domain

Alice...Bob...Dave

Here are the CapWiki: Finances
Dave

Receives: BobReceive
Better Web Access Control

• No more passwords – Send a <me>Send to a <service>Send. They know who you are, you know who they are.

• Side benefit – SPAM resistance. Don’t like a source of SPAM, cut it off to any delegation level.

• Principle Of Least Authority (POLA) sharing that can facilitate cross site services.
Is Carol a pseudonym for Alice?