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

b.foo(c)
Delegating Least Authority

A \( \rightarrow \) foo(\text{...}) \rightarrow B

A \rightarrow C

B \rightarrow C
Delegating Least Authority

A → C → B
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)
Polaris, Plash
Bitfrost?

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

“Hybrid” Cap Systems (SCAP, Sys/38)
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
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)

Horton

ACLs
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.
Alice

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
b.foo()
Do I still use Bob’s services?
Bob, deliver \texttt{foo()} to B
Do I still honor Alice’s requests?
Deliver `foo()` to B for Alice
Three-party intermediation

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

Alice

foo()  

Bob

B

C

Carol
Carol, please provide Bob access to C

A

B

Alice

Bob

C

Carol

foo(·)
Alice, please provide Bob access to C.
Carol, please provide Bob access to C

Gift wrap it for Bob
Alice

foos(·)

Bob

Gift wrap it for Bob

Carol, please provide Bob access to C

To Bob
From Carol
Carol, please provide Bob access to C

To Bob
From Carol

return Bob
's
gift

Alice

Bob

A

B

C

To Bob
From Carol

return Bob’s gift

intro( )

foo( )
Bob, deliver foo() to B with Carol's ()
Alice

Bob

Carol

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

To Bob From Carol
Alice

To Bob

From Carol

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

Carol

Unwrap Carol’s gift from Alice

To Bob From Carol
Unwrap Carol’s gift from Alice

foo( )

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
Make a stub for Bob's use
Make a stub for Bob’s use
Gift wrap it for Bob
wrap(s3, whoBob, beCarol)

deliver("foo", [])

tracked()
wrap(s3, whoBob, beCarol)
deliver("foo", [])
tracked(•)
pr
seal(•)
wrap(s3, whoBob, beCarol)
unwrap( , whoCarol, beBob)
unwrap(, whoCarol, beBob)
unwrap(, whoCarol, beBob)
unwrap(, whoCarol, beBob)
unwrap( , whoCarol, beBob)
unwrap( , whoCarol, beBob)
unwrap( , whoCarol, beBob)
unwrap( , whoCarol, beBob)
unwrap(, whoCarol, beBob)
unwrap( , whoCarol, beBob)
makeProxy(..)
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

Bob’s Domain

Receives:
*AliceReceive

Sends:
AliceSend
DaveSend
Alice’s Domain

Sends:
- BobSend
- EveSend
- IvanSend

CapWiki: CapWiki Stuff: Concepts Finances Other

Bob’s Domain

Receives:
- AliceReceive

Sends:
- AliceSend
- DaveSend

CapWiki Finances: Investor Market
Here are the CapWiki: Finances

Dave's Domain

Sends: BobSend
EveSend
IvanSend

Receives: AliceReceive

Sends: AliceSend
DaveSend

Receives: BobReceive

Alice's Domain

Sends: BobSend
EveSend
IvanSend

CapWiki: CapWiki Stuff: Concepts
Finances
Other

CapWiki Finances: Investor
Market

Bob's Domain

Receives: AliceReceive

Sends: AliceSend
DaveSend

Here are the CapWiki: Finances
Dave
Here are the CapWiki: Finances

CapWiki: CapWiki Stuff: Concepts Finances Other

Sends:

BobSend
EveSend
IvanSend

Receives:

*AliceReceive

Sends:

AliceSend
DaveSend

Here are the CapWiki: Finances

Daves’s Domain

Bobs’s Domain

Alice’s Domain

CapWiki: Investor Market

Sends:

BobSend
EveSend
IvanSend

Receives:

*AliceReceive

Sends:

AliceSend
DaveSend

Here are the CapWiki: Finances

Dave
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.