Authority Propagation Models: PoP vs PoC and the Confused Deputy Problem

Nicola Gallo

1 December 2025

1 Model

Let P be a set of principals (e.g., U, PDF, $Storage$). Let O be a set of operations (e.g., convert, delete), and R a set of resources.

A privilege is a pair $(o, r) \in O \times R$. Each principal p has a privilege set:

$$Priv(p) \subseteq O \times R.$$

A request is a message req sent between principals and may contain a payload.

2 Confused Deputy

Definition 1 (Confused Deputy)**.** *A confused deputy occurs when there exist principals U (user) and D (deputy) such that:*

1. $(o, r) \notin Priv(U)$,
2. $(o, r) \in Priv(D)$,
3. *U sends a request req to D,*
4. *as a consequence of req, D executes (o, r).*

This definition does not depend on implementation details, only on the mismatch of authority and causality.

3 Proof-of-Possession (PoP)

A token t grants a set of privileges:

$$Auth(t) \subseteq O \times R.$$

PoP Semantics: Possession implies usability: if a principal holds t, it may exercise all $(o, r) \in Auth(t)$.

PoP systems do not constrain authority by causality or provenance.

3.1 Vulnerability Condition

Assume:

$$(\text{delete}, r) \notin Priv(U) \tag{H1}$$

$$(\text{delete}, r) \in Priv(PDF) \tag{H2}$$

Further assume:

> The payload of a request may influence control flow in PDF, including code paths that call internal privileges such as delete.

Theorem 1 (PoP admits confused deputy)**.** *Under assumptions (H1)–(H2), there exists a request req such that PDF executes $(delete, r)$ as a result of processing req.*

Proof. Since $(\text{delete}, r) \in Priv(PDF)$ (H2), some code path in PDF invokes Storage.delete(r). Because processing is influenced by payload, there exists an adversarial payload that triggers that path. Since U lacks (delete, r) (H1), and PDF acts on behalf of U, conditions for a confused deputy are satisfied. □

This applies to OAuth tokens and sealed capabilities: sealing protects transport, not authority semantics.

4 Proof-of-Continuity (PoC / PIC)

Execution is modeled as a causal chain of hops:

$$p_0 \to p_1 \to \cdots \to p_n,$$

with $p_0 = U$.

Each hop transfers a privilege subset:

$$ops_i \subseteq O \times R$$

and must satisfy:

$$ops_{i+1} \subseteq ops_i. \tag{C1}$$

Let π be a verifiable sequence:

$$\pi = \langle (p_0, ops_0), (p_1, ops_1), \ldots, (p_n, ops_n) \rangle.$$

4.1 Authorization Rule

The final service authorizes (o, r) if and only if:

$$(o, r) \in \bigcap_{i=0}^{n} ops_i$$

and π is valid.

Since the chain is monotonic decreasing,

$$\bigcap_{i=0}^{n} ops_i = ops_n.$$

Thus no hop may acquire new authority not present at the origin.

5 Safety Property

Definition 2 (Origin-bounded authority)**.** *A model enforces origin-bounded authority if every executable privilege at hop n is a privilege originally granted by hop 0.*

Theorem 2 (PoC prevents confused deputy)**.** *If $ops_0 = \{(convert, r)\}$ and (C1) holds for every hop, then $(delete, r)$ can never be authorized at hop n.*

Proof. Authorization requires:

$$(\text{delete}, r) \in \bigcap_{i=0}^{n} ops_i.$$

But $(\text{delete}, r) \notin ops_0$ and each $ops_{i+1} \subseteq ops_i$. Therefore $(\text{delete}, r) \notin ops_i$ for all i, hence not in the intersection. The request is rejected. □

6 Discussion

PoP systems conflate authority and possession, enabling confused deputy attacks whenever privileged internal code paths exist. PIC/PoC systems propagate only non-expansive subsets of authority, ensuring that downstream services cannot perform operations not explicitly authorized at the origin.