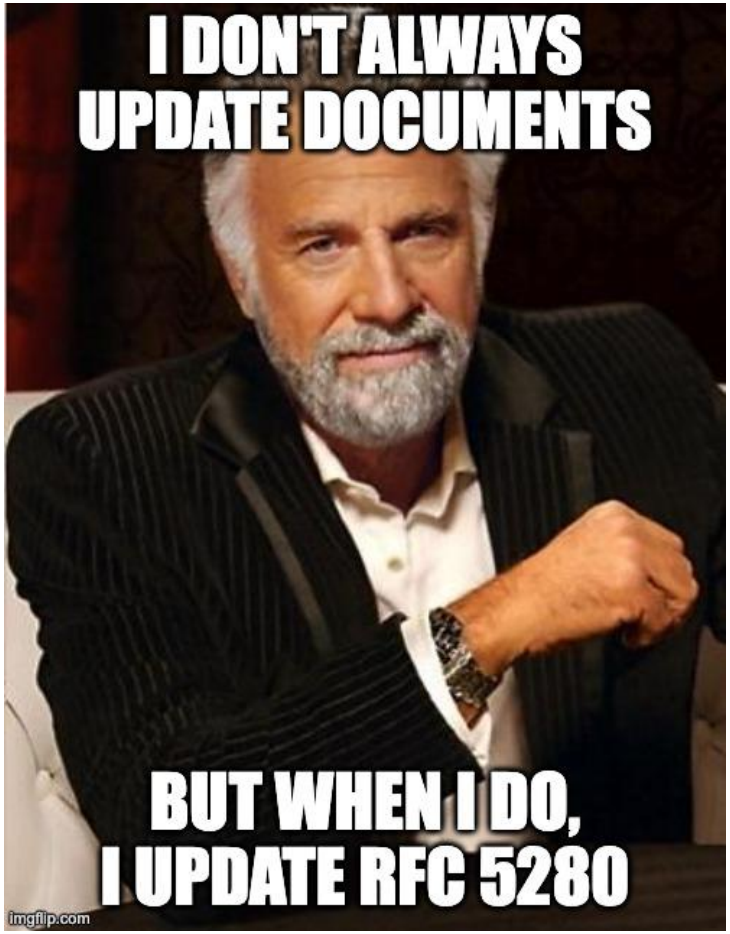
Updates to X.509 Policy Validation

draft-davidben-x509-policy-graph


I DON'T ALWAYS
UPDATE DOCUMENTS
BUT WHEN I DO,
I UPDATE RFC 5280
imgflip.com

X.509 policy validation

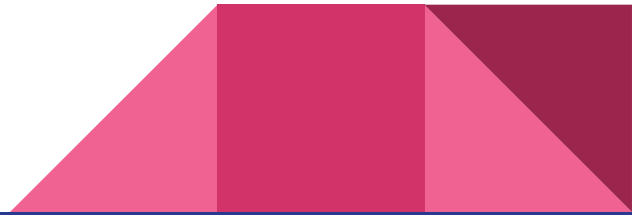
Certificate policies (RFC 5280, section 4.2.1.4)

Policies asserted by the certificate, act as constraints in CAs

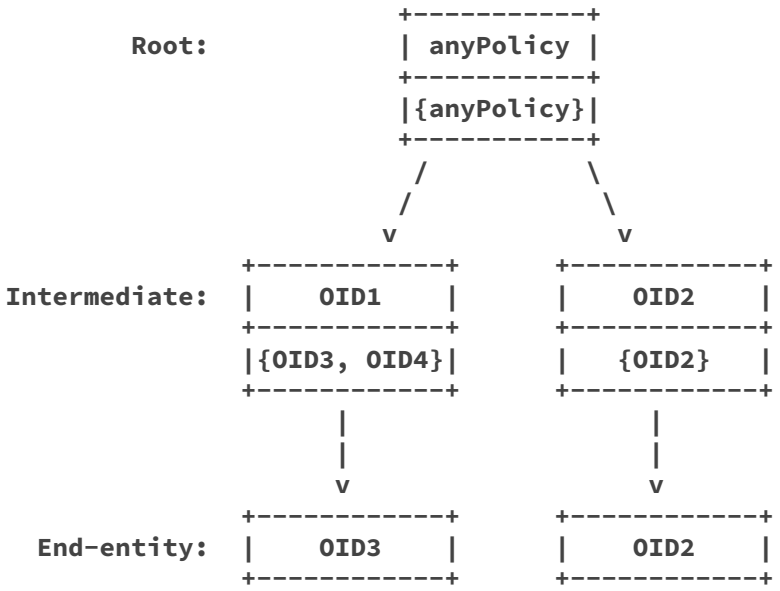
Policy mappings (RFC 5280, section 4.2.1.5)

Allows CAs to *rename* policy OIDs down the chain

(Other complexity omitted here. anyPolicy, user-initial-policy-set, inhibit anyPolicy, inhibit mappings, require explicit policy, ...)



Policy trees



Intermediate

certificate policies: OID1, OID2, OID5

policy mappings: OID1 ↦ OID3, OID1 ↦ OID4

End-entity

certificate policies: OID2, OID3, OID6

Follow this very simple algorithm...

(a) valid_policy_tree: A tree of certificate policies with their optional qualifiers; each of the leaves of the tree represents a valid policy at this stage in the certification path validation. If valid policies exist at this stage in the certification path validation, th equal to been proc in the ce NULL. O ceases.

(d) explicit_policy: an in valid_policy_tree is re number of non-self-issu this requirement is imp decreased, but may not certificate in the path valid_policy_tree, a la requirement. If initia initial value is 0, oth

Cooper, et al. Standa

RFC 5280 PKIX Certifica

(e) inhibit_anyPolicy: an anyPolicy policy identi integer indicates the n to be processed before certificate other than certificate, is ignored decreased, but may not certificate in the path later certificate canno inhibit is set, then th initial value is n+1.

(f) policy_mapping: an int is permitted. The inte issued certificates to inhibited. Once set, t not be increased. That specifies that policy m overridden by a later c mapping-inhibit is set, otherwise the initial v

(d) If the certificate policies certificate and the valid_p the policy information by p order:

(1) For each policy P not eq certificate policies ext for policy P and P-Q den P. Perform the followin

(i) For each node of dep where P-OID is in th child node as follow set the qualifier_se expected_policy_set {P-OID}.

For example, conside of depth i-1 where t White}. Assume the Silver appear in the certificate i. The Silver policy is not node of depth i for shown as Figure 4.

Figure 4. Processing an Exact Match

to the value from the expected_policy_set in the parent node, set the qualifier_set to AP-Q, and set the expected_policy_set to the value in the valid_policy from this node.

For example, consider a valid_policy_tree with a node of depth i-1 where the expected_policy_set is {Gold, Silver}. Assume anyPolicy appears in the certificate policies extension of certificate i with no policy qualifiers, but Gold and Silver do not appear. This rule will generate two child nodes of depth i, one for each policy. The result is shown below as Figure 6.

Figure 6. Processing Unmatched Pol Certificate Policies Extension Spec

(3) If there is a node in the valid_pol or less without any child nodes, de this step until there are no nodes without children.

For example, consider the valid_pol Figure 7 below. The two nodes at d marked with an 'X' have no children Applying this rule to the resulting node at depth i-2 that is marked wi In the resulting tree, there are no less without children, and this ste

(ii) If there was no mat valid_policy_tree i the valid_policy an the following value set the qualifier_s expected_policy_set

For example, consid of depth i-1 where Assume the certific in the certificate i. The Gold policy Silver policy has t Silver were not mat generate two child policy. The result

Figure 5. Processing Unmatched Policies when a Leaf Node Specifies anyPolicy

(2) If the certificate policies extension includes the policy anyPolicy with the qualifier set AP-Q and either (a) inhibit_anyPolicy is greater than 0 or (b) i<n and the certificate is self-issued, then:

For each node in the valid_policy_tree of depth i-1, for each value in the expected_policy_set (including anyPolicy) that does not appear in a child node, create a child node with the following values: set the valid_policy

(e) If the certificate policies extension is not present, set the valid_policy_tree to NULL.

(f) Verify that either explicit_policy is greater than 0 or the valid_policy_tree is not equal to NULL;

If any of steps (a), (b), (c), or (f) fails, the procedure terminates, returning a failure indication and an appropriate reason.

If i is not equal to n, continue by performing the preparatory steps listed in Section 6.1.4. If i is equal to n, perform the wrap-up steps listed in Section 6.1.5.

Figure 7. Pruning the valid_policy_tree

6.1.4. Preparation for Certificate i+1

...continued

(h) If certificate i is not self-issued:

(1) If explicit_policy is not 0, decrement expl... 1.

(2) If policy_mapping is not 0, decrement polic...

(3) If inhibit_anyPolicy is not 0, decrement in... by 1.

(i) If a policy constraints extension is included ... certificate, modify the explicit_policy and po... state variables as follows:

(1) If requireExplicitPolicy is present and is ... explicit_policy, set explicit_policy to the ... requireExplicitPolicy.

(2) If inhibitPolicyMapping is present and is l... policy_mapping, set policy_mapping to the v... inhibitPolicyMapping.

(j) If the inhibitAnyPolicy extension is included in the certificate and is less than inhibit anyPolicy, set

(a) If explicit_policy is not 0, decrement explicit_policy by 1.

(b) If a policy constraints extension is included in the certificate and requireExplicitPolicy is present and has a value of 0, set the explicit_policy state variable to 0.

(g) Calculate the intersection of the valid_policy_tree and the user-initial-policy-set, as follows:

(i) If the valid_policy_tree is NULL, the intersection is NULL.

(ii) If the valid_policy_tree is not NULL and the user-initial-policy-set is any-policy, the intersection is the entire valid_policy_tree.

(iii) If the valid_policy_tree is not NULL and the user-initial-policy-set is not any-policy, calculate the intersection o... initial-policy...

1. Determine the... have a valid_... valid_policy_...

2. If the valid_... valid_policy_... policy-set an... all its child...

3. If the valid_policy_tree includes a node of depth n with the valid_policy anyPolicy and the user-initial-policy-set is not any-policy, perform the following steps:

a. Set P-Q to the qualifier_set in the node of depth n with valid_policy anyPolicy.

b. For each P-OID in the user-initial-policy-set that is not the valid_policy of a node in the valid_policy_node_set, create a child node whose parent is the node of depth n-1 with the valid_policy anyPolicy. Set the values in the child node as follows: set the valid_policy to P-OID, set the qualifier_set to P-Q, and set the expected_policy_set to {P-OID}.

c. Delete the node of depth n with the valid_policy anyPolicy.

4. If there is a node in the valid_policy_tree of depth n-1 or less without any child nodes, delete that node. Repeat this step until there are no nodes of depth n-1 or less without children.

If either (1) the value of explicit_policy variable is greater than zero or (2) the valid_policy_tree is not NULL, then path processing has succeeded.

6.1.4. Preparation for Certif...

To prepare for processin... following steps for cert...

(a) If a policy mapping ... special value anyPo... issuerDomainPolicy ...

(b) If a policy mapping ... issuerDomainPolicy ...

(1) If the policy_ma... node in the vali... valid_policy, se... subjectDomainPol... equivalent to ID...

If no node of de... valid_policy of ... valid_policy of ... the node of dept... as follows:

(i) set the val...

(ii) set the qua... policy anyP... extension o...

(iii) set the exp... subjectDoma... equivalent ...

(2) If the policy_ma...

(i) delete each ... where ID-P ...

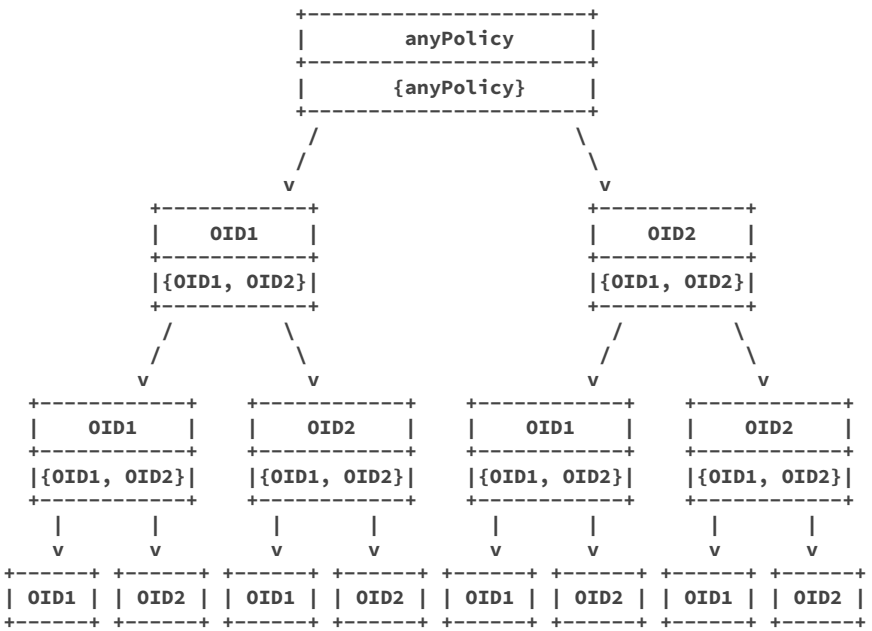
(ii) If there is a node in the v... i-1 or less without any chi... node. Repeat this step unt... depth i-1 or less without c...

Duplicate nodes

(1) **For each policy P** not equal to anyPolicy **in the certificate policies extension**, let P-OID denote the OID for policy P and P-Q denote the qualifier set for policy P. Perform the following steps in order:

(i) **For each node of depth i-1 in the valid_policy_tree** where P-OID is in the expected_policy_set, **create a child node** as follows: set the valid_policy to P-OID, set the qualifier_set to P-Q, and set the expected_policy_set to {P-OID}.

X.509 policy trees grow exponentially



Certificate policies

OID1, OID2

Policy mappings

OID1 ↦ OID1, OID1 ↦ OID2,

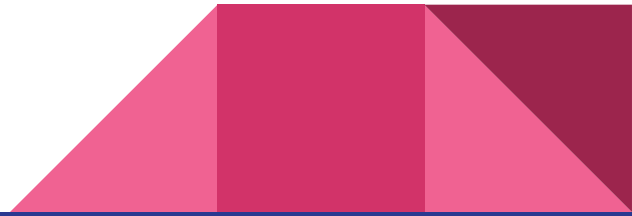
OID2 ↦ OID1, OID2 ↦ OID2

Repeat

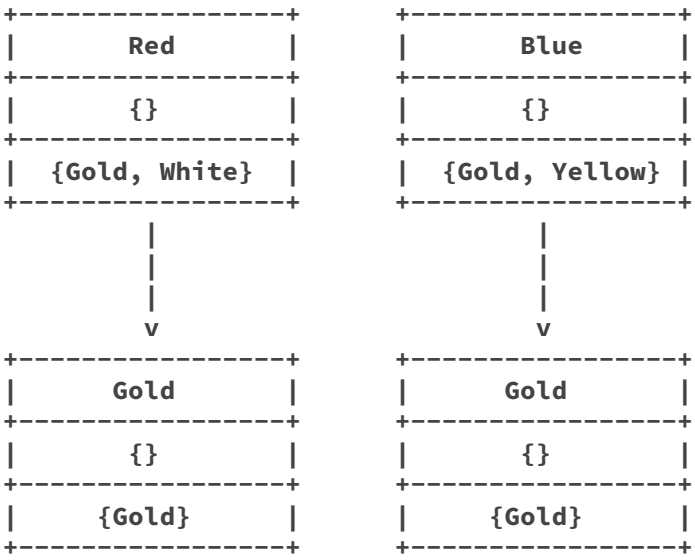
Denial of service vulnerability

Hosting providers may evaluate untrusted PKIs

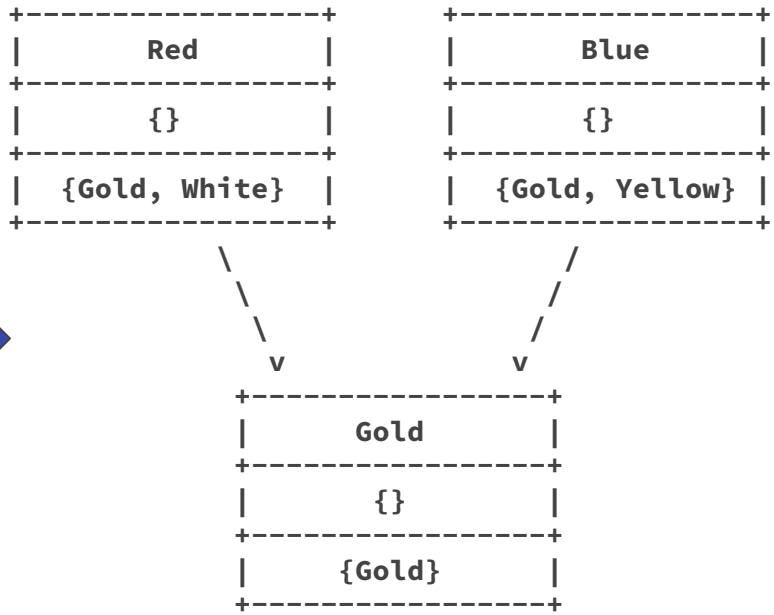
A trusted CA may issue a constrained intermediate to an untrusted party



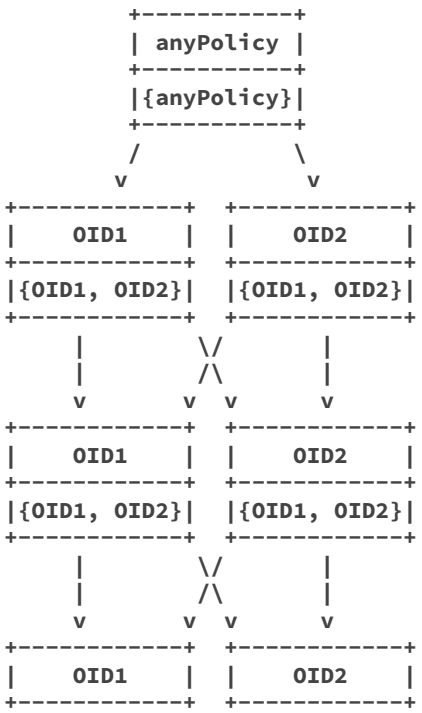
Tree



Directed acyclic graph



Policy graphs grow linearly



Certificate policies

OID1, OID2

Policy mappings

OID1 ↦ OID1, OID1 ↦ OID2,

OID2 ↦ OID1, OID2 ↦ OID2

Repeat

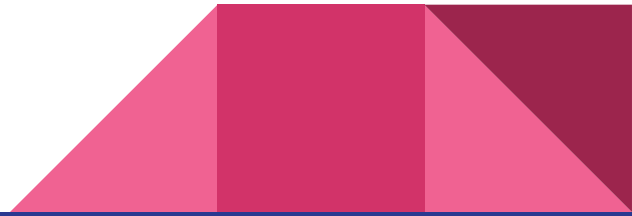
draft-davidben-x509-policy-graph

Updates RFC 5280 with the new algorithm

Updates verification output

Discusses other mitigations

- Limit certificate depth
- Limit policy tree size
- Inhibit policy mapping
- Disable policy checking
- Verify signatures first (partial mitigation only)



Questions?

draft-davidben-x509-policy-graph