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
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

Root: anyPolicy

Intermediate: OID1, OID2, OID5
Policies: OID1
Mappings: OID1 → OID3, OID1 → OID4

End-entity: 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.

(b) certified_policy: The certificate policy of the certificate in the current path.

(c) unexpanded_policies: A list of unexpanded certificate policies.

(d) if the certificate is revoked, then return NULL.

(e) If there is an explicit policy, then check it; if it applies, return the policy information.

(f) error.

(g) If any other policy succeeds, then return the result of that policy.

(h) If the policy is not found, then return NULL.

(i) If the policy is not found, then return NULL.

(j) Process the certificate policies in the policy tree, starting with the policy at the root.

(k) For each policy P, check if it is a valid_policy_tree.

(l) For each policy P, check if it is a valid_policy_tree.

(m) For each policy P, check if it is a valid_policy_tree.

(n) For each policy P, check if it is a valid_policy_tree.

(o) For each policy P, check if it is a valid_policy_tree.

(p) For each policy P, check if it is a valid_policy_tree.

(q) For each policy P, check if it is a valid_policy_tree.

(r) For each policy P, check if it is a valid_policy_tree.

(s) For each policy P, check if it is a valid_policy_tree.

(t) For each policy P, check if it is a valid_policy_tree.

(u) For each policy P, check if it is a valid_policy_tree.

(v) For each policy P, check if it is a valid_policy_tree.

(w) For each policy P, check if it is a valid_policy_tree.

(x) For each policy P, check if it is a valid_policy_tree.

(y) For each policy P, check if it is a valid_policy_tree.

(z) For each policy P, check if it is a valid_policy_tree.

(A) For each policy P, check if it is a valid_policy_tree.

(B) For each policy P, check if it is a valid_policy_tree.

(C) For each policy P, check if it is a valid_policy_tree.

(D) For each policy P, check if it is a valid_policy_tree.

(E) For each policy P, check if it is a valid_policy_tree.

(F) For each policy P, check if it is a valid_policy_tree.

(G) For each policy P, check if it is a valid_policy_tree.

(H) For each policy P, check if it is a valid_policy_tree.

(I) For each policy P, check if it is a valid_policy_tree.

(J) For each policy P, check if it is a valid_policy_tree.

(K) For each policy P, check if it is a valid_policy_tree.

(L) For each policy P, check if it is a valid_policy_tree.

(M) For each policy P, check if it is a valid_policy_tree.

(N) For each policy P, check if it is a valid_policy_tree.

(O) For each policy P, check if it is a valid_policy_tree.

(P) For each policy P, check if it is a valid_policy_tree.

(Q) For each policy P, check if it is a valid_policy_tree.

(R) For each policy P, check if it is a valid_policy_tree.

(S) For each policy P, check if it is a valid_policy_tree.

(T) For each policy P, check if it is a valid_policy_tree.

(U) For each policy P, check if it is a valid_policy_tree.

(V) For each policy P, check if it is a valid_policy_tree.

(W) For each policy P, check if it is a valid_policy_tree.

(X) For each policy P, check if it is a valid_policy_tree.

(Y) For each policy P, check if it is a valid_policy_tree.

(Z) For each policy P, check if it is a valid_policy_tree.

(1) For each policy P, check if it is a valid_policy_tree.

(2) For each policy P, check if it is a valid_policy_tree.

(3) For each policy P, check if it is a valid_policy_tree.

(4) For each policy P, check if it is a valid_policy_tree.

(5) For each policy P, check if it is a valid_policy_tree.

(6) For each policy P, check if it is a valid_policy_tree.

(7) For each policy P, check if it is a valid_policy_tree.

(8) For each policy P, check if it is a valid_policy_tree.

(9) For each policy P, check if it is a valid_policy_tree.

(10) For each policy P, check if it is a valid_policy_tree.

(11) For each policy P, check if it is a valid_policy_tree.

(12) For each policy P, check if it is a valid_policy_tree.

(13) For each policy P, check if it is a valid_policy_tree.

(14) For each policy P, check if it is a valid_policy_tree.

(15) For each policy P, check if it is a valid_policy_tree.

(16) For each policy P, check if it is a valid_policy_tree.

(17) For each policy P, check if it is a valid_policy_tree.

(18) For each policy P, check if it is a valid_policy_tree.

(19) For each policy P, check if it is a valid_policy_tree.

(20) For each policy P, check if it is a valid_policy_tree.

(21) For each policy P, check if it is a valid_policy_tree.

(22) For each policy P, check if it is a valid_policy_tree.

(23) For each policy P, check if it is a valid_policy_tree.

(24) For each policy P, check if it is a valid_policy_tree.

(25) For each policy P, check if it is a valid_policy_tree.

(26) For each policy P, check if it is a valid_policy_tree.

(27) For each policy P, check if it is a valid_policy_tree.

(28) For each policy P, check if it is a valid_policy_tree.

(29) For each policy P, check if it is a valid_policy_tree.

(30) For each policy P, check if it is a valid_policy_tree.

(31) For each policy P, check if it is a valid_policy_tree.

(32) For each policy P, check if it is a valid_policy_tree.

(33) For each policy P, check if it is a valid_policy_tree.

(34) For each policy P, check if it is a valid_policy_tree.

(35) For each policy P, check if it is a valid_policy_tree.

(36) For each policy P, check if it is a valid_policy_tree.

(37) For each policy P, check if it is a valid_policy_tree.

(38) For each policy P, check if it is a valid_policy_tree.

(39) For each policy P, check if it is a valid_policy_tree.

(40) For each policy P, check if it is a valid_policy_tree.

(41) For each policy P, check if it is a valid_policy_tree.

(42) For each policy P, check if it is a valid_policy_tree.

(43) For each policy P, check if it is a valid_policy_tree.

(44) For each policy P, check if it is a valid_policy_tree.

(45) For each policy P, check if it is a valid_policy_tree.

(46) For each policy P, check if it is a valid_policy_tree.

(47) For each policy P, check if it is a valid_policy_tree.

(48) For each policy P, check if it is a valid_policy_tree.

(49) For each policy P, check if it is a valid_policy_tree.

(50) For each policy P, check if it is a valid_policy_tree.

(51) For each policy P, check if it is a valid_policy_tree.

(52) For each policy P, check if it is a valid_policy_tree.

(53) For each policy P, check if it is a valid_policy_tree.

(54) For each policy P, check if it is a valid_policy_tree.

(55) For each policy P, check if it is a valid_policy_tree.

(56) For each policy P, check if it is a valid_policy_tree.

(57) For each policy P, check if it is a valid_policy_tree.

(58) For each policy P, check if it is a valid_policy_tree.

(59) For each policy P, check if it is a valid_policy_tree.

(60) For each policy P, check if it is a valid_policy_tree.

(61) For each policy P, check if it is a valid_policy_tree.

(62) For each policy P, check if it is a valid_policy_tree.

(63) For each policy P, check if it is a valid_policy_tree.

(64) For each policy P, check if it is a valid_policy_tree.

(65) For each policy P, check if it is a valid_policy_tree.

(66) For each policy P, check if it is a valid_policy_tree.

(67) For each policy P, check if it is a valid_policy_tree.

(68) For each policy P, check if it is a valid_policy_tree.

(69) For each policy P, check if it is a valid_policy_tree.

(70) For each policy P, check if it is a valid_policy_tree.

(71) For each policy P, check if it is a valid_policy_tree.

(72) For each policy P, check if it is a valid_policy_tree.

(73) For each policy P, check if it is a valid_policy_tree.

(74) For each policy P, check if it is a valid_policy_tree.

(75) For each policy P, check if it is a valid_policy_tree.

(76) For each policy P, check if it is a valid_policy_tree.

(77) For each policy P, check if it is a valid_policy_tree.

(78) For each policy P, check if it is a valid_policy_tree.

(79) For each policy P, check if it is a valid_policy_tree.

(80) For each policy P, check if it is a valid_policy_tree.

(81) For each policy P, check if it is a valid_policy_tree.

(82) For each policy P, check if it is a valid_policy_tree.

(83) For each policy P, check if it is a valid_policy_tree.

(84) For each policy P, check if it is a valid_policy_tree.

(85) For each policy P, check if it is a valid_policy_tree.

(86) For each policy P, check if it is a valid_policy_tree.

(87) For each policy P, check if it is a valid_policy_tree.

(88) For each policy P, check if it is a valid_policy_tree.

(89) For each policy P, check if it is a valid_policy_tree.

(90) For each policy P, check if it is a valid_policy_tree.

(91) For each policy P, check if it is a valid_policy_tree.

(92) For each policy P, check if it is a valid_policy_tree.

(93) For each policy P, check if it is a valid_policy_tree.

(94) For each policy P, check if it is a valid_policy_tree.

(95) For each policy P, check if it is a valid_policy_tree.

(96) For each policy P, check if it is a valid_policy_tree.
...continued

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

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

(2) If policy_mapping is not 0, decrement policy_mapping by 1.

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

If a policy constraints extension is included in the certificate, modify the explicit_policy and policy_mapping state variables as follows:

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

(2) If inhibitPolicyMapping is present and is true, set policy_mapping to
inhibitPolicyMapping.

(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 of the valid_policy_tree with the valid_policy_set
and the user-initial-policy-set, as follows:

1. Determine the depth of the node in the valid_policy_tree
that corresponds to the user-initial-policy-set.

2. If the valid_policy_tree does not contain a node of
the specified depth, the intersection is NULL.

3. If the valid_policy_tree includes a node of depth n
with the valid_policy anyPolicy and the user-initial-
policy-set is 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.
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
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