# Configuring Access Control Policies

draft-petithuguenin-p2psip-access-control

Marc Petit-Huguenin
03/31/2011

# What is an Access Control Policy?

Definition of Access Control Policies (ACP) in p2psip-base (section 6.3):

"...defines whether a request from a given node to operate on a given value should succeed or fail."

Four ACP defined in p2psip-base:

- USER-MATCH
- NODE-MATCH
- USER-NODE-MATCH
- NODE-MULTIPLE

# More new ACP than anticipated

The paragraph continues: "[i]t is anticipated that only a small number of generic access control policies are required."

But in fact more than 50% of Usages require a new ACP

| I-D | New ACPs |
|---|---|
| draft-maenpaa-p2psip-service-discovery | NODE-ID-MATCH |
| draft-knauf-p2psip-disco-02 | |
| draft-knauf-p2psip-share-00 | USER-CHAIN-ACL<br>USER-PATTERN-MATCH |
| draft-liao-p2psip-dcmp-01 | |
| draft-peng-p2psip-snmp-00 | |
| draft-rosenberg-dispatch-vipr-reload-usage-03 | (no name) |
| draft-xiao-ppsp-reload-distributed-tracker-01 | |

# So, what's the problem?

Peers not implementing a new ACP will be kicked out of the overlay when deployed.

Peers have no incentive to upgrade if they are not using the new feature, but are just storing data for people using the new feature.

Difficult to have all implementations in the same overlay at the same level.

# Proposal

The proposal is to carry portable code for the new ACP in the configuration file.

Because the configuration file is signed by the operators of the overlay, it is not as scary as it looks.

ACP code uses ECMAScript (aka JavaScript): popularity (5$^{th}$ on http://langpop.com/), multiple FOSS implementations available.

# Example : USER-MATCH

```
String.prototype['bytes'] = function() {
        var bytes = [];
        for (var i = 0; i < this.length; i++) {
            bytes[i] = this.charCodeAt(i);
        }
        return bytes;
    };

return
  resource.equalsHash(signature.user_name.bytes());
```

# Example: NODE-MULTIPLE

```
Number.prototype['width'] = function(w) {
    var bytes = [];
    for (var i = 0; i < w; i++) {
        bytes[i] = (this >>> ((w - i - 1) * 8)) & 255;
    }
    return bytes;
};

for (var i = 0; i < kind.params['max-node-multiple']; i++) {
    if (resource.equalsHash(signature.node_id, i.width(4))) {
        return true;
        }
    }
return false;
```

# Example: NODE-ID-MATCH (REDIR)

```
var checkIntervals = function(node_id, level, node, factor) {
    var size = new BigNumber(2).pow(128);
    var node = toBigNumber(node_id);
    for (var f = 0; f < factor; f++) {
        var temp = size.multiply(new BigNumber(f)
          .pow(new BigNumber(level).negate()));
        var min = temp.multiply(node.add(new BigNumber(f)
          .divide(factor)));
        var max = temp.multiply(node.add(new BigNumber(f + 1)
          .divide(factor)));
        if (node.compare(min) === -1 || node.compare(max) == 1
          || node.compare(max) == 0) return false;
    }
    return true;
};
```

# NODE-ID-MATCH cont.

```
var level = function(value) {
    var length = value[16] * 256 + value[17];
    return value[18 + length] * 256 + value[18 + length + 1];
};

var node = function(value) {
    var length = value[16] * 256 + value[17];
    return value[18 + length + 2] * 256+ value[18 + length + 3];
};

var namespace = function(value) {
    var length = value[16] * 256 + value[17];
    return String.fromCharCode(value.slice(18, length));
};
```

# NODE-ID-MATCH cont.

```
var equals = function(a, b) {
    if (a.length !== b.length) return false;
    for (var i = 0; i < a.length; i++) {
        if (a[i] !== b[i]) return false;
    }
    return true;
};

return equals(entry.key, signature.node_id)
  && (!entry.exists || checkIntervals(entry.key,
    level(entry.value), node(entry.value),
    kind.params['branching-factor']))
  && (!entry.exists
    || resource.equalsHash(namespace(entry.value),
      level(entry.value), node(entry.value)));
```

# Mandatory extensions

Not all overlays will need this mechanism, so it is OK to develop this as an extension.

But overlays that use this extension must be sure that all peers are implementing it.

Propose to add an extension in the configuration file (in p2psip-base) to make an extension mandatory.

# Conclusions

FOSS implementation in the coming months.

Send me your new ACP.

Adoption by WG?

Adding support for mandatory extensions in -base?

Questions?

Found any interesting beer to try?