Establishing Policy-based Resource Quotas at Software-defined Exchanges

Marshall Brinn, GPO

June 16, 2015

GEC23

Sponsored by the National Science Foundation

Outline

- Policies and SDX
- Policy-based Authorization in Aggregates
- Quota-based Policies

Policies and SDX

- One of the key enablers of federation is **trust**
  - At a human level, between the federates
  - At a network level, in the ability of software to enforce agreements between federates
- Quota-based policies are a critical element of establishing such trust for cross-domain resource exchanges
  - Enabling each federation member to set and enforce its own prioritization on resource allocation among different federation peers

Policies and SDX: Examples

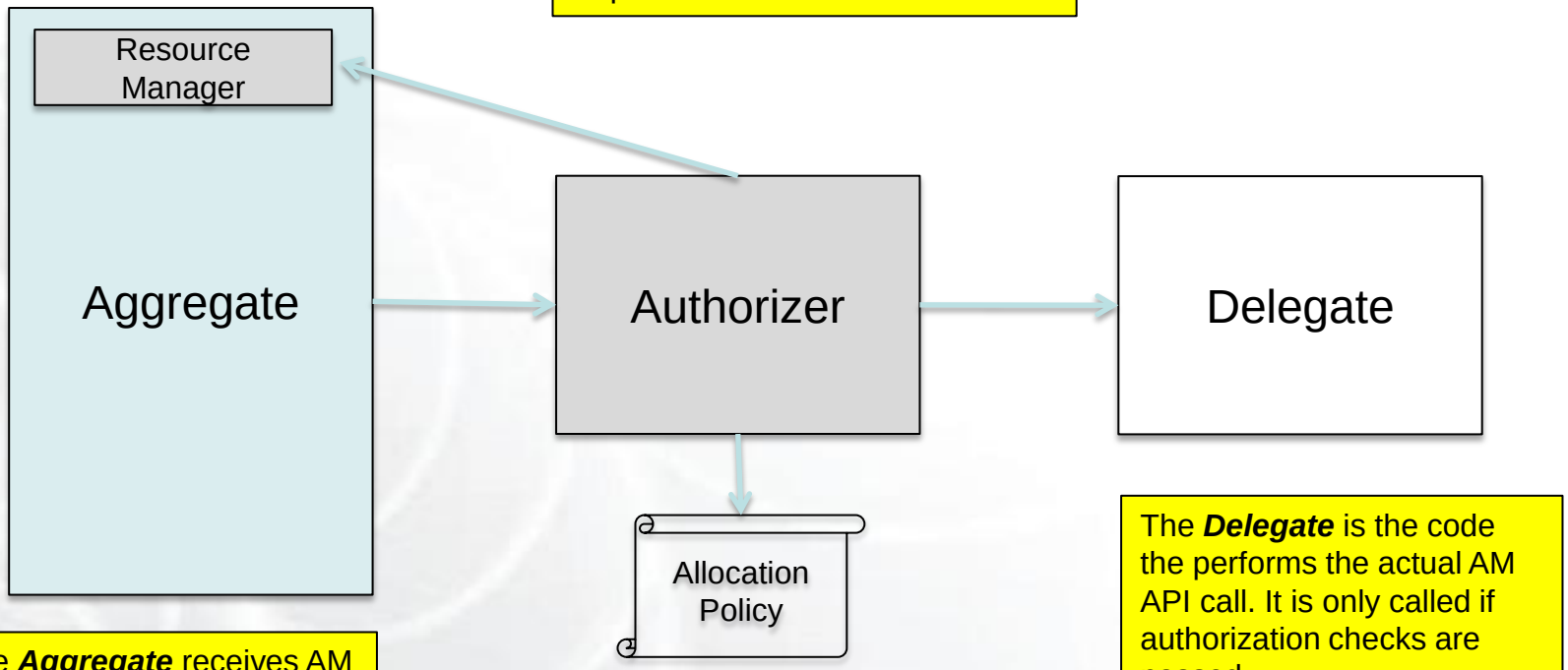
What are the kinds of things one would want to say in policy and enforce at exchange points?

- Consider an example federation between GENI and iMinds. They might want to agree to (and enforce) such statements as:
  - *GENI*: “I want GENI users to be able to allocate up to 500 mbps at WIX, but iMinds users to only allocate 200 mbps at WIX”
  - iMinds: “I want to limit the number of VMs that any GENI slice can allocate on my resources during the summer”
  - *GENI*: “I want to limit the total number of VLANs allocated at WIX at any one time over all iMinds users to 10”

Policy-based Authorization in Aggregates

The ***Resource Manager*** computes a summary of current PLUS requested resource allocations from Aggregate internal state. The authorizer uses this to compute policy adherence.

The ***Authorizer*** determines if the current request is authorized by evaluating the required and forbidden policies in the context of the current call and current and requested allocation state.



The ***Aggregate*** receives AM API calls using XMLRPC over SSL. It thus knows the identity (SSL public cert) of the caller, and the authority to whom the caller belongs.

The ***Allocation Policy*** specifies requirements of what is required or forbidden for the user's call to be authorized.

The ***Delegate*** is the code the performs the actual AM API call. It is only called if authorization checks are passed.

ABAC-based Policy Format

```
{
    "identities" : {
    …
    },
    "constants" : {
     …
    },
    "binders" : [
     …
        ],
    "conditional_assertions" : [
                    …
    ],
    "policies" : [
                    …
    ],
    "queries" : [
                    …
    ]
}
```

Identities: Establish a set of named identities linked to specified certificates

Constants: define a set of named constants for improving readability and maintainability

Binders: A set of Python classes to bind variables based on current allocation context (caller, time, authority, resources)

Conditional Assertions: Rules to determine whether to make ABAC assertions/statements based on current context

Policies: ABAC statements of which assertions imply which other assertions

Queries: Set of statements that must be proven (if positive) or not proven (if negative) for authorization

The policies are in a particular JSON format that combines Python logic and ABAC (Attribute-based Access Control) assertions and identities to express positive and negative requirements.

Quota-based Policies

- The aggregate resource manager reports allocation metrics of the ***proposed future state*** (current + requested)
- Allocation metrics can be anything coordinated by the aggregate and policy developers
  - A common example is ‘NODE’ : # of NODES (VM’s, PC’s) allocated
- These are then
  - Aggregated by AUTHORITY, PROJECT, SLICE, USER
  - Computed by HOURS, MAX, TOTAL

Thus, e.g. we can write policies in terms of variables such as AUTHORITY_NODE_TOTAL or PROJECT_NODE_HOURS

Quota-based Policies [2]

- Most critical for SDX, we want to be able to manage quotas on cross-domain network allocations
- For this we define these allocation metrics, gleaned from the stitching request RSpec:
  - CAPACITY: Bandwidth requested for cross-domain traffic
    - Note: This is not necessarily the same as what BW is provided in real-time
    - Rather, it establishes an upper bound at allocation time.
  - VLAN: The number of VLANs allocated for cross-domain traffic

Prototype Capability

- Xi Yang and Tom Lehman (MAX), Brecht Vermeulen (iMinds) and I have developed a working prototype SDX that enforces quota-based policies in this manner.
  - Integrated in the current WIX aggregate manager which is the US side of GENI/iMinds exchange.
  - Integrated in the current iMinds aggregate manager which is the EU side of the GENI/iMinds exchange
- The policies are proof-of-concept (i.e. for demonstration purposes) but the architecture and implementation are real.

Quota Enforcement: Example output from omni

Allocation of slivers in slice urn:publicid:IDN+ch.geni.net:MAX-GENI+slice+max-gram-test1 at maxgram3 failed: Error from Aggregate: code 2: Authorization Failure, Quota Exceeded.
Allocate slivers in slice urn:publicid:IDN+ch.geni.net:MAX-GENI+slice+max-gram-test1 failed at http://max-myplc.dragon.maxgigapop.net:5001

Quota Enforcement: Example output from AM

```
2015-06-15 11:01:42,415 Authorizing Allocate_V3 urn:publicid:IDN+ch.geni.net+user+xyang #Creds = 1 Args = ['slice_urn', 'rspec'] Opts =['geni_true_caller_cert', 'geni_am_urn']
2015-06-15 11:01:42,420 RAS = [{'slice_urn': 'urn:publicid:IDN+ch.geni.net:MAX-GENI+slice+wix-sdx-test1', 'user_urn': 'urn:publicid:IDN+ch.geni.net+user+xyang', 'start_time': '06-
15-2015 11:08:53.000000', 'measurements': {'VLAN': 1, 'CAPACITY': 100000}, 'end_time': '06-16-2015 10:58:54.000000', 'sliver_urn': 'not_set_yet'}, {'slice_urn': 'urn:publicid:IDN+
ch.geni.net:MAX-GENI+slice+wix-sdx-test2', 'user_urn': 'urn:publicid:IDN+ch.geni.net+user+xyang', 'start_time': '2015-06-15 15:01:42.373989', 'measurements': {'VLAN': 1, 'CAPACITY
': 400000}, 'end_time': '2015-06-22 14:55:58', 'sliver_urn': 'not_set_yet'}]
2015-06-15 11:01:42,420 In ABAC AUTHORIZER...
2015-06-15 11:01:42,709 BINDINGS = {'$AUTHORITY_VLAN_MAX': '2', '$SLICE_VLAN_MAX': '2', '$AUTHORITY_VLAN_HOURS': '191.737777778', '$REQUESTED_STITCH_POINTS': '[]', '$SLICE_VLAN_HO
URS': '191.737777778', '$USER_CAPACITY_TOTAL': '500000', '$MONTH': '6', '$USER_CAPACITY_MAX': '500000', '$SLICE_CAPACITY_MAX': '500000', '$DAY_OF_WEEK': '0', '$AUTHORITY_CAPACITY_
MAX': '500000', '$PROJECT_CAPACITY_HOURS': '69545027.7778', u'$GENI_AUTHORITY': u'urn:publicid:IDN+ch.geni.net+authority+ca', '$HOUR': '11', u'$IMINDS_CAPACITY_QUOTA': u'300000',
'$CALLER_AUTHORITY': 'urn:publicid:IDN+ch.geni.net+authority+ca', '$CALLER': 'urn:publicid:IDN+ch.geni.net+user+xyang', '$AUTHORITY_CAPACITY_HOURS': '69545027.7778', '$USER_VLAN_T
OTAL': '2', '$YEAR': '2015', '$PROJECT_URN': 'urn:publicid:IDN+ch.geni.net+project+MAX-GENI', '$SLICE_CAPACITY_HOURS': '69545027.7778', '$PROJECT_VLAN_MAX': '2', '$PROJECT_CAPACIT
Y_MAX': '500000', '$USER_VLAN_MAX': '2', '$SLICE_CAPACITY_TOTAL': '500000', u'$GENI_CAPACITY_QUOTA': u'500000', '$SFA_AUTHORIZED': 'True', u'$IMINDS_AUTHORITY': u'urn:publicid:IDN
+wall2.ilab2.iminds.be+authority+ca', '$SLICE_VLAN_TOTAL': '2', '$PROJECT_VLAN_TOTAL': '2', '$PROJECT_CAPACITY_TOTAL': '500000', '$METHOD': 'Allocate_V3', '$USER_VLAN_HOURS': '191
.737777778', '$USER_NUM_SLICES': '1', '$USER_NUM_PROJECTS': '1', '$USER_CAPACITY_HOURS': '69545027.
UOTA': u'5', '$PROJECT_VLAN_HOURS': '191.737777778', '$AUTHORITY_CAPACITY_TOTAL': '500000', '$SLICE_URN': 'urn:publicid:IDN+ch.geni.net:MAX-GENI+slice+wix-sdx-test1'}
2015-06-15 11:01:42,715 EVAL : 'urn:publicid:IDN+ch.geni.net+authority+ca' == 'urn:publicid:IDN+ch.geni.net+authority+ca' and 500000 > 500000
2015-06-15 11:01:42,715 EVAL : 'urn:publicid:IDN+ch.geni.net+authority+ca' == 'urn:publicid:IDN+ch.geni.net+authority+ca' and 2 > 5
2015-06-15 11:01:42,716 EVAL : 'urn:publicid:IDN+ch.geni.net+authority+ca' == 'urn:publicid:IDN+wal
2015-06-15 11:01:42,716 EVAL : 'urn:publicid:IDN+ch.geni.net+authority+ca' == 'urn:publicid:IDN+wall2.ilab2.iminds.be+authority+ca' and 2 > 3
2015-06-15 11:01:42,718 ASSERTIONS = [u'AM.MAY_SHUTDOWN<-GENI_CH_CA.MAY_SHUTDOWN']
2015-06-15 11:01:42,720 QUERY (False) : AM.EXCEEDS_QUOTA<-urn:publicid:IDN+ch.geni.net+user+xyang
```

resource information for existing and new slivers

binding to policy constraint variables

evaluating against policy assertions

query on a policy rule return 'False'

```
2015-06-15 11:03:56,463 EVAL : 'urn:publicid:IDN+ch.geni.net+authority+ca' == 'urn:publicid:IDN+ch.geni.net+authority+ca' and 600000 > 500000
2015-06-15 11:03:56,464 EVAL : 'urn:publicid:IDN+ch.geni.net+authority+ca' == 'urn:publicid:IDN+ch.geni.net+authority+ca' and 3 > 5
2015-06-15 11:03:56,464 EVAL : 'urn:publicid:IDN+ch.geni.net+authority+ca' == 'urn:publici
2015-06-15 11:03:56,464 EVAL : 'urn:publicid:IDN+ch.geni.net+authority+ca' == 'urn:publici
2015-06-15 11:03:56,466 ASSERTIONS = [u'AM.EXCEEDS_QUOTA<-urn:publicid:IDN+ch.geni.net+user+xyang', u'AM.MAY_SHUTDOWN<-GENI_CH_CA.MAY_SHUTDOWN']
2015-06-15 11:03:56,482 QUERY (True) : AM.EXCEEDS_QUOTA<-urn:publicid:IDN+ch.geni.net+user+xyang
2015-06-15 11:03:56,483 PROOF_CHAIN : [u'AM.EXCEEDS_QUOTA<-urn:publicid:IDN+ch.geni.net+user+xyang']
Traceback (most recent call last):
  File "/opt/gcf/src/gcf/geni/am/am_method_context.py", line 137, in __enter__
    requested_allocation_state)
  File "/opt/gcf/src/gcf/geni/auth/abac_authorizer.py", line 143, in authorize
    raise Exception(msg)
Exception: Quota Exceeded
2015-06-15 11:03:56,503 Quota Exceeded
2015-06-15 11:03:56,505 Result from Allocate_V3: {'output': 'Quota Exceeded', 'code': {'am_type': 'gcf', 'geni_code': 2, 'am_code': 0}, 'value': ''}
```

a 3rd sliver triggers assertion after total capacity exceeds quota

query on policy rule returns 'True' that causes "quota excetion"

Summary

- Quota-based policies are a critical enabler of cross-domain resource exchanges
- A prototype for such policies is currently implemented and distributed with the current ***geni-tools*** (gcf) software package

- Anyone interested in using this framework to establish such policy-based authorization in their aggregates, please contact help@geni.net or mbrinn@bbn.com.

ABAC-BASED POLICY: ANNOTATED EXAMPLE

ABAC-based Policy Format: Annotated

```
{
   "identities" : {
   …
   },
   "constants" : {
    …
   },
   "binders" : [
    …
      ],
   "conditional_assertions" : [
                    …
   ],
   "policies" : [
                    …
   ],
   "queries" : [
                    …
   ]
}
```

```
{
   "identities" : {
   "GENI_CH_CA" :
      "/etc/gram/trusted_roots/ch.geni.net-ca.pem"
 },
```

Identities: Establish a set of named identities linked to specified certificates

ABAC-based Policy Format: Annotated

```
{
    "identities" : {
    …
    },
    "constants" : {
     …
    },
    "binders" : [
     …
        ],
    "conditional_assertions" : [
                        …
    ],
    "policies" : [
                        …
    ],
    "queries" : [
                        …
    ]
}
```

```
{
"constants" : {
      "$GENI_VLAN_QUOTA" : "5”,
      "$GENI_CAPACITY_QUOTA" : "500000”,
      "$GENI_AUTHORITY" :
  "urn:publicid:IDN+ch.geni.net+authority+ca”,
      "$IMINDS_VLAN_QUOTA" : "3”,
      "$IMINDS_CAPACITY_QUOTA" : "300000”,
      "$IMINDS_AUTHORITY" :
   "urn:publicid:IDN+wall2.ilab2.iminds.be+authority+ca”
 },
```

Constants: define a set of named constants for improving readability and maintainability

ABAC-based Policy Format: Annotated

```
{
    "identities" : {
    …
    },
    "constants" : {
     …
    },
    "binders" : [
     …
       ],
    "conditional_assertions" : [
                    …
    ],
    "policies" : [
                    …
    ],
    "queries" : [
                    …
    ]
}
```

```
"binders" : [
    "gcf.geni.auth.binders.Standard_Binder",
    "gcf.geni.auth.binders.SFA_Binder",
    "gcf.geni.auth.binders.Stitching_Binder",
    "gcf.geni.auth.resource_binder.MAX_Binder",
    "gcf.geni.auth.resource_binder.TOTAL_Binder",
    "gcf.geni.auth.resource_binder.HOURS_Binder",
    "gcf.geni.auth.resource_binder.User_Slice_Binder"
],
```

Binders: A set of Python classes to bind variables based on current allocation context (caller, time, authority, resources)

ABAC-based Policy Format: Annotated

```
{
    "identities" : {
    …
    },
    "constants" : {
     …
    },
    "binders" : [
     …
        ],
    "conditional_assertions" : [
                    …
    ],
    "policies" : [
                    …
    ],
    "queries" : [
                    …
    ]
}
```

```
"conditional_assertions" : [
      {
                    "precondition" : "True",
                    "exclusive" : false,
                    "clauses" : [
          {
          "condition" : "'$CALLER_AUTHORITY' == '$GENI_AUTHORITY' and
                        $AUTHORITY_CAPACITY_TOTAL >= $GENI_CAPACITY_QUOTA",
          "assertion" : "AM.EXCEEDS_QUOTA<-$CALLER"
          },
          {
          "condition" : "'$CALLER_AUTHORITY' == '$GENI_AUTHORITY' and
                         $AUTHORITY_VLAN_TOTAL >= $GENI_VLAN_QUOTA",
          "assertion" : "AM.EXCEEDS_QUOTA<-$CALLER"
          },
          {
          "condition" : "'$CALLER_AUTHORITY' == '$IMINDS_AUTHORITY' and
                $AUTHORITY_CAPACITY_TOTAL >= $IMINDS_CAPACITY_QUOTA",
          "assertion" : "AM.EXCEEDS_QUOTA<-$CALLER"
          },
          {
          "condition" : "'$CALLER_AUTHORITY' == '$IMINDS_AUTHORITY' and
                $AUTHORITY_VLAN_TOTAL >= $IMINDS_VLAN_QUOTA",
          "assertion" : "AM.EXCEEDS_QUOTA<-$CALLER"
          }
        ]
      }
   ],
```

Conditional Assertions: Rules to determine whether to make ABAC assertions/statements based on current context

ABAC-based Policy Format: Annotated

```
{
    "identities" : {
    …
    },
    "constants" : {
     …
    },
    "binders" : [
     …
        ],
    "conditional_assertions" : [
                    …
    ],
    "policies" : [
                    …
    ],
    "queries" : [
                    …
    ]
}
```

```
"policies" : [
      "AM.MAY_SHUTDOWN<-GENI_CH_CA.MAY_SHUTDOWN"
 ],
```

Policies: ABAC statements of which assertions imply which other assertions

ABAC-based Policy Format: Annotated

```
{
    "identities" : {
    …
    },
    "constants" : {
     …
    },
    "binders" : [
     …
        ],
    "conditional_assertions" : [
                    …
    ],
    "policies" : [
                    …
    ],
    "queries" : [
                    …
    ]
}
```

```
"queries" : [
        {
            "statement" : "AM.EXCEEDS_QUOTA<-$CALLER”,
            "is_positive" : false,
            "message" : "Quota Exceeded”
            },
        {
            "statement" : "AM.MAY_SHUTDOWN<-$CALLER”,
            "is_positive" : true,
            "message" : "Privilege Failure”,
            "condition" : "'$METHOD' in
                                ['Shutdown_V2', 'Shutdown_V3']”
            }
]
```

Queries: Set of statements that must be proven (if positive) or not proven (if negative) for authorization