LISA ’09
Federated access control and workflow enforcement in systems configuration

Bart Vanbrabant, Thomas Delaet and Wouter Joosen

DistriNet, Dept. of Computer Science, K.U.Leuven, Belgium

November 6, 2009
Systems configuration
  Context
  Problems

Our solution: ACHEL
  Access control and workflow
  Generating meaningful changes

Prototype

Evaluation
  Case 1
  Case 2

Conclusion
Outline

Systems configuration
  Context
  Problems

Our solution: ACHEL
  Access control and workflow
  Generating meaningful changes

Prototype

Evaluation
  Case 1
  Case 2

Conclusion
System configuration tools
System configuration tools
System configuration tools
System configuration tools
System configuration tools
System configuration tools
Access control rules

[@netadmins]
lib = r
hosts = r
lib/ net = rw

[@senior]
= rw

[@mail]
lib/mail = rw
lib/file = rw

[userA]
hosts/verdana.cs.kuleuven.be.cf = rw

Repository

lib/
  net/
    dhcp.cf
    routing.cf
  web/
    cluster.cf
    ...
  mail/
    ...
  file/
    ...
hosts/
  ...

[@senior]
= rw

[@mail]
lib/mail = rw
lib/file = rw

[userA]
hosts/verdana.cs.kuleuven.be.cf = rw
UserA can not be trusted

hosts/verdana.cs.kuleuven.be.cf

Some global network configuration!
Workflow

Dev repository

Updates approved by security officer

Q&A repository

Production repository
Federated infrastructures

Central Repository

Site 1 Repository
Site 2 Repository
Site 3 Repository
Site 4 Repository
Outline

Systems configuration
  Context
  Problems

Our solution: ACHEL
  Access control and workflow
  Generating meaningful changes

Prototype

Evaluation
  Case 1
  Case 2

Conclusion
ACHEL manages access to repositories of configuration specification by implementing access control and enforcing workflows

- fine-grained access control interpreting the semantics of changes
- access control is applied at the abstraction level of the configuration specification
- support for workflow in federated infrastructures
- a (configuration) language agnostic solution
Update 1: an allowed change
Current specification for managing the motd file written by Bart:

```python
motd_file = File()
motd_file.name = "/etc/motd"
motd_file.content = "Welcome to $hostname"
motd_file.owner = "root"
motd_file.group = "root"
motd_file.perm = "0644"
```
Thomas changes the content of the motd file:

```python
motd_file = File()
motd_file.name = "/etc/motd"
motd_file.content = template("motd.tmpl")
motd_file.owner = "root"
motd_file.group = "root"
motd_file.perm = "0644"
```
Access control policy

# list of admins
define admins as
  bart.vanbrabant@cs.kuleuven.be,
  wouter.joosen@cs.kuleuven.be

# allow admins to create the motd
allow admins to:
  * assign File() to motd_file
  * assign "/etc/motd" to motd_file.name

# allow everyone to manage the motd
allow to:
  * assign * to motd_file.content

# demand approval by an admin to change
# the permissions (all other attributes)
allow to:
  /(add|modify)/ assign * to motd_file.*
  authorised by 1 admins

update {
  action => modify
  operation => assign
  lhs => motd_file.content
  rhs => template("motd.tmpl")
  old_rhs => "Welcome to $hostname"
  owner => bart.vanbrabant@cs.kuleuven.be
  author => thomas.delaet@cs.kuleuven.be
}
Update 1: an allowed change

Output from our prototype for the motd example:

Rev 1 has 6 changes and 0 signatures
- allowed bart.vanbrabant@cs.kuleuven.be to add assign "/etc/motd" to motd_file.name
- allowed bart.vanbrabant@cs.kuleuven.be to add assign "Welcome at $hostname" to motd_file.content
- allowed bart.vanbrabant@cs.kuleuven.be to add assign "root" to motd_file.group
- allowed bart.vanbrabant@cs.kuleuven.be to add assign File() to motd_file
- allowed bart.vanbrabant@cs.kuleuven.be to add assign "root" to motd_file.owner
- allowed bart.vanbrabant@cs.kuleuven.be to add assign "0644" to motd_file.perm

Rev 2 has 1 changes and 0 signatures
- allowed thomas.delaet@cs.kuleuven.be to modify assign template("motd.tmpl") to motd_file.content
Update 1: an allowed change

Sysadmin

Commit change in system

ACL Check on:
- Content
- Author
- Owner

DVCS

Sysadmin pushes changes

ACL Check on:
- Content
- Author
- Owner

DVCS

Enforce rule
Warn rule
Configuration specification
Host profile

Tool

Enforce configuration on infrastructure

BCFG2
Update 2: a change requiring authorisation
Thomas changes the permissions of the motd file:

```python
motd_file = File()
motd_file.name = "/etc/motd"
motd_file.content = template("motd.tmpl")
motd_file.owner = "root"
motd_file.group = "wheel"
motd_file.perm = "0644"
```
Access control policy

```plaintext
# list of admins
define admins as
  bart.vanbrabant@cs.kuleuven.be,
  wouter.joosen@cs.kuleuven.be

# allow admins to create the motd
allow admins to:
  * assign File() to motd_file
  * assign "/etc/motd" to motd_file.name

# allow everyone to manage the motd
allow to:
  * assign * to motd_file.content

# demand approval by an admin to change
# the permissions (all other attributes)
allow to:
  /(add|modify)/ assign * to motd_file.*
  authorised by 1 admins
```

Update 2: a change requiring authorisation

```plaintext
update {
  action => modify
  operation => assign
  lhs => motd_file.group
  rhs => "wheel"
  old_rhs => "root"
  owner => bart.vanbrabant@cs.kuleuven.be
  author => thomas.delaet@cs.kuleuven.be
}
```
Output from our prototype for the motd example:

Rev 1 has 6 changes and 0 signatures
  allowed bart.vanbrabant@cs.kuleuven.be to add assign "/etc/motd" to motd_file.name
  allowed bart.vanbrabant@cs.kuleuven.be to add assign "Welcome at $hostname" to motd_file.content
  allowed bart.vanbrabant@cs.kuleuven.be to add assign "root" to motd_file.group
  allowed bart.vanbrabant@cs.kuleuven.be to add assign File() to motd_file
  allowed bart.vanbrabant@cs.kuleuven.be to add assign "root" to motd_file.owner
  allowed bart.vanbrabant@cs.kuleuven.be to add assign "0644" to motd_file.perm

Rev 2 has 1 changes and 0 signatures
  allowed thomas.delaet@cs.kuleuven.be to modify assign template("motd.tmpl") to motd_file.content

Rev 3 has 1 changes and 0 signatures
  authorisation (1) required for thomas.delaet@cs.kuleuven.be to modify assign "wheel" to motd_file.group owned by bart.vanbrabant@cs.kuleuven.be
Update 2: a change requiring authorisation
Generating meaningful changes

Algorithm based on:

- Meaningful change detection in structured data. CHAWATHE AND GARCIA-MOLINE. 1997
- Change Distilling: Tree Differencing for Fine-Grained Source Code Change Extraction. FLURI, WUERSCH, PINZGER AND GALL. 2007
Outline

Systems configuration
  Context
  Problems

Our solution: ACHEL
  Access control and workflow
  Generating meaningful changes

Prototype

Evaluation
  Case 1
  Case 2

Conclusion
Prototype in Python

- built on Mercurial
- simple configuration language and BCFG2 for deployment
- PGP for signatures and authentication
- access control language using regular expressions for pattern matching
Outline

Systems configuration
  Context
  Problems

Our solution: ACHEL
  Access control and workflow
  Generating meaningful changes

Prototype

Evaluation
  Case 1
  Case 2

Conclusion
Case 1: access control and simple workflow

- Small infrastructure
- Team with junior and senior sysadmins
- Enforce responsibilities
- Enforce coding guidelines
- Manage network configuration
Case 1: access rules

# enforce some conventions on everyone
deny to:
  * assign File() to /^[^\]+(?!file_)[\S]+$/
  * assign Package() to /^[^\]+(?!pkg_)[\S]+$/
  * assign Service() to /^[^\]+(?!service_)[\S]+$/
  * assign Directory() to /^[^\]+(?!dir_)[\S]+$/
  * assign Symlink() to /^[^\]+(?!ln_)[\S]+$/
  * assign Permissions() to /^[^\]+(?!perm_)[\S]+$/

# senior admins can do anything else
allow senioradmin to:
  * * *

# allow admins to do everything if a senior admin approves
allow to:
  * * *
    authorised by 1 senioradmin

# network related configuration
deny netadmins to:
  # deny files other than those in /etc/network
  * assign /^(?!/etc/\!network\/)\S+/$ to /^net_file_\w+\!\.name$/
  # deny services other than dhcpd and network
  * assign /^(?!dhcpd\$|network\$))\w+$/ to /^net_service_\w+\!\.name$/

allow netadmins to:
  * import /^dhcp/
  # allow adding a list of values to the net_dhcp_clients list
  * add /^\^[\^\]\]$/ to /^net_dhcp_clients$/
  # allow only variables prefixed with net (ignore rhs)
  * assign * to /^(?!net_)[\S]+$/
Case 1: configuration

```python
# configure network interfaces
net_file_interfaces = File()
net_file_interfaces.name =
    
"/etc/network/interfaces"
net_file_interfaces.owner = "root"
net_file_interfaces.group = "root"
net_file_interfaces.perms = "0644"
net_file_interfaces.content = source("net/interfaces.$hostname")

# network service needs to be enabled
net_service_network = Service()
net_service_network.name = "network"
net_service_network.status = "on"

# use template for /etc/hosts
net_file_hosts = File()
net_file_hosts.name = 
    
"/etc/hosts"
net_file_hosts.owner = "root"
net_file_hosts.group = "root"
net_file_hosts.perms = "0644"
net_file_hosts.content = template("net/hosts.tmpl")
```
Case 2: complex workflow in federated infrastructures

- Large federated grid infrastructure
- Several administrative domains
- Shared and site specific configuration
- Based on the description of BeGrid in *Devolved Management of Distributed Infrastructures With Quattor*, *LISA ’08*
Case 2: complex workflow in federated infrastructures

Central Repository at Belnet

- KULeuven Repository
- UGent Repository
- UA Repository
- UCL Repository
Outline

Systems configuration
   Context
   Problems

Our solution: ACHEL
   Access control and workflow
   Generating meaningful changes

Prototype

Evaluation
   Case 1
   Case 2

Conclusion
Future work

Validate ACHEL on a complex real-life configuration language. Key challenges:

- develop an access control language that integrates with the configuration language
- provide integration with the tools used with the configuration language
ACHEL’s contributions

- *fine-grained* access control interpreting the *semantics* of changes
- *access control* is applied at the *abstraction level* of the configuration specification
- support for workflow in *federated* infrastructures
- a language *agnostic* approach