



Study of Security of the HUBCAP Sandbox Architecture

Tomas Kulik, Hugo Daniel Macedo, Prasad Talasila and Peter
Gorm Larsen

PROJECT PARTNERS

Aarhus University

Fortiss GmbH

Fundazione Bruno Kessler

University "Lucian Blaga" of Sibiu

Research Institutes of Sweden

Politecnico di Milano

Controllab Products

Verified Systems International

Technology Transfer Systems

Newcastle University

Virtual Vehicle Research

KTH Royal Institute of Technology

Engineering Ingegneria Informatica

F6S Network Limited

Unparallel Innovation

BEIA Consult

Validas





Agenda

- Contribution
- HUBCAP project
- HUBCAP Sandbox Architecture
- Sandbox access control
- VDM Model
- Security analysis
- Conclusion
- Future work





Contribution

- Formally analyzed sandbox access model
- Improvements to the access model
- Move towards distributed Sandboxing
- Increased Security for Model based Engineering platform
- Approach to combinatorial testing of access control properties





HUBCAP

ABOUT HUBCAP

• Who we are

- **Innovation Action** co-financed by the European Commission, DT-ICT-01-2019 Smart Anything Everywhere initiative.
- **Coordinator** Aarhus University, Denmark
- **Project duration** January 2020 - December 2022, 36 months
- **Total EC contribution** EUR ~7.95M
- HUBCAP will provide a one-stop-shop for European SMEs wanting to join the Cyber-Physical Systems (CPS) revolution using Model-Based Design (MBD) techniques.
- **Vision** Lower barriers for SMEs to realize the potential of growing autonomy in CPS by accessing advanced model-based design (MBD) technology, providing training and guidance.

HUBCAP Ecosystem





HUBCAP

ABOUT HUBCAP

• Project setup

Network of DIHs:

- Inventory of service offerings
- Ecosystem building
- Cross-DIH collaboration
- Network sustainability

Seed SMEs

- Enabling quicker start for the platform
- Early-stage prototypical usage of HUBCAP
- Awareness-raising demonstrations



Collaboration Platform:

- Cloud-enabled, based on DIHIWARE
- “Access to” and “Collaborate with”:
 - Ecosystem
 - Community-building
 - Marketplace
 - **Sandbox**

Open Calls

- Engage early-adopters
- 3 open call series

Model-Based Design:

- Populating the platform
- Enabling model-based services in sandbox
- Multi-user, validation and logging capabilities



HUBCAP Sandbox



HUBCAP

Home

Catalogues

Innovation ecosystem and networking

Skills and training

Funding & Opportunities

Collaboration Space

Innovation Space

Sandbox Environment

Catalogues / Models Catalogue /

MODELS CATALOGUE

Search for snippets, click on caret

Model

Disturbance

INVERTED PENDULUM

The model is an electronically controlled inverted pendulum that demonstrates the co-simulation capabilities of AutoFOCUS 3 and how this feature can be leveraged for tool-interoperability. For the tool interaction, we use the Functional Mock-up Interface (F...

ADI/ADAS ON MPSOC PLATFORM

The model represents a set of software-defined ADI/ADAS functions that are deployed to MPSoC architecture integrating both general-purpose microprocessor cores and a GPU. It originates from the Industrial Challenge that accompanies the yearly WATERS worksh...

DUAL CHANNEL

The model shows an example of dual channel design. The requirements are formalized and structured into contracts. The formal properties are validated with formal techniques.

ADAPTIVE CRUISE CONTROL

The model represents a set of software-defined functions that implement an Adaptive Cruise Control. It manages the speed of a vehicle to remain at a user-defined value while keeping a user and regulation-defined distance to a potentially present front car. T...

TRIPLE MODULAR GENERATOR

The Triple Modular Generator represents a power generator with Triple Modular Redundancy. It can be used to try various functionalities of xSAP including Model-Based

ROBOTR3

The RobotR3 model from the Modelica Standard Library. This example animates a motion of a detailed model of the robot with predefined axes' angles over time. For animation,

ROSACE

The model represents a set of software-defined functions of a baseline flight-controller whose components shall be deployed on a platform consisting of two many-core tiles. In

Welcome to HUBCAP Sandbox pietrog | [admin] | Password expiration: 16 Oct 07:01

Home Log out

Operating Systems

Tools

Models

pietrog [admin]

IntoCPS_Ubu18_Wine_Empty1t

ENG_Test13t

20-sim_windows_demo4t

Windows_Test32t

Connected to QEMU (20201001-160941-280730935_20-sim_windows_demo4t)

Ctrl Alt Win Tab Esc Fullscreen

Activate Windows
Go to Settings to activate Windows.

Eliminate All Sbox Data and Tools

Destroy

Archive From Local PC to SBox

Browse... No file selected.

Upload

Archive From SBox To Local PC

Download

Save Current System To Tools Repo

Tool name

Tool Description

Save 20-sim_windows_demo4t

Share SBox With Other Users

Invite as guest

adrian.pop

angelo.marguglio

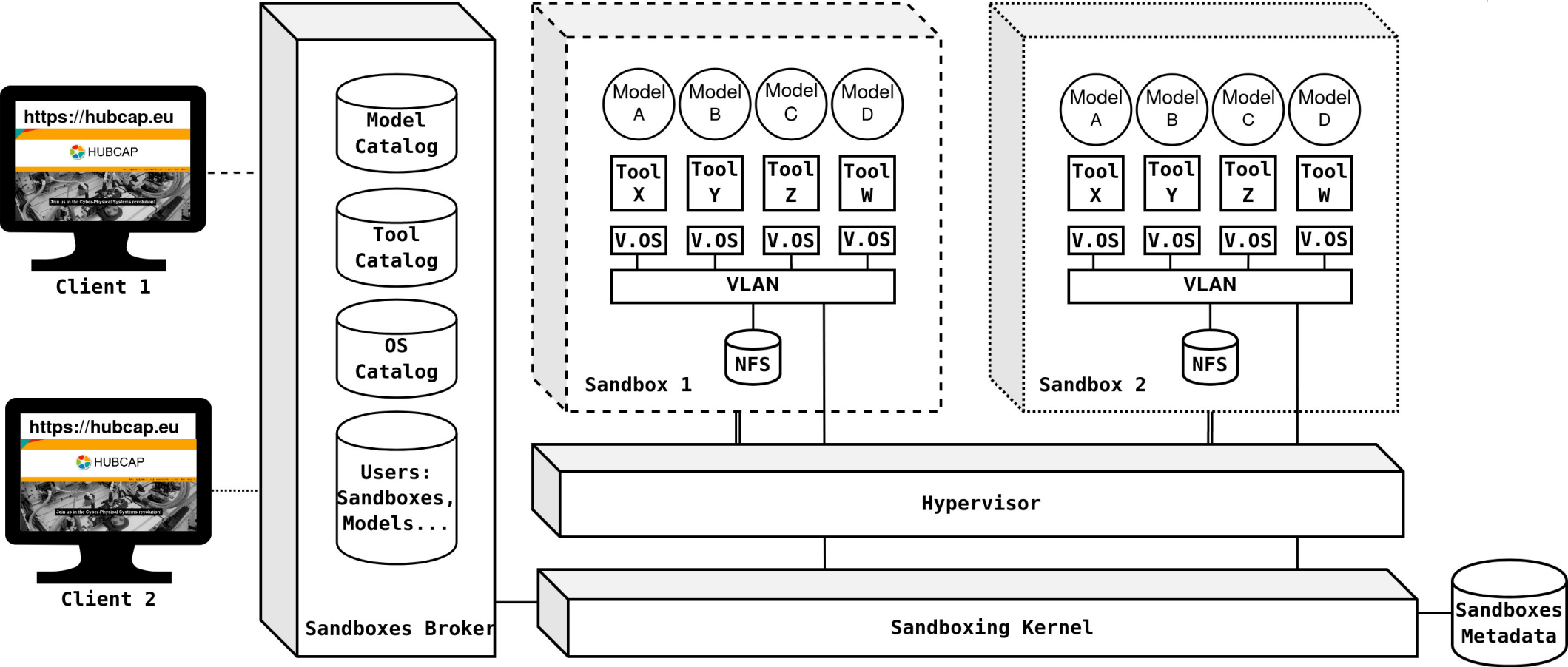
barner.simon





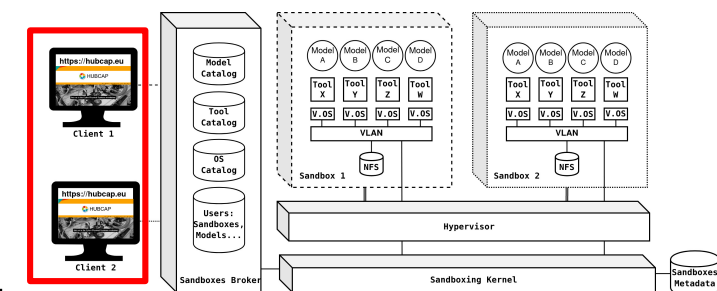
HUBCAP

HUBCAP Platform architecture - Sandbox



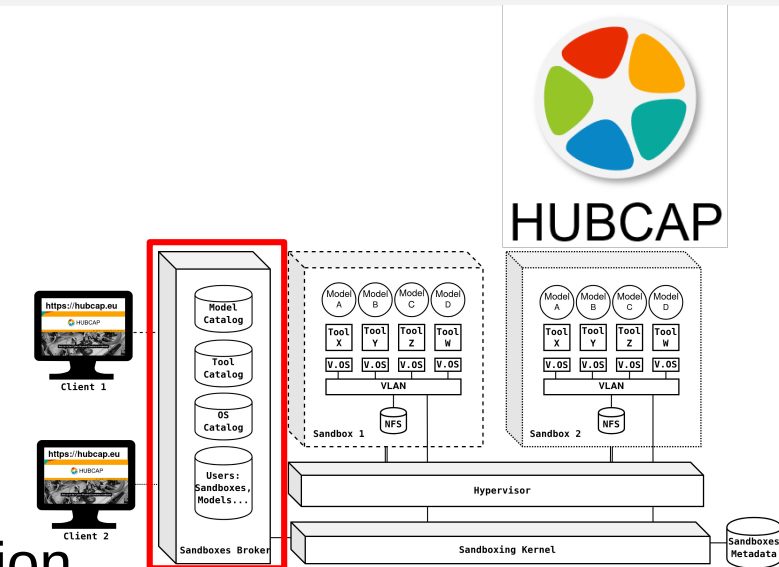
HUBCAP Platform architecture - Primary building blocks

- **Client**
- **Remote access to sandbox** → Connect to Sandbox remotely based on the access rights
- **Interact with the HUBCAP Platform** → Manage existing Sandboxes or create new ones
- **Interact with the HUBCAP Platform** → Select existing repository items or provide new ones



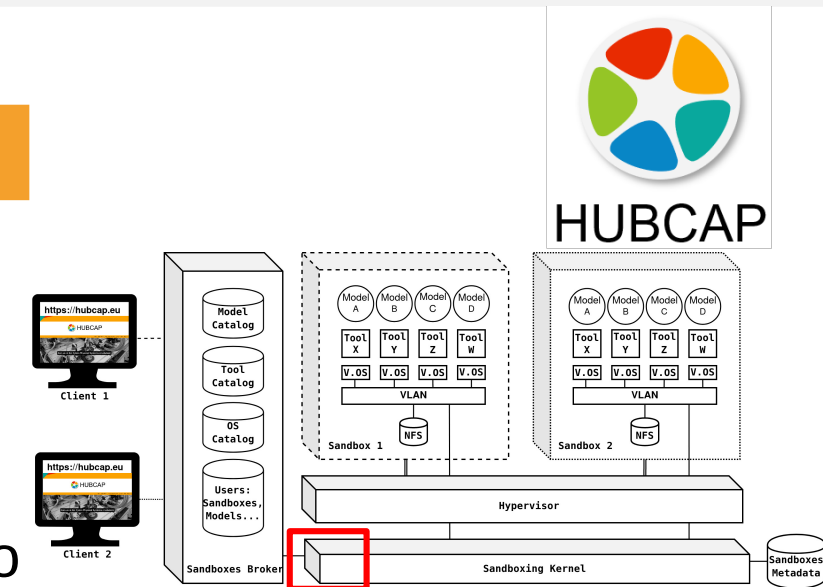
HUBCAP Platform architecture - Primary building blocks

- **Broker**
- **Connection handling to a Sandbox** → Facilitates client connection to a Sandbox
- **Component management within the platform** → Manages components of the HUBCAP platform such as the repository and Sandboxes
- **Persistence of Sandbox settings** → The broker records Sandbox metadata such as identities of the servers under the Sandbox



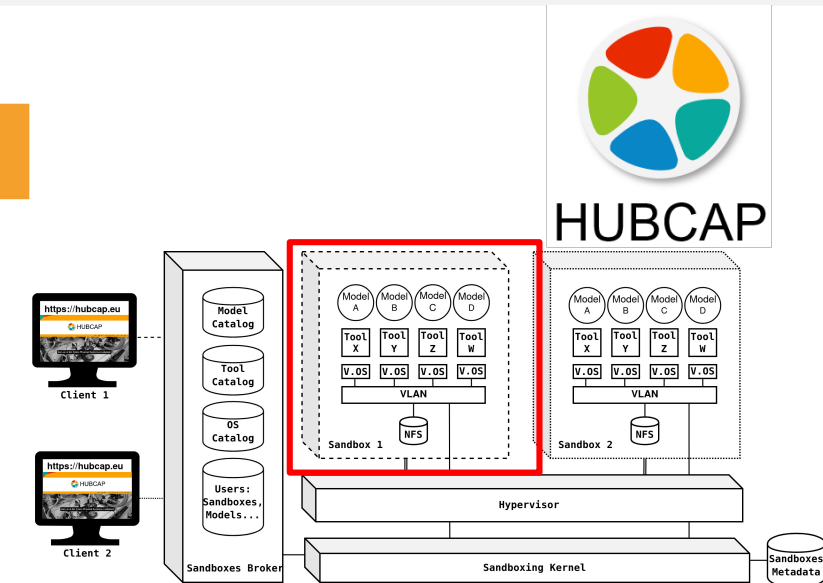
HUBCAP Platform architecture - Primary building blocks

- **Gateway**
- **Direct connection from Client to** → Keeps client connection to a Sandbox open
- **Server connection handling** → Manages connections from clients towards specific servers constituting a Sandbox



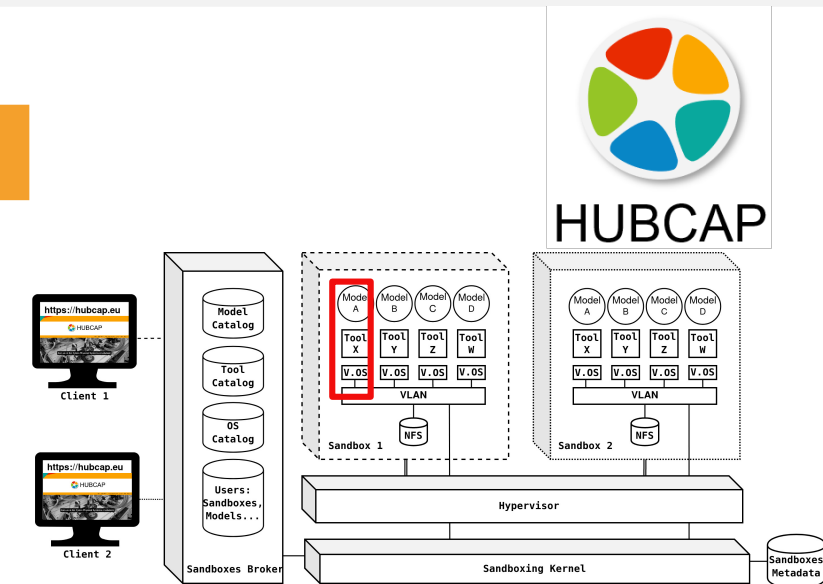
HUBCAP Platform architecture - Primary building blocks

- **Sandbox**
- **Container for Servers** → A Sandbox is a collection of servers combining different tools and potentially models
- **Isolated internal network** → Sandbox provides isolated environment from the rest of the platform
- **Collaboration space** → Used for collaboration on specific modeling tasks for multiple clients



HUBCAP Platform architecture - Primary building blocks

- **Server**
- **Virtualization** → Servers within the Sandbox are considered virtual machines, facilitating fast deployment
- **Different Operating Systems** → Servers can be installed with different operating systems





Sandbox Access Control

- Limit the user access to Sandbox
- Distinguish between providers and consumers
- Distinguish between owners and guests
- Limit the functionality based on the role or profile
- Ensure intellectual property protection
- Iterate to accommodate new functionality





Sandbox Access Control

Feature	Provider	Provider	Consumer	Consumer
	Owner	Guest	Owner	Guest
Access to remote viewer	X	X	X	X
Upload archive	X	X	X	X
Download archive	X		X	
Invite guests	X		X	
Destroy sandbox	X		X	
Select tool	X		X	
Select model	X		X	
Select operating system	X			
Save tool	X			
Upload new model	X			
Delete repository tool	X			





VDM-SL Model

- Model the different components
 - Model the System behavior on top
 - Capture the access table
-
- Single module model
 - Multiple traces for analysis
 - Use of Combinatorial Testing





Client VDM-SL Model

-- InviteGuest

```
InviteGuest: ClientId * ClientId * SandboxId ==> ()
InviteGuest(cId, cguestId, sId)==
  BrokerAddGuest(cId, cguestId, sId)
pre cId in set validClients;
```

-- Select OS

```
SelectOS : ClientId * OSId ==> ()
SelectOS(cId, osId)==
  clientst.selectedOS := SelectOperatingSystem(osId, cId)
pre cId in set validClients;
```

-- Select Tool

```
SelectToolFromRepository: ClientId * ToolId ==> ()
SelectToolFromRepository(cId, tId) ==
  clientst.selectedTool := SelectTool(tId, cId)
pre cId in set validClients;
```

types

```
ClientSt::
  selectedTool :
  SelectedTool
  selectedOS : SelectedOS
  selectedModel :
  SelectedModel;
```

```
ClientId = nat;
KnownSandboxes = set of
  SandboxId;
Errors = set of token;
SelectedTool = [nat];
SelectedOS = [nat];
SelectedModel = [nat]
```

State





Broker VDM-SL Model

functions

ClientIsNull: ClientId * Providers * Consumers * Owners * Guests -> **bool**

ClientIsNull(cId, ps, cs, os, gs) ==

```
cId not in set ps and
cId not in set cs and
cId not in set dom os and
cId not in set dom gs;
```

operations

BrokerInitiateSandboxAccess: ClientId * SandboxId ==> **bool**

BrokerInitiateSandboxAccess(cId, sId) ==

```
let sandboxes = GetSystemSandboxes(),
    servers = dunion {s.sandboxServers | s in set rng sandboxes &
s.sandboxId = sId}
in
```

```
(for all x in set servers do
  UpdateConnections(cId, x, sId, true);
  return true) -- one needs to enable the UpdateConnections
```

operation to report if it was ok

```
pre not ClientIsNull(cId, brokerst.providers, brokerst.consumers,
brokerst.owners, brokerst.guests)
and
```

```
((cId in set dom brokerst.owners and sId in set brokerst.owners(cId))
  or (cId in set dom brokerst.guests and sId in set
brokerst.guests(cId)));
```

types

State

BrokerSt ::

```
providers : Providers
consumers : Consumers
validModels : ValidModels
activeSandboxes : ActiveSandboxes
validTools : ValidTools
validOSs : ValidOSs
owners : Owners
guests : Guests
errorLog : ErrorLog
sandboxModels : SandboxModels
sandboxTools : SandboxTools
sandboxOSs : SandboxOSs;
```

Owners = **map** ClientId **to set of** SandboxId;

Guests = **map** ClientId **to set of** SandboxId;

ActiveSandboxes = **set of** SandboxId;

...



Gateway + Server + Sandbox VDM-SL Model

types

Gateway

```
GateWaySt ::  
connectedClients : ConnectedClients  
connectedServers : ConnectedServers;
```

```
ConnectedClients = set of ClientId;  
ConnectedServers = set of ServerId;
```

types

Server

```
ServerId = nat;
```

Sandbox

types

```
SandboxId = nat;  
SandboxServers = set of ServerId;
```

```
Sandbox::
```

```
  sandboxId : SandboxId  
  sandboxServers : SandboxServers
```

Destroying a Sandbox

```
-- Destroying the sandbox removes it from  
-- known system sandboxes  
DestroySandbox : ClientId * SandboxId ==> ()  
DestroySandbox(cId, sId) ==  
  (systemSandboxes := {sId} <-:  
   systemSandboxes;  
   brokerst.owners(cId) :=  
   brokerst.owners(cId) \ {sId})  
pre cId in set dom brokerst.owners  
and  
sId in set brokerst.owners(cId)  
and  
not sId in set brokerst.activeSandboxes  
post  
sId not in set brokerst.owners(cId);
```





Initial State

```
GenerateNewSandboxId: () ==> nat
GenerateNewSandboxId() ==
  return Max(systemSandboxes) + 1;
```





Overture - Combinatorial Testing

The screenshot shows the Visual Studio Code interface with the following components:

- File Explorer (Left):** Displays the project structure, including 'HUBCAP_Sandbox' and its subdirectories: 'generated', 'Broker.vdmsl', 'Client.vdmsl', 'Gateway.vdmsl', 'Sandbox.vdmsl', 'Server.vdmsl', 'System.vdmsl', and 'Test.vdmsl'.
- Main Editor:** Shows the 'Test.vdmsl' file with the following content:


```

1  operations
2  SetupClients: ClientId ==>()
3  SetupClients(cId) == validClients union {cId};
4
5  SetupOSs: OSId ==> ()
6  SetupOSs(osId) == brokerst.validOSs := brokerst.validOSs union {osId};
7
8  SetupTools: ToolId ==> ()
9  SetupTools(tId) == brokerst.validTools := brokerst.validTools union {tId};
10
11 SetupModels: ModelId ==> ()
12 SetupModels(mId) == brokerst.validModels := brokerst.validModels union {mId};
13
14 SetupProviders: ClientId ==> ()
15 SetupProviders(cId) == brokerst.providers := brokerst.providers union {cId};
16
17 SetupConsumers: ClientId ==> ()
18 SetupConsumers(cId) == brokerst.consumers := brokerst.consumers union {cId};
19
20 traces
21 --Create Sandbox
22 CreateSandbox:
23   SetupClients(1);
24   SetupProviders(1);
25   SetupOSs(1);
26   SetupTools(1);
27   SetupModels(1);
28   let clientId in set {1}
29   in
30     SelectOS(clientId, 1);
31     SelectToolFromRepository(clientId, 1);
32     SelectModelFromRepository(clientId, 1);
33     LaunchNewSandbox(clientId)
34   );
35
36 --Create Sandbox multiple clients
37 CreateSandboxMultipleClients:
38   SetupClients(1);
39   SetupClients(2);
40   SetupClients(3);
41   SetupProviders(1);
42   SetupConsumers(2);
43   SetupOSs(1);
44   SetupTools(1);
45   SetupModels(1);
46   let clientId in set {1, 2, 3}
47   in
48     SelectOS(clientId, 1);
49     SelectToolFromRepository(clientId, 1);
50     SelectModelFromRepository(clientId, 1);
51     LaunchNewSandbox(clientId)
52   );
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```
- CT Overview (Right):** Lists all test cases and their execution status:
 - DEFAULT
 - CreateSandbox (1) [Pass]
 - CreateSandboxAndConnect (1) [Pass]
 - CreateSandboxAndConnectAndDisconnect (1) [Pass]
 - CreateSandboxAndInviteGuest (1) [Pass]
 - CreateSandboxAndInviteGuestAndGuestConnect (1) [Pass]
 - CreateSandboxAndUninvitedConnect (1) [Pass]
 - CreateSandboxConsumer (1) [Pass]
 - CreateSandboxMultipleClients (3) [Pass]
 - CreateSandboxNoModel (1) [Pass]
 - CreateSandboxNoModelNoTool (1) [Pass]
 - CreateSandboxNull (1) [Pass]
- Problems/Console (Bottom):** Shows the 'CT Test Case result' tab with a table of test results:

Trace Test case	Result
SelectOS(3, 1)	Error 4071: Precondition failure: pre_SelectOperatingSystem in 'DEFAULT' at line 56:35
SelectToolFromRepository(3, 1)	N/A
SelectModelFromRepository(3, 1)	N/A



Combinatorial Testing traces

• Validating the Sandbox Creation


operations


```
SetupClients: ClientId ==>()
SetupClients(cId) == validClients:= validClients union
{cId};
```


```
SetupOSs: OSId ==> ()
SetupOSs(osId) == brokerst.validOSs:= brokerst.validOSs
union {osId};
```


```
SetupProviders: ClientId ==> ()
SetupProviders(cId) == brokerst.providers :=
brokerst.providers union {cId};
```

```
SetupConsumers: ClientId ==> ()
SetupConsumers(cId) == brokerst.consumers :=
brokerst.consumers union {cId};
```

 CreateSandboxMultipleClients (3)

 Test 000001

 Test 000002

 Test 000003

CreateSandboxMultipleClients:

SetupClients(1);

SetupClients(2);

SetupClients(3);

SetupProviders(1);

SetupConsumers(2);

SetupOSs(1);

SetupTools(1);

SetupModels(1);

let clientId **in set** {1, 2, 3}

in

(SelectOS(clientId, 1);

SelectToolFromRepository(clientId, 1);

SelectModelFromRepository(clientId, 1);

LaunchNewSandbox(clientId)

);

Null user





Results

- Suggestions to the implementation team
- Explicit roles for Sandbox creation
- Covered the current access functionality
- 11 traces expanding to 13 tests
- 2 seconds analysis time
- Small effort in validation – security properties captured as pre and post conditions





Conclusion and future work

- VDM-SL is a good fit for the access analysis
- Combinatorial Testing provides a powerful analysis tool
- An explicit permission for Sandbox creation proposed to the implementation team
- Expand the model to cover aspects of federated cloud
- Use the model in order to iterate on the table of permissions
- Utilize combinatorial testing to cover more scenarios



Thank you



Thank you for your attention!

