PD-3 "Policy-based Distributed Data Dissemination" project

Rohit Ranchal, Denis Ulybyshev, Prof. Bharat Bhargava, Dr. Pelin Angin, Dr. Lotfi ben Othmane, Prof. Leszek Lilien,

Tutorial

Purdue University, 2015

Table of Contents

Initial Setup

Supported OS: Linux

1. Create a <Project_Folder>, go to that folder
   $ cd Project_Folder

and clone the source code from the repository:
   $ git clone https://code.google.com/p/absoa/

2. go to folder *absoa/console* and install ***npm***:
   $ cd console
   $ sudo npm install

node_modules folder must appear in the project directory. Then do the global update of “Express” framework :
  $ npm update -g express

3. Go to *Project_Folder/absoa/scenarios* , then go to every particular scenario and run 'install' script. It will install ***npm*** for every service. For example, for *healthcare* scenario, go to folder *absoa/scenarios/healthcare* and run 'install' script, which will install *npm* for 5 services: doctor, hospital, laboratory, paramedic, pharmacy.
   $ ./install

4. Install ***mysql*** :
   $ apt-get install mysql-server-5.6

5. Set up MySQL database:
  $ mysql -u root -p < db/db.sql

6. Install ***Java Runtime Environment***. The project was tested under JRE version 8.

7. Run the project:
  $ ./start

8. Open the browser, go to *http://localhost:3000/* (default port for Node is 3000).
Internet connection is required. You should see the following page:

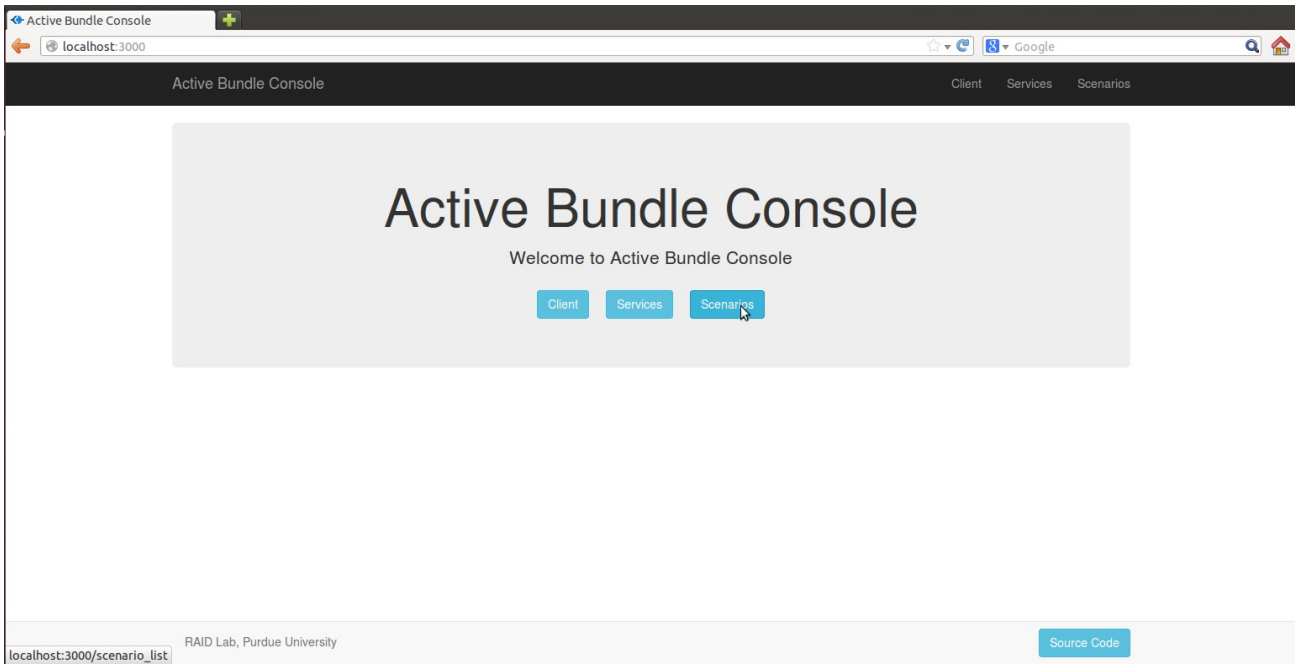

Fig.1. Active Bundle Console

9. Press “Scenarios” button. Then you will see the following page with scenarios:

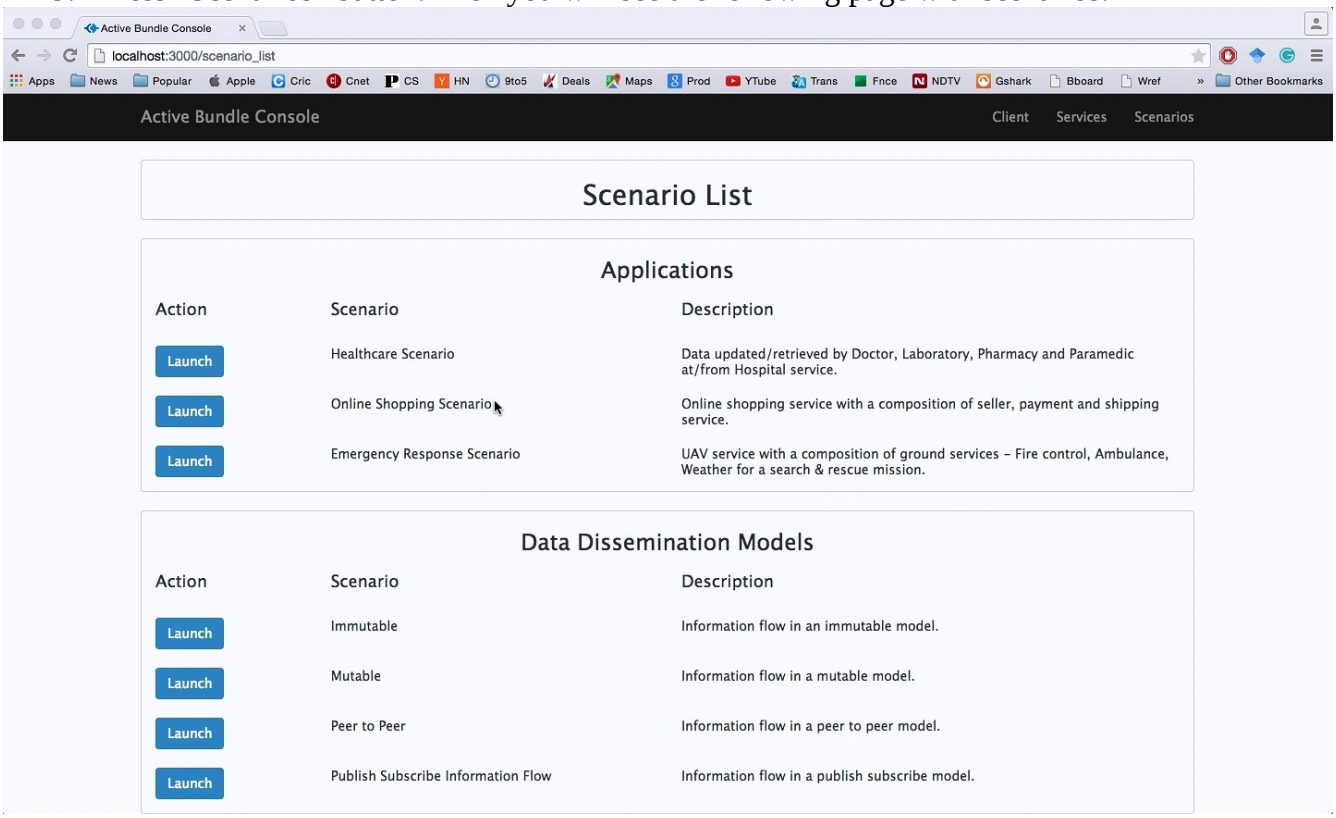

Fig.2. Scenario List

10. Choose the scenario you want to try and press the corresponding “Launch” button.

Emergency response scenario

In this scenario, there are 4 services: UAV, Fire Control, Ambulance, Weather. Their composition is a basis for “search and rescue” mission.

If the scenario is launched in an insecure browser then the Active Bundle (AB) is considered to be untrusted because it was created in an insecure browser. Therefore, AB is sent to the cloud for an execution. AB can't be executed in a service's domain due to privacy concerns, because browser was detected as an insecure one.

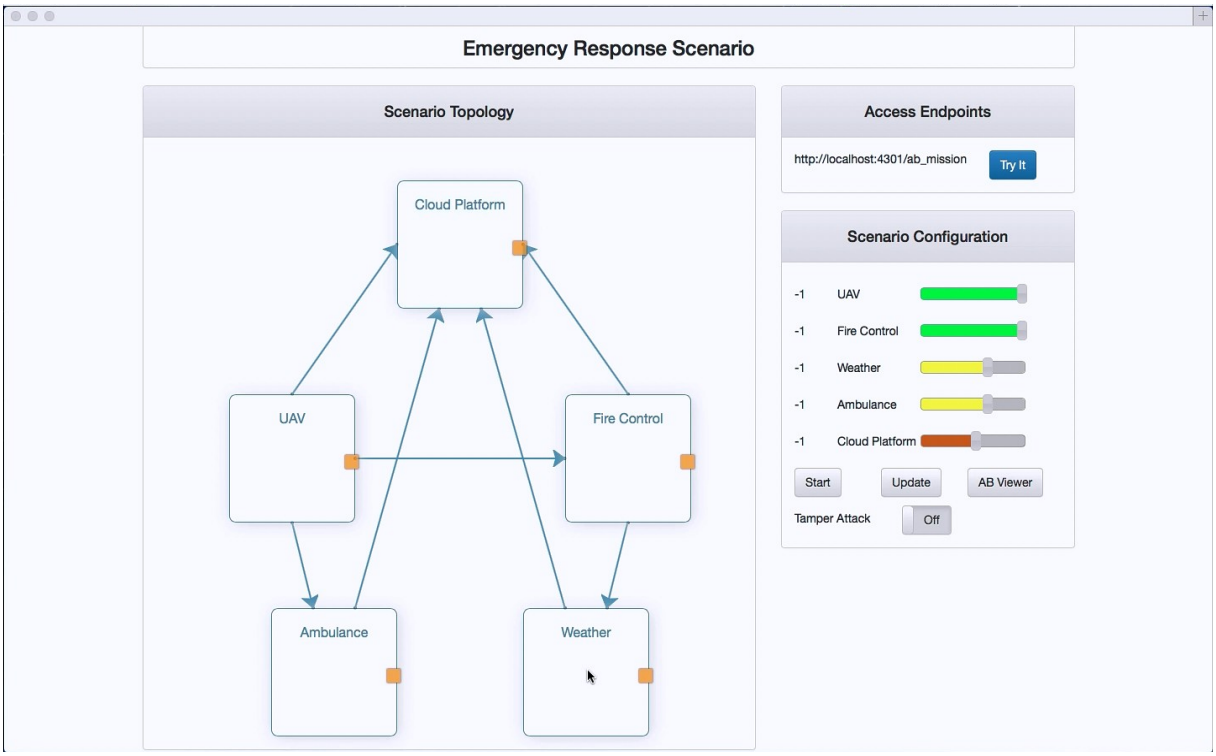

Fig.3. Initial state of 'Emergency Response' scenario in an insecure browser

To start the services, press 'Start' button. Button text will change from 'Start' to 'Stop' and the ID for every service will change from -1 to a new service ID, assigned by an operating system.

As you can see in the 'Scenario Topology' section in the figure 4 below, the cloud platform has received an AB: message 'AB received' appeared for the cloud platform.

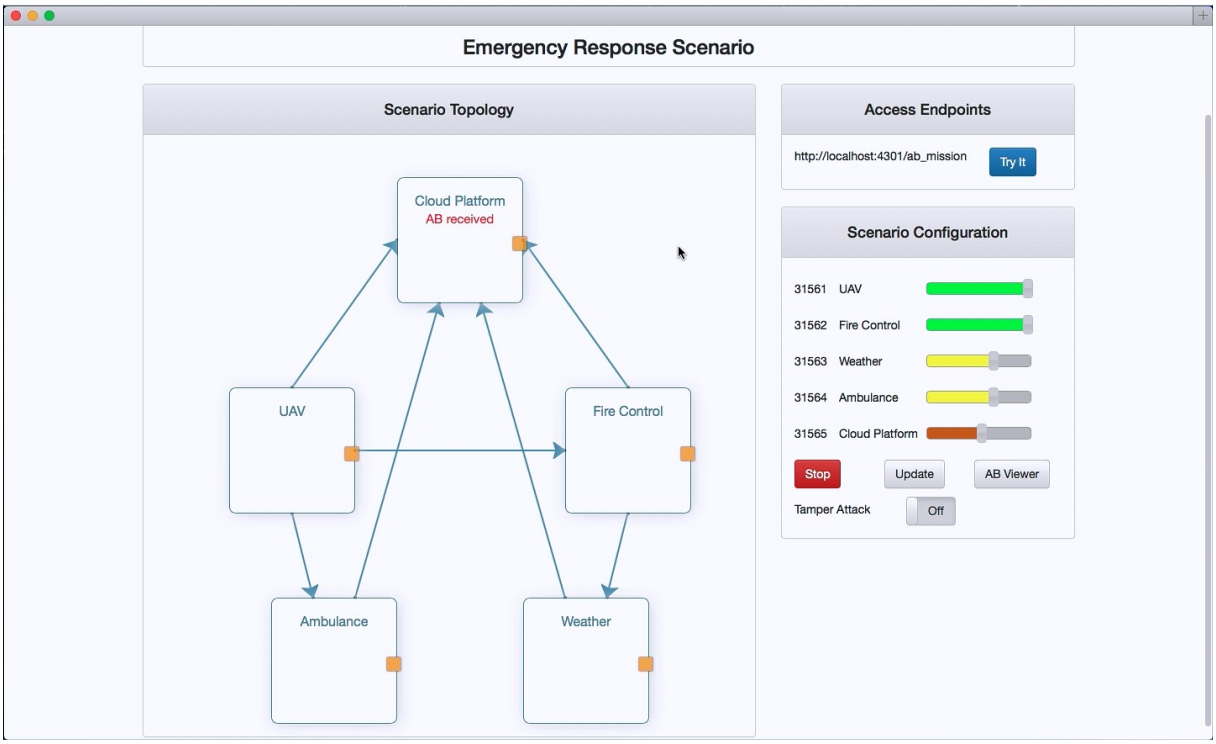

Fig.4. Services have been started in 'Emergency Response' scenario in an insecure browser

Now to start a 'search and rescue' mission we need to press 'Try It' button. After that the pop-up window with message “Search & Rescue started” will appear. Press 'OK' button in this pop-up window.

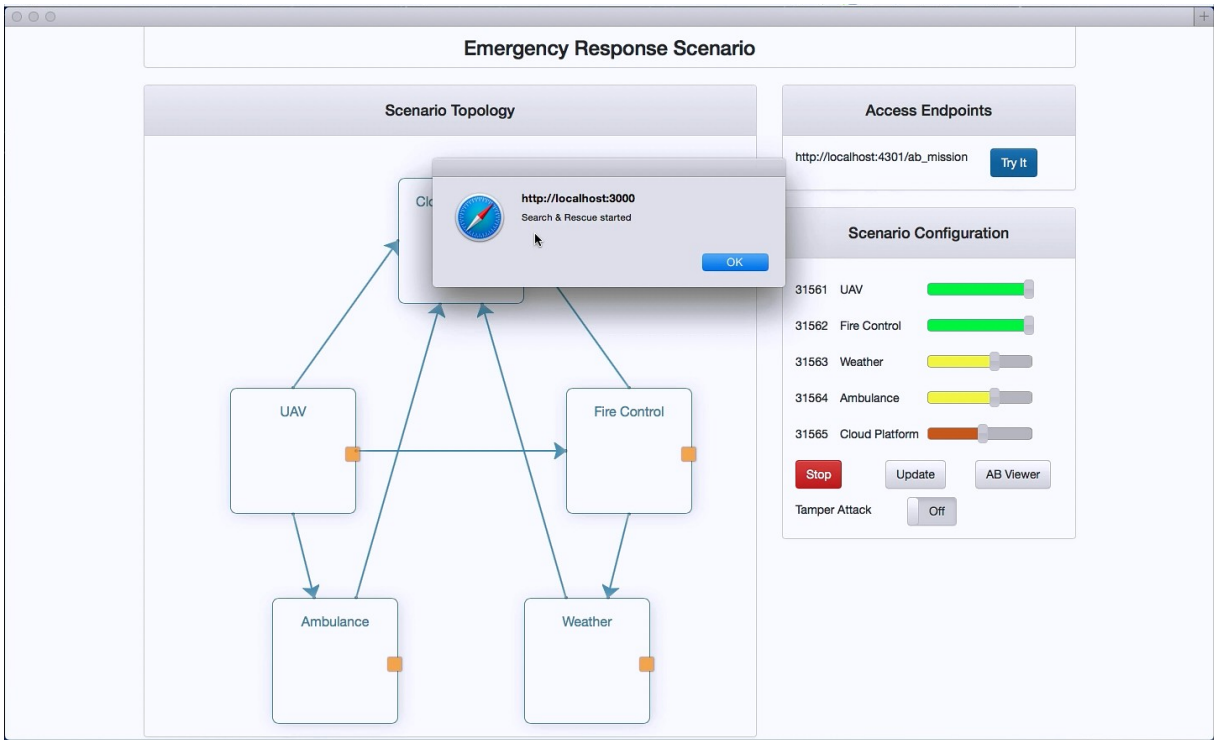

Fig.5. 'Search & Rescue' mission has been started in an insecure browser

As a result, each service interacts with a cloud platform to access only those data from AB for which the service is authorized.

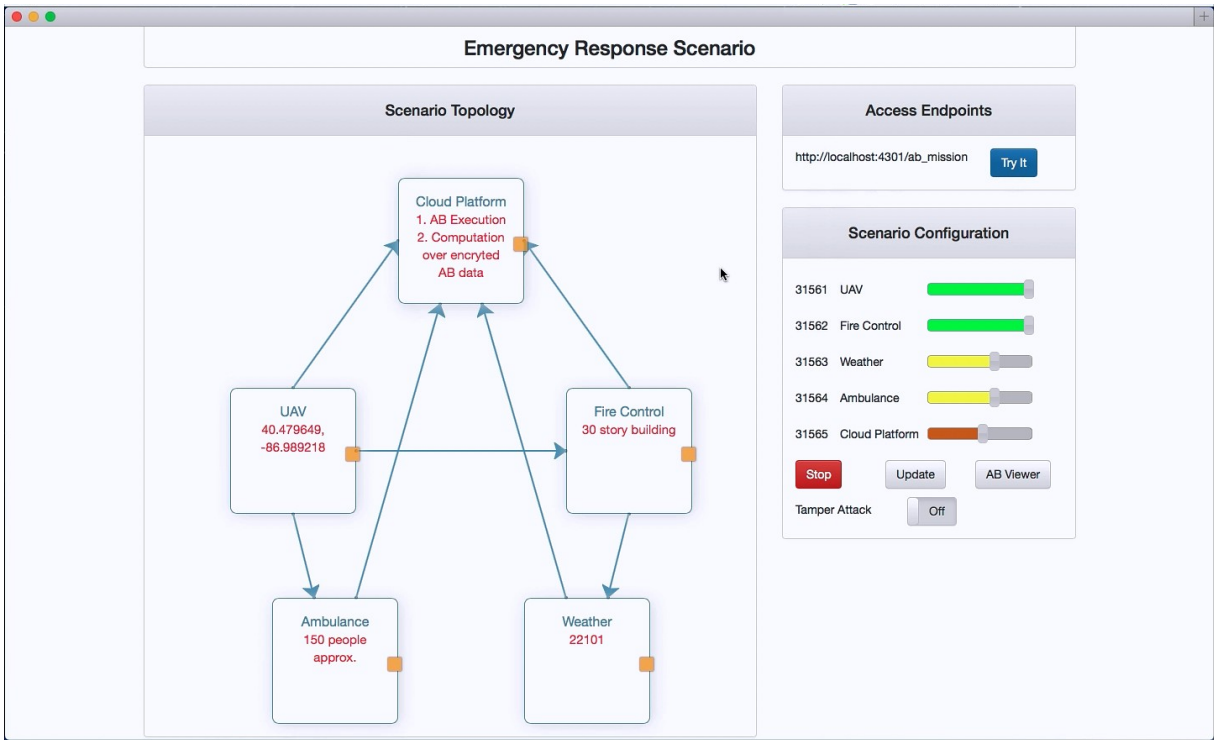

Fig.6. Data dissemination between services in 'Search & Rescue' mission in an insecure browser

AB can be executed on a cloud platform depending on a trust level of the platform. Data from AB can be disclosed to services according to policies. As you can see from Fig.6 above, 'UAV' service gets access only to geographical coordinates of an area that needs help. 'Fire Control' service gets access only to the building-related information, e.g. how many levels are there in the building. 'Weather' service gets access only to ZIP-code of an area which needs to be served by the mission. 'Ambulance' service gets access only to the number of people that need help in the corresponding area.

If the same 'Emergency Response' scenario is launched in a secure browser then AB can be executed in a service's domain.

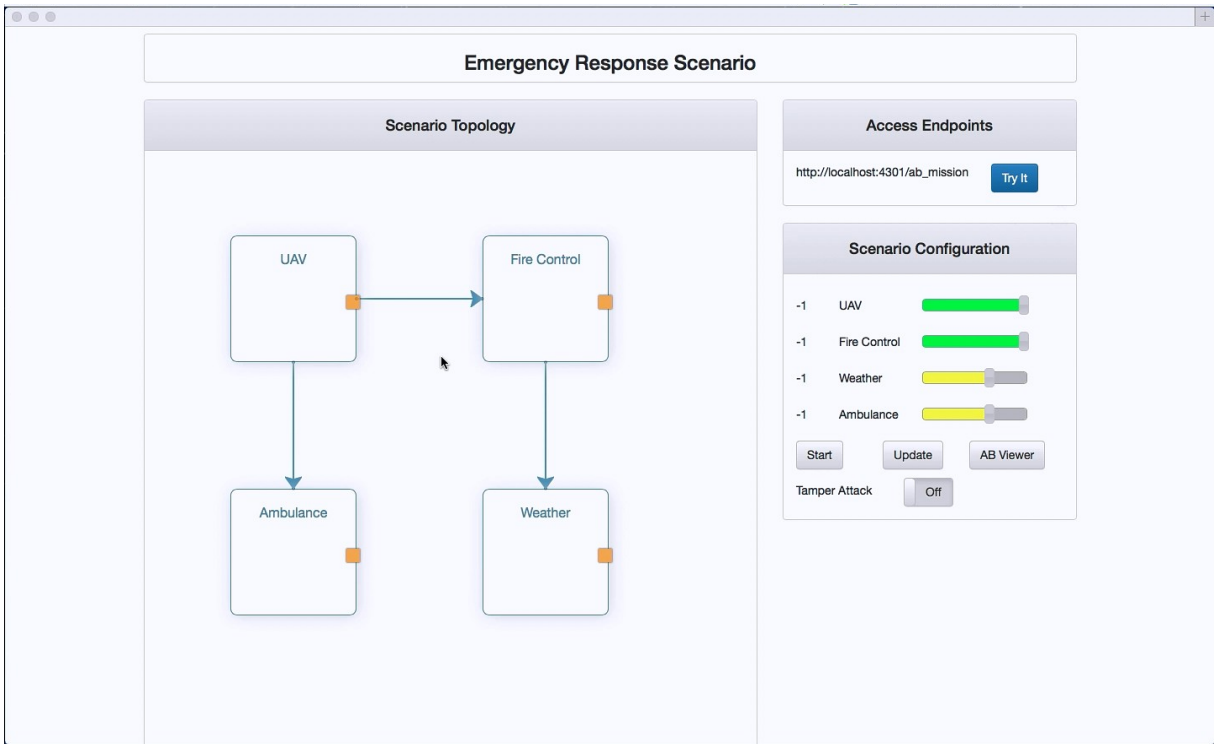

Fig.7. Initial state of 'Emergency Response' scenario in a secure browser

To start the services, press 'Start' button. Button text will change from 'Start' to 'Stop' and the ID for every service will change from -1 to a new service ID, assigned by an operating system.

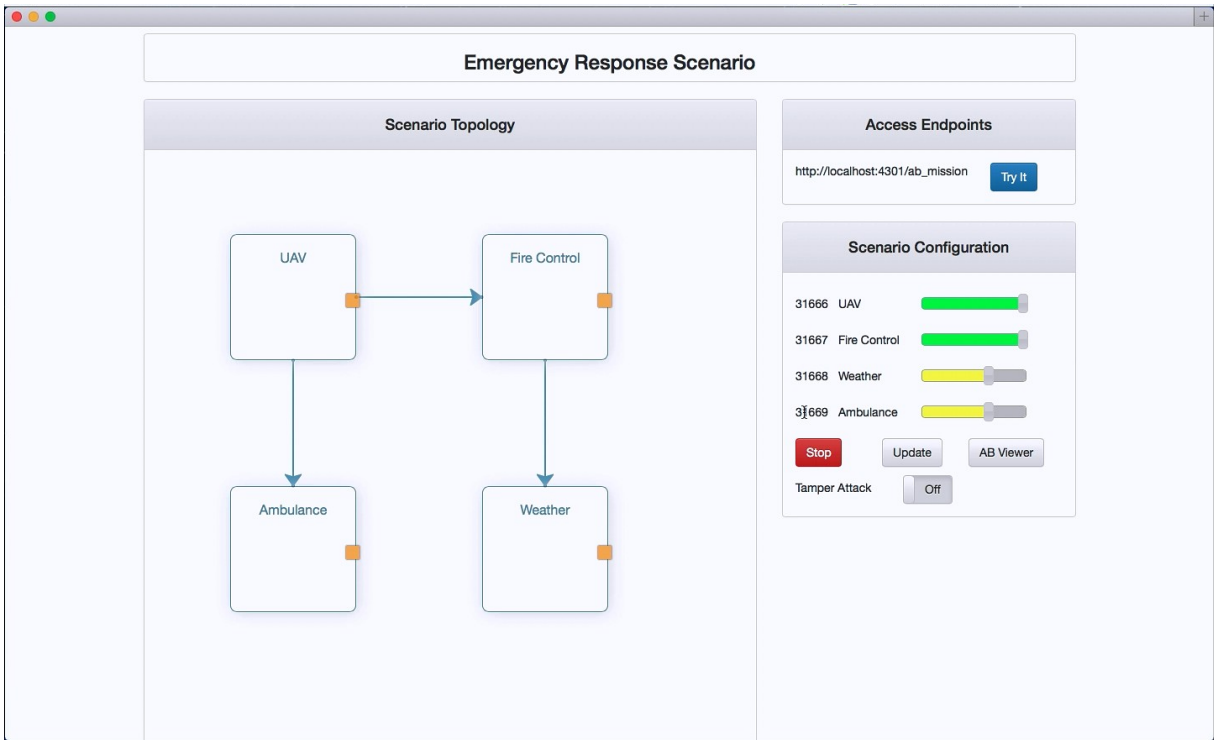

Fig.8. Services have been started in 'Emergency Response' scenario in a secure browser

Now, in the same way as we did in case of an insecure browser, we need to press 'Try It' button to start a 'search and rescue' mission. After that the pop-up window with message “Search & Rescue started” will appear. Press 'OK' button in this pop-up window.

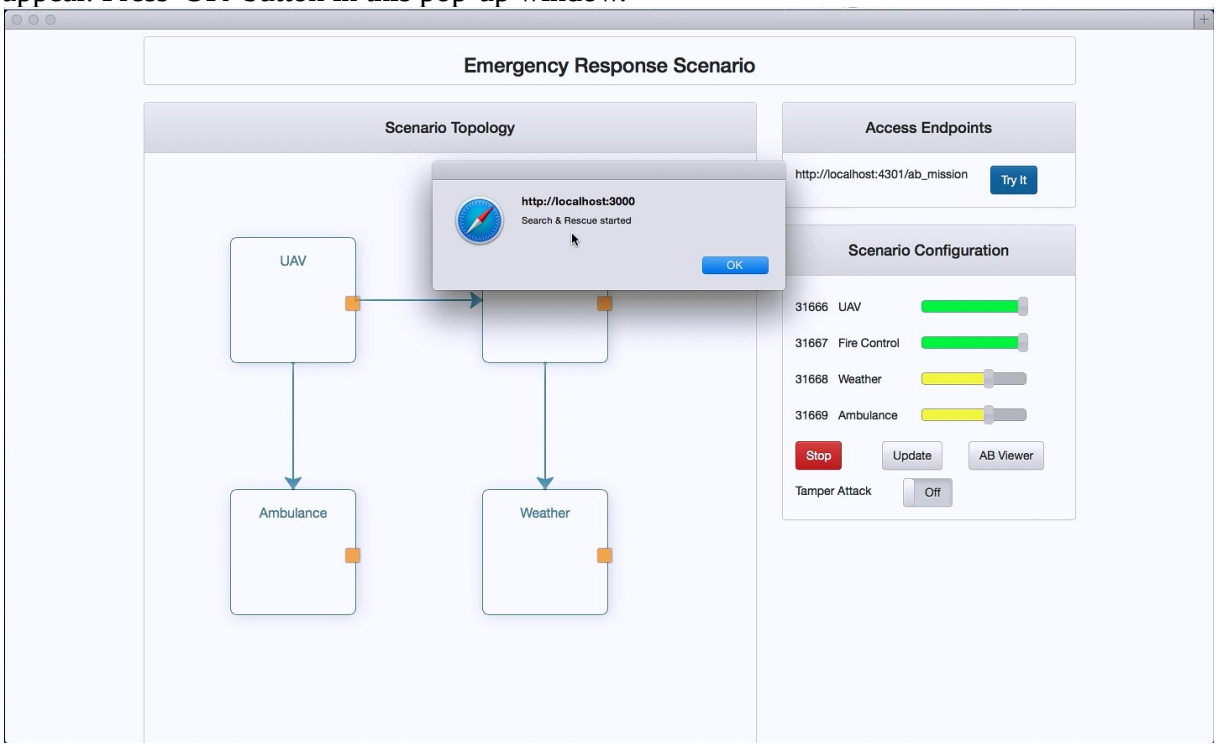

Fig.9. 'Search & Rescue' mission has been started in a secure browser

As a result, each service receives an AB and interacts with it to access the data from AB for which the service is authorized.

Fig.10. Data dissemination between services in 'Search & Rescue' mission in a secure browser

The services get access to the corresponding data from AB on a 'need-to-know' basis, as defined by policies of AB. Each of 4 services gets access to the same type of data as described above for an insecure browser.

Health care scenario

In this scenario, there are 5 services: Hospital, Doctor, Laboratory, Paramedic and Pharmacy. AB contains patient's Electronic Healthcare Record (EHR).
Doctor and Laboratory not only can get the corresponding data from AB according to the policies, but they can also update information to AB. For example, doctor can update AB with a test prescription (e.g. X-Ray, blood test etc) and with a prescription (e.g. Omeprazole or some Allergy Medicine etc).
Laboratory can update AB with test results.
Paramedic and Pharmacy can only read the corresponding data they are authorized for from AB.
Hospital service plays a role of a Hospital Information System.

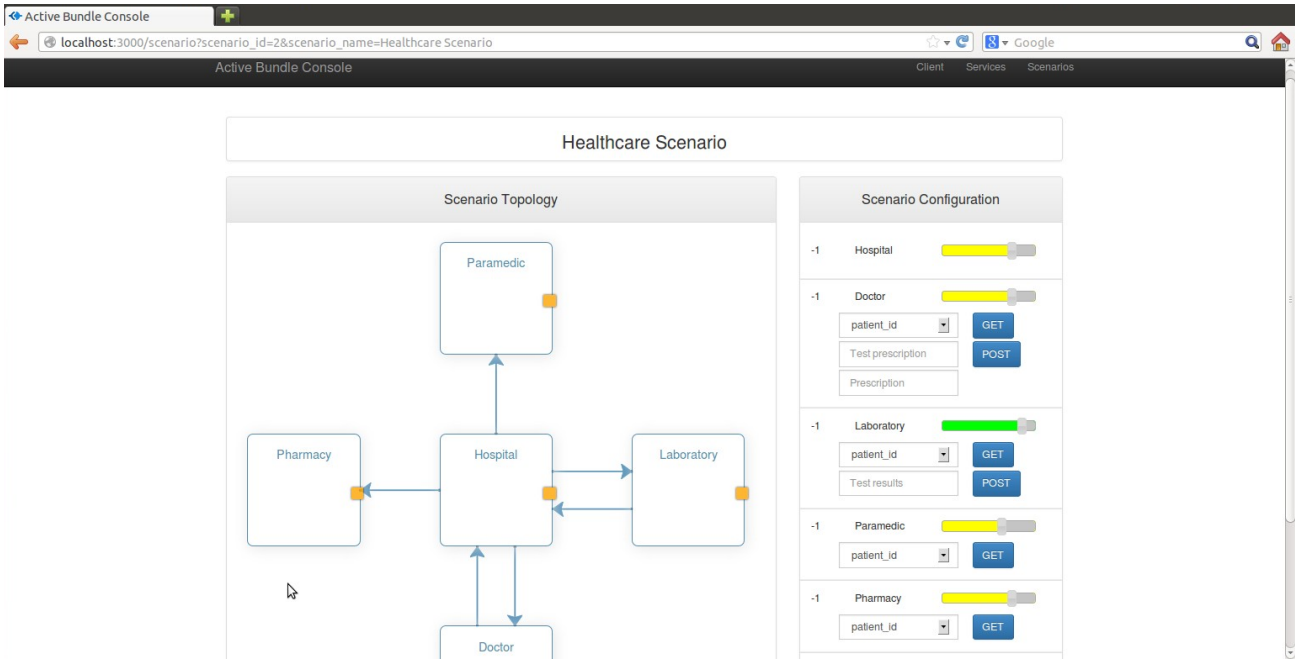

Fig.11. Initial state of a 'Health care' scenario

To start the services, press 'Start' button. Then the five services of a health care scenario will be started and their corresponding process IDs, assigned by an operating system, will be displayed instead of -1 to the left side from process's name. Button text will change from 'Start' to 'Stop'.

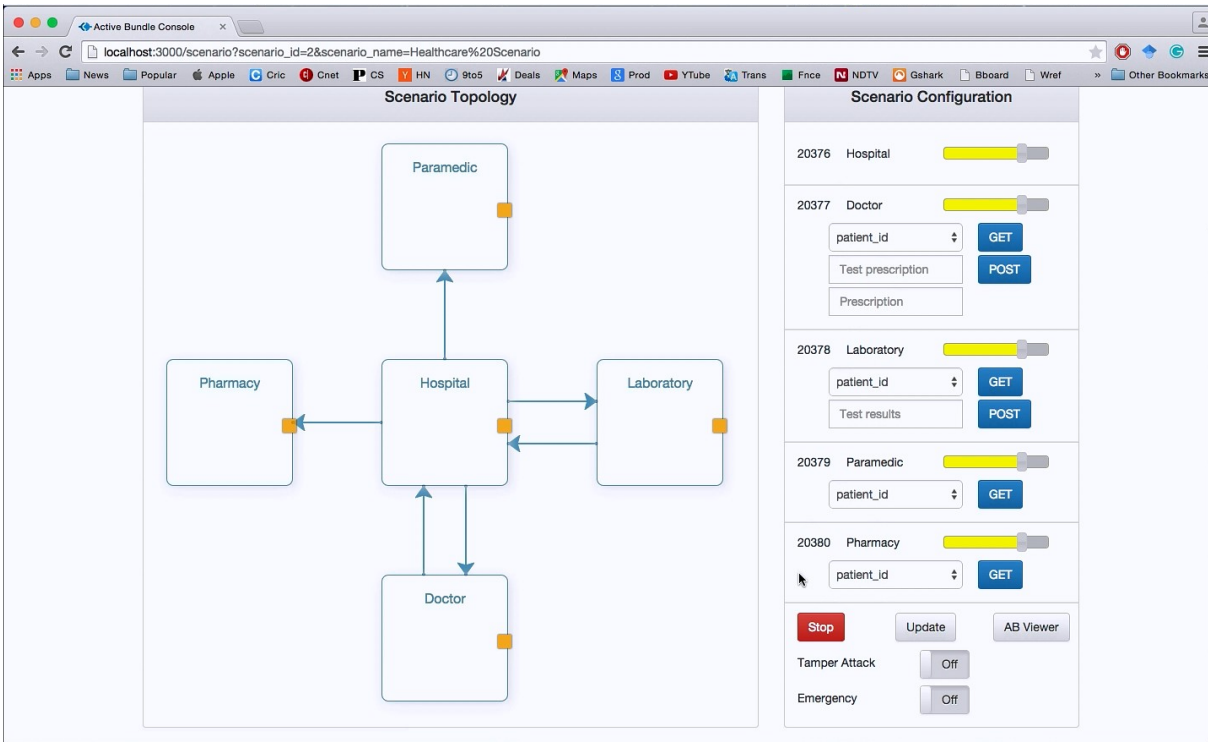

Fig.12. Services have been started in a 'Health care' scenario

Policy-based data dissemination

Every service can access only those data from AB it is authorized for. In a drop-down box below the service's name the user can try to select the type of data to be accessed by a corresponding service and then press 'GET' button. For example, 'Doctor' service (application) can get access to medical data, but can't get access to insurance ID.

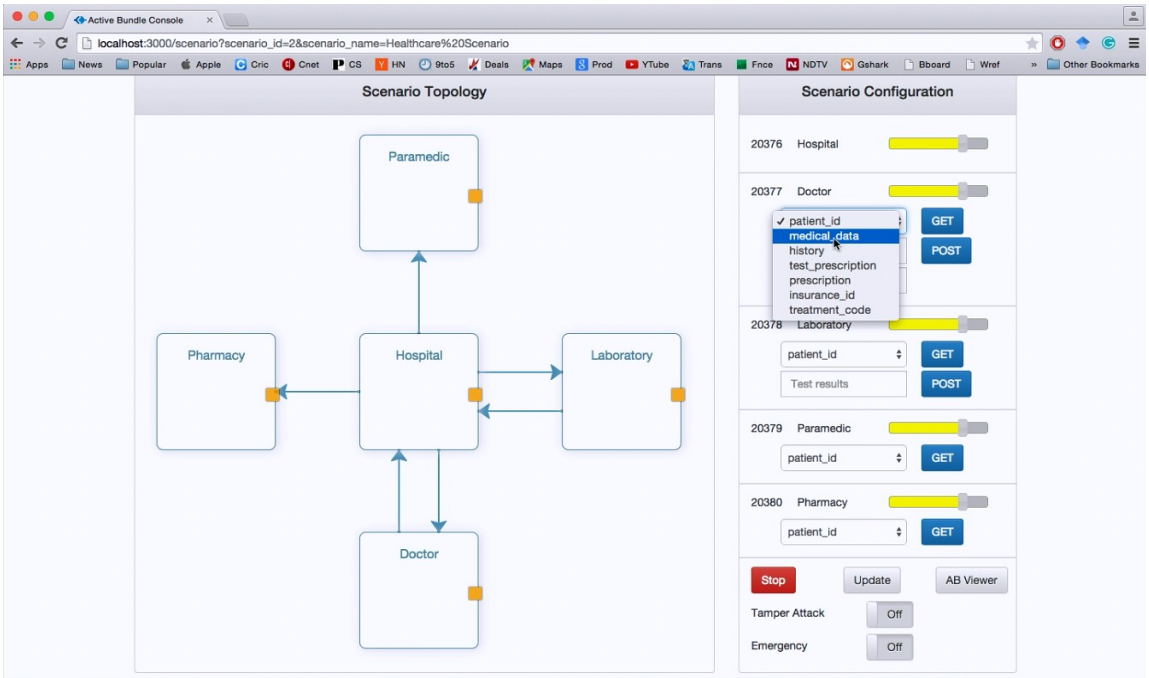

Fig.13. Selection of data to be accessed from 'Doctor' scenario

As we can see in the figure 14 below, doctor is authorized to get access to medical data of a patient and as a result the doctor gets access to patient's cholesterol level [“hdl: 60”]

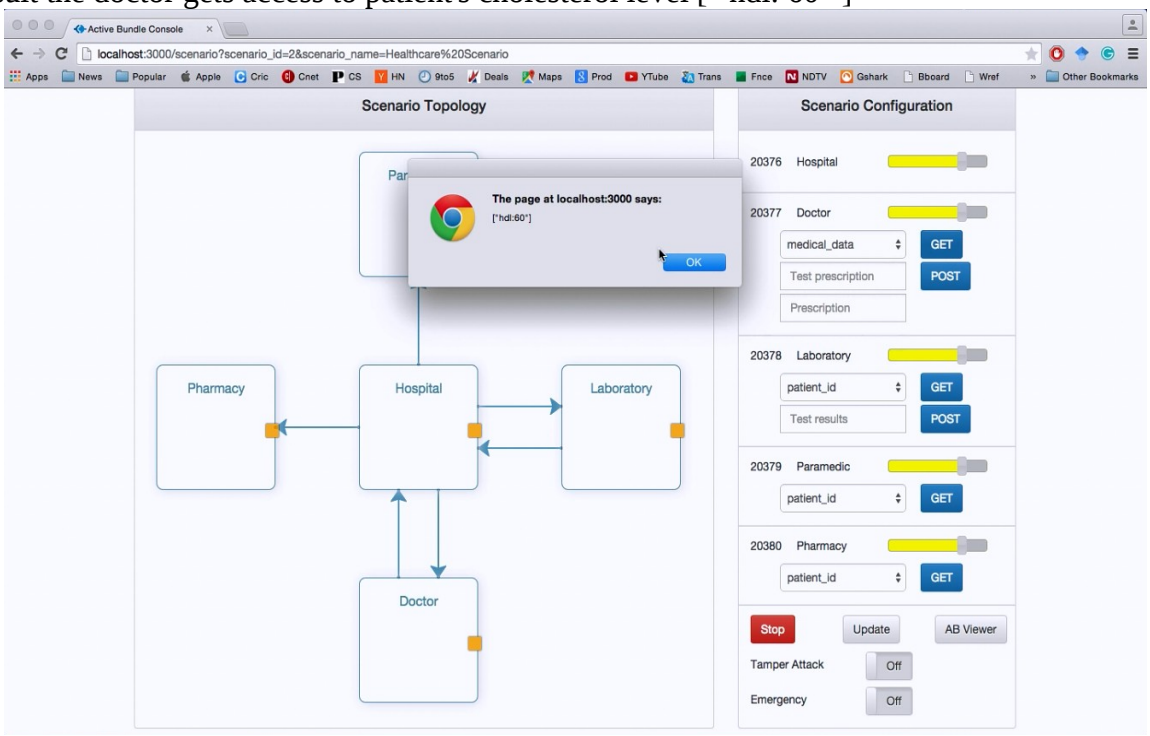

Fig.14. 'Doctor' is able to get access to patient's medical data

Laboratory is authorized to get the test prescription data from AB. As a result, pop-up window with a message [“X-Ray”] appears.

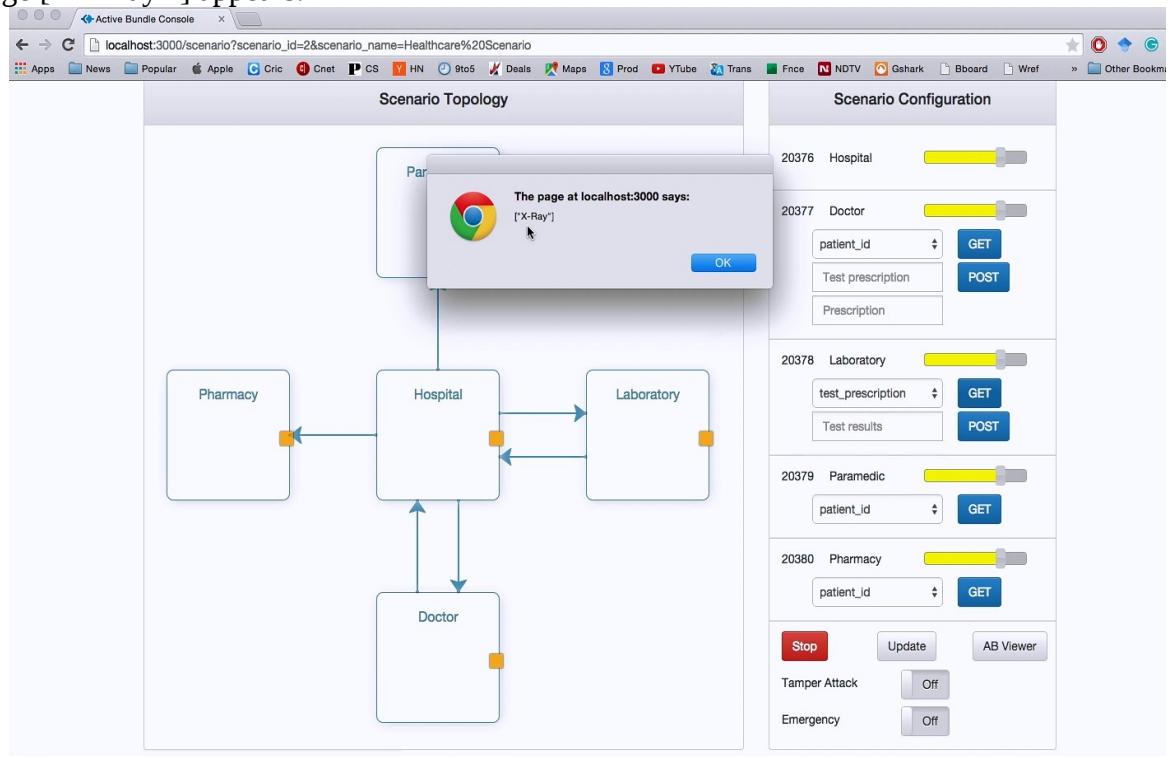

Fig.15. 'Laboratory' is able to get access to patient's test prescription

On the other hand, laboratory can't get access to patient's medical data because policies don't allow that. As a result, pop-up window with a message [“Unauthorized access”] appears.

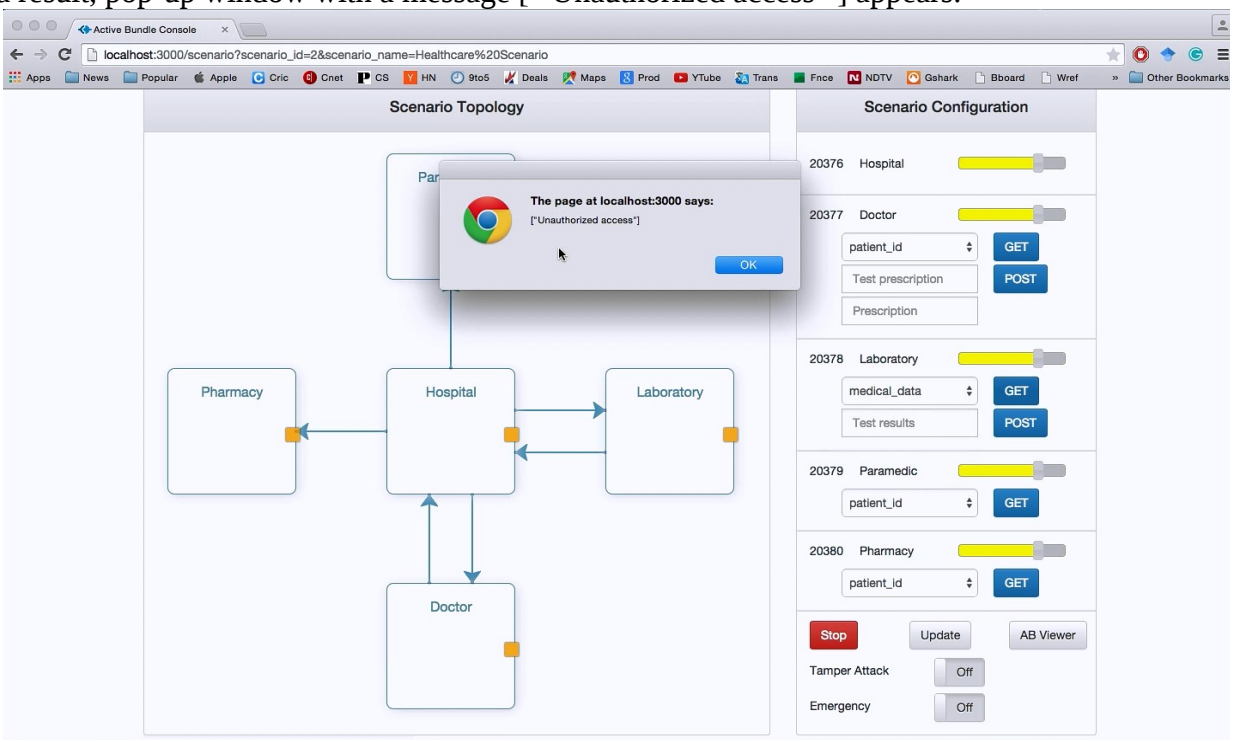

Fig.16. 'Laboratory' is not able to get access to patient's medical data

Trust-based data dissemination

If by using the corresponding slider on the right side of service's name we decrease the trust level for Doctor's application to 2 out of 10 (it is required to press 'Update' button after that) then Doctor will not be able to access patient's medical data anymore.

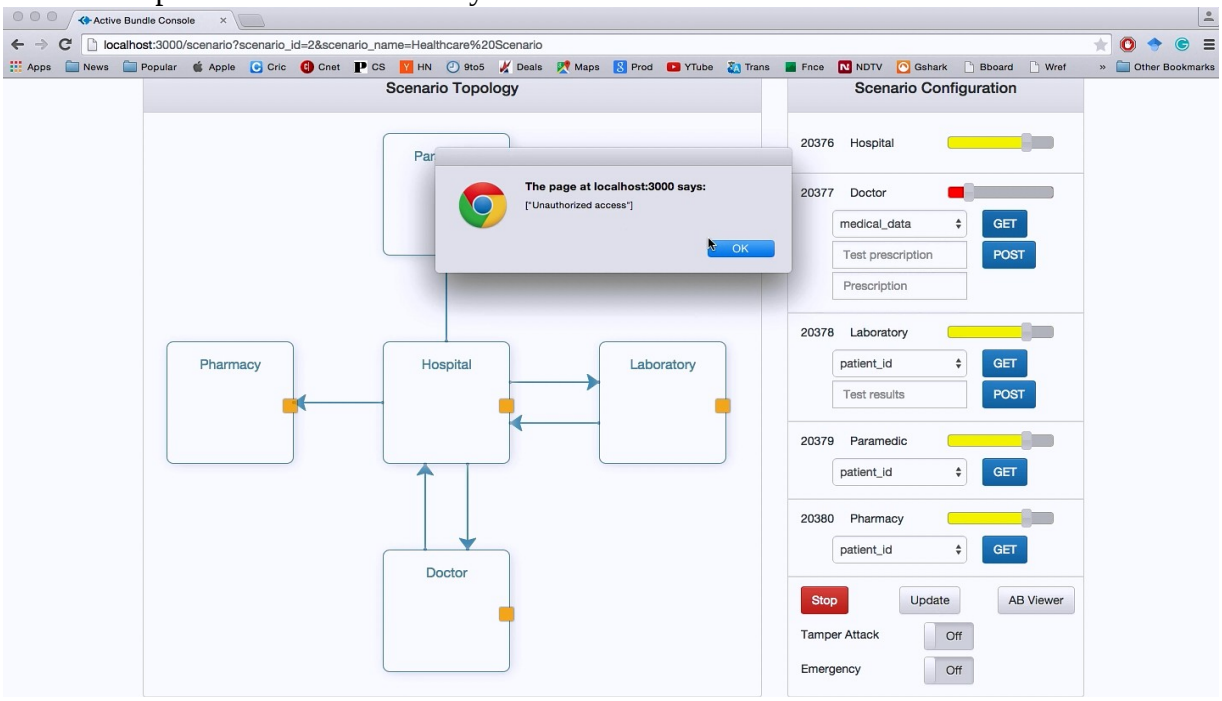

Fig.17. 'Doctor' with low trust level is not able to get access to patient's medical data

As a result, pop-up window with a message [“Unauthorized access”] appears.

In contrast to denied access to medical data due to low trust level, Doctor is able to access medical data of a patient, as shown on fig. 14, if the trust level of a service is sufficient.

Context-based data dissemination

Our implementation supports context-based data dissemination. The idea is that under different contexts the same services can get different set of data from AB. E.g. under emergency context the 'Paramedic' service can get access to more data compared to normal context.

To turn the emergency context on the user needs to move the 'Emergency' switch in 'Scenario Configuration' section to “ON” position. After that the pop-up window with a message “Context changed to: EMERGENCY” appears.

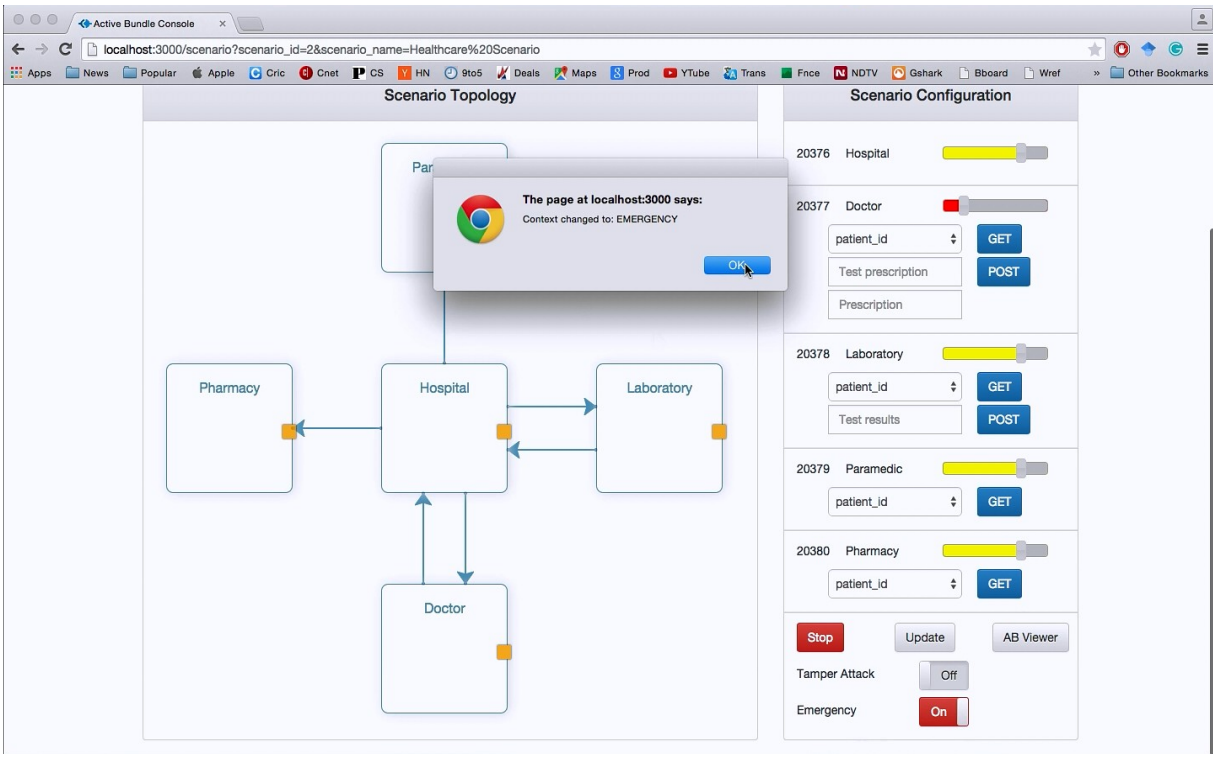

Fig.18. Turning the Emergency context on

If emergency context is ON then 'Paramedic' service can get access to medical data of a patient, as shown on fig. 19 below. After selecting 'medical_data' in a corresponding drop-down box below the 'Paramedic' label user needs to press the corresponding 'GET' button and then the pop-up window with patient's cholesterol level [“hdl: 60”] appears.

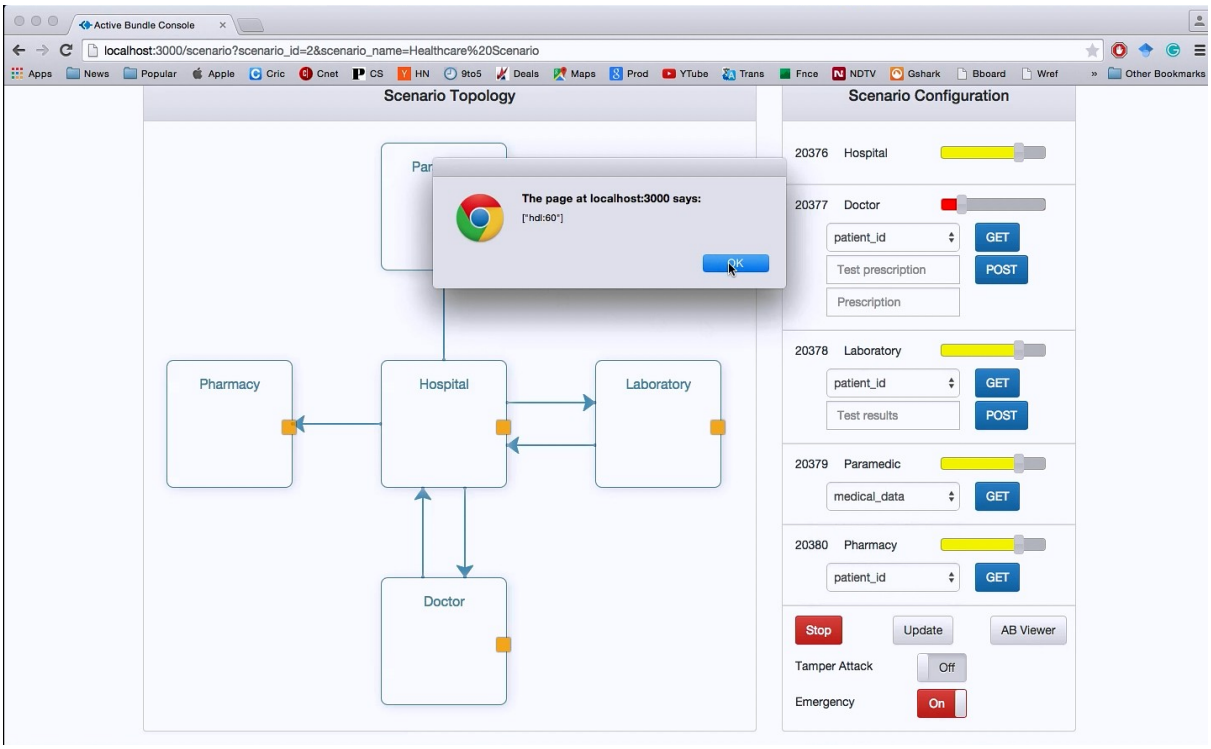

Fig.19. 'Paramedic' is able to get access to patient's medical data in Emergency context

However, if the 'Paramedic' service tries to access patient's medical data under a normal (non-emergency) context then access is denied and pop-up window with a message [“Unauthorized access”] appears.

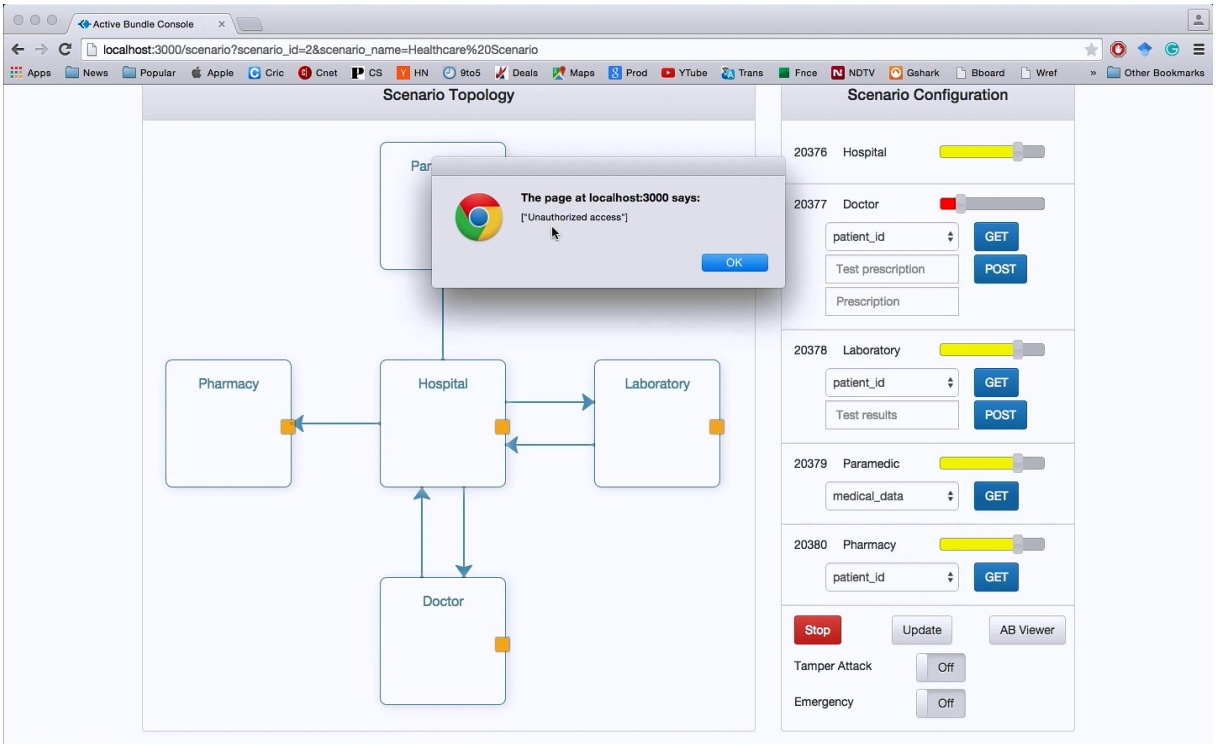

Fig.20. 'Paramedic' is not able to get access to patient's medical data in non-emergency context

AB under tamper attacks

Tamper attack tries to modify policies of AB in order to insert a new malicious subject to policies to get unauthorized access to data.
To turn tamper attack on the user needs to move the 'Tamper Attack' switch in 'Scenario Configuration' section to “ON” position. After the tamper attack on AB is completed a pop-up window with a message [“Active Bundle Tampered”] appears.

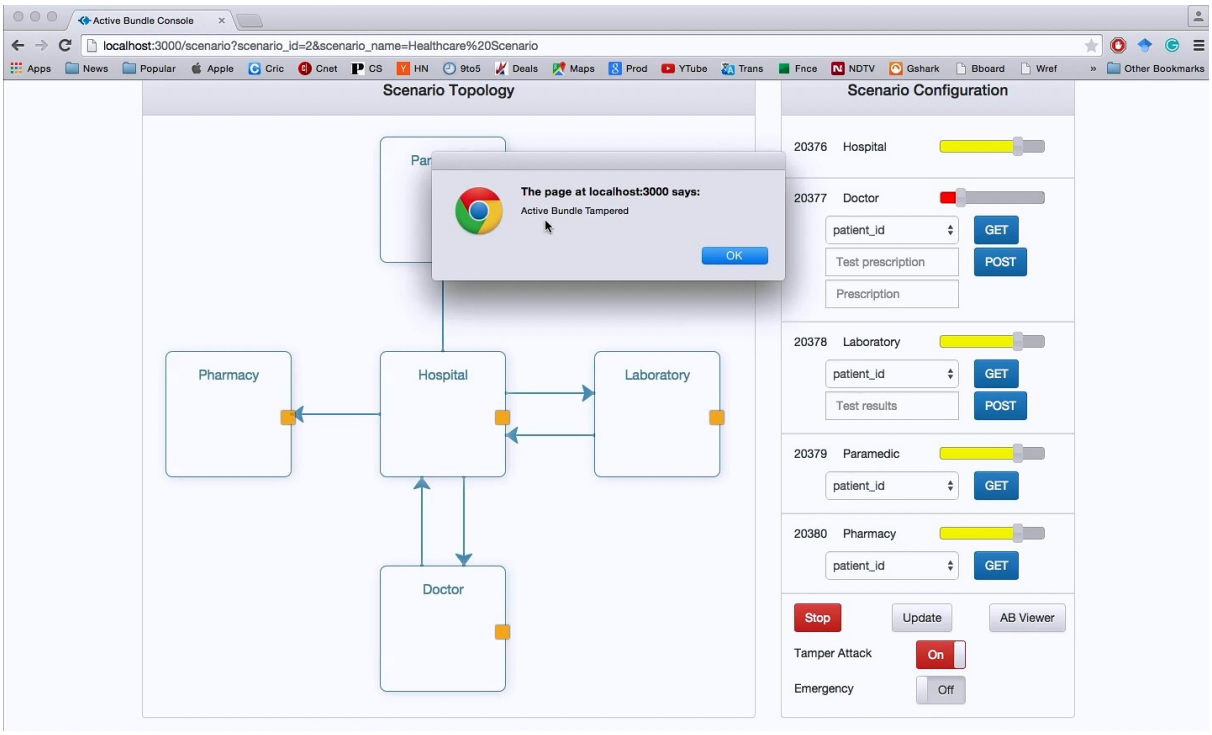

Fig.21. Turning on the tamper attack on AB

To see the current state of AB press the 'AB Viewer' button in the bottom of 'Scenario Configuration' section. JarZilla application needs to be installed to use AB Viewer.

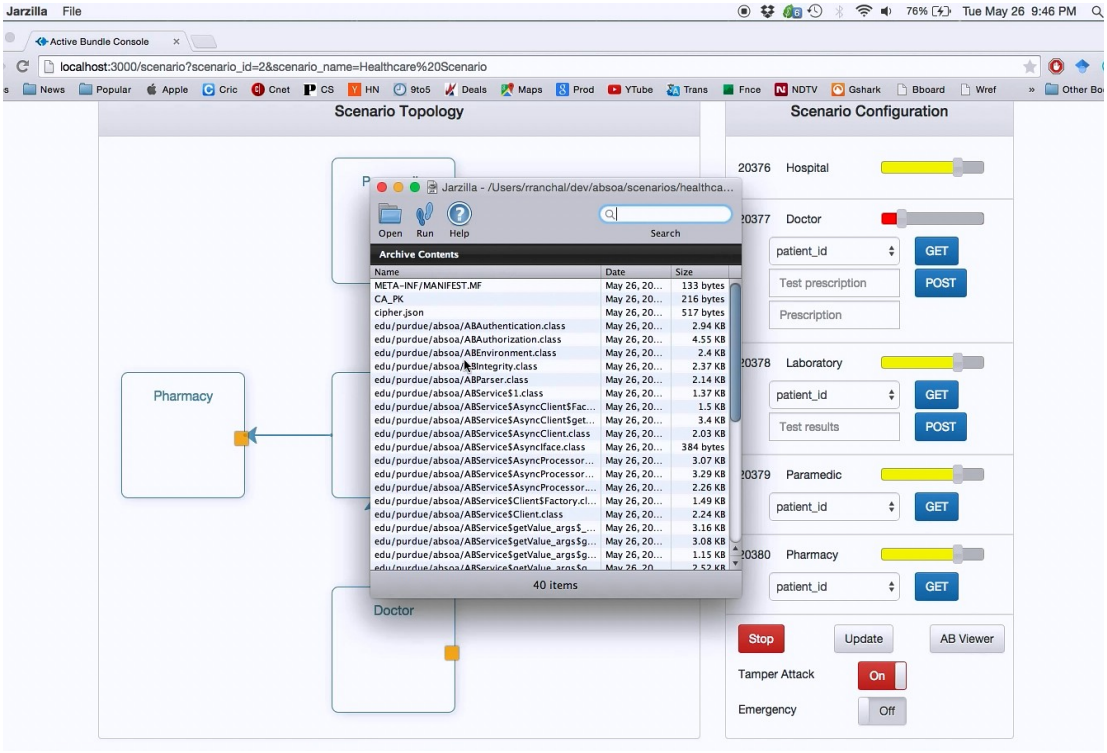

Fig.22. State of AB in AB viewer

To see the encrypted data inside AB double click on cipher.json file.

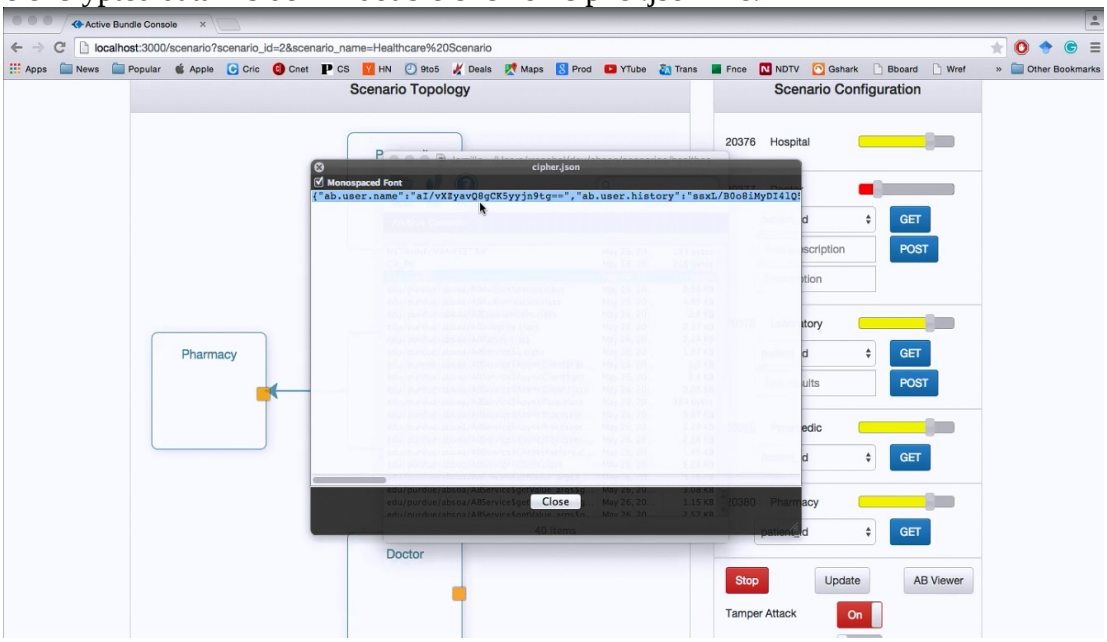

Fig.23. Encrypted data of AB in cipher.json

To see policies double click on policy/policy-0. You can see that malicious subject 'Eve' appeared for each data item.

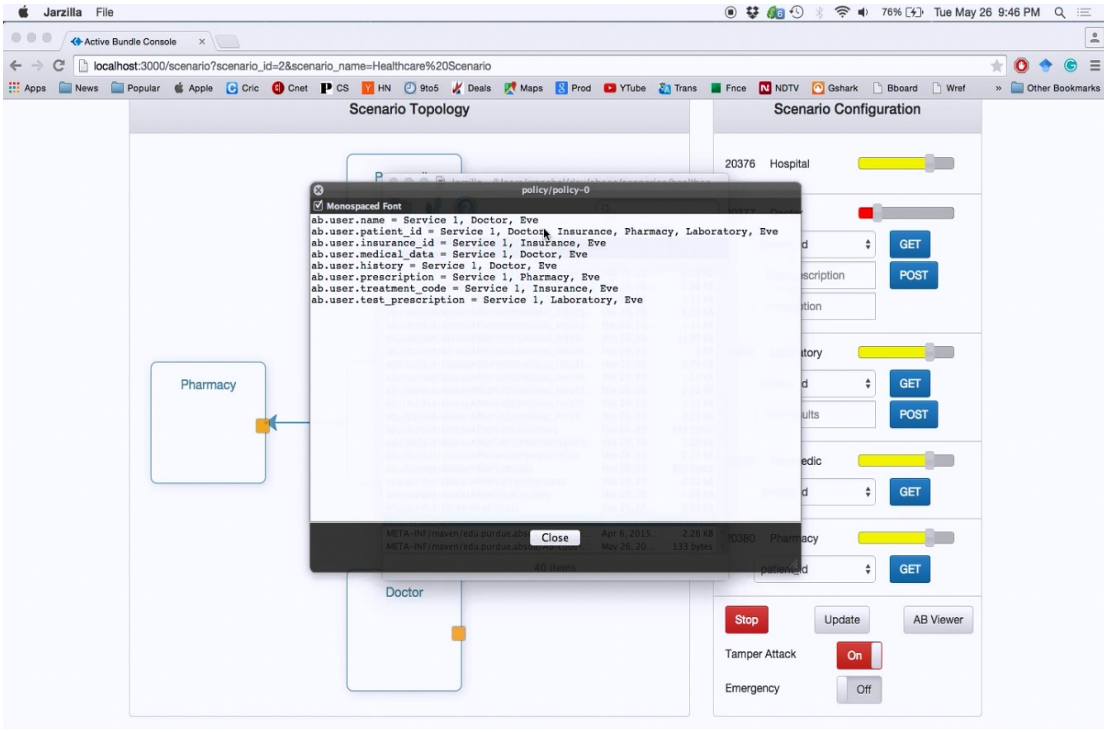

Fig.24. Policies of attacked AB with malicious subject 'Eve' for each data item

Now if 'Pharmacy' service tries to get access to prescription data, for which it is authorized, the access will be denied because AB has been tampered. As a result, pop-up window with a message [“Unauthorized access”] appears.

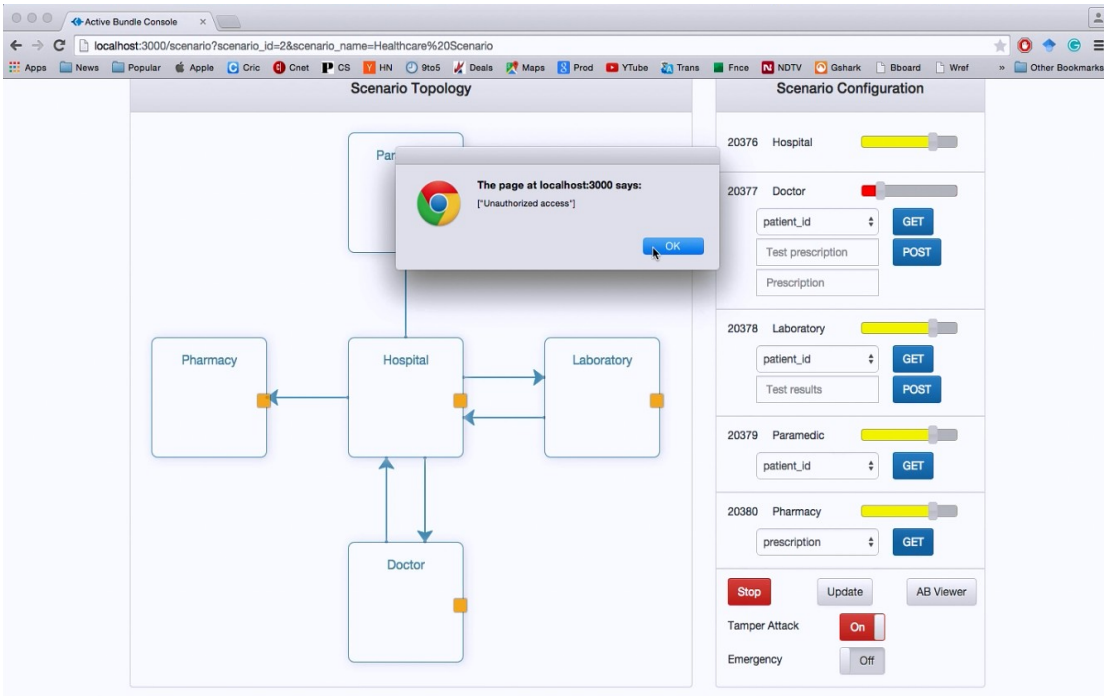

Fig.25. 'Pharmacy' is not able to get access to prescription under tamper attack

If to turn 'Tamper Attack' off then AB will be restored (rolled back to the previous non-malicious state) and a pop-up window with a message [“Active Bundle Restored”] will appear.

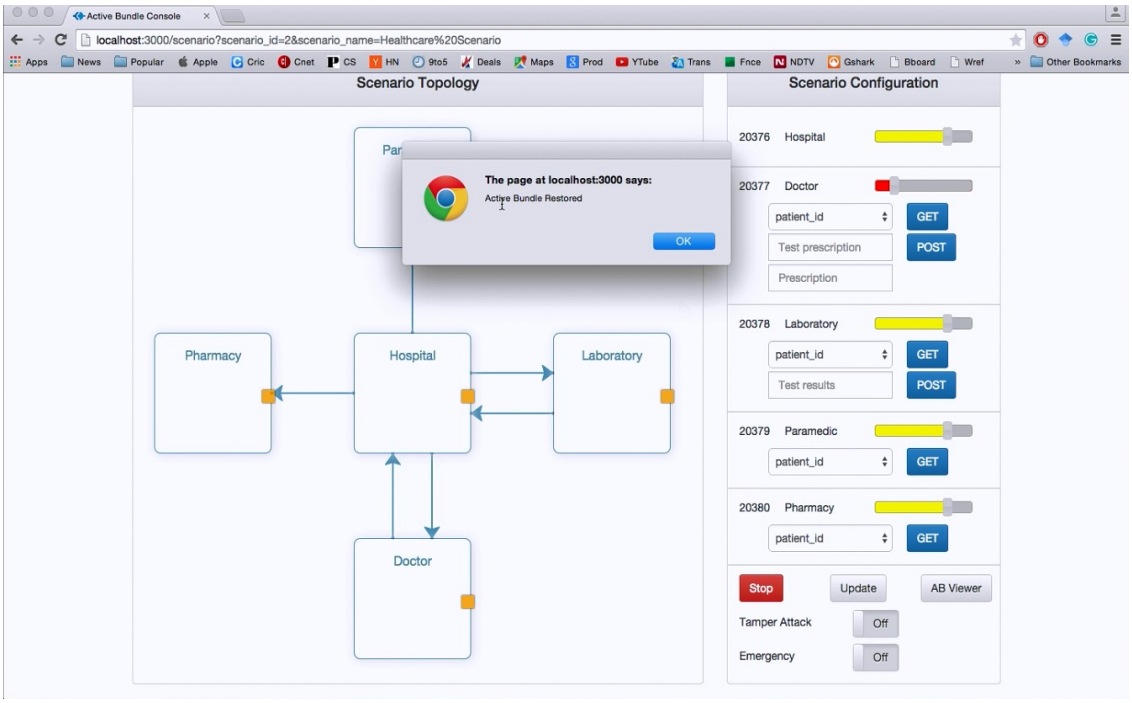

Fig.26. Turning off the tamper attack on AB

AB Viewer can be run in order to see how policies of AB have been changed after restoring AB. To see policies double click on policy/policy-0. You will see that malicious subject 'Eve' disappeared for each data item.

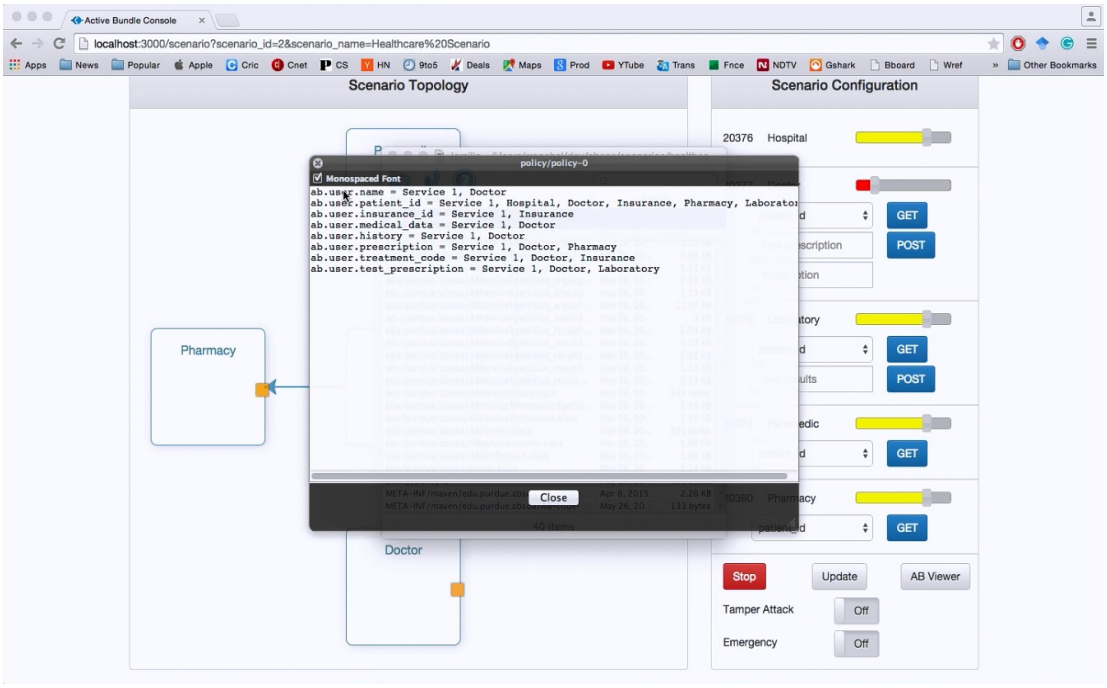

Fig.27. Policies of a normal AB (not under attack)

If we try to access the prescription data from 'Pharmacy' service now it will result in a successful access (in opposite to fig. 25 when AB is under tamper attack) and a pop-up window with a message [“Allergy Medicine”] appears.

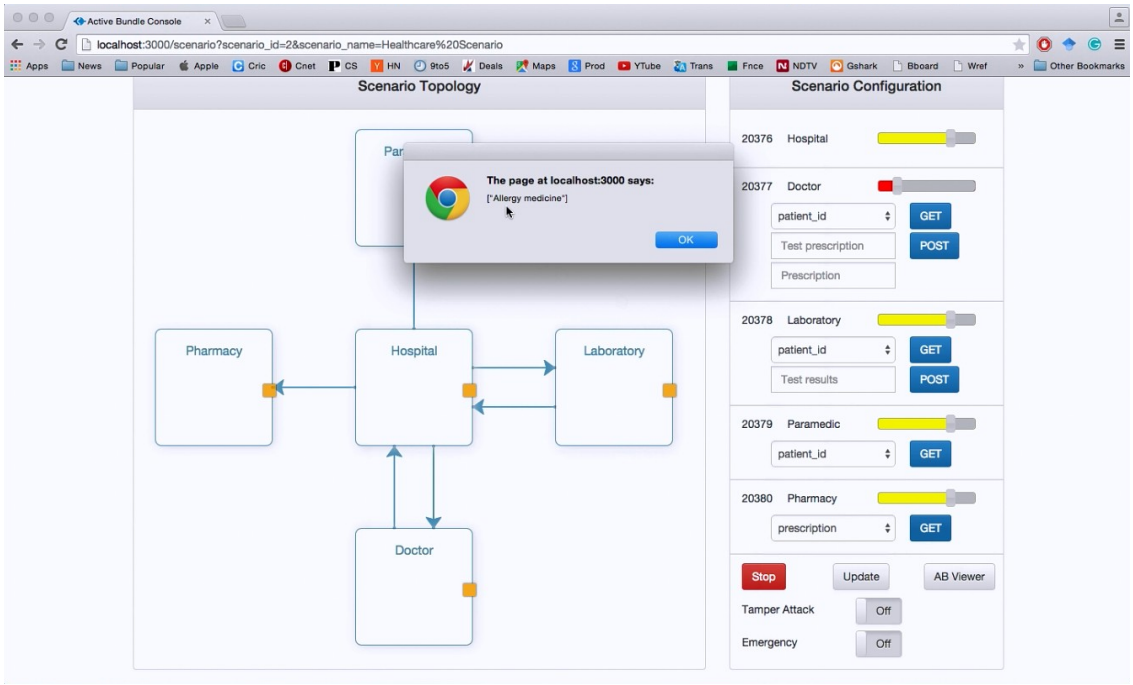

Fig.28. 'Pharmacy' is able to get access the prescription data if AB is not under tamper attack