icoami-5

D7001D Network programming and distributed applications

LAB 4 – Agents

Malin Christoph, icoami-5
Schnitker Jan, jansic-4
Solis Sara, sarsol-5
winter semester 2015

Date: 19.10.2015

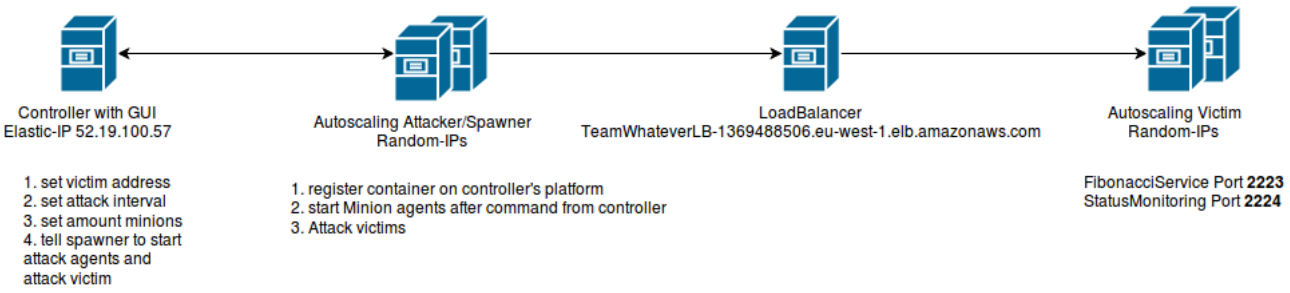
Inhaltsverzeichnis

Agents, the rise of agents

1. Attacker/Spawner

The coordinator agent shall be able to coordinate up to 10.000 attacker agents.

The following diagram shows the architecture of our solution.



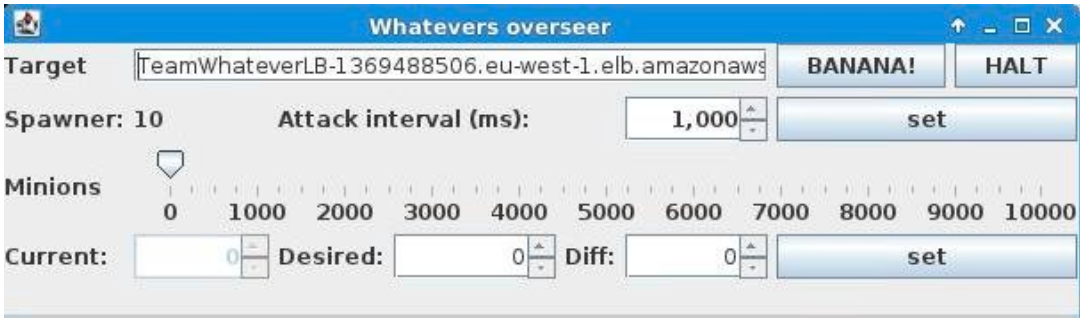
Auto scaling victims
As you can see on the right side there are the victim servers. They are part of an auto scaling group. When the CPU utilization of them is higher than 75% for more than 1 minute they start three additional servers. When the server boots a script get's the latest victim version from github.com and starts the server configured in a script. Next to the Fibonacci calculator also a resource monitoring service is started on port 2224. The extra service gives a good overview about actual CPU, RAM utilization and how many threads are running.

LoadBalancer
The victims stand behind the LoadBalancer which takes all the incoming requests from the attacker agents and distributes them equally on the victims. The load balancer forwards requests to the ports 2223 and 2224 directly to the victims.

Auto scaling attacker/spawner
When an attacker/spawner agents is booting he gets the latest attacker version from github.com and starts a SpawnerContainer(randomNumber) and registers himself at the DFService of the main plattform running on the controller. For all communication between the agents port 4321 is used.

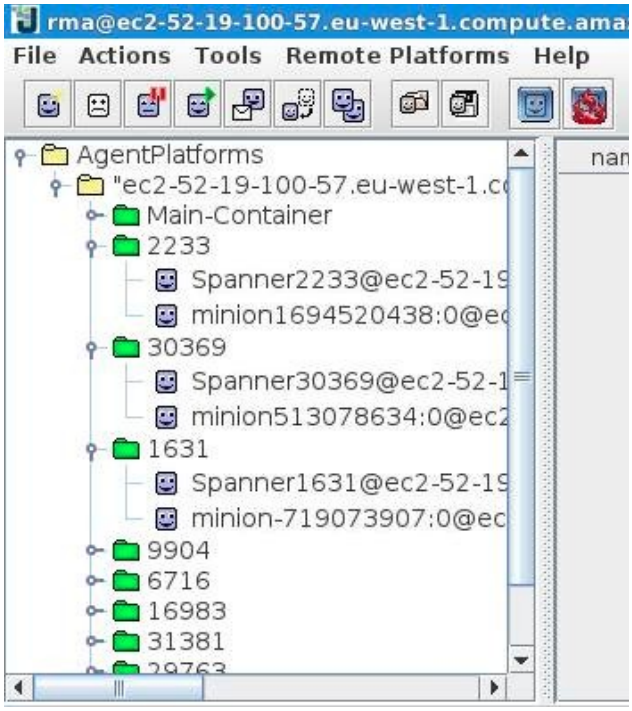
The spawner has a TickerBehaviour where he waits for ACLMessages from the controller giving him new commands. When he get's a command he looks that he has the desired amount of attacker agents running and waits for the message which tells him to start the attack against the victim.

Controller with GUI
The controller has the GUI where the victim address, the attack interval, the amount of attackers/minions can be set.



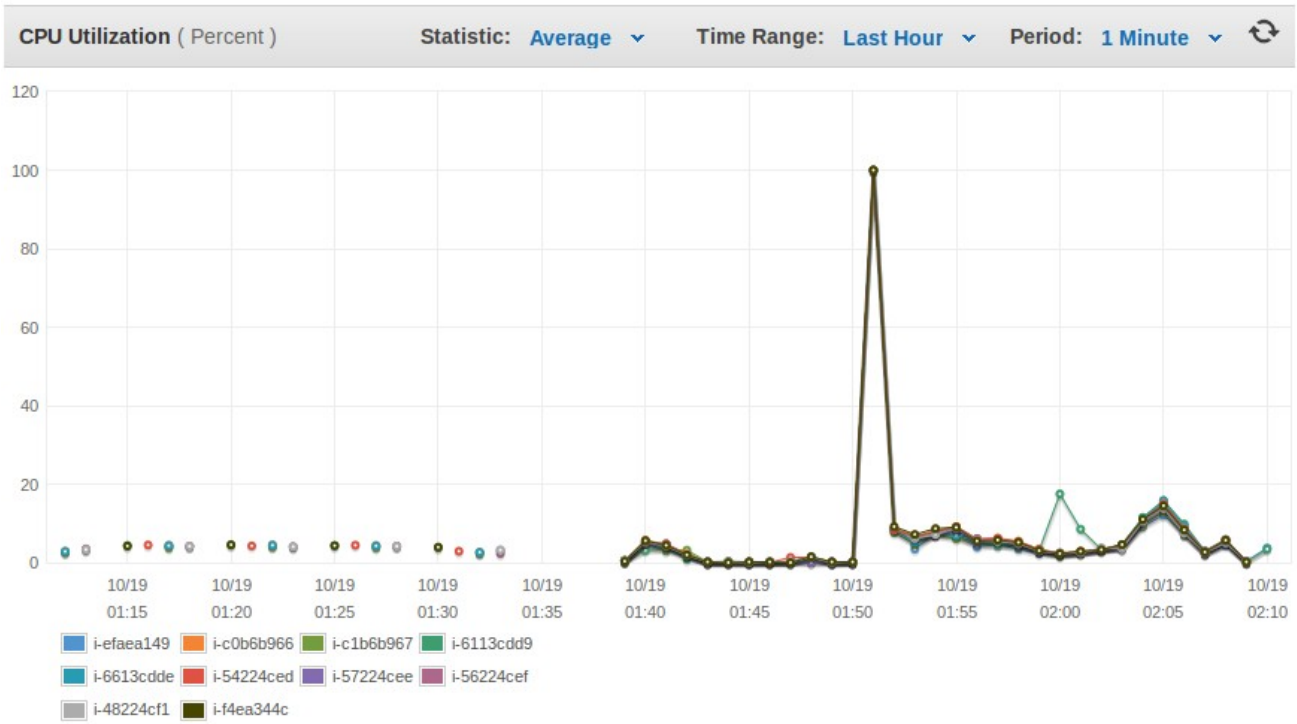
When you have 10 Spawners and you set desired attackers to 1000, every spawner will start 100 attacker agents. By klicking on the BANANA! button messagers are sent to the spawners. They will tell the attackers to attack the victim.

In the picture on the right side you can see the platform running on the controler server. As you can see different containers of the spawners are available. Each spawner container contains the spawner agent and also zero or more minion/attacker agents.



Measuring data

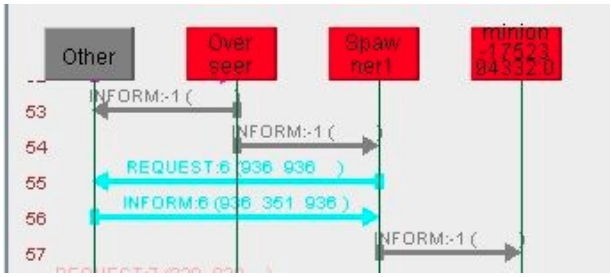
We used amazon cloud watch which allowed us to see detailed data about resource utilization of the servers. The following graphic shows the CPU Utilizations of the Spawner servers when they have to create several thousan agents.



Agent communication

The following diagram shows the communication when the interval of a minion gets updated. The Overseeer/coordinator sends a inform message to the spawner. The Spawner forwards the message to the minion.

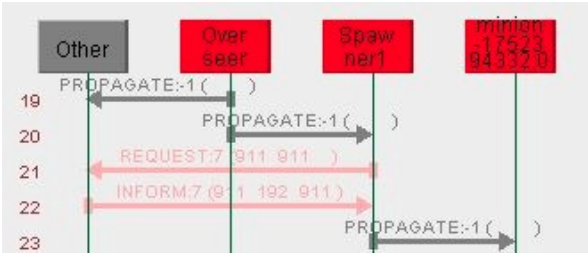
Message: Performative: INFORM, Content: period/interval in milliseconds



Agents

The following diagram shows the communication when the overseeer sends the attack command to the spawner which will forward it to the minion.

Message: Performative: PROPAGATE, Content: dns and port, separated by a colon

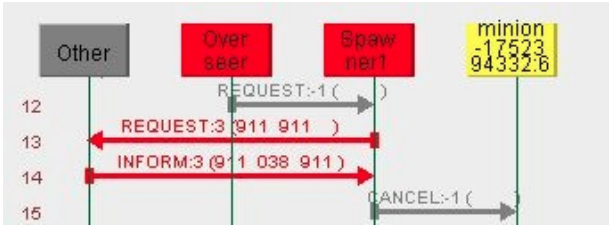


The following diagram shows the communication when the overseeer sends the amount of desired agents to the spawner.

Message: Performative: REQUEST, Content: desired number of agents

The following diagram shows the communication when the overseeer sends the amount of desired agents to the spawner. It has to decrease the amount of minions, so one minion gets cancelled.

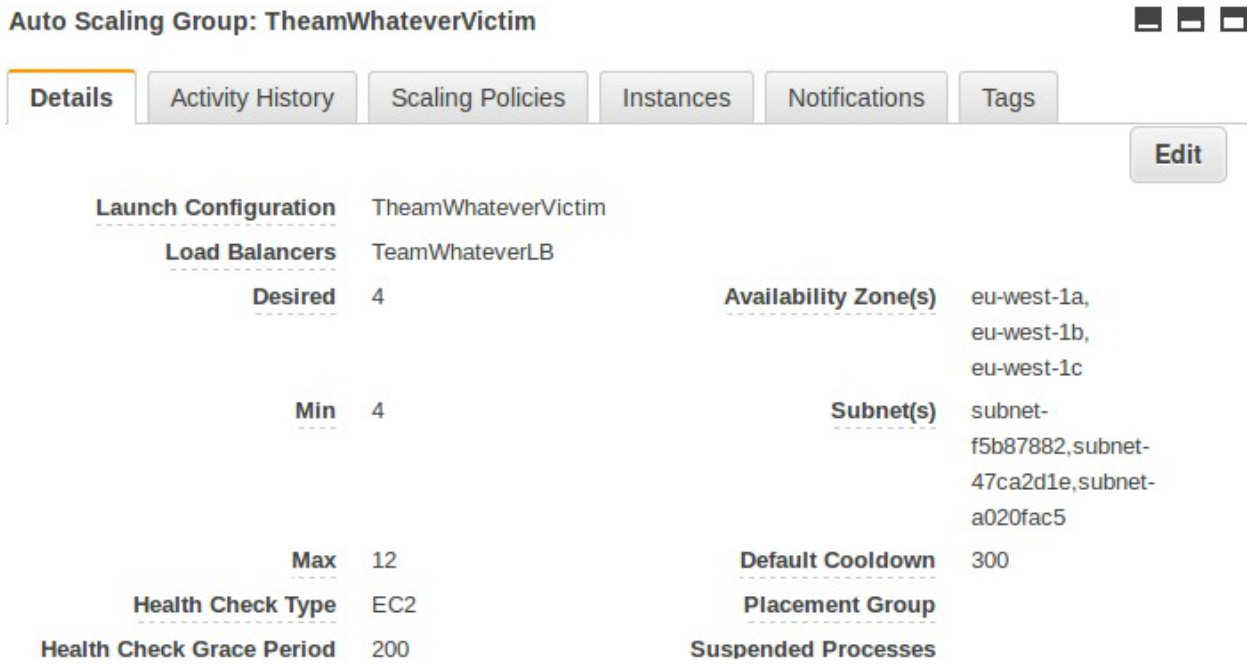
Message: Performative: CANCEL, Content: empty



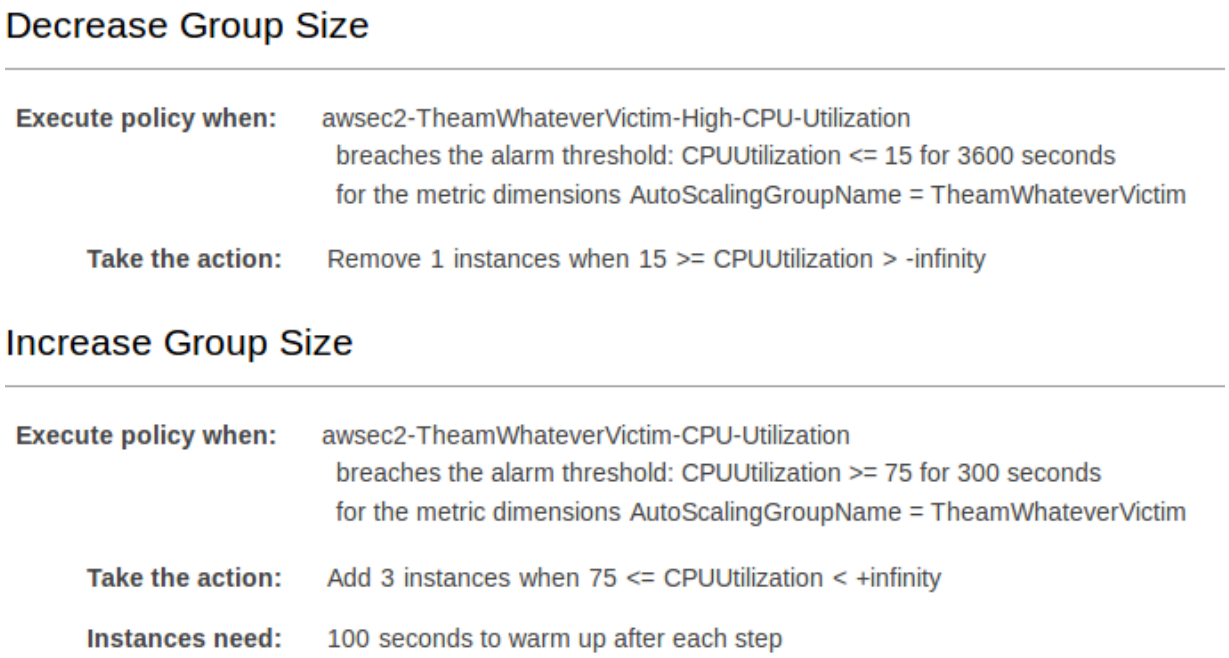
We used the architecture described berfore because it is very dynamic and allows us to scale the amount of attackers and the amount of victims very easily without much effort.

2. Victim

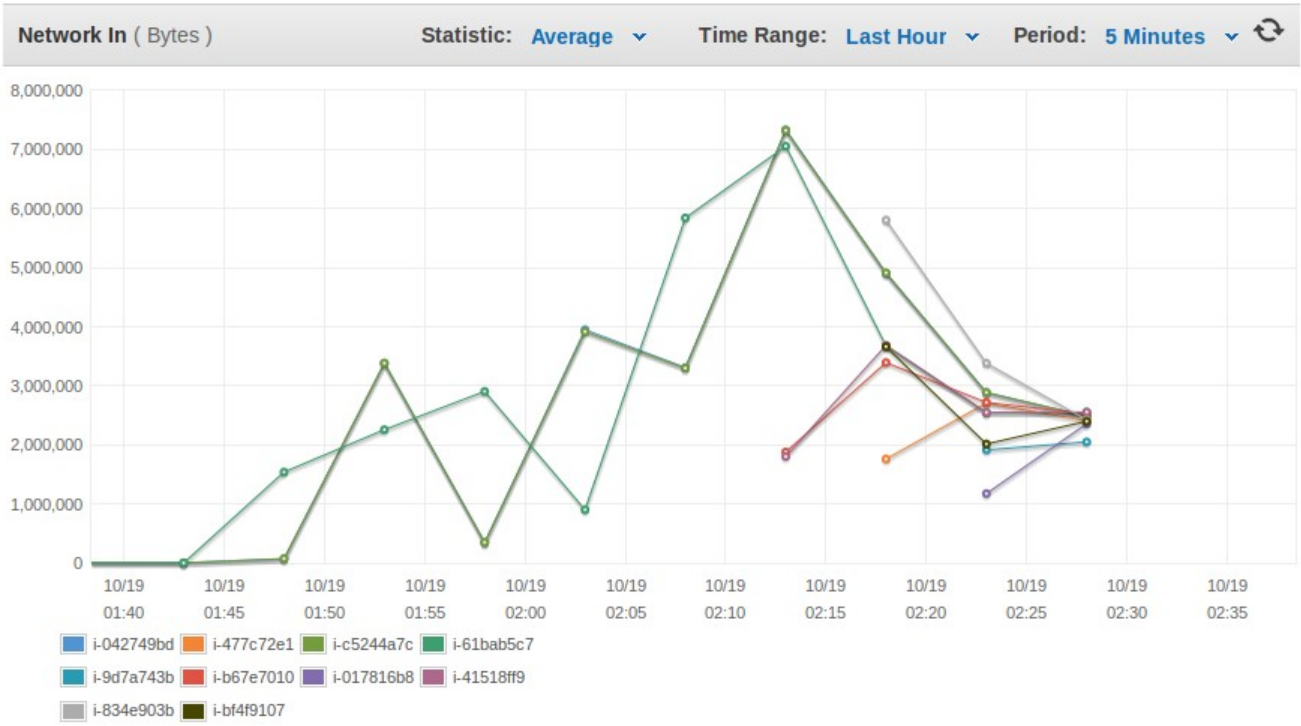
As already explained the victims stand behind a load balancer and are part of a auto scaling group. As you can see 4 victims are running from the beginning. Max is 12 in total.



The scaling policy of the victims:



As you can see in the following two diagrams an attack is happening and the increase policy triggers.



3. The war of worlds → Attacking the victims

4000 agents with 1 attack per second each were enough to utilize all ressources of 4 victims. As a reaction the auto scaling group added more victims. The victims start a calculation with a duration of one second after every connection.

Install JADE

Download all failes from website and add jade to your Java CLASSPATH variable.

The file ~/.bashrc

```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

# User specific aliases and functions
export CLASSPATH=~/jade/lib/commons-codec/commons-codec-1.3.jar:$CLASSPATH
export CLASSPATH=~/jade/lib/jadeExamples.jar:$CLASSPATH
export CLASSPATH=~/jade/lib/jade.jar:$CLASSPATH
```

Let the os execute a script on boot time:

Create the file **/etc/init.d/myserverBootScript.sh**

chmod +x /etc/init.d/myserverBootScript.sh

```
#!/bin/sh
#
### BEGIN INIT INFO
# Provides: myJAVAServer
# Required-Start: $remote_fs $syslog
# Required-Stop: $remote_fs $syslog
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: Initialize MyJavaServer for amazonaws
# Description: starts myjavaserver
### END INIT INFO
#Put the script to start with the system (using SysV). Just run the following command (as root):
#update-rc.d myscript defaults
cd /home/ec2-user/

case "$1" in
        start)
                echo executing bootScript
```

```
                su - ec2-user -c "/bin/bash ./bootScript.sh"
                ;;
        stop)
                echo Stopping not implemented
                ;;
        restart)
                echo Restarting not implemented
                ;;
        *)
                echo Usage: \$0 "{start|stop}"
                exit 1
esac
exit 0
```

Now you need to tell the amazon ami/centos to execute the script when it boots. update-rc.d **myserverBootScript** defaults.

Now you need to create the bootScript inside your home directory. The script can clone/pull the github project with git and call a script inside the pulled directory.