Vanet Simulator

Report for the Computer Security exam at the Politecnico di Torino

Walter Dal Mut (161600)
Armand Sofack (124515)
tutor: Giorgio Calandriello

June 2009

Contents

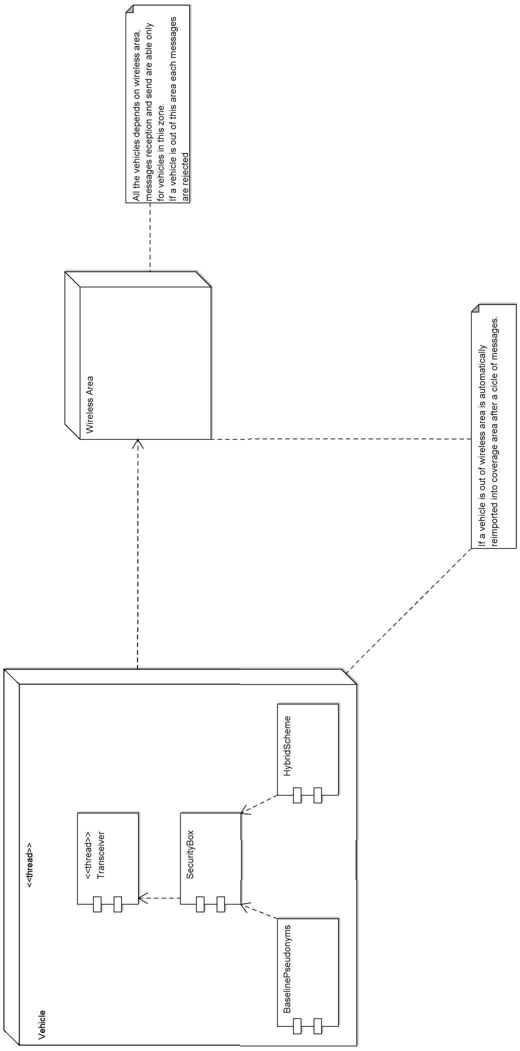
1 Introduction

2 UML Diagrams

2.1 Framework implementation

The Vanet Simulator is based on stack representation (see figure 1 at page 5), in particoular is composed by a vehicles which a transceiver for send and receive messages on the network and under that leve a security box which use the security implementation which you have set during simulation.

2.2 Class Diagram



All the vehicles depends on wireless area,
messages reception and send are able only
for vehicles in this zone.
If a vehicle is out of this area each messages
are rejected
Wireless Area
If a vehicle is out of wireless area is automatically
reimported into coverage area after a cicle of messages.
Vehicle
<<thread>>
<<thread>>
Transceiver
SecurityBox
HybridScheme
BaselinePseudonyms

Figure 1: Vanet Stack

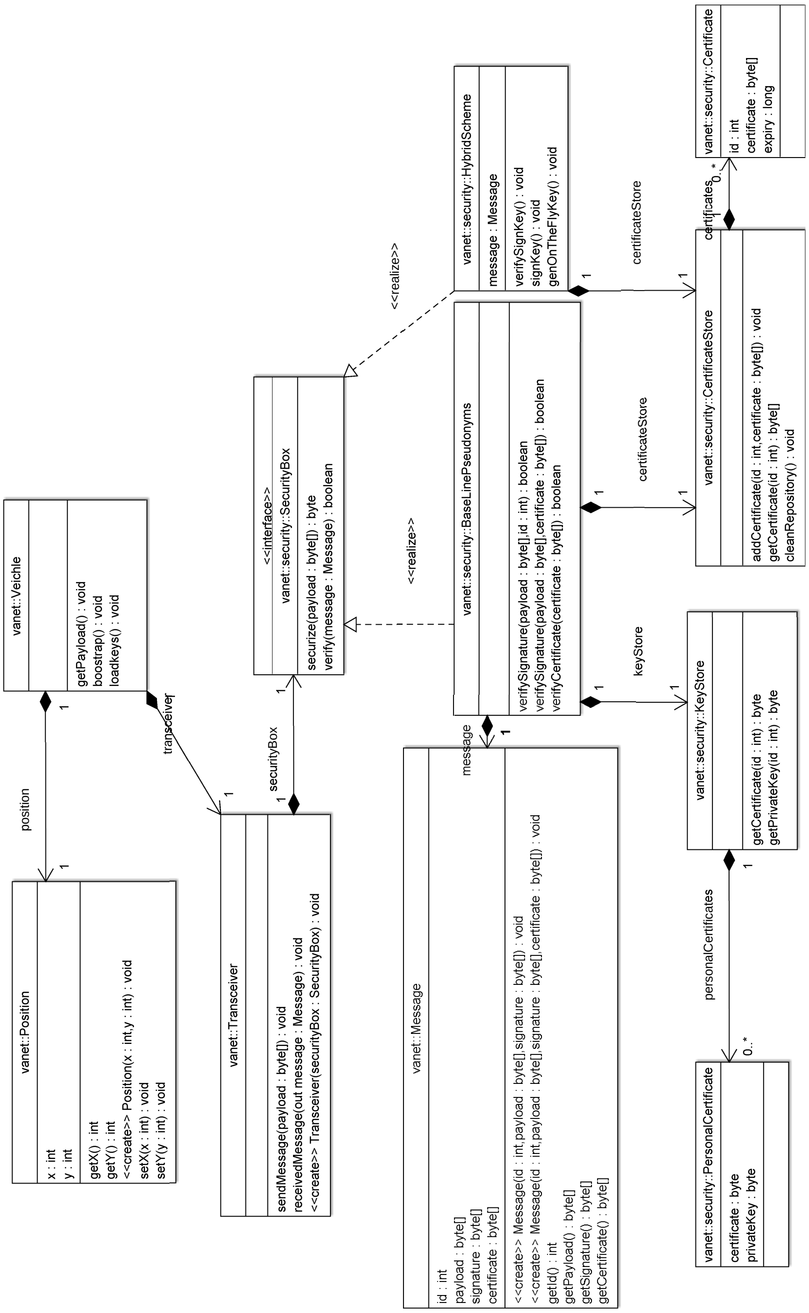

vanet::Veichle
getPayload() : void
boostrap() : void
loadkeys() : void
position
1
1
vanet::Position
x : int
y : int
getX() : int
getY() : int
<<create>> Position(x : int,y : int) : void
setX(x : int) : void
setY(y : int) : void
transceiver
1
vanet::Transceiver
sendMessage(payload : byte[]) : void
receivedMessage(out message : Message) : void
<<create>> Transceiver(securityBox : SecurityBox) : void
securityBox
1
1
<<interface>>
vanet::security::SecurityBox
securize(payload : byte[]) : byte
verify(message : Message) : boolean
<<realize>>
<<realize>>
vanet::security::HybridScheme
message : Message
verifySignKey() : void
signKey() : void
genOnTheFlyKey() : void
vanet::security::BaseLinePseudonyms
verifySignature(payload : byte[],id : int) : boolean
verifySignature(payload : byte[],certificate : byte[]) : boolean
verifyCertificate(certificate : byte[]) : boolean
message
1
vanet::Message
id : int
payload : byte[]
signature : byte[]
certificate : byte[]
<<create>> Message(id : int,payload : byte[],signature : byte[]) : void
<<create>> Message(id : int,payload : byte[],signature : byte[],certificate : byte[]) : void
getId() : int
getPayload() : byte[]
getSignature() : byte[]
getCertificate() : byte[]
certificateStore
1
1
certificateStore
1
1
vanet::security::CertificateStore
addCertificate(id : int,certificate : byte[]) : void
getCertificate(id : int) : byte[]
cleanRepository() : void
certificates
1
0..*
vanet::security::Certificate
id : int
certificate : byte[]
expiry : long
keyStore
1
1
vanet::security::KeyStore
getCertificate(id : int) : byte
getPrivateKey(id : int) : byte
personalCertificates
1
0..*
vanet::security::PersonalCertificate
certificate : byte
privateKey : byte

Figure 2: Class Diagram

3 Security Implementations

3.1 Baseline Pseudonyms

In the Baseline Pseudonyms implementation each vehicle V is equipped with a set of pseudonyms, a rich set of certificate certified by the certification authority without any information for identify the vehicle.
For the i-th pseudonym the CA provide a valid certificate with the public key signed by the CA and a private key for signing messages.
The easy way for realize this security implementation is to attach after every message a digital signature and the certificate used for signing message. In this way a vehicle which receive the message can verify the validity of certificate and so verify the digital signature of message. It's really important that pseudonyms is used only for a few time τ for provide security against the track message reception, because if I use the same certificate for a long period of time an attacker can follow the certitificate and identify the car which send messages. This implementation it's really difficult to implement because the message is too long because if we sum the message length plus the signature plus certificate with public key we have a big packet to send on network. For resolve the overhead problem we can use two optimization[1].

3.2 Hybrid Scheme

4 Simulation Framework implementation

4.1 Simulation of Baseline Pseudonyms

For starting simulations in baseline psedonyms you have to modify the configuration file *base.properties* [1] under folder *properties* in root directory of Vanet Simulator.

[1] See the user manual for more information around configurations. Section 7.1.1 at page 12

5 Test and profile of simulations

6 Comparisons and conclusions

7 Documentation

7.1 User Manual

In this part of this monography we explain the possibilities offred by configurations for using the Vanet Simulator in all of it parts. In particoular the main features offered by simulator are changing the security implementation passing by Baseline Pseudonyms and Hybrid Scheme implementations and configure vehicles on the road, the number of beacons sent during the simulations and modifing logging system for understanding results of simulation.
The Vanet Simulator is completly configurable modifing itś configurations files under the folder *properties* in the root directoty of simulator.

7.1.1 Base Configurations

The base configurations provide modifications in the core of Vanet Simulator, in particoular you can change the *base.properties* file for change core properties like beacons sent, security implementations and others base properties. If you open the configuration file you see:

```
#Max speed in km/h
max_speed = 140
#Max 802.11 cover in meters
wifi_cover = 200
#Access Point broadcast point
server_broadcast_point = 127.255.255.255
#Server Port
server_port = 55055
#Beacons/sec
beacons_sec = 0.1
#If you want no moves of veichles
no_moves = true
#choose simulator BP or GS
simulator = bp
#max certificate validity into area in seconds
maxCertificateValidityTime = 33
#Reattach certificate every tot beacons
reattachCertificate = 5
#MYSQL properties
mysql_host=127.0.0.1
mysql_username=root
mysql_password=
mysql_database=vanet
#Define the log system
#  0 MYSQL log
#  1 File log
#  2 StdOut log
logSystem=0
```

For detailed information you can see table 1 at page 13.

7.1.2 Vehicle Configurations

The vehicles configurations set the status of roads into the simulator. In particoular you can modify the number of vehicles into the road, velocity and initial position.
The configuration file for vehicle is XML (eXtensible Markup Language) based and is named *vehicles.xml* and positioned into folder *vehicles*; if you open this file you see:

```
<?xml version="1.0" encoding="UTF-8"?>
<Vehicles>
    <Vehicle id="1" speed="100" x="10" y="20" />
    <Vehicle id="2" speed="120" x="20" y="50" />
</Vehicles>
```

Table 1: Base Configuration specifications

Property Name	Property Translation	Property Type
max_speed	The maximum speed of vehicles	int
wifi_cover	The maximum wireless area coverage	int
server_broadcast_point	The broadcast node for send messages	string
server_port	The port for receive messages	int
beacons_sec	The number of beacons sent in one second	float
no_moves	Lock vehicles into the map	boolean
simulator	The simulator which you want use. BP for baseline implementations. GS for group signature implementation	string
maxCertificateValidityTime	The maximum time for certificate validity	int
reattachCertificate	Reattach the certificate every tot beacons	int
mysql_host	The host for mysql	string
mysql_username	Username for authenticate into mysql	string
mysql_password	Password for authenticate into mysql	string
mysql_database	Database to use	string
logSystem	The log system which you want use. 0 for mysql log system. 1 for log data into a files 2 for log data on console	int

Every tag, excluding root tag, identify a new vehicle with attributes like options, in paticoular you can modify the vehicle identification number changing the *id* attribute or you can change the veichle speed modifing the *speed* attribute or the position of the veichle using the x or y attributes. For the Baseline Pseudonyms operating mode you have to link certificates and private keys to each vehicles, for doing that you have to follow instruction in section 7.1.8 at page 15

7.1.3 Why many log configurations

The log system for this application is really difficult, infact the normal stdout or file log system is too slow and produce conflicts if you send a lot of beacons during sign and verify operation but it's really useful because you can understand immediatly what the system are doing in real time, the other methods are a middle solution for see result and velocity during sign and verify and the best solution for velocity but difficult to understand in real time the system but it's useful for post-processing. For this reason we have written three type of log system for use the best method when that are compatible with the simulation.

7.1.4 Log configuration

The log system use the *log4j* module for write sensible information of simulator. The system provide three log configurations, on standard output stream, file stream or on MySQL datbase. The configuration of log system it's really powerful and you can set the level of logging or change the log representation for standard out stream or file stream. The configuration of log system is divided into three file, ones for each method and it's collected into folder *properties* which names *stdout.properties* for standard out, *file.properties* for file stream or *mysql.properties* for MySQL database log system.

7.1.5 MySQL database configuration

For using MySQL database log system you have to configure the database before launching the Vanet Simulator. You have to create or import a database with tables definition into MySQL using the *vanet.sql* file under the *properties* directoty.
For import the database and tables definition you have to enter in you MySQL command line and create a new database using command:

```
mysql> CREATE DATABASE vanet;
```

After this step you have to create a table in the new database using commands:

```
mysql> use vanet;
...
mysql>CREATE TABLE IF NOT EXISTS ‘logs‘ (
  ‘log_id‘ int(11) unsigned zerofill NOT NULL auto_increment,
  ‘level‘ varchar(255) NOT NULL,
  ‘class_name‘ varchar(255) NOT NULL,
  ‘method_name‘ varchar(255) NOT NULL,
  ‘message‘ text NOT NULL,
  PRIMARY KEY  (‘log_id‘),
  KEY ‘level‘ (‘level‘)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;
...
```

After this step you have configured the MySQL database for record logs from Vanet Simulator.

7.1.6 Install Vanet Simulator on Windows

For install Vanet simulator on Windows operating system you can use the installer *setupVanetSimulator.exe* and follow the sceen information for complete the setup of application.
After install procedure you have to open a new console and go into install directory and send the command

```
C:\Program File\Vanet Simulator\>java -jar vanetSimulator.jar
```

After this command you see the boostrap procedure and after the system run. The default log operation is the standard out and you can see directly all the informations.
The common output on screen are this:

```
.:: Boostrap ::.
Loading base properties
Base properties loaded
Loading veichles configuration
security/certificates/1/c
security/certificates/2/c
Veichles configuration loaded
.:: Boostrap end ::.
```

7.1.7 Install Vanet Simulator on generic OS

For install Vanet Simulator you have to setup all folders and executable jar manually. Create new folder in a point of file system and enter in it, after that copy the content of *build* directory and now you can send the command for start the simulator.

```
name@domain$ java -jar vanetSimulator.jar
```

After this command you see the boostrap procedure and after the system run. The default log operation is the standard out and you can see directly all the informations.
The common output on screen are this:

```
.:: Boostrap ::.
Loading base properties
Base properties loaded
Loading veichles configuration
security/certificates/1/c
security/certificates/2/c
Veichles configuration loaded
.:: Boostrap end ::.
```

7.1.8 Add certificates and private keys for Baseline Pseudonyms

When the system doing the boostrap in Baseline Pseudonyms mode research certificate and private keys for doing digital signatures.
Security properties are in *security* folder and for Baseline Pseudonyms implementation only you have another folder active during the simulation named *certificates* and under that folder you have another one folder for each vehicle named with the *vehicle id* (section 7.1.2 at page 12). Under this folder you have to create two folders named *c* for certificates and *p* for private keys; after this steps you have to copy private keys in folder *p* and certificates in folder *c*.

References

[1] P. Papadimitratos, G. Calandriello, J.-P. Hubaux, A. Lioy, “Efficient and Robust Pseudonymous Authentication in VANET”, MOVE 2008: IEEE INFOCOM-2008 workshop on Mobile Networking for Vehicular Environments, Phoenix (AZ, USA), April 13-18, 2008, pp. 19-27

[2] P. Papadimitratos, M. Raya, J.-P. Hubaux, “Securing Vehicular Communications”, MOVE 2006: IEEE Wireless Communications Intervehicular Communications, October, 2006, pp. 8-15

[3] F. Kargl, E. Schoch, B. Wiedersheim, T. Leinmuller, “Secure and Efficient Beaconing for Vehicular Networks”, VANET 2008: IEEE INFOCOM-2008 workshop on Mobile Networking for Vehicular Environments, San Francisco (CA, USA), September 15, 2008, pp. 1-2