TECHNICAL REPORT

Report No. UI-SE-MDSERG-2018-10
Date: July 23, 2018

ATL rules and OCL queries implemented in VAnDroid

Atefeh Nirumand
Bahman Zamani
Behrouz Tork Ladani

Department of Software Engineering
University of Isfahan
Hezar-Jerib Ave.
Isfahan

Tel: +98-31-37934537
Fax: +98-31-36699529 , +98-31-37932670
zamani@eng.ui.ac.ir
http://mdse.ui.ac.ir/TR

ATL rules and OCL queries implemented in VAnDroid

Atefeh Nirumand, Bahman Zamani, and Behrouz Tork Ladani
Department of Software Engineering
University of Isfahan
Isfahan, Iran.
{atefehnirumand, zamani, ladani}@eng.ui.ac.ir

Abstract: *This report provides the implementation details of the VAnDroid. VAnDroid is a framework based on Model Driven Reverse Engineering (MDRE) to analyze Android apps. The approach conducted in this framework has three phases. The feasibility and pertinence of this approach is demonstrated by developing an Eclipse-based tool called VAnDroid which implements the three phases of this approach. This tool has been developed by Nirumand et al. VAnDroid automatically identifies the Intent Spoofing and Unauthorized Intent Receipt as two attacks related to the Android application communication model.*

ATL rules and OCL queries implemented in VAnDroid

Atefeh Nirumand, Bahman Zamani, and Behrouz Tork Ladani
Department of Software Engineering
University of Isfahan
Isfahan, Iran.
{atefehnirumand, zamani, ladani}@eng.ui.ac.ir

Abstract: *This report provides the implementation details of the VAnDroid. VAnDroid is a framework based on Model Driven Reverse Engineering (MDRE) to analyze Android apps. The approach conducted in this framework has three phases. The feasibility and pertinence of this approach is demonstrated by developing an Eclipse-based tool called VAnDroid which implements the three phases of this approach. This tool has been developed by Nirumand et al. VAnDroid automatically identifies the Intent Spoofing and Unauthorized Intent Receipt as two attacks related to the Android application communication model.*

Contents

1 Introduction **2**

2 The ATL rules used in the Transformations and Integration phase **2**

3 The ATL rules and OCL queries used in the Analysis phase **11**

List of Figures

1 Android Application Security Aspects Metamodel. 3

1 Introduction

VAnDroid is a framework based on Model Driven Reverse Engineering (MDRE). In fact, the MDRE process is used for this framework to analyze Android apps and detect vulnerabilities in the Inter-Component Communication (ICC). The approach conducted in this framework has three phases. The security information is first extracted from the Android application. This is then transformed into a single model called Android Applications Security Aspects using model-to-model transformation of the ATL. Based on this model, some operations, such as queries and model manipulations, can be applied later to analyze and manage the security configurations. In order to detect vulnerabilities, the ATL and OCL rules have been used. In this report, the details of these ATL rules and OCL queries are described.

2 The ATL rules used in the Transformations and Integration phase

In order to extract the needed information and integrate them in the single model, the proposed metamodel is used. This metamodel is shown in Figure 1. In the following, the ATL transformation rules used for the transformations and integration phase are shown.

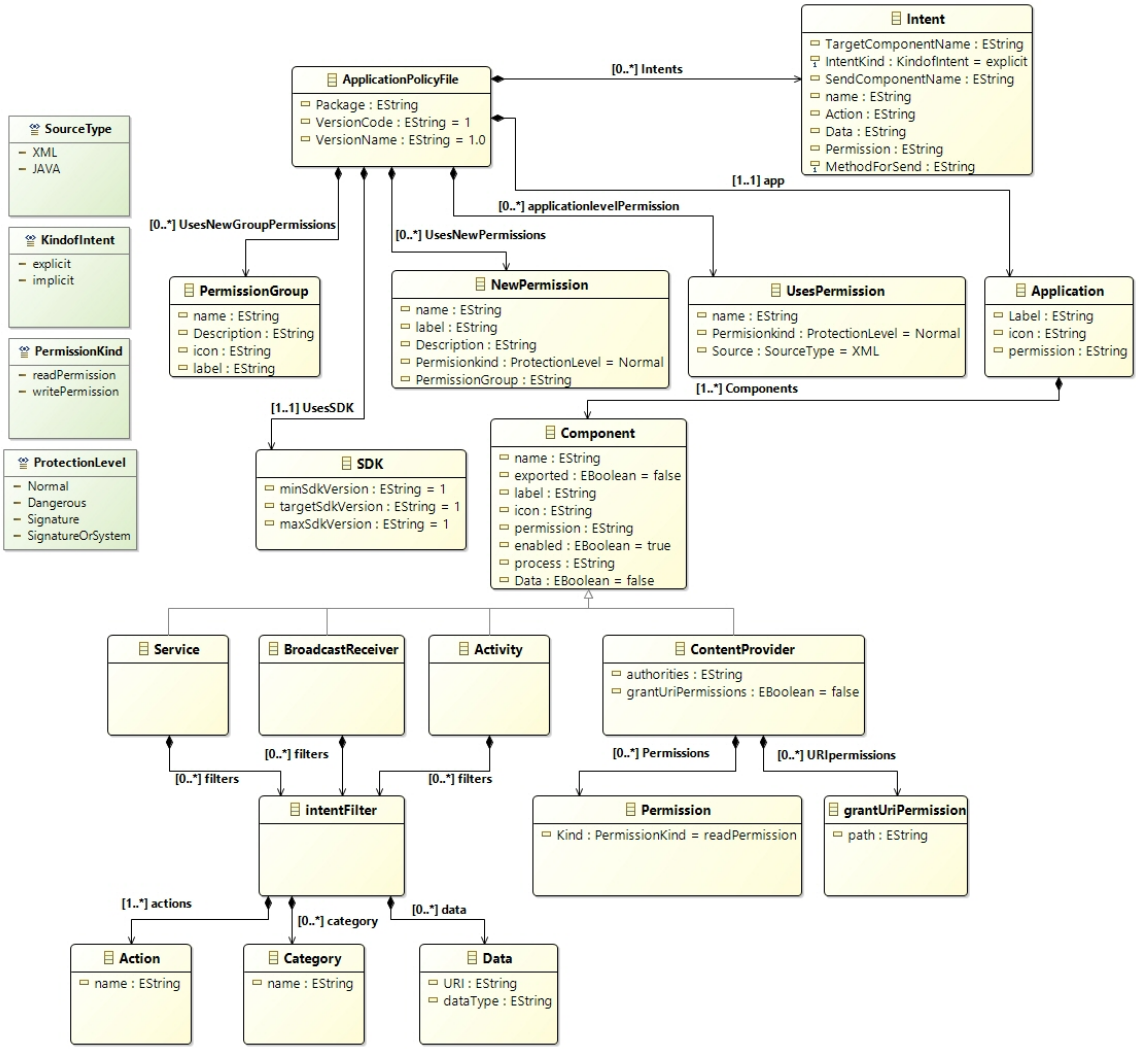

Figure 1: Android Application Security Aspects Metamodel.

```
1  rule Element2SDK{
2  from
3  s:XML!Element (s.name='uses-sdk')
4  to
5  t:ASEC!SDK
6  (
7  targetSdkVersion <-
8   if s.getAttribute('android:targetSdkVersion').oclIsUndefined
       () then
9  OclUndefined
10 else
11 s.getAttribute('android:targetSdkVersion').value
12 endif,
13
14 minSdkVersion <- if
15  s.getAttribute('android:minSdkVersion').oclIsUndefined()
       then
16 OclUndefined
17 else
18 s.getAttribute('android:minSdkVersion').value
19 endif,
20
21 maxSdkVersion <- if
22  s.getAttribute('android:maxSdkVersion').oclIsUndefined()
       then
23 OclUndefined
24 else
25 s.getAttribute('android:maxSdkVersion').value
26 endif
27 )
28 }
```

Listing 1: ATL rule to create SDK element.

```
1  rule Element2UsesPermission{
2  from
3  s:XML!Element   (s.name='uses-permission')
```

```
4   to
5   t:ASEC!UsesPermission
6   (
7   name <- s.getAttribute('android:name').value,
8   Permisionkind <-
9    thisModule.stringOfPermission2ProtectionLevel.get(
10                  s.getAttribute('android:name').value)
11  )
12  }
```

Listing 2: ATL rule to create UsesPermission element.

```
1   rule Element2NewPermission {
2   from
3   s:XML!Element (s.name='permission')
4   to
5   t:ASEC!NewPermission
6   (
7   name <- s.getAttribute('android:name').value,
8
9   label <-
10   if s.getAttribute('android:label').oclIsUndefined() then
11   OclUndefined
12   else
13   s.getAttribute('android:label').value
14   endif,
15
16  Description <-
17   if s.getAttribute('android:description').oclIsUndefined()
       then
18   OclUndefined
19   else
20   s.getAttribute('android:description').value
21   endif,
22
23  Permisionkind <-
```

```
24    if s.getAttribute('android:protectionLevel').oclIsUndefined
        () then
25    OclUndefined
26    else
27    thisModule.string2ProtectionLevel.get(
28        s.getAttribute('android:protectionLevel').value)
29    endif,
30
31   PermissionGroup <-
32    if s.getAttribute('android:permissionGroup').oclIsUndefined
        () then
33    OclUndefined
34    else
35    s.getAttribute('android:permissionGroup').value
36    endif
37   )
38   }
```

Listing 3: ATL rule to create NewPermission element.

```
1    rule Element2PermissionGroup{
2    from
3    s:XML!Element (s.name='permission-group')
4    to
5    t:ASEC!PermissionGroup
6    (
7    name <- s.getAttribute('android:name').value,
8
9    Description <-
10   if s.getAttribute(
11    'android:description').oclIsUndefined() then
12   OclUndefined
13   else
14   s.getAttribute('android:description').value
15   endif,
16
17   icon <- if s.getAttribute('android:icon').oclIsUndefined()
       then
```

```
18  OclUndefined
19  else
20  s.getAttribute('android:icon').value
21  endif,
22
23  label <- if s.getAttribute('android:label').oclIsUndefined()
        then
24  OclUndefined
25  else
26  s.getAttribute('android:label').value
27  endif
28  )
29  }
```

Listing 4: ATL rule to create PermissionGroup element.

```
1   rule Element2Application{
2   from
3   s:XML!Element   (s.name='application')
4   to
5   t:ASEC!Application
6   (
7   Label <- s.getAttribute('android:label').value,
8
9   icon <- if s.getAttribute('android:icon').oclIsUndefined()
        then
10  OclUndefined
11  else
12  s.getAttribute('android:icon').value
13  endif,
14
15  permission <-
16      if s.getAttribute('android:permission').oclIsUndefined()
        then
17      OclUndefined
18      else
19      s.getAttribute('android:permission').value
20  endif,
```

```
21
22  Components <- s.children->
23          select(c | thisModule.name_of_components.includes(c.
        name))
24  )
25  }
```

Listing 5: ATL rule to create Application element.

```
1   rule Element2Component {
2   from
3   s:XML!Element (thisModule.name_of_components.includes(s.name)
        )
4   to
5   t:ASEC!Component
6   (
7   name <- if s.getAttribute('android:name').oclIsUndefined()
        then
8   OclUndefined
9   else
10  s.getAttribute('android:name').value
11  endif,
12
13  label <- if s.getAttribute('android:label').oclIsUndefined()
        then
14  OclUndefined
15  else
16  s.getAttribute('android:label').value
17  endif,
18
19  icon <- if s.getAttribute('android:icon').oclIsUndefined()
        then
20  OclUndefined
21  else
22  s.getAttribute('android:icon').value
23  endif,
24
```

```
25  permission <-
26  if s.getAttribute('android:permission').oclIsUndefined() then
27  OclUndefined
28  else
29  s.getAttribute('android:permission').value
30  endif,
31
32  enabled <-
33  if s.getAttribute('android:enabled').oclIsUndefined() then
34  false
35  else
36  thisModule.string2Boolean.get(
37   s.getAttribute('android:enabled').value)
38  endif,
39
40  process <-
41  if s.getAttribute('android:process').oclIsUndefined() then
42  OclUndefined
43  else
44  s.getAttribute('android:process').value
45  endif,
46
47  exported <-
48  if s.getAttribute('android:exported').oclIsUndefined() then
49  if not s.getIntentFilter.notEmpty() then
50  false
51  else
52  true
53  endif
54  else
55  thisModule.string2Boolean.get(
56    s.getAttribute('android:exported').value)
57  endif,
58
59  Data <-
60  if thisModule.getAllgetIntentMethod(
61   s.getAttribute('android:name').value+'.java').notEmpty()
       then
62  true
63  else
```

```
64  false
65  endif
66  )
67  }
```

Listing 6: ATL rule to create Component element.

```
1   rule Element2intentFilter {
2   from
3   s:XML!Element (s.name='intent-filter')
4   to
5   t:ASEC!intentFilter
6   (
7   actions <- s.children ->select(c | c.name='action'),
8   category <- s.children ->select(c | c.name='category'),
9   data <- s.children->select(c | c.name='data')
10  )
11  }
```

Listing 7: ATL rule to create intentFilter element.

3 The ATL rules and OCL queries used in the Analysis phase

All the security information of the application is gathered and represented in the form of an integrated model corresponding to our Android Applications Security Aspects metamodel. the security information obtained in the form of the model is received as input to be evaluated. In order to detect vulnerabilities, the ATL and OCL rules have been used. In the following, some of the ATL transformation rules and OCL queries used for the analysis phase are shown.

```
1   rule Component2IntentSpoofing
2   {
3   from
4   s:ASEC!Component(s.isPublic and (s.permission.oclIsUndefined
        ()
5    or not thisModule.StrongPermission.includes(s.permission) ))
6   to
7   t:ANO!IntentSpoofingComponent
8   (
9   description <- s.getIntentSpoofing
10   ->iterate(up; output: String = '' | output->concat(up)+'\n')
11   )
12   do {
13   t.trace<-Sequence{s};
14       }
15  }
```

Listing 8: ATL rule to check Intent Spoofing vulnerability.

```
1   helper context ASEC!Component
2          def:getIntentSpoofing :Sequence(String) =
```

```
3   let ComponentsOfApp : Sequence(ASEC!Component) =
4   self.refImmediateComposite().Components in
5   ComponentsOfApp->
6    iterate(n ; output : Sequence(String) = Sequence{} |
7   if self.haveIntentSpoofing then
8   if self.Data=true then
9   if self.oclIsTypeOf(ASEC!Activity) then
10  output->including(
11  ': The Application have
12   Activity Launch(with data) in component with name:  '+
13  self.name+
14  ' '+''+'... /the reasons of this vulnerability are :
15  this component is public
16  and component does not have Strong Permission')
17  else
18  if self.oclIsTypeOf(ASEC!Service) then
19  output->including(
20  ': The Application
21  have Service Launch(with data)  in component with name:  '+
22  self.name+
23  ' '+''+'... /the reasons of this vulnerability are :
24  this component is public
25  and component does not have Strong Permission')
26  else
27  if self.oclIsTypeOf(ASEC!BroadcastReceiver) then
28  output->including(
29  ': The Application
30  have Broadcast Injection(with data)  in component with name:
        '+
31  self.name+
32  ' '+''+'... /the reasons of this vulnerability are :
33  this component is public
34  and component does not have Strong Permission')
35  else
36  output->reject(c | c.name=self.name)
37  endif
38  endif
39  endif
40  else
41  if self.oclIsTypeOf(ASEC!Activity) then
```

```
42  output->including(
43  ': The Application
44  have Activity Launch(without data)  in component with name:
       '+
45  self.name+
46  ' '+''+'... /the reasons of this vulnerability are :
47  this component is public
48  and component does not have Strong Permission')
49  else
50  if self.oclIsTypeOf(ASEC!Service) then
51  output->including(
52  ': The Application
53  have Service Launch(without data)  in component with name:
       '+
54  self.name+
55  ' '+''+'... /the reasons of this vulnerability are :
56  this component  is public
57  and component does not have Strong Permission')
58  else
59  if self.oclIsTypeOf(ASEC!BroadcastReceiver) then
60  output->including(
61  ': The Application
62  have Broadcast Injection(without data)  in component with
       name:  '+
63  self.name+
64  ' '+''+'... /the reasons of this vulnerability are :
65  this component is public
66  and component does not have Strong Permission')
67  else
68  output->reject(c | c.name=self.name)
69  endif
70  endif
71  endif
72  endif
73  else
74  output->reject(c | c.name=self.name)
75  endif)->asSet()->asSequence();
```

Listing 9: Helper to check Intent Spoofing vulnerability.

```
1  rule Intent2UnauthorizedIntentReceipt
2  {
3  from
4  s:ASEC!Intent (not s.Action.oclIsUndefined() and s.IntentKind
       =#implicit )
5  to
6  t:ANO!UnauthorizedIntentReceipt
7  (
8  description <- s.getUnauthorizedIntentReceipt->
9      iterate(up; output: String = '' | output->concat(up)+'\n
       ')
10 )
11 do {
12 t.trace<-Sequence{s};
13 }
14 }
```

Listing 10: ATL rule to check Unauthorized Intent Receipt vulnerability.

```
1  helper context ASEC!Intent
2      def:getUnauthorizedIntentReceipt :Sequence(String) =
3  let IntentsOfApp : Sequence(ASEC!Intent) =
4   self.refImmediateComposite().Intents in
5  IntentsOfApp->iterate(n ; output : Sequence(String) =
       Sequence{} |
6  if self.IntentKind=#explicit then
7  output->reject(c | c.name=self.name)
8  else
9  if self.Action.oclIsUndefined() then
10 output->reject(c | c.name=self.name)
11 else
12 --permission is undefined
13 if self.Permission.oclIsUndefined() then
14 if self.Data='yes' then
15 if thisModule.Method_for_Activity.includes(self.MethodForSend
       ) then
16 output->including(
17 ': The Application
```

```
18  have Activity Hijacking(with data) attack in Intent with name
        :  '+
19  self.name+ ' ' +'and in Component with Package : '+
20  self.SendComponentName +
21  '... /the reasons of this vulnerability are:
22  the Intent does not have  Strong permission,
23  while it have Action and it is of implicit type./.../
24  Intent-sending mechanisms is use for Activity./     ')
25  else
26  if thisModule.Method_for_Service.includes(self.MethodForSend)
        then
27  output->including(
28  ': The Application
29  have Service Hijacking(with data) attack in Intent with name:
        '+
30  self.name +' ' +'and in Component with Package : '+
31  self.SendComponentName+
32  '... /the reasons of this vulnerability are:
33  the Intent does not have strong permission,
34  while it have Action and it is of implicit type./.../
35  Intent-sending mechanisms is use for Service./     ')
36  else
37  if thisModule.Method_for_Receiver.includes(self.MethodForSend
        ) then
38  output->including(
39  ': The Application
40  have Broadcast Theft(with data) attack in Intent with name:
        '+
41  self.name+ ' ' +'and in Component with Package : '+
42  self.SendComponentName+
43  '... /the reasons of this vulnerability are:
44  the Intent does not have strong permission,
45  while it have Action and it is of implicit type./.../
46  Intent-sending mechanisms is use for Receiver./     ')
47  else
48  output->reject(c | c.name=self.name)
49  endif
50  endif
51  endif
52  else
```

```
53  -- not Data
54  if thisModule.Method_for_Activity.includes(self.MethodForSend
        ) then
55  output->including(
56  ': The Application
57  have Activity Hijacking(without data) attack in Intent with
        name:  '+
58  self.name+ ' ' +'and in Component with Package : '+
59  self.SendComponentName+
60  '... /the reasons of this vulnerability are:
61  the Intent does not have strong permission,
62  while it have Action and it is of implicit type./.../
63  Intent-sending mechanisms is use for Activity./     ')
64  else
65  if thisModule.Method_for_Service.includes(self.MethodForSend)
        then
66  output->including(
67  ': The Application
68  have Service Hijacking(without data) attack in Intent with
        name:  '+
69  self.name + ' '+'and in Component with Package : '+
70  self.SendComponentName+
71  '... /the reasons of this vulnerability are:
72  the Intent does not have strong permission,
73  while it have Action and it is of implicit type./.../
74  Intent-sending mechanisms is use for Service./     ')
75  else
76  if thisModule.Method_for_Receiver.includes(self.MethodForSend
        ) then
77  output->including(
78  ': The Application
79  have Broadcast Theft(without data) attack in Intent with name
        :  '+
80  self.name +' '+'and in Component with Package : '+
81  self.SendComponentName+
82  '... /the reasons of this vulnerability are:
83  the Intent does not have strong permission,
84  while it have Action and it is of implicit type./.../
85  Intent-sending mechanisms is use for Receiver./     ')
86  else
```

```
87  output->reject(c | c.name=self.name)
88  endif
89  endif
90  endif
91  endif
92  else
93  --permission is strong
94  if thisModule.StrongPermission.includes(self.Permission) then
95  output->reject(c | c.name=self.name)
96  else
97  --permission is not strong
98  if self.Data='yes' then
99  if thisModule.Method_for_Activity.includes(self.MethodForSend
    ) then
100 output->including(
101 ': The Application
102 have Activity Hijacking(with data) attack in Intent with name
    : '+
103 self.name +' '+'and in Component with Package : '+
104 self.SendComponentName+
105 '... /the reasons of this vulnerability are:
106 the Intent have permission but permission is not Strong ,
107 while it have Action and it is of implicit type./.../
108 Intent-sending mechanisms is use for Activity./     ')
109 else
110 if thisModule.Method_for_Service.includes(self.MethodForSend)
     then
111 output->including(
112 ': The Application
113 have Service Hijacking(with data) attack in Intent with name:
     '+
114 self.name +' '+'and in Component with Package : '+
115 self.SendComponentName+
116 '... /the reasons of this vulnerability are:
117 the Intent have permission but permission is not Strong,
118 while it have Action and it is of implicit type./.../
119 Intent-sending mechanisms is use for Service./     ')
120 else
121 if thisModule.Method_for_Receiver.includes(self.MethodForSend
    ) then
```

```
122  output->including(
123  ': The Application
124  have Broadcast Theft(with data) attack in Intent with name:
        '+
125  self.name+' ' +'and in Component with Package : '+
126  self.SendComponentName+
127  '... /the reasons of this vulnerability are:
128  the Intent have permission but permission is not Strong,
129  while it have Action and it is of implicit type./.../
130  Intent-sending mechanisms is use for Receiver./      ')
131  else
132  output->reject(c | c.name=self.name)
133  endif
134  endif
135  endif
136  else
137  if thisModule.Method_for_Activity.includes(self.MethodForSend
        ) then
138  output->including(
139  ': The Application
140  have Activity Hijacking(without data) attack in Intent with
        name:  '+
141  self.name +' '+'and in Component with Package : '+
142  self.SendComponentName+
143  '... /the reasons of this vulnerability are:
144  the Intent have permission but permission is not Strong,
145  while it have Action and it is of implicit type./.../
146  Intent-sending mechanisms is use for Activity./      ')
147  else
148  if thisModule.Method_for_Service.includes(self.MethodForSend)
         then
149  output->including(
150  ': The Application
151  have Service Hijacking(without data) attack in Intent with
        name:  '+
152  self.name+' ' +'and in Component with Package : '+
153  self.SendComponentName+
154  '... /the reasons of this vulnerability are:
155  the Intent have permission but permission is not Strong,
156  while it have Action and it is of implicit type./.../
```

```
157   Intent-sending mechanisms is use for Service./      ')
158   else
159   if thisModule.Method_for_Receiver.includes(self.MethodForSend
          ) then
160   output->including(
161   ': The Application
162   have Broadcast Theft(without data) attack in Intent with name
          :  '+
163   self.name+' ' +'and in Component with Package : '+
164   self.SendComponentName+
165   '... /the reasons of this vulnerability are:
166   the Intent have permission but permission is not Strong,
167   while it have Action and it is of implicit type./.../
168   Intent-sending mechanisms is use for Receiver./      ')
169   else
170   OclUndefined
171   endif
172   endif
173   endif
174   endif
175   endif
176   endif
177   endif
178   endif
179   )->asSet()->asSequence();
```

Listing 11: Helper to check Unauthorized Intent Receipt vulnerability.