Declarative Aspect-Oriented Security Policies and Verification for In-lined Reference Monitors

Kevin W. Hamlen and Micah Jones
University of Texas at Dallas
Outline

1. Background
   – In-lined Reference Monitors (IRMs)
   – Aspect-Oriented Programming (AOP)

2. Contributions
   – SPoX (Security Policy XML)
   – Automated policy analysis
   – Service-oriented IRMs

3. Next paper: Automated IRM certification
Publications


Service-Oriented Security Problem

Policy: User is willing to divulge city but *not* street to server.
End Goal: Formally Certified IRMs

Trusted

Policy

Rewriter

Untrusted

Untrusted binary

Rewritten binary
End Goal: Formally Certified IRMs

- Trusted
  - Policy
  - Verifier
    - yes: accept
    - no: reject

- Untrusted
  - Untrusted binary
  - Rewriter
    - Rewritten binary
End Goal: Formally Certified IRMs

Trusted

Verifier

Policy

Untrusted

Rewriter

Untrusted binary

Rewritten binary

yes → accept

no → reject
Security Automata

An established way to describe security policies*

**Network Send Policy (N):** Untrusted applets may not send messages over the network after they have read from the street field.

* [Alpern & Schneider, J.Distrib. Comp. '86]
Security Automata

Resource Bound Policy (R): Do not create more than 3 new files on the user’s desktop.
Enforcement Approaches

1. Add runtime security checks to the OS or VM
   – OS/VM is large and difficult to modify
   – Hard to verify that customized OS/VM is secure

2. Statically verify that the binary code is secure
   – Undecidable in general
   – Especially difficult for temporal policies

3. Automatically inject security checks into the untrusted binary code
   – The approach used in In-lined Reference Monitors (IRMs)
In-lined Reference Monitors

Uninstrumented code:

```java
int s = 0;

if (Regex.matches(arg1, ".*\Users\.*\Desktop\.*")

File.new(arg1);
    if (s == 3)
        halt();
    if (s >= 0 && s <= 2)
        s := s + 1;
}
```

Instrumented code:
In-lining Process

Original code:
...
File.new(arg1);
...

Rewritten code:
...
if (Regex.matches(arg1, ".*\Users\.*\Desktop\.*")
{
  if (s == 3)
    halt();
  if (s >= 0 && s <= 2)
    s := s + 1;
}
File.new(arg1);
...

[Jones & Hamlen, ISI ‘09]
IRM Systems

• SASI [Erlingsson & Schneider, NSPW ‘99]
  – Early IRM for Java and x86 binaries

• PoET/PSLang [Erlingsson, PhD Thesis ‘04]
  – Successor to SASI, IRM for Java
  – Enforces more complex policies than SASI

• Java-MaC [Kim et al., FMSD ’04]
  – Uses language with limited ability to specify security-relevant events

• ConSpec [Aktug & Naliuka, SCP ‘08]
  – Uses limited, effect-free version of PSLang
  – Guard instructions are trusted
Aspect-Oriented Programming (AOP)

- Emerging form of software development
- Builds upon (does not replace) OOP
- Two components:
  - **Pointcuts** describe sets of instructions
  - **Advice** provides code to run before, after, or around instructions matched by pointcuts
- The combination of a pointcut and its corresponding advice is an **Aspect**
- In-lining a security automaton is really AOP
aspect FaultHandler {
    private boolean Server.disabled = false;
    private void reportFault() { System.out.println("Failure! Please fix!"); }
    public static void fixServer(Server s) { s.disabled = false; }
    pointcut services(Server s): call(public * *(..)) && target(s);
    before(Server s): services(s) {
        if (s.disabled) throw new DisabledException();
    }
    after(Server s) throwing (FaultException e): services(s) {
        s.disabled = true;
        reportFault();
    }
}
AspectJ Pointcut Language

**call(MethodPattern)** – matches call sites

**execution(MethodPattern)** – matches method entrypoints

**get(FieldPattern)** – matches field-reads

**set(FieldPattern)** – matches field-writes

$p_1 \&\& p_2$ – matches places where both $p_1$ and $p_2$ match

$p_1 \mid\mid p_2$ – matches places where $p_1$ or $p_2$ matches (or both)

$p$ – matches when $p$ doesn’t match

**within(TypePattern)** – matches all code within a class

**within(MethodPattern)** – matches all code within a method

**cflow(p)** – matches anytime there is a stack frame matching $p$
CFlow Examples

• `cflow(call(f)) && cflow(call(g)) && call(h)`
  – f and g are both on the call stack when h is called

• `cflow(call(f) && call(g)) && call(h)`
  – there is a stack frame that is both a call to f and a call to g
  – impossible! (unless f and g are the same function)

• `cflow(call(f) && !within(g)) && call(h)`
  – h is called while there is a call to f on the stack, and the call to f was not located within g
Aspect-Oriented IRMs

Enforcing 3-desktop-files policy using Java-MOP*:

```java
import java.util.regex.*;

SafeFile() {
    static int counter = 0;
    static final String REGEX = ".*\Users\\.*\Desktop\\.*";
    static final int MAX_FILES = 3;

    event createFile before() : call(File.new(filename)) {
        if (Pattern.matches(REGEX, filename)) {
            if (counter >= MAX_FILES) System.exit(1);
            else counter++;
        }
    }
}
```

*[Chen & Roșu, TACAS ‘05]*
Aspect-Oriented IRMs

An alternate version:

```java
import java.util.regex.*;

SafeFile() {
    static int counter = 0;
    static final String REGEX = ".*\Users\.*\Desktop\.*";
    static final int MAX_FILES = 3;
    event createFile before() : call(File.new(filename)) {
        if (Pattern.matches(REGEX, filename)) counter++;
    }
    event illegalCreateFile before() : call(*.new(filename)) {
        if (Pattern.matches(REGEX, filename) && counter >= MAX_FILES)
            System.exit(1);
    }
}
```
(policy
  (state name="s")

  (pointcut name="new_desktop_file"
    (and (call "File.new")
      (argval 1 (streq ".*\Users\.*\Desktop\.*"))))

  (forall var="i" from="0" to="2"
    (edge name="count"
      (pointcutid "new_desktop_file")
      (nodes var="s" i, i+1)))

  (edge name="too_many_files"
    (pointcutid "new_desktop_file")
    (nodes var="s" 3, #)))
SPOX Pointcut Extensions

\textbf{argval}(i, ValuePredicate) – arg \textit{i} of operation satisfies \textit{ValPred}

\textbf{argtyp}(i, TypePattern) – arg \textit{i} has type \textit{TypePattern}

\textbf{Value Predicates}

\textbf{intle}(n) – matches integer values no greater than \textit{n}

\textbf{streq}(Regexp) – matches string values matching \textit{Regexp}
SPoX

- SPoX specifications denote security automata
- Denotational semantics:

\[ q \in Q = (SV \uplus (Obj \times SV)) \rightarrow N \]
\[ S \in SM = (SV \uplus (ID \times SV)) \rightarrow N \]
\[ I \in IM = IV \rightarrow N \]
\[ b \in Bnd = ID \rightarrow Obj \]
\[ r \in OBnd = Bnd \uplus \{ \text{Fail} \} \]
\[ P : pol \rightarrow (\gamma \times 2^Q \times \gamma \times ((Q \times JP) \rightarrow 2^Q)) \]
\[ E_S : edg \rightarrow IM \rightarrow 2^{(JP \rightarrow OBnd)} \times SM \times SM \]
\[ PC : pcd \rightarrow JP \rightarrow OBnd \]
\[ EP : s \rightarrow IM \rightarrow (SM \times SM) \]
\[ A : a \rightarrow IM \rightarrow N \]

security states
state-variable maps
iteration var maps
bindings
optional bindings
policy denotations
edgeset denotations
pointcut denotations
endpoint constraints
arithmetic

\[ \mathcal{P}[edg_1 \ldots edg_n] = (Q, \{ q_0 \}, JP, \delta) \]
where \( q_0 = (SV \uplus (Obj \times SV)) \times \{ 0 \} \)
and \( \delta(q, jp) = \{ q[S'[b]] \mid (f, S, S') \in \cup_{1 \leq i \leq n} E_S[edg_i] \downarrow, f(jp) = b, S[b] \subseteq q \} \)

\[ E_S[\text{forall var} = \text{iv} \text{ from} = a_1 \text{ to} = a_2 <\text{edg}</\text{forall}>]I = \cup_{A_1 \leq A_2} (E_S[edg](I[j/iv])) \]

\[ E_S[\text{edge} > pcd \text{ ep}_1 \ldots \text{ ep}_n</\text{edge}>] I = \{ (PC[pcd], \cup_{1 \leq j \leq n} S_j, \cup_{1 \leq j \leq n} S'_j) \} \]
where \( \forall j \in N. (1 \leq j \leq n) \Rightarrow ((S_j, S'_j) = EP[ep_j]I) \)

\[ PC[pcd]jp = \text{match-pcd}(pcd)jp \]

\[ EP[\text{nodes} \text{ var} = \text{sv} > a_1, a_2</\text{nodes}>]I = \{ ((sv, A[a_1]I)), ((sv, A[a_2]I)) \} \]

\[ EP[\text{nodes} \text{ obj} = \text{id} \text{ var} = \text{sv} > a_1, a_2</\text{nodes}>]I = \{ (((id, sv), A[a_1]I)), (((id, sv), A[a_2]I)) \} \]

\[ A[n]I = n \quad A[a_1 + a_2]I = A[a_1]I + A[a_2]I \]
\[ A[iv]I = I(iv) \quad A[a_1 - a_2]I = A[a_1]I - A[a_2]I \]
\[ A[a_1 \cdot a_2]I = A[a_1]I \cdot A[a_2]I \]
\[ A[a_1/a_2]I = A[a_1]I/A[a_2]I \]

[Hamlen & Jones, PLAS ‘08]
Debugging SPOX Policies

Buggy policy:

(policy
  (state name="s")

  (pointcut name="new_desktop_file"
    (and (call "File.new")
      (argval 1 (streq ".*\Users\\.*\Desktop\\.*")))

  (forall var="i" from="0" to="3"
    (edge name="count"
      (pointcutid "new_desktop_file")
      (nodes var="s" i, i+1)))

  (edge name="too_many_files"
    (pointcutid "new_desktop_file")
    (nodes var="s" 3, #)))
Debugging SPoX Policies

Non-deterministic automaton:

```plaintext
s=0 ----new_desktop_file----> s=1
     ^         |         |
     |         v         |
     s=2 ----new_desktop_file----> s=3

     ^         |         |
     |         v         |
     s=3 ----new_desktop_file----> s=4

s=#
```
Aspect-Oriented IRMs

An alternate version:

```java
import java.util.regex.*;

SafeFile() {
    static int counter = 0;
    static final String REGEX = ".*\Users\.*\Desktop\.*";
    static final int MAX_FILES = 3;
    event createFile before() : call(File.new(filename)) {
        if (Pattern.matches(REGEX, filename)) counter++;
    }
    event illegalCreateFile before() : call(*.new(filename)) {
        if (Pattern.matches(REGEX, filename) && counter >= MAX_FILES)
            System.exit(1);
    }
}
```
Debugging SPoX Policies

• Declarative policies allow detection of non-determinism

• SPoX non-determinism detection is reducible to:
  – Integer linear programming
    • Determines ambiguous state transitions
  – Boolean satisfiability (SAT)
    • Determines overlap of pointcuts
  – Regular language non-emptiness
    • Determines overlap of regular expressions in pointcuts

[Jones & Hamlen, AOSD ‘10]
Debugging SPoX Policies

- SAT reduction for pointcuts

\[
E_1 \text{ label:} \\
(\text{and} \\
(\text{call "File.open"}) \\
(\text{argval 1 (streq "filename"))}) \\
\rightarrow A*B
\]

\[
E_2 \text{ label:} \\
(\text{or} \\
(\text{call "File.open"}) \\
(\text{call "File.close"})) \\
\rightarrow A+C
\]

Constraint: \(\neg (A*C)\)

\[
S = (A*B)*(A+C)*\neg (A*C)
\]
## Constraint Chart

<table>
<thead>
<tr>
<th></th>
<th>call</th>
<th>exec</th>
<th>get</th>
<th>set</th>
<th>argv</th>
<th>targ</th>
<th>argt</th>
<th>with</th>
</tr>
</thead>
<tbody>
<tr>
<td>call</td>
<td>CR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>exec</td>
<td>E</td>
<td>CR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>get</td>
<td>E</td>
<td>E</td>
<td>CR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>set</td>
<td>E</td>
<td>E</td>
<td>E</td>
<td>CR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>argv</td>
<td>C</td>
<td>C</td>
<td>E</td>
<td>C</td>
<td>C</td>
<td>RV</td>
<td></td>
<td></td>
</tr>
<tr>
<td>targ</td>
<td>CR</td>
<td>CR</td>
<td>C</td>
<td>C</td>
<td>I</td>
<td>CR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>argt</td>
<td>C</td>
<td>C</td>
<td>E</td>
<td>C</td>
<td>T</td>
<td>I</td>
<td>CR</td>
<td></td>
</tr>
<tr>
<td>with</td>
<td>I</td>
<td>E</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>CR</td>
</tr>
</tbody>
</table>

**Legend:**

- **I**: independent (no constraint required)
- **E**: mutually exclusive (use constraint $\neg(a \land b)$)
- **C**: independent except for known classes
- **R**: regular expression non-emptiness check
- **V**: argval check
- **T**: argval–argtyp compatibility check
Non-Determinism Tool Runtimes

• Implemented in Java using Prolog CLP
• 9200 lines of Java; Prolog code is dynamically generated
• Median runtime per line of policy code: 3.2 ms

<table>
<thead>
<tr>
<th>Policy</th>
<th>Size (chars)</th>
<th>Pointcut vars</th>
<th>CNF vars</th>
<th>CNF clauses</th>
<th>Runtimes</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileMode</td>
<td>1488</td>
<td>9</td>
<td>1764</td>
<td>2061</td>
<td>2243ms</td>
</tr>
<tr>
<td>FileModeFixed</td>
<td>1570</td>
<td>8</td>
<td>706</td>
<td>850</td>
<td>248ms</td>
</tr>
<tr>
<td>Logger</td>
<td>722</td>
<td>2</td>
<td>10</td>
<td>12</td>
<td>235ms</td>
</tr>
<tr>
<td>LoggerFixed</td>
<td>956</td>
<td>3</td>
<td>34</td>
<td>40</td>
<td>156ms</td>
</tr>
<tr>
<td>GetPermission</td>
<td>938</td>
<td>5</td>
<td>44</td>
<td>57</td>
<td>90ms</td>
</tr>
<tr>
<td>GetPermissionFixed</td>
<td>1189</td>
<td>5</td>
<td>49</td>
<td>61</td>
<td>85ms</td>
</tr>
<tr>
<td>FileExists</td>
<td>437</td>
<td>3</td>
<td>10</td>
<td>14</td>
<td>35ms</td>
</tr>
<tr>
<td>FileExistsFixed</td>
<td>423</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>25ms</td>
</tr>
<tr>
<td>NoFreeride</td>
<td>1024</td>
<td>3</td>
<td>22</td>
<td>28</td>
<td>797ms</td>
</tr>
<tr>
<td>NoFreerideFixed</td>
<td>1075</td>
<td>3</td>
<td>18</td>
<td>22</td>
<td>825ms</td>
</tr>
<tr>
<td>Encrypt</td>
<td>986</td>
<td>3</td>
<td>34</td>
<td>40</td>
<td>163ms</td>
</tr>
<tr>
<td>Log&amp;Encrypt</td>
<td>2281</td>
<td>4</td>
<td>972</td>
<td>1180</td>
<td>538ms</td>
</tr>
<tr>
<td>Log&amp;EncryptFixed</td>
<td>2321</td>
<td>4</td>
<td>578</td>
<td>703</td>
<td>391ms</td>
</tr>
</tbody>
</table>
Service-Oriented IRMs

Geospatial data server

Applet server

Client

Applet requests drug stores in city

Returns drug stores in city

SPoX Java Rewrite service

JAR File:  
Policy File:  
Rewrite

Trusted Computing Base (TCB)

Request applet

Provide applet

Submit applet and policy

Return safe applet
Service-Oriented IRMs

- Web service provides trusted in-liner
- Under submission to MobiWIS 2011
- Statistics:

<table>
<thead>
<tr>
<th>Program</th>
<th>Original Size (KB)</th>
<th>Rewritten size (KB)</th>
<th>Rewrite time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>jWeatherWatch</td>
<td>140</td>
<td>146</td>
<td>4.3</td>
</tr>
<tr>
<td>Google.mE</td>
<td>921</td>
<td>513</td>
<td>21.7</td>
</tr>
<tr>
<td>Jeti</td>
<td>533</td>
<td>474</td>
<td>20.4</td>
</tr>
</tbody>
</table>
End Goal: Formally Certified IRMs

- Policy
- Rewriter
- Verifier
- Untrusted binary
- Rewritten binary

Yes -> accept
No -> reject
Conclusion

• **Main Contributions**
  1. **SPoX (Security Policy XML)**
     - Fully declarative, aspect-oriented policy language
     - Working rewriter
  2. **Automated policy analysis**
     - Tool detects non-determinism in SPoX policies
  3. **Service-oriented IRMs**
     - Web service provides trusted rewriting service
  4. **Automated IRM certification**
     - Next time!
References


References

