Security through Static Analysis

Chris Hankin, Imperial College

Thanks to: SecSafe partners, David Clark and Sebastian Hunt
Overview

- SecSafe objectives
- Carmel and Security
- Flow Logic
- Information Flow
- CFA for Carmel
- Conclusions
Security and Safety

Algorithms

model
realisation / constraints

Static Analysis

proof
implementation

Semantics

Automatic Tools

Verificard Workshop, January 2002, Marseille
A subset of Javacard (Carmel) - Motivation:
- hiding of uninteresting language and JCVM details
- focus on salient features
- reduction of specification and development effort
- (almost) direct translation from JCVM language

⇒ the essence [JCVMLe]

<table>
<thead>
<tr>
<th>JCVM language</th>
<th>Carmel</th>
</tr>
</thead>
<tbody>
<tr>
<td>185 low-level instructions</td>
<td>30 high-level instructions</td>
</tr>
<tr>
<td>AID, tokens, offsets</td>
<td>names</td>
</tr>
</tbody>
</table>
Memory allocation control

- Dynamic memory allocation must be bounded.
- No memory must be allocated after personalization.

Information flow control

- Given types of information must not flow outside the applet.

Service control

- Given program points must be executable only if given conditions are satisfied.

Error prediction

- No exception must reach the toplevel except ISOExceptions.
Flow logic:
a multi-paradigmatic approach to static analysis

- Specification oriented
- Semantics based *not* semantics directed
- Integrates state-of-the-art from abstract interpretation and data flow analysis
- Multi-paradigmatic: functional, imperative, concurrent ...
Information Flow for Algol-like Languages

- Information Flow Analysis
  - Prevent flow from *high* to *low*

- Flow Logic specification
  - Simple imperative language
  - Idealised Algol

- Extended with probabilistic constructs
Following Denning, it is possible to categorise information flows into **direct vs indirect** and **explicit vs implicit** flows.

**Indirect flows:** transitive flows (a flow from $x$ to $y$ followed by a flow from $y$ to $z$ implies a flow from $x$ to $z$)

**Direct explicit flows:** arise from assignments; for example, $x := y + z$ causes explicit information flows from both $y$ and $z$ to $x$. 
Direct implicit flows:

- **Local flows** arise from guards in conditionals:

  \[
  \text{if } x \text{ then } y := z \text{ else } y := w.
  \]

- **Global flows** arise from guards in while loops

  \[
  x := y;(\text{while } w \text{ do } x := z);\ldots.
  \]
We will illustrate the approach for a simple imperative language:

\[ S \in \text{Statement}, \ C \in \text{Command} \]
\[ \ell \in \text{Lab}, \ x \in \text{Ide} \]
\[ a \in \text{Arith-exp}, \ b \in \text{Bool-exp} \]

\[
S \ ::= \ C^\ell \\
C \ ::= \ \text{skip} \mid x := a \mid S_1;S_2 \mid \\
\text{if } b \text{ then } S_1 \text{ else } S_2 \mid \text{while } b \text{ do } S \mid \\
\text{new } x.S
\]

Verificard Workshop, January 2002, Marseille
$\text{skip}^\ell, \sigma \Downarrow \text{void}, \sigma$

$a, \sigma \Downarrow v, \sigma$

$(x:=a)^\ell, \sigma \Downarrow \text{void}, \sigma[x \mapsto v]$

$S_1, \sigma \Downarrow \text{void}, \sigma' \quad S_2, \sigma' \Downarrow \text{void}, \sigma''$

$(S_1;S_2)^\ell, \sigma \Downarrow \text{void}, \sigma''$
\[
\frac{b, \sigma \downarrow 1, \sigma \quad S_1, \sigma \downarrow \text{void}, \sigma'}{(\text{if } b \text{ then } S_1 \text{ else } S_2)^{\ell}, \sigma \downarrow \text{void}, \sigma'}
\]
\[
\frac{b, \sigma \downarrow 0, \sigma \quad S_2, \sigma \downarrow \text{void}, \sigma'}{(\text{if } b \text{ then } S_1 \text{ else } S_2)^{\ell}, \sigma \downarrow \text{void}, \sigma'}
\]
\[
\frac{b, \sigma \downarrow 0, \sigma}{(\text{while } b \text{ do } S)^\ell, \sigma \downarrow \text{void}, \sigma}
\]

\[
\frac{b, \sigma \downarrow 1, \sigma \quad S, \sigma \downarrow \text{void}, \sigma' \quad (\text{while } b \text{ do } S)^\ell, \sigma' \downarrow \text{void}, \sigma''}{(\text{while } b \text{ do } S)^\ell, \sigma \downarrow \text{void}, \sigma''}
\]

\[
\frac{S, \sigma[x \mapsto 0] \downarrow \text{void}, \sigma'}{(\text{new } x. \ S)^\ell, \sigma \downarrow \text{void}, \sigma'[x \mapsto \sigma x]}
\]
We write

$$(\hat{X}, \hat{G}, \hat{D}) \models S$$

when $(\hat{X}, \hat{G}, \hat{D})$ is an acceptable Information Flow Analysis of the statement $S$.

$$\hat{X} \in \text{Assign} = \text{Lab} \rightarrow \mathcal{P}(\widehat{\text{Ide}})$$

$$\hat{G} \in \text{Global} = \text{Lab} \rightarrow \mathcal{P}(\widehat{\text{Ide}})$$

$$\hat{D} \in \text{Dep} = \text{Lab} \rightarrow \mathcal{P}(\widehat{\text{Ide}} \times \widehat{\text{Ide}})$$

where $\widehat{\text{Ide}} = \text{Ide} \cup \{\bullet\}$. 
We use $;$ for relational composition, thus: $x R ; S z$ iff $\exists y. x R y S z$. We also overload this notation to allow the ‘composition’ of a set with a relation, thus:

$$Y ; R \overset{\text{def}}{=} \{z \mid \exists y \in Y. y R z\}.$$  

We use the notation $f \setminus x$ to restrict the range of a partial function, thus: $(f \setminus x)(y)$ is undefined if $x = y$ and is $f(y)$ otherwise. We apply the same notation to binary relations:

$$R \setminus x \overset{\text{def}}{=} \{(y, z) \in R \mid y \neq x\}.$$  

Where convenient, we treat $D(\ell)$ as a function of type $\widehat{\text{Id}_e} \rightarrow \mathcal{P}(\widehat{\text{Id}_e})$. In particular, we use a ‘function update’ notation on relations thus: $R[x \mapsto Y] \overset{\text{def}}{=} R \setminus x \cup \{x\} \times Y$. 
\[(\hat{X}, \hat{G}, \hat{D}) \models \text{skip}^\ell \text{ iff } \hat{D}(\ell) \supseteq \text{Id}\]

\[(\hat{X}, \hat{G}, \hat{D}) \models (x := a)^\ell \]
iff \[\hat{X}(\ell) \supseteq \{x\} \land \hat{D}(\ell) \supseteq \text{Id}[x \mapsto \text{FV}(a)]\]

\[(\hat{X}, \hat{G}, \hat{D}) \models (C_1^{\ell_1}; C_2^{\ell_2})^\ell \]
iff \[\hat{X}(\ell) \supseteq \hat{X}(\ell_1) \cup \hat{X}(\ell_2) \land \hat{D}(\ell) \supseteq \hat{D}(\ell_1) \land \hat{G}(\ell_1) \cup \hat{G}(\ell_2) ; \hat{D}(\ell_1) \land \hat{D}(\ell) \supseteq \hat{D}(\ell_2) ; \hat{D}(\ell_1)\]
\[(\hat{X}, \hat{G}, \hat{D}) \models (\text{if } b \text{ then } C_1^{\ell_1} \text{ else } C_2^{\ell_2})^l\]

iff

\[(\hat{X}, \hat{G}, \hat{D}) \models C_1^{\ell_1} \land (\hat{X}, \hat{G}, \hat{D}) \models C_2^{\ell_2} \land\]

\[\hat{X}(\ell) \supseteq \hat{X}(\ell_1) \cup \hat{X}(\ell_2) \land\]

\[\hat{G}(\ell) \supseteq \hat{G}(\ell_1) \cup \hat{G}(\ell_2) \land\]

\[(\bullet \in \hat{G}(\ell) \Rightarrow \hat{G}(\ell) \supseteq \text{FV}(b)) \land\]

\[\hat{D}(\ell) \supseteq \hat{D}(\ell_1) \cup \hat{D}(\ell_2) \land\]

\[\hat{D}(\ell) \supseteq \hat{X}(\ell) \times \text{FV}(b)\]
\((\hat{X}, \hat{G}, \hat{D}) \models (\text{while } b \text{ do } C'^{l_1})^l\)

equiv \[(\hat{X}, \hat{G}, \hat{D}) \models C'^{l_1} \land \]
\[\hat{X}(l) \supseteq \hat{X}(l_1) \land \]
\[\hat{G}(l) \supseteq \{\bullet\} \cup \text{FV}(b) \cup \hat{G}(l_1) \cup \hat{G}(l) ; \hat{D}(l_1) \land \]
\[\hat{D}(l) \supseteq \text{Id} \cup \hat{D}(l) ; \hat{D}(l_1) \land \]
\[\hat{D}(l) \supseteq \hat{X}(l) \times \text{FV}(b)\]

\[(\hat{X}, \hat{G}, \hat{D}) \models (\text{new } x. \ C'^{l_1})^l\]

equiv \[(\hat{X}, \hat{G}, \hat{D}) \models C'^{l_1} \land \]
\[\hat{X}(l) \supseteq \hat{X}(l_1) \setminus \{x\} \land \]
\[\hat{G}(l) \supseteq \hat{G}(l_1) \setminus \{x\} \land \]
\[\hat{D}(l) \supseteq \hat{D}(l_1) \setminus \{x\} \cup \{(x, x)\}\]
We are concerned with three aspects of correctness:

- First, that the analysis is well-defined.
- Second, that the analysis results are a proper abstraction of the semantics.
- Third, that every program has an acceptable information flow analysis and that the constraints have solutions.

Having analysed a program, $C^\ell$, we determine that there is a breach of security if either

- $H \cap \hat{G}(\ell) \neq \emptyset$, or
- $\exists x \in L. \exists y \in H. x \hat{D}(\ell) y$
First we consider an example:

\[
( ( (\text{while} (x < 3) \\
\quad \text{do} (((\text{if} (p = g) \\
\quad \text{then} (f := 1)_{l_{10}} \\
\quad \text{else} (f := 0)_{l_{11}})_{l_{8}}; \\
\quad (x := x + 1)_{l_{9}})_{l_{6}}; \\
\quad (g := g + 10)_{l_{7}})_{l_{5}})_{l_{3}}; \\
\quad (f := 2)_{l_{4}})_{l_{1}}; \\
\quad (x := 0)_{l_{2}})_{l_{0}})
\]

Vericard Workshop, January 2002, Marseille
The analysis of this program produces a set of constraints to be solved:

\[
\begin{align*}
\hat{X}(\ell_0) \supseteq & \hat{X}(\ell_1) \cup \hat{X}(\ell_2) \land \hat{G}(\ell_0) \supseteq \hat{G}(\ell_1) \cup \hat{G}(\ell_2) ; \hat{D}(\ell_1) \\
& \land \hat{D}(\ell_0) \supseteq \hat{D}(\ell_2) ; \hat{D}(\ell_1) \\
\hat{X}(\ell_1) \supseteq & \hat{X}(\ell_3) \cup \hat{X}(\ell_4) \land \hat{G}(\ell_1) \supseteq \hat{G}(\ell_3) \cup \hat{G}(\ell_4) ; \hat{D}(\ell_3) \\
& \land \hat{D}(\ell_1) \supseteq \hat{D}(\ell_4) ; \hat{D}(\ell_3) \\
\hat{X}(\ell_2) \supseteq & \{x\} \land \hat{D}(\ell_2) \supseteq \text{Id}[x \mapsto \emptyset] \\
\vdots
\end{align*}
\]
Iterating over these constraints beginning from $\hat{X} = \lambda x.\emptyset$, $\hat{G} = \lambda x.\emptyset$, and $\hat{D} = \lambda x.\emptyset$ to a fixed point giving the least solution yields:

$$
\begin{align*}
\hat{X}(\ell_0) &= \{f, x, g\} \\
\hat{G}(\ell_0) &= \{\bullet, x\} \\
\hat{D}(\ell_0) &= \{(p, p), (g, g), (g, x)\}
\end{align*}
$$

which satisfy the security criteria for the while language (cf type-based approaches).

We now return to the correctness . . .
The specification of the analysis is essentially defining the relation:

\[ \models : (\text{Assign} \times \text{Global} \times \text{Dep} \times \text{Statement}) \rightarrow \{\text{true}, \text{false}\} \]

\[ Q : ((\text{Assign} \times \text{Global} \times \text{Dep} \times \text{Statement}) \rightarrow \{\text{true}, \text{false}\}) \rightarrow ((\text{Assign} \times \text{Global} \times \text{Dep} \times \text{Statement}) \rightarrow \{\text{true}, \text{false}\}) \]

\[ Q_1 \sqsubseteq Q_2 \iff \forall(\hat{X}, \hat{G}, \hat{D}, S) : (Q_1(\hat{X}, \hat{G}, \hat{D}, S) = \text{true}) \Rightarrow (Q_2(\hat{X}, \hat{G}, \hat{D}, S) = \text{true}) \]
Given a set of variables $X$, we write $\sigma_1 \sim_X \sigma_2$ to mean that the two stores agree on all $x \in X$:

$$\sigma_1 \sim_X \sigma_2 \Leftrightarrow \forall x \in X. \sigma_1(x) = \sigma_2(x)$$

Clearly, $\sim_X$ is an equivalence relation for any choice of $X$. We sometimes write $\sim_x$ to mean $\sim\{x\}$. 
Assignment Freedom

Suppose \((\hat{X}, \hat{G}, \hat{D}) \models C^\ell\) and let \(X' = \{x \in \text{Ide} \mid x \not\in \hat{X}(\ell)\}\). Then:

1. if \(C^\ell, \sigma \downarrow \text{void}, \sigma'\) then \(\sigma' \sim_{X'} \sigma\)
2. if \(x \not\in \hat{X}(\ell)\) then \(x \hat{D}(\ell) x\)

Store Independence

Suppose \((\hat{X}, \hat{G}, \hat{D}) \models C^\ell\), then, for all \(x\):

\[
\text{if } (\sigma_1 \sim_{D_x} \sigma_2) \text{ then } (C^\ell, \sigma_1 \downarrow \text{void}, \sigma'_1 \land C^\ell, \sigma_2 \downarrow \text{void}, \sigma'_2) \text{ then } \sigma'_1 \sim_x \sigma'_2
\]

where \(D_x = \hat{D}(\ell)(x)\).

Proof: Proof is by induction on the height of the first derivation.
Termination Independence

Suppose \((\widehat{X}, \widehat{G}, \widehat{D}) \models C^\ell\). Then:

1. if \(\bullet \not\in \widehat{G}(\ell)\) then \(C^\ell, \sigma \Downarrow\) for all \(\sigma\).
2. if \(\sigma_1 \sim_{\widehat{G}(\ell)} \sigma_2\) then \(C^\ell, \sigma_1 \Downarrow \iff C^\ell, \sigma_2 \Downarrow\)

Proof: Part 1 is by structural induction. Part 2 is by induction on the height of the derivation.
Existence of solutions

For all \( S \in \text{Statement} \) the set \( \{ (\hat{X}, \hat{G}, \hat{D}) \mid (\hat{X}, \hat{G}, \hat{D}) \models S \} \) is a Moore family.

Recall that a subset \( Y \) of a complete lattice \( L = (L, \sqsubseteq) \) is a Moore family if and only if \( \sqcap Y' \in Y \) for all \( Y' \subseteq Y \).

An immediate corollary of our result is that there is always an acceptable information flow analysis for a statement and that, moreover, there is a least analysis.
We now return to Carmel.

The main issue is that we have to deal with method invocation. This essentially means that we need an inter-procedural information flow analysis.

But … which methods are being invoked?
Flow Logic for Carmel (Rene Rydhof Hansen)

- Control Flow Analysis for Carmel
- Proved correct wrt. semantics
- Extensions for exceptions, ownership (firewall) etc.
- Basis for prototype implementation
\( \hat{K} \) tracks values of static fields for each class, \( \hat{H} \) tracks values of instance fields for individual objects, \( \hat{L} \) is the local heap and \( \hat{S} \) is the abstract operand stack. Judgements are of the form:

\[
(\hat{K}, \hat{H}, \hat{L}, \hat{S}) \models \text{addr} : \text{instr}
\]

**Analysing push:**

\[
(\hat{K}, \hat{H}, \hat{L}, \hat{S}) \models (m_0, pc_0) : \text{push} \ t \ v \\
\text{iff} \quad \{v\} :: \hat{S}(m_0, pc_0) \subseteq \hat{S}(m_0, pc_0 + 1) \\
\hat{L}(m_0, pc_0) \subseteq \hat{L}(m_0, pc_0 + 1)
\]
Analysing \texttt{invokevirtual}:

\[(\hat{K}, \hat{H}, \hat{L}, \hat{S}) \models (m_0, pc_0): \textbf{invokevirtual} m \text{ iff} \]
\[A_1 :: \cdots :: A_{|m|} :: B :: X < \hat{S}(m_0, pc_0) : \]
\[\forall (\text{Ref}\sigma) \in B : \]
\[m_v = \text{methodLookup}(m.\text{id}, \sigma) \]
\[\{(\text{Ref}\sigma)\} :: A_1 :: \cdots :: A_{|m|} \sqsubseteq \hat{L}(m_v, 1)[0..|m|] \]
\[T :: Y < \hat{S}(m_v, \text{END}_{m_v}) : \]
\[T :: X \sqsubseteq \hat{S}(m_0, pc_0 + 1) \]
\[\hat{L}(m_0, pc_0) \sqsubseteq \hat{L}(m_0, pc_0 + 1) \]
The information flow analysis of \texttt{invokevirtual} might look something like:

\[
(\hat{X}, \hat{G}, \hat{D}) \models (K, \hat{H}, \hat{L}, \hat{S}) (m_0, pc_0) : \texttt{invokevirtual } m \iff \\
A_1 :: \cdots :: A_{|m|} :: B :: X \triangleleft \hat{S}(m_0, pc_0) :: \\
\forall (\text{Ref} \sigma) \in B : \\
\begin{align*}
  m_v & = \text{methodLookup}(m.id, \sigma) \\
  \hat{D}(m_0, pc_0)^+ & \subseteq \hat{D}(m_v, 1) \\
  \hat{X}(m_v, \text{END}_{m_v}) & \subseteq \hat{X}(m_0, pc_0 + 1) \\
  \hat{G}(m_v, \text{END}_{m_v}) & \subseteq \hat{G}(m_0, pc_0 + 1) \\
  \hat{D}(m_v, \text{END}_{m_v})^- & \subseteq \hat{D}(m_0, pc_0 + 1)
\end{align*}
\]
Conclusions

We have seen:

- SecSafe objectives
- Flow Logic
- Information Flow Analysis

It remains to:

- develop the Information Flow Logic for Carmel
- to develop other security analyses