SWORD

Front End Filter

Verification, Phase 1

A Specification and Security Proof
in
Higher Order Logic
using
ProofPower

Aims of the Contract:

- Assessment of applicability of computer assisted formal proof in development of secure systems.

- To undertake formal parts of development of prototype secure DBMS.

Specific Objectives:

- Verification of security of SSQL specifications

- Verification of formal model of implementation of .

Policy (I)

SML

```
declare_type_abbrev
("BEHAVIOURS",["'QUERY","'DATA"],
 ⌜: ('QUERY × Class)LIST → (Class × 'DATA)LIST⌝);
```

HOL Constant

same_ins : $Class \rightarrow$
$(('QUERY \times Class)LIST \leftrightarrow ('QUERY \times Class)LIST)$

$\forall clear$: $Class$; $si1\ si2$: $('QUERY \times Class)LIST$
• $(si1,si2) \in same\_ins\ clear$
$\Leftrightarrow$
$let\ v = \{(q,c)|(clear\ dominates\ c)\}$
in
$si1 \restriction v = si2 \restriction v$

HOL Constant

same_outs : $Class \rightarrow$
$((Class \times 'DATA)LIST \leftrightarrow (Class \times 'DATA)LIST)$

$\forall clear$: $Class$; $so1\ so2$: $(Class \times 'DATA)LIST$
• $(so1,so2) \in same\_outs\ clear$
$\Leftrightarrow$
$let\ v = \{(c,d)|(clear\ dominates\ c)\}$
in
$so1 \restriction v = so2 \restriction v$

Policy (II)

HOL Constant

secure: $('QUERY,'DATA)BEHAVIOURS\ \mathbb{P}$

$\forall bm{:}('QUERY,'DATA)BEHAVIOURS$

$\bullet$ $bm \in secure$

$\Leftrightarrow$

$\forall\ clear : Class;\ si1\ si2 : ('QUERY \times Class)LIST$

$\bullet$ $(si1,si2) \in same\_ins\ clear$

$\Rightarrow$

$(bm\ si1, bm\ si2) \in same\_outs\ clear$

State Transition Models

SML
```
declare_type_abbrev("Stf",[],
 ⌜: (Query × Class) × State →
  State × (Class × (Data LIST LIST × Errors))⌝);
```

An "Abstract Machine" is a transition function together with an initial state:

SML
```
declare_type_abbrev("Am",[],⌜: Stf × State⌝);
```


Behavioural Abstraction

HOL Constant

iterate :

$$(((' QUERY \times Class) \times ' STATE) \rightarrow (' STATE \times (Class \times ' DATA))) \rightarrow (((' QUERY \times Class) LIST \times ' STATE) \rightarrow (' STATE \times ((Class \times ' DATA) LIST)))$$

$$T$$

We define *behaviours* as an iterated transition function from the initial state:

HOL Constant

behaviours :

$$((((' QUERY \times Class) \times ' STATE) \rightarrow (' STATE \times (Class \times ' DATA))) \times ' STATE)$$

$$\rightarrow (' QUERY, ' DATA)\ BEHAVIOURS$$

$$\forall\ tf : (((' QUERY \times Class) \times ' STATE) \rightarrow (' STATE \times (Class \times ' DATA)));\ istate : ' STATE;\ si : (' QUERY \times Class) LIST$$

$$\bullet$$

$$behaviours(tf, istate)\ si = Snd((iterate\ tf)(si, istate))$$

Top Level Structure of SSQL Specification

The security relevant aspects of the SSQL specification are separated from other aspects by splitting the specification into three parts:

"Hide" filters the state to obtain a view from a particular classification.

SML

```
declare_type_abbrev("Hide",[],
    ⌜: Class × State → State⌝);
```

"Process" encapsulates the non-security aspects of the semantics of the Query Language.

SML

```
declare_type_abbrev("Process",[],
    ⌜: Query × Class × State → Effect × Errors⌝);
```

"Ustate" updates the state in a secure way using the results of the untrusted query processing.

SML

```
declare_type_abbrev("Ustate",[],
    ⌜: Class × (Effect × Errors) × State →
      State × (Class × (Data LIST LIST × Errors))⌝);
```


Building the Transition Function

The Hide, Process and Ustate specifications are combined to give a specification of the transition function:

HOL Constant

mkTf : $Hide \rightarrow Process \rightarrow Ustate \rightarrow Stf$

$\forall h{:}Hide;\ p{:}Process;\ u{:}Ustate;\ q : Query;$
$c : Class;\ s : State$
• $(mkTf\ h\ p\ u)\ ((q,c),s)$
$=$
$u\ (c,\ p(q,\ c,\ h(c,\ s)),\ s)$

HOL Constant

isstate : $State$

T

HOL Constant

SSQLam : Am

$SSQLam =$
$(mkTf\ hide\ processQuery\ updateState,\ isstate)$

?⊢ $behaviours\ SSQLam \in secure$

"Critical Requirements" on Components
Hide

HOL Constant

$$\mathbf{secureHide} : \mathit{Hide}\ \mathbb{P}$$

$$\forall\ h{:}\mathit{Hide} \bullet$$
$$h \in \mathit{secureHide} \Leftrightarrow$$
$$\forall c_1\ c_2 : \mathit{Class};\ s_1\ s_2 : \mathit{State} \bullet$$
$$h(c_1,s_1) = h(c_1,s_2) \wedge c_1\ \mathit{dominates}\ c_2$$
$$\Rightarrow h(c_2,s_1) = h(c_2,s_2)$$

"Critical Requirements" on Components Update

HOL Constant

$$\textbf{secureUpdate} : \mathit{Hide} \leftrightarrow \mathit{Ustate}$$

$$\forall\ h : \mathit{Hide}\ ; u : \mathit{Ustate} \bullet$$
$$(h,u) \in \mathit{secureUpdate}$$
$$\Leftrightarrow$$
$$(\forall c_1\ c_2 : \mathit{Class};\ s : \mathit{State};\ e{:}\mathit{Effect} \times \mathit{Errors} \bullet$$
$$\mathit{let}\quad s' = \mathit{Fst}(u(c_1,e,s))$$
$$\mathit{in}\quad \neg(h(c_2,s) = h(c_2,s'))$$
$$\Rightarrow c_2\ \mathit{dominates}\ c_1)$$
$$\wedge$$
$$(\forall c_1\ c_2{:}\mathit{Class};\ s_1\ s_2{:}\mathit{State};\ e{:}\mathit{Effect} \times \mathit{Errors} \bullet$$
$$\mathit{let}\quad s'_1 = \mathit{Fst}(u(c_2,e,s_1))$$
$$\mathit{and}\quad s'_2 = \mathit{Fst}(u(c_2,e,s_2))$$
$$\mathit{in}\quad h(c_1,s_1) = h(c_1,s_2)$$
$$\wedge\ c_1\ \mathit{dominates}\ c_2$$
$$\Rightarrow h(c_1,s'_1) = h(c_1,s'_2))$$
$$\wedge$$
$$(\forall c : \mathit{Class};\ s_1\ s_2{:}\mathit{State};\ e{:}\mathit{Effect} \times \mathit{Errors} \bullet$$
$$\mathit{let}\quad o_1 = \mathit{Snd}(u(c,e,s_1))$$
$$\mathit{and}\quad o_2 = \mathit{Snd}(u(c,e,s_2))$$
$$\mathit{in}\quad h(c,s_1) = h(c,s_2) \Rightarrow o_1 = o_2)$$
$$\wedge$$
$$(\forall c : \mathit{Class};\ s{:}\mathit{State};\ e{:}\mathit{Effect} \times \mathit{Errors} \bullet$$
$$\mathit{Fst}(\mathit{Snd}(u(c,e,s))) = c)$$

The "Unwinding" Lemma

HOL Constant

Lemma2 : $BOOL$

$$
\begin{aligned}
&Lemma2 = \\
&\quad ((hide \in secureHide \\
&\quad \wedge \\
&\quad (hide, updateState) \in secureUpdate) \\
&\quad \Rightarrow \\
&\quad behaviours\ SSQLam \in secure)
\end{aligned}
$$

Phase 1 Results

- Specifications reorganised, machine checked and debugged.
- "Unwinding" Proof completed.
- Security conjecture proven.
- All specifications and proofs developed and checked using ProofPower v0.1 (since transferred to ProofPower v0.3).

Success Factors

- Customised Formal Security Model.
- Simple Security Model.
- Well Structured Specifications.
- Good Tool Support ProofPower
- Experienced and Competent Staff.
- Good Customer/Supplier Relations

