

Using Types to Embed Authorization Policies into Spi

Michael Carl Tschantz

October 3, 2007

This is presentation on
“A Type Discipline for Authorization Policies”
by Fournet, Gordon, and Maffeis (2005)

What We Want

- ▶ TAs may record grades.
- ▶ AI is a TA.

What We're Willing to Have

```
record(GRD) :- ta(U),graded(U,GRD).  
ta('al').
```

What We Have

```
#!/usr/bin/perl

$grades = ();
while(<STDIN>){
    $user = <STDIN>;
    $pw = <STDIN>;
    if($user eq 'al' && $pw eq '456'){
        $grade = <STDIN>;
        push(@grades,$grade);
    }
    printf(FILE, @grades);
}
```

If We're Lucky

```
#!/usr/bin/perl
use strict;
my $grades = ();
while(<STDIN>){
    my $user = <STDIN>;
    my $pw = <STDIN>;
    if($user eq 'al' && $pw eq '456'){
        my $grade = <STDIN>;
        push(@grades,$grade);
    }
    printf(FILE, @grades);
}
```

If We're Really Lucky

```
new grades;  
  !(in stdin(i);  
  match i as ('al', '456', g);  
    !(out grades(g))  
  | !(in grades(g); g)
```

The Gap

```
record(GRD) :- ta(U),graded(U,GRD).  
ta('al').
```

```
new grades;  
  !(in stdin(i);  
    match i as ('al', '456', g);  
      !(out grades(g)))  
| !(in grades(g); g)
```

Combine Them

```
record(GRD) :- ta(U),graded(U,GRD)
| ta('al')
| new grades;
  !(in stdin(i);
    match i as ('al', '456', g);
      !(out grades(g)))
| !(in grades(g); g)
```

```
record(GRD) :- ta(U),graded(U,GRD)
| ta('al')
| new grades;
  !(in stdin(i);
    match i as ('al', '456', g); graded('al', g) |
      !(out grades(g) | expect record(g)))
| !(in grades(g); g | expect record(g))
```

Type Them

```
record(GRD) :- ta(U),graded(U,GRD)
| ta('al')
| new grades:(g:Un,Ok(graded('al', g)));
  !(in stdin(i:(Un,Un,Un)));
  match i as ('al', '456', g:Un); graded('al', g) |
    !(out (grades(g), ok) | expect record(g))
| !(in grades(g:Un, x:Ok(record(g)));
  g | expect record(g))
```

Key Points

- ▶ Use annotations from an authorization logic to mark security-related events
- ▶ Use opponent threat model
- ▶ Use dependent type system to ensure safety against opponent

Authorization Logic

$(\mathcal{C}, fn, \models)$ is an *authorization logic* if for all $C \in \mathcal{C}$, $S \subseteq \mathcal{C}$, and substitutions σ of messages for names,

- ▶ \mathcal{C} is closed under substitutions σ ;
- ▶ $fn(C)$ is a finite set of free names;
- ▶ $C\sigma = C$ if $dom(\sigma) \cap fn(C) = \emptyset$;
- ▶ $fn(C\sigma) \subseteq (fn(C) \setminus dom(\sigma)) \cup fn(\sigma)$;
- ▶ if $S \models C$, then $S \cup \{C'\} \models C$; and
- ▶ if $S \models C$, then $S\sigma \models C\sigma$.

Message Syntax

$M, N ::=$

- x names of a key or channel
- $\{M\}N$ authenticated encryption of M with key N
- (M, N) message pair
- ok** distinguished message

Process Syntax

$P, Q, R ::=$

out $M(N)$

in $M(x:T); P$

new $x:T; P$

$P|Q$

$!P$

0

decrypt M **as** $\{y:T\}N; P$

split M **as** $(x:T, y:U); P$

match M **as** $(N, y:U); P$

C

expect C

asynchronous output of N to channel M

input from channel M bound to x in P

fresh name bound to x in P

parallel composition of P and Q

unbounded parallel composition of P

inactivity

bind y to decryption of M with key N in P

solve $(x, y) = M$ and bind x and y in P

solve $(N, y) = M$ and bind and y in P

statement of a clause C

expectation that clause C is derivable

Structural Equivalence \equiv (1 of 3)

$$\frac{}{P \equiv P}$$

$$\frac{P \equiv Q}{Q \equiv P}$$

$$\frac{P \equiv Q \quad Q \equiv R}{P \equiv R}$$

Structural Equivalence \equiv (2 of 3)

$$\overline{(P|Q)|R} \equiv \overline{P|(Q|R)}$$

$$\overline{P|Q} \equiv \overline{Q|P}$$

$$\frac{P \equiv Q}{\overline{P|R} \equiv \overline{Q|R}}$$

$$\frac{P \equiv Q}{\overline{!P} \equiv \overline{!Q}}$$

$$\overline{!P} \equiv \overline{P|!P}$$

$$\overline{!!P} \equiv \overline{!P}$$

$$\overline{!(P|Q)} \equiv \overline{!P|!Q}$$

$$\overline{P|0} \equiv \overline{P}$$

$$\overline{!0} \equiv \overline{0}$$

Structural Equivalence \equiv (3 of 3)

$$\frac{P \equiv Q}{\mathbf{new} \ x:T; P \equiv \mathbf{new} \ x:T; Q} \qquad \frac{x \notin \mathit{fn}(P)}{\mathbf{new} \ x:T; (P|Q) \equiv P|\mathbf{new} \ x:T; Q}$$

$$\frac{x_1 \neq x_2 \quad x_1 \notin \mathit{fn}(T_2) \quad x_2 \notin \mathit{fn}(T_1)}{\mathbf{new} \ x_1:T_1; \mathbf{new} \ x_2:T_2; P \equiv \mathbf{new} \ x_2:T_2; \mathbf{new} \ x_1:T_1; P}$$

Reduction \rightarrow

$$\frac{P \rightarrow P'}{P|Q \rightarrow P'|Q}$$

$$\frac{P \rightarrow P'}{\mathbf{new} \ x:T; P \rightarrow \mathbf{new} \ x:T; P'}$$

$$\frac{P \equiv Q \quad Q \rightarrow Q' \quad Q' \equiv P'}{P \rightarrow P'}$$

$$\frac{}{\mathbf{out} \ a(M) \mid \mathbf{in} \ a(x:T); P \rightarrow P[M/x]}$$

$$\frac{}{\mathbf{decrypt} \ \{M\}k \ \mathbf{as} \ \{y:T\}k; P \rightarrow P[M/y]}$$

$$\frac{}{\mathbf{split} \ (M, N) \ \mathbf{as} \ (x:T, y:U); P \rightarrow P[M/x][N/y]}$$

$$\frac{}{\mathbf{match} \ (M, N) \ \mathbf{as} \ (M, y:U); P \rightarrow P[N/y]}$$

Example

```
record(GRD) :- ta(U),graded(U,GRD)
| ta('al')
| new grades;
  !(in stdin(i);
    match i as ('al', '456', g); graded('al', g) |
      !(out grades(g) | expect record(g)))
| !(in grades(g); g | expect record(g))
```

Note that statements and expectations do not get reduced. Rather they are used to define *safety*.

A process P is *safe* iff whenever

$$P \rightarrow_{\equiv}^* (\mathbf{expect} C \mid Q)$$

we have that

$$Q \equiv (C_1 \mid \cdots \mid C_n \mid R)$$

such that

$$\{C_1, \dots, C_n\} \models C$$

where $P \rightarrow_{\equiv}^* P'$ iff $P \equiv P'$ or $P \rightarrow^* P'$

Too weak: This should be unsafe, but is judged safe:

```
new x; expect Foo(x)
```

Too strong: This should be safe, but is judged unsafe:

```
Bar() :- Foo(X)|expect Bar()|new y; Foo(y)
```

A process P is *safe* iff whenever

$$P \rightarrow_{\equiv}^* \mathbf{new} \vec{x}:\vec{T}; (\mathbf{expect} C \mid Q)$$

we have that

$$Q \equiv \mathbf{new} \vec{y}:\vec{U}; (C_1 \mid \dots \mid C_n \mid R)$$

such that

$$\{C_1, \dots, C_n\} \models C$$

with $\{\vec{y}\} \cap \mathit{fn}(C) = \emptyset$

Opponents

- ▶ What is another process O attacking our process P ?
- ▶ We model such opponents as an extended SPI process that only uses public data (of the type **Un**) and has no expectations
- ▶ A process is *robustly safe* iff $P|O$ is safe for all opponents O

Example

Let P be

```
record(GRD) :- graded(U,GRD)
| graded(al, bob-hw1-A)
| out grades(bob-hw1-A)
| in grades(g); expect record(g)
```

- ▶ P is Safe
- ▶ P is not robustly safe: Let O be **out** grades(bogus). $P|O$ requires that **expect** record(bogus) be satisfied, which it isn't.
- ▶ **new** grades; P is robustly safe.

Type Syntax

$T, U ::=$

- Un** public data
- Ch**(T) channel for messages of type T
- Key**(T) secret key for plaintexts of type T
- $(x:T, U)$ dependent pair with x bound in U
- Ok**(S) ok to assume the clauses S

$E ::=$

· empty

$E, x:T$ x has type T

E, C C is a valid clause

Let $E(x) = T$ iff $E = E', x:T, E''$

$$\text{dom}(E, C) = \text{dom}(E)$$

$$\text{dom}(E, x:T) = \text{dom}(E) \cup \{x\}$$

$$\text{dom}(\cdot) = \emptyset$$

$$\text{clauses}(E, C) = \text{clauses}(E) \cup \{C\}$$

$$\text{clauses}(E, x:\mathbf{Ok}(S)) = \text{clauses}(E) \cup S$$

$$\text{clauses}(E, x:T) = \text{clauses}(E) \quad \text{where } T \neq \mathbf{Ok}(S)$$

$$\text{clauses}(\cdot) = \emptyset$$

Well-Formed Environments

$$\overline{\cdot \vdash \diamond}$$

$$\frac{E \vdash \diamond \quad \text{fn}(T) \subseteq \text{dom}(E) \quad x \notin \text{dom}(E)}{E, x:T \vdash \diamond}$$

$$\frac{E \vdash \diamond \quad \text{fn}(C) \subseteq \text{dom}(E)}{E, C \vdash \diamond}$$

Typing Messages (1 of 2)

$$\frac{E \vdash \diamond \quad x \in \text{dom}(E)}{E \vdash x : E(x)} \quad \frac{E \vdash M : T \quad E \vdash N : \mathbf{Key}(T)}{E \vdash \{M\}N : \mathbf{Un}}$$

$$\frac{E \vdash M : T \quad E \vdash N : U[M/x]}{E \vdash (M, N) : (x:T, U)}$$

$$\frac{E \vdash \diamond \quad \text{fn}(S) \subseteq \text{dom}(E) \quad \forall C \in S, \text{clauses}(E) \models C}{E \vdash \mathbf{ok} : \mathbf{Ok}(S)}$$

Typing Messages (2 of 2)

$$\frac{E \vdash M : \mathbf{Un} \quad E \vdash N : \mathbf{Un}}{E \vdash \{M\}N : \mathbf{Un}}$$

$$\frac{E \vdash M : \mathbf{Un} \quad E \vdash N : \mathbf{Un}}{E \vdash (M, N) : \mathbf{Un}}$$

$$\frac{E \vdash \diamond}{E \vdash \mathbf{ok} : \mathbf{Un}}$$

Well-Typed Processes (1 of 3)

$$\frac{E \vdash \diamond}{E \vdash \mathbf{0}}$$

$$\frac{E, x:T \vdash P \quad T \text{ is generative (i.e., } \mathbf{Un}, \mathbf{Ch}(U), \text{ or } \mathbf{Key}(U))}{E \vdash \mathbf{new } x:T; P}$$

$$\frac{E \vdash M : \mathbf{Un} \quad E \vdash N : \mathbf{Key}(T) \quad E, y:T \vdash P}{E \vdash \mathbf{decrypt } M \text{ as } \{y:T\} N; P}$$

Well-Typed Processes (2 of 3)

$$\frac{E, C \vdash \diamond \quad \text{clauses}(E) \models C}{E \vdash \mathbf{expect} C}$$

$$\frac{E, C \vdash \diamond}{E \vdash C}$$

Well-Typed Processes (3 of 3)

$$\frac{E, \text{env}(Q)^{\vec{x}} \vdash P \quad E, \text{env}(P)^{\vec{y}} \vdash Q \quad \text{fn}(P|Q) \subseteq \text{dom}(E)}{E \vdash P|Q}$$

$$\begin{aligned} \text{env}(P|Q)^{\vec{x}, \vec{y}} &= \text{env}(P)^{\vec{x}}, \text{env}(Q)^{\vec{y}} && \text{where } \{\vec{x}, \vec{y}\} \cap \text{fn}(P|Q) = \emptyset \\ \text{env}(\mathbf{new } x:T; P)^{x, \vec{x}} &= x:T, \text{env}(P)^{\vec{x}} && \text{where } \{x\} \cap \text{fn}(P|Q) = \emptyset \\ \text{env}(!P)^{\vec{x}} &= \text{env}(P)^{\vec{x}} \\ \text{env}(C)^{\emptyset} &= C \\ \text{env}(P)^{\emptyset} &= \emptyset && \text{otherwise} \end{aligned}$$

(Other inference rules also exist for $E \vdash P$)

Example

```
record(GRD) :- ta(U),graded(U,GRD)
| ta('al')
| new grades:(g:Un,Ok(graded('al', g)));
  !(in stdin(i:(Un,Un,Un)));
  match i as ('al', '456', g:Un); graded('al', g) |
    !(out (grades(g), ok) | expect record(g))
| !(in grades(g:Un, x:Ok(record(g)));
  g | expect record(g))
```

Type Preservation

If $E \vdash P$ and either $P \equiv P'$ or $P \rightarrow P'$, then $E \vdash P'$.

Safety

If $E \vdash P$ and E is generative, then P is safe.

Robust Safety

If $\vec{x}:\vec{\mathbf{Un}} \vdash P$, then P is robustly safe.

Meeting Points

- ▶ **Un**
- ▶ **ok**
- ▶ **Ok(S)**
- ▶

$$\frac{E \vdash \diamond \quad \text{fn}(S) \subseteq \text{dom}(E) \quad \forall C \in S, \text{clauses}(E) \models C}{E \vdash \mathbf{ok} : \mathbf{Ok}(S)}$$

- ▶

$$\frac{E, C \vdash \diamond \quad \text{clauses}(E) \models C}{E \vdash \mathbf{expect} C}$$