Security-by-Contract using Automata Modulo Theory (AMT)

Ida S.R. Siahaan
Is an application trustworthy?

Contract:
specification of application’s behavior concerning security-relevant actions
Is an application trustworthy?

- **Reveal what it does**
  - Design software with security claims

- **Demonstrate its evidence**
  - Check that the application fulfills its claims

- **Verify its compliance**
  - Compliance of Contracts with Policies

- **Assurance for trustworthiness**
  - Inline security policy into the application
  - Run-time monitor the services

**Contract:** specification of application’s behavior concerning security-relevant actions
Is an application trustworthy?

- **Reveal what it does**
  - Design software with security claims

- **Demonstrate its evidence**
  - Check that the application fulfills its claims

- **Verify its compliance**
  - Compliance of Contracts with Policies

- **Assurance for trustworthiness**
  - Inline security policy into the application
  - Run-time monitor the services

---

**Policy:**
specification of application’s acceptable behavior to be executed on a platform concerning security-relevant actions
Is an application trustworthy?

- **Reveal what it does**
  - Design software with security claims

- **Demonstrate its evidence**
  - Check that the application fulfills its claims

- **Verify its compliance**
  - Compliance of Contracts with Policies

- **Assurance for trustworthiness**
  - Inline security policy into the application
  - Run-time monitor the services

Flowchart:
- Check Evidence of Contract
  - correct?
    - Yes: Check Contract-Policy Compliance
    - No: Optimize Policy
- Contract-Policy Compliance
  - match?
    - Yes: Execute
    - No: Inline Policy
- Inline-Policy
  - in-lining?
    - Yes: Execute
    - No: Run-time monitoring
Road Map

Security-by-Contract

Automata Modulo Theory

On-the-fly Matching

Simulation Matching

IRM Optimization
Check Evidence of Contract

correct ?

Yes

Check Contract-Policy Compliance

match ?

Yes

Optimize Policy

No

Inlining ?

No

Run-time monitoring

Yes

Inline Policy

Execute
Check Evidence of Contract

Correct? Yes

Check Contract-Policy Compliance

Match? Yes

Optimize Policy

Inline Policy

Run-time monitoring

In-lining? Yes

Execute
Thesis Works

Check Evidence of Contract

- Correct?
  - Yes: Check Contract-Policy Compliance
  - No: Optimize Policy

Check Contract-Policy Compliance

- Match?
  - Yes: Optimize Policy
  - No: Run-time monitoring

Optimize Policy

- In-lining?
  - Yes: Inline Policy
  - No: Execute
Why not Security Automata?

- Class of Büchi automata accepting safety properties (recognizers) [Schneider-TISSec’00]
  - a countable set $Q$ of *automaton states*,
  - a countable set $I$ of *input symbols*
  - a *transition function* $\delta : (Q \times Q) \rightarrow 2^Q$, and
  - a countable set $Q_0 \subseteq Q$ of *initial automaton states*
• **It is rare, but it exists**
  
  – Example: A security requirement for banking applets
    
    • an application should use all the permissions it requires
    • to avoid over-entitlement which can be the source of potential (and possibly unknown) attacks
Infinite Transitions

Example of a Policy:
"After PIM is accessed only secure connections can be opened"
Example of a Policy:
"After PIM is accessed only secure connections can be opened"
Infinite Transitions

Example of a Policy:
"After PIM is accessed only secure connections can be opened"
Security Policies Enforcement Mechanisms

- Expressiveness
- Practical usage

High-accuracy mechanisms

2010-04-20
Expressiveness vs. Practical usage

- high-accuracy mechanisms
- light-weight mechanisms
Security Policies Enforcement
Mechanisms

Expressiveness

Practical usage

- High-accuracy mechanisms
- Flexible mechanisms
- Light-weight mechanisms

2010-04-20
Road Map

Security-by-Contract
Automata Modulo Theory
On-the-fly Matching
Simulation Matching
IRM Optimization
Automata Modulo Theory (AMT) as flexible mechanism

**AMT = Büchi automata + Satisfiability Modulo Theories**

Satisfiability Modulo Theories (SMT) [Sebastiani-JSAT’07]

- The problem of deciding the satisfiability of a first-order formula with respect to some decidable first-order theory $T$ (SMT($T$))
  - A $\Sigma$-theory is a set of first-order sentences with signature $\Sigma$

- Examples of theories of interest:
  - Equality and Uninterpreted Functions (EUF),
  - Linear Arithmetic (LA): both over the reals (LA($Q$)) and the integers (LA($Z$))
  - Combination of two or more theories $T_1,\ldots,T_n$.

- Examples of SMT tools:
  - Z3, MathSAT
• Let $A = < S, \Sigma, \mathcal{C}, \mathcal{E}, \Delta, s_0, F >$ be an AMT [MS-NordSec’07]
  – a finite set $S$ of automaton states,
  – a set $\mathcal{E}$ of formulae in the language of the $\Sigma$-Theory $\mathcal{C}$ as input symbols,
  – an initial state $s_0 \in S$,
  – a set $F \subseteq S$ of accepting states, and
  – a labeled transition relation $\Delta \subseteq S \times \mathcal{E} \times S$
Examples of AMT

Example of a Contract
"After PIM is opened no connections are allowed"
Examples of AMT

\[ \neg Jop \rightarrow s_0 \]

\[ \neg Joc(url) \rightarrow s_1 \]

\[ \text{Example of a Contract} \]
"After PIM is opened no connections are allowed"

\[ Joc(url) \triangleq Joc(joc,url) \]

\[ Jop \triangleq Jop(jop,x_1,\ldots,x_n) \]

\[ p(url) = type \triangleq url.startsWith(type) \]

\[ joc \triangleq \text{javax.microedition.io.Connector.open} \]

\[ jop \triangleq \text{javax.microedition.pim.PIM.openPIMList} \]
Examples of AMT

Example of a Contract
"After PIM is opened no connections are allowed"

Example of a Policy
"After PIM is accessed only secure connections can be opened"

\[
\begin{align*}
Joc(url) & \equiv Joc(joc,url) \\
Jop & \equiv Jop(jop,x_1,\ldots,x_n) \\
p(url) = type & \equiv url.startsWith(type) \\
joc & \equiv javax.microedition.io.Connector.open \\
jop & \equiv javax.microedition.pim.PIM.openPIMList
\end{align*}
\]
Examples of AMT

Example of a Contract
"After PIM is opened no connections are allowed"

Example of a Policy
"After PIM is accessed only secure connections can be opened"

\[
\begin{align*}
\text{Joc}(\text{url}) & \equiv \text{Joc}(\text{joc}, \text{url}) \\
\text{Jop} & \equiv \text{Jop}(\text{jop}, x_1, \ldots, x_n) \\
p(\text{url}) = \text{type} & \equiv \text{url.startsWith}(\text{type}) \\
\text{joc} & \equiv \text{javax.microedition.io.Connector.open} \\
\text{jop} & \equiv \text{javax.microedition.pim.PIM.openPIMList}
\end{align*}
\]
Symbolic Run in AMT

- Let $A = < S, \Sigma, \mathcal{T}, \mathcal{E}, \Delta, s_0, F>$ be an AMT
- A symbolic run of $A$ is a sequence of states alternating with expressions $\sigma = < q_0 e_1 q_1 e_2 q_2 ... >$:
  - $q_0 = s_0$
  - $(q_i, e_{i+1}, q_{i+1}) \in \Delta$ and $e_{i+1}$ is $\mathcal{T}$-satisfiable:
    - that is there exists some valuation $\nu$ over $\Sigma$ and $\mathcal{T}$ s.t. $\nu \models e_{i+1}$
    - valuation $\nu$ is a pair $(M, \alpha)$: $M$ a model of $\mathcal{T}$ and $\alpha$ an assignment
  - Finite symbolic run $\sigma = < q_0 e_1 q_1 e_2 q_2 ... q_n >$
  - Infinite symbolic run $\sigma = < q_0 e_1 q_1 e_2 q_2 ... >$

- Accepting symbolic run:
  - Finite run: $q_n \in F$
  - Infinite run: there exists some $k$ s.t. $q_k \in F$ and $q_k$ is visited infinitely often
Concrete Run in AMT

• Let $A = \langle S, \Sigma, \mathcal{T}, \mathcal{E}, \Delta, s_0, F \rangle$ be an AMT

• A concrete run of $A$ is a sequence of states alternating with valuations $\sigma = \langle q_0 v_1 q_1 v_2 q_2 \ldots \rangle$:
  
  - $q_0 = s_0$
  
  - there exists $e_{i+1} \in \mathcal{E}$:
    
    • $(q_i, e_{i+1}, q_{i+1}) \in \Delta$
    
    • there exists some valuation $v$ over $\Sigma$ and $\mathcal{T}$ s.t. $v \models e_{i+1}$

  
  - Finite concrete run $\sigma = \langle q_0 v_1 q_1 v_2 q_2 \ldots q_n \rangle$
  
  - Infinite concrete run $\sigma = \langle q_0 v_1 q_1 v_2 q_2 \ldots \rangle$

• Acceptance condition as symbolic run
Example of an Accepting Run in AMT

Symbolic Run

\[ t_0 \quad \text{Jop}(\text{jop},\text{file},\text{permission}) \quad t_1 \quad \text{Joc}(\text{joc},\text{url}) \wedge \text{p}(\text{url})=\text{"https"} \]

Concrete Run

\[ t_0 \quad (\text{jop},\text{PIM.CONTACT_LIST},\text{PIM.READ_WRITE}) \]
\[ t_1 \quad (\text{joc},\text{"https://www.esse3.unitn.it/"}) \]
\[ t_1 \quad (\text{jop},\text{PIM.CONTACT_LIST},\text{PIM.READ_ONLY}) \]
\[ t_1 \quad (\text{joc},\text{"https://online.unicreditbanca.it/login.htm"}) \]
Deterministic AMT

• \( A = \langle S, \Sigma, \mathcal{T}, \mathcal{E}, \Delta, s_0, F \rangle \) is a deterministic AMT
  – \( S, \Sigma, \mathcal{T}, \mathcal{E}, s_0, F \) as before
  – a labeled transition function \( \Delta \subseteq S \times \mathcal{E} \times S \):
    • for every \( s, s_1, s_2 \in S \) and every \( e_1, e_2 \in \mathcal{E} \)
    • if \( (s, e_1, s_1) \in \Delta \) and \( (s, e_2, s_2) \in \Delta \) where \( s_1 \neq s_2 \)
    • then \( (e_1 \land e_2) \) is unsatisfiable in the \( \Sigma \)-Theory \( \mathcal{T} \)

• Why determinism matters?
  – nondeterministic complementation is complex and exponential blow-up

• Why considering only the complementation of deterministic automata?
  – security policies are naturally deterministic
    • a platform owner should have a clear idea on what to allow or disallow
AMT Complementation and Intersection

• **Complementation:**
  – For each deterministic AMT automaton $A$ there exists a (possibly nondeterministic) AMT that accepts all the words which are not accepted by automaton $A$.

• **Intersection:** Let $<S^a, \Sigma^a, \mathcal{T}^a, \mathcal{E}^a, \Delta^a, s_0^a, F^a>$ and $<S^b, \Sigma^b, \mathcal{T}^b, \mathcal{E}^b, \Delta^b, s_0^b, F^b>$ be AMT, the *intersection* automaton $A = <S, \Sigma, \mathcal{T}, \mathcal{E}, \Delta, s_0, F>$:
    
    – $\Sigma = \Sigma^a \cup \Sigma^b$, $\mathcal{T} = \mathcal{T}^a \cup \mathcal{T}^b$, $\mathcal{E} = \mathcal{E}^a \cup \mathcal{E}^b$,
    
    – $S = S^a \times S^b \times \{1,2\}$, $s_0=(s_0^a, s_0^b, 1)$, $F = F^a \times S^b \times \{1\}$,
    
    – for every $s \in S$ and for every $e \in \mathcal{E}$:
      
      $\Delta = \{(s^a, s^b, x), (e^a \land e^b), (t^a, t^b, y) \mid (s^a, e^a, t^a) \in \Delta^a \text{ and } (s^b, e^b, t^b) \in \Delta^b \text{ and } \text{DecisionProcedure}(e^a \land e^b) = SAT \}$

    $y = \begin{cases} 
    2 & \text{if } x = 1 \text{ and } s^a \in F^a \text{ or if } x = 2 \text{ and } s^b \not\in F^b \\
    1 & \text{if } x = 1 \text{ and } s^a \not\in F^a \text{ or if } x = 2 \text{ and } s^b \in F^b 
      \end{cases}$
AMT Complementation and Intersection

- **Complementation:**
  - For each deterministic AMT automaton $A$ there exists a (possibly nondeterministic) AMT that accepts all the words which are not accepted by automaton $A$.

- **Intersection:** Let $< S^a, \Sigma^a, \mathcal{C}^a, \mathcal{E}^a, \Delta^a, s^a_0, F^a >$ and $< S^b, \Sigma^b, \mathcal{C}^b, \mathcal{E}^b, \Delta^b, s^b_0, F^b >$ be AMT, the intersection automaton $A = < S, \Sigma, \mathcal{C}, \mathcal{E}, \Delta, s^a_0, F >$:
  - $\Sigma = \Sigma^a \cup \Sigma^b$, $\mathcal{C} = \mathcal{C}^a \cup \mathcal{C}^b$, $\mathcal{E} = \mathcal{E}^a \cup \mathcal{E}^b$,
  - $S = S^a \times S^b \times \{1,2\}$, $s_0 = (s^a_0, s^b_0, 1)$, $F = F^a \times S^b \times \{1\}$,
  - for every $s \in S$ and for every $e \in \mathcal{E}$:
    
    $\Delta = \{(s^a, s^b, x), (e^a \land e^b), (t^a, t^b, y) | (s^a, e^a, t^a) \in \Delta^a \text{ and } (s^b, e^b, t^b) \in \Delta^b \text{ and } \text{DecisionProcedure}(e^a \land e^b) = \text{SAT}\}$

    $y = \begin{cases} 
    2 & \text{if } x = 1 \text{ and } s^a \in F^a \text{ or if } x = 2 \text{ and } s^b \notin F^b \\
    1 & \text{if } x = 1 \text{ and } s^a \notin F^a \text{ or if } x = 2 \text{ and } s^b \in F^b 
    \end{cases}$
AMT Intersection

(a) Example of Automata
(a) Example of Automata

AMT Intersection
(a) Example of Automata

(b) Boolean Abstraction
AMT Intersection

(a) Example of Automata

(b) Boolean Abstraction

(c) AMT Intersection
AMT Intersection

(a) Example of Automata

(b) Boolean Abstraction

(c) AMT Intersection

(d) Normal Intersection
So, What is Contract-Policy Compliance Check?

- **Security policies as AMT**
- **Matching:**
  - Language Inclusion:
    - Given two automata $A^c$ and $A^p$ representing respectively a contract and a policy, we have a match when the set execution traces of the $A^c$ is a subset of the set of acceptable traces of $A^p$.
  - Simulation:
    - every security-relevant action invoked by $A^c$ can also be invoked by $A^p$
Road Map

- Security-by-Contract
- Automata Modulo Theory
- On-the-fly Matching
- Simulation Matching
- IRM Optimization
• Matching between a contract with a security policy problem can be reduced to an emptiness test of the product automaton between a contract with a complement of policy.
• Matching between a contract with a security policy problem can be reduced to an emptiness test of the product automaton between a contract with a complement of policy.
• Matching between a contract with a security policy problem can be reduced to an emptiness test of the product automaton between a contract with a complement of policy.
• Matching between a contract with a security policy problem can be reduced to an emptiness test of the product automaton between a contract with a complement of policy.
Matching between a contract with a security policy problem can be reduced to an emptiness test of the product automaton between a contract with a complement of policy.
Contract-Policy Matching Algorithm

- **Input:** a contract and a complement policy
- **Output:** fail or succeed
- **Process:**
  - starts a depth first search procedure check\_safety from initial state
  - **IF** an accepting state in AMT is reached:
    - **IF** the state contains an error state of complemented policy **THEN** report a security policy violation without further ado
    - **IF** the state does not contain an error state of complemented policy **THEN** start a new depth first search check\_availability from the candidate state to determine whether it is in a cycle
    - **IF** cycle **THEN** report an availability violation
Proposition 4.1.

Let the theory $\mathcal{C}$ be decidable with an oracle for the SMT problem in the complexity class $C$ then:

The contract-policy matching problem for AMT using language inclusion is decidable in

- **time:** $\text{LIN} \sim \text{TIME}^C$
- **space:** $\text{NLOG} \sim \text{SPACE-complete}^C$
Contract-Policy Architecture

OFF-DEVICE

Policy Automaton → Complement Policy → Co-Policy Automaton

Contract Automaton

ON-DEVICE

Decision Procedure

Add Constraints
Solve
Remove Constraints

NuSMV library

Matching algorithm

Declare variables
OnTheFly emptiness check

match succeed/fail
Road Map

- Security-by-Contract
- Automata Modulo Theory
- On-the-fly Matching
- Simulation Matching
- IRM Optimization
Contract-Policy Matching

Policy Automaton

Contract Automaton

- **Matching = Simulation**
  - Every security-relevant action invoked by Contract can also be invoked by Policy

- **Compliance Game**
  - Concrete: Contract tries to make a concrete move and Policy follows accordingly to show that the Contract move is allowed
  - Symbolic: IF expression of Contract implies expression of Policy is VALID (modulo theory) THEN exists a move

  Adaptation of Jurdzinski’s algorithm on parity games (Jurdzinski 2000)
Contract-Policy Matching

- **Matching = Simulation**
  - Every security-relevant action invoked by Contract can also be invoked by Policy

- **Compliance Game**
  - Concrete: Contract tries to make a concrete move and Policy follows accordingly to show that the Contract move is allowed
  - Symbolic: IF expression of Contract implies expression of Policy is VALID (modulo theory) THEN exists a move
  - Adaptation of Jurdzinski’s algorithm on parity games (Jurdzinski 2000)
Contract-Policy Matching

- **Matching = Simulation**
  - Every security-relevant action invoked by Contract can also be invoked by Policy

- **Compliance Game**
  - Concrete: Contract tries to make a concrete move and Policy follows accordingly to show that the Contract move is allowed
  - Symbolic: IF expression of Contract implies expression of Policy is VALID (modulo theory) THEN exists a move
  - Adaptation of Jurdzinski’s algorithm on parity games (Jurdzinski 2000)
• **Matching = Simulation**
  – Every security-relevant action invoked by Contract can also be invoked by Policy

• **Compliance Game**
  – Concrete: Contract tries to make a concrete move and Policy follows accordingly to show that the Contract move is allowed
  – Symbolic: IF expression of Contract implies expression of Policy is VALID (modulo theory) THEN exists a move
  – Adaptation of Jurdzinski’s algorithm on parity games (Jurdzinski 2000)
Simulation as Compliance Game

• **Winner of the game:**
  – Policy cannot move: Contract wins.
  – Otherwise, two infinite concrete runs $s$ and $t$ resp. of Contract and Policy:
    • $s$ is an accepting concrete run and $t$ is not an accepting concrete run: Contract wins.
    • Other cases: Policy wins

• **Failure of Matching**
  – Policy cannot move => Contract is non-compliant
Symbolic vs Concrete Automaton

(a) Splitting Edges

\[(\text{Joc(url)} \land \text{p(url)} = "https") \rightarrow \text{s}_0 \quad \text{and} \quad (\text{Joc(url)} \land \text{p(url)} = "http") \rightarrow \text{s}_1\]

(b) Disjuncting Expressions

\[(\text{Joc(url)} \land \text{p(url)} = "https") \lor (\text{Joc(url)} \land \text{p(url)} = "http") \]

\[\text{s}_0 \rightarrow \text{s}_1\]
Symbolic vs Concrete Automaton

- **IF** A\(^c\) **complies with** A\(^p\) **THEN** A\(^c\) **concretely complies with** A\(^p\)
  - The converse does not hold in general.
  - Contrast to the simulation notions of (Hennessy and Lin 1995)
- **AMT fair simulation is stronger than AMT language inclusion**
Normalized AMT

• For every $q, q_1$ in set of states $S$ there is at most one expression $e_1$ in set of expressions $\mathcal{E}$ s.t. $(q, e_1, q_1)$ is in set of transitions $\Delta$
  – Example: from previous figure (a) is NOT normalized, (b) is normalized

• Normalization is possible when:
  – theory $\mathcal{T}$ is convex and closed under disjunction.

• Normalization preserves AMT determinism

• For normalized AMT: $A^c$ concretely complies with $A^p$ IFF $A^c$ complies with $A^p$
Simulation Policy-Contract Algorithm

- Matching between a contract with a security policy problem can be reduced to compliance game between a contract with a policy.
- Input: a contract and a policy
- Output: fail or succeed
- Process:
  - Create compliance game graph $G = \langle V, E, I \rangle$
  - $\mu(v) := 0$ for all $v \in V$
  - WHILE $\mu(v) \neq \mu_{\text{new}}(\mu, v)$ for some $v \in V$ DO
    - $\mu := \mu_{\text{new}}(\mu, v)$
  - IF $\mu(v(s_0^c, s_0^p)) < \infty$ THEN
    - succeed (Simulation exists)
Proposition 6.2.

Let the theory $\mathcal{C}$ be decidable with an oracle for the SMT problem in the complexity class $C$ then:

The contract-policy matching problem for AMT using fair simulation is decidable in

- time: $O(2 \cdot |E| \cdot |V_1|)$
- space: $O(|V|)$

- By Lemma 6.1.
  - $|V_1|$ is in $O(|S^c| \cdot |S^p|)$
  - $|V_0|$ is in $O(|S^c| \cdot |S^p| \cdot |\Delta^c|)$
  - $|E|$ is in $O(|S^c| \cdot |S^p| \cdot |\Delta^c|)^c$
Simulation Contract-Policy Architecture

- Decision Procedure
  - Add Constraints
  - Solve
  - Remove Constraints

- Matching algorithm
  - Declare variables
  - Construct game graph
  - Parity game simulation

- NuSMV library

- Contract Automaton

- Policy Automaton

- match succeed/fail
Matching Experiment

• Goal: proof-of-concept and deciding the best configuration of integrating matching algorithm with decision procedure
• Collected data: number of visited states, number of visited transitions, and running time for each problem in each design alternative
• Problem suite:
  – sample of policy-contract (mis)matching pairs
  – artificial problem to mimic large number of states
• Setup:
  – Desktop:
    • PC (Intel(R) Pentium D CPU 3.40GHz, 3389.442MHz, 1.99GB of RAM, 2048 KB cache)
    • On-the-fly: OS Linux version 2.6.20-16-generic, Kubuntu 7.04 (Feisty Fawn)
    • Simulation: Microsoft(R) Windows XP Professional Version 2002 Service Pack 3
  – Mobile device:
    • HTC P3600 (3G PDA phone) with ROM 128MB, RAM 64MB, 400MHz, Samsung(R) SC32442A
    • OS Microsoft(R) Windows Mobile 5.0 with Direct Push technology
On-the-fly Matching Experiment on Desktop

(a) Match succeeds for real policies

(b) Match fails for real policies
On-the-fly Matching Experiment on Desktop

(a) Match succeeds for real policies

(b) Match fails for real policies

(c) Matches among synthetic contracts and policies
On-the-fly Matching Experiment
Device vs Desktop

(a) Match succeeds
On-the-fly Matching Experiment
Device vs Desktop

(b) Match fails
Matching Experiment
Simulation vs On-the-fly on Desktop

(a) Match succeeds

(b) Match fails
IRM Optimization Models

Model 1: Contract Extractor on Trusted part

- Extract security relevant behaviors from code

![Diagram showing trusted and untrusted code with ContractExtractor and Contract nodes]
IRM Optimization Models

Model 1: Contract Extractor on Trusted part

Policy

extract security relevant behaviors from code

Contract

ContractExtractor

SimulationChecker

Code

Untrusted

Compliance Proof
IRM Optimization Models

Model 1: Contract Extractor on Trusted part

- **Policy**: Extract security relevant behaviors from code
- **Contract**: ContractExtractor
- **SimulationChecker**: Trusted
- **Optimizer**: Untrusted
- **Code**: Code
- **Compliance Proof**: Code
IRM Optimization Models

Model 1: Contract Extractor on Trusted part

- Policy
- Extract security relevant behaviors from code
- Contract
- ContractExtractor
- SimulationChecker
  - Yes
- Optimizer

Untrusted

Code

Compliance Proof

2010-04-20
UNITN - Siahaan
68
Model 1: Contract Extractor on Trusted part

- Extract security relevant behaviors from code

**Policy**

```
ContractExtractor
\downarrow
SimulationChecker
```

**Contract**

```
OptPolicy
```

**OptPolicy**

- Yes
- No

**Code**

Model 1: Contract Extractor on Trusted part
**IRM Optimization Models**

**Policy**
- extract security relevant behaviors from code
- check policy simulates contract

**Contract**
- ContractExtractor
- SimulationChecker
- Optimizer

**OptPolicy**
- Rewriter

**Code**
- Model1: Contract Extractor on Trusted part
extract security relevant behaviors from code

check policy simulates contract

discharge behaviors which are already enforced by code

Model 1: Contract Extractor on Trusted part

IRM Optimization Models

Trusted

Contract Extractor

SimulationChecker

OptPolicy

Rewriter

Execute

SafeCode

Untrusted

Code

OptPolicy

Policy

Execute

Compliance Proof

Optimized Policy
IRM Optimization Models

Model 1: Contract Extractor on Trusted part

Extract security relevant behaviors from code

Contract Extractor

Policy

SimulationChecker

Contract

Optimizer

OptPolicy

Rewriter

SafeCode

Execute

Trusted

Untrusted

Check policy simulates contract

Discharge behaviors which are already enforced by code

Inject policy to the code

2010-04-20
Optimizer and Rewriter on Untrusted part

Model 6: Contract Extractor on Untrusted part

Yes

No

OptPolicy

Rewriter

SafeCode

Execute

No

Yes

Policy

ClaimChecker

SimulationChecker

ClaimChecker

Execute

Yes

No

Contract

ContractExtractor

OptPolicy

Trust

Code

Yes

No

ClaimChecker

Reject

No

Yes

Execute
Optimizer and Rewriter on Untrusted part

Model 6: Contract Extractor on Untrusted part

- **ClaimChecker**
  - Policy: No → Reject
  - Yes → SimulChecker
  - SimulChecker:
    - Policy: No → Reject
    - Yes → Optimizer
    - Optimizer:
      - OptPolicy: Yes → Rewriter
      - No → SafeCode

- **ContractExtractor**
  - Contract
  - SafeCode
  - Execute

- **OptPolicy**
  - Yes → Rewriter
  - No → SafeCode

- **SafeCode**
  - Execute
Optimizer and Rewriter on Untrusted part

Model 6:
Contract Extractor on Untrusted part
Model 6: Contract Extractor on Untrusted part

- **Policy**:
  - No → ClaimChecker
  - Yes → SimulationChecker

- **Untrusted Code**
  - ContractExtractor

- **Optimizer**:
  - OptPolicy
  - SafeCode

- **ClaimChecker**
  - Yes → Optimizer
  - No → SafeCode

- **SimulationChecker**
  - No → Execute
  - Yes → Optimizer

- **Execute**
  - Yes → Optimizer
  - No → SafeCode

- **OptPolicy**
  - Yes → SafeCode
  - No → SafeCode

- **SafeCode**
  - Yes → Reject
  - No → SafeCode

- **Verify**:
  - Verify that the injected optimized policy complies to the code
Optimier and Rewriter on Untrusted part

Model 6: Contract Extractor on Untrusted part

- **Contract Extractor**
  - **Contract**
    - **Optimizer**
      - **OptPolicy**
        - **Rewriter**
          - **SafeCode**

- **Claim Checker**
  - **Policy**
    - **Yes**
      - **Simulation Checker**
        - **No**
          - **Execute**
            - **Claim Checker**
              - **No**
                - **Reject**
              - **Yes**
                - **Execute**
                  - **Claim Checker**
                    - **No**
                      - **Reject**
                    - **Yes**
                      - **OptPolicy**

- **Code**

- **OptPolicy**

- **SafeCode**

- **Verify that the injected optimized policy complies to the code**

2010-04-20
Optimizer and Rewriter on Untrusted part

Model 6: Contract Extractor on Untrusted part

Untrusted Code

ContractExtractor

Contract

OptPolicy

Rewriter

SafeCode

Yes

No

Yes

No

Yes

No

Execute

ClaimChecker

SimulationChecker

Policy

Yes

No

Yes

No

Yes

No

Execute

verify that the injected optimized policy complies to the code

ClaimChecker

ClaimChecker

Reject

Execute
Optimizing Security Policy or Rewriter

Security Policy → IRM Rewriter → Secured Application
Optimizing Security Policy or Rewriter

- **Security Automata SFI Implementation (SASI)** [Erlingson-etal-NSPW’99]
  - Minimizing TCB by working at the level of object code
- **Trade off between moving more processes out of trusted part and the complexity of the whole process** [Hamlen-Thesis’06]
- **Efficient IRM Enforcement** [Yan-etal-ASIACCS’09]
  - a constrained representation of history-based access control policies
  - exploit the structure of this policy representation
  - extended into a distributed optimization protocol
Optimizing Security Policy or Rewriter

- **Security Automata SFI Implementation (SASI) [Erlingson-et al-NSPW’99]**
  - Minimizing TCB by working at the level of object code
- **Trade off between moving more processes out of trusted part and the complexity of the whole process [Hamlen-Thesis’06]**
- **Efficient IRM Enforcement [Yan-et al-ASIACCS’09]**
  - A constrained representation of history-based access control policies
  - Exploit the structure of this policy representation
  - Extended into a distributed optimization protocol
Optimizing Security Policy or Rewriter

- **Security Automata SFI Implementation (SASI) [Erlingson-etal-NSPW’99]**
  - Minimizing TCB by working at the level of object code
- **Trade off between moving more processes out of trusted part and the complexity of the whole process [Hamlen-Thesis’06]**
- **Efficient IRM Enforcement [Yan-etal-ASIACCS’09]**
  - a constrained representation of history-based access control policies
  - exploit the structure of this policy representation
  - extended into a distributed optimization protocol
• Given two automata C and P representing resp. the formal specification of a contract and of a policy, we have an efficient IRM O derived from P with respect to C when:
  – every security-relevant event invoked by the intersection of O and C can also be invoked by P [sound]
  – O has smaller or equal number of transitions or states compared to P [optimal]
Searching an Optimized Policy

- Given two automata C and P representing resp. the formal specification of a contract and of a policy, we have an efficient IRM O derived from P with respect to C when:
  - every security-relevant event invoked by the intersection of O and C can also be invoked by P [sound]
  - O has smaller or equal number of transitions or states compared to P [optimal]
Searching an Optimized Policy

Given two automata C and P representing resp. the formal specification of a contract and of a policy, we have an efficient IRM O derived from P with respect to C when:

- every security-relevant event invoked by the intersection of O and C can also be invoked by P [sound]
- O has smaller or equal number of transitions or states compared to P [optimal]
### Inline Type Examples

<table>
<thead>
<tr>
<th>Inline-type</th>
<th>Contract</th>
<th>Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>C=P</td>
<td><img src="image" alt="Contract Diagram" /></td>
<td><img src="image" alt="Policy Diagram" /></td>
</tr>
</tbody>
</table>

#### Diagrams:

- **Contract Diagram**
  - Node C₀ connected to C₁ with edge labeled b.
  - Edge from a to C₀.
  - Edge from c to C₁.

- **Policy Diagram**
  - Node P₀ connected to P₁ with edge labeled b.
  - Edge from a to P₀.
  - Edge from c to P₁.
## Inline Type Examples

<table>
<thead>
<tr>
<th>Inline-type</th>
<th>Contract</th>
<th>Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>C=P</td>
<td><img src="image1" alt="Diagram" /></td>
<td><img src="image2" alt="Diagram" /></td>
</tr>
<tr>
<td>P=C</td>
<td><img src="image3" alt="Diagram" /></td>
<td><img src="image4" alt="Diagram" /></td>
</tr>
</tbody>
</table>
## Inline Type Examples

<table>
<thead>
<tr>
<th>Inline-type</th>
<th>Contract</th>
<th>Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="triangle.png" alt="Triangle" /> ( C=P )</td>
<td><img src="contract.png" alt="Contract Diagram" /></td>
<td><img src="policy.png" alt="Policy Diagram" /></td>
</tr>
<tr>
<td><img src="triangle.png" alt="Triangle" /> ( P=C )</td>
<td><img src="contract.png" alt="Contract Diagram" /></td>
<td><img src="policy.png" alt="Policy Diagram" /></td>
</tr>
</tbody>
</table>

Inline Type Examples

Inline nothing
### Inline Type Examples

<table>
<thead>
<tr>
<th>Inline-type</th>
<th>Contract</th>
<th>Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="C=P" /></td>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td><img src="image" alt="Triangle" /></td>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td><img src="image" alt="Triangle" /></td>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
</tbody>
</table>

### Inline nothing
## Inline Type Examples

<table>
<thead>
<tr>
<th>Inline-type</th>
<th>Contract</th>
<th>Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="C=P" /></td>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td><img src="image" alt="P C" /></td>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td><img src="image" alt="C P" /></td>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td><img src="image" alt="C P" /></td>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
</tbody>
</table>
## Inline Type Examples

<table>
<thead>
<tr>
<th>Inline-type</th>
<th>Contract</th>
<th>Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="C=P" alt="Triangle" /></td>
<td><img src="C_0_to_C_1" alt="Contract" /> <img src="P_0_to_P_1" alt="Policy" /></td>
<td><img src="C_0_to_C_1" alt="Contract" /> <img src="P_0_to_P_1" alt="Policy" /></td>
</tr>
<tr>
<td><img src="P=C" alt="Triangle" /></td>
<td><img src="C_0_to_C_1" alt="Contract" /> <img src="P_0_to_P_1" alt="Policy" /></td>
<td><img src="C_0_to_C_1" alt="Contract" /> <img src="P_0_to_P_1" alt="Policy" /></td>
</tr>
<tr>
<td><img src="C_P" alt="Two Triangles" /></td>
<td><img src="C_0_to_C_1" alt="Contract" /> <img src="P_0_to_P_1" alt="Policy" /></td>
<td><img src="C_0_to_C_1" alt="Contract" /> <img src="P_0_to_P_1" alt="Policy" /></td>
</tr>
<tr>
<td><img src="P_C" alt="Inverse Triangle" /></td>
<td><img src="C_0_to_C_1" alt="Contract" /> <img src="P_0_to_P_1" alt="Policy" /></td>
<td><img src="C_0_to_C_1" alt="Contract" /> <img src="P_0_to_P_1" alt="Policy" /></td>
</tr>
</tbody>
</table>
Optimization Example

<table>
<thead>
<tr>
<th>Inline-type</th>
<th>Contract</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="inline_type_contract.png" alt="Diagram" /></td>
<td><img src="contract_policy.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="policy_optimized_policy.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

Optimized Policy
 Optimization Example

| Inline-type | ![](image) |
| Contract | ![](image) |
| Policy | ![](image) |
| Optimized Policy | ![](image) |
Optimization Example

<table>
<thead>
<tr>
<th>Inline-type</th>
<th>Contract</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Diagram" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Optimized Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Diagram" /></td>
</tr>
</tbody>
</table>
Publications

Journals:

Conferences:
Publications

Workshops:

Conclusions

• Security policies of both safety and liveness properties
• Mechanism for defining a general security policies (not platform-specific)
• Mechanism for representing an infinite structure as a finite structure
• Goal:
  – to provide contract-policy matching
  – issues: small memory footprint, efficient computations
  – the tractability limit is the complexity of the satisfiability procedure for the background theories used to describe expressions
• Results:
  – Contract-policy matching problem for AMT using language inclusion and simulation
  – Policy optimization problem for AMT using fair simulation
Thank you
References

• J.R. Büchi, "On a decision method in restricted second-order arithmetic. ", Int. Cong. on Logic, Methodology and Philosophy of Science, 1962.
• F. Yan, P.W.L. Fong, “Efficient IRM Enforcement of History-Based Access Control Policies.”, ASIACCS 2009