Hiding Information in Completeness Holes

New perspectives in code obfuscation and watermarking

Roberto Giacobazzi

Dipartimento di Informatica
Università di Verona
Italy

SEFM’08, Cape Town November 2008
THE PROBLEM: PROTECTION

In SW much of the know-how is located in the product itself!

According to Business Software Alliance (BSA):

- the worldwide weighted average piracy rate is 35%, the median piracy rate is 62%, meaning half of the countries have a piracy rate of 62% or higher of the market, which grows to 75% in one-third of the countries
- In 2007, every 2.00USD worth of software purchased legitimately, 1.00USD worth was obtained illegally!!

knowledge extraction by static and dynamic analysis

program decomposition for code reuse

source code disassembly and decompilation for reverse engineering

integrity corruption for code hacking
THE PROBLEM: PROTECTION

We need adequate strategies for

- Intellectual Property Protection (IPP)
- Digital Right Management (DRM)
- Make difficult source code analysis
- Make difficult program decomposition, disassembly and decompilation
- Steganography (watermarking and fingerprinting) against theft
- Tamper proofing against integrity corruption
**THE PROBLEM: ATTACK**

Malware represents malicious software.

Malware detector is a program $D$ that determines whether another program $P$ is infected with a malware $M$.

$$D(P, M) = \begin{cases} 
\text{True} & \text{if } P \text{ is infected with } M \\
\text{False} & \text{otherwise} 
\end{cases}$$
Malware represents malicious software.

Malware detector is a program \( D \) that determines whether another program \( P \) is infected with a malware \( M \).

\[
D(P, M) = \begin{cases} 
  \text{True} & \text{if } P \text{ is infected with } M \\
  \text{False} & \text{otherwise}
\end{cases}
\]

An ideal malware detector detects all and only the programs infected with \( M \), i.e., it is sound and complete.

- **Sound** = no false positives (no false alarms)
- **Complete** = no false negatives (no missed alarms)
Malware Trends

There is more malware every year.

New Malware

- 2002: 445
- 2003
- 2004
- 2005: 10992
There is more malware every year.

But the number of malware families has almost no variation.

Beagle family has 197 variants (as on Jan. 2007).
Warezov family has 218 variants (as on Jan. 2007).
SW protection vs. SW attacks

- Host
- Attack
- SW
- Malicious SW

- SW
- Attack
- Host
- Malicious Host
SW protection vs. SW attacks

- host
- attack
- SW
  - malicious SW
  - viruses
  - worms
- SW
  - malicious host
- attack
- host
SW protection vs. SW attacks

- Malicious SW: viruses, worms
- Malicious host: IP integrity
- Attack on host
- Attack on SW
SW protection vs. SW attacks

misuse detection

SW

attack

host

malicious SW

SW

attack

host

malicious host
SW protection vs. SW attacks

misuse detection (syntactic)

SW
attack

malicious SW

code obfuscation

SW
attack

host

malicious host
SW PROTECTION VS. SW ATTACKS

- host
- attack
- malicious SW
- SW
- code obfuscation
- malicious host
- reverse engineering

misuse detection (syntactic)
SW protection vs. SW attacks

- Host
  - Misuse detection (syntactic)
  - Code obfuscation

- SW
  - Malicious SW
  - Code obfuscation

- Host
  - Malicious host
  - Reverse engineering (behaviour)
SW protection vs. SW attacks

- Host
  - Misuse detection (syntactic)
  - Deobfuscation

- SW
  - Code obfuscation

- Malicious SW
  - Code obfuscation

- Host
  - Malicious host
  - Reverse engineering (behaviour)
  - Deobfuscation
PROTECTION BY OBSCURITY: CODE OBFUSCATION

\[ \tau : P \rightarrow P \] is a code obfuscation if it is an obfuscating compiler:

- it is potent: \( \tau(P) \) is more complex (ideally unintelligible) than \( P \);
- it preserves the observational behaviour of programs \( \llbracket \tau(P) \rrbracket = \llbracket P \rrbracket \) [C. Collberg et al. ’97, ’98].
PROTECTION BY OBSCURITY: CODE OBFUSCATION

\( \tau : \mathcal{P} \rightarrow \mathcal{P} \) is a code obfuscation if it is an obfuscating compiler:

- **Potent**: \( \tau(P) \) is more complex (ideally unintelligible) than \( P \);

- It preserves the observational behaviour of programs \( \llbracket \tau(P) \rrbracket = \llbracket P \rrbracket \) [C. Collberg et al. ’97, ’98].

The limit. Obfuscating programs is (im)possible:

Even under restrictive hypothesis a general purpose obfuscator generating perfectly unintelligible code (virtual black-box) does not exist! [Barak et al. ’01].

The challenge. Design obfuscators that work against specific attacks

Extensional properties of programs are undecidable [Rice ’53].

....so formal methods and static analysis are born!
(Pseudo-)Code:

```
mov eax, [edx+0Ch]
push ebx
push [eax]
call ReleaseLock
```
(Pseudo-)Code:

```Assembly
mov eax, [edx+0Ch]
push ebx
push [eax]
call ReleaseLock
```

Obfuscated code (junk):

```Assembly
mov eax, [edx+0Ch]
inc eax
push ebx
dec eax
push [eax]
call ReleaseLock
```
**Example**

**(Pseudo-)Code:**

```
mov eax, [edx+0Ch]
push ebx
push [eax]
call ReleaseLock
```

**Obfuscated code (junk + reordering):**

```
mov eax, [edx+0Ch]
jmp +3
push ebx
dec eax
jmp +4
inc eax
jmp -3
call ReleaseLock
jmp +2
push [eax]
jmp -2
```
[Collberg et al. ’97, ’98]

- opaque predicate insertion
- code flattening,
- variable splitting,
- bogus code insertion,
- spurious aliases

**Potency measure by standard metrics:**

code size, number of predicates, number of methods in OO code, height of inheritance, and variable dependence length
State of the Art

[Wang et al. ’00]

- spurious aliases

Potency measure by complexity of static analysis

- 1-level aliasing is easy \( \mathbf{P} \) [Banning ’79]

- \( \geq \) 2-level aliasing is hard \( \mathbf{NP} \) [Horowitz ’97]

- with dynamic memory allocation is undecidable!!

understanding control-flow = solve a \( \geq \) 2-level aliasing problem
[Cloackware ’00]

**code flattening**

Potency is related with the PSPACE complexity of reachability in dispatchers
[Cloackware ’00]

[Diagram]

Potency is related with the PSPACE complexity of reachability in dispatchers
State of the Art

[Drape et al ’05 and ’07]

- data obfuscation
- slicing obfuscation: enlarging slices by adding dependencies

Potency is related with data-refinement

If $D$ is a data-type, $\mathcal{D}$ is a refinement of $D$ if $\langle \mathcal{D}, \alpha, \gamma, D \rangle$ is a GI

Correctness: $\llbracket P \rrbracket = \alpha \circ \llbracket \tau(P) \rrbracket \circ \gamma$

...i.e.: $P$ and $\gamma; \tau(P); \alpha$ are observationally equivalent!

Obfuscation corresponds precisely to concretise (in the sense of abstract interpretation) a data-type
THE PROBLEM: HIDING AND UNVEILING IN SW

Understanding programs corresponds to understand their semantics

✓ The attacker is an interpreter (static or dynamic)

Potency is related with the degree of precision of the interpreter

✓ \( \tau(P) \) is an obfuscation of \( P \) if the interpretation of \( \tau(P) \) fails (is less precise) than the same interpretation of \( P \): \( \llbracket P \rrbracket \leq \llbracket \tau(P) \rrbracket \)

✓ In this case \( \tau \) defeats \( \llbracket \cdot \rrbracket \)!!

We need a theory of interpreters at different levels of abstraction

We need Abstract Interpretation
THE PROBLEM: HIDING AND UNVEILING IN SW

Input

Deobfuscation

Output

Reverse Engineering

α

malicious user

δ

SW
Abstract Interpretation (1977) is the most general model for the (static or dynamic) approximation of semantics of discrete dynamic systems

Including: Static program analysis, type checking and type inference, model checking and predicate abstraction, trajectory evaluation, testing, proof systems, etc.
Design approximate semantics of programs [Cousot & Cousot '77, ’79].

**Galois Connection:** \( \langle C, \alpha, \gamma, A \rangle \), \( A \) and \( C \) are complete lattices.

\( \langle uco(C), \sqsubseteq \rangle \) set of all possible abstract domains,

\( A_1 \sqsubseteq A_2 \) if \( A_1 \) is more concrete than \( A_2 \)
A program $P$

A domain of computation for $P$: $C$ typically a complete lattice

Semantic specification (interpreter): $\llbracket P \rrbracket : C \rightarrow C$

(Approximate) observable properties: $\rho \in uco(C)$

**Derive a sound approximate specification $\llbracket P \rrbracket ^\#$**

$$\rho(\llbracket P \rrbracket(x)) \leq \llbracket P \rrbracket ^\#(x)$$

**The limit case: Completeness**

$$\rho(\llbracket P \rrbracket(x)) = \llbracket P \rrbracket ^\#(x) \text{ iff } \rho(\llbracket P \rrbracket(x)) = \rho(\llbracket P \rrbracket(\rho(x)))$$
**Completeness in Abstract Interpretation**

**Backward Soundness:** No information is lost by approximating the input/output

\[ \rho \circ f \preceq \rho \circ f \circ \rho \]
Completeness in Abstract Interpretation

Backward Completeness: no loss of precision is accumulated by approximating the input

\[ \rho \circ f = \rho \circ f \circ \rho \]
**Completeness in Abstract Interpretation**

- **Forward Completeness**: no information is lost by approximating the output

\[ f \circ \rho \leq \rho \circ f \circ \rho \]
Completeness in Abstract Interpretation

Forward Completeness: no information is lost by approximating the output

\[ f \circ \rho = \rho \circ f \circ \rho \]

\[ f(x) \rightarrow f(\rho(x)) = \rho(f(\rho(x))) \rightarrow f^\sharp(\rho(x)) \]

\[ \rho \]

\[ f \]

\[ \rho \]

\[ f \]
A simple example in Interval analysis

A simple domain of intervals
A simple example in Interval analysis

A simple domain of intervals

\[ sq(X) = \left\{ x^2 \mid x \in X \right\} \]

\{\mathbb{Z}, [0, +\infty], [0, 10]\} is Forward but not Backward complete
A simple example in Interval analysis

A simple domain of intervals

$\text{sq}(X) = \{ x^2 \mid x \in X \}$

$\{\mathbb{Z}, [0, +\infty], [0, 10]\}$ is Forward but not Backward complete

$\{\mathbb{Z}, [0, 2], [0, 0]\}$ is Backward but not Forward complete
Obfuscation by incompleteness

Failing precision means failing completeness!

*Obfuscating programs is making abstract interpreters incomplete*

Let \( \rho \in \text{uco}(\Sigma) \) with \( \Sigma \) semantic objects (data, traces etc)

A program transformation \( \tau : \mathbb{P} \rightarrow \mathbb{P} : [P] = [\tau(P)] \).

\( \rho \) \( \mathcal{B} \)-complete for \([\cdot] : \rho([P]) = [P]^\rho \)

\( \tau \) obfuscates \( P \) if

\[ [P]^\rho \subseteq [\tau(P)]^\rho \iff \rho([\tau(P)]) \subseteq [\tau(P)]^\rho \]
OBSURITY BY INCOMPLETENESS

Failing precision means failing completeness!

*Obfuscating programs is making abstract interpreters incomplete*

\[ C : x = a \ast b \]

*Sign* is an abstraction of \( \wp(\mathbb{Z}) \):

\[
\begin{align*}
\wp(\mathbb{Z}) & \\
\ldots & \\
0^- & 0^+ \\
\ldots & \\
\{-1, -3, -4\} & \ldots & \{2, 3, 5\} \\
\ldots & \\
0 & 1 & \\
\ldots & \\
\emptyset & \\
\end{align*}
\]
Obscurity by incompleteness

Failing precision means failing completeness!

*Obfuscating programs is making abstract interpreters incomplete*

\[ C : x = a \ast b \]

*Sign* is an abstraction of \( \varphi(\mathbb{Z}) \):

\[ \varphi(\mathbb{Z}) \]

\[ \{ -1, -3, -4 \} \quad \ldots \quad \{ 2, 3, 5 \} \]

\[ \emptyset \quad 0 \quad 1 \quad \ldots \]

\[ 0^- \quad 0^+ \]

\[ 0 \quad 0 \]

\[ \emptyset \]

\[ \emptyset \quad 0 \quad 0^+ \]

\[ 0^- \quad 0^+ \]

\[ \emptyset \]

SEFM’08 – Cape Town – p. 16/37
Failing precision means failing completeness!

*Obfuscating programs is making abstract interpreters incomplete*

\[ x = 0; \]

\[ C : \quad x = a \times b \quad \rightarrow \quad \tau(C) : \quad \text{if } b \leq 0 \text{ then } \{ a = -a; b = -b \}; \]

\[ \text{while } b \neq 0 \{ x = a + x; b = b - 1 \} \]

\[ \text{Sign} \text{ is complete for } C \]

\[ \\checkmark \quad [C]^{\text{Sign}} = \lambda a, b. \ \text{Sign}(a \times b) \]

\[ \text{Sign} \text{ is incomplete for } \tau(C) \]

\[ \\checkmark \quad [\tau(C)]^{\text{Sign}} = \lambda a, b. \begin{cases} 0 & \text{if } a = 0 \lor b = 0 \\ \wp(\mathbb{Z}) & \text{otherwise} \end{cases} \]
We consider variable splitting

\[ v \in \text{Var}(P) \text{ is split into } \langle v_1, v_2 \rangle \text{ such that } \]

\[ v_1 = f_1(v), \; v_2 = f_2(v) \text{ and } v = g(v_1, v_2) \]

\[ f_1(v) = v \div 10 \]
\[ f_2(v) = v \mod 10 \]
\[ g(v_1, v_2) = 10 \cdot v_1 + v_2 \]

And the interval analysis: \( v(x) = [\min(x), \max(x)] \)

\[
P: \begin{cases} 
  v = 0; \\
  \text{while } v < N \{ v ++ \} 
\end{cases}
\]

\[ [P]^t = \lambda v. [0, N] \]
We consider variable splitting

\[ v \in \text{Var}(P) \text{ is split into } \langle v_1, v_2 \rangle \text{ such that } v_1 = f_1(v), \ v_2 = f_2(v) \text{ and } v = g(v_1, v_2) \]

\[ f_1(v) = v \div 10 \]
\[ f_2(v) = v \mod 10 \]
\[ g(v_1, v_2) = 10 \cdot v_1 + v_2 \]

And the interval analysis: \( \iota(x) = [\min(x), \max(x)] \)

\[ \tau(P) : \begin{cases} v_1 = 0; \\ v_2 = 0; \end{cases} \begin{cases} \textbf{while } 10 \cdot v_1 + v_2 < N \{ \\ v_1 = v_1 + (v_2 + 1) \div 10 \\ v_2 = (v_2 + 1) \mod 10 \} ; \end{cases} \[
\begin{align*}
\llbracket \tau(P); c \rrbracket^\iota = \lambda v. 10 \odot [0, \frac{N \Theta [0, 9]}{10}] \oplus [0, 9] = \\
\lambda v. [0, N] \oplus [0, 9] = \\
\lambda v. [0, N + 9]
\end{align*}
\]
\[ c : v = 10 \cdot v_1 + v_2 \]
We consider array splitting for weakening the invariant of Fibonacci’s
\[
\text{Inv} = 2 \leq i \leq N \land \forall j \in [2, i]. \ a[j] = a[j - 1] + a[j - 2]
\]

The invariant \text{Inv} can be generated by relational interval-\text{Fib} analysis

\[
\eta = \alpha^+ \circ \alpha \text{ where}
\]

\[
\alpha(X) = \begin{cases} 
\text{Fib} & \text{if } \forall \langle S, x \rangle \in X. \ S \subseteq D_x \land (S = \{0\} \land x[0] = 0) \lor \\
& (S = \{0, 1\} \land x[0] = 0 \land x[1] = 1) \lor \\
& (\forall j \in S. \ x[j] = x[j - 1] + x[j - 2]) \\
\text{Any} & \text{otherwise}
\end{cases}
\]

\[
I \longrightarrow \text{Fib} \text{ represents Fibonacci’s sequences until } \max(I)
\]

\[
I \longrightarrow \text{Any} \text{ represents any array with domain including } I \text{ (no overflow)}
\]

\[
[n, m] \longrightarrow \text{Fib} = [n, m - 1] \longrightarrow \text{Fib} \oplus [n, m - 2] \longrightarrow \text{Fib}
\]
We consider array splitting for weakening the invariant of Fibonacci’s

\[ \text{Inv} = 2 \leq i \leq N \land \forall j \in [2, i]. \ a[j] = a[j - 1] + a[j - 2] \]

\[
\begin{align*}
& \quad a[0] = 0; \\
& \quad a[1] = 1; \\
& \quad i = 2;
\end{align*}
\]

\[ P : \quad \text{while } i \leq N \{ \\
& \quad a[i] = a[i - 1] + a[i - 2]; \\
& \quad i++; \\
\} \]

\[
[P]^u \xrightarrow{n} = a \in [0, N] \longrightarrow \text{Fib} \land i \in [2, N + 1]
\]
We consider array splitting for weakening the invariant of Fibonacci's

\[ \text{Inv} = 2 \leq i \leq N \land \forall j \in [2, i]. \ a[j] = a[j - 1] + a[j - 2] \]

\[
\begin{align*}
\tau(P) : \\
& b[0] = 0; \\
& c[0] = 1; \\
& i = 2; \\
& \text{while } i \leq N \{ \\
& \quad \text{if } i \mod 2 == 0 \\
& \quad \quad \{ b[i \div 2] = c[(i - 1) \div 2] + b[(i - 2) \div 2] \} \\
& \quad \quad \{ c[i \div 2] = b[(i) \div 2] + c[(i - 2) \div 2] \}; \\
& \quad i += 0 \\
& \} \\
\end{align*}
\]

\[ [\tau(P)]^{u \rightarrow \eta} = b, c \in [0, N \div 2] \rightarrow \text{Any} \land i \in [2, N + 1] \]
We consider array splitting for weakening the invariant of Fibonacci’s
\[ \text{Inv} = 2 \leq i \leq N \land \forall j \in [2, i]. \ a[j] = a[j - 1] + a[j - 2] \]

How can we attack \( \tau(P) \) and get \( \text{Inv} \) back?
THE GEOMETRY OF ATTACKERS

Abstract

\( X \)

\( \mathcal{R}(X) \)

Concrete

Ico – REFINEMENT
THE GEOMETRY OF ATTACKERS

Concrete

Abstract

$S(X)$

$X$

uco – SIMPLIFICATION
Let $P$ be completeness.
Let $p$ be completeness
DOMAIN COMPLETENESS: SHELL/ CORE

BACKWARD COMPLETENESS: $\eta \circ f \circ \rho = \eta \circ f$
DOMAIN COMPLETENESS: SHELL/CORE

BACKWARD IN-COMPLETENESS: \( \eta \circ f \circ \rho \geq \eta \circ f \)
Domain Completeness: Shell/Core

Making BACKWARD COMPLETE: Refining input domains [GRS’00]
Making BACKWARD COMPLETE: Simplifying output domains [GRS’00]
Domain Completeness: Shell/Core

Forward completeness: $\eta \circ f \circ \rho = f \circ \rho$
Domain Completeness: Shell/Core

Forward in-completeness: $\eta \circ f \circ \rho \geq f \circ \rho$
**Making FORWARD COMPLETE: Refining output domains [GQ'01]**
Making FORWARD COMPLETE: Simplifying input domains [GQ’01]
A domain is *backward complete* wrt $f$ iff it is *forward complete* wrt $f^+ = \lambda X. \bigcup \{ Y \mid f(Y) \subseteq X \}$;

A (not trivial) partition is *backward stable* wrt $f$ iff it is *forward stable* wrt $f^{-1} = \lambda X. \{ y \mid f(y) \in X \}$;

If $f$ is injective, a (not trivial) partition is *forward stable* wrt $f$ iff it is *backward stable* wrt $f^{-1}$;
A domain is **backward complete** wrt \( f \) iff it is **forward complete** wrt \( f^+ = \lambda X. \bigcup \{ Y \mid f(Y) \subseteq X \} \);

A (not trivial) partition is **backward stable** wrt \( f \) iff it is **forward stable** wrt \( f^{-1} = \lambda X. \{ y \mid f(y) \in X \} \);

If \( f \) is **injective**, a (not trivial) partition is **forward stable** wrt \( f \) iff it is **backward stable** wrt \( f^{-1} \);

A **backward** problem can always be transformed in a **forward** one, but the viceversa is not always possible!
The complete shell $S = \mathcal{R}^B_{[\tau(P)]}(u \rightarrow \eta)$ includes odd and even Fibonacci's sequences:

\[
[\tau(P)]^S = b \in [0, N \div 2] \rightarrow \text{eFib} \land c \in [0, N \div 2] \rightarrow \text{oFib} \land i \in [2, N + 1]
\]

Inv = $2 \leq i \leq N \land \forall j \in [2, i]. a[j] = a[j - 1] + a[j - 2]$
Can we make SW obscure by transforming semantics?
Program Transformation

Subject program $P$   Syntactic transformation $\tau$   Transformed program $\tau[P]$

Subject program semantics $S[P]$   Semantic transformation $t$   Transformed program semantics $t[S[P]] \subseteq S[\tau[P]]$

Syntactic transformation: $\tau = p \circ t \circ S$
THE GEOMETRY OF SEMANTICS TRANSFORMERS

Making semantics complete (from above and below):

\[ F_{\eta, \rho}^\uparrow(f) = \bigcap \{ h : C \to C \mid f \sqsubseteq h, \ \rho \circ h \circ \eta = h \circ \eta \} \]

\[ F_{\eta, \rho}^\downarrow(f) = \bigcup \{ h : C \to C \mid f \sqsupseteq h, \ \rho \circ h \circ \eta = h \circ \eta \} \]

\( F_{\eta, \rho}^\uparrow(f) \) and \( F_{\eta, \rho}^\downarrow(f) \) are (Forward) complete

Making semantics maximally in-complete (from above and below):

\[ O_{\eta, \rho}^\uparrow(f) = \bigcup \{ g : C \to C \mid F_{\eta, \rho}^\downarrow(g) = F_{\eta, \rho}^\downarrow(f) \} \]

\[ O_{\eta, \rho}^\downarrow(f) = \bigcap \{ g : C \to C \mid F_{\eta, \rho}^\uparrow(g) = F_{\eta, \rho}^\uparrow(f) \} \]

\( O_{\eta, \rho}^\uparrow(f) \) and \( O_{\eta, \rho}^\downarrow(f) \) are generally in-complete
THE GEOMETRY OF SEMANTICS TRANSFORMERS

Minimal complete transformation from above

Minimal complete transformation from below

Maximal incomplete transformation from below

Maximal incomplete transformation from above

\[(F^\uparrow)^+ = F^\downarrow\]

and

\[(F^\uparrow)^- = O^\downarrow\]
Making FORWARD COMPLETENESS: Transforming the semantics upwards

$$\mathbb{F}^{\uparrow}_{\eta, \rho} = \lambda f. \lambda x. \begin{cases} 
\rho \circ f(x) & \text{if } x \in \eta(C) \\
 f(x) & \text{otherwise} 
\end{cases}$$
The Geometry of Semantics Transformers

Making **Forward Completeness:** Transforming the semantics downwards

\[
\begin{align*}
F_{\eta,\rho}^\downarrow &= \lambda f . \lambda x. \begin{cases} 
\rho^+ \circ f(x) & \text{if } x \in \eta(C) \\
 f(x) & \text{otherwise}
\end{cases} 
\end{align*}
\]
Making **FORWARD IN-COMPLETENESS**: Transforming the semantics upwards

\[
\mathcal{O}^\uparrow_{\eta, \rho}(f)(x) = \begin{cases} 
(\rho^+)^+(f(x)) = \vee \{ y \mid \rho^+(y) = \rho^+(f(x)) \} & \text{if } x \in \eta \\
\rho^+(f(x)) & \text{otherwise}
\end{cases}
\]
Making **FORWARD IN-COMPLETENESS**: Transforming the semantics downwards

\[
\Diamond_{\eta, \rho} (f)(x) = \left\{ \begin{array}{ll}
\rho^{-1}(f(x)) = \bigwedge \{ y \mid \rho(y) = \rho(f(x)) \} & \text{if } x \in \eta \\
 f(x) & \text{otherwise}
\end{array} \right.
\]
We transform semantics in order to induce maximal incompleteness

\[
P : \begin{cases}
  x = x \ast x; \\
  c : \text{if } 10 \leq x \leq 100 \{ y = 5 \} \{ y = 5000 \}; \\
  \text{return} (y)
\end{cases}
\]

\[\llbracket P \rrbracket^i (x \in [5, 8]) = x \in [25, 64] \land y \in [5]\]

\[\text{wlp} \llbracket c \rrbracket^i (y \leq 100) = x \in [10, 100] \land y \in [5]\]

\[\text{wlp} \llbracket x = x * x \rrbracket^i (x \in [10, 100]) = x \in [4, 10].\]

Find \( c' \) such that

\[\text{wlp} \llbracket c' \rrbracket^i (x \in [10, 100]) =
\]

\[\bigotimes^i (\lambda X. \text{wlp} \llbracket x = x \ast x \rrbracket^i (X)) (x \in [10, 100]) =
\]

\[\iota^{-1} (\text{wlp} \llbracket x = x \ast x \rrbracket^i (x \in [10, 100])) = \{4, 10\}.\]
OBfuscation as incompleteness

We transform semantics in order to induce maximal incompleteness

\[
P : \quad \begin{align*}
    x &= x \times x; \\
    c & : \quad \text{if } 10 \leq x \leq 100 \{ y = 5 \} \{ y = 5000 \}; \\
    \text{return}(y)
\end{align*}
\]

\[
c' : \text{if } x == 4 \lor x == 10 \{ x = 16 \} \{ x = x \times 200 \}
\]

In order to ensure behaviour equivalence we derive

\[
\text{if } 4 \leq x \leq 10 \\
\{ x = x - (x - 4) \} \quad \square x = x - (x - 10) \} \\
\{ \text{nil} \}
\]
**Obfuscation as Incompleteness**

We transform semantics in order to induce maximal incompleteness.

\[
P : \begin{cases} 
  x = x \cdot x; \\
  c : \begin{cases} 
    \text{if } 10 \leq x \leq 100 \{ y = 5 \} \{ y = 5000 \}; \\
    \text{return} (y) 
  \end{cases} 
\end{cases}
\]

The resulting obfuscated code is:

\[
\tau(P) : \begin{cases} 
  \text{if } 4 \leq x \leq 10 \\
  \{ x = x - (x - 4) \} \{ x = x - (x - 10) \} \\
  \{ \text{nil} \}; \\
  \text{if } x == 4 \lor x == 10 \{ x = 16 \} \{ x = x \cdot 200 \}; \\
  \text{if } 10 \leq x \leq 100 \{ y = 5 \} \{ y = 5000 \}; \\
  \text{return} (y) 
\end{cases}
\]

For \( x = 7 \) we have

\[
[[\tau(P)](x \in [5, 8])] = x \in [16, 1400] \land y \in [5, 5000]
\]
We transform semantics in order to induce maximal incompleteness

\[ P : \begin{align*}
& x = x \times x; \\
& c : \text{if } 10 \leq x \leq 100 \{ y = 5 \} \{ y = 5000 \}; \\
& \text{return}(y)
\end{align*} \]

The resulting obfuscated code is:

\[ \tau(P) : \begin{align*}
& \text{if } 4 \leq [5, 8] \leq 10 \\
& \{ x = [5, 8] - ([5, 8] - 4) \square x = x - (x - 10) \} \\
& \{ \text{nil} \}; \\
& \{ x \in [1, 7] \} \\
& \text{if } x == 4 \lor x == 10 \{ x = 16 \} \{ x = x \times 200 \}; \\
& \text{if } 10 \leq x \leq 100 \{ y = 5 \} \{ y = 5000 \}; \\
& \text{return}(y)
\end{align*} \]

For \( x = 7 \) we have

\[ [\tau(P)]^t(x \in [5, 8]) = x \in [16, 1400] \land y \in [5, 5000] \]
Obfuscation as Incompleteness

We can derive a method for systematically making code obscure:

\[ P = M_1; \ldots; M_j; \Phi_j M_{j+1}; \ldots; M_n \]

Assume the invariant \( \Phi_j \) can be generated with abstract interpretation \( \alpha \)

Find \( C \) such that:

\[
\text{wlp}[C]^{\alpha}(\Phi_j) = \Phi_j M_j + 1; \ldots; M_n
\]

Adjust \( C \) in order to keep concrete observational (I/O) behaviour \( (C \models \Phi_j) \)

\[
\tau(P) = M_1; \ldots; C; \Phi_j M_{j+1}; \ldots; M_n
\]
We generalize Cousot’ Abstract Watermarking [Cousot & Cousot ’04]

- **Stegomarker**: $M : S \rightarrow P$ encodes the signature $s \in S$ into a program $M(s) \in P$ (the stegomark)

- **Stegolayer**: $L : P \times P \rightarrow P$ is used to compose the stegomark with the source (cover) program.

- **Stegoprogram**: $G : P \times S \rightarrow P$ such that $G(P, s) = L(P, M(s))$

**Static Watermarking**

Watermarks are encoded as syntactic (static) properties of $G(P, s)$
HIDING IN OBScurity

We generalize Cousot’ Abstract Watermarking [Cousot & Cousot ’04]

- **Stegomarker**: $M : S \rightarrow P$ encodes the signature $s \in S$ into a program $M(s) \in P$ (the stegomark)

- **Stegolayer**: $L : P \times P \rightarrow P$ is used to compose the stegomark with the source (cover) program.

- **Stegoprogram**: $G : P \times S \rightarrow P$ such that $G(P, s) = L(P, M(s))$

**Dynamic Watermarking**

Watermarks are encoded as semantic (dynamic) properties of $G(P, s)$
We generalize Cousot’ Abstract Watermarking [Cousot & Cousot ’04]

- **Stegomarker**: $\mathcal{M} : S \rightarrow \mathbb{P}$ encodes the signature $s \in S$ into a program $\mathcal{M}(s) \in \mathbb{P}$ (the stegomark)

- **Stegolayer**: $\mathcal{L} : \mathbb{P} \times \mathbb{P} \rightarrow \mathbb{P}$ is used to compose the stegomark with the source (cover) program.

- **Stegoprogram**: $\mathcal{G} : \mathbb{P} \times S \rightarrow \mathbb{P}$ such that $\mathcal{G}(P, s) = \mathcal{L}(P, \mathcal{M}(s))$

**Abstract Watermarking**

Watermarks are encoded as abstract properties of $\mathcal{G}(P, s)$
Static and dynamic are instances of Abstract Watermarking!

Let $P \in \mathbb{P}$ (source), $\alpha, \omega, \eta \in \text{uco}(\Sigma)$ be program properties such that $\alpha \subseteq \omega$.

If $\{\mathcal{M}(s)\}^\alpha \in \omega$ then $\mathcal{L}$ is a stegolayer for $P$ and $\mathcal{M}(s)$ if

$$\{\mathcal{L}(P, \mathcal{M}(s))\}^\alpha = \lambda x. \begin{cases} \{\mathcal{M}(s)\}^\alpha(x) & \text{if } x \in \eta \\ \{P\}^\alpha(x) & \text{otherwise} \end{cases}$$

**Static Watermarking**

$\alpha$ decidable (static) and $\eta = id$

$\mathcal{S}(P, s)$ always reveals the watermark.
Hiding in Obscurity

Static and dynamic are instances of Abstract Watermarking!

\[ P \in \mathcal{P} \text{ (source), } \alpha, \omega, \eta \in uco(\Sigma) \text{ be program properties such that } \alpha \subseteq \omega \]

If \( \{\mathcal{M}(s)\}^\alpha \in \omega \) then \( \mathcal{L} \) is a stegolayer for \( P \) and \( \mathcal{M}(s) \) if

\[
\{\mathcal{L}(P, \mathcal{M}(s))\}^\alpha = \lambda x. \begin{cases} \{\mathcal{M}(s)\}^\alpha(x) & \text{if } x \in \eta \\ \{P\}^\alpha(x) & \text{otherwise} \end{cases}
\]

Dynamic Watermarking

\( \alpha \) generic interpreter (dynamic) and \( \eta \neq id \)

\[ \Downarrow \]

\( \mathcal{G}(P, s) \) reveals the watermark only on input \( \eta \)
A stegoprogram reveals the watermark $\omega$ under input $\eta$ if its abstract semantics is $\mathcal{F}$-complete for $\omega$ and $\eta$.

$\mathcal{S}(s, P)$ is a stegoprogram if:

$$\{ \mathcal{S}(s, P) \}_{\alpha}^\uparrow \downarrow_{\eta, \mathfrak{M}[s]} (\{ P \}_{\alpha})$$

$\{ \cdot \}_{\alpha}^\uparrow \downarrow$ performs watermark extraction (an abstract interpretation)

Credibility: $\{ P \}_{\alpha}^\uparrow \not\in \omega$ (i.e., $\omega(\{ P \}_{\alpha}) \approx \top$)

Resilience: $\alpha$ is preserved by most program transformations

Stealthy: $\alpha$ hard to guess + good stegolayer
Static watermarking ($\eta = id$) with traces in $\Sigma^+$ as semantic objects

\[ \mathcal{E} : \mathbb{N} \rightarrow \mathcal{G} \] encoding of numbers in graphs

\[ \mathcal{M}\{s\} \] is the atomic closure $\{\mathcal{G}_s, \Sigma^+\} \in uco(\Sigma^+) \] where

\[ \mathcal{G}_s = \left\{ \sigma \in \Sigma^+ \mid \mathcal{E}(s) = \text{CFG}(\sigma) \right\} \]

\[ \|P\|^\alpha \] extracts the CFG of $P$, which is an (incomplete) abstract interpretation of the trace semantics $\|P\|

\[ 2 \cdot 5^3 + 0 \cdot 5^2 + 1 \cdot 5^1 + 4 \cdot 5^0 = 259 \]
Dynamic watermarking ($\eta \neq \text{id}$) states $\langle c, \mathcal{R}, \mathcal{H}, i \rangle$, where $\mathcal{H} \in \mathbb{H}$ is a heap, $c$ is the current instruction, $i$ is an input sequence, and $\mathcal{R} : \text{Var}(P) \rightarrow \mathcal{R}$ is register allocation.

- $\mathcal{E} : \mathbb{N} \rightarrow \mathcal{G}$ encoding of numbers in graphs

- $\mathcal{M}[s]$ is the atomic closure $\{\mathcal{E}(s), \Sigma^+\} \in \text{uco}(\Sigma^+)$

- $\mathcal{H} : \mathbb{H} \rightarrow \mathcal{G}$ extracts the set of all graphs allocated in memory with root allocated as last,

- $\alpha = \delta^+ \circ \delta$ where $\delta : \varphi(\Sigma^+) \rightarrow \mathcal{G}$ is such that:

$$
\delta(X) = \left\{ \begin{array}{l}
\mathcal{G} \in \mathcal{H}(\mathcal{H}_n), \ \text{root}(\mathcal{G}) \notin \mathcal{H}_j \\
\forall j \in [0, n - 1]. \ \text{root}(\mathcal{G}) \notin \mathcal{H}_j \\
\end{array} \right.
$$
**Discussion: The FUCSIA idea**

Obfuscation and Steganography by Abstract Interpretation

Define a uniform framework for information concealment in programming languages

- **General** enough to include most known methods
- **Formal** enough to provide a (possibly) provable secure environment for obfuscation and steganography
- **Rich** enough to provide advanced design and evaluation tools
- **Practical** enough to become a standard in the obfuscation and steganographic design and evaluation

The goal: develop a theory and practice for code obfuscation and steganography in order to make these technologies as practical as analogous ones in other media (e.g., in DRM of audio and video)

- The code is a new media
- Known concepts in digital media (compression, noise etc.) have to be studied on software
FUTURE DIRECTIONS

- Move from syntactic to semantic-based metrics
  - measuring incompleteness
  - measuring complexity of complete refinements

- Obscuring and watermarking require program integration $f: P \times P \rightarrow P$

- Explore (HO)ANI for isolating completeness holes?
  - The obfuscated parts and the stegomarks have to preserve the semantics of the cover program when integrated
  - $P$ is partitioned in
    - cover programs $P \subseteq P$
    - secret programs $Q \subseteq P$
HOANI FOR SW WATERMARKING?

\[(\eta) \mathcal{I}(\phi \sim \rho) : [P_1]^{\eta} = [P_2]^{\eta} \implies \rho^\rho([\mathcal{I}]^\phi, [Q_1]^{\phi}, [P_1]^{\eta}) = \rho^\rho([\mathcal{I}]^\phi, [Q_2]^{\phi}, [P_2]^{\eta})\]
HOANI FOR SW WATERMARKING?

\[(\eta)\mathcal{J}(\phi \sim \rho) : [P_1]^{\eta} = [P_2]^{\eta} \implies \]
\[\rho^\rho([\mathcal{J}]^{\phi, \eta}([Q_1]^{\phi}, [P_1]^{\eta})) = \rho^\rho([\mathcal{J}]^{\phi, \eta}([Q_2]^{\phi}, [P_2]^{\eta}))\]
HOANI FOR SW WATERMARKING?

\[(\eta) \mathcal{J}(\phi \sim \rho) : [P_1]^{\eta} = [P_2]^{\eta} \implies \rho^{\phi}([\mathcal{J}]^{\phi,\eta}( [Q_1]^{\phi}, [P_1]^{\eta} )) = \rho^{\phi}([\mathcal{J}]^{\phi,\eta}( [Q_2]^{\phi}, [P_2]^{\eta} ))\]
HOANI FOR SW WATERMARKING?

\[(\eta)\mathcal{I}(\phi \sim \rho) : [P_1]^{\eta} = [P_2]^{\eta} \implies \rho^\phi([\mathcal{I}]^{\phi,\eta}([Q_1]^{\phi}, [P_1]^{\eta})) = \rho^\phi([\mathcal{I}]^{\phi,\eta}([Q_2]^{\phi}, [P_2]^{\eta}))\]
HOANI FOR SW WATERMARKING?

\[(\eta) \mathcal{J}(\phi \sim \rho) : [P_1]^{\eta} = [P_2]^{\eta} \implies \rho^\phi ([\mathcal{J}]^\phi, [Q_1]^{\eta}, [P_1]^{\eta}) = \rho^\phi ([\mathcal{J}]^\phi, [Q_2]^{\phi}, [P_2]^{\eta})\]
HOANI for SW watermarking?

\[
(\eta) \mathcal{J}(\phi \sim \eta) : [P_1]^{\eta} = [P_2]^{\eta} \implies \\
\rho^\phi([\mathcal{J}]^{\phi, \eta}([Q_1]^{\phi}, [P_1]^{\eta})) = \rho^\phi([\mathcal{J}]^{\phi, \eta}([Q_2]^{\phi}, [P_2]^{\eta}))
\]
Many Thanks!!