Automatic Discovery and Quantification of Information Leaks

Aravind Machiry

University of California

machiry@cs.ucsb.edu

December 2, 2015

Overview

1. Motivation
2. Idea Overview
3. Approach Details
   - DisQUANT
   - Computing Quantitative Information flow metrics

Existing Works

- Assume equivalence relation is given.
- Only provide the number of secret/high bits leaked.

In this paper

Given a program:

- Automated way to find equivalence classes (DISCO) and their sizes (QUANT).
- Answer various questions:
  - Number of leaked bits.
  - Successful guess probability.

Notations

They model a Program P as a transition system (S, T, I, F):

- S: a set of program states.
- T: a finite set of transitions, each transition $\tau \in T$ is associated with a binary transition relation $\rho_\tau \subseteq SxS$.
- I: a set of initial states, $I \subseteq S$ and $I = I_{hi} x I_{lo}$.
- F: a set of final states, $F \subseteq S$ and $F = F_{hi} x F_{lo}$.
- σ: A program computation. Which is a sequence of program states $s_1, s_2, ..., s_n$, where $s_1 \in I$ and $s_n \in F$. All consecutive pairs of states should be a valid transition.
- π: A program path. Which is a non-empty sequence of program transitions i.e $\pi \in T^+$. $\pi = \tau_1, \tau_2, ..., \tau_n$, Such that for each $1 \leq i < n$, if $(s_1, s_2) \in \rho_{\tau_i}$, $(s_1^1, s_2^1) \in \rho_{\tau_{i+1}}$ then $s_2 = s_1^1$.

Finding Equivalence Classes (DISCO)

Equivalence Relation of High inputs is modelled as integer constraint on them. i.e $R : I_{hi} x I_{hi} \rightarrow \{true, false\}$

Example

Consider High inputs: $H = h_1,\ h_2$.
An example relation as constraint would be:
$R(H, \bar{H}) = (h_1 \leq \bar{h}_2 \wedge h_1 > h_2) \vee (\bar{h}_1 \leq \bar{h}_2 \wedge h_1 \geq h_2)$.
If we want to find equivalence class of High inputs:$(1, 2)$, then replace $\bar{h}_1 = 1,\ \bar{h}_2 = 2$ in the above constraint resulting in :
$(h_1 \leq 2 \wedge h_1 > h_2) \vee (h_1 \geq h_2)$.
All possible values of $h_1,\ h_2$, which satisfy the second constraint belong to the same equivalence class.

DISCO: $Leak_p$

Notations:

- R: Equivalence relation over I_{hi}, i.e $R \subseteq I_{hi} x I_{hi}$
  - $All_{hi} = I_{hi} x I_{hi}$: Constant output program. All inputs belong to same equivalent class. **Non-inference**
  - $=_{hi} \equiv \{(s_{hi}, s_{hi}) \mid s_{hi} \in I_{hi}\}$: Each input belong to its own class. **Leaks everything**.
- E or Experiments: Set of Low inputs ($E \in I_{lo}$, can be controlled by the attacker) used to compute various metrics. This models the threat model, one wants to use to compute various information flow metrics.

$Leak_p$

There is an information leak in Program P w.r.t R, if there is a pair of program paths π and η that start from initial states with R-equivalent high components and equal low components in E, and lead to final states with different low components: $\exists s, t \in I \exists s', t' \in F : (s, s') \in \rho_\pi \wedge (t, t') \in \rho_\eta \wedge s_{lo} = t_{lo} \wedge (s_{hi}, t_{hi}) \in R \wedge s_{lo} \in E \wedge s'_{lo} \neq t'_{lo}$

DISCO: *Confine$_p$* and *Refine$_p$*

$Confine_p(R, E)$

This relation indicates if R correctly over-approximates the maximal information that is leaked when Program P is run on the experiments E. In short there are no leaks: $\forall \pi, \eta \in T^+ : \neg Leak_p(R, E, \pi, \eta)$.
The largest equivalence relation R with $Confine_p(R, E)$ is the most precise charecterization of the leaked information, denoted by $\approx_E$.
$\approx_E \equiv \bigcup\{R \mid Confine_p(R, E)\}$.

$Refine_E(\pi, \eta)$

This represents refinement of Relation R w.r.t paths π and η. In short, creates new equivalence classes such that there is no leak w.r.t π and η.
$Refine_E(\pi, \eta) \equiv \{(s_{hi}, t_{hi}) \mid \forall s, t \in I \forall s', t' \in F : (s, s') \in \rho_\pi \wedge (t, t') \in \rho_\eta \wedge s_{lo} = t_{lo} \wedge s_{lo} \in E \rightarrow s'_{lo} = t'_{lo}\}$.

Finding Equivalence Classes (DISCO): Overview

Algorithm 1 Overview of Disco

1: P = Program to Test
2: R = $I_{hi} x I_{hi}$
3: **while** exists $\pi, \eta \in T^+ : Leak_P(R, E, \pi, \eta)$ **do**
4: $\quad R = R \cap Refine_E(\pi, \eta)$
5: **end while**
6: $R = R \cup =_{I_{hi}}$ // Adding identity relation for deterministic programs.

Implementation Details (DISCO) 1

For a given program P, a modified version $\bar{P}$ is created where every variable x is replaced with $\bar{x}$. Then $Leak_p$ is implemented as:

$Leak_p$

if $(l = \bar{l} \wedge l \in E \wedge (h, \bar{h}) \in R)$
  $P(h, l)$
  $\bar{P}(\bar{h}, \bar{l})$
  if $l \neq \bar{l}$
    error

Here, reachability of **error** indicates possibility of leak. Model checker *ARMC* is used, when **error** is reached this results in counter-example as paths π in P and η in $\bar{P}$ along with a formula in **linear arithmetic** that characterizes all initial states i.e pairs of $((h, l), (\bar{h}, \bar{l}))$ i.e $\bar{R}$.

Implementation Details (DISCO) 2

Let $\bar{R_E}$ be projected high inputs from $\bar{R}$. In short, $\bar{R_E}$ characterizes all pairs of high inputs from which the error state can be reached with an experiment from E. Then
$\bar{R_E} \equiv \{(h, \bar{h}) \mid \exists l \in E : ((h, l), (\bar{h}, l)) \in \bar{R}\}$.

$Refine_E(\pi, \eta)$

Given $\bar{R_E}$, $Refine_E$ can be defined as:
$Refine_E(\pi, \eta) \equiv I_{hi} x I_{hi} \setminus \bar{R_E}$ or $I_{hi} x I_{hi} \wedge \neg(\bar{R_E})$.

To Note

- $h, l, \bar{h}, \bar{l}$ and $\bar{R_E}$ are linear arithmetic constraints.
- E is defined also as a constraint. Ex: $E = i^1_{lo} > 0 \wedge i^2_{lo} < 2^{31} \wedge i^2_{lo} \geq 0$.

Example

Consider the example below:

```
int l = 0;
for(int i=0; i<n; i++) {
        if(h[i] > h[l])
                l = i;
}
```

They unroll the loop with every $h[i]$ replaced by h_i.

Equivalence Class Constraint (after DISCO with n = 3)

$R \equiv (h_1 < h_3 \wedge h_2 < h_3 \wedge \bar{h}_1 < \bar{h}_3 \wedge \bar{h}_2 < \bar{h}_3) \vee (\bar{h}_1 < \bar{h}_3 \wedge \bar{h}_3 \leq \bar{h}_2 \wedge h_1 < h_2 \wedge h_3 \leq h_2) \vee \ldots$ Refer paper for complete formula. The first conjunction represents equivalence class of inputs where element h_3 is the greatest.

Finding Equivalence Classes Sizes (QUANT): Overview

Algorithm 2 Overview of QUANT

i $= 1$
$Q = I_{hi} \leftarrow$ All Inputs satisfy this constraint
while $Q \neq \emptyset \leftarrow$ Is Q satisfiable **do**
    $s_i =$ select in $Q \leftarrow$ Find an input that satisfies Q
    $n_i = Count([s_i]_R)$
    $Q = Q \wedge \neg([s_i]_R)$
    $i = i + 1$
end while
return $\{n_1, n_2, ..., n_{i-1}\}$

Implementation Details (QUANT)

$Q \neq \emptyset$

Satisfiability check of Q.

select in Q

Assignment of high inputs which satisfy Q.

$Count([s_i]_R)$

Count number of solutions that satisfy R when its¯inputs are replaced by s_i. They use LATTE (Lattice Point Enumeration Tool).

Example

Consider the case where $(\bar{h_1}, \bar{h_2}, \bar{h_3}) = (1, 2, 3)$, Replacing corresponding values in R, gives us:
$R = (h_1 < h_3 \wedge h_2 < h_3 \wedge 1 < 3 \wedge 2 < 3) \vee \mathit{false} \vee \mathit{false}... = (h_1 < h_3 \wedge h_2 < h_3)$
which is the constraint for all high inputs which belong to the same class as $(1, 2, 3)$.
Limiting to 32-bit numbers i.e $0 \leq h_1, h_2, h_3 \leq 2^{32} - 1$ size of the equivalence class or number of solutions to the above constraint are 26409387495531407161709035520

Information flow metrics 1

Consider $p : I_{hi} \rightarrow R$, probability distribution of high inputs.

Guessing Entropy (average number of guesses) : G or $G(U)$

Let all high inputs be arranged in their decreasing order of distribution.
$p(I^i_{hi}) \geq p(I^j_{hi})$ whenever $i \leq j$.
$G = \sum_{1 \leq i \leq |I_{hi}|} i.p(I^i_{hi})$

Guessing Entropy when equivalence classes are known: G_R or $G(U|\nu_R)$

Let $\nu_R : I_{hi} \rightarrow [I_{hi}]_R$ be the map of secret inputs to its equivalence classes computed when run on experiments E. Given ν_R, p could be modified depending on the size of equivalence classes. Lets say: p_{ν_R}.
$G_R = \sum_{1 \leq i \leq |I_{hi}|} i.p_{\nu_R}$

Information flow metrics 2

Minimal Guessing Entropy ($\hat{G}_R$ or $\hat{G}(U|\nu_R)$)

Minimal guessing effort for the weakest secrets i.e high inputs with large equivalence classes.
$\hat{G}_R = min(G_R|\nu_R = [s_{hi}]_R|s_{hi} \in I_{hi})$.

Shannon entropy can be computed in exactly the same way as explained by Lucas in his first presentation

Example

Simple Password checker:

```
if (l==h)
  out = 1;
else
  out = 0;
```

Relation computed by DISCO would be:
$R = (\bar{h} = l \wedge l - h \leq -1) \vee (\bar{h} = l \wedge l - h \geq 1) \vee (h = l \wedge \bar{h} - l \leq -1) \vee (h = l \wedge l - \bar{h} \leq -1)$.
To compute password entropy after one guess, Lets consider $E = (l == 0)$, then the constraint R_2 computed by DISCO would be:
$R_2 = (\bar{h} = 0 \wedge 0 - h \leq -1) \vee (\bar{h} = 0 \wedge 0 - h \geq 1) \vee (h = 0 \wedge \bar{h} - 0 \leq -1) \vee (h = 0 \wedge 0 - \bar{h} \leq -1)$

Example (cont)

Given R_2, considering 32 bit numbers for h, QUANT will compute 2 equivalence classes, $B_1 \equiv h = 0, B_2 \equiv h \leq -1 \vee h \geq 1$ and corresponding sizes are $|B_1| = 0$ and $|B_2| = 2^{32} - 2$.
Shannon entropy $= \frac{1}{2^{32}} \sum_{i=1}^{2} |B_i|.log(|B_i|) = 31.99999999992$, after single guess.