統計的手法によるプログラムの定量的情報流解析

Tom Chothia[1], Yusuke Kawamoto[2], and Chris Novakovic[3]
[1]University of Birmingham, UK, [2]AIST, Japan, [3]Imperial College London, UK

概 要

定量的情報流解析は，システムやプログラムの出力データに漏洩する秘密情報の量を計算する技術であり，盛んに研究されてきた．しかし，従来技術では，システムやプログラムの規模が大きい場合，状態爆発によって扱えないという問題があった．そこで，本研究では，状態爆発を避けるために，統計的手法を用いて漏洩情報量を推定する方法を提案する．また，開発したツールを用いて，この方法の有効性を実証する．

1 背景

情報システムの安全性は，必ずしも完璧である必要はなく，実用上問題のない水準であれば十分である．例えば，クレジットカードのレシートにはカード番号の下 4 桁が表示されることが多く，秘密情報の一部が漏洩してしまっているが，カード番号全体を推測できる確率は非常に小さいため実害がない．また，パスワードによる認証システムでは，推測したパスワードが本物と一致するかどうかが分かるが，パスワードの推測に成功する確率は非常に小さいため，実用上問題がない．このような確率的システムが実用上問題のない安全性を備えているかどうかを判断するには，システムがどのぐらい安全なのかを**定量的に評価**する技術が欠かせない．

2 定量的情報流解析とは

システムの出力データに漏洩する秘密情報の量を計算する解析技術としては，**定量的情報流解析**（quantitative information flow）があり，最近十数年間，盛んに研究されてきた [1, 2]．この解析技術では，システムの出力データと秘密情報の間の対応関係を，情報理論における**通信路**としてモデル化し，エントロピーを用いて秘密情報の量を定義する．

通信路は，秘密情報の集合 $\mathcal{S}$，出力情報の集合 $\mathcal{O}$，通信路行列 C の三つ組で定義される．秘密 $s \in \mathcal{S}$ と出力 $o \in \mathcal{O}$ に対し，通信路行列の要素 $C[s,o]$ は，秘密が s であるときに出力が o である条件付き確率 $p(o|s)$ として定義される．

```
r := { 0 ↦ 0.5, 1 ↦ 0.5 };   // 0 と 1 の確率が半々
secret sec0 := { 0 ↦ 0.5, 1 ↦ 0.5 };
secret sec1 := { 0 ↦ 0.5, 1 ↦ 0.5 };
observe obs0 := sec0 xor r;
observe obs1 := sec1 xor r;
```

図 1. 排他的論理和を用いた単純なプログラム

例えば，図 1 のプログラムに対応する通信路は，表 1 に示す行列 C を用いて，$(\{sec0, sec1\}, \{obs0, obs1\}, C)$ で定式化される．このプログラムの出力から漏洩する情報は，秘密のビット *sec0* と *sec1* が一致するかどうかである．

$sec \setminus obs$	(0, 0)	(0, 1)	(1, 0)	(1, 1)
(0, 0)	0.5	0	0	0.5
(0, 1)	0	0.5	0.5	0
(1, 0)	0	0.5	0.5	0
(1, 1)	0.5	0	0	0.5

表 1. 通信路行列 C

情報量の定義として最もよく使われるのが**相互情報量**（mutual information）であり，秘密情報の分布 π と通信路行列 C に対して

$$I(\pi, C) = \sum_{\substack{s \in \mathcal{S}, \\ o \in \mathcal{O}}} \pi[s] C[s,o] \log_2 \left(\frac{C[s,o]}{\sum_{s' \in \mathcal{S}} \pi[s'] C[s',o]} \right)$$

で定義される．上の例では秘密の分布 π は一様であり，漏洩ビット数は $I(\pi, C) = 1$ となる．

3 プログラムからの漏洩情報量の計算

プログラムからの漏洩情報量の計算は，以下のようにして自動的に行うことができる [3]．まず，与えられたプログラムに対して，その状態遷移図（離散時間マルコフ連鎖）を生成する．状態遷移図では，各ノードが変数とその値を保持し，枝には遷移確率がついている．次にこの状態遷移図をもとにして，秘密情報の分布 π と通信路行列 C を計算する．これらを用いて，漏洩情報量 $I(\pi, C)$ を計算する．

しかし，従来技術では，大きなプログラムを扱おうとすると，状態遷移図を計算する際に**状態爆発**を起こしてしまうという問題がある．

4 プログラムからの漏洩情報量の推定

そこで，我々は，この状態爆発を避けるため，**統計的手法**を用いて，プログラムから漏洩する

秘密情報の量を推定する新たな方法を考案した．

具体的には，まず，プログラムをランダムに何度も実行して，秘密情報と出力情報を含む**実行トレース**を記録する．次に，こうして得られた実行トレースの集合を用いて，経験分布 $\hat{p}(s,o)$ を計算し，相互情報量 $\hat{I}$ を求める．

ここで，$\hat{I}$ には誤差があり，真の値 I よりも大きいことが統計学の研究 [4] で知られている．具体的には，真の値 I が 0 のとき，標本の大きさ（実行トレースの総数）n に対して，$2n\hat{I}$ が自由度 $(\#\mathcal{S}-1)(\#\mathcal{O}-1)$ の χ^2 分布に従う．また，$I>0$ の場合も，真の値からのずれの平均と分散が知られている．そこで，これらを用いて，相互情報量の**誤差を補正**し，さらに**95%信頼区間**を計算して，相互情報量の推定値の正確さの度合いを評価することができる．

なお，漏洩情報量としては，相互情報量だけでなく，*min-entropy leakage* もよく用いられる．この情報量は 1 回の推測攻撃による情報漏洩の評価に用いられる．我々は，実行トレースの集合から min-entropy leakage を統計的に推定する手法を考案した．詳細は [5, 6] にある．

5　解析ツールと解析例

我々は，前節の新たな統計的手法を実装して情報漏洩解析ツール leakiEst と LeakWatch を開発し，公開した [7]．これらのツールは，簡単なリバースエンジニアリングやサイドチャネル攻撃の発見に応用できる．

解析ツール leakiEst [8] は，実行トレースの集合から漏洩情報量を統計的に推定する．leakiEst では，ソースコードがなくとも，実行トレースの集合さえあれば，漏洩情報量を推定できる．そのため，ハードウェアの実行や，ネットワーク上での通信プロトコルの実行から得られるトレース集合を用いて，漏洩情報量を推定できる．例えば，Tor プロトコルのトレース集合を用いて，メッセージの送信者情報がパケット長やパケット数に何ビット漏洩するのかを leakiEst で調べ，匿名性を定量的に評価できる．

解析ツール LeakWatch [5] は，与えられた Java のソースコードに対して，ランダムに何度も実行トレースを生成し，内部で leakiEst を用いて漏洩情報量を統計的に推定する．例えば，LeakWatch を用いると，Java の Random.class と SecureRandom.class の擬似乱数の精度を比較できる．(詳しく述べると，2 個目の擬似乱数は 1 個目の擬似乱数に依存して生成されるため，前者には後者の情報がある程度漏洩している．LeakWatch を用いると，何ビット漏洩しているかを調べられる．)

一般に，我々の手法では，プログラムの状態数が膨大であっても，秘密情報と出力情報の取り得る値の数 $\#\mathcal{S}$ と $\#\mathcal{O}$ がどちらも十分小さければ，非常に効率よく正確に漏洩情報量を推定できる．しかし，$\#\mathcal{S}$ や $\#\mathcal{O}$ が大きくなると，統計的推定に必要な実行トレースの数が膨大なってしまうという欠点がある．

そこで，今後の課題として，より大規模なプログラムを解析可能にするために，統計的手法だけでなく，合成的手法 [9, 10] や静的コード解析を組み合わせた新たな手法を開発中である．

参考文献

[1] D. Clark, S. Hunt, and P. Malacaria. Quantitative analysis of the leakage of confidential data. In *Proc. of QAPL*, volume 59 (3) of *ENTCS*, pages 238–251. Elsevier, 2001.

[2] G. Smith. On the foundations of quantitative information flow. In *Proc. of FOSSACS*, pages 288–302, 2009.

[3] T. Chothia, Y. Kawamoto, C. Novakovic, and D. Parker. Probabilistic point-to-point information leakage. In *Proc. of CSF*, pages 193–205. IEEE, June 2013.

[4] R. Moddemeijer. On estimation of entropy and mutual information of continuous distributions. *Signal Processing*, 16:233–248, 1989.

[5] T. Chothia, Y. Kawamoto, and C. Novakovic. LeakWatch: Estimating information leakage from java programs. In *Proc. of ESORICS*, pages 219–236, 2014.

[6] T. Chothia and Y. Kawamoto. Statistical estimation of min-entropy leakage, April 2014. Manuscript available at `http://www.cs.bham.ac.uk/research/projects/infotools/`.

[7] `http://www.cs.bham.ac.uk/research/projects/infotools/`.

[8] T. Chothia, Y. Kawamoto, and C. Novakovic. A Tool for Estimating Information Leakage. In *Proc. of CAV*, pages 690–695, 2013.

[9] Y. Kawamoto, K. Chatzikokolakis, and C. Palamidessi. Compositionality results for quantitative information flow. In *Proc. of QEST*, pages 368–383, 2014.

[10] Y. Kawamoto and T. Given-Wilson. Quantitative information flow for scheduler-dependent systems. In *Proc. of QAPL*, 2015.