Leakage in presence of an active and adaptive adversary

Cristian Ene , Laurent Mounier
e-mail : Cristian.Ene@univ-grenoble-alpes.fr
Laurent.Mounier@univ-grenoble-alpes.fr

January 14, 2021

Measuring the information leakage of a system is very important for security. From side-channels to biases in random number generators, quantifying how much information a system leaks about its secret inputs is crucial for preventing adversaries from exploiting it; this has been the focus of intensive research efforts in the areas of privacy and of quantitative information flow (QIF). For example, both programs in Figure 1 are leaking some additional information about the secret if one can measure the execution time or if one can observe the instruction cache. Moreover, by interacting iteratively with the application, the adversary is able to improve his knowledge [4].

```
void compare(int l, int  s){
  if (s<l)
      {write_log(‘‘too large’’);}    // 1 sec.
  else
      {some_computation();}          // 2 sec.
}
```

```
int pwdCheck(char *l, char*  pwd){
  unsigned i;
  for (i=0; i<B_Size; i++)
       if (l[i]!=pwd[i])
            {return 0;}
  return 1;
}
```

Figure 1: Leaking programs

Hence the overall scenario (Figure 2) is the following one:

- Some secret $x \in \mathcal{X}$ is generated and provided to the application
- Iteratively and adaptively,
  1. The adversary provides some public input $l \in \mathcal{L}$ to the application
  2. The application does some computation and outputs some $y \in \mathcal{Y}$

The adversary's knowledge about the the secret $x \in \mathcal{X}$ at some moment i is called the prior probability π_i (e.g. initially, π_0 would be the uniform distribution on $\mathcal{X}$). In our context, the application corresponds to a family of probabilistic channels $(\mathcal{C}_l)_{l\in\mathcal{L}}$, such that for each $x \in \mathcal{X}$ and $l \in \mathcal{L}$, it returns a $y \in \mathcal{Y}$ according to some distribution $\mathcal{P}_{\mathcal{C}_l}(Y = y \mid X = x)$. In the considered

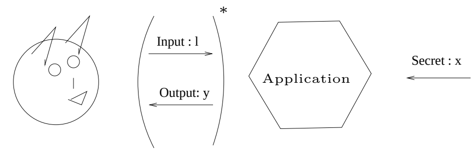

Figure 2: The target scenario

scenario, the adversary interacts iteratively (Figure 3) with the application until his knowledge π_k achieves some desired vulnerability level $\mathcal{V}(\pi_k)$.

```
π←π₀;                                        // (1)
while V(π) ≤ ϵ do                            // (2)
    l₀←argmax_{l∈L}  V[π▷C_l];               // (3)
    Execute App with input l₀;
    Get the output y₀ returned by App;
    Update π according to  y₀                // (4)
```

where

- (1) π_0 is the initial probabilistic information about the secret x[1] (called the prior)
- (2) ϵ is the intended level of knowledge (modelled by some measure $\mathcal{V}$) about the secret
- (3) find the "best" input l_0 that optimises the leakage ; $\pi \triangleright \mathcal{C}_{l_0}$ is the hyper-distribution corresponding to executing App with prior π and input l_0, i.e. the distribution of posteriors $\mathcal{P}(X \mid Y = y_0, L = l_0)$, each with probability $\mathcal{P}(Y = y_0 \mid L = l_0)$
- (4) use the Bayes law to update the belief: $\pi \leftarrow \mathcal{P}(X \mid Y = y_0, L = l_0)$

Figure 3: Attacker's strategy

Several issues can be investigated in this internship:

- What are the best choices for the measures $\mathcal{V}$ and $\mathcal{V}[\pi \triangleright C]$?
- How to compute/approximate for each input l, the associated probabilistic channel $\mathcal{C}_l$ (do we have the source code or not for App)?
- How to compute/approximate $argmax_{l \in \mathcal{L}}\ \mathcal{V}[\pi \triangleright \mathcal{C}_l]$? , given that in most of the cases the sets of inputs $\mathcal{L}$, secrets $\mathcal{X}$ and observables $\mathcal{Y}$ can be very large, and that in the most realistic scenario, the output Y will rather be a continuous random variable [2].
- Information-theoretic vs. probabilistic polynomial-time adversary

The topic of this internship can be oriented in various directions:

- refining the scenario from Figure 3 in a grey-box case, where the attacker has the binary code of the application
- apply machine learning (ML) methods [3, 5] in order to get the necessary scalability (whenever the sets of inputs $\mathcal{L}$ or secrets $\mathcal{X}$ are very large or if the output Y is a continuous random variable) for the grey- or black-box measurements
- implementing the scenario from Figure 3 in order to synthesis an adaptive attack [6] or to measure the vulnerability for a concrete application.

References

[1] Mário S. Alvim et al. "An Axiomatization of Information Flow Measures". In: *Theoretical Computer Science* 777 (2019), pp. 32–54. DOI: `10.1016/j.tcs.2018.10.016`. URL: `https://hal.archives-ouvertes.fr/hal-01995712`.

[2] Lucas Bang, Nicolás Rosner, and Tevfik Bultan. "Online synthesis of adaptive side-channel attacks based on noisy observations". In: *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE. 2018, pp. 307–322.

[3] Giovanni Cherubin, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. "F-BLEAU: Fast Black-Box Leakage Estimation". In: *S&P 2019 - 40th IEEE Symposium on Security and Privacy*. San Francisco, United States: IEEE, May 2019, pp. 835–852. DOI: `10.1109/SP.2019.00073`. URL: `https://hal.archives-ouvertes.fr/hal-02422945`.

[4] Quoc-Sang Phan et al. "Synthesis of adaptive side-channel attacks". In: *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*. IEEE. 2017, pp. 328–342.

[5] Marco Romanelli et al. "Estimating g-Leakage via Machine Learning". In: *CCS '20 - 2020 ACM SIGSAC Conference on Computer and Communications Security*. This is the extended version of the paper which appeared in the Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS), November 9-13, 2020, Virtual Event, USA. Online, United States: ACM, Nov. 2020, pp. 697–716. URL: `https://hal.archives-ouvertes.fr/hal-03091469`.

[6] Seemanta Saha et al. "Incremental Attack Synthesis". In: *ACM SIGSOFT Software Engineering Notes* 44.4 (2019), pp. 16–16.