Protecting Control Flow Confidentiality in Cloud-based Computation

Yongzhi Wang, Jinpeng Wei
Florida International University

Motivation

- More and more innovative algorithmic computations are being deployed to the public cloud in important applications (e.g., business analytics, geospatial mapping/searching, bioinformatic analysis, and image processing).
- Control flow, which decides the sequence of instructions (e.g., statements) to be executed, directly reflects the algorithm of a program.
- How to protect control flow confidentiality (and thus the confidentiality of the algorithm) of an outsourced program deployed to the public cloud?

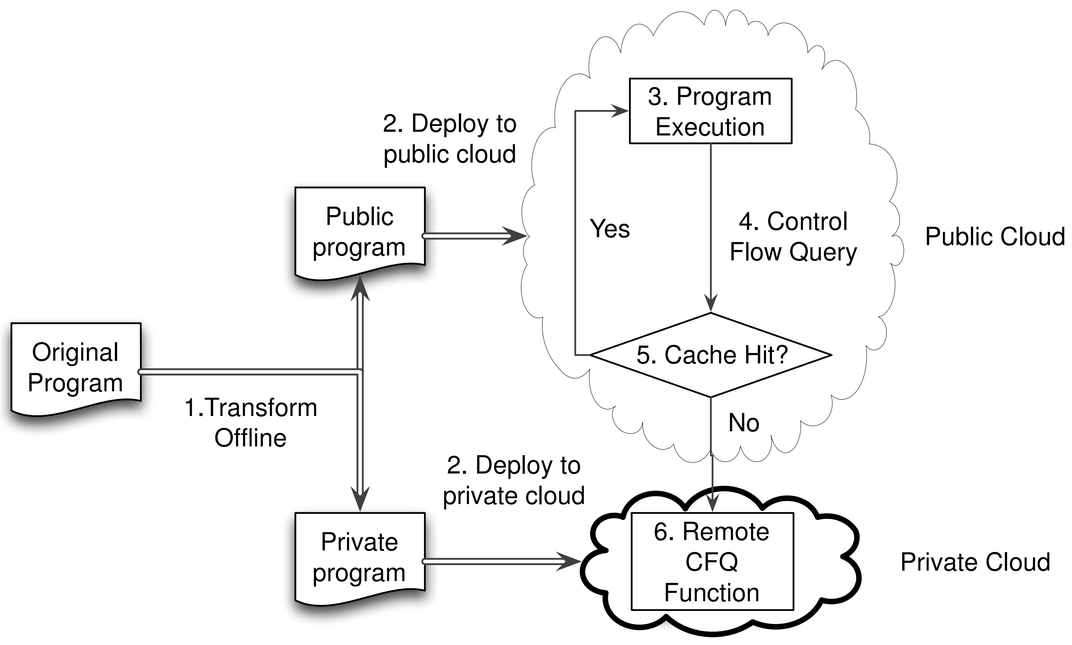
Solution: Runtime Control Flow Obfuscation

Idea:

- Transform each original program into a public program and a private program.
- Public program: execute on the public cloud; perform most execution except for evaluating predicates of branch statements.
- Private program: execute on private cloud; evaluate predicates.

Proposed Techniques:

- Replace predicates of the branch statements in the original program with control flow query (CFQ) invocations to the private program.
- Insert indistinguishable fake branch statements in the original program to raise the bar for the attacker to understand the algorithm's control flow.
- Maintain a continuous cache to reduce the cost of cross-cloud communication.



RCFO Example

The original function of the K-means clustering algorithm

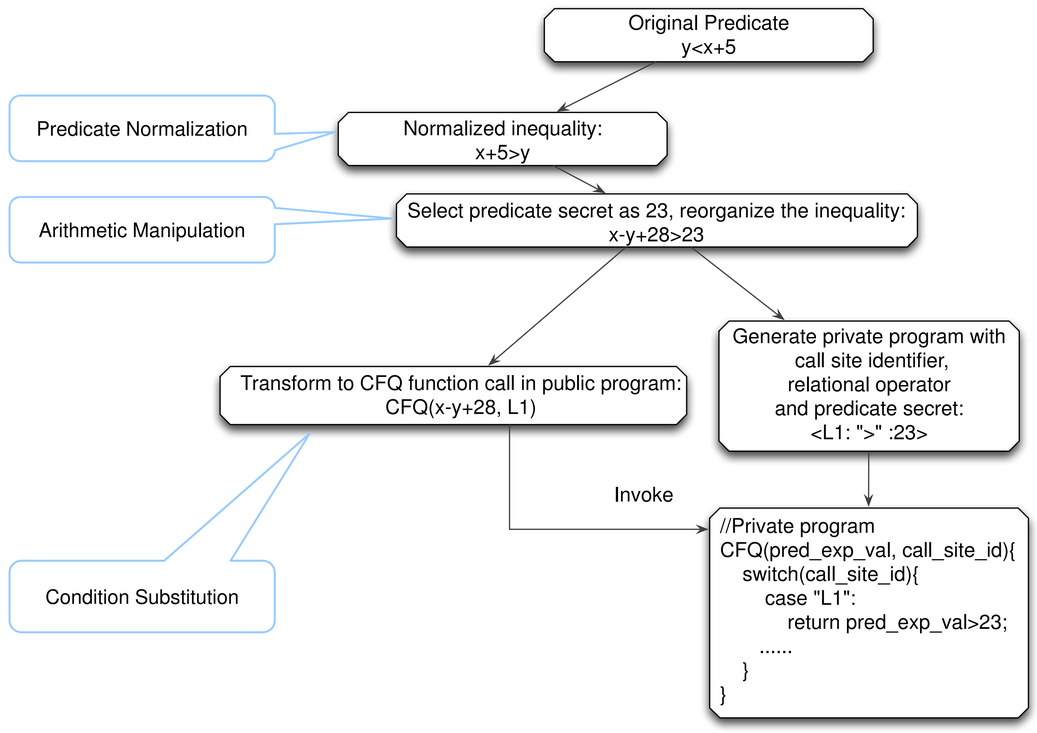
```
  boolean computeConvergence(
        DistanceMeasure measure,
        double convergenceDelta) {
    Vector centroid=computeCentroid();
    double distance=measure.distance(
        centroid.getLengthSquared(),
        centroid, getCenter());
L1: if (distance <= CONVERGENCEDELTA)
        //CONVERGENCEDELTA is a constant
        return true;
    else
        return false;
  }
```

The function after RCFO: the constant variable CONVERGENCEDELTA is hidden in the private program.

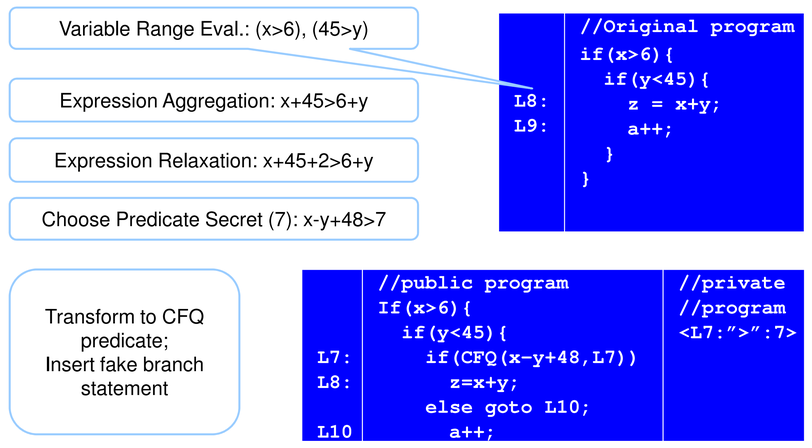
```
  boolean computeConvergence(
        DistanceMeasure measure) {
    Vector centroid=computeCentroid();
    double distance = measure.distance(
        centroid.getLengthSquared(),
        centroid, getCenter());
L1: if(CFQ(0.05-distance,L1))
        //CONVERGENCEDELTA is now hidden
        //in the private program
        return true;
    else
        return false;
  }
```

Yongzhi Wang, Jinpeng Wei. "Toward Protecting Control Flow Confidentiality in Cloud-based Computation". Elsevier Journal of Computers & Security (in press)

Transform original predicates

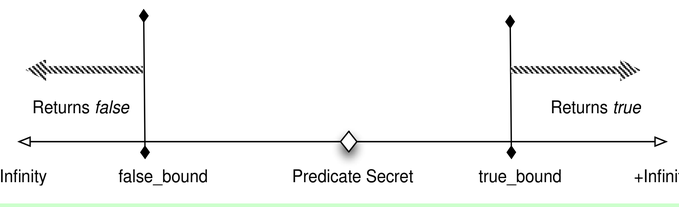


Insert fake branch statements



Continuous Cache

- For each *CFQ(exp, Ln)* on the public cloud, maintain a *true_bound* and a *false_bound* and implement a public program version.
- For each invocation of *CFQ(exp, Ln)* on the public program, If *exp>true_bound*, the public program version returns *true*. If *exp<false_bound*, the public version returns *false*. Otherwise, invoke on the real CFQ function in the private program.



Experiments Result

Execution time of MapReduce jobs with different obfuscation degree d

- The experiments are performed on a hybrid cloud (a private cloud and Amazon Elastic MapReduce).
- Obfuscation degree: The probability of inserting a fake branch statement before each original statement