Scribe Notes

Thursday, March 6

Andrew Ruef

Summary

Declassification: Dimensions and Principles provides a framework for discussing declassification policies and implementations centered around four axes: *what* information is released, *where* in the code is it released, *who* is the information released by and to, and *when* is this information released. It also gives some principles for declassification mechanisms: *consistency*, *conservativity*, *monotonicity*, and *non-occlusion*.

Gradual Release: Unifying Declassification, Encryption and Key Release Policies presents a system that performs declassification by releasing information that has been encrypted with a secret key and then releasing the key to precisely release the information encrypted with that key. They present a type system that prevents the use of a key on high security information once released.

Questions

What is the similarity between immediate and gradual declassification? You can perform immediate declassification by composing newkey, encrypt, release, and decrypt.

Is the gradual release type system flow sensitive? Yes

What is a *what* policy? Aseem: what information is released, specifying the bits in the program to release.

What is a *when* policy? Aseem: Preconditions for release, relative to time

Kris: This has a parallel to “release in 30 years”, release after the relevant information is stale

Kris: Can’t you encode most of this as what? What under which circumstances, with some notion of condition.

James: I like where policies better, specifies locality, minimizes checking.

An example (1):

```
if(h1)
  declassify
else
  skip
```

Classic implicit case, lines aren’t important the context is important. Check that the pc can flow to a label at a specific point.

Kris: Differentiate between intensional vs extensional guarnatees, information control vs access control?

Impose a monitor? But that's bogus because the right people do need the information

Aseem: is 1. A when policy? Or a who? It feels more like a principle.

What about with delimited release? We have a notion of non-interference, all high inputs not related to low outputs. If we relax this, what if the attackers know some things, how does this change the relationship to the low output? It's important that the attacker can't learn anything *more*. The attacker should only learn declassify statements.

Aseem: Back to 1, with delimited release, is it okay? No, you learn about h which is disallowed.

Aseem: But the low memories are equal, so it is okay? Oh yeah, then yes. If the else branch assigned something to x, then it would be bad.

Kris: Relational time, happens-before. A big problem is time-varying data.

Nate: When analysis seems parallel to crypto guarantees? Yes.

What is semantic consistency? Security policies ought to be semantic.

James: Two programs reduce to the same program? Yes, and so do their events. When does a policy fail this condition? If the policy depends on a decidable thing it doesn't work. If the policy is syntactic, there is trouble.

There is a difference between flows-to and noninterference. Non-interference is semantic, Denning-style analyses are syntactic. Want the policy to "stand alone" and be as precise as possible.

What is conservativity? What is it? The program should provide no new information to the attacker.

Aseem: Policy doesn't follow if h1, h2 premise is false?

Aseem: Adding declassify statements doesn't change the program? No, declassification has no semantic effect on the program.

Luis: If a program is insecure, is adding a declassify making the program secure? Yes

What is non-occlusion? James: Can't smuggle more information through? Yes

Quantify information leakge: How many bits leak, is it a what or a who? Nate: A what? Yes

Intransitive non-interference?

Kris: Explicitly cast between levels.

When would this be meaningful?

James: We can guarantee encryption or sanitization of inputs.

Aseem: Is this similar to a shim? Have all declassifications or sanitizations go through a shim?

What about integrity vs confidentiality?

James: Flip the lattice upside down? Yes

Aseem: High shows data is trusted, rather than doing declassification you are doing sanitization. You

could also name the structure of the input that you want.

There is a dual relationship between integrity and confidentiality. Where is appealing because you (the author) know your program, you study the declassify statements and know what you are releasing. Where policies are also easy to check.

Criticisms

What vs where – where requires comprehension of a potentially large amount of code, what doesn't care

Ideas

James: Can you check gradual release statically? It would be super annoying.