COL876: SPECIAL TOPICS IN FORMAL METHODS

Formal verification of security protocols

Lecture 9, 9 October 2023

RECAP

- Saw some tools: ProVerif, Tamarin…
- There's a whole zoo of tools
- Some more specialized than others
- Scyther, Avispa, APTE, DeepSec, SAT-Equiv &c.

INSECURITY PROBLEM

- Given a protocol Pr, is there an attack?
- Undecidable in the general case
  - saw reduction from 2CM reachability
- Decidable for boundedly many sessions! [RT03]
- Consider the "K-bounded insecurity problem"
  - Given a protocol Pr, is there an attack involving $\leq$ K sessions?

K-BOUNDED INSECURITY PROBLEM

- Make copies of each role systematically, renaming variables
- Bake freshly generated names into the copy; no need to generate at runtime
- Suffices to check existence of attack involving K copies in all
- Each role thought of as a sequence of recv → send implications

K-BOUNDED INSECURITY PROBLEM

- Attack is a sequence $\xi = r_0 s_0 r_1 s_1 \ldots r_n s_n$ and substitution σ with
  - dom(σ) = vars(ξ)
  - for every $i \leq n$ and x ∈ vars(s_i): there is $j \leq i$ s.t. x ∈ vars(r_j)
  - for each $i \leq n$: $X_I^0 \cup \{s_0\sigma, \ldots, s_{i-1}\sigma\} \vdash r_i\sigma$ and $X_I^0 \cup \{s_0\sigma, \ldots, s_n\sigma\} \vdash$ secret.
- [RT03]: If there is an attack (ξ, σ), then there is one (ξ, τ) where $\tau(x) < B$ for all x such that B is a bound obtained from only the protocol description
- Guess interleaving ξ, small substitution τ, check if above derivations hold.

CONSTRAINT SATISFACTION

- Used in many tools; more systematic way of "guessing" τ
- Express derivation checks as constraints over vars(ξ) and B
- Solution to this constraint system is a substitution τ which
  - preserves derivability requirements, and
  - respects the bound B

CONSTRAINT SYSTEM

- Constraints $C = \{(S_1 \Vdash u_1), \ldots, (S_n \Vdash u_n)\}$ s.t. for every $i \leq n$:
  - $S_{i-1} \subseteq S_i$ for $i > 1$
  - If $x \in vars(S_i)$, then there is $j \leq i$ s.t. $x \in vars(u_j)$
- Solution is a substitution τ with
  - $dom(\tau) = vars(C)$, and
  - $S\tau \vdash u\tau$ for every $(S \Vdash u) \in C$

EXAMPLE

$$A \longrightarrow B : A, \mathsf{enc}(m, \mathsf{pk}(B))$$

$$B \longrightarrow A : \mathsf{enc}(m, \mathsf{pk}(A))$$

- Initiator a talks to b: [] → (pk(sk_a), aenc(m, pk(sk_b)))
- Responder b replies to a: (pk(sk_a), aenc(x, pk_b)) → aenc(x, pk(sk_a))
- Constraint system C defined as follows.

$$S_0, (\mathrm{pk}(sk_a), \mathrm{aenc}(m, \mathrm{pk}(sk_b)) \Vdash \mathrm{aenc}(x, \mathrm{pk}(sk_a))$$

$$\text{with } S_0 = \{\mathrm{pk}(sk_a), \mathrm{pk}(sk_b), sk_i\}$$

- Potential solution: $\tau = \{x \mapsto m\}$

EXAMPLE: ATTACK

$$A \longrightarrow B : A, \mathsf{enc}(m, \mathsf{pk}(B))$$
$$B \longrightarrow A : \mathsf{enc}(m, \mathsf{pk}(A))$$

- Initiator a talks to b: $[] \rightarrow (\mathrm{pk}(sk_a), \mathrm{aenc}(m, \mathrm{pk}(sk_b)))$
- Responder b replies to a: $(y, \mathrm{aenc}(x, pk_b)) \rightarrow \mathrm{aenc}(x, y)$
- Constraint system C defined as follows.

$$S_0, (\mathrm{pk}(sk_a), \mathrm{aenc}(m, \mathrm{pk}(sk_b)) \Vdash (y, \mathrm{aenc}(x, \mathrm{pk}(sk_b)))$$
$$S_0, (\mathrm{pk}(sk_a), \mathrm{aenc}(m, \mathrm{pk}(sk_b)), \mathrm{aenc}(x, y) \Vdash m$$
$$\text{with } S_0 = \{\mathrm{pk}(sk_a), \mathrm{pk}(sk_b), sk_i, \mathrm{pk}(sk_i)\}$$

- Potential solution: $\tau = \{x \mapsto m, y \mapsto \mathrm{pk}(sk_i)\}$

EXAMPLE

$$A \longrightarrow B : \mathsf{enc}((A, n_a), \mathsf{pk}(B))$$

$$B \longrightarrow A : \mathsf{enc}((n_a, n_b), \mathsf{pk}(A))$$

$$A \longrightarrow B : \mathsf{enc}(n_b, \mathsf{pk}(B))$$

CONSTRAINT SOLVING

- Algorithm: Sequence of rules to simplify a constraint system
- Non-deterministic; more than one rule might be applicable
- Each application implicitly builds τ
- If we keep applying these rules, can arrive at a "simple" constraint system — terms to the right of each $\Vdash$ are just single variables
- Decidable if a simple constraint system has a solution or not

CONSTRAINT SOLVING

- Depth-first search; might arrive at an insoluble system due to applying rules in a particular order
  - Backtrack and retry with a different sequence of rules!
- If all paths explored but no solution, insoluble system
- If current system is soluble, so is the original
- Every reduction path will end in a simple constraint system!

SIMPLIFICATION RULES

- Redundancy rule: Remove $T \Vdash u$ if u is already deducible from T along with variables from solved constraints
- Function rule: Guess that the attacker built f(u, v) from u and v
- Unsatisfiable rule: There is some ground constraint $T \Vdash u$ such that u is not deducible from T
- Unification rules: Guess a possible instantiation σ of variables by unifying two subterms of a constraint

SIMPLIFICATION RULES

- Formally, this procedure is:
  - Sound: any solution found by the procedure is indeed a solution of the constraint system.
  - Complete: whenever there is a solution of the constraint system, there is a path in the tree of possible simplifications that leads to a solution.
  - Terminating: there is no infinite path in the tree.
- Can be extended with various equational theories and security properties.