CS590U

Access Control: Theory and Practice

Lecture 10 (February 14)

Safety Analysis in HRU

Summary of Known Results in Safety Analysis in HRU

- Undecidable in the general case
  - Turing Machine can be reduced to Protection System in HRU
- Undecidable in the monotonic case (no delete/destroy)
  - PCP can be reduced to it
- Undecidable in bi-conditional monotonic case
- PSPACE-complete in the case of no create
  - whole thing becomes finite
- coNP-complete in the mono-operational case
  - only needs to consider one more new subject

Turing Machine

- A Turing Machine is a 7-tuple
  $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$
  - Q is the set of states
  - Σ is the input alphabet
  - Γ is the tape alphabet
  - δ is the transition function
  - $q_0 \in Q$ is the start state
  - $q_{accept} \in Q$ is the accept state
  - $q_{reject} \in Q$ is the reject state, $q_{reject} \neq q_{accept}$

Basic Results about Turing Machines

- Universal Turing Machines
  - there exists a UTM that take the description of a TM M and an input string α and outputs M(α)
- It is undecidable to determine whether an arbitrary Turing machine halts or not
  - there exists no algorithm that can take as inputs the description of a Turing machine and an input and decides whether the Turing machine halts

Simulating Turing Machines using Protection Systems

- Given a Turing machine, we construct a protection system
  - The set of generic rights include
    - the states of the Turing machine
    - the tape symbols of the Turing machine
    - and two special rights: `own’, `end’
  - Turing Machine instructions are mapped to commands of the protection system

Mapping a Tape to an Access Matrix

- The j'th cell on the tape = the subject s_j
- The j'th cell has symbol X $\Rightarrow$ X $\in$ (s_j , s_j)
- The head is at the j'th cell and the current state is q $\Rightarrow$ q $\in (s_j , s_j)$
- The k'th cell is the last $\Rightarrow$ 'end' $\in$ (s_k , s_k)
- For 1=j<k, `own' $\in$ (s_j , s_{j+1})

Moving Left: (q, X) -> (p, Y, left)

```
command $C_{qX}$(s, s′)
  if  q in (s′, s′) and X in (s′, s′)
      and `own′ in (s, s′)
  then   delete q from (s′, s′)
         delete X from (s′, s′)
         enter Y into (s′, s′)
         enter p into (s, s)
end
```

Moving Right (case one): (q, X) -> (p, Y, right)

```
command $C_{qX}$(s, s′)
  if  q in (s, s) and X in (s, s)
      and `own′ in (s, s′)
  then   delete q from (s, s)
         delete X from (s, s)
         enter Y into (s, s)
         enter p into (s′, s′)
end
```

Moving Right (case two): (q, X) -> (p, Y, right)

```
command $C_{qX}$(s, s′)
  if  q in (s, s) and X in (s, s)
      and `end′ in (s, s)
  then    delete q from (s, s)        delete X from (s, s)
          enter Y into (s, s)
          create subject s′           enter `own′ into (s, s′)
          enter p into (s′, s′)       enter B into (s′, s′)
          delete end from (s, s)      enter ‘end′ into (s′, s′)
  end
```

Summary

- Given a Turing Machine, it can be encoded as a protection system, so that the Turing Machine enters the accept state iff the HRU protection system leaks the right corresponding to q_{accept}
- Safety in HRU is thus undecidable.

What about the monotonic case?

- Use a reduction from the Post Correspondence Problem (PCP)

Six Notions of Safety

- leak safety (whether a right can be leaked by a command, used in the HRU paper)
  - (r)-leak-safety, (o,r)-leak-safety, and (s,o,r)-leak-safety
- simple safety (whether a right that does not exist in the initial state can be added, used in most follow-up work)
  - (r)-simple-safety, (o,r)-simple-safety, and (s,o,r)-simple-safety

Mono-operational HRU Systems

- Definition: each command has only one primitive operation in its body
- Key implications:
  - when an subject/object is created, no right can be added at the same time
  - a new subject/object is no different from any other new subject/object
- Theorem: Safety analysis is decidable in mono-operational HRU systems

Proof of Decidability

- General approach to prove decidability:
  - show that one only needs to consider a bounded number of possibilities
- Safety in mono-operational HRU
  - show that one only needs to add at most one subject

Argument Taken from the HRU paper

- The proof hinges on two simple observations. First, commands can test for the presence of rights, but not the absence of rights or objects. This allows delete and destroy commands to be removed from computations leading to a leak (since the system is mono-operational, we can identify the command by the type of primitive operation). Second, a command can only identify objects by the rights in their row and column of the access matrix. No mono-operational command can both create an object and enter rights, so multiple creates can be removed from computations, leaving the creation of only one subject. This allows the length of the shortest `leaky' computation to be bounded.

Examining the argument

- The above argument is flawed, why?

“Reduction” of (o,r)-safety to (r)-safety in [HRU]

- Given an instance of (o,r)-safety
  - Add two new generic rights r′ and r″,
  - Add r′ to (o,o)
  - Add the following command

```
Command DUMMY(x,y)
if    r in (x,y)   and   r′ in (y,y)
then      enter r″ into (y,y)
end
```

  - We get an instance of (r)-safety

Is this a reduction?

- What if a right is leaked in transit for (o,r)-safety?
  - this is not a reduction for the definition of safety (leak safety) in the paper
- What if the object o is removed and then added back in order to leak the right (o,r)?
  - in Unix, a none-owner having write permission can destroy the file and recreate it
- Even if a reduction exists, this does not mean that (o,r) safety is undecidable.

Open Problems in Safety Analysis in HRU

- What is the computational complexity with limited number of rights and limited number of commands?
  - what if there is only one generic right and one command?
    - seems still coNP-hard, but should be decidable
  - what if there is only one generic right?
  - what if there are only two generic rights?
  - what is there is only one command?

Issues in the Definition of Safety Problem

- Trusted subjects
- Whether to use leak-safety or simple-safety?
  - whether transient states should
- Beyond safety

Removing trusted subjects is a problem

- Why: also remove possible attacks
- Source of the problem: no concept of initiator of a command. Without it, cannot define concurrence or truly untrusted.

Whether Transient Right Should be Considered?

- Depends on whether a command is atomic and which states are considered to be reachable.
- Depends on intention of modeling
- In most usage, e.g., modeling of Graham-Denning, commands are atomic.
  - Atomic commands must exist
  - How about breaking up commands that are not atomic?

Beyond Safety

- The notion of safety is problematic
  - some subjects are entitled access, the list of these subjects may not be pre-determined
- Other notions of security are also needed
  - Availability: a subject always has access
  - An object always has an owner
  - Every subject that can read an object o has the control right over another subject s’
  - State-transition-based security properties

End of Lecture 10

- Next lecture
  - Understanding safety analysis and other work on safety analysis