

Declarative model

Preliminaries

Hierarchy of concepts

host

 gateway (i.e., every gateway is a host)

network

agent

 process (i.e., a running program)

 human

 ATTACKER (i.e., ATTACKER is a special human agent)

service

privilege

datum

 credential (i.e., every credential can be represented by some data, e.g., passwords, keys, biometric data)

channel (e.g., a TLS connection between remote processes)

storage (e.g., a disk in a host)

Constructions for describing system specific facts

(we may call it a static system model)

host:H is_connected_to network:N

process:P runs_on host:H

process:P provides service:S

process:P relies_on service:S

```

    // check if "process:P can_use service:S" can be derived from the given facts!
process:P relies_on datum:D

    // check if "process:P can_read datum:D" can be derived from the given facts!
service:S is_exposed
service:S offers capability:C

    // capabilities:
    // can_access(service:S')
    // can_perform(operation:O) ???
    // can_stop(process:P), can_update(process:P)
    // can_read(datum:D), can_write(datum:D)
human:U can_attach_to host:H
human:U can_attach_to network:N
datum:D is_a_copy_of datum:D'
datum:D is_a_function_of datum:D'
channel:C carries datum:D
channel:C traverses host:H
channel:C traverses network:N
storage:S stores datum:D
storage:S resides_on host:H
agent:A has credential:Cr
agent:A can_obtain privilege:Pr

privileges:
    S.use          // where S is a service
    C.send         // where C is a channel
    C.receive      // where C is a channel
    S.store        // where S is a storage
    S.retrieve     // where S is a storage
    H.user_U       // where H is a host, and U may be a user of H
    H.root         // where H is a host, and root is the distinguished sys admin user

```

credentials:

// passwords, keys, and biometric data of and used by agents

ATTACKER can_attach_to network:N (where N is the Internet)

Constructions for describing system specific rules

(we may call it a dynamic system model)

Capability of obtaining privileges

// [] means optional

[agent:A can_obtain privilege:Pr

pre:

agent:A can_obtain credential:Cr]

// such a rule exists if a privilege:Pr is connected to a credential:Cr

// e.g., if an agent knows a password, then it has some privilege

Capability of using services

agent:A can_use service:S

pre:

agent:A can_access service:S [AND

agent:A can_obtain privilege:S.use] // or other system specific guard conditions

Capability of sending and receiving on channels

agent:A can_send_on channel:C

pre:

agent:A can_access channel:C [AND

agent:A can_obtain privilege:C.send] // or other system specific guard conditions

agent:A can_receive_on channel:C

pre:

agent:A can_access channel:C [AND

agent:A can_obtain privilege:C.receive] // or other system specific guard conditions

Capability of storing in and retrieveing from storages

agent:A can_store_in storage:S

pre:

agent:A can_access storage:S [AND

agent:A can_obtain privilege:S.store] // or other system specific guard conditions

agent:A can_retrieve_from storage:S

pre:

agent:A can_access storage:S [AND

agent:A can_obtain privilege:S.retrieve] // or other system specific guard conditions

Capability of partially or fully controlling hosts

human:U can_fully_control host:H

pre:

human:U can_attach_to host:H AND

human:U can_obtain privilege:H.root

human:U can_partially_control host:H

pre:

human:U can_attach_to host:H AND

human:U can_obtain privilege:H.user_U

General rules for any system

Capability of accessing services

agent:A can_access service:S

pre:

agent:A can_use service:S' AND

service:S' offers_capability:can_access(service:S)

process:P can_access service:S

pre:

process:P provides service:S' AND

service:S' offers_capability:can_access(service:S)

OR

process:P runs_on host:H AND

process:P' runs_on host:H AND

process:P' provides service:S

OR

process:P runs_on host:H AND

host:H is_connected_to network:N AND

process:P' runs_on host:H' AND

host:H' is_connected_to network:N AND
process:P' provides service:S AND
service:S is_exposed

human:U can_access service:S

pre:

human:U can_attach_to host:H AND
process:P runs_on host:H AND
process:P provides service:S

OR

human:U can_attach_to network:N AND
host:H is_connected_to network:N AND
process:P runs_on host:H AND
process:P provides service:S AND
service:S is_exposed

Capability of performing operations

agent:A can_perform operation:O

pre:

agent:A can_use service:S AND
service:S offers capability:can_perform(operation:O)

process:P can_perform operation:O

pre:

process:P provides service:S AND
service:S offers capability:can_perform(operation:O)

Capability of stopping processes

agent:A can_stop process:P

pre:

agent:A can_use service:S AND

service:S offers capability:can_stop(process:P)

process:P can_stop process:P'

pre:

process:P provides service:S AND

service:S offers capability:can_stop(process:P')

Capability of updating processes

agent:A can_update process:P

pre:

agent:A can_use service:S AND

service:S offers capability:can_update(process:P)

process:P can_update process:P'

pre:

process:P provides service:S AND

service:S offers capability:can_update(process:P')

Capability of accessing channels

process:P can_access channel:C

pre:

channel:C traverses host:H AND

process:P runs_on host:H

OR

channel:C traverses network:N AND

host:H is_connected_to network:N AND

process:P runs_on host:H

human:U can_access channel:C

pre:

channel:C traverses host:H AND

human:U can_attach_to host:H

OR

channel:C traverses network:N AND

human:U can_attach_to network:N

Capability of accessing storages

process:P can_access storage:S

pre:

storage:S resides_on host:H AND

process:P runs_on host:H

human:U can_access storage:S

pre:

storage:S resides_on host:H AND

human:U can_attach_to host:H

Capability of reading data

agent:A can_read datum:D

pre:

agent:A can_read datum:D' AND
datum:D' is_a_copy_of datum:D
OR
agent:A can_use services:S AND
service:S offers capability:can_read(datum:D)
OR
agent:A can_receive_on channel:C AND
channel:C carries datum:D
OR
agent:A can_retrieve_from storage:S AND
storage:S stores datum:D

process:P can_read datum:D

pre:

process:P provides service:S AND
service:S offers capability:can_read(datum:D)

Capability of writing data

agent:A can_write datum:D

pre:

agent:A can_write datum:D' AND
datum:D is_a_function_of datum:D'
OR
agent:A can_use services:S AND
service:S offers capability:can_write(datum:D)
OR
agent:A can_send_on channel:C AND
channel:C carries datum:D
OR

agent:A can_store_in storage:S AND
storage:S stores datum:D

process:P can_write datum:D

pre:

process:P provides service:S AND
service:S offers capability:can_write(datum:D)

Capability of obtaining credentials

agent:A can_obtain credential:Cr

pre:

agent:A has credential:Cr
OR
datum:D is_a_copy_of credential:Cr AND
agent:A can_read datum:D

General rules only for the ATTACKER

Obtaining capabilities from controlled agents

Capability of accessing services

ATTACKER can_access service:S

pre:

agent:A can_access service:S AND
ATTACKER can_control agent:A

Capability of using services

ATTACKER can_use service:S

pre:

agent:A can_use service:S AND

ATTACKER can_control agent:A

Capability of performing operations

ATTACKER can_perform operation:O

pre:

agent:A can_perform operation:O AND

ATTACKER can_control agent:A

Capability of obtaining privileges

ATTACKER can_obtain privilege:Pr

pre:

agent:A can_obtain privilege:Pr AND

ATTACKER can_control agent:A

Capability of obtaining credentials

ATTACKER can_obtain credential:Cr

pre:

agent:A can_obtain credential:Cr AND

ATTACKER can_control agent:A

OR

storage:S stores credential:Cr AND

ATTACKER can_store_in storage:S // overwrite with own credential

OR

datum:D is_a_function_of credential:Cr AND // e.g., password hash

storage:S stores datum:D AND

ATTACKER can_store_in storage:S // overwrite password hash

OR

ATTACKER can_access service:S AND

privilege:S.use requires credetial:Cr

ATTACKER can_guess credential:Cr (*) // login attempts

OR

datum:D is_a_function_of credential:Cr AND // e.g., password hash

storage:S stores datum:D AND

ATTACKER can_retrieve_from storage:S // read password hash

ATTACKER can_guess credential:Cr (*) // John the Ripper

Capability of attaching to networks and hosts

ATTACKER can_attach_to host:H

pre:

human:U can_attach_to host:H AND

ATTACKER can_control human:U (*)

ATTACKER can_attach_to network:N

pre:

human:U can_attach_to network:N AND

ATTACKER can_control human:U (*)

Capability of accessing channels

ATTACKER can_access channel:C

pre:

process:P can_access channel:C AND

ATTACKER can_control process:P

OR

human:U can_access channel:C AND

ATTACKER can_control human:U (*)

Capability of accessing storages

ATTACKER can_access storage:S

pre:

process:P can_access storage:S AND

ATTACKER can_control process:P

OR

human:U can_access storage:S AND

ATTACKER can_control human:U (*)

Achieving typical low-level attack goals

Attacks on processes

ATTACKER can_control process:P

pre:

ATTACKER can_update process:P

OR

process:P provides service:S AND

ATTACKER can_access service:S AND

ATTACKER can_inject_code_into process:P through service:S (*)

OR

process:P runs_on host:H AND

ATTACKER can_fully_control host:H

ATTACKER can_corrupt process:P

pre:

ATTACKER can_update process:P

OR

process:P provides service:S AND

ATTACKER can_access service:S AND

ATTACKER can_inject_code_into process:P through service:S (*)

OR

process:P relies_on service:S AND

ATTACKER can_alter service:S

OR

process:P relies_on datum:D AND

ATTACKER can_write datum:D

OR

process:P runs_on host:H AND

ATTACKER can_fully_control host:H

ATTACKER can_disable process:P

pre:

ATTACKER can_stop process:P

OR

process:P provides service:S AND

ATTACKER can_access service:S AND

ATTACKER can_crash process:P through service:S (*)

OR

process:P relies_on service:S AND

ATTACKER can_make_unavailable service:S

OR

process:P relies_on datum:D AND

ATTACKER can_make_unavailable datum:D

OR

process:P runs_on host:H AND

ATTACKER can_fully_control host:H

ATTACKER can_exhaust process:P

pre:

process:P provides service:S AND

ATTACKER can_overload process:P through service:S (*)

OR

process:P runs_on host:H AND

ATTACKER can_partially_control host:H

OR

process:P runs_on host:H AND

ATTACKER can_fully_control host:H

Attacks on hosts

ATTACKER can_fully_control host:H

pre:

process:P runs_on host:H AND

ATTACKER can_control process:P AND

ATTACKER can_gain_full_control_over host:H through process:P (*)

OR

process:P runs_on host:H AND

process:P provides service:S AND

ATTACKER can_access service:S AND

ATTACKER can_gain_full_control_over host:H through service:S (*)

ATTACKER can_partially_control host:H

pre:

ATTACKER can_obtain_privilege:H.user_U

OR

process:P runs_on host:H AND

ATTACKER can_control process:P

OR

process:P runs_on host:H AND

process:P provides service:S AND

ATTACKER can_access service:S AND

ATTACKER can_gain_partial_control_over host:H through service:S (*)

Achieving typical high-level attack goals

Attacks on data

// for "ATTACKER can_read datum:D"

// see rules for human agents to read data

ATTACKER can read datum:data

pre:

storage:S stores datum:data

storage:S resides_on host:H

ATTACKER can_fully_control host:H

// for "ATTACKER can_write datum:D"

// see rules for human agents to write data

ATTACKER can_write datum:D

pre:

storage:S stores datum:D

storage:S resides_on host:H

ATTACKER can_fully_control host:H

ATTACKER can_make_unavailable datum:D

pre:

ATTACKER can_write datum:D // e.g., make unavailable by encrypting it

OR

storage:S stores datum:D AND

storage:S resides_on host:H AND

ATTACKER can_fully_control host:H

OR

channel:C carries datum:D AND

channel:C traverses host:H AND

ATTACKER can_fully_control host:H

Attacks on services

// for "ATTACKER can_use service:S"

// see rules for agents to use a service

ATTACKER can_alter service:S

pre:

process:P provides service:S AND

ATTACKER can_control process:P

OR

process:P provides service:S AND

ATTACKER can_corrupt process:P

ATTACKER can_make_unavailable service:S

pre:

ATTACKER can_access service:S AND

ATTACKER can_overload service:S (*)

OR

process:P provides service:S AND

ATTACKER can_control process:P

OR

process:P provides service:S AND

ATTACKER can_disable process:P

OR

process:P provides service:S AND

ATTACKER can_exhaust process:P

OR

process:P provides service:S AND

process:P relies_on service:S' AND

ATTACKER can_make_unavailable service:S'

OR

process:P provides service:S AND

process:P relies_on datum:D AND

ATTACKER can_make_unavailable datum:D

=====

Testable low-level attack goals

ATTACKER can_control human:U (*)

// e.g., bribing, extortion, insider threat, ...

// this will be just a theoretical possibility, we will not test it, of course

process:P provides service:S AND

ATTACKER can_access service:S AND

ATTACKER can_crash process:P through service:S (*)

// we can rely on the results of the component tester (CT)

// e.g., fuzzing leads to crashing

process:P provides service:S AND

ATTACKER can_access service:S AND

ATTACKER can_inject_code_into process:P through service:S (*)

// we can rely on the results of the component tester (CT)

// e.g., fuzzing leads to crashing

process:P provides service:S AND

ATTACKER can_overload process:P through service:S (*)

// attacker overloads process:P, such that it cannot provide its services (not only service:S)

ATTACKER can_access service:S AND

ATTACKER can_overload service:S (*)

// attacker overloads only service:S, such that others cannot use it

process:P runs_on host:H AND

ATTACKER can_control process:P AND

ATTACKER can_gain_full_control_over host:H through process:P (*)

// the attacker already fully controls a process

// can he break out and obtain full control over the host?

// i.e., essentially privilege escalation:

// how to become root when you already have some user level privileges?

process:P runs_on host:H AND

process:P provides service:S AND

ATTACKER can_access service:S AND

ATTACKER can_gain_full_control_over host:H through service:S (*)

// same as above, but the attacker is in a weaker position:

// he does not control any process yet, but can only access a process

// through one of its services

process:P runs_on host:H AND

process:P provides service:S AND

ATTACKER can_access service:S AND

ATTACKER can_gain_partial_control_over host:H through service:S (*)

// same as above, but aiming only at user level control over the host

// e.g., command injection through a service of a process

ACM Classification

```
\begin{CCSXML}
<ccs2012>
  <concept>
    <concept_id>10002978.10002986.10002989</concept_id>
    <concept_desc>Security and privacy~Formal security models</concept_desc>
    <concept_significance>500</concept_significance>
  </concept>
  <concept>
    <concept_id>10002978.10003006</concept_id>
    <concept_desc>Security and privacy~Systems security</concept_desc>
    <concept_significance>300</concept_significance>
  </concept>
</ccs2012>
\end{CCSXML}

\ccsdesc[500]{Security and privacy~Formal security models}
\ccsdesc[300]{Security and privacy~Systems security}
```