

Suspicion-driven analysis of security requirements

Nuno Amálio

University of Luxembourg

Introduction

- Increasingly: need to engineer secure systems
- Traditionally: security treated as *after-thought*:
 - Security requirements treated as non-functional
 - and kept separate from functional reqs
 - But functionality impacts on security
- Security and systems engineering need to be unified
- Best way to do this still an open problem

Analysing security

- Sw verification techniques applicable to security
- Traditionally, emphasis is on:
 - **Safety**: something bad does not happen
 - **Liveness**: Something good must happen
- Security analysis needs to look for vulnerabilities
 - Find threats and windows of malicious opportunity

This Talk

- Two experiments:
 - focussing on **confidentiality** and **integrity**
- Proved to be **safety-secure**
- But are in fact **not secure**
- Suspicion-based analysis exposes vulnerabilities
- System requirements formalised in Event-Calculus
- Formal analysis conducted using planning

Event-Calculus

- Based on 1st order predicate calculus
- Enables reasoning about action and change
- EC's basic elements are:
 - Events: actions that occur in the world
 - fluents: time-varying property of world
 - timepoints: instances of time

Event-Calculus

- Based on 1st order predicate calculus
- Enables reasoning about action and change
- EC's basic elements are:
 - Events: actions that occur in the world
 - fluents: time-varying property of world
 - timepoints: instances of time

Basic Predicates:

$$\textit{HoldsAt} (f, t)$$

Fluent f holds at timepoint t

Event-Calculus

- Based on 1st order predicate calculus
- Enables reasoning about action and change
- EC's basic elements are:
 - Events: actions that occur in the world
 - fluents: time-varying property of world
 - timepoints: instances of time

Basic Predicates:

Happens (e, t)

Event e may occur at timepoint t

Event-Calculus

- Based on 1st order predicate calculus
- Enables reasoning about action and change
- EC's basic elements are:
 - Events: actions that occur in the world
 - fluents: time-varying property of world
 - timepoints: instances of time

Basic Predicates:

Initiates (e, f, t)

If event e occurs at timepoint t , f is true after t

Event-Calculus

- Based on 1st order predicate calculus
- Enables reasoning about action and change
- EC's basic elements are:
 - Events: actions that occur in the world
 - fluents: time-varying property of world
 - timepoints: instances of time

Basic Predicates:

Terminates (e, f, t)

If event e occurs at timepoint t , f is false after t

Event-Calculus

- Based on 1st order predicate calculus
- Enables reasoning about action and change
- EC's basic elements are:
 - Events: actions that occur in the world
 - fluents: time-varying property of world
 - timepoints: instances of time

Basic Predicates:

Initially (f)

f is true at timepoint 0.

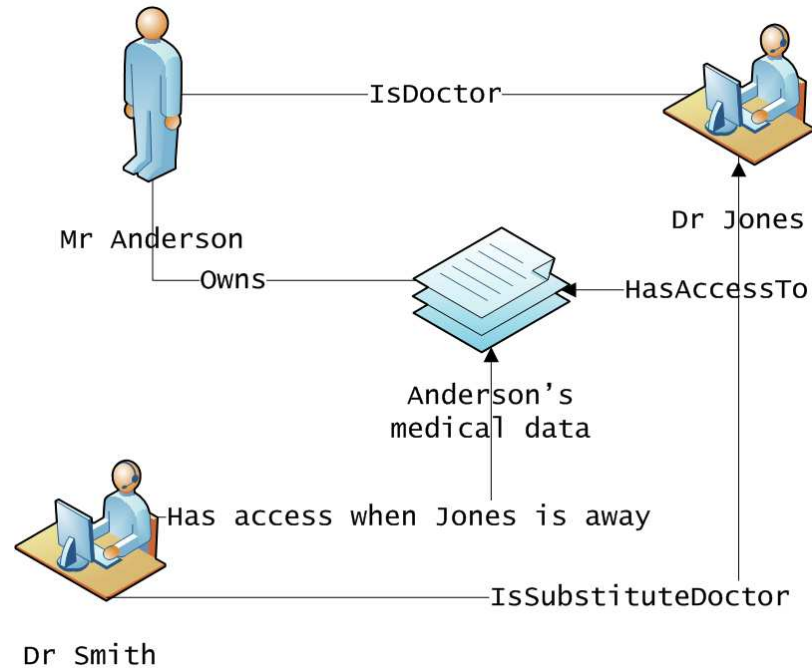
Security analysis based on planning

- Here, analysis is study of reachability using planning
- Framework of Event-Calculus with tool-support
- Planning builds plans to achieve goal
- A plan is a system trace
- Analysis uses two types of goals:
 - **security-analysis** (safety-analysis)
 - **suspicious** (what-happens-if)

Confidentiality Example

R1	Doctors must be able to access their patient's medical data to provide effective medical care.
R2	A doctor may nominate a substitute doctor who may be able to access the patient's medical data only when the main doctor is on leave.

Confidentiality Example



EC Model

$$\begin{aligned} & \forall d : Doctor; p : Patient; t : Time \mid \\ & \quad HoldsAt (CanAccessMD (d, p), t) \\ & \quad \Leftrightarrow HoldsAt (IsDoctorOf (d, p), t) \\ & \quad \vee ((\exists d' : Doctor) HoldsAt (IsDoctorOf (d', p), t) \\ & \quad \quad \wedge HoldsAt (IsSubstituteDoctor (d, d'), t) \\ & \quad \quad \wedge HoldsAt (OnLeave (d'), t)) \end{aligned} \tag{1}$$

EC Model

$$\begin{aligned} & \forall d : Doctor; p : Patient; t : Time \mid \\ & \quad HoldsAt (CanAccessMD (d, p), t) \\ & \quad \Leftrightarrow HoldsAt (IsDoctorOf (d, p), t) \\ & \quad \vee ((\exists d' : Doctor) HoldsAt (IsDoctorOf (d', p), t) \\ & \quad \quad \wedge HoldsAt (IsSubstituteDoctor (d, d'), t) \\ & \quad \quad \wedge HoldsAt (OnLeave (d'), t)) \end{aligned} \tag{1}$$

$$\begin{aligned} & \forall d : Doctor; p : Patient; t : Time \mid \\ & \quad HoldsAt (CanAccessMD (d, p), t) \\ & \quad \Rightarrow Initiates (AuthoriseAccess (d, p), CredentialMD (d, p, t), t) \end{aligned} \tag{2}$$

EC Model

$$\begin{aligned} &\forall d : Doctor; p : Patient; t : Time \mid \\ &\quad HoldsAt (CanAccessMD (d, p), t) \\ &\quad \Rightarrow Initiates (AuthoriseAccess (d, p), CredentialMD (d, p, t), t) \end{aligned} \quad (2)$$

$$\begin{aligned} &\forall d : Doctor; p : Patient; t : Time \mid \\ &\quad HoldsAt (HasValidCredential (d, p), t) \\ &\quad \Leftrightarrow (\exists t_2 : Time) HoldsAt (CredentialMD (d, p, t_2), t) \wedge (t_2 + 3) \geq t \end{aligned} \quad (3)$$

EC Model

$$\begin{aligned} &\forall d : Doctor; p : Patient; t : Time \mid \\ &\quad HoldsAt (CanAccessMD (d, p), t) \\ &\quad \Rightarrow Initiates (AuthoriseAccess (d, p), CredentialMD (d, p, t), t) \end{aligned} \quad (2)$$

$$\begin{aligned} &\forall d : Doctor; p : Patient; t : Time \mid \\ &\quad HoldsAt (HasValidCredential (d, p), t) \\ &\quad \Leftrightarrow (\exists t_2 : Time) HoldsAt (CredentialMD (d, p, t_2), t) \wedge (t_2 + 3) \geq t \end{aligned} \quad (3)$$

$$\begin{aligned} &\forall d : Doctor; p : Patient; t : Time \mid \\ &\quad Happens (GetMD(d, p), t) \Rightarrow HoldsAt (HasValidCredential (d, p), t) \end{aligned} \quad (4)$$

EC Model

$\forall d : Doctor; p : Patient; t : Time \mid$

$HoldsAt (HasValidCredential (d, p), t)$

$\Leftrightarrow (\exists t_2 : Time) HoldsAt (CredentialMD (d, p, t_2), t) \wedge (t_2 + 3) \geq t \quad (3)$

$\forall d : Doctor; p : Patient; t : Time \mid$

$Happens (GetMD(d, p), t) \Rightarrow HoldsAt (HasValidCredential (d, p), t) \quad (4)$

$\forall d : Doctor; p : Patient; t : Time \mid$

$Initiates (GetMD (d, p), ExposedToAt (d, p, t), t) \quad (5)$

EC Model

$\forall d : Doctor; p : Patient; t : Time \mid$

$$Happens (GetMD(d, p), t) \Rightarrow HoldsAt (HasValidCredential (d, p), t) \quad (4)$$

$\forall d : Doctor; p : Patient; t : Time \mid$

$$Initiates (GetMD (d, p), ExposedToAt (d, p, t), t) \quad (5)$$

$\forall a : Agent, d_1, d_2 : Doctor; t : Time \mid$

$$Initiates (SetSubstituteDoctor (a, d_1, d_2), IsSubstituteDoctor (d_2, d_1), t) \quad (6)$$

EC Model

$\forall d : Doctor; p : Patient; t : Time \mid$

$Initiates (GetMD (d, p), ExposedToAt (d, p, t), t)$ (5)

$\forall a : Agent, d_1, d_2 : Doctor; t : Time \mid$

$Initiates (SetSubstituteDoctor (a, d_1, d_2), IsSubstituteDoctor (d_2, d_1), t)$ (6)

$\forall a : Agent, d : Doctor; t : Time \mid$

$Initiates (SetDoctorOnLeave (a, d), OnLeave (d), t)$ (7)

EC Model

$\forall a : Agent, d_1, d_2 : Doctor; t : Time \mid$

$Initiates (SetSubstituteDoctor (a, d_1, d_2), IsSubstituteDoctor (d_2, d_1), t) \quad (6)$

$\forall a : Agent, d : Doctor; t : Time \mid$

$Initiates (SetDoctorOnLeave (a, d), OnLeave (d), t) \quad (7)$

$\forall a : Agent, d : Doctor; t : Time \mid$

$Terminates (DoctorNoLongerOnLeave (a, d), OnLeave (d), t) \quad (8)$

EC Model

$$\forall a : Agent, d : Doctor; t : Time |$$
$$\textit{Initiates} (\textit{SetDoctorOnLeave} (a, d), \textit{OnLeave} (d), t) \quad (7)$$

$$\forall a : Agent, d : Doctor; t : Time |$$
$$\textit{Terminates} (\textit{DoctorNoLongerOnLeave} (a, d), \textit{OnLeave} (d), t) \quad (8)$$

$$\forall d : Doctor; p : Patient | \textit{Initially} (\neg \textit{ExposedTo} (d, p)) \quad (9)$$

$$\forall d_1, d_2 : Doctor | \textit{Initially} (\neg \textit{IsSubstituteDoctor} (d_1, d_2)) \quad (10)$$

$$\forall d : Doctor | \textit{Initially} (\neg \textit{OnLeave} (d)) \quad (11)$$

$$\textit{Initially} (\textit{IsDoctorOf} (jones, anderson)) \quad (12)$$

Security violations

Can confidentiality of medical data be breached directly?

$$\begin{aligned} &\exists d : Doctor; p : Patient; t_1, t_2 : Time \mid \\ &\quad HoldsAt (ExposedToAt (d, p, t_2), t_1) \\ &\quad \wedge \neg HoldsAt (HasValidCredential (d, p), t_2) \end{aligned}$$

Security violations

Can confidentiality of medical data be breached directly?

$$\begin{aligned} &\exists d : Doctor; p : Patient; t_1, t_2 : Time \mid \\ &\quad HoldsAt (ExposedToAt (d, p, t_2), t_1) \\ &\quad \wedge \neg HoldsAt (HasValidCredential (d, p), t_2) \end{aligned}$$

- Tool does not find plans satisfying goal
- No state that directly violates security property

Suspicion analysis

It is suspicious that a doctor that is not the patient's doctor accesses medical file

$$\begin{aligned} &\exists d : Doctor; p : Patient; t_1, t_2 : Time \mid \\ &\quad HoldsAt (ExposedToAt (d, p, t_2), t_1) \\ &\quad \wedge \neg HoldsAt (IsDoctorOf (d, p), t_2) \end{aligned}$$

Suspicion analysis

It is suspicious that a doctor that is not the patient's doctor accesses medical file

$$\begin{aligned} \exists d : Doctor; p : Patient; t_1, t_2 : Time \mid \\ HoldsAt (ExposedToAt (d, p, t_2), t_1) \\ \wedge \neg HoldsAt (IsDoctorOf (d, p), t_2) \end{aligned}$$

Generated plans now give interesting scenarios:

- Some doctor can set himself as substitute of other doctor
- Some doctor can say that some other is on-leave.

Suspicion analysis

It is suspicious that a doctor that is not the patient's doctor accesses medical file

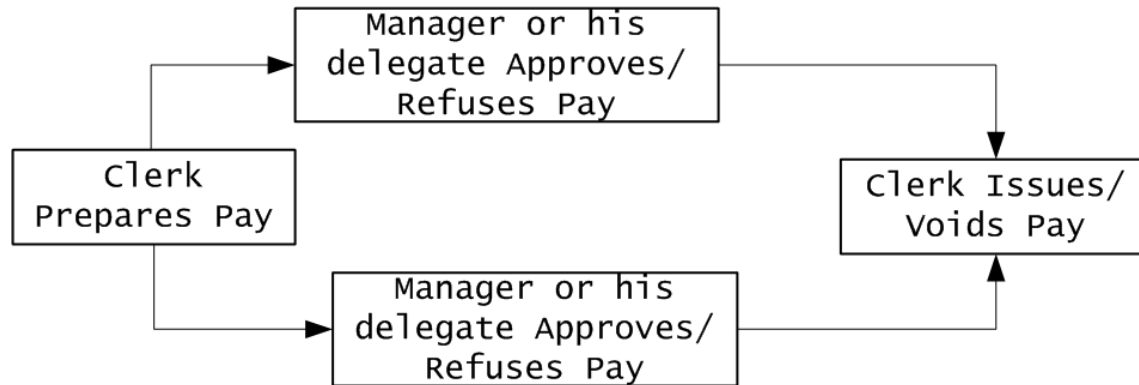
$$\begin{aligned} \exists d : Doctor; p : Patient; t_1, t_2 : Time \mid \\ HoldsAt (ExposedToAt (d, p, t_2), t_1) \\ \wedge \neg HoldsAt (IsDoctorOf (d, p), t_2) \end{aligned}$$

Generated plans now give interesting scenarios:

- Some doctor can set himself as substitute of other doctor
- Some doctor can say that some other is on-leave.

Requirements elaborated to fix problem.

Integrity Example



Integrity Example

R1	There are two types of users clerks and managers. Managers can performs the tasks that usually the clerks do, but clerks should not usually perform manager's tasks (exception is delegation, below).
R2	Clerks are responsible for starting the refund procedure, and for issuing or cancelling the refund.
R3	The refund shall be issued by a clerk if approved by the managers, or cancelled otherwise.
R4	A refund must by approved by two different managers.
R5	A clerk shall not both prepare and issue or cancel a refund.
R6	Managers can delegate the authority on approval of refunds to one of their administrators.

Security violations

Is it possible to reach state where SoD is breached?

$\exists u : User; s : Session; t : Time \mid HoldsAt (BreaksSoD (u, s), t)$

Security violations

Is it possible to reach state where SoD is breached?

$\exists u : User; s : Session; t : Time \mid HoldsAt (BreaksSoD (u, s), t)$

- No plans can be found for this goal
- No state directly violates security property

Suspicion analysis (I)

It is suspicious to execute two tasks in workflow

$$\begin{aligned} &\exists u : User; s : Session; t : Time \mid \\ &\quad HoldsAt (IsWrkfComplete (s), t) \\ &\quad \wedge HoldsAt (HasExecutedTwoTasks (u, s), t) \end{aligned}$$

Suspicion analysis (I)

It is suspicious to execute two tasks in workflow

$$\begin{aligned} &\exists u : User; s : Session; t : Time \mid \\ &\quad HoldsAt (IsWrkfComplete (s), t) \\ &\quad \wedge HoldsAt (HasExecutedTwoTasks (u, s), t) \end{aligned}$$

We now get interesting scenarios

- Managers prepare payments and then approve them
- Managers can approve and then issue payment

Suspicion analysis (I)

It is suspicious to execute two tasks in workflow

$$\begin{aligned} &\exists u : User; s : Session; t : Time \mid \\ &\quad HoldsAt (IsWrkfComplete (s), t) \\ &\quad \wedge HoldsAt (HasExecutedTwoTasks (u, s), t) \end{aligned}$$

We now get interesting scenarios

- Managers prepare payments and then approve them
- Managers can approve and then issue payment

New requirement fixes problem:

- Users can play at most one rôle in workflow session

Suspicion analysis (II)

Delegation is legal, but suspicious. Let's investigate.

$$\begin{aligned} &\exists u : User; ta, : Task; s : Session; t : Time \mid \\ &\quad HoldsAt(IsWrkfComplete(s), t) \\ &\quad \wedge HoldsAt(ExecutedTaskAsDelegator(u, ta, s), t) \end{aligned}$$

Suspicion analysis (II)

Delegation is legal, but suspicious. Let's investigate.

$$\begin{aligned} &\exists u : User; ta, : Task; s : Session; t : Time \mid \\ &\quad HoldsAt(IsWrkfComplete(s), t) \\ &\quad \wedge HoldsAt(ExecutedTaskAsDelegator(u, ta, s), t) \end{aligned}$$

We now get interesting scenarios

- Admin approves payment on behalf of Manager and then manager issues payment

Suspicion analysis (II)

Delegation is legal, but suspicious. Let's investigate.

$$\begin{aligned} &\exists u : User; ta, : Task; s : Session; t : Time \mid \\ &\quad HoldsAt(IsWrkfComplete(s), t) \\ &\quad \wedge HoldsAt(ExecutedTaskAsDelegator(u, ta, s), t) \end{aligned}$$

We now get interesting scenarios

- Admin approves payment on behalf of Manager and then manager issues payment

New requirement fixes problem:

- Delegate may execute tasks on behalf of manager, but system must also record that manager executed those tasks.

Conclusions

- Important to find states violating security properties
- Also important to look for **vulnerabilities**
- Question **secure or not** without straightforward yes/no answer
- Rigid security proofs may mislead security analysts
- Here, search for vulnerabilities in suspicious space
- Attack scenarios generated automatically from goals
- Experiments confirm importance of unified approach to systems and security engineering