Simulated Penetration Testing: From "Dijkstra" to "Turing Test++"

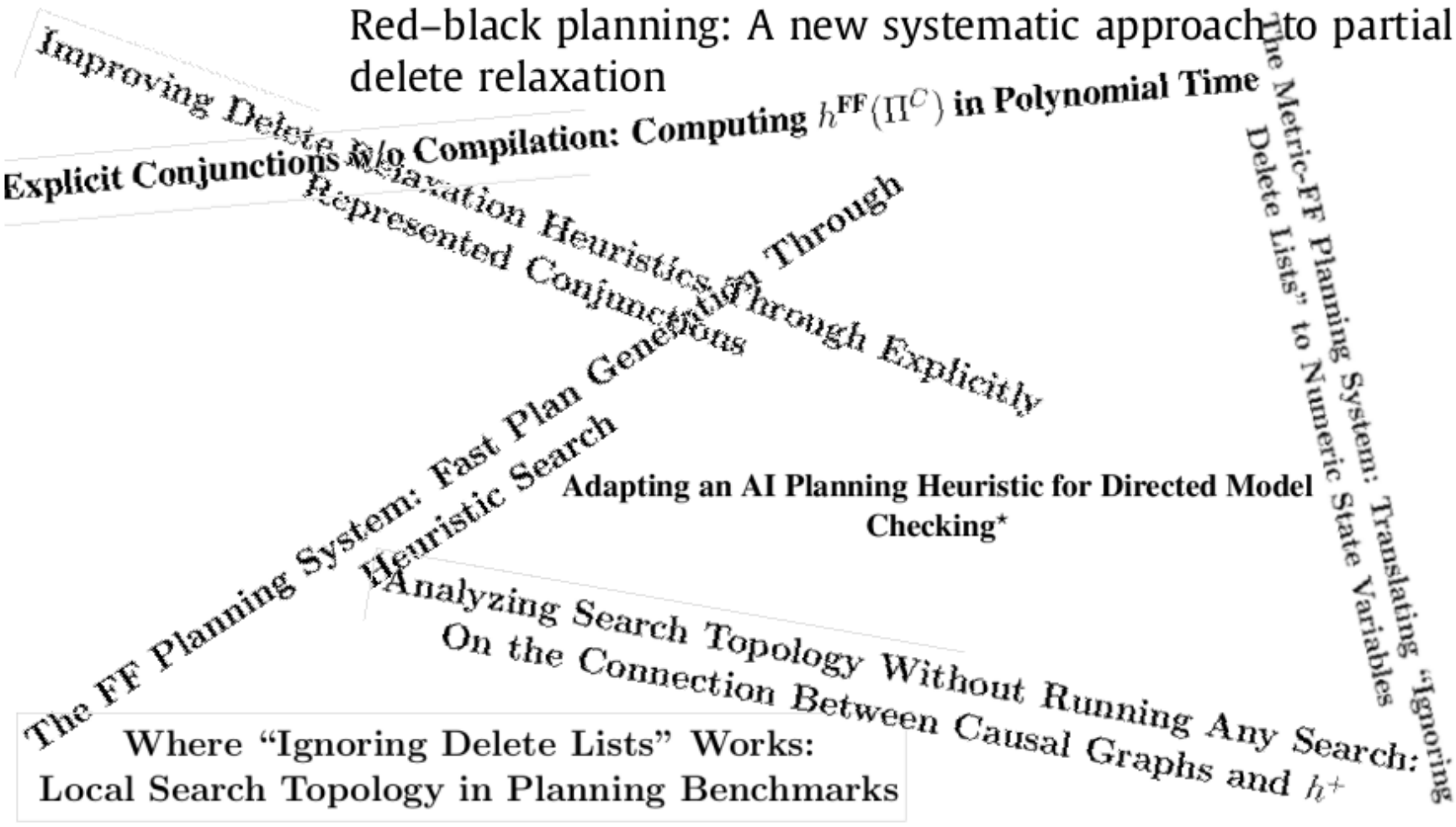
Jörg Hoffmann

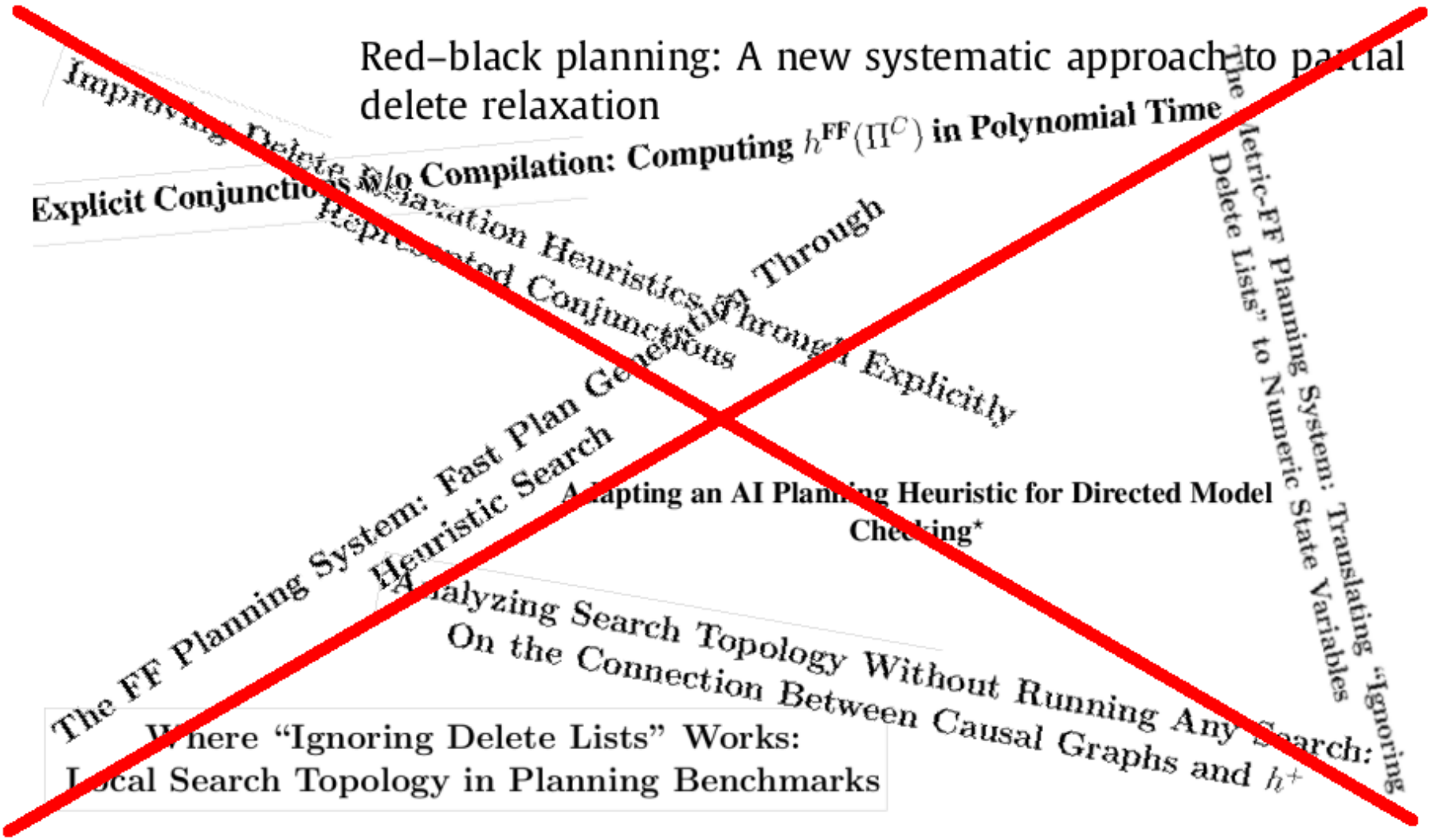
SAARLAND UNIVERSITY
COMPUTER SCIENCE

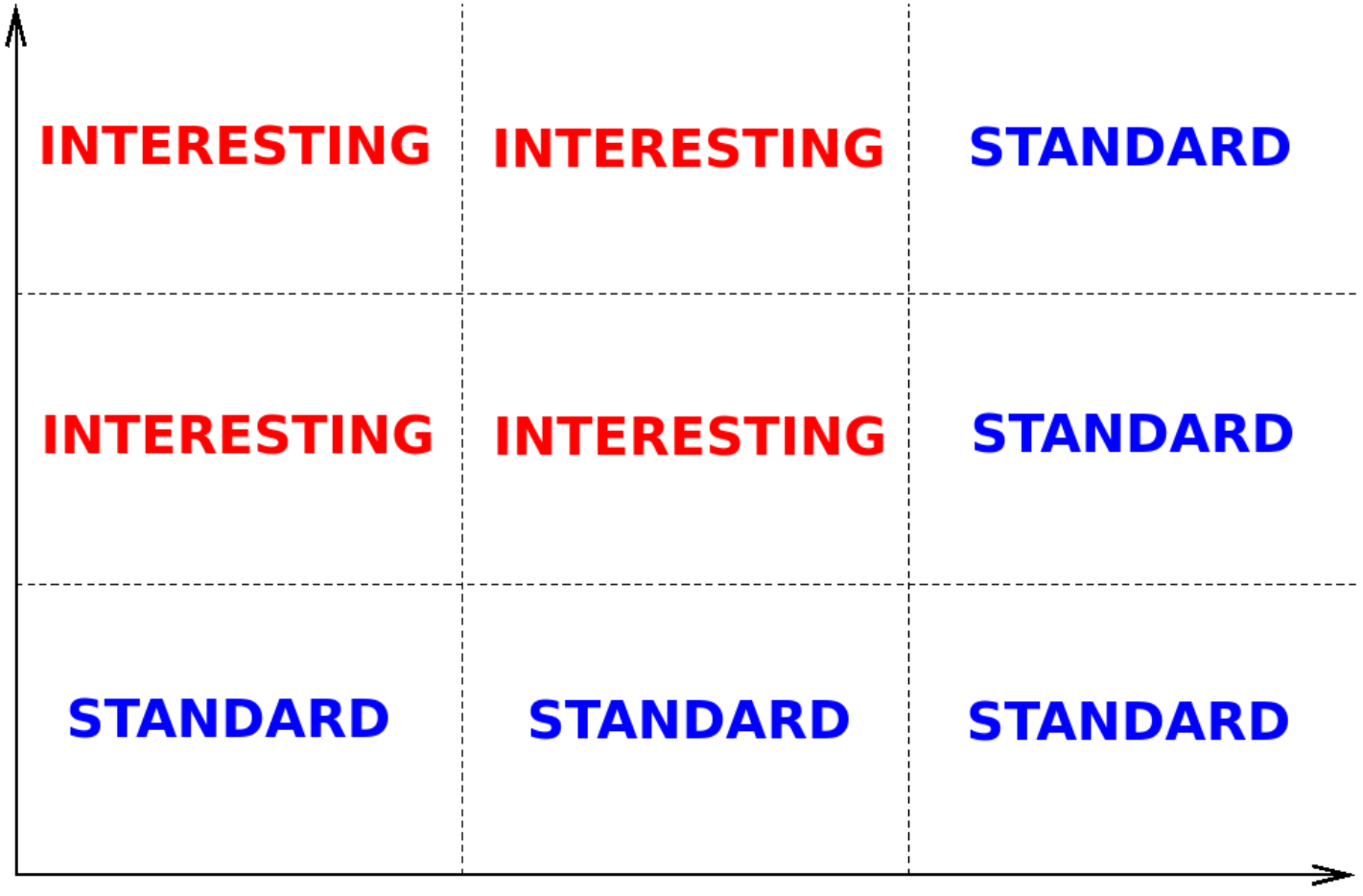
June 26, 2015


©Ron Leishman * illustrationsOf.com/439451


©Ron Leishman * illustrationsOf.com/43945


Red–black planning: A new systematic approach to partial delete relaxation
Improving Delete Relaxation Heuristics Through Explicit Conjunctions
Improving Delete Relaxation Heuristics Through Explicitly Represented Conjunctions
Explicit Conjunctions w/o Compilation: Computing $h^{FF}(\Pi^C)$ in Polynomial Time
The FF Planning System: Fast Plan Generation Through Heuristic Search
Adapting an AI Planning Heuristic for Directed Model Checking*
The Metric-FF Planning System: Translating "Ignoring Delete Lists" to Numeric State Variables
Analyzing Search Topology Without Running Any Search: On the Connection Between Causal Graphs and h^+
Where "Ignoring Delete Lists" Works: Local Search Topology in Planning Benchmarks

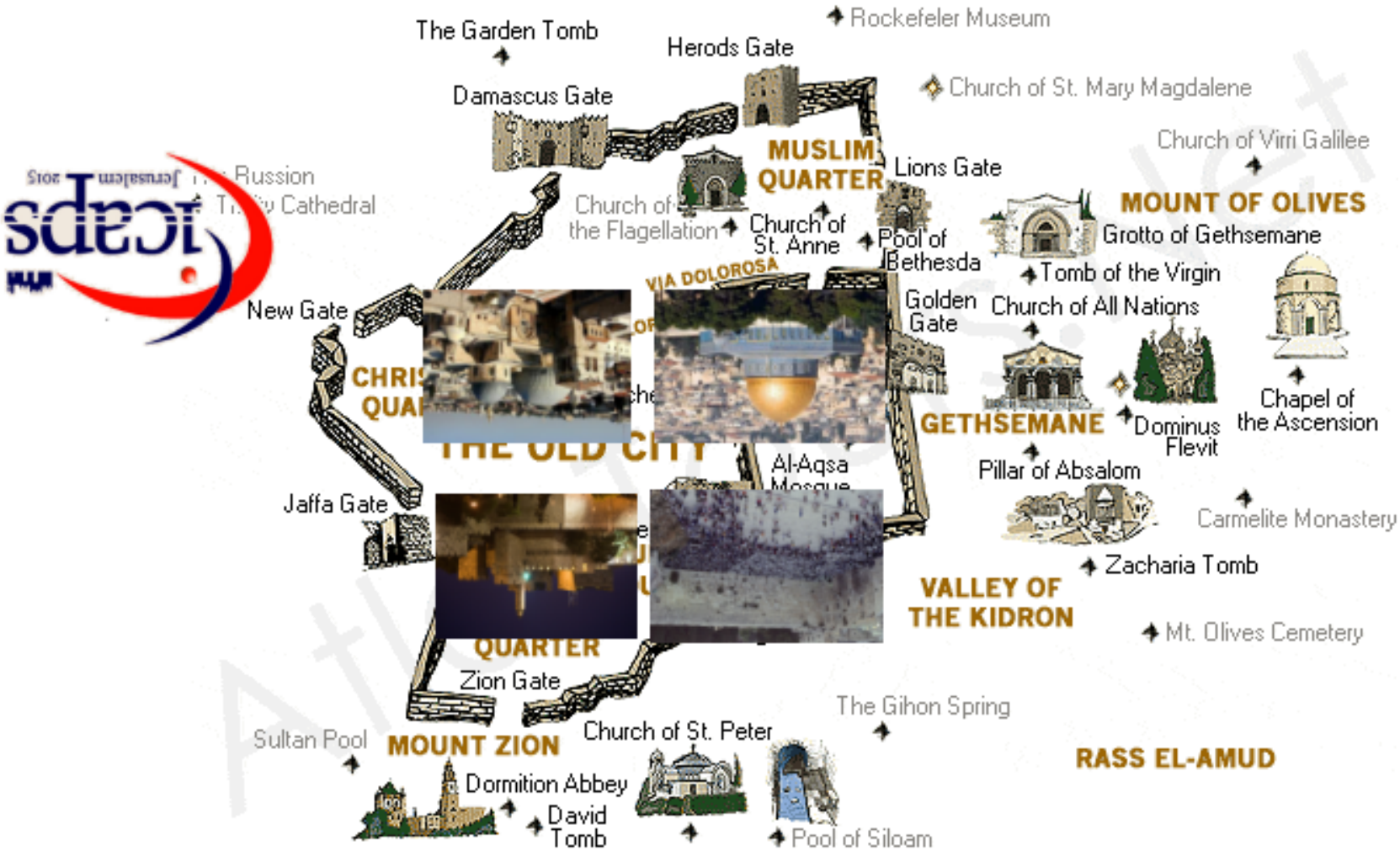

Red–black planning: A new systematic approach to partial delete relaxation
Improving Delete Relaxation Heuristics Through Explicit Conjunctions Represented Explicitly
Delete Relaxation Heuristics Through Explicit Conjunctions
Improving Delete Relaxation Heuristics Through Explicit Conjunctions
Explicit Conjunctions w/o Compilation: Computing $h^{FF}(\Pi^C)$ in Polynomial Time
The FF Planning System: Fast Plan Generation Through Heuristic Search
Adapting an AI Planning Heuristic for Directed Model Checking*
Analyzing Search Topology Without Running Any Search: On the Connection Between Causal Graphs and h^+
Where "Ignoring Delete Lists" Works: Local Search Topology in Planning Benchmarks
The Metric-FF Planning System: Translating "Ignoring Delete Lists" to Numeric State Variables



Details: See paper.

Details: See old town.

Details: See old town.



Simulated Penetration Testing: From "Dijkstra" to "Turing Test++"

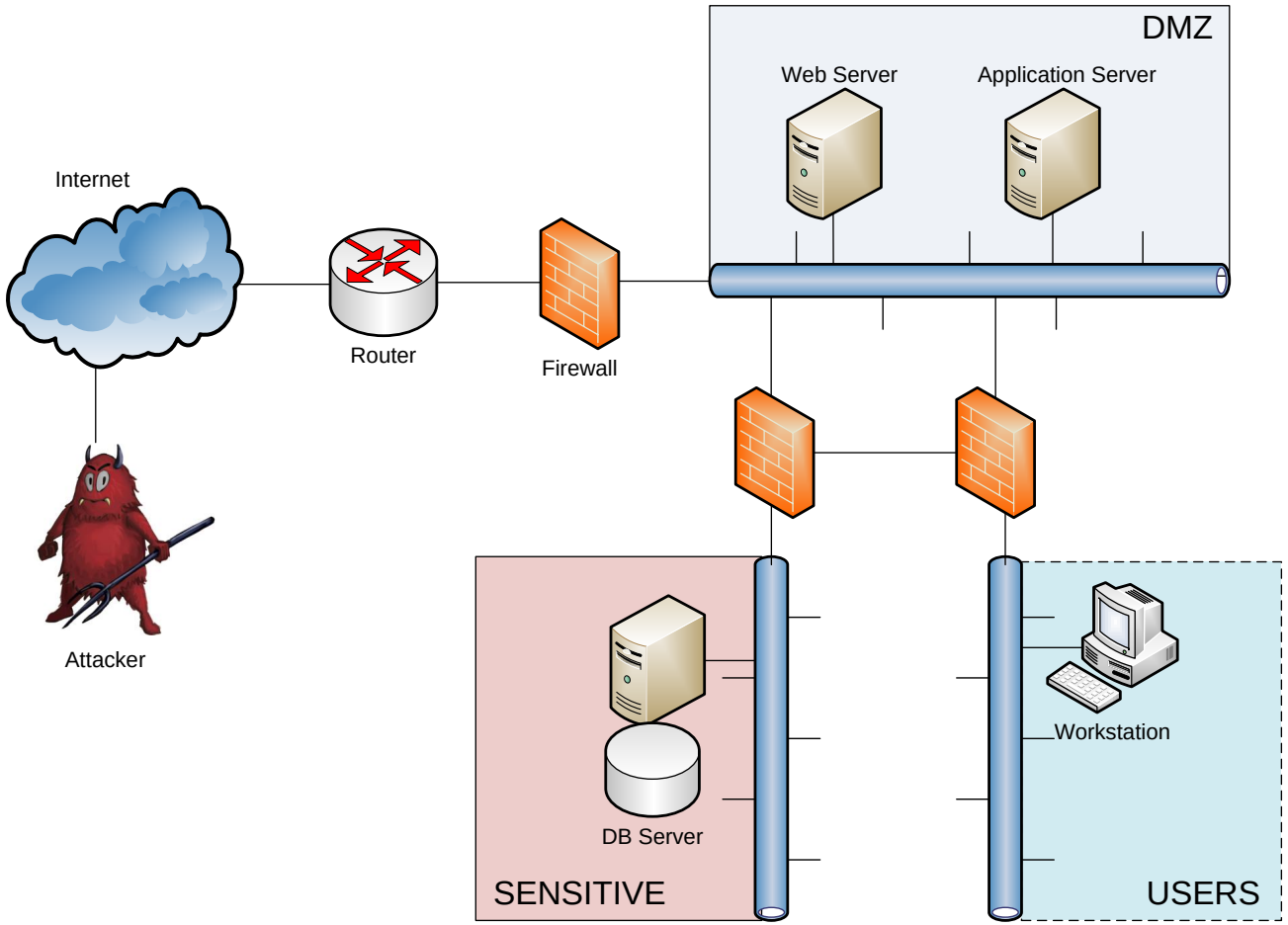
Jörg Hoffmann

SAARLAND UNIVERSITY
COMPUTER SCIENCE

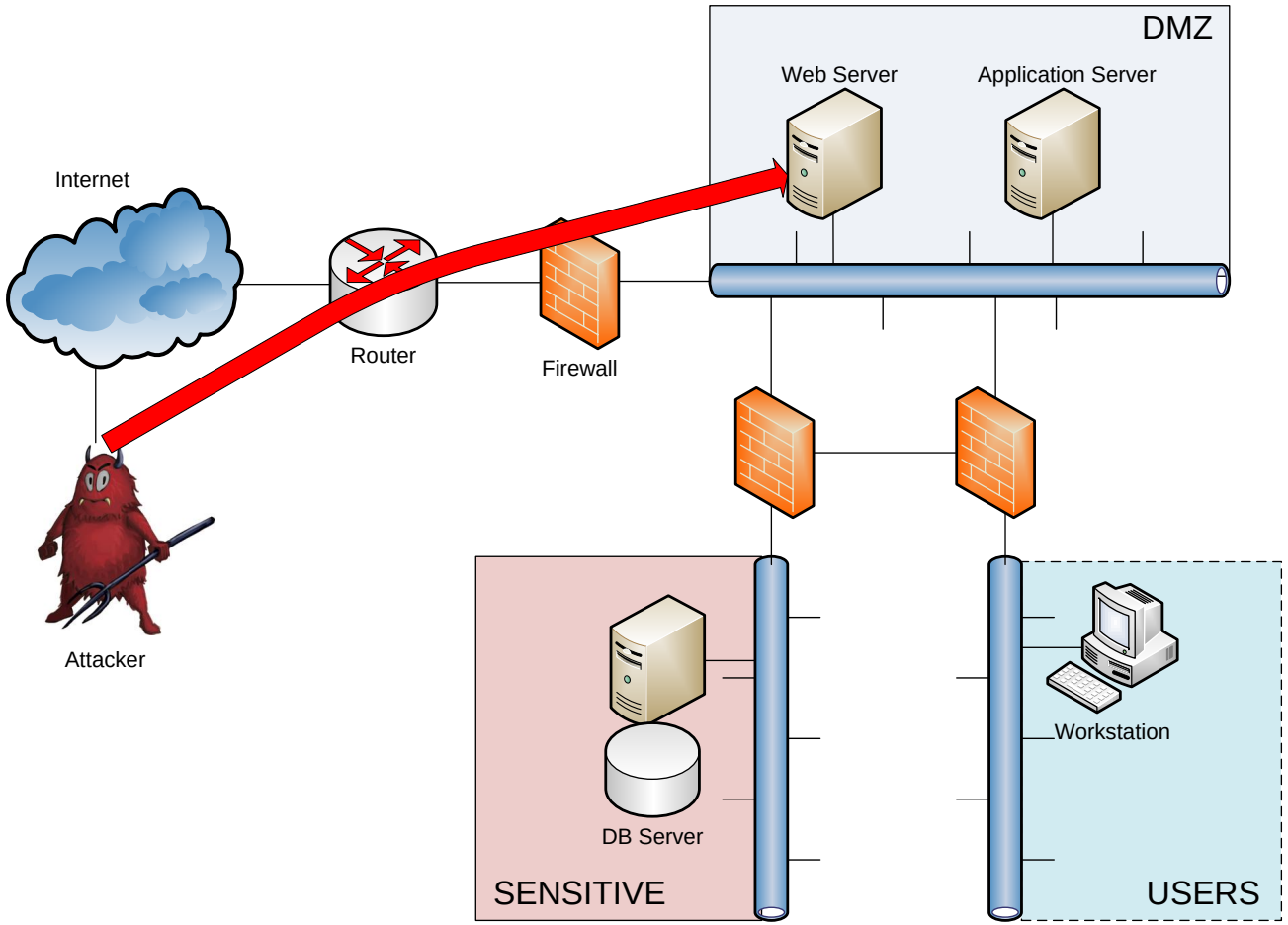
June 26, 2015

Agenda

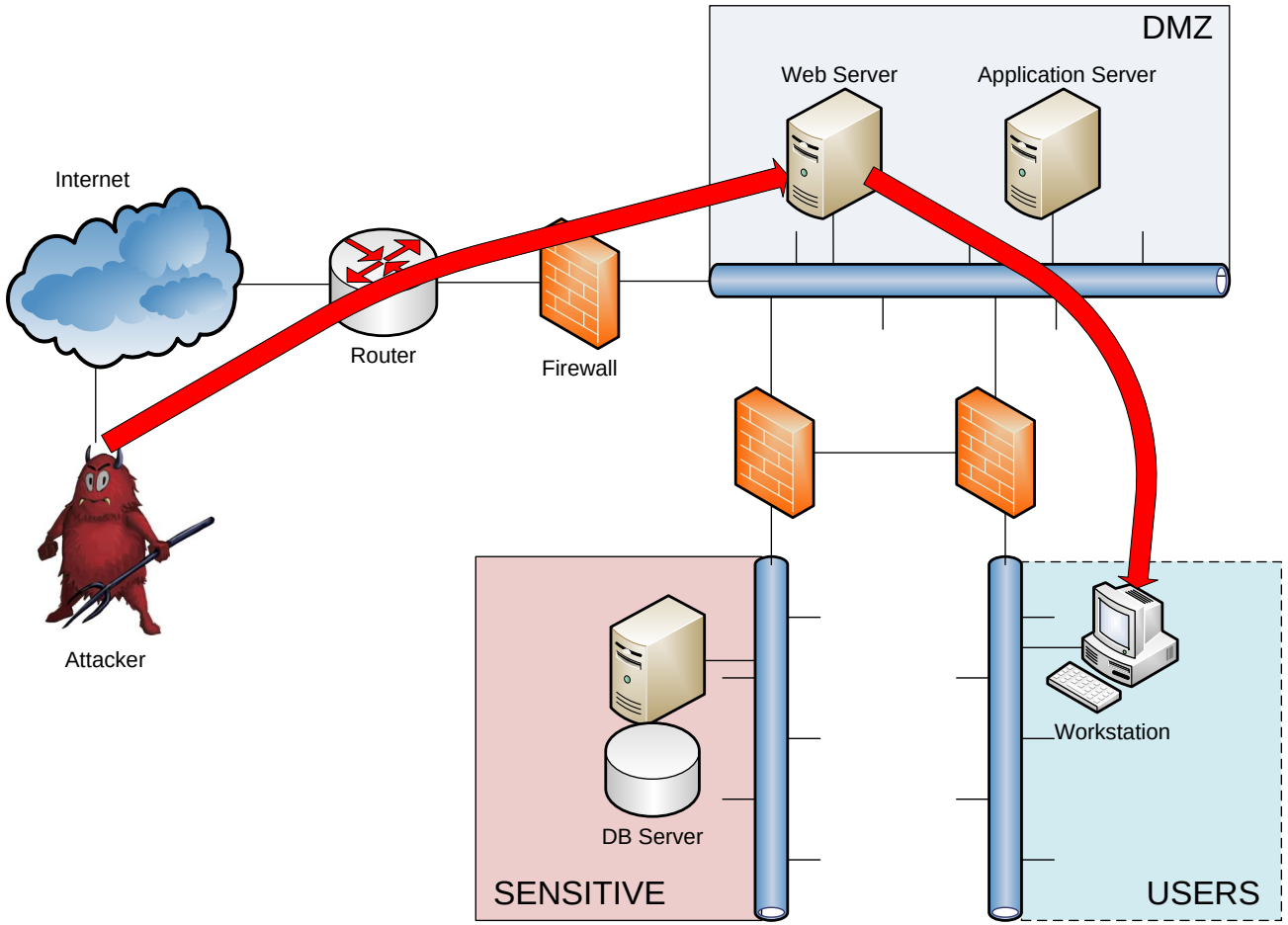
Network Hacking



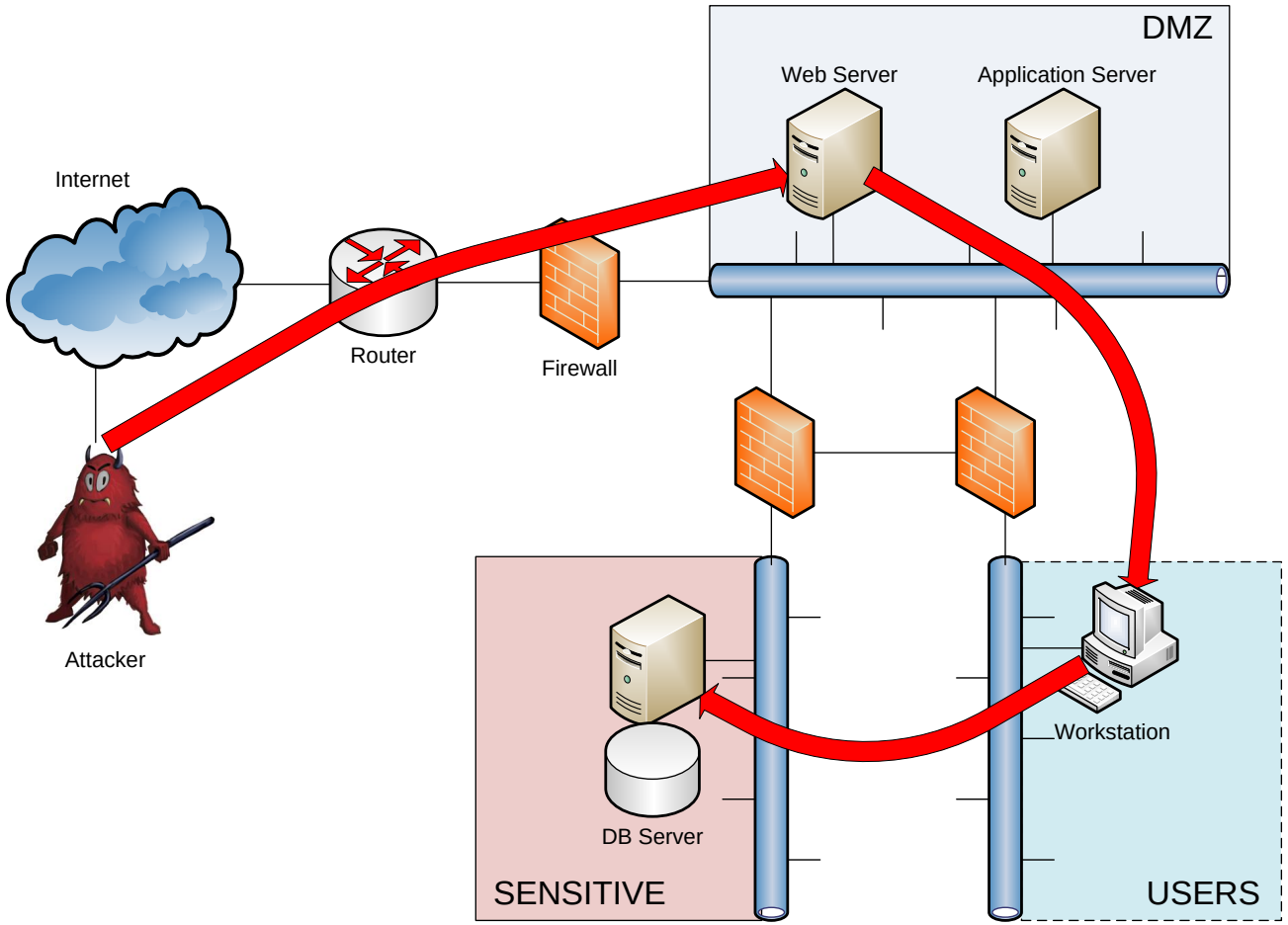
Network Hacking



Network Hacking



Network Hacking



Penetration Testing (Pentesting)

Pentesting

Actively verifying network defenses by conducting an intrusion in the same way an attacker would.

Penetration Testing (Pentesting)

Pentesting

Actively verifying network defenses by conducting an intrusion in the same way an attacker would.

- Well-established industry (roots back to the 60s).
- Points out specific dangerous attacks (as opposed to vulnerability scanners).

Penetration Testing (Pentesting)

Pentesting

Actively verifying network defenses by conducting an intrusion in the same way an attacker would.

- Well-established industry (roots back to the 60s).
- Points out specific dangerous attacks (as opposed to vulnerability scanners).
- Pentesting tools sold by security companies, like Core Security.

  → Core IMPACT (since 2001); Immunity Canvas (since 2002); Metasploit (since 2003).
- Run security checks launching exploits.

Penetration Testing (Pentesting)

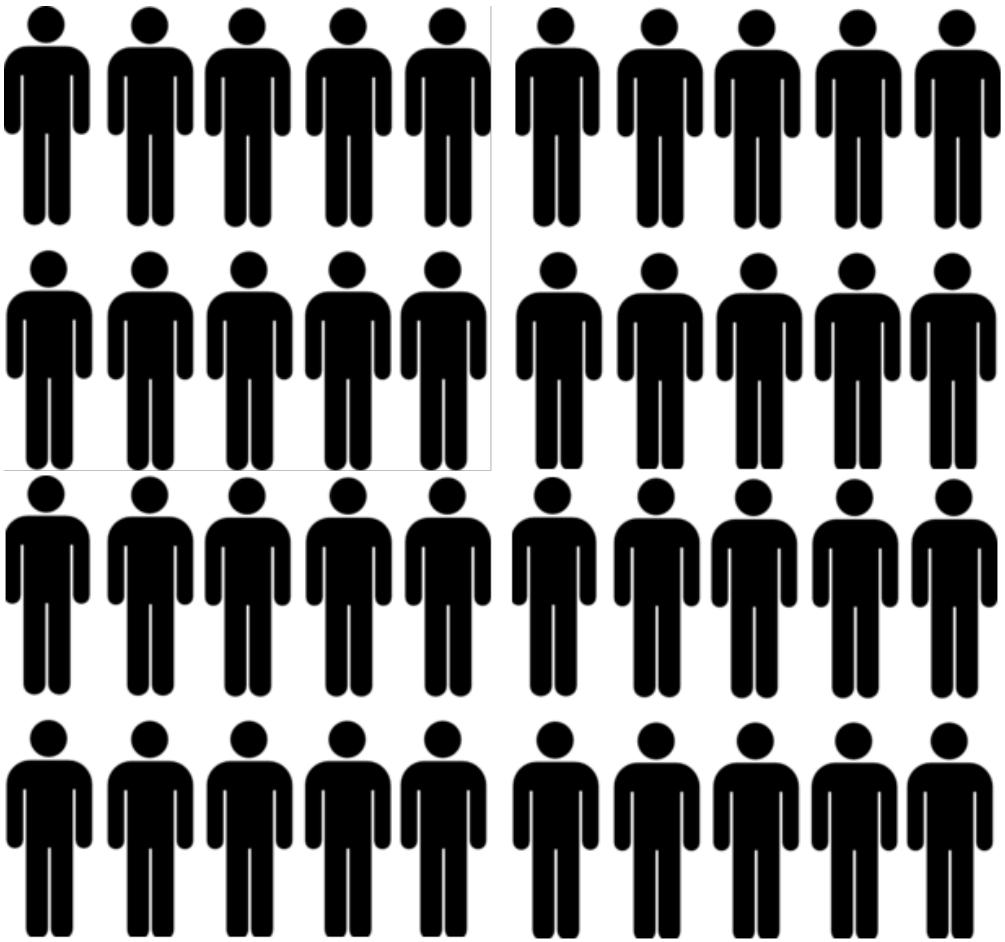
Pentesting

Actively verifying network defenses by conducting an intrusion in the same way an attacker would.

- Well-established industry (roots back to the 60s).
- Points out specific dangerous attacks (as opposed to vulnerability scanners).
- Pentesting tools sold by security companies, like Core Security.
  → Core IMPACT (since 2001); Immunity Canvas (since 2002); Metasploit (since 2003).
- Run security checks launching exploits.
- Core IMPACT uses Metric-FF for automation since 2010.

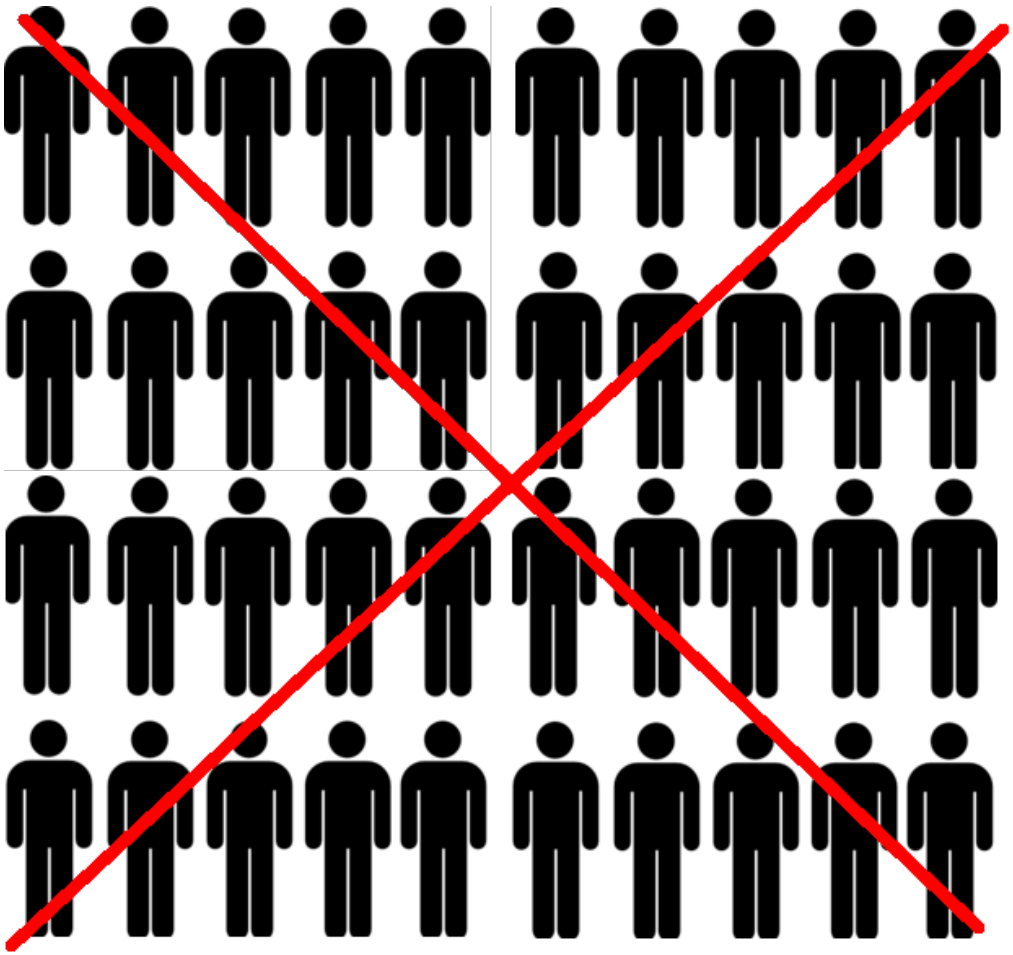
Automation

Security teams are typically small:



Automation

Security teams are typically small:



Automation

Increase testing coverage:

Automation

The security officer's "rat race":

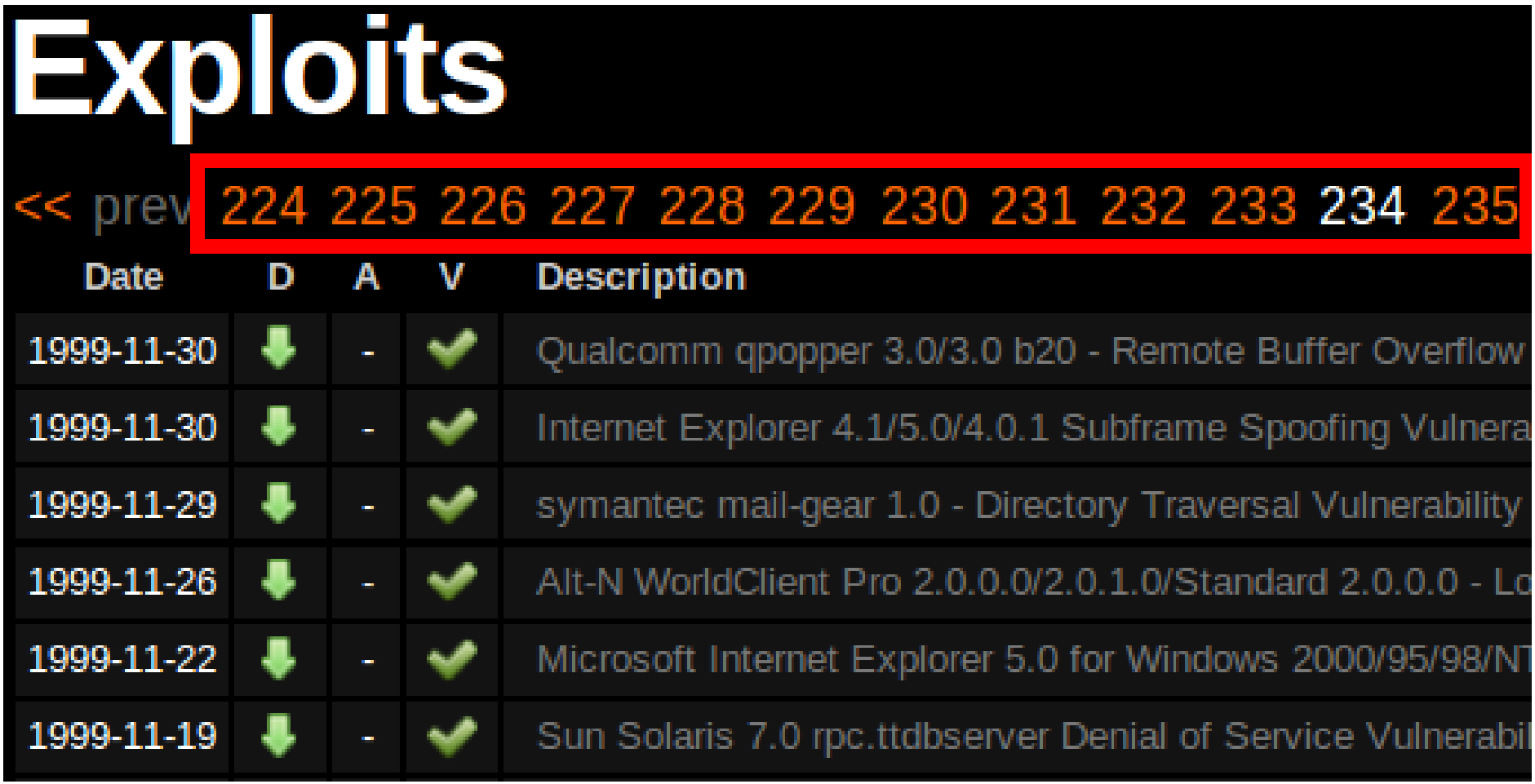
Exploits

<< prev 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 next >>

Date	D	A	V	Description	Plat.	Author
1999-11-30		-		Qualcomm qpopper 3.0/3.0 b20 - Remote Buffer Overflow Vulnerability (2)	unix	Synnergy Networks
1999-11-30		-		Internet Explorer 4.1/5.0/4.0.1 Subframe Spoofing Vulnerability	windows	Georgi Guninski
1999-11-29		-		symantec mail-gear 1.0 - Directory Traversal Vulnerability	multiple	Ussr Labs
1999-11-26		-		Alt-N WorldClient Pro 2.0.0.0/2.0.1.0/Standard 2.0.0.0 - Long URL DoS Vulnerability	windows	Ussr Labs
1999-11-22		-		Microsoft Internet Explorer 5.0 for Windows 2000/95/98/NT 4 XML HTTP Redirect Vulnerability	windows	Georgi Guninksi
1999-11-19		-		Sun Solaris 7.0 rpc.ttdbserver Denial of Service Vulnerability	solaris	Elias Levy
1999-11-19		-		Microsoft SQL Server 7.0/7.0 SP1 NULL Data DoS Vulnerability	windows	Kevork Belian
1999-11-18		-		Dick Lin ZetaMail 2.1 Login DoS Vulnerability	windows	Ussr Labs
1999-11-17		-		Tektronix Phaser Network Printer 740/750/750DP/840/930 PhaserLink Webserver Vulnerability	hardware	Dennis W. Mattiso.
1999-11-16		-		Matt Wright FormHandler.cgi 2.0 Reply Attachment Vulnerability	unix	m4rcyS
1999-11-15		-		Admiral Systems EmailClub 1.0.0.5 - Buffer Overflow Vulnerability	windows	UNYUN
1999-11-15		-		Antelope Software W4-Server 2.6 a/Win32 Cgitest.exe Buffer Overflow	windows	UNYUN
1999-11-15		-		International TeleCommunications WebBBS 2.13 login & password Buffer Overflow	windows	UNYUN
1999-11-14		-		Microsoft Internet Explorer 5.0 Media Player ActiveX Error Message Vulnerability	windows	Georgi Guninski
1999-11-13		-		ETL Delegate 5.9.x / 6.0.x - Buffer Overflow Vulnerabilities	linux	scut

Automation

The security officer's "rat race":



Automation

$\Longrightarrow$ **Simulated Pentesting:**

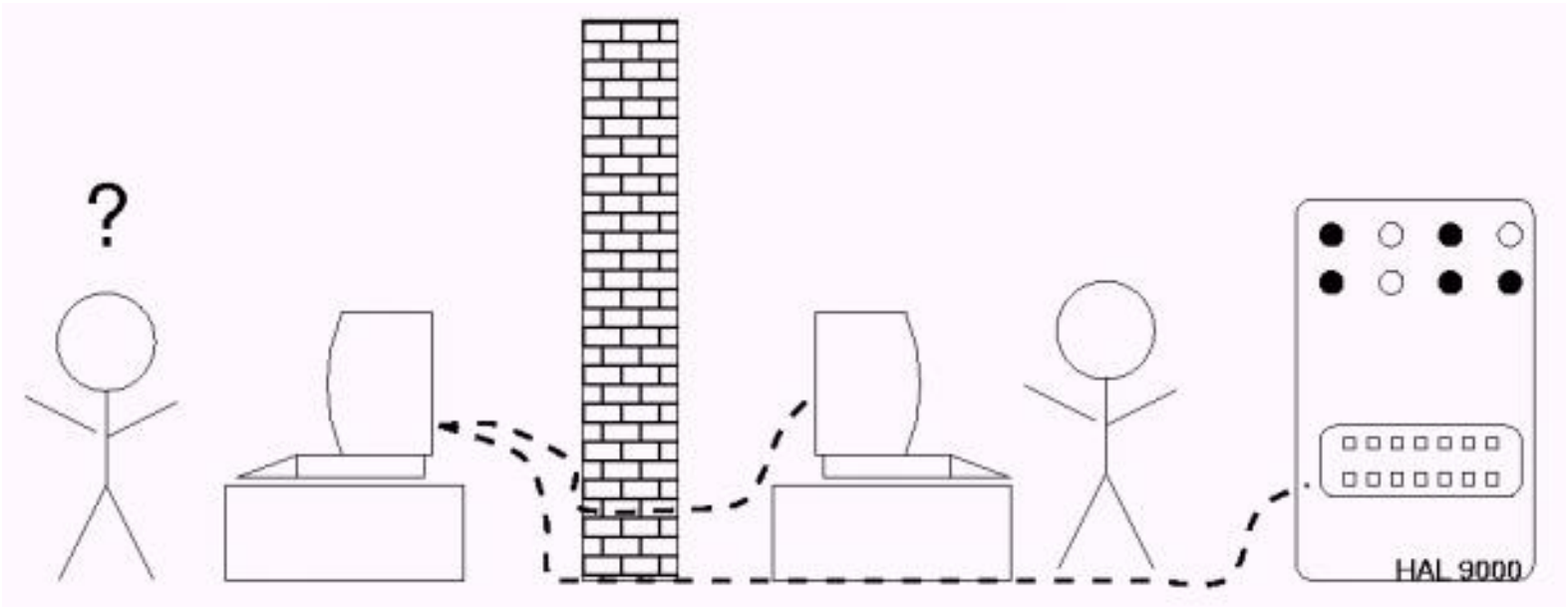
- Make a model of the network and exploits.
- Run attack planning on the model to simulate attacks.

Automation

$\Longrightarrow$ **Simulated Pentesting:**

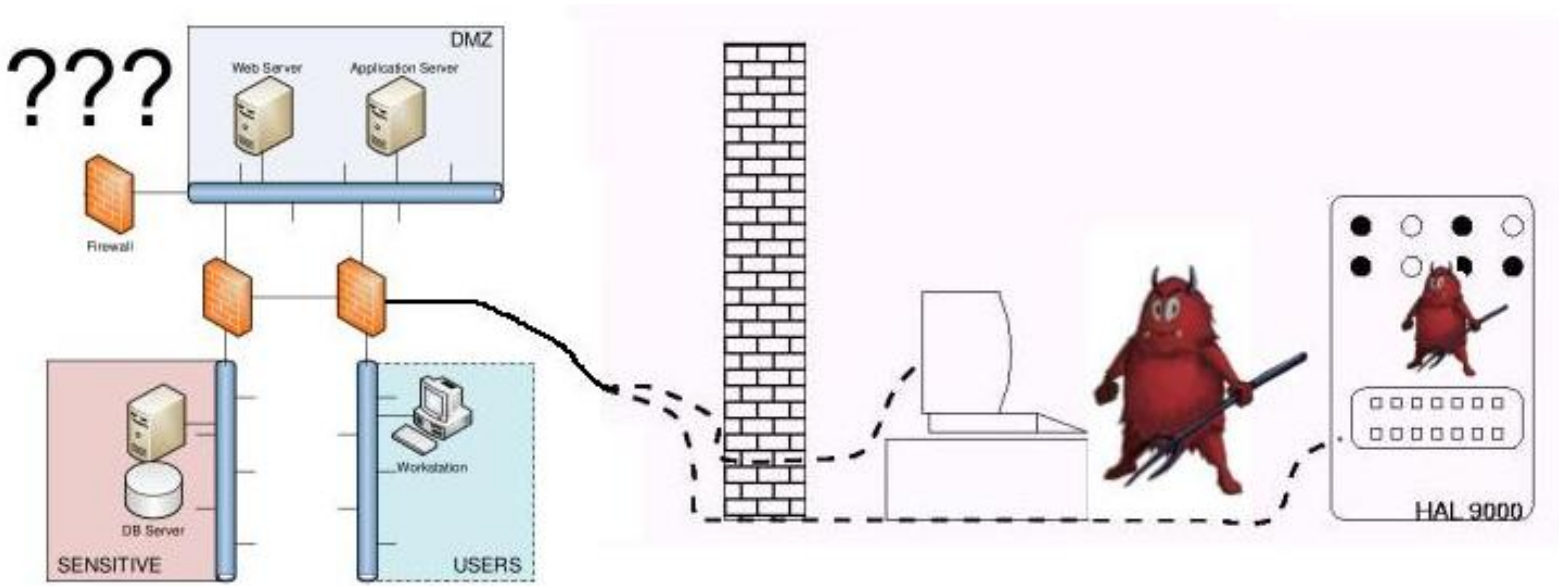
- Make a model of the network and exploits.
- Run attack planning on the model to simulate attacks.
- Running the rat race $\approx$ update the model, go drink a coffee.

The Turing Test



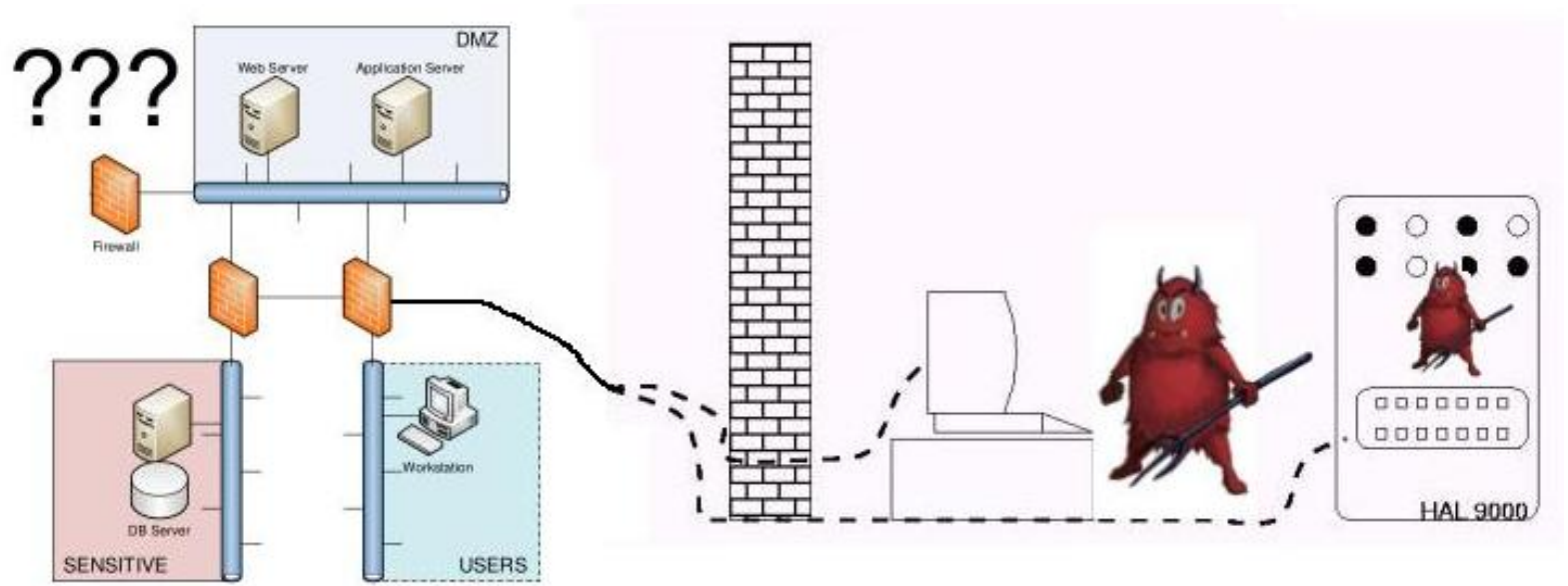
The Turing Test++: "Hacking, not Talking!"

Ultimate vision: realistically simulate a human hacker!



The Turing Test++: "Hacking, not Talking!"

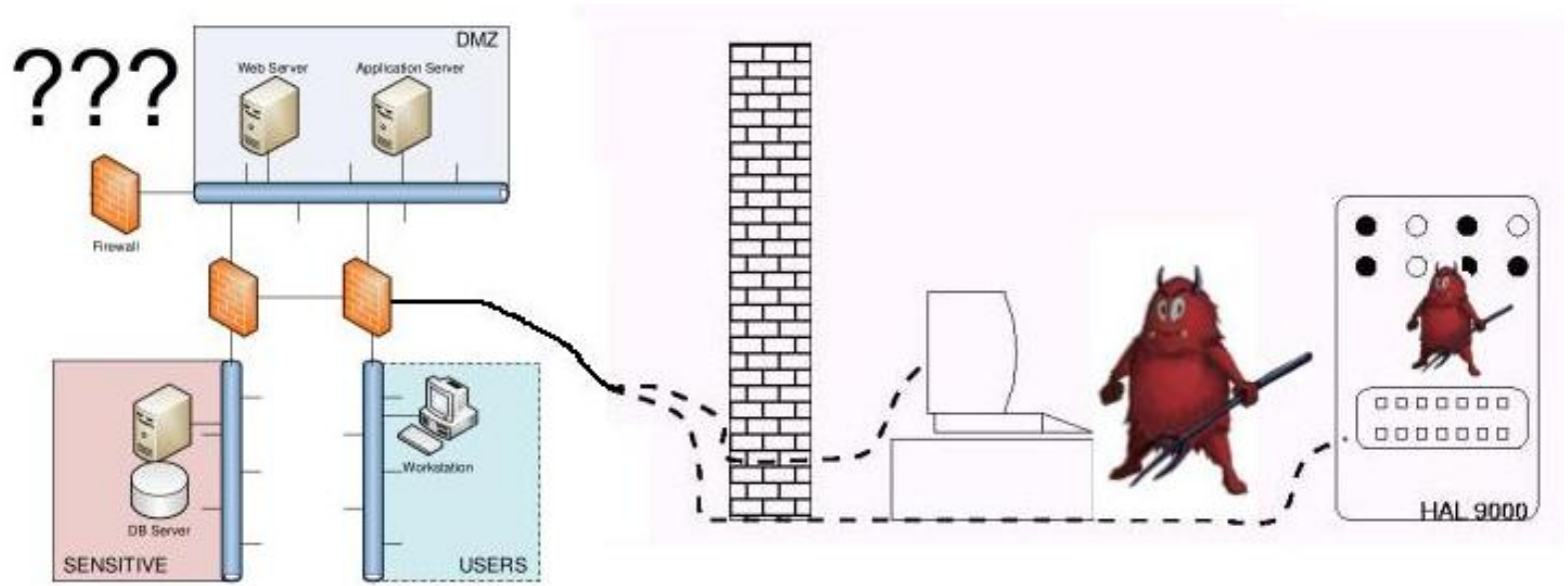
Ultimate vision: realistically simulate a human hacker!



- Yes hacking is more technical.

The Turing Test++: "Hacking, not Talking!"

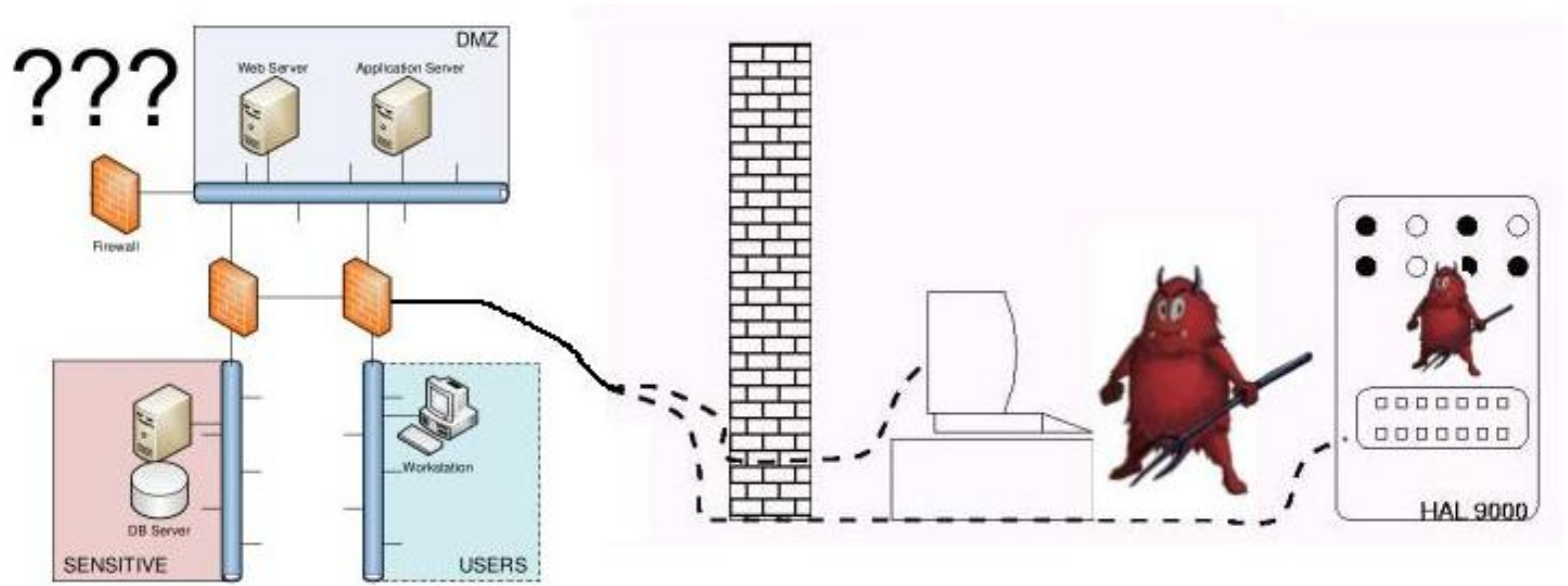
Ultimate vision: realistically simulate a human hacker!



- Yes hacking is more technical.
- However: socio-technical attacks, e. g. social network reconnaissance.

The Turing Test++: "Hacking, not Talking!"

Ultimate vision: realistically simulate a human hacker!



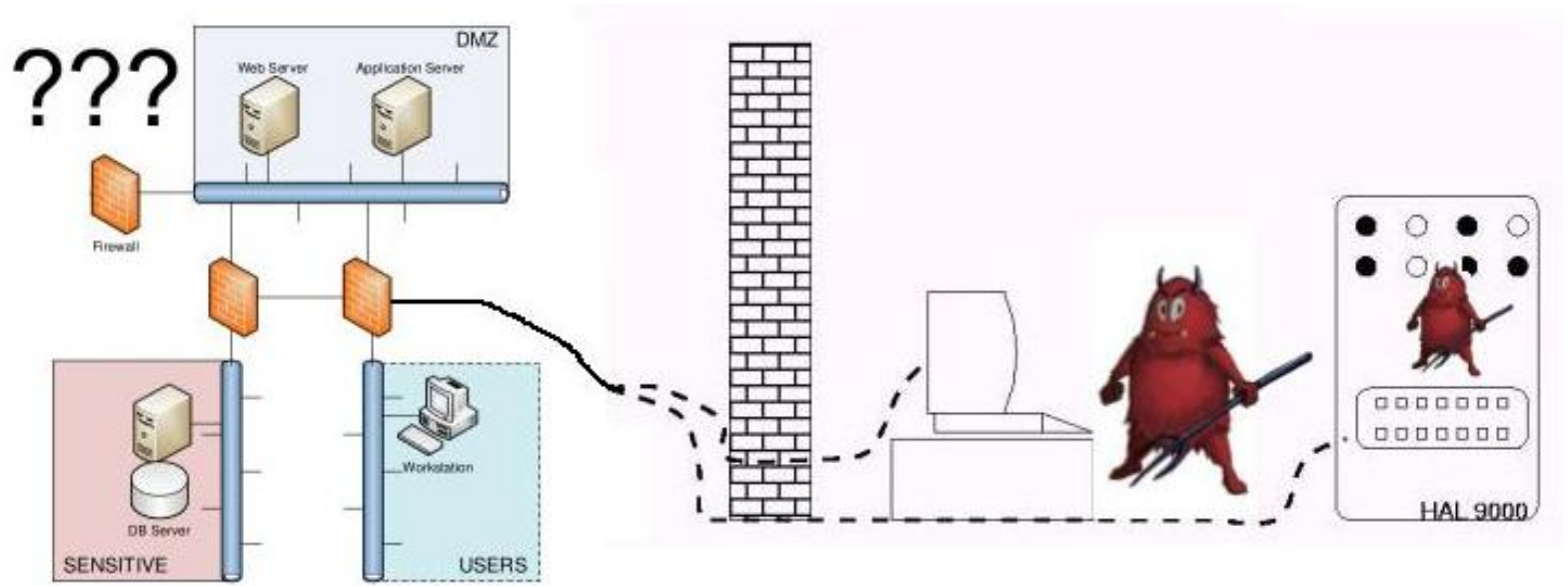
- Yes hacking is more technical.
- However: socio-technical attacks, e. g. social network reconnaissance.

  → Turing Test as a sub-problem of spying on people

.

The Turing Test++: "Hacking, not Talking!"

Ultimate vision: realistically simulate a human hacker!



- Yes hacking is more technical.
- However: socio-technical attacks, e. g. social network reconnaissance.

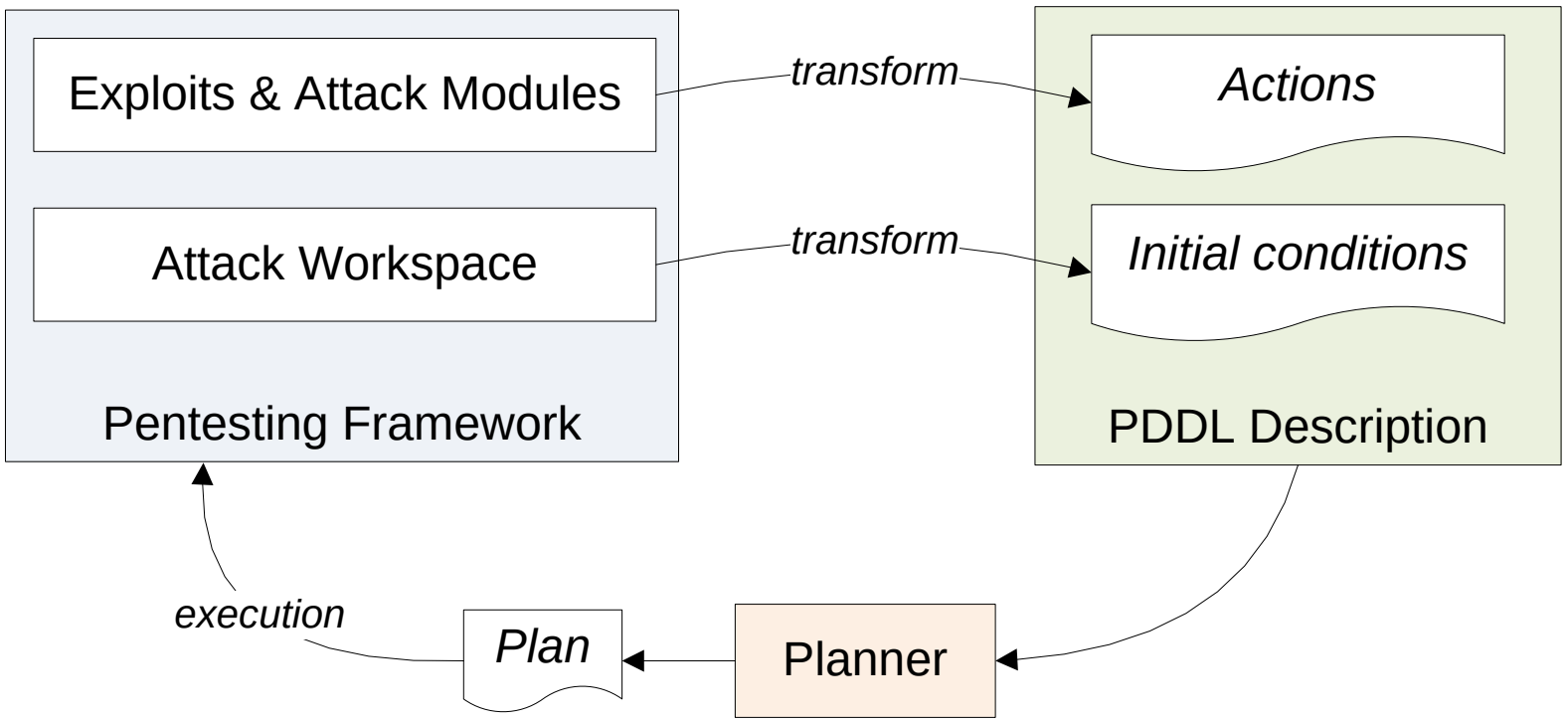
  → Turing Test as a sub-problem of spying on people (e. g. [Huber *et al.* (2009)]).

Agenda

Agenda

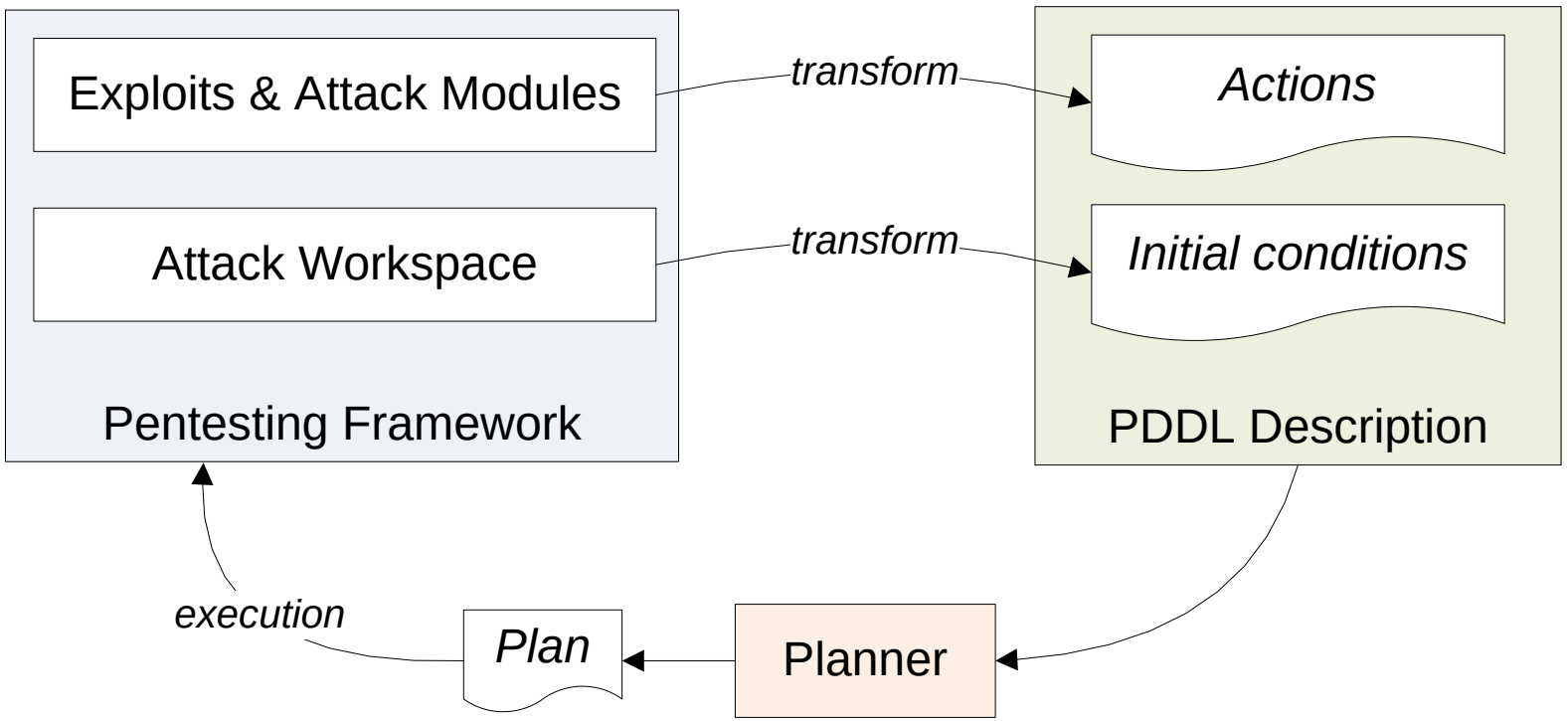
Simulated Pentesting at Core Security

Core IMPACT system architecture:



Simulated Pentesting at Core Security

Core IMPACT system architecture:



→ In practice, the attack plans are being used to point out to the security team where to look.

Simulated Pentesting at Core Security

"Point out to the security team where to look"

Simulated Pentesting at Core Security

"Point out to the security team where to look"

Classical Planning

Definition

A STRIPS planning task is a tuple $\langle \mathcal{P}, \mathcal{A}, s_0, G \rangle$:

- $\mathcal{P}$: set of facts (Boolean state variables).
- $\mathcal{A}$: set of actions a, each a tuple $\langle pre(a), add(a), del(a), c(a) \rangle$ of precondition, add list, delete list, and non-negative cost.
- s_0: initial state; G: goal.

Classical Planning

Definition

A STRIPS planning task is a tuple $\langle \mathcal{P}, \mathcal{A}, s_0, G \rangle$:

- $\mathcal{P}$: set of facts (Boolean state variables).
- $\mathcal{A}$: set of actions a, each a tuple $\langle pre(a), add(a), del(a), c(a) \rangle$ of precondition, add list, delete list, and non-negative cost.
- s_0: initial state; G: goal.

Definition

A STRIPS planning task's state space is a tuple $\langle \mathcal{S}, \mathcal{A}, T, s_0, S_G \rangle$:

- $\mathcal{S}$: set of all states; $\mathcal{A}$: actions as above.
- T: state transitions (s, a, s')
- s_0: initial state as above; S_G: goal states.

$\rightarrow$ Objective: Find cheapest path from s_0 to (a state in) S_G.

Core Security Attack Planning PDDL

Actions:

```
(:action HP_OpenView_Remote_Buffer_Overflow_Exploit
    :parameters (?s - host ?t - host)
    :precondition (and (compromised ?s)
        (connected ?s ?t)
        (has_OS ?t Windows)
        (has_OS_edition ?t Professional)
        (has_OS_servicepack ?t Sp2)
        (has_OS_version ?t WinXp)
        (has_architecture ?t I386)
        (has_service ?t ovtrcd))
    :effect (and (compromised ?t) (increase (time) 10)))
```

Core Security Attack Planning PDDL

Actions:

```
(:action HP_OpenView_Remote_Buffer_Overflow_Exploit
    :parameters (?s - host ?t - host)
    :precondition (and (compromised ?s)
        (connected ?s ?t)
        (has_OS ?t Windows)
        (has_OS_edition ?t Professional)
        (has_OS_servicepack ?t Sp2)
        (has_OS_version ?t WinXp)
        (has_architecture ?t I386)
        (has_service ?t ovtrcd))
    :effect (and (compromised ?t) (increase (time) 10)))
```

Core Security Attack Planning PDDL

Actions:

```
(:action HP_OpenView_Remote_Buffer_Overflow_Exploit
    :parameters (?s - host ?t - host)
    :precondition (and (compromised ?s)
        (connected ?s ?t)
        (has_OS ?t Windows)
        (has_OS_edition ?t Professional)
        (has_OS_servicepack ?t Sp2)
        (has_OS_version ?t WinXp)
        (has_architecture ?t I386)
        (has_service ?t ovtrcd))
    :effect (and (compromised ?t) (increase (time) 10)))
```

Core Security Attack Planning PDDL

Actions:

```
(:action HP_OpenView_Remote_Buffer_Overflow_Exploit
    :parameters (?s - host ?t - host)
    :precondition (and (compromised ?s)
        (connected ?s ?t)
        (has_OS ?t Windows)
        (has_OS_edition ?t Professional)
        (has_OS_servicepack ?t Sp2)
        (has_OS_version ?t WinXp)
        (has_architecture ?t I386)
        (has_service ?t ovtrcd))
    :effect (and (compromised ?t) (increase (time) 10)))
```

Core Security Attack Planning PDDL

Actions:

```
(:action HP_OpenView_Remote_Buffer_Overflow_Exploit
    :parameters (?s - host ?t - host)
    :precondition (and (compromised ?s)
        (connected ?s ?t)
        (has_OS ?t Windows)
        (has_OS_edition ?t Professional)
        (has_OS_servicepack ?t Sp2)
        (has_OS_version ?t WinXp)
        (has_architecture ?t I386)
        (has_service ?t ovtrcd))
    :effect (and (compromised ?t) (increase (time) 10)))
```

Action cost:

- Average execution time.
- Success statistic against hosts with the same/similar observable configuration parameters.

Core Security Attack Planning PDDL, ctd.

Actions:

```
(:action HP_OpenView_Remote_Buffer_Overflow_Exploit
    :parameters (?s - host ?t - host)
    :precondition (and (compromised ?s)
        (connected ?s ?t)
        (has_OS ?t Windows)
        (has_OS_edition ?t Professional)
        (has_OS_servicepack ?t Sp2)
        (has_OS_version ?t WinXp)
        (has_architecture ?t I386)
        (has_service ?t ovtrcd))
    :effect (and (compromised ?t) (increase (time) 10)))
```

Initial state:

- "connected" predicates: network graph.
- "has_*" predicates: host configurations.
- One compromised host: models the internet.

Goal: Compromise one or several goal hosts.

Remarks

History:

- Planning domain "of this kind" (less IT-level, including also physical actions like talking to somebody) first proposed by [Boddy *et al.* (2005)]; used as benchmark in IPC'08 and IPC'11.

Remarks

History:

- Planning domain "of this kind" (less IT-level, including also physical actions like talking to somebody) first proposed by [Boddy *et al.* (2005)]; used as benchmark in IPC'08 and IPC'11.
- Presented encoding proposed by [Lucangeli *et al.* (2010)].
- Used commercially by Core Security in Core INSIGHT since 2010, running a variant of Metric-FF [Hoffmann (2003)].

Remarks

History:

- Planning domain "of this kind" (less IT-level, including also physical actions like talking to somebody) first proposed by [Boddy *et al.* (2005)]; used as benchmark in IPC'08 and IPC'11.
- Presented encoding proposed by [Lucangeli *et al.* (2010)].
- Used commercially by Core Security in Core INSIGHT since 2010, running a variant of Metric-FF [Hoffmann (2003)].

Do Core Security's customers like this?

- I am told they do.

Remarks

History:

- Planning domain "of this kind" (less IT-level, including also physical actions like talking to somebody) first proposed by [Boddy *et al.* (2005)]; used as benchmark in IPC'08 and IPC'11.
- Presented encoding proposed by [Lucangeli *et al.* (2010)].
- Used commercially by Core Security in Core INSIGHT since 2010, running a variant of Metric-FF [Hoffmann (2003)].

Do Core Security's customers like this?

- I am told they do.
- In fact, they like it so much already that Core Security is very reluctant to invest money in making this better ...

Remarks

And now:

. . . some remarks about the model.

Assumption (iii)

```
:precondition (and (compromised ?s)
     (connected ?s ?t)
     (has_OS ?t Windows)
     (has_OS_edition ?t Professional)
     (has_OS_servicepack ?t Sp2)
     (has_OS_version ?t WinXp)
     (has_architecture ?t I386)
     (has_service ?t ovtrcd))
:effect (and (compromised ?t) (increase (time) 10)))
```

→ Which of the predicates are static?

Assumption (iii)

```
:precondition (and (compromised ?s)
     (connected ?s ?t)
     (has_OS ?t Windows)
     (has_OS_edition ?t Professional)
     (has_OS_servicepack ?t Sp2)
     (has_OS_version ?t WinXp)
     (has_architecture ?t I386)
     (has_service ?t ovtrcd))
:effect (and (compromised ?t) (increase (time) 10)))
```

→ Which of the predicates are static? All except "compromised".

Assumption (iii)

```
:precondition (and (compromised ?s)
     (connected ?s ?t)
     (has_OS ?t Windows)
     (has_OS_edition ?t Professional)
     (has_OS_servicepack ?t Sp2)
     (has_OS_version ?t WinXp)
     (has_architecture ?t I386)
     (has_service ?t ovtrcd))
:effect (and (compromised ?t) (increase (time) 10)))
```

→ Which of the predicates are static? All except "compromised".

Assumption (iv)

```
(:action HP_OpenView_Remote_Buffer_Overflow_Exploit
    :parameters (?s - host ?t - host)
    ...
    :effect (and (compromised ?t) (increase (time) 10)))
```

→ Are you missing something?

Assumption (iv)

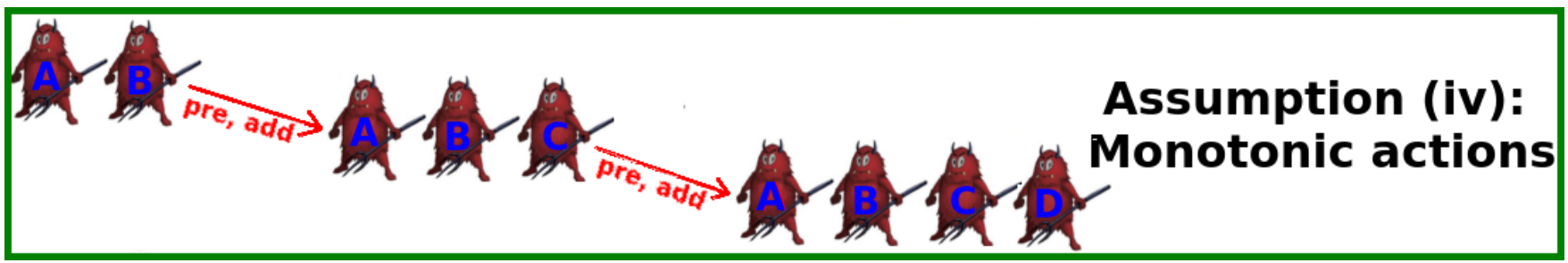
```
(:action HP_OpenView_Remote_Buffer_Overflow_Exploit
    :parameters (?s - host ?t - host)
    ...
    :effect (and (compromised ?t) (increase (time) 10)))
```

→ Are you missing something? There are no delete effects.

Assumption (iv)

```
(:action HP_OpenView_Remote_Buffer_Overflow_Exploit
    :parameters (?s - host ?t - host)
    ...
    :effect (and (compromised ?t) (increase (time) 10)))
```

→ Are you missing something? There are no delete effects.

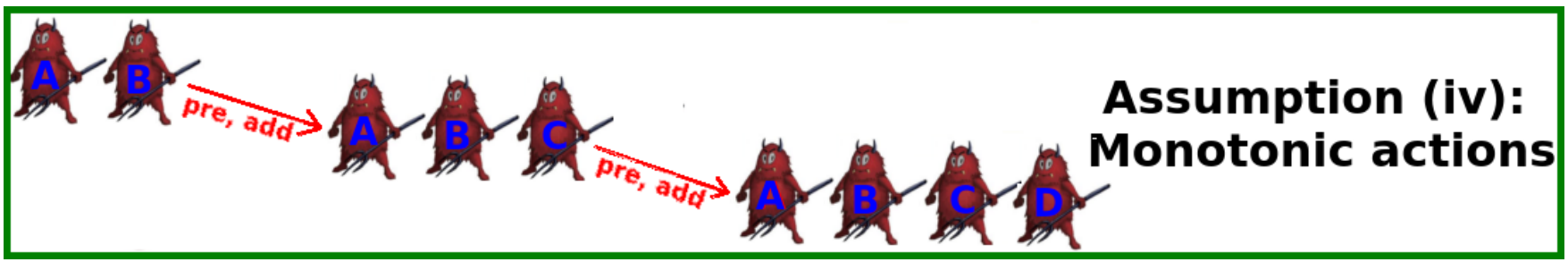


- The attack is monotonic (growing set of attack assets).
- = delete-relaxed planning.

Assumption (iv)

```
(:action HP_OpenView_Remote_Buffer_Overflow_Exploit
    :parameters (?s - host ?t - host)
    ...
    :effect (and (compromised ?t) (increase (time) 10)))
```

→ Are you missing something? There are no delete effects.



- The attack is monotonic (growing set of attack assets).
- = delete-relaxed planning.
- Metric-FF solves this once in every search state . . .
- Generating an attack is polynomial-time. Generating an optimal attack is **NP**-complete.

Assumption (v)

```
:precondition (and (compromised ?s) (connected ?s ?t)
    (has_OS ?t Windows) (has_OS_edition ?t Professional)
    (has_OS_servicepack ?t Sp2) (has_OS_version ?t WinXp)
    (has_architecture ?t I386) (has_service ?t ovtrcd))
```

→ Which preconditions are not static?

Assumption (v)

```
:precondition (and (compromised ?s) (connected ?s ?t)
    (has_OS ?t Windows) (has_OS_edition ?t Professional)
    (has_OS_servicepack ?t Sp2) (has_OS_version ?t WinXp)
    (has_architecture ?t I386) (has_service ?t ovtrcd))
```

→ Which preconditions are not static? Just 1: "(compromised ?s)".

Assumption (v)

```
:precondition (and (compromised ?s) (connected ?s ?t)
    (has_OS ?t Windows) (has_OS_edition ?t Professional)
    (has_OS_servicepack ?t Sp2) (has_OS_version ?t WinXp)
    (has_architecture ?t I386) (has_service ?t ovtrcd))
```

→ Which preconditions are not static? Just 1: "(compromised ?s)".

- 1 positive precondition, 1 positive effect.

Assumption (v)

```
:precondition (and (compromised ?s) (connected ?s ?t)
    (has_OS ?t Windows) (has_OS_edition ?t Professional)
    (has_OS_servicepack ?t Sp2) (has_OS_version ?t WinXp)
    (has_architecture ?t I386) (has_service ?t ovtrcd))
```

→ Which preconditions are not static? Just 1: "(compromised ?s)".

- 1 positive precondition, 1 positive effect.
- Optimal attack planning for single goal host = Dijkstra.

Assumption (v)

```
:precondition (and (compromised ?s) (connected ?s ?t)
    (has_OS ?t Windows) (has_OS_edition ?t Professional)
    (has_OS_servicepack ?t Sp2) (has_OS_version ?t WinXp)
    (has_architecture ?t I386) (has_service ?t ovtrcd))
```

→ Which preconditions are not static? Just 1: "(compromised ?s)".

- 1 positive precondition, 1 positive effect.
- Optimal attack planning for single goal host = Dijkstra.

  Fixed # goal hosts polynomial-time [Bylander (1994)].
  Scaling # goal hosts = Steiner tree [Keyder and Geffner (2009)].

Concluding Remarks?

Simulated Pentesting at Core Security

$\approx$

Dijkstra in the graph over network hosts where weighted edges are defined as a function of configuration parameters and available exploits.

Concluding Remarks?

Simulated Pentesting at Core Security

$\approx$

Dijkstra in the graph over network hosts where weighted edges are defined as a function of configuration parameters and available exploits.

Why they use planning & Metric-FF anyway:

Concluding Remarks?

Simulated Pentesting at Core Security

$\approx$

Dijkstra in the graph over network hosts where weighted edges are defined as a function of configuration parameters and available exploits.

Why they use planning & Metric-FF anyway:

- Extensibility to more fine-grained models of exploits, socio-technical aspects, detrimental side effects.

Concluding Remarks?

Simulated Pentesting at Core Security

$\approx$

Dijkstra in the graph over network hosts where weighted edges are defined as a function of configuration parameters and available exploits.

Why they use planning & Metric-FF anyway:

- Extensibility to more fine-grained models of exploits, socio-technical aspects, detrimental side effects.
- Bounded sub-optimal search to suggest several solutions not just a single "optimal" one.

Concluding Remarks?

Simulated Pentesting at Core Security

$\approx$

Dijkstra in the graph over network hosts where weighted edges are defined as a function of configuration parameters and available exploits.

Why they use planning & Metric-FF anyway:

- Extensibility to more fine-grained models of exploits, socio-technical aspects, detrimental side effects.
- Bounded sub-optimal search to suggest several solutions not just a single "optimal" one.
- Quicker & cheaper than building a proprietary solver.

Agenda

Attack Graphs in a Nutshell

Community: Application-oriented security.

Approach: Describe attack actions by preconditions and effects. Identify/give overview of dangerous action combinations.

Attack Graphs in a Nutshell

Community: Application-oriented security.

Approach: Describe attack actions by preconditions and effects. Identify/give overview of dangerous action combinations.

Example model:

RSH_Connection_Spoofing:

requires	**with**
Trusted_Partner: TP;	TP.service is RSH;
Service_Active: SA;	SA.service is RSH;
...	...
provides	**with**
push_channel: PSC;	PSC_using := RSH;
remote_execution: REX;	REX.using := RSH;
...	...

Attack Graphs in a Nutshell, ctd.

Brief overview of variants:

Who and When?	What?	Terminology

Attack Graphs in a Nutshell, ctd.

Brief overview of variants:

Who and When?	What?	Terminology
Schneier (1999); Templeton and Levitt (2000)	STRIPS actions	"attack graph" = action descriptions

Attack Graphs in a Nutshell, ctd.

Brief overview of variants:

Who and When?	What?	Terminology
Schneier (1999); Templeton and Levitt (2000)	STRIPS actions	"attack graph" = action descriptions
Ritchey and Ammann (2000)	BDD model checking	"attack graph" = state space

Attack Graphs in a Nutshell, ctd.

Brief overview of variants:

Who and When?	What?	Terminology
Schneier (1999); Templeton and Levitt (2000)	STRIPS actions	"attack graph" = action descriptions
Ritchey and Ammann (2000)	BDD model checking	"attack graph" = state space
Ammann *et al.* (2002)	"Attacks are monotonic!"	

Attack Graphs in a Nutshell, ctd.

Brief overview of variants:

Who and When?	What?	Terminology
Schneier (1999); Templeton and Levitt (2000)	STRIPS actions	"attack graph" = action descriptions
Ritchey and Ammann (2000)	BDD model checking	"attack graph" = state space
Ammann *et al.* (2002)	"Attacks are monotonic!"	
Since then, e. g. Ammann *et al.* (2002); Noel *et al.* (2009)	Relaxed planning	"attack graph" = relaxed planning graph

Attack Graphs in a Nutshell, ctd.

Brief overview of variants:

Who and When?	What?	Terminology
Schneier (1999); Templeton and Levitt (2000)	STRIPS actions	"attack graph" = action descriptions
Ritchey and Ammann (2000)	BDD model checking	"attack graph" = state space
Ammann *et al.* (2002)	"Attacks are monotonic!"	
Since then, e. g. Ammann *et al.* (2002); Noel *et al.* (2009)	Relaxed planning	"attack graph" = relaxed planning graph

→ Attack graphs ≈ practical security-analysis tools based on variants of, and analyses on, relaxed planning graphs.

→ "AI ⇔ attack graphs" community bridge could be quite useful . . .

Dimension (B): Action Model

Two major dimensions for simulated pentesting models:

(A) **Uncertainty Model:** Up next.

(B) **Action Model:** Degree of interaction between individual attack components.

Dimension (B): Action Model

Two major dimensions for simulated pentesting models:

(A) **Uncertainty Model:** Up next.

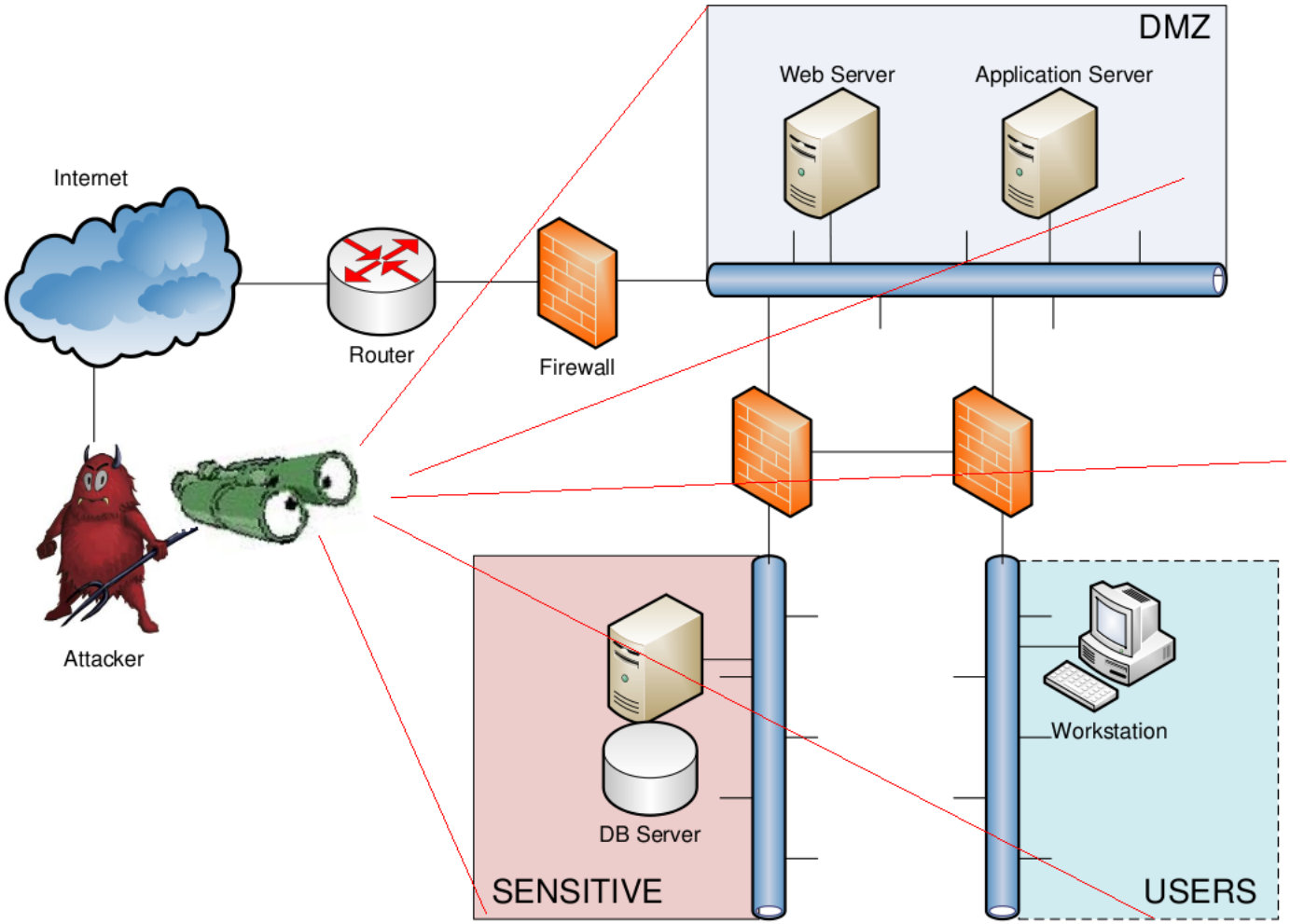
(B) **Action Model:** Degree of interaction between individual attack components.

Dimension (B) distinction lines:

- **Explicit Network Graph:** Actions = "hops from ?s to ?t".
  1 positive precond, 1 positive effect. Subset of compromised hosts.
- **Monotonic actions:** Attacker can only gain new attack assests.
  Installed software, access rights, knowledge (e. g. passwords) etc.
- **General actions:** No restrictions (STRIPS, in simplest case).
  Can model detrimental side effects.

Agenda

An Additional Assumption ...

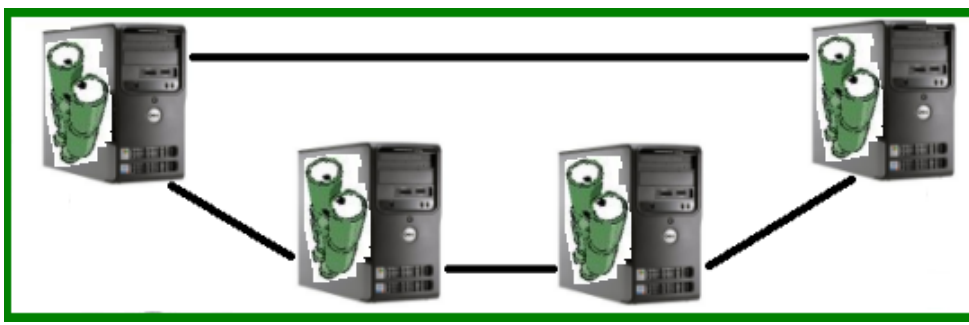


Assumptions (i) and (ii)

Known network graph: No uncertainty about network graph topology.

Assumption (i):
Known network graph
(recommended)

Known host configurations: No uncertainty about host configurations.



Assumption (ii):
Known host configurations

An Overview Before We Begin . . .

Uncertainty Model, Dimension (A):

- **None:** Classical planning.

  → CoreSec-Classical: Core Security's model, as seen. Assumptions (i)–(v).

- **Uncertainty of action outcomes:** MDPs.

  → CoreSec-MDP: Minimal extension of CoreSec-Classical. Assumptions (ii)–(viii).

- **Uncertainty of state:** POMDPs.

  → CoreSec-POMDP: Minimal extension of CoreSec-Classical. Assumptions (ii)–(vii).

Partially Observable MDP (POMDP)

Definition

A POMDP is a tuple $\langle \mathcal{S}, \mathcal{A}, T, \mathcal{O}, O, b_0 \rangle$:

- $\mathcal{S}$ states, $\mathcal{A}$ actions, $\mathcal{O}$ observations.
- $T(s, a, s')$: probability of coming to state s' when executing action a in state s.
- $O(s, a, o)$: probability of making observation o when executing action a in state s.
- b_0: initial belief, probability distribution over $\mathcal{S}$.

Respectively, some (possibly factored) description thereof.

Partially Observable MDP (POMDP)

Definition

A POMDP is a tuple $\langle \mathcal{S}, \mathcal{A}, T, \mathcal{O}, O, b_0 \rangle$:

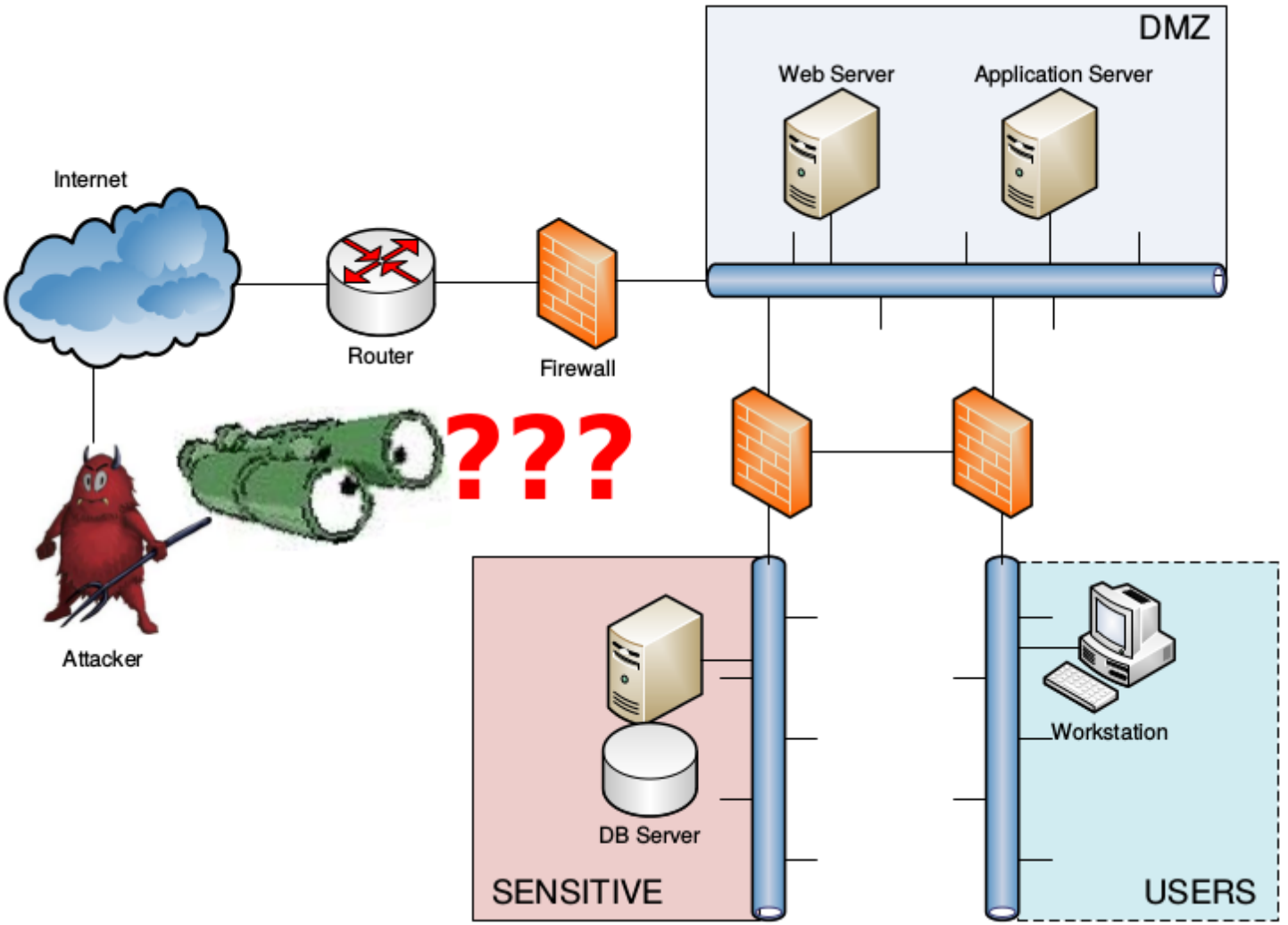
- $\mathcal{S}$ states, $\mathcal{A}$ actions, $\mathcal{O}$ observations.
- $T(s, a, s')$: probability of coming to state s' when executing action a in state s.
- $O(s, a, o)$: probability of making observation o when executing action a in state s.
- b_0: initial belief, probability distribution over $\mathcal{S}$.

Respectively, some (possibly factored) description thereof.

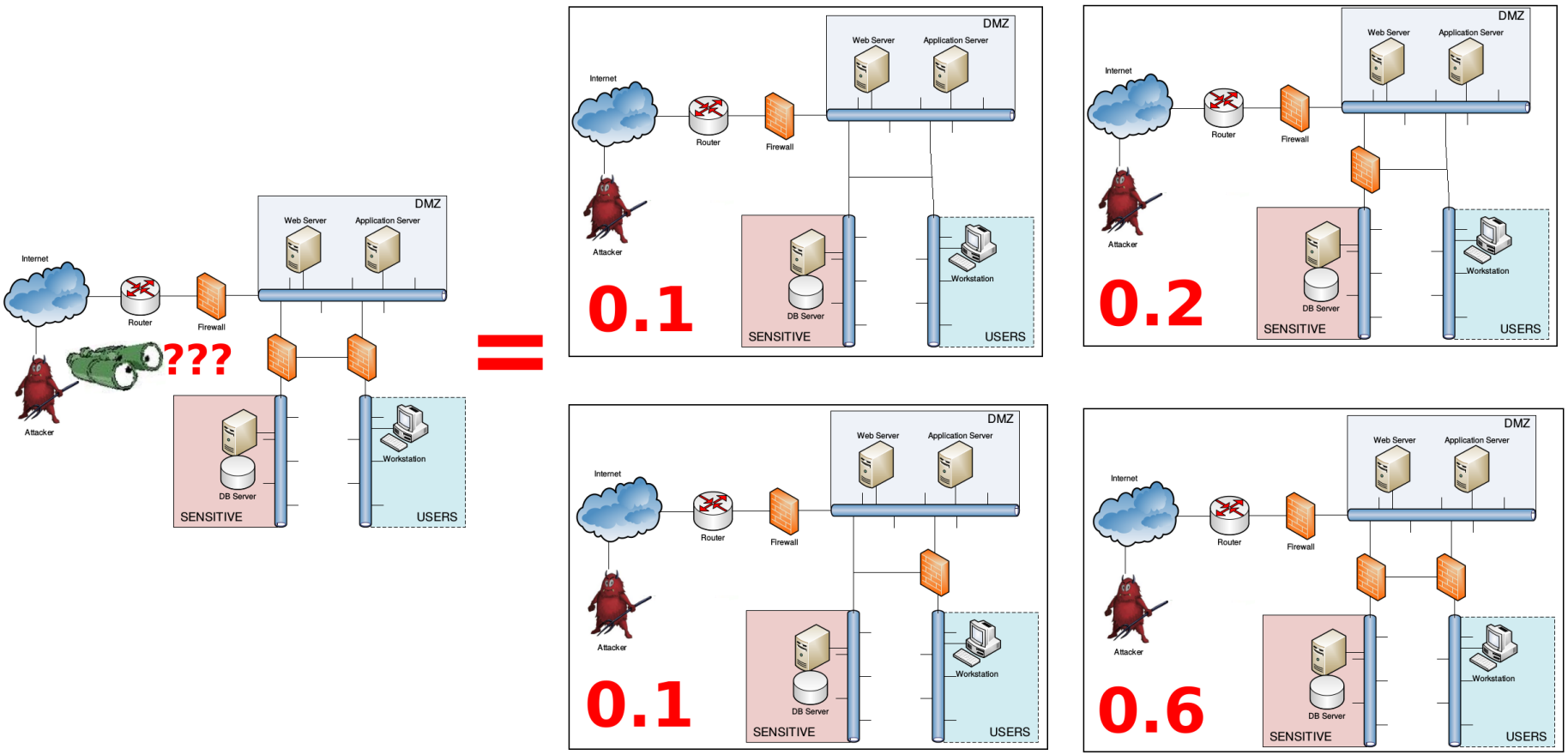
$\rightarrow$ I'll discuss optimization objectives later on.

For now, assume observable goal states S_g, minimizing undiscounted expected cost-to-goal in a Stochastic Shortest Path (SSP) formulation.

The Basic Problem



The Basic Idea [Sarraute *et al.* (2012)]



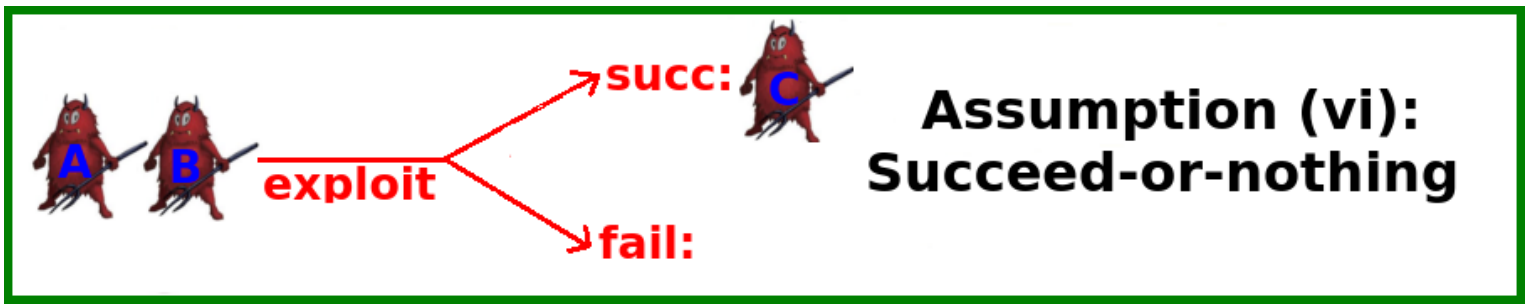
States

```
H0-win2000
H0-win2000-p445
H0-win2000-p445-SMB
H0-win2000-p445-SMB-vuln
H0-win2000-p445-SMB-agent

H0-win2003
H0-win2003-p445
H0-win2003-p445-SMB
H0-win2003-p445-SMB-vuln
H0-win2003-p445-SMB-agent

H0-winXPsp2
H0-winXPsp2-p445
H0-winXPsp2-p445-SMB
H0-winXPsp2-p445-SMB-vuln
H0-winXPsp2-p445-SMB-agent

terminal
```

"H0": the host. "winXXX": OS. "p445": is port 445 open?
"SMB": if so, SAMBA server?
"vuln": SAMBA server vulnerable?
"agent": has attacker exploited that vulnerability yet?
"terminal": attacker has given up.

Assumptions (vi) and (vii)

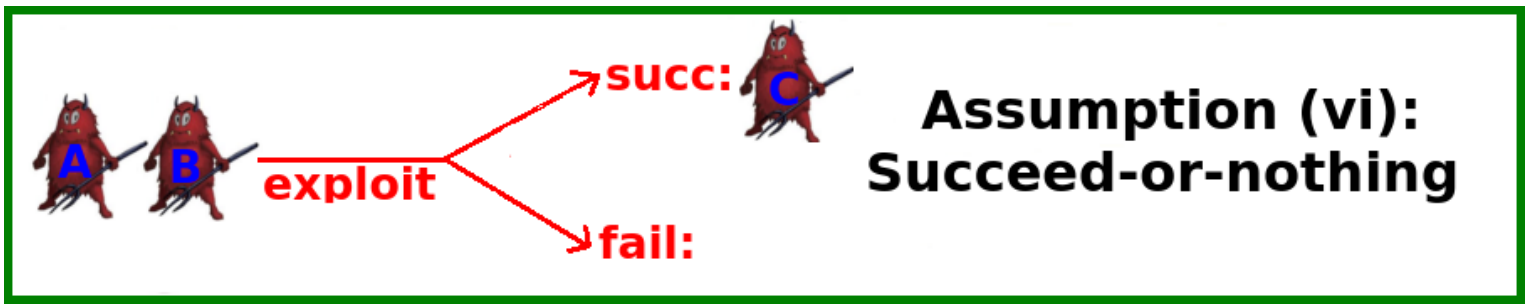
Succeed-or-nothing: Exploits have only two possible outcomes, succeed or fail. Fail has an empty effect.



→ Abstraction mainly regarding detrimental side effects.

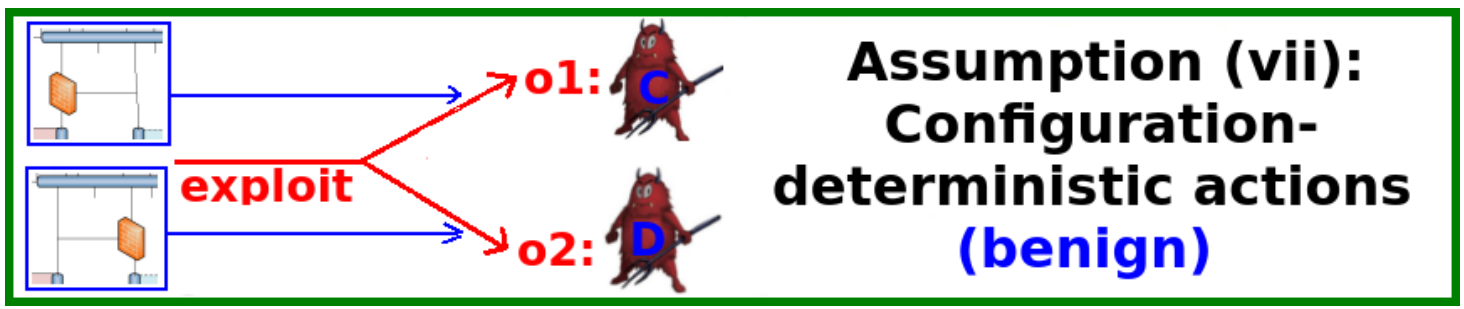
Assumptions (vi) and (vii)

Succeed-or-nothing: Exploits have only two possible outcomes, succeed or fail. Fail has an empty effect.



→ Abstraction mainly regarding detrimental side effects.

Configuration-deterministic actions: Action outcome depends deterministically on network configuration.



→ Abstraction only in case of more fine-grained dependencies.

Exploit Actions

Same syntax:

```
(:action HP_OpenView_Remote_Buffer_Overflow_Exploit
    :parameters (?s - host ?t - host)
    :precondition (and (compromised ?s)
        (connected ?s ?t)
        (has_OS ?t Windows)
        (has_OS_edition ?t Professional)
        (has_OS_servicepack ?t Sp2)
        (has_OS_version ?t WinXp)
        (has_architecture ?t I386)
        (has_service ?t ovtrcd))
    :effect (and (compromised ?t) (increase (time) 10)))
```

Exploit Actions

Same syntax:

```
(:action HP_OpenView_Remote_Buffer_Overflow_Exploit
    :parameters (?s - host ?t - host)
    :precondition (and (compromised ?s)
        (connected ?s ?t)
        (has_OS ?t Windows)
        (has_OS_edition ?t Professional)
        (has_OS_servicepack ?t Sp2)
        (has_OS_version ?t WinXp)
        (has_architecture ?t I386)
        (has_service ?t ovtrcd))
    :effect (and (compromised ?t) (increase (time) 10)))
```

. . . but with a different semantics: Consider $s \xrightarrow{a} s'$

$$T(s,a,s') = \begin{cases} 1 & s \models pre(a), s' = appl(s,a) \\ 1 & s \not\models pre(a), s' = s \\ 0 & \text{otherwise} \end{cases}$$

$$O(s,a,o) = \begin{cases} 1 & s \models pre(a), s' = appl(s,a), o = \text{“success”} \\ 1 & s \not\models pre(a), s' = s, o = \text{“fail”} \\ 0 & \text{otherwise} \end{cases}$$

Sensing Actions

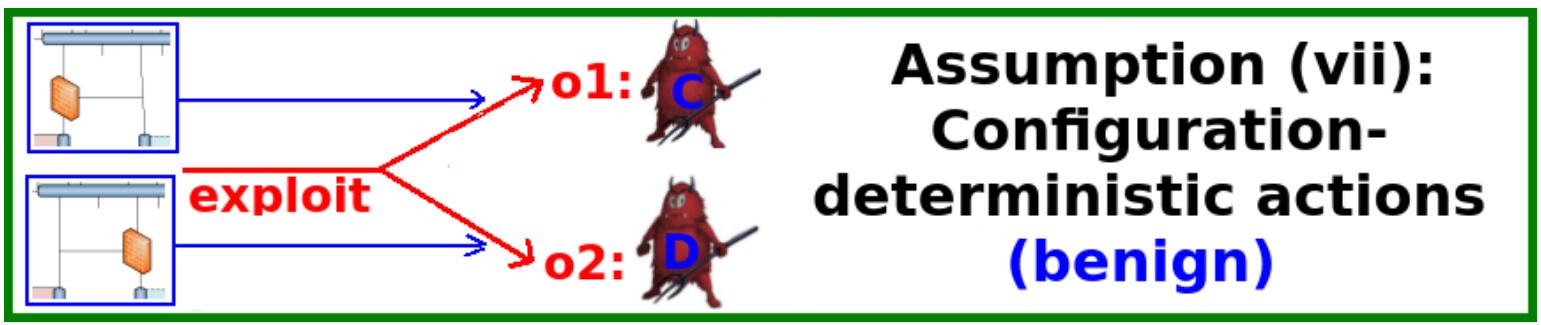
Example:

```
(:action OS_Detect
    :parameters (?s - host ?t - host)
    :precondition (and (compromised ?s) (connected ?s ?t))
    :observe (and
        (when (has_OS ?t Windows2000) ("win"))
        (when (has_OS ?t Windows2003) ("win"))
        (when (has_OS ?t WindowsXPsp2) ("winXP"))
        (when (has_OS ?t WindowsXPsp3) ("winXP")))
```

Sensing Actions

Example:
```
(:action OS_Detect
    :parameters (?s - host ?t - host)
    :precondition (and (compromised ?s) (connected ?s ?t))
    :observe (and
        (when (has_OS ?t Windows2000) ("win"))
        (when (has_OS ?t Windows2003) ("win"))
        (when (has_OS ?t WindowsXPsp2) ("winXP"))
        (when (has_OS ?t WindowsXPsp3) ("winXP")))
```

Network reconnaissance also satisfies the benign assumption:



→ Non-injective but deterministic function of configuration.

So, we're done, right?

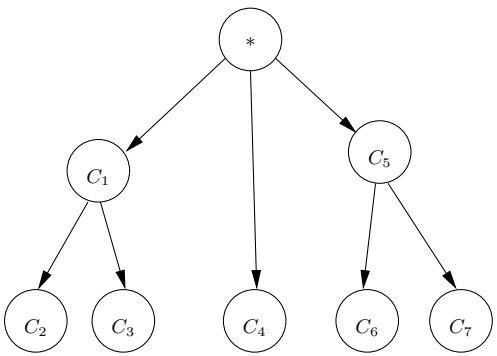
So, we're done, right?

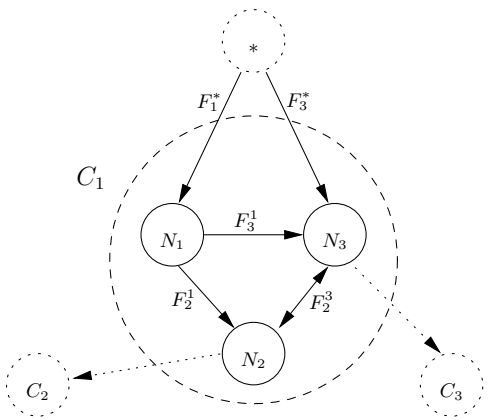
Computation!

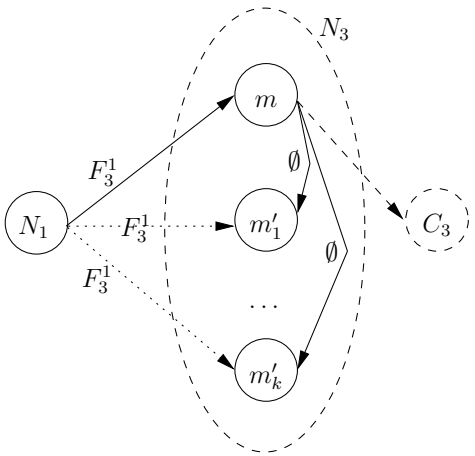
So, we're done, right?

Computation!

But: Can use single-machine case + decomposition.

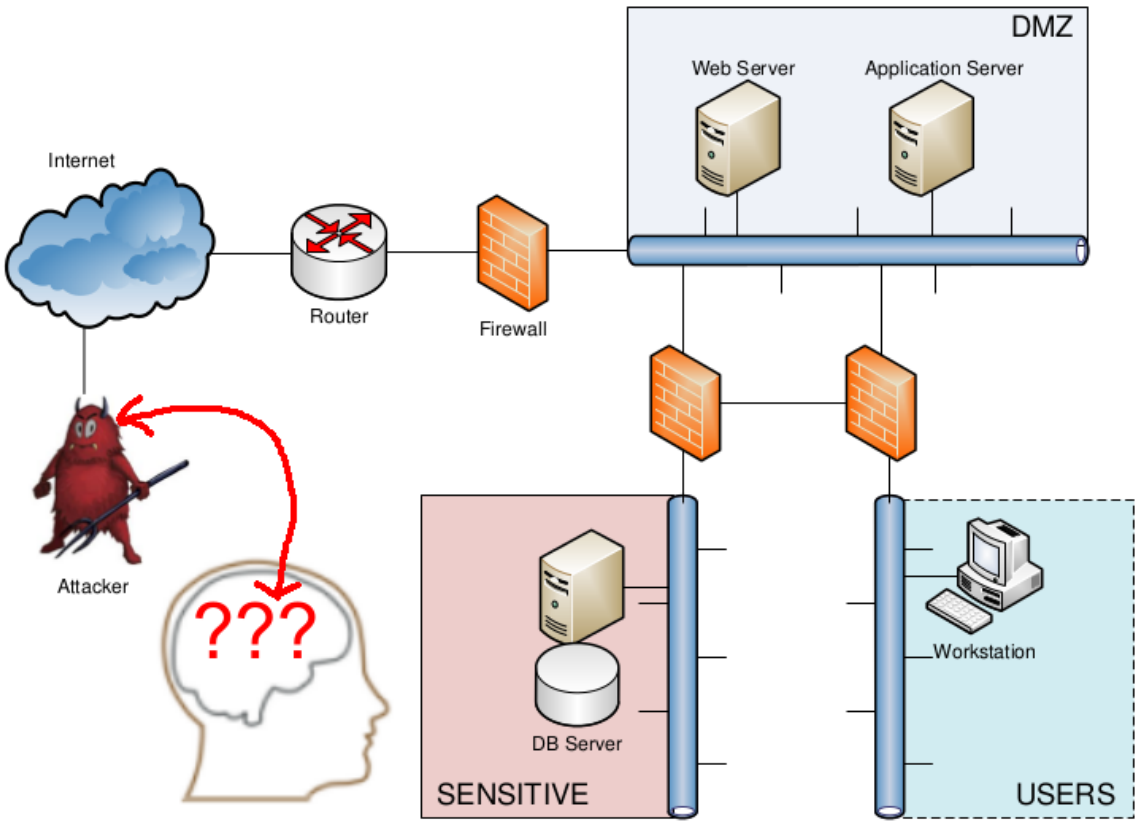






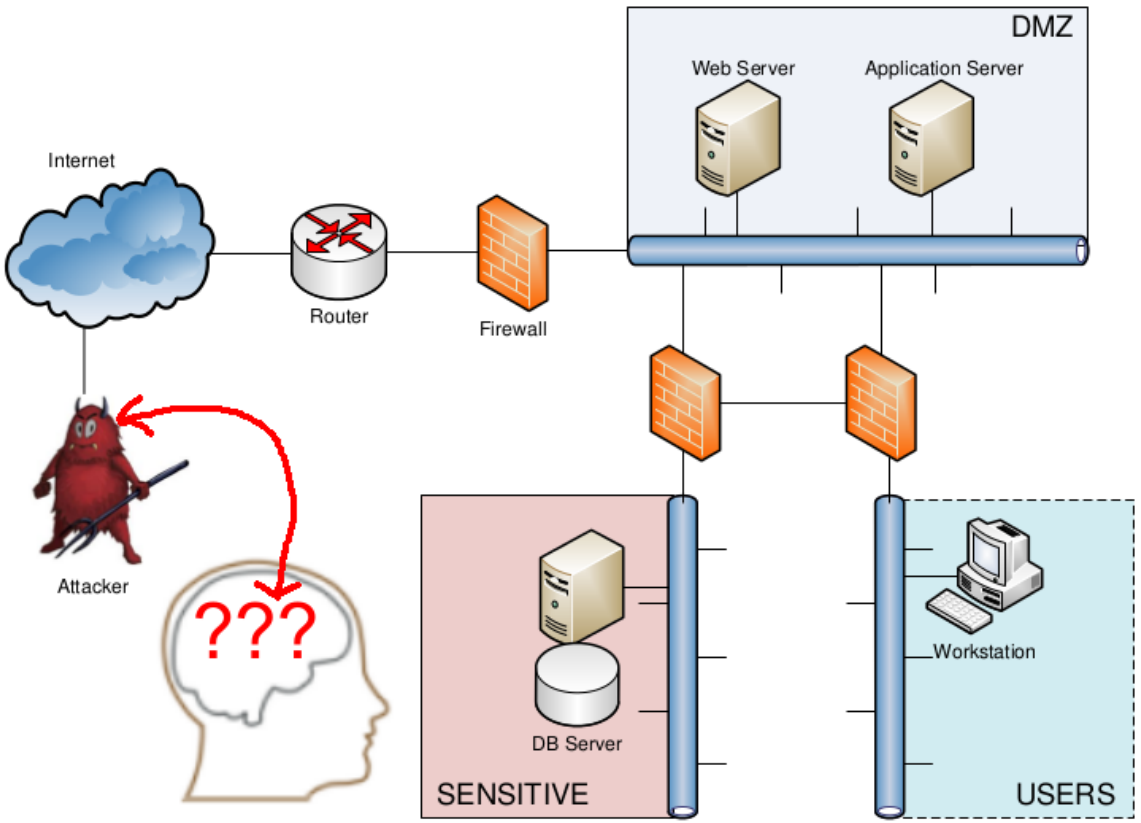
So, we're done, right?

Modeling!



So, we're done, right?

Modeling!



But: Can use outcome of standard scanning scripts?

Agenda

Markov Decision Process (MDP)

Definition

An MDP is a tuple $\langle \mathcal{S}, \mathcal{A}, T, s_0 \rangle$:

- $\mathcal{S}$ states, $\mathcal{A}$ actions.
- $T(s, a, s')$: probability of coming to state s' when executing action a in state s.
- s_0: initial state.

Respectively, some (possibly factored) description thereof.

$\rightarrow$ I'll discuss optimization objectives later on.

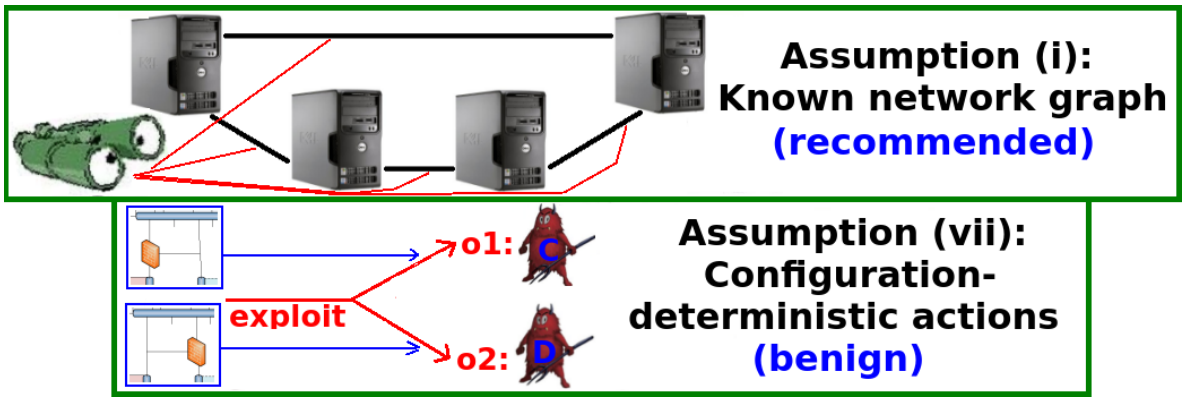
For now, assume goal states S_g, minimizing undiscounted expected cost-to-goal in a Stochastic Shortest Path (SSP) formulation.

The Basic Idea

```
(:action HP_OpenView_Remote_Buffer_Overflow_Exploit
    :parameters (?s - host ?t - host)
    :precondition (and (compromised ?s) (connected ?s ?t)
        (has_OS ?t Windows)
        (has_OS_edition ?t Professional)
        (has_OS_servicepack ?t Sp2)
        (has_OS_version ?t WinXp)
        (has_architecture ?t I386)
        (has_service ?t ovtrcd))
    :effect (and (compromised ?t)
        (increase (time) 10)))
```

```
(:action HP_OpenView_Remote_Buffer_Overflow_Exploit
    :parameters (?s - host ?t - host)
    :precondition (and (compromised ?s) (connected ?s ?t))
    :effect (and (probabilistic 0.3 (compromised ?t))
        (increase (time) 10)))
```

How to Obtain Action Outcome Probabilities?



$\Longrightarrow$ outcome occurs iff ϕ(host configurations)

How to Obtain Action Outcome Probabilities?

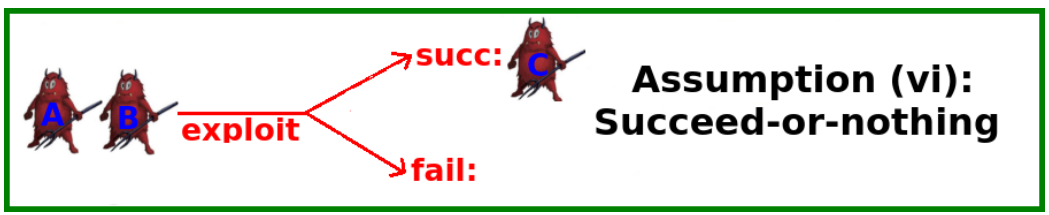
$\Longrightarrow$ outcome occurs iff ϕ(host configurations)

$\Longrightarrow$ outcome probability $\approx P(\phi(\text{host configurations}), b_0)$

How to Obtain Action Outcome Probabilities?

$\Longrightarrow$ outcome occurs iff ϕ(host configurations)

$\Longrightarrow$ outcome probability $\approx P(\phi(\text{host configurations}), b_0)$

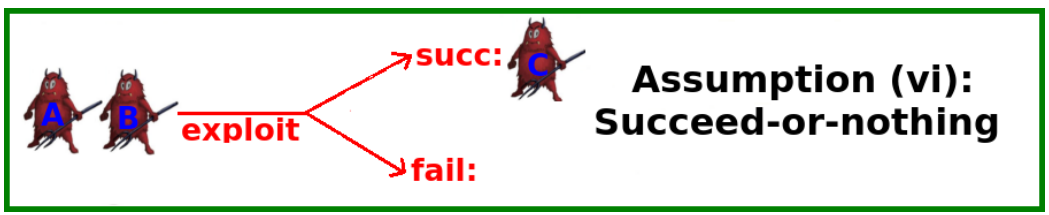


$\Longrightarrow$ just need success probability as function of host configurations in b_0

How to Obtain Action Outcome Probabilities?

$\Longrightarrow$ outcome occurs iff ϕ(host configurations)

$\Longrightarrow$ outcome probability $\approx P(\phi(\text{host configurations}), b_0)$



$\Longrightarrow$ just need success probability as function of host configurations in b_0

$\Longrightarrow$ **Use Core Security success statistics as success probabilities.**

MDP vs. POMDP

Where did we cheat on the previous slide?

MDP vs. POMDP

Where did we cheat on the previous slide?

$\Longrightarrow$ outcome prob $\approx P(\phi(\text{host configs}), b_0)$

$\rightarrow$ b_0 just captures the attacker's *initial knowledge*.

MDP vs. POMDP

Where did we cheat on the previous slide?

$\Longrightarrow$ outcome prob $\approx P(\phi(\text{host configs}), b_0)$

$\rightarrow b_0$ just captures the attacker's *initial knowledge*.

Hence: Inability to learn. Success probabilities develop with knowledge in the POMDP, but remain constant in the MDP.

MDP vs. POMDP

Where did we cheat on the previous slide?

$\Longrightarrow$ outcome prob $\approx P(\phi(\text{host configs}), b_0)$

$\rightarrow$ b_0 just captures the attacker's *initial knowledge*.

Hence: Inability to learn. Success probabilities develop with knowledge in the POMDP, but remain constant in the MDP.

(But: Maintain flags for partial belief-tracking in the MDP?)

Assumption (viii)

Assume that ?t doesn't have the required configuration:

```
(:action HP_OpenView_Remote_Buffer_Overflow_Exploit
    :parameters (?s - host ?t - host)
    :precondition (and (compromised ?s) (connected ?s ?t))
    :effect (and (probabilistic 0.3 (compromised ?t))
        (increase (time) 10)))
```

Assumption (viii)

Assume that ?t doesn't have the required configuration:

```
(:action HP_OpenView_Remote_Buffer_Overflow_Exploit
    :parameters (?s - host ?t - host)
    :precondition (and (compromised ?s) (connected ?s ?t))
    :effect (and (probabilistic 0.3 (compromised ?t))
        (increase (time) 10)))
```

→ The probability of breaking into ?t eventually is

Assumption (viii)

Assume that ?t doesn't have the required configuration:

```
(:action HP_OpenView_Remote_Buffer_Overflow_Exploit
    :parameters (?s - host ?t - host)
    :precondition (and (compromised ?s) (connected ?s ?t))
    :effect (and (probabilistic 0.3 (compromised ?t))
        (increase (time) 10)))
```

→ The probability of breaking into ?t eventually is 1.

Assumption (viii)

Assume that ?t doesn't have the required configuration:

```
(:action HP_OpenView_Remote_Buffer_Overflow_Exploit
    :parameters (?s - host ?t - host)
    :precondition (and (compromised ?s) (connected ?s ?t))
    :effect (and (probabilistic 0.3 (compromised ?t))
        (increase (time) 10)))
```

→ The probability of breaking into ?t eventually is 1.

This contradicts our benign assumptions (iii) and (vii).

Assumption (viii)

Assume that ?t doesn't have the required configuration:

```
(:action HP_OpenView_Remote_Buffer_Overflow_Exploit
    :parameters (?s - host ?t - host)
    :precondition (and (compromised ?s) (connected ?s ?t))
    :effect (and (probabilistic 0.3 (compromised ?t))
        (increase (time) 10)))
```

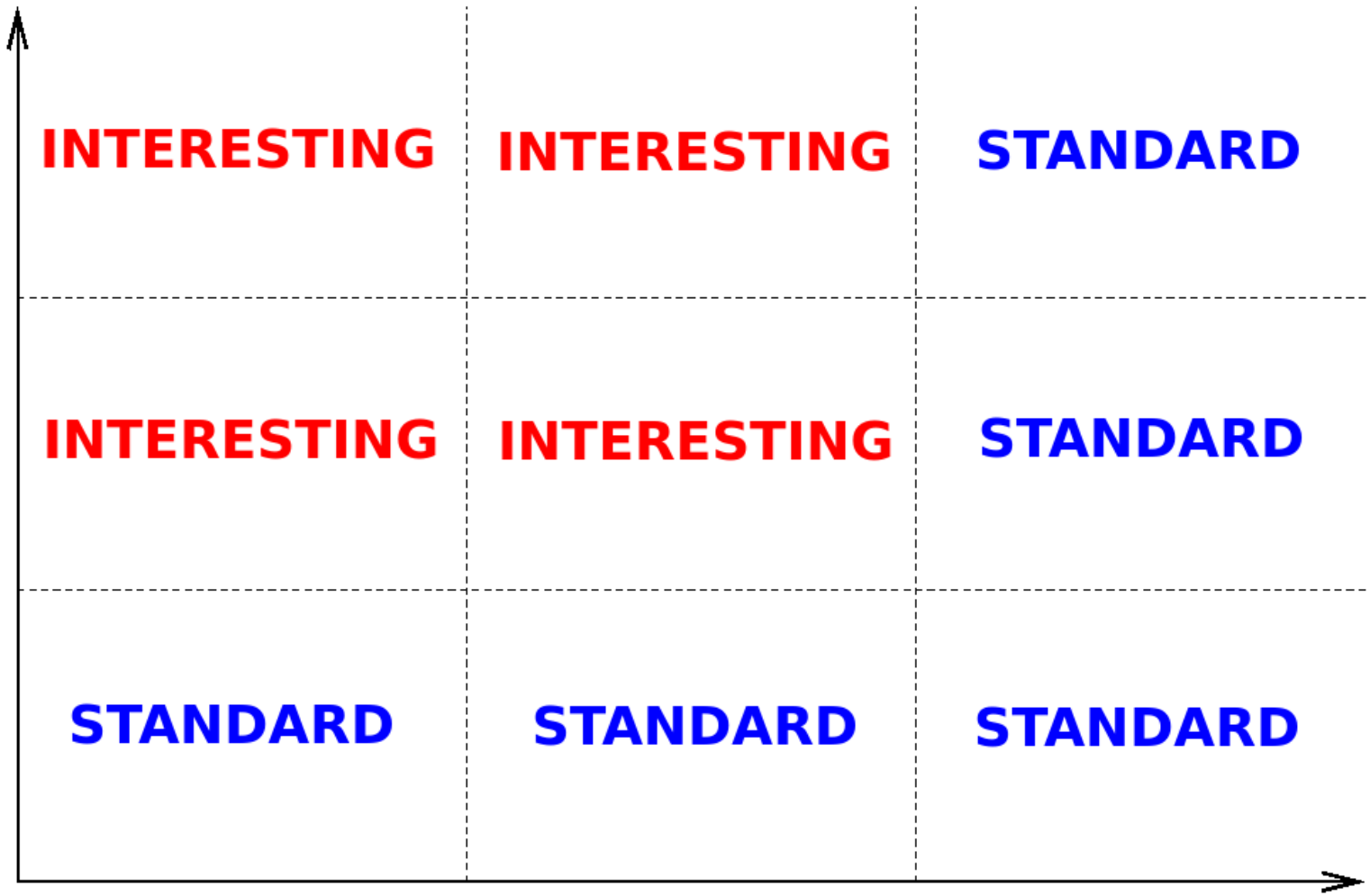
→ The probability of breaking into ?t eventually is 1.

This contradicts our benign assumptions (iii) and (vii). Hence:

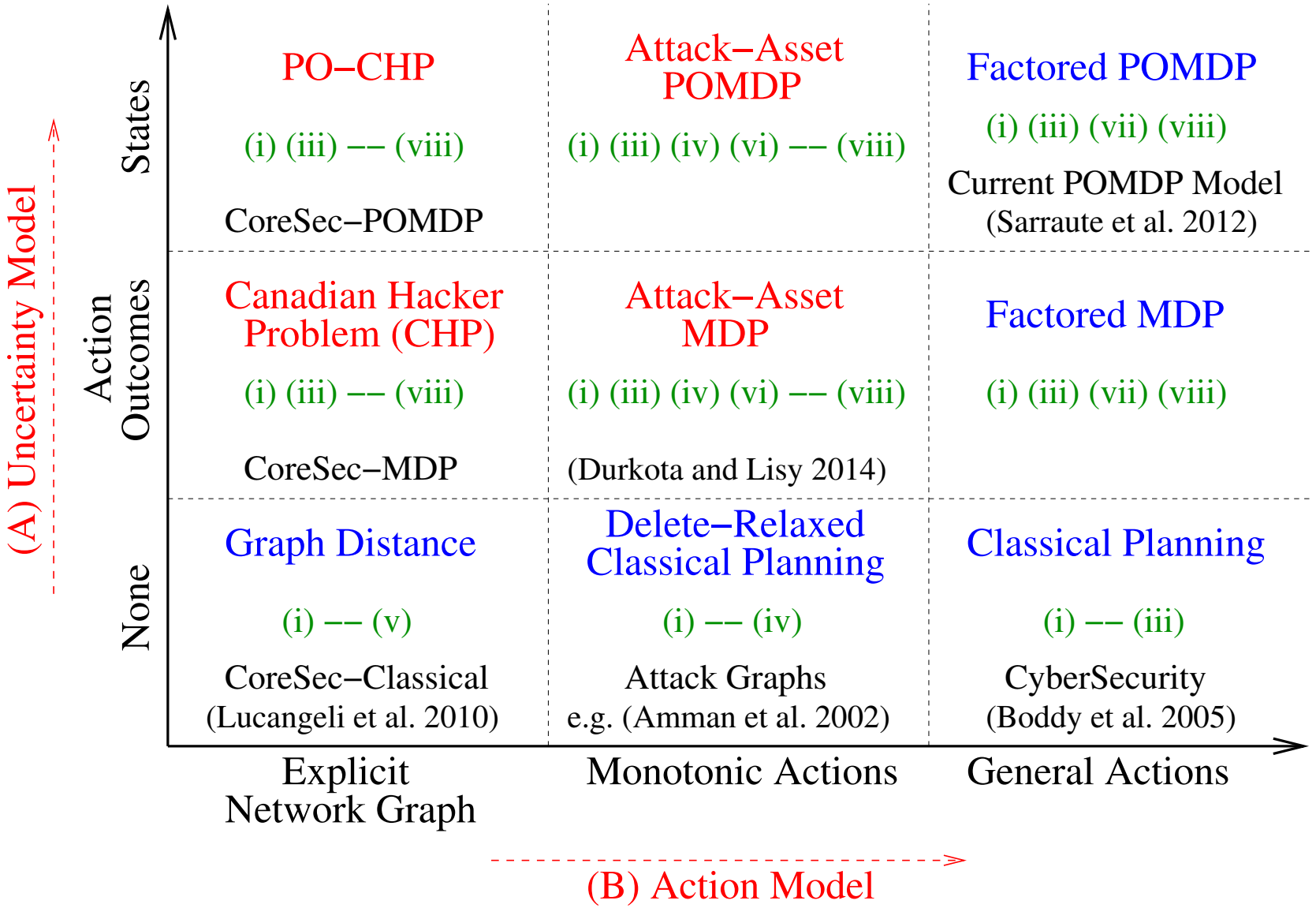
Apply-once constraint: Allow to apply each exploit, on each target host, at most once.

Agenda

Remember?


INTERESTING
INTERESTING
STANDARD
INTERESTING
INTERESTING
STANDARD
STANDARD
STANDARD
STANDARD

A Model Taxonomy



The 3rd Dimension

Three major dimensions for simulated pentesting models:

(A) **Uncertainty Model.**

(B) **Action Model.**

(C) **Optimization objective:** What is the atttacker trying to achieve?

The 3rd Dimension

Three major dimensions for simulated pentesting models:

(A) **Uncertainty Model.**

(B) **Action Model.**

(C) **Optimization objective:** What is the atttacker trying to achieve?

Options:

- Finite-horizon: Ok. **But:** Offline problem, horizon not meaningful unless for overall attack (see below).
- Maximize discounted reward: Ok. **But:** Discounting unintuitive. And who's to set the rewards?

The 3rd Dimension

Three major dimensions for simulated pentesting models:

(A) **Uncertainty Model.**

(B) **Action Model.**

(C) **Optimization objective:** What is the atttacker trying to achieve?

Options:

- Finite-horizon: Ok. **But:** Offline problem, horizon not meaningful unless for overall attack (see below).
- Maximize discounted reward: Ok. **But:** Discounting unintuitive. And who's to set the rewards?
- Minimize non-discounted expected cost-to-goal (SSP): Seems good. Non-0 action costs, give-up action. **But:** Give-up cost?

The 3rd Dimension

Three major dimensions for simulated pentesting models:

(A) **Uncertainty Model.**

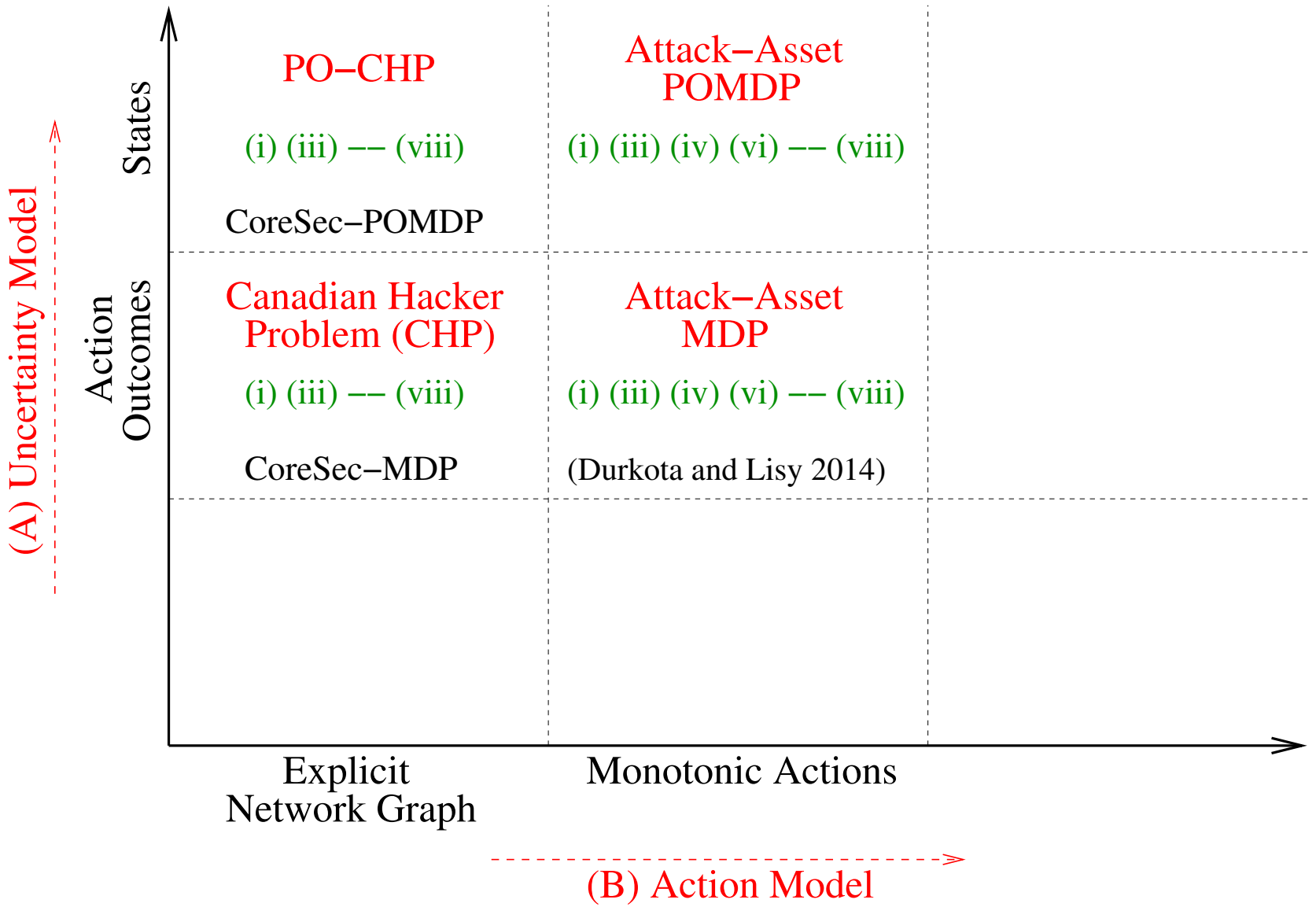
(B) **Action Model.**

(C) **Optimization objective:** What is the atttacker trying to achieve?

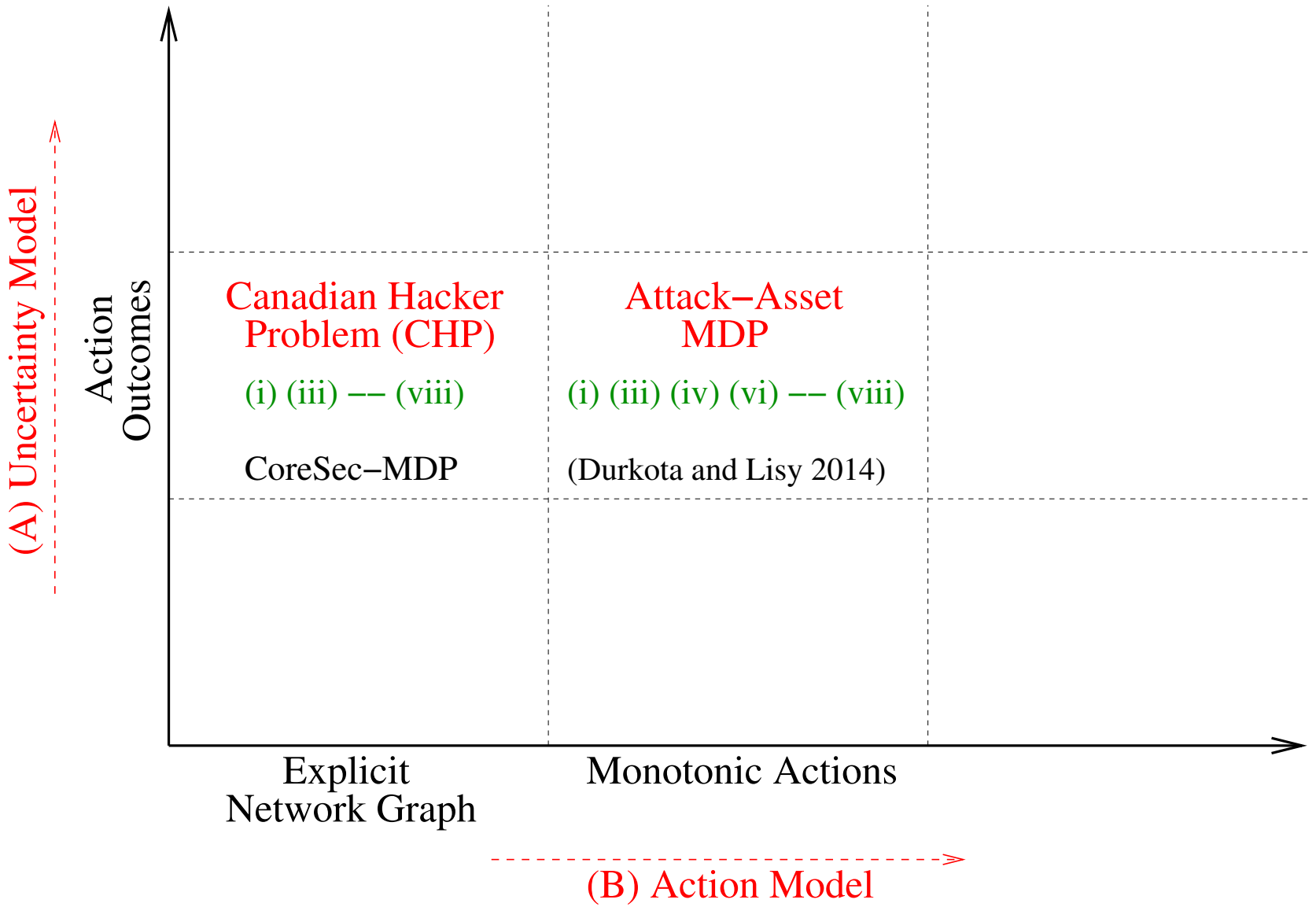
Options:

- Finite-horizon: Ok. **But:** Offline problem, horizon not meaningful unless for overall attack (see below).
- Maximize discounted reward: Ok. **But:** Discounting unintuitive. And who's to set the rewards?
- Minimize non-discounted expected cost-to-goal (SSP): Seems good. Non-0 action costs, give-up action. **But:** Give-up cost?
- Limited-budget goal probability maximization (MAXPROP): My favorite. Non-0 action costs, give-up action, hence finite-runs SSP. No "but" I can think of.

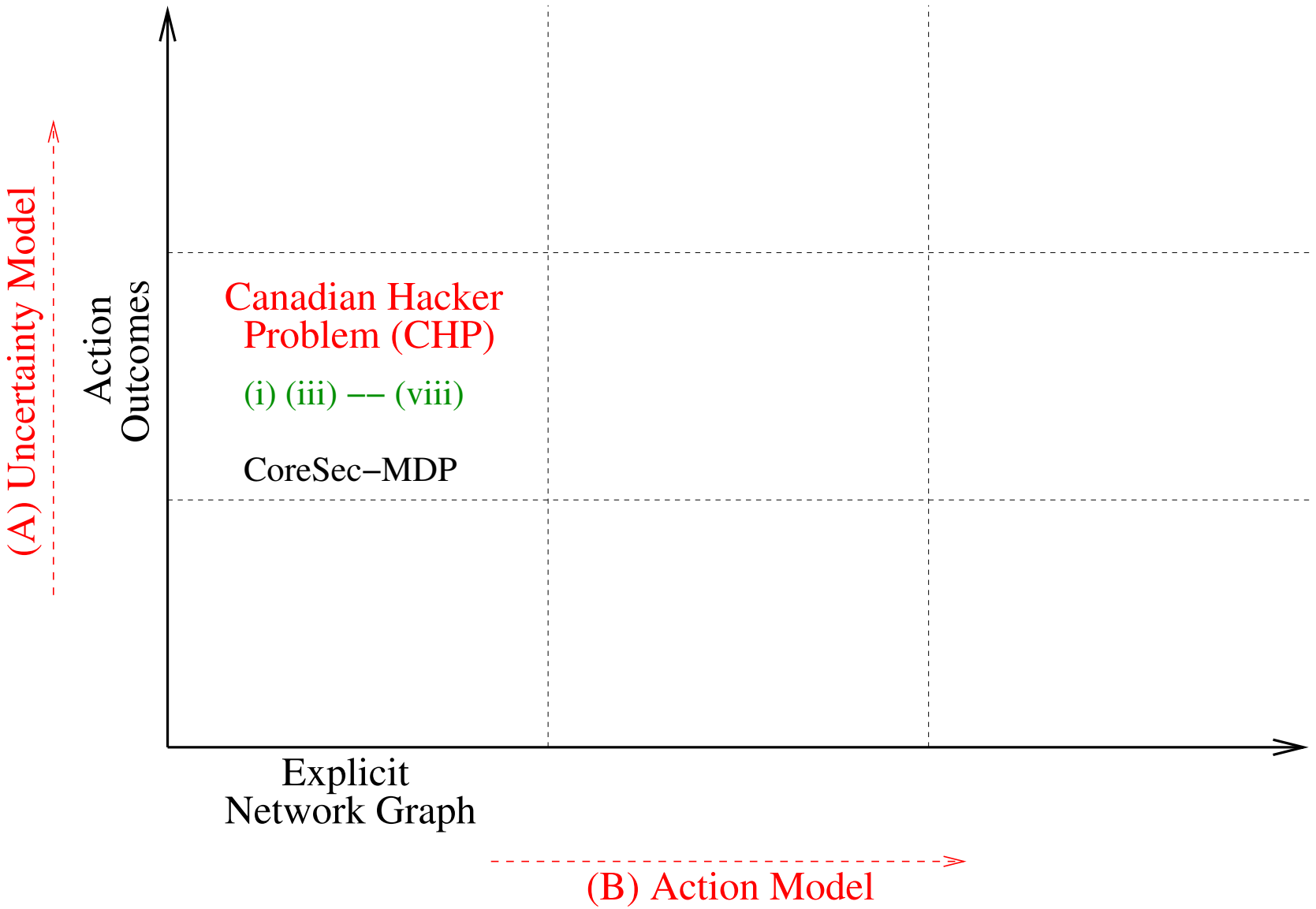
The Interesting Sub-Classes



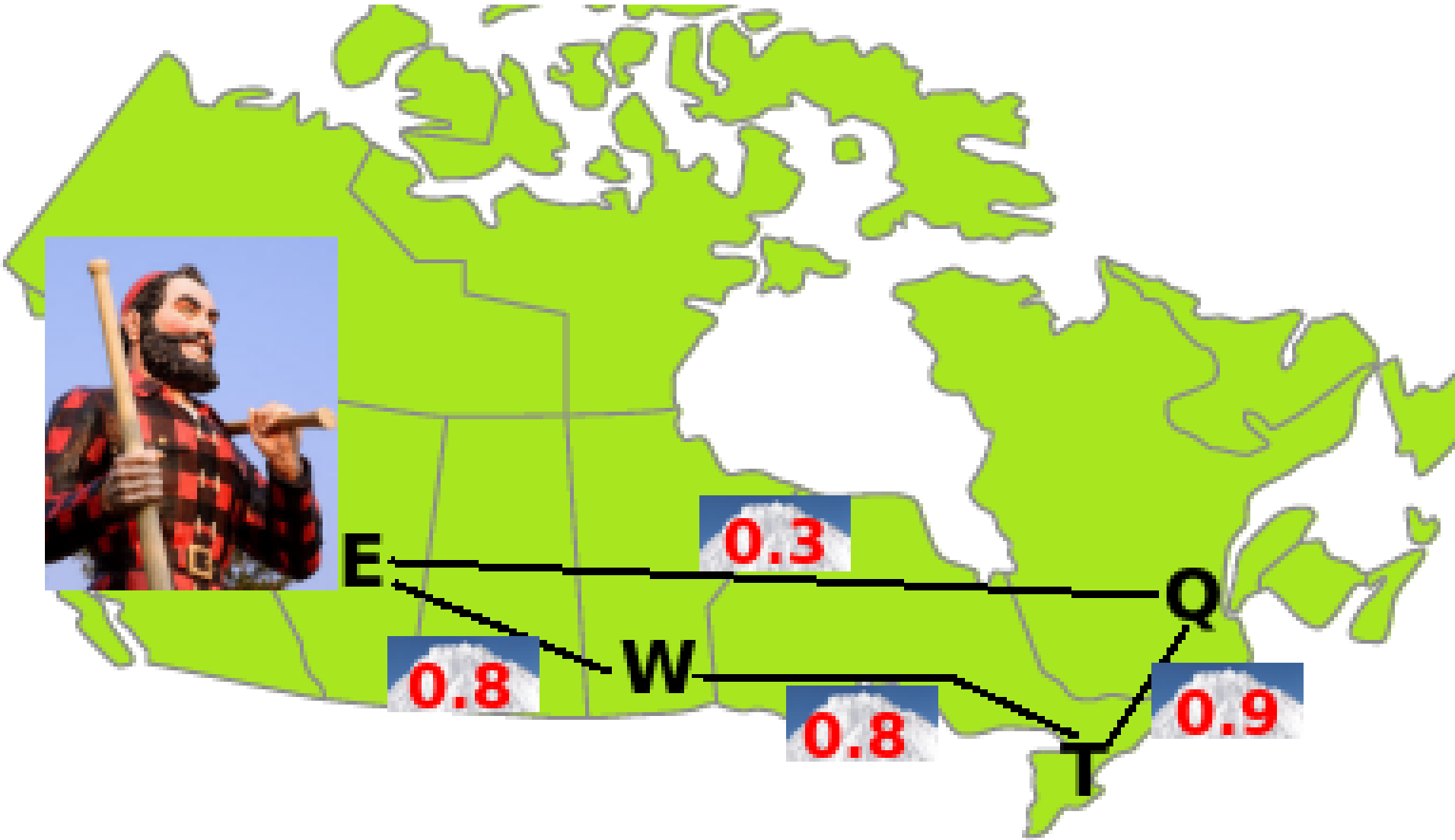
The Interesting Sub-Classes we will Discuss



We Start With:



The Canadian Traveller Problem



The Canadian Hacker Problem

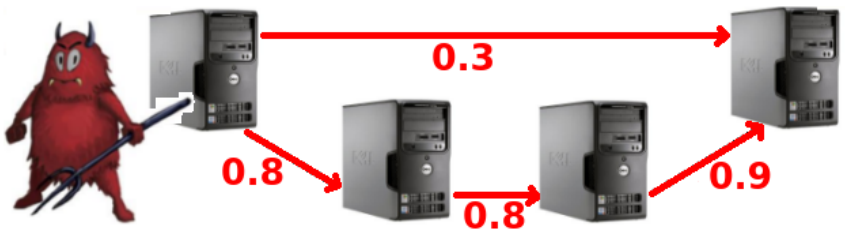
The Canadian Hacker Problem

+

action-outcome uncertainty =

The Canadian Hacker Problem

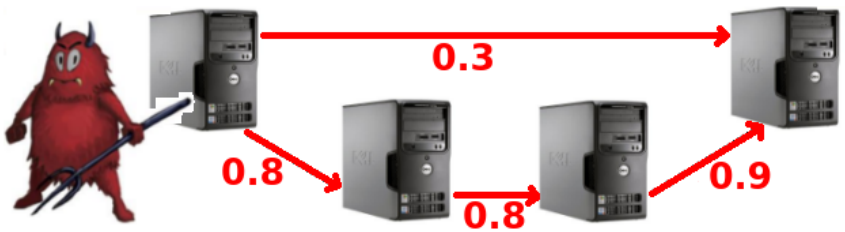
action-outcome uncertainty =



The Canadian Hacker Problem

+

action-outcome uncertainty =



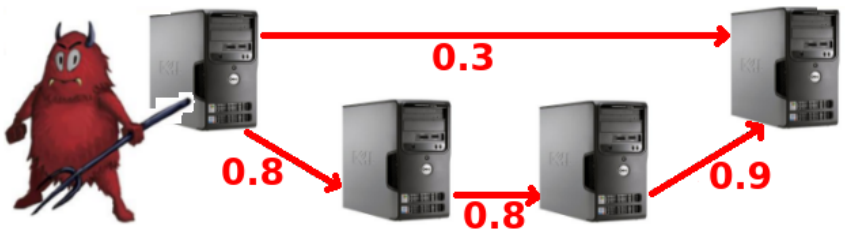
+

=

The Canadian Hacker Problem

+

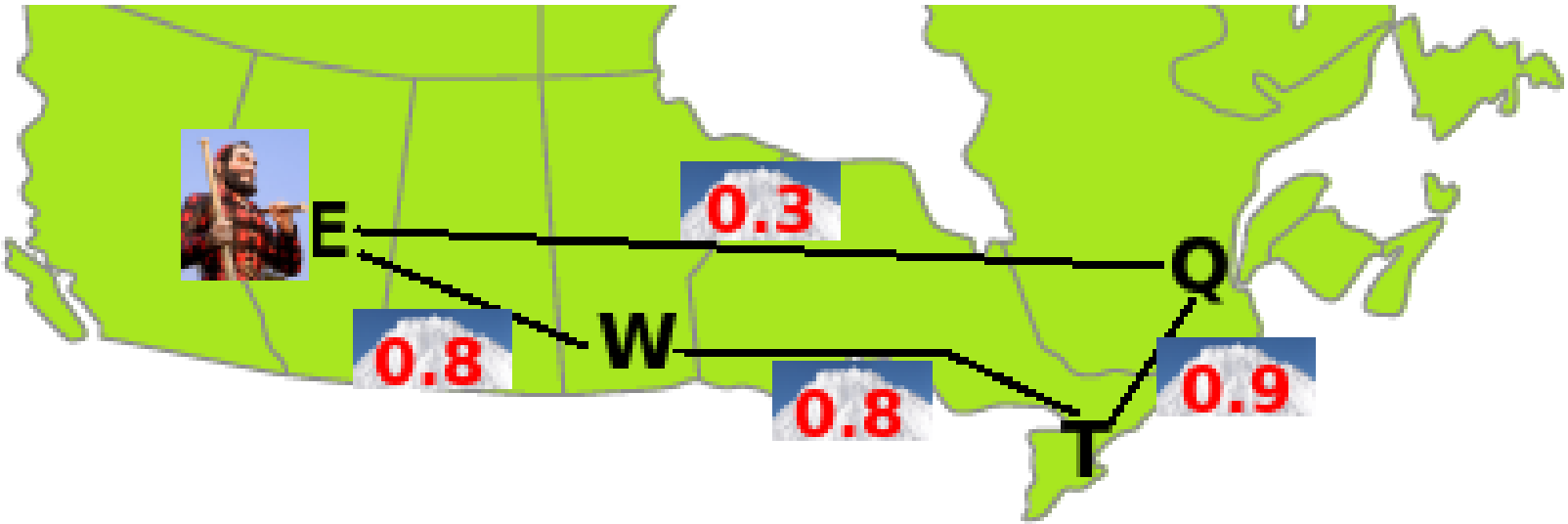
action-outcome uncertainty =

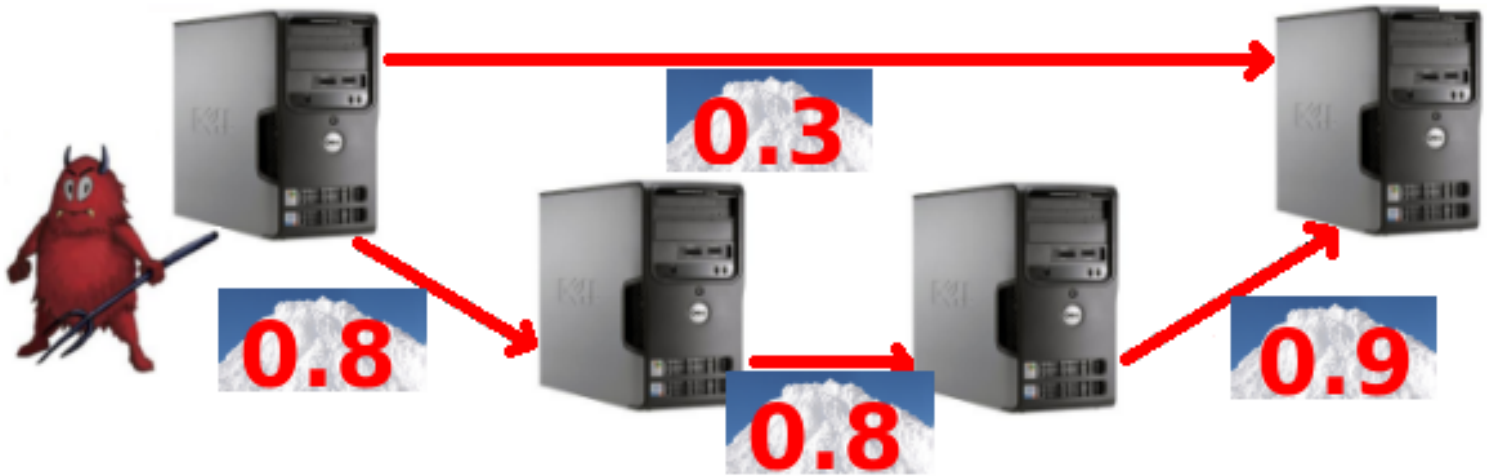


+

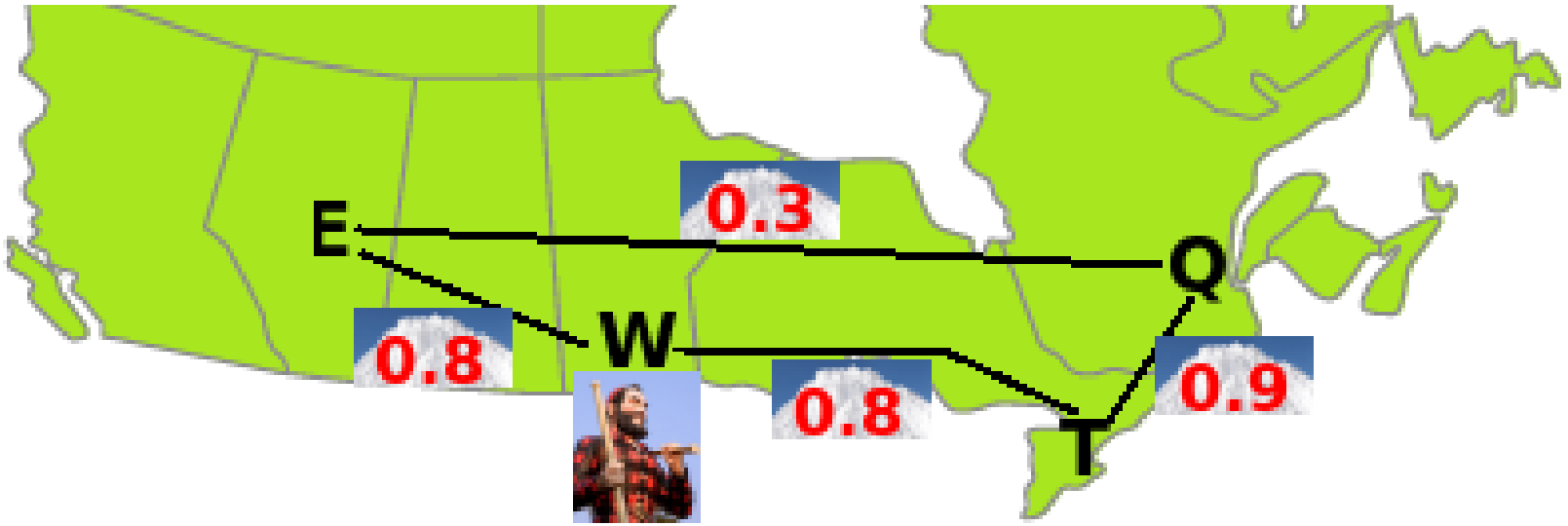
=

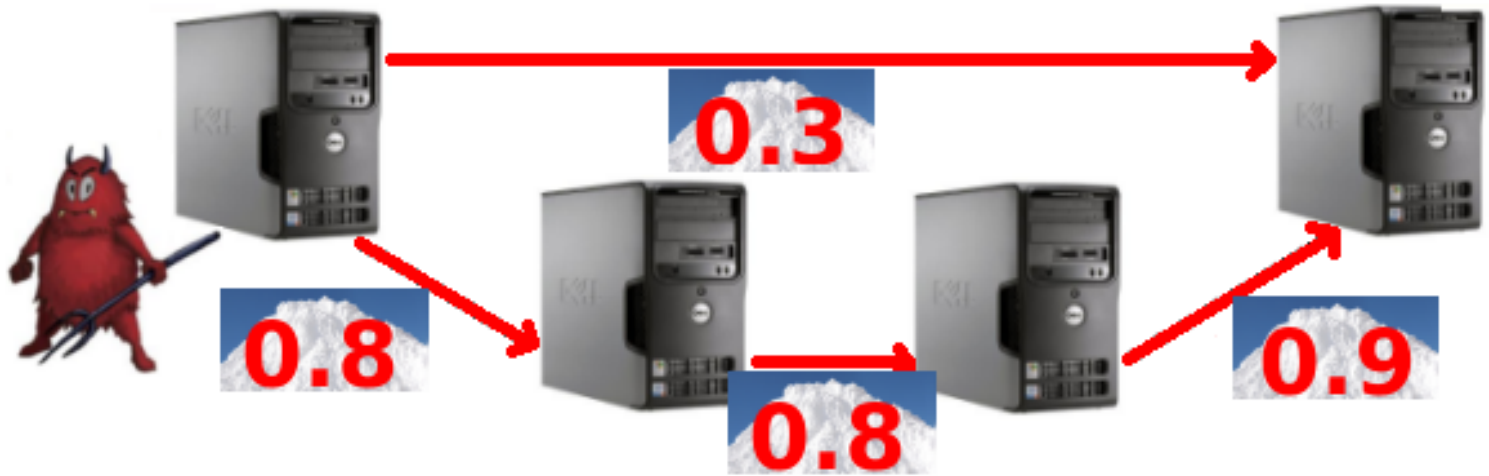
The Canadian Hacker Problem



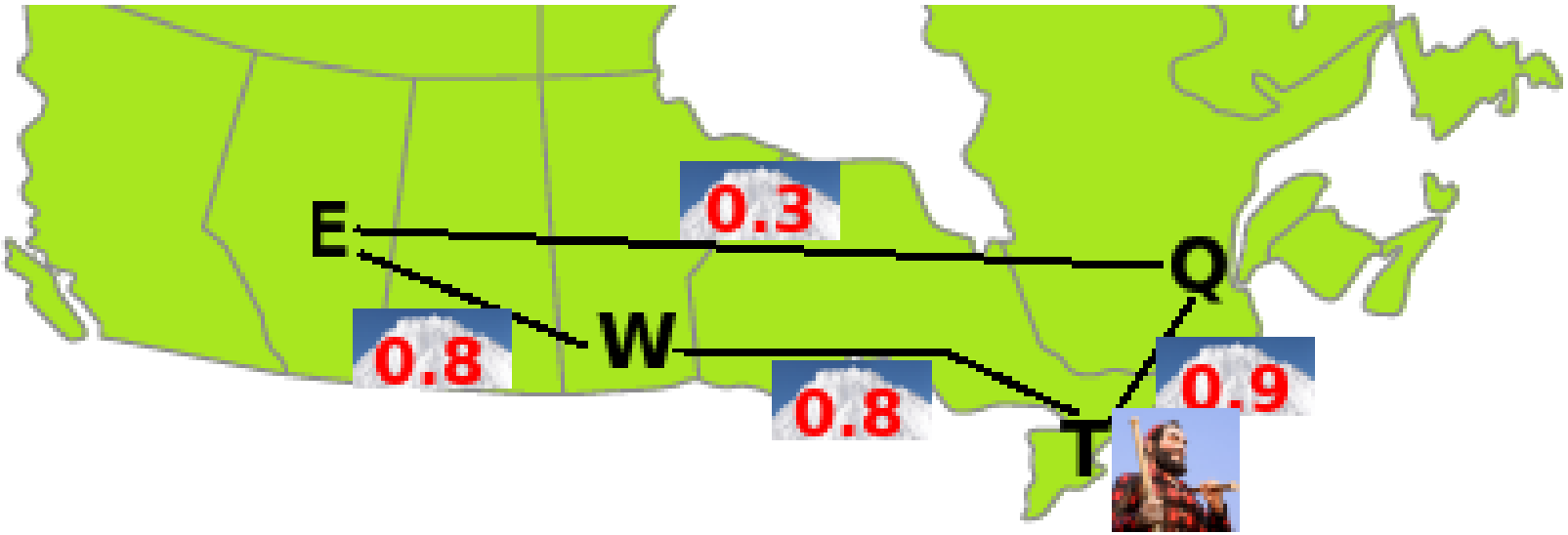


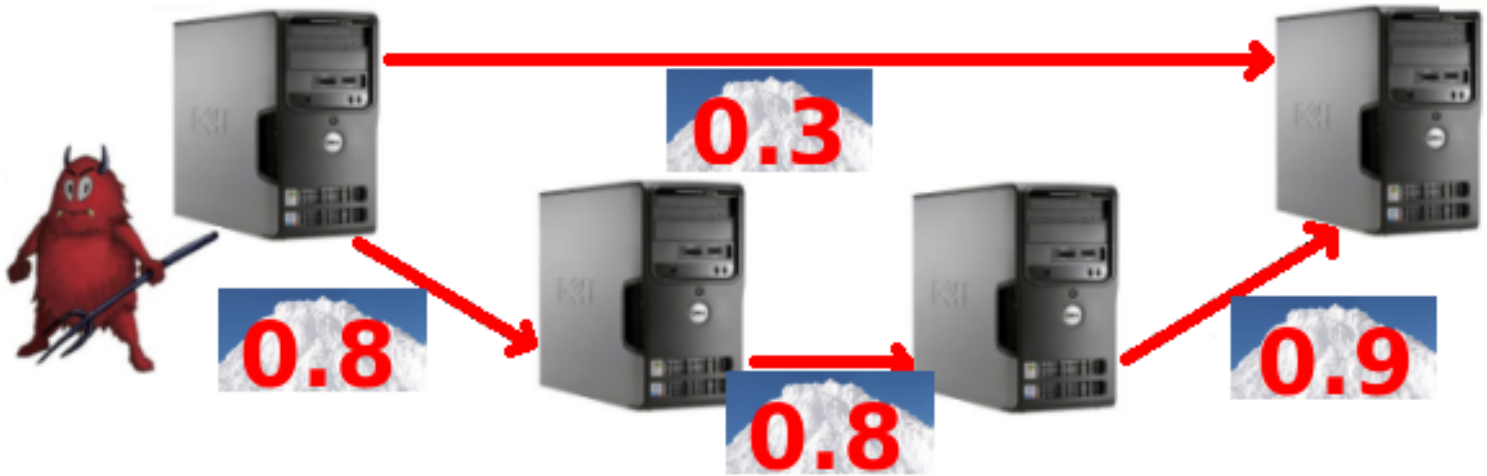
The Canadian Hacker Problem



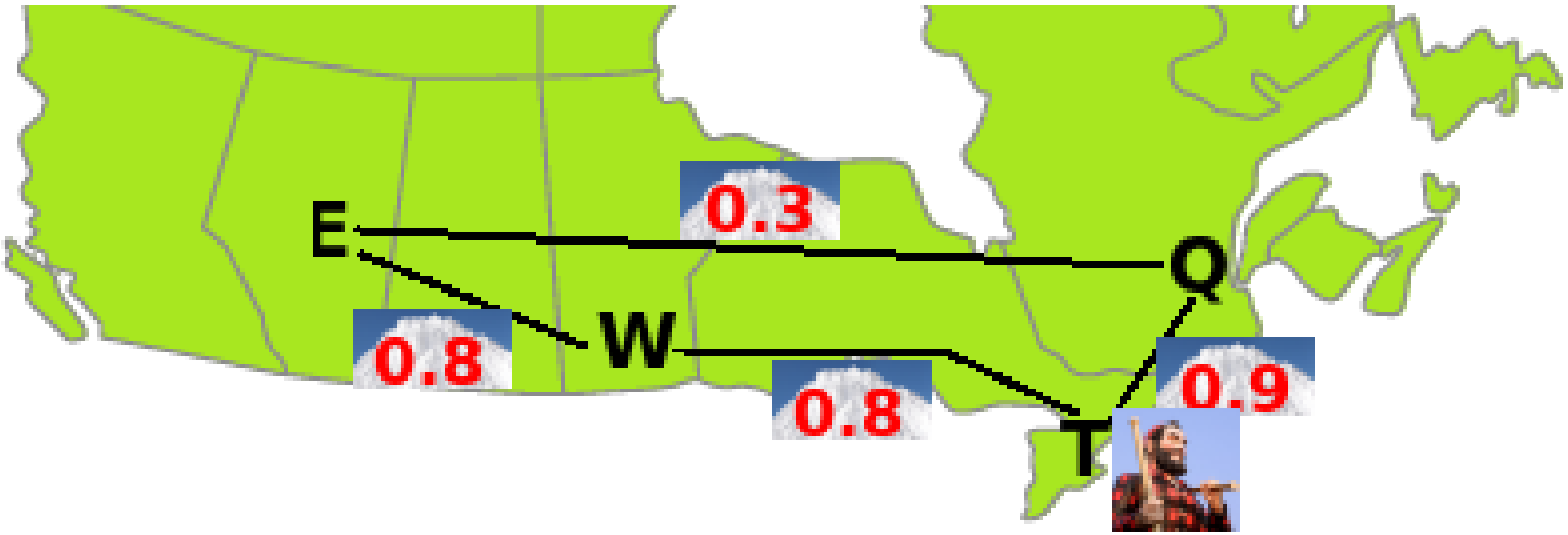


The Canadian Hacker Problem

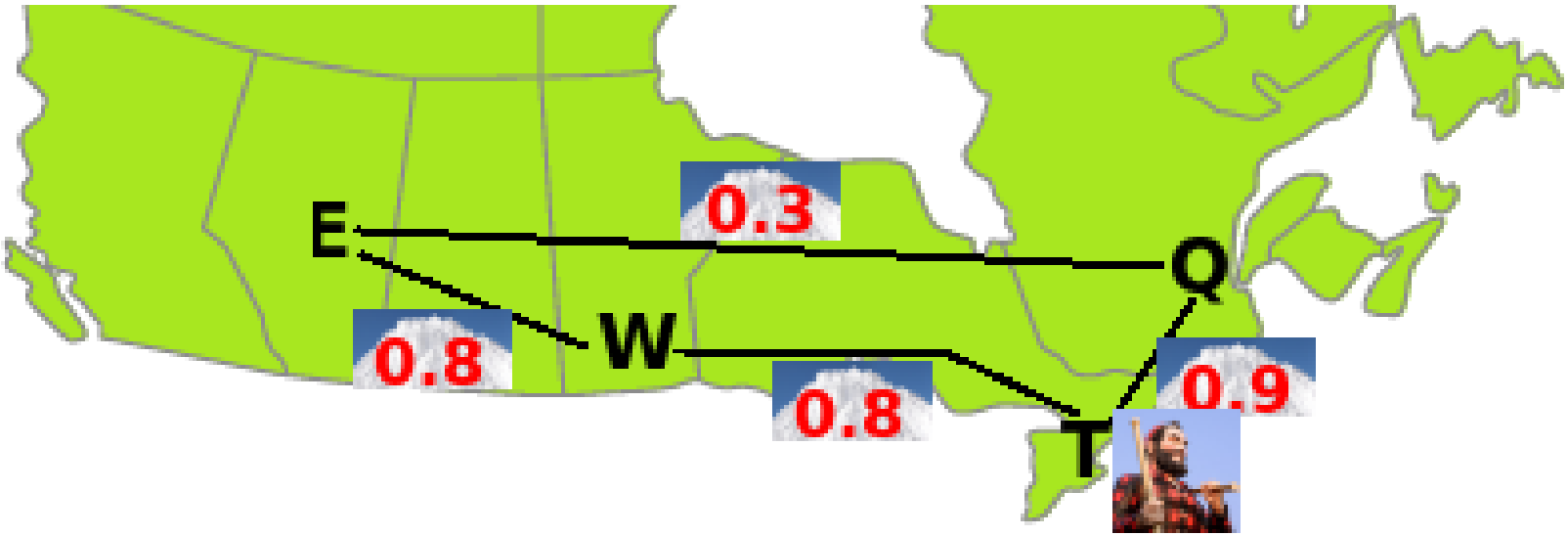




The Canadian Hacker Problem



The Canadian Hacker Problem



The Canadian Hacker Problem: And Now?

Wrap-up: Variant of Canadian Traveller Problem where we "have" a monotonically growing set of nodes ("no need to drive back").

The Canadian Hacker Problem: And Now?

Wrap-up: Variant of Canadian Traveller Problem where we "have" a monotonically growing set of nodes ("no need to drive back").

Research Challenges/Opportunities:

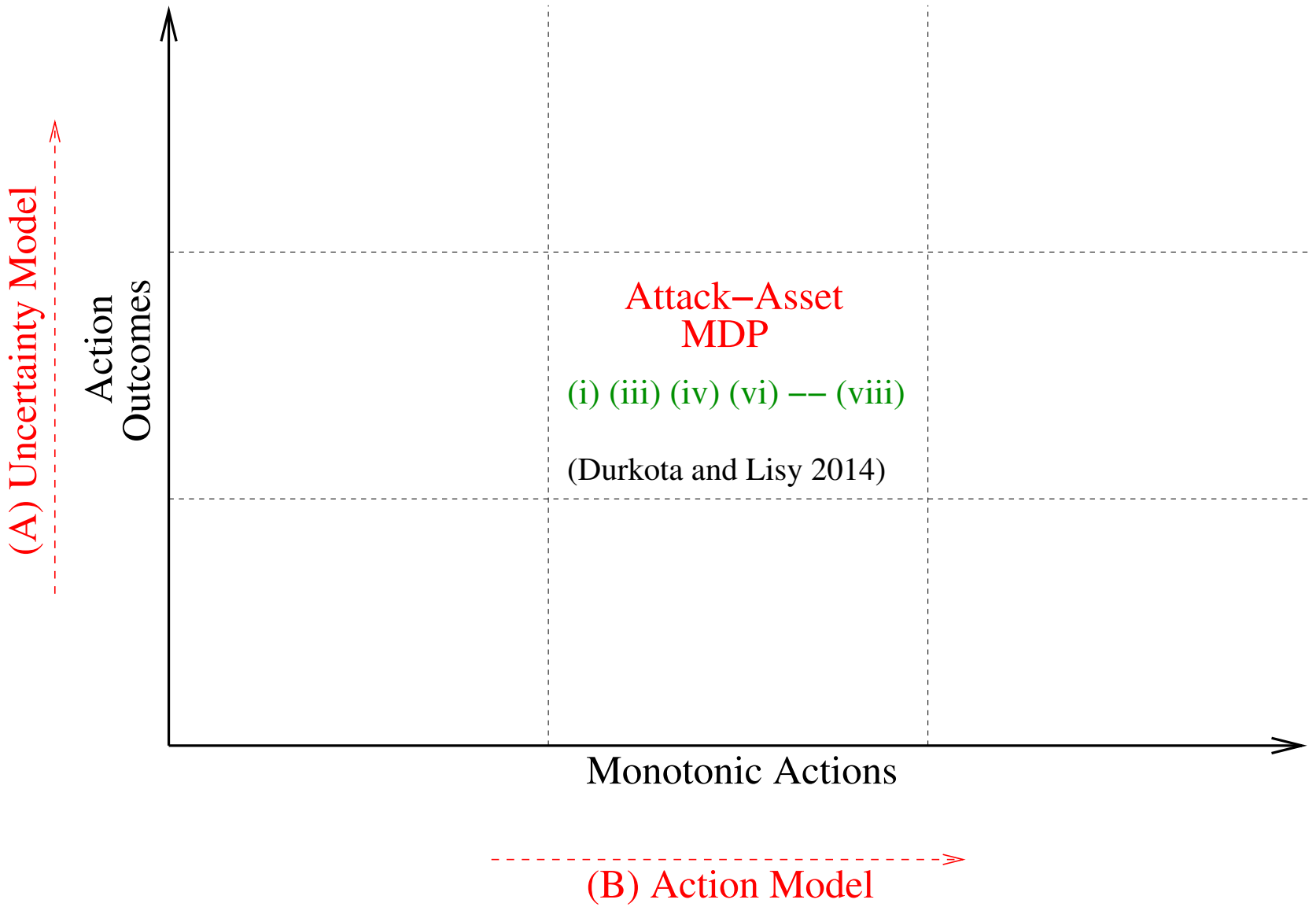
The Canadian Hacker Problem: And Now?

Wrap-up: Variant of Canadian Traveller Problem where we "have" a monotonically growing set of nodes ("no need to drive back").

Research Challenges/Opportunities:

- 1001 CTP papers to be adapted to this ...

Attack-Asset MDPs



Attack-Asset MDPs

Definition

An Attack-Asset MDP is a tuple $\langle \mathcal{P}, \mathcal{A}, s_0, G \rangle$:

- $\mathcal{P}$: set of facts (Boolean state variables).
- $\mathcal{A}$: set of actions a, each a tuple $\langle pre(a), add(a), p(a), c(a) \rangle$ of precondition, add list, success probability, and non-negative cost.
- s_0: initial state; G: goal.

Attack-Asset MDPs

Definition

An Attack-Asset MDP is a tuple $\langle \mathcal{P}, \mathcal{A}, s_0, G \rangle$:

- $\mathcal{P}$: set of facts (Boolean state variables).
- $\mathcal{A}$: set of actions a, each a tuple $\langle pre(a), add(a), p(a), c(a) \rangle$ of precondition, add list, success probability, and non-negative cost.
- s_0: initial state; G: goal.

The probabilistic transitions T arise from these rules:

- States: STRIPS s, available actions $A \subseteq \mathcal{A}$.
- a is applicable to (s, A) if $pre(a) \subseteq s$ and $a \in A$.

Attack-Asset MDPs

Definition

An Attack-Asset MDP is a tuple $\langle \mathcal{P}, \mathcal{A}, s_0, G \rangle$:

- $\mathcal{P}$: set of facts (Boolean state variables).
- $\mathcal{A}$: set of actions a, each a tuple $\langle pre(a), add(a), p(a), c(a) \rangle$ of precondition, add list, success probability, and non-negative cost.
- s_0: initial state; G: goal.

The probabilistic transitions T arise from these rules:

- States: STRIPS s, available actions $A \subseteq \mathcal{A}$.
- a is applicable to (s, A) if $pre(a) \subseteq s$ and $a \in A$.
- With probability $p(a)$ we obtain $s' = s \cup add(a)$, and with probability $1 - p(a)$ we obtain $s' = s$.
- In both cases, we pay cost $c(a)$, and remove a from A.

Attack-Asset MDPs: And Now?

Wrap-up: Probabilistic delete-free STRIPS with success probabilities, no effect in case of failure, each action at most once.

Attack-Asset MDPs: And Now?

Wrap-up: Probabilistic delete-free STRIPS with success probabilities, no effect in case of failure, each action at most once.

Research Challenges/Opportunities:

Attack-Asset MDPs: And Now?

Wrap-up: Probabilistic delete-free STRIPS with success probabilities, no effect in case of failure, each action at most once.

Research Challenges/Opportunities: E.g. determinization.

Attack-Asset MDPs: And Now?

Wrap-up: Probabilistic delete-free STRIPS with success probabilities, no effect in case of failure, each action at most once.

Research Challenges/Opportunities: E.g. determinization.

- Only two outcomes, of which one is "nothing happens".

Attack-Asset MDPs: And Now?

Wrap-up: Probabilistic delete-free STRIPS with success probabilities, no effect in case of failure, each action at most once.

Research Challenges/Opportunities: E.g. determinization.

- Only two outcomes, of which one is "nothing happens".
- Every probabilistic action yields a single deterministic action.

Attack-Asset MDPs: And Now?

Wrap-up: Probabilistic delete-free STRIPS with success probabilities, no effect in case of failure, each action at most once.

Research Challenges/Opportunities: E.g. determinization.

- Only two outcomes, of which one is "nothing happens".
- Every probabilistic action yields a single deterministic action.
- These deterministic actions have no delete effects.

Attack-Asset MDPs: And Now?

Wrap-up: Probabilistic delete-free STRIPS with success probabilities, no effect in case of failure, each action at most once.

Research Challenges/Opportunities: E.g. determinization.

- Only two outcomes, of which one is "nothing happens".
- Every probabilistic action yields a single deterministic action.
- These deterministic actions have no delete effects.
- Weak plans and determinization heuristics = standard delete relaxation heuristics.

Attack-Asset MDPs: And Now?

Wrap-up: Probabilistic delete-free STRIPS with success probabilities, no effect in case of failure, each action at most once.

Research Challenges/Opportunities: E.g. determinization.

- Only two outcomes, of which one is "nothing happens".
- Every probabilistic action yields a single deterministic action.
- These deterministic actions have no delete effects.
- Weak plans and determinization heuristics = standard delete relaxation heuristics.
- "Landmark action outcomes" = deterministic delete-relaxation landmarks.

Attack-Asset MDPs: And Now?

Wrap-up: Probabilistic delete-free STRIPS with success probabilities, no effect in case of failure, each action at most once.

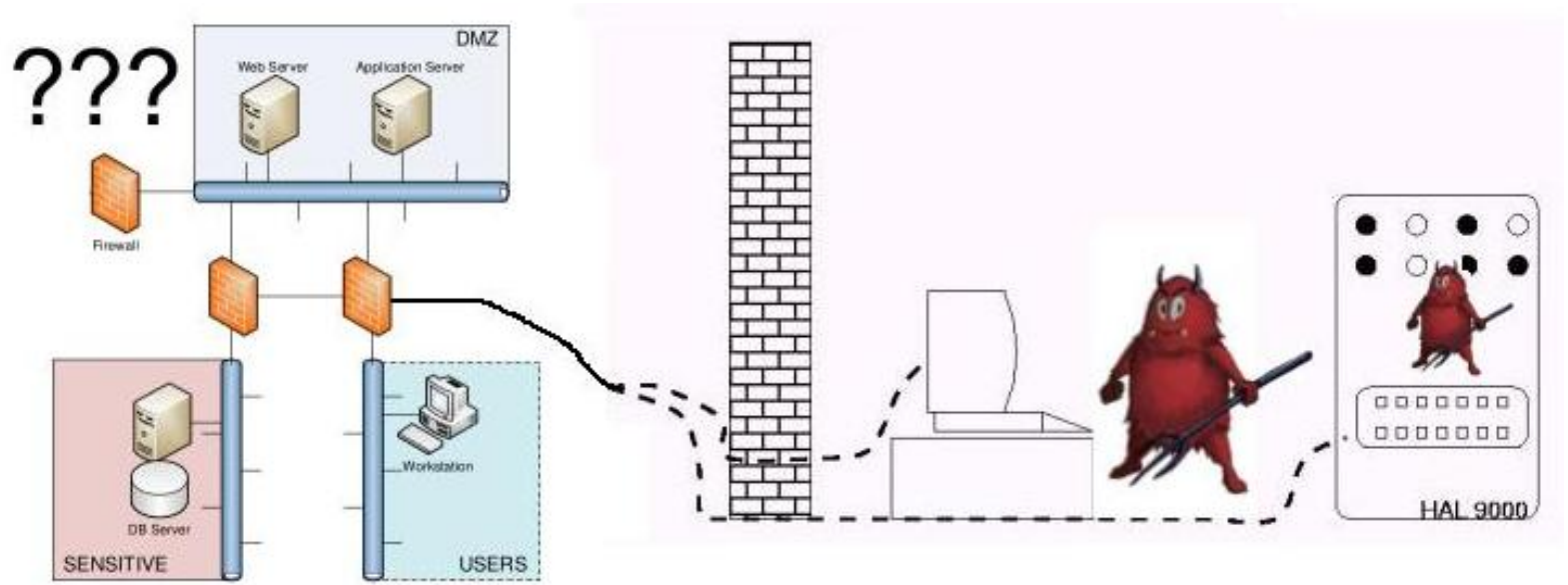
Research Challenges/Opportunities: E.g. determinization.

- Only two outcomes, of which one is "nothing happens".
- Every probabilistic action yields a single deterministic action.
- These deterministic actions have no delete effects.
- Weak plans and determinization heuristics = standard delete relaxation heuristics.
- "Landmark action outcomes" = deterministic delete-relaxation landmarks.
- Limited-budget goal probability maximization: landmarks reduce budget à la [Mirkis and Domshlak (2014)].

Agenda

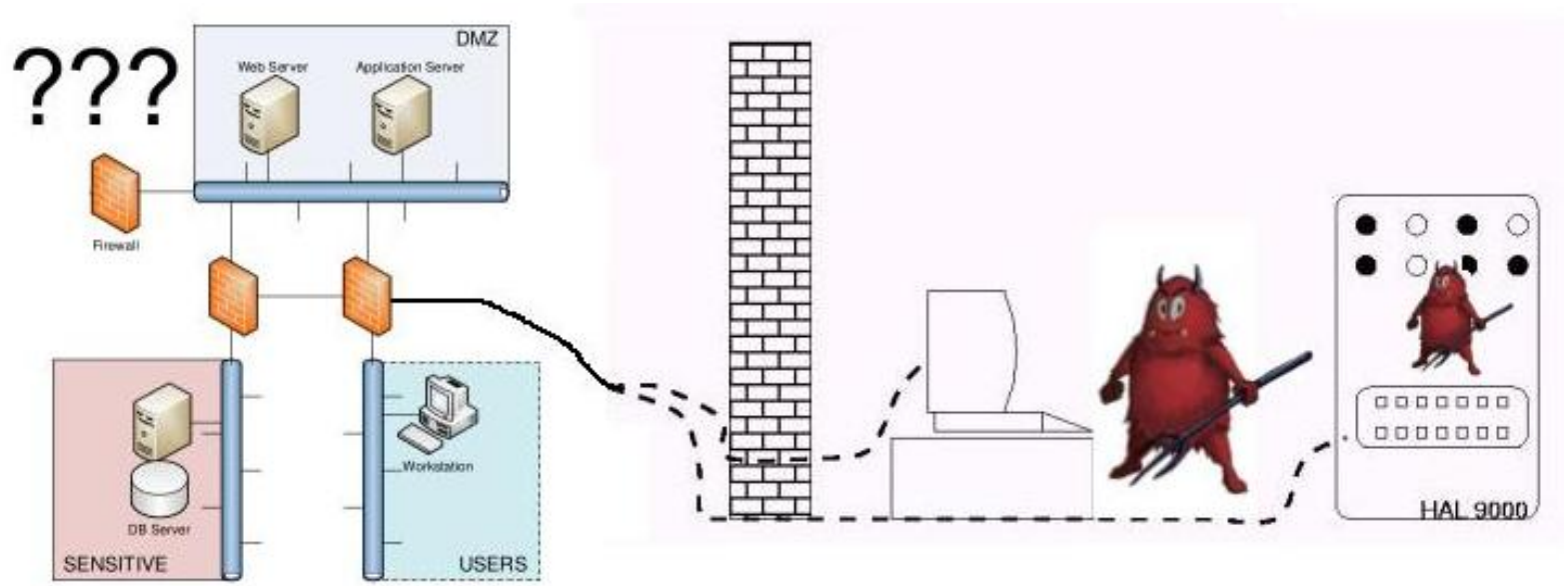
An AI (Sequential Decision Making) Challenge

Realistically simulate a human hacker!



An AI (Sequential Decision Making) Challenge

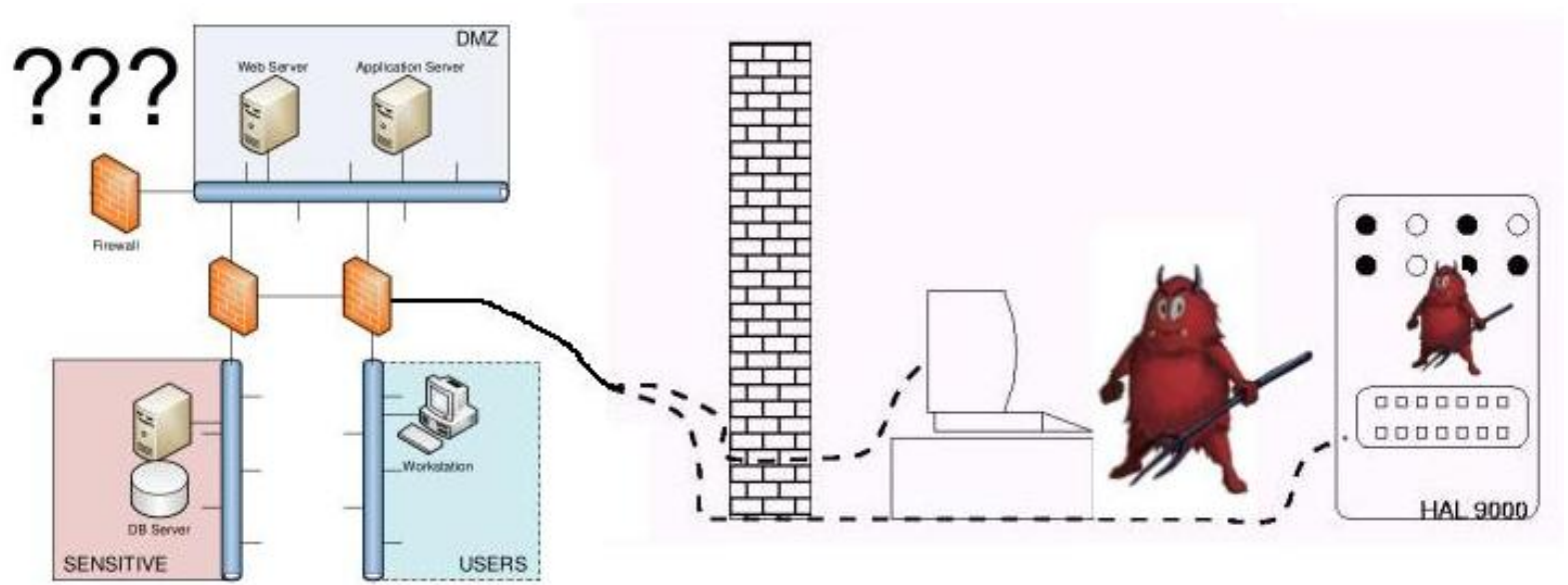
Realistically simulate a human hacker!



- Model and algorithm design in wide space of relevant complexity/accuracy trade-offs.

An AI (Sequential Decision Making) Challenge

Realistically simulate a human hacker!

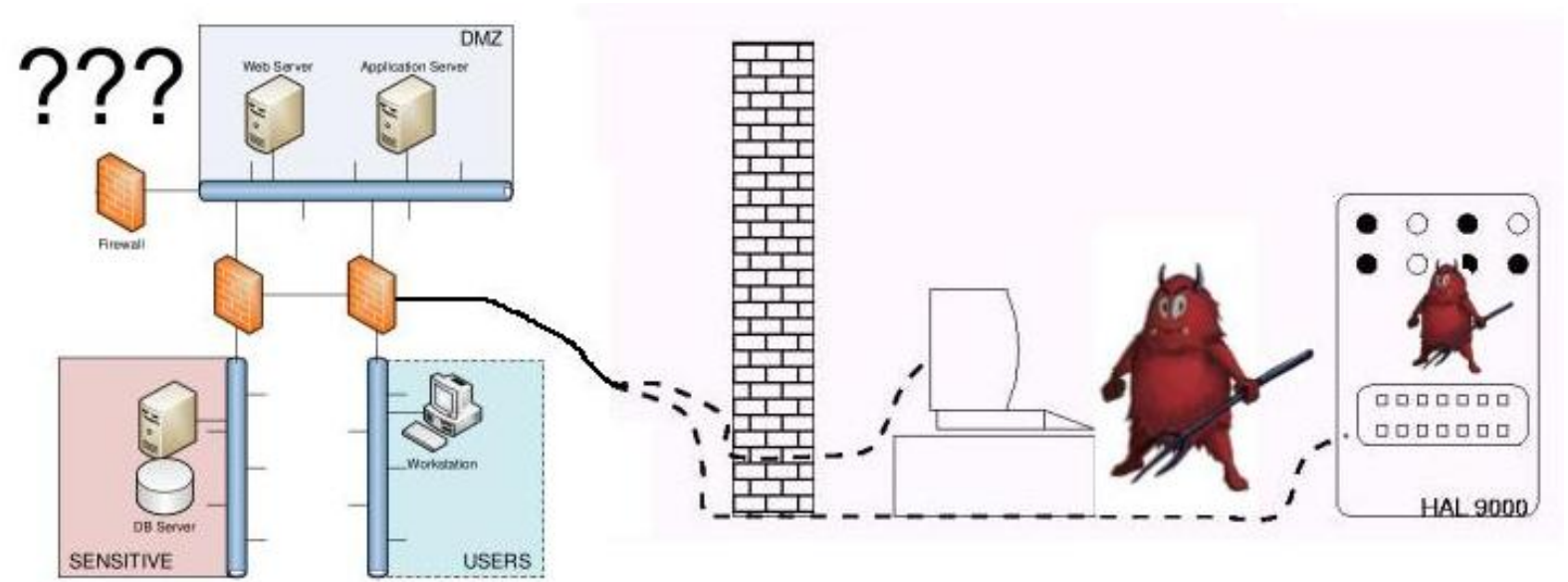


- Model and algorithm design in wide space of relevant complexity/accuracy trade-offs.

  (Sorry Scott – best modeled in PPDDL, at least the MDP variants.)

An AI (Sequential Decision Making) Challenge

Realistically simulate a human hacker!

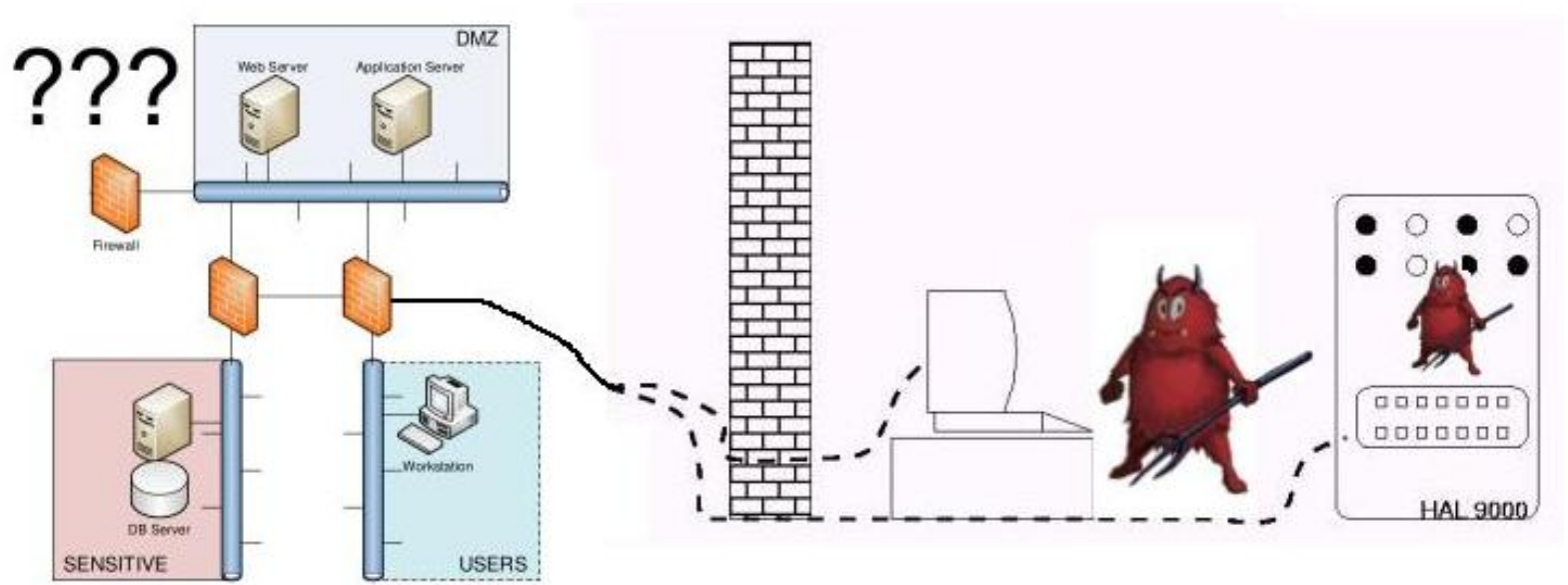


- Model and algorithm design in wide space of relevant complexity/accuracy trade-offs.

  (Sorry Scott – best modeled in PPDDL, at least the MDP variants.)
- Diverse attacks,

An AI (Sequential Decision Making) Challenge

Realistically simulate a human hacker!

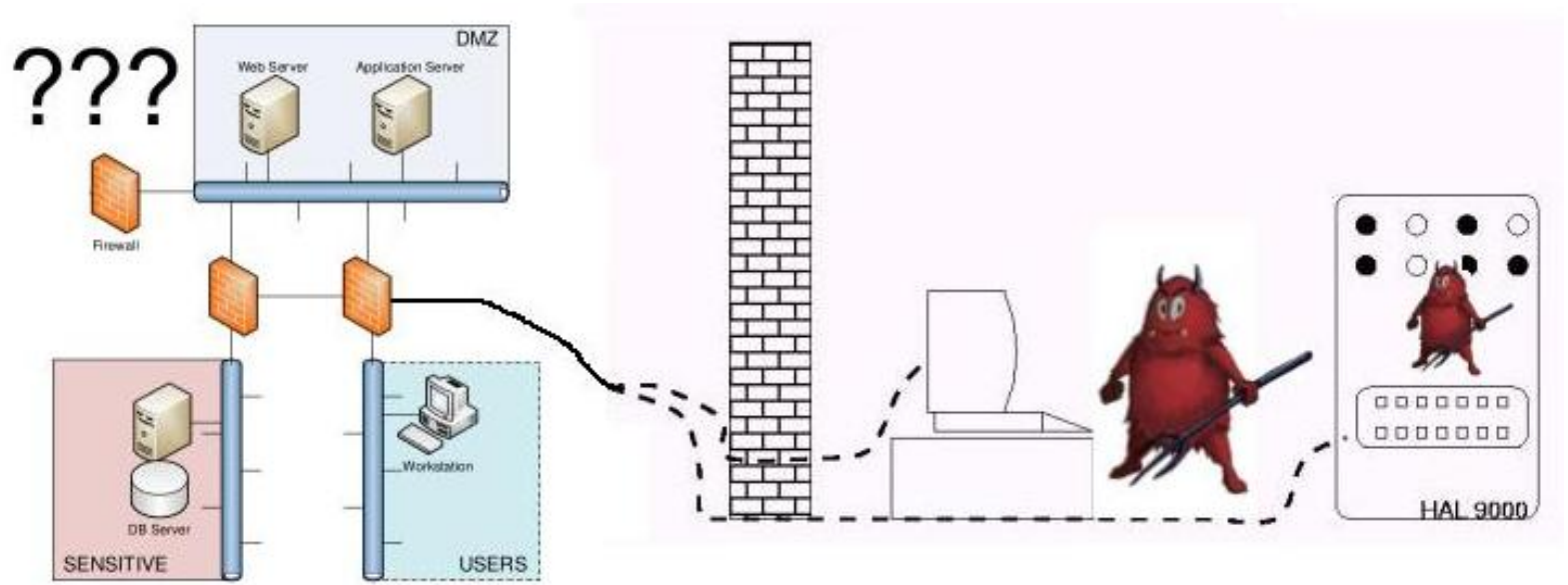


- Model and algorithm design in wide space of relevant complexity/accuracy trade-offs.

  (Sorry Scott – best modeled in PPDDL, at least the MDP variants.)
- Diverse attacks, meta-criteria,

An AI (Sequential Decision Making) Challenge

Realistically simulate a human hacker!

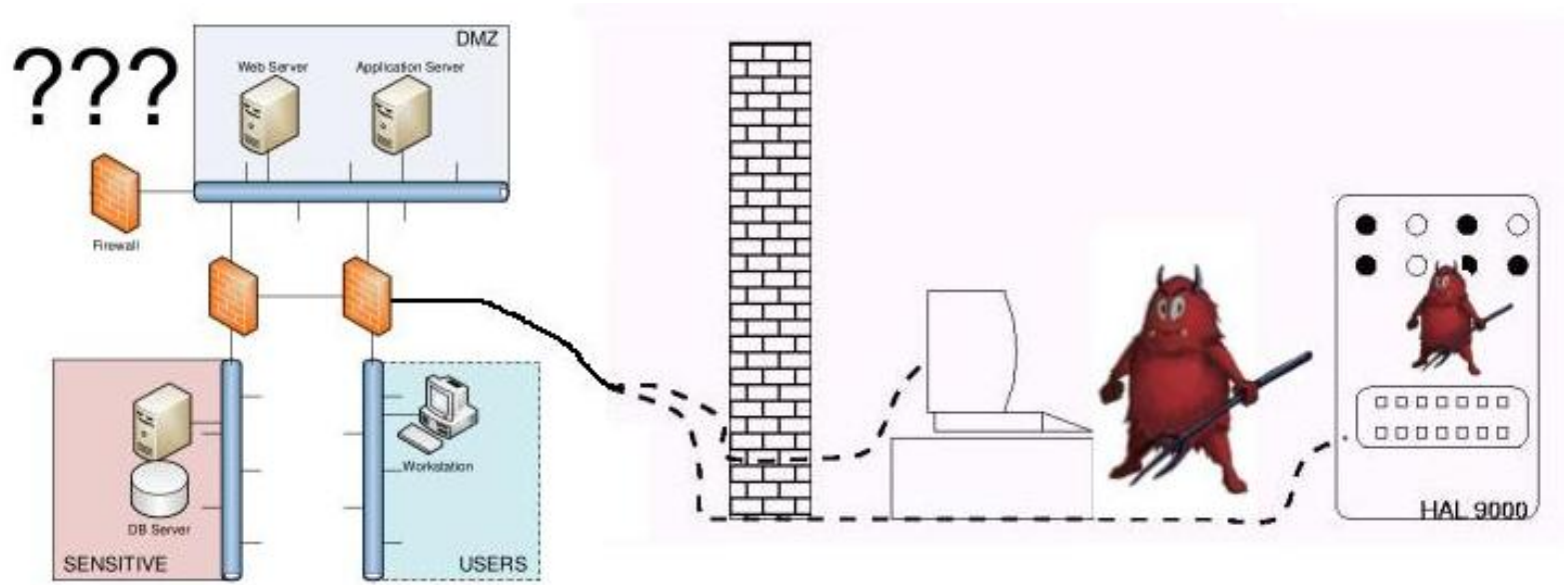


- Model and algorithm design in wide space of relevant complexity/accuracy trade-offs.

  (Sorry Scott – best modeled in PPDDL, at least the MDP variants.)
- Diverse attacks, meta-criteria, situation report,

An AI (Sequential Decision Making) Challenge

Realistically simulate a human hacker!

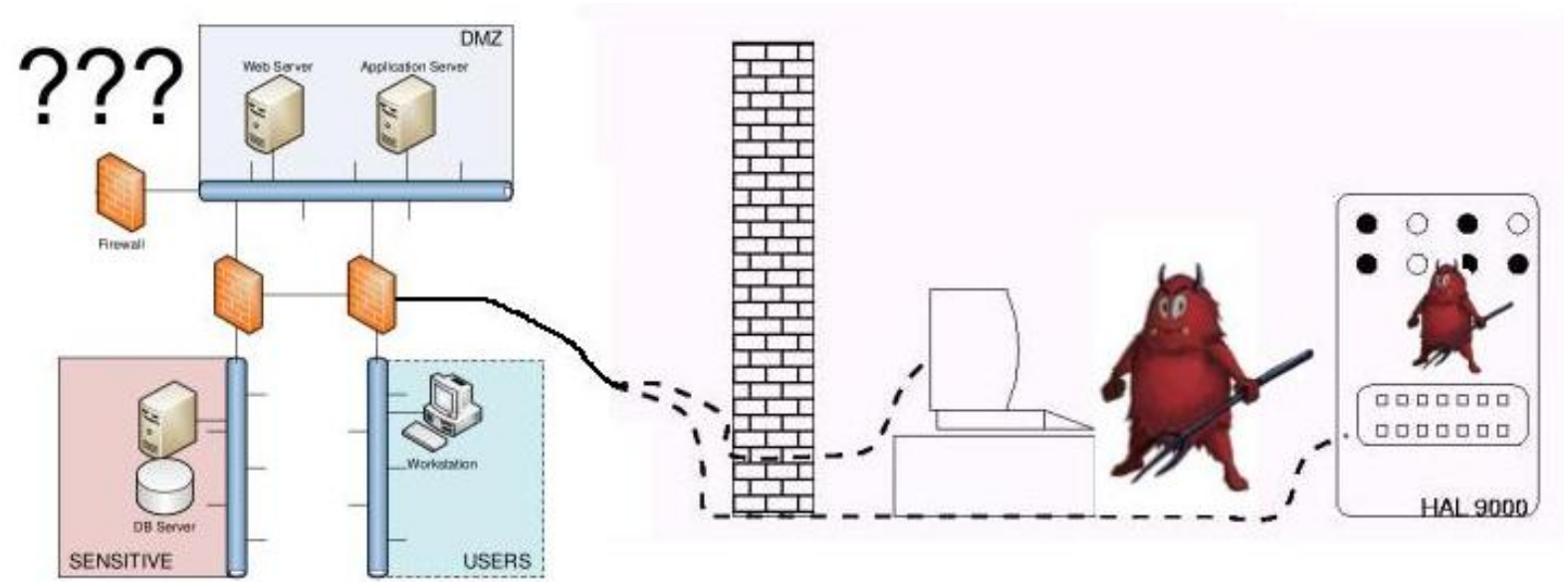


- Model and algorithm design in wide space of relevant complexity/accuracy trade-offs.

  (Sorry Scott – best modeled in PPDDL, at least the MDP variants.)
- Diverse attacks, meta-criteria, situation report, suggest fixes.

An AI (Sequential Decision Making) Challenge

Realistically simulate a human hacker!



- Model and algorithm design in wide space of relevant complexity/accuracy trade-offs.

  (Sorry Scott – best modeled in PPDDL, at least the MDP variants.)
- Diverse attacks, meta-criteria, situation report, suggest fixes.
- Ultimately, an AI-complete problem.

Thanks for Your Attention!

. . . and enjoy the old city tour.

References I

Paul Ammann, Duminda Wijesekera, and Saket Kaushik. Scalable, graph-based network vulnerability analysis. In *ACM Conference on Computer and Communications Security*, pages 217–224, 2002.

Mark Boddy, Jonathan Gohde, Tom Haigh, and Steven Harp. Course of action generation for cyber security using classical planning. In Susanne Biundo, Karen Myers, and Kanna Rajan, editors, *Proceedings of the 15th International Conference on Automated Planning and Scheduling (ICAPS-05)*, pages 12–21, Monterey, CA, USA, 2005. Morgan Kaufmann.

Rainer Böhme and Márk Félegyházi. Optimal information security investment with penetration testing. In *Proceedings of the 1st International Conference on Decision and Game Theory for Security (GameSec'10)*, pages 21–37, 2010.

Tom Bylander. The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69(1–2):165–204, 1994.

Russell Greiner, Ryan Hayward, Magdalena Jankowska, and Michael Molloy. Finding optimal satisficing strategies for and-or trees. *Artificial Intelligence*, 170(1):19–58, 2006.

References II

Russell Greiner. Finding optimal derivation strategies in redundant knowledge bases. *Artificial Intelligence*, 50(1):95–115, 1991.

Jörg Hoffmann. The Metric-FF planning system: Translating "ignoring delete lists" to numeric state variables. *Journal of Artificial Intelligence Research*, 20:291–341, 2003.

Markus Huber, Stewart Kowalski, Marcus Nohlberg, and Simon Tjoa. Towards automating social engineering using social networking sites. In *Proceedings of the 12th IEEE International Conference on Computational Science and Engineering (CSE'09)*, pages 117–124. IEEE Computer Society, 2009.

Emil Keyder and Hector Geffner. Trees of shortest paths vs. Steiner trees: Understanding and improving delete relaxation heuristics. In C. Boutilier, editor, *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009)*, pages 1734–1739, Pasadena, California, USA, July 2009. Morgan Kaufmann.

Barbara Kordy, Sjouke Mauw, Sasa Radomirovic, and Patrick Schweitzer. Foundations of attack-defense trees. In *Proceedings of the 7th International Workshop on Formal Aspects in Security and Trust (FAST'10)*, pages 80–95, 2010.

References III

Barbara Kordy, Piotr Kordy, Sjouke Mauw, and Patrick Schweitzer. ADTool: security analysis with attack-defense trees. In *Proceedings of the 10th International Conference on Quantitative Evaluation of Systems (QEST'13)*, pages 173–176, 2013.

Viliam Lisý and Radek Píbil. Computing optimal attack strategies using unconstrained influence diagrams. In *Pacific Asia Workshop on Intelligence and Security Informatics*, pages 38–46, 2013.

Jorge Lucangeli, Carlos Sarraute, and Gerardo Richarte. Attack planning in the real world. In *Proceedings of the 2nd Workshop on Intelligent Security (SecArt'10)*, 2010.

Vitaly Mirkis and Carmel Domshlak. Landmarks in oversubscription planning. In Thorsten Schaub, editor, *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI'14)*, pages 633–638, Prague, Czech Republic, August 2014. IOS Press.

Steven Noel, Matthew Elder, Sushil Jajodia, Pramod Kalapa, Scott O'Hare, and Kenneth Prole. Advances in topological vulnerability analysis. In *Proceedings of the 2009 Cybersecurity Applications & Technology Conference for Homeland Security (CATCH'09)*, pages 124–129, 2009.

References IV

Ronald W. Ritchey and Paul Ammann. Using model checking to analyze network vulnerabilities. In *IEEE Symposium on Security and Privacy*, pages 156–165, 2000.

Carlos Sarraute, Olivier Buffet, and Jörg Hoffmann. POMDPs make better hackers: Accounting for uncertainty in penetration testing. In Jörg Hoffmann and Bart Selman, editors, *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI'12)*, pages 1816–1824, Toronto, ON, Canada, July 2012. AAAI Press.

B. Schneier. Attack trees. *Dr. Dobbs Journal*, 1999.

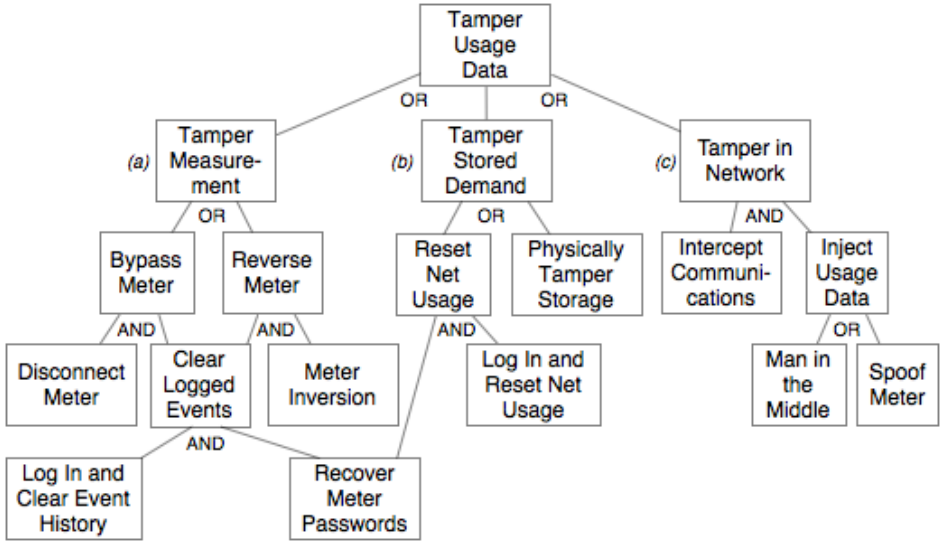
Milind Tambe. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*. Cambridge University Press, 2011.

Steven J. Templeton and Karl E. Levitt. A requires/provides model for computer attacks. In *Proceedings of the Workshop on New Security Paradigms (NSPW'00)*, pages 31–38, 2000.

Attack Trees

Community: Application-oriented security, some academic research.

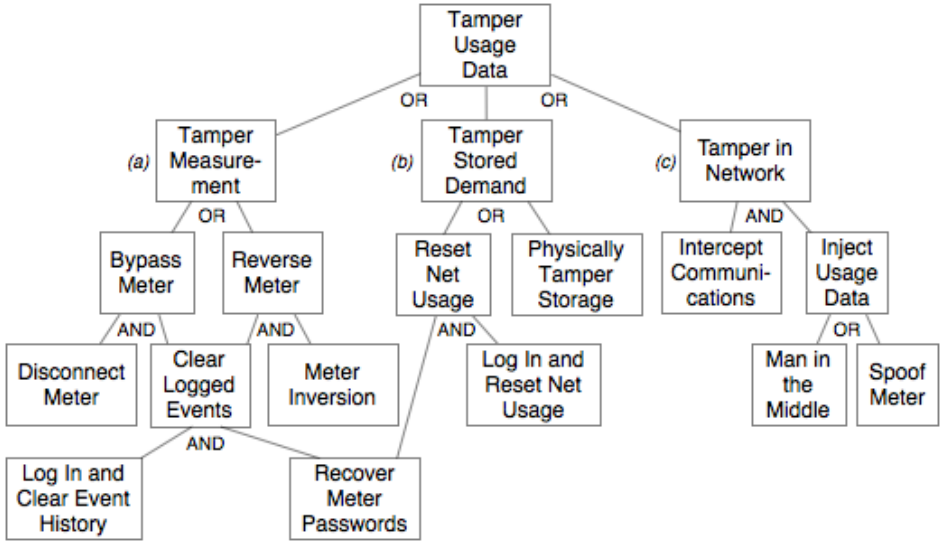
Approach: "Graphical Security Models". Organize known possible attacks by top-down refinement over attack actions and sub-actions.



Attack Trees

Community: Application-oriented security, some academic research.

Approach: "Graphical Security Models". Organize known possible attacks by top-down refinement over attack actions and sub-actions.

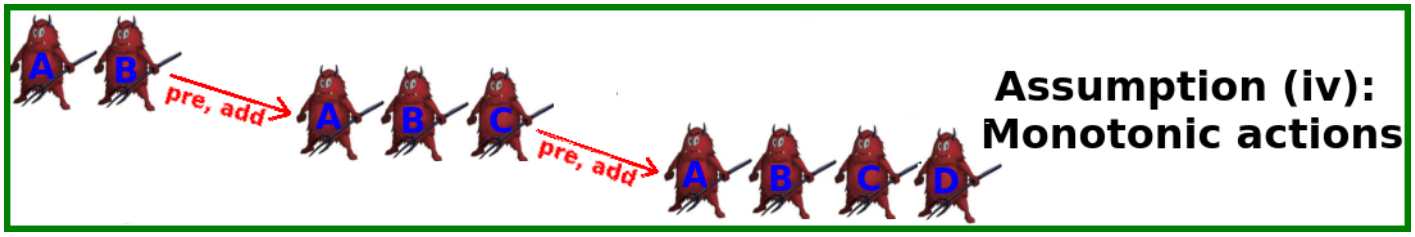


On the side: Many attack tree models are equivalent to AI "formula evaluation" [e. g. Greiner (1991); Greiner *et al.* (2006)]. Apparently unnoticed by both communities; pointed out by Lisý and Píbil (2013).

Dimension (B): In Other Words

Explicit Network Graph: Actions = "hops from ?s to ?t".

Monotonic actions: Attacker can only gain new attack assests.

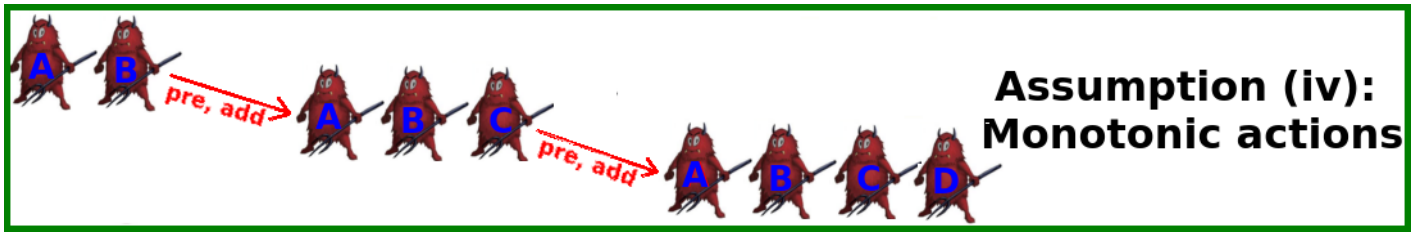


General actions: No restrictions.

Dimension (B): In Other Words

Explicit Network Graph: Actions = "hops from ?s to ?t".

Monotonic actions: Attacker can only gain new attack assests.



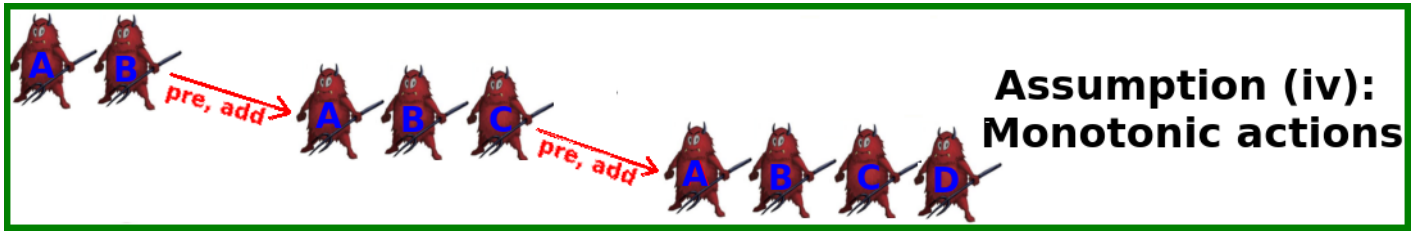
General actions: No restrictions.

→ Note that (v) implies (iv).

Dimension (B): In Other Words

Explicit Network Graph: Actions = "hops from ?s to ?t".

Monotonic actions: Attacker can only gain new attack assests.



General actions: No restrictions.

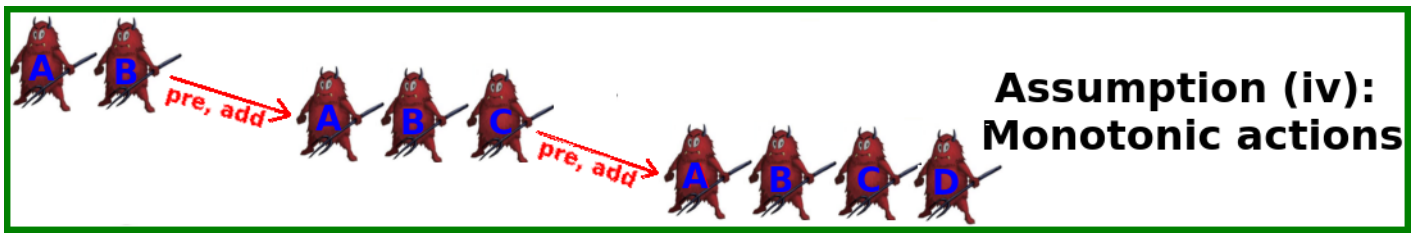
→ Note that (v) implies (iv). And each of (iv) and (v) implies (iii):

Dimension (B) Assumptions: Overview

Explicit Network Graph: Actions = "hops from ?s to ?t".

Relax: More general attack assets (software/passwords . . .).

Monotonic actions: Attacker can only gain new attack assests.



Relax: E.g. detrimental side effects, crashing the host.

Static network: Host connections & configurations not affected.

Relax: E.g. detrimental side effects, crashing the host.

Game-Theoretic Models

What about modeling the defender?

Game-Theoretic Models

What about modeling the defender?

My 5 cents:

- How to get realistic models? Is a network intrusion actually a game? $\rightarrow$ Typically mentioned, if at all, as "detection risk" as in "potential detrimental side effect of an attack action".

Game-Theoretic Models

What about modeling the defender?

My 5 cents:

- How to get realistic models? Is a network intrusion actually a game? $\rightarrow$ Typically mentioned, if at all, as "detection risk" as in "potential detrimental side effect of an attack action".
- GameSec series `http://www.gamesec-conf.org/`

Game-Theoretic Models

What about modeling the defender?

My 5 cents:

- How to get realistic models? Is a network intrusion actually a game?
  $\rightarrow$ Typically mentioned, if at all, as "detection risk" as in "potential detrimental side effect of an attack action".
- GameSec series `http://www.gamesec-conf.org/`
  Böhme and Félegyházi (2010) introduce a model of the entire pentesting life cycle, and prove that pentesting pays off.

Game-Theoretic Models

What about modeling the defender?

My 5 cents:

- How to get realistic models? Is a network intrusion actually a game?
  $\rightarrow$ Typically mentioned, if at all, as "detection risk" as in "potential detrimental side effect of an attack action".
- GameSec series `http://www.gamesec-conf.org/`
  Böhme and Félegyházi (2010) introduce a model of the entire pentesting life cycle, and prove that pentesting pays off.
  Attack-defense trees [Kordy *et al.* (2010, 2013)].

Game-Theoretic Models

What about modeling the defender?

My 5 cents:

- How to get realistic models? Is a network intrusion actually a game?
  $\rightarrow$ Typically mentioned, if at all, as "detection risk" as in "potential detrimental side effect of an attack action".
- GameSec series `http://www.gamesec-conf.org/`
  Böhme and Félegyházi (2010) introduce a model of the entire pentesting life cycle, and prove that pentesting pays off.
  Attack-defense trees [Kordy *et al.* (2010, 2013)].
- Security games (e. g. Tambe (2011)): Completely different application.