Evolving Secure Information Systems through Attack Simulation

Elmar Kiesling, Andreas Ekelhart, Bernhard Grill,
Christine Strauß, Christian Stummer

January 7, 2014; Waikoloa, Big Island, Hawaii

Funded by the Austrian Science Fund under project number P 23122-N23

Core ideas

Security is. . .

- not the result of any particular technical measure
- a system property that emerges from **interactions**
- not an absolute concept, but involves **tradeoffs**
- meaningless without a specific **threat model**

"Best" approach to secure a system is highly **context-dependent**:

- system characteristics
- threat landscape
- available resources
- decision-makers' risk preferences

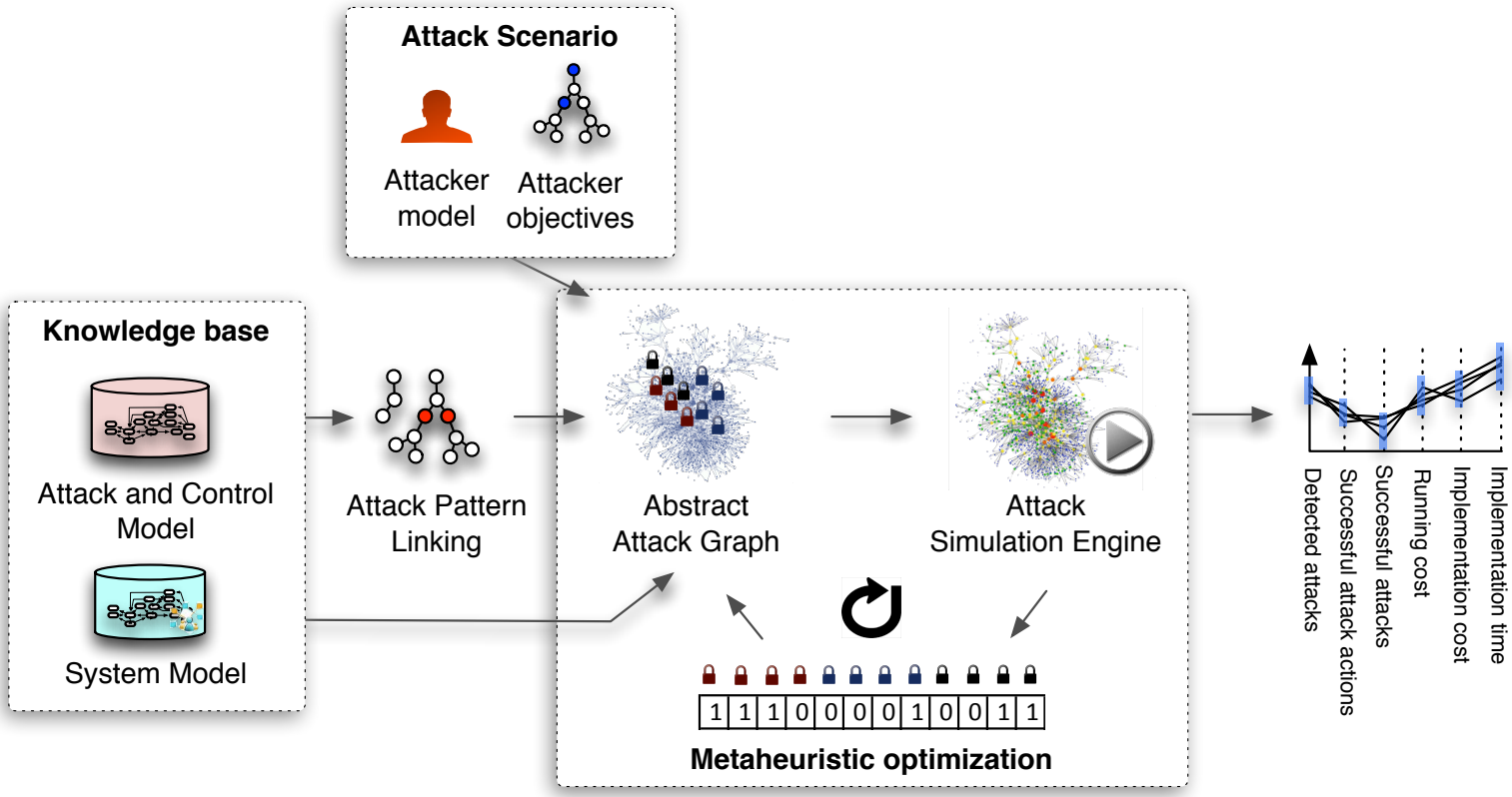
Problem definition and approach

Objective: choose an "optimal" set of security controls

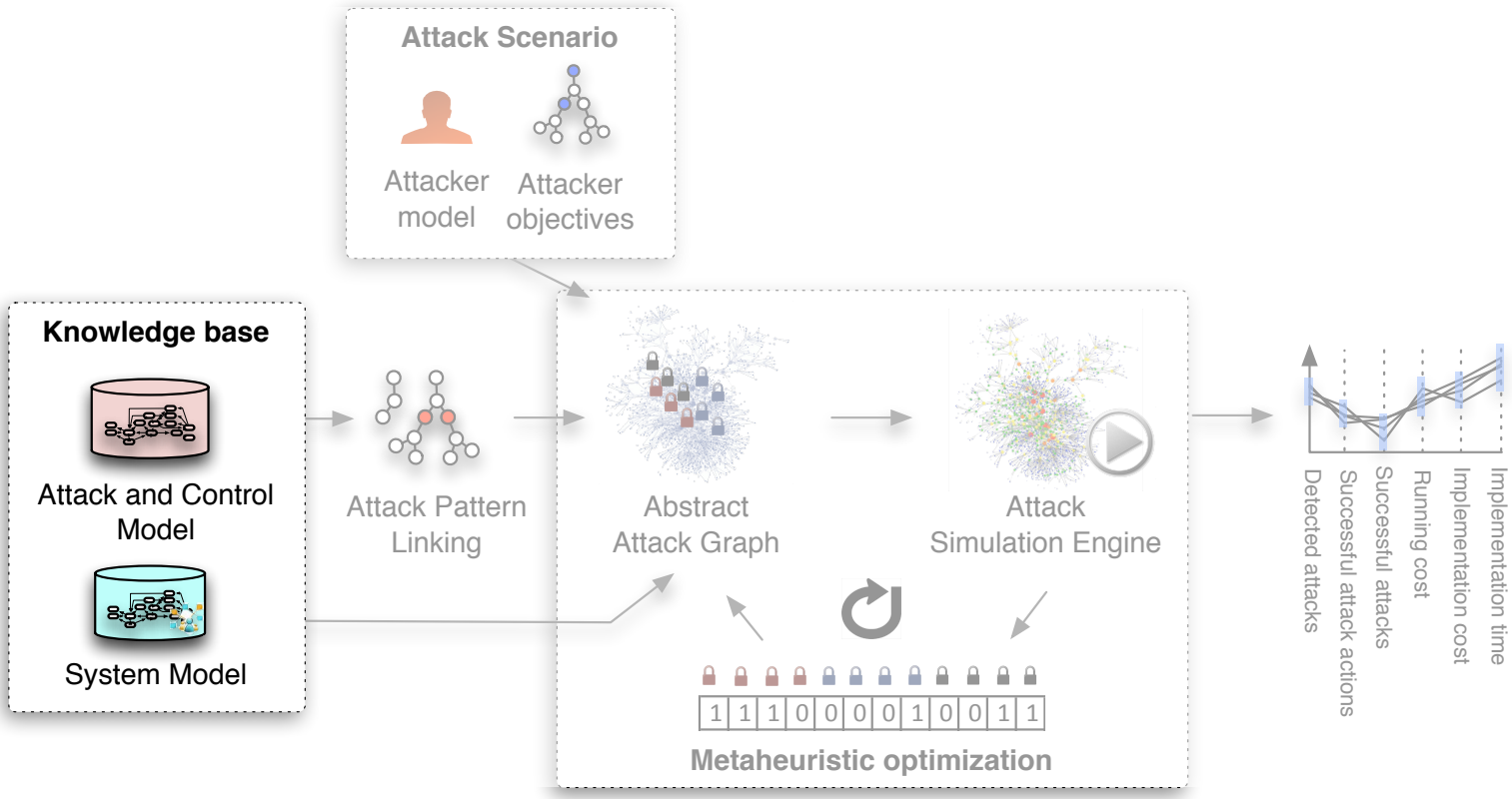
Solution approach:

1. **Model**
   - **a)** abstract causal interdependencies
   - **b)** the information system and its context
   - **c)** adversaries and their behavior
2. Apply sets of security controls and **simulate** attacks
3. **Optimize** control sets w.r.t. multiple objectives
4. Support decision-maker in the selection of control

Overview

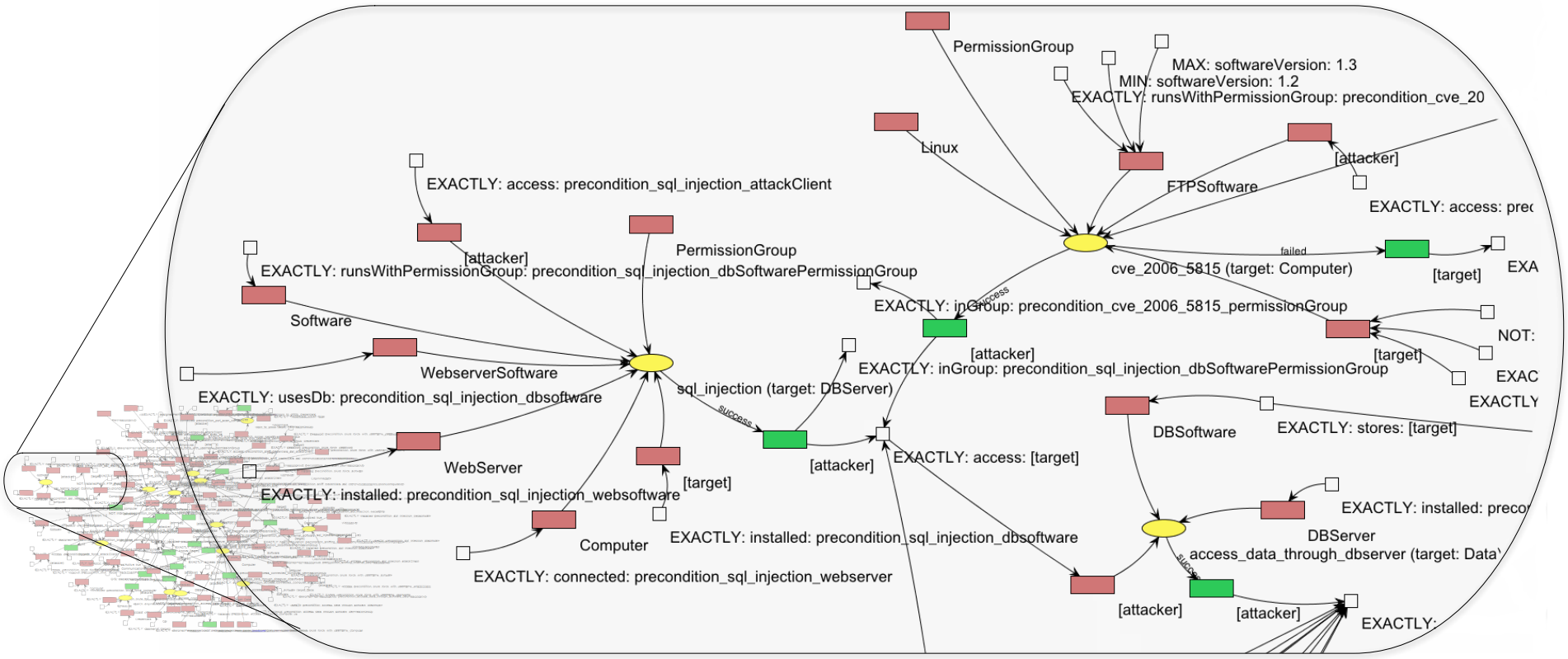

Attack Scenario
Attacker model
Attacker objectives
Knowledge base
Attack and Control Model
System Model
Attack Pattern Linking
Abstract Attack Graph
Attack Simulation Engine
1 1 1 0 0 0 0 1 0 0 1 1
Metaheuristic optimization
Detected attacks
Successful attack actions
Successful attacks
Running cost
Implementation cost
Implementation time

Knowledge base



Knowledge base

- Captures abstract attack knowledge
- Actions linked through pre- and post-conditions



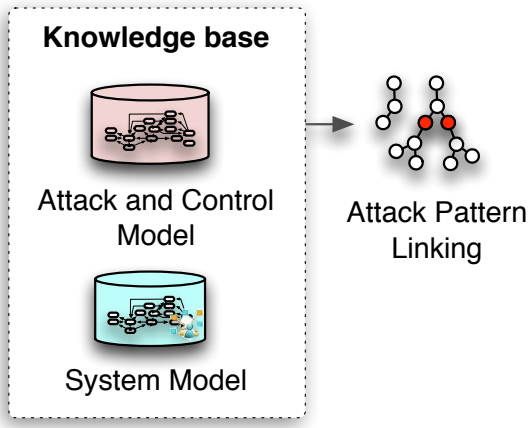
Atomic attack actions

Pre-Conditions

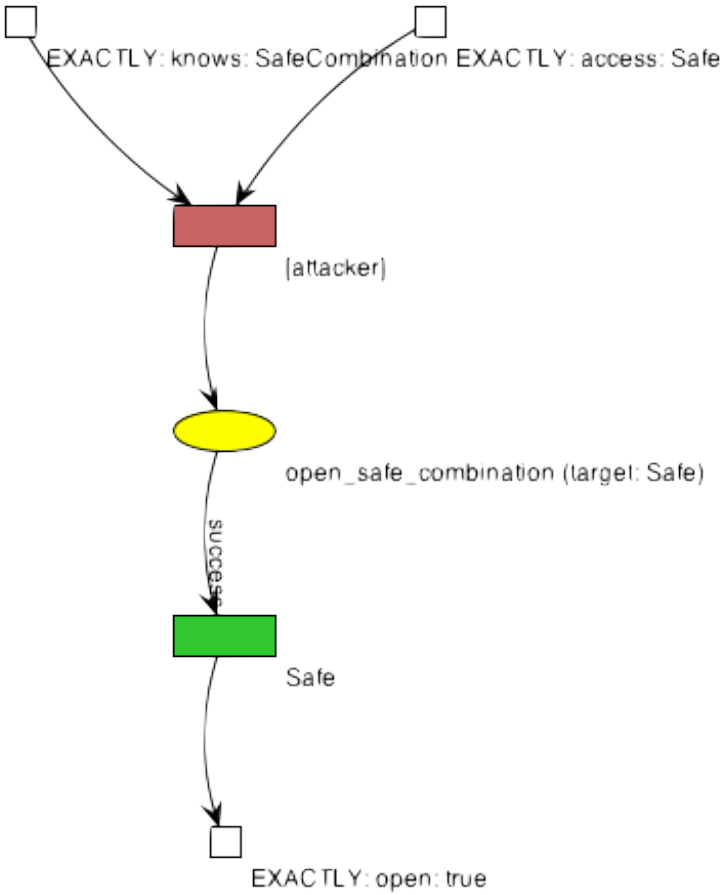
Condition properties

Post-Conditions

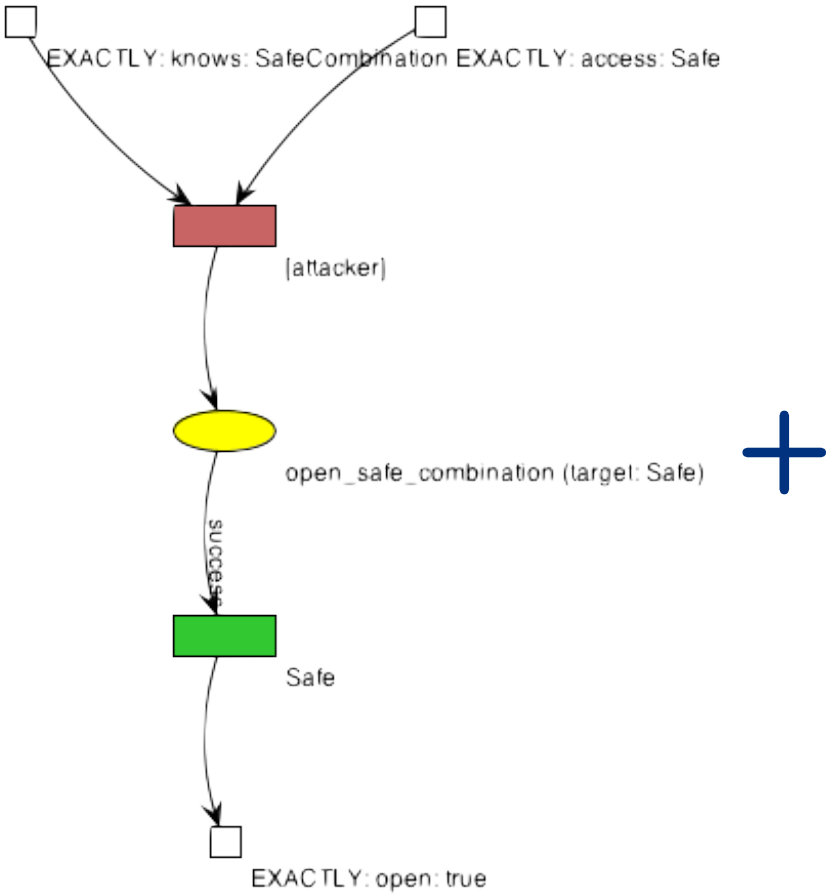
Attack patterns



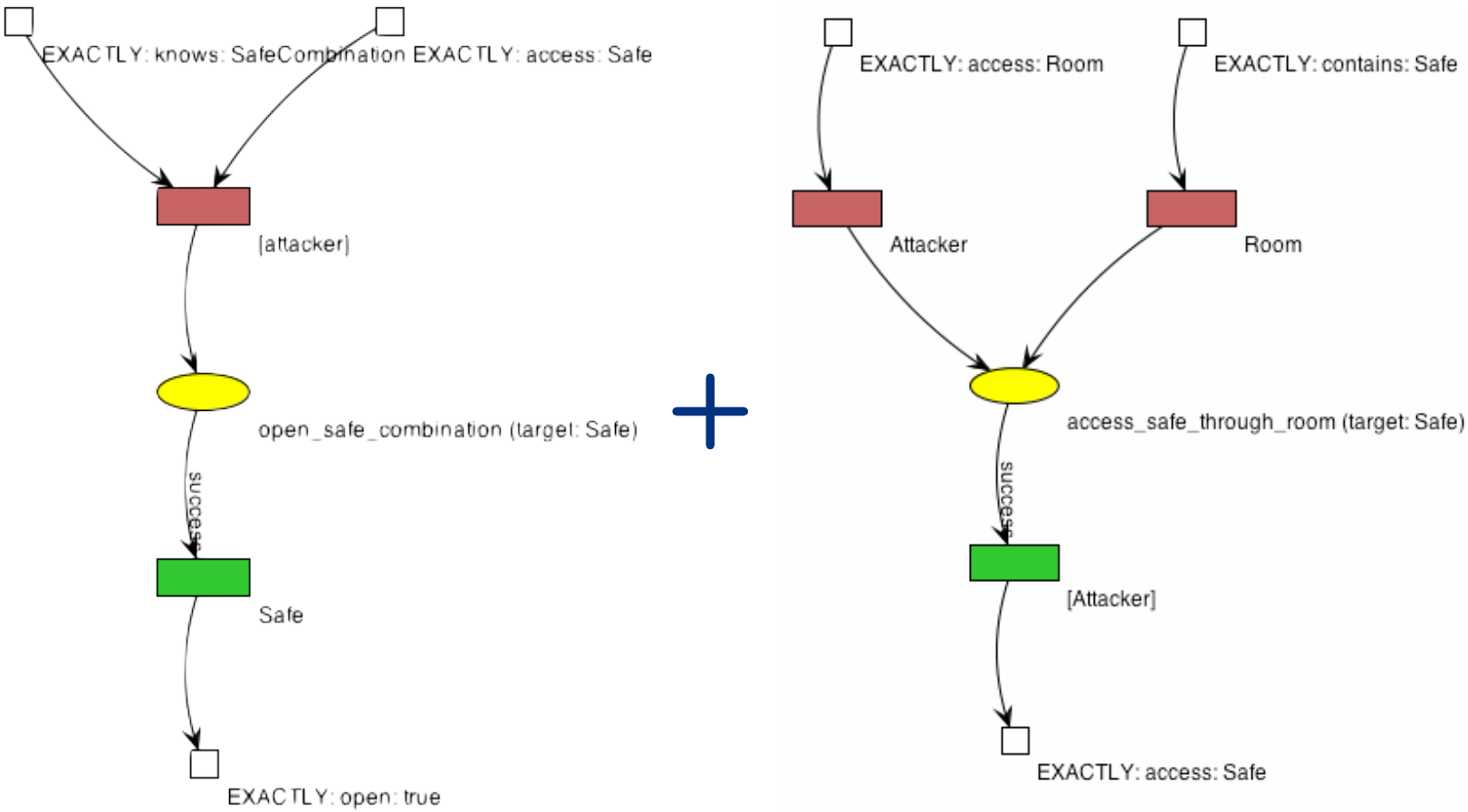
Attack pattern linking



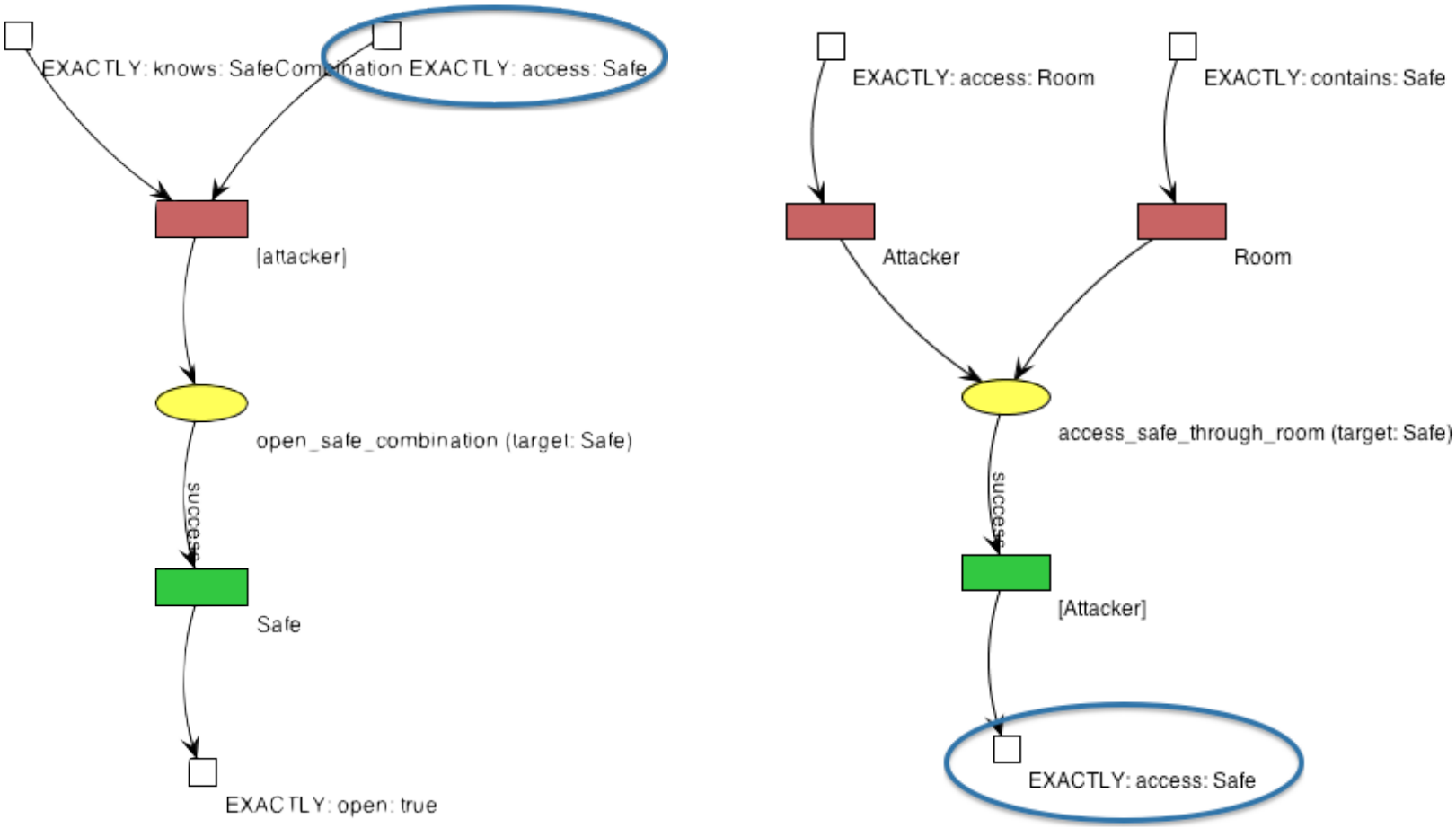
Attack pattern linking



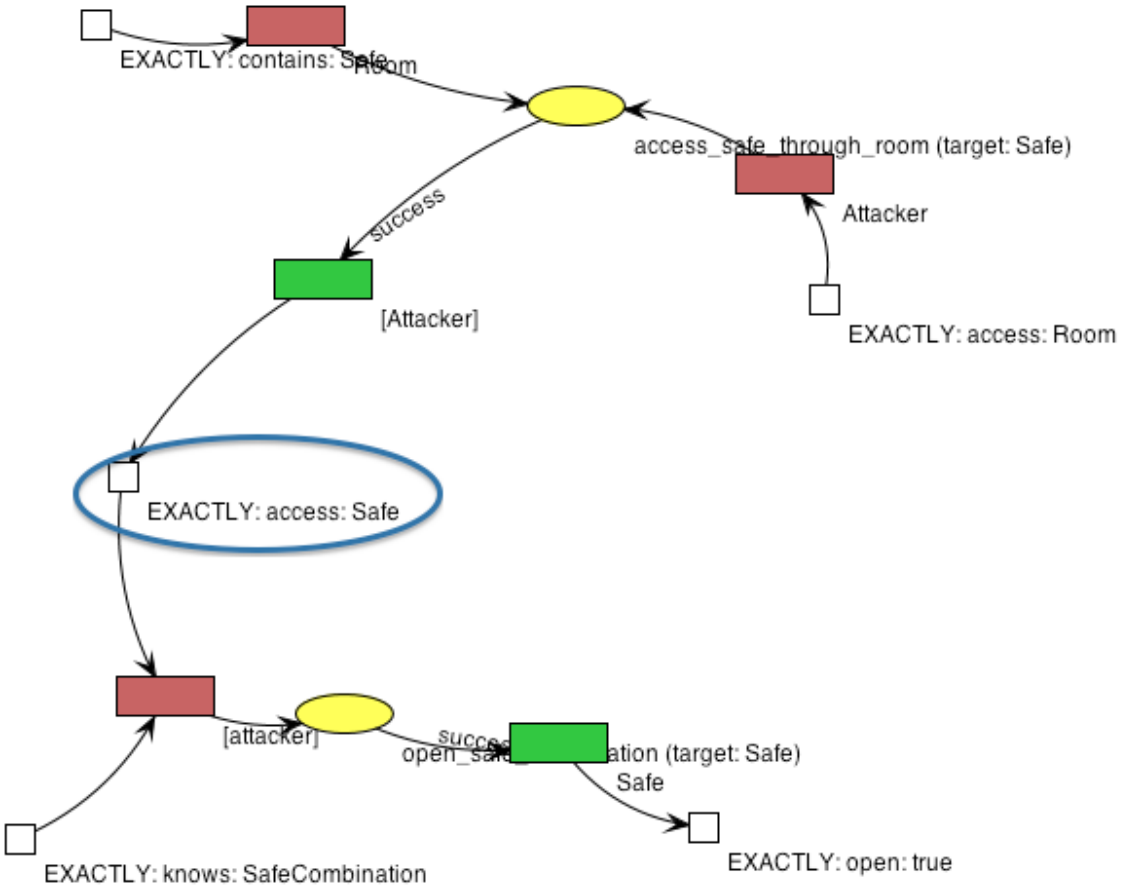
Attack pattern linking



Attack pattern linking



Attack pattern linking



CAPEC [?]

- Publicly available list of common attack patterns
- 413 patterns described in varying levels of detail
- Not fully formalized (textual descriptions)

Transformation:

1. Generic CAPEC pattern $\rightarrow$ more specific actions

   e.g., “134 Email Injection” $\rightarrow$ `emailKeylogger`, `emailBackdoor`
2. Single CAPEC pattern $\rightarrow$ sequential atomic actions

   e.g., “49 Brute Forcing" $\rightarrow$ `bruteForce`, `accessHost`, `accessData`
3. Add additional actions

   e.g., accessData, accessHost
4. Formalize
   - preconditions
   - postconditions
   - impact

CAPEC example: Brute Force (1)

Brute Force

Attack Pattern ID: 112 *(**Standard** Attack Pattern Completeness: **Complete**)*

Typical Severity: High

Status: Draft

Description

Summary

In this attack, some asset (information, functionality, identity, etc.) is protected by a finite secret value. The attacker attempts to gain access to this asset by using trial-and-error to exhaustively explore all the possible secret values in the hope of finding the secret (or a value that is functionally equivalent) that will unlock the asset. Examples of secrets can include, but are not limited to, passwords, encryption keys, database lookup keys, and initial values to one-way functions.

The key factor in this attack is the attacker's ability to explore the possible secret space rapidly. This, in turn, is a function of the size of the secret space and the computational power the attacker is able to bring to bear on the problem. If the attacker has modest resources and the secret space is large, the challenge facing the attacker is intractable. While the defender cannot control the resources available to an attacker, they can control the size of the secret space. Creating a large secret space involves selecting one's secret from as large a field of equally likely alternative secrets as possible and ensuring that an attacker is unable to reduce the size of this field using available clues or cryptoanalysis. Doing this is more difficult than it sounds since elimination of patterns (which, in turn, would provide an attacker clues that would help them reduce the space of potential secrets) is difficult to do using deterministic machines, such as computers. Assuming a finite secret space, a brute force attack will eventually succeed. The defender must rely on making sure that the time and resources necessary to do so will exceed the value of the information. For example, a secret space that will likely take hundreds of years to explore is likely safe from raw-brute force attacks.

CAPEC example: Brute Force (2)

Attack Execution Flow

Explore

1. **Determine secret testing procedure**:

   Determine how a potential guess of the secret may be tested. This may be accomplished by comparing some manipulation of the secret to a known value, use of the secret to manipulate some known set of data and determining if the result displays specific characteristics (for example, turning cryptotext into plaintext), or by submitting the secret to some external authority and having the external authority respond as to whether the value was the correct secret. Ideally, the attacker will want to determine the correctness of their guess independently since involvement of an external authority is usually slower and can provide an indication to the defender that a brute-force attack is being attempted.

   Attack Step Techniques

ID	Attack Step Technique Description	Environments
1	Determine if there is a way to parallelize the attack. Most brute force attacks can take advantage of parallel techniques by dividing the search space among available resources, thus dividing the average time to success by the number of resources available. If there is a single choke point, such as a need to check answers with an external authority, the attacker's position is significantly degraded.	env-All

2. **Reduce search space**:

   Find ways to reduce the secret space. The smaller the attacker can make the space they need to search for the secret value, the greater their chances for success. There are a great many ways in which the search space may be reduced.

   Attack Step Techniques

ID	Attack Step Technique Description	Environments
1	If possible, determine how the secret was selected. If the secret was determined algorithmically (such as by a random number generator) the algorithm may have patterns or dependencies that reduce the size of the secret space. If the secret was created by a human, behavioral factors may, if not completely reduce the space, make some types of secrets more likely than others. (For example, humans may use the same secrets in multiple places or use secrets that look or sound familiar for ease of recall.)	env-All
2	If the secret was chosen algorithmically, cryptoanalysis can be applied to the algorithm to discover patterns in this algorithm. (This is true even if the secret is not used in cryptography.) Periodicity, the need for seed values, or weaknesses in the generator all can result in a significantly smaller secret space.	env-All
3	If the secret was chosen by a person, social engineering and simple espionage can indicate patterns in their secret selection. If old secrets can be learned (and a target may feel they have little need to protect a secret that has been replaced) hints as to their selection preferences can be gleaned. These can include character substitutions a target employs, patterns in sources (dates, famous phrases, music lyrics, family members, etc.). Once these patterns have been determined, the initial efforts of a brute-force attack can focus on these areas.	env-All
4	Some algorithmic techniques for secret selection may leave indicators that can be tested for relatively easily and which could then be used to eliminate large areas of the search space for consideration. For example, it may be possible to determine that a secret does or does not start with a given character after a relatively small number of tests. Alternatively, it might be possible to discover the length of the secret relatively easily. These discoveries would significantly reduce the search space, thus increasing speed with which the attacker discovers the secret.	env-All

3. **Expand victory conditions**:

   It is sometimes possible to expand victory conditions. For example, the attacker might not need to know the exact secret but simply needs a value that produces the same result using a one-way function. While doing this does not reduce the size of the search space, the presence of multiple victory conditions does reduce the likely amount of time that the attacker will need to explore the space before finding a workable value.

Exploit

1. **Gather information so attack can be performed independently.**:

   If possible, gather the necessary information so a successful search can be determined without consultation of an external authority. This can be accomplished by capturing cryptotext (if the goal is decoding the text) or the encrypted password dictionary (if the goal is learning passwords).

CAPEC example: Brute Force (3)

Attack Prerequisites

The attacker must be able to determine when they have successfully guessed the secret. As such, one-time pads are immune to this type of attack since there is no way to determine when a guess is correct.

Methods of Attack

- Brute Force

Attacker Skills or Knowledge Required

Skill or Knowledge Level: Low

The attack simply requires basic scripting ability to automate the exploration of the search space. More sophisticated attackers may be able to use more advanced methods to reduce the search space and increase the speed with which the secret is located.

Resources Required

Ultimately, the speed with which an attacker discovers a secret is directly proportional to the computational resources the attacker has at their disposal. This attack method is resource expensive: having large amounts of computational power do not guarantee timely success, but having only minimal resources makes the problem intractable against all but the weakest secret selection procedures.

Indicators-Warnings of Attack

Description

Repeated submissions of incorrect secret values may indicate a brute force attack. For example, repeated bad passwords when accessing user accounts or repeated queries to databases using non-existent keys.

Description

Attempts to download files protected by secrets (usually using encryption) may be a precursor to an offline attack to break the file's encryption and read its contents. This is especially significant if the file itself contains other secret values, such as password files.

Description

If the attacker is able to perform the checking offline then there will likely be no indication that an attack is ongoing.

Obfuscation Techniques

Description

The attack is impossible to detect if the attacker can test for successful discovery of the secret value independently, without needing to consult an external authority.

Description

If an external authority must be consulted, the attacker can attempt to space out their guesses to avoid a large number of failed guesses in a short period of time, but doing so slows the attack to the point of making it unworkable against all but the most trivial secret spaces. As such, if an external authority must be consulted the attacked is unlikely to be able to keep the attack secret.

CAPEC example: Brute Force (4)

Solutions and Mitigations

Select a provably large secret space for selection of the secret. Provably large means that the procedure by which the secret is selected does not have artifacts that significantly reduce the size of the total secret space.

Do not provide the means for an attacker to determine success independently. This forces the attacker to check their guesses against an external authority, which can slow the attack and warn the defender. This mitigation may not be possible if testing material must appear externally, such as with a transmitted cryptotext.

Attack Motivation-Consequences

Scope	Technical Impact	Note
Confidentiality	Read application data	
Confidentiality Access_Control Authorization	Gain privileges / assume identity	

Related Weaknesses

CWE-ID	Weakness Name	Weakness Relationship Type
330	Use of Insufficiently Random Values	Secondary
326	Inadequate Encryption Strength	Secondary
521	Weak Password Requirements	Secondary

Related Attack Patterns

Nature	Type	ID	Name	Description	V
ChildOf	C	223	Probabilistic Techniques		**1000**
HasMember	C	344	WASC Threat Classification 2.0 - WASC-11 - Brute Force		333
ParentOf	A	20	Encryption Brute Forcing		**1000**
ParentOf	A	49	Password Brute Forcing		**1000**

Relevant Security Requirements

Protect sensitive data, even when the data is encrypted. If an attacker can gain access to encrypted data, they can mount a brute-force attack independently. The defender will not be aware of this attack or be able to do anything about it and at that point it is purely a function of the attacker's available resources as to how long it takes them to learn the secret.

Monitor activity logs for suspicious activity. An attacker that must use an external authority to check their brute-force guesses is easy to detect, but only if that external authority is monitoring activity and detects the abnormally large number of failed guesses.

Related Guidelines

- Do not assume secrets will protect sensitive data in the long-term
- Monitor systems for suspicious activity.

Purposes

- Penetration

Brute force: Prolog rule formulation

Preconditions

```
action_bruteForce(Attacker, TargetHost, TargetGroup):-
   technicalSkillLevel(Attacker, TechnicalSkillLevel),
   TechnicalSkillLevel >= 1,
   owned(Attacker, AttackHost),
   connected(AttackHost, TargetHost, rdpProtocol, rdpPort),
   accessHost(TargetGroup, TargetHost, _),
   not(inGroup(Attacker, TargetGroup)).
```

Postcondition

```
exec_success_action_bruteForce(Attacker, TargetHost, TargetGroup):-
   assert(inGroup(Attacker, TargetGroup)).
```

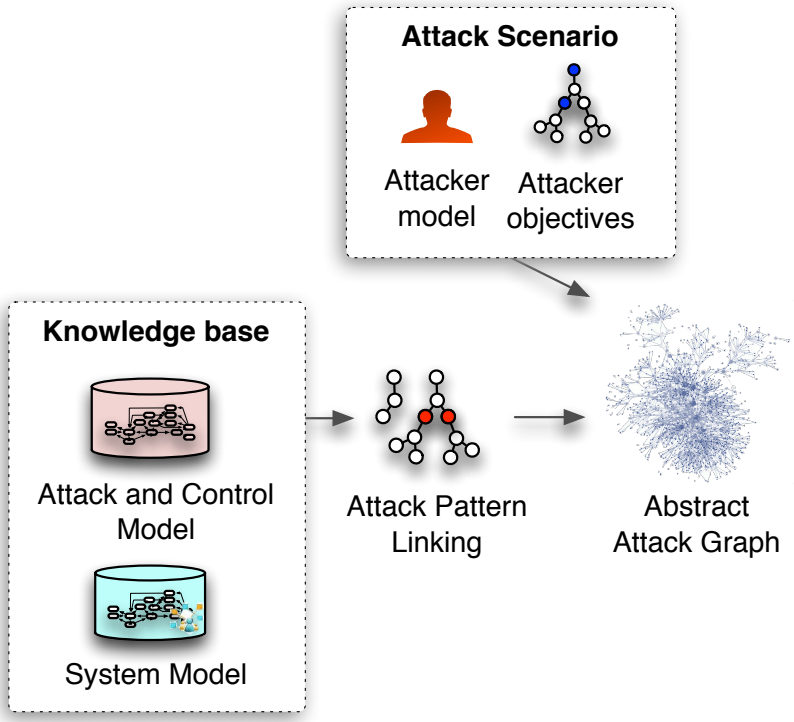
Impact

```
action_impact(action_bruteForce, confidentiality).
impact_success_bruteForce(Attacker, TargetHost, TargetGroup, SecurityAttribute, Impact):-
   importance(TargetGroup, SecurityAttribute, Impact).
```

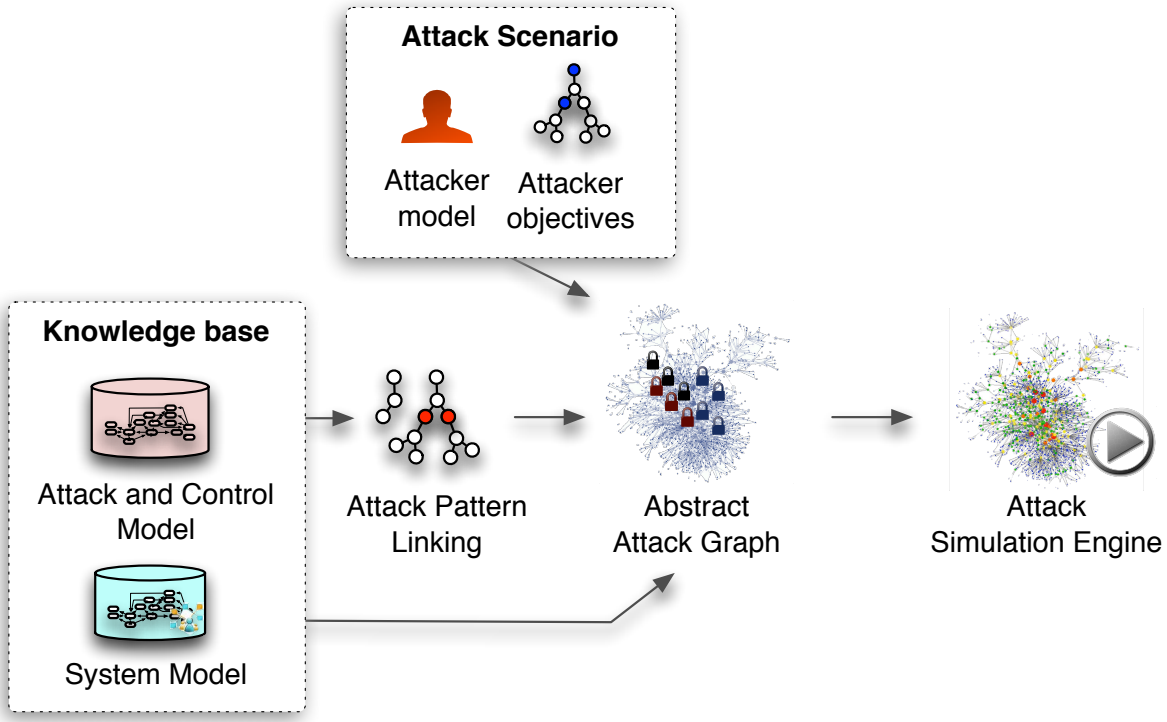
Simulation attributes

```
/** cost, time, base probability, maxTries, simultaneous **/
action_properties(action_bruteForce, 0, 18000, 0.01, 0, true).
available_action(action_bruteForce).
```

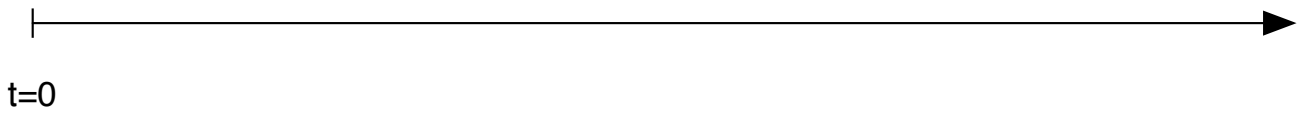
Simulation


Attack Scenario
Attacker model
Attacker objectives
Knowledge base
Attack and Control Model
System Model
Attack Pattern Linking
Abstract Attack Graph

Simulation

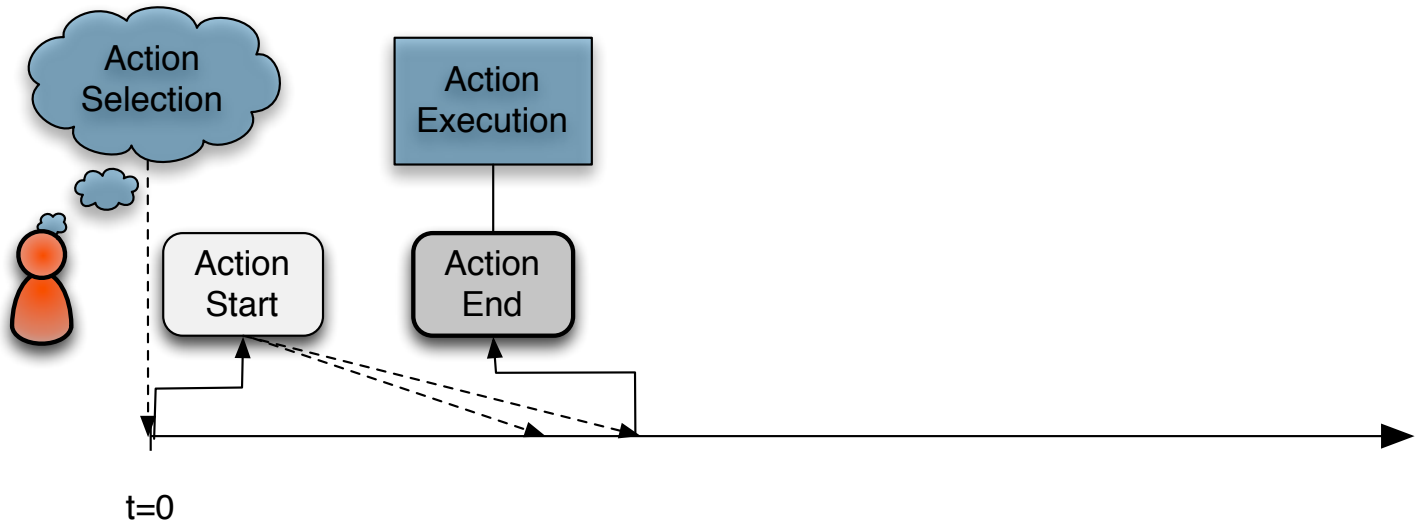

Attack Scenario
Attacker model
Attacker objectives
Knowledge base
Attack and Control Model
System Model
Attack Pattern Linking
Abstract Attack Graph
Attack Simulation Engine

Discrete Event Scheduling

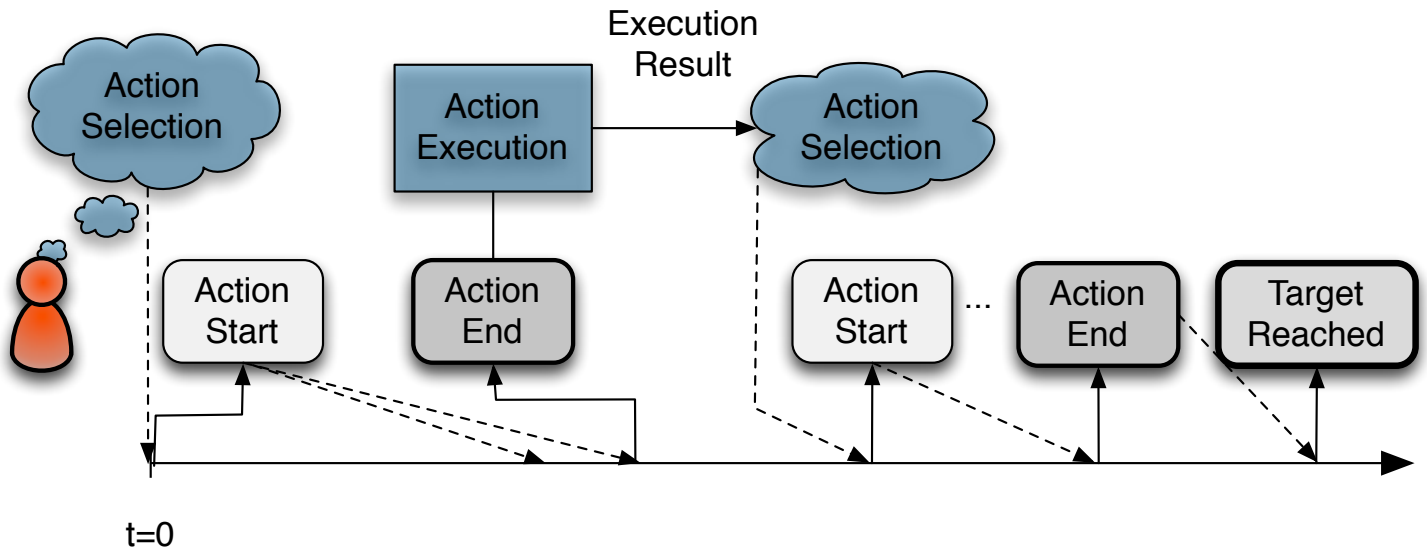


Discrete Event Scheduling

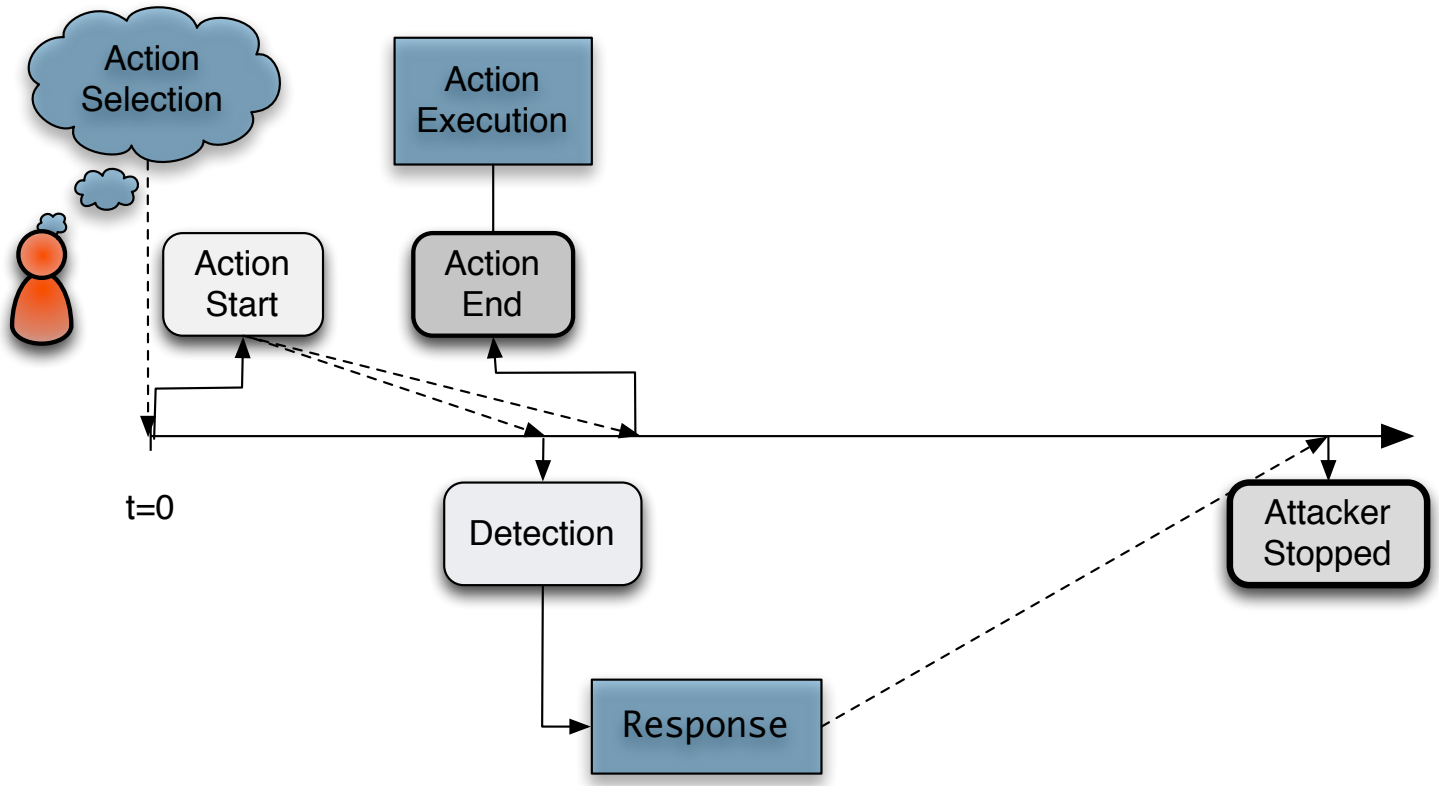
Discrete Event Scheduling



Discrete Event Scheduling

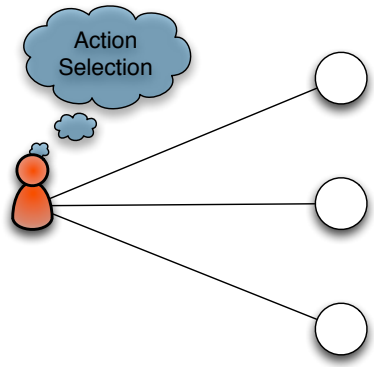


Discrete Event Scheduling



Behavioral model

Choice set:



Choice function: for all considered actions $a \in A$

1. Calculate distance in abstract graph:

$$d_a^{rel} \leftarrow \frac{d(a,t)}{max(d(a,t))+1}$$

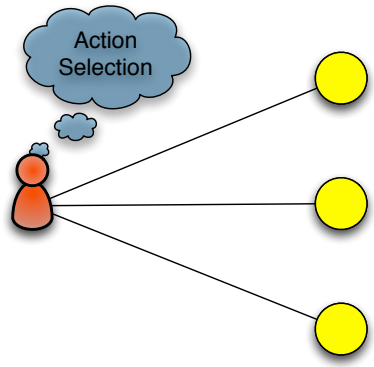
2. Calculate weight:

$$W_a \leftarrow p_{suc}(a)^{w_{suc}} \left(1 - p_{det}(a)\right)^{w_{det}} \left(1 - d_a^{rel}\right)^{w_{dist}}$$

3. **return** $weightedChoice(A, W)$

Behavioral model

Choice set:



Choice function: for all considered actions $a \in A$

1. Calculate distance in abstract graph:

$$d_a^{rel} \leftarrow \frac{d(a,t)}{max(d(a,t))+1}$$

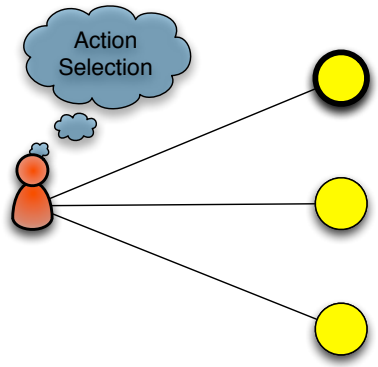
2. Calculate weight:

$$W_a \leftarrow p_{suc}(a)^{w_{suc}} \left(1 - p_{det}(a)\right)^{w_{det}} \left(1 - d_a^{rel}\right)^{w_{dist}}$$

3. **return** $weightedChoice(A, W)$

Behavioral model

Choice set:



Choice function: for all considered actions $a \in A$

1. Calculate distance in abstract graph:

$$d_a^{rel} \leftarrow \frac{d(a,t)}{max(d(a,t))+1}$$

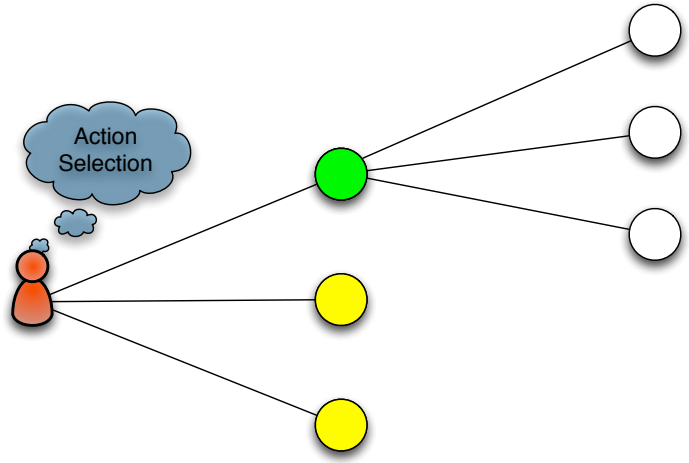
2. Calculate weight:

$$W_a \leftarrow p_{suc}(a)^{w_{suc}} \left(1 - p_{det}(a)\right)^{w_{det}} \left(1 - d_a^{rel}\right)^{w_{dist}}$$

3. **return** $weightedChoice(A, W)$

Behavioral model

Choice set:



Choice function: for all considered actions $a \in A$

1. Calculate distance in abstract graph:

$$d_a^{rel} \leftarrow \frac{d(a,t)}{max(d(a,t))+1}$$

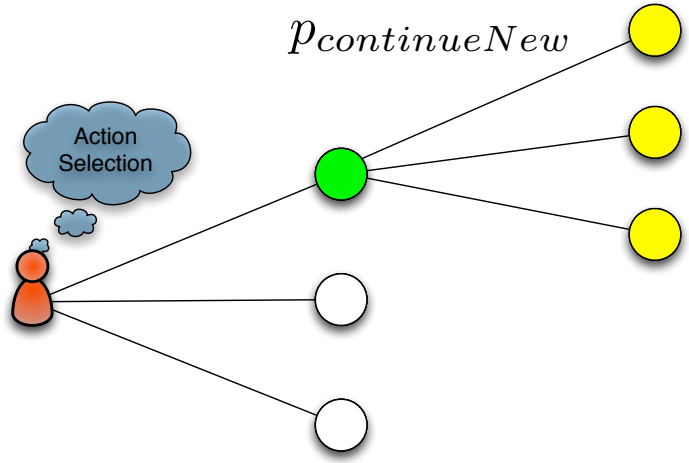
2. Calculate weight:

$$W_a \leftarrow p_{suc}(a)^{w_{suc}} \left(1 - p_{det}(a)\right)^{w_{det}} \left(1 - d_a^{rel}\right)^{w_{dist}}$$

3. **return** $weightedChoice(A, W)$

Behavioral model

Choice set:



Choice function: for all considered actions $a \in A$

1. Calculate distance in abstract graph:

$$d_a^{rel} \leftarrow \frac{d(a,t)}{max(d(a,t))+1}$$

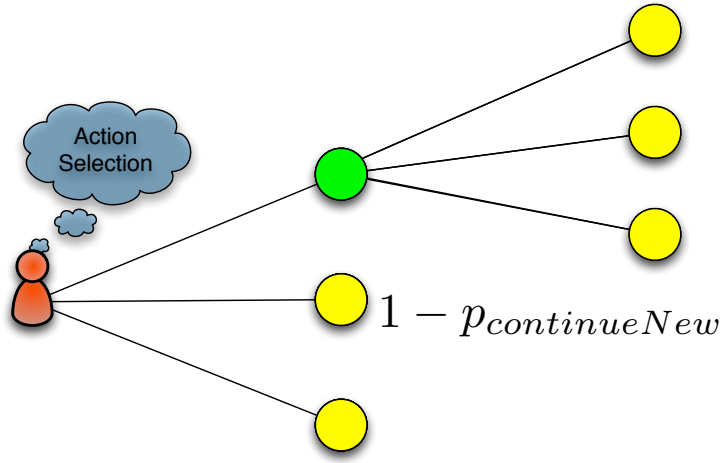
2. Calculate weight:

$$W_a \leftarrow p_{suc}(a)^{w_{suc}} \left(1 - p_{det}(a)\right)^{w_{det}} \left(1 - d_a^{rel}\right)^{w_{dist}}$$

3. **return** $weightedChoice(A, W)$

Behavioral model

Choice set:



Choice function: for all considered actions $a \in A$

1. Calculate distance in abstract graph:

$$d_a^{rel} \leftarrow \frac{d(a,t)}{max(d(a,t))+1}$$

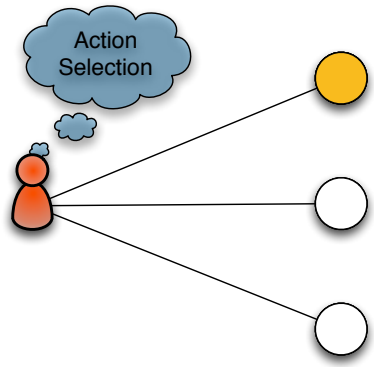
2. Calculate weight:

$$W_a \leftarrow p_{suc}(a)^{w_{suc}} \left(1 - p_{det}(a)\right)^{w_{det}} \left(1 - d_a^{rel}\right)^{w_{dist}}$$

3. **return** $weightedChoice(A, W)$

Behavioral model

Choice set:



Choice function: for all considered actions $a \in A$

1. Calculate distance in abstract graph:

$$d_a^{rel} \leftarrow \frac{d(a,t)}{max(d(a,t))+1}$$

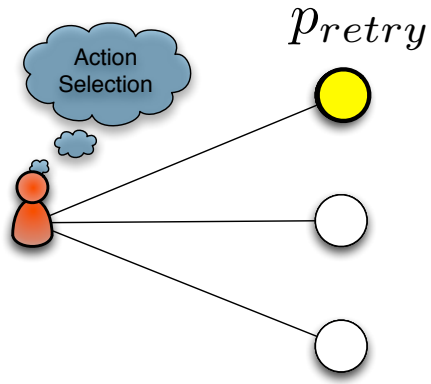
2. Calculate weight:

$$W_a \leftarrow p_{suc}(a)^{w_{suc}} \left(1 - p_{det}(a)\right)^{w_{det}} \left(1 - d_a^{rel}\right)^{w_{dist}}$$

3. **return** $weightedChoice(A, W)$

Behavioral model

Choice set:

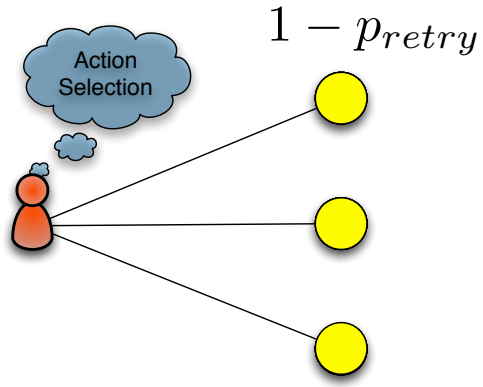


Choice function: for all considered actions $a \in A$

1. Calculate distance in abstract graph:
$$d_a^{rel} \leftarrow \frac{d(a,t)}{max(d(a,t))+1}$$
2. Calculate weight:
$$W_a \leftarrow p_{suc}(a)^{w_{suc}} \left(1 - p_{det}(a)\right)^{w_{det}} \left(1 - d_a^{rel}\right)^{w_{dist}}$$
3. **return** $weightedChoice(A, W)$

Behavioral model

Choice set:

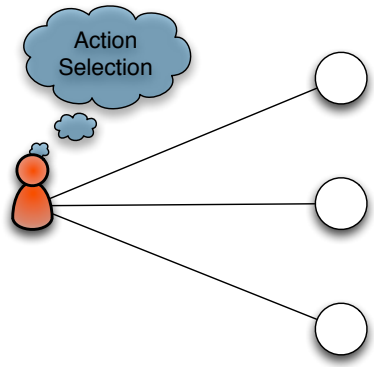


Choice function: for all considered actions $a \in A$

1. Calculate distance in abstract graph:
$$d_a^{rel} \leftarrow \frac{d(a,t)}{max(d(a,t))+1}$$
2. Calculate weight:
$$W_a \leftarrow p_{suc}(a)^{w_{suc}} \left(1 - p_{det}(a)\right)^{w_{det}} \left(1 - d_a^{rel}\right)^{w_{dist}}$$
3. **return** $weightedChoice(A, W)$

Behavioral model

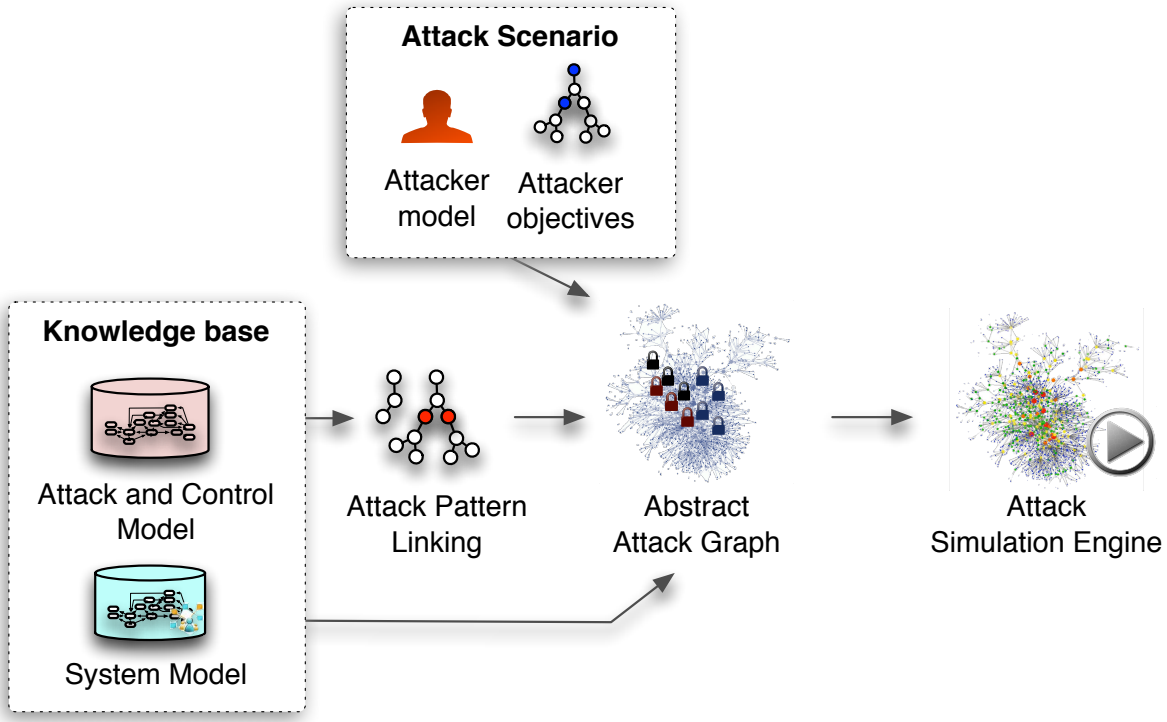
Choice set:



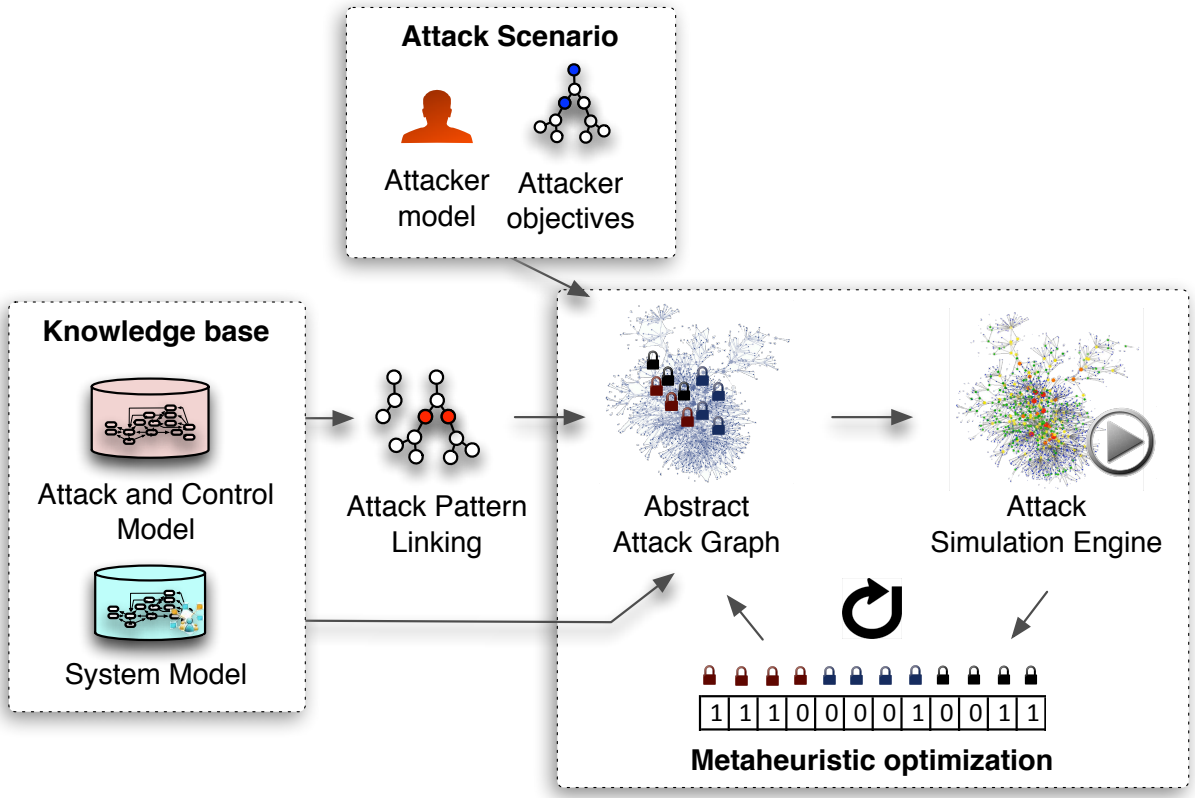
Choice function: for all considered actions $a \in A$

1. Calculate distance in abstract graph:
   $d_a^{rel} \leftarrow \frac{d(a,t)}{max(d(a,t))+1}$
2. Calculate weight:
   $$W_a \leftarrow p_{suc}(a)^{w_{suc}} \left(1 - p_{det}(a)\right)^{w_{det}} \left(1 - d_a^{rel}\right)^{w_{dist}}$$
3. **return** $weightedChoice(A, W)$

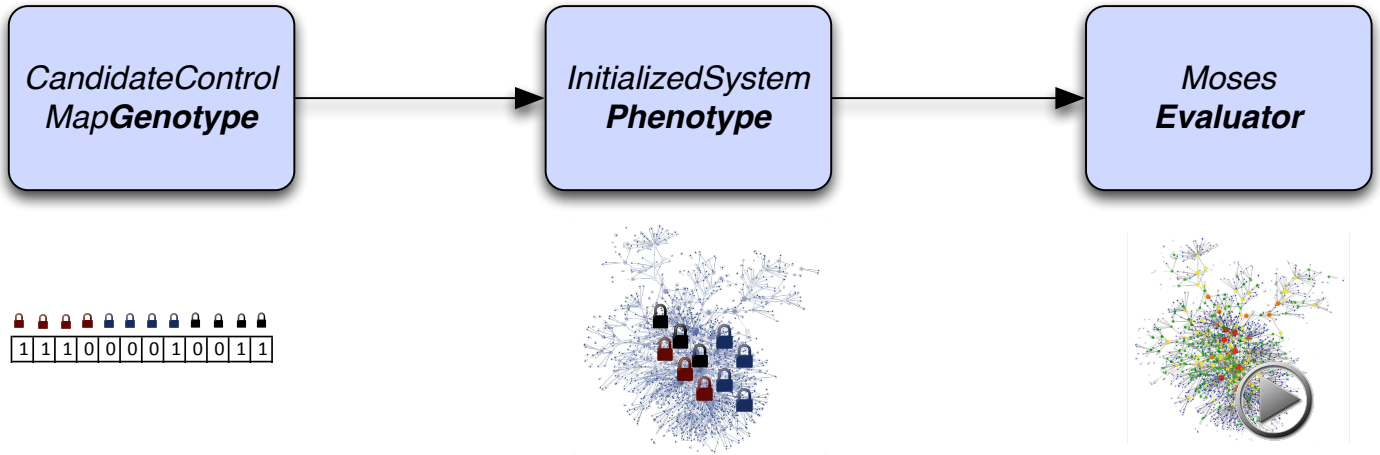
Optimization



Optimization

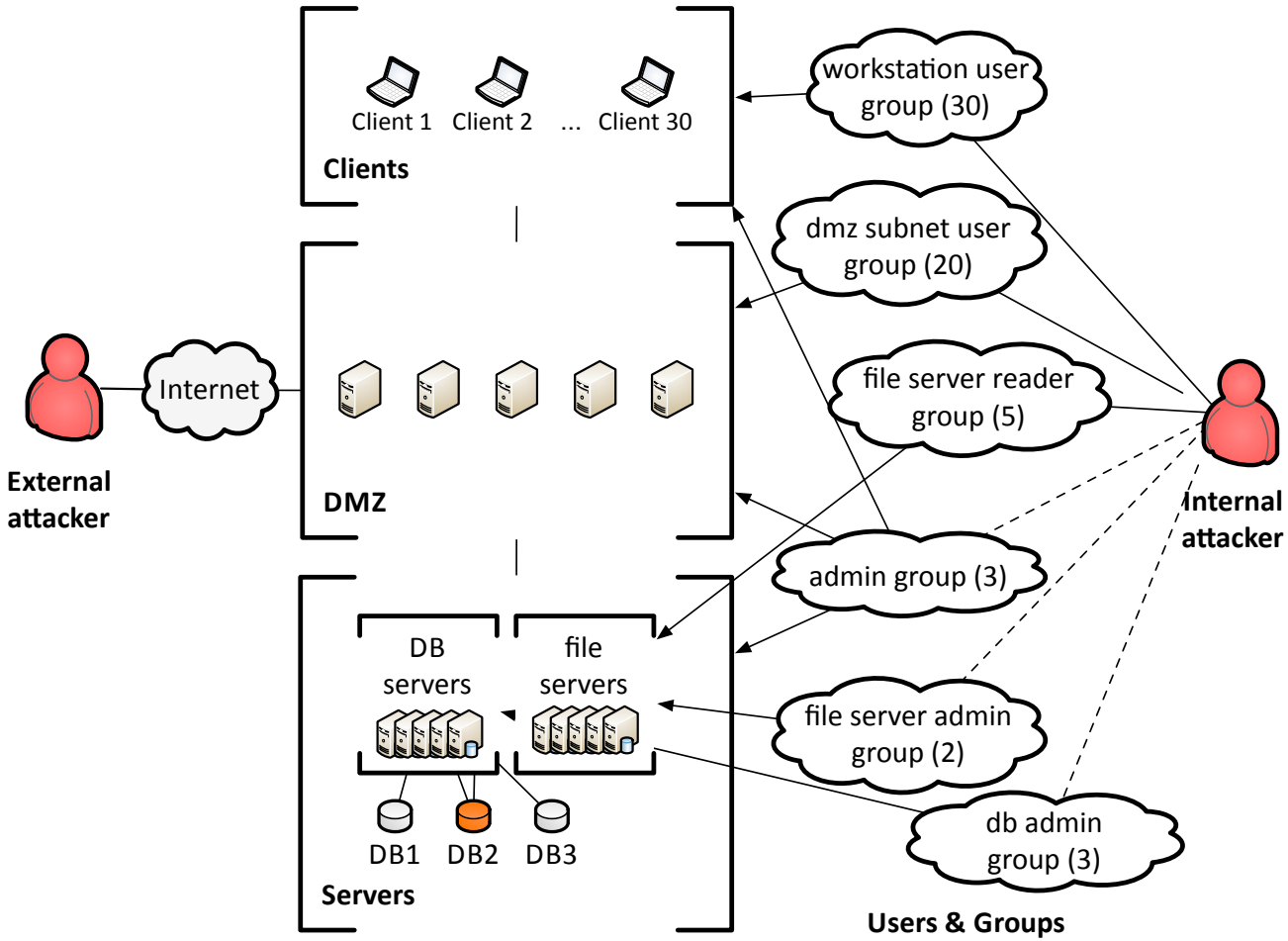


Evaluation of control portfolios

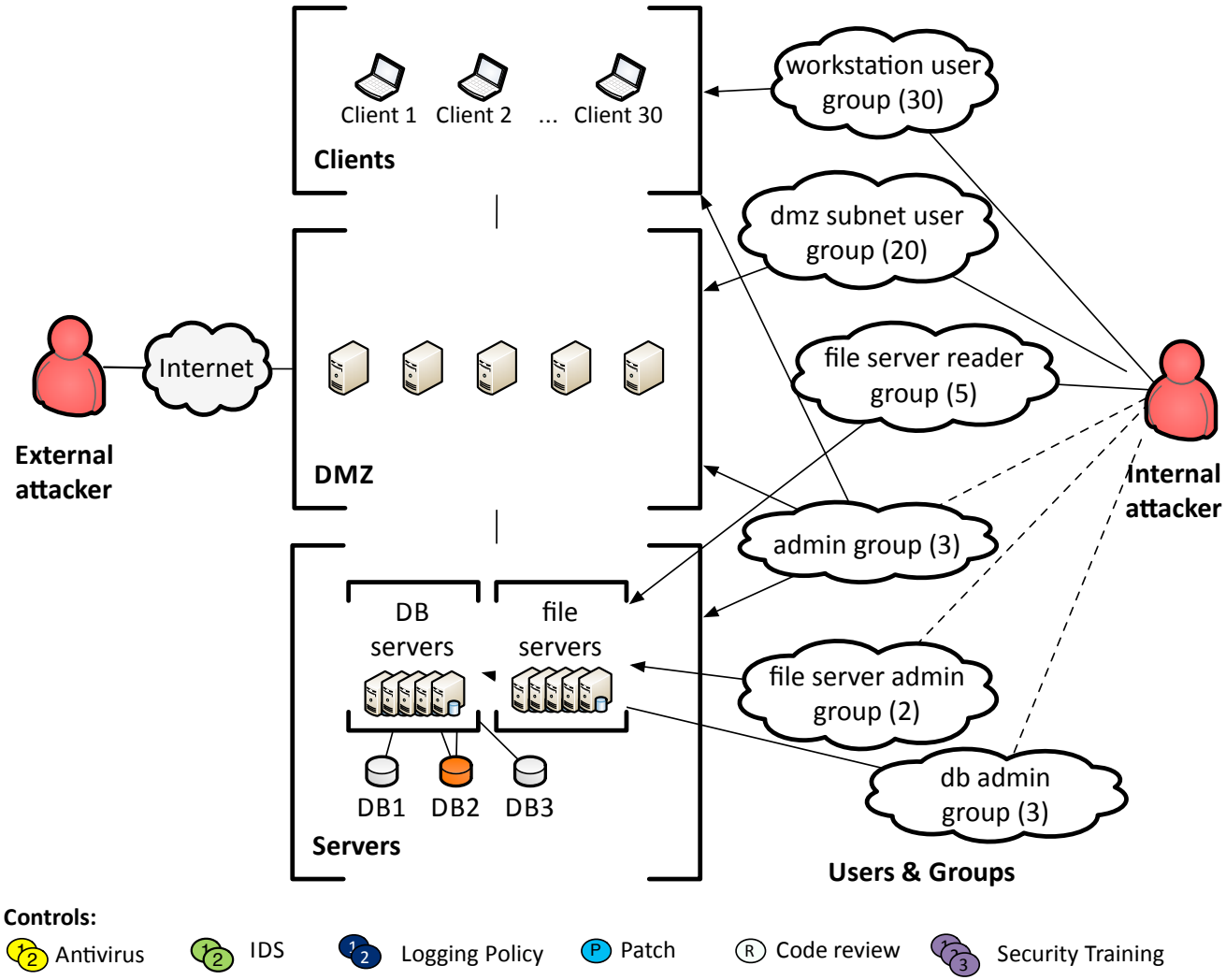


- Genetic algorithm adapts the system
- Probabilistic $\rightarrow$ multiple replications per control set
- Reduced to a deterministic problem using expected/median/worst case values etc.

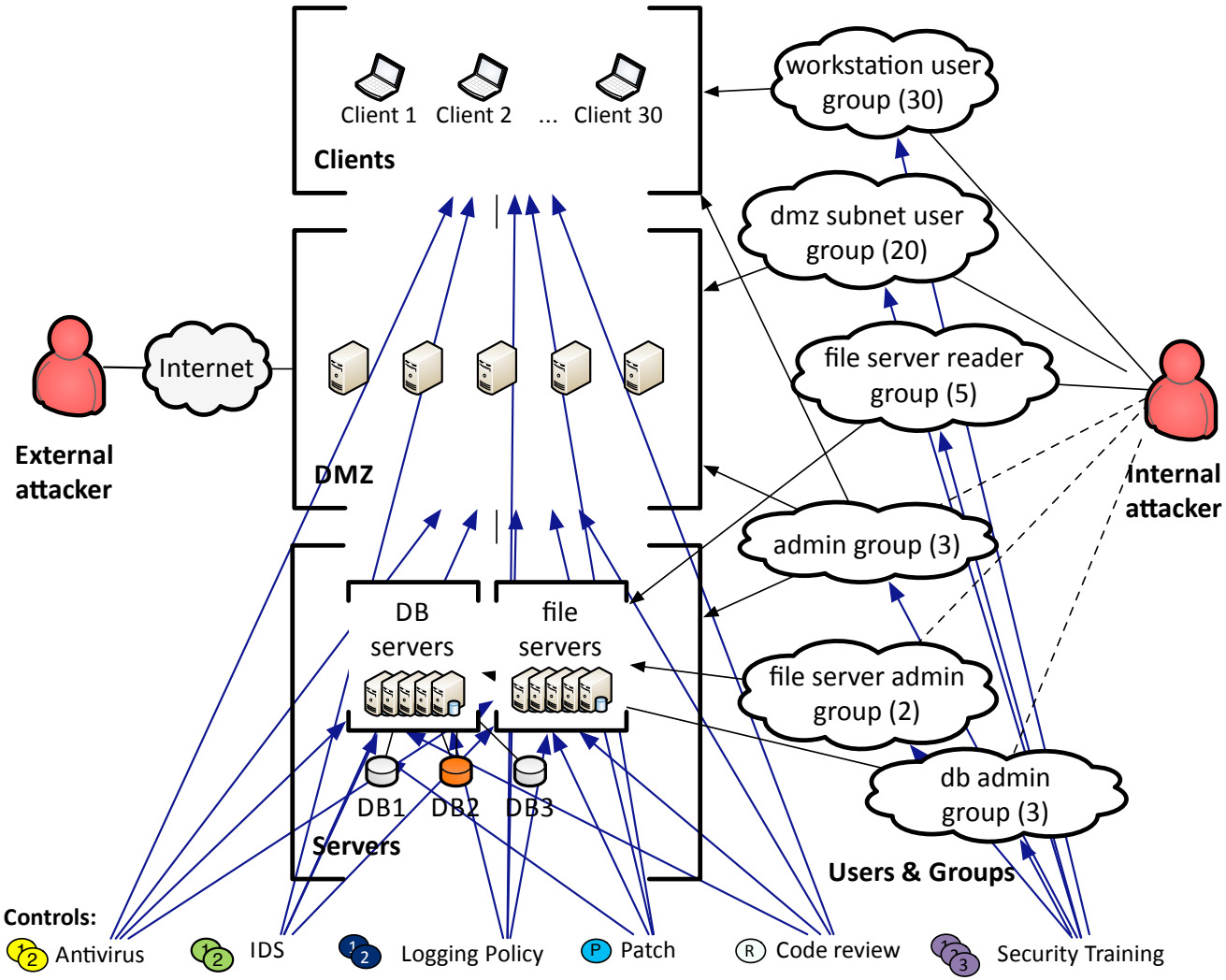
Scenario domain



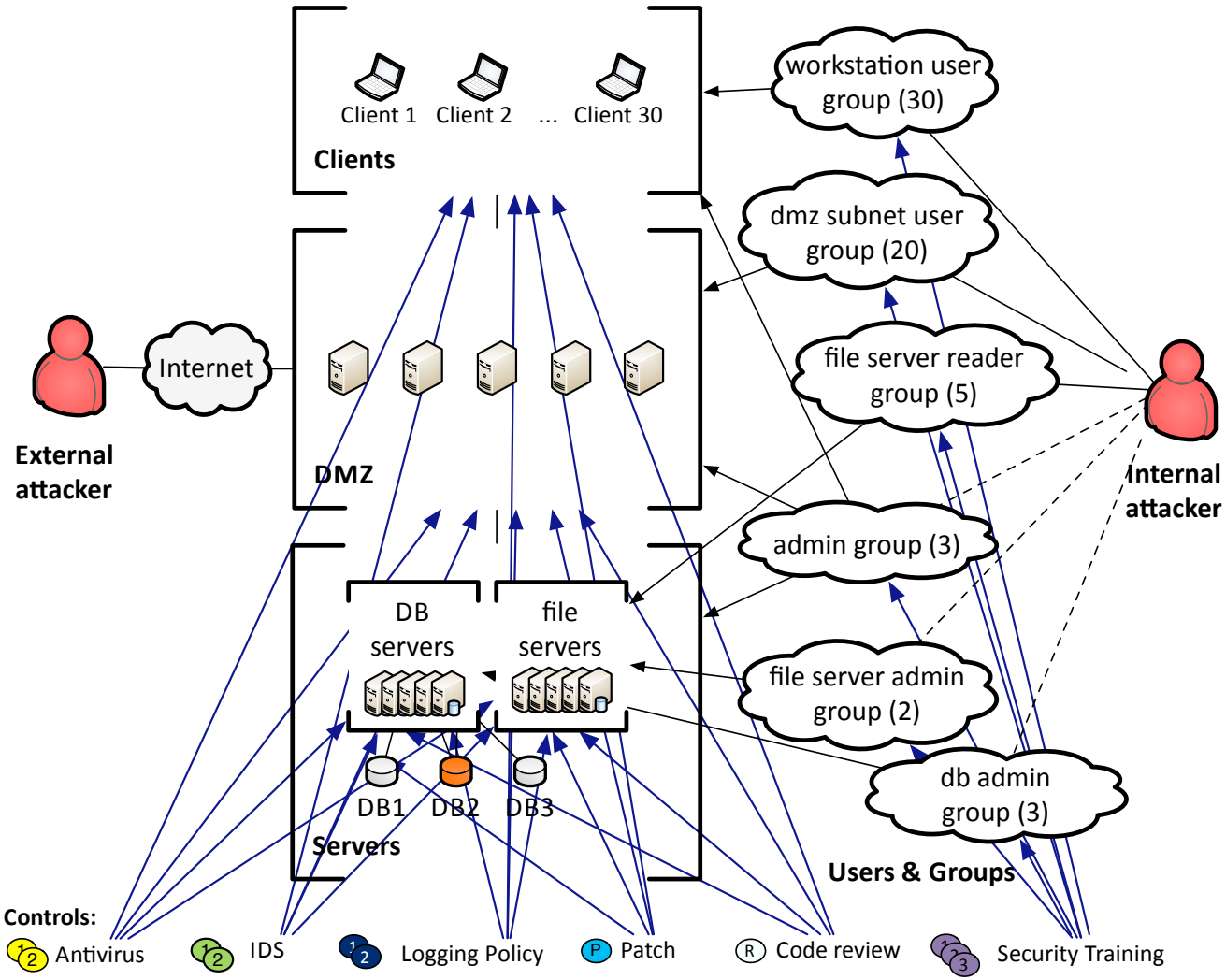
Scenario domain



Scenario domain



Scenario domain



58 binary decision variables (control-asset assignments)

Adversary types

Characteristics

	time (mins)	w_{det}	w_{suc}	w_{dist}	access
Employee	2500	0.45	0.25	0.30	workstations
Administrator	5000	0.50	0.20	0.30	all hosts
Skilled External	3333	0.30	0.40	0.30	-
Unskilled External	1667	0.30	0.40	0.30	-
APT	∞	0.50	0.20	0.30	-

Available actions (based on skill level, access)

Employee (skill: 0)	`shoulderSurfing`
Unskilled external (skill: 1)	`spearfish`
	`sqlInjection`
	`socialAttack`
	`bruteForce`
	`emailKeylogger`
	`emailBackdoor`
Skilled external (skill: 2)	`+ bufferOverflow`
	`+ directoryTraversal`
Admin (skill: 2)	(all above)
Advanced persistent threat (skill: 3)	`+ zeroDay`

Optimization objectives

1. Minimize cost of controls
2. Minimize target condition achievement
3. Maximize detection of attacks
4. Minimize confidentiality impact (L/M/H)
5. Minimize integrity impact (L/M/H)
6. Minimize availability impact (L/M/H)

L/M/H: low, medium, high in lexicographic order

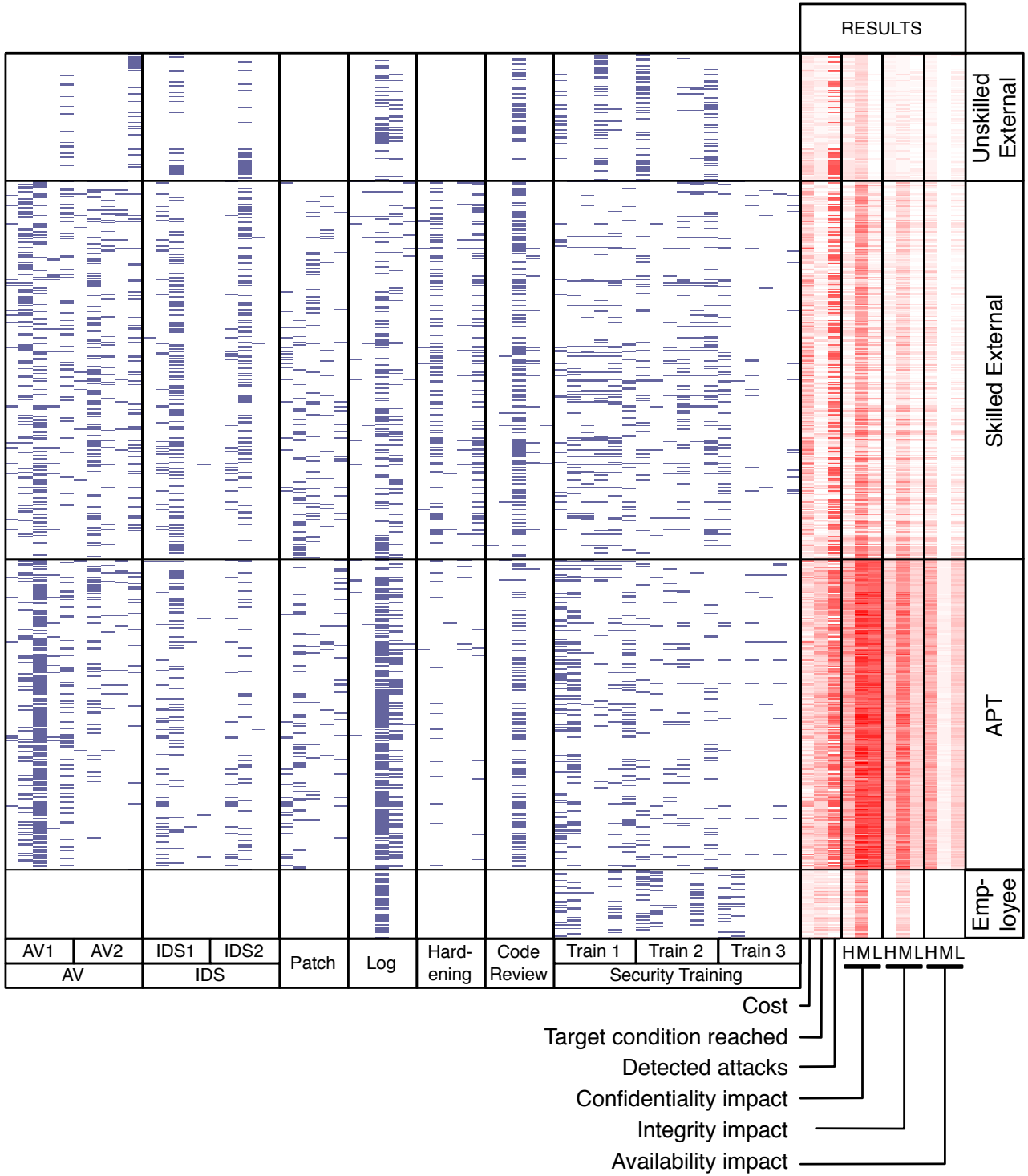
Parameter settings

Simulation: 50 replications per control set

Optimization: 500 generations

- **Population**
  - $\alpha = 100$ (population size)
  - $\mu = 25$ (number of parents per generation)
  - $\lambda = 25$ (number of offsprings per generation)
  - Initialization: $\vec{1}$, $\vec{0}$, remaining random (i.e., each control included with $p = 0.5$)
- **Selection:** NSGA2, 2 tournaments
- **Crossover:** 2-point crossover @ rate 0.95
- **Mutation:** mixed permutation (insert, revert, swap) rate $1/n$

Results: Overview



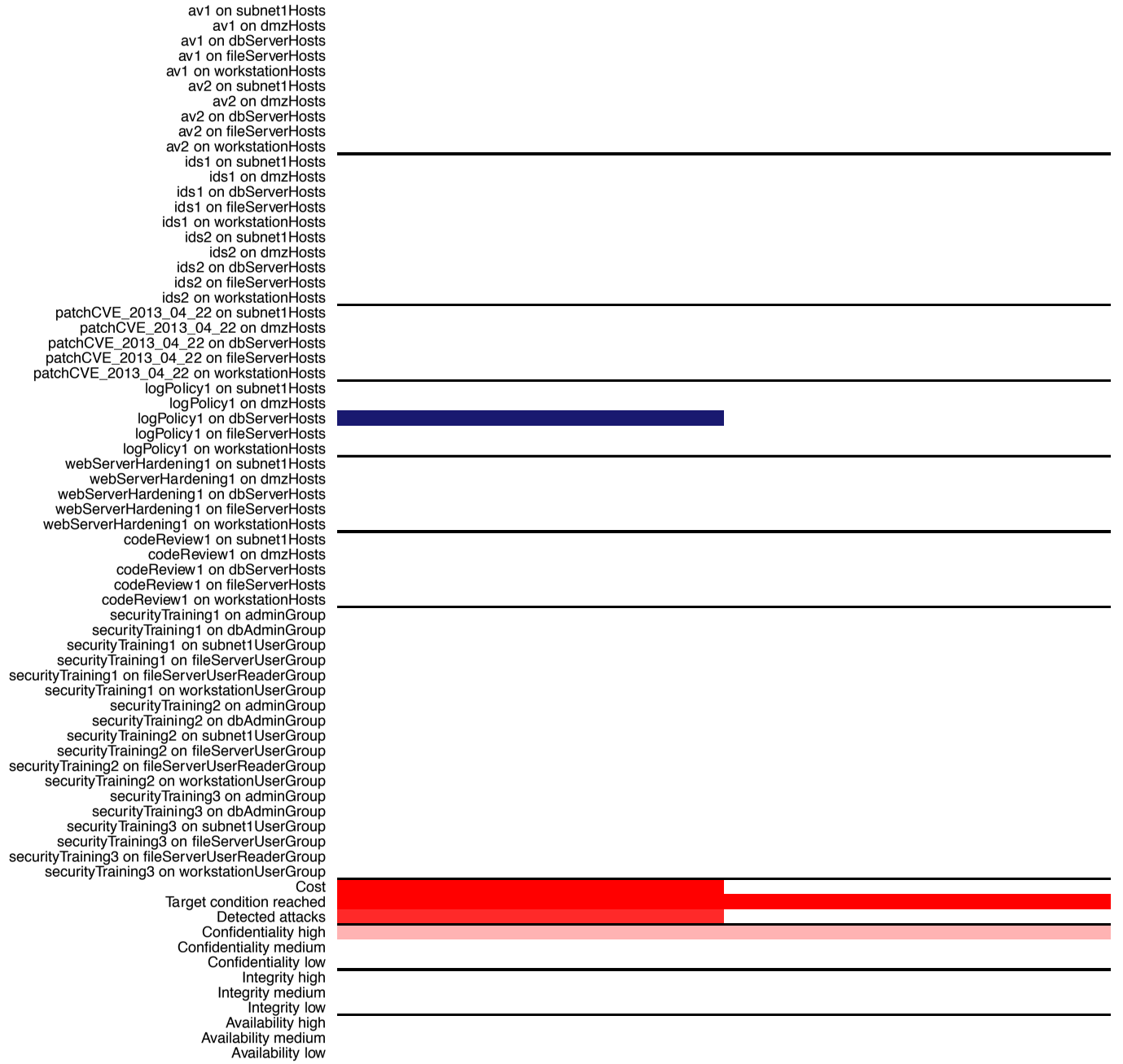
Administrator example attack trace

action_accessData

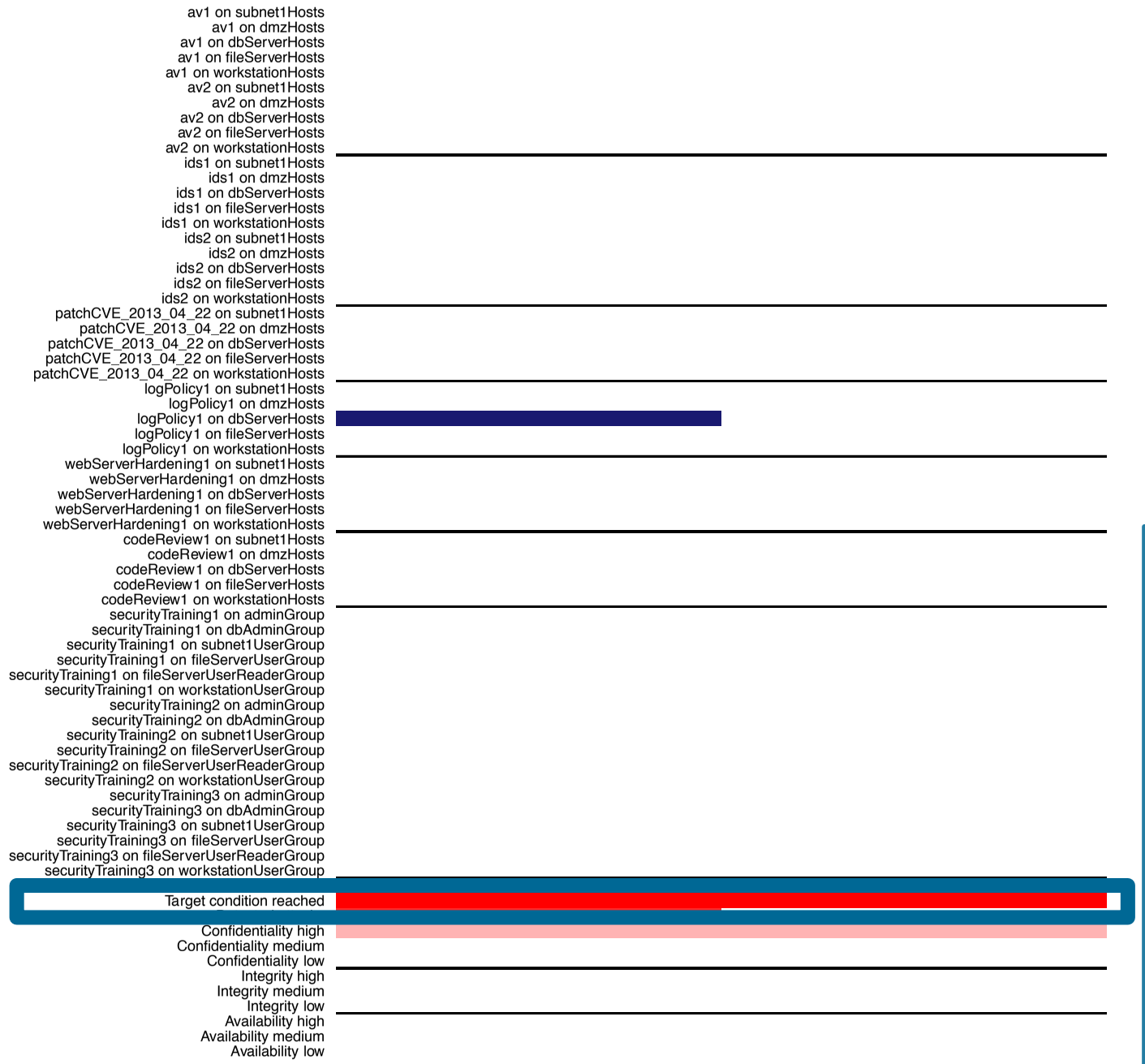
Target:
null

Asset mapping:
TargetHost: dbServerHosts
Data: db2

Results: Administrator

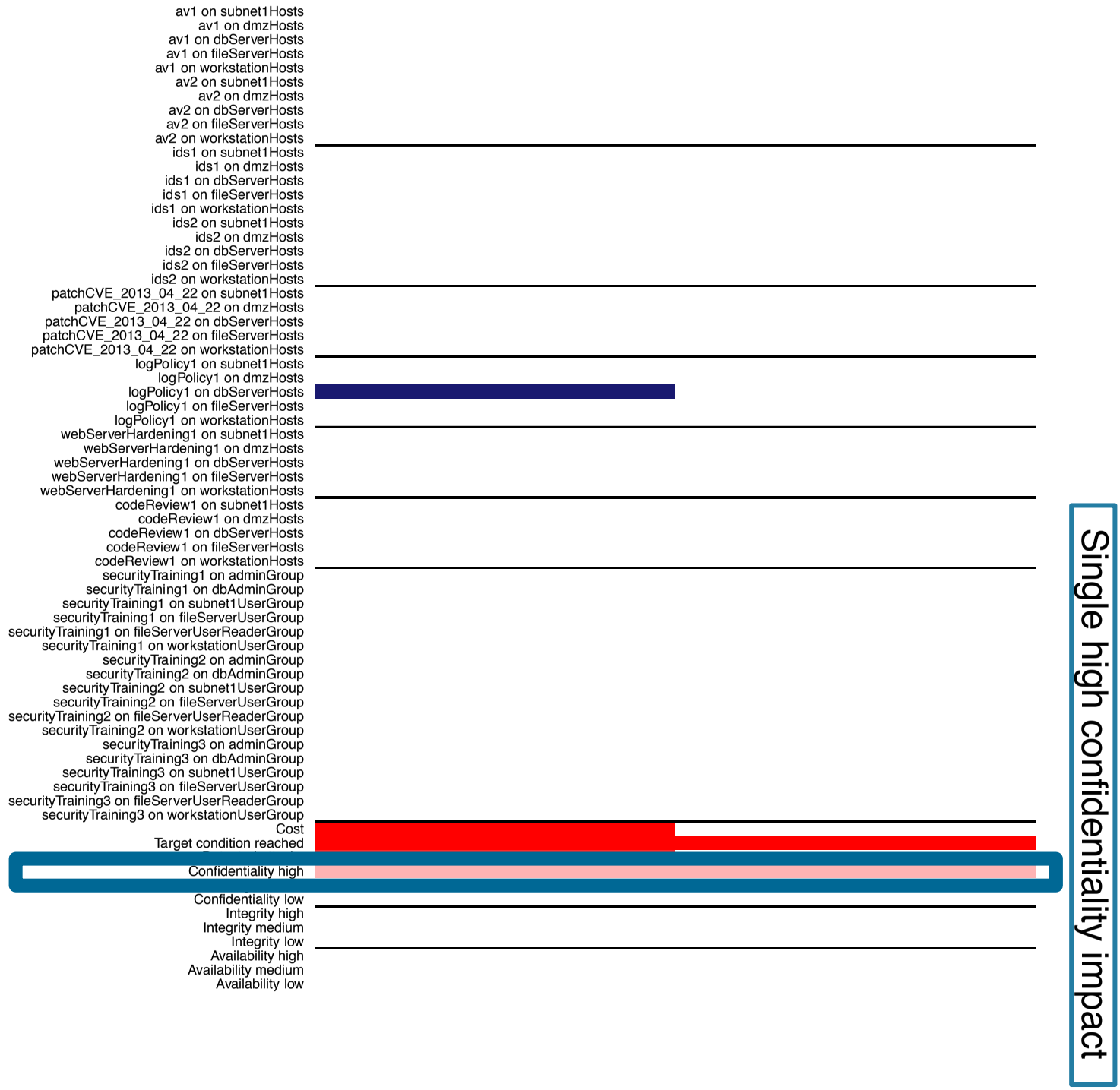


Results: Administrator

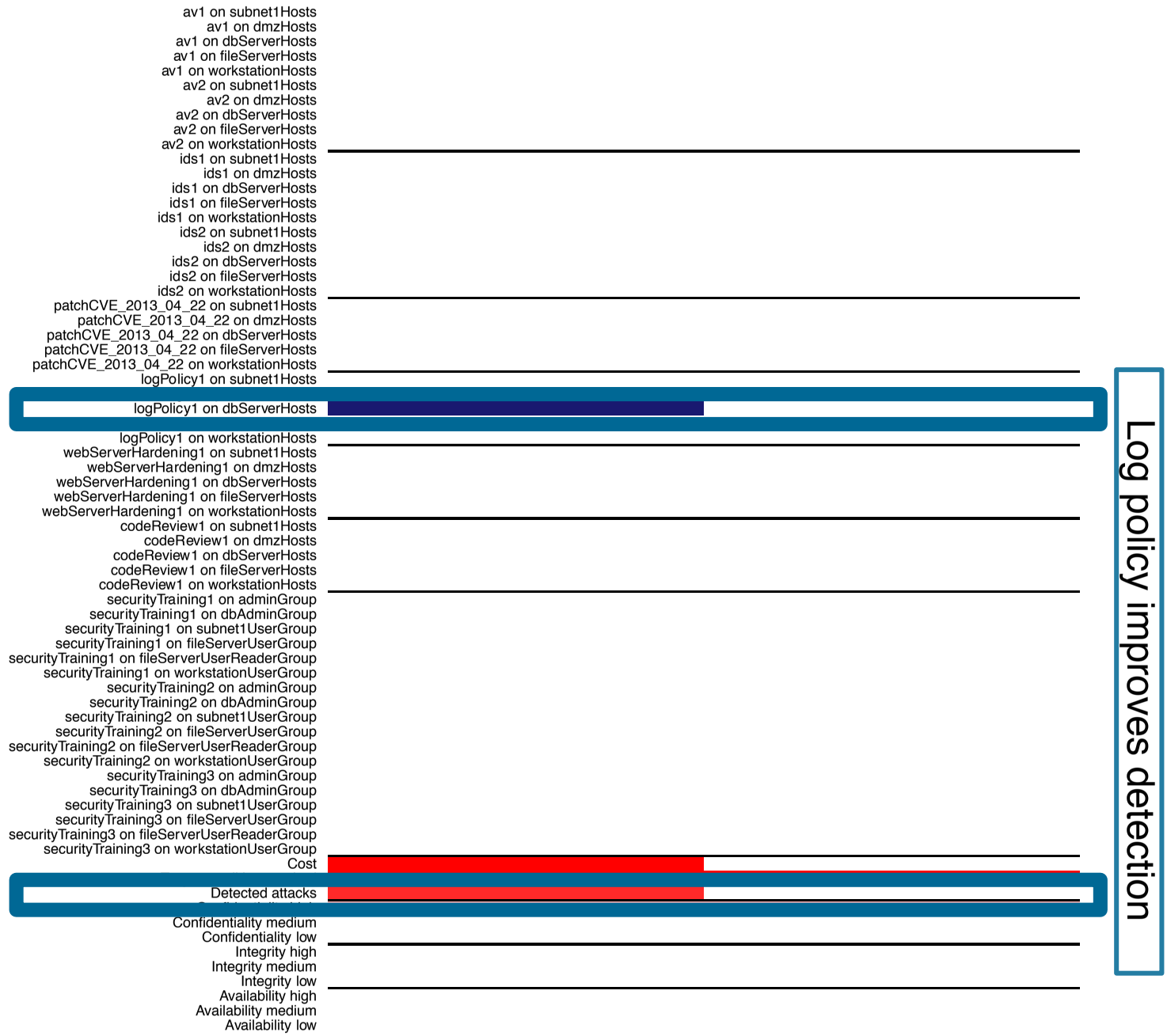


Target condition always reached

Results: Administrator



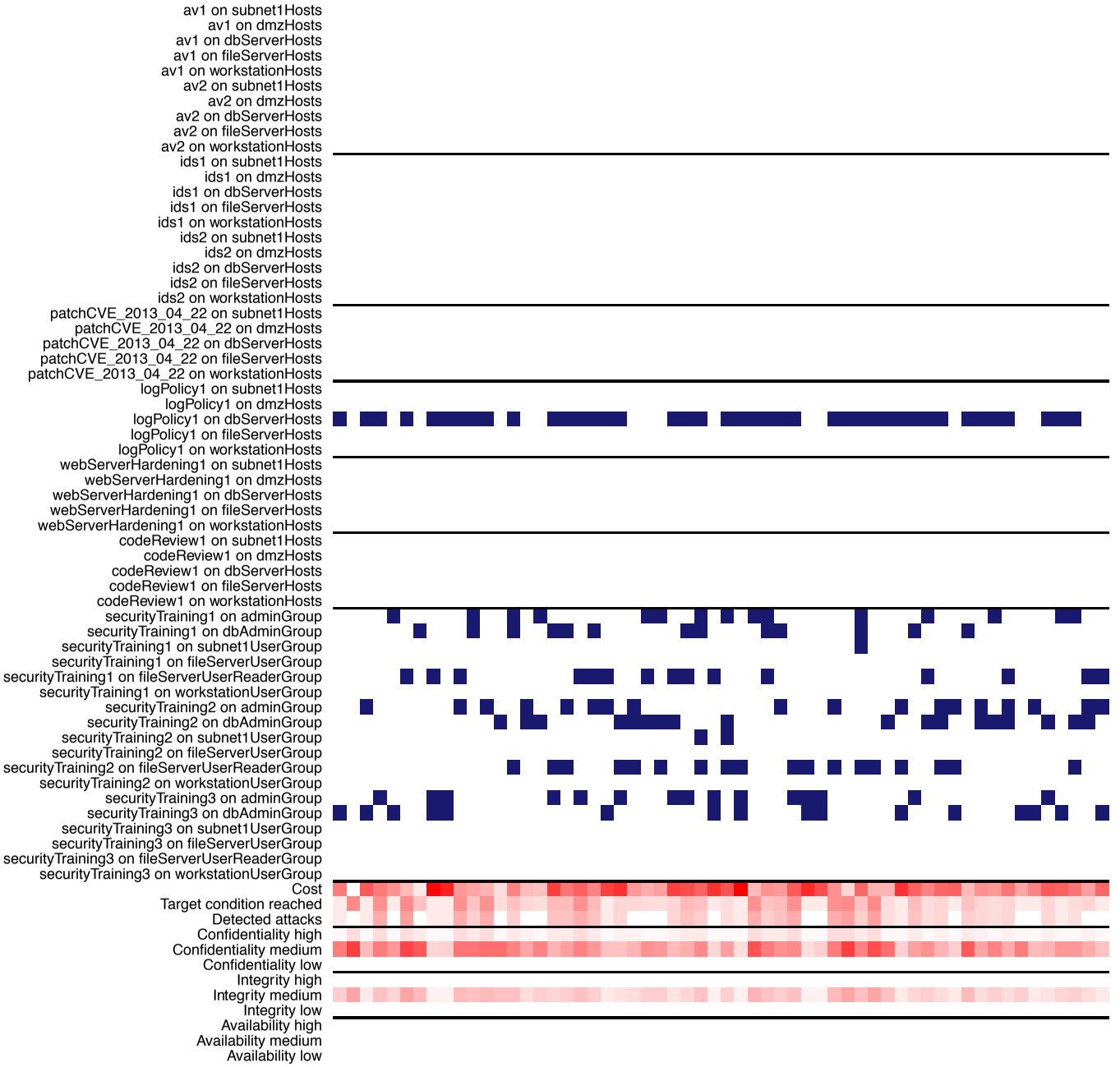
Results: Administrator



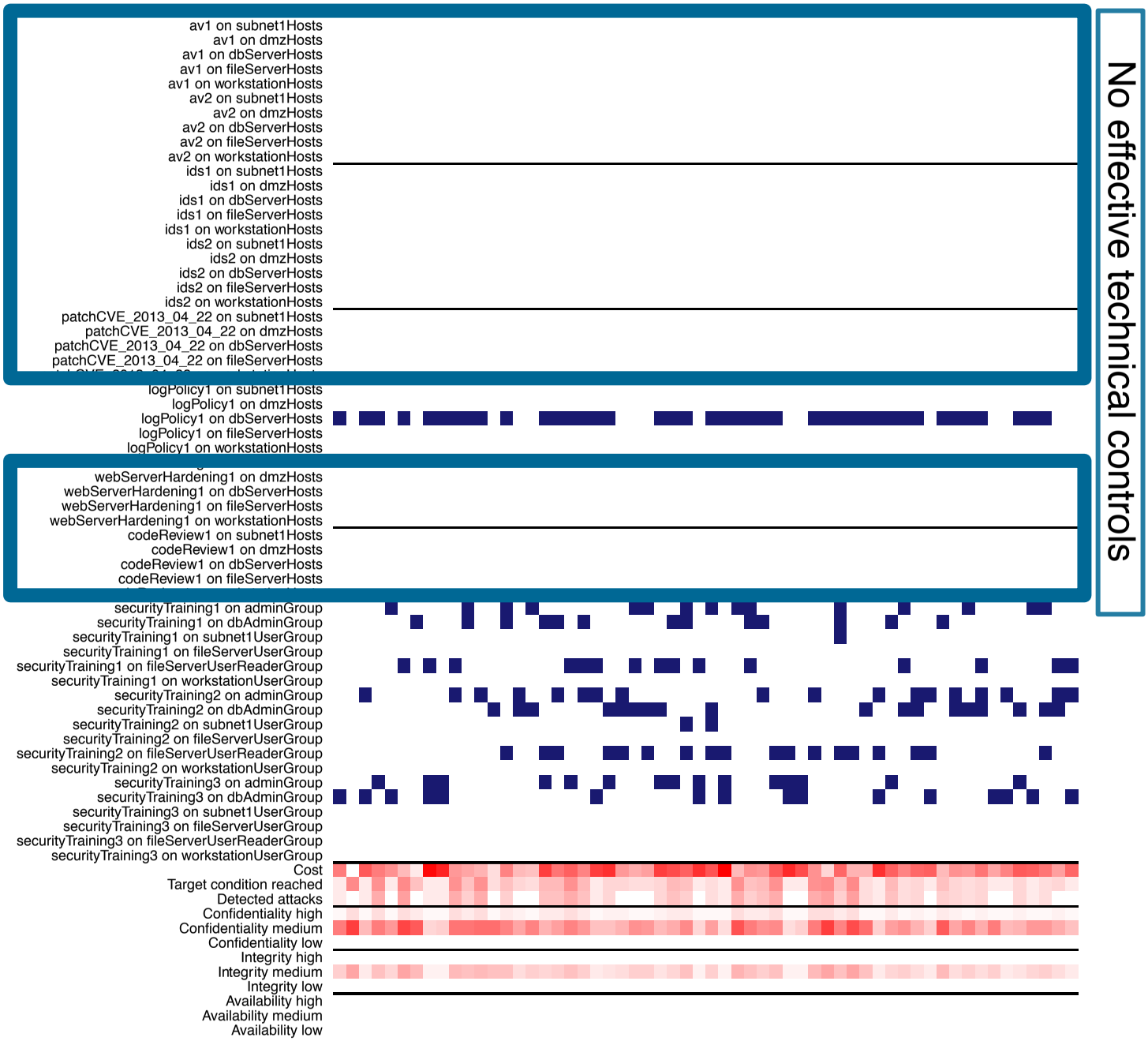
Log policy improves detection

Employee example attack trace

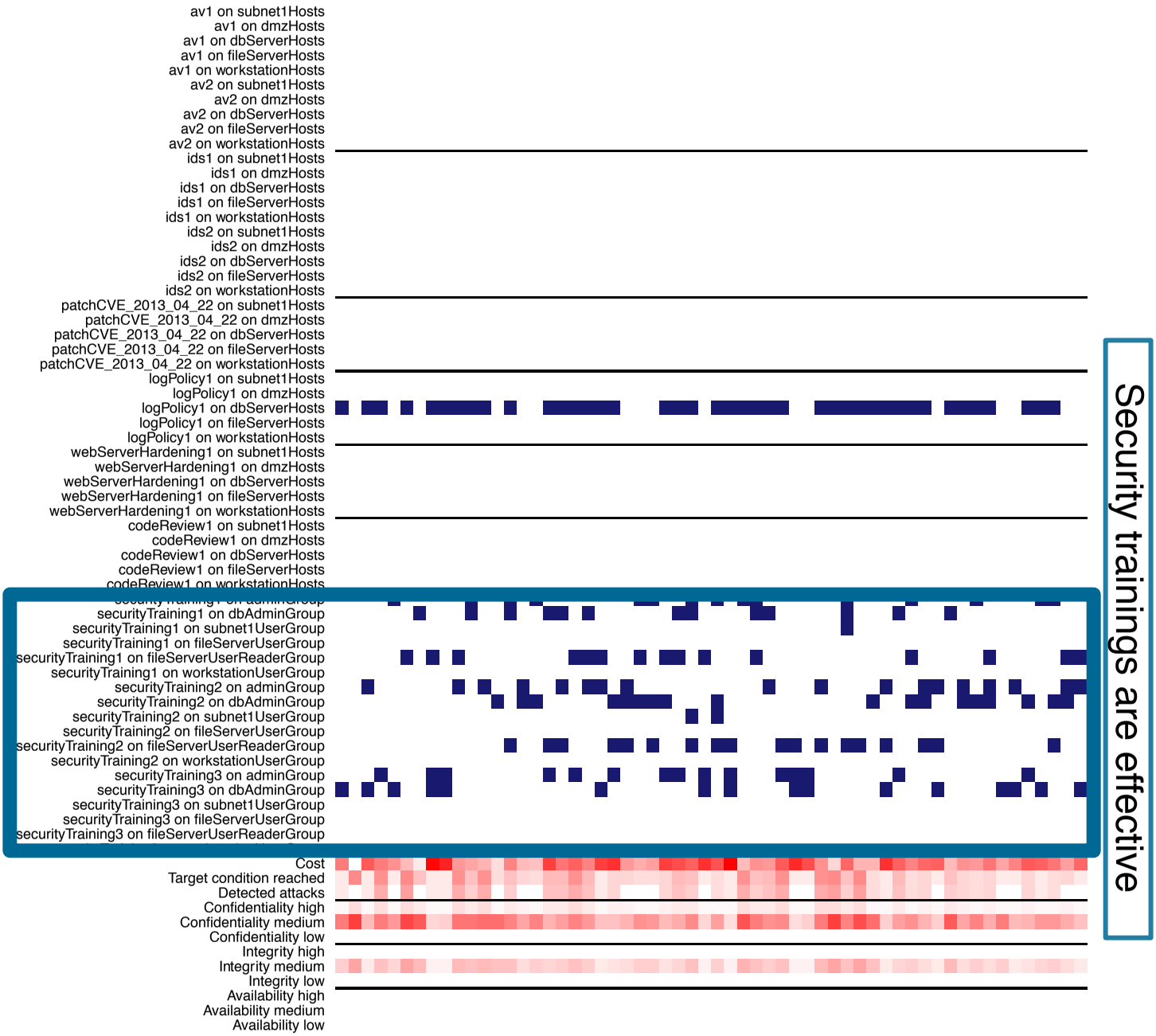
Results: Employee



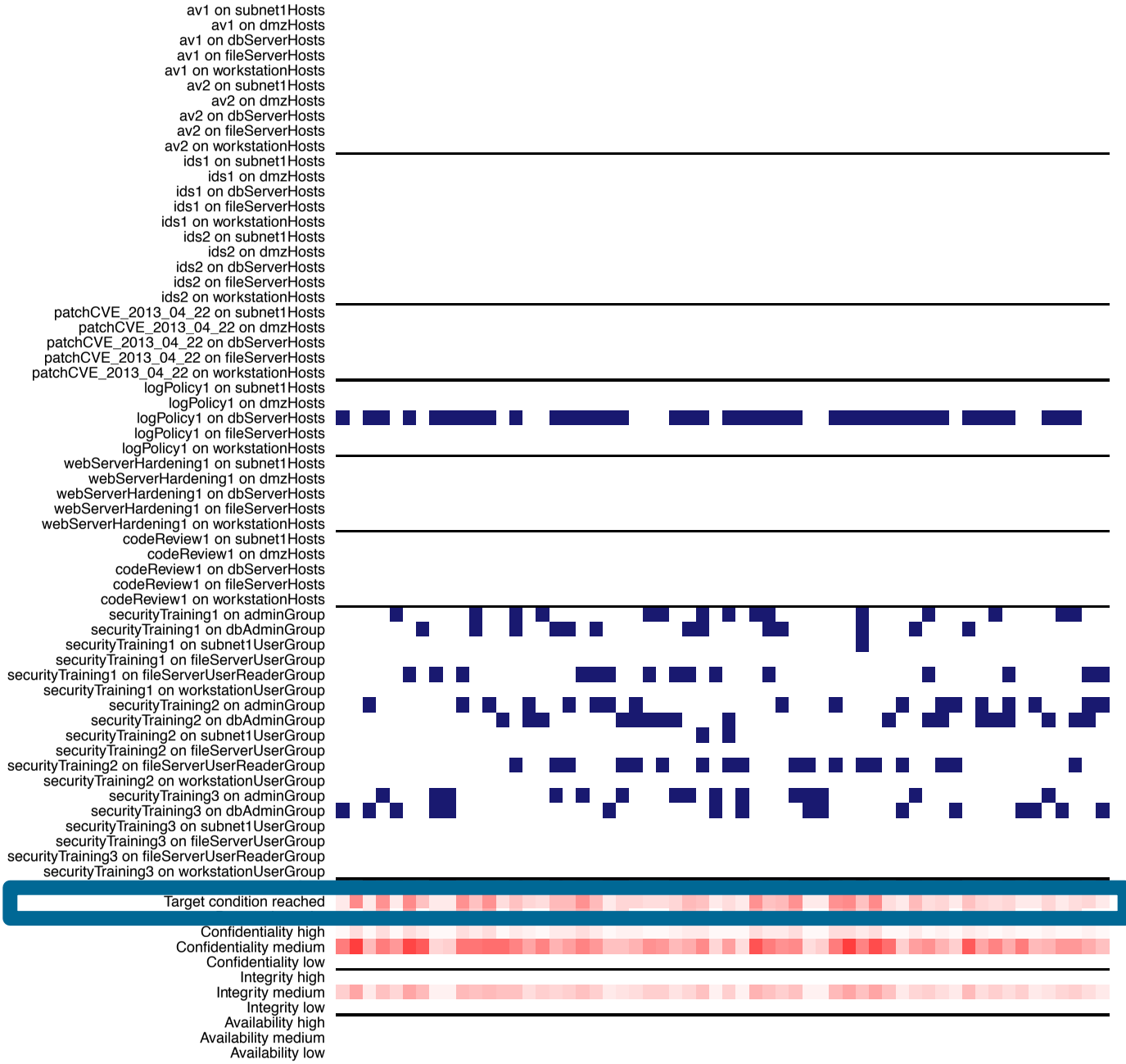
Results: Employee



Results: Employee

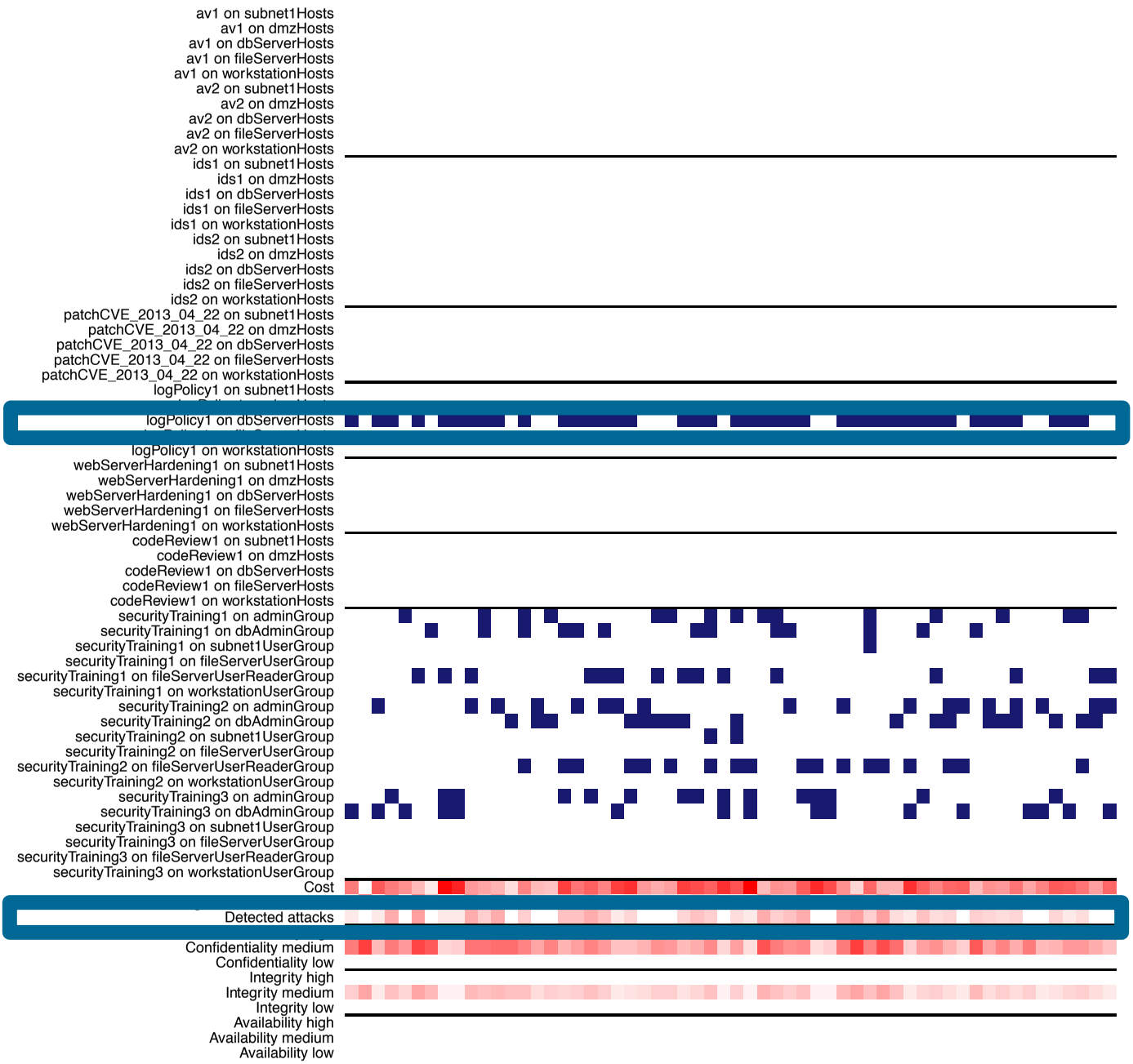


Results: Employee



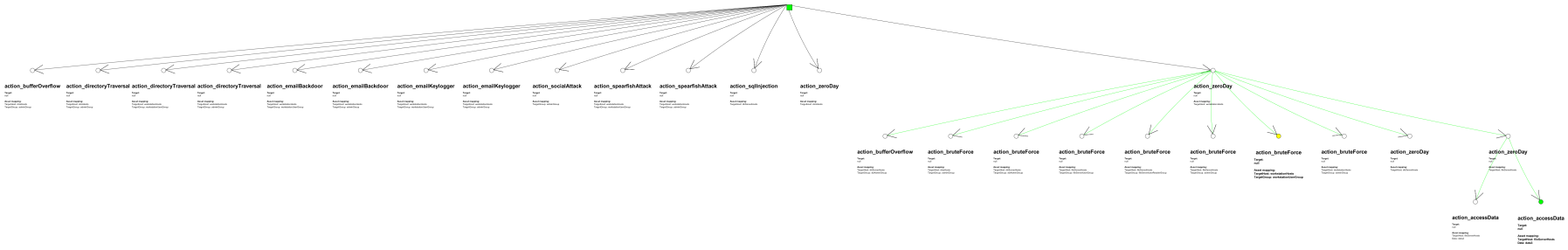
Success rate can be reduced from 46% to 6%

Results: Employee

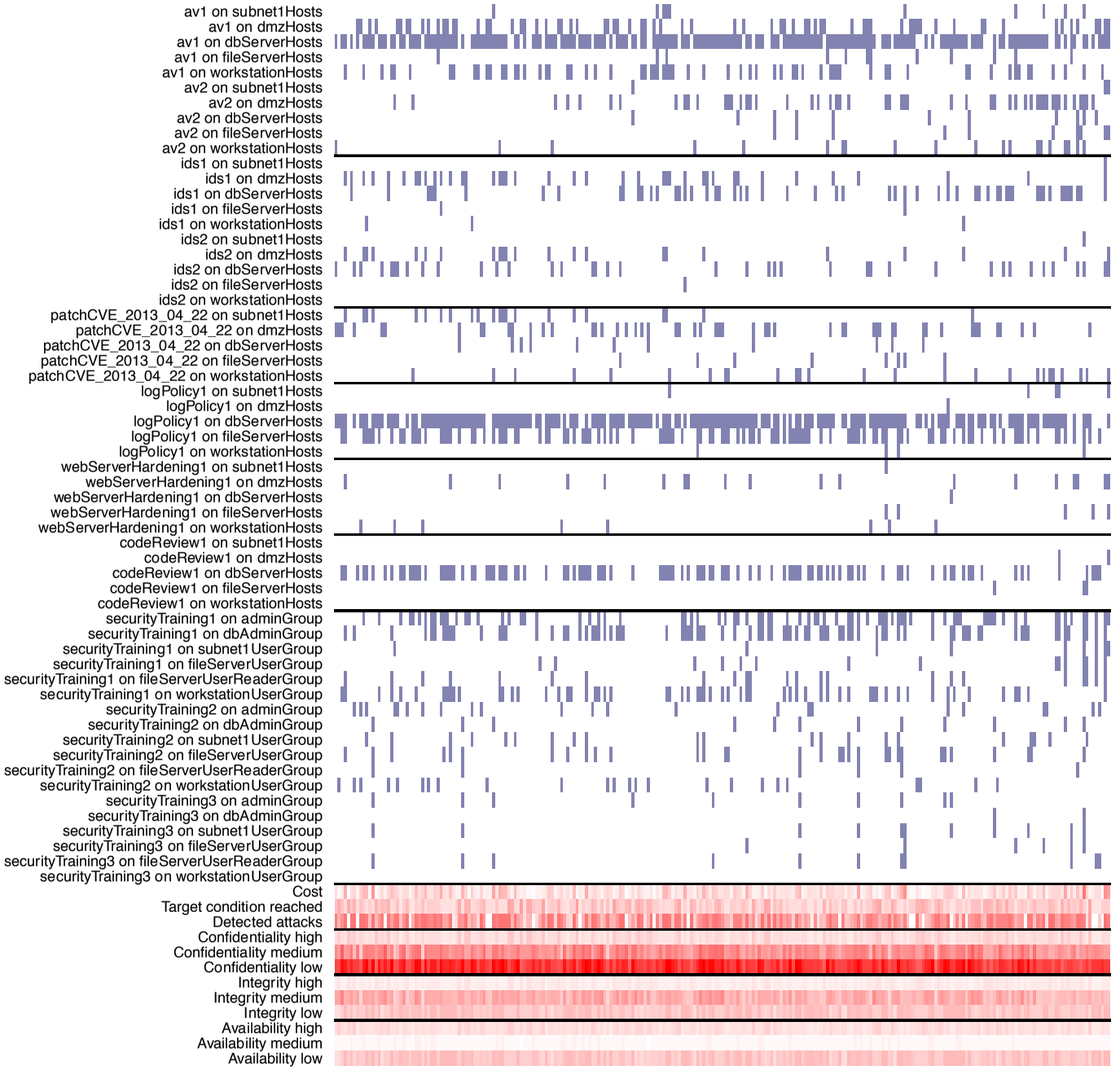


Log policy can increases detection rate to ~ 1/3

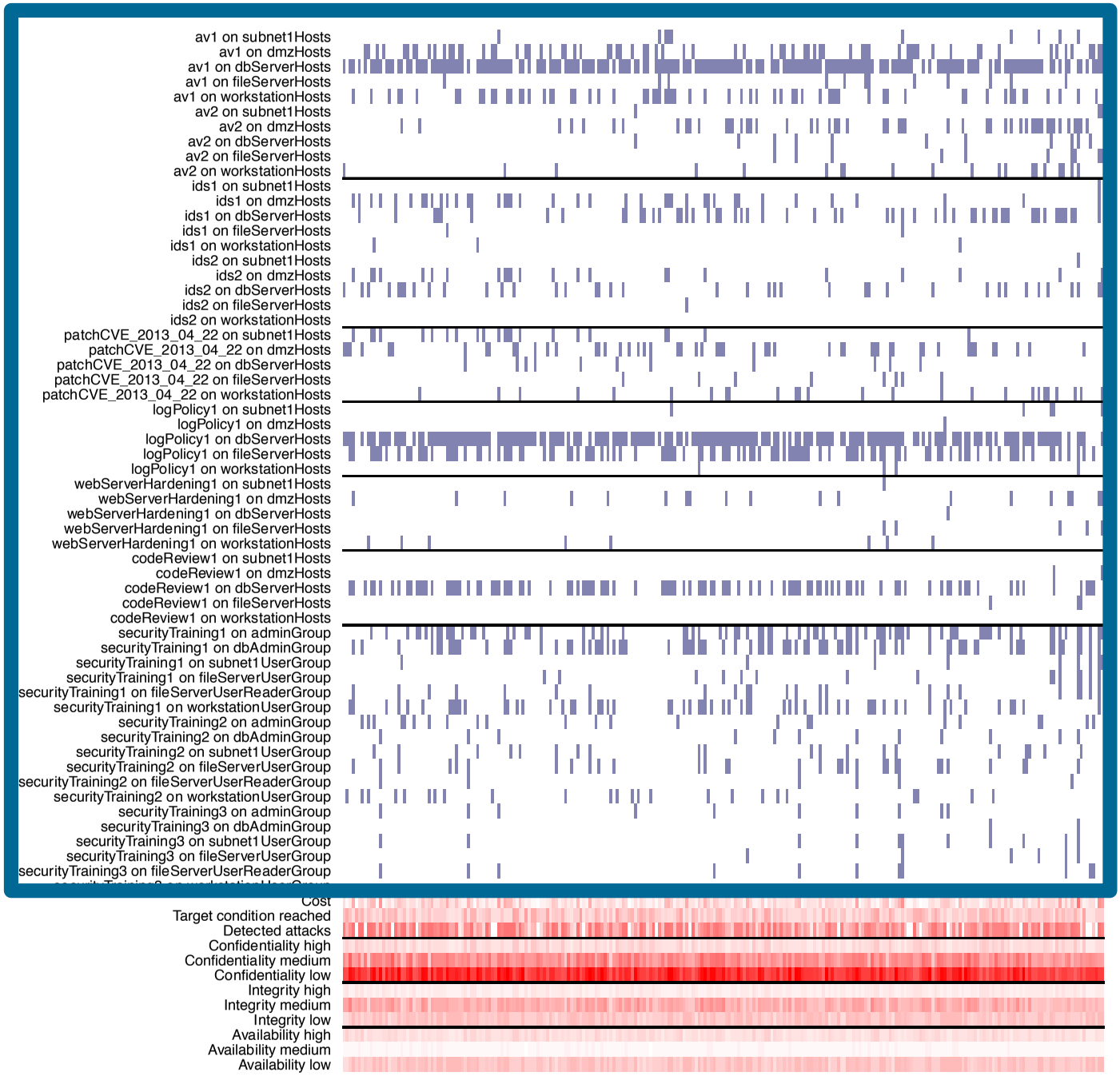
APT example attack trace



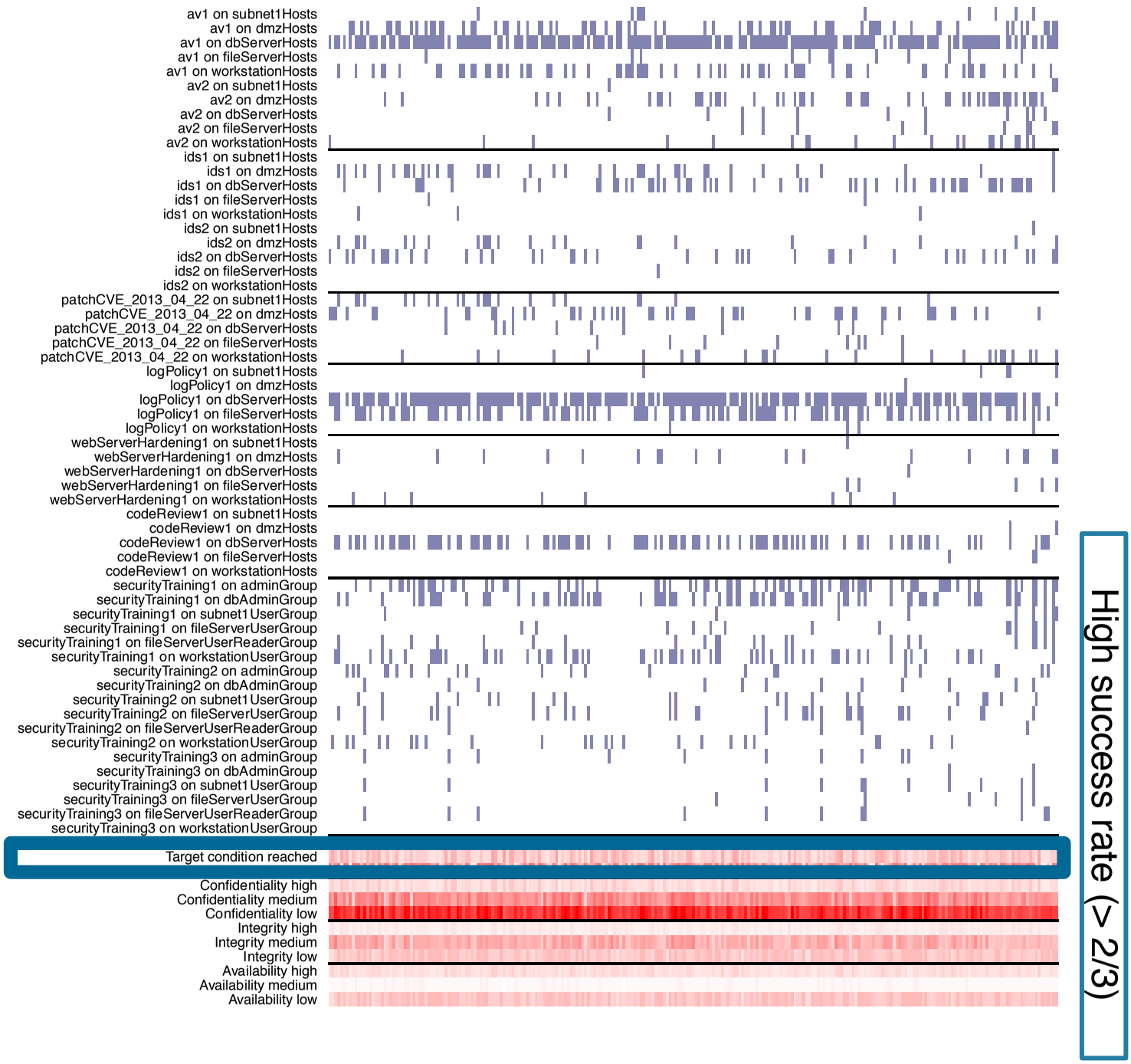
Results: Advanced persistent threat



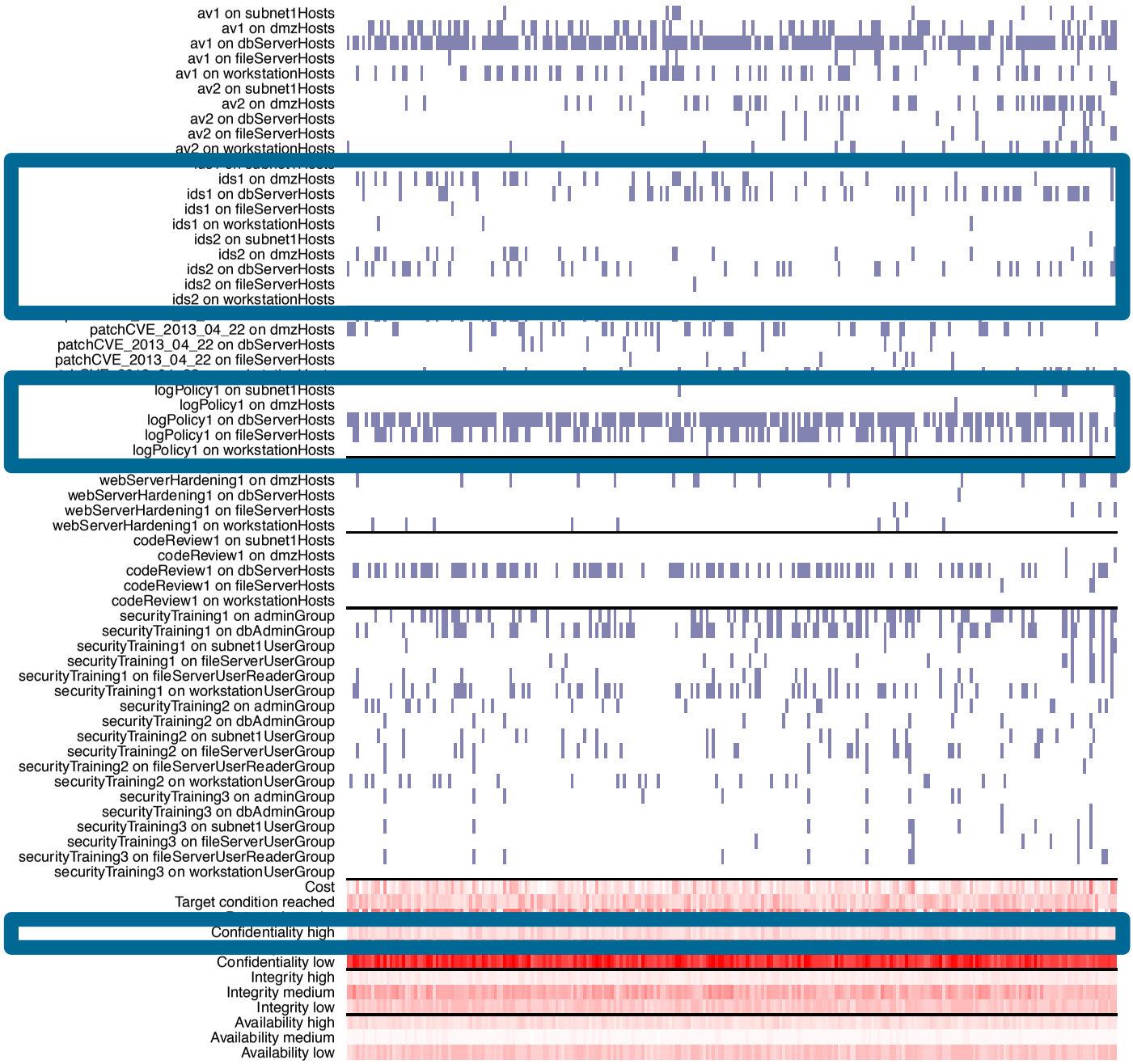
Results: Advanced persistent threat



Results: Advanced persistent threat

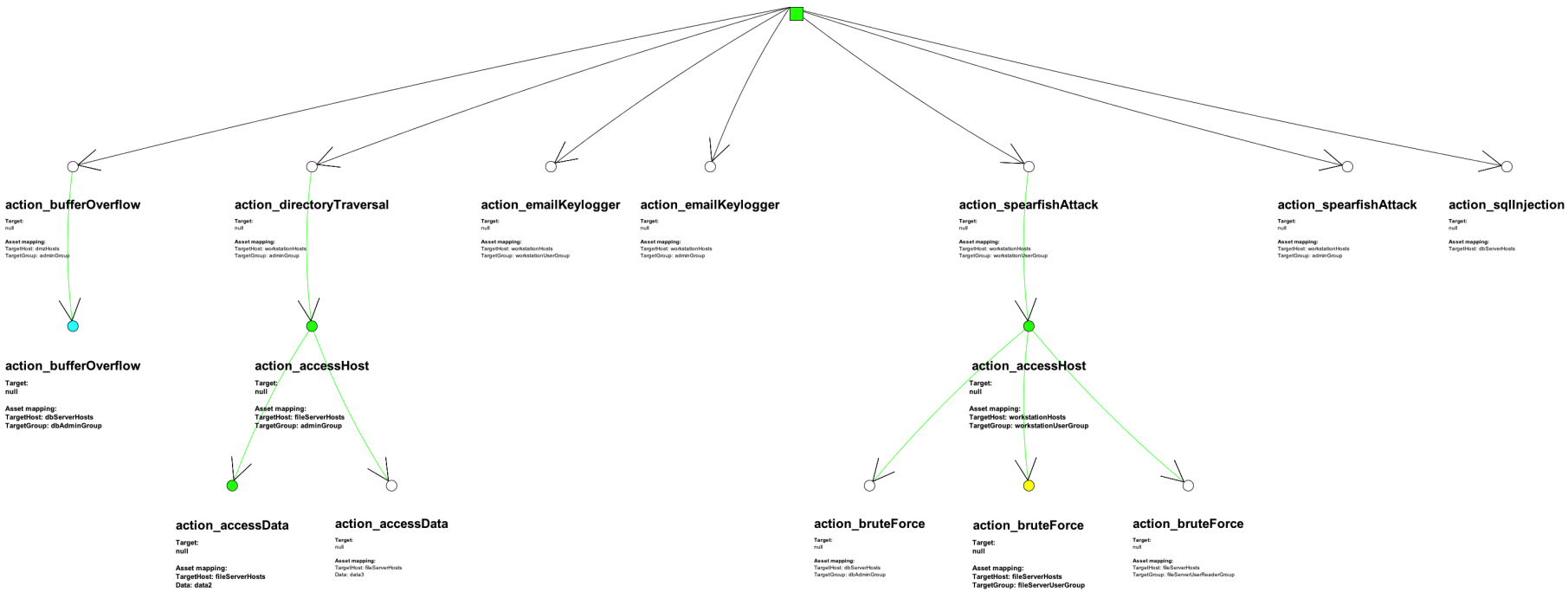


Results: Advanced persistent threat

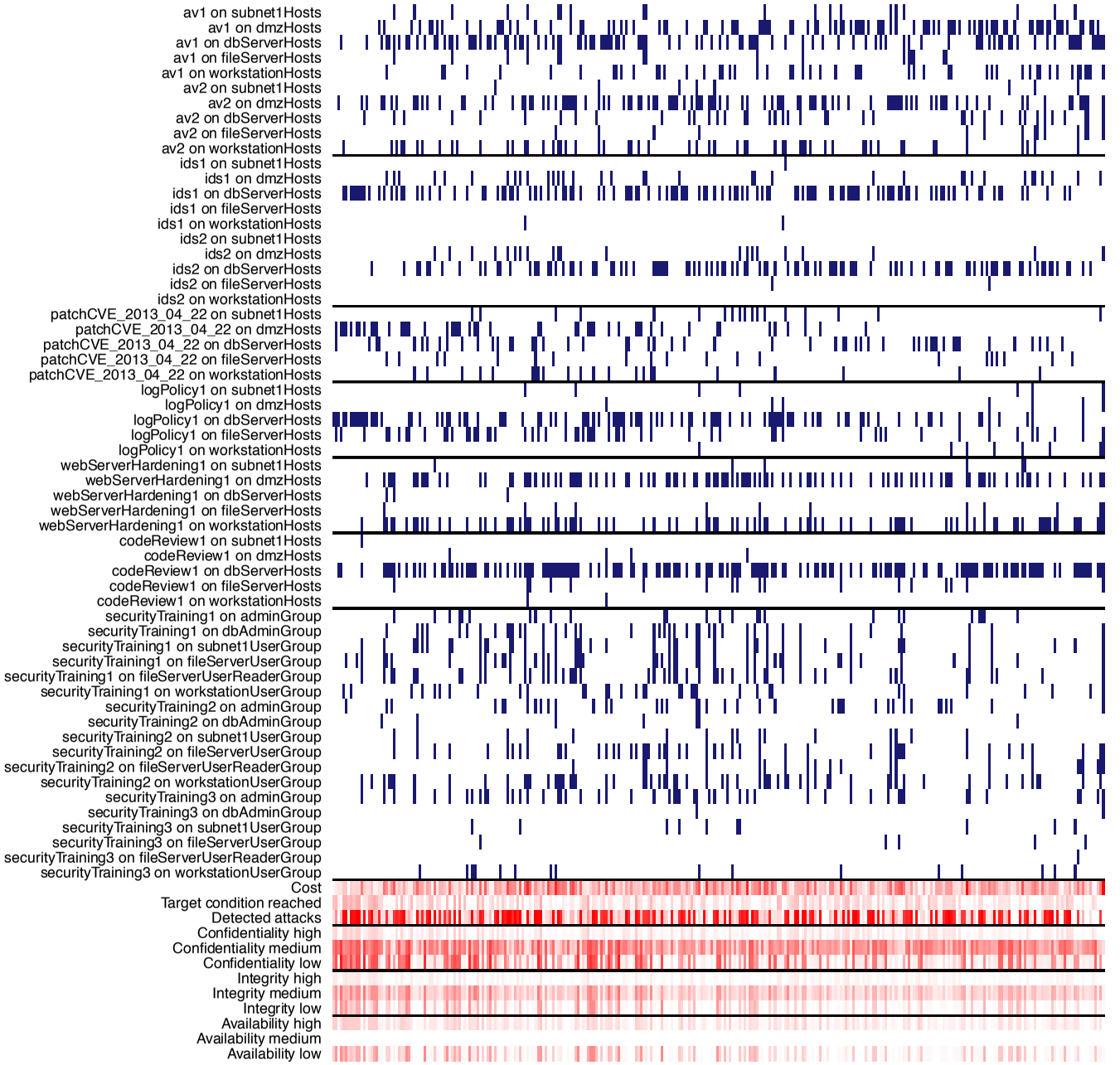


IDS and log policies can raise detection rates to ~ 2/3

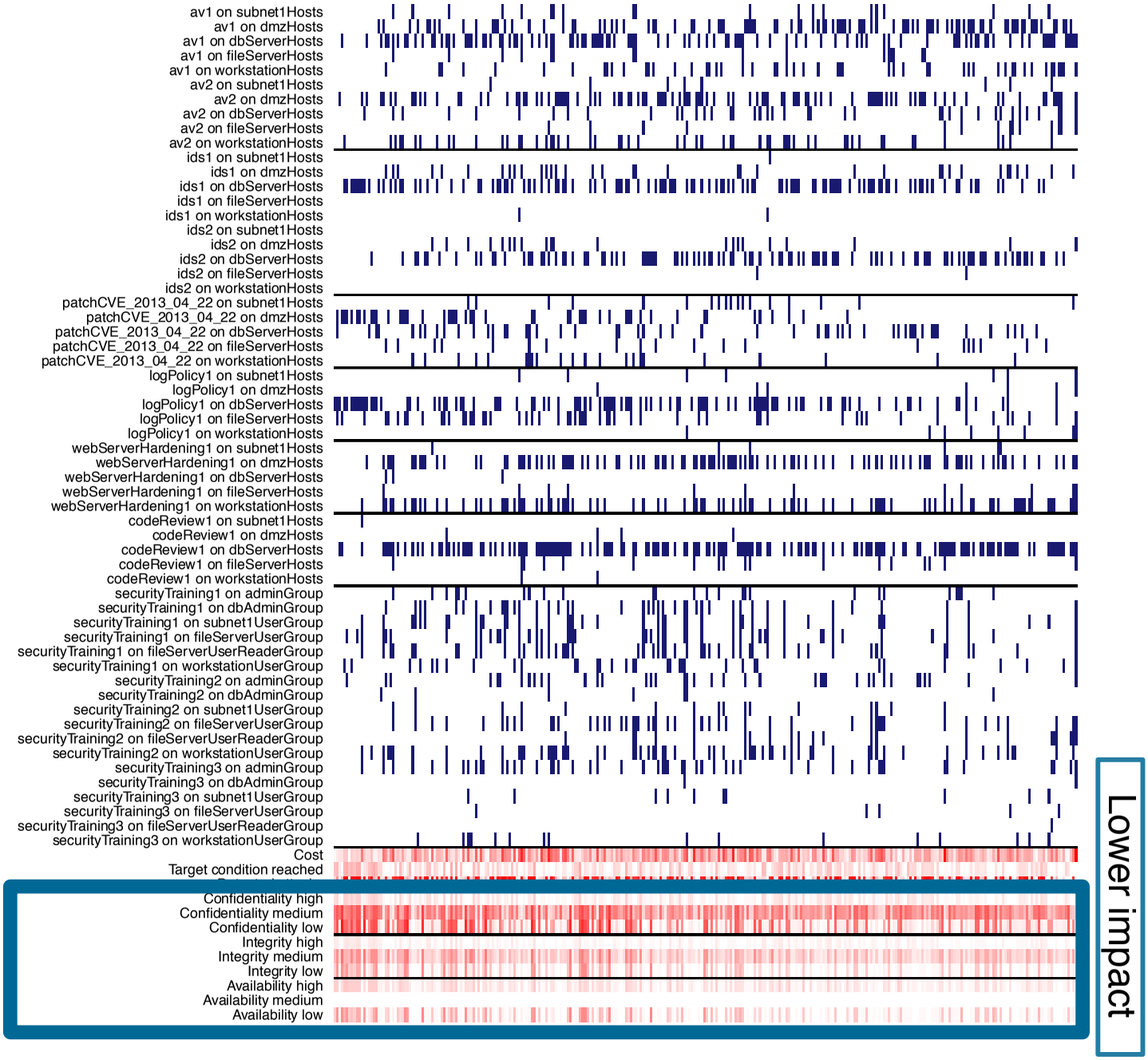
Skilled external example attack trace



Results: Skilled external

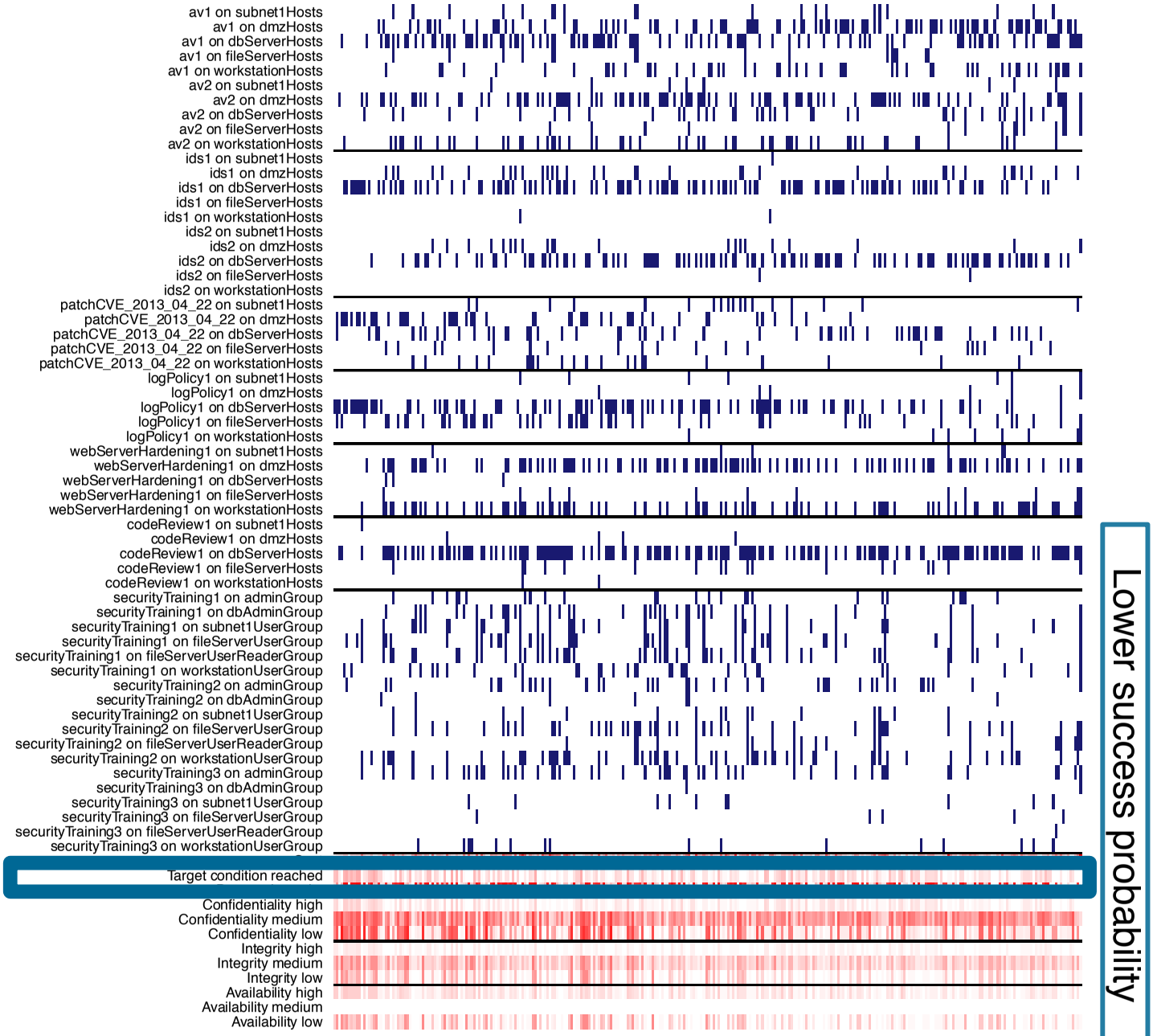


Results: Skilled external

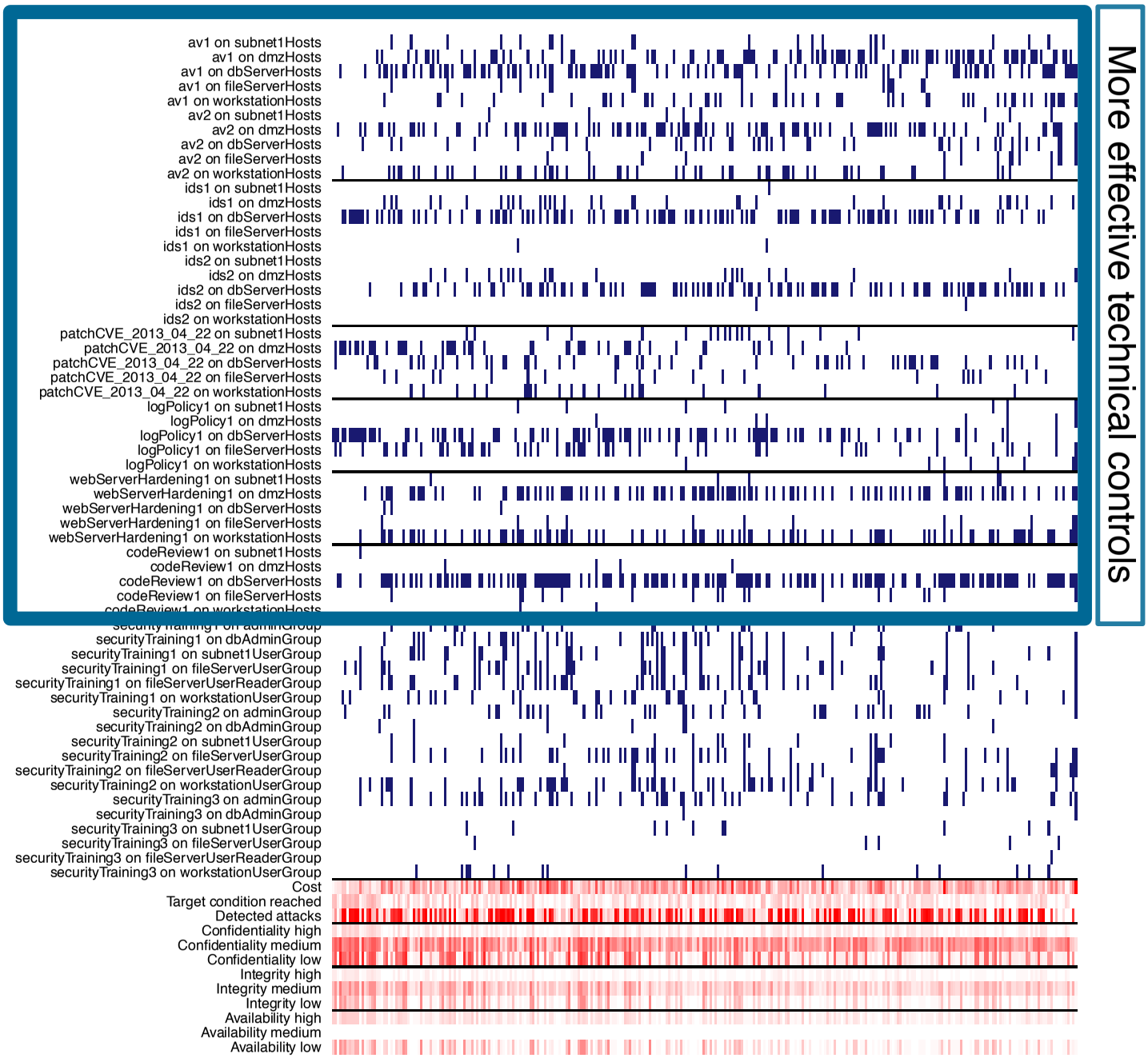


Lower impact

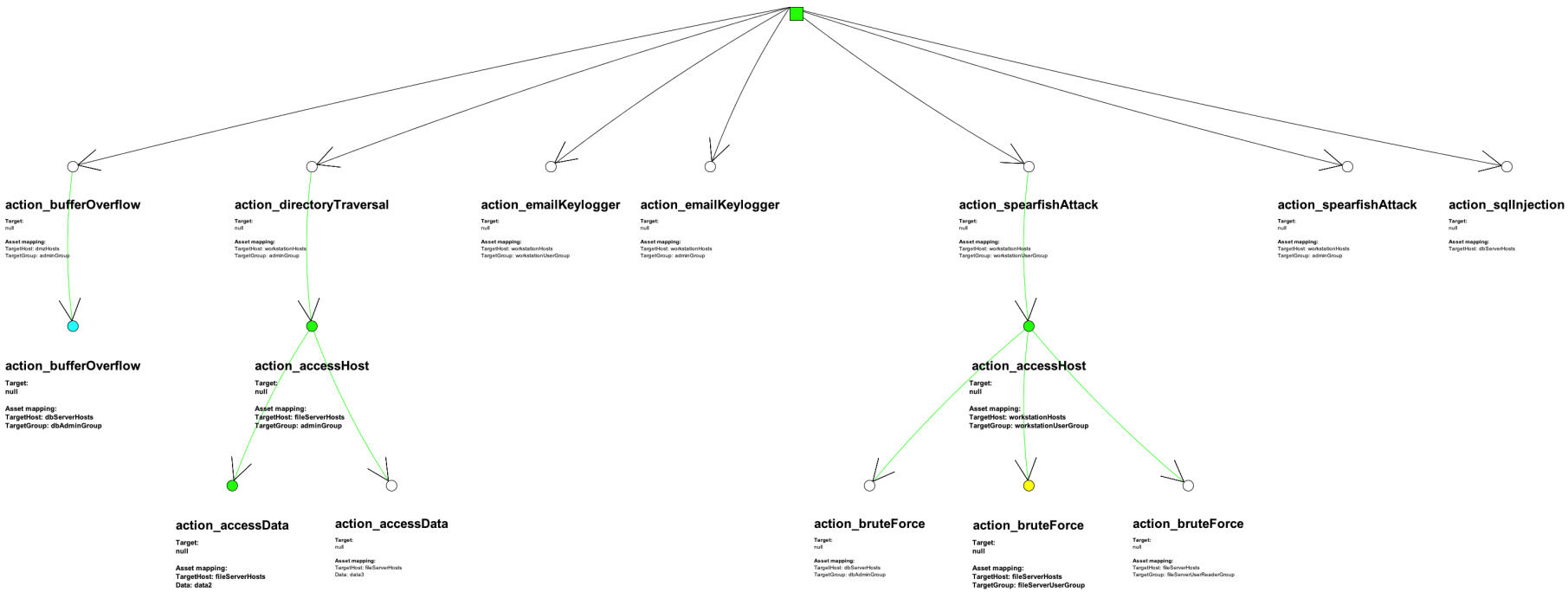
Results: Skilled external



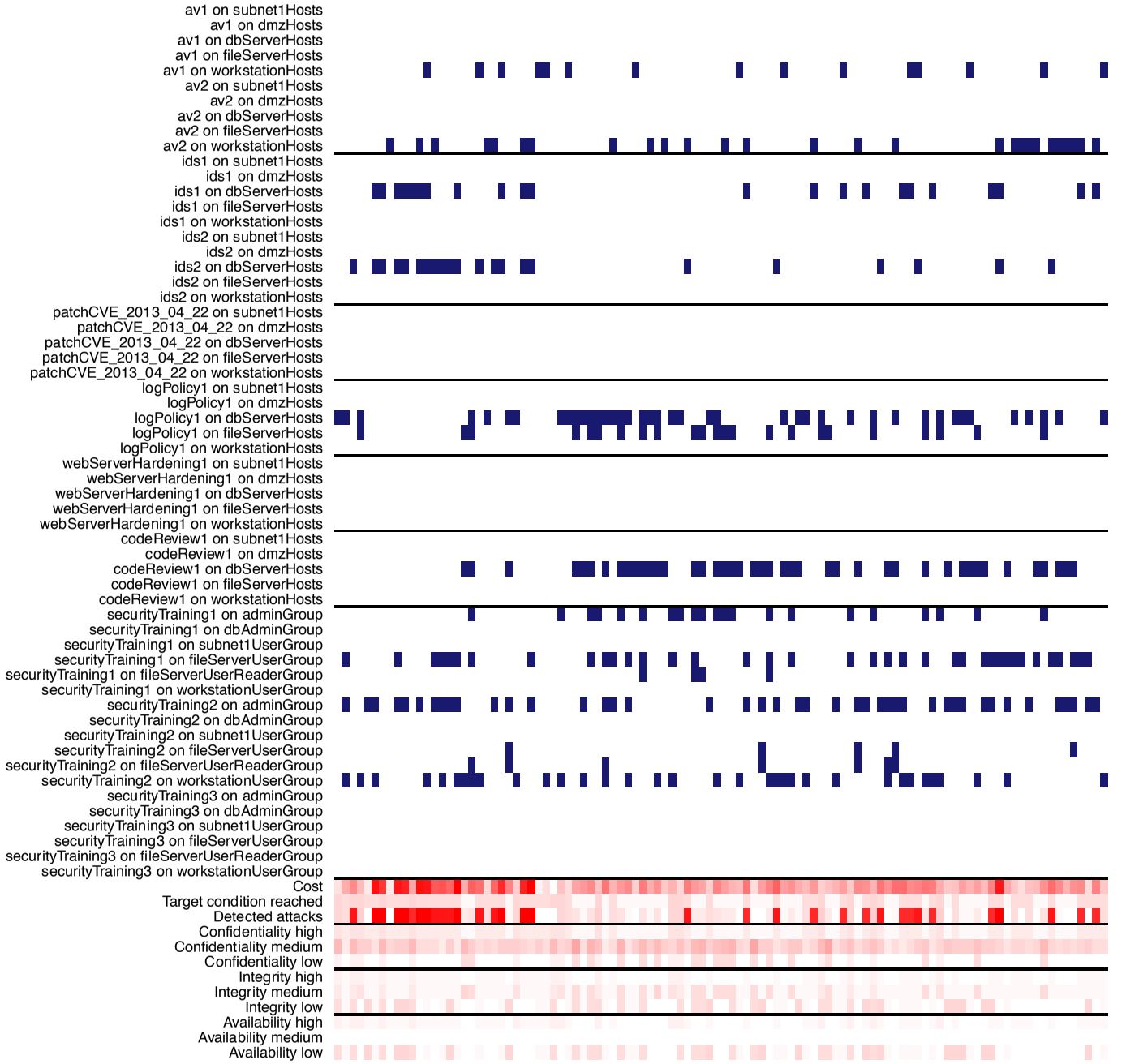
Results: Skilled external



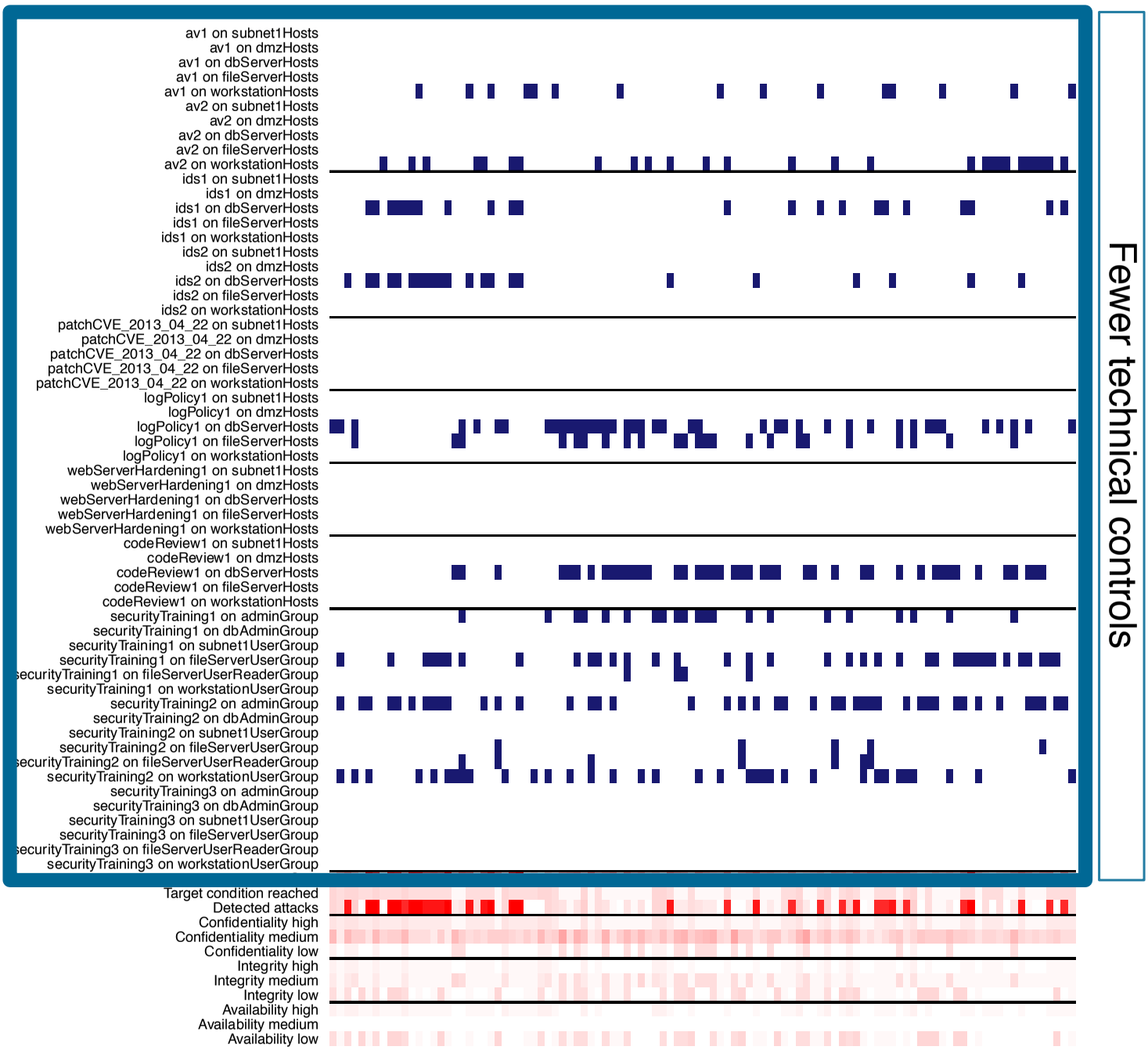
Unskilled external example attack trace



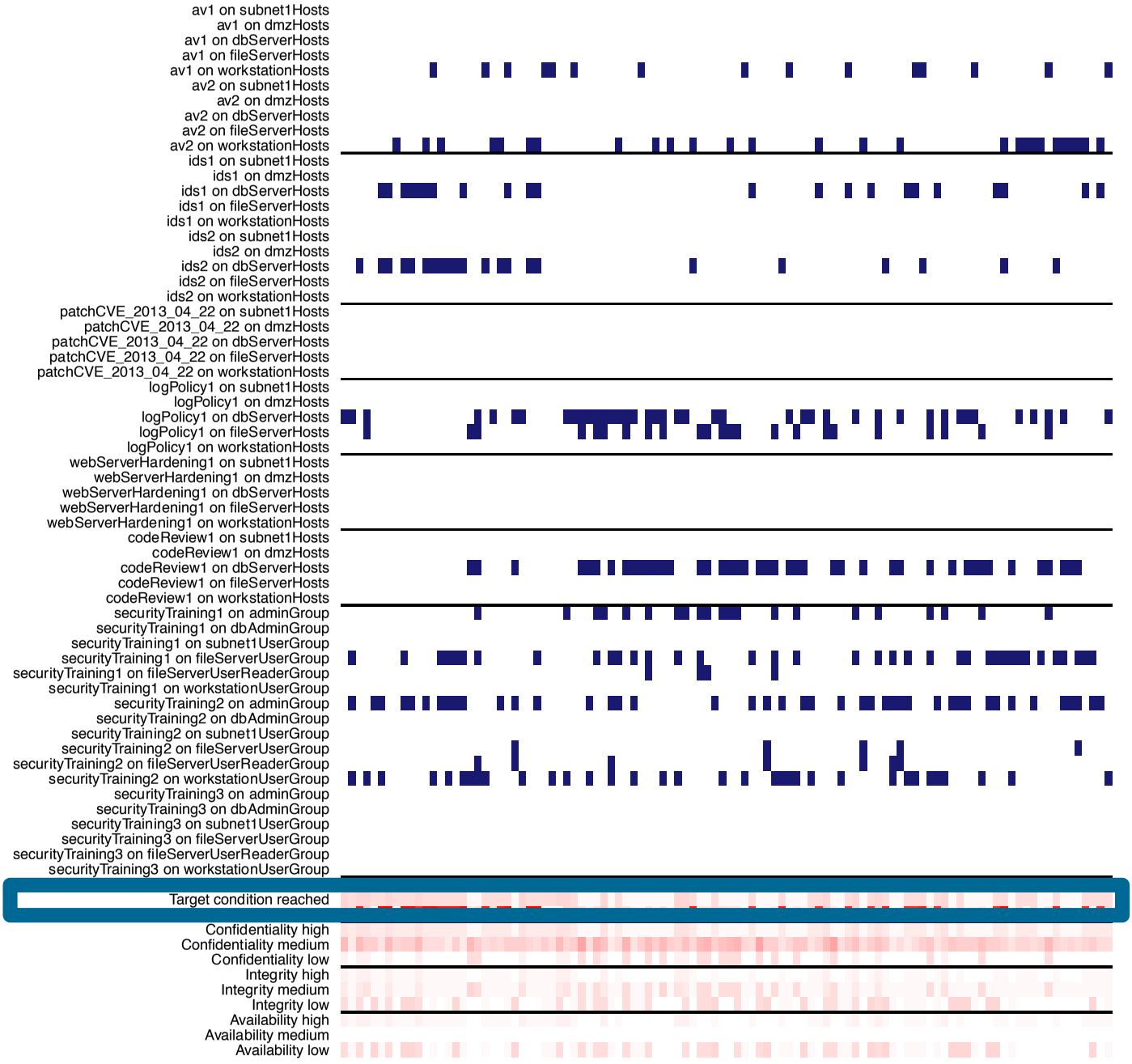
Results: Unskilled external



Results: Unskilled external



Results: Unskilled external



Success probability can be lowered to ~ 3%

Conclusions

Summary

- Simulation-Optimization framework for IT security
- Attacker-centric approach

Current research challenges

- Knowledge base: attack pattern formalization
- Simulation: cognitive and behavioral model
- Optimization:
  - cost of portfolio evaluations
  - cost of permutations

Future work

- Control selection $\rightarrow$ system design

  (very large design space + constraints)
- Problem-specific genotype structure

Q & A

Contact:
ekiesling@sba-research.org

Major security management challenges

- **Growing complexity** of information systems
- **Malicious threats** and targeted attacks
- **Increasingly sophisticated** attacks that exploit
  - software vulnerabilities
  - network vulnerabilities
  - social vulnerabilities
  - insider knowledge and access
  - etc.
- **Heterogeneous adversaries**

  hacktivists, script kiddies, insiders, advanced persistent threats . . .

$\rightarrow$ **Best way to cope with diverse threats?**

Implementation

Knowledge base

- Initial experiments with OWL ontologies
- `SWI-Prolog`:[1] current rule-based implementation
- `JPL`:[2] Java access

Simulation

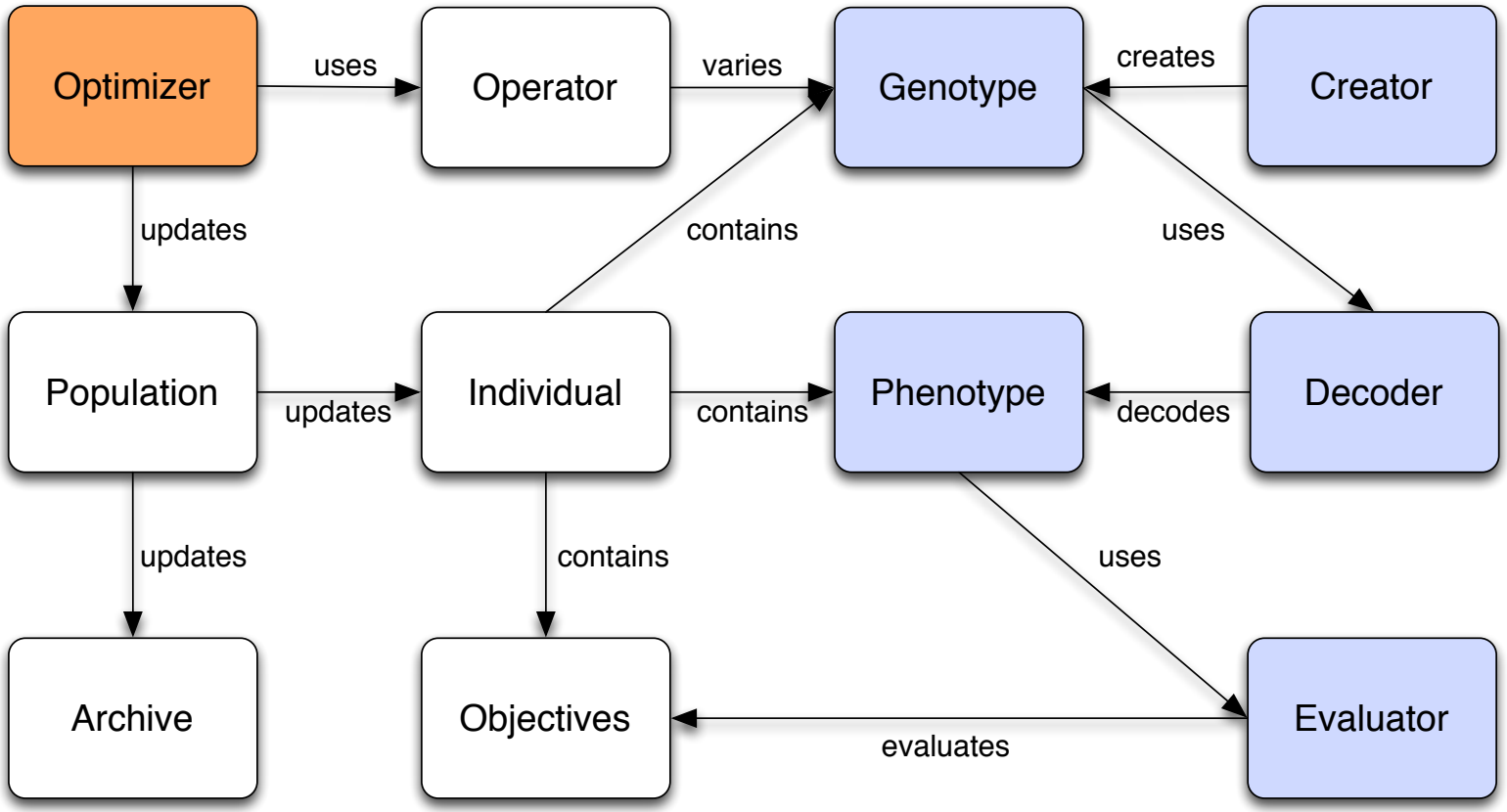
- `Java 1.6`
- `Mason 14`:[3] discrete-event core
- `Colt 1.2`:[4] random distributions
- `Jung 2.0.1`:[5] graph structures and visualization
- `Log4j`, `XStream`, `JUnit`, `Commons`, ...

Optimization

- `Opt4j 2.7`[6]: evolutionary computation framework

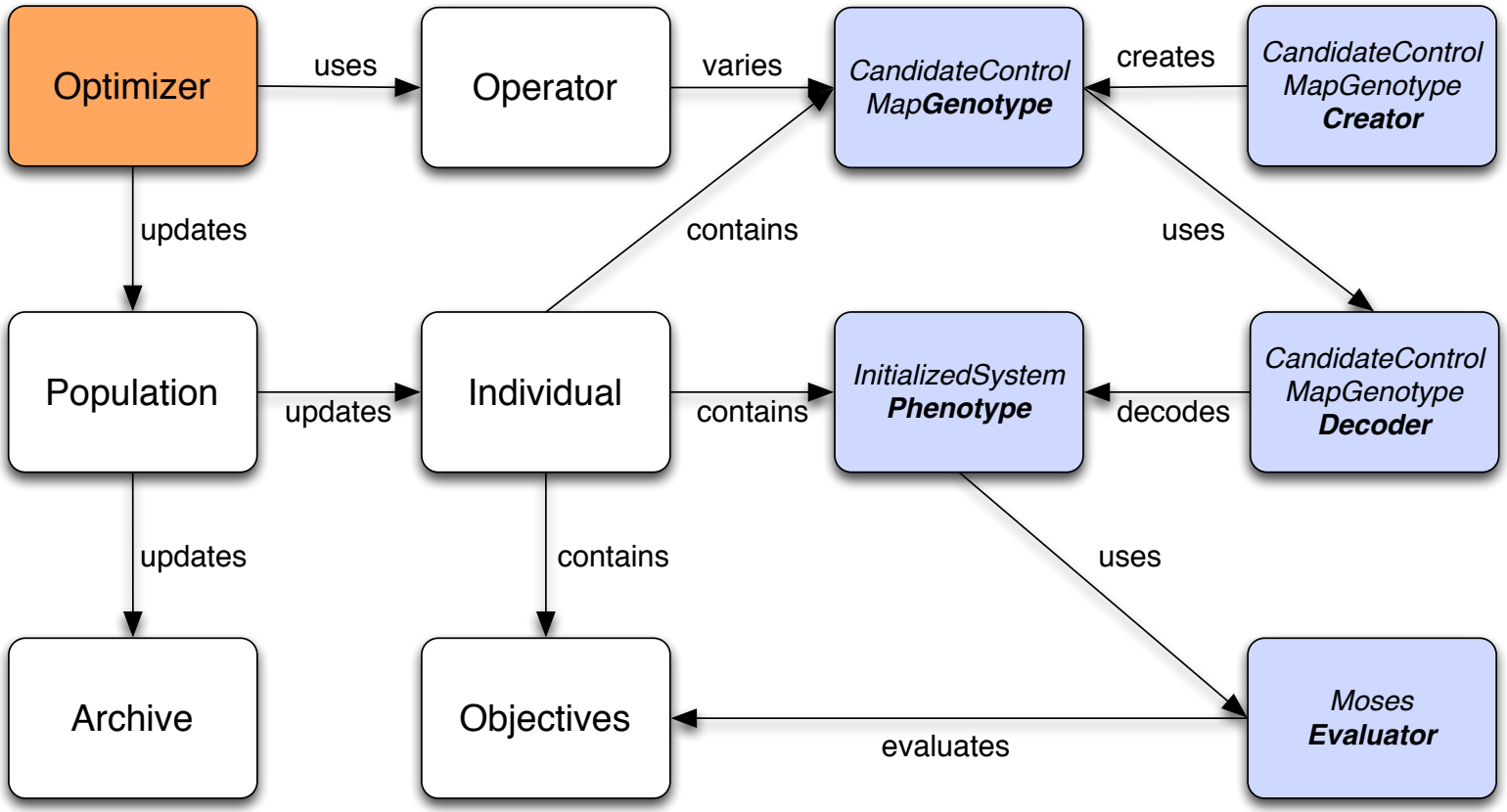
[1] `http://www.swi-prolog.org`
[2] `http://www.swi-prolog.org/packages/jpl`
[3] `http://cs.gmu.edu/~eclab/projects/mason/`
[4] `http://acs.lbl.gov/software/colt/`
[5] `http://jung.sourceforge.net/`
[6] `http://opt4j.sourceforge.net/`

Implementation: Optimization



Source: adapted from [?]

Implementation: Optimization



Source: adapted from [?]

Simulation approach - Motivation

1. Model dynamic attacks

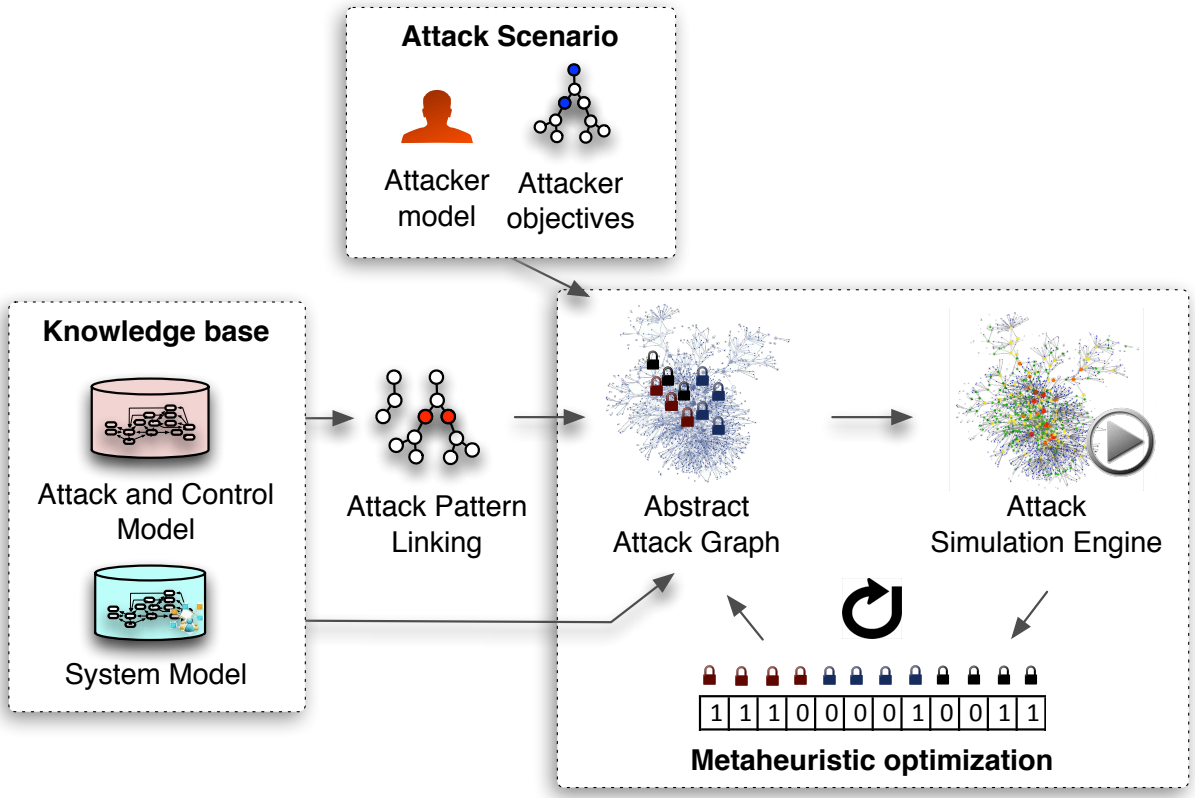
- Strategic behavior: Attackers adapt to the system architecture
- Attacks consist of multiple sequential steps
- Each step potentially changes the system state
- Steps chosen depend on results of previous actions
- Controls influence attacker strategy
- Not all attackers behave the same way

2. Tackle complexity

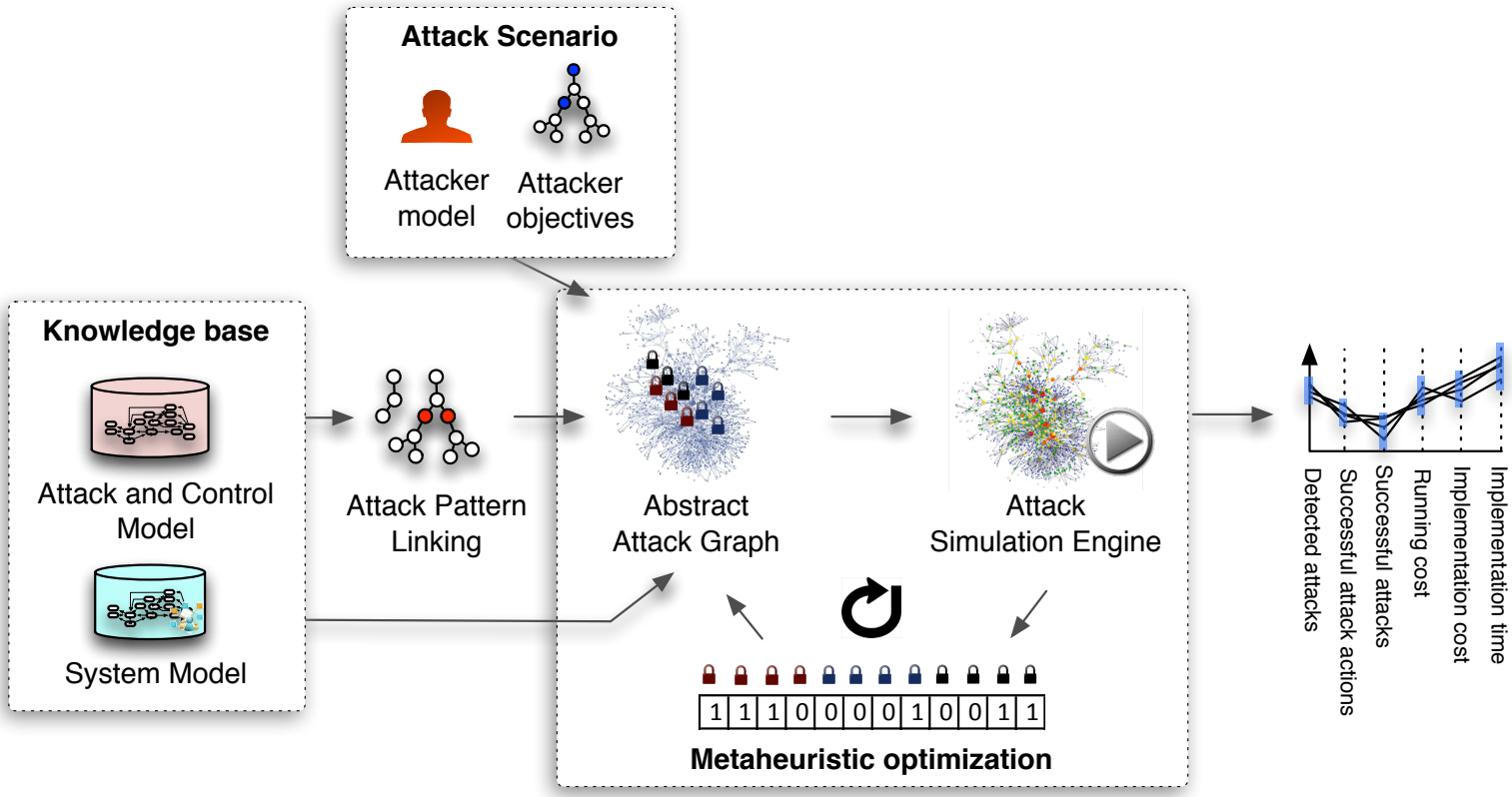
- Constructing and analysing full attack graphs infeasible for large systems
- Security problems inherently stochastic

3. Capture inherent variability

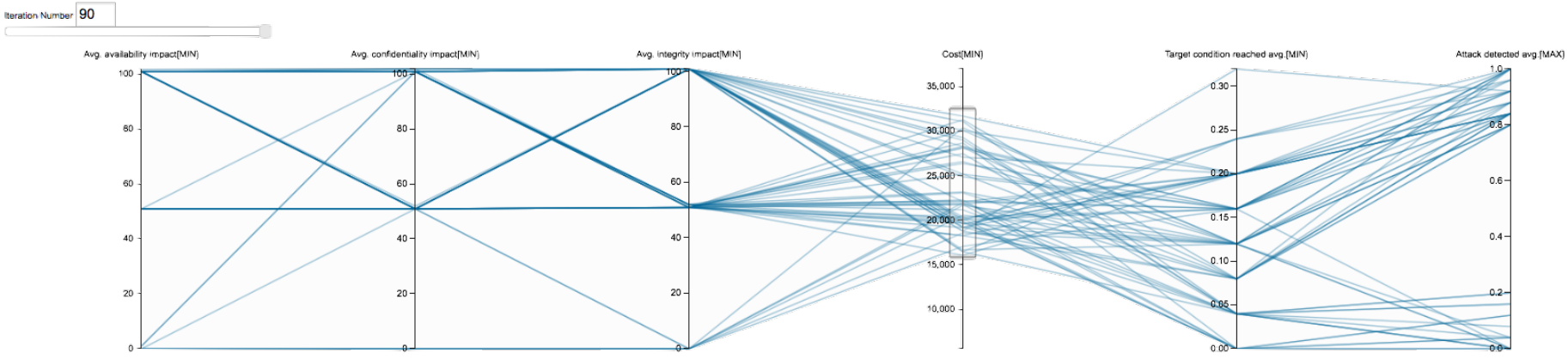
Decision support



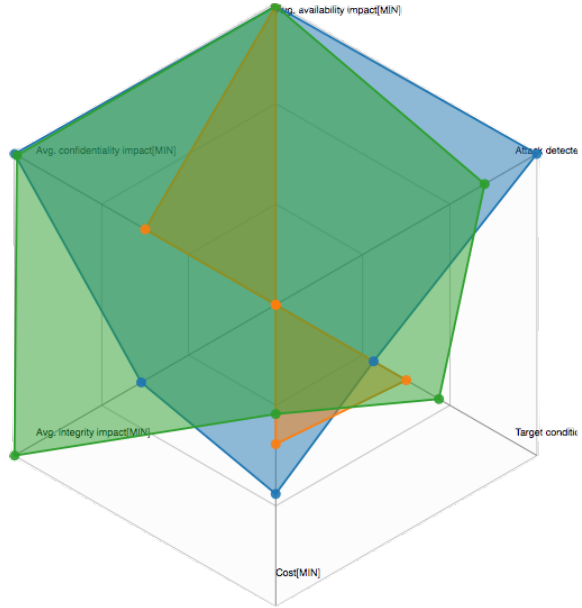
Decision support

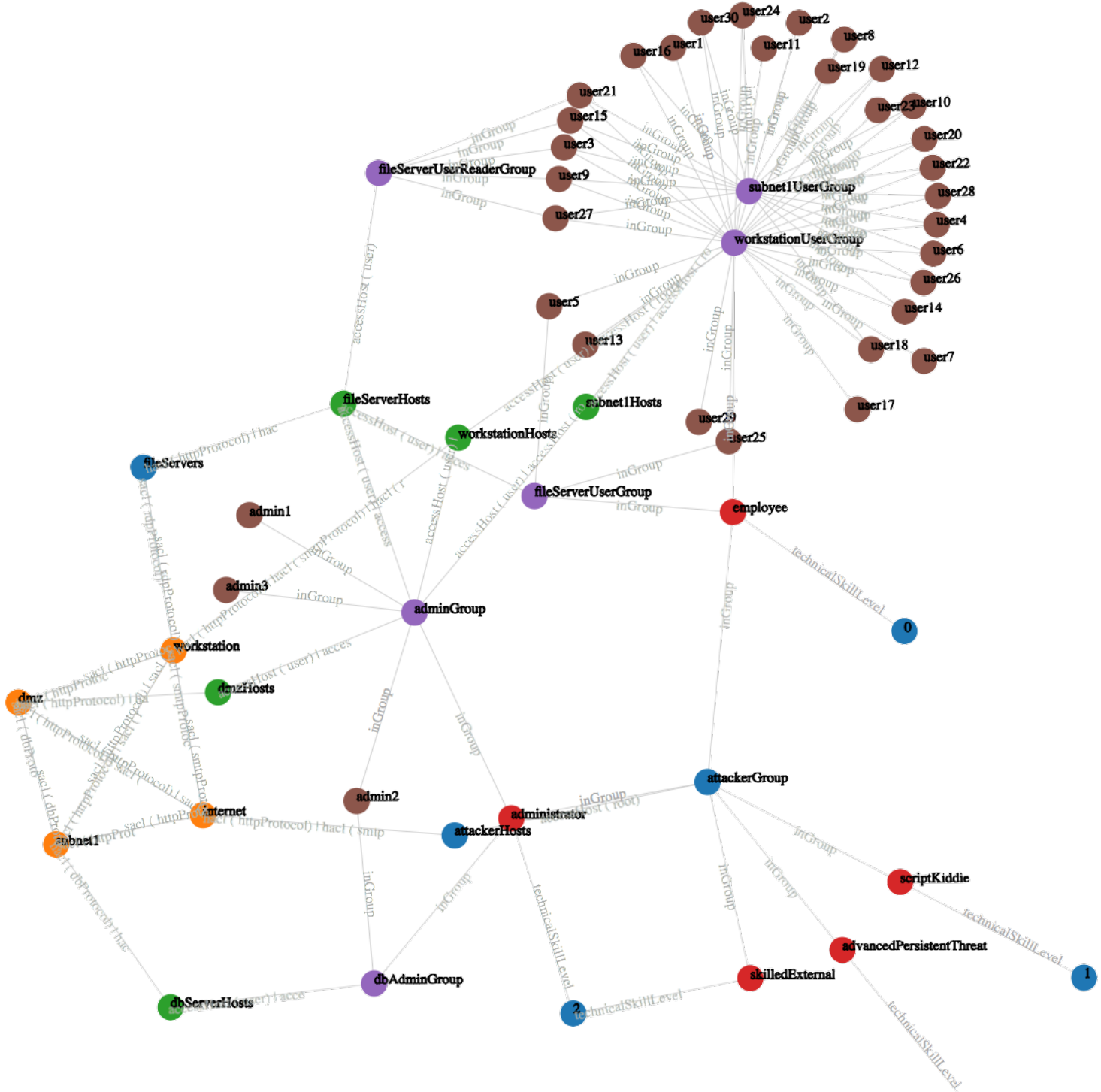


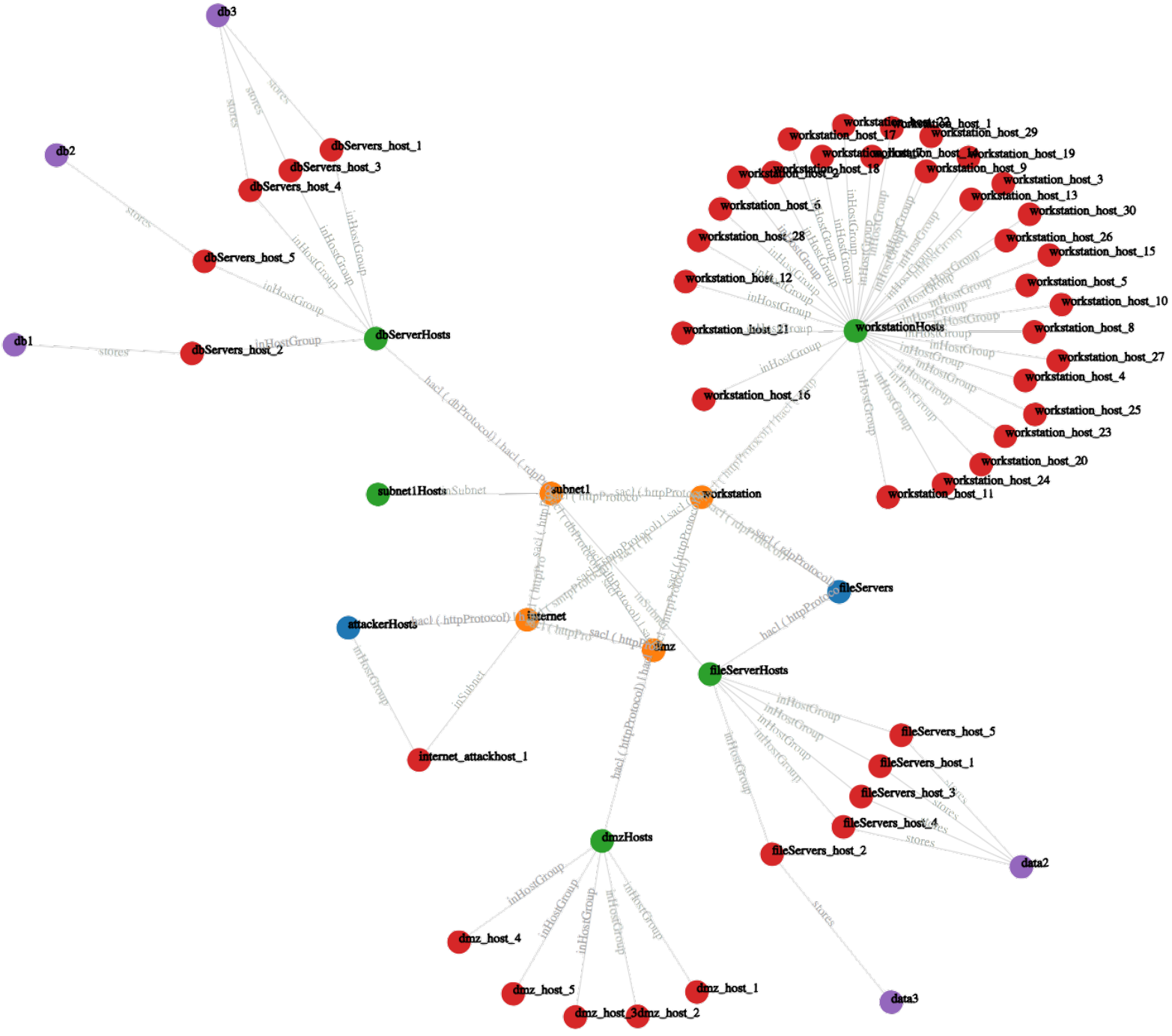
Decision support



iteration	evaluations	runtime[s]	Genotype	Avg. availab…	Avg. confide…	Avg. integrit…	Cost[MIN]	Target condi…	Attack detec…	id	
90	2250	6648.466	0001001001…	101	51	0	17100.0	0.16	0.0	3	☐
90	2250	6648.466	0001100000…	101	102	52	23200.0	0.12	1.0	5	☐
90	2250	6648.466	0101000100…	101	101	101	31100.0	0.04	0.2	8	☑
90	2250	6648.466	0101000000…	101	101	51	18300.0	0.08	0.8	12	☑
90	2250	6648.466	0101000100…	101	51	101	27100.0	0.2	0.88	13	☐
90	2250	6648.466	1000100000…	101	101	52	16100.0	0.2	0.84	14	☑
90	2250	6648.466	0101000000…	51	51	51	19700.0	0.12	0.8	16	☐
90	2250	6648.466	1010100000…	101	101	101	18800.0	0.32	0.92	18	☐
90	2250	6648.466	0101000000…	51	51	51	28700.0	0.0	0.04	20	☐
90	2250	6648.466	0101000100…	102	102	101	16500.0	0.04	0.16	21	☐
90	2250	6648.466	1000001101…	101	101	52	21200.0	0.04	0.08	22	☐




user30
user24
user2
user8
user16
user1
user11
user19
user12
user21
user23
user10
user15
user20
user3
user22
fileServerUserReaderGroup
user9
subnet1UserGroup
user28
user27
user4
workstationUserGroup
user6
user26
user5
user14
user13
user18
user7
fileServerHosts
subnet1Hosts
user17
user29
workstationHosts
user25
fileServers
fileServerUserGroup
employee
admin1
technicalSkillLevel
admin3
adminGroup
0
workstation
dmz
dmzHosts
attackerGroup
admin2
administrator
internet
attackerHosts
scriptKiddie
subnet1
advancedPersistentThreat
dbAdminGroup
skilledExternal
1
dbServerHosts
2
inGroup
accessHost (user)
technicalSkillLevel


db3
db2
db1
dbServers_host_1
dbServers_host_3
dbServers_host_4
dbServers_host_5
dbServers_host_2
dbServerHosts
workstation_host_17
workstation_host_29
workstation_host_19
workstation_host_18
workstation_host_9
workstation_host_2
workstation_host_3
workstation_host_13
workstation_host_6
workstation_host_30
workstation_host_28
workstation_host_26
workstation_host_15
workstation_host_12
workstation_host_5
workstation_host_10
workstation_host_21
workstationHosts
workstation_host_8
workstation_host_27
workstation_host_4
workstation_host_16
workstation_host_25
workstation_host_23
workstation_host_20
workstation_host_24
workstation_host_11
subnet1Hosts
subnet1
workstation
fileServers
attackerHosts
internet
dmz
fileServerHosts
internet_attackhost_1
fileServers_host_5
fileServers_host_1
fileServers_host_3
fileServers_host_4
fileServers_host_2
data2
dmzHosts
dmz_host_4
dmz_host_5
dmz_host_1
dmz_host_3
dmz_host_2
data3
stores
inHostGroup
inSubnet
hacl (httpProtocol)
hacl (dbProtocol)
hacl (rdpProtocol)

Results

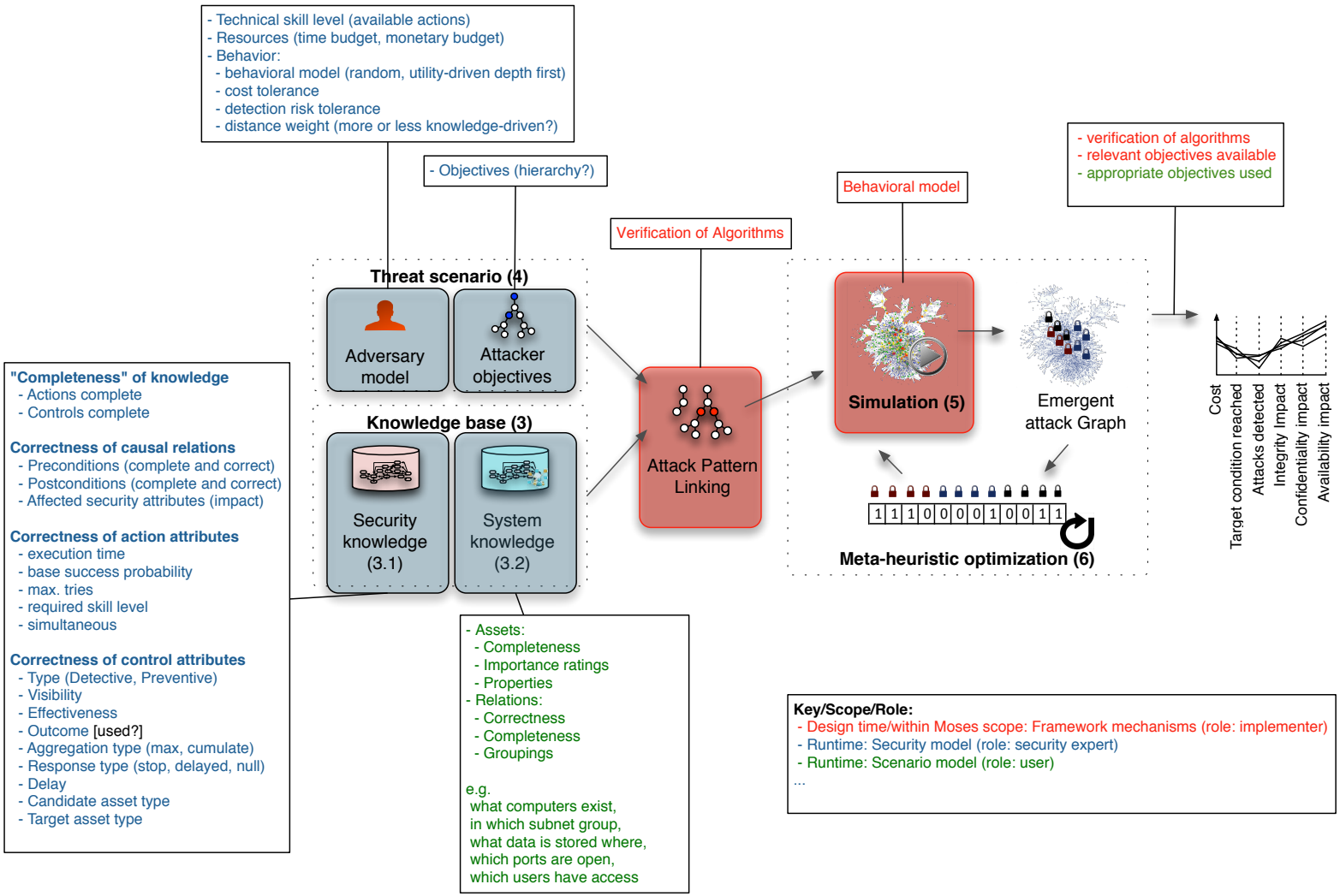
Runtime (3GHz Xeon, currently only single core used)

~ 90 mins (admin) – ~ 50 hrs (APT)

Proposed efficient solutions

- administrator: 2
- employee: 58
- unskilled external: 104
- skilled external: 306
- advanced persistent threat: 251

Validation


- Technical skill level (available actions)
- Resources (time budget, monetary budget)
- Behavior:
- behavioral model (random, utility-driven depth first)
- cost tolerance
- detection risk tolerance
- distance weight (more or less knowledge-driven?)
- Objectives (hierarchy?)
Verification of Algorithms
Behavioral model
- verification of algorithms
- relevant objectives available
- appropriate objectives used
Threat scenario (4)
Adversary model
Attacker objectives
Knowledge base (3)
Security knowledge (3.1)
System knowledge (3.2)
Attack Pattern Linking
Simulation (5)
Emergent attack Graph
1 1 1 0 0 0 0 1 0 0 1 1
Meta-heuristic optimization (6)
Cost
Target condition reached
Attacks detected
Integrity Impact
Confidentiality impact
Availability impact
"Completeness" of knowledge
- Actions complete
- Controls complete
Correctness of causal relations
- Preconditions (complete and correct)
- Postconditions (complete and correct)
- Affected security attributes (impact)
Correctness of action attributes
- execution time
- base success probability
- max. tries
- required skill level
- simultaneous
Correctness of control attributes
- Type (Detective, Preventive)
- Visibility
- Effectiveness
- Outcome [used?]
- Aggregation type (max, cumulate)
- Response type (stop, delayed, null)
- Delay
- Candidate asset type
- Target asset type
- Assets:
- Completeness
- Importance ratings
- Properties
- Relations:
- Correctness
- Completeness
- Groupings
e.g.
what computers exist,
in which subnet group,
what data is stored where,
which ports are open,
which users have access
Key/Scope/Role:
- Design time/within Moses scope: Framework mechanisms (role: implementer)
- Runtime: Security model (role: security expert)
- Runtime: Scenario model (role: user)
...

References I