FOSAD 2013 Open Session

# Security Evaluation with Adaptive Attacker Model

Leanid Krautsevich, Fabio Martinelli, Artsiom Yautsiukhin

Istituto di Informatica e Telematica, Consiglio Nazionale delle Ricerche

September 06, 2013

# Outline

- Background
  - Attack graphs and their analysis
- Motivation
- Attacker models
- Conclusions

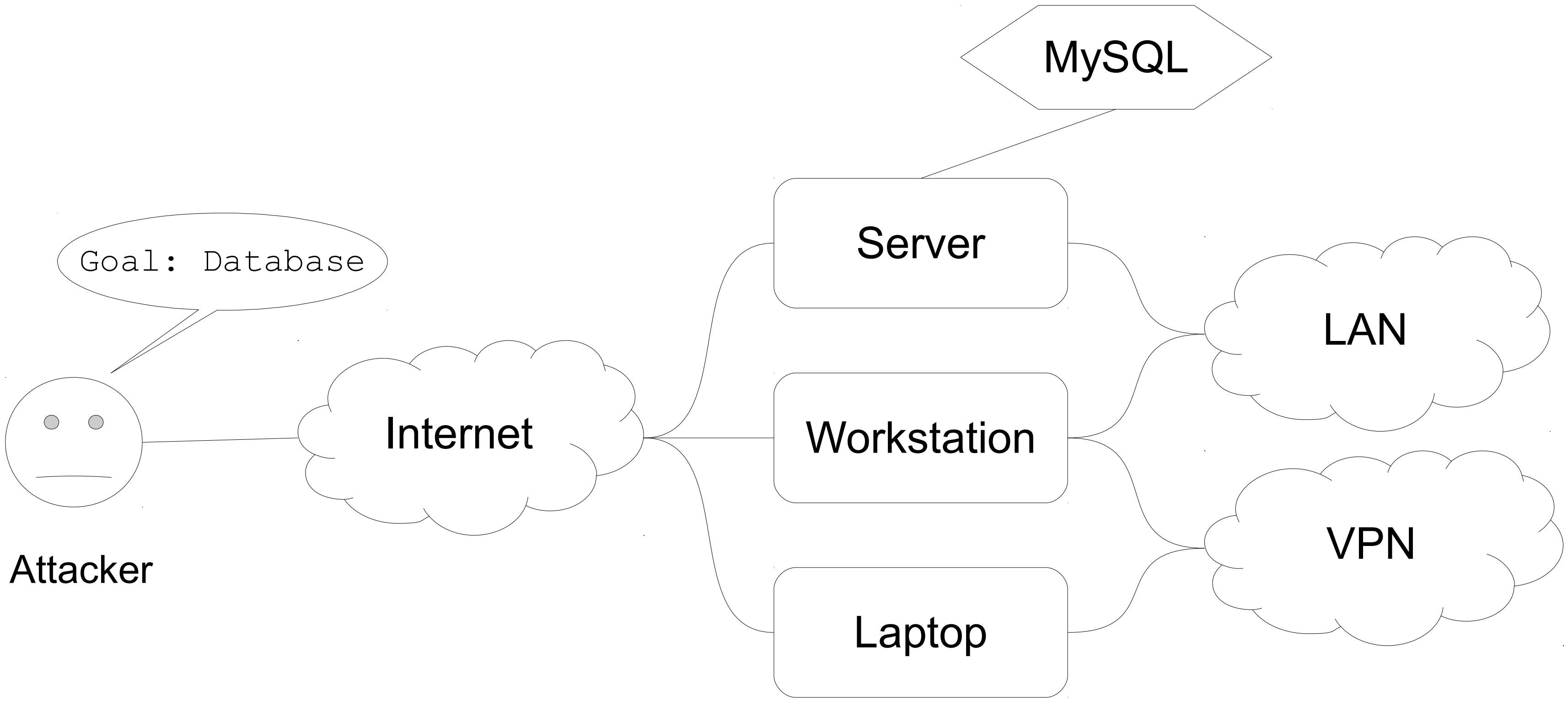
# Attack graphs

- A tool to describe security of a network
  - Contains attack paths
- Attack graph is a tuple $G = (S, A)$:
  - $S$ – the set of vertices that denote successfully executed vulnerabilities
    - $s_{init} \in S$ – the initial node
    - $S_{end}$ – the set of end nodes (subset of $S$)
  - $A$ – the set of edges that denote attempts to exploit vulnerabilities (i.e., atomic attacks)

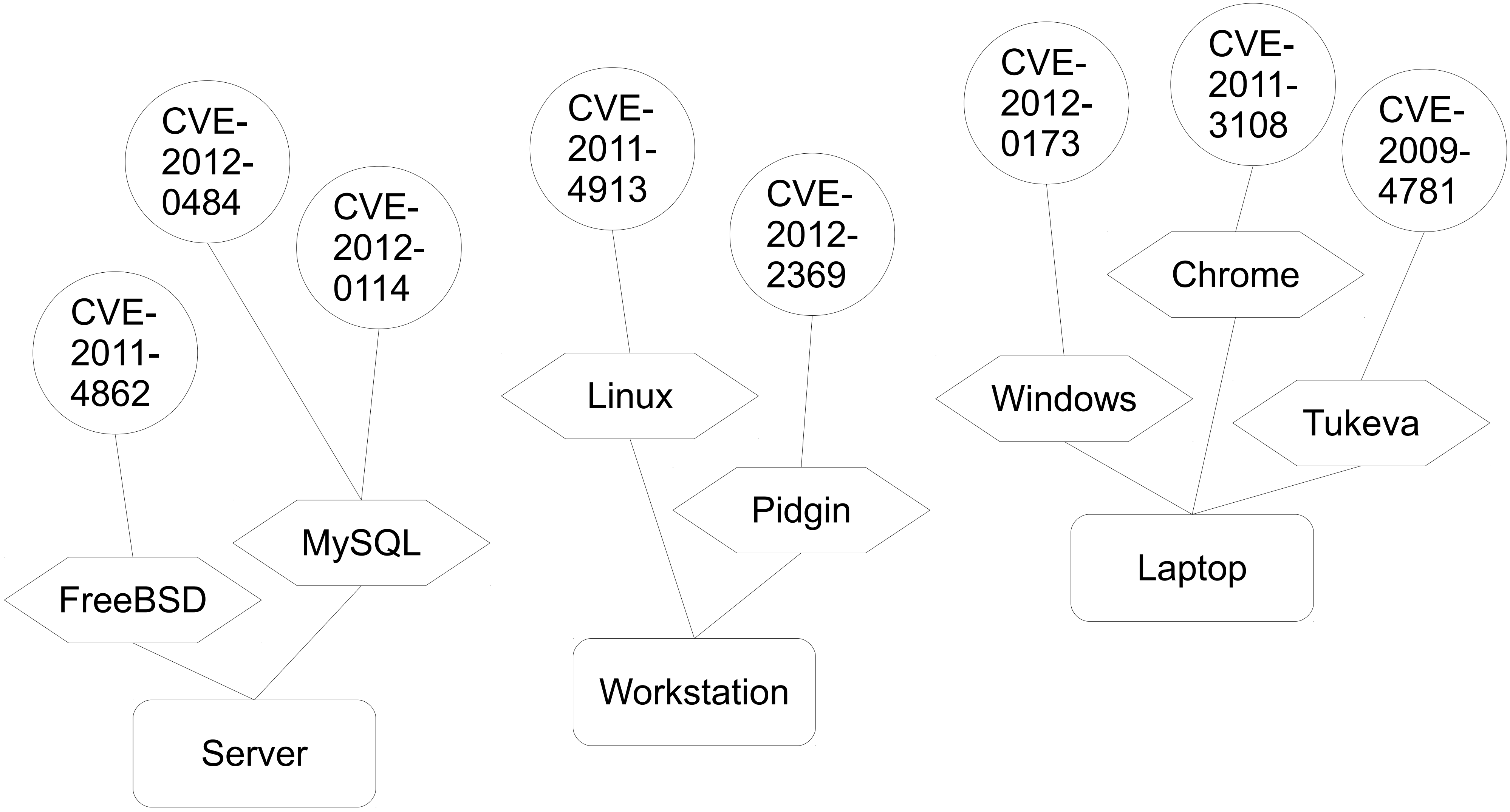
# Construction of Attack Graph

- Determine vulnerabilities of hosts
  - Manually
  - Automatically (Nessus, OpenVAS)
- Produce the attack graph
  - Use information about vulnerabilities
  - Use information about the network

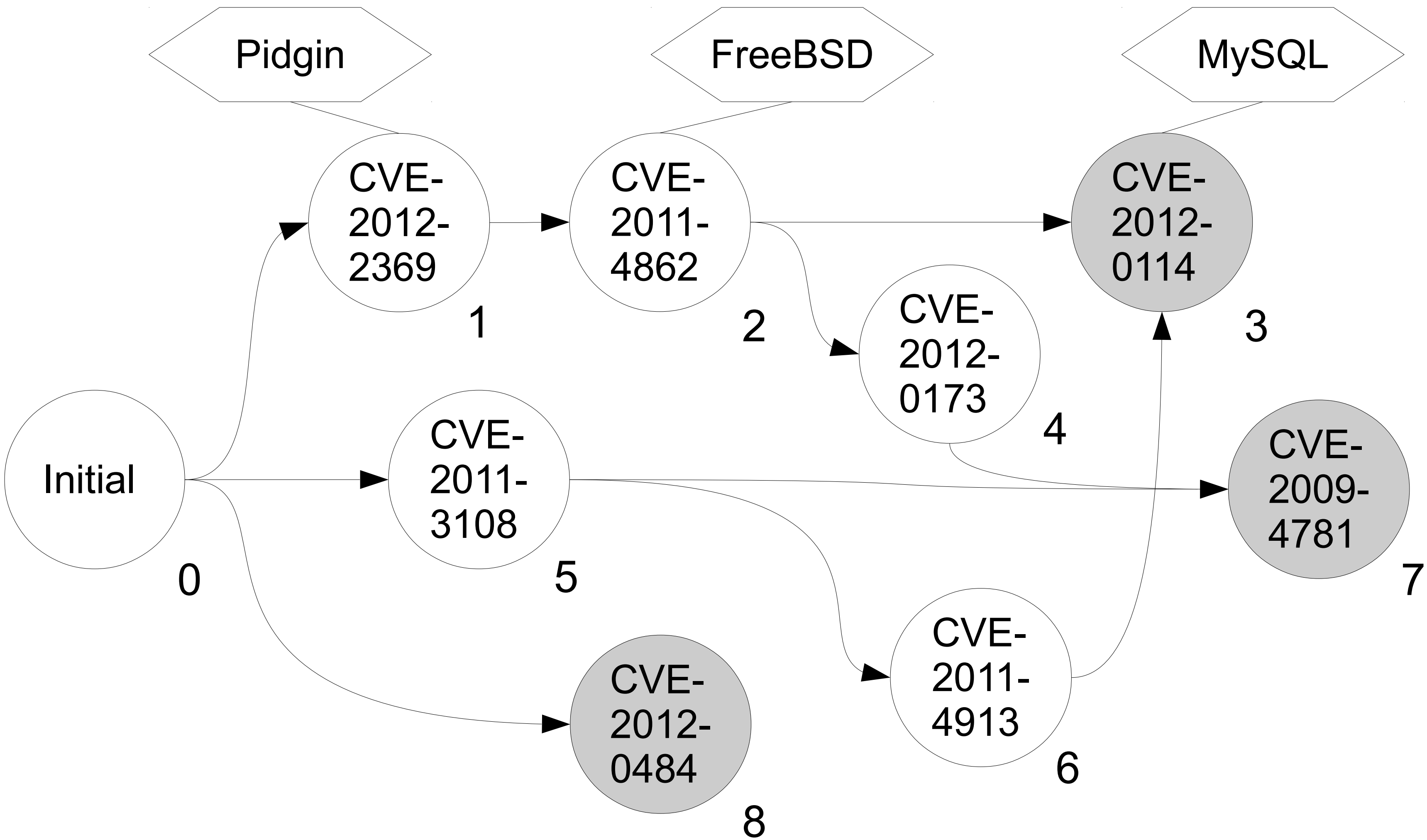
# Example: Network System



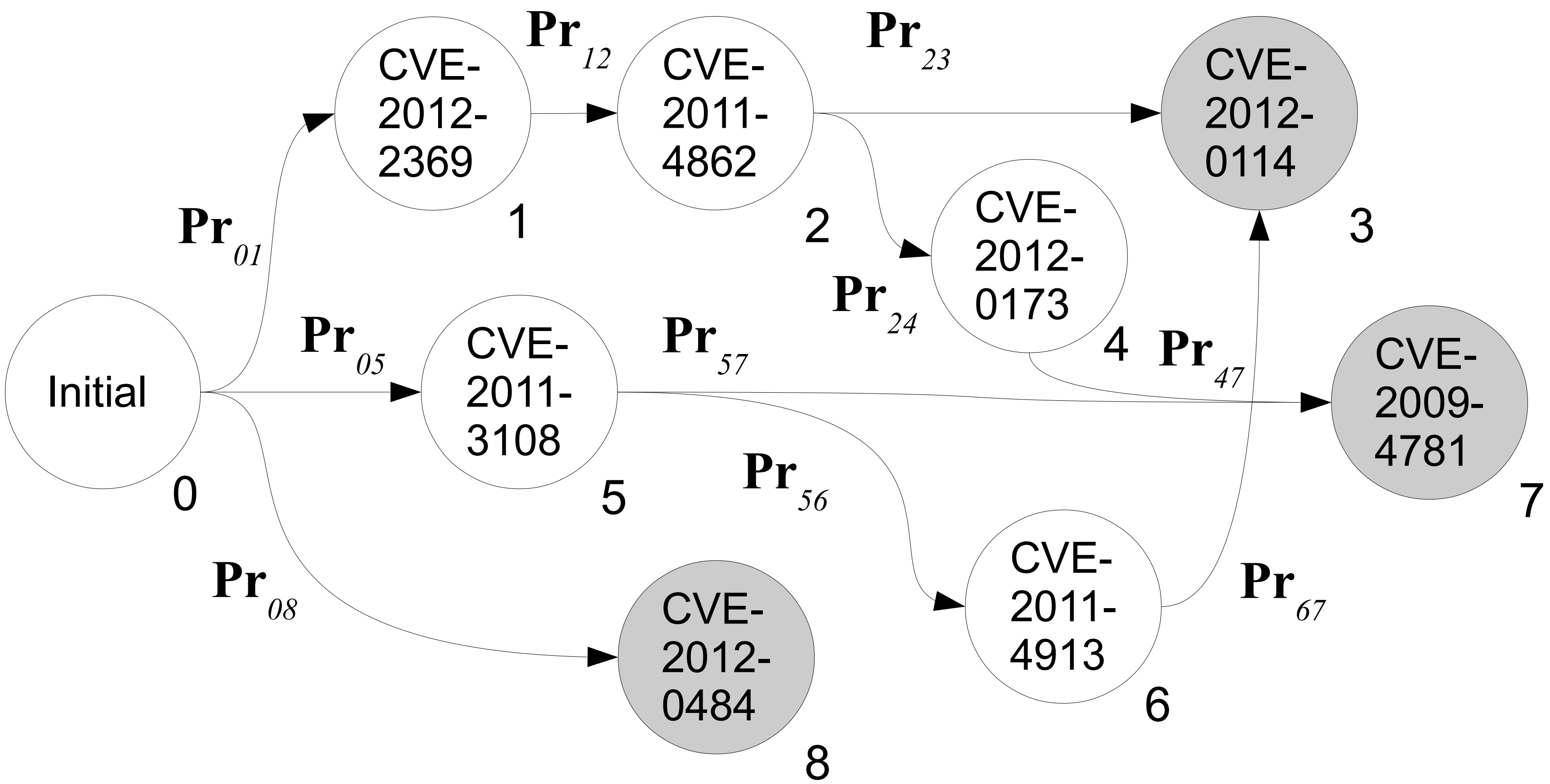
# Example: Vulnerabilities



# Example: Attack Graph



# Probabilistic Analysis (1)



# Probabilistic Analysis (2)

- Under assumption that the system follows Markov property

$$\mathbf{Pr}_{reliable} = 1 - \sum_{s \in S_{end}} \mathbf{Pr}_{steady}(s)$$

- Find the most probable attack path

# Motivation (1)

- Currently
  - Attacker is omniscient
    - Knows all vulnerabilities in the system
  - Attacker is deterministic
    - Always follows initially selected attack path

# Motivation (2)

- At the same time
  - Attacker does not know all vulnerabilities
  - Attacker studies a system during the attack
    - Finds new vulnerabilities
    - Figures out that older ones are patched
  - Attacker can change the attack path
    - When cannot complete current one

We aim at modelling ***adaptive*** attacker with ***partial knowledge*** to make evaluation of security more versatile

# Model of Attacker (1)

- Attacker is a tuple of the following values:
  - $goal$ – the goal of the attacker
  - $\Gamma$ – the set of known attacks
  - $tang$ – tangible resources possessed by the attacker (e.g., money)
  - $intang$ – intangible resources possessed by the attacker (e.g., time)
  - $skill$ – attacker's skills

# Model of Attacker (2)

- Attacker has own ***belief*** about the system

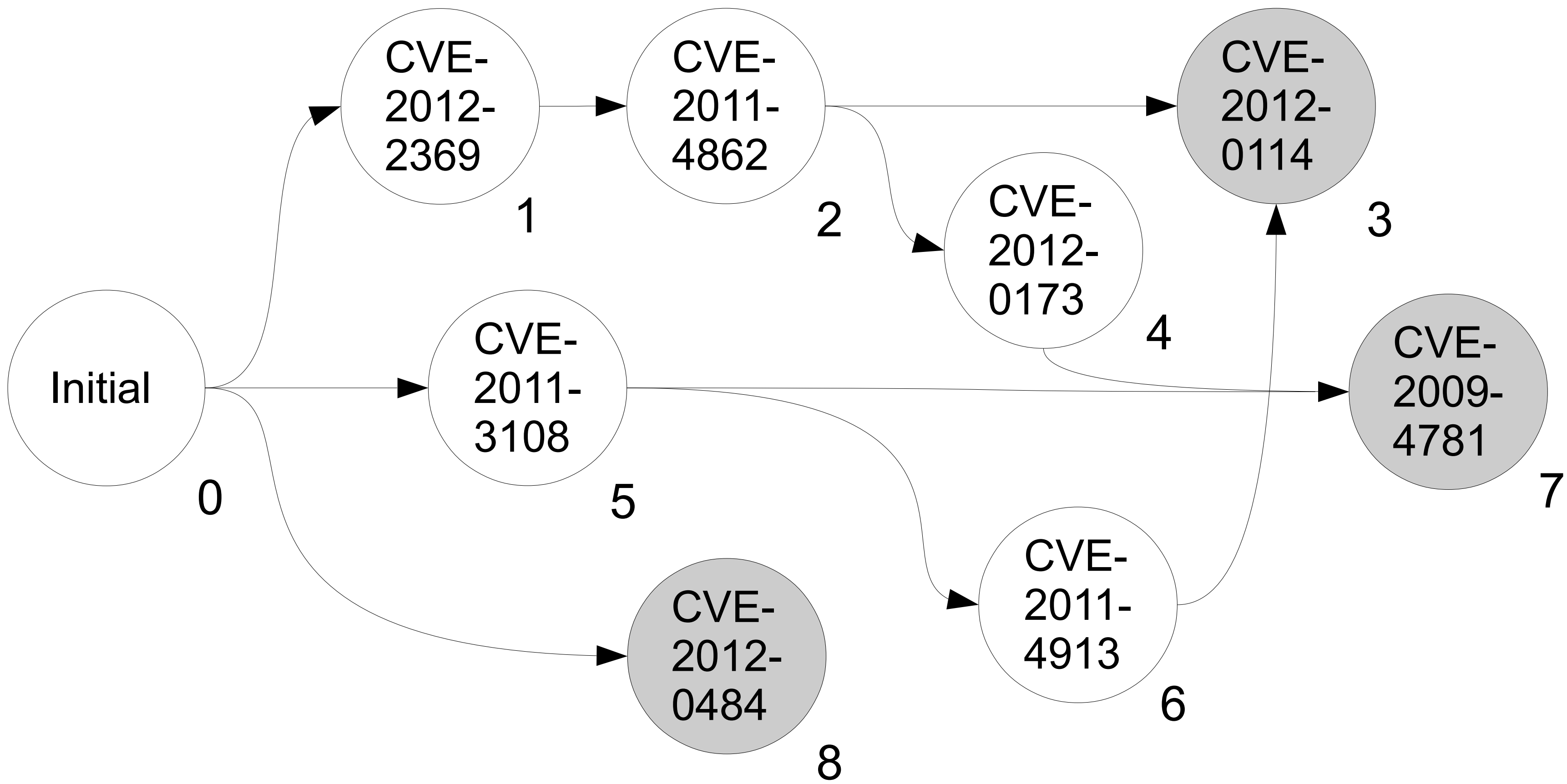
$$G_B = (S_B, A_B) : S_B = S_{true} \cup S_{false}; A_B = A_{true} \cup A_{false}$$

$$S_{true} \subseteq S; A_{true} \subseteq A$$

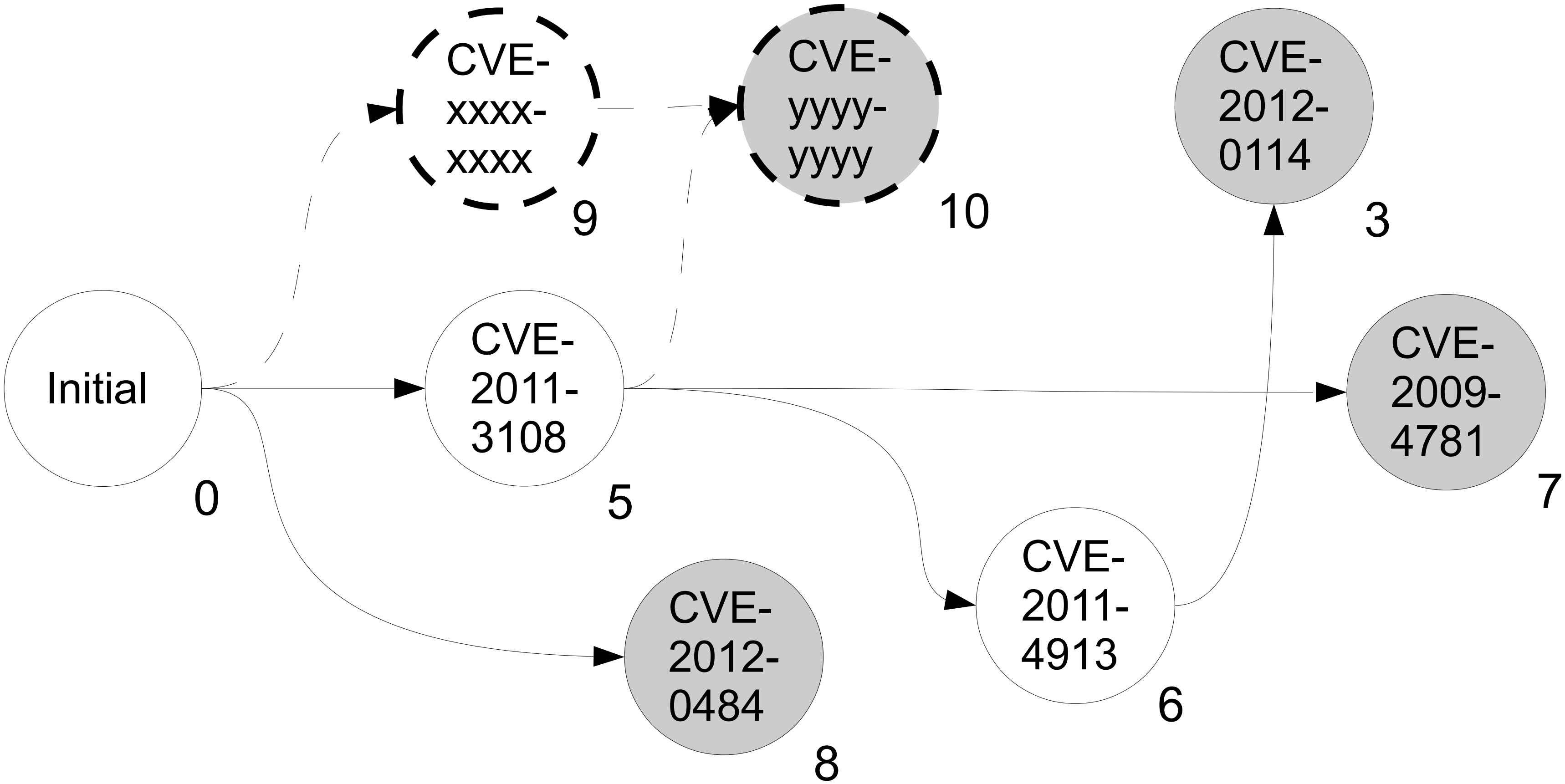
- Attacker has own ***view*** of the system

$$G_{\mathcal{X}} = (S_{\mathcal{X}}, A_{\mathcal{X}}) : S_{\mathcal{X}} \subseteq S_B; A_{\mathcal{X}} \subseteq A_B$$

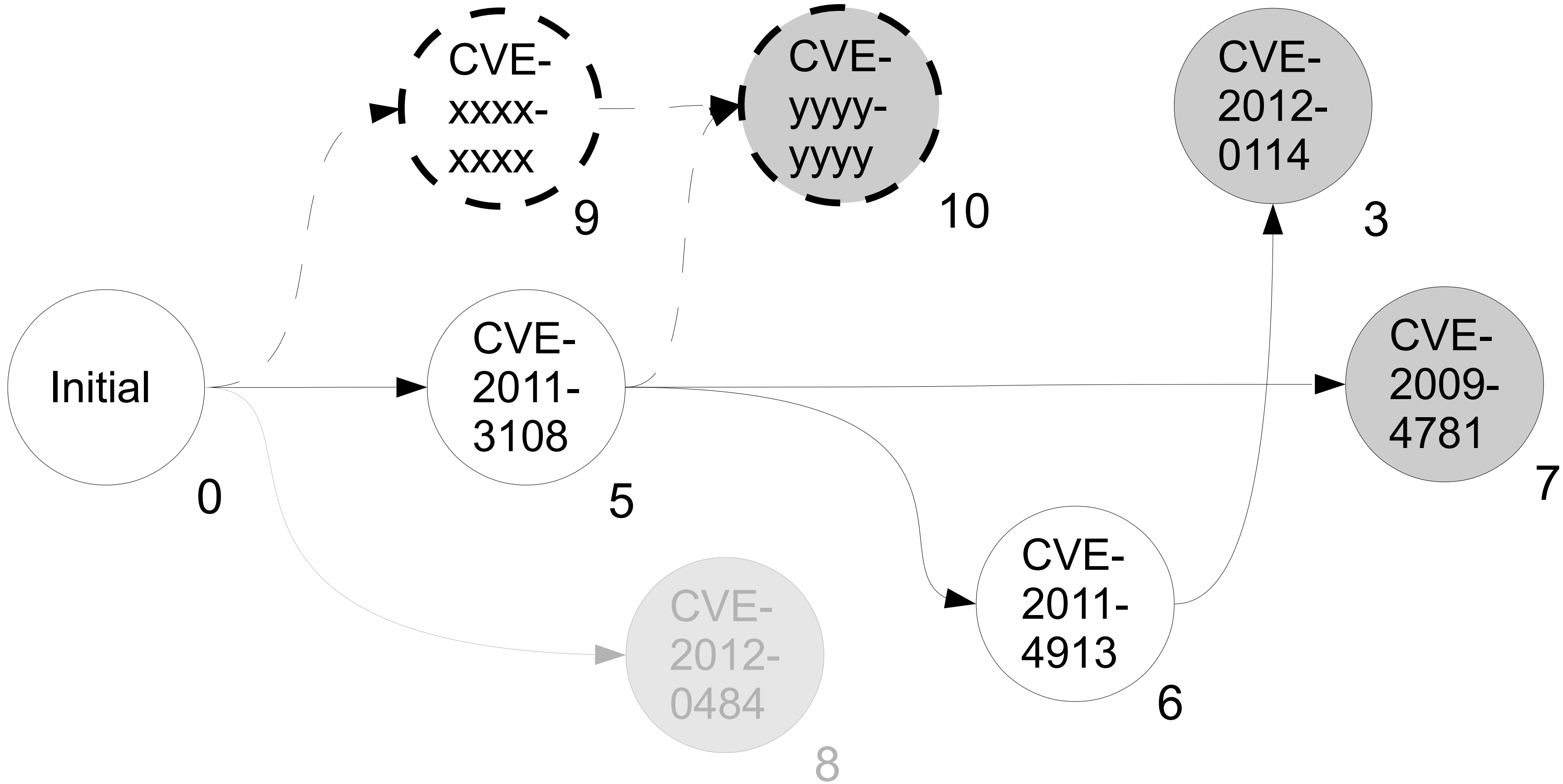
# Example: Real System



# Example: Modelling Belief



# Example: Modelling View



# Model of Attacker (3)

- The system behaves probabilistically
- For the attacker, probability of successful exploitation of a vulnerability is:

$$\mathbf{Pr}_{ij} = \mathbf{Pr}_{ij}^{p} \cdot \mathbf{Pr}_{ij}^{exp}$$

- $\mathbf{Pr}^{p}_{ij}$ is a probability that the vulnerability presents in system
- $\mathbf{Pr}^{exp}_{ij}$ is a probability to successfully exploit the vulnerability

# Model of Attacker (4)

- Probability of presence $\mathbf{Pr}^{p}_{ij}$ depends on time passed from its discovery

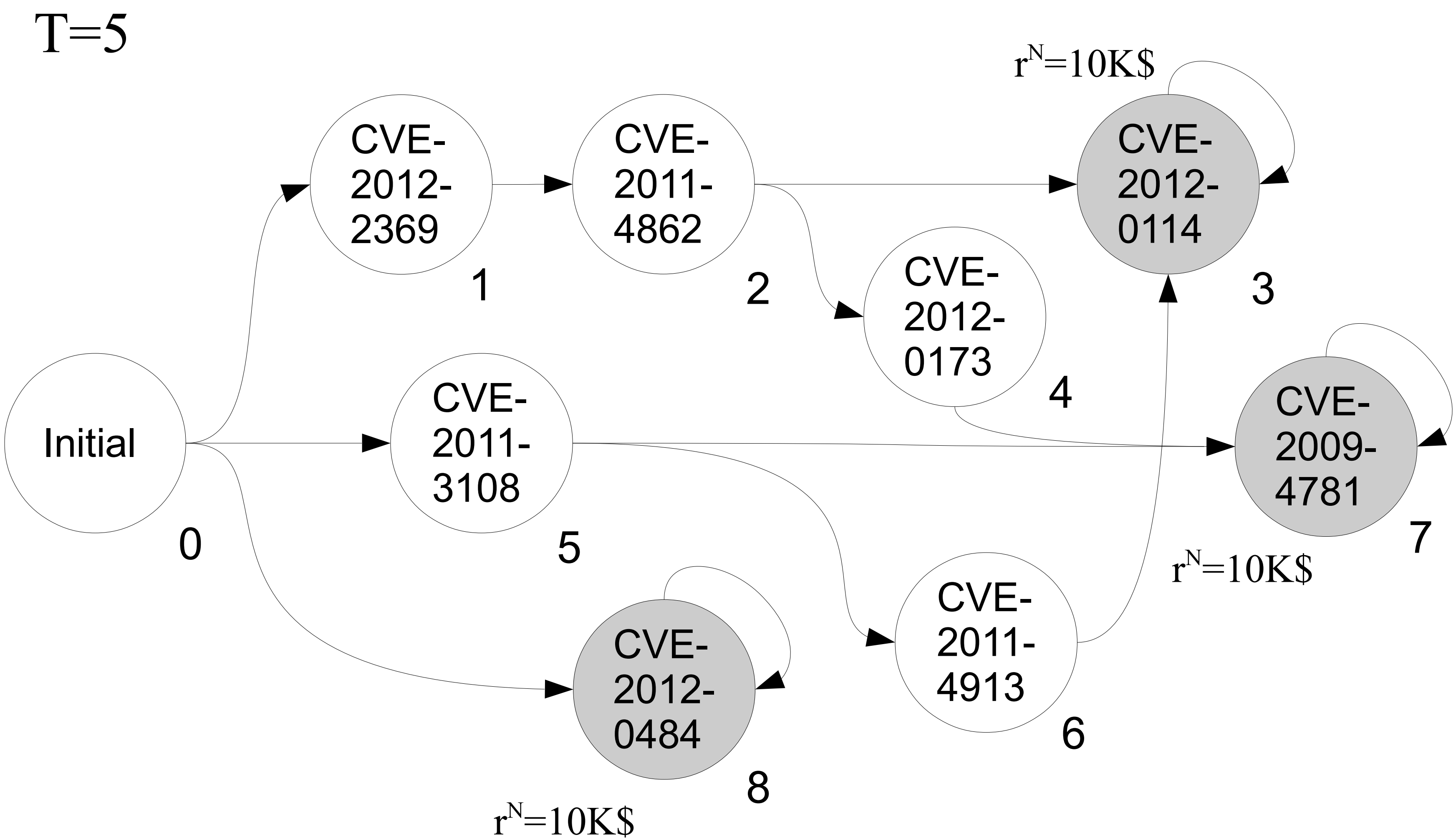
$$\mathbf{Pr}^{p}_{ij} = -\frac{1}{T_{patch}} \cdot t + 1 \qquad \text{if } T_{patch} \geq t$$
$$\mathbf{Pr}^{p}_{ij} = 0 \qquad \text{if } T_{patch} < t$$

- Probability of exploitation $\mathbf{Pr}^{exp}_{ij}$ can be evaluated on the basis of a score of the vulnerability in a vulnerability database

# Modelling Attacker's Behaviour

- Markov Decision Process:
  - $S$ – the set of system states
  - $A$ – the set of available to the attacker actions
  - $\mathbf{Pr}_{ij}$ – the set of transition probabilities
  - $T$ – the set of decision epochs
  - $R$ – the set of rewards
- Goal of a decision process – maximal total reward
  - $\pi$ – a policy

# Example: Time and Rewards

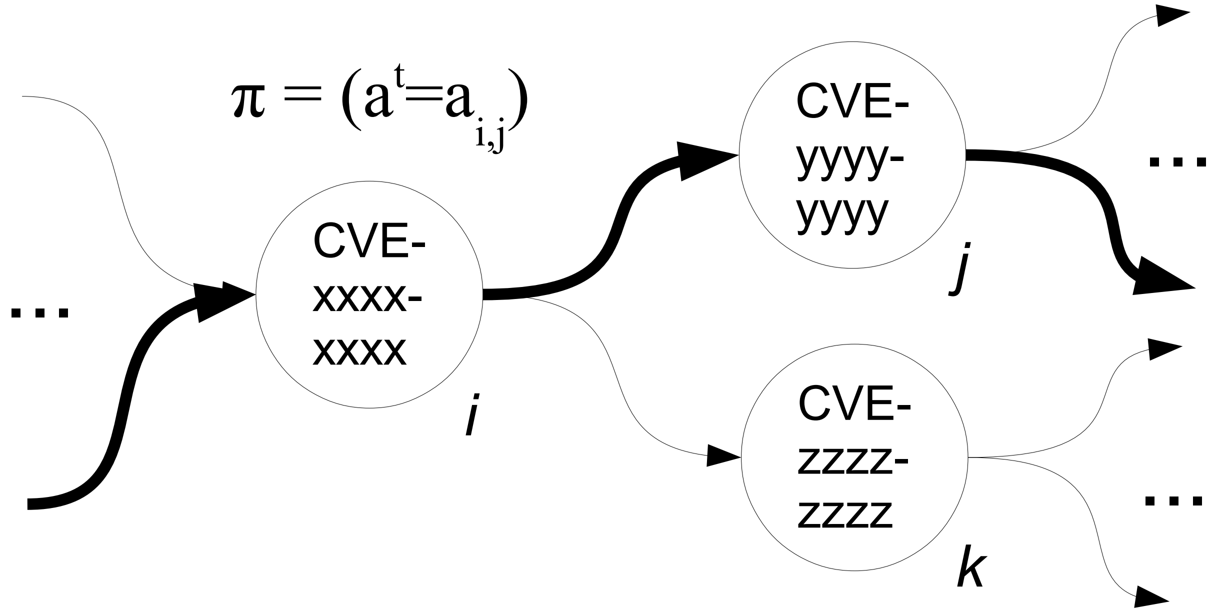


# Policies

- Consist of decision rules
  - Deterministic
  - Randomized

# Deterministic Attacker

- Policy for the deterministic attacker is computed using ***backward induction***

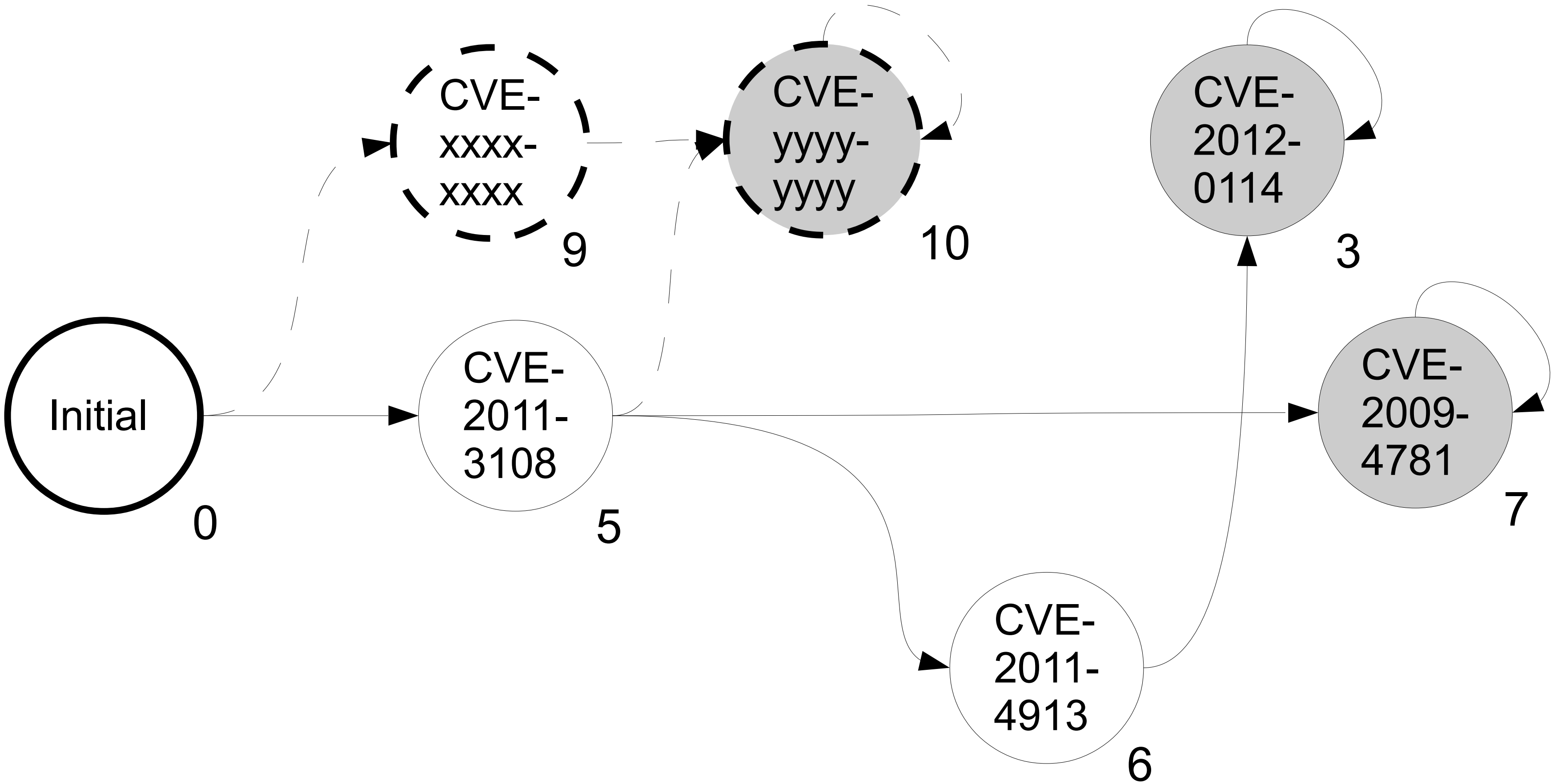


# Adaptive Attacker

- Find a set of initial deterministic policies
- After each step modify MDP depending on whether the step is successful or it is not
  - Modify $\mathbf{Pr}_{ij}$
  - Reduce the number of decision epochs
- Recompute deterministic policies

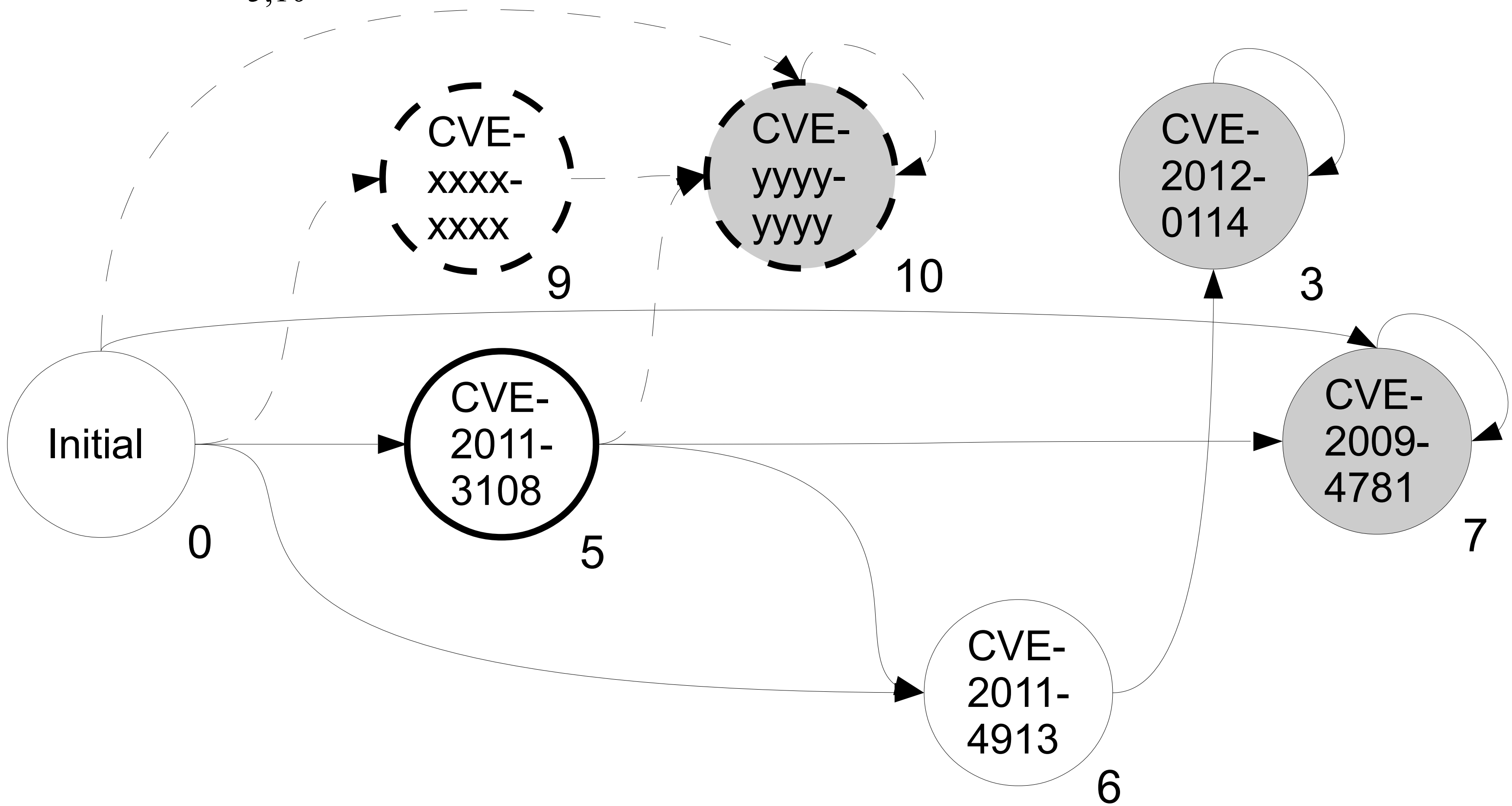
# Example: Adaptive Attacker (1)

$\pi = (a^1 = a_{0,5})$, $T=5$



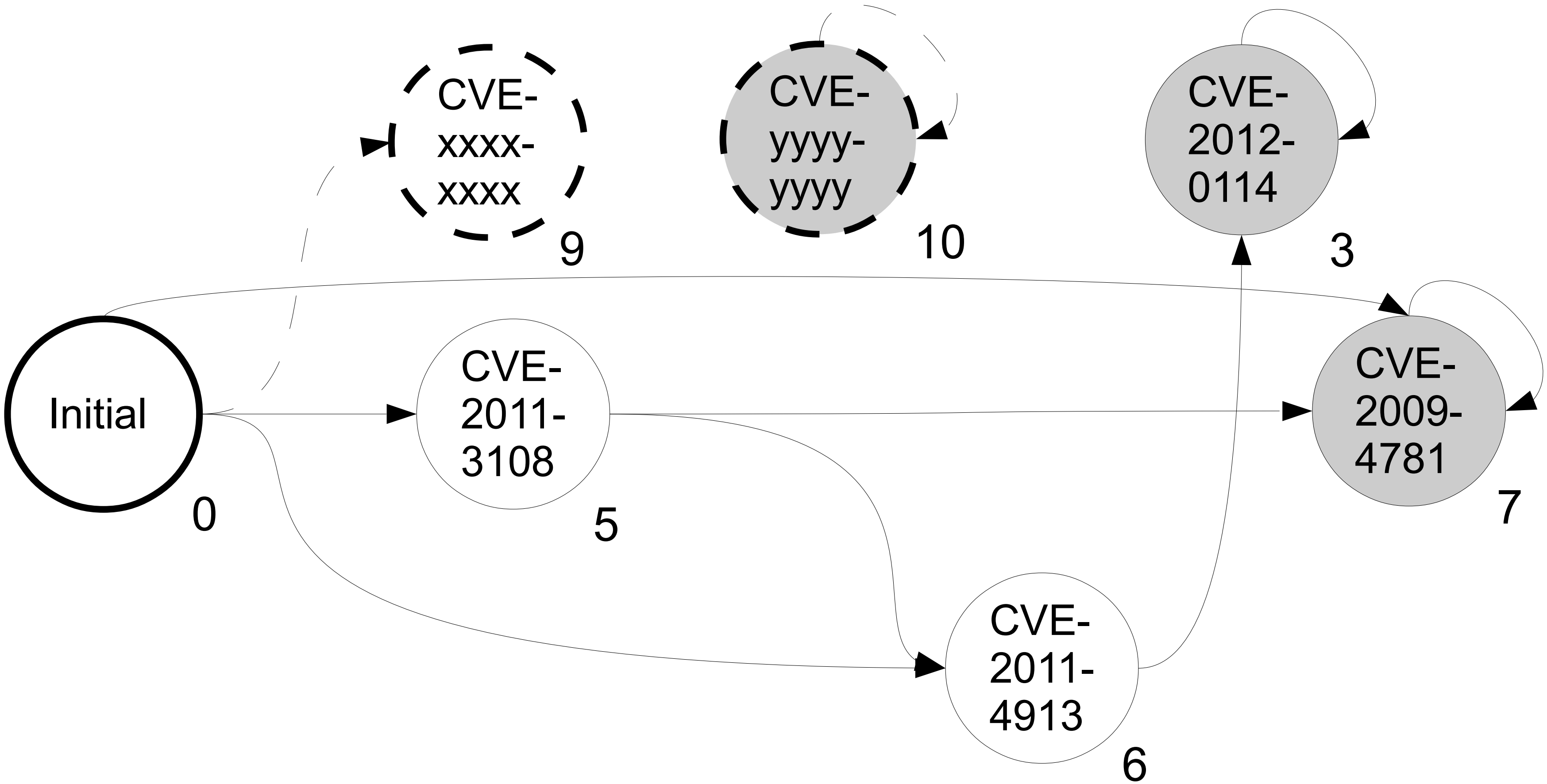
# Example: Adaptive Attacker (2)

$\pi = (a^2 = a_{5,10})$, T=4



# Example: Adaptive Attacker (3)

$\pi = (a^3 = a_{0,6})$, T=3



# Computation of Metrics

- Run several simulation
- Compute reliability as

$$\mathbf{Pr}_{reliable} = 1 - \frac{N_{sim}^{scc}}{N_{sim}^{ttl}}$$

# Conclusions

- We proposed a refined model of attacker's behaviour (published at FPS '12)
  - Attacker gains knowledge step by step
  - Attacker can reconsider her initial plan
- Future work
  - Compare the model to the usual model of attacker
  - Improve the model and introduce
    - decreasing tangible resources
    - zero-day vulnerabilities