Contextual Multi-Armed Bandits for Web Server Defense

Tobias Jung, Sylvain Martin, Damien Ernst, and Guy Leduc

Montefiore Institute

University of Liège

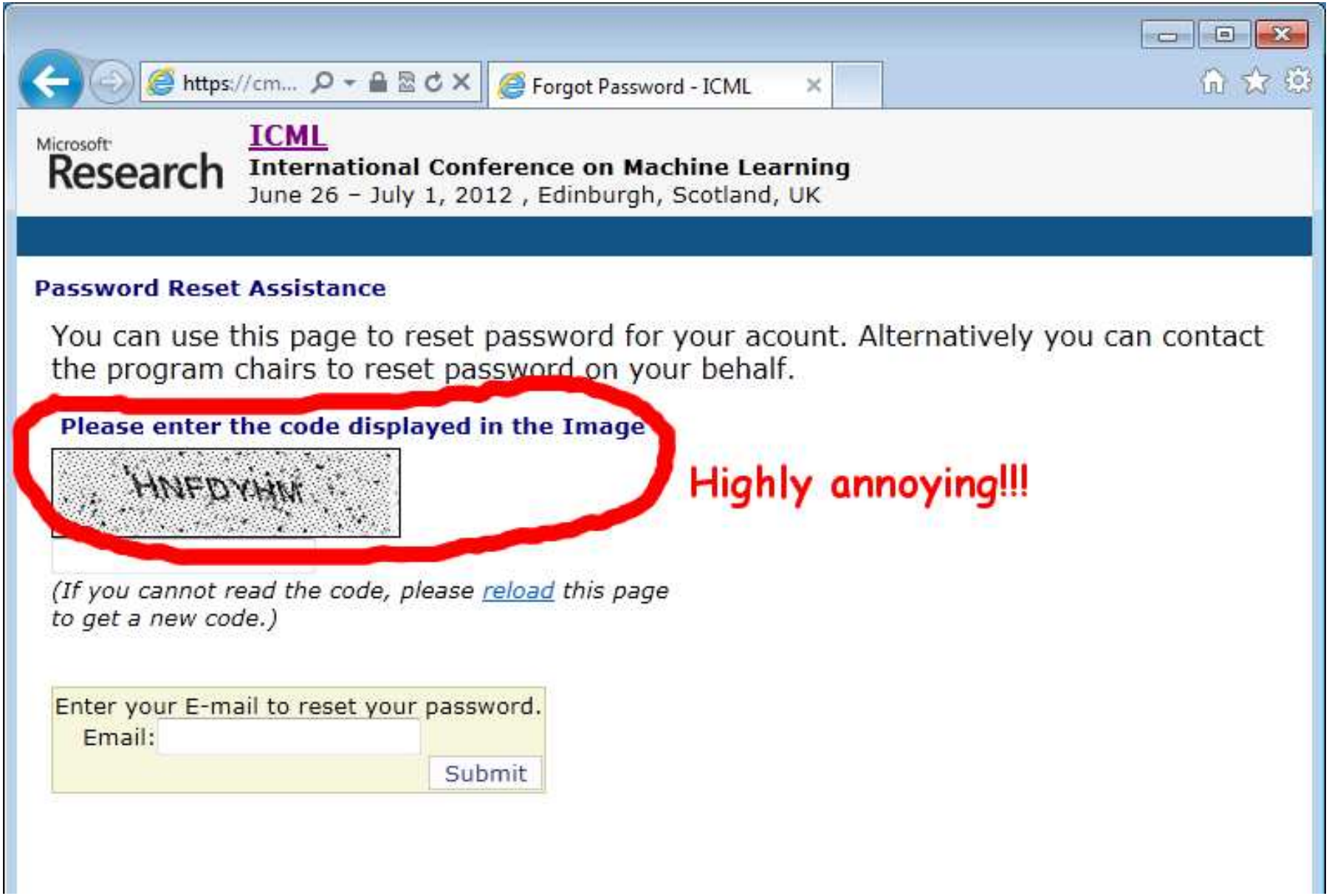
{tjung}@ulg.ac.be

Outline: Two contributions

1. **Conceptual:** autonomic web server defense as reinforcement learning task
2. **Algorithmical:** new algorithm CMABFAS (contextual multi-armed bandits for finite action spaces)

The Application

We've all seen this before ...

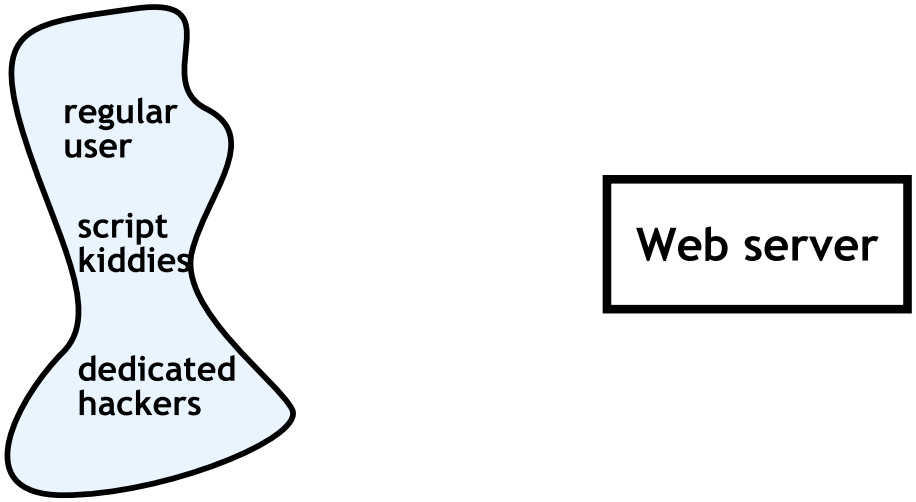


Example

Case study: detecting and preventing HTTP-based attacks on web servers

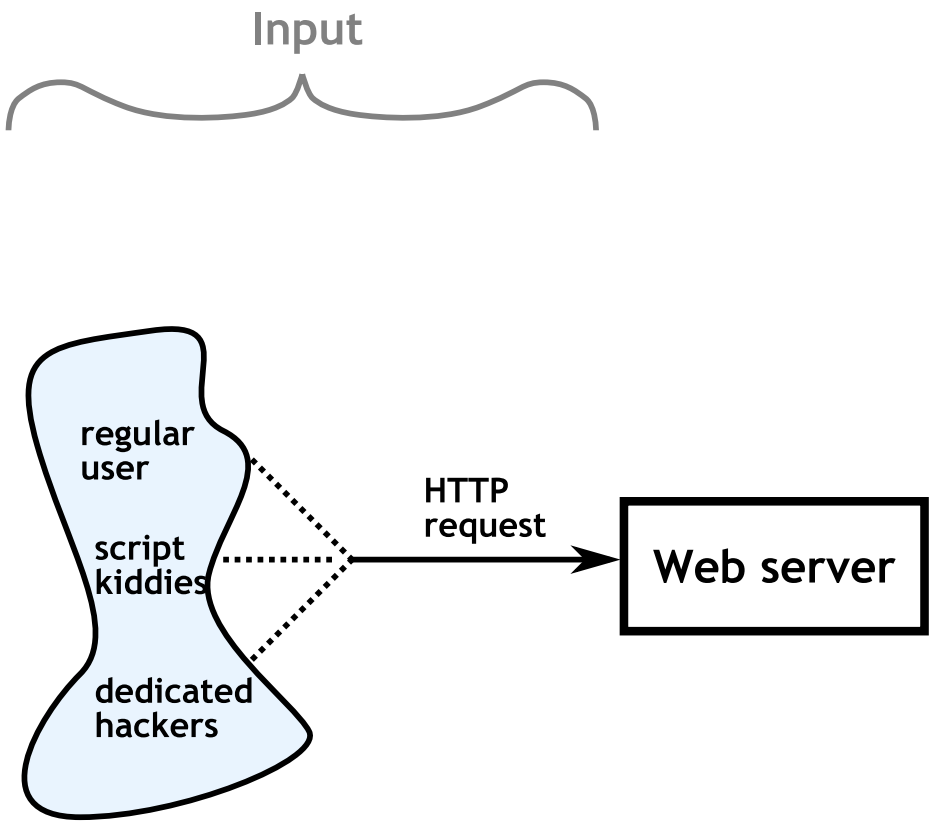
Example

Case study: detecting and preventing HTTP-based attacks on web servers



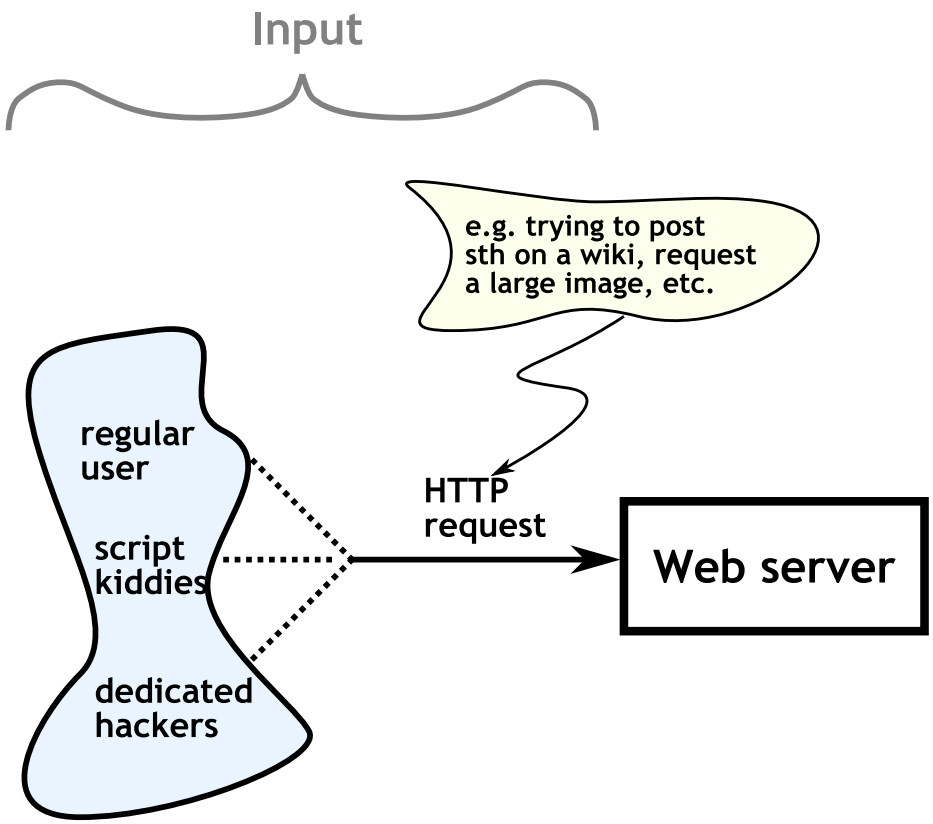
Example

Case study: detecting and preventing HTTP-based attacks on web servers



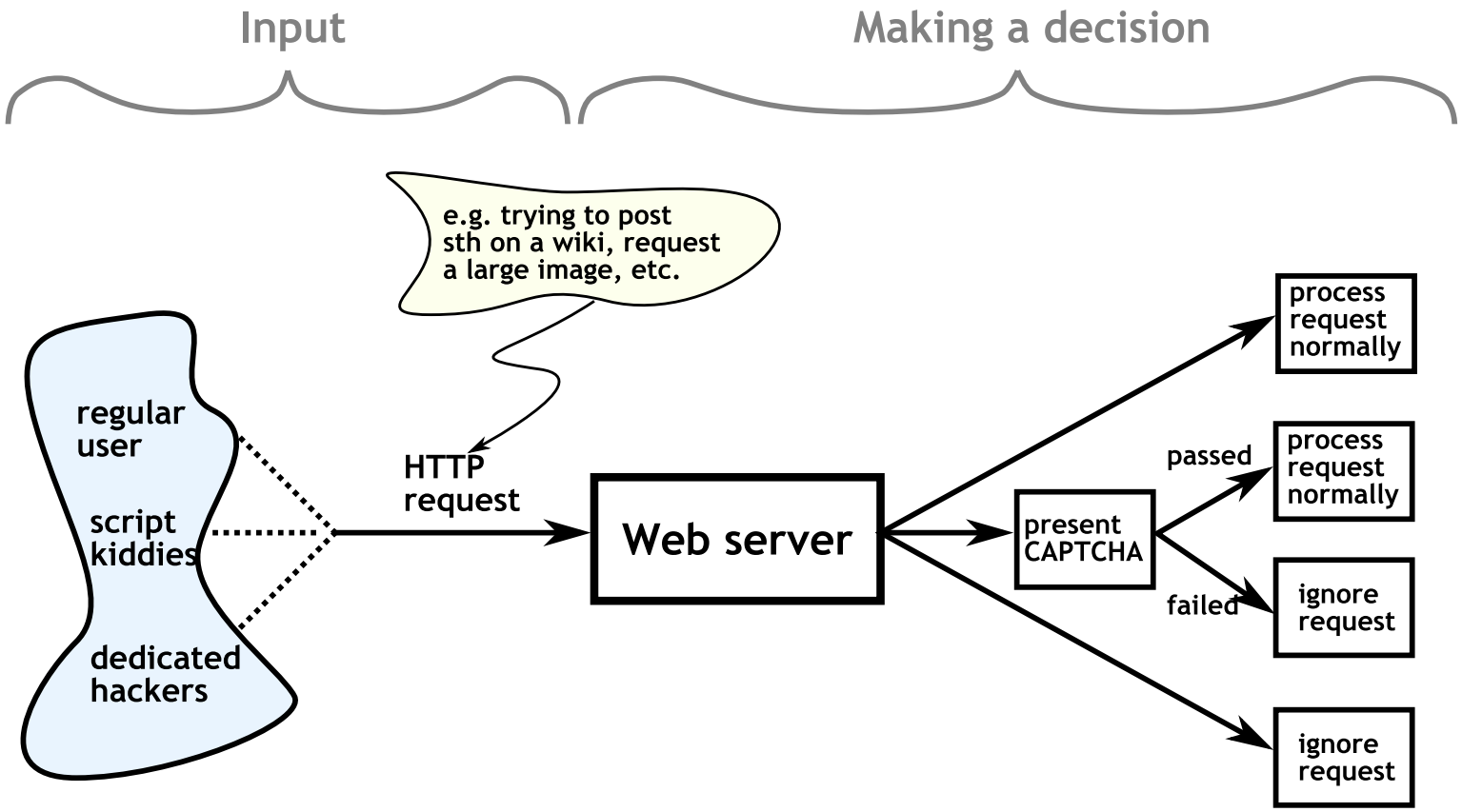
Example

Case study: detecting and preventing HTTP-based attacks on web servers



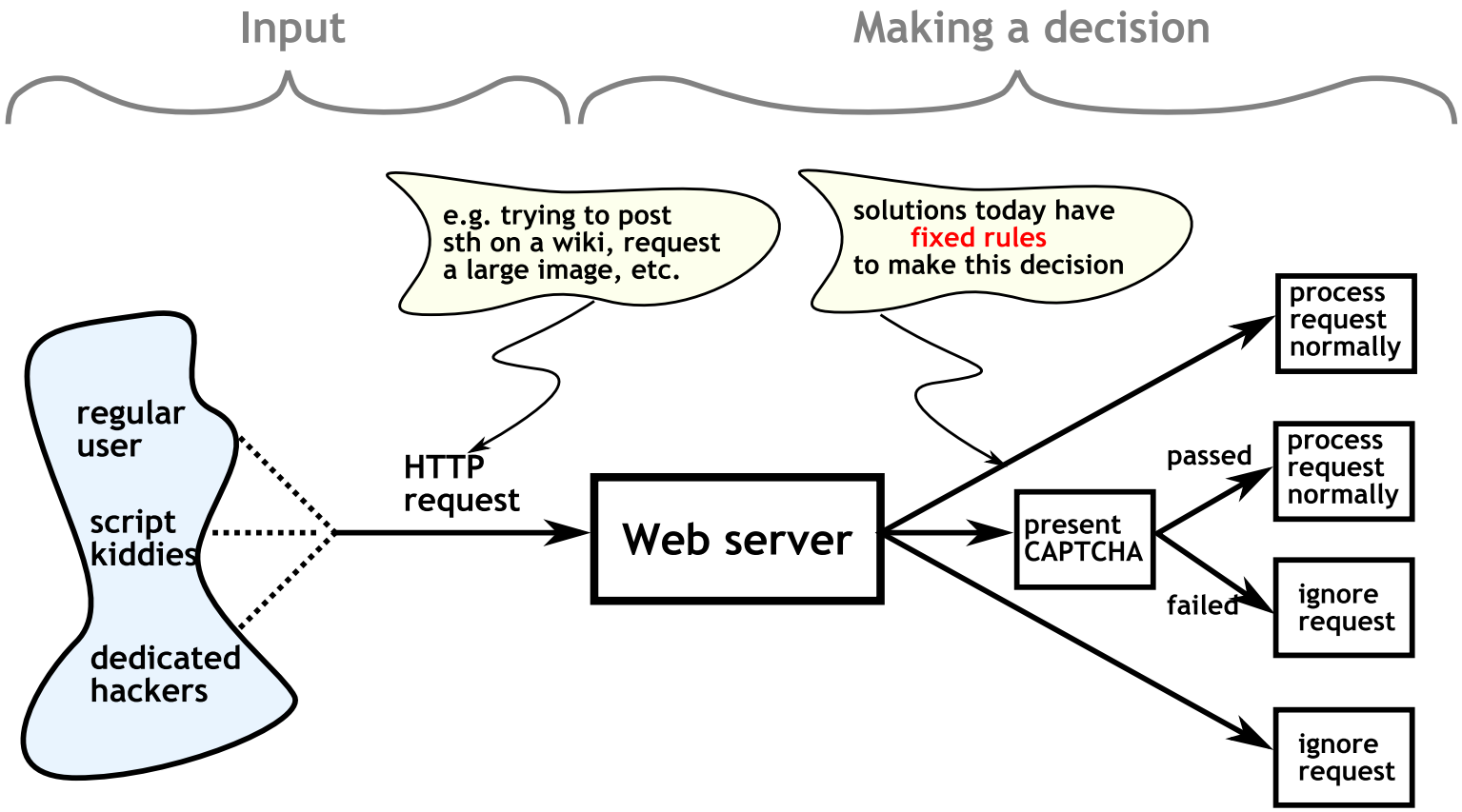
Example

Case study: detecting and preventing HTTP-based attacks on web servers



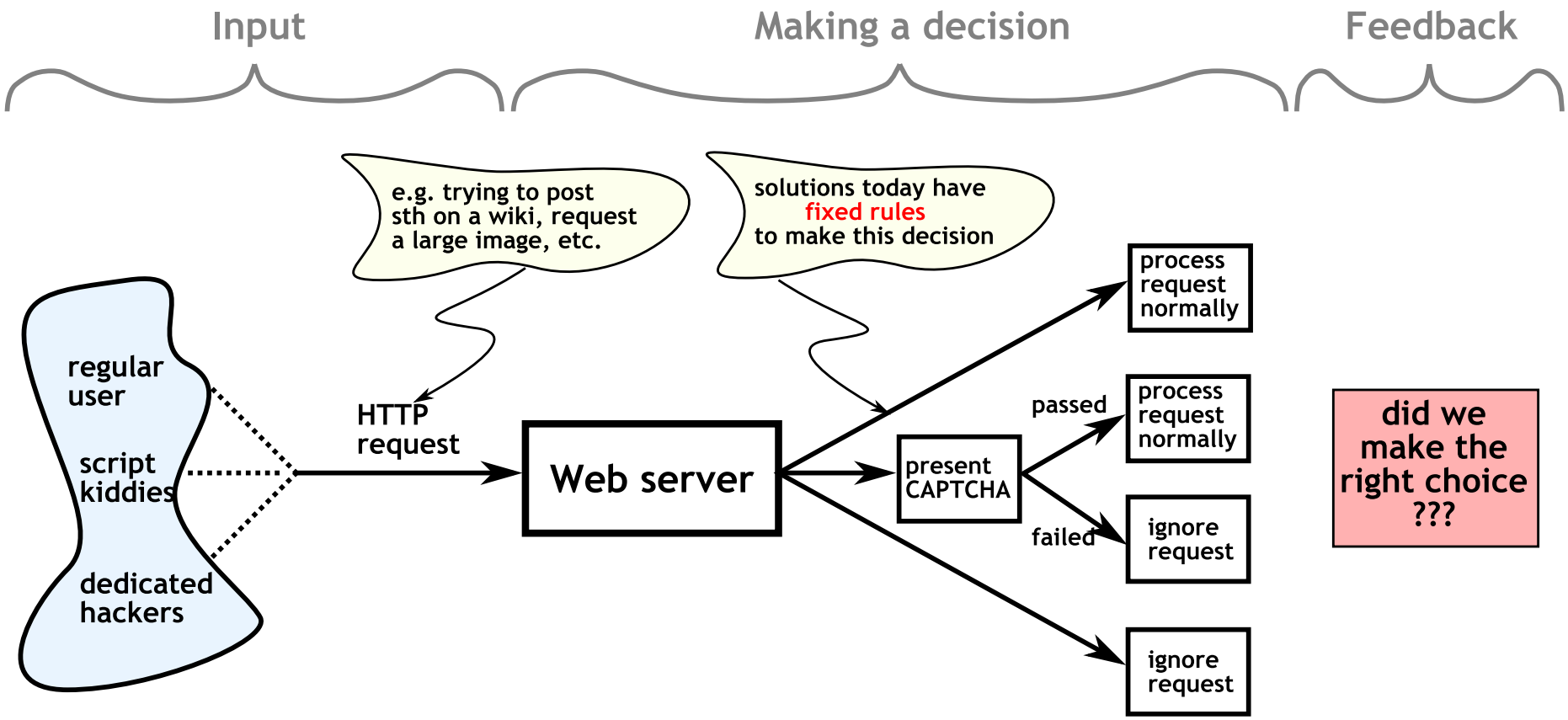
Example

Case study: detecting and preventing HTTP-based attacks on web servers



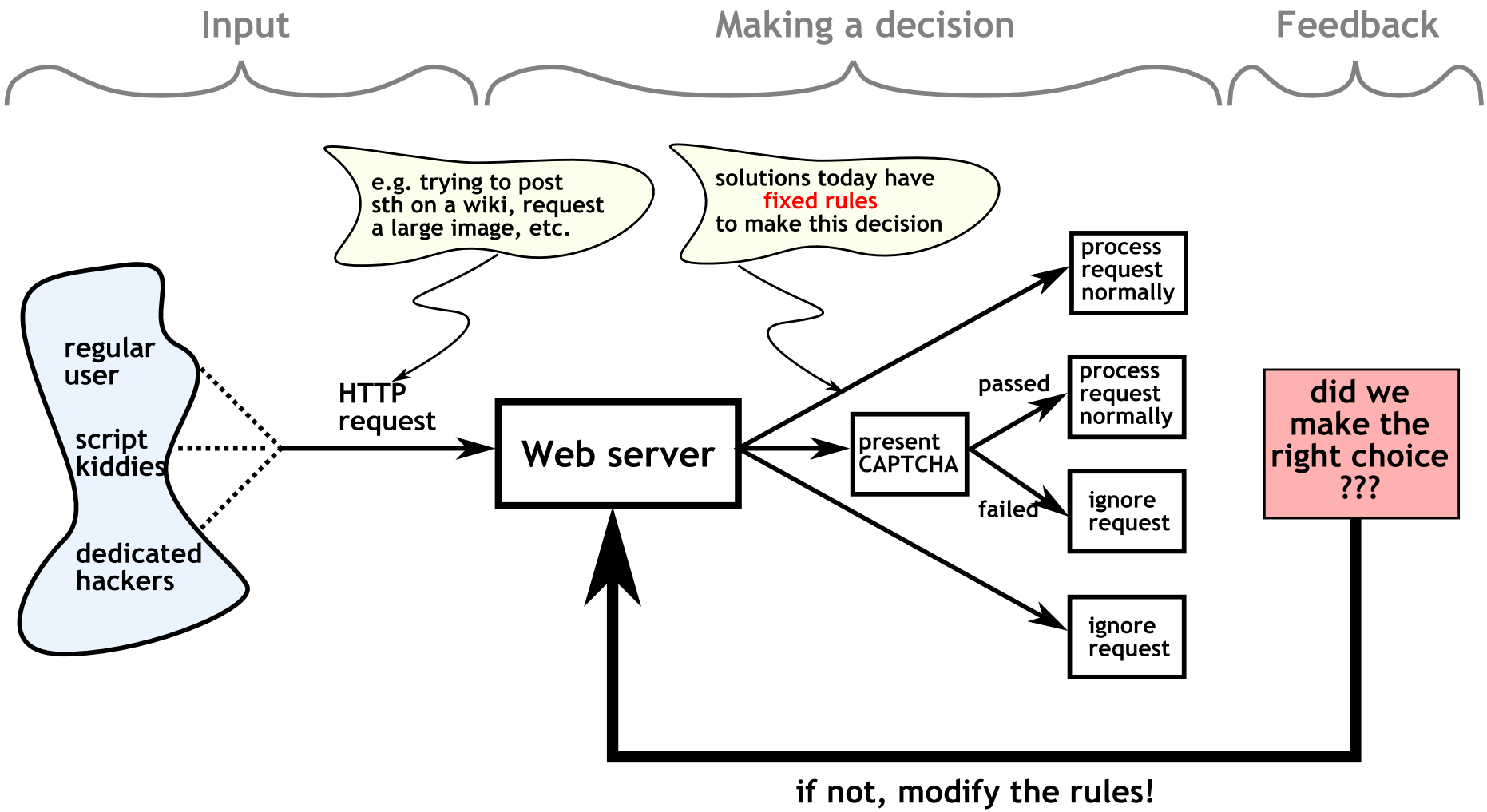
Example

Case study: detecting and preventing HTTP-based attacks on web servers



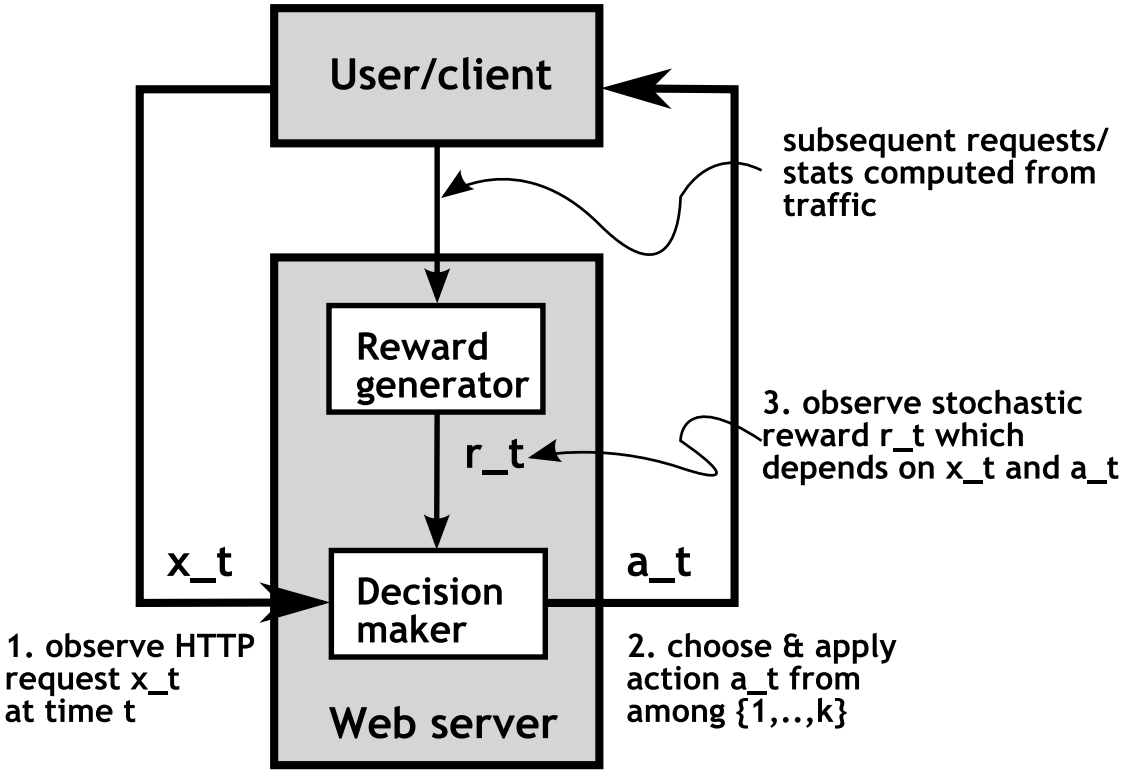
Example

Case study: detecting and preventing HTTP-based attacks on web servers



Question: how can we do this?

Answer: Reinforcement learning (RL)



Benefits:

- does not require human experts to write/update rules
- does not require correctly labeled training data
- allows the system to become self-learning
- allows the system to defend against novel attacks

Learning Algorithm: CMABFAS

Notation

$\mathcal{X}$	context space
$a \in \{1, \ldots, k\}$	possible actions for $x \in \mathcal{X}$
$\mathcal{R}^a(x)$	reward distribution for situation x and action a bounded support: $\operatorname{supp} \mathcal{R}^a(x) \subseteq [0, 1]$
$r^a(x)$	samples drawn from $\mathcal{R}^a(x)$
$\mu^a(x)$	mean of the reward distribution $\mathcal{R}^a(x)$

Goal: when presented with any $x \in \mathcal{X}$, we want to choose the action with the highest expected reward

$$\text{given } x, \text{ return } \operatorname*{argmax}_a \ \mu^a(x)$$

The catch: $\mathcal{R}^a(x)$ and $\mu^a(x)$ is not known!!!

Solution: estimate $\mu^a(x)$ from samples => exploration vs. exploitation

MAB vs contextual MAB

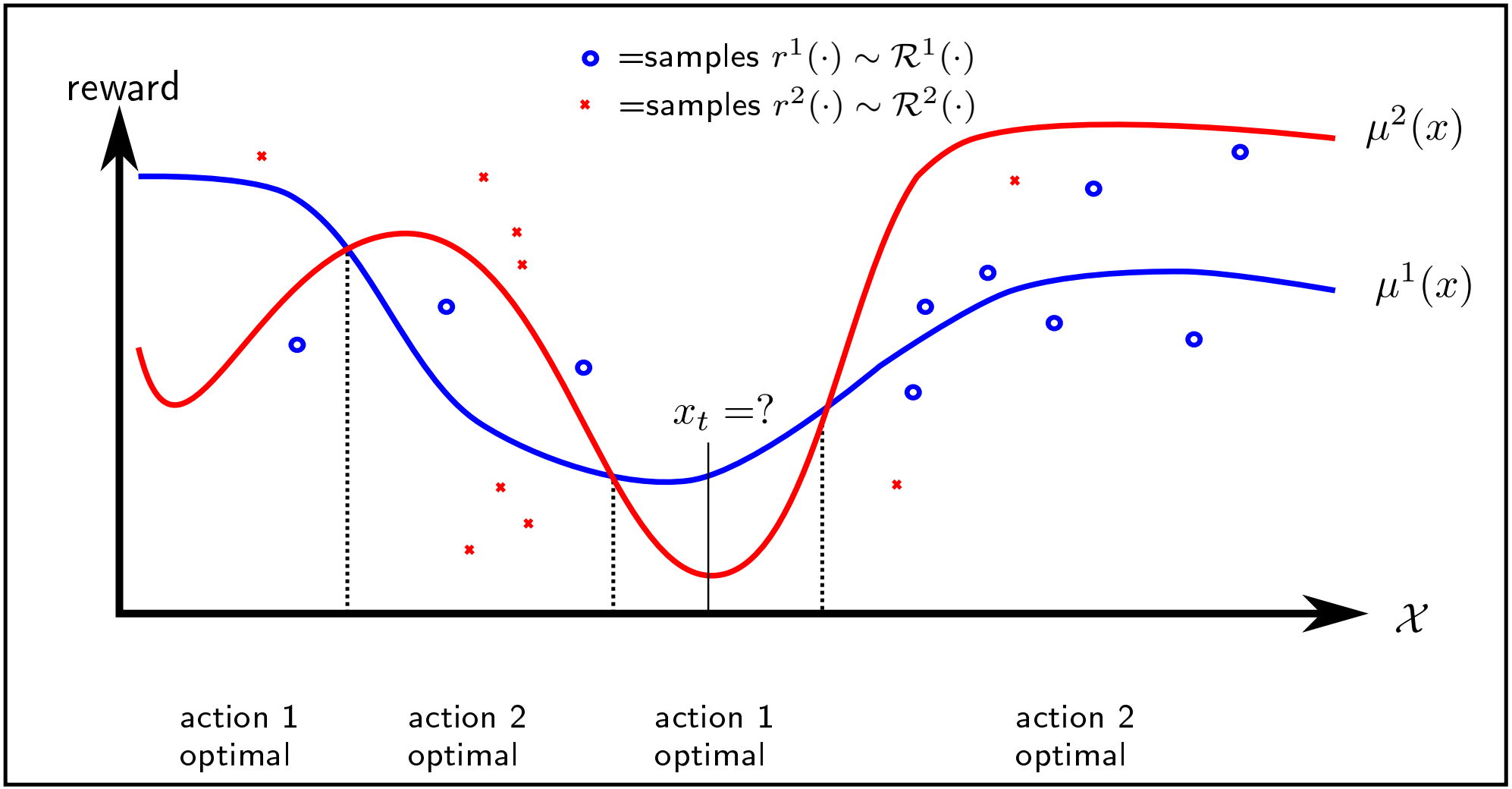
- So far, this looks a lot like a traditional k-armed bandit.
- The main difference is: in k-armed bandits the reward does not depend on x.
- Learning in contextual MAB is significantly more difficult since we have to aggregate samples over $\mathcal{X}$. These samples are no longer iid samples (different base distributions).
- In order to make learning possible, we need to make some additional structural assumptions:

  The expected reward is smooth over $\mathcal{X}$

- **Formally:** let $\mathcal{X}$ be equipped with pseudo-metric $d(x, x')$ (with $\sup_{x,x'} \; d(x, x') = 1$). We then assume that each $\mu^a(\cdot)$ is Lipschitz:

$$|\mu^a(x) - \mu^a(x')| \leq \lambda \cdot d(x, x') \quad \forall x, x' \in \mathcal{X}, \forall a$$

Illustration (2 actions)

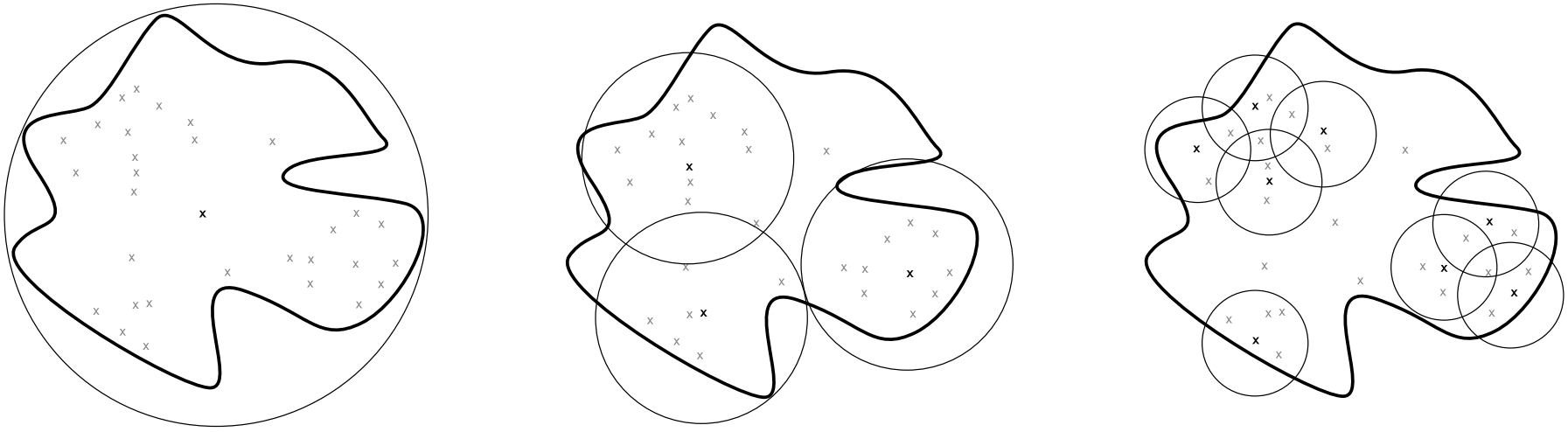


CMABFAS–Overview

- For each action a separately, we incrementally construct over time $t = 1, 2, \ldots$ a cover of $\mathcal{X}$.
- The cover consists of ball-shaped regions where individual balls are centered on some of the $\{x_1, \ldots, x_{t-1}\}$ seen so far.
- The cover is hierarchical with the radius of the balls decreasing as $1, \frac{1}{2}, \frac{1}{4}, \ldots$.
- Each ball aggregates the reward samples lying within.
- Each ball covering x_t can be used to upper-bound $\mu^a(x_t)$ and produces a score which consists of
  1. sample average within the ball (exploitation)
  2. how many samples are in the ball (exploration)
  3. how large is the ball (exploration)
- The best ball *for each action* is then the one with the lowest score (tightest upper bound).
- The best action overall is the highest such score.

CMABFAS–Overview 2

Context space (x=sample locations)



Level 1 (radius=1) Level 2 (radius=1/2) Level 3 (radius=1/4)

Adaptive refinement: New balls are created according to the following rules:

- A new ball is created centered at (x_t, a_t) at 1/2 radius of the parent (where a_t is the action chosen at time t).
- A new ball can only be created if the number of samples in the parent ball exceeds a certain threshold (i.e., the ball is full).
- A new ball is only created if it will not overlap with already existing balls at the same level of the hierarchy.

CMABFAS–Algorithm

- For $t = 1, 2, \ldots$
  - Get context x_t.
  - For each action $a = 1 \ldots k$
    - Determine set of **active balls** $A^a(x_t)$
      (=balls which cover x_t)
    - Determine set of **relevant balls** $R^a(x_t)$
      (=active balls which are either not full, or are full and can produce a child)
    - Determine score

$$u(x_t, a) := \min_{B \in R^a(x_t)} \Big[\underbrace{\frac{\varrho_t(B)}{n_t(B)}}_{\substack{\text{avg rew} \\ \text{in ball B}}} + \underbrace{c \cdot \sqrt{\log T / n_t(B)}}_{\substack{\text{uncertainty} \\ \text{due to} \\ \text{\#samples in B}}} + \underbrace{2 \cdot \lambda \cdot r(B)}_{\substack{\text{uncertainty} \\ \text{due to} \\ \text{size of B}}} \Big]$$

  - Perform best action $a_t := \text{argmax}_{a \in 1,\ldots,k} \; u(x_t, a)$
  - Observe reward $r_t \sim \mathcal{R}^{a_t}(x_t)$ and update counters.
  - **Adaptive refinement:** Add new ball at (x_t, a_t) with 1/2 radius if winning ball is full
    (and thus allows a child to be created)

CMABFAS–Properties

Analysis: CMABFAS works because

- the score $u(x_t, a)$ is a high-probability bound for $\mu^a(x_t)$ (Azuma-Hoeffding for martingales with bounded increments)
- the radius of the "winning" ball at step t can thus be used to upper-bound the loss at step t (smaller balls, better accuracy)
- the adaptive refinement rules ensure that the balls get smaller, but only in those regions of $\mathcal{X}$ where the corresponding action is optimal.
- the number of times large balls are "winning" balls can be upper-bound by the r-packing number of $\mathcal{X}$ (and the *near optimality dimension* of the problem, see [Slivkins, 2009; Bubeck et al. 2008; Munos 2011]) $\Longrightarrow$ regret bounds!

Implementation: From a practical point of view CMABFAS has two nice properties:

- it is a cheap algorithm with little computational cost and low storage requirements
- it is an anytime algorithm

Experiments

Web server defense scenario

Our current modeling of the web server defense scenario as contextual MAB looks like this:

$\mathcal{X}$ attributes extracted from the initial HTTP request of a session
(essentially a vector of text strings)

$d(x, x')$ comparing two text strings (for details see paper)

a {action #1,action #2, action #3}
- action #1: apply no security measure
- action #2: apply security measure Type-1 (abstract action)
- action #3: apply security measure Type-2 (abstract action)

$r(x, a)$
- each HTTP session was manually labeled and assigned to one of 9 classes
- each class was manually assigned a reward profile (Bernoulli distribution)
- WIP: originally we envisioned an automated procedure to internally generate rewards

Note: Unlike in classification or anomaly detection, we cannot work with training data to verify our approach. Due to the interactive nature the system choosing actions must be able to observe how the environments responds. This is very difficult to simulate.

Experimental protocol

Data: obtained from a real, locally hosted web server

- 57,389 HTTP requests grouped into 1126 sessions (logged between Jan-Dec 2011)
- mostly regular traffic, but also
  - scans for vulnerable versions of installed packages
  - remote code injection attempts
  - spam on public wiki

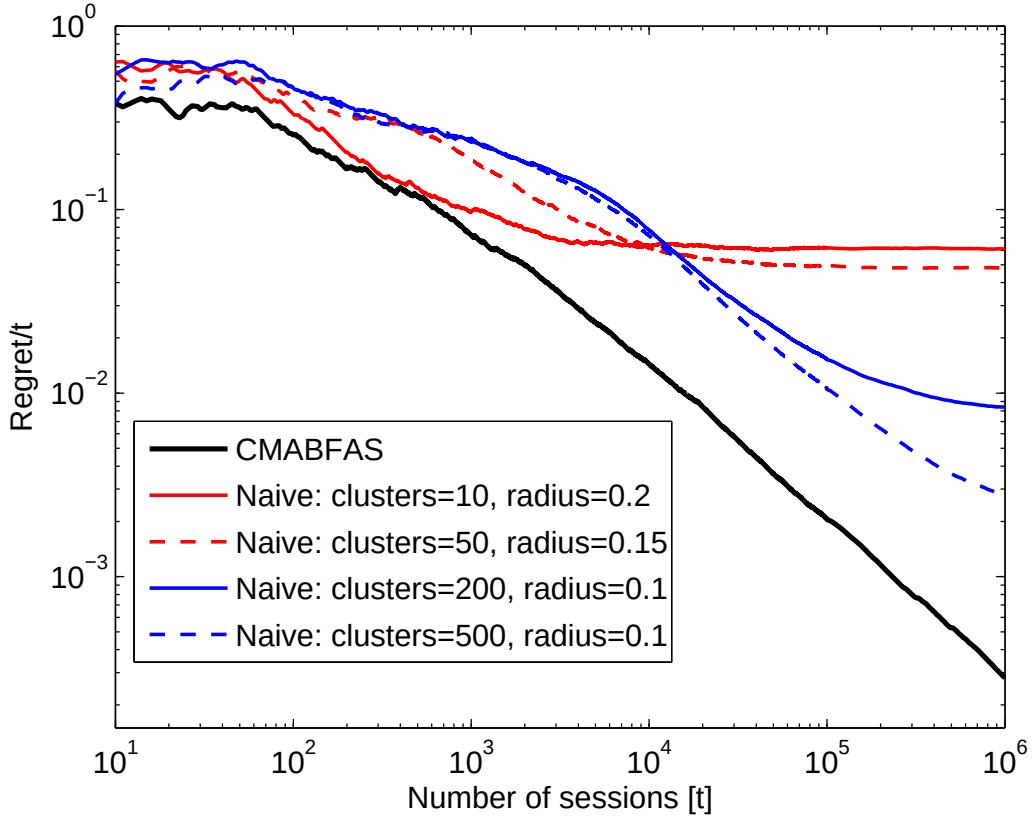
Experiment:

- Each of the sessions was semi-automatically assigned to a class/reward profile
- For $t = 1 \ldots 1,000,000$, draw random session x_t from corpus and present it to the CMABFAS defense system

Contestant: compare against a naive baseline method which

- incrementally but non-adaptively clusters $\mathcal{X}$
- assigns each incoming x_t to the nearest cluster
- uses a traditional k-armed bandit to handle each cluster separately (UCB-1.2).

Results



Overall:

- CMABFAS: $\sim$ 450 mistakes (out of 1,000,000 possible ones)
- Best naive MAB: $\sim$ 3200 mistakes (out of 1,000,000 possible ones)

Summary

Two contributions:

1. **Conceptual:**
   - Architecture for a self-learning web server defense via RL

2. **Algorithmical:**
   - CMABFAS (a new algorithm for contextual MAB)

Note: All of this is ongoing work. Experiments so far were merely "simulated".