Method of Detecting Vulnerability in Web Apps
Using Machine Learning

CODE BLUE 2016
Presented by Isao Takaesu

Isao Takaesu

(@bbr_bbq)

About the speaker

- Occupation: Web security engineer
- Company: Mitsui Bussan Secure Directions
- Hobbies: Making scanners, Machine Learning
- Blog: http://www.mbsd.jp/blog/
- Presented in “Black Hat Asia 2016” Arsenal
- hosted “AISECjp”

Agenda

1. Introduction
2. Objectives of research
3. Overview of SAIVS
4. Realized methods
5. Validation results
6. Demonstration
7. Future Prospects

Introduction

In Japan, there is

a **significant lack** of security engineers.

Introduction

About **240,000** people short.

(according to a survey by IPA)

Introduction

Are we facing **the limit** of human-only resources?

Objectives of research

Fully automated information security.

Objectives of research

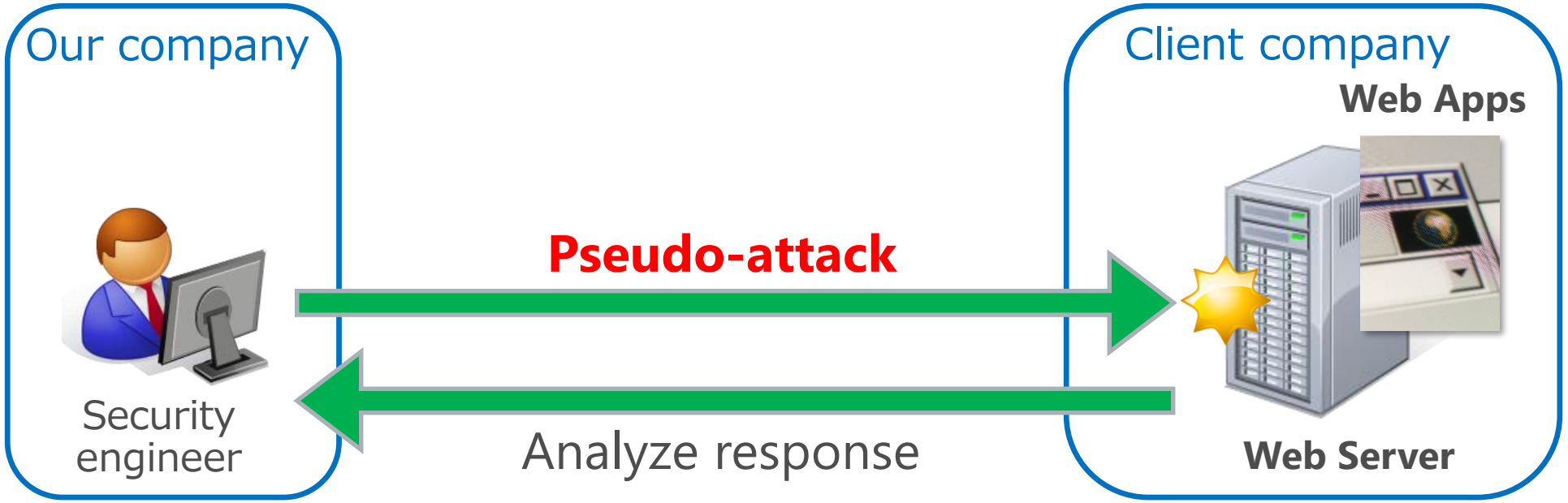
The first step is

Web application vulnerability assessment.

What is web app vul assessment ?

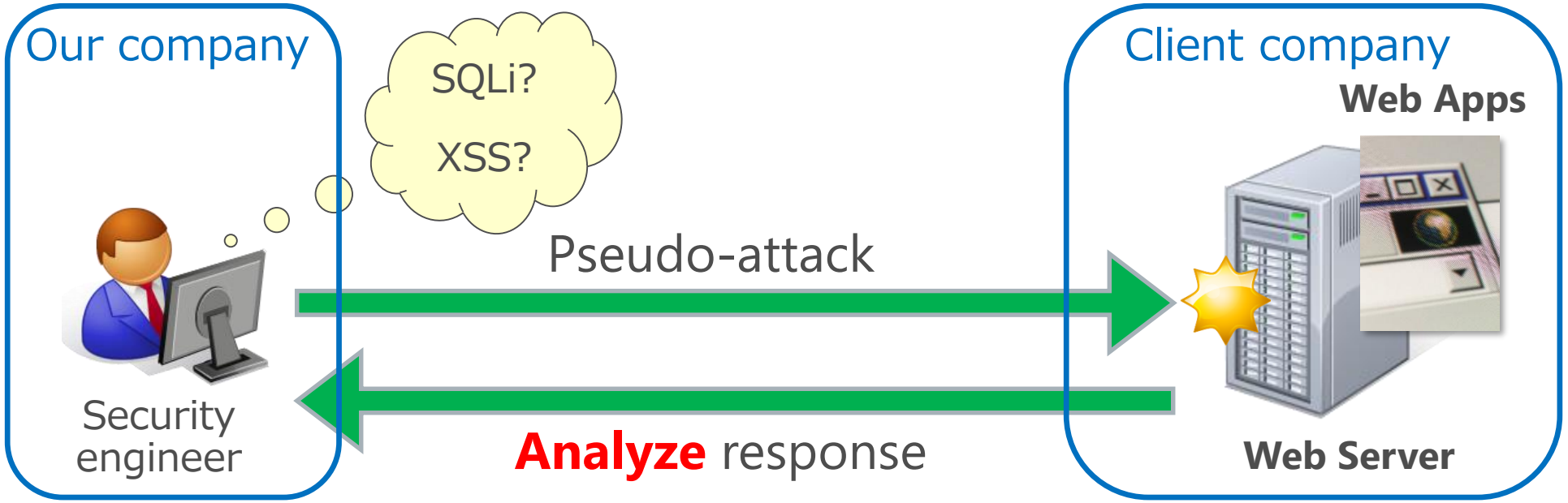
Detect vulnerabilities on web apps.

What is web app vul assessment ?



- **Performs pseudo-attacks while crawling the pages** of web apps.
- Analyzes the response and detects vulnerabilities.

What is web app vul assessment ?



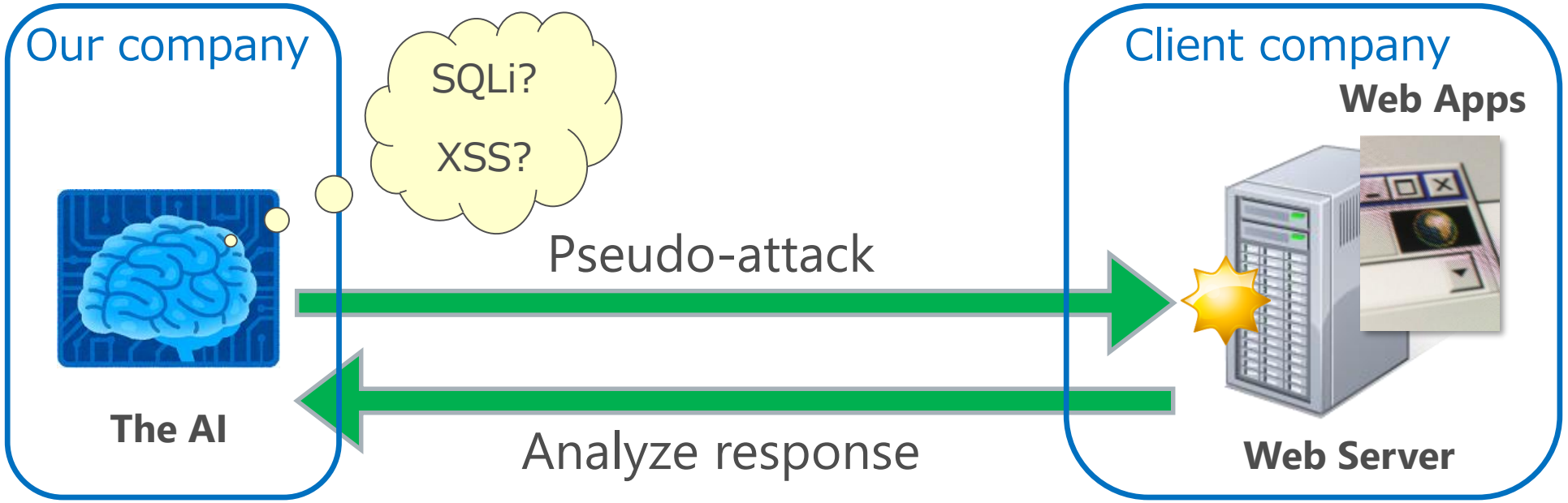
- Performs pseudo-attacks while crawling the pages of web apps.
- **Analyzes** the response and **detects vulnerabilities**.

Issues of web app vul assessment

Depends on the skilled craftsmanship

of security engineers.

Our “GOAL”



Realize **ALL MACHINE** web app vulnerability assessment.

SAIVS

Spider **A**rtificial **I**ntelligence **V**ulnerability **S**canner

AI that performs web app vul assessment.

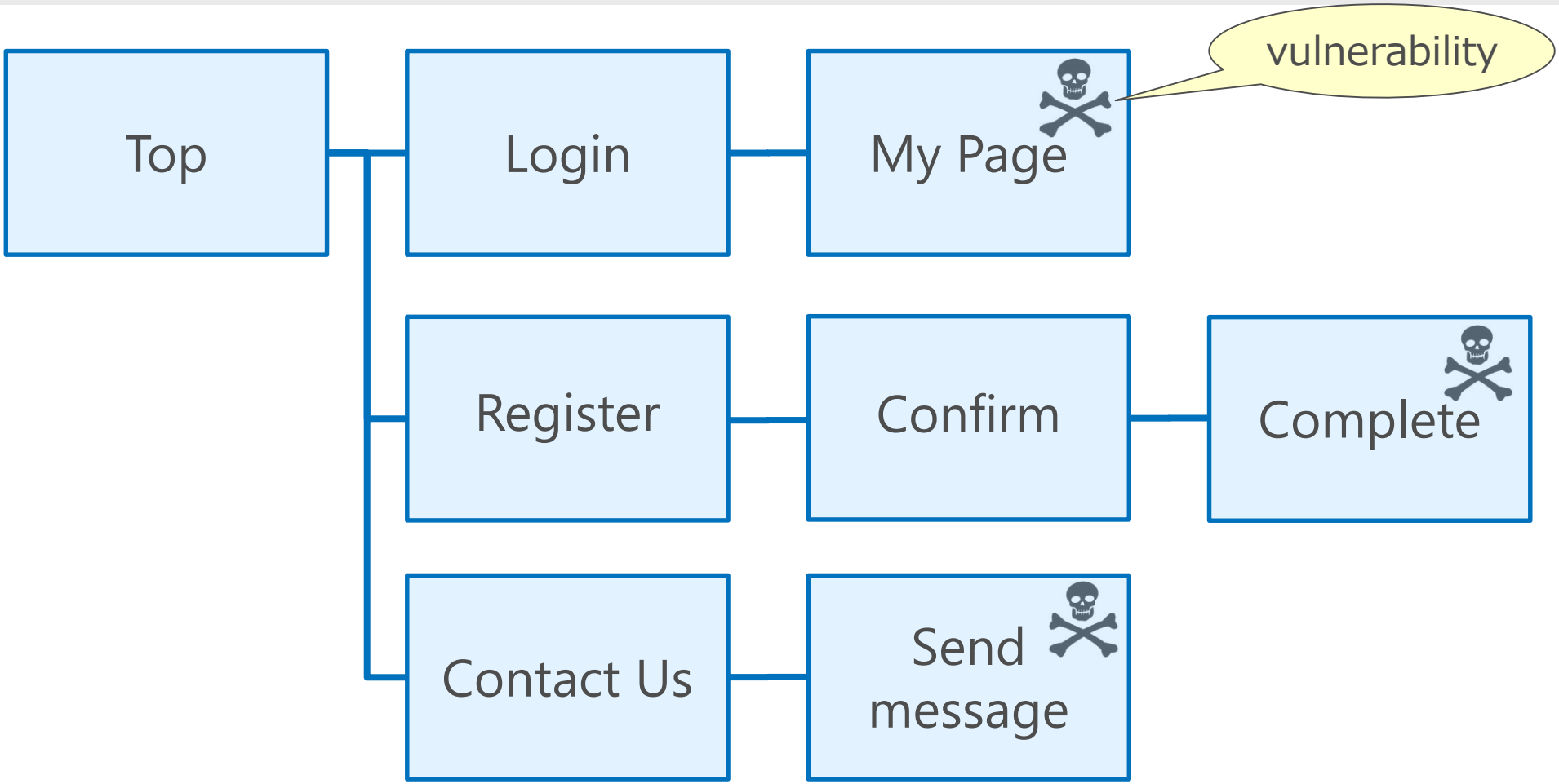
Abilities of SAIVS

- Crawls web apps
- Detects vulnerabilities

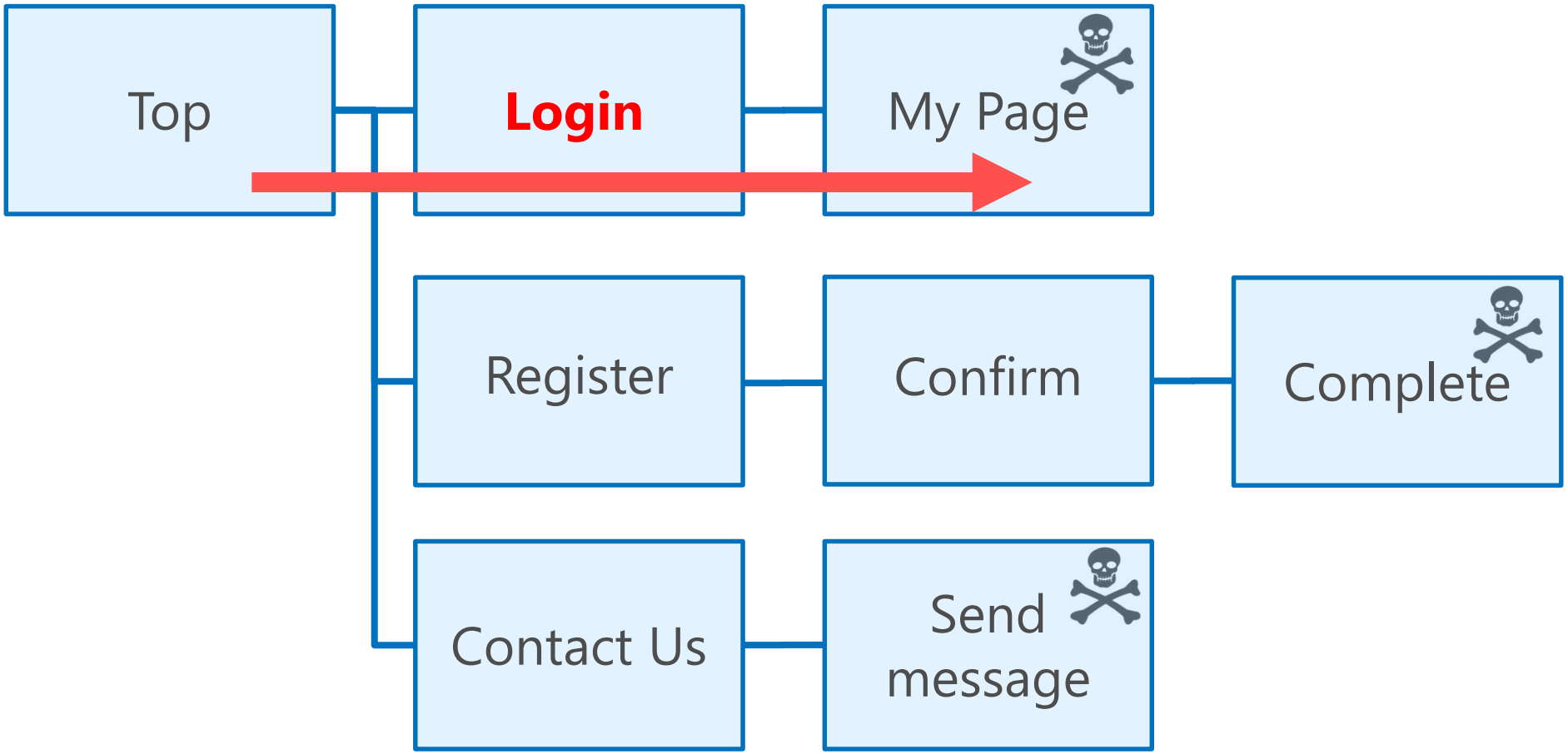
Abilities of SAIVS

- **Crawls web apps**
- Detects vulnerabilities

Importance of crawling

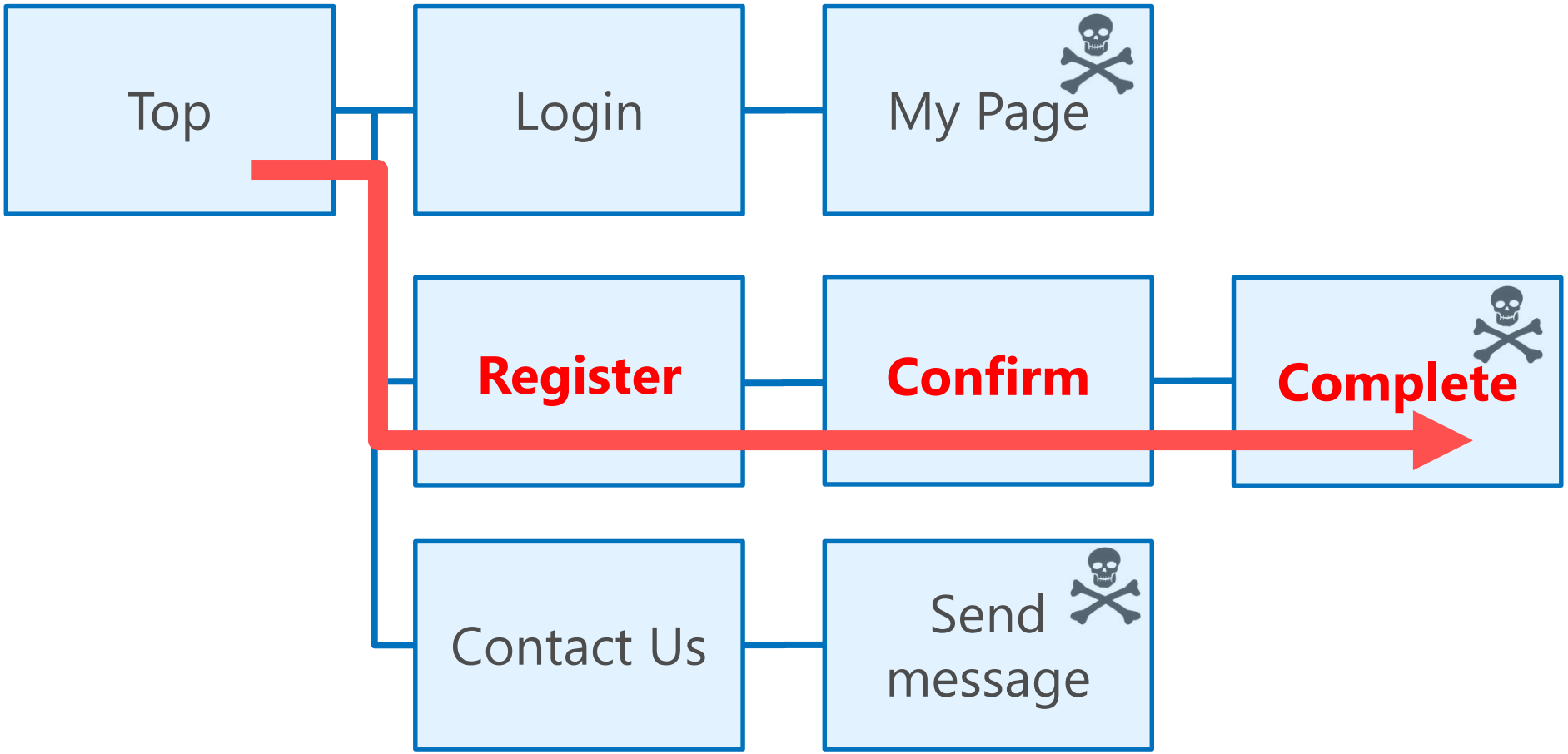


Importance of crawling



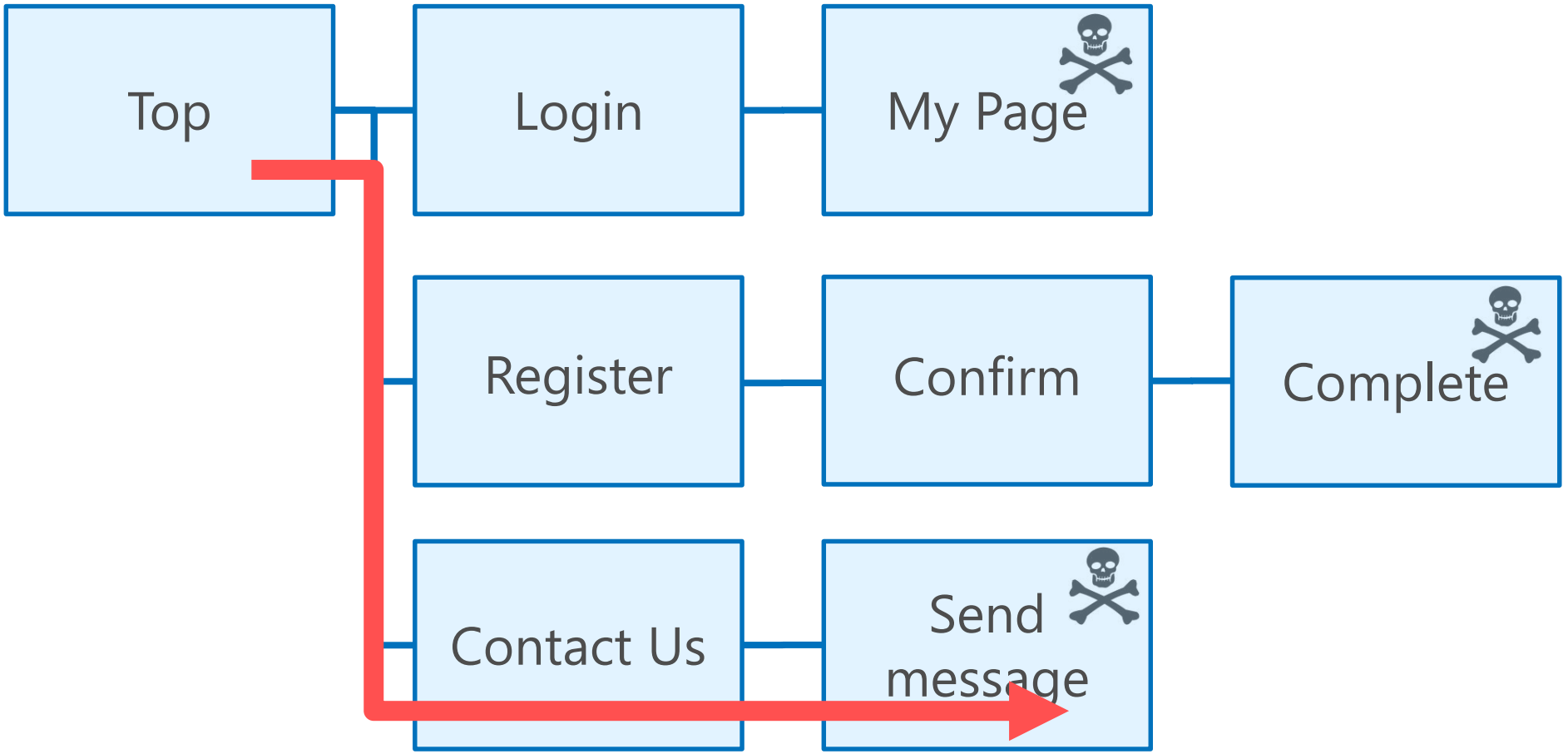
It is necessary to **log in** correctly.

Importance of crawling.



It is necessary to **register** correctly.

Importance of crawling

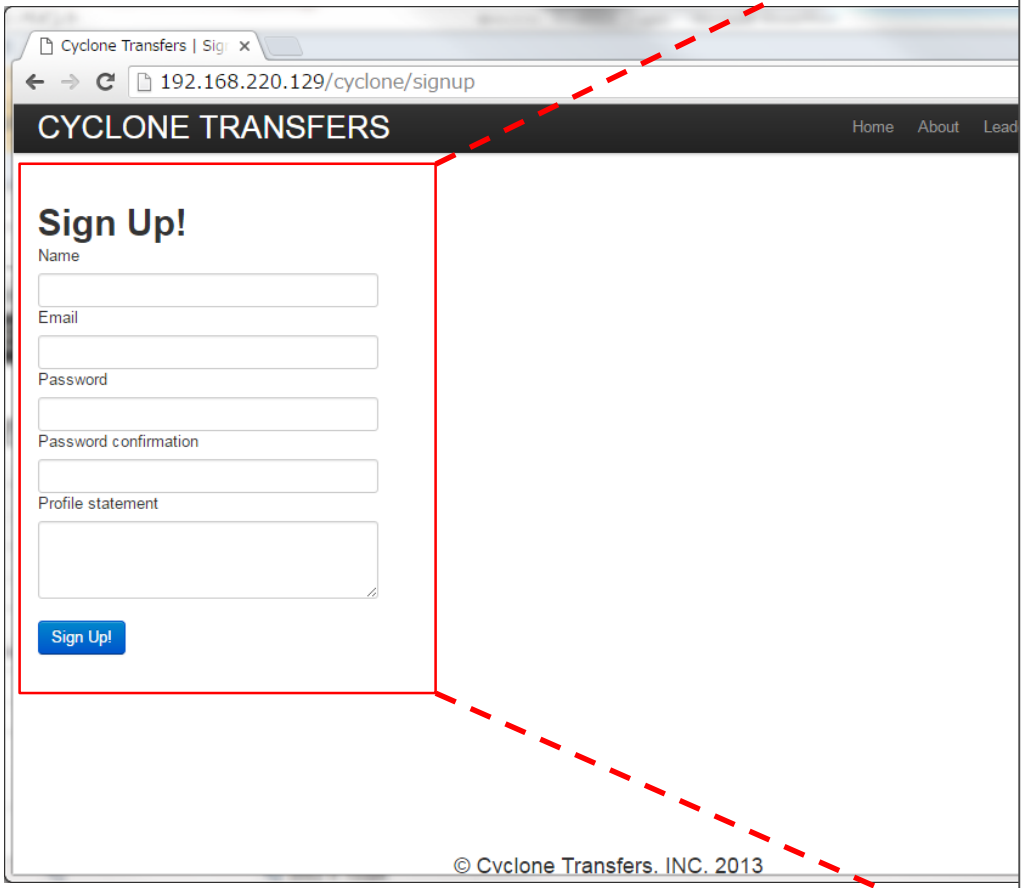


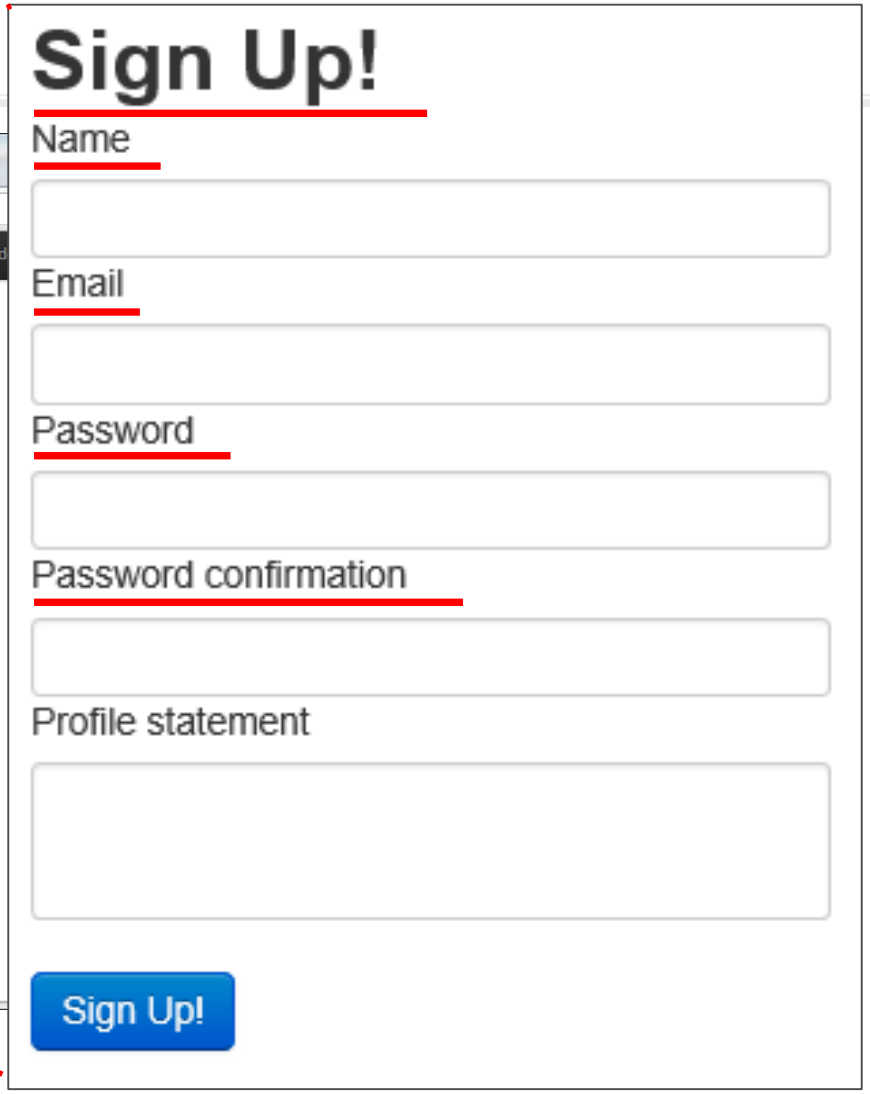
It is important to **exhaustively crawl pages**.

Crawling web apps is **difficult**.

Why ?

What is the type of this page ?

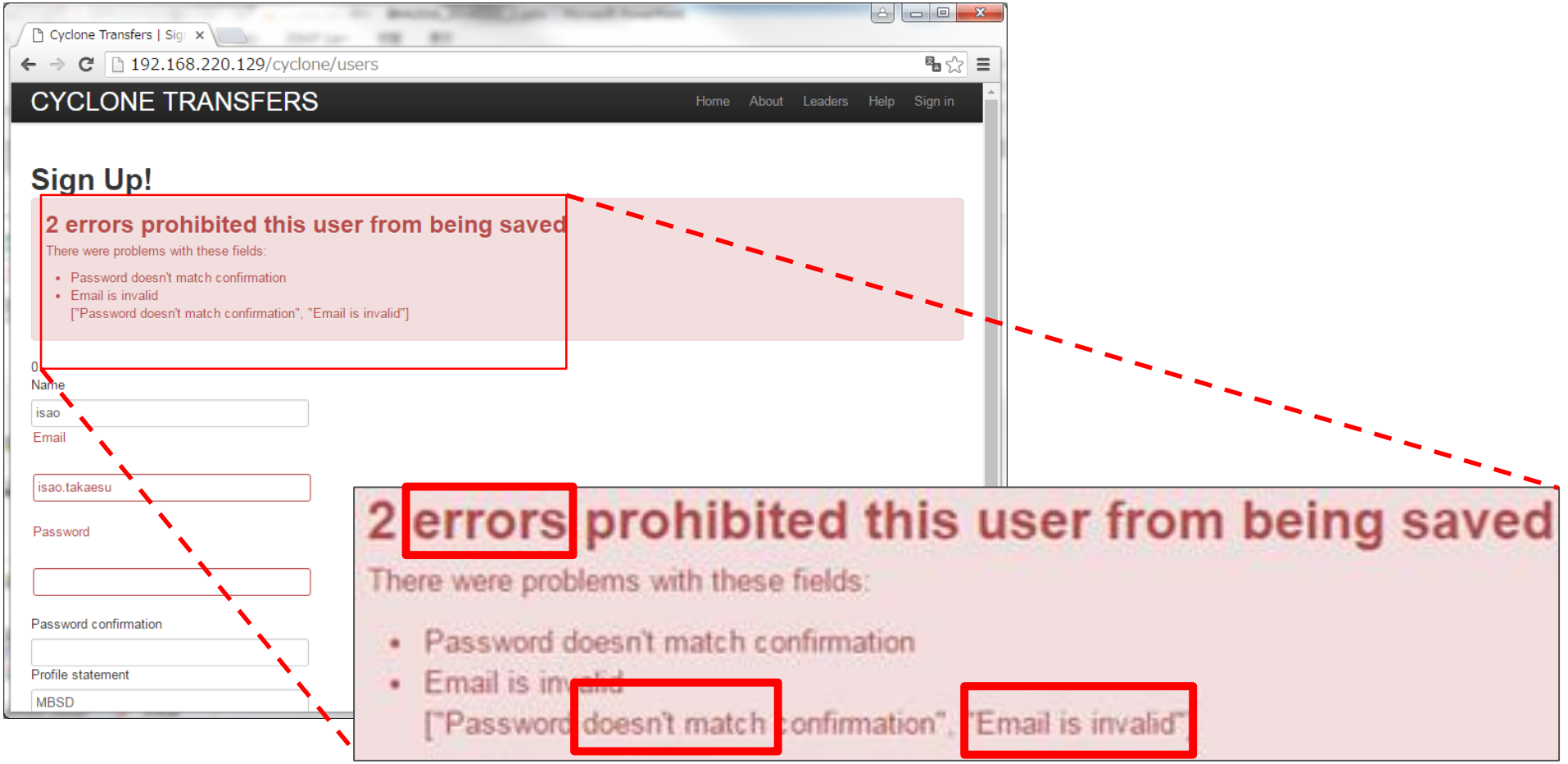




Humans : Easily **recognize the page type.**

Machine : Difficult. (Log in? Register?)

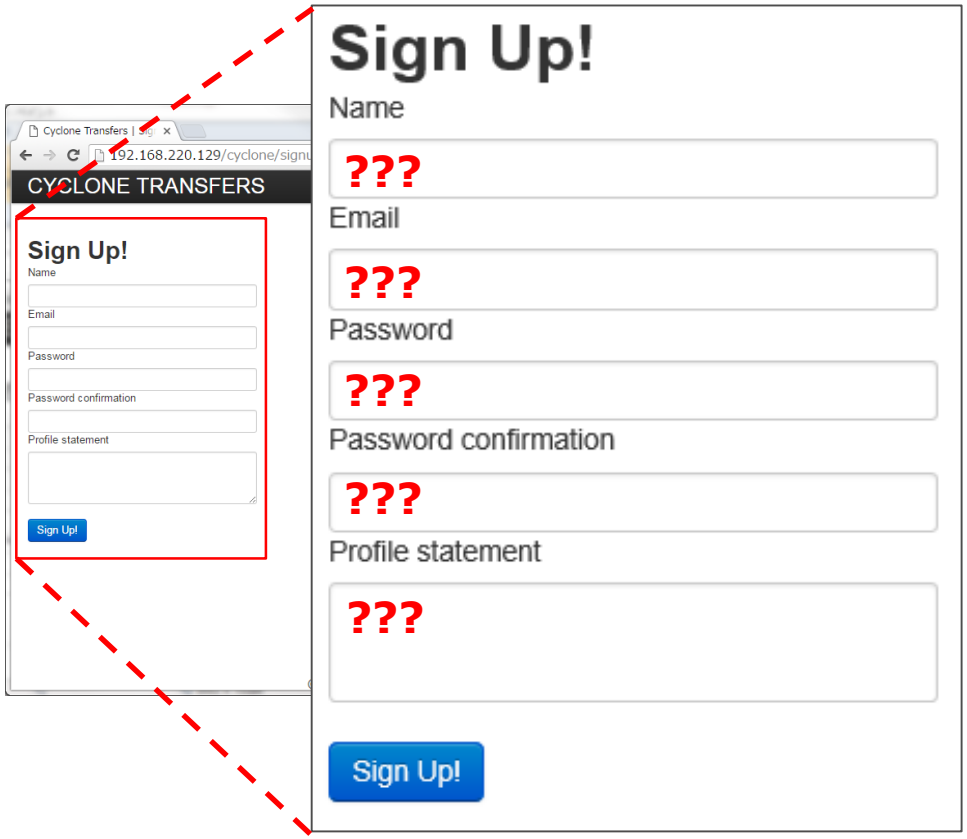
What has just happened?



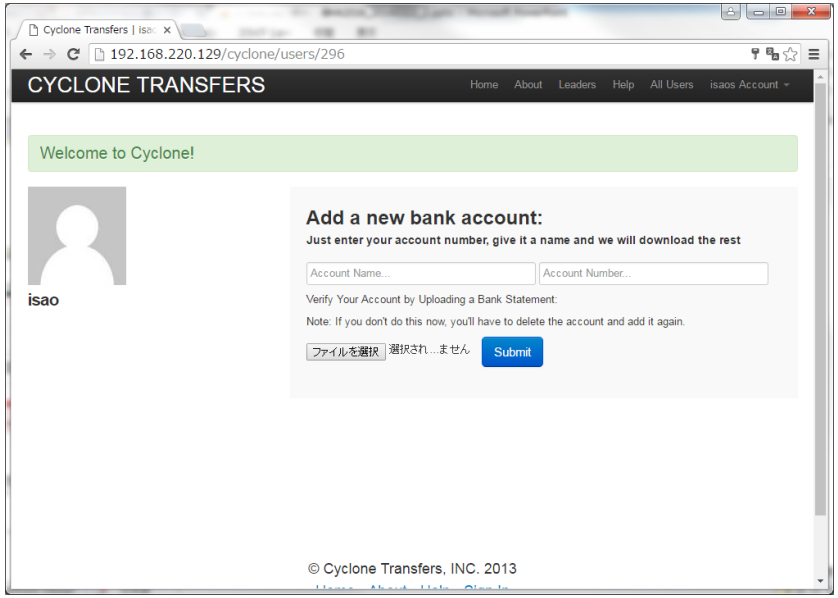
Humans : **Recognize failed crawling** attempts from the message.

Machine : Difficult to interpret the message.

What are your input values to crawl to the next page ?



crawl



Humans : **Input correct values** in the form.

Machine : Difficult to interpret the meaning of the form.

Crawling requires complex thinking.

3 requirements for crawling

- Recognize the page type
- Recognize the success/fail of crawling
- Input correct values in the forms

How do you achieve them ?

Our approach

Reverse engineering of the human brain.

Our approach

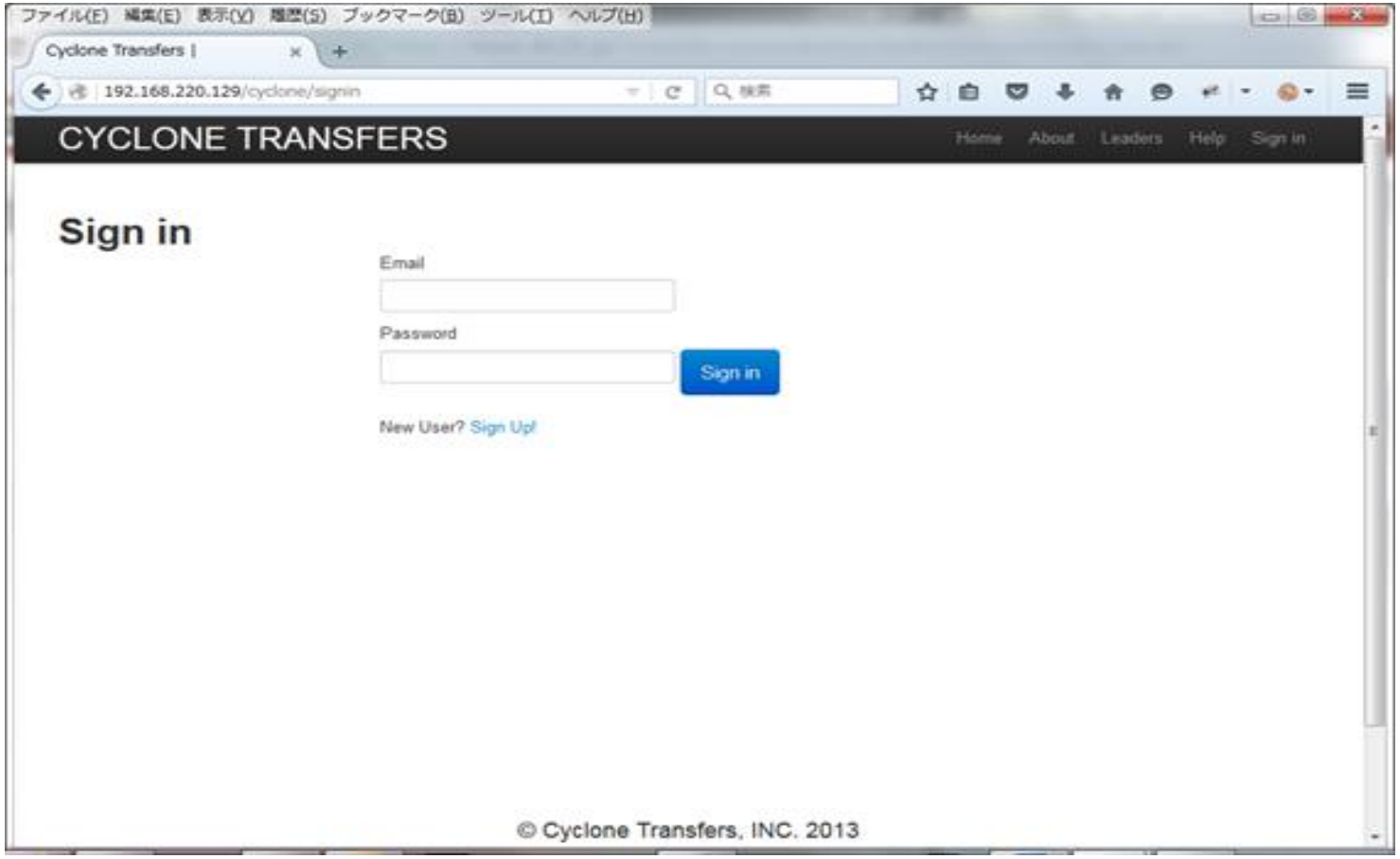
Each thinking pattern

implemented in machine learning.

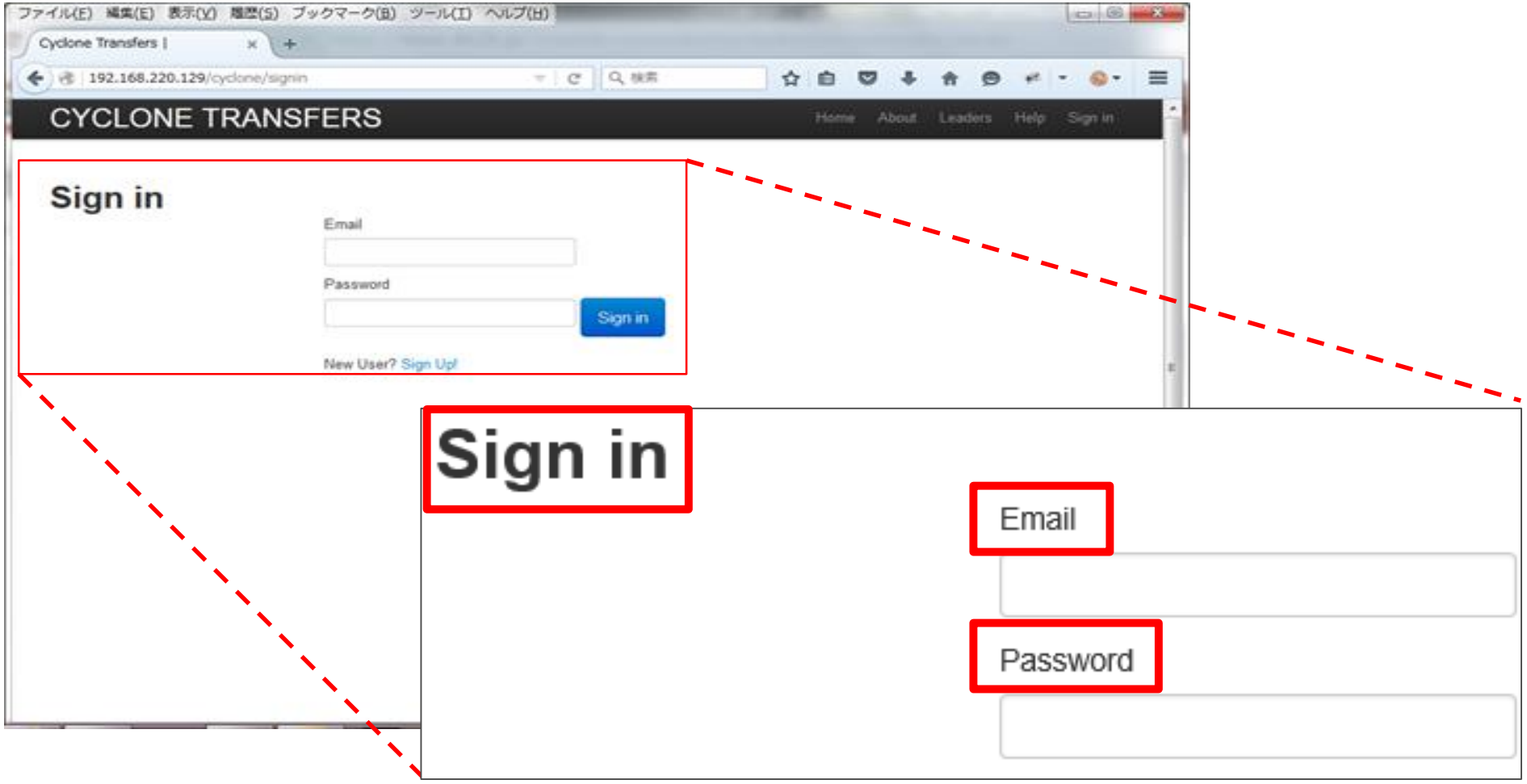
3 requirements for crawling

- **Recognize the page type**
- Recognize the success/fail of crawling
- Input correct values in the forms

Page recognition by humans



Page recognition by humans



Recognize the **keywords that characterize the page**.

Page recognition by machines

Using Naive Bayes.

What is Naive Bayes?

Naive Bayes is used for auto classification of texts.

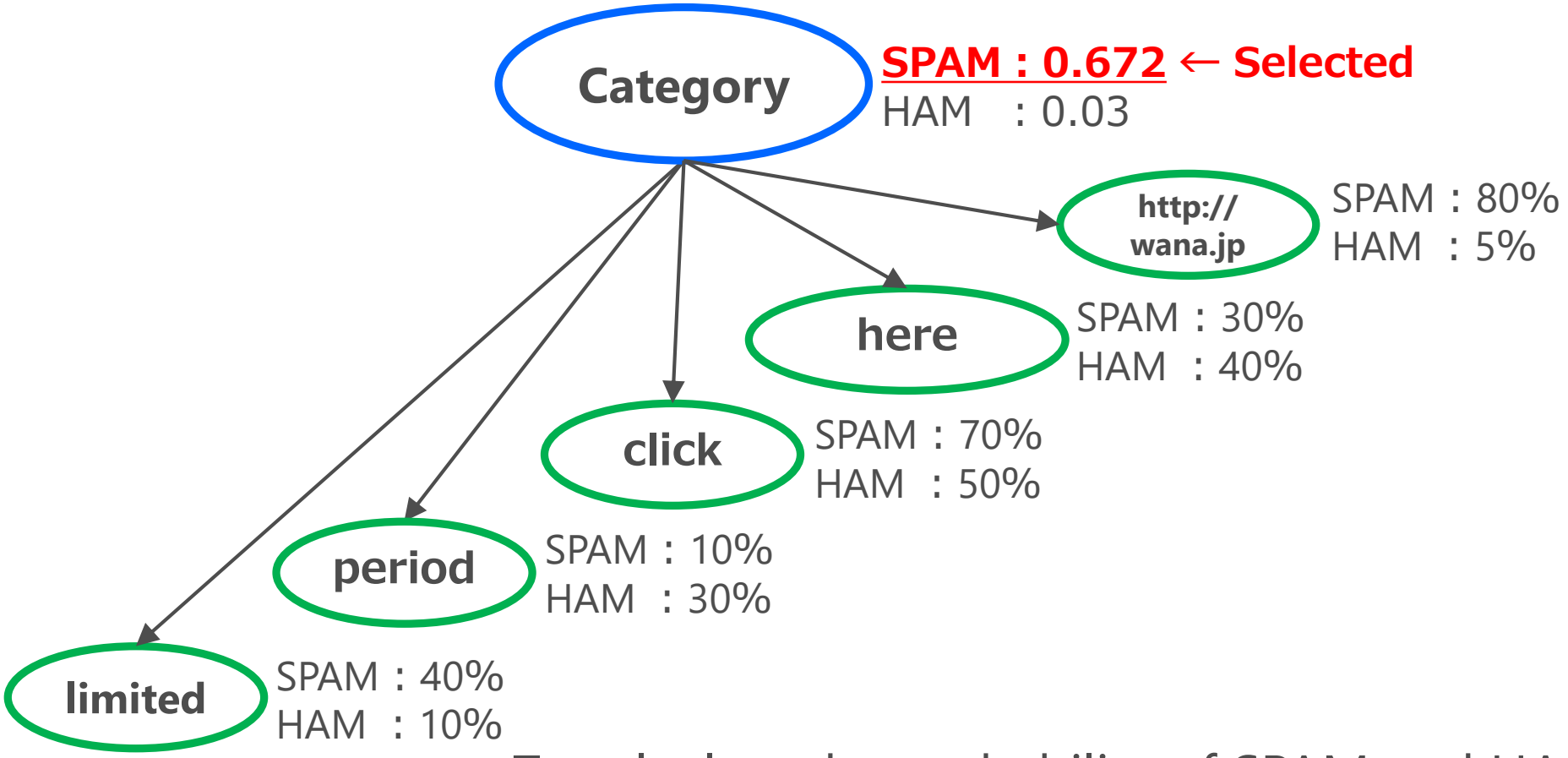
Uses **pre-defined categories** & **laws of probability**.

examples)

- Spam mail filter.
- Classify blog post genres.
- Improvement of defensive accuracy for WAF.

Example of the use of the spam e-mail filter.

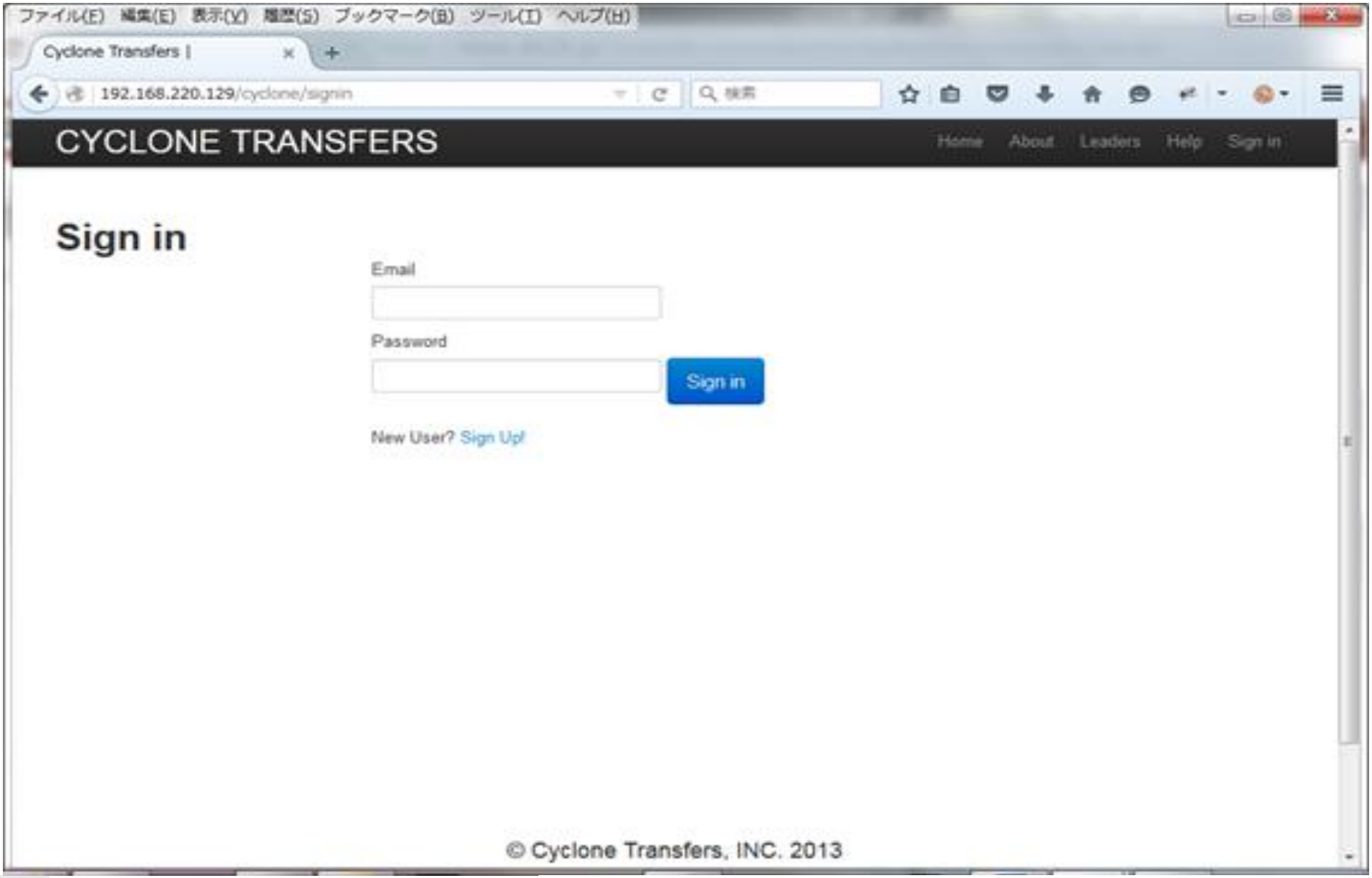
Extracting a keyword from the mail text.



To calculate the probability of SPAM and HAM,

to select the probability is high category.

Page recognition by Naive Bayes



Page recognition by Naive Bayes

```
<h1>Sign in</h1>
<form action="/cyclone/sessions" method="post">
<label for="email">Email</label>
<input id="email" name="email" type="text" />
<label for="password">Password</label>
<input id="password" name="password" type="password" />
</form>
```

Extracts keywords that characterize the page.

Excludes “Stop words”.

Category table for page recognition

Categories	Keywords
Log in	Email, User ID, Password, Sign in …
Register user	Email, Password, Confirm, Sign up …
Search	Word, Text, String, Sort, Search …
Purchase goods	Credit, Account, Expire, Purchase …
Change password	Password, Old Password, Change …

Selects the category that contain the many keywords.

Keywords : Sign in, Email, Password

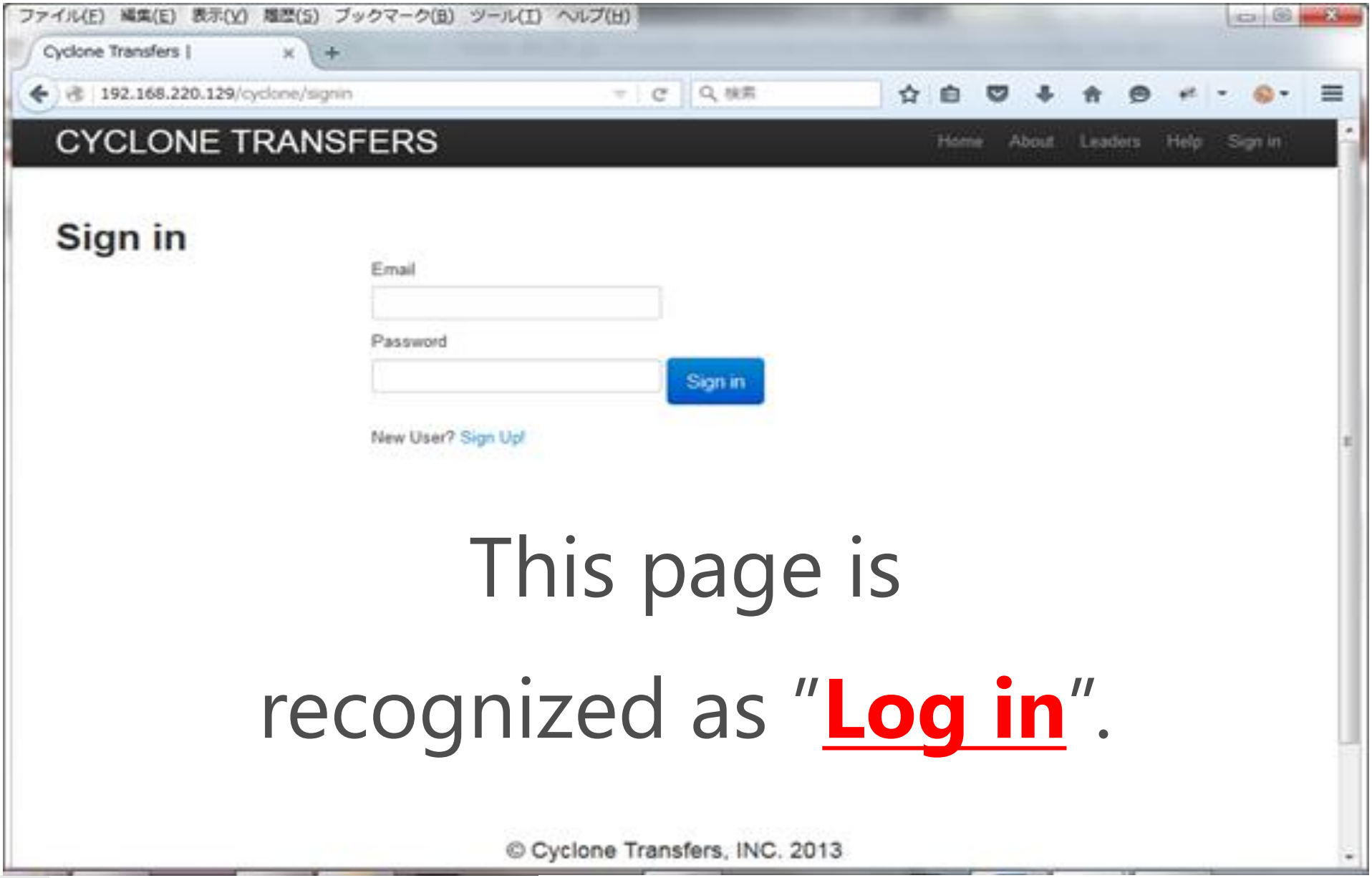
Category table for page recognition

Categories	Keywords
Log in	**Email**, User ID, **Password**, **Sign in** …
Register user	**Email**, **Password**, Confirm, Sign up …
Search	Word, Text, String, Sort, Search …
Purchase goods	Credit, Account, Expire, Purchase …
Change password	**Password**, Old Password, Change …

"Log in" category contains many of the keywords.

⇒ **The highest probability** of the category "**Log in**".

Page recognition by Naive Bayes



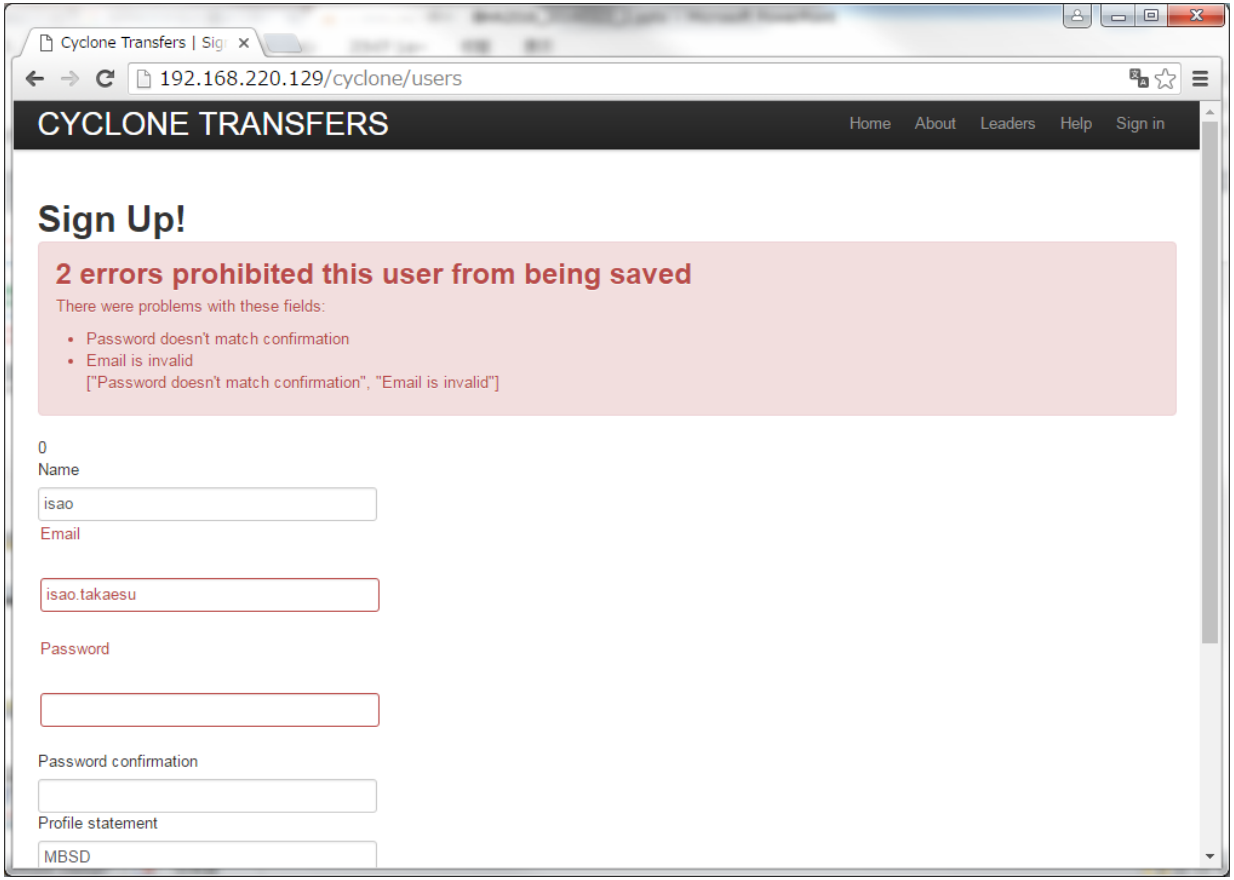
This page is

recognized as "**Log in**".

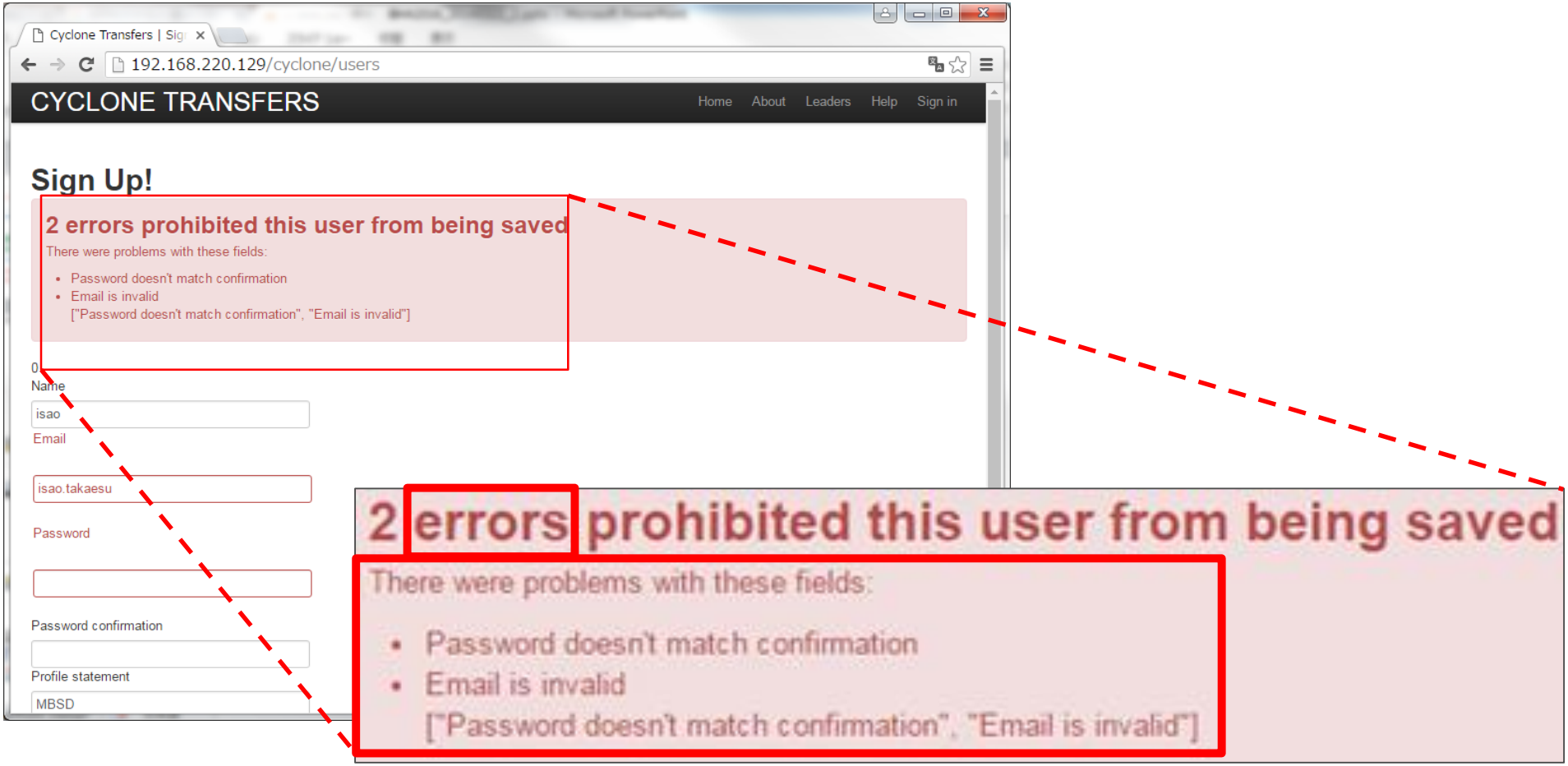
3 requirements for crawling

- Recognize the page type (✓)
- **Recognize the success/fail of crawling**
- Input correct values in the forms

Recognition of page crawling success/fail by humans



Recognition of page crawling success/fail by humans

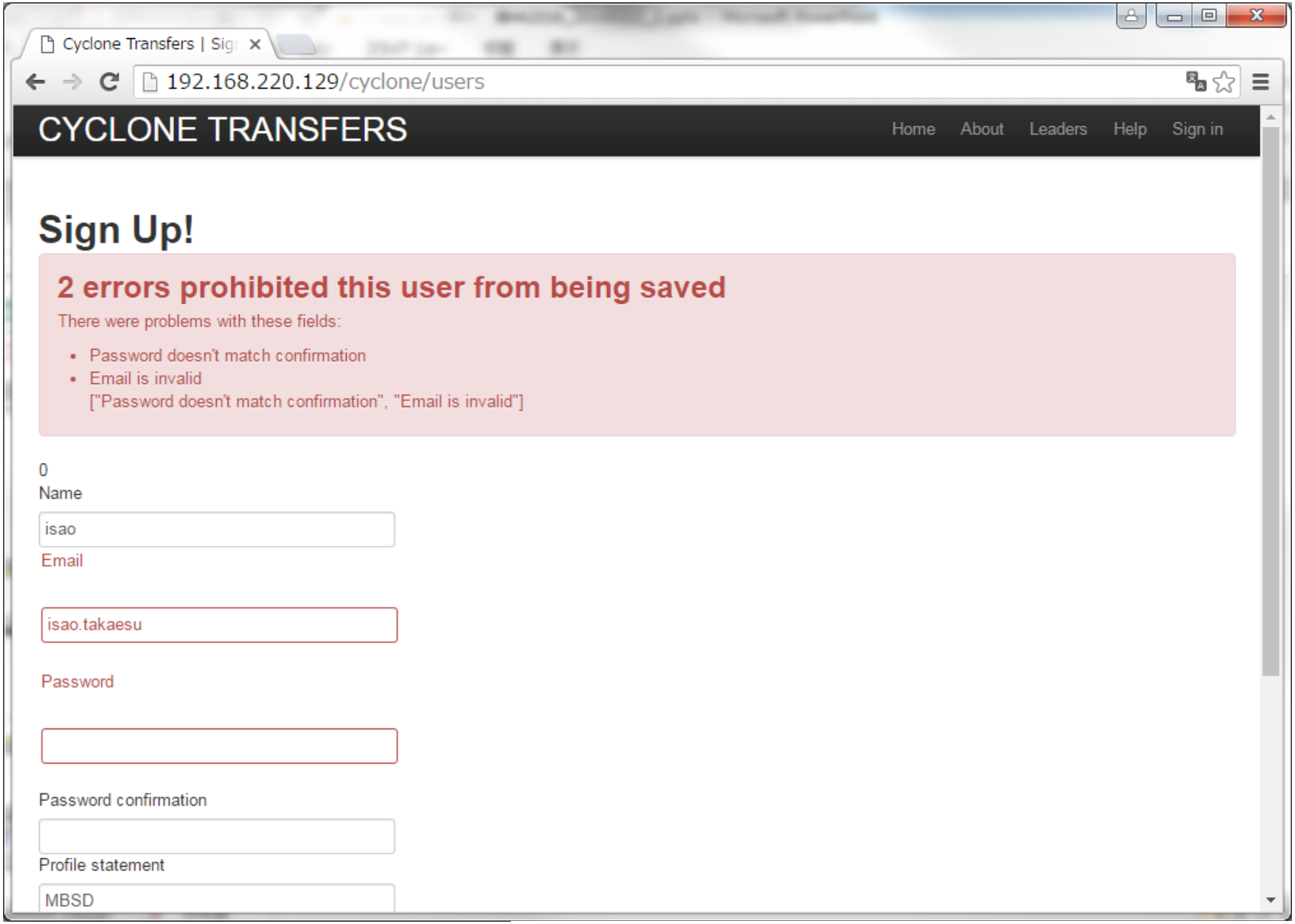


Recognizes the **keywords that characterize failure**.

Recognition of page crawling success/fail by machines

Using Naive Bayes.

Recognition of page crawling success/fail by machines



Recognition of page crawling success/fail by machines

```
<h2>2 errors prohibited this user from being saved </h2>
<p>There were problems with these fields:</p>
<ul>
<li> Password doesn't match confirmation </li>
<li> Email is invalid </li>
["Password doesn't match confirmation","Email is invalid
"]
</ul>
```

Extracts keywords that characterize failure.

Excludes “Stop words”.

Category table for page crawling success/fail

Categories	Keywords
Success	good, valid, success, normal, fine, clean, nice, can, match, confirmation, ok, finish, thank …
Failure	bad, invalid, failure, error, problem, unmatch, doesn't match, can't, too, wrong, ng, blank …

Selects the category that contain the many keywords.

Keywords : errors, problem, doesn't match, invalid

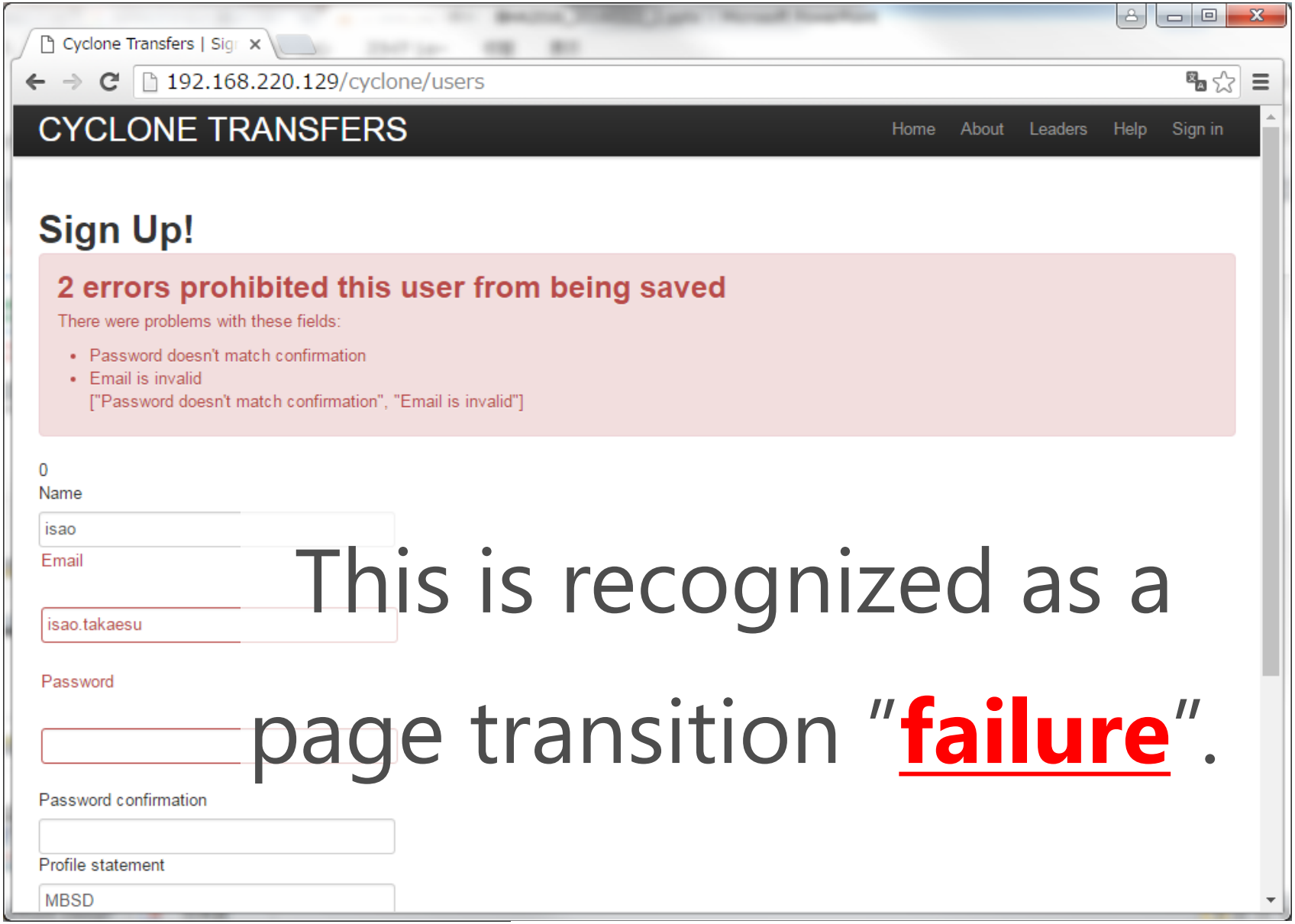
Category table for page crawling success/fail

Categories	Keywords
Success	good, valid, success, normal, fine, clean, nice, can, **match**, confirmation, ok, finish, thank …
Failure	bad, **invalid**, failure, **error**, **problem**, wrong, **doesn't match**, can't, too, ng, blank …

"Failure" category contains many of the keywords.

⇒ **The highest probability** of the category "**Failure**".

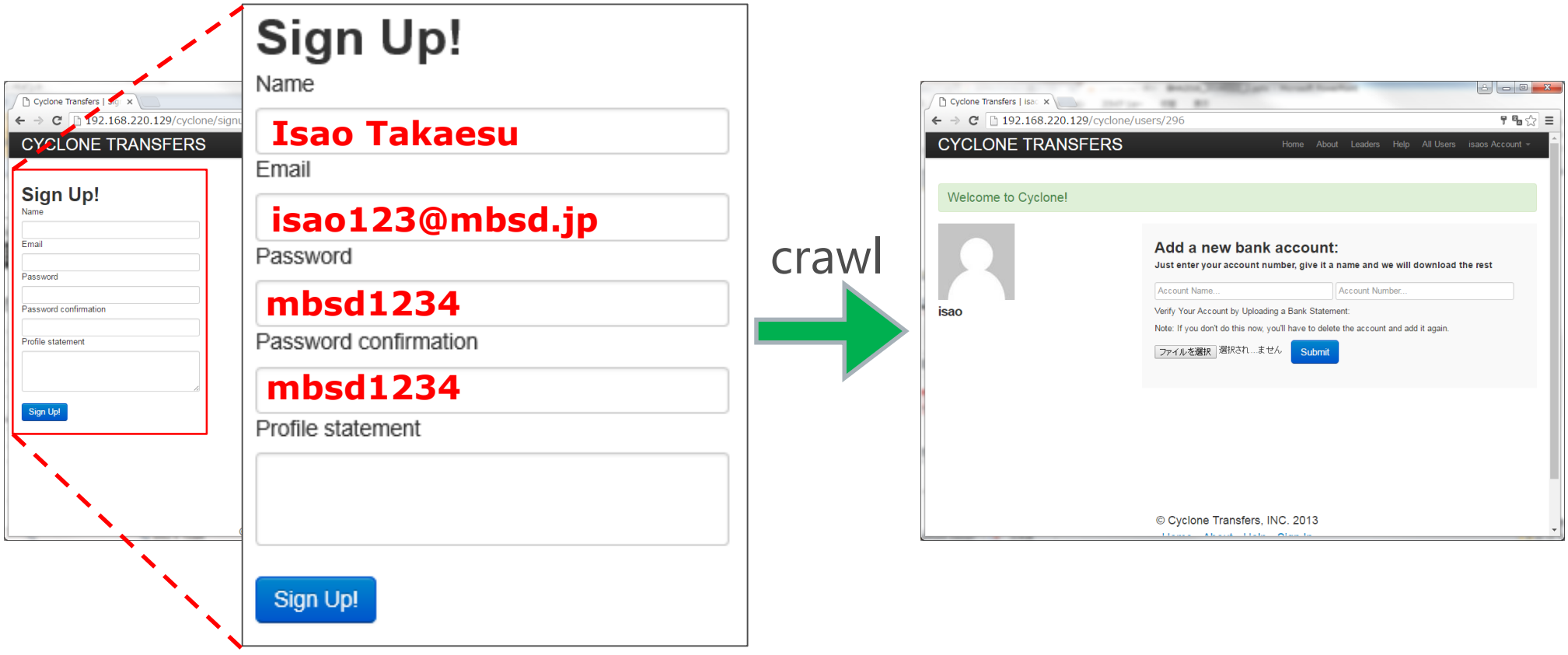
Recognize the success/fail of crawling by machines



3 requirements for crawling

- Recognize the page type (✓).
- Recognize the success/fail of crawling (✓).
- **Input correct values in the forms.**

Inputting correct values by humans



Interprets the meaning of the form and **inputs correct values**.

Inputting correct values by machines

Using multilayer perceptron &

Q-Learning.

What is multilayer perceptron（MLP） ?

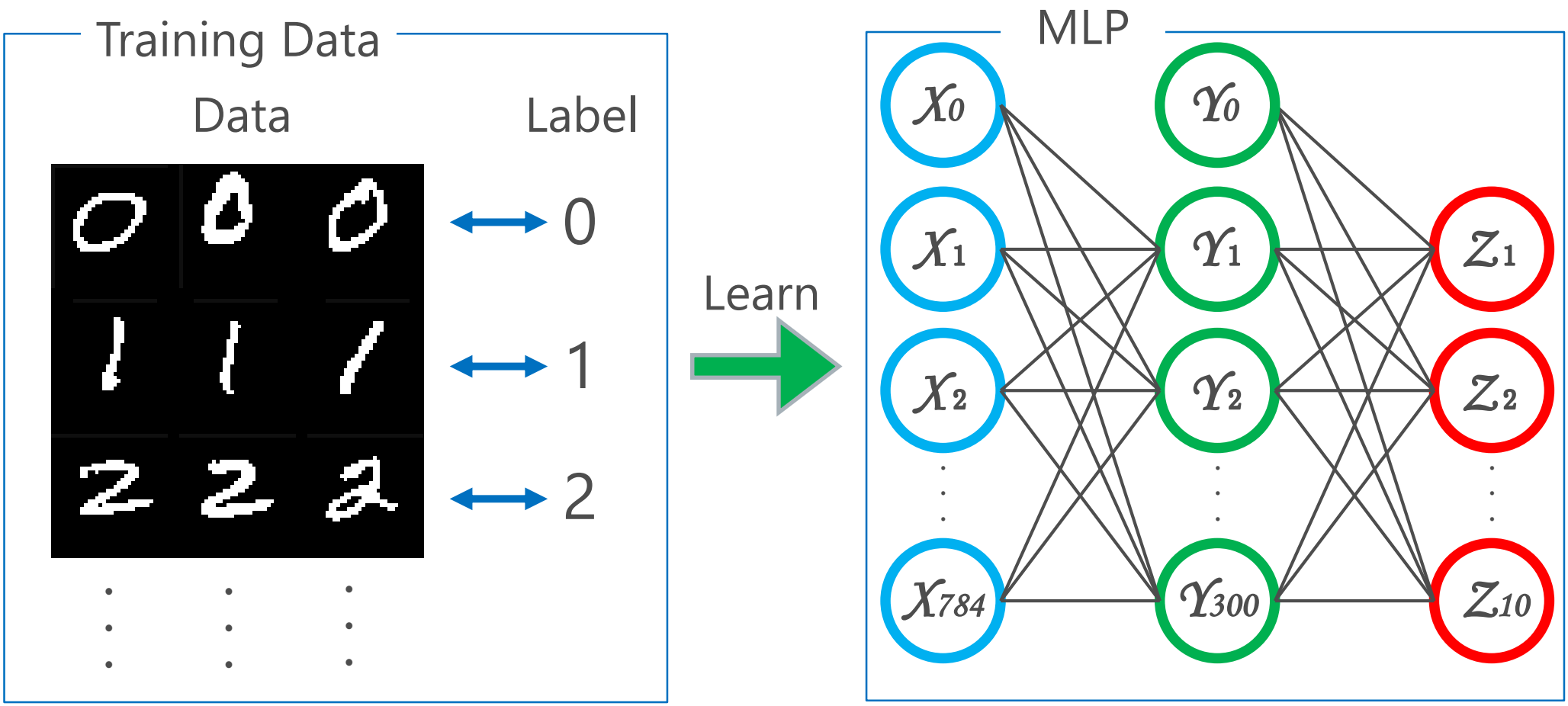
MLP is used for **image recognition**.

The model simulates the neural structure and function.

examples)

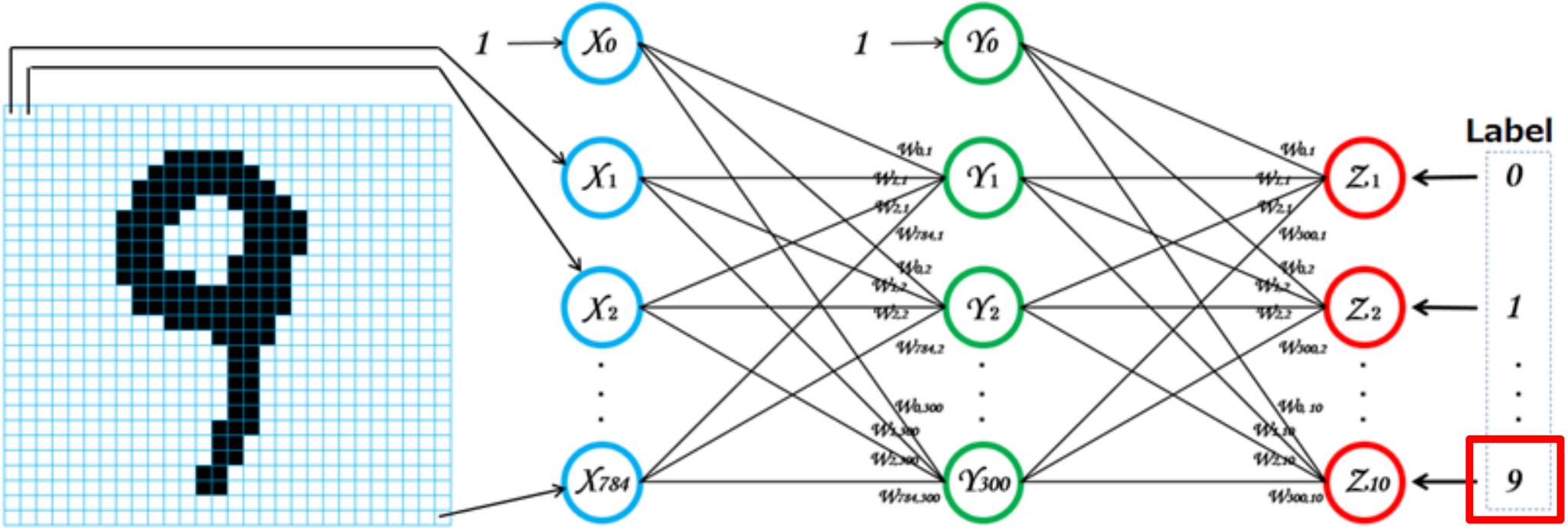
- Image recognition.
- Hand-written number recognition.

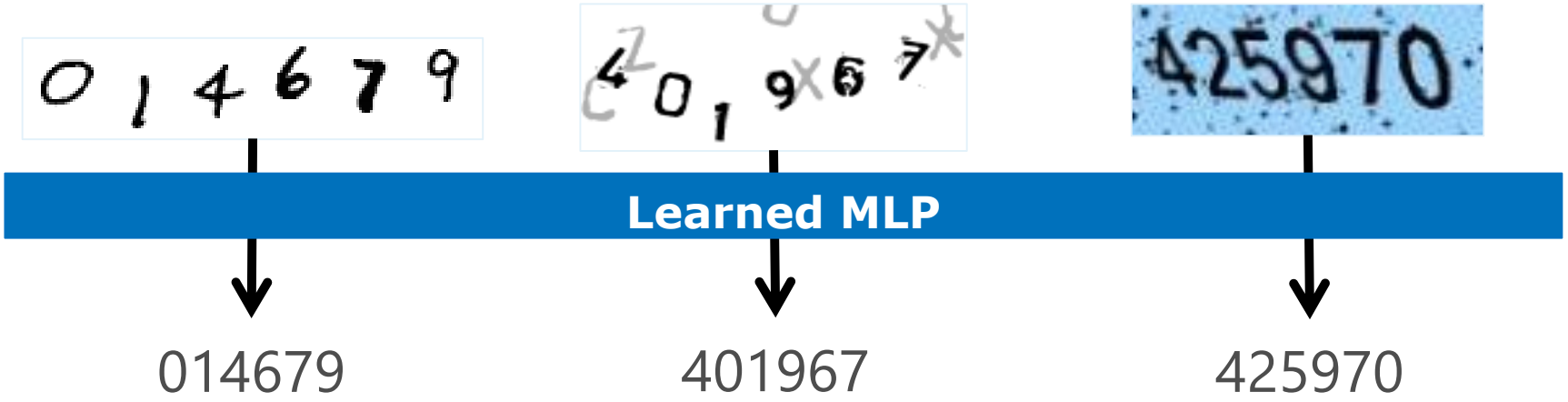
Example of the use of the hand-written number recognition



It can **output the answer depending on the input data**.

Example of the use of the hand-written number recognition





What is Q-Learning ?

An algorithm that learns the best behavior of the agent.

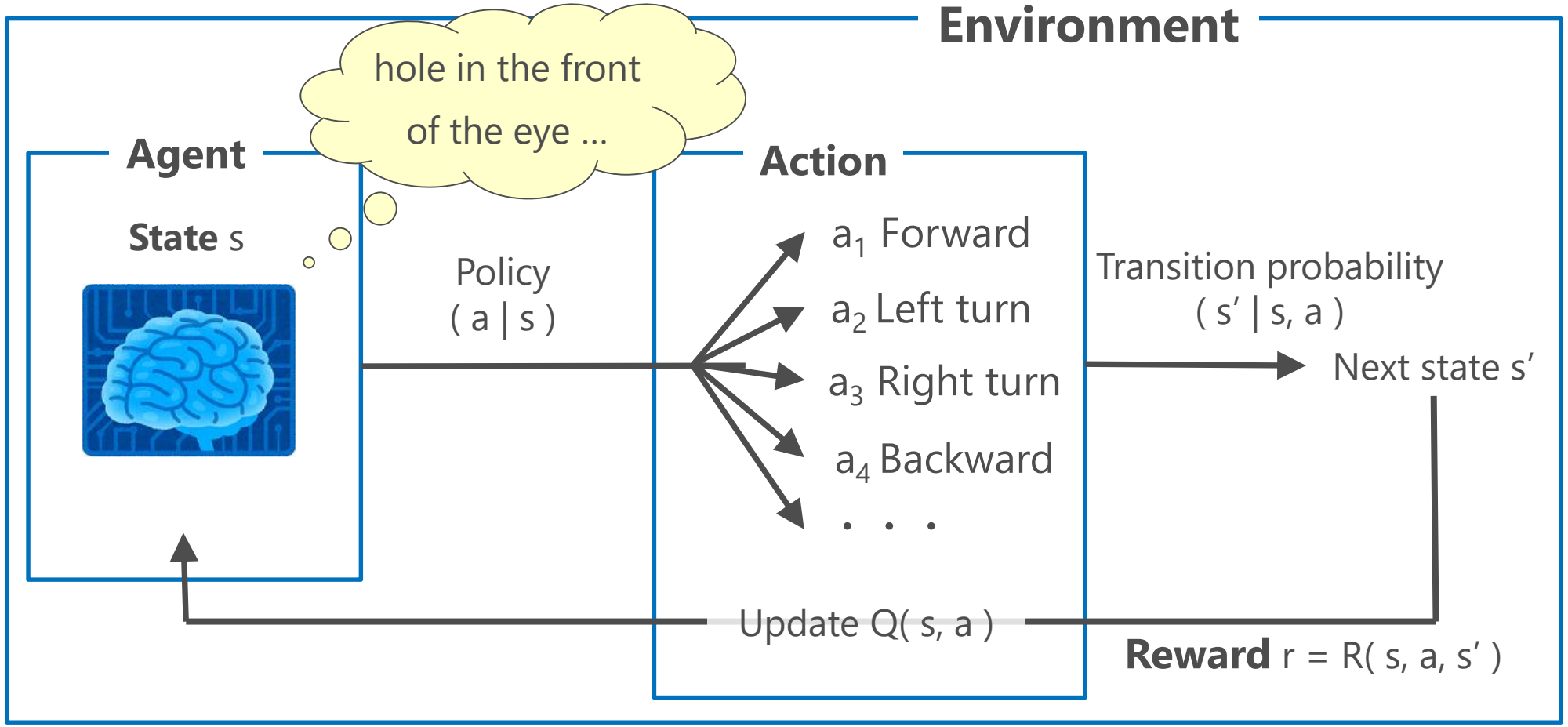
Evaluates effectiveness of the behavior using the Q-value.

examples)

- Locomotive learning in robots.
- Playing video games.
- Route search.

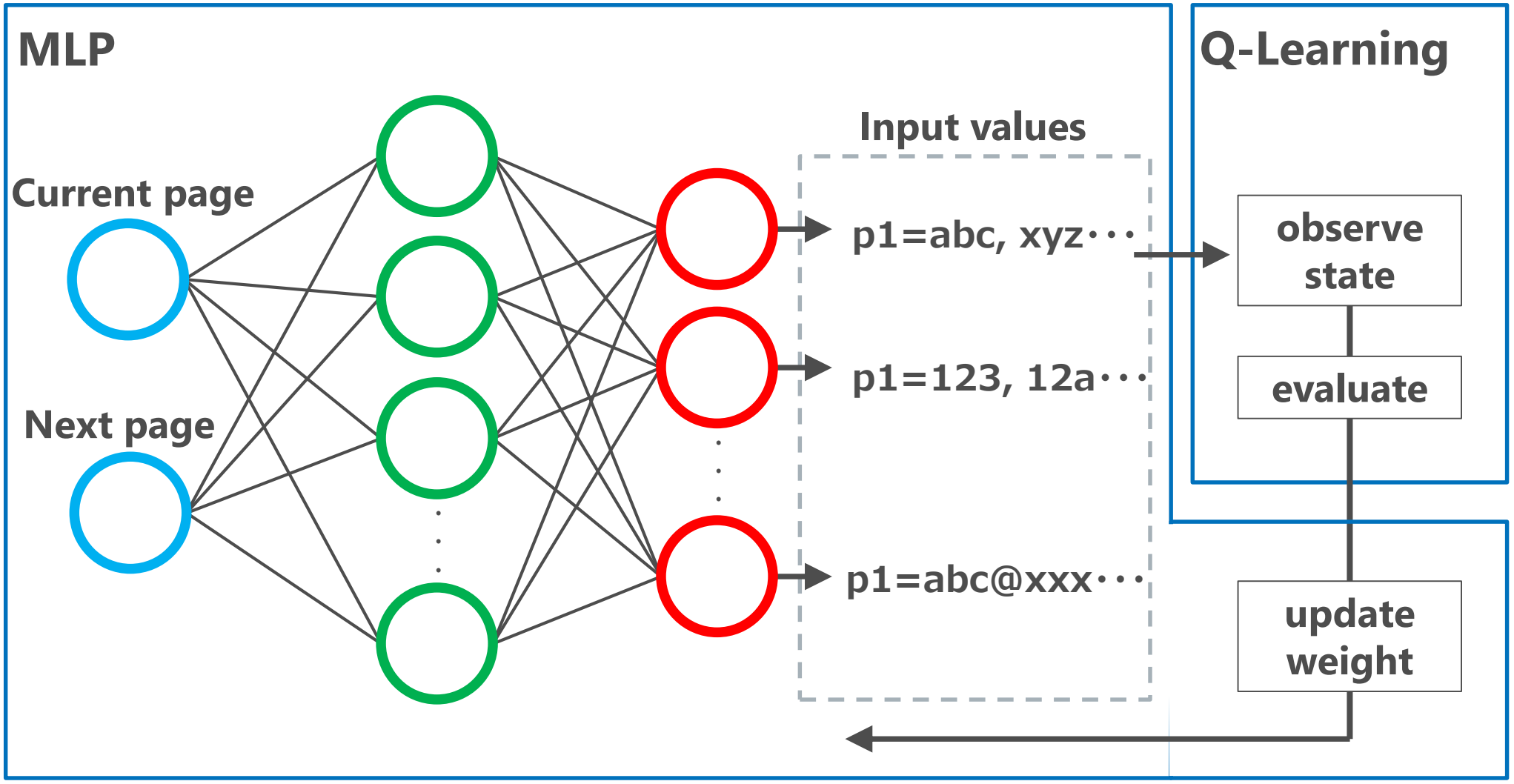
Example of the use of the locomotive learning in robots

Task : The goal to avoid obstacles.



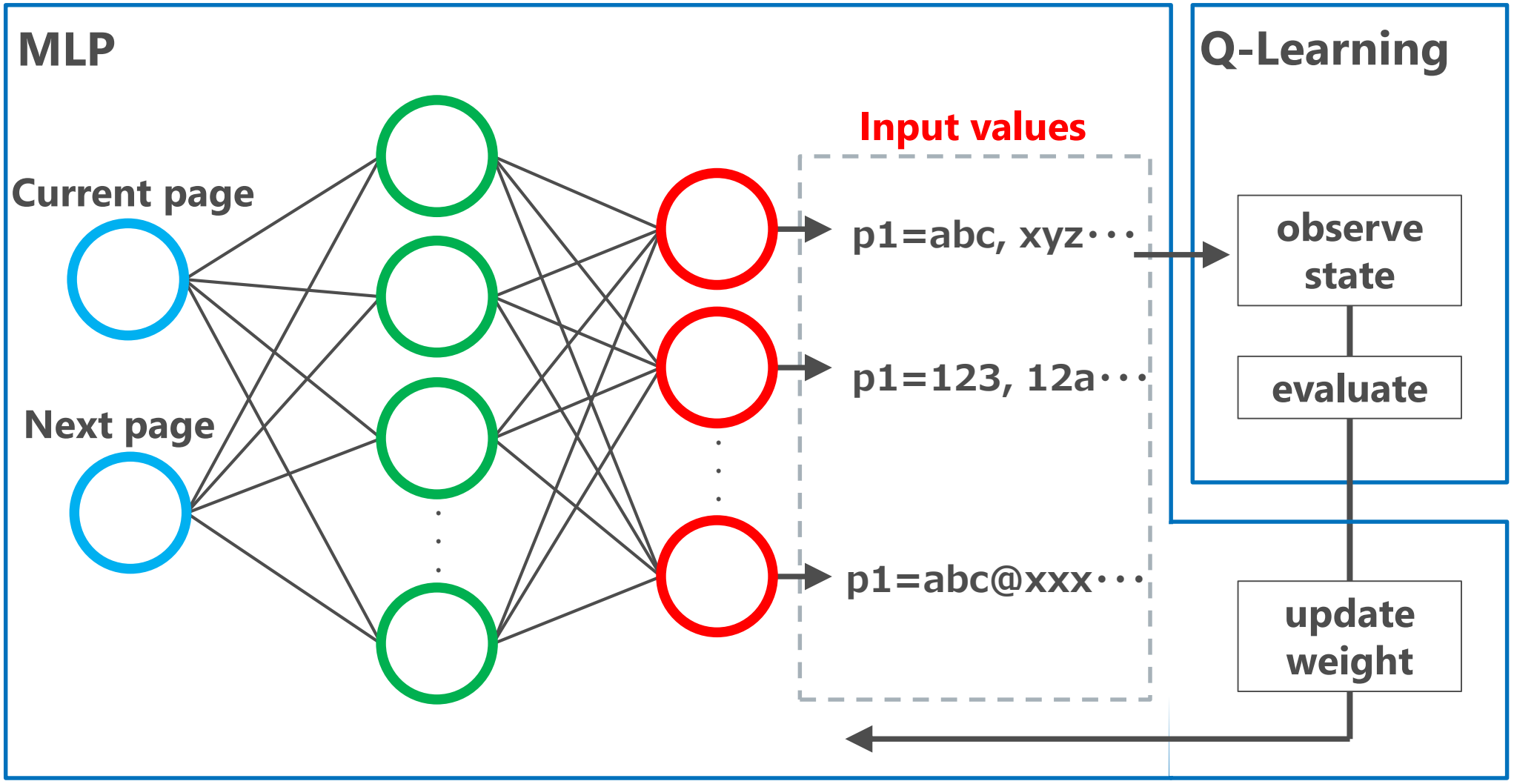
Learn the policies that **total of reward is the maximum**.

Model to attain correct input values



Learns correct input values while attempting to crawl.

Model to attain correct input values



Learns correct input values while attempting to crawl.

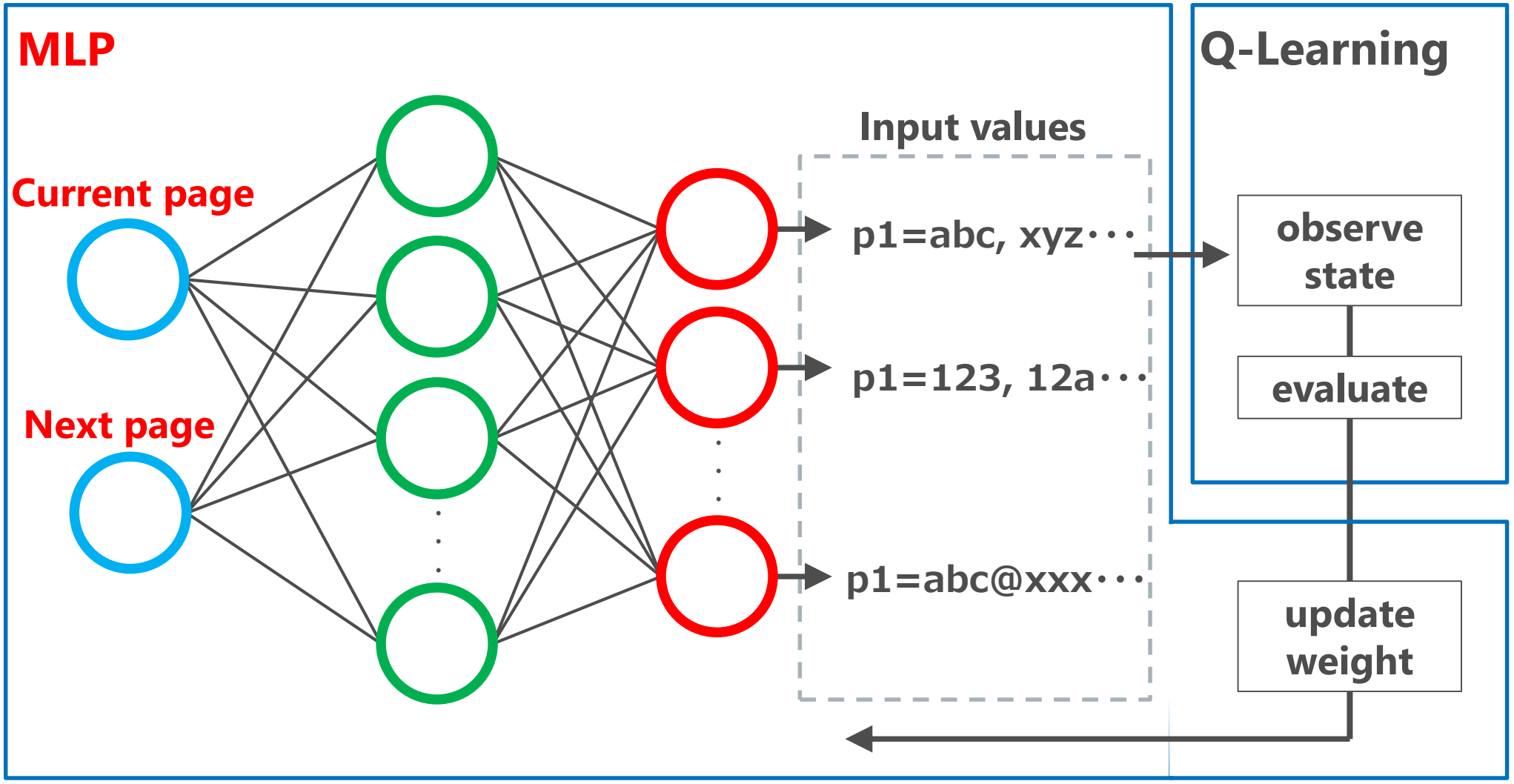
Patterns of input value

Pattern	Input value
Char	abc, abcdef, aBc, aBcdEf, ABCDEF …
Num	123, 12345, 4111111111111111 …
Char,Num	abc123, 123abc, aBc123, 1a2b3c …
Char,Symbol	abc!, abc!#$, abcdef!, abcdef!#$ …
Num,Symbol	123!, 123!#$, 12345!, !#$12345 …
Char,Num,Symbol	abc123!, 123abc!, abc!123, !#$%&a1 …
Mail address	abc@hoge.com, abc123@hoge.com …

Set a combination of input values to the parameter.

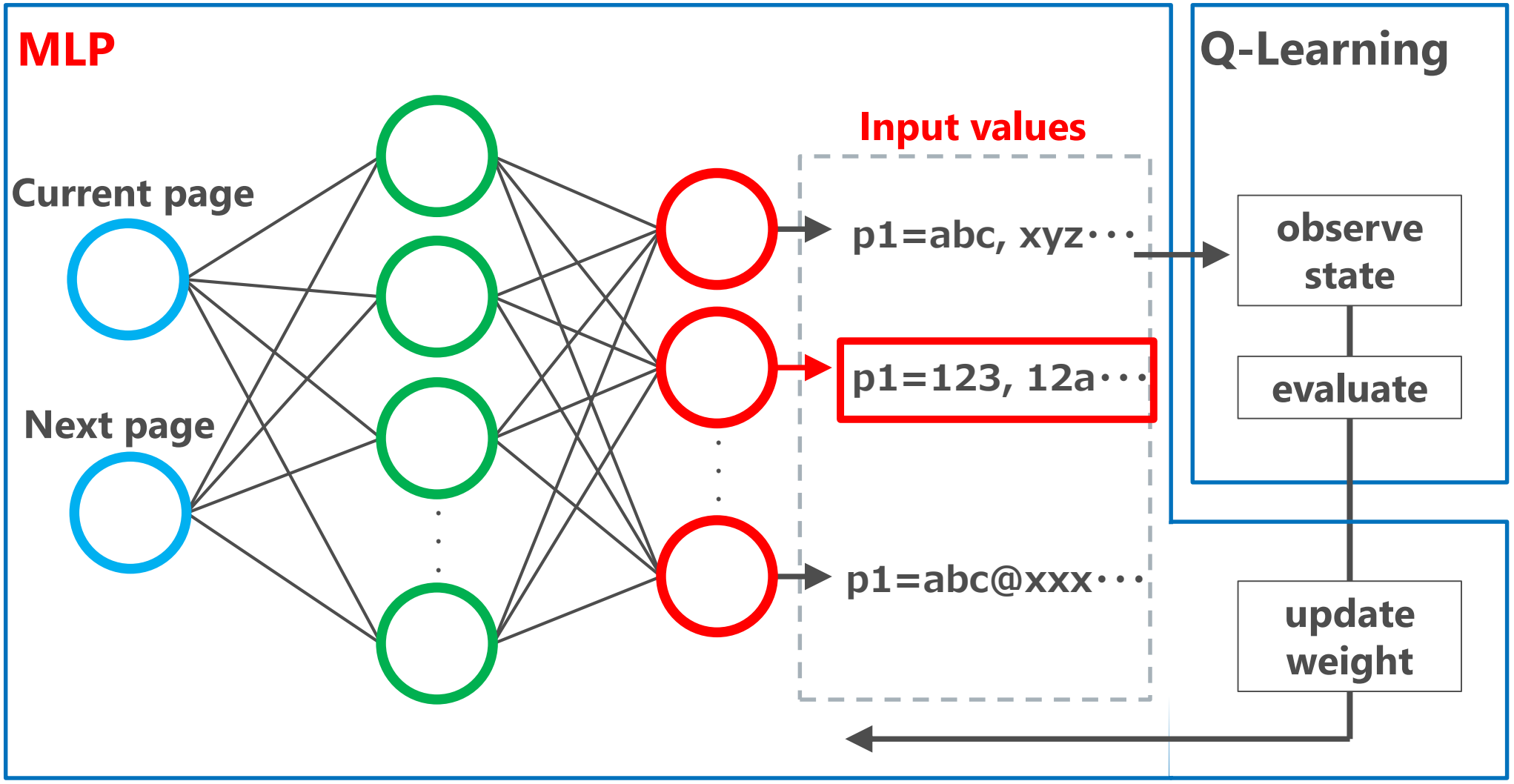
Target : INPUT TYPE ＝ text, password

Learning flow



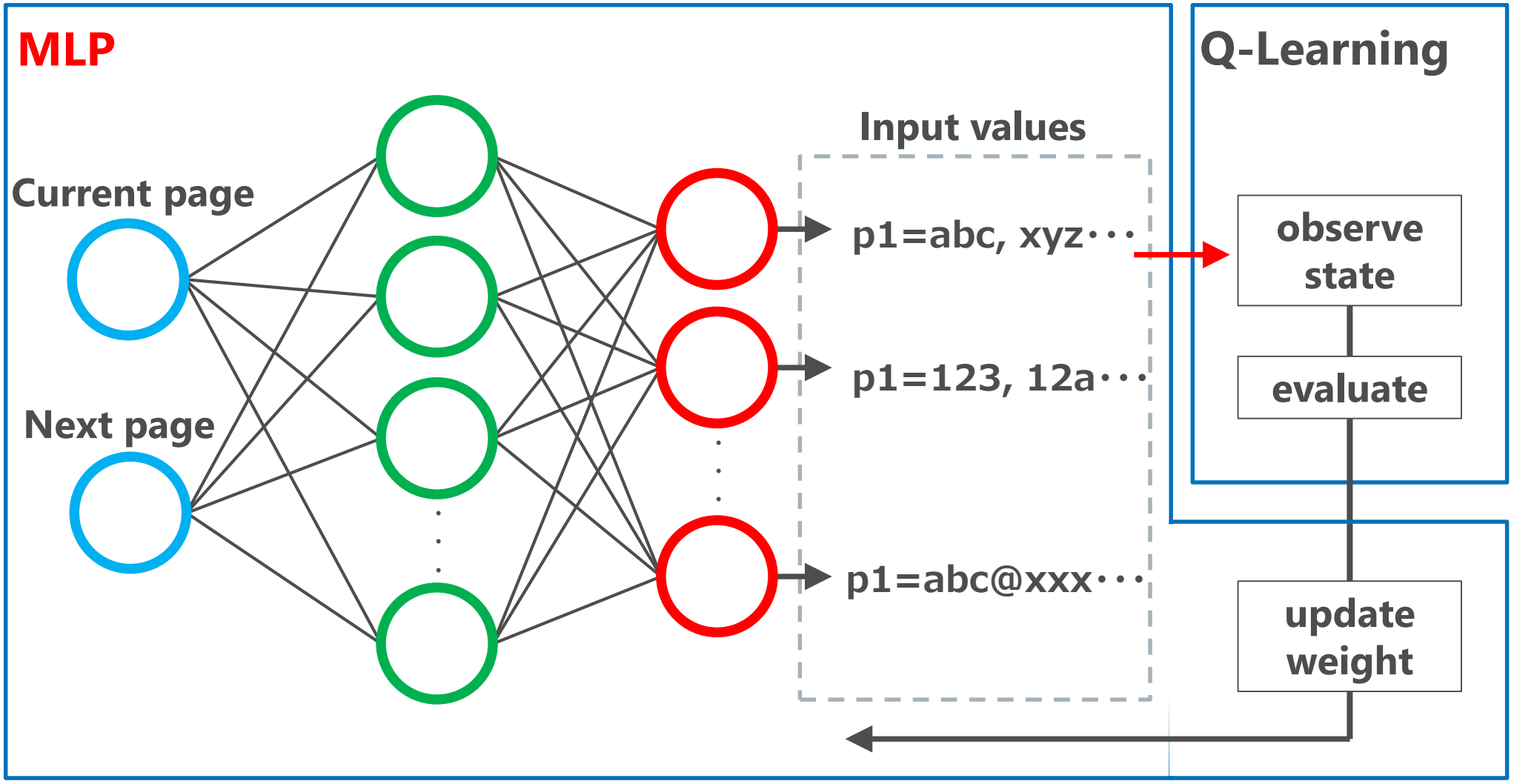
Inputs "current" and "next" page to MLP.

Learning flow



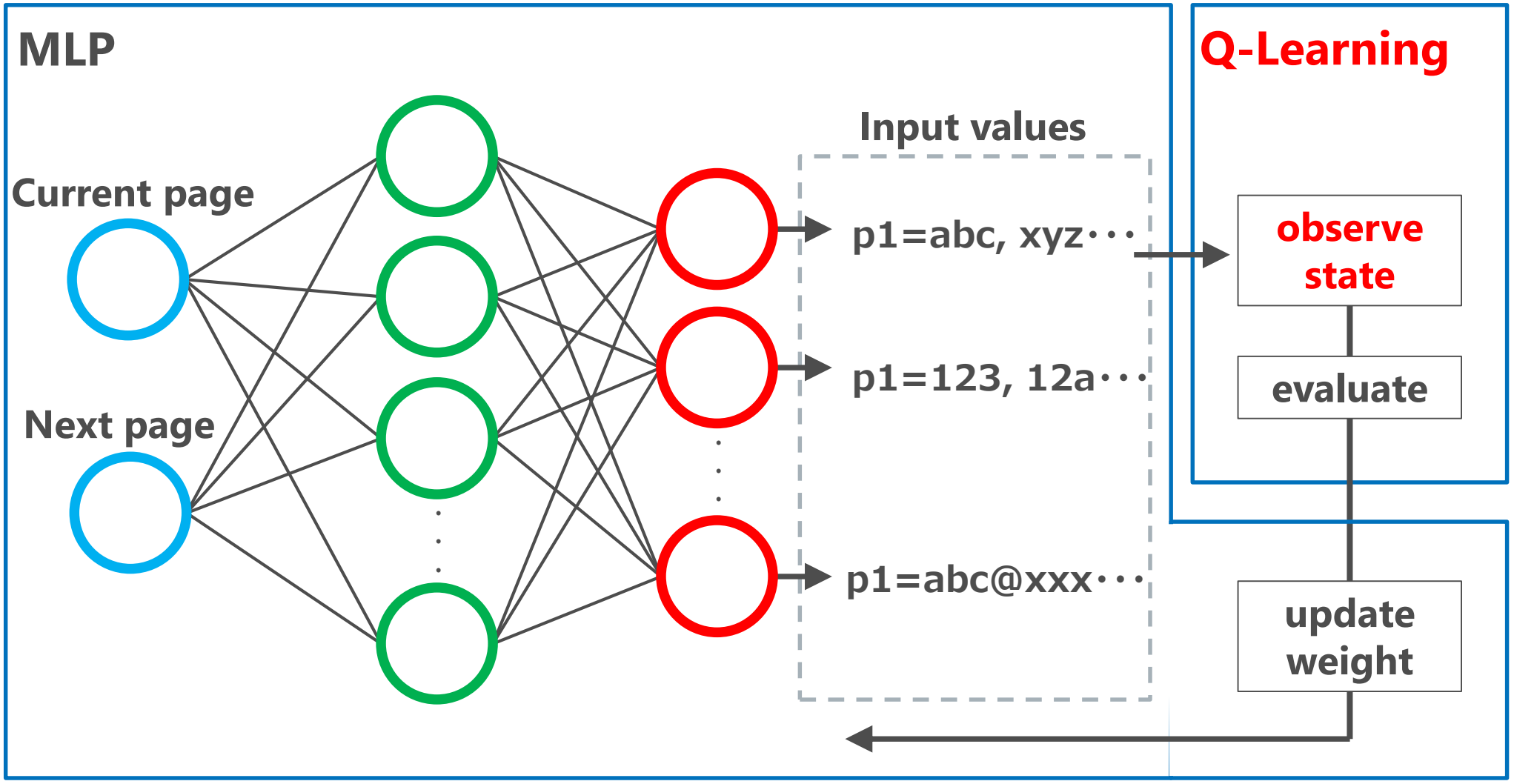
Selects any input pattern from the input of MLP.

Learning flow



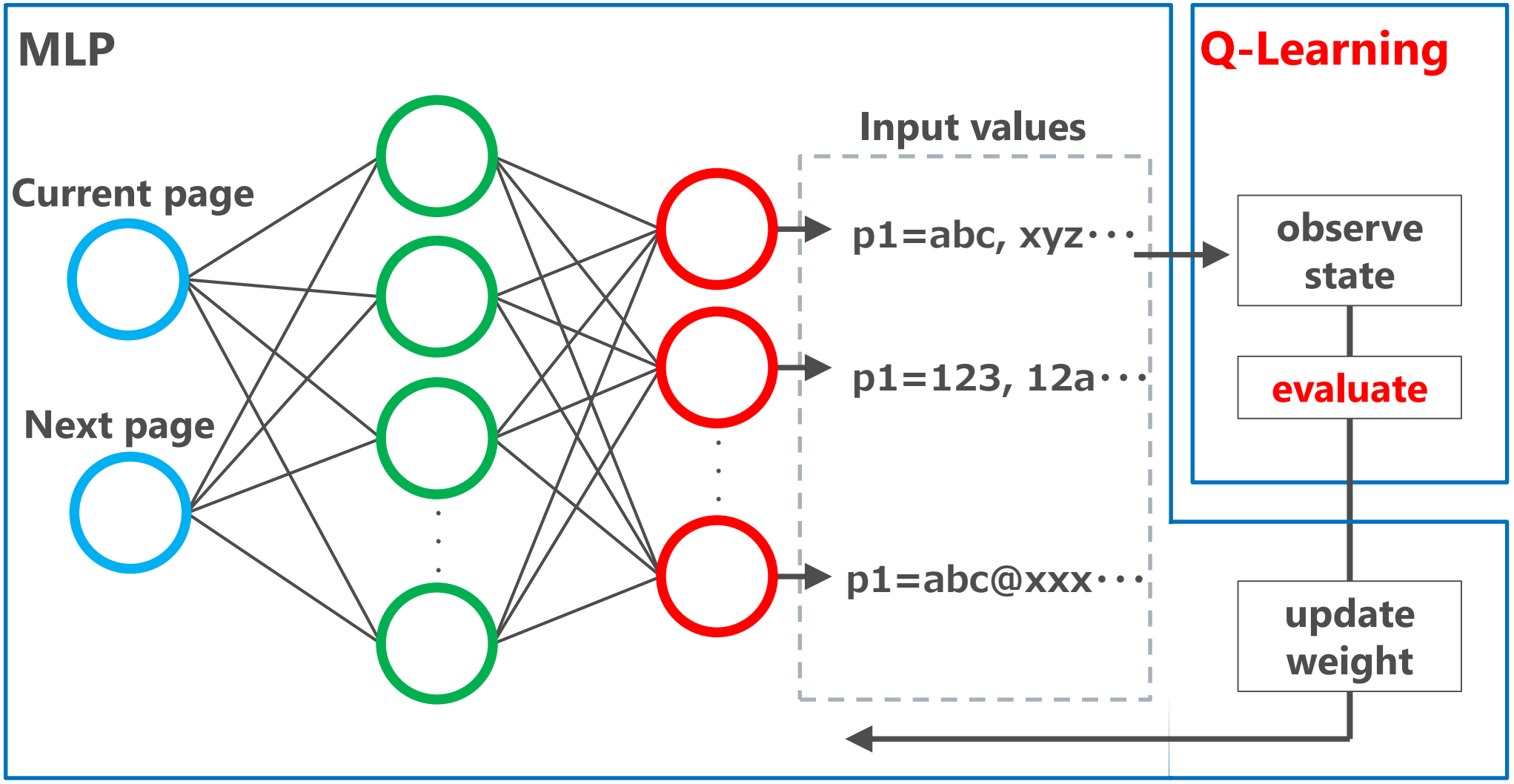
Attempts to crawl using the selected input pattern of MLP.

Learning flow



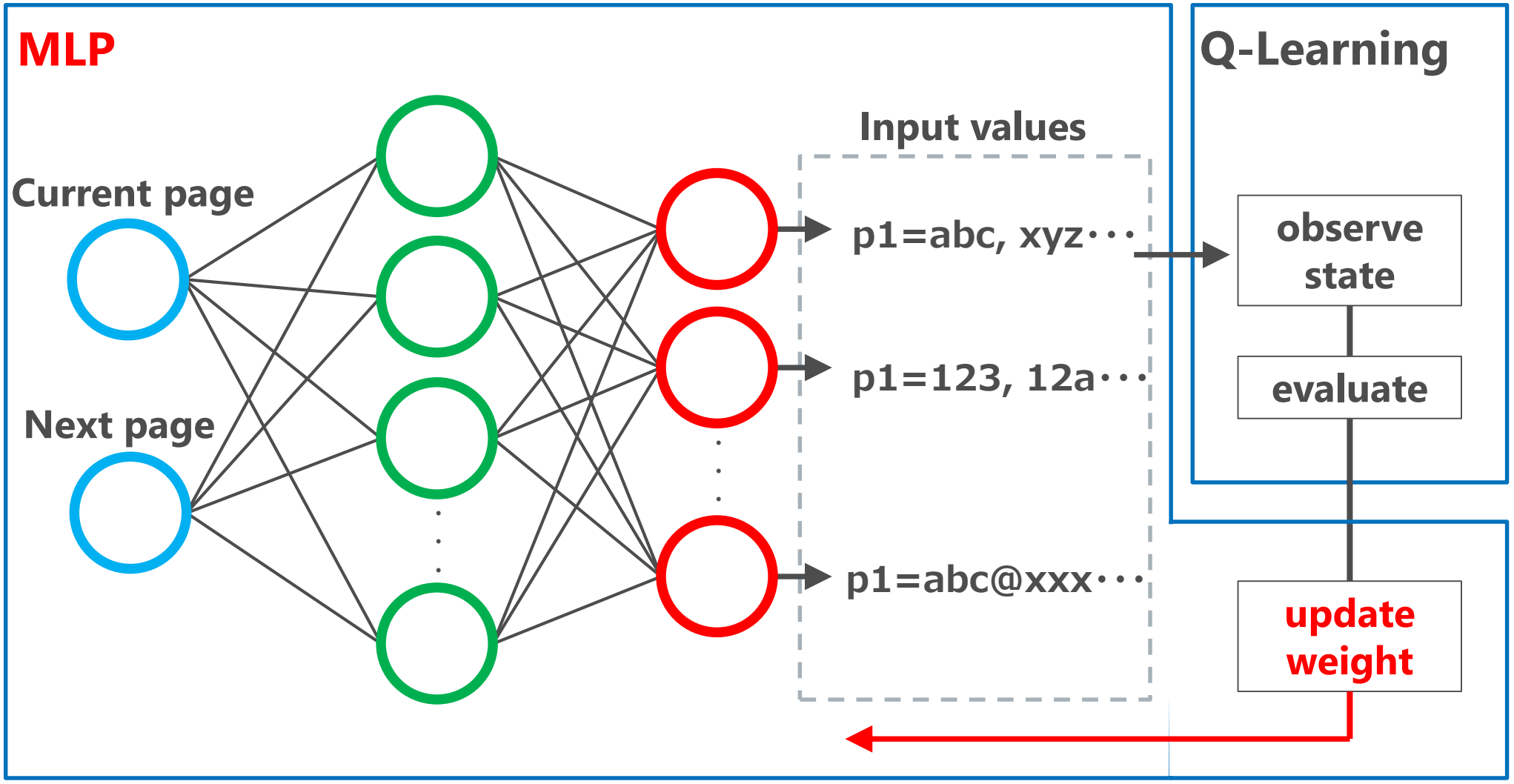
Observe the result of crawling.

Learning flow



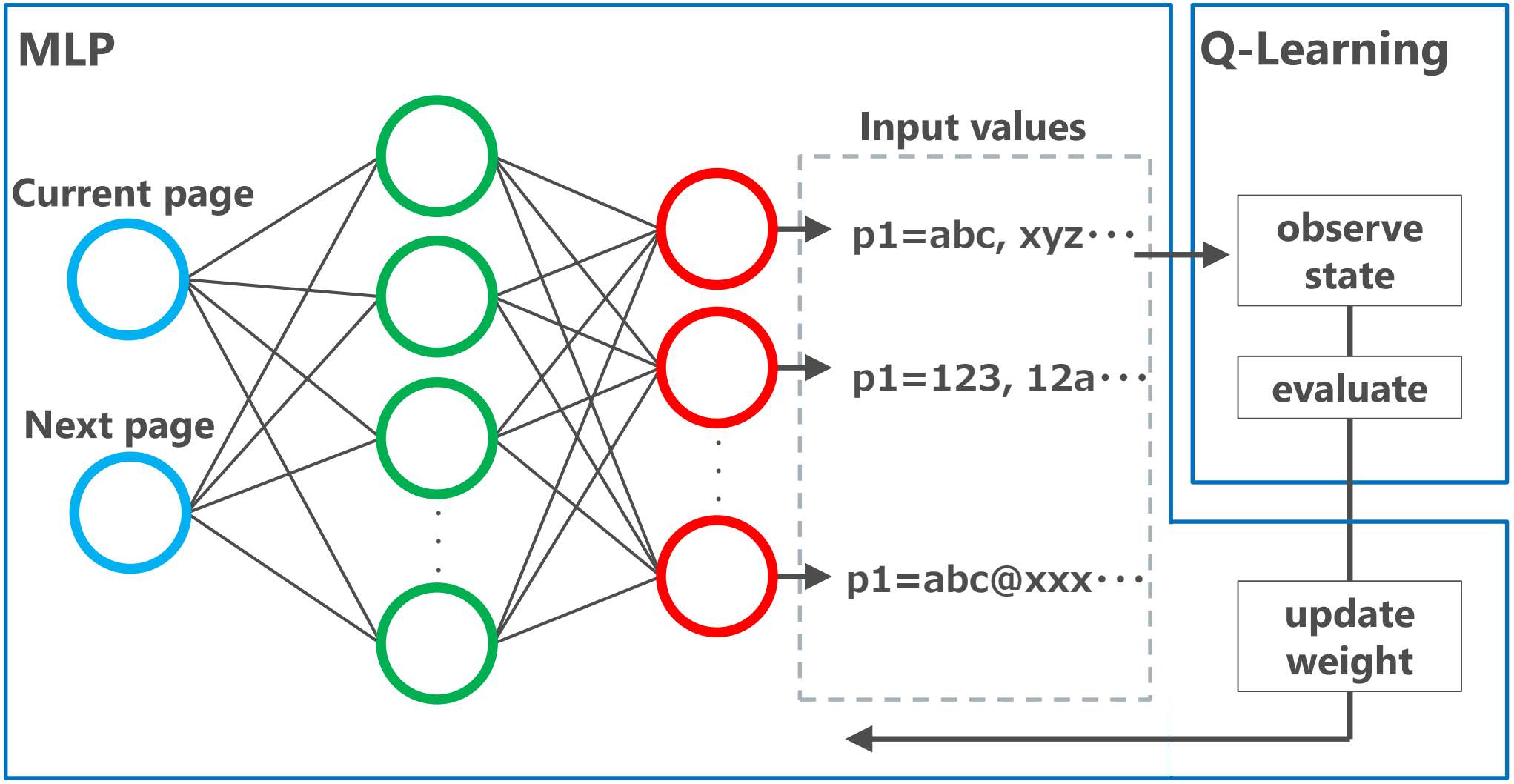
Gives a reward depending on the result of crawling.

Learning flow



Update the weight of MLP using backpropagation.

Learning flow



Learning can be completed **at approx. 300 attempts.**

Problem

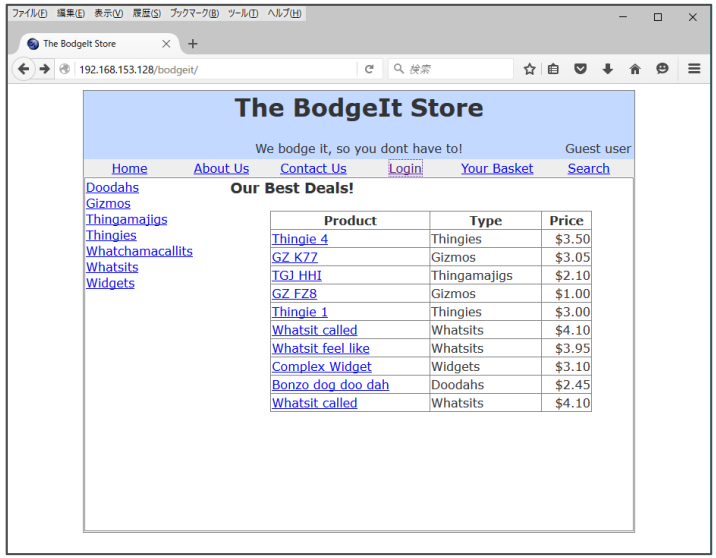
300 attempts ⇒ Inefficient

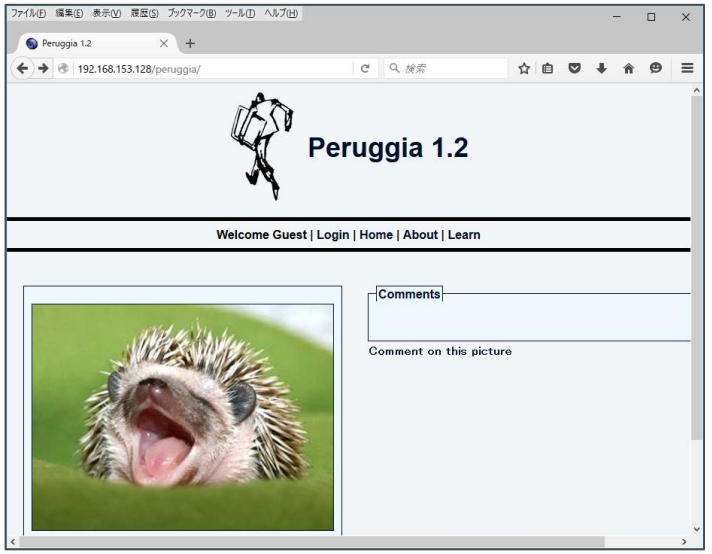
Solution

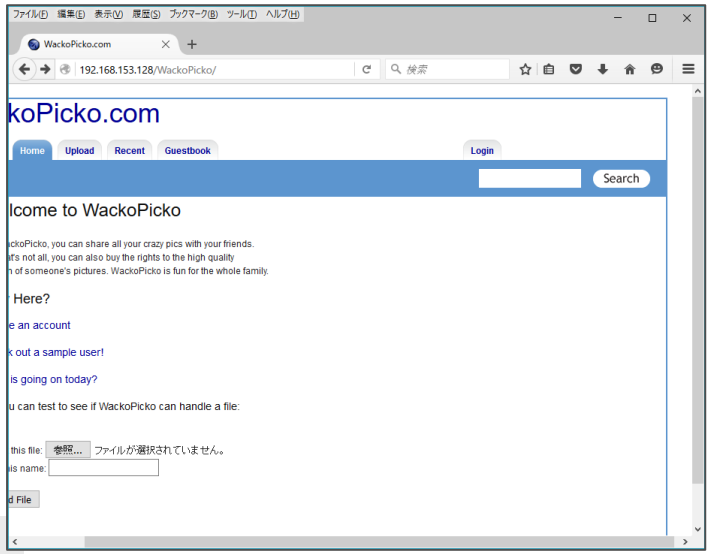
Pre-training.

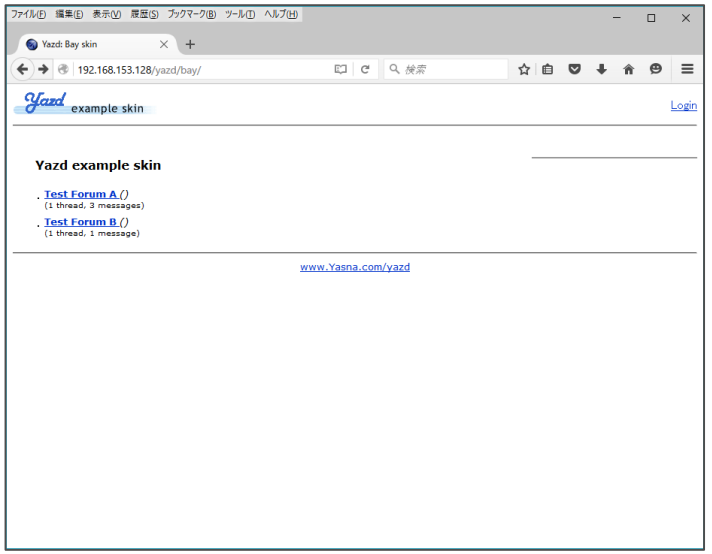
Web apps for pre-training（one example）.

OWASP Broken Web Apps

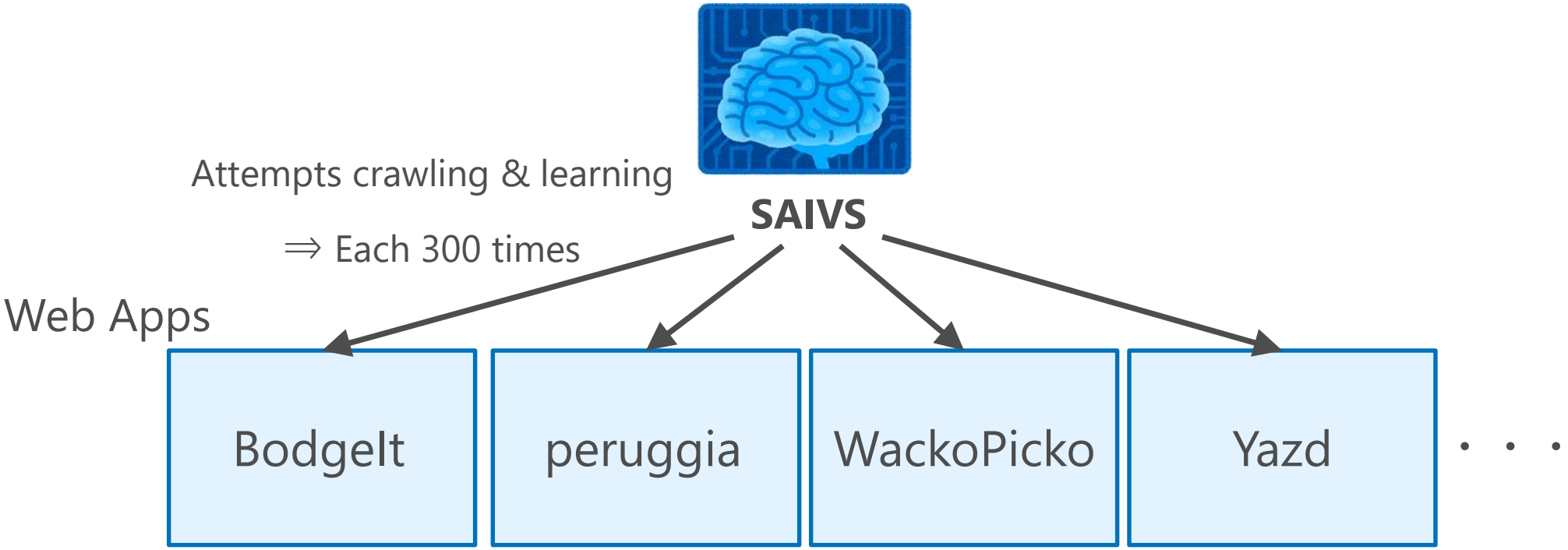








Training SAIVS



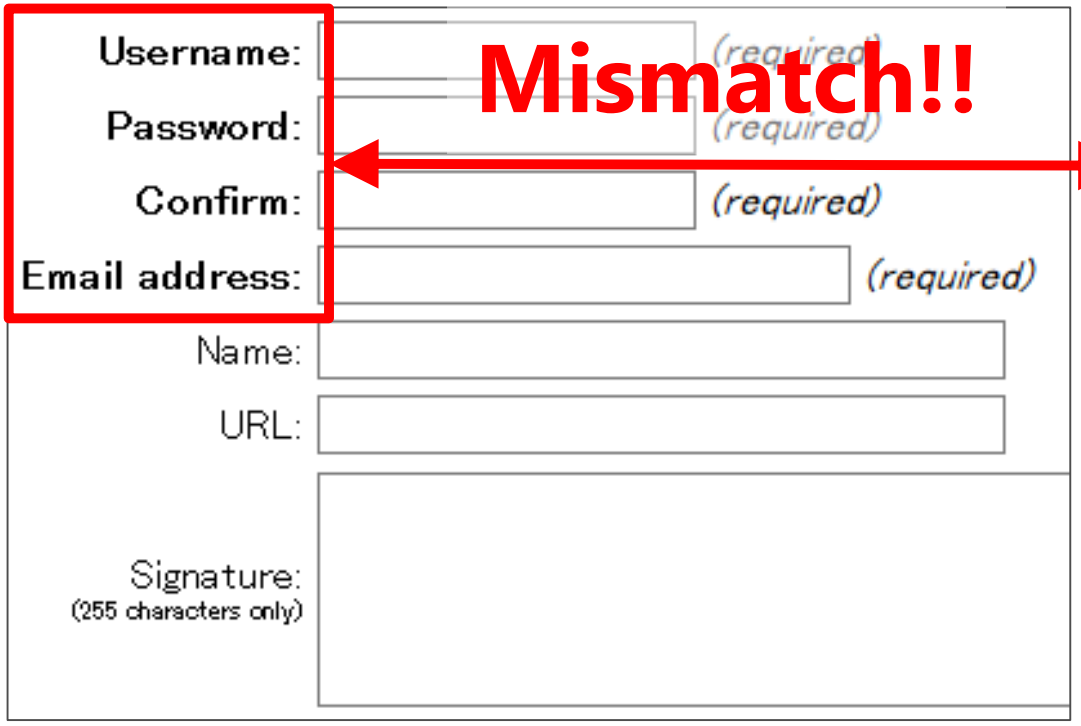
Achieves the correct input values for each form.

Achieved correct input values (one example).

Input form	Correct input value
ID	abc, abcdef, aBc, aBcdEf, ABCDEF …
Password	abc123!, 123abc!, abc!123, !#$%&a1 …
FirstName	abc, abcdef, aBc, aBcdEf, ABCDEF …
LastName	abc, abcdef, aBc, aBcdEf, ABCDEF …
Email address	abc@hoge.com, abc123@hoge.com …
Username	abc, abcdef, aBc, aBcdEf, ABCDEF …
Signature	abc, abcdef, aBc, aBcdEf, ABCDEF …

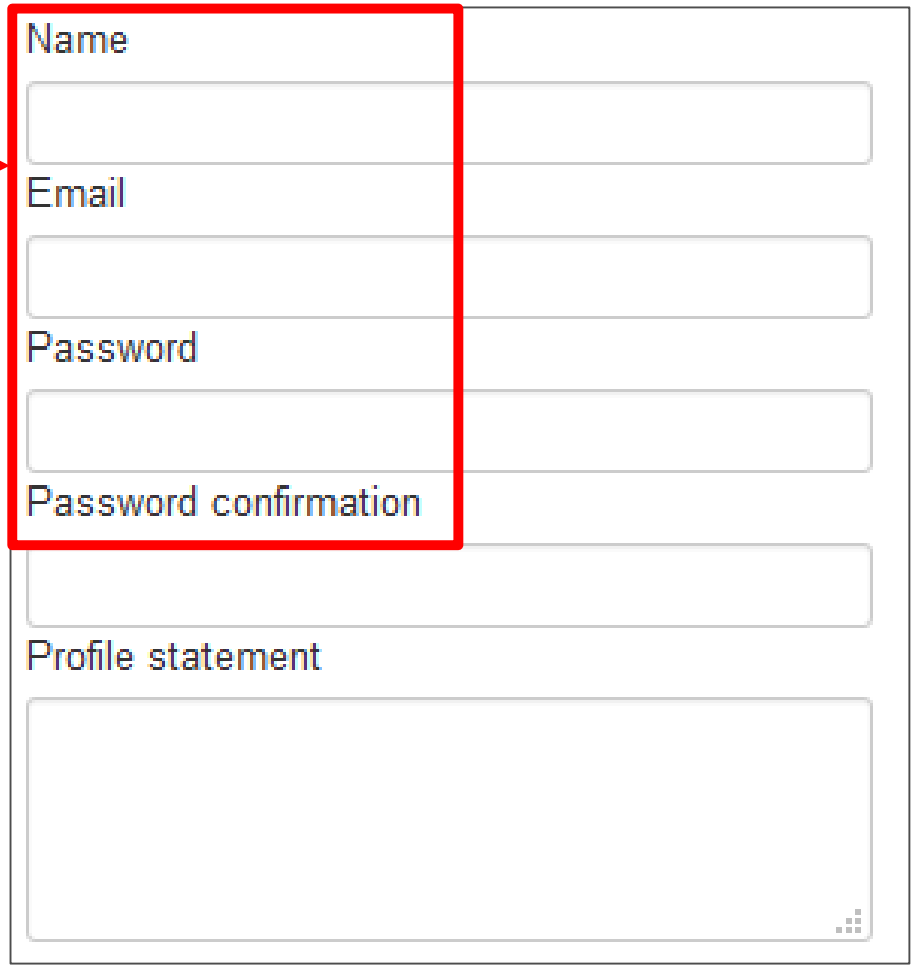
Problem：mismatch of form strings

Used in Training



Mismatch!!

Used in actual assessment



Solution

Calculate the

similarity of form strings.

Calculate similarity

Using word2vec.

What is word2vec ?

Calculation method for **similar words** in NLP.

A word is represented by a vector.

examples)

- Similarity between words

  Input : e-mail

word	cos distance
email	0.956302
mail	0.927386
reply	0.920610

- Adding and subtracting words

formula	answer
Iraq - Violence	Jordan
Human - Animal	Ethics
Japan – Tokyo + France	Paris

Example of the use of the similarity calculation

Calculate similarity using **words around the word of interest**.

Training data)

interpretation further. However, if anyone wishes to **discuss** this, I'm certainly willing (either **offline** - **e-mail** - or Stephen In **article bevans@carina.unm.edu** (Mathemagician) **writes**: Just what do gay people do that straight・・・

carries archives of old alt.atheism.moderated **articles** and assorted other files. For more information, **send mail** to **archive-server@mantis.co.uk** saying help **send** atheism/index and it will **mail** back a **reply**. mathew An・・・

send mail to **archive-server@mantis.co.uk** saying help **send** atheism/index and it will **mail** back a **reply**. mathew An Introduction to Atheism by Mathew. This **article** attempts to provide a general introduction・・・

The similarity of the "e-mail", "mail", "reply" is high.

Word2vec training data

The 20 Newsgroups data set.

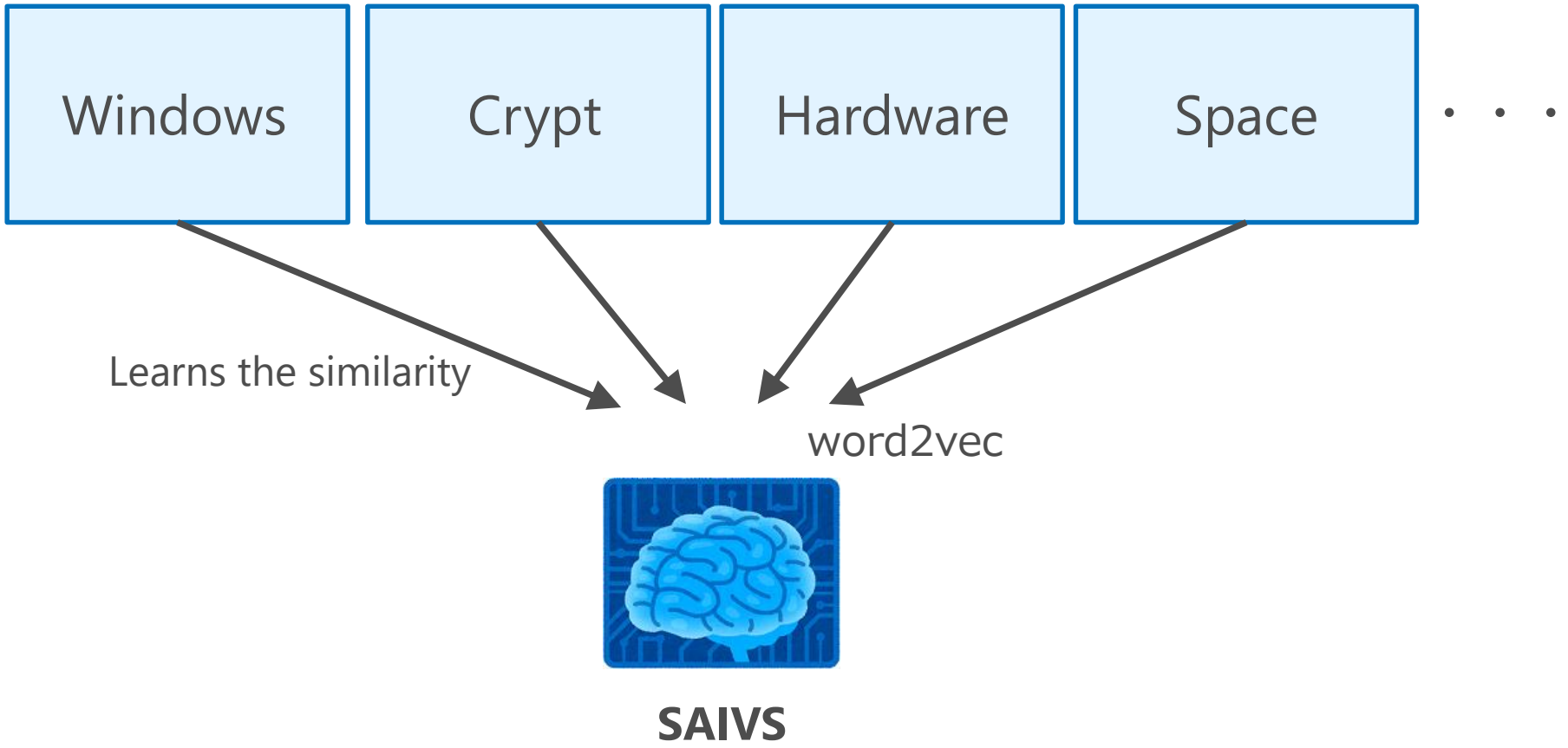
A collection of newsgroups : about **20,000** news.

examples of newsgroup)

- Computer（Graphics, MS-Windows, Hardware）
- Science（Cryptography, Electronics, Space）
- Hobby（Motorcycles, Baseball, Hockey）

Training SAIVS

The 20 Newsgroups data set



Learns the similarity between the words.

Learned similarities in the training

Target ： e-mail

candidate	similarity
email	0.956302
mail	0.927386
E-mail	0.900249
address	0.893337
reply	0.865438
contact	0.846801
message	0.792930
chat	0.754903
newsgroup	0.747636

Target ：name

candidate	similarity
names	0.962508
username	0.939661
nickname	0.933694
naming	0.898254
surname	0.863966
initials	0.861093
firstname	0.849641
lastname	0.847604
title	0.782467

Target ：website

candidate	similarity
homepage	0.794415
blog	0.752945
site	0.708534
webpage	0.701838
portal	0.701374
forum	0.692067
com	0.641086
archive	0.537914
org	0.531096

Extracts candidate words of Top 10.

Learned similarities in the training

Target ： e-mail

candidate	similarity
email	0.956302
mail	0.927386
E-mail	0.900249
address	0.893337
reply	0.865438
contact	0.846801
message	0.792930
chat	0.754903
newsgroup	0.747636

Target ：name

candidate	similarity
names	0.962508
username	0.939661
nickname	0.933694
naming	0.898254
surname	0.863966
initials	0.861093
firstname	0.849641
lastname	0.847604
title	0.782467

Target ：website

candidate	similarity
homepage	0.794415
blog	0.752945
site	0.708534
webpage	0.701838
portal	0.701374
forum	0.692067
com	0.641086
archive	0.537914
org	0.531096

Selects a candidate word that **matches the form string of training**.

Learned similarities in the training

Target ： e-mail

candidate	similarity
email	0.956302
mail	**0.927386**
E-mail	0.900249
address	0.893337
reply	0.865438
contact	0.846801
message	0.792930
chat	0.754903
newsgroup	0.747636

⇒ **abc@hoge.com**

Target ：name

candidate	similarity
names	0.962508
username	0.939661
nickname	**0.933694**
naming	0.898254
surname	0.863966
initials	0.861093
firstname	0.849641
lastname	0.847604
title	0.782467

⇒ **aBcdEf**

Target ：website

candidate	similarity
homepage	0.794415
blog	0.752945
site	0.708534
webpage	**0.701838**
portal	0.701374
forum	0.692067
com	0.641086
archive	0.537914
org	0.531096

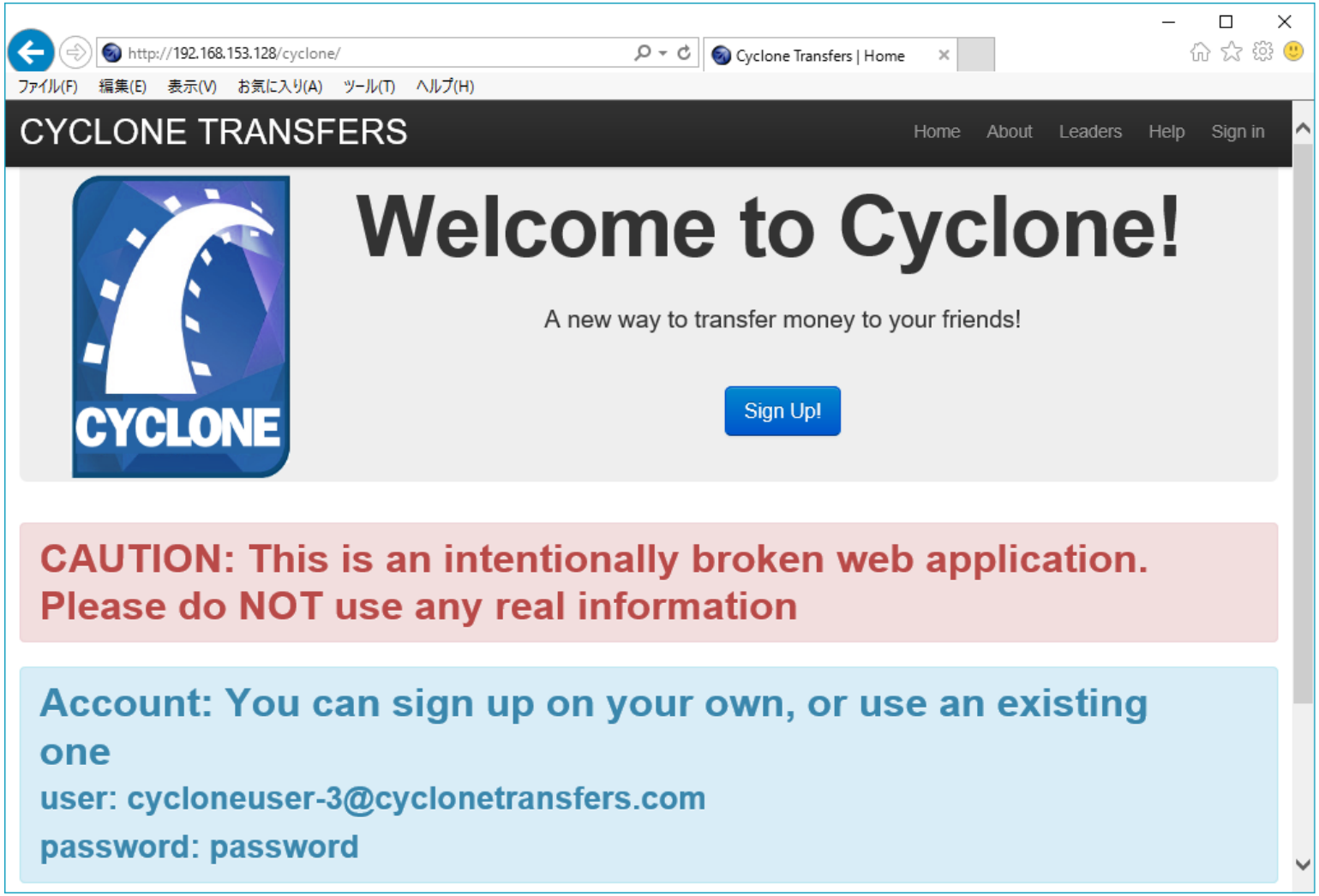
⇒ **http://hoge.com**

3 requirements for crawling

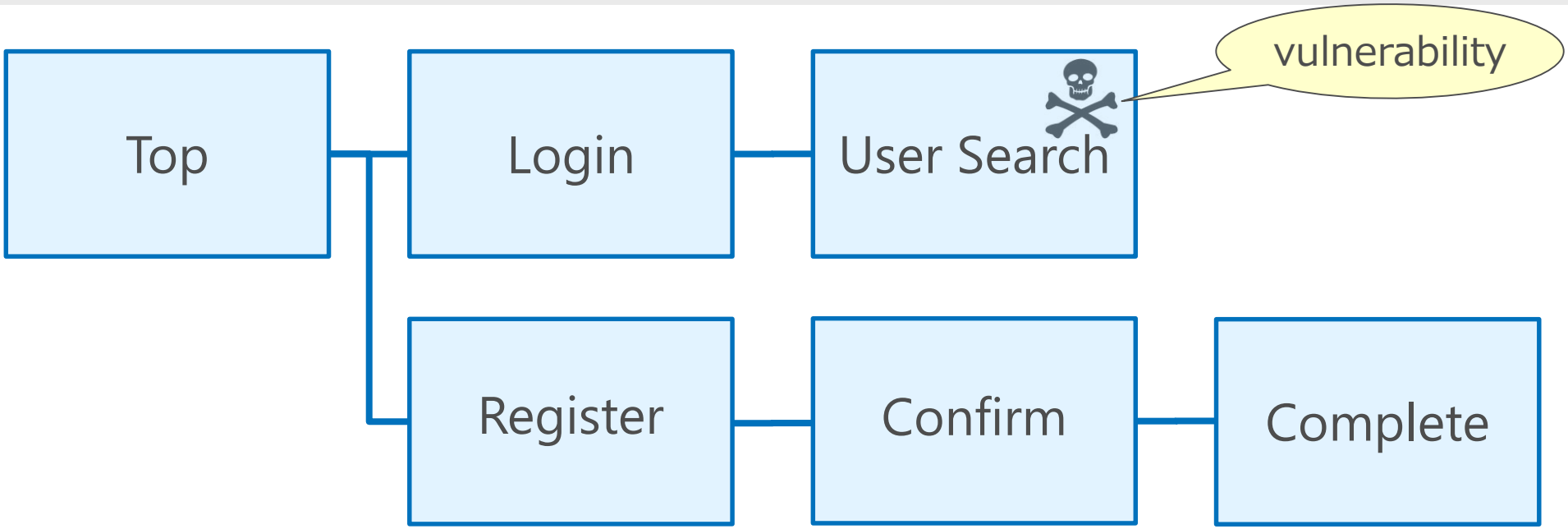
- Recognize the page type (✓).
- Recognize the success/fail of crawling (✓).
- Input correct values in the forms (✓).

Demonstration of crawling

Target：OWASP Broken Web Apps **Cyclone**

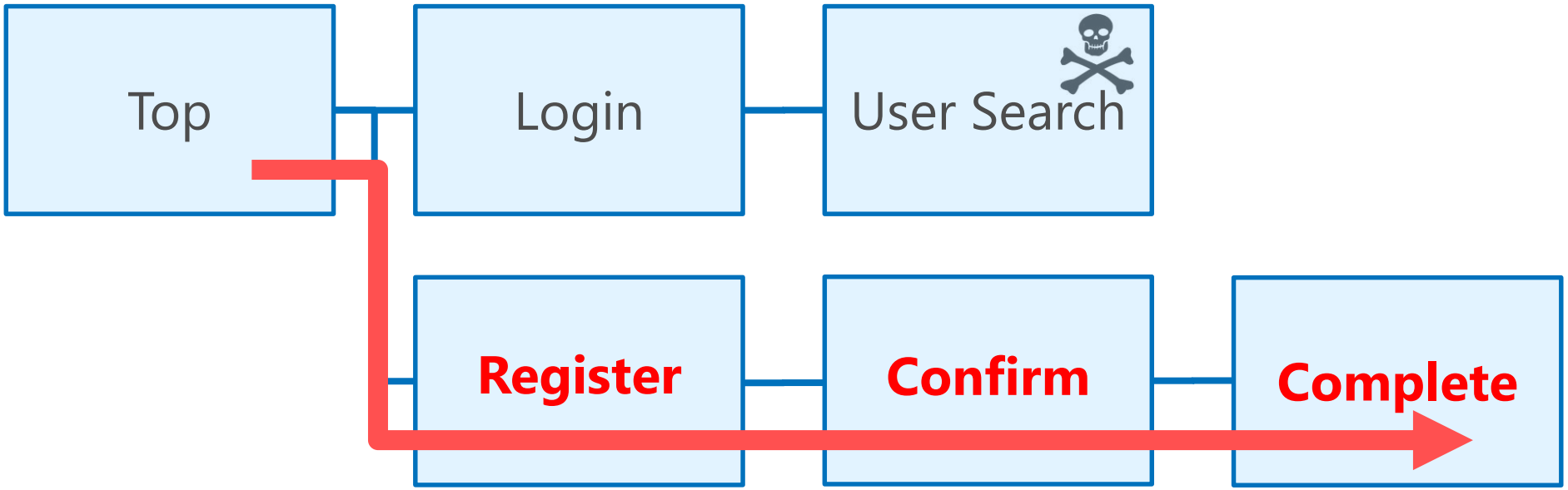


Demonstration of crawling



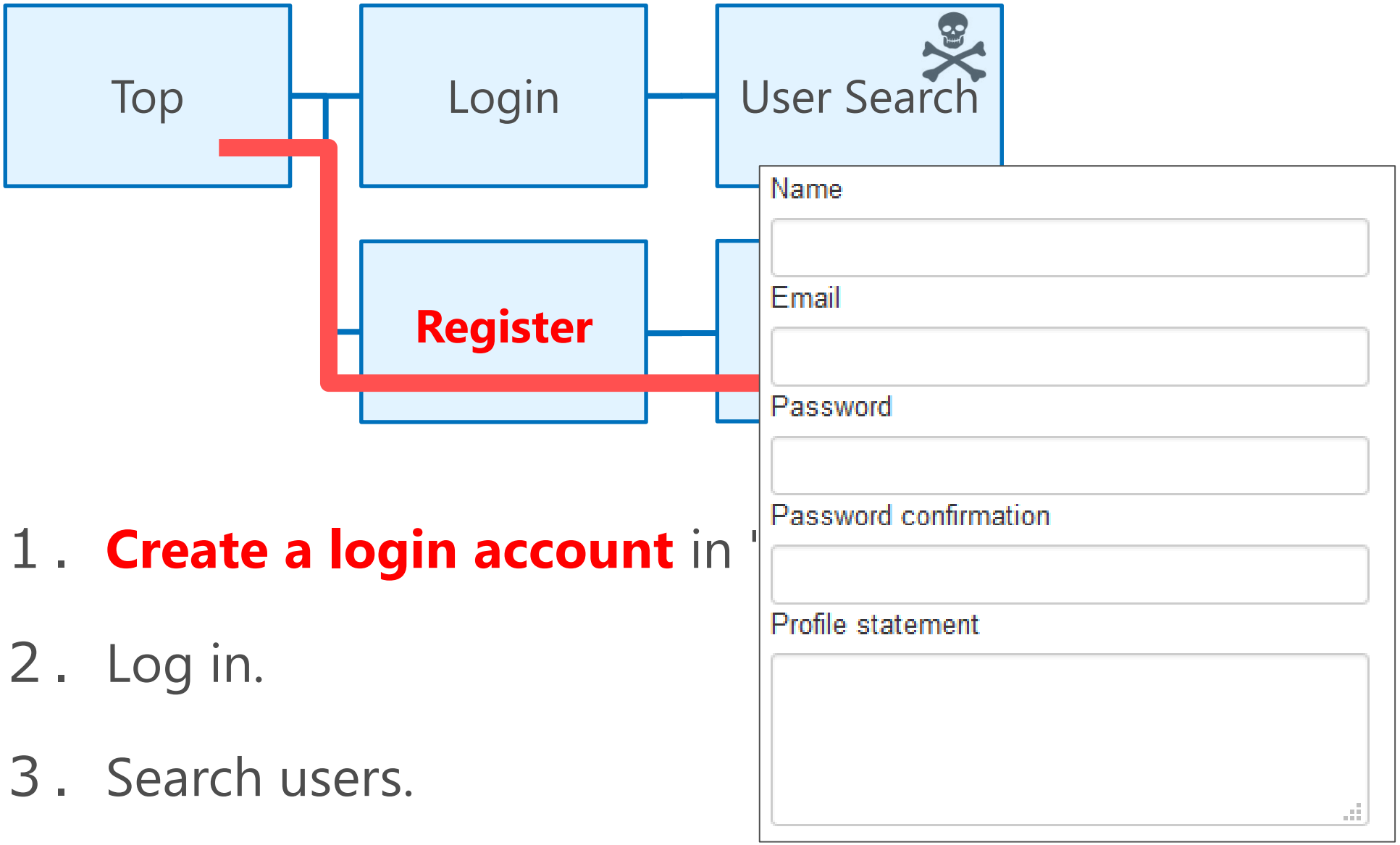
1. Create a login account in "register".
2. Log in.
3. Search users.

Demonstration of crawling



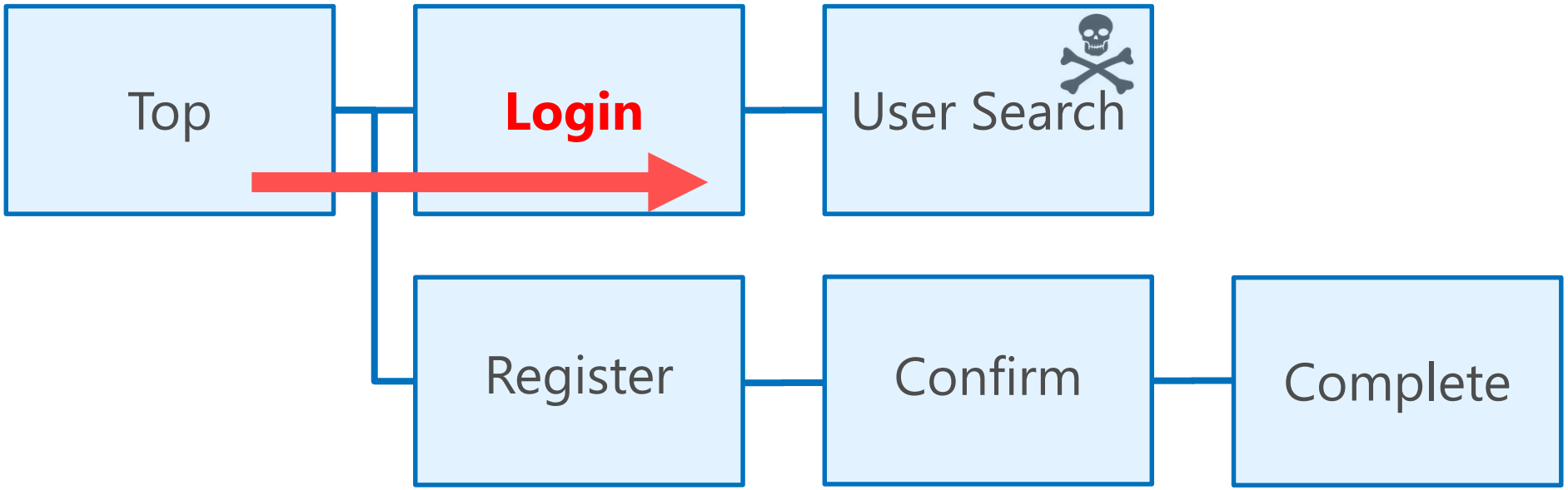
1. **Create a login account** in "register".
2. Log in.
3. Search users.

Demonstration of crawling



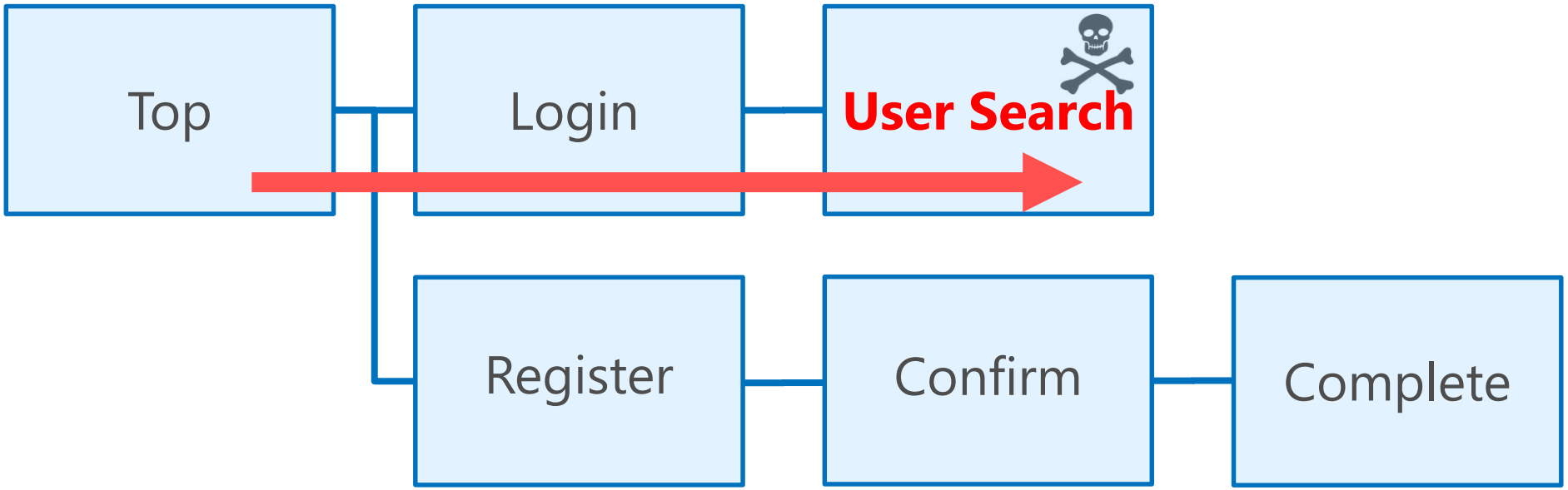
1. **Create a login account** in '
2. Log in.
3. Search users.

Demonstration of crawling



1. Create a login account in "register".
2. **Log in**.
3. Search users.

Demonstration of crawling



1. Create login account in "register".
2. Log in.
3. **Search users**.

Demonstration movie of crawling

https://www.youtube.com/watch?v=aXw3vgXbl1U

Abilities of SAIVS

- Crawls web apps. (✓)
- **Detects vulnerabilities.**

Objective of this research

To detect vulnerabilities with **less trouble.**

There are many web application vulnerabilities...

Target vulnerability of this research

Reflected Cross-Site Scripting

(RXSS)

Example of RXSS Pattern 1：very vulnerable web app

http://xxx/case3/?input=testData

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html">
<title>Case 3 - RXSS</title>
</head>
<body>
<input type="text" value="testData">
</body>
</html>
```

Input value is echoed in the **"VALUE" attribute of the INPUT tag**.

Example of RXSS Pattern 1：attacking a web app

http://xxx/case3/?input="/><script>alert('XSS');</script>

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html">
<title>Case 3 - RXSS</title>
</head>
<body>
<input type="text"
value=""/><script>alert('XSS');</script>">
</body>
</html>
```

JavaScript is inserted in the HTML syntax.

Example of RXSS Pattern 2：a little more secure web app

http://xxx/case4/?input="/><script>alert('XSS');</script>

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html">
<title>Case 4 - RXSS</title>
</head>
<body>
<input type="text" value="/> alert('XSS');">
</body>
</html>
```

SCRIPT tag is sanitized by the web app.

Example of RXSS Pattern 2：attacking web app

http://xxx/case4/?input=“onmouseout=alert(‘XSS')”

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html">
<title>Case 4 - RXSS</title>
</head>
<body>
<input type="text" value=""onmouseout=alert('XSS')"">
</body>
</html>
```

JavaScript is inserted to **avoid sanitization.**

3 requirements for detecting RXSS

- Understanding HTML syntax.
- Understanding JavaScript syntax.
- Avoiding sanitization.

How do you achieve them ?

Our approach

Reverse engineering the human brain.

Our approach

Each thinking pattern

implemented in machine learning.

3 requirements for detecting RXSS

- **Understanding HTML syntax.**
- **Understanding JavaScript syntax.**
- Avoiding sanitization.

Using LSTM

What is LSTM（Long-Short Term Memory）？

It can **learn chronological data.**

It can learn dependencies in short & long term data.

examples)

- Machine translation.
- Text generation. (novel, lyrics, source code)
- Sound generation. (music, human voice)

What is LSTM（Long-Short Term Memory） ?

It can learn chronological data.

It can learn dependencies in short & long term data.

examples)

- Machine translation.
- **Text generation.** (novel, Lyrics, source code)
- Sound generation. (music, human voice)

Example of the use of the text generation.

Generation of source code.（from Andrej Karpathy blog）

Training Data

```
static int indicate_policy(void)
{
  int error;
  if (fd == MARN_EPT) {
    if (ss->segment < mem_total)
      unblock_set_blocked();
    else
      ret = 1;
    goto bail;
  }
  segaddr = in_SB(in.addr);
  selector = seg / 16;
. . .
```

Generated source code by LSTM

```
static void settings(struct *tty)
{
  if (tty == tty)
    disable_single_st_p(dev);
    pci_disable_spool(port);
    return 0;
}

static void command(struct seq_file *m)
{
  int column = 32 << (cmd[2] & 0x80);
  if (state)
. . .
```

It can generate a code from the starting point (seed).

Why LSTM ?

http://xxx/textarea1?in=xxx

```
<!doctype html><html><head><title>Reflected XSS in
textarea (textarea1)</title></head><body>
<H2>Textarea injection test</H2>
<FORM>
<textarea name="in" rows="5" cols="60">xxx</textarea>
 . . .
```

Input value is echoed in the **TEXTAREA tag**.

Why LSTM ?

http://xxx/textarea1?in=<script>alert('XSS');</script>

```
<!doctype html><html><head><title>Reflected XSS in
textarea (textarea1)</title></head><body>
<H2>Textarea injection test</H2>
<FORM>
<textarea name="in" rows="5" cols="60">
<script>alert('XSS');</script></textarea>
 . . .
```

JavaScript doesn't run.

⇒ **Simple script injection is not allowed.**

Why LSTM ?

http://xxx/textarea1?in=xxx

```
<!doctype html><html><head><title>Reflected XSS in
textarea (textarea1)</title></head><body>
<H2>Textarea injection test</H2>
<FORM>
<textarea name="in" rows="5" cols="60">xxx</textarea>
 . . .
```

"**Is JavaScript inserted after closing the TEXTAREA tag ?**"

Why LSTM ?

http://xxx/textarea1?in=</textarea><script>alert('XSS');</script>

```
<!doctype html><html><head><title>Reflected XSS in
textarea (textarea1)</title></head><body>
<H2>Textarea injection test</H2>
<FORM>
<textarea name="in" rows="5"
cols="60">
```
`</textarea><script>alert('XSS');</script>`
```
</t
extarea>
 . . .
```

Understands the context of HTML and inserts a script.

⇒ **JavaScript run**.

How can LSTM be learned ?

Learning data for LSTM (HTML)

20,000 pages of HTML syntax. (about 12,000 types)

```
<abbr class="" data-utime="" title=""></abbr>
<abbr class='' title=''></abbr>
<abbr data-utime='' title=''></abbr>
 . . .
<input name="" type="" value=""/>
<input alt="" id="" onclick="" src="" type=""/>
<input alt='' id="" src='' type=''/>
<input alt='' name='' src='' type=''/>
 . . .
<video autoplay="" loop="" muted=""></video>
<video class="" height="" id="" width=""></video>
<video src='' tabindex=''></video>
<video src=''></video>
```

Learning data for LSTM (JavaScript)

10,000 pages of JavaScript syntax.

```
_satellite.pageBottom();']
    (function(window) {
      var _gaq = _gaq || [];
      var methods = ['log', 'dir', 'trace'];
      if (typeof window.console === 'undefined') {
        window.console = {};
      }
      for (var i in methods) {
        if (!(methods[i] in window.console)) {
          window.console[methods[i]] = function() {};
        }
      }
    }(window));
```

Generated syntax by learned LSTM (one example)

Seed	Generated syntax
<textarea cols="60">**xxx**	**</textarea>**
<!– mbsdtest **xxx**	**-->**
<input type="" value="**xxx**	**">**
var hoge = ['log', '**xxx**	**red']¥r¥n**
/* mbsdtest **xxx**	***/**
function(){ **xxx**	**}¥r¥n**

It can **generate a syntax from the seeds.**

3 requirements for detecting RXSS

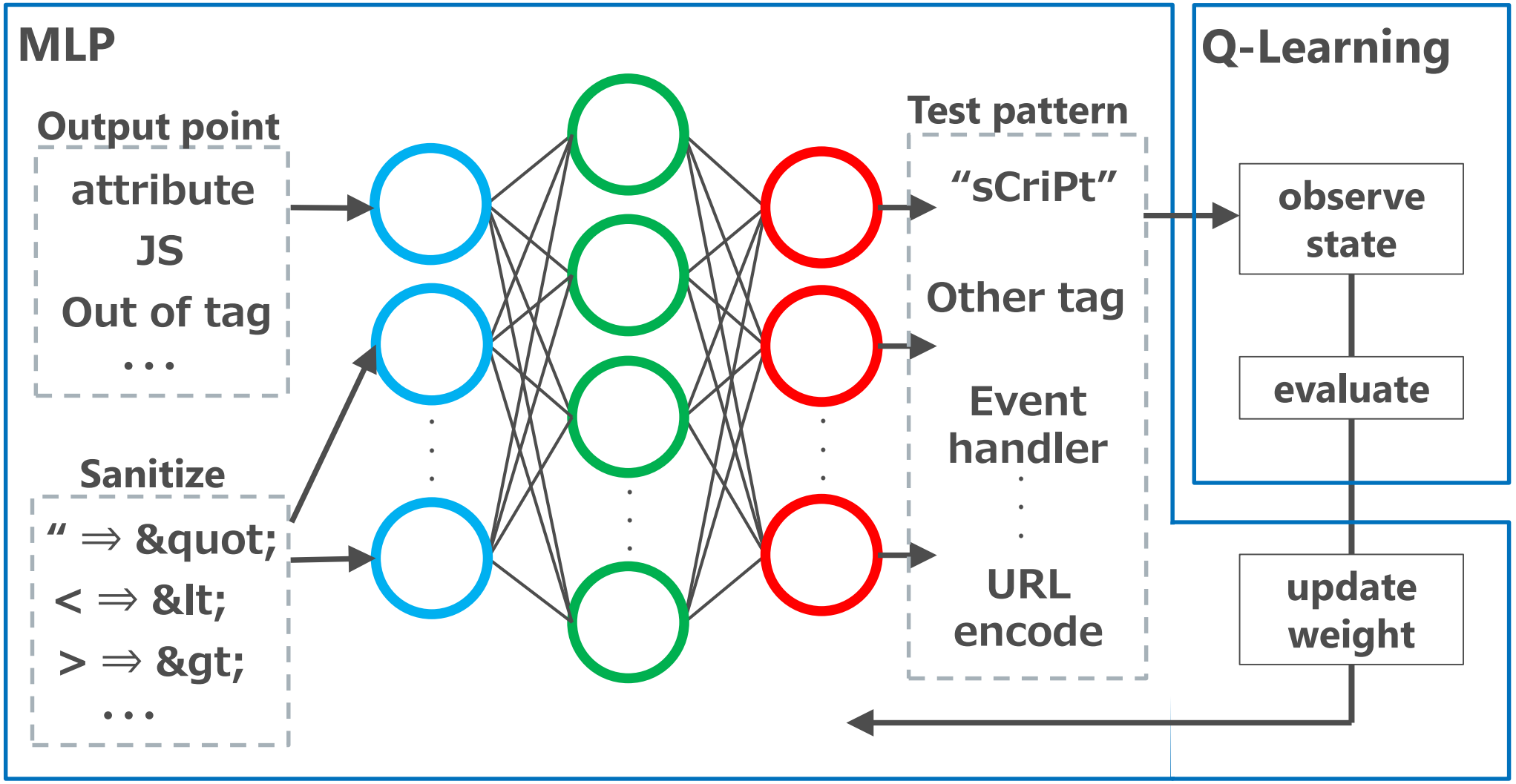
- Understanding HTML syntax (✓).
- Understanding JavaScript syntax (✓).
- **Avoiding sanitization.**

Avoidance of sanitization by machines

Using Multilayer perceptron &

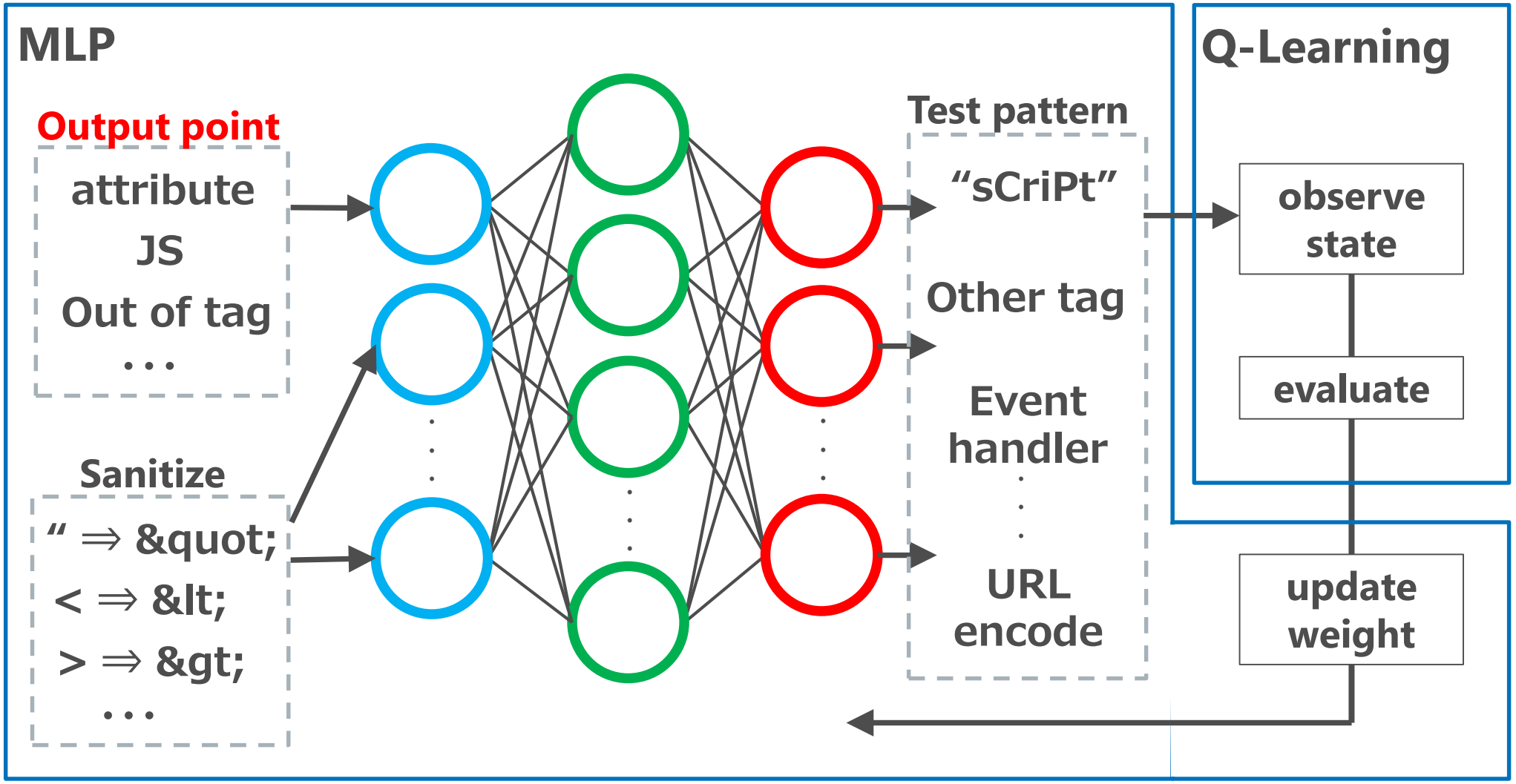
Q-Learning.

Model to avoid sanitization



Learns the pattern of avoiding sanitization while attempting to test.

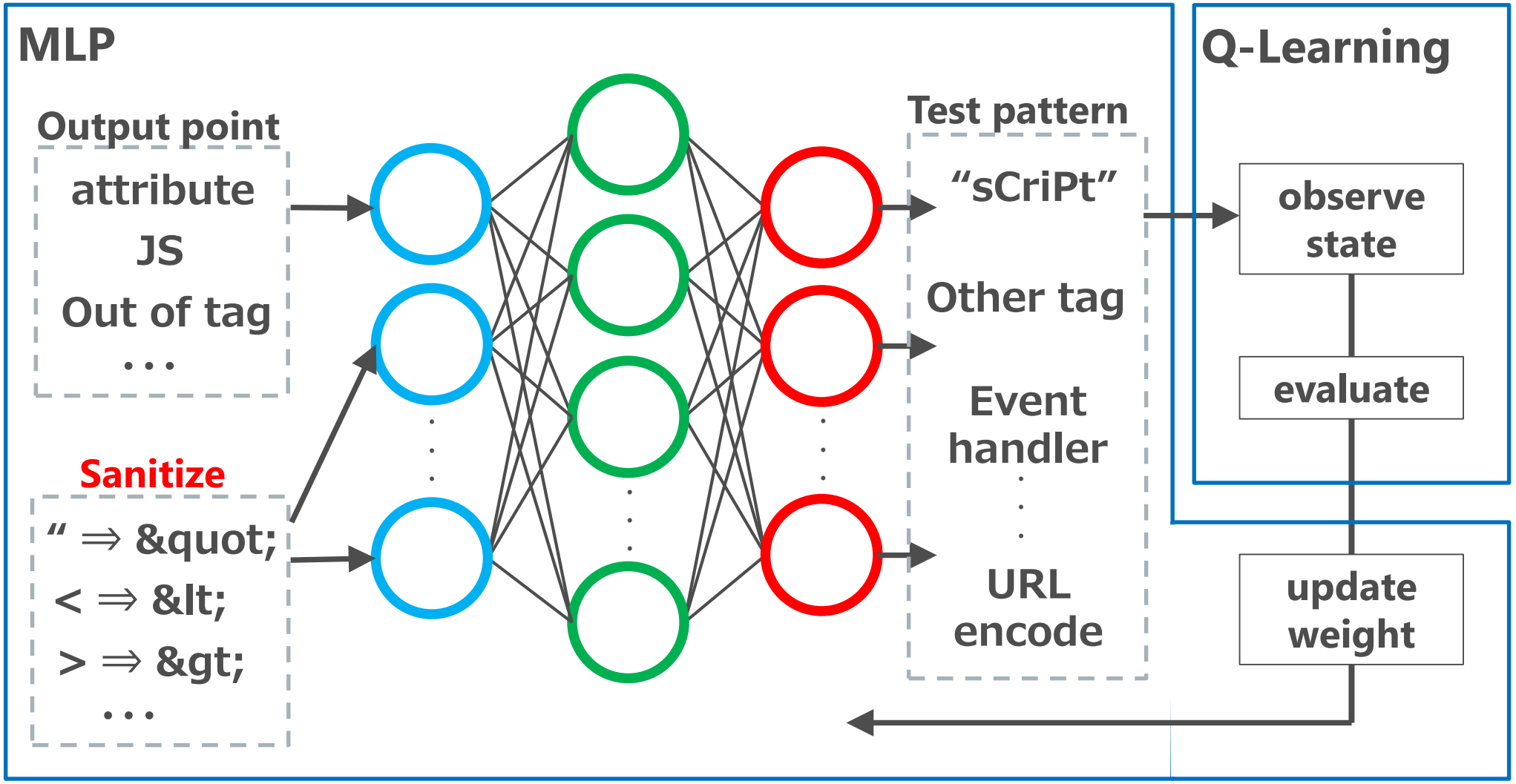
Model to avoid sanitization.



Patterns of Output point

Output point	Example
attribute value : 「"」	<～value="**xxx**">
attribute value : 「'」	<～value='**xxx**'>
attribute value : noquote	<～value=**xxx**>
JavaScript	<script>**xxx**</script>
outside of HTML tags	<～>**xxx**</～>

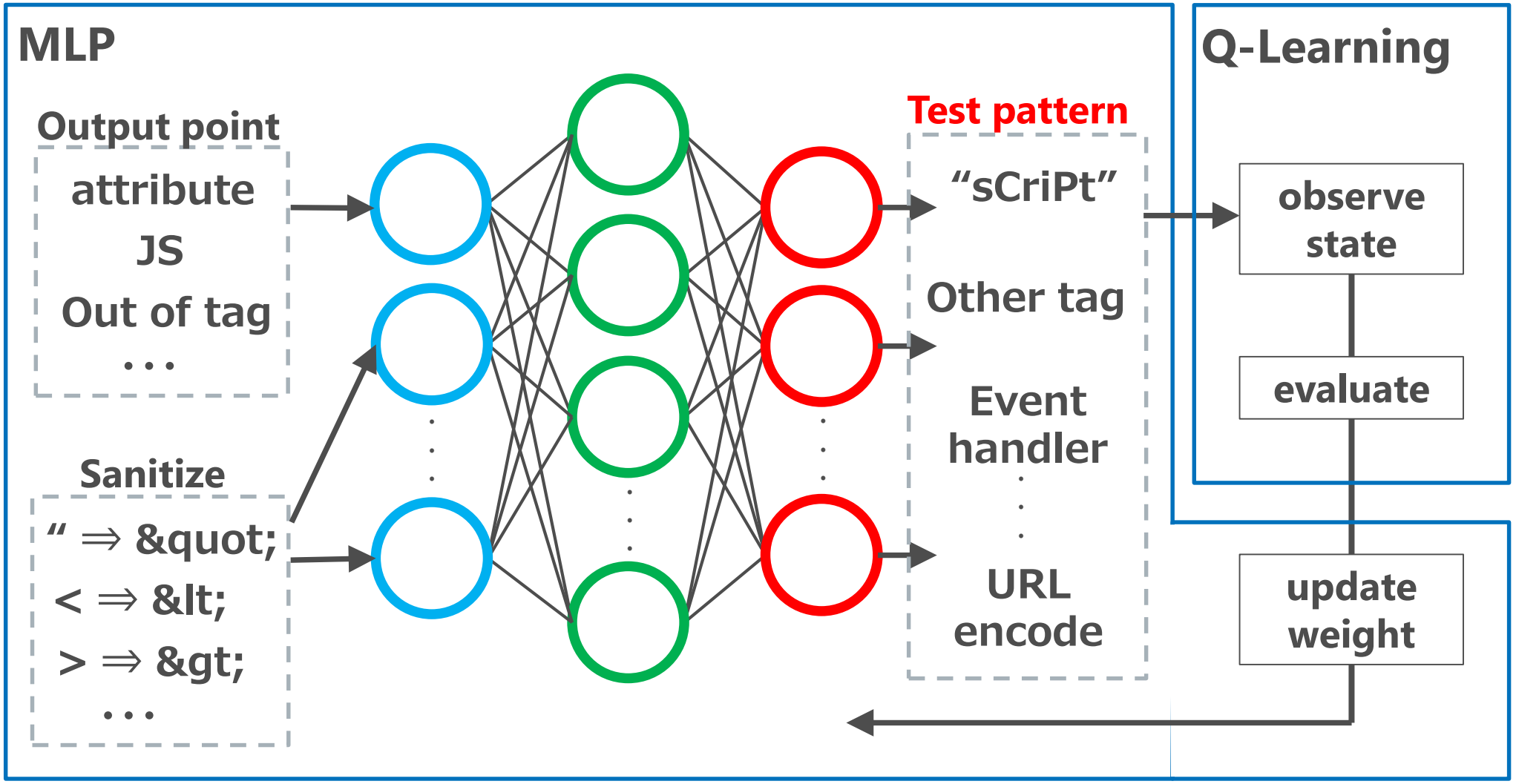
Model to avoid sanitization



Patterns of sanitization

Sanitization	Example
「"」 : translate entity reference, exclude	「"」 ⇒ 「**"**」
「'」 : translate entity reference, exclude	「'」 ⇒ 「**'**」
「<」 : translate entity reference, exclude	「<」 ⇒ 「**<**」
「>」 : translate entity reference, exclude	「>」 ⇒ 「**>**」
「alert();」 : exclude	「alert();」 ⇒ 「」

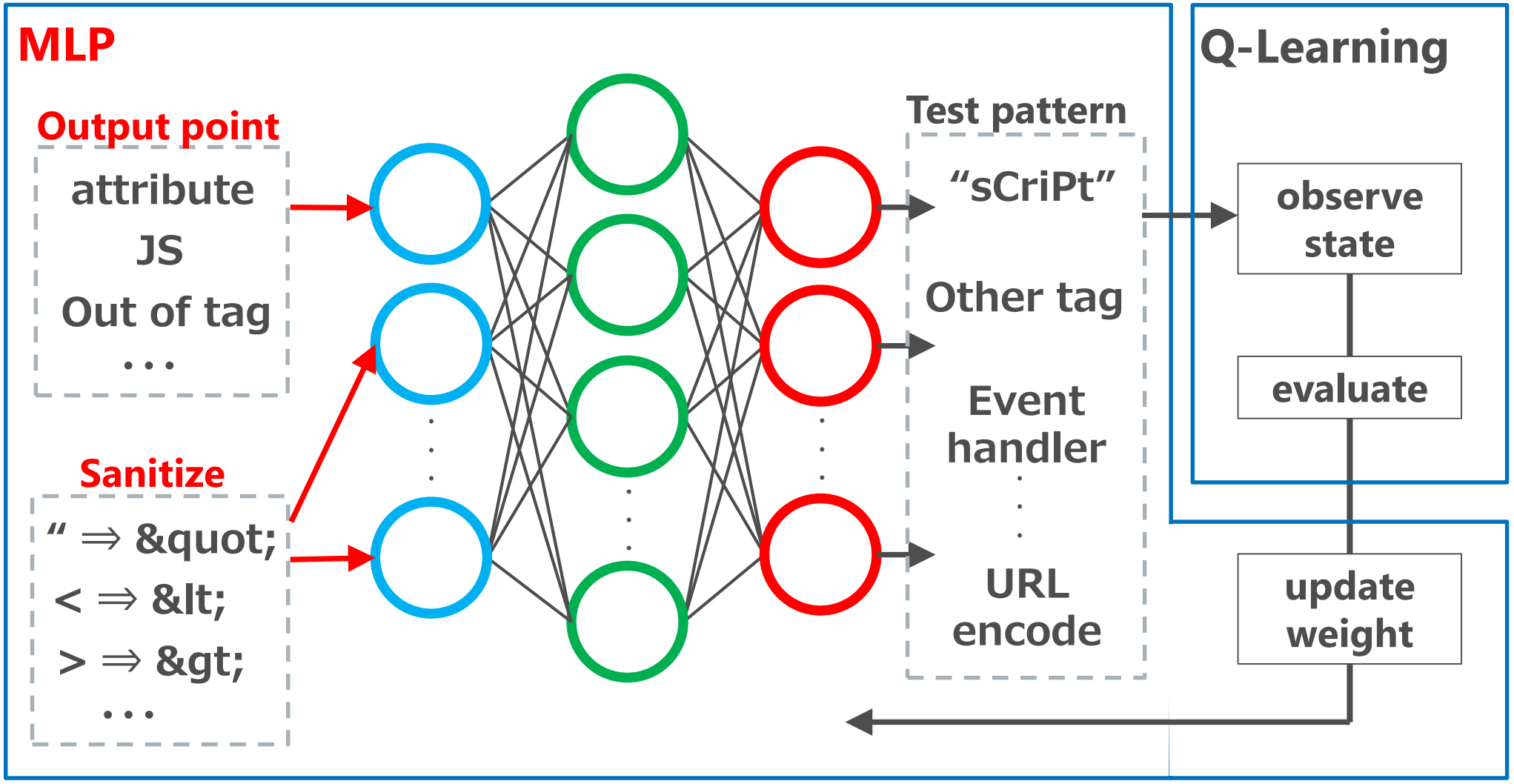
Model to avoid sanitization



Test patterns（one example）

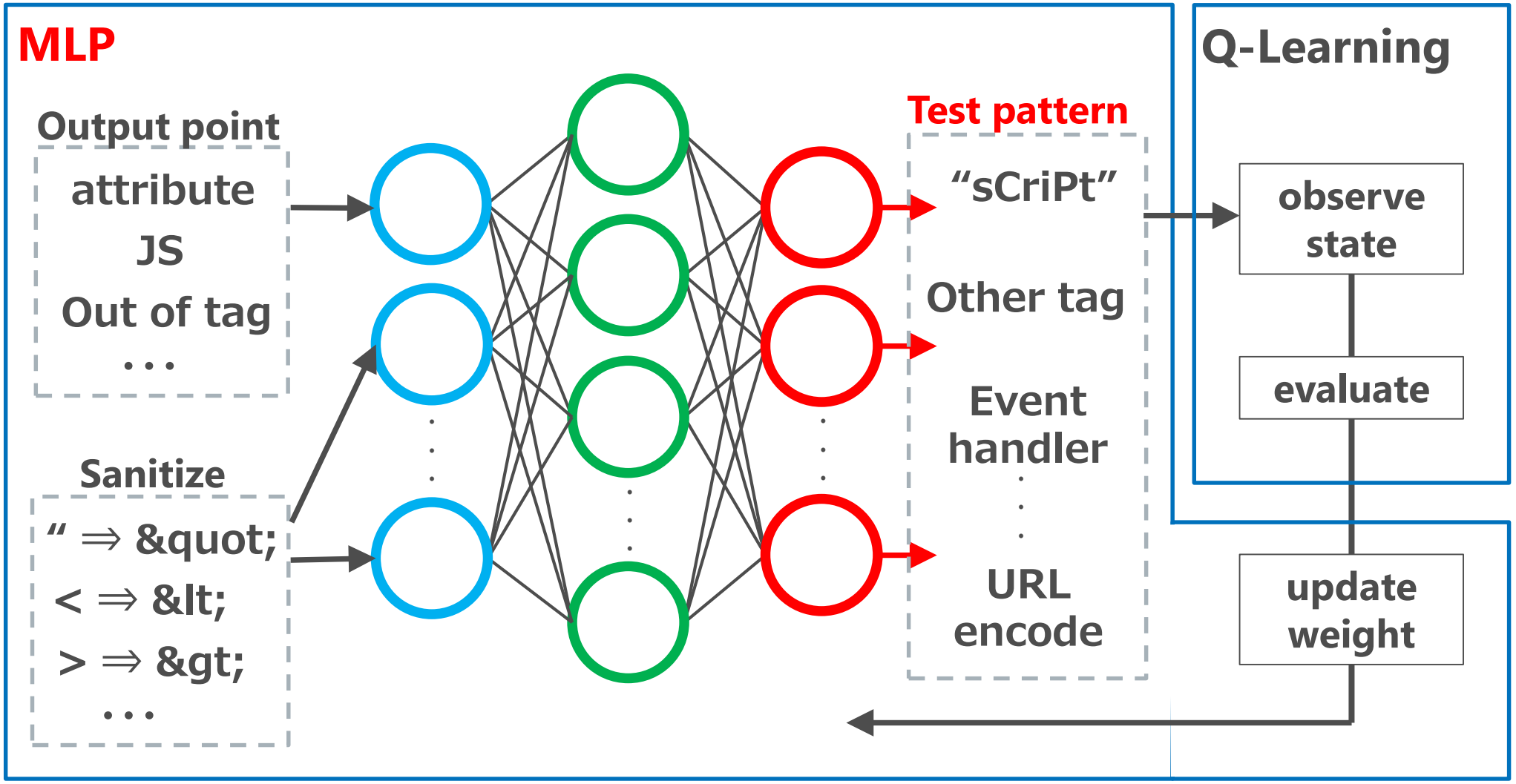
Output point	Test pattern
attribute value :「”」 attribute value :「'」 attribute value : noquote outside of HTML tags	“event handler
	"><sCriPt>xxx
	"><img src=xxx
	‘event handler
	‘><sCriPt>xxx
	<svg/onload=alert()>
	<sCriPt>xxx</sCriPt>
JavaScript	";alert();//
	[CR][LF]alert();
	¥";alert();//

Learning flow



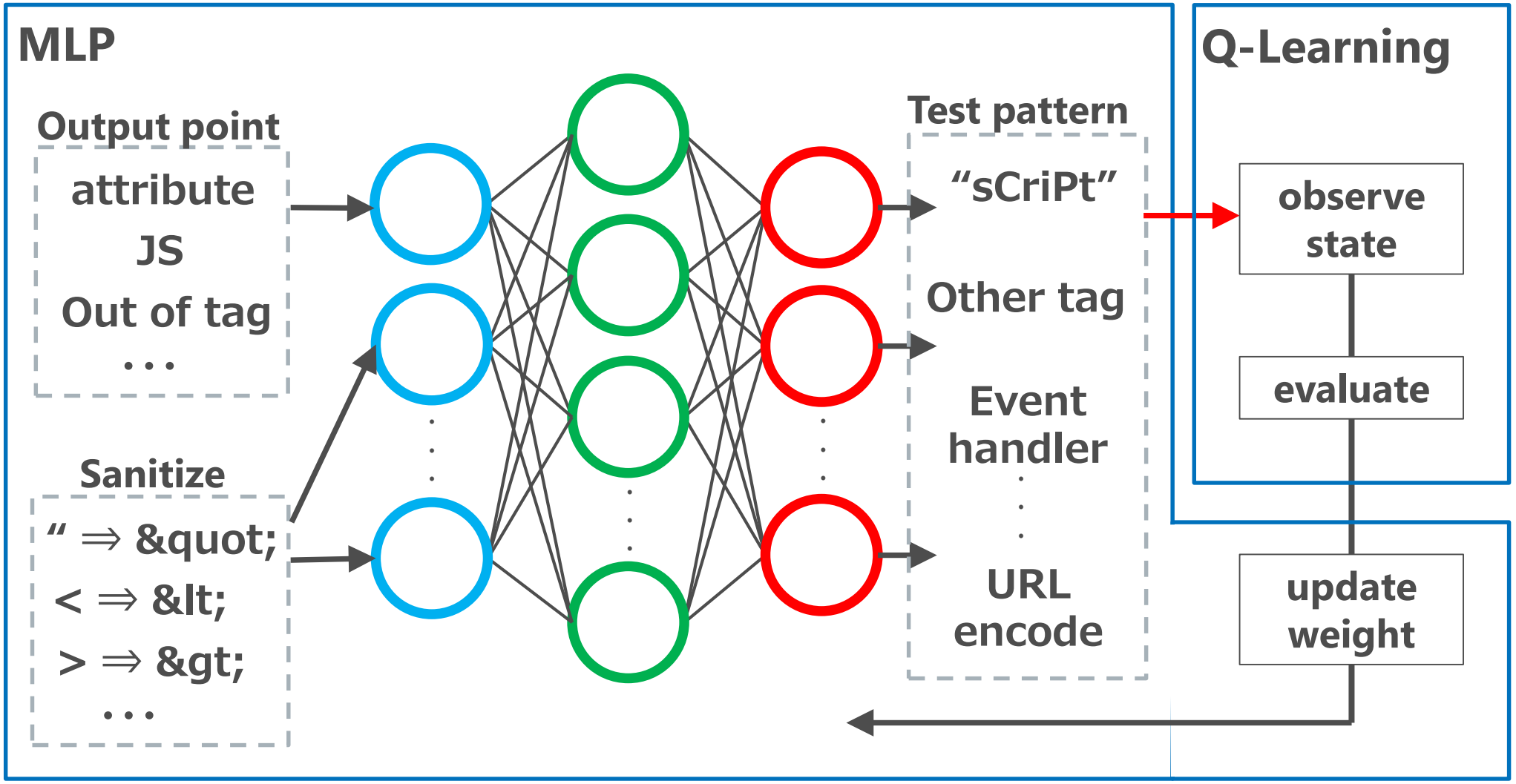
Inputs "Output point" and "pattern of sanitization" to MLP.

Learning flow



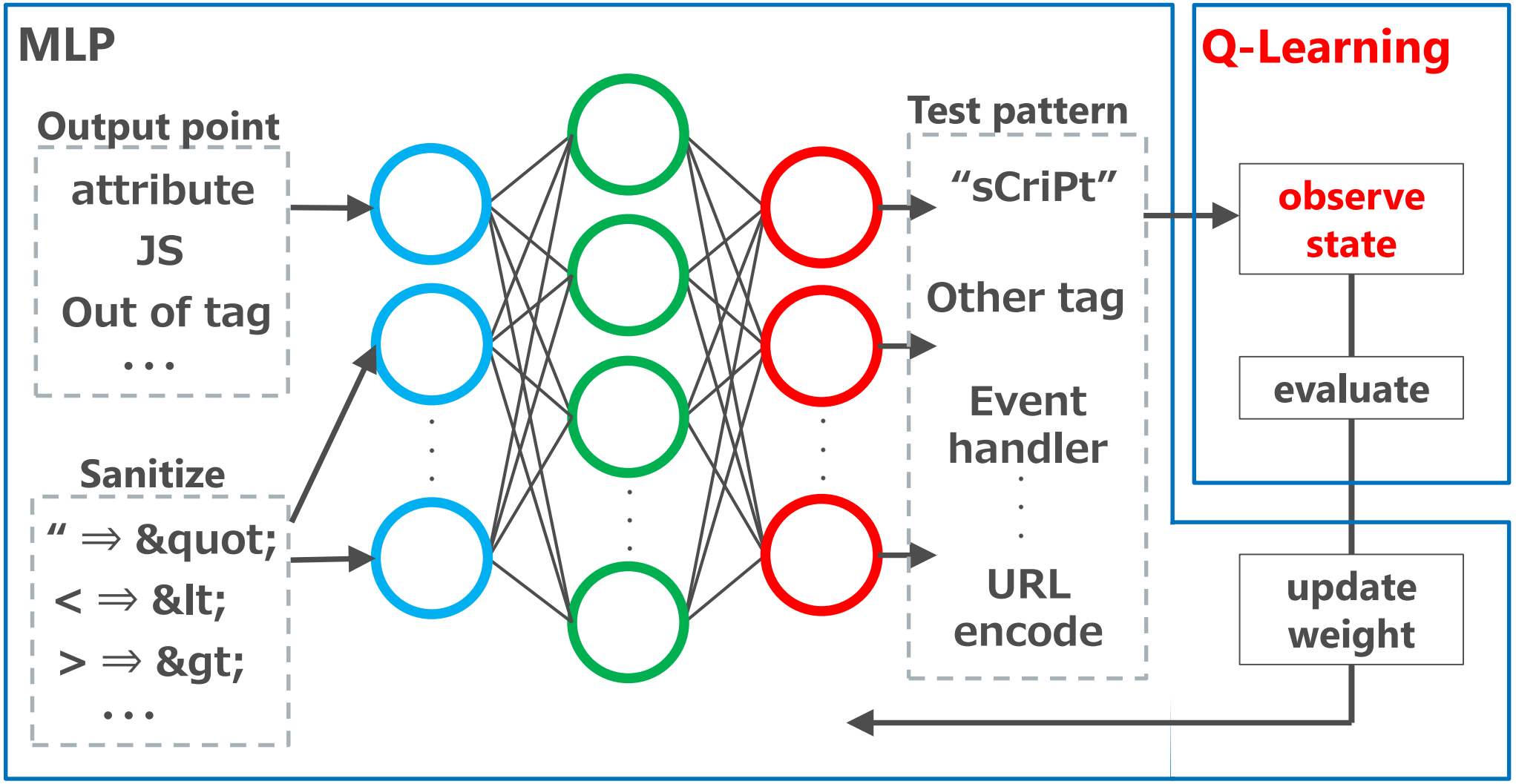
Selects any test pattern from the input of MLP.

Learning flow



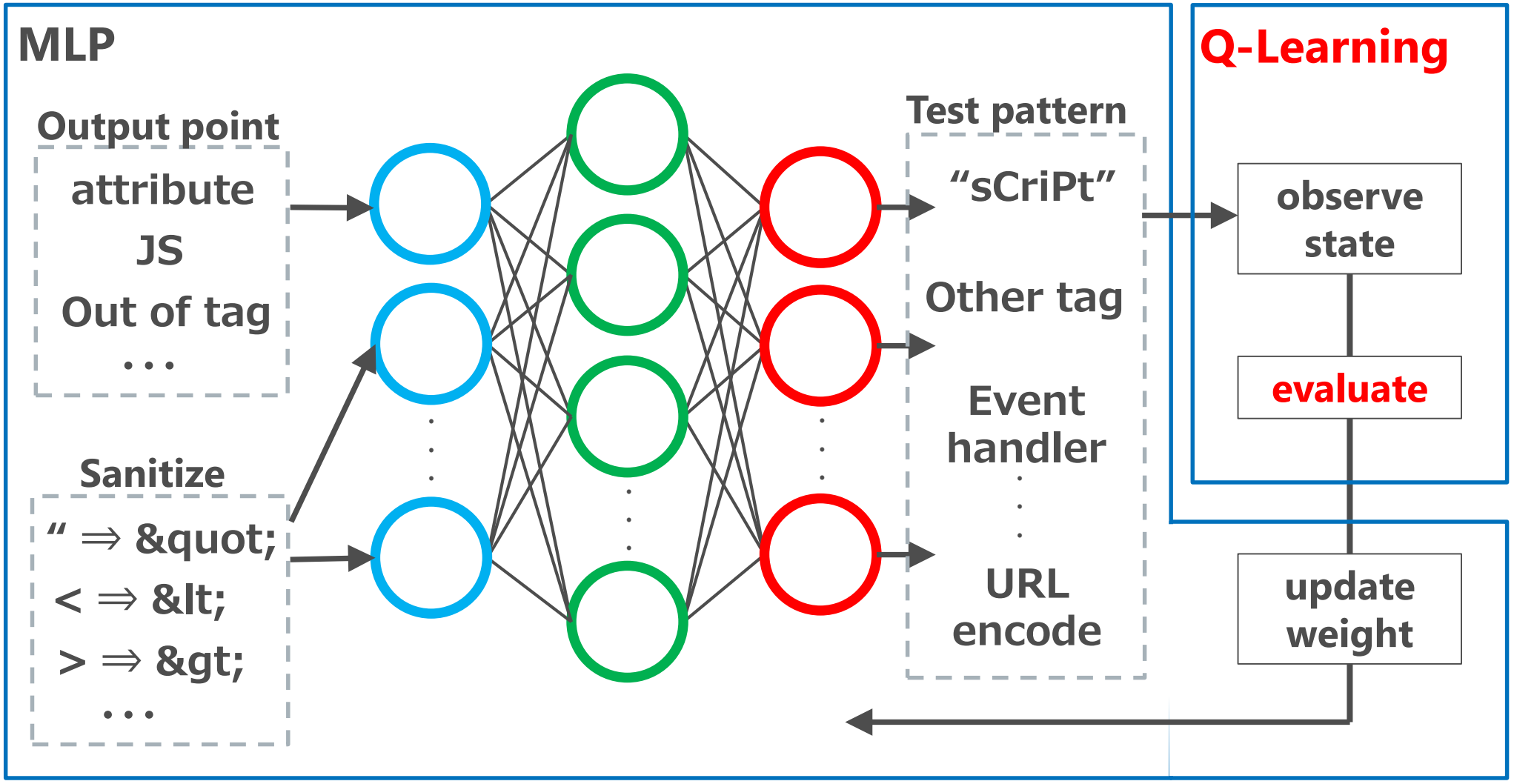
Attempts to test using the selected test pattern of MLP.

Learning flow



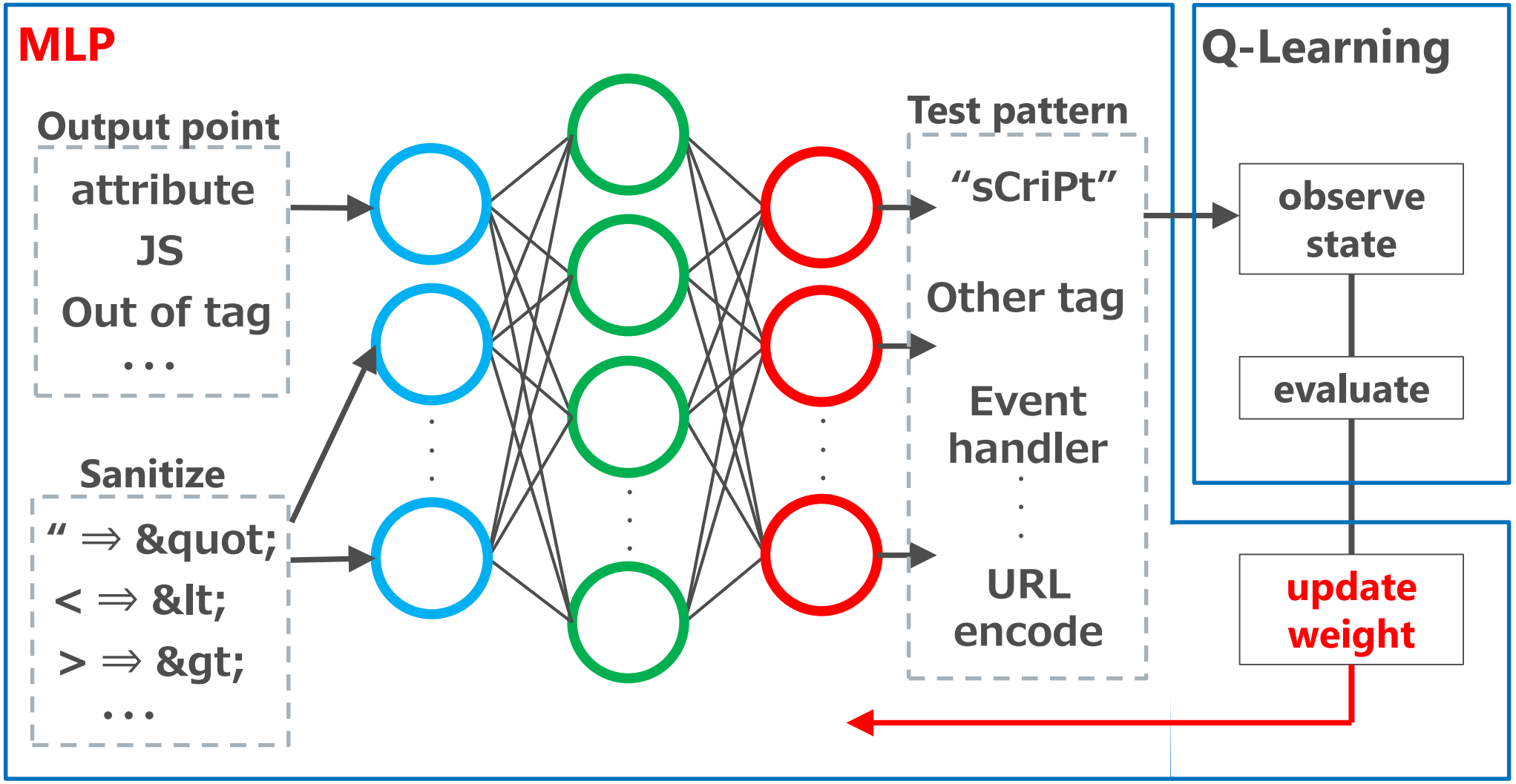
Observe the result of the test.

Learning flow



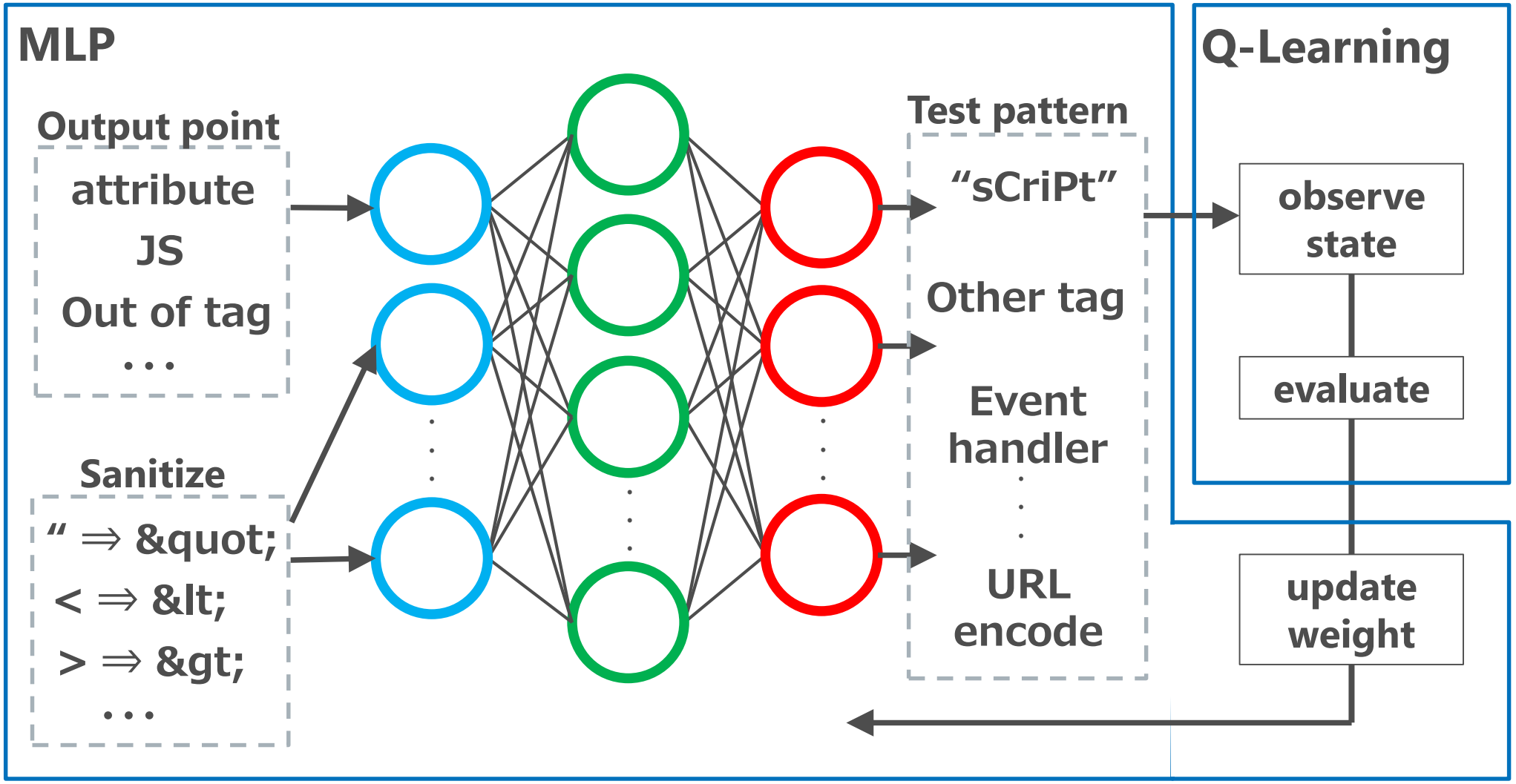
Gives a reward depending on the result of the test.

Learning flow



Update the weight of MLP using backpropagation.

Learning flow.



Learning is completed at **approx. 100 attempts.**

Problem

100 attempts ⇒ Inefficient

Solution

Pre-training.

Web Apps for pre-training（one example）.

WAVSEP

ReflectedXSS GET Input Vector	
Case06	output : SRC attr value of IMG tag. sanitize : < , > ⇒ < , > testing value : "onmouseover=alert(3122);"
Case10	output : onClick value of SCRIPT tag. sanitize : “ , < , > ⇒ " , < , > testing value : ';alert(3122);//
Case27	output : single line comment of JavaScript sanitize : comment out testing value : [CR][LF]alert(3122);//

Training SAIVS

Attempts testing & learning

⇒ Each 100 times.

SAIVS

WAVSEP

Case06 | Case08 | Case10 | Case27 . . .

Achieve the best test string to avoid each sanitize pattern.

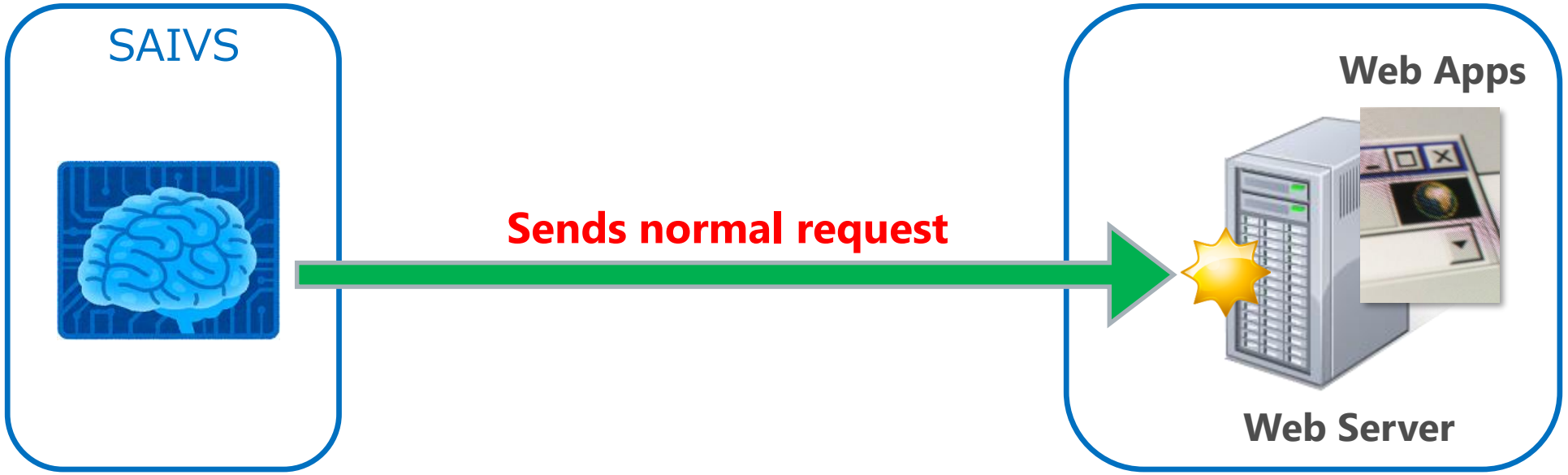
3 requirements for detecting RXSS

- Understanding HTML syntax (✓).
- Understanding JavaScript syntax (✓).
- Avoiding sanitization (✓).

Test flow of SAIVS

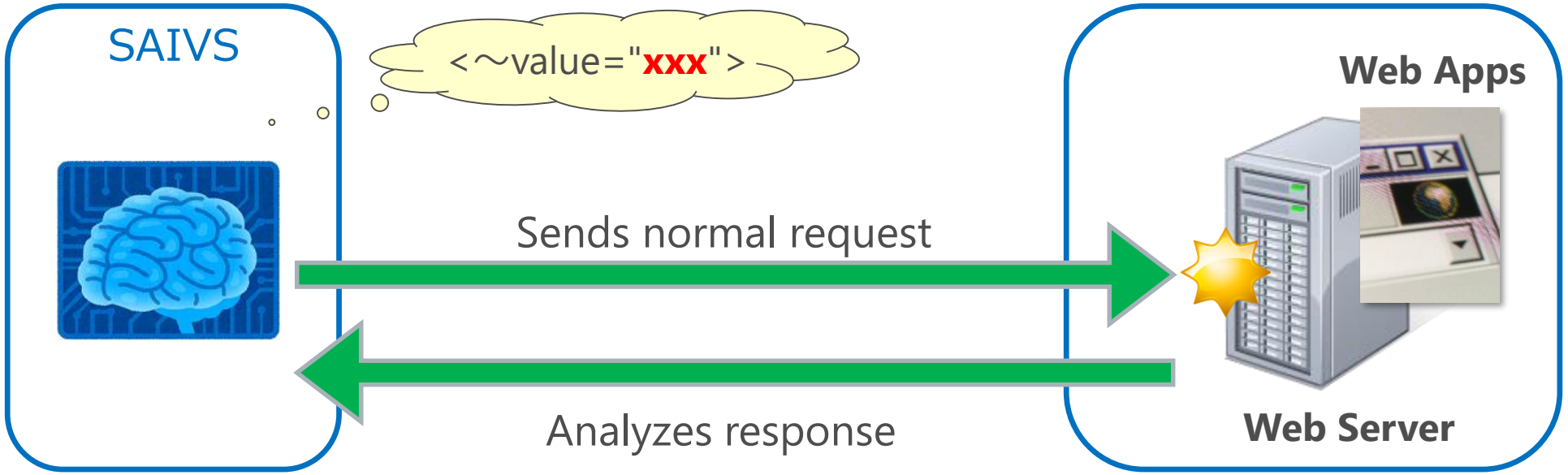
- Primary inspection.
- Secondary inspection.

Primary inspection



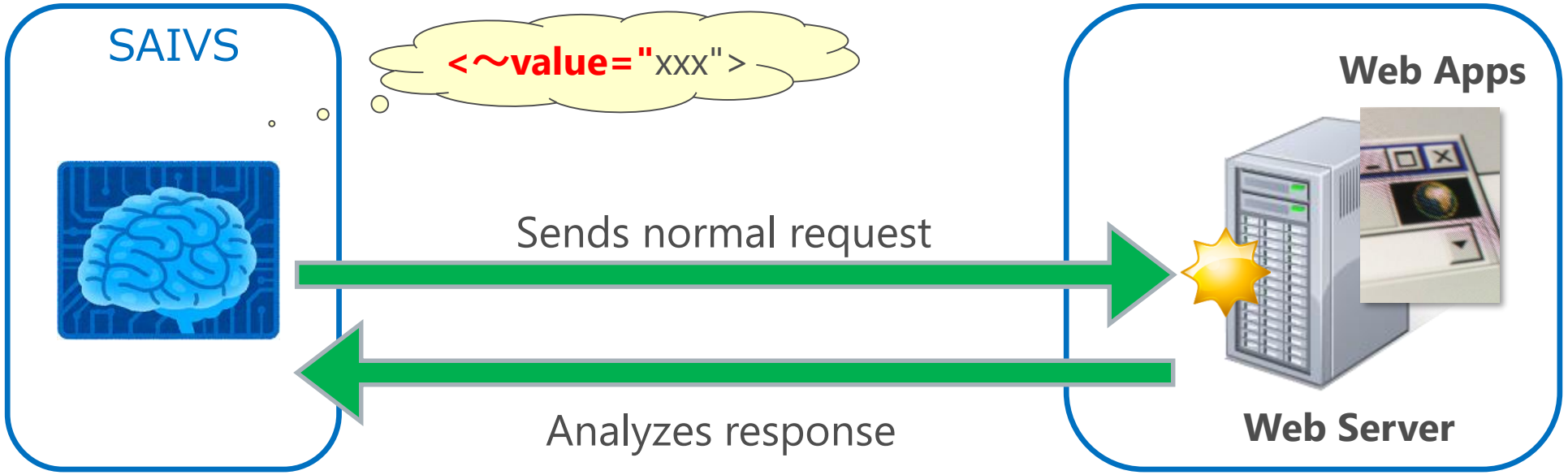
Sends normal request to **examine the output point**.

Primary inspection.



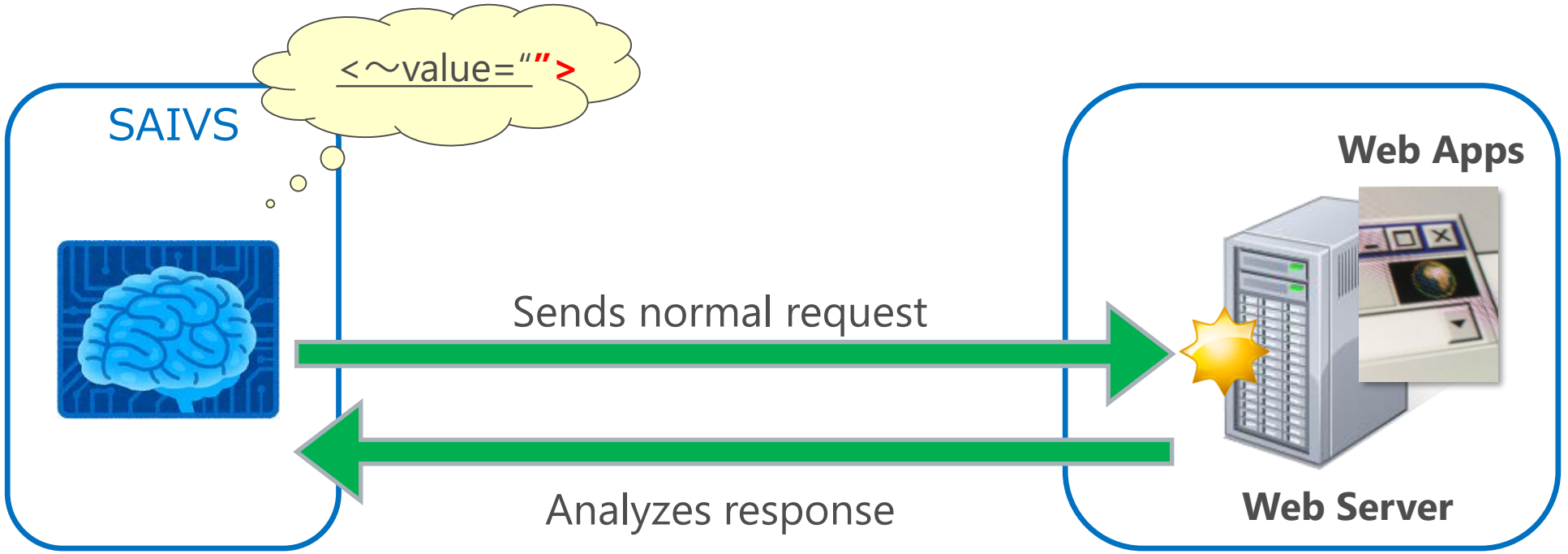
Recognizes the output point of the input value.

Primary inspection



Extracts the seed based on the output point.

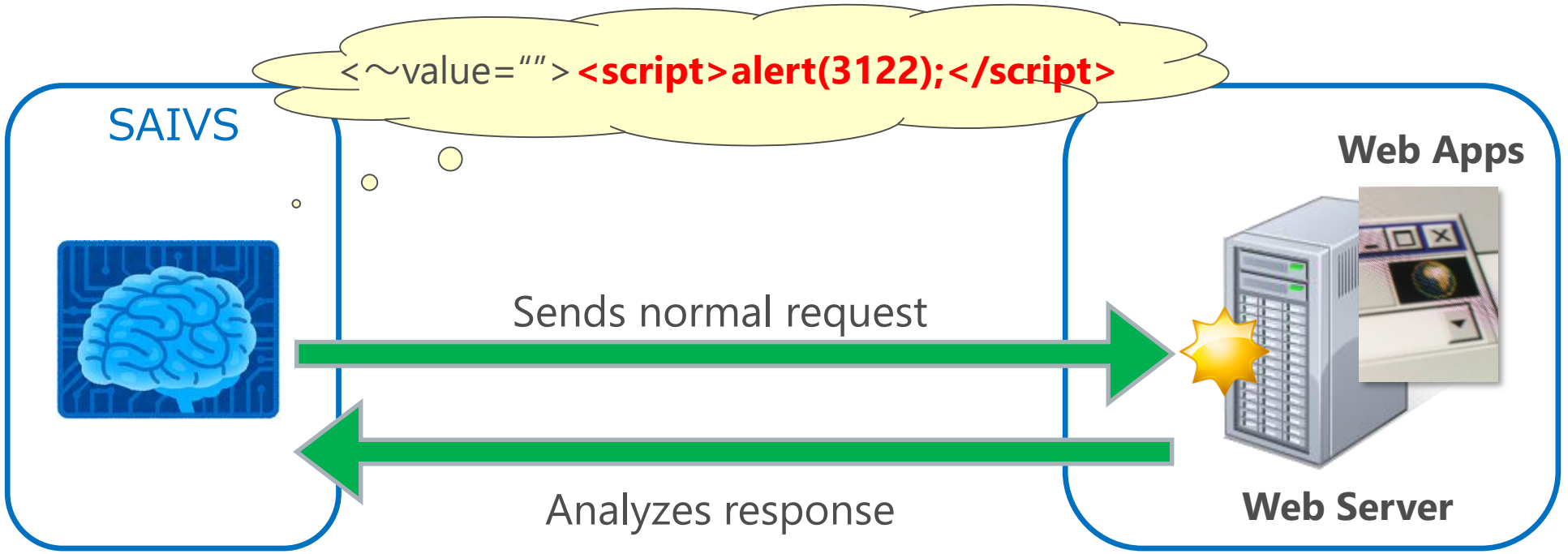
Primary inspection



Generates HTML / JavaScript syntax

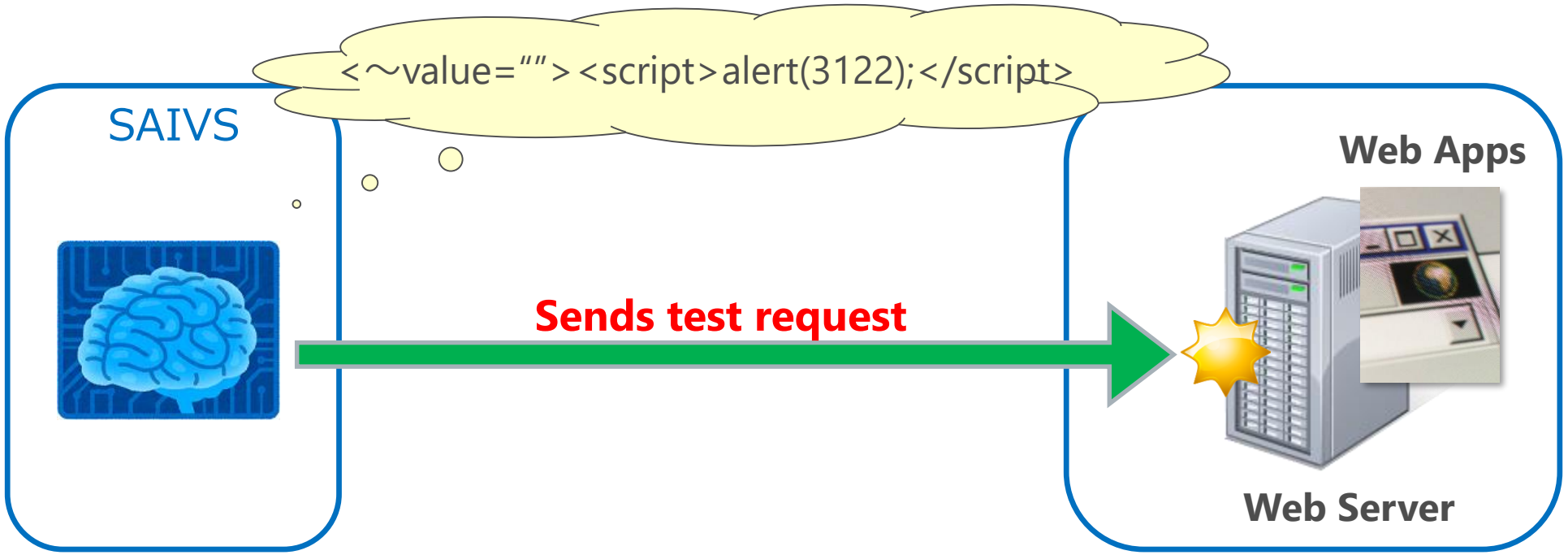
from the seed. (using learned LSTM)

Primary inspection



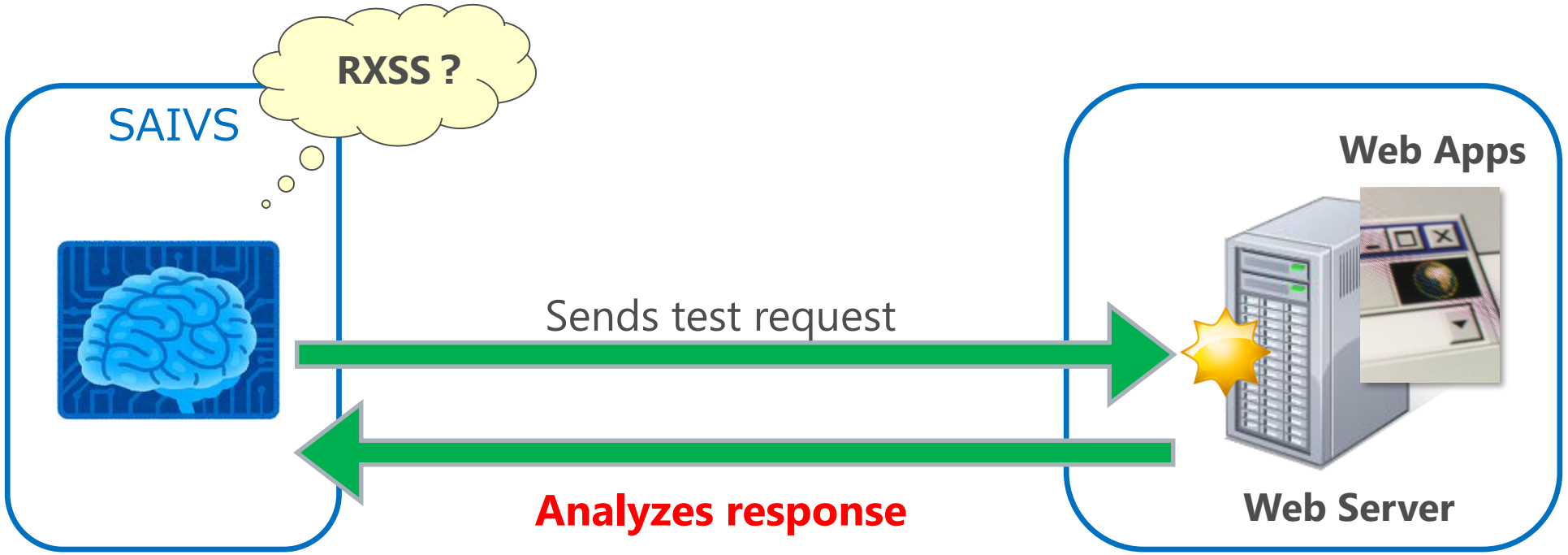
Adds the script to the generated HTML syntax.

Primary inspection



Sends test request to detect RXSS.

Primary inspection

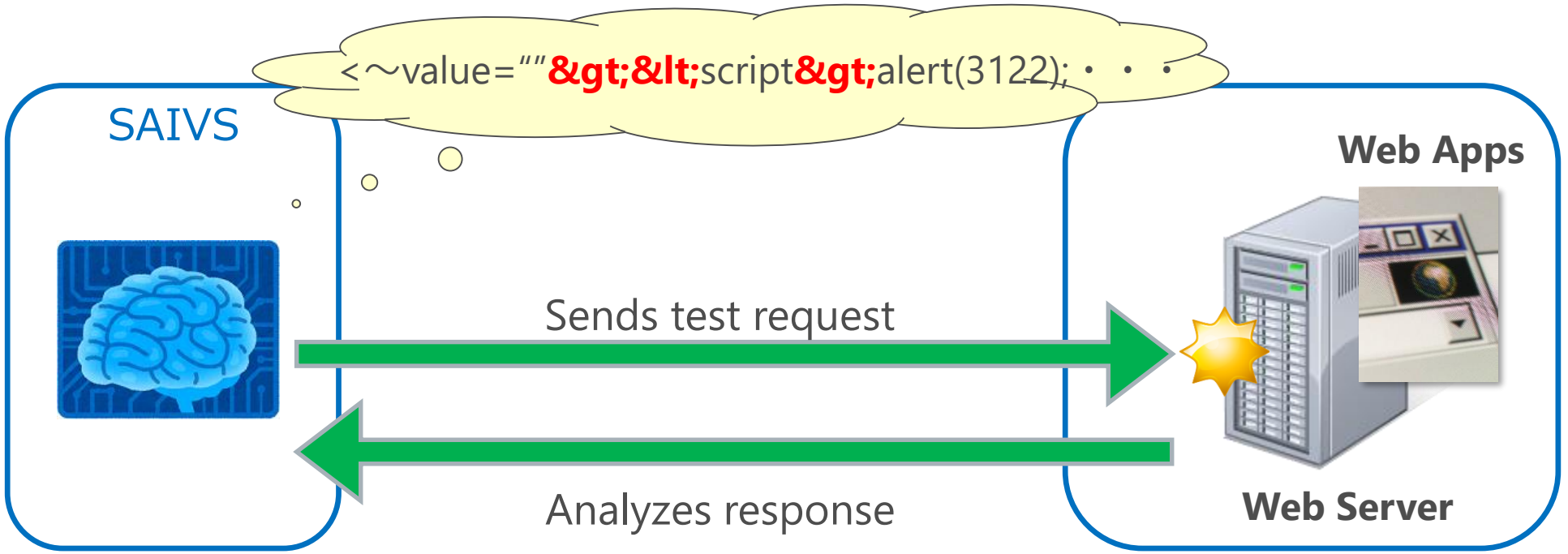


Determines the presence or absence of RXSS.

RXSS detected ⇒ Test finishes.

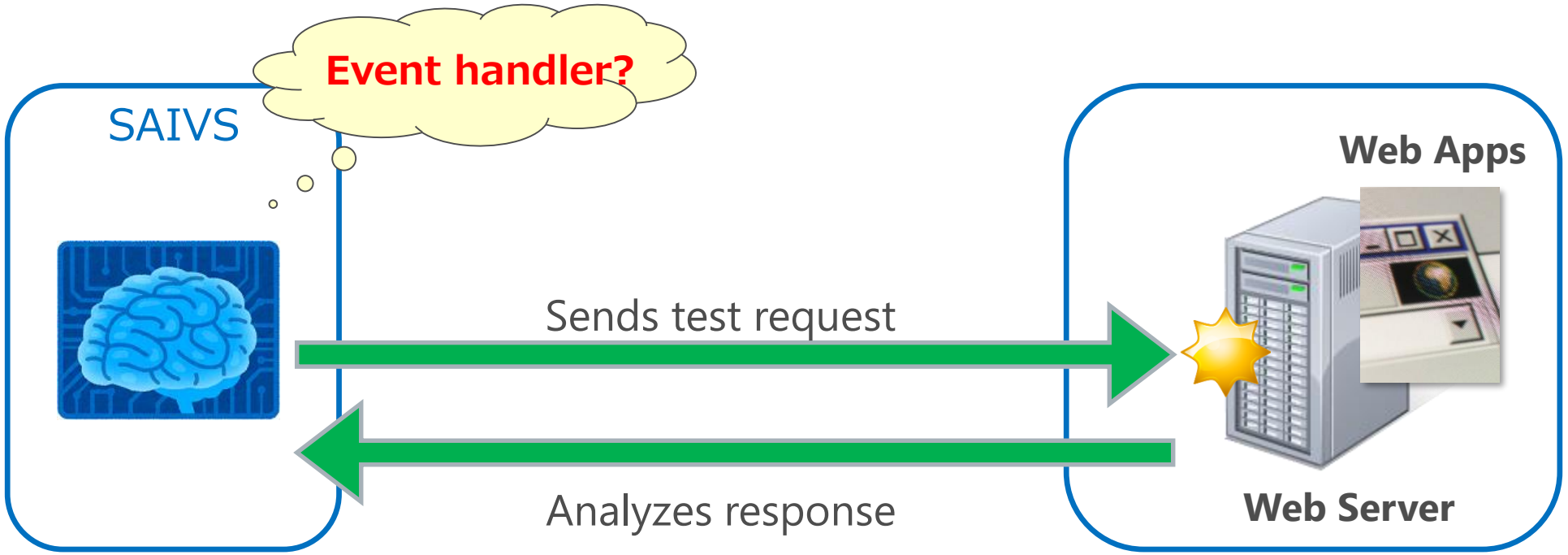
Not detected⇒ Secondary inspection.

Secondary inspection



Recognizes the sanitizing pattern.

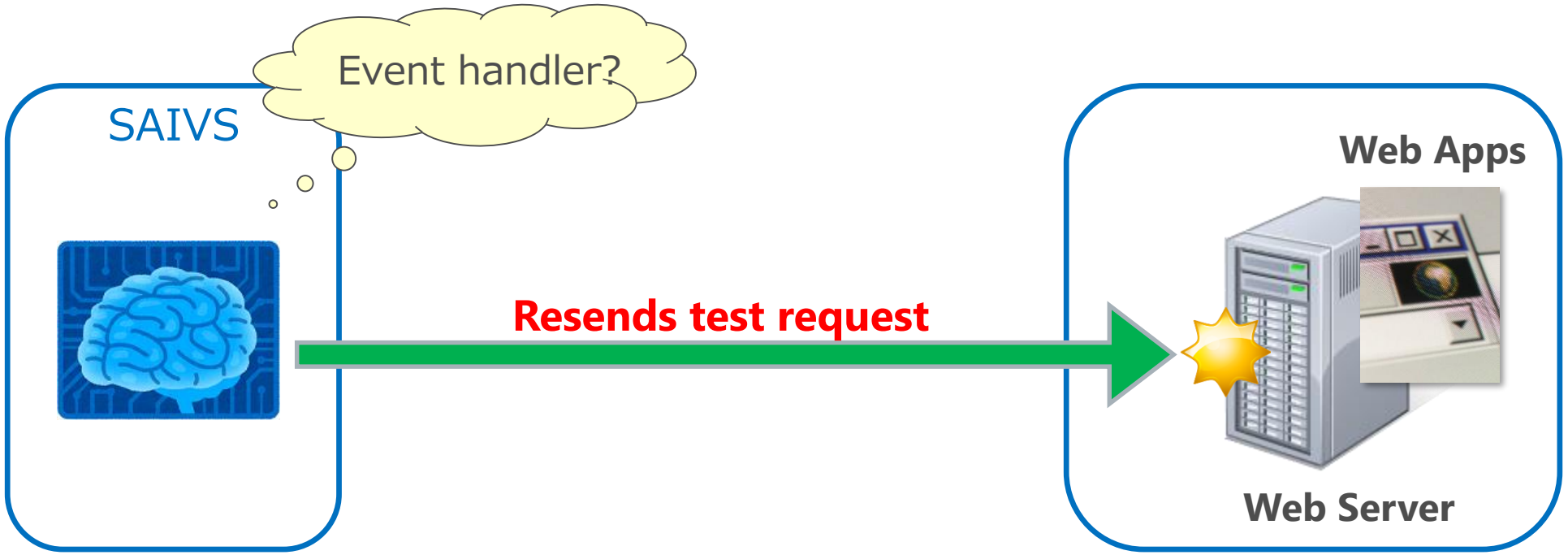
Secondary inspection



Selects the test pattern to **avoid sanitization**.

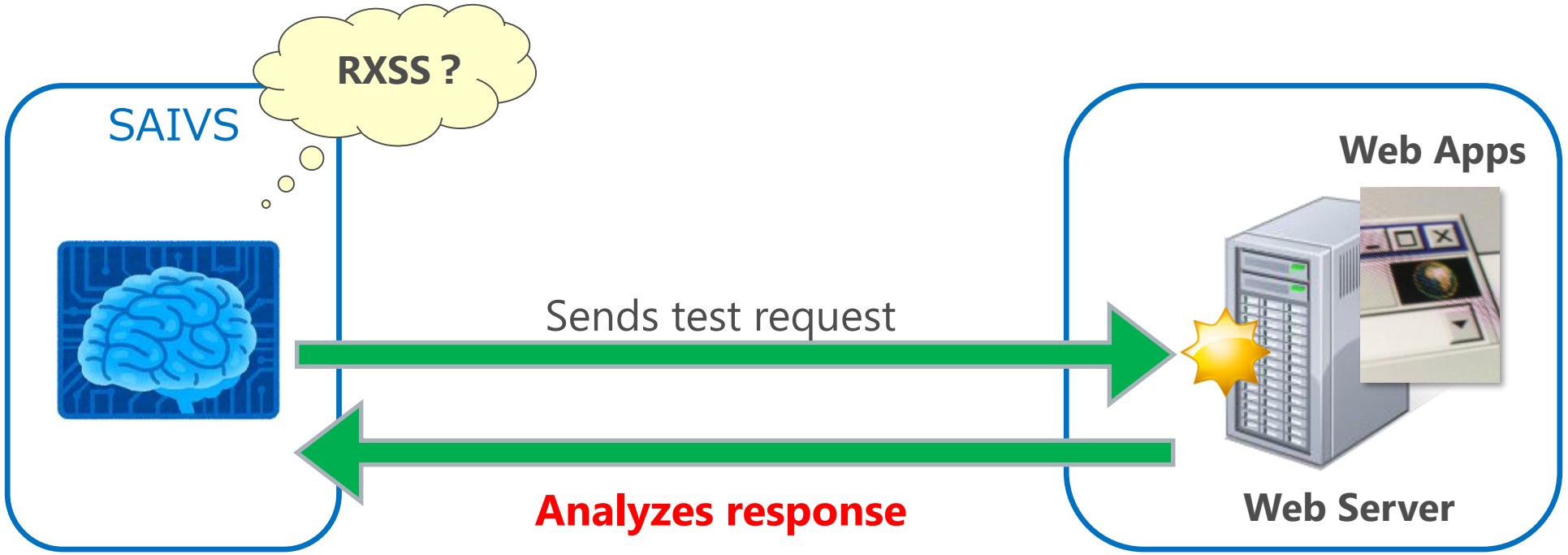
(Using learned MLP)

Secondary inspection



Sends the test request to **avoid sanitization.**

Secondary inspection



Determines presence or absence of RXSS.

RXSS detected ⇒ Test finishes.

Not detected ⇒ Repeat secondary inspection.

Demonstration of detecting RXSS

Target：**webseclab**

Case	Overview
/reflect/full1	output ：BODY tag sanitize : none
/reflect/textarea1	output ：TEXTAREA tag sanitize : none
/reflect/onmouseover	output ：attribute value of INPUT tag sanitize : exclude “</script>”
/reflect/js4_dq	output ：SCRIPT tag sanitize : none

Demonstration of detecting RXSS

Target：**webseclab**

Case	Overview
/reflect/full1	output ：BODY tag sanitize : none
/reflect/textarea1	output ：TEXTAREA tag sanitize : none
/reflect/onmouseover	output ：attribute value of INPUT tag sanitize : exclude “</script>”
/reflect/js4_dq	output ：SCRIPT tag sanitize : none

Case1：Sending normal request

http://xxx/reflect/full1?in=saivs12345

```
<!doctype html><html><head><title>Full Javascript
Injection (full.1)</title></head><body>
Hello!<BR>
The value of cgi parameter &quot;in&quot;
is:saivs12345
</body></html>
```

output ：BODY tag

sanitize：None

Case1：Sending test request

http://xxx/reflect/full1?in=lasther=''%3E%3C/form%3ED0i7Q%22VW53N'nT7t0%3Cscript%3Ealert(3122);kc5i3%3C/script%3EueFj8

```
<!doctype html><html><head><title>Full Javascript
Injection (full.1)</title></head><body>
Hello!<BR>
The value of cgi parameter &quot;in&quot; is:
lasther=''></form>D0i7Q"VW53N'nT7t0<script>alert(3122)
;kc5i3</script>ueFj8
</body></html>
```

attempted times : 1

Case1：Demo movie

https://www.youtube.com/watch?v=3RkhSED5DQU

Demonstration of detecting RXSS

Target：**webseclab**

Case	Overview
/reflect/full1	output ：BODY tag sanitize : none
/reflect/textarea1	output ：TEXTAREA tag sanitize : none
/reflect/onmouseover	output ：attribute value of INPUT tag sanitize : exclude “</script>”
/reflect/js4_dq	output ：SCRIPT tag sanitize : none

Case2：Sending normal request

http://xxx/reflect/textarea1?in=saivs12345

```
<!doctype html><html><head><title>Reflected XSS in
textarea (textarea1)</title></head><body>
<H2>Textarea injection test</H2>
<p>
<FORM>
<textarea name="in" rows="5" cols="60">saivs12345
</textarea>
<p>
```

output ：TEXTAREA tag

sanitize：None

Case2：Sending test request

http://xxx/reflect/textarea1?in=%3C/textarea%3E7Q7pN%22MBPcc'PA6tz%3Cscript%3Ealert(3122);WKr8J%3C/script%3EfowCP

```
<!doctype html><html><head><title>Reflected XSS in
textarea (textarea1)</title></head><body>
<H2>Textarea injection test</H2>
<p>
<FORM>
<textarea name="in" rows="5" cols="60">
```**`</textarea>7Q7pN"MBPcc'PA6tz<script>alert(3122);WKr8J</script>`**`fowCP</
textarea>
<p>
```

attempted times : 1
```

Case2：Demo movie

https://www.youtube.com/watch?v=6UHbMGdqr_0

Demonstration of detecting RXSS

Target：**webseclab**

Case	Overview
/reflect/full1	output ：BODY tag sanitize : none
/reflect/textarea1	output ：TEXTAREA tag sanitize : none
/reflect/onmouseover	output ：attribute value of INPUT tag sanitize : exclude “</script>”
/reflect/js4_dq	output ：SCRIPT tag sanitize : none

Case 3：Sending normal request

http://xxx/xss/reflect/onmouseover?in="><script>alert()</script>

```
<!doctype html><html>
<head><title>Reflected XSS - attribute injection in ta
gs (dq.2)</title></head><body>
<H2>Update Your Preferences</H2><p>
<FORM>
Homepage: <input value=""><script>alert()" name="in" s
ize="40"><BR>
```

output ：INPUT tag

sanitize：exclude “</script>”

Case 3：Sending test request.

http://xxx/xss/reflect/onmouseover?in=%22%3E%3C/option%3E%3

Coption%20s%20onmouseover=alert(3122);//

```
<!doctype html><html>
<head><title>Reflected XSS - attribute injection in
tags (dq.2)</title></head><body>
<H2>Update Your Preferences</H2><p>
<FORM>
Homepage: <input value="">  <option s onmouseover=alert
(3122);//" name="in" size="40"><BR>
```

attempted times : 3

Case 3：Demo movie

https://www.youtube.com/watch?v=-r3C1moUVqU

Demonstration of detecting RXSS

Target：**webseclab**

Case	Overview
/reflect/full1	output ：BODY tag sanitize : none
/reflect/textarea1	output ：TEXTAREA tag sanitize : none
/reflect/onmouseover	output ：attribute value of INPUT tag sanitize : exclude “</script>”
/reflect/js4_dq	output ：SCRIPT tag sanitize : none

Case 4：Sending normal request

http://xxx/xss/reflect/js4_dq?in=saivs12345

```
<!doctype html><html><head><title>JavaScript and
double-quote injection in JS block (js.4)</title>
</head><body>
<script language="javascript">
var f = {
 date: "",
 week: "1",
 bad: "saivs12345",
 phase: "2",
```

output ：SCRIPT tag

sanitize：None

Case 4：Sending test request

http://xxx/xss/reflect/js4_dq?in=6%22,%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20skuI;alert(3122);//1VU7k

```
<!doctype html><html><head><title>JavaScript and
double-quote injection in JS block (js.4)</title>
</head><body>
<script language="javascript">
var f = {
 date: "",
 week: "1",
 bad: "6",
 skuI;alert(3122);//1VU7k",
 phase: "2",
```

attempted times : 1

Case 4：Demo movie

https://www.youtube.com/watch?v=Pf2lSB25C3M

Abilities of SAIVS

- Crawls web apps. (✓)
- Detects vulnerabilities. (✓)

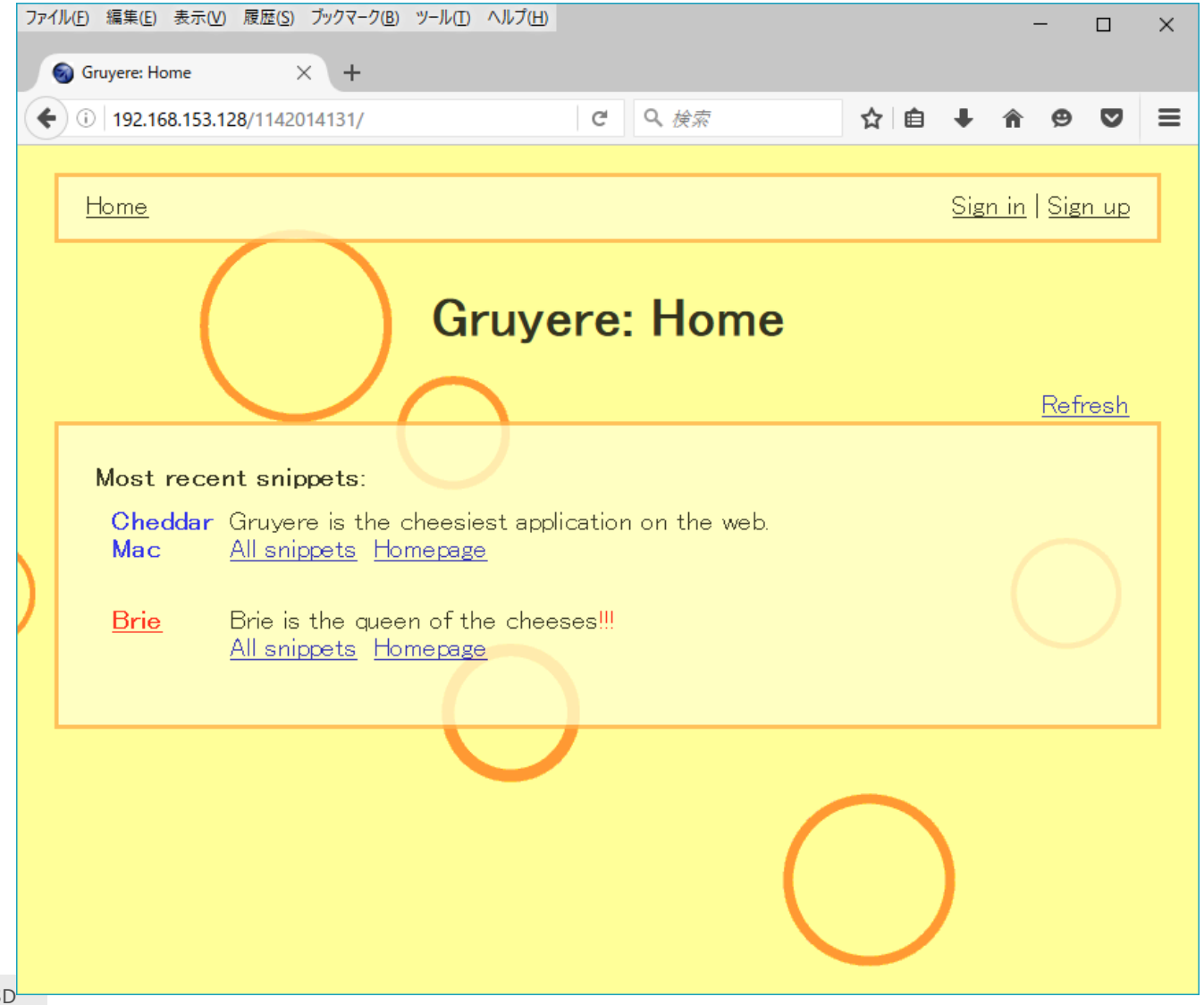
Abilities of SAIVS

"**Detect RXSS** while **crawling** web apps."

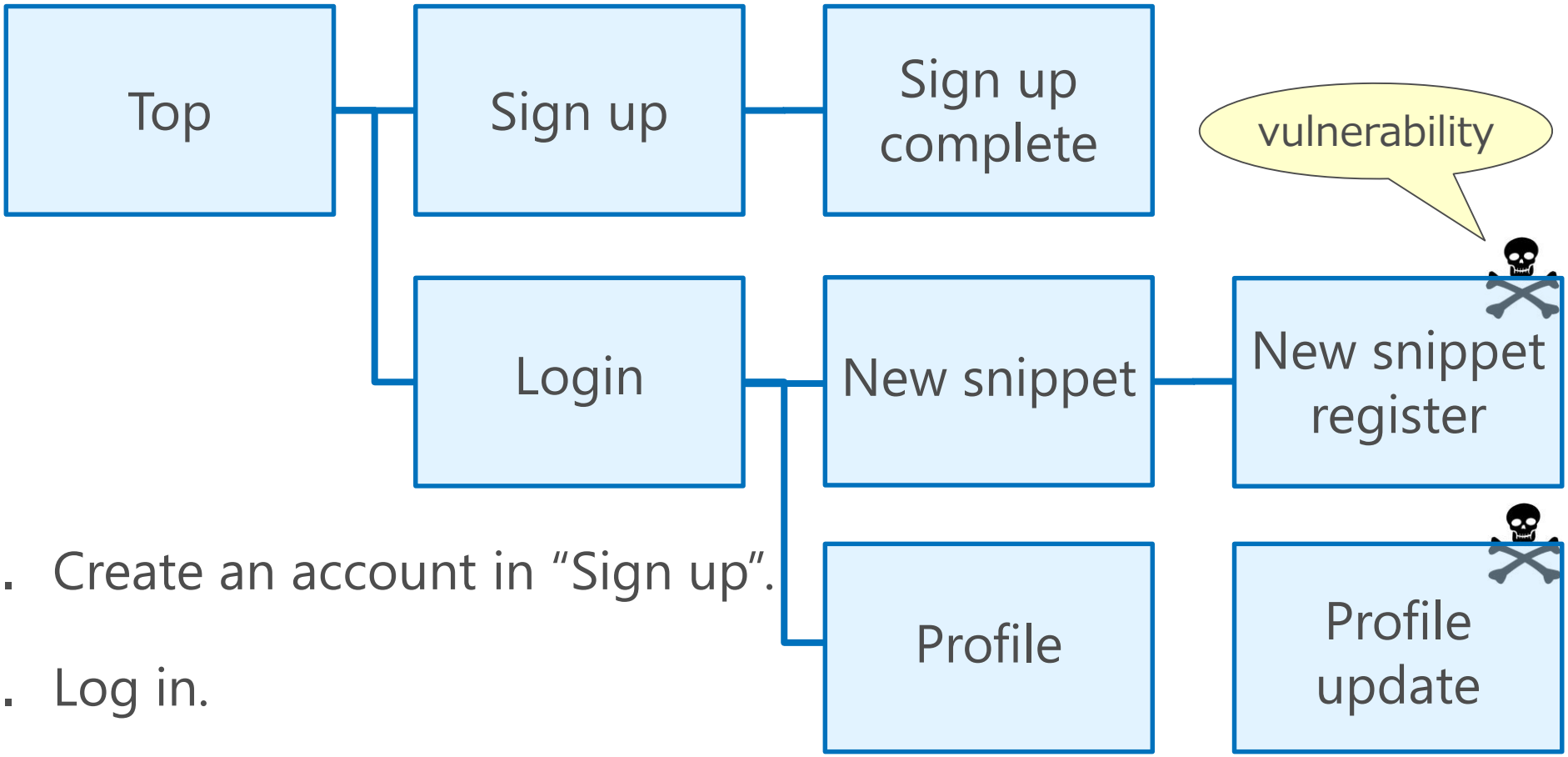
Realized fully automatically.

Demonstration of SAIVS

Target：OWASP Broken Web Apps **Google Gruyere**

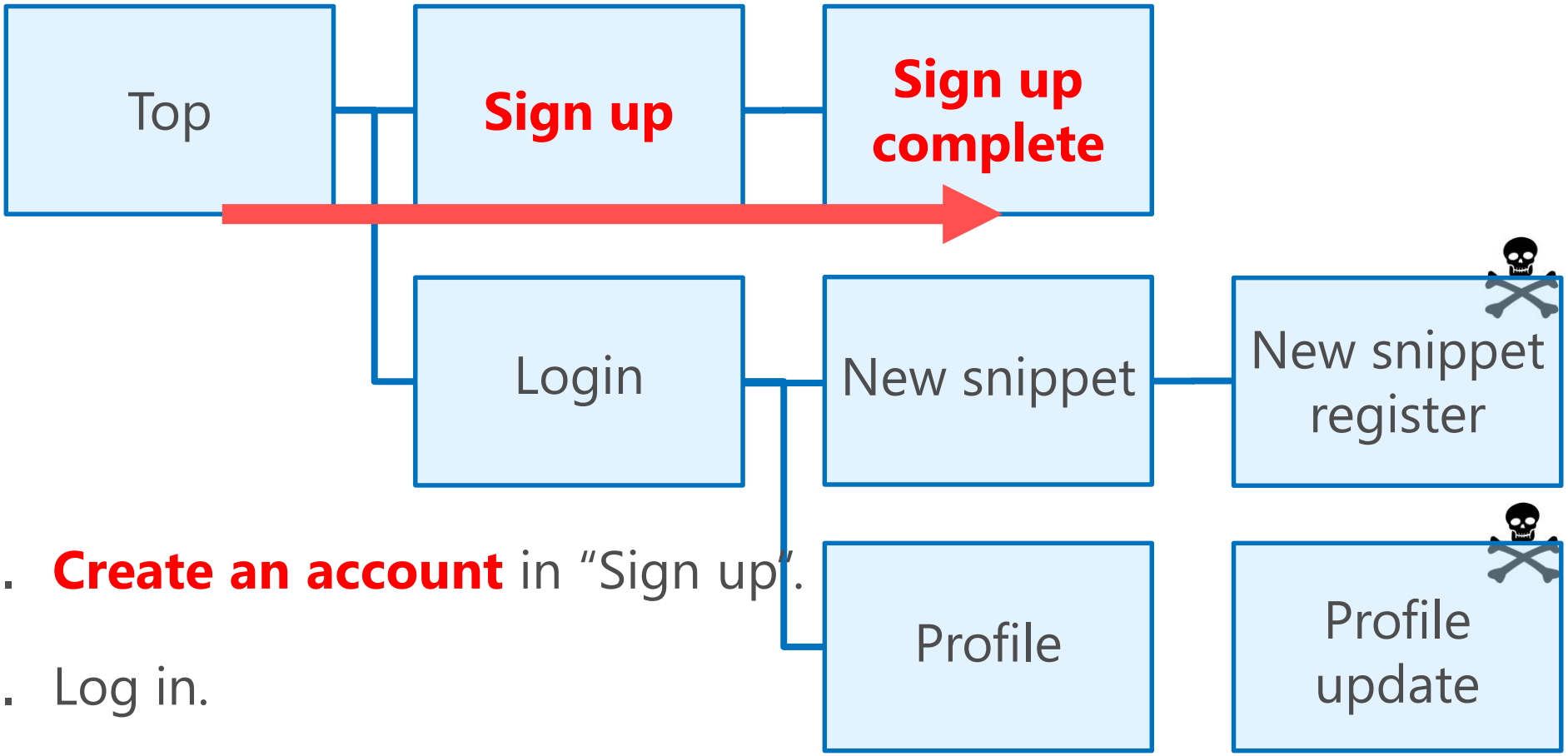


Demonstration of SAIVS



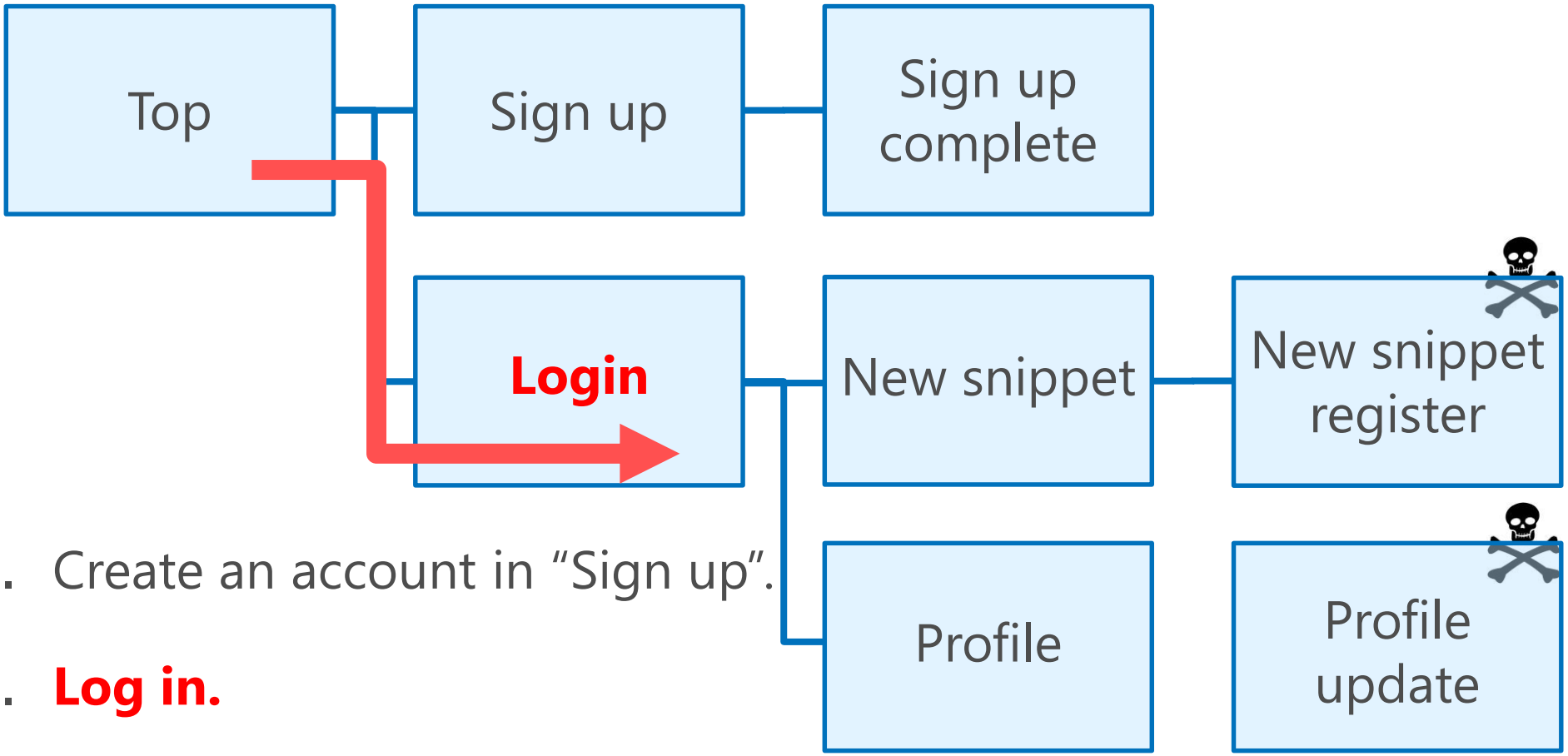
1. Create an account in “Sign up”.
2. Log in.
3. Register Snippet.
4. Update Profile.

Demonstration of SAIVS



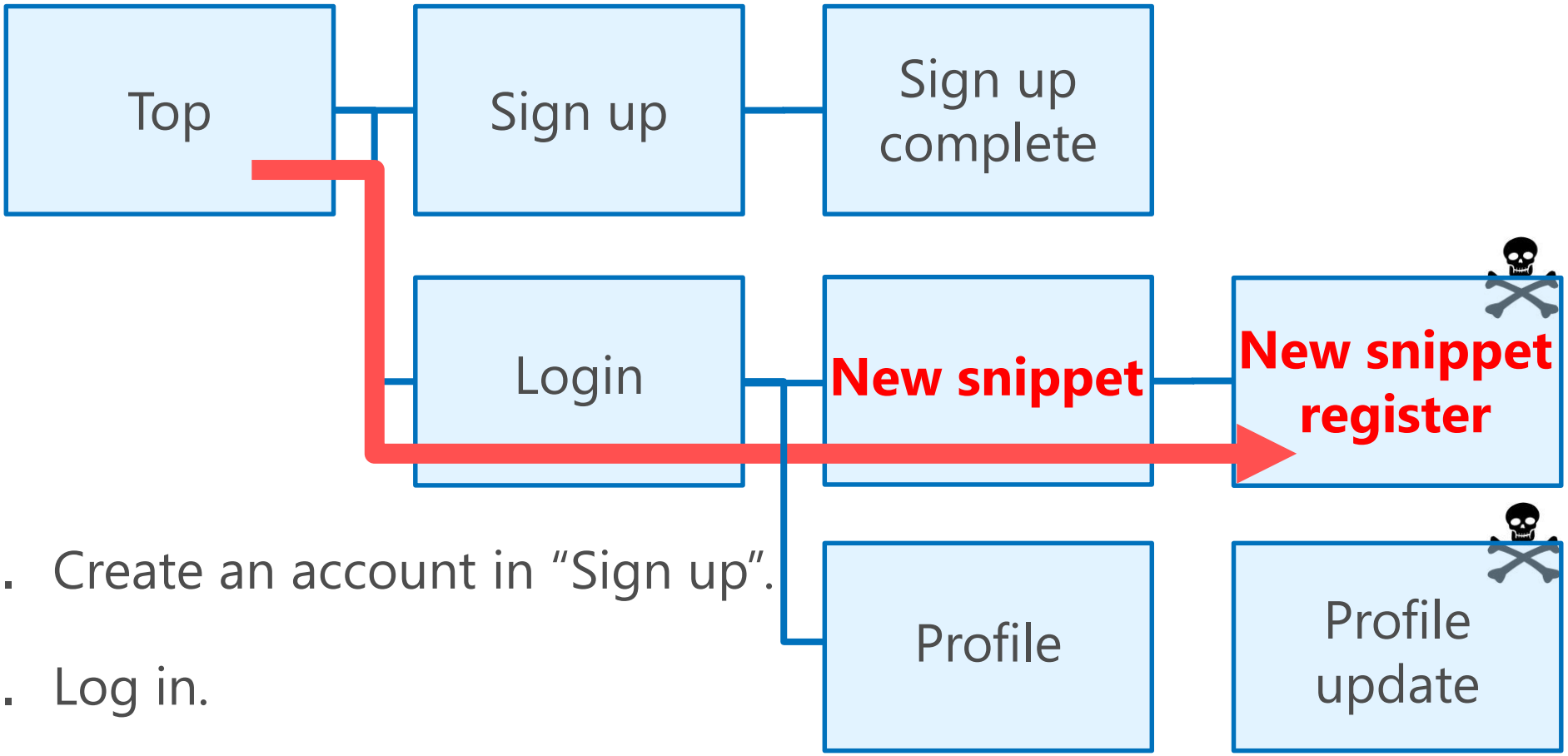
1. **Create an account** in “Sign up”.
2. Log in.
3. Register Snippet.
4. Update Profile.

Demonstration of SAIVS



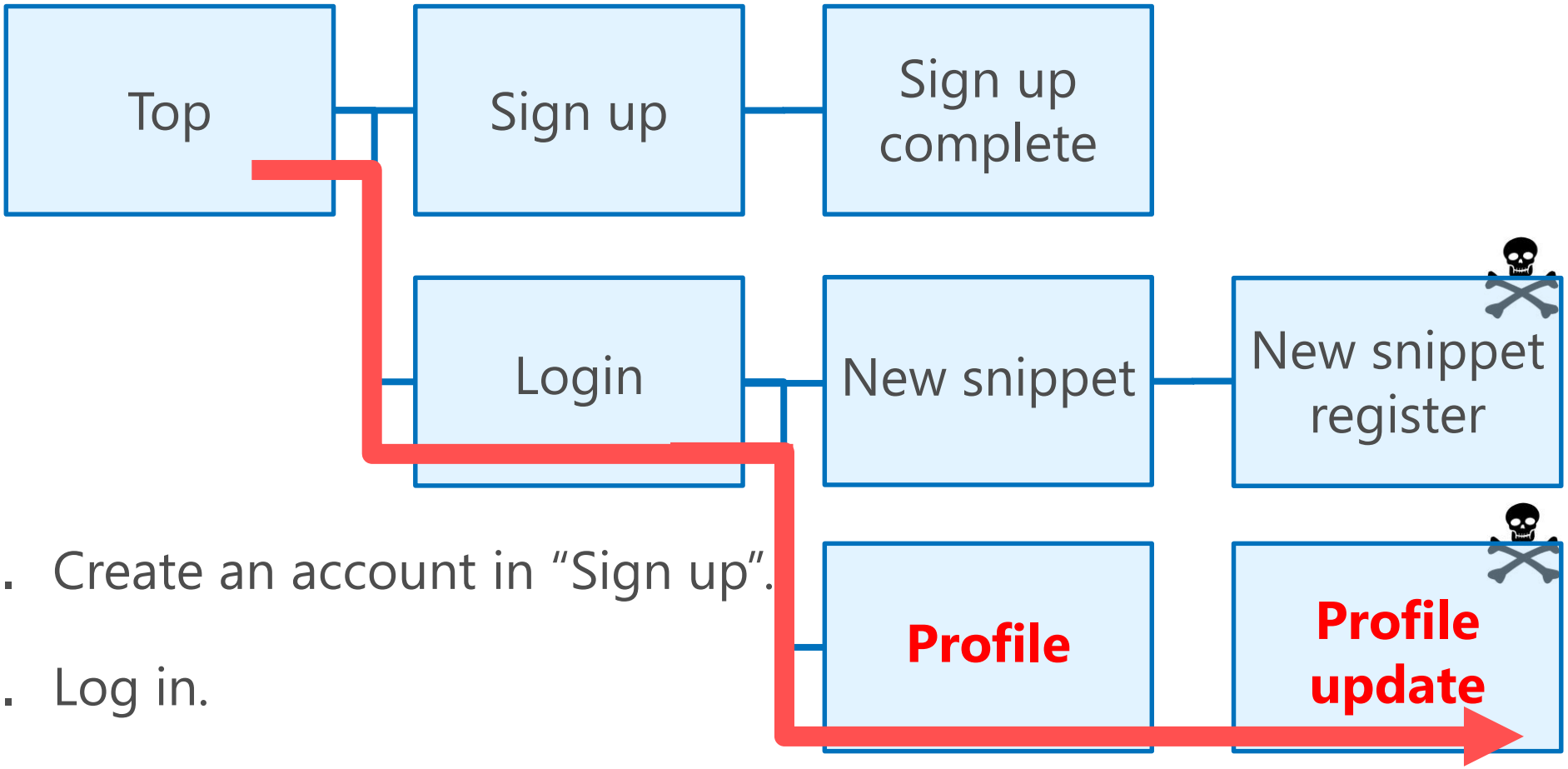
1. Create an account in “Sign up”.
2. **Log in.**
3. Register Snippet.
4. Update Profile.

Demonstration of SAIVS



1. Create an account in “Sign up”.
2. Log in.
3. **Register Snippet.**
4. Update Profile.

Demonstration of SAIVS



1. Create an account in “Sign up”.
2. Log in.
3. Register Snippet.
4. **Update Profile.**

Demonstration of SAIVS

Function name	Overview
New snippet register	output : BODY tag sanitize : exclude SCRIPT tag
Profile update	output : A tag sanitize : < , > ⇒ < , >

Demonstration of SAIVS

https://www.youtube.com/watch?v=N5d9oM0NcM0

Future prospects

- Strengthening the **page crawling** capability.
  - ✓ Responding to complex web apps.
  - ✓ Responding to CAPTCHA.
- Strengthening the **detecting vulnerabilities** capability.
  - ✓ Responding to complex RXSS patterns.
  - ✓ Responding to other vulnerabilities.

THANK YOU!

- **Download ".PDF" version of this document:**
  ≫ http://www.mbsd.jp