Black-box adversarial attacks on XSS attack detection model

《Computers & Security》

Qiuhua Wang, Hui Yang, Guohua Wu

2024.09.05
張家維

CONTENT

01.

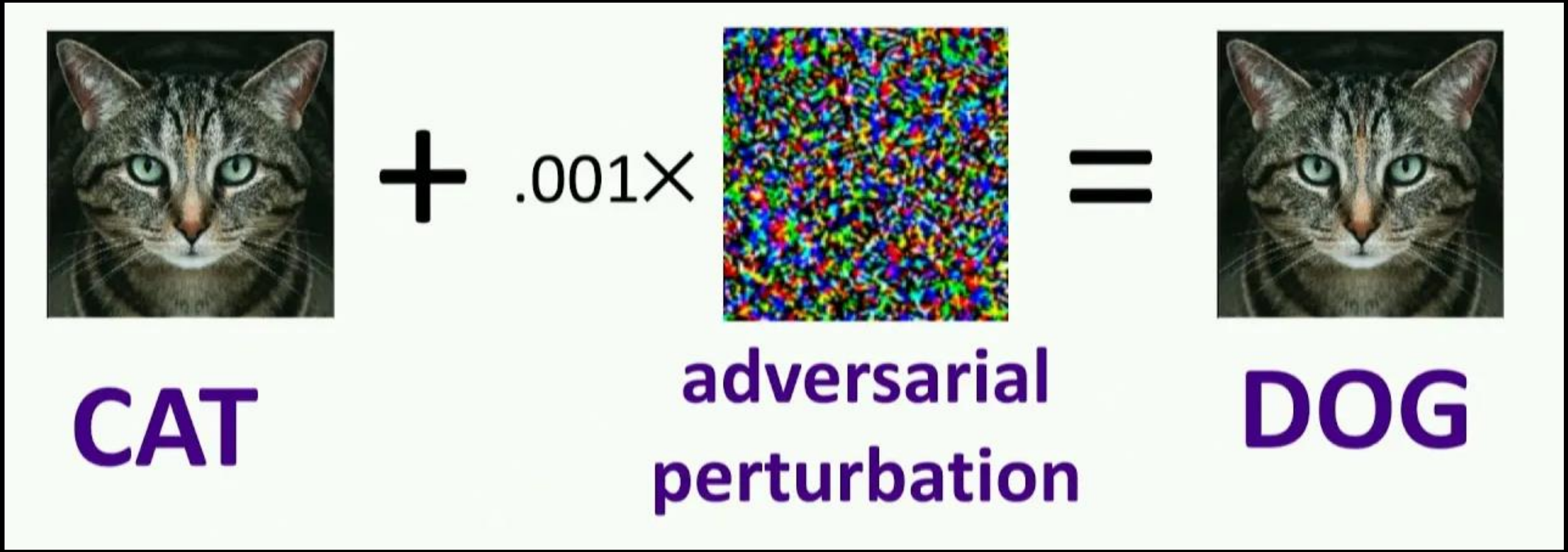
Introduction

Adversarial attack

Fool ML, DL models by introducing small, intentional perturbations to the input data.

- Defending against attacks caused vital processes by
    1. adversarial attack examples
    2. minimizing adversarial examples’ impact
- XSS scripts can only be changed based on fixed rules
- Few studies have focused on generating XSS adversarial attack examples.
- very low escape rate (ER)

Adversarial attack



Adversarial attack

檢測率 (Detection Rate, DR)

安全系統能夠成功檢測出所有已知威脅的比例

$$DR = \frac{\text{成功檢測的威脅數量}}{\text{所有已知威脅的總數}} \times 100\%$$

逃逸率 (Escape Rate, ER)

指未被檢測系統發現或攔截的威脅比例

$$ER = \frac{\text{未被檢測到的威脅數量}}{\text{所有已知威脅的總數}} \times 100\%$$

DR + ER = 100 %

- **DR (Detection Rate)**:
  The proportion of malicious XSS samples correctly identified as malicious.
- **ER (Escape Rate)**:
  The proportion of malicious XSS samples incorrectly identified as benign.

Soft Q-learning

Definition: Soft Q-learning is a variant of traditional Q-learning that incorporates a softmax policy to promote exploration by adding an entropy term to the reward.

- **Entropy Regularization:**

Encourages the agent to explore more diverse actions by penalizing deterministic policies.

- **Soft Bellman Equation:**

Updates the Q-function by considering both the expected reward and the entropy of the policy.

- **Objective:**

Maximizes a trade-off between the expected cumulative reward and the entropy of the policy, leading to more robust and exploratory behavior.

02.

Related Work

XSS Attacks

(1) Reflected (Non-Persistent) XSS Attack

Malicious scripts are reflected off the server and executed in the user's browser via a crafted link or form.

(2) Stored (Persistent) XSS Attack

Malicious scripts are stored on the server and executed when users visit the infected page.

(3) DOM-based XSS Attack

The attack happens within the browser, manipulating the DOM to execute malicious scripts.

Semantic analysis

(1) JavaScript source code into an abstract syntax tree

- analyze obfuscated malicious samples
- syntactic unit sequences
- FastText algorithm & Bi-LSTM

(2) XSSChop

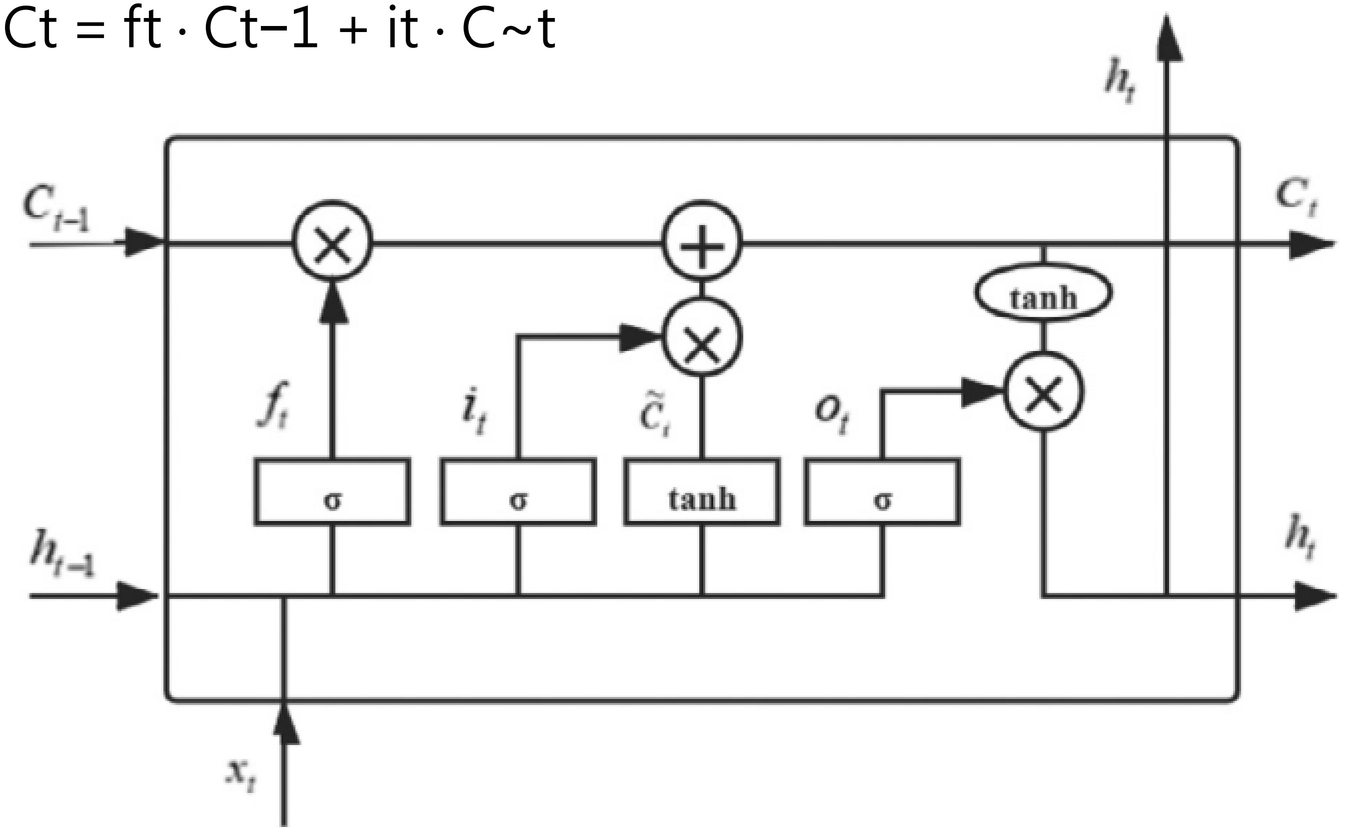
- scored the HTML semantic analysis and JS syntax
- high accuracy rate and a very low false positive rate
- low recall rate

LSTM

The LSTM network can converge better and faster, and can effectively improve the prediction accuracy.

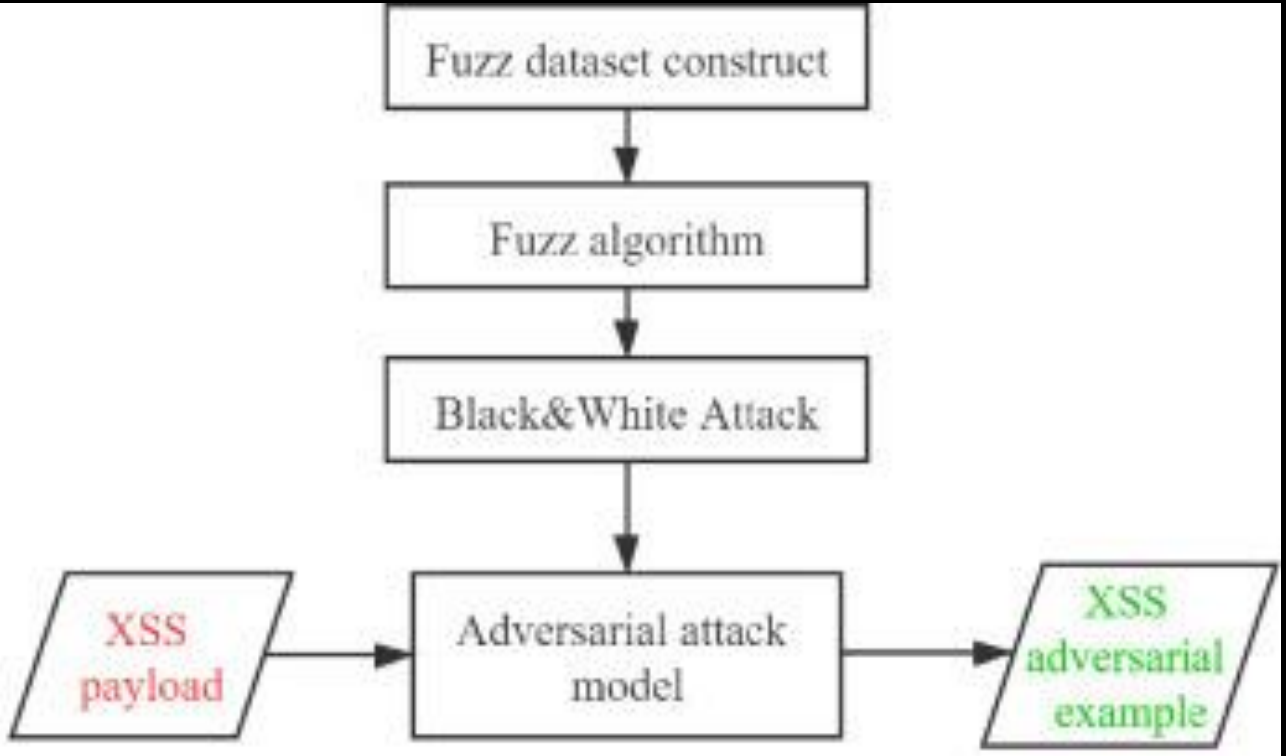
- ➢ forget gate ft
  - outputs 0 - 1
  - cell state Ct−1
- ➢ input gate it
  - sigmoid function
- ➢ cell gate gt
  - candidate cell state
- ➢ output gate ot.
  - determines next hidden state

Ct = ft · Ct−1 + it · C~t



03. Proposed approach

Conceptual Design |



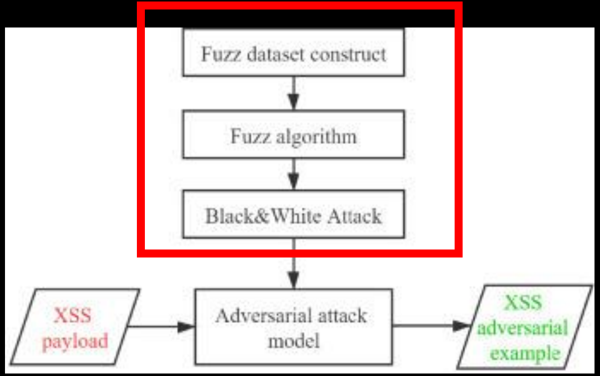
>_ Obtain the benign samples

- > crawler technology
- > construct a fuzz dataset

>_ Implement Black & White attack

- > improve the confidence coefficient of malicious samples
- > bypass strategy

FUZZ Algo and black&white attack



>_ Increase the confidence coefficient

> The strings with some regularity, some normal English words, and some benign examples in the XSS attack

>_ Construct FUZZ dataset

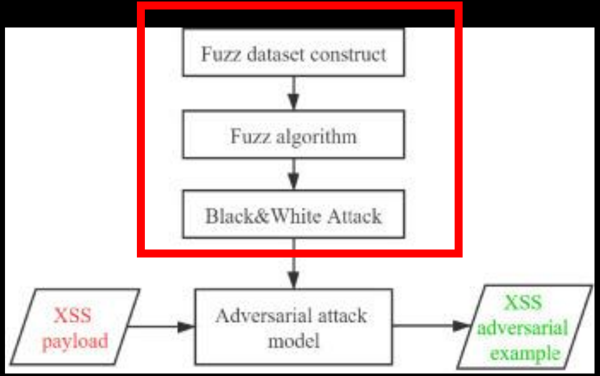
> 1. Web crawler: Alexa top 1000 website in depth first. maximum depth of 4 layers, , obtaining 142,621 request parameters

> 2. 100,000 commonly used words.

> 3. alphabetical order rules and numerical order rules 1000 strings.

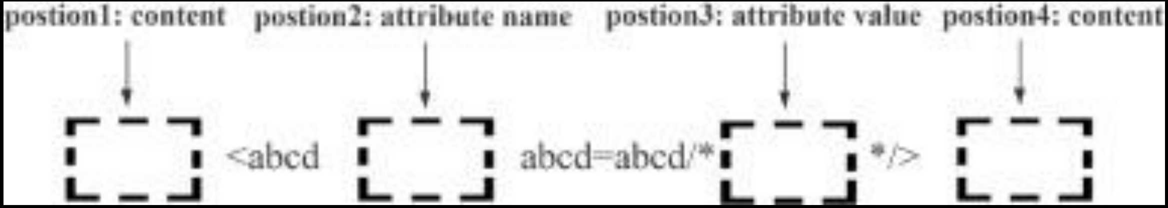
Alexa top1000 parameters:	jQuery112400655284765504 6845_1602697106492 jQuery11240655426066126 8738_1602724289306 wp_portlet_css__0.0%3Ahead_css osaka-sakaishi-miharaku MediaWiki:Stylesheets-Tabs.css AuthPortalPoolUS ......
English words top10w:	sense close subject turn town ......
Regular strings:	abcdefg abc%20abcd%20abcde abc%20abcd%20abcde%20abcdefgh 123456789 987654321 ......

FUZZ Algo and black&white attack



>_ XSS characteristics

> tag name, attribute name, attribute value, event and malicious JavaScript



>_ coefficient prediction:

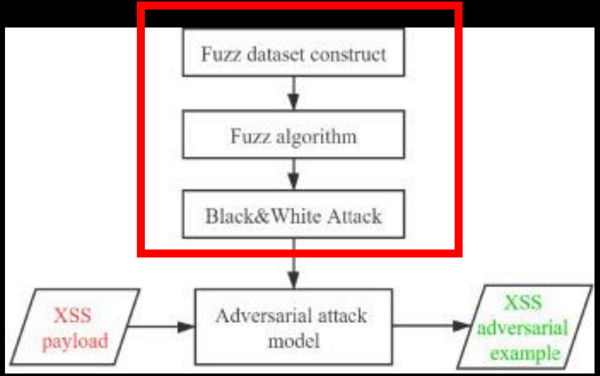
> LSTM-based: **0.6911**

> MLP-based: **0.4892**

>_ confidence coefficient:

XSS attack detection model identifies the data as malicious

FUZZ Algo and black&white attack



>_ Fuzz Algorithm

> Input Parameters

- **WORDS[]:** fuzz data strings
- **DM[]:** XSS detection models
- **P[]:** fuzz positions
- **Payload:** initial fuzz example

```
input  : WORDS[]:list fuzz data; DM[]: list XSS detection model; P[]: list fuzz postions; payload: a fuzz e
output: RES[]:fuzz results
WORDS[] ← list fuzz words;
DM[] ← list XSS detection model;
RES[] ← results;
POS[] ← list fuzz postions;
payload ← a fuzz example;
foreach fuzz postion p in POS do
    foreach each XSS detection model dm in DM do
        foreach each fuzzword in WORDS do
            Compute the confidence coefficient of the model dm when the fuzzword is added to the position
            Update RES
        end
    end
end
```

> Output

- **RES[]: stores the results of the fuzzing process**
- Sorted results

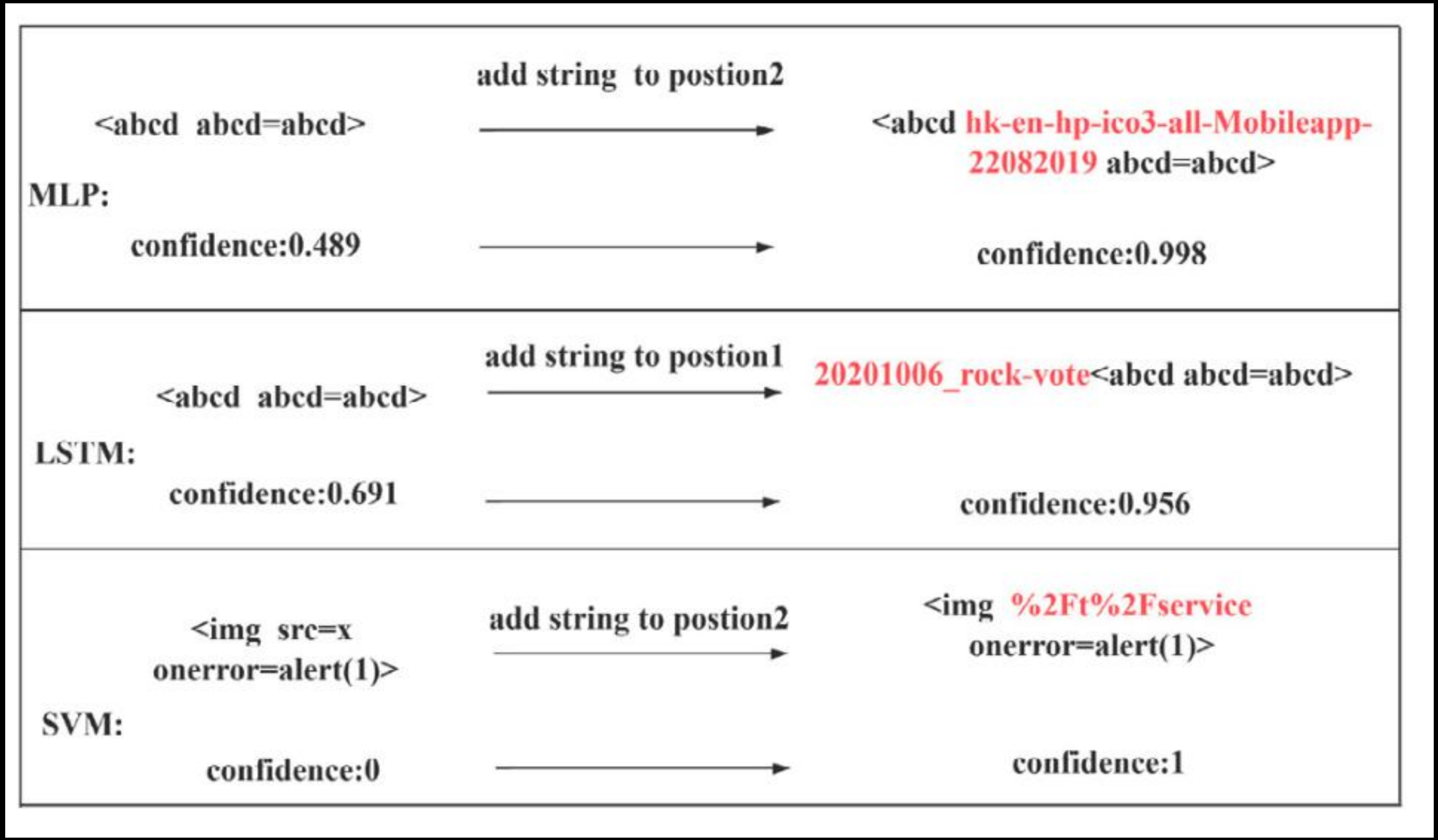
FUZZ Algo and black&white attack

>_ Fuzz Algorithm

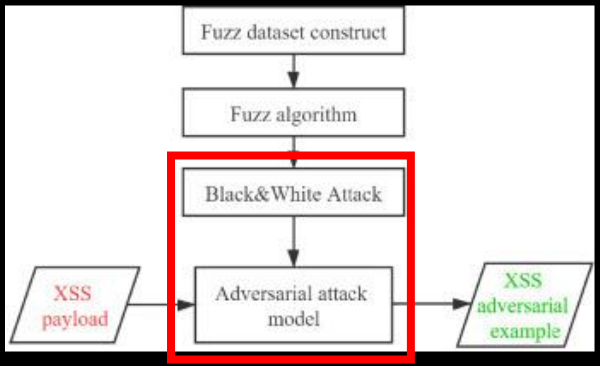
- **DM[]:** XSS detection models

> Detection Model:

1. DeepXSS
2. SVM
3. LSTM
4. MLP
5. SafeDog
6. XSSchop

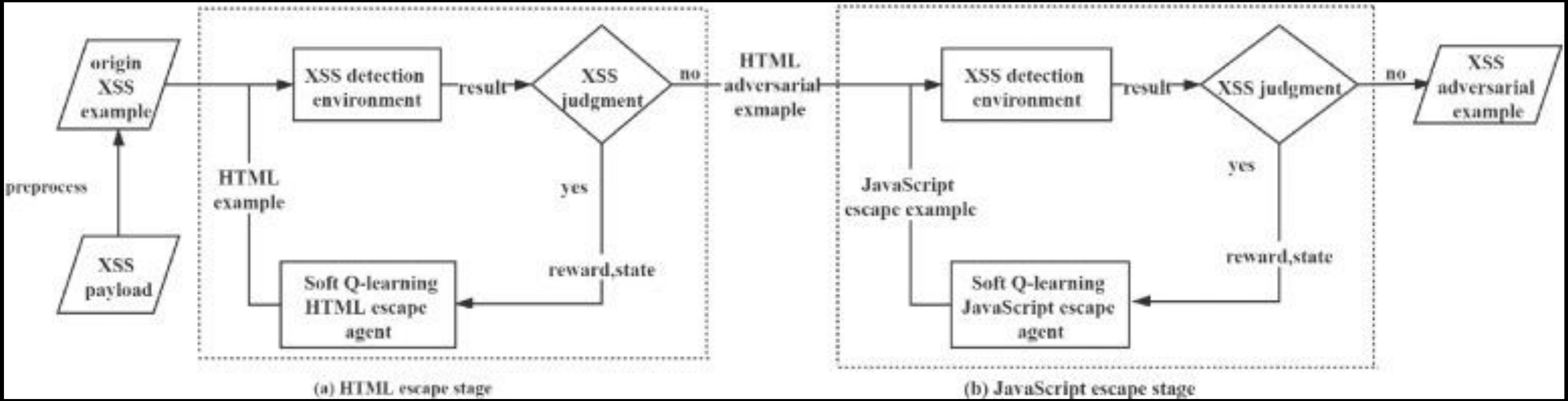


Black-box adversarial attack model



>_ Bypass XSS attack detection

- > Preprocess
- > HTML escape
- > JavaScript escape



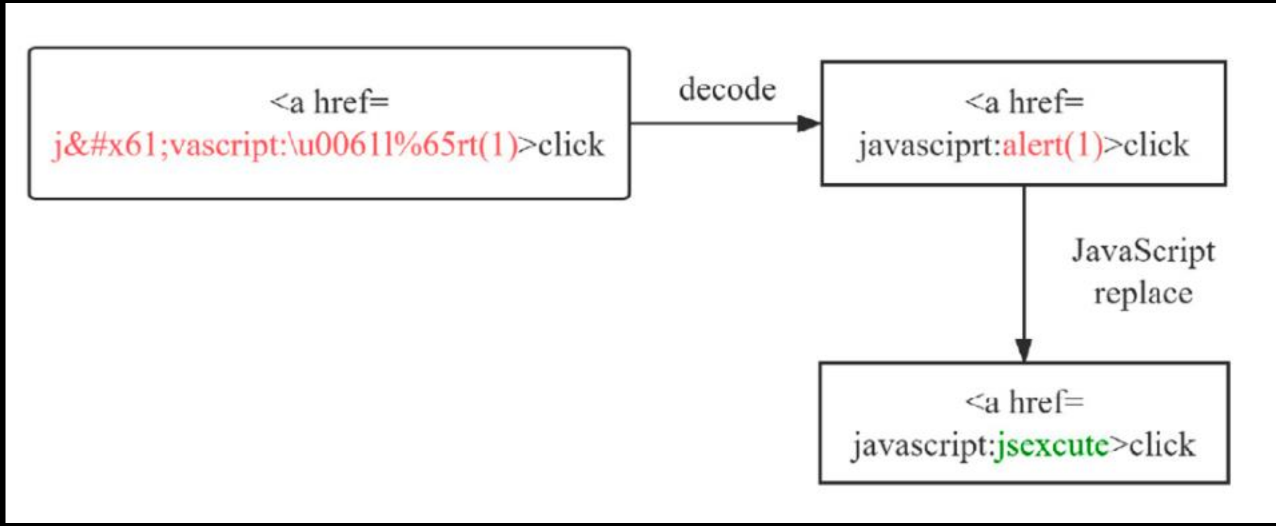
Black-box adversarial attack model

Black-box adversarial attack model

>_ Preprocess

(1) URL, HTML, and JavaScript decoding

(2) Malicious example replaced with a normal string



XSS Payload → execute JavaScript but not contain malicious code.

Black-box adversarial attack model

>_ HTML escape stage

(1) Black-box XSS detection environment

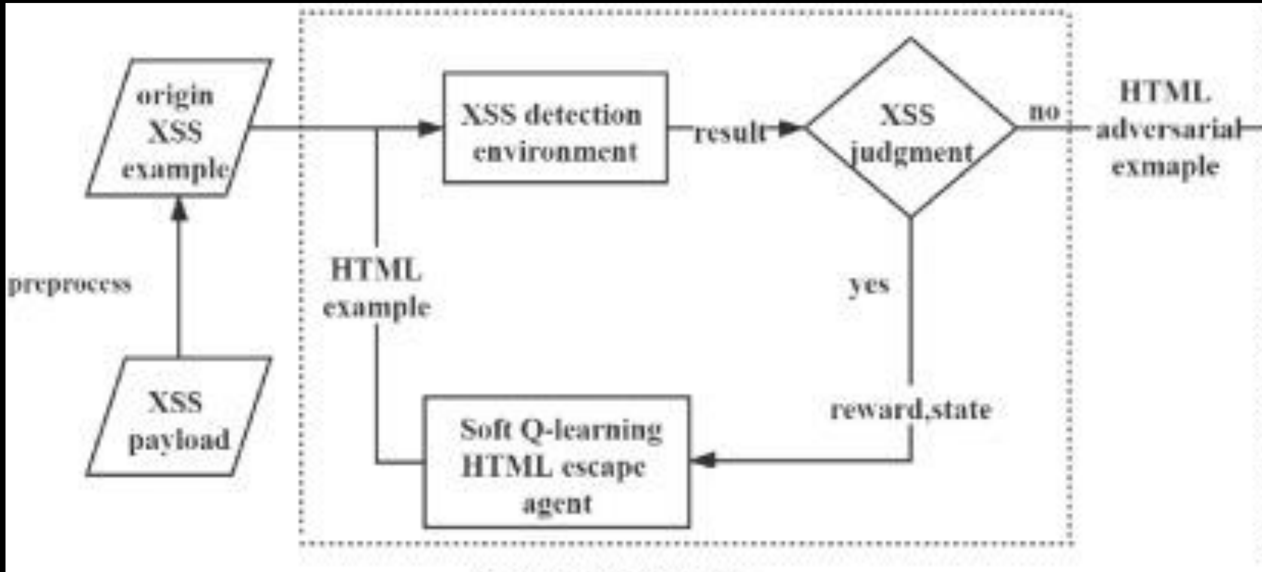
> only know the confidence coefficient

> State: payload structure

> Reward: (result new – result old) * score

> score is a fixed reward value

1. DeepXSS
2. SVM
3. LSTM
4. MLP
5. SafeDog
6. XSSchop



(a) HTML escape stage

Black-box adversarial attack model

>_ HTML escape stage

(2) Soft Q-learning-based HTML escape agent

> Soft Q-learning

> select the best HTML escape action

> HTML AST Analyzer

> Parse5(2020) HTML parsing engine

>> HTML adversarial example

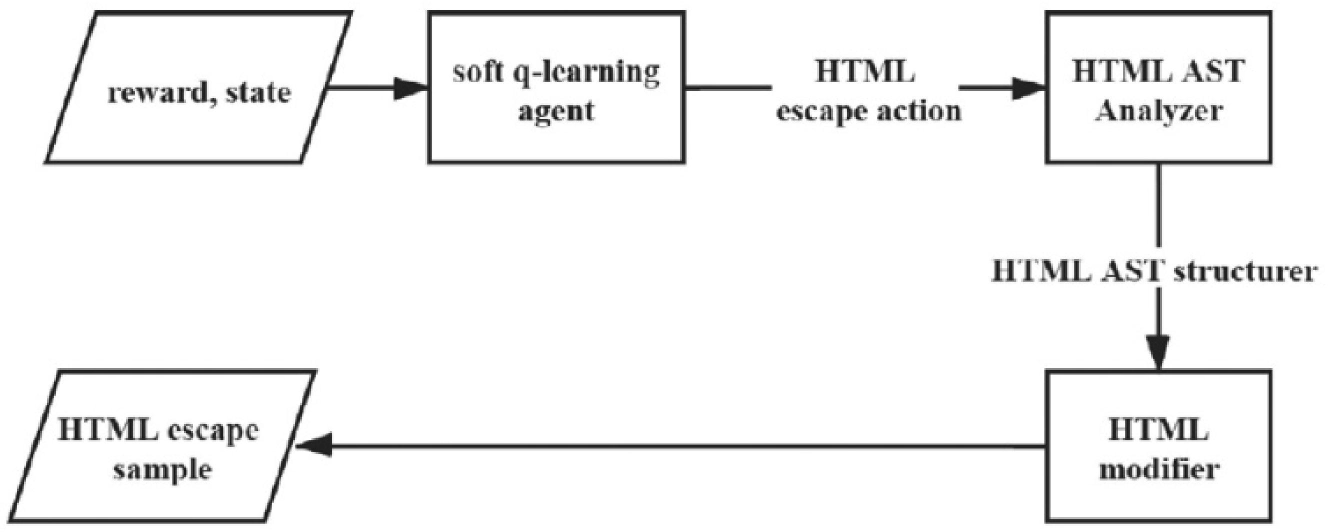


Fig. 10. HTML Escape Agent.

Black-box adversarial attack model

>_ HTML escape stage

(3) HTML escape action

> 3.1 String substitution

tag name and attribute name

%20, %09, %0a, %0d, and %0c

(space, Tab, LF, CR, FF)

> 3.2 Character coding

(a) HTML encoding;
(b) Add zero to the middle of HTML entity codes;
(c) Convert the decimal and hexadecimal.

> 3.3 String addition

(a) Add %0d, : 	
 %09 and %0a between ”javascript:”;

(b)Add %10-%1f before ”javascript:”;

(c)Add random strings before or after the tag;

(d)Add the corresponding special string

(e)Add ”%26%2362%3B” string between attributes

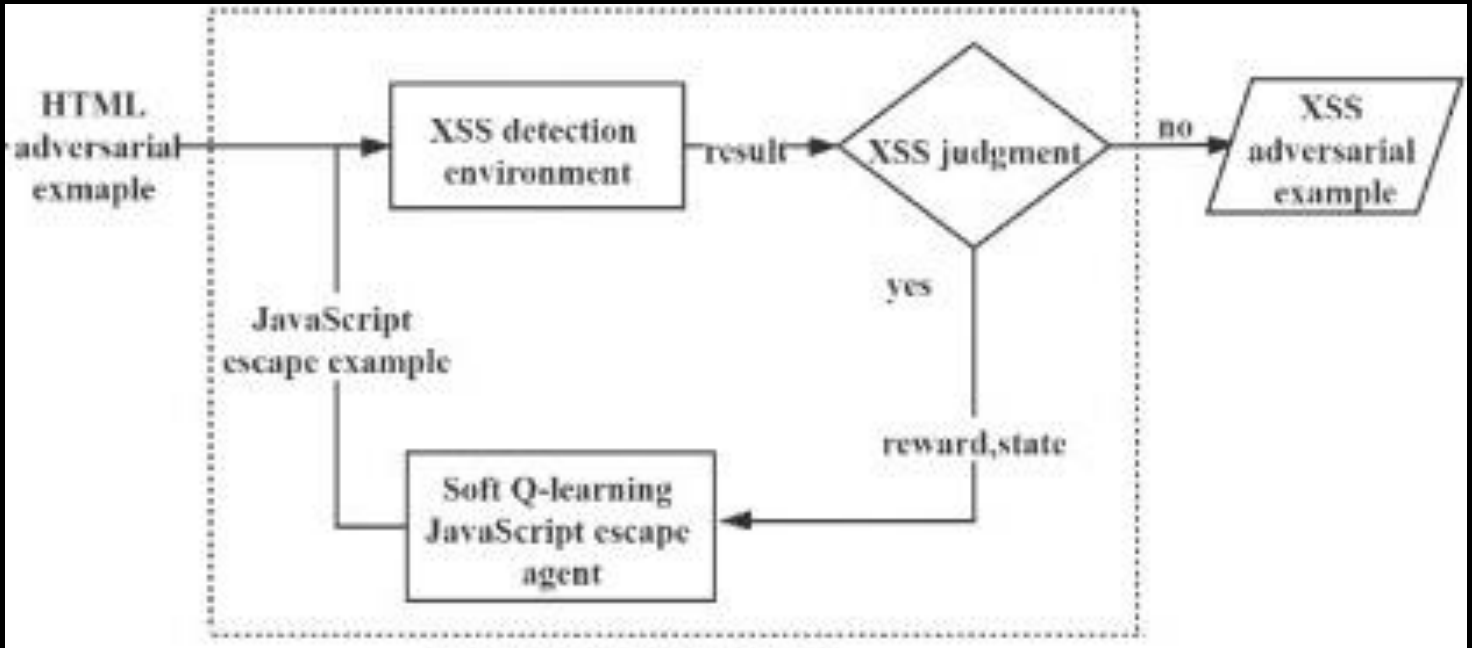
(f)Add %0a, %0c, %0d, %09, before attributes.

Black-box adversarial attack model

>_ Javascript escape stage

> Soft Q-learning agent is similar to HTML escape stage

> JavaScript AST analyzer (@babel/parser 2020)



(b) JavaScript escape stage

Black-box adversarial attack model

>_ Javascript escape action

> Add %0a, %0b, %0c, %0d, %09 before (, [

> JavaScript is randomly encoded into HTML entities

> Unicode encoding of the JavaScript code

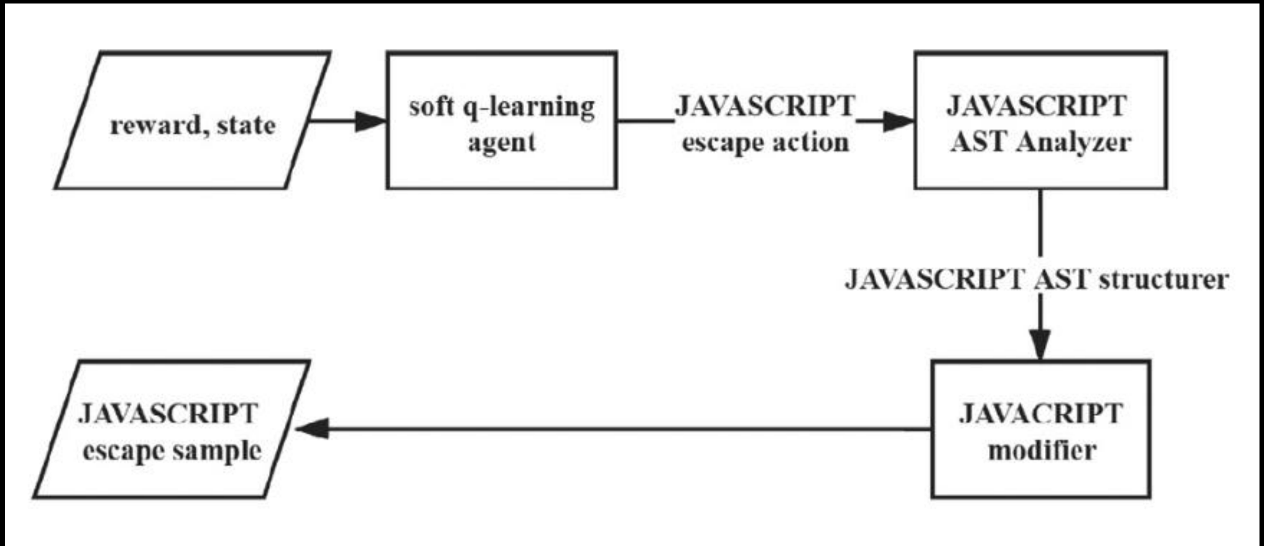
> Add a random string at the comment

> URL encoding of the string ”javascript”

> Add the corresponding special string at the comment

> Jsfuck encoding

> JavaScript emoticons encryption.



04. Experiment Result

Dateset

> DeepXSS dataset (2018)

>> 40,637 malicious XSS examples

>> 31,407 normal requests on XSSED

> PortSwigger XSS cheat sheet (2020)

> construct malicious examples according to ECMAScript (2020)

> extracted 20 malicious JavaScript examples from XSSED (2012)

Experimental Results

Table 1. Comparative Experiment Of Different XSS Attack Detection Model.

Classifier	Precision	Recall	F1	Accuracy
XSSChop	0.9999	0.8243	0.9036	0.8494
MLP	0.9983	0.9837	0.9910	0.9908
LSTM	0.9991	0.9842	0.9916	0.9914
SafeDog	0.9972	0.9286	0.9617	0.9366
SVM	0.9984	0.9776	0.9879	0.9877
DeepXSS	0.995	0.979	0.987	-

> **MLP and LSTM models are higher than others**

Experimental Results

> Accuracy & Loss

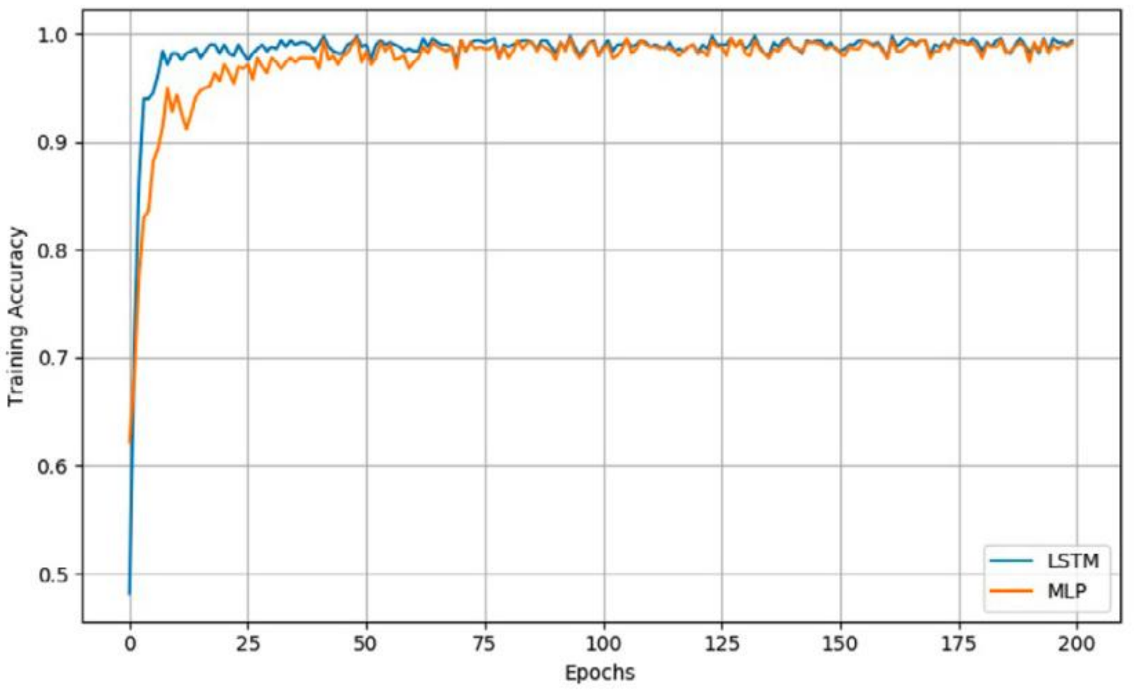


Fig. 12. Accuracy Of The Training Dataset.

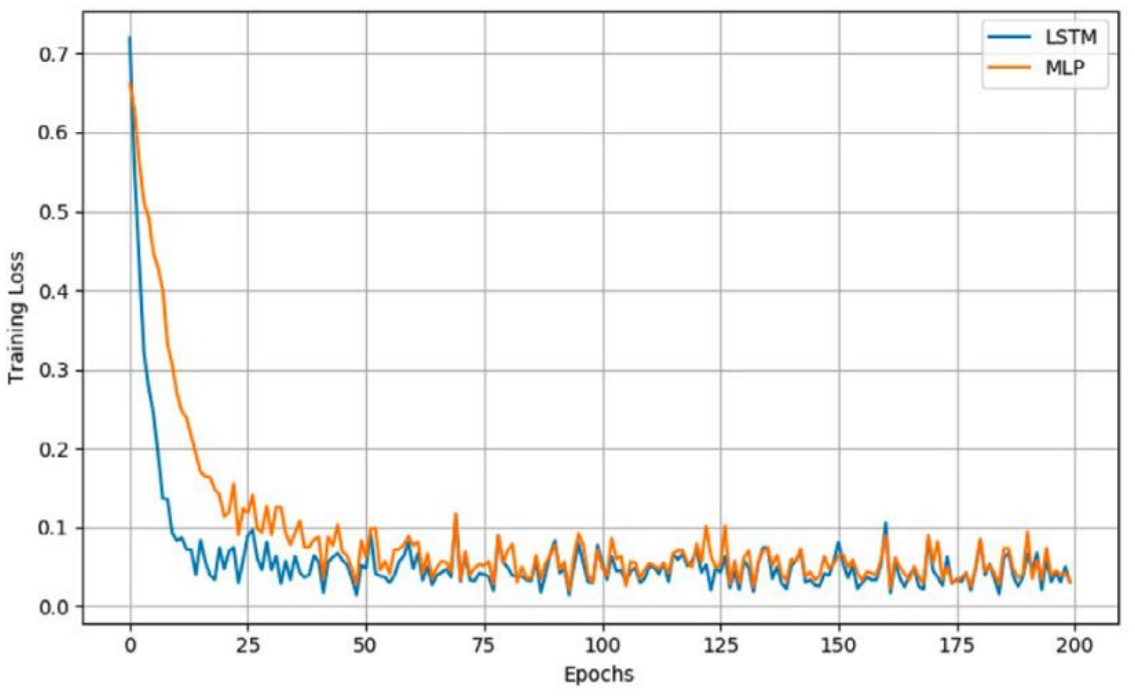


Fig. 13. Loss Of The Training Dataset.

Experimental Results

> Adversarial attack model

Table 2. Evaluating results.

Model	Our method (Our dataset)		Our method (RLXSS dataset)		RLXSS	
	DR	ER	DR	ER	DR	ER
SafeDog	0.031	**0.969**	0.057	0.943	0.859	0.141
XSSChop	0.058	**0.942**	0.879	0.121	0.907	0.093
LSTM	0.138	**0.862**	0.241	0.759	0.9175	0.0825
MLP	0.047	**0.953**	0.119	0.881	-	-
SVM	0.141	**0.859**	0.198	0.802	-	-

ER over 85%, far exceeding RLXSS (2019) results. → dataset

05. Conclusions

Contributions

1. Adversarial attack model based on Soft Q-learning

2. The Black&White attack method tests XSS detection models using machine learning and deep learning

3. Interfere with XSS attack detection models' classification

4. XSS adversarial attack examples

Q & A