ConvXSS: A deep learning-based smart ICT framework against code injection attacks for HTML5 web applications in sustainable smart city infrastructure

《Sustainable Cities and Society》


Koundinya Kuppa

Institution and department
Birla Institute of Technology and Science Pilani ·
Department of Computer Science and Information Systems

Anushka Dayal

Institution and department
Birla Institute of Technology and Science Pilani ·
Department of Humanities and Languages


Shashank Gupta
PhD · Assistant Professor

Institution and department
Birla Institute of Technology and Science Pilani ·
Department of Computer Science and Information Systems

2023.06.13
張家維

Sustainable Cities and Society

ISSN

2210-6707

EISSN

2210-6715

JCR ABBREVIATION

SUSTAIN CITIES SOC

ISO ABBREVIATION

Sust. Cities Soc.

Journal information

EDITION

Science Citation Index Expanded (SCIE)

CATEGORY

GREEN & SUSTAINABLE SCIENCE & TECHNOLOGY - SCIE

ENERGY & FUELS - SCIE

CONSTRUCTION & BUILDING TECHNOLOGY - SCIE

LANGUAGES	REGION	1ST ELECTRONIC JCR YEAR
English	NETHERLANDS	2015

Publisher information

PUBLISHER	ADDRESS	PUBLICATION FREQUENCY
ELSEVIER	RADARWEG 29, 1043 NX AMSTERDAM, NETHERLANDS	8 issues/year

Journal Citation Reports™ Journals Categories Publishers Countries/Regions My favorites Sign In Register

Search results > Journal profile

Favorite Export

JCR Year
2022

Sustainable Cities and Society

ISSN
2210-6707

EISSN
2210-6715

Journal information

EDITION
Science Citation Index Expanded (SCIE)

CATEGORY
GREEN & SUSTAINABLE SCIENCE & TECHNOLOGY - SCIE

CONSTRUCTION & BUILDING TECHNOLOGY - SCIE

《Sustainable Cities and Society》

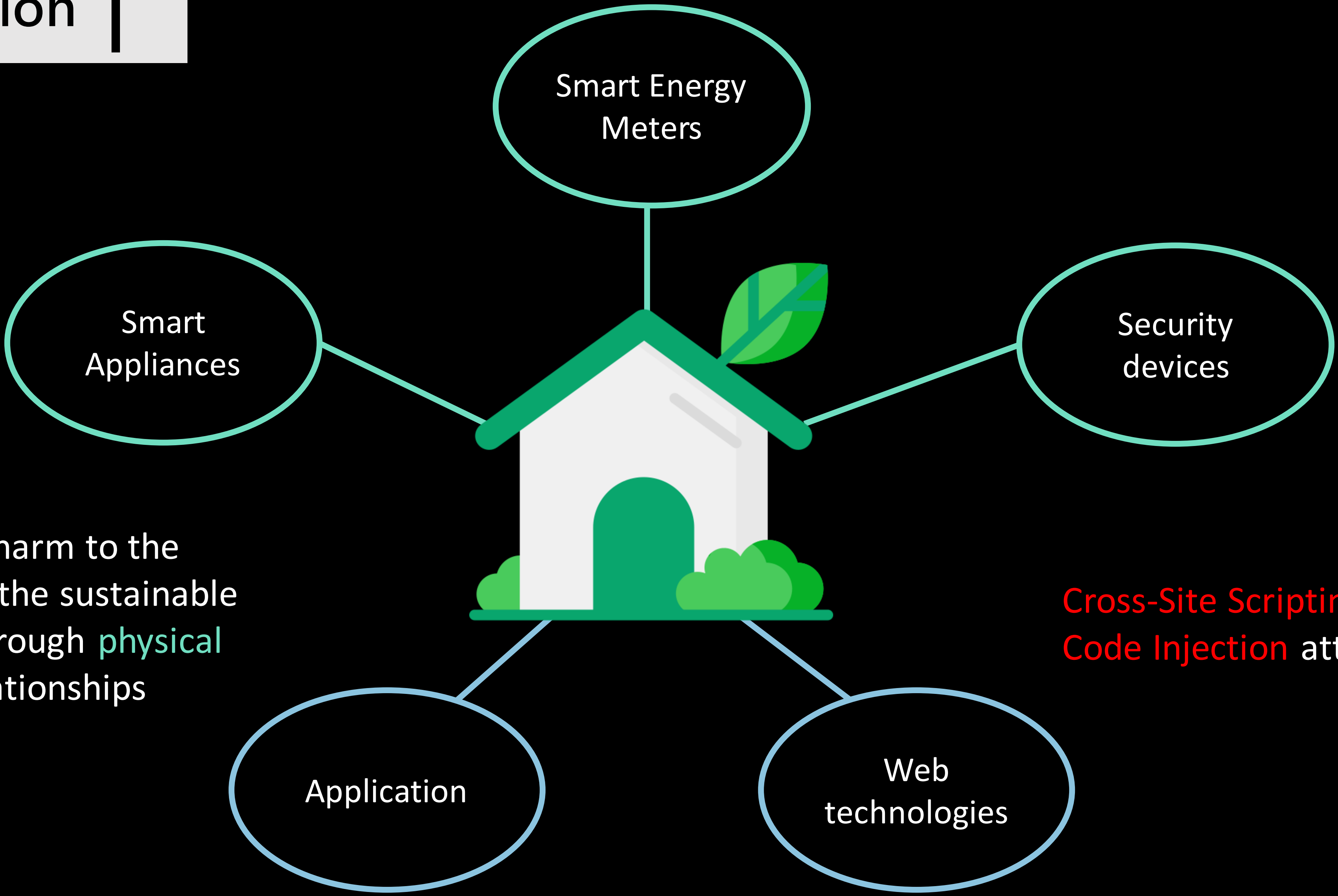
Year	Total Citations	Journal impact factor	JIF without self cites	5 Year Impact Factor	Immediacy Index	Citable items	% of articles in Citable items	Average JIF Percentile
2022	36,111	11.7	10.1	10.6	3.3	827	96.01	91.6
2021	26,556	10.696	8.678	9.908	3.685	918	96.73	90.313
2020	14,373	7.587	6.046	7.308	2.724	709	95.35	86.410
2019	7,140	5.268	4.115	5.143	1.660	521	91.75	80.383
2018	3,924	4.624	3.484	4.466	1.169	485	92.78	80.114
2017	1,788	3.073	2.521	3.160	0.879	281	94.31	64.930
2016	682	1.777	1.701	1.968	0.391	169	96.45	48.808
2015	204	1.044	1.009	1.023	0.172	134	96.27	32.535

CONTENT

01.

Introduction

Introduction



Prevention of harm to the inhabitants of the sustainable smart cities through physical and digital relationships

Cross-Site Scripting (XSS) and Code Injection attacks

Information and Communications Technology

資訊和通訊技術

ICT 是一個廣泛的主題

- 通訊技術著重於訊息傳播的傳送技術
- 資訊科技則著重於資訊的編碼或解碼，以及在通訊

載體上的傳輸方式。

中華民國統計資訊網 ICT產業範圍:

CR.電子零組件製造業

CS.電腦、電子產品及光學製品製造業

JB.電信業

JC.電腦相關及資訊服務業

ICT Development Index (IDI)

United Nations

International Telecommunication Union

- 一項綜合指標，用於評估世界各國在信息和

通信技術ICT 發展方面的相對進步和現狀。

Access & Usage & Skills

		Aggregate Scores (0-100)			IDI 2023 Indicator values (2021)						
					Universal Connectivity indicators			Meaningful Connectivity i			
	Country/Territory	ICT Development Index (IDI)	Universal Connectivity Pillar	Meaningful Connectivity Pillar	Individuals using the Internet (%)	Households with Internet access at home (%)	Active mobile-broadband subscriptions per 100 inhabitants	Population covered by at least a 3G mobile network (%)	Population covered by at least a 4G/LTE mobile network (%)	Mobile broadband Internet traffic per subscription (GB)	Fixed broadband Internet traffic per subscription (GB)
1	Kuwait	**98.2**	97.0	99.3	99.7	99.4	136.6	100.0	100.0	657.8	8 205.6
2	Singapore	**97.4**	99.4	95.4	96.9	99.3	147.5	100.0	100.0	87.9	N/A
3	Qatar	**97.3**	98.7	96.0	99.7	95.0	144.0	100.0	99.8	140.2	10 484.5
4	Denmark	**96.9**	98.2	95.6	98.9	96.1	141.8	100.0	100.0	176.8	4 132.5
5	Estonia	**96.9**	97.5	96.4	91.0	91.8	180.1	100.0	99.0	222.8	N/A
6	Finland	**96.7**	98.1	95.2	92.8	91.7	157.2	99.9	99.9	398.9	1 013.9
7	United States	**96.6**	99.1	94.1	96.8	92.5	165.8	99.9	99.9	101.5	N/A
8	Bahrain	**96.5**	96.7	96.2	100.0	100.0	135.2	100.0	100.0	294.5	4 773.3
9	Hong Kong	**96.5**	99.1	93.8	93.1	94.4	160.3	99.0	99.0	99.1	3 853.0
10	United Arab Emirates	**96.4**	100.0	92.8	100.0	99.9	241.2	100.0	99.8	52.7	5 183.2

Reference: https://en.wikipedia.org/wiki/Information_and_communications_technology

Introduction

Rank	Name of Risk	Attack Vectors		Security Weakness		Impacts		Score
		Threat Agents	Exploitability	Prevalence	Detectability	Technical	Business	
1	Injection	App Specific	Easy: 3	Common:2	Easy: 3	Severe: 3	App Specific	8.0
2	Authentication	App Specific	Easy: 3	Common:2	Average: 2	Severe: 3	App Specific	7.0
3	Sens. Data Exposure	App Specific	Average: 2	Widespread:3	Average: 2	Severe: 3	App Specific	7.0
4	XML External Entities (XXE)	App Specific	Average: 2	Common:2	Easy: 3	Severe: 3	App Specific	7.0
5	Broken Access Control	App Specific	Average: 2	Common:2	Average: 2	Severe: 3	App Specific	6.0
6	Security Misconfiguration	App Specific	Easy: 3	Widespread:3	Easy: 3	Moderate: 2	App Specific	6.0
7	Cross-Site Scripting	App Specific	Easy: 3	Widespread:3	Easy: 3	Moderate: 2	App Specific	6.0
8	Insecure Deserialization	App Specific	Difficult: 1	Common:2	Average: 2	Severe: 3	App Specific	5.0
9	Vulnerable Components	App Specific	Average: 2	Widespread:3	Average: 2	Moderate: 2	App Specific	4.7
10	Insufficient Logging & Monitoring	App Specific	Average: 2	Widespread:3	Difficult: 1	Moderate: 2	App Specific	4.0

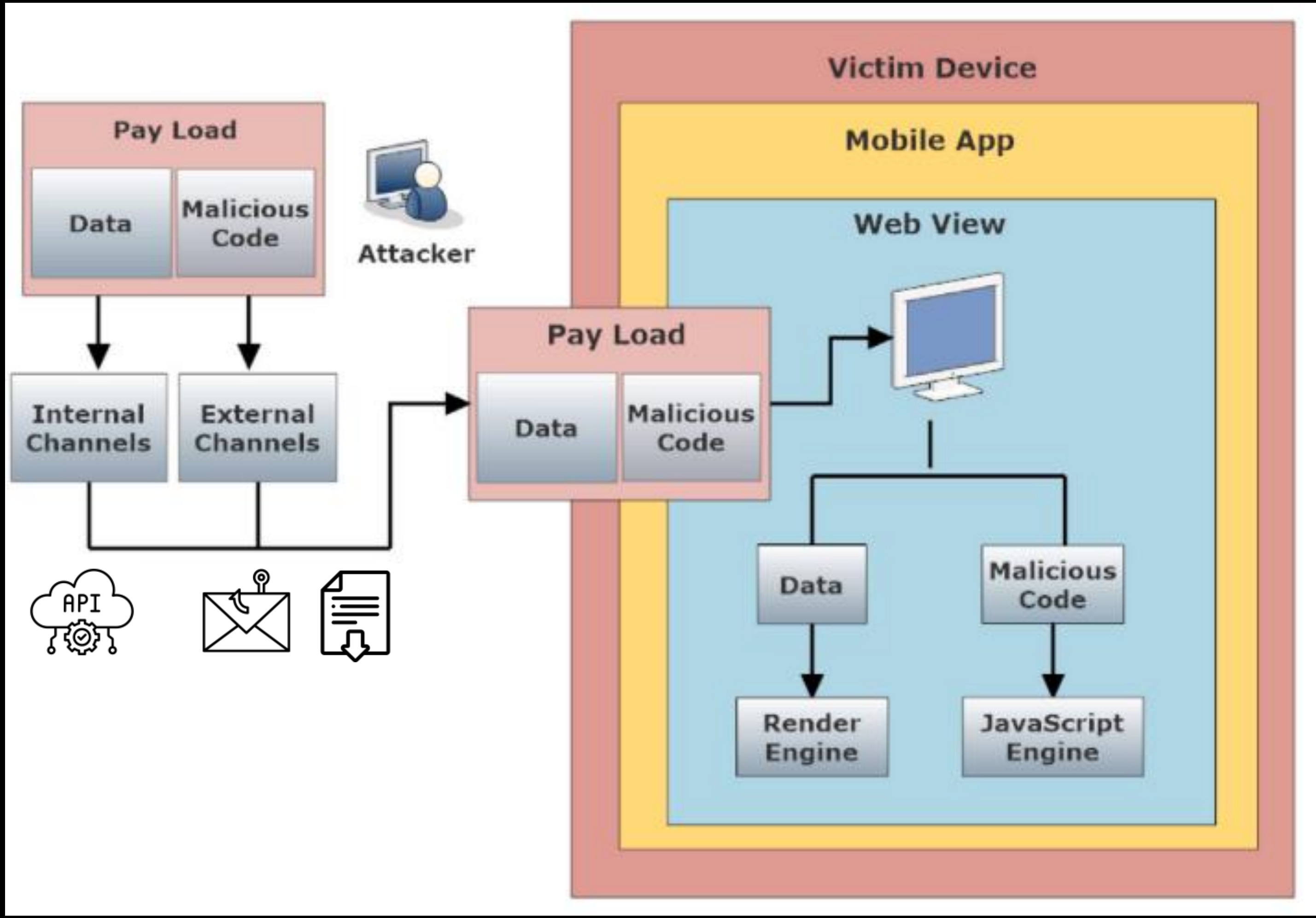
OWASP 2017 Top10

02.

Related Work

Attack Flow

Code Injection Attack on HTML-5 based Apps to compromise security of sustainable smart cities.



Based on the approach of the attack:

- **Data Channels:**
pathways for transmitting actual data.
Ex. 2D barcode, RFID tag.

- **Metadata Channels:**
transmission of descriptive or attribute information
Ex. file titles, artist names, file types

- **ID Channels:**
identification information used to identify and connect devices
Ex. Wi-Fi SSIDs, Bluetooth device names

Deep Learning

"Why Deep Learning?"

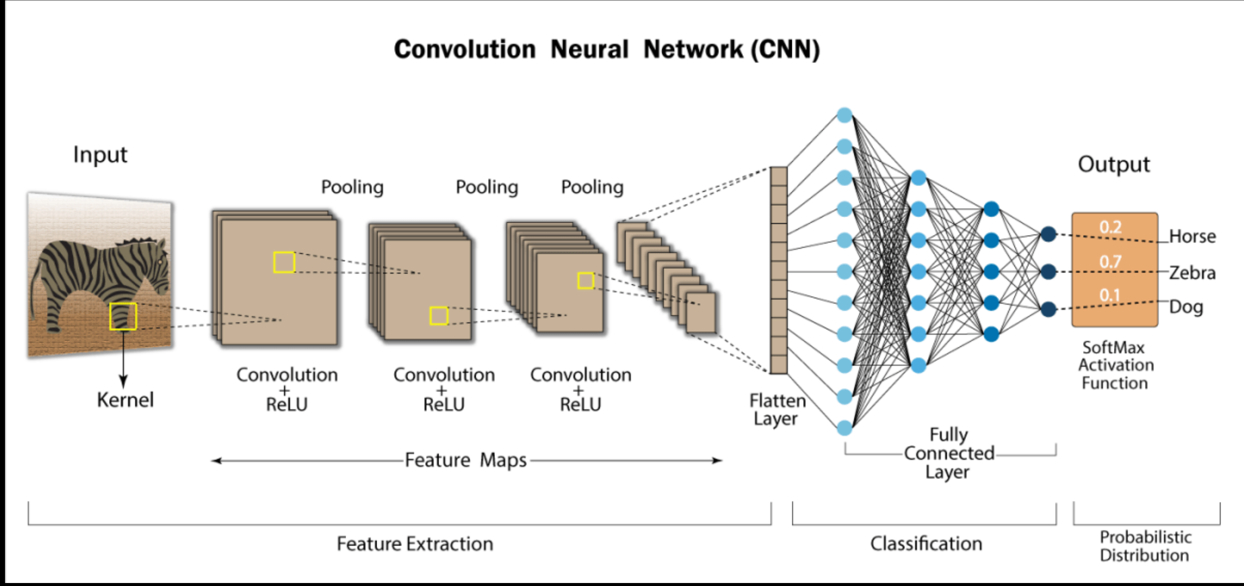
~~Deep Learning, a looming field of research in machine learning~~.

Deep learning could be supervised, semi-supervised or unsupervised.

Supervised	Unsupervised	Semi-supervised
所有資料都被「標註」(label)，告訴機器相對應的值。 常見的監督學習算法包括: 回歸（如線性回歸、邏輯回歸） 分類（如支持向量機、決策樹、隨機森林、神經網絡等） 優: 高精確度 缺: label易受人為錯誤影響	所有資料都沒有標註，機器透過尋找資料的特徵，自己進行分類。 常見的非監督學習算法包括: 分群（如K-means、層次聚類） 降維（如主成分分析（PCA）、自編碼器）。 優: 大規模，適用探索性數據分析 缺: 解釋困難、可能無意義	對少部分資料進行「標註」，電腦只要透過labeled data找出特徵並對其它的資料進行分類。 標註成本較低，並且可以適用於大部分的任務。~~目前最常用~~ 優: 減少對標註數據的依賴 缺: 複雜度增加

CNN

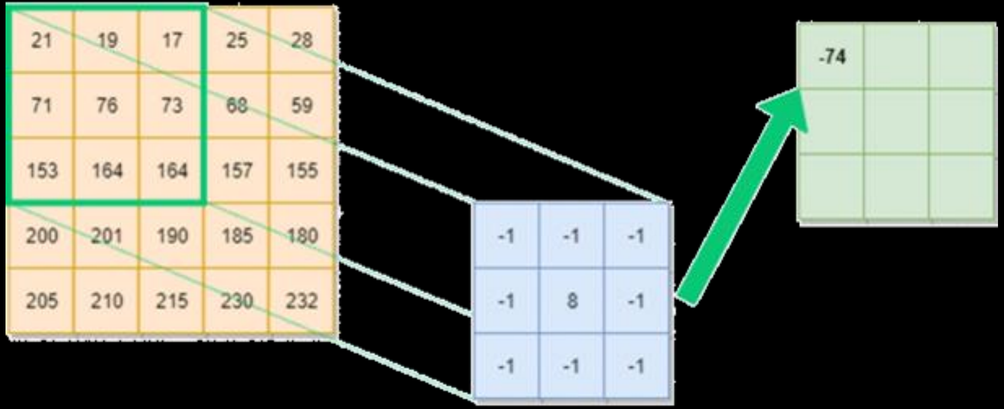
“ConvXSS , Why CNN ? ”



Convolutional Neural Networks (CNN)

- 卷積層（Convolution layer）：

  卷積層可以產生一組平行的feature map，通過在輸入圖像上滑動不同的卷積核並執行一定的運算而組成。



- 池化層（Pooling Layer）：

  在保留重要特徵情況下，減少特徵圖的空間維度，從而降低計算量和存儲需求。

- 攤平（Flatten）:

  將高維數據（如 2D 或 3D 特徵圖）轉換為一維向量。

258	258
258	258

63	89.5
130.75	97.75

CNN

“ConvXSS , Why CNN ? ”

- Low computational prerequisites (no consecutive output)
- Not a single-dimensional input
- Diminished parameter set
- Better at the efficiency

Detection

"How to detect malicious code? "

- ~~Virus signature is found in a list~~

☞ Lack of up-to-date signature files

- ~~Pre-established rules by security experts~~

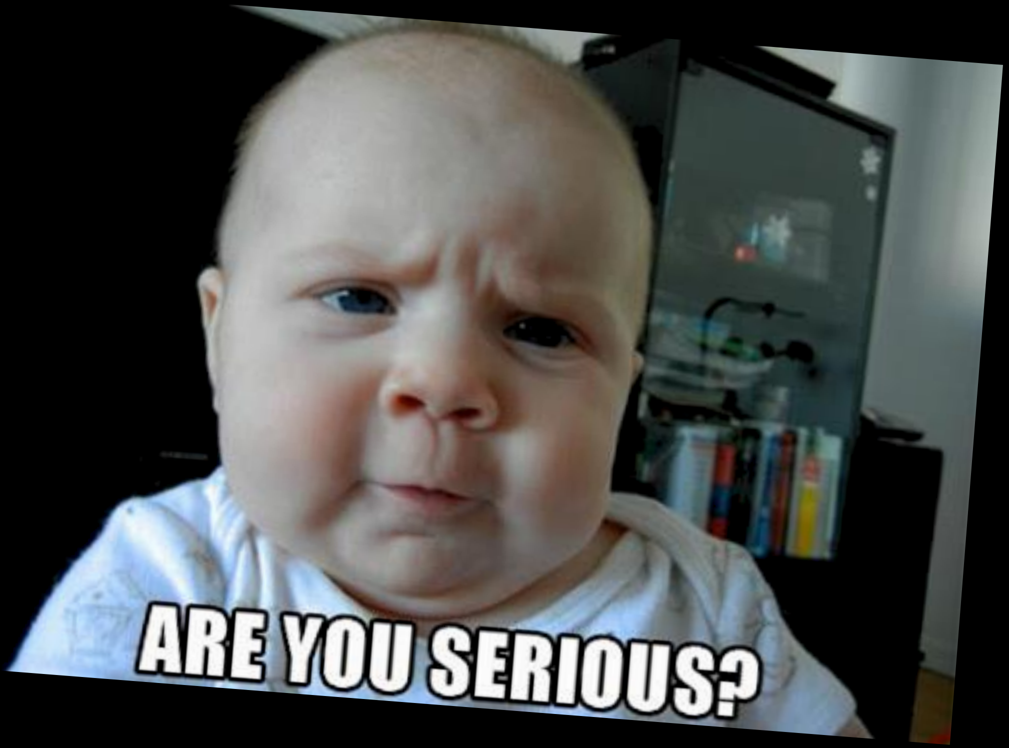
☞ Only identify the familiar code

- ~~Dynamic analysis methods~~

☞ Time-consuming & Can not dynamically protect

- ~~Honeypot, drive-by downloads and heap spraying~~

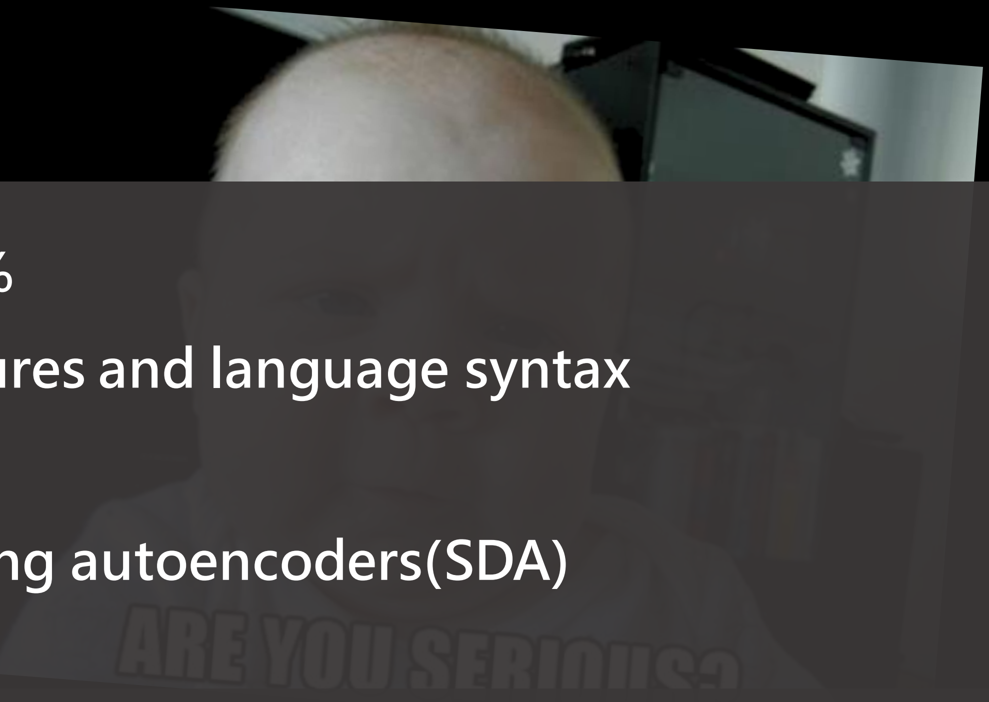
☞ designed for specific attacks & cannot cater the ever-evolving attacks



Detection

"How to detect malicious code? "

- static, non-linear, SVM: accuracy 94.38%
- Random Forests: detect behavioral features and language syntax
- binary measures: Top rate of accuracy
- word frequency(TFID) / stacked denoising autoencoders(SDA)
- Logistic Regression, SVM + SDA feature



CNN model

&

Sanitize malicious code

03. Proposed approach

Framework: ConvXSS

1. Input: JavaScript code

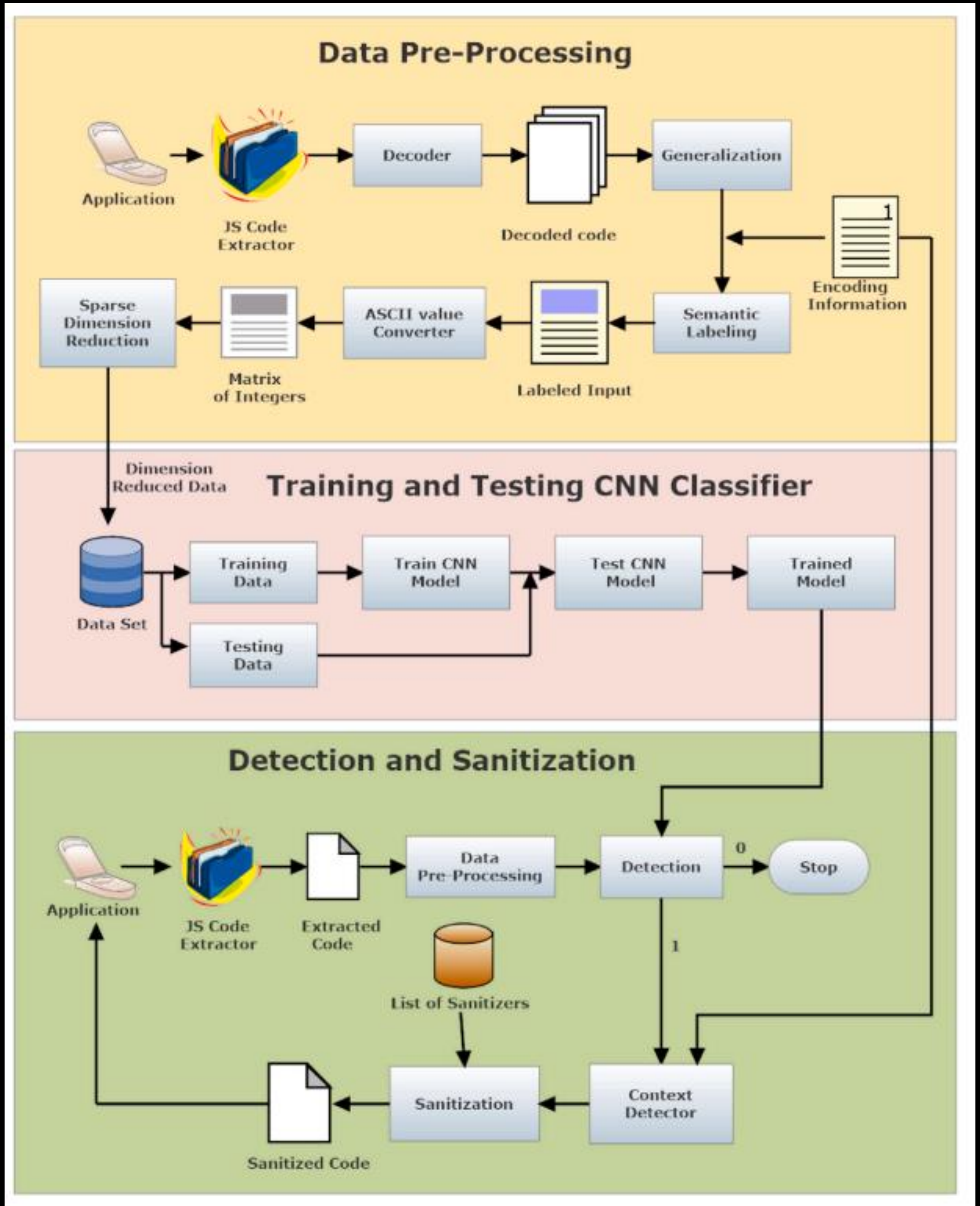
(1) decoding, generalization, semantic labeling
(2) binary vectors based on the ASCII
(3) dataset of 2d matrix

2. Training and Testing

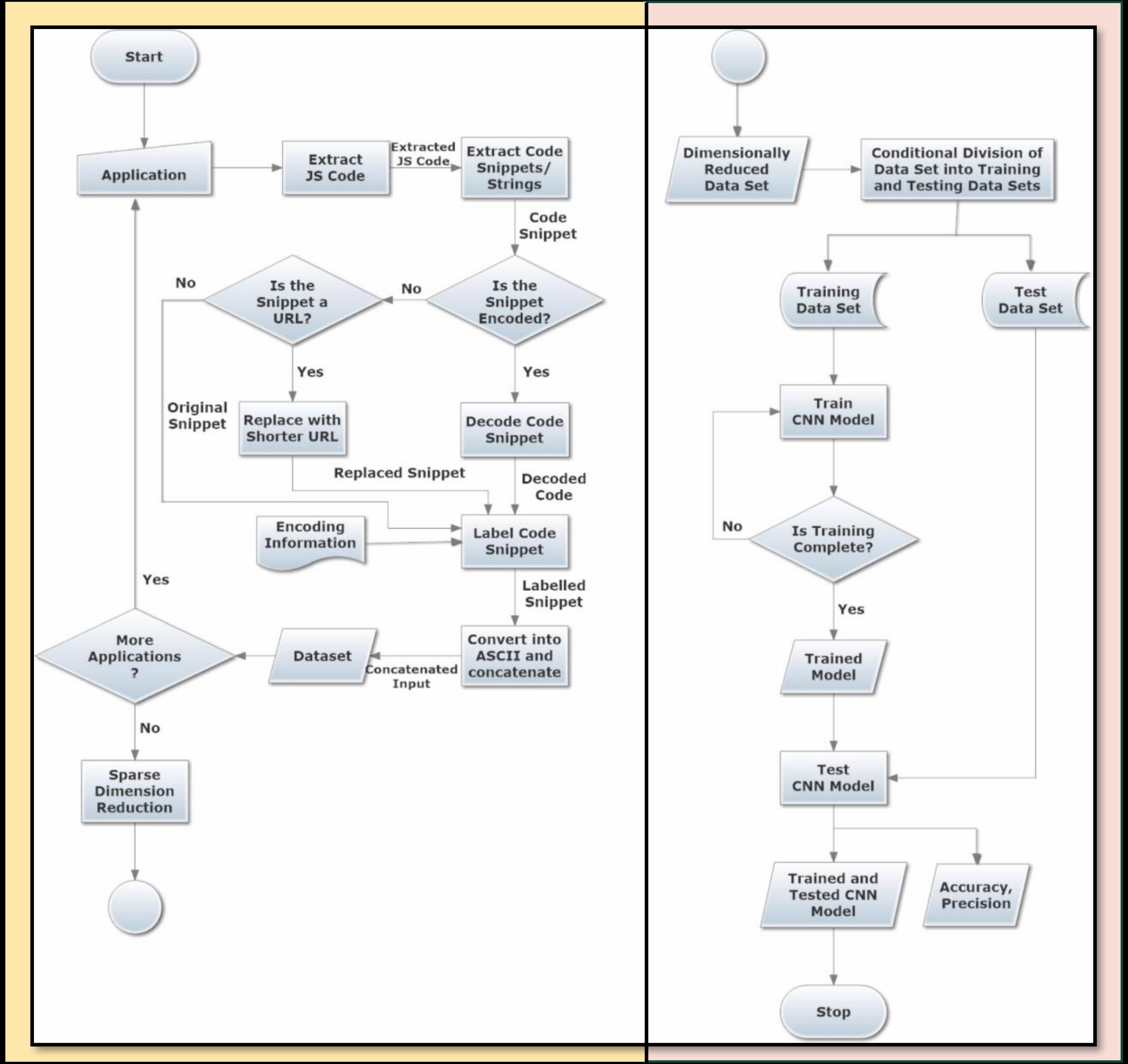
(1) CNN classifier
(2) Test CNN model
(3) find out the accuracy

3. Sanitization

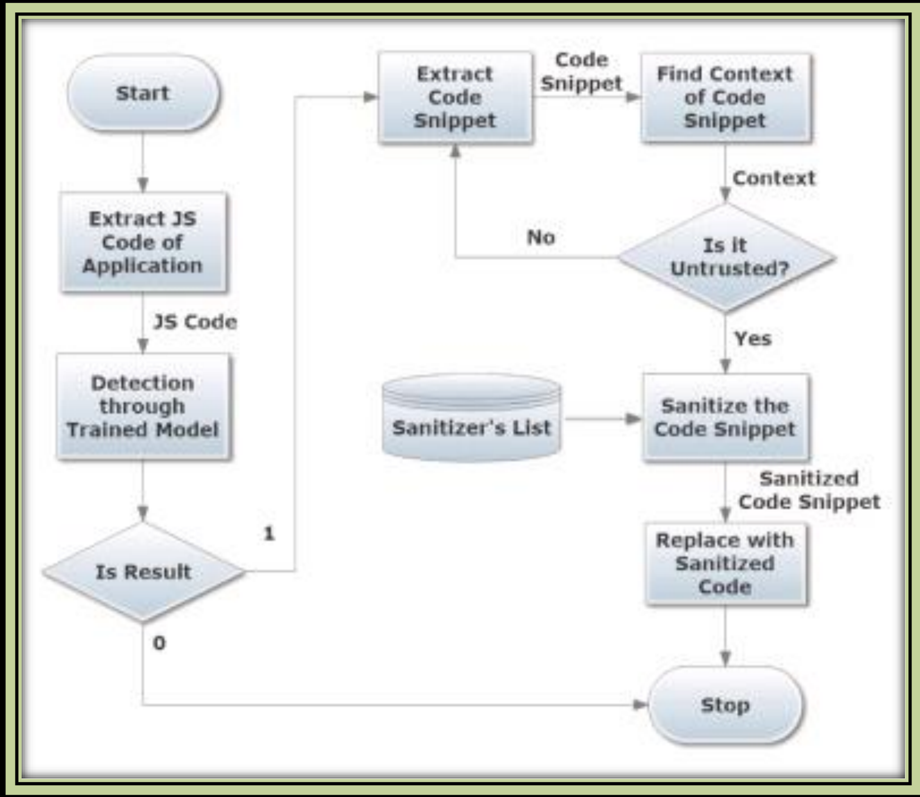
(1) extracted JS code
(2) malicious code from CNN
(3) snippet be sanitized



Framework: ConvXSS



- ∢ ∢ Data Pre-Processing
- ∢ Training and Testing
- Sanitization ∀



Data pre-processing

- ☞ Cleaning Data
- ☞ Binary vectors

- **JS Code Extractor Algo**

1. Single line: StringTokenizer(st1, ηη)

2. Divide into word: sa[i]

3. Decode

4. Http reduce

5. Labelling & Conversion

Algorithm 1: Data Pre-Processing

```
Input: Set of JS Codes (Set)
Output: Dataset of Integers (3D array of integers where each 2D array is pre-processed input.)
begin
  Dataset ← NULL; // Declaration of Dataset Variable to add the converted data in the end.
  String A ← NULL; // Declaration of a temporary string for computations.
  String [ ] sa ← NULL; // Declaration of array of strings whose contents will be tokens of code snippet.
  int k ← 0;
  for ( SεSet ) {
    // Application of the algorithm for each JavaScript Code in the set.
    String [ ][ MAX]B ← NULL;
    for ( stεS ) {
      // Application of Division of JS code into lines.
      int m ← 0;
      String st1 ← StringTokenizer(st, ); // Taking one single line of the code at a time and storing in st1.
      int i ← 0;
      while (st1.hasMoreTokens( )) do
        // Conversion of lines into array of words for further computation.
        sa[ i] ← st1.nextToken(); // Dividing the single line of code into words and storing it in the array.
        i ← i + 1;
      end while
      int j ← 0;
      Ignore the part till one of the elements in sa is ε < script > ε // Avoid Unnecessary tags for computation since th
        initial code contains noise elements which do not add any value.
      while (sa[ j]! =< /script >) do
        // Conversion of strings into required format.
        if (sa[ j].substring(0, 2) == ε&#ε) then
          A ← Decode(sa[j]); // Decoding of the Encoded Strings using Algorithm 2
        end if
        else if (sa[ j].substring(0, 7) == εhttp : //ε) then
          A ← εhttp : //websiteε; // Replaced with shorter string to reduce data.
        end if
        else
          A ← sa[ j];
        end if
        Label(A);// Semantic labelling of the string using encoding information as mentioned in Algorit
          3.
        B[ l][ m] ← Convert(A); // Conversion of the labelled string into required input using Algorithm 4.
        m + +;
      end while
      Dataset[ k] ← B[ l]; // Entry of the converted data into data set.
      Increment k and l and continue this until all the data is converted
    }
  }
  Append_Zero(Dataset); // Appending zeroes to each row of data set for consistent length.
  Sparse_Dimension_Reduction(Dataset); // Reduction technique to reduce dimensionality.
  return Dataset ;
end
```

Data pre-processing

- **Decoder**

URL Encoding

Unicode Encoding

Hex Encoding

HTML Entity Encoding

UTF-7 Encoding

```
document.write("<script onmouseover="alert(1)">test </script>");
```

HTML Entity Encoding

```
document.write(&quot;&lt;script onmouseover=&quot;alert(1)
&quot;&gt;test &lt;/script& gt;&quot;);
```

```
<article draggable="true" ondragstart="alert(1)" >test </article >
```

HTML Decimal Coding

```
<article draggable="true" ondragstart="&#97;&#108;&#101;
&#114;&#116;&#40;&#49;&#41;">test </article >
```

HTML Hexadecimal Coding

```
<article draggable="true" ondragstart="&#x61;&#x6c;&#x65;
&#x72;&#x74;&#x28;&#x31;&#x29;&#xa;">test </article >
```

Data pre-processing

- **Decoder**

S = string
ch = char
Num = 臨時字串
Number[ ]: encode value

> Num.append(ch);

將ch 加到String Num尾端

> char c <- n;

數字 n 轉換為字元 c

> a.append(c);

將c 加到String a 尾端

Algorithm 2: Decoding of Data

```
Input: String (The String to be Decoded)
Output: Decoded String after the computation of the algorithm
begin
    String a, Num ← NULL; // Num is a temporary string used to append numbers and later on convert to integer.
    int[ ]Number ← 0; // Declaration of integer array to store converted numbers
    // This method is used as Encoded strings are formed by appending '&' and '#' with ASCII values of each
       character and ';'
    for ( ch∈S ) {
        if (ch == &||ch == #) then
            // Taking each character and ignore if character is & or #
            Ignore ch;
        end if
        else if (ch>=0&&ch<=9) then
            // If the character is a number, then appending it with the existing string of numbers.
            Num.append(ch);
        end if
        else if (ch==;) then
            // If the character reaches ;, the string is converted to number and put into the array Num.
            Ignore ch;
            Number ← (int)Num; // converting the string to integer and adding it to the array.
            Num ← NULL;
        end if
    }
    for ( n∈Number ) {
        // Each number is taken and converted to its corresponding character using ASCIIand appended to form
           a string.
        char c ← n;
        a.append(c);
    }
    return a; // Output is the decoded string.
end
```

Data pre-processing

- **Generalization**

  the string of websites >> http://website

- **Semantic labeling of strings**

  - Input Data:

    String S

    Encoding Information

  - EI[].contains(S)

  - A.append(S)

Algorithm 3: Labelling of Data

```
Input: String S, Encoding Information EI
begin
    String A ← NULL; // Declaration of a String for Modification
    if (EI[0].contains(S)) then
        // Labelling based on the context of the given string.
        A ← 0;
    end if
    else if (EI[1].contains(S)) then
        A ← 1;
    end if
    else if (EI[2].contains(S)) then
        A ← 2; // Concatenation of a label based on the context.
    end if
    A.append(S); // Appending the existing string with the label.
    S ← A;
end
```

Data pre-processing

- **Conversion of strings into the required input**

  UPPERcase letter >> lower case

  zeros appended to maximum length

- **Dimensionality reduction**

  ☞ Decrease time of the training

  ~~Factor Analysis~~, ~~ICA~~, ~~PCA~~

  “Sparse Random Projection (SRP)”

  - 使用**稀疏隨機矩陣**將數據從高維空間投影到低維空間。
  - 計算效率：非常高，由於投影矩陣是稀疏的，計算量小。
  - 保持距離：根據 Johnson-Lindenstrauss 引理，能夠大概率地保持數據點之間的距離。
  - 隨機性：結果具有隨機性，不同次的投影結果可能不同。
  - 適用大規模數據集的快速處理和預處理。

Training and Testing

- **Training and testing of CNN model for classification**

1. training and testing data

2. Convolutional Neural Network
   2.1 single or multiple layer of convolutional
   2.2 pooling layer
   2.3 single or multiple fully connected layers

3. The reason to choose CNN:

“Performance of CNN for image recognition is better than the other classifiers”

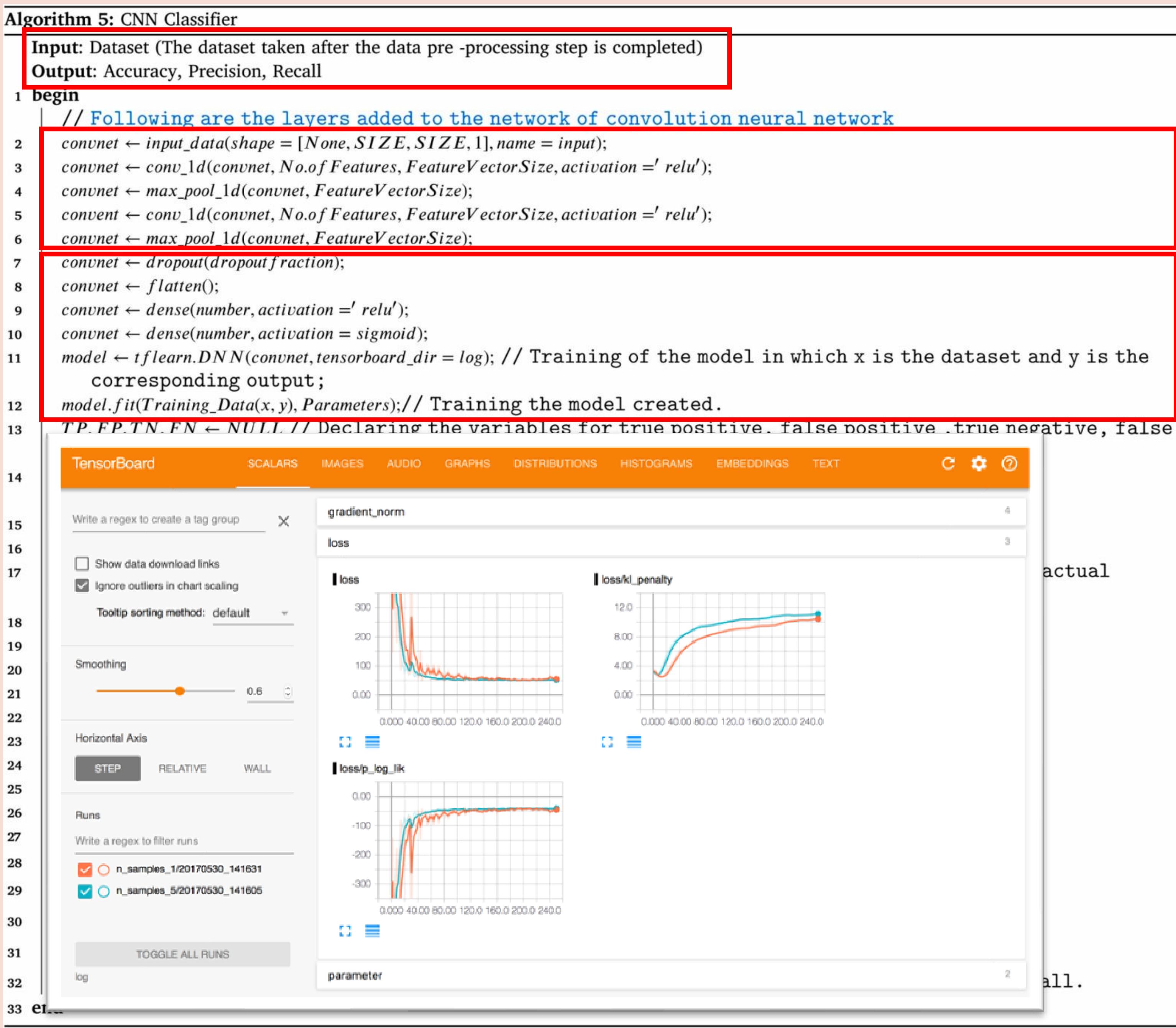
Training and Testing

- **Training and testing of CNN ConvXSS Algo.**

Input = Dataset
Output = Accuracy, Precision, Recall

- Input Data: $[None, SIZE, SIZE, 1]$
- conv_1d:
  No.ofFeatures, FeatureVectorSize
  ReLU
- max_pool_1d: FeatureVectorSize
- dropout() & flatten()
- dense ('relu' & sigmoid)

- tf learn.DNN: TensorBoard
- model.fit:



Algorithm 5: CNN Classifier

Input: Dataset (The dataset taken after the data pre -processing step is completed)
Output: Accuracy, Precision, Recall

```
begin
  // Following are the layers added to the network of convolution neural network
  convnet ← input_data(shape = [None, SIZE, SIZE, 1], name = input);
  convnet ← conv_1d(convnet, No.ofFeatures, FeatureVectorSize, activation =' relu');
  convnet ← max_pool_1d(convnet, FeatureVectorSize);
  convent ← conv_1d(convnet, No.ofFeatures, FeatureVectorSize, activation =' relu');
  convnet ← max_pool_1d(convnet, FeatureVectorSize);
  convnet ← dropout(dropoutfraction);
  convnet ← flatten();
  convnet ← dense(number, activation =' relu');
  convnet ← dense(number, activation = sigmoid);
  model ← tflearn.DNN(convnet, tensorboard_dir = log); // Training of the model in which x is the dataset and y is the
     corresponding output;
  model.fit(Training_Data(x, y), Parameters);// Training the model created.
  TP, FP, TN, FN ← NULL // Declaring the variables for true positive, false positive ,true negative, false
  ... actual
  ... all.
```

Training and Testing

- **Training and testing of CNN ConvXSS Algo.**

  - x,y ϵ Test_Data: (x,y)

    x: 預設值

    y: 實際值

  - Accuracy
  - Precision
  - Recall

Algorithm 5: CNN Classifier

```
Input: Dataset (The dataset taken after the data pre -processing step is completed)
Output: Accuracy, Precision, Recall
begin
    // Following are the layers added to the network of convolution neural network
    convnet ← input_data(shape = [None, SIZE, SIZE, 1], name = input);
    convnet ← conv_1d(convnet, No.of Features, FeatureVectorSize, activation =' relu');
    convnet ← max_pool_1d(convnet, FeatureVectorSize);
    convent ← conv_1d(convnet, No.of Features, FeatureVectorSize, activation =' relu');
    convnet ← max_pool_1d(convnet, FeatureVectorSize);
    convnet ← dropout(dropoutfraction);
    convnet ← flatten();
    convnet ← dense(number, activation =' relu');
    convnet ← dense(number, activation = sigmoid);
    model ← tflearn.DNN(convnet, tensorboard_dir = log); // Training of the model in which x is the dataset and y is the
        corresponding output;
    model.fit(Training_Data(x, y), Parameters);// Training the model created.
    TP, FP, TN, FN ← NULL // Declaring the variables for true positive, false positive ,true negative, false
        negative
    for ( x, yϵTest_Data ) {
        // Testing the trained model for every value.
        output ← model.predict(x);
        if (output ==1 && y==1) then
            TP++;// Incrementing the respective values based on the output given by model and the actual
                output.
        end if
        else if (output == 0 && y==1) then
            FN++;
        end if
        else if (output == 1 && y==0) then
            FP++;
        end if
        else if (output == 0 && y==0) then
            TN++;
        end if
    }
    Accuracy ← ((TP+TN)/(TP+TN+FP+FN)) * 100; // Accuracy Calculation based on the values of TN, TP, FP, FN
    Precision ← (TP/(TP+FN)) * 100; // Precision Calculation based on the values TP, FN.
    Recall ← (TP/(TP+FP)) * 100; // Calculation of Recall based on the values TP,FP.
    return Accuracy, Precision, Recall; // Output values of this algorithm is Accuracy,Precision and Recall.
end
```

	Actual malicious code	Actual benign code
Predicted malicious code	True Positive (TP)	False Positive (FP)
Predicted benign code	False Negative (FN)	True Negative (TN)

Detection and Sanitization

- **Context based sanitization**

If (the code in the app is detected malicious) {

// list of sanitizers and encoding information

//the code for which the classifier detected as malicious

Algorithm 6: Context Based Sanitization

```
Input: JavaScript Code, List of Sanitizers, Encoding Information
Output: JS Document (After the sanitization of the untrusted variables)
begin
    String[ ]U ← NULL // Declaration of a string for purpose of modification.
    for ( linesl∈JS ) {
        // Taking each line of the code into consideration
        for ( S∈sl ) {
            // Considering each word in the line taken
            int l ← label_integer(S) // Taking the labelled integer of the word into account
            if (l==untrusted_num) then
                U.push(S);// If the context of the string is untrusted, the word is listed
                Replace(Sanitize(S),S); // Respective sanitizer is applied based on context and replaced.
            end if
        }
    }
    return JS // After the completion of sanitization of the variables and replacement, the code is returned.
end
```

- each line of JS code
- each word in line
- (label) int l
- U.push
- Replace(Sanitize(S),S)

}

04. Implementation

Environment

Dataset: **105,470 samples**

31,407 benign

74,063 malicious

XSS Payload Dataset (Payloadbox, 2020),
Cross-site scripting dataset (Shah, 2020),
XSS Filter Evasion Cheat Sheet (OWASP, 2017)
others(Cozamanis, 2019), Balaji (2019),
Cross-site scripting (XSS) cheat sheet (2021),
HTML5 security cheatsheet (2021).

Windows 10
Intel(R) Core i7 CPU @ 1.8 GHz, 16 GB RAM

Kaggle

Colab

Python Language

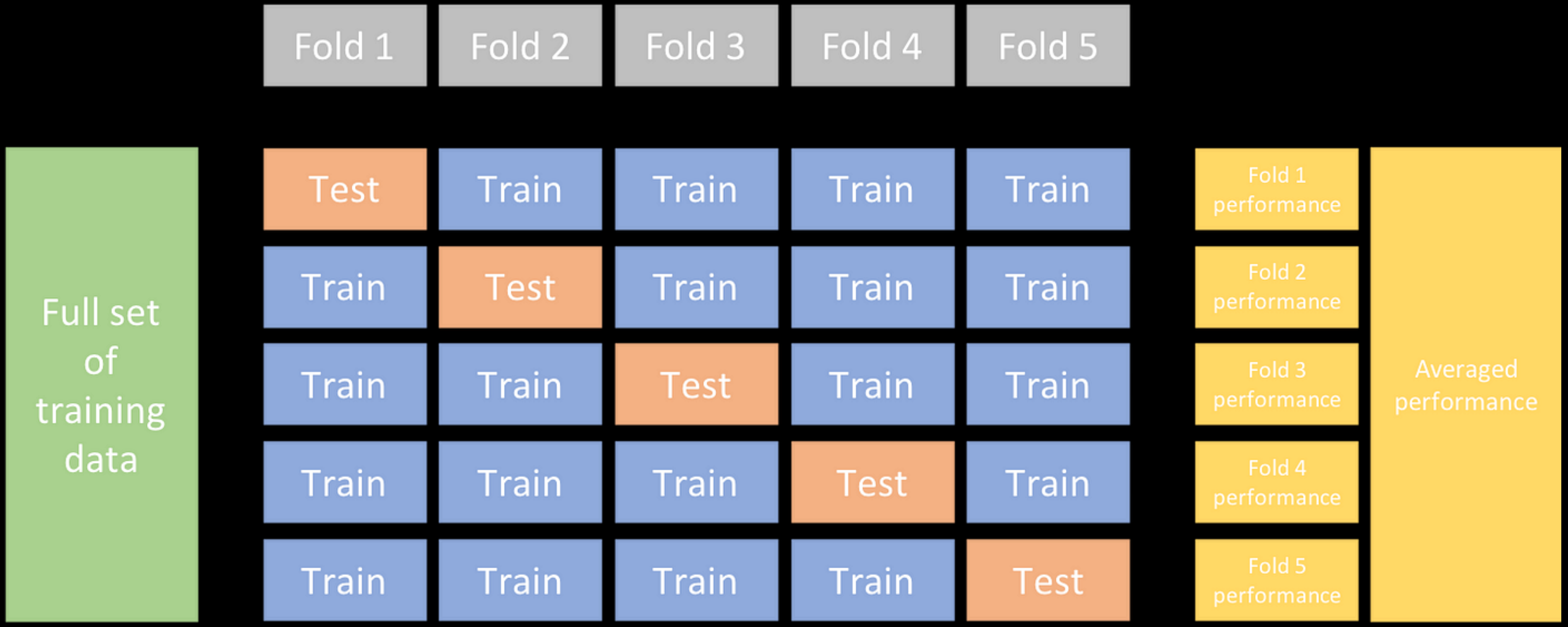
TensorFlow, Keras

ConvXSS

Dividing dataset

- **K-fold cross-validation (K=10)**

K-Fold 方法將訓練資料再依序切割訓練集與測試集
K-Fold 裡面的測試集可以當成驗證集。K-Fold 的方法中 K 是由我們自由調控的，在每次的迭代中會選擇一組作為驗證集，其餘 (k-1) 組作為訓練集。透過這種方式學習，不同分組訓練的結果進行平均來減少方差，因此模型的性能對數據的劃分就不會那麼敏感。



Dividing dataset

- **K-fold cross-validation (K=10)**

Epoch value: 20

Hidden Layer: 8-10

1d convolutional layer & Max-pooling

CNN Model 1			
Evaluation	Accuracy	Precision	Recall
Fold 1	97.7866352	97.7655231	98.0902314
Fold 2	98.3178437	98.3915687	98.4461665
Fold 3	98.8784015	98.8002300	99.0766644
Fold 4	98.6865461	98.5807896	98.9860237
Fold 5	99.3358970	99.4206965	99.3384838
Fold 6	99.2178321	99.2961645	99.2692828
Fold 7	99.2916226	99.2161274	99.4388402
Fold 8	99.2621064	99.5977521	99.0664184
Fold 9	99.3801713	99.5587468	99.2849290
Fold 10	98.9817083	98.5597790	99.5608091
Average	98.9138764	98.9187378	99.0557849

CNN Model 4			
Evaluation	Accuracy	Precision	Recall
Fold 1	98.1260180	97.5709617	98.9482403
Fold 2	98.6719787	98.2957661	99.2230773
Fold 3	99.0112245	98.7218678	99.4124234
Fold 4	99.1292894	99.1511524	99.2326677
Fold 5	99.4539618	99.2863119	99.6968031
Fold 6	99.4096875	99.4052529	99.5128572
Fold 7	99.2325902	99.0502775	99.4949520
Fold 8	99.4982362	99.6524990	99.4398534
Fold 9	99.4687200	99.5321989	99.4774461
Fold 10	99.4982362	99.4248211	99.6431589
Average	99.14999425	99.0091109	99.4081479

Evaluative measures for models with different hidden layers and number of neurons.

Model no.	Number of hidden layers	Number of neurons[a]	Overall accuracy	Precision	Recall
1	8	100c,0.2dp,100c,fl,250d,1d	98.9138764	98.9187378	99.0557849
2	8	200c,0.2dp,100c,fl,250d,1d	98.8976467	98.8437295	99.102354
3	8	300c,0.2dp,100c,fl,250d,1d	98.6497152	98.5406047	98.9487636
4	10	100c,100c,0.2dp,100c,fl,250d,1d	99.1499942	99.0091109	99.4081479
5	10	200c,100c,0.2dp,100c,fl,250d,1d	99.0791547	99.0867633	99.1955662
6	10	300c,100c,0.2dp,100c,fl,250d,1d	99.0319312	99.0377957	99.1575849

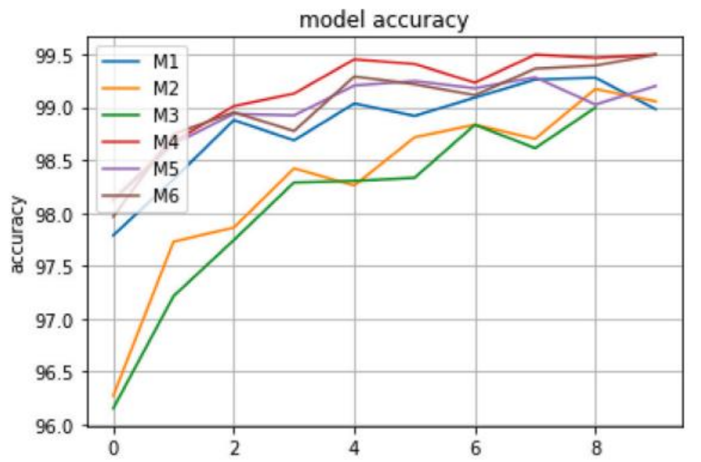
[a]c: Conv1D + MaxPool1D, dp: Dropout, fl: Flatten, d: Dense.

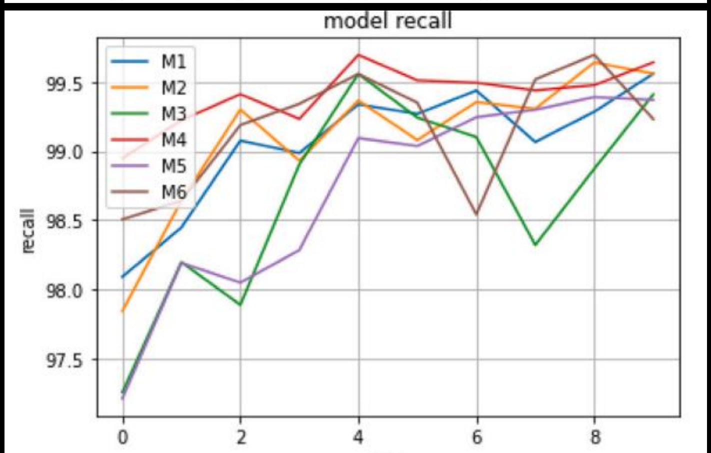
Performance analysis

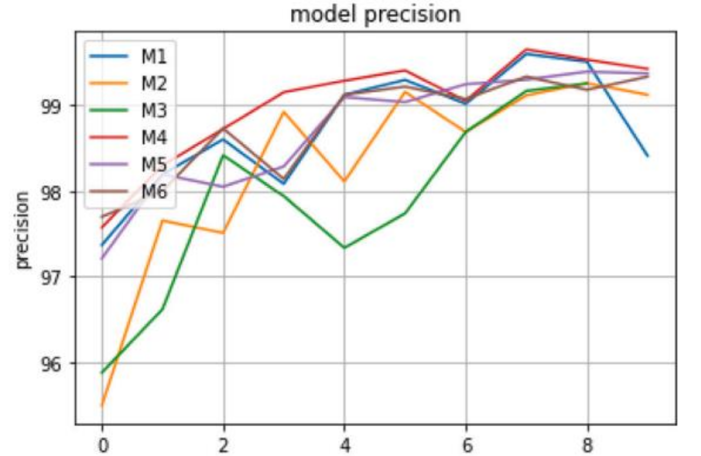
- **Influence the process of optimization**
  - Count of neurons in hidden layers
  - Training cycles
  - Quantity of hidden layers
  - Types of the optimizers

Comparison with Models of other papers.

Model	Accuracy overall	Precision	Recall
Our Model - ConvXSS	99.42	99.81	99.35
CODDLE - S. Abaimov and G. Bianchi (Abaimov & Bianchi, 2019)	95.7	99.0	91.2
Selvam, M. K. Selvam (2018)	98.54	98.65	98.40
Wang, Y., Cai, W. D., Wei, P. C. Wang et al. (2016)	94.82	94.9	94.8
DeepXSS - Fang, Y., Li, Y., Liu, L., Huang, C. Fang, Li, Liu, and Huang (2018)	98.5	99.5	97.9
Adaboost - Wang, R., Jia, X., Li, Q., Zhang, S. Wang, Jia, Li, and Zhang (2014)	94.1	93.9	93.9



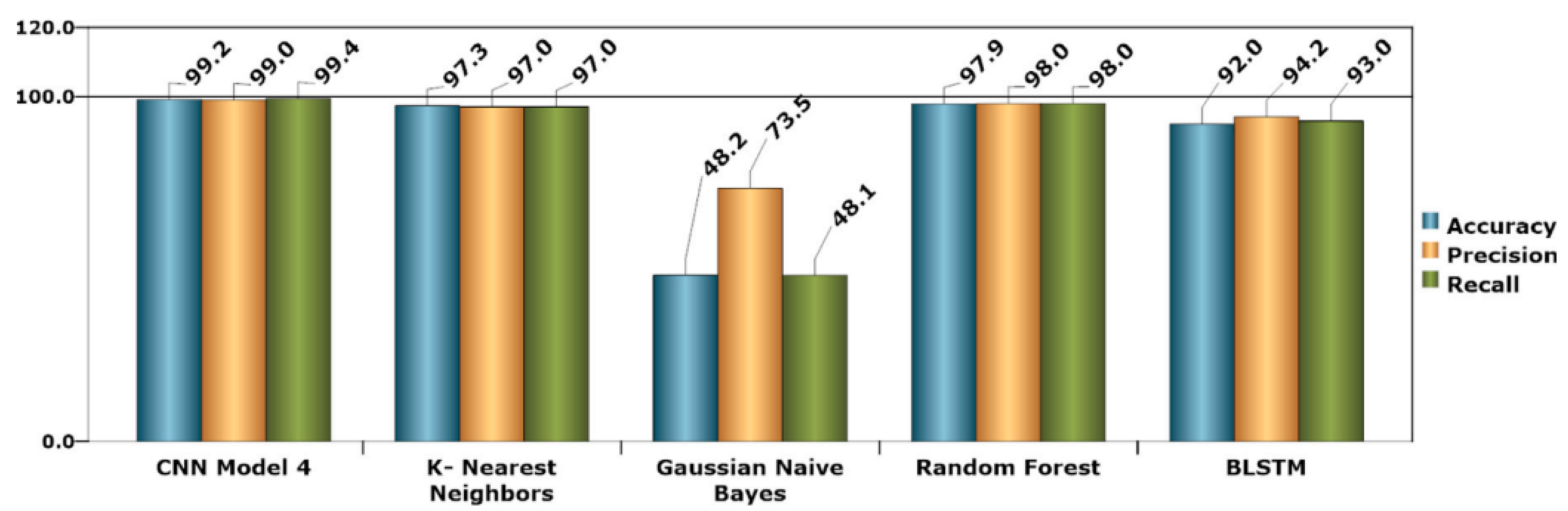




Performance analysis

- **Comparative study with other models**

  - K-nearest neighbors (KNN)
  - Naive Bayes
  - Random Forest (RF)
  - Bidirectional Long Short-Term Memory (BLSTM)



Limitations

☑ Detect the malicious code
☑ Sanitize the code
☒ Encoding Information limited (different languages)
☒ Number of sanitizers
☒ More efficient algorithms

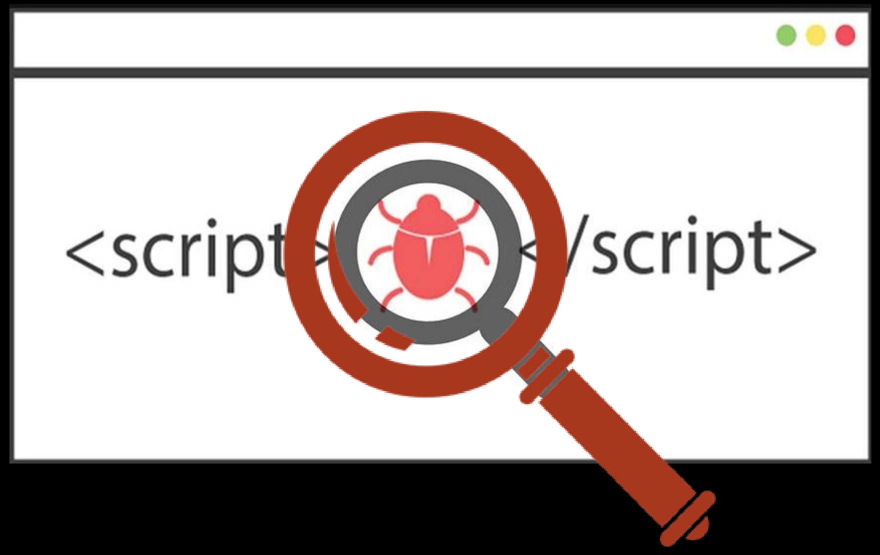
05. Conclusions

Contributions

1. Definition of security and privacy in the sustainable smart cities
2. Identified channels through the code be injected and spread
3. ConvXSS: Convolutional Neural Networks
4. Decoding, Generalizing, and Labeling, binary vectors
5. Accuracy: 99.42% 、 Precision: 99.81%

Prospective possibilities

1. Real life ex. insufficient data, noise in data. (ML algorithms)
2. Real-time and drawing out the JS code in Website
3. OS can immediately block/delete/sanitize the compromised website/app.
4. Add new malicious JavaScript samples. end-to-end Deep Learning



Q & A