Securing heterogeneous embedded devices against XSS attack in intelligent IoT system

《Computers & Security》

Pooja Chaudhary
Department of Computer Engineering, National Institute of Technology Kurukshetra, India

Brij Bhooshan Gupta
Department of Computer Science and Information Engineering, Asia University, Taichung 413, Taiwan

Awadhesh Kumar Singh
Department of Computer Engineering, National Institute of Technology Kurukshetra, India

2023.02.29

張家維

CONTENT

01.

Introduction

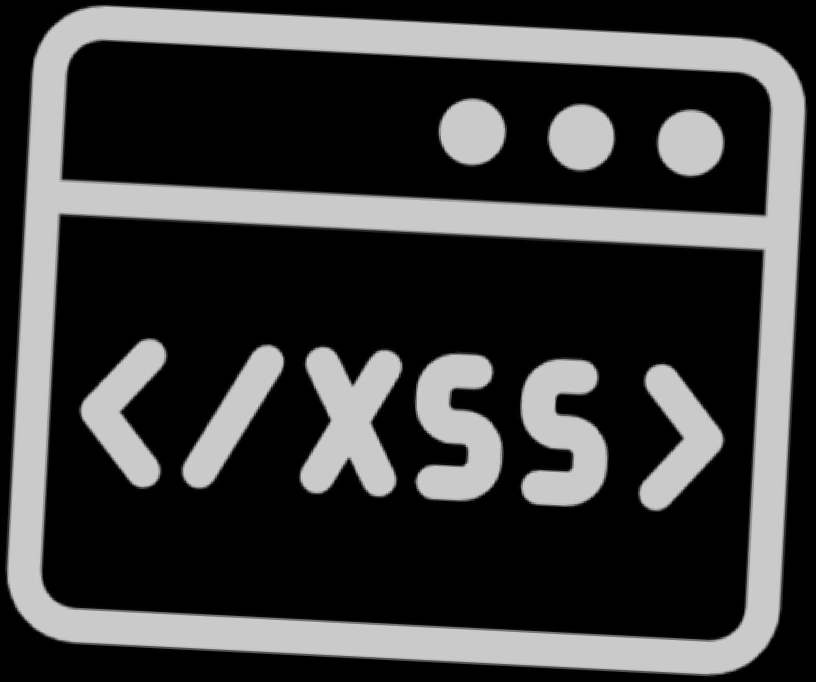
XSS attack

"Why Cross-Site Scripting ?

Cross-Site Scripting (XSS) is one such
~~commonly found and threatened web application vulnerability~~.

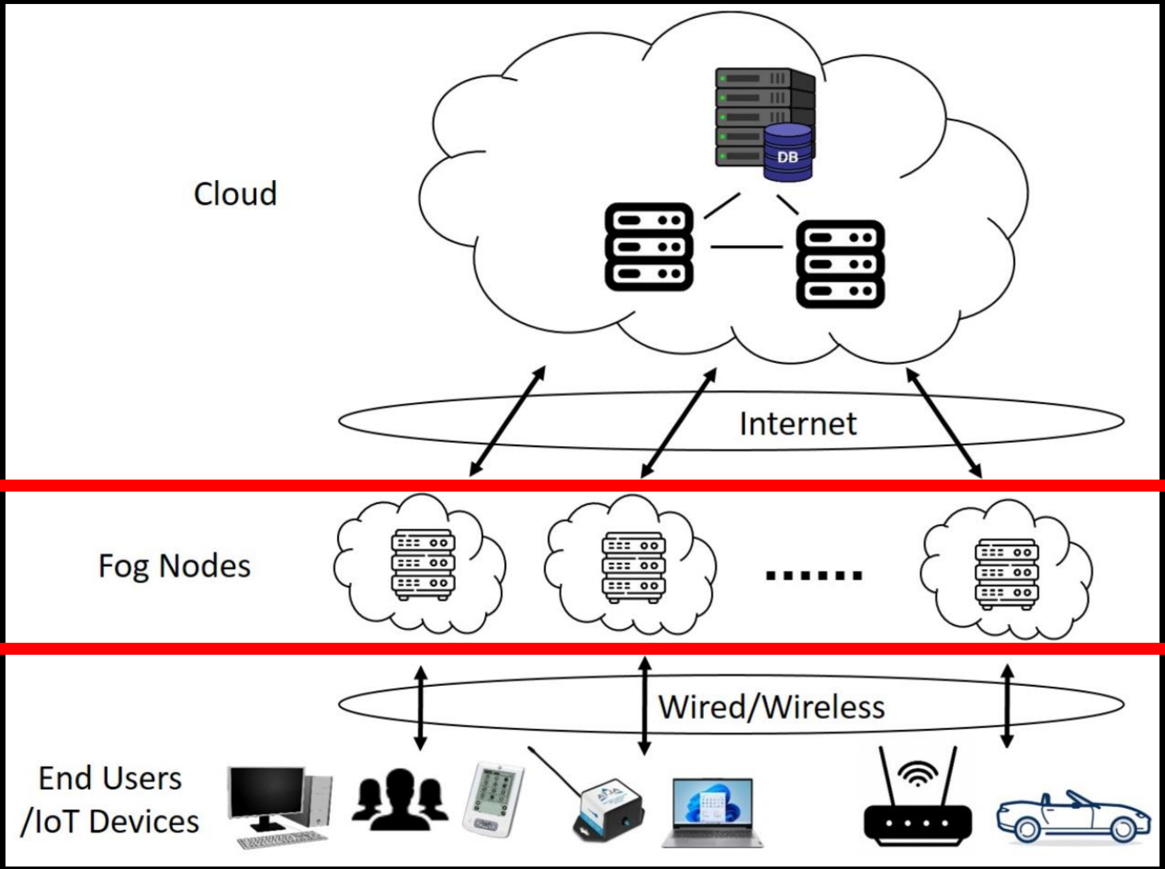
Massive scale:

- Unauthorized access to sensitive data.
- Redirecting the user to attacker control web site.
- Device exploitation in building botnet army for Distributed Denial of Service (DDoS)
- Reconfiguration of devices settings.



Fog-based IOT

IoT system



Fog Computing

- A distributed computing model that places computing resources and services between IoT devices and the cloud
- This model positions computing resources in proximity to the point of data generation, aiming to reduce latency in the data transmission process.
- Enables real-time processing and analysis of data at the device edge

Difference Between Edge Computing and Fog Computing

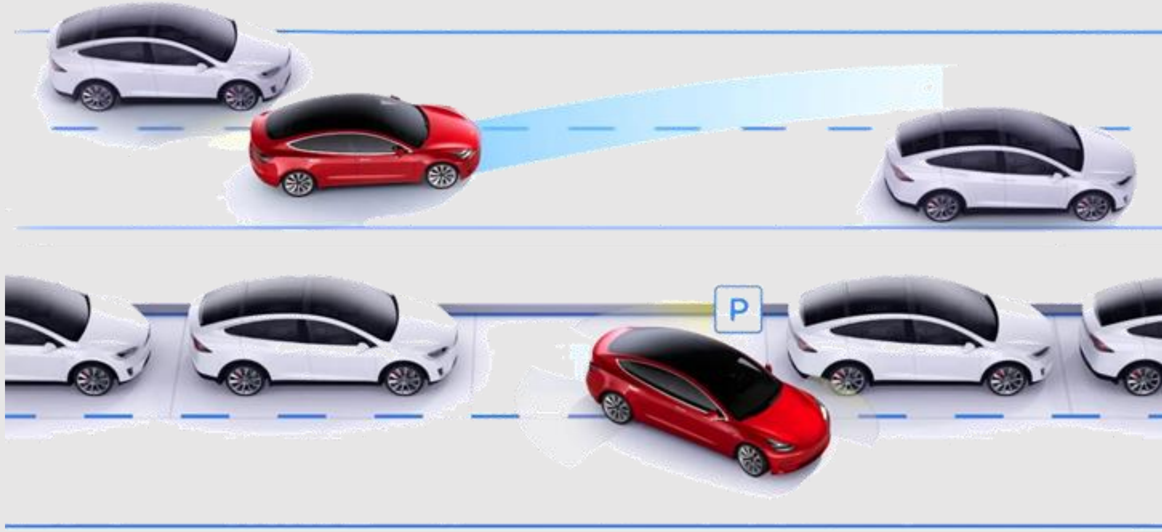
Fog Computing

•在資料來源和中央雲端平台之間放置了一個分散的企業運算層。與邊緣運算一樣，霧運算也能使處理能力更接近資料擷取的位置。

•雲端運算的延伸。當邊緣電腦向雲端發送大量資料時，霧節點接收資料並分析重要內容。然後霧節點將重要資料傳輸到雲端進行存儲，並刪除不重要的資料或保留它們以供進一步分析。

Edge Computing

•邊緣運算讓處理和儲存系統盡可能靠近產生和收集資料的應用程式、設備或元件。透過消除將資料傳輸到中央處理系統並返回端點的需要，有助於最大限度地減少處理時間

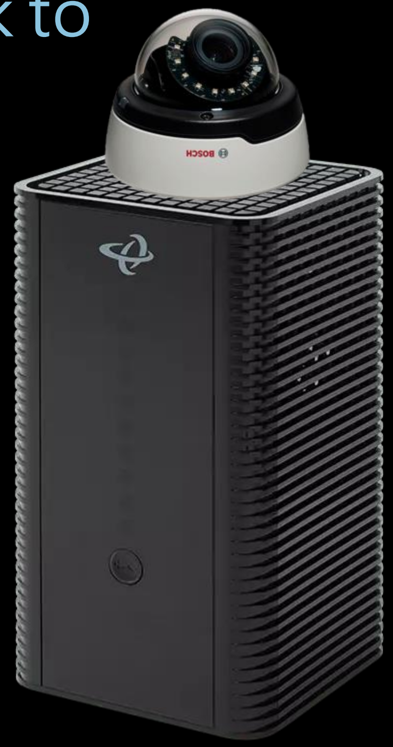


Reference: https://www.spiceworks.com/tech/cloud/articles/edge-vs-fog-computing/

About Research

1. Purpose

The development of an approach to defend against XSS attack to safeguard embedded devices deployed in intelligent IoT system.



2. Method

Fog-enabled approach

- Comparing injected strings with the blacklisted
- Implementing filtering method

3. Demo environment

Digital IP Camera and wireless router

Hitron CODA 4582u router & Bosch Flexidome IP indoor 5000 HD camera.

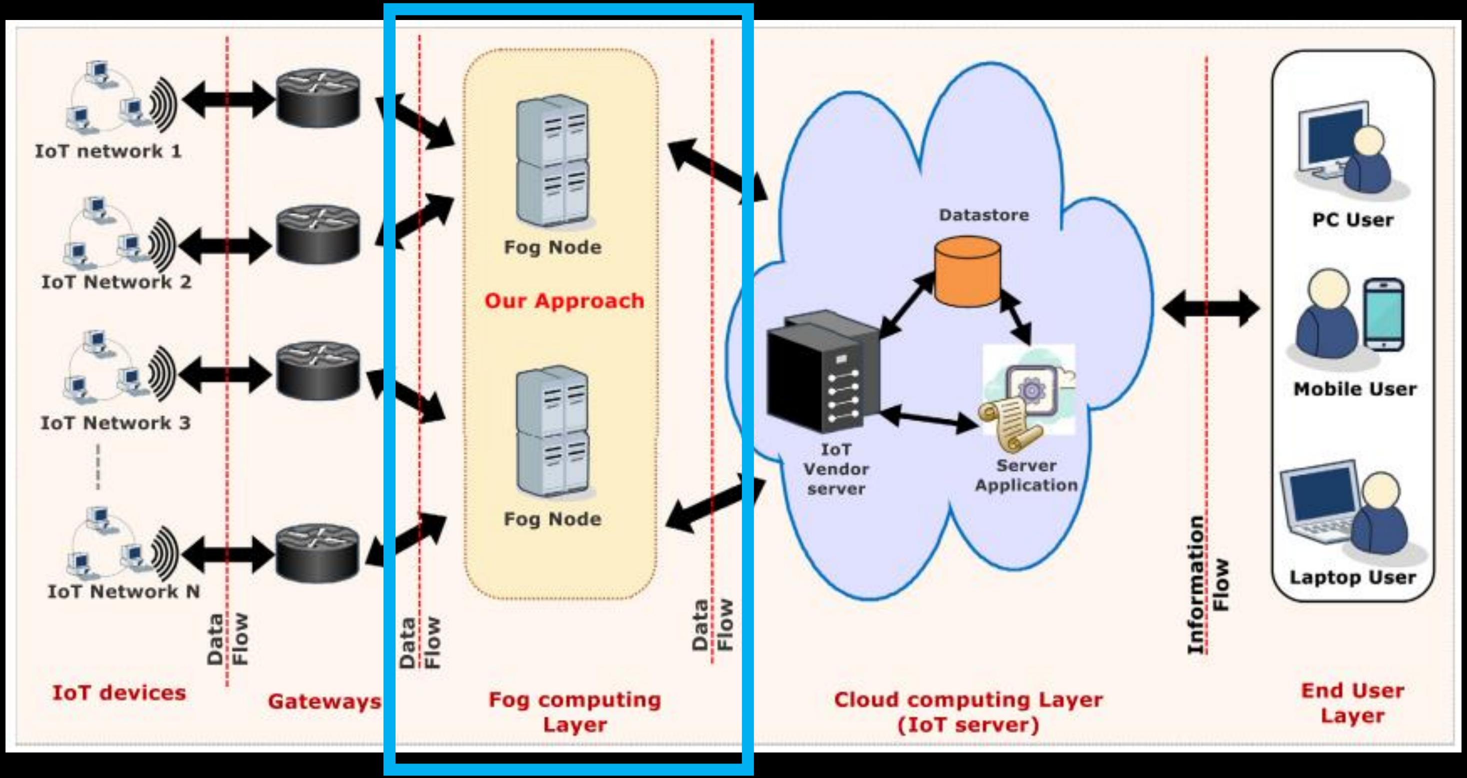
4. Experimental results

Accuracy over 0.9

02. Proposed approach

Conceptual Design

Abstract Design Overview of the proposed approach.



Objective 1:

Identify reflected XSS attack.
(parameter & class of attack string)
yi ∈ Q and yi ⊆ AS

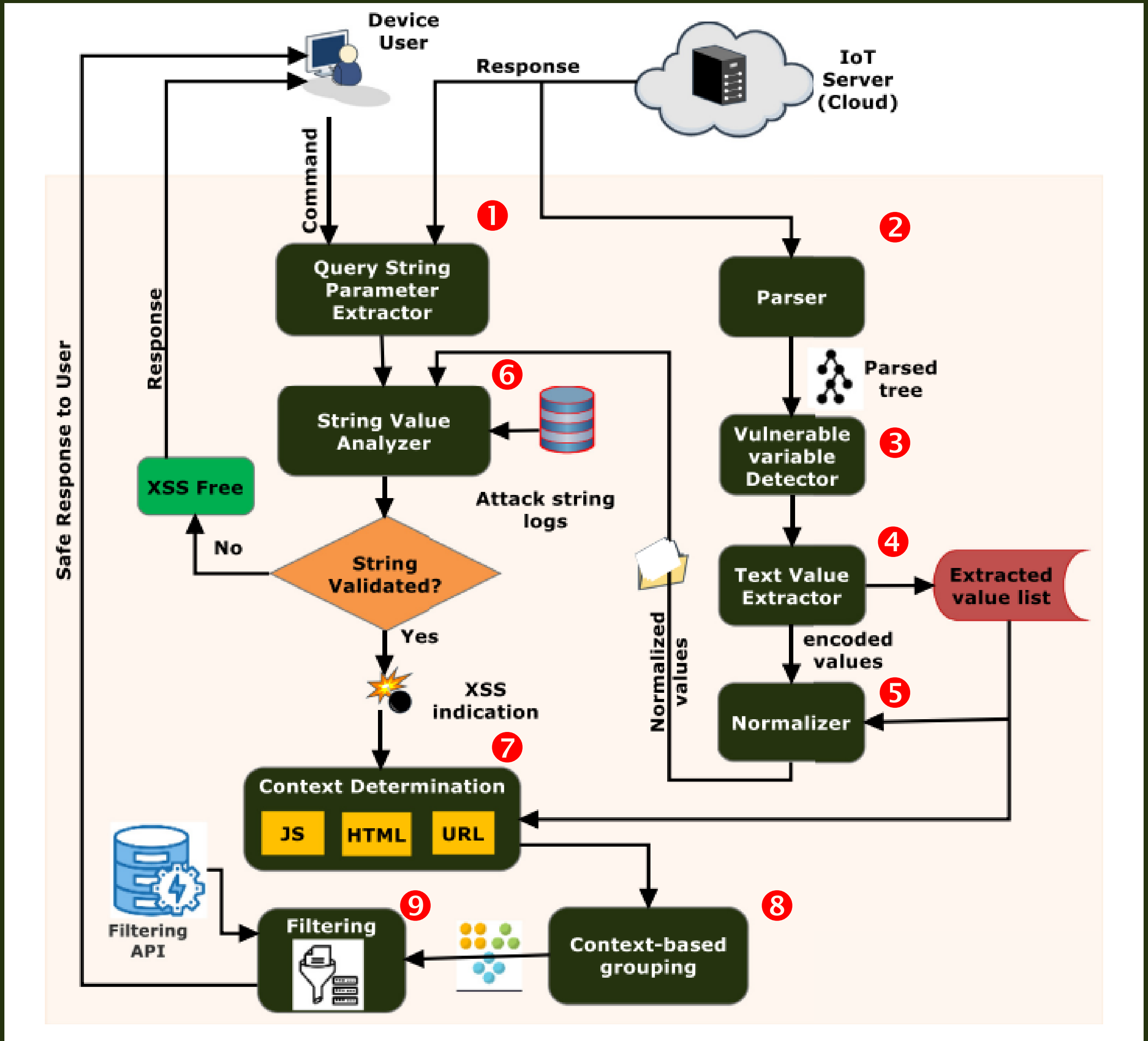
Objective 2:

Identify stored XSS attack.
(injected input , HTTP response & class of attack string)
(wi ∈ I and xi ∈ L: wi present at xi) and (wi ⊆ AS)

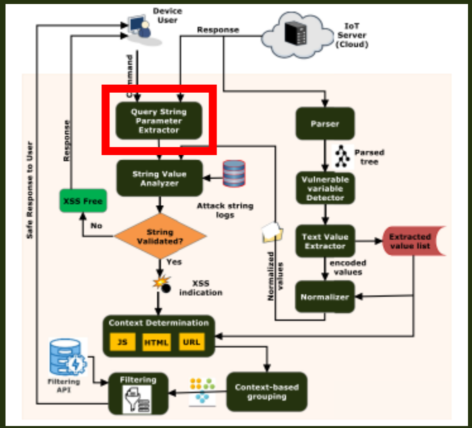
Objective 3:

Nullify the effects of XSS attack strings.
Grouping (JS-based, HTML- based and their content)

Detailed Design



Detailed Design |



Query string parameter extractor:

- Seizes each user's command i.e., HTTP request
- Correspondingly generated server's response i.e., HTTP response.
- Extracts the parameter values (query string)
- HTML decoding, URL decoding

Request

Raw Hex GraphQL

```
GET /level1/frame?query=%3Cscript%3Ealert%28%29%3B%3C%2Fscript%3E HTTP/2
Host: xss-game.appspot.com
Sec-Ch-Ua: "Chromium";v="121", "Not A(Brand";v="99"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/121.0.6167.85 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/
apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: iframe
Referer: https://xss-game.appspot.com/level1/frame
Accept-Encoding: gzip, deflate, br
```

Response

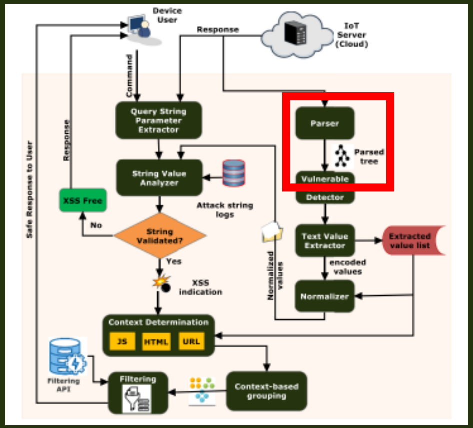
Pretty Raw Hex Render

```
HTTP/2 200 OK
Content-Type: text/html; charset=utf-8
Cache-Control: no-cache
X-Xss-Protection: 0
X-Cloud-Trace-Context: 4cbe9fa9e6c0b1d84ec15f36cf23baf8;o=1
Vary: Accept-Encoding
Date: Fri, 23 Feb 2024 07:06:31 GMT
Server: Google Frontend
Content-Length: 421
Alt-Svc: h3=":443"; ma=2592000,h3-29=":443"; ma=2592000


<!doctype html>
<html>
  <head>
    <!-- Internal game scripts/styles, mostly boring stuff -->
    <script src="/static/game-frame.js">
```

%3Cscript%3Ealert%28%29%3C%2Fscript%3E. may be decoded as
<script>alert()</script>

Detailed Design |



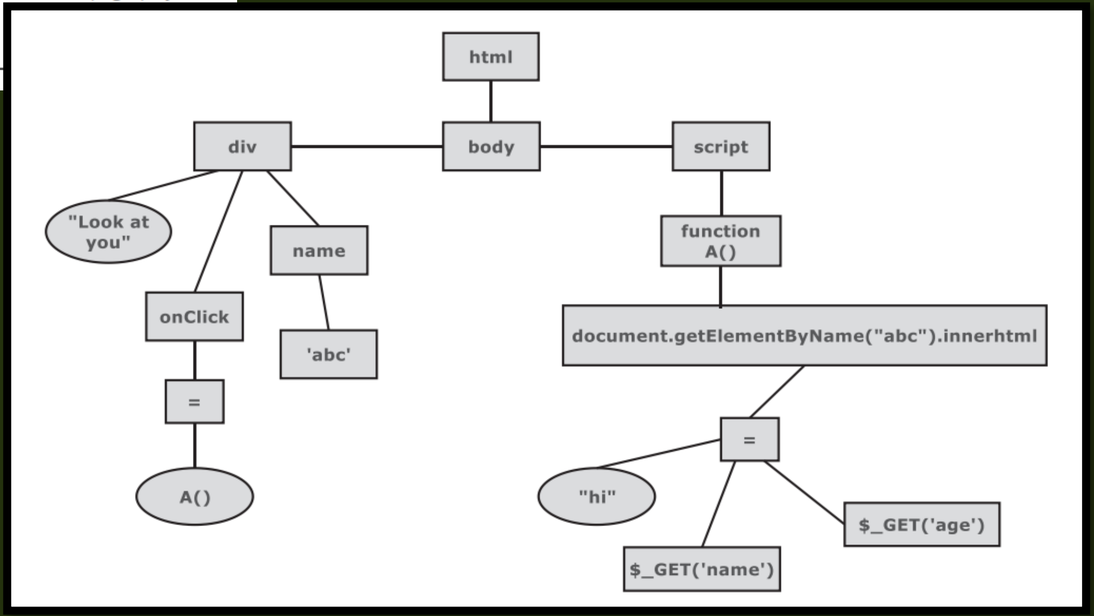
Parser:

- constructs parse tree of the received web page
- html5lib parser

Listing 1
Dummy Code snippet vulnerable to XSS attack.

```
<html>
<body>
<div name= "abc" onClick= "A()"> look at you!!! </div>
<script>
function A() {
document.getElementByName("abc").innerhtml= "Hi" + "$_GET('name')" + "$_GET('age')";}
</script>
</body></html>
```

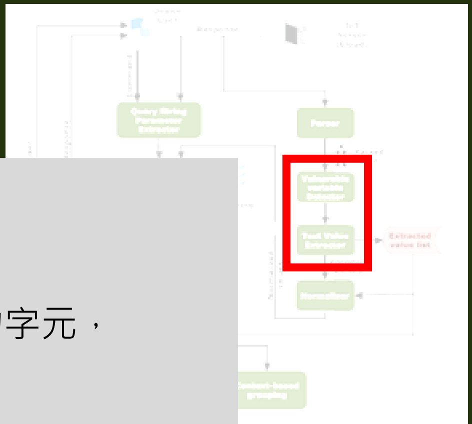
html5lib parser



Detailed Design

Vulnerable variable detector & text value extractor:

- Identifies the vulnerable locations (1/2)



Each known vulnerable context of the HTML page

List of malicious contexts in HTML page.

Elements	Context
	PCDATA
HTML	RCDATA
	CDATA
	Tag name
	Attribute name
	Attribute value: Quoted
	Attribute value: Unquoted
	Event attribute
	Tag text: String
	Attribute value: String
JavaScript	Method name
	Method value: REGEX
URL	Query: String

- **PCDATA（Parsed Character Data）:** 可解析字元
  <p>Hello</p>

- **RCDATA （Replaceable Character Data）:**可替換的字元，
  < 代表 <

- **CDATA（Character Data）：**
  CDATA 是字符數據，通常用於標記內容，例如 <script> 或 <style> 標記中的 JavaScript 或 CSS 代碼。

- **Tag name**：HTML元素的名稱，例如 <p> 標記中的 "p"。

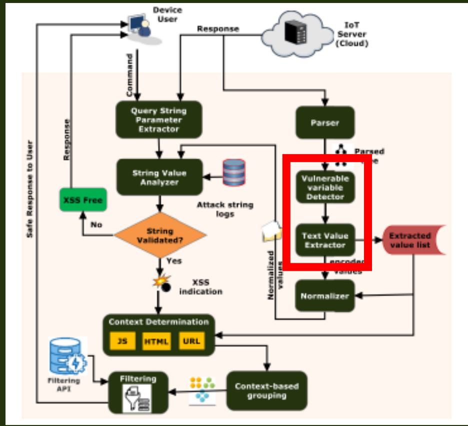
- **Attribute name**：HTML元素的屬性的名稱
  <a href="https://example.com"> 中的 "href"。

- **Attribute value**：HTML元素的屬性的值，
  <img src="image.jpg"> 中的 "image.jpg"。

- **Event attribute**：HTML元素的屬性，應執行的JavaScript代碼
  <button onclick="myFunction()"> 中的 "onclick"。

- **Tag text: String**：
  標記文本是HTML元素內的純文本內容，不包含HTML標記。
  例如，對於 <p>This is a paragraph.</p>，"This is a paragraph." 就是標記文本。

Detailed Design



Vulnerable variable detector & text value extractor:

- Identifies the vulnerable locations (2/2)

H{}　（HTML 標記、屬性）

JS{}　（JavaScript 字串）

U{}　（URL 屬性）

X_{PV}{}（漏洞字串）

Vulnerable variable detection and value extraction.

Input: parsed tree of response web page P(N, E)
Output: Vulnerable string payload vector (X_{PV})
H_tag ← HTML vulnerable tags list
H_att ← HTML vulnerable attribute list
H_event ← HTML vulnerable event handler list
JS_fun ← JS vulnerable function list
JS_prop ← JS vulnerable properties list
URL_prop ← URL vulnerable properties list
Start
H{}← ∅;
JS{}← ∅;
U{}← ∅;
X_{PV}{}← ∅;
//Extract every tag, attribute and event handler from parse tree
For each node $n_i \in$ P(N, E) **do**
If (n_i.matches(H_att)) **then**
H ← H ∪ n_i.value;
elseif (ni.matches(H_event)) **then**
H ← H ∪ n_i.value;
elseif (ni.matches(H_tag)) **then**
H ← H ∪ n_i.value;
//collect JS string values from every possible place
If ((n_i.value ∈ JS_prop) && (n_i.value ∈ JS_fun)) **then**
JS ← JS ∪ n_i.value;
End if
elseif (ni.matches(URL_prop) **then**
U ← U ∪ n_i.value;
End if
End for
X_{PV} ← H ∪ JS ∪ U;
Return vulnerable string payload vector X_{PV};
End

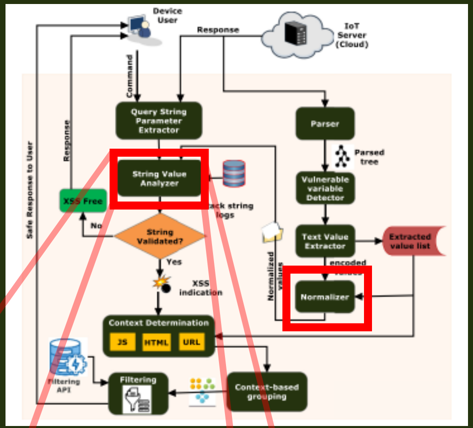
Detailed Design

Normalizer:

- Normalizes the extracted vulnerable strings
- Performs decoding of these values

Relevant encodings category for (“<”).

Encoding name	code
URL encoding	%3C
HTML character entity	<
HTML decimal character	<
JS single escape character	\<
JS hex escape sequence	\x3C
HTML hexadecimal character	<
JS Unicode escape sequence	\u003C

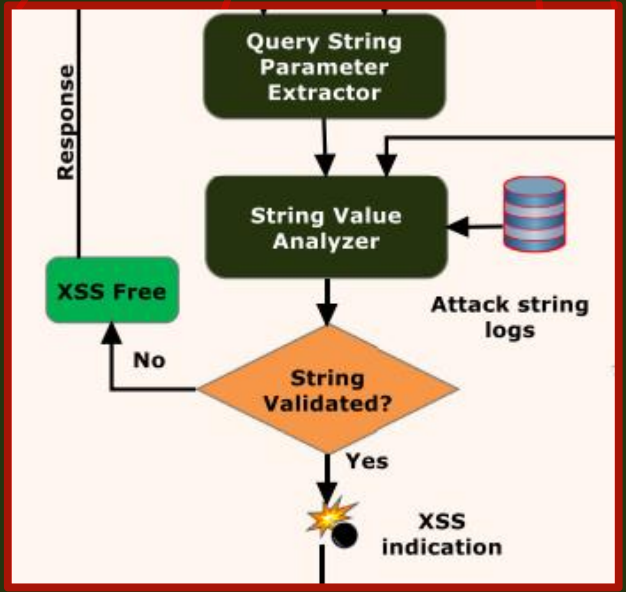


String value analyzer:

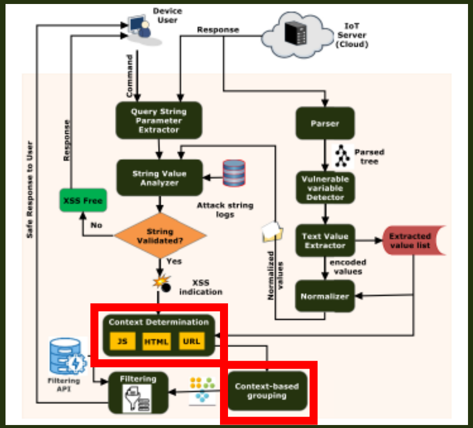
- Key task
- Extracted scripts from the HTTP request and response
- Vulnerable location

- **Boyer-Moore algorithm**
高效率的字串搜尋演算法，用於在一個主字符串中尋找特定模式的出現位置，是一種基於字符比對的算法。

- Found a match in blacklisted logs



Detailed Design |



Context determination:

- Recognize the context of the malicious attack string
- Execute correct attack filter APIs

Ex. <IMG SRC=‘javascript:alert("RSnake says, ’XSS’")‘>

  - HTML tag name
  - Attribute value: quoted

```
For each Cᵢ ∈ Temp do
X_I ← extract value X from CT(X) for Cᵢ;
//X is the placeholder for variable.
If (X_I ∈ STRING) then
Γ↦CT_I: String;
Else if (X_I ∈ NUMERIC) then
Γ↦CT_I: Number;
Else if (X_I ∈ REGEX) then
Γ↦ CT_I: Regular expression;
Else if (X_I ∈Quoted Data) then
Γ↦ CT_I: Quoted Data;
Else if (X_I ∈ PCDATA) then
Γ↦ CT_I: Parsed character data;
Else if (X_I ∈ CDATA) then
Γ↦ CT_I: Character data;
End If
Temp ← modified(CT);
End for
Return Temp;
End
```

Context-based determination:

- Grouping of such kind of scripts (p.15 Algo)
- Levenshtein distance measure the difference

- **萊文斯坦距離 Levenshtein distance**
量化兩字串間差異的演算法。

<script>alert(48a$bc);</script>
<script>alert(48xv&ez);</script>
S 代表字母 ； N 代表數字
➲ **<script>alert(48-S-);</script>**

Detailed Design

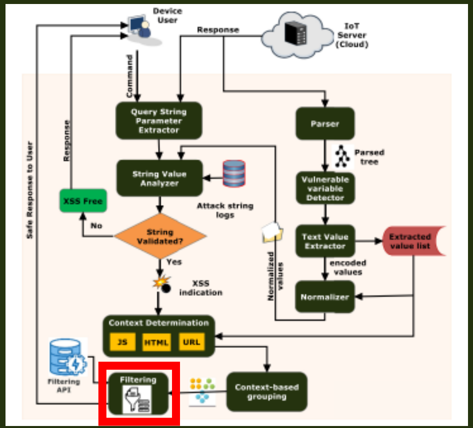
Filtering:

- Help of filtering APIs
- Produces safe response

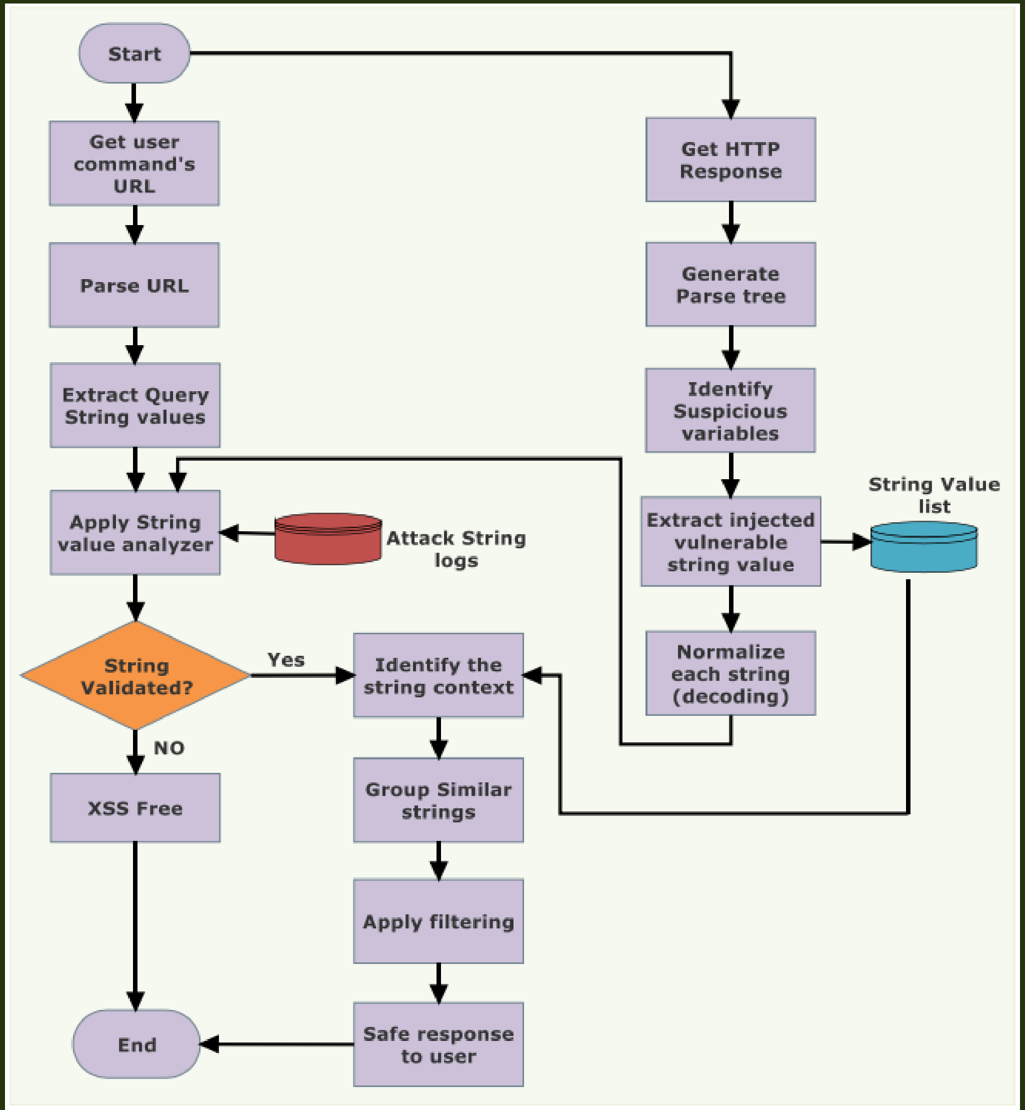
Input: context information and extracted string value
Output: modified response (H_M')
H ← extracted vulnerable HTML values;
JS ← extracted vulnerable JS values;
U ← extracted vulnerable URL values;
Temp ← context of each vulnerable string value;
F_API ← Externally available Filtered APIs (F_1, F_2, F_3... F_N);

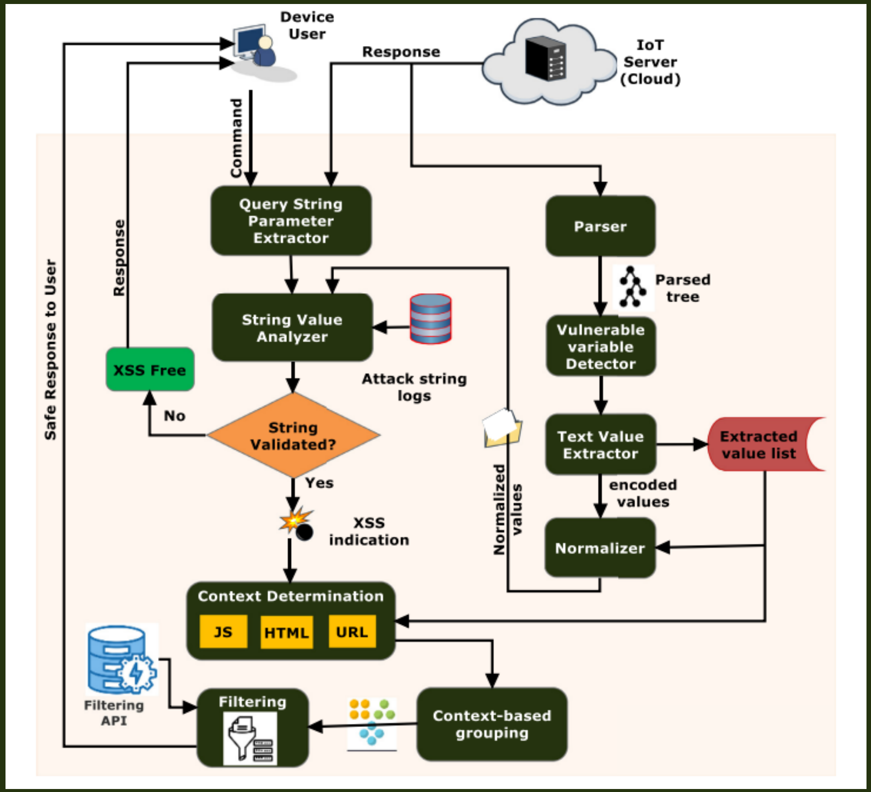
- G_TP (Grouped Templates): 儲存生成的分組模板
- m (Grouped Templates): 單個分組模板
- P (Levenshtein Distance): 儲存Levenshtein distance
- C (Context): 儲存上下文
- H_M' (Modified Response): 儲存修改後的網頁

Start
G_TP ← Ø; m ← Ø;
P ← Ø;
C ← Ø;
//generate grouped template for each category of extracted string value
For each $h_i \in$ H **do**
P ← **Levenshtein-distance (h_i, h_{i+1});**
If (P > β) **then**
Accept (h_i, h_{i+1});
m ← create grouped template (h_i, h_{i+1});
G_TP ← G_TP ∪m;
else
Discard (h_i, h_{i+1});
Select other pair;
End for
For each $Js_i \in$ JS **do**
P ← **Levenshtein-distance (Js_i, Js_{i+1});**
If (P > β) **then**
Accept (Js_i, Js_{i+1});
m ← create grouped template (Js_i, Js_{i+1});
G_TP ← G_TP ∪m;
else
Discard (Js_i, Js_{i+1});
Select other pair;
End for
For each $u_i \in$ U**do**
P ← **Levenshtein-distance (u_i, u_{i+1});**
If (P > β) **then**
Accept (u_i, u_{i+1});
m ← create grouped template (u_i, u_{i+1});
G_TP ← G_TP ∪m;
else
Discard (u_i, u_{i+1});
Select other pair;
End for
// apply filtering API on each template
For each $m_i \in$ G_TP **do**
C ← context(m(X_i)) ∈ Temp; *//Xi is the placeholder for vulnerable value*
F_i ← ($F_i \in$ F_API) && ($F_i \in$ matches C);
Apply F_i to X_i;
H_M' ← Modify X_i in received response web page;
End for
Return H_M';
End



Detailed Design



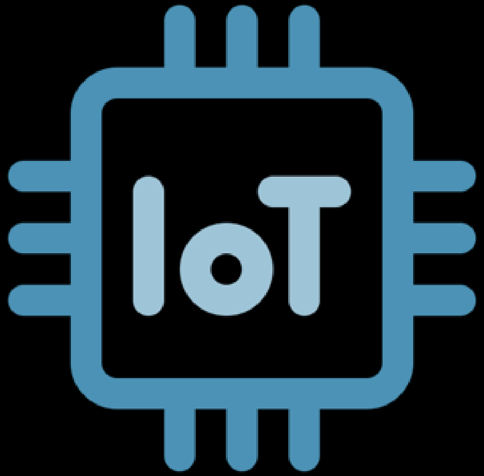


03. Experiment Result

Environment

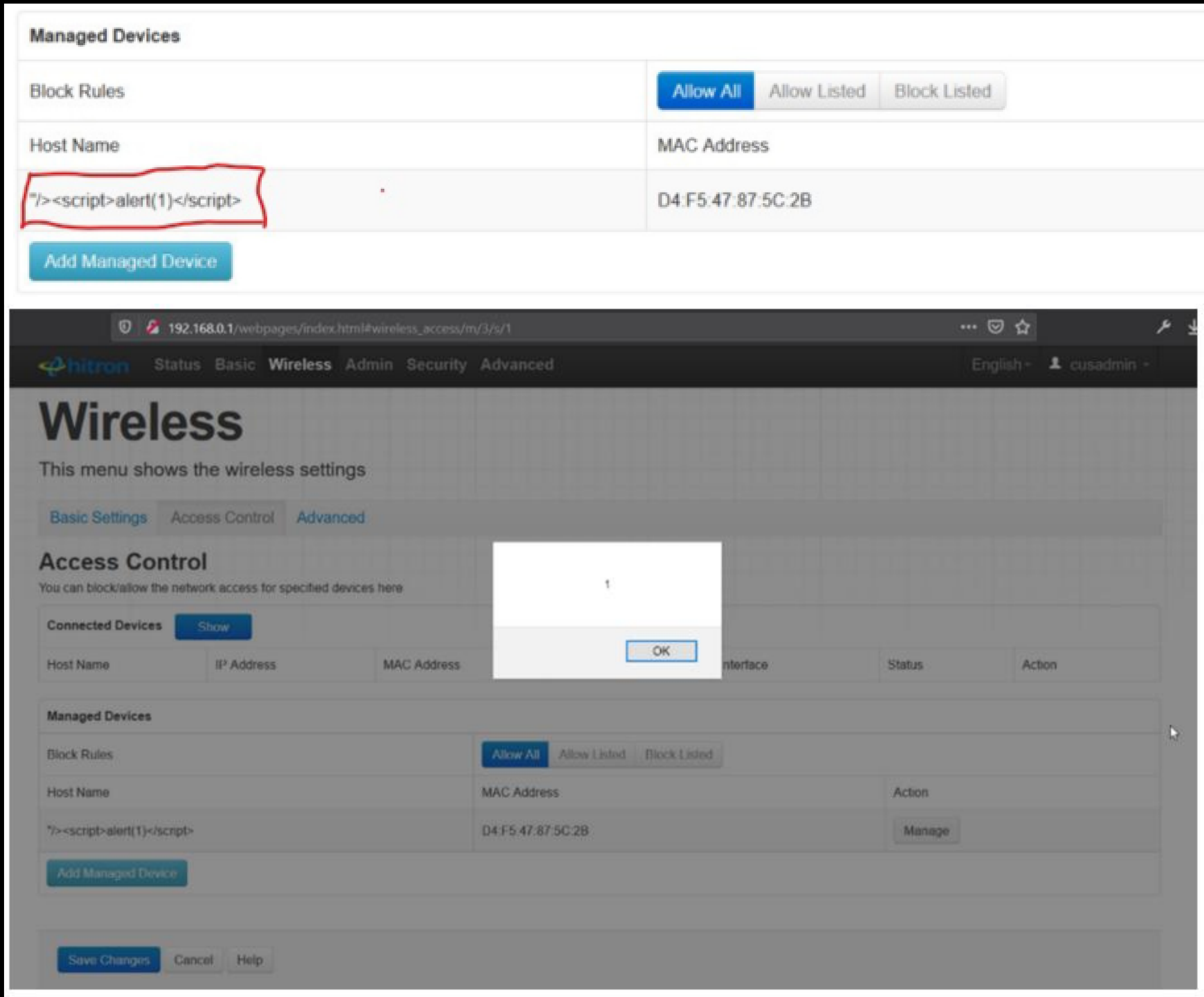
- Python programming language
- Intel® Core TM i5-6600k, 3.9GHz CPU
- 16 GB RAM, 1 TB HDD and 256 GB SSD
- html5lib parser
- BeautifulSoup python library (2022)
- Boyer-Moore algorithm

Azure
Cloud service



Experimental Results

Device Name	Hitron CODA router	Bosch IP camera CPP4
Model No.	4582u	Flexidome IP indoor 5000 HD
Vulnerable firmware version	7.1.1.30	7.10
XSS Vulnerability	CVE-2020-8824	CVE-2021-23848
Type of XSS attack	Stored	Reflected



Experimental Results

- HTML tag
- Event
- JS method

No.	Attack Vector Categories	Example Patterns
1.	Malicious HTML Tags (MHT)	<INPUT TYPE="IMAGE" SRC="javascript:alert('XSS');"> <BODY BACKGROUND="javascript:alert('XSS')"> <BODY ONLOAD=alert('XSS')> <BGSOUND SRC="javascript:alert('XSS');"> <BR SIZE="&{alert('XSS')}"> <TABLE BACKGROUND="javascript:alert('XSS')"> <TABLE><TD BACKGROUND="javascript:alert('XSS')">
2.	Script Embedded Malicious Attributes (SEMA)	XSS</A> XSS</A> <video src=1 href=1 onerror="javascript:alert(1)"></video> <body src=1 href=1 onerror="javascript:alert(1)"></body> <image src=1 href=1 onerror="javascript:alert(1)"></image>
3.	Exploited HTML Event Method (EHEM)	<IMG SRC= onmouseover="alert('xss')"> <IFRAME SRC=# onmouseover="alert(document.cookie)"></IFRAME> xxs link xxs link <IMG SRC=# onmouseover="alert('xxs')"> <html onMouseOver html onMouseOver="javascript:javascript:alert(1)"></html onMouseOver> <html onMouseEnter html onMouseEnter="javascript:parent.javascript:alert(1)"></html onMouseEnter>
4.	Malicious JS Variable (MJV)	<SCRIPT =">" SRC="http://ha.ckers.org/xss.js"></SCRIPT> <SCRIPT a=">" " SRC="http://ha.ckers.org/xss.js"></SCRIPT> <SCRIPT "a='>'" SRC="http://ha.ckers.org/xss.js"></SCRIPT> <SCRIPT a='>' SRC="http://ha.ckers.org/xss.js"></SCRIPT> <SCRIPT a=">'>" SRC="http://ha.ckers.org/xss.js"></SCRIPT>
5.	Malicious JS Methods (MJM)	<script>({set/**/$($){_/**/setter=$,_=javascript:alert(1)}}).$=eval</script> <script>{0:#0=eval/#0#/#0#(javascript:alert(1))}</script> <script\x0Atype="text/javascript">javascript:alert(1);</script> ""><\x3Cscript>javascript:alert(1)</script> ""><\x00script>javascript:alert(1)</script> <a href="data:application/x-x509-user-cert;
base64
,PHNjcmlwdD5hbGVydCgxKTwvc2NyaXB0Pg=="& - #09;& -#10;& - #11>X</a https://www.google<script.com>alert(document.location)</script
6.	Obfuscated Script Embedded URLs (OSEU)	test test <img alt="'><img src=x:x onerror=javascript:alert(1)>"> <div style="list-style:url(http://foo.f)\20url(javascript:javascript:alert(1));">X <META HTTP-EQUIV="refresh" CONTENT="0;url=javascript:javascript:alert(1);"> <META HTTP-EQUIV="refresh" CONTENT="0; URL=http://;URL=javascript:javascript:alert(1);">

[(Manico and Hansen, 2022, XSS Vectors Cheat Sheet 2022,
XSS Script Cheat Sheet for Web Application 2022, Cross-site scripting (XSS) cheat sheet 2022, Heiderich, 2022)]

Experimental Results

$$TNR = \frac{TN}{TN + FP}$$

$$FPR = \frac{FP}{FP + TN}$$

$$FNR = \frac{FN}{FN + TP}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Injected 100 attack strings of each six categories

Hitron CODA 4582u

Observed experimental results of integrating the proposed approach on router's device web interface.

Attack Categories	Total	TP	FP	FN	TN	TNR	FPR	FNR	Accuracy
MHT	100	88	1	3	8	0.889	0.111	0.033	0.96
SEMA	100	85	2	3	10	0.833	0.167	0.034	0.95
EHEM	100	90	1	2	7	0.875	0.125	0.022	0.97
MJV	100	84	2	3	11	0.846	0.154	0.034	0.95
MJM	100	86	1	1	12	0.923	0.077	0.011	0.98
OSEU	100	88	2	5	5	0.714	0.286	0.054	0.93

Bosch IP camera

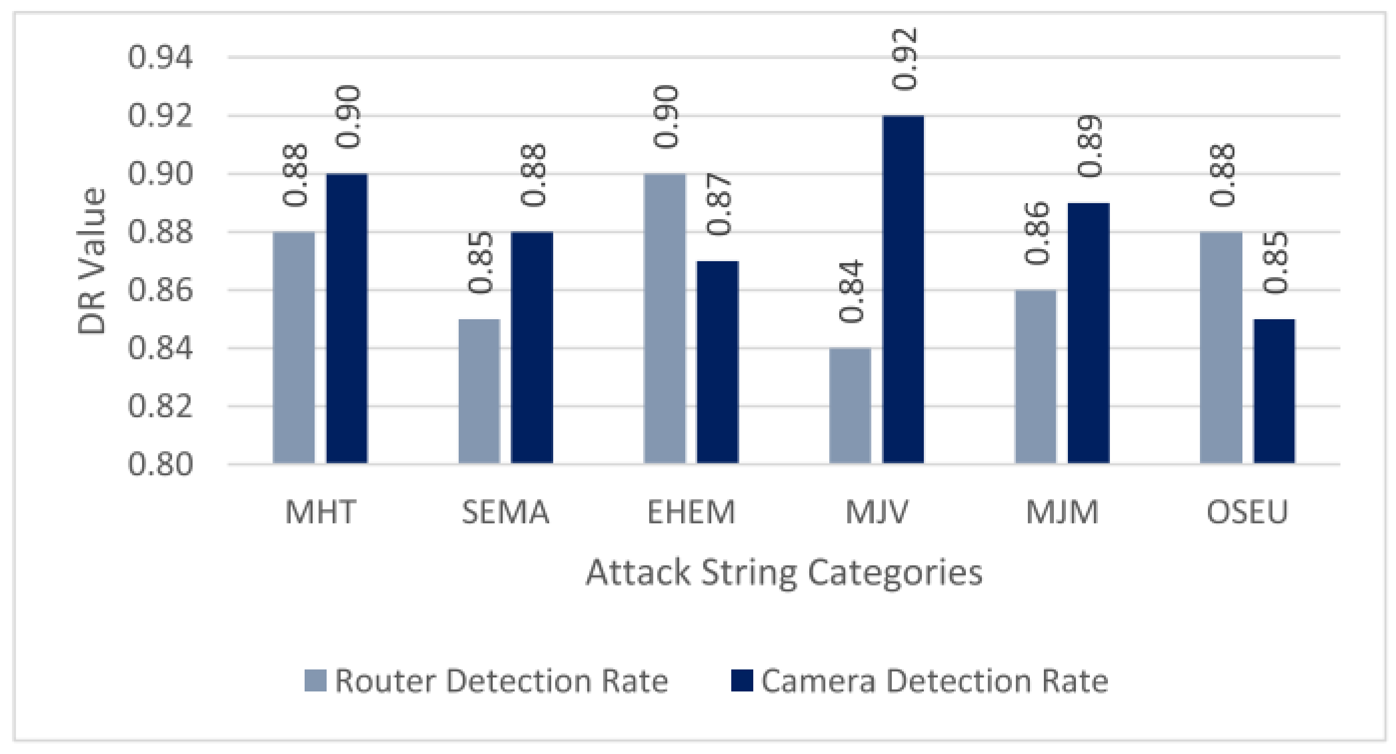
Observed experimental results of integrating the proposed approach on Camera's device web interface.

Attack Categories	Total	TP	FP	FN	TN	TNR	FPR	FNR	Accuracy
MHT	100	90	1	2	7	0.875	0.125	0.022	0.97
SEMA	100	88	2	2	8	0.800	0.200	0.022	0.96
EHEM	100	87	2	2	9	0.818	0.182	0.022	0.96
MJV	100	92	1	1	6	0.857	0.143	0.011	0.98
MJM	100	89	1	1	9	0.900	0.100	0.011	0.98
OSEU	100	85	3	1	11	0.786	0.214	0.012	0.96

[(Manico and Hansen, 2022, XSS Vectors Cheat Sheet 2022, XSS Script Cheat Sheet for Web Application 2022, Cross-site scripting (XSS) cheat sheet 2022, Heiderich, 2022)]

Experimental Results

$$\textit{Attack detection rate} = \frac{TP}{TP + TN + FP + FN}$$



All attack categories ranges: 0.8-0.9

Performance evaluation

F – measure

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F - Measure = \frac{2 * Precision * Recall}{Precision + Recall}$$

Performance evaluation outcomes of the proposed approach.

Attack Categories	Router Precision	Recall	F-measure	Camera Precision	Recall	F-measure
MHT	0.989	0.967	0.978	0.989	0.978	0.983
SEMA	0.977	0.966	0.971	0.978	0.978	0.978
EHEM	0.989	0.978	0.983	0.978	0.978	0.978
MJV	0.977	0.966	0.971	0.989	0.989	0.989
MJM	0.989	0.989	0.989	0.989	0.989	0.989
OSEU	0.978	0.946	0.962	0.966	0.988	0.977

High value of F-measure > 0.9

F-test hypothesis

S1: malicious attack strings injected 數量

S2: malicious attack strings detected數量

- Null Hypothesis: S1=S2
- Alternate Hypothesis: S1>S2
- Significance level (α = 0.05)

F_{tab}=5.503

Router F_{cal}=1.14345

Camera F_{cal}= 1.12044

Fcal1 < Ftab and Fcal2 < Ftab

➲ accept alternate hypothesis

Limitations

☑ Identifies attack string
☑ Resemblance between attack strings
☑ good results
☒ New features attack string

➲ Automatically updating the available attack vector repository

04. Conclusions

Contributions

1. Design a fog-based intelligent IoT system infrastructure.

2. Boyer-Moore string matching algorithm, to detect reflected XSS.

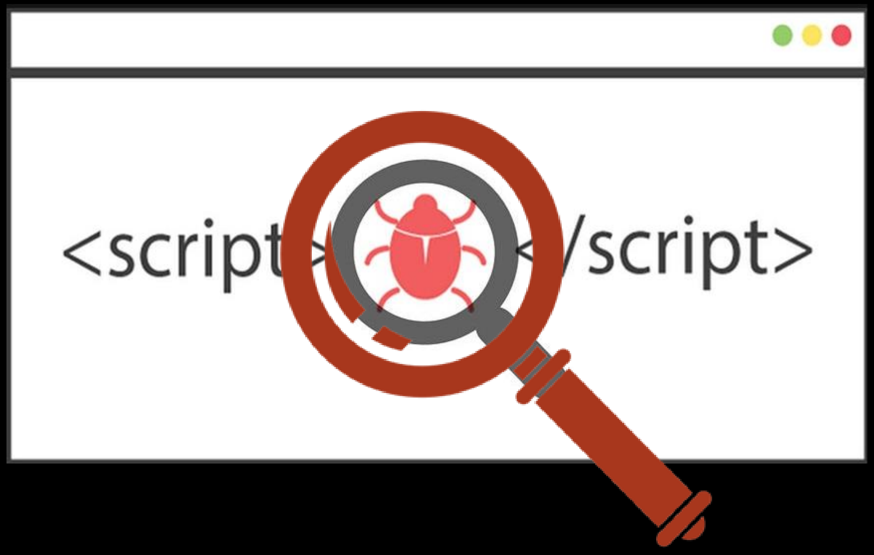
3. Construct parse tree, compare with debarred attack strings using Boyer-Moore to identify stored XSS.

4. Attack demonstration exploiting known XSS vulnerability in Hitron CODA 4582u router and Bosch Flexidome IP indoor 5000 HD camera.

5. Examine the performance using prominent metrices comprising precision, recall, F-measure, and FPR.

Conclusion

1. First of its kind research study.
2. Fog computing environment to reduce the latency and bandwidth
3. Context-based grouping, highest accuracy of 0.98
4. (Future) Automatically updating the available attack vector



Q & A