

# **An Efficient and Usable Client-Side Cross Platform Compatible Phishing Prevention Application**

## **THIRD REVIEW**

### **Guide**

Dr. Angelin Gladston  
Associate Professor  
Department of CSE

### **Submitted by**

N. Dhanush	2016103021
G. Santhosh	2016103057
S. Ben Stewart	2016103513

# **OUTLINE**

1. INTRODUCTION
2. OVERALL OBJECTIVE
3. LITERATURE SURVEY
4. PROPOSED SYSTEM
5. HIGH LEVEL BLOCK DIAGRAM
6. MODULE LIST
7. IMPLEMENTATION
8. EVALUATION METRICS
9. TEST CASES
10. REFERENCES

# INTRODUCTION

- Phishing
- Lists of such sites
- Time constraints
- Computational resources
- Vulnerabilities
- Cross platform

# OVERALL OBJECTIVE

- Create a phishing list
- Cross Platform application
- Web browser add-on
- Provide temporal resilience
- Remove false positives from list

# LITERATURE SURVEY

- Previous work by Samuel Marchal, Giovanni Armano, Tommi Grondahl, Kalle Saari, Nidhi Singh, and N. Asokan
- IEEE Trans. Comput., vol. 66, no. 10, pp. 1717-1733, Oct. 2017
- Implemented a client-side phishing prevention application.
- Had background tasks communicate with a browser add-on.
- Not platform independent.

# AUTOMATIC PHISHING CLASSIFICATION

- Colin Whittaker, Brian Ryner and Marria Nazif for Google
- Proc. Netw. Distrib. Syst. Security Symp., 2010
- Features used
  - 1. The URL of the page
  - 2. The HTML page contents
  - 3. The host server details
- Needs blacklist updating.

# CANTINA

- Guang Xiang, Jason Hong, Carolyn P. Rose and Lorrie Cranor
- ACM Trans. Inf. Syst. Secur., 2011
- Page similarity
- SHA 1 algorithm
- Easy to break
- Performance gains

# AUTO UPDATED WHITELIST

- Ankit Kumar Jain and B. B. Gupta
- EURASIP J. Inf. Secur., vol. 2016, no. 1, Dec. 2016
- Whitelist
  - a. the domain name
  - b. the IP address
- Reverts to old system if not in whitelist

# FUZZY ROUGH SET FEATURE SELECTION TO ENHANCE PHISHING ATTACK DETECTION

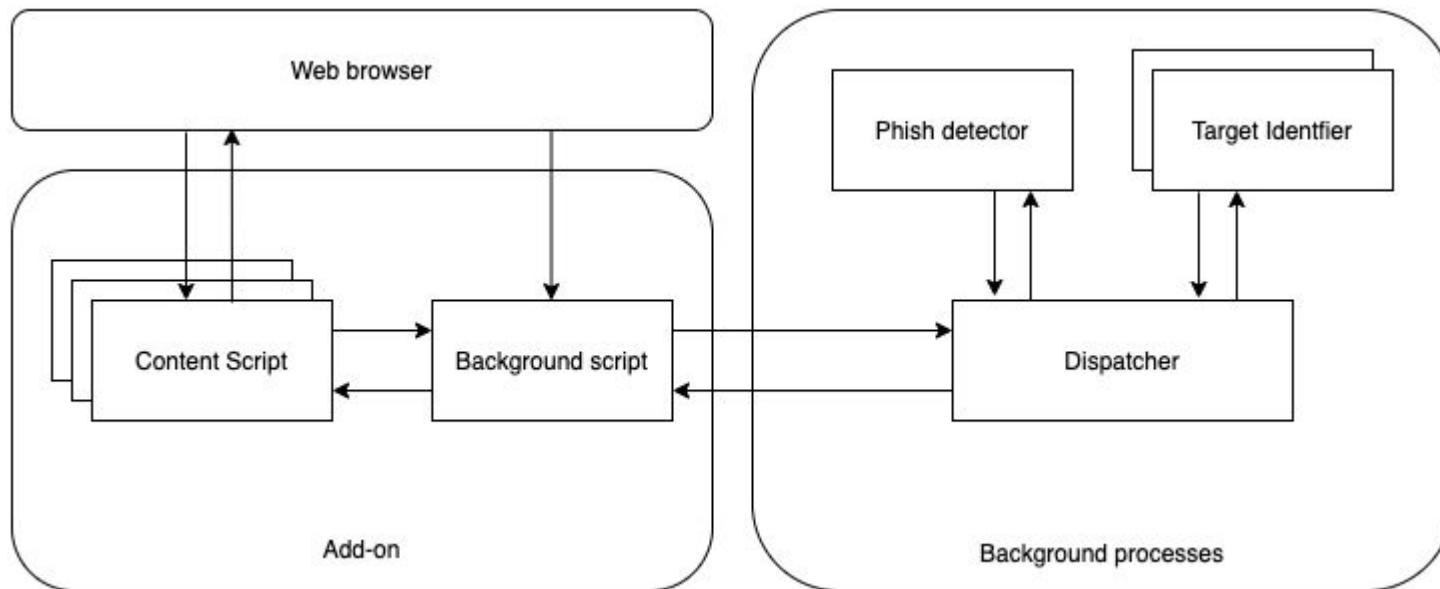
- Mahdieh Zabihimayvan and Derek Doran
- IEEE International Conference on Fuzzy Systems, June 2019
- Fuzzy Rough Set (FRS) theory
- Feature selection algorithm
- Random Forest classification
- No third party features

<b>Paper</b>	<b>Journal/Conf. , Year</b>	<b>Contributions</b>	<b>Limitations</b>
Large-Scale Automatic Classification of Phishing Pages	Proc. Netw. Distrib. Syst. Security Symp., 2010	Machine learning model can be used with reliable accuracy.	Needs blacklist for updating.
CANTINA: A feature-rich machine learning framework for detecting phishing Web sites	ACM Trans. Inf. Syst. Secur., 2011	SHA1 based similarity check for similar looking sites.	SHA1 could be manipulated.
A novel approach to protect against phishing attacks at client side using auto-updated white-list	EURASIP J. Inf. Secur., vol. 2016, no. 1, Dec. 2016	Auto-updated whitelist for faster detection of sites on average.	Not temporally resilient.
Fuzzy Rough Set Feature Selection to Enhance Phishing Attack Detection	IEEE International Conference on Fuzzy Systems, June 2019	Feature selection.	Not a user oriented application.

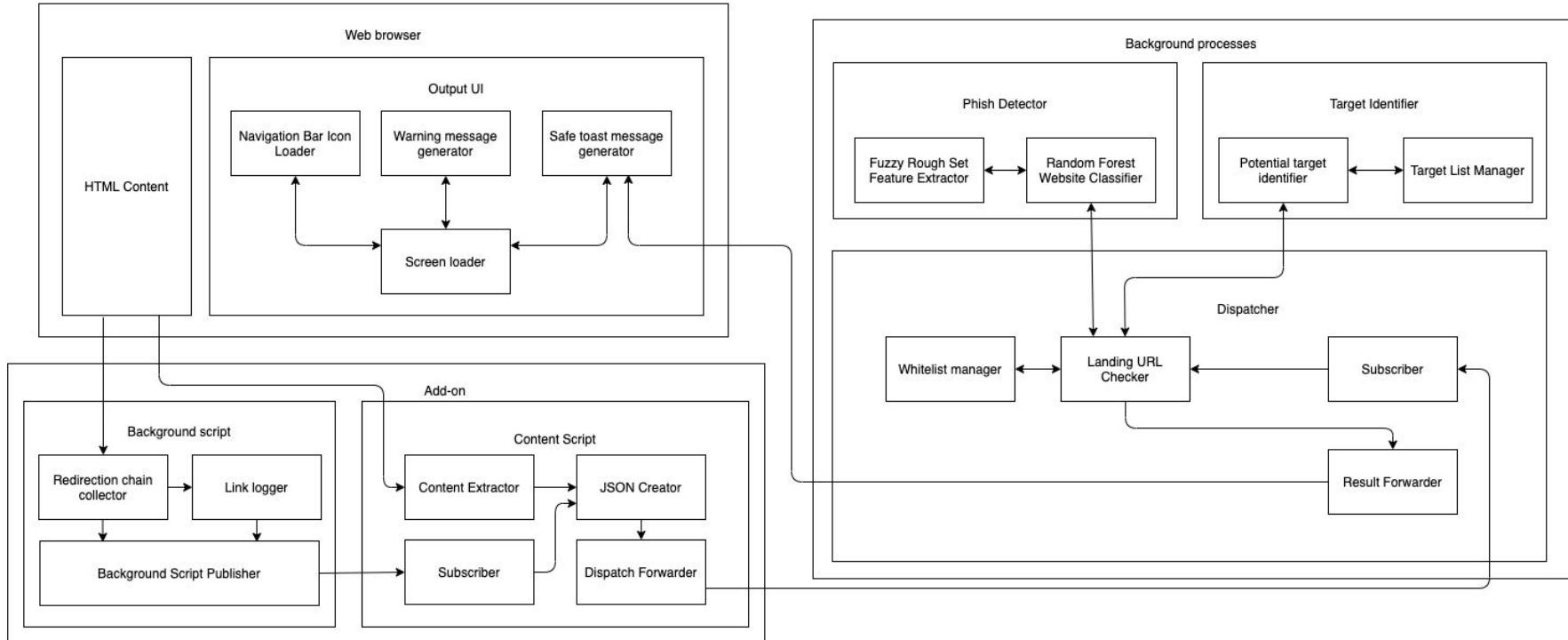
# **PROPOSED SYSTEM**

- Platform independent
- Browser add-on
- Reduce false warnings
- Context independent detection
- Static observations

# SYSTEM ARCHITECTURE



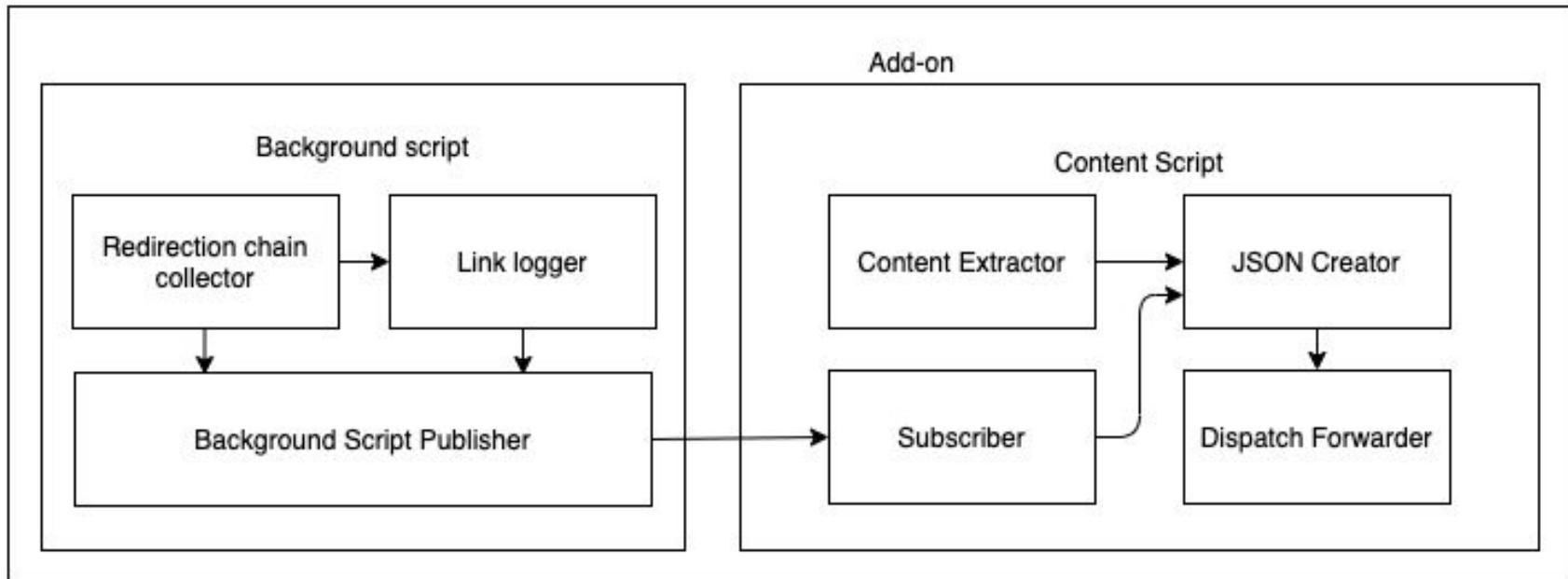
# HIGH LEVEL BLOCK DIAGRAM



# MODULE LIST

- Add on
  - a. Background script
  - b. Content script
- Background process
  - a. Dispatcher
  - b. Phish Detector
  - c. Target Identifier
- Web Browser
  - a. HTML content
  - b. Output UI

# ADD-ON



# BACKGROUND SCRIPT

*Begin*

*For each page load redirect*

*Add listener to that event*

*Get the list of redirects from listener*

*If page is fully loaded*

*Send the list of redirects to content script*

*Done*

*End*

# CONTENT SCRIPT

*Begin*

*For each page load redirect*

*If page is fully loaded*

*Get the URL from the tab*

*Get the HTML content from innerHTML tag*

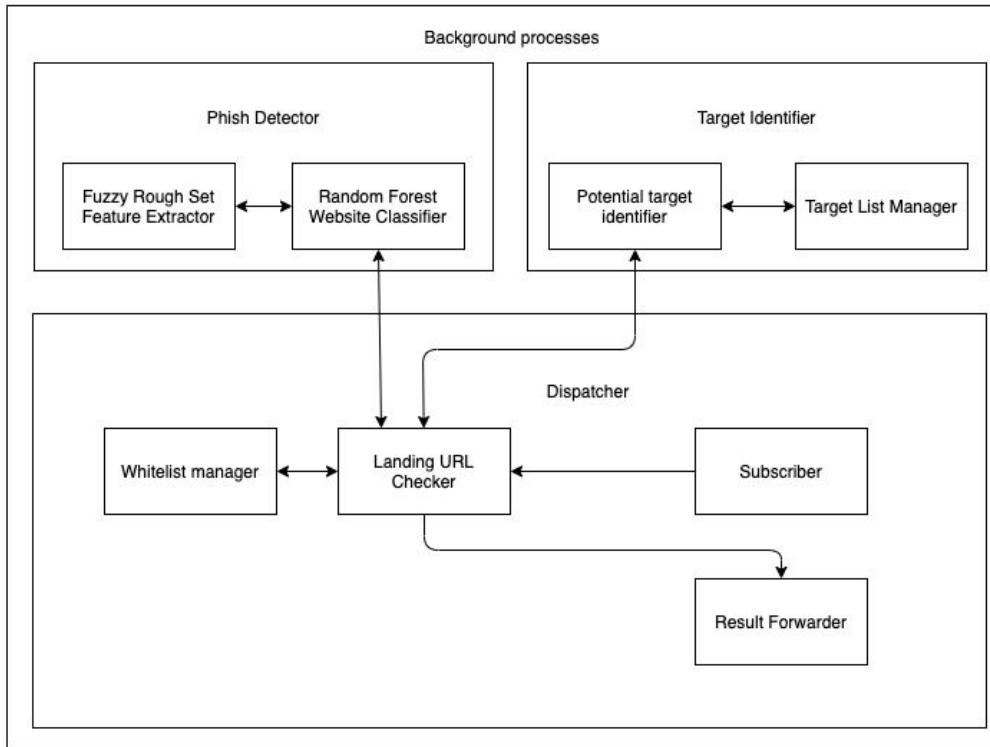
*Get redirection list from background script*

*Send them to the background process*

*Done*

*End*

# BACKGROUND PROCESS



# DISPATCHER

*Begin*

*If page address is in whitelist*

*Send the GREEN signal*

*Else*

*Send content to phish detector*

*Get results from phish detector*

*If phish is FALSE*

*Send the GREEN signal*

*Else*

*Send the RED signal*

*Send content to target identifier*

*If target is found*

*Publish target*

*Else*

*No target matched*

*End*

# PHISH DETECTOR

*Begin*

*For each page URL*

*Get the fuzzy set feature values for the URL*

*Load the saved random forest model*

*Publish the result*

*Done*

*End*

# FUZZY ROUGH SET

*Begin*

*Compute indiscernibility matrix  $M(A)$*

*Reduce  $M$  using absorption laws*

*d - number of non-empty fields*

*Initialise all fields*

*For all fields*

*Compute fields using formulas  $R=SUT$*

*Done*

*End*

# RANDOM FOREST MODEL

*Begin*

*For each record in dataset*

*Get the fuzzy set feature values*

*Create an arff file to save results*

*Done*

*Train the dataset with at least 7 splits as random forest*

*Save the model as pkl file*

*End*

# TARGET IDENTIFIER

*Begin*

*Remove all href tags in page*

*Get the hash value for page content*

*Compare with values in hash list*

*If match*

*Display target*

*Else*

*No target found*

*End*

# SHA

*Begin*

*Input is an array 8 items long where each item is 32 bits.*

*Calculate all the function boxes and store those values.*

*Store input, right shifted by 32 bits, into output.*

*Store the function boxes.*

*Store (Input H + Ch + ( (Wt+Kt) AND 2^31 ) ) AND 2^31 As  
mod1*

*Store (sum1 + mod1) AND 2^31 as mod2*

*Store (d + mod2) AND 2^31 into output E*

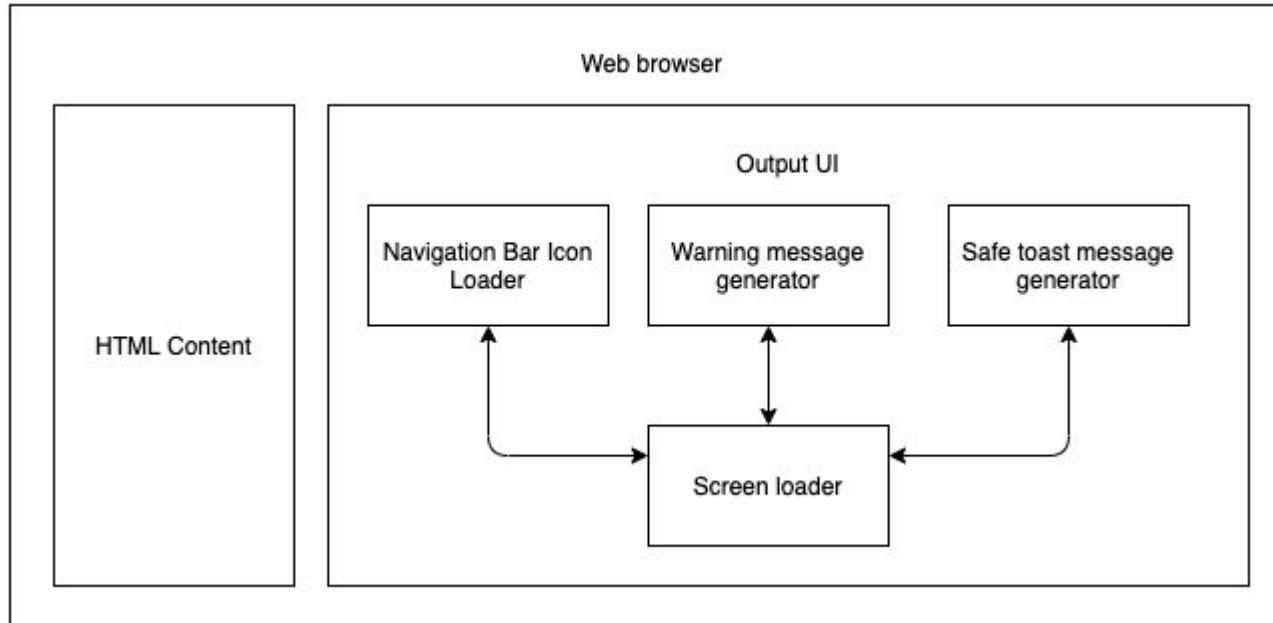
*Store (MA + mod2) AND 2^31 as mod3*

*Store (sum0 + mod3) AND 2^31 into output A*

*Output is an array 8 items long where each item is 32 bits.*

*End*

# WEB BROWSER



# OUTPUT UI

*Begin*

*If site is phish*

*Change icon to red*

*Display warning message*

*If site has target*

*Display target link*

*Else*

*Display no target*

*Else*

*Change icon to green*

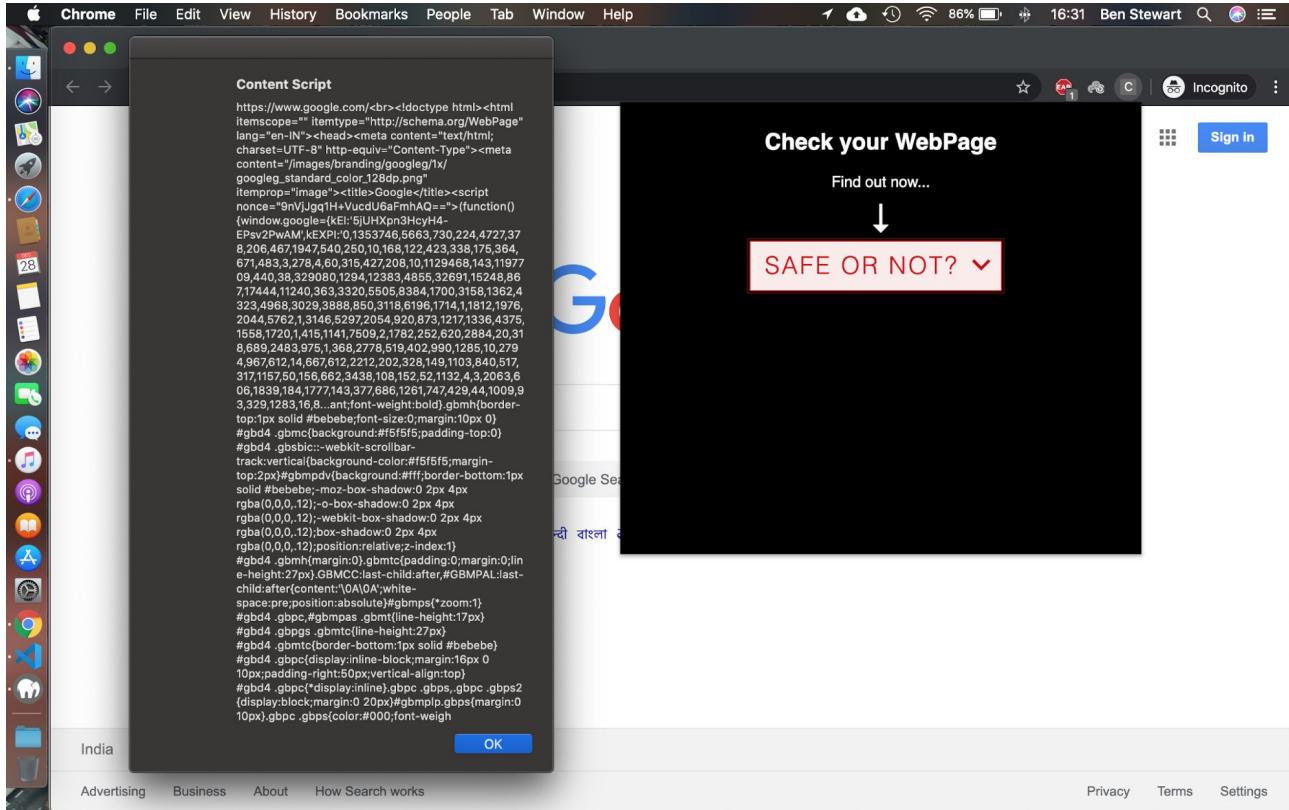
*Display safe to proceed message*

*End*

# IMPLEMENTATION

- Add on
  - a. Background script
  - b. Content script
- Background process
  - a. Dispatcher
  - b. Phish Detector
  - c. Target Identifier
- Web Browser
  - a. HTML content
  - b. Output UI

# CONTENT SCRIPT



# CONTENT SCRIPT

```
//Retrieve URL JS  
tablink = tab.url;
```

```
//Retrieve Page content PHP  
$site=$_POST['url'];  
$html = file_get_contents($site);
```

# BACKGROUND SCRIPT

The screenshot shows a Mac desktop with a Chrome browser window open to the Amazon Kindle Wireless Reading Display product page. The browser's status bar at the bottom right shows a green checkmark icon and the URL [https://www.amazon.com/Kindle-Wireless-Reading-Display-Globally/dp/B003FSUDM4/ref=amb\\_link\\_353259562\\_2?pf\\_rd\\_m=ATVPDKIKX...](https://www.amazon.com/Kindle-Wireless-Reading-Display-Globally/dp/B003FSUDM4/ref=amb_link_353259562_2?pf_rd_m=ATVPDKIKX...). To the right of the browser window, a series of status bar notifications are displayed, each with an orange downward arrow icon and a URL:

- <http://tinyurl.com/KindleWireless>  
307: Internal (browser cached) redirect to <https://tinyurl.com/KindleWireless>
- <https://tinyurl.com/KindleWireless>  
301: Permanent redirect to <http://www.amazon.com/Kindle-Wireless-Re>
- <https://r.eablink.com/?key=a982cfabb5482be54a4d3a52b21>  
302: Temporary redirect to <http://www.amazon.com/Kindle-Wireless-Re>
- <http://www.amazon.com/Kindle-Wireless-Reading-G>  
307: Internal (browser cached) redirect to [https://www.amazon.com/Kindle-Wireless-Reading-Display-Globally/dp/B003FSUDM4/ref=amb\\_link\\_353259562\\_2?pf\\_rd\\_m=ATVPDKIKX...](https://www.amazon.com/Kindle-Wireless-Reading-Display-Globally/dp/B003FSUDM4/ref=amb_link_353259562_2?pf_rd_m=ATVPDKIKX...)
- <https://www.amazon.com/Kindle-Wireless-Reading-Display-G>  
200: HTTP/1.1 200

The main content of the browser window shows the Amazon Kindle Wireless Reading Display product page, featuring a hand holding a Kindle device displaying a book cover.

# BACKGROUND SCRIPT

```
//URL path item  
url: pathItem.url,  
status: pathItem.status_line,  
redirect_type: pathItem.redirect_type,  
redirect_url: pathItem.redirect_url,  
meta_timer: pathItem.meta_timer
```

# **DATASET**

- 30 features
- 23827 records
- Scrapped from PhishTank
- Up-to-date

# FEATURE SELECTION

```
benstewart@ben:~/phish_detector$ master$ python sample.py  
(11054, 1)  
(11054, 30)  
[[-1]  
 [-1]  
 [-1]  
 ...  
 [-1]  
 [-1]  
 [-1]]  
[[[-1 1 1 ... 1 1 -1]  
 [ 1 1 1 ... 1 1 1]  
 [ 1 0 1 ... 1 0 -1]  
 ...  
 [-1 1 1 ... 1 -1 1]  
 [ 1 -1 1 ... 1 0 1]  
 [-1 -1 1 ... 1 1 1]]]
```

# FEATURE SELECTION

```
selector = RoughSetsSelector()  
X_selected = selector.fit(X, y).transform(X)
```

# RANDOM FOREST MODEL WITHOUT FEATURE SELECTION

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.  
[Parallel(n_jobs=1)]: Done 10 out of 10 | elapsed: 0.0s finished  
precision      recall    f1-score   support  
  
      -1       0.99      0.97      0.98      460  
       1       0.98      0.99      0.98      594  
  
  micro avg       0.98      0.98      0.98     1054  
  macro avg       0.98      0.98      0.98     1054  
weighted avg       0.98      0.98      0.98     1054  
  
The accuracy is: 0.9810246679316889
```

```
[[446 14]  
 [ 6 588]]
```

```
benstewart@ben ~/phish detector
```

# RANDOM FOREST MODEL WITH FEATURE SELECTION

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.  
[Parallel(n_jobs=1)]: Done 10 out of 10 | elapsed: 0.0s finished
```

	precision	recall	f1-score	support
-1	0.96	0.95	0.95	460
1	0.96	0.97	0.97	594
micro avg	0.96	0.96	0.96	1054
macro avg	0.96	0.96	0.96	1054
weighted avg	0.96	0.96	0.96	1054

```
The accuracy is: 0.961100569259962
```

```
[[435 25]  
 [ 16 578]]
```

```
benstewart@ben ~/phish detector
```

# RANDOM FOREST MODEL

```
//create model  
clf4=RandomForestClassifier(min_samples_split=7)  
clf4.fit(features_train, labels_train)  
//save the model  
joblib.dump(clf4, 'classifier/random_forest.pkl', compress=9)  
  
//feature weightage  
importances = clf4.feature_importances_  
//confusion matrix  
print metrics.confusion_matrix(labels_test, pred4)
```

# TARGET IDENTIFIER

```
[ benstewart@ben ~ ] ➤ """/target identifier ➤ python3 score-compare-tags.py compare-tags.json
Maximum f1 0.944 at threshold=35 tp=84 fp=4 fn=6 prec=0.955 rec=0.933
benstewart@ben ~ ] ➤ """/target identifier
```

```
tags1 = get_tags(lxml.html.parse(path1))
tags2 = get_tags(lxml.html.parse(path2))
diff = difflib.SequenceMatcher()
diff.set_seq1(tags1)
diff.set_seq2(tags2)
```

```
params['url'] = url
response=requests.get(url, headers=headers, params=params)
```

# AUTO UPDATED WHITELIST

Screenshot of the Network tab in the developer tools showing a single request to "clientServer.php".

Network Tab Headers:

- Elements
- Console
- Sources
- Network**
- Performance
- Memory
- >
- ⚠ 1
- ⋮

Network Tab Filters:

- Preserve log (unchecked)
- Disable cache (checked)
- Online
- Filter (empty)
- Hide data URLs (unchecked)

Network Tab Metrics:

- All XHR JS CSS Img Media Font Doc WS Manifest Other
- Only show requests with SameSite issues (unchecked)

Network Timeline:

1000 ms	2000 ms	3000 ms	4000 ms	5000 ms	6000 ms	7000 ms
---------	---------	---------	---------	---------	---------	---------

Network Requests Table:

Name	Status	Type	Initiator	Size	Time	Waterfall
clientServer.php	200	xhr	popup.js:12	405 B	7.50 s	

Network Summary:

- 1 requests
- 405 B transferred
- 4 B resources

Screenshot of the Network tab in the developer tools showing multiple requests from a popup window.

Network Tab Headers:

- Elements
- Console
- Sources
- Network**
- Performance
- Memory
- >
- ⚠ 2
- ⋮

Network Tab Filters:

- Preserve log (unchecked)
- Disable cache (checked)
- Online
- Filter (empty)
- Hide data URLs (unchecked)

Network Tab Metrics:

- All XHR JS CSS Img Media Font Doc WS Manifest Other
- Only show requests with SameSite issues (unchecked)

Network Timeline:

1000 ms	2000 ms	3000 ms	4000 ms	5000 ms	6000 ms	7000 ms
---------	---------	---------	---------	---------	---------	---------

Network Requests Table:

Name	Status	Type	Initiator	Size	Time	Waterfall
popup.html	200	document	Other	501 B	2 ms	
popup.js	200	script	popup.html	762 B	17 ms	
style.css	200	stylesheet	popup.html	1.8 KB	18 ms	
jquery.min.js	200	script	popup.html	29.6 KB	678 ms	
clientServer.php	200	xhr	popup.js:12	405 B	3.27 s	

Network Summary:

- 5 requests
- 33.1 KB transferred
- 87.7 KB resources
- Finish: 6.92 s
- DOMContentLoaded: 713 ms
- Load: 7

# AUTO UPDATED WHITELIST

```
//insert into whitelist  
white_list_file=open('whitelist.txt', "a+")  
white_list_file.write(url)
```

```
//search whitelist  
white_list_file = open('whitelist.txt').read()  
white_list = white_list_file.split('\n')  
if url in white_list:  
    #url is safe
```

# DISPATCHER

```
$decision=exec("python test.py $site 2>&1 ");
echo $decision;
```

*//temporal resilience*

```
$decision=exec("python scrape.py single-sites.json 1 data");
```

# OUTPUT UI

**Check your WebPage**

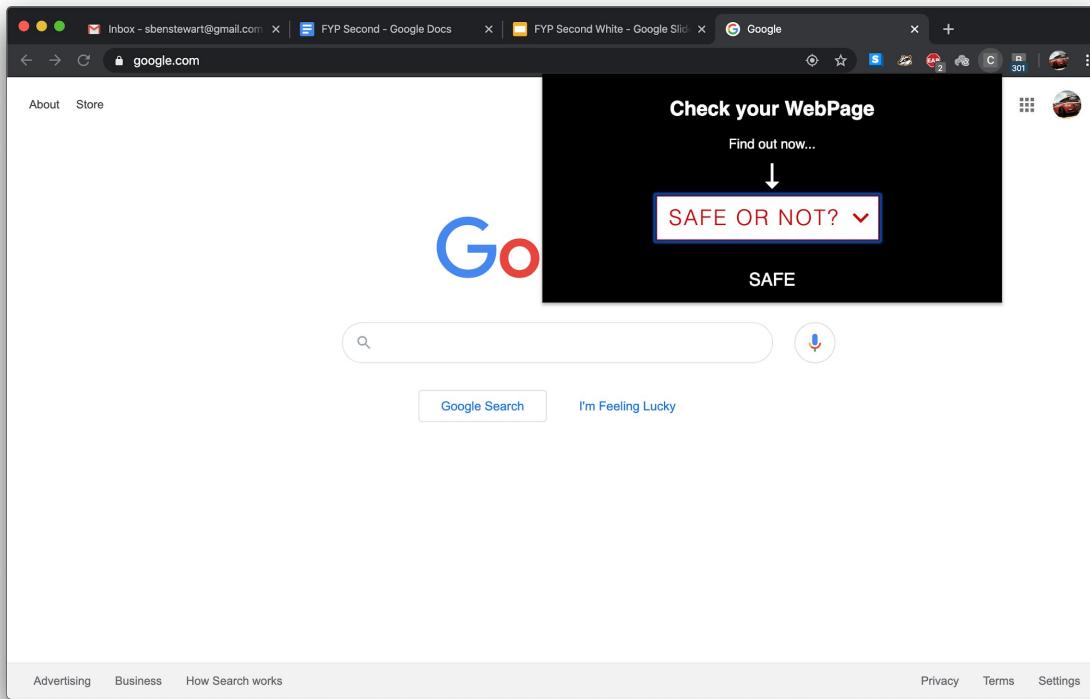
Find out now...



**SAFE OR NOT? ✓**

PHISHING and the most similar site is verizon with  
similarity 23.4957020057

# SAFE HTML CONTENT

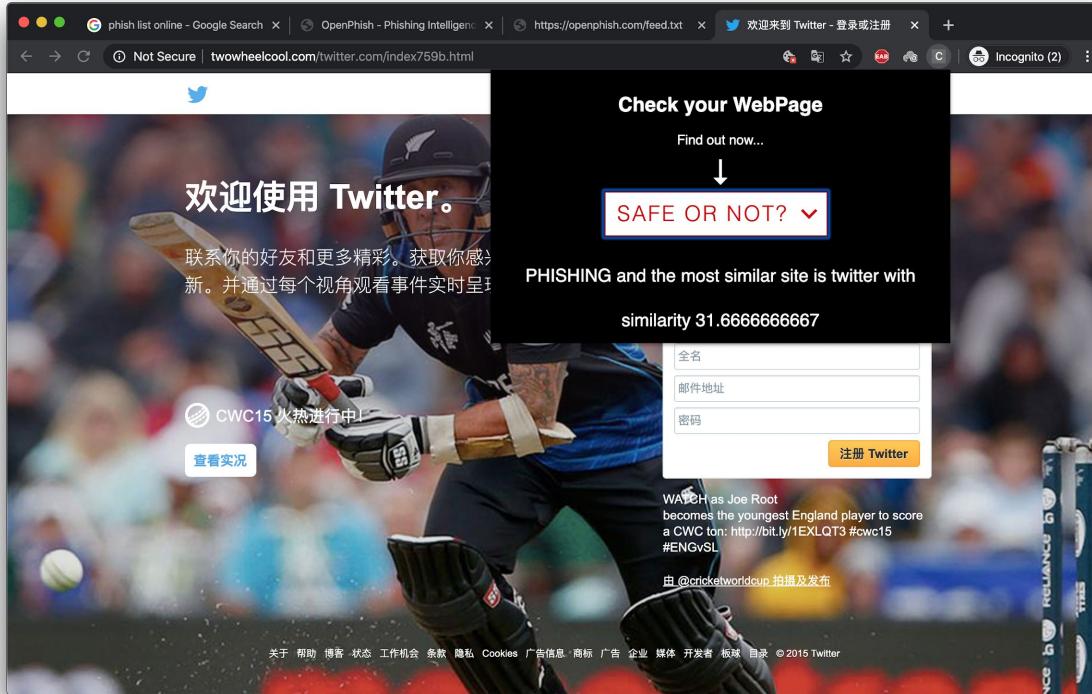


# SAFE HTML CONTENT

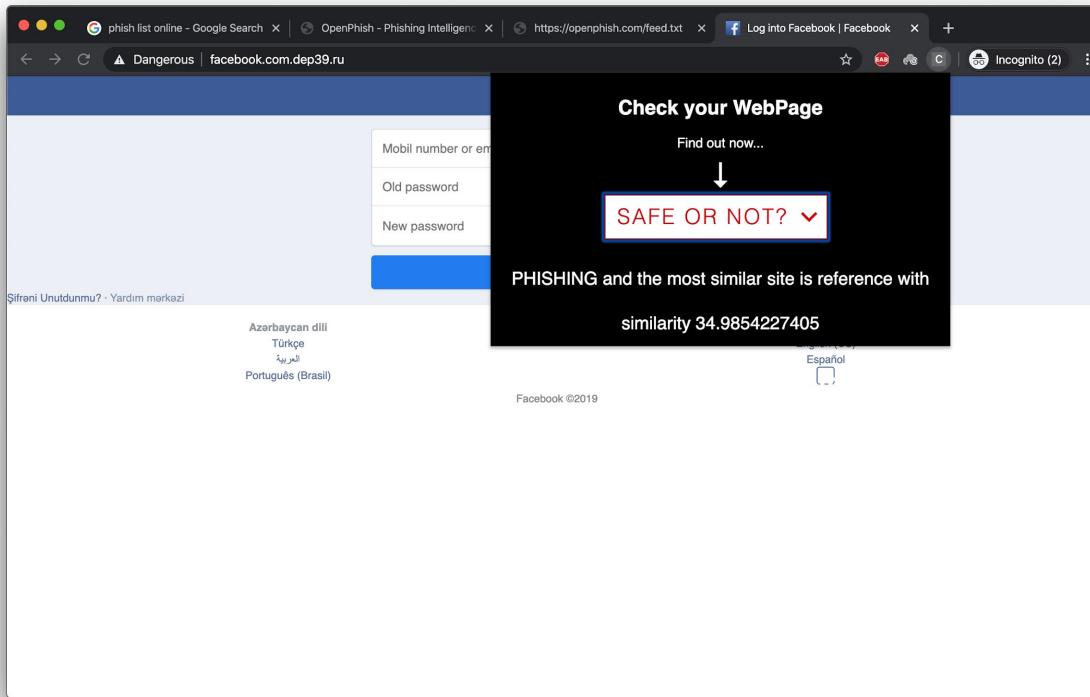
The screenshot shows a browser window with the URL [annauniv.edu](http://annauniv.edu). The main content is the official website of Anna University, featuring sections for TANCET 2020, GRADUATION DAY 2019, and TANCA 2019. A central sidebar displays news items and links to various university services. Overlaid on the right side is a dark rectangular box containing the text "Check your WebPage" at the top, followed by "Find out now.." with a downward arrow, and a large red button labeled "SAFE OR NOT?". Below this button, the word "SAFE" is displayed in white capital letters. To the right of the "SAFE" button is a vertical list of university services, each with a blue link:

- CoE - Genuineness Verification
- Constituent Colleges
- Distance Education
- Health Centre
- International Relations
- National Apprenticeship Schemes
- NIRF
- Patents
- Harassment Cell
- Ramanujan Computing Centre
- Recruitment Cell
- Regional Offices
- RTI Mandatory Disclosure
- Services - Transcripts Online
- Students' Activities
- Tender
- University Departments
- University Library

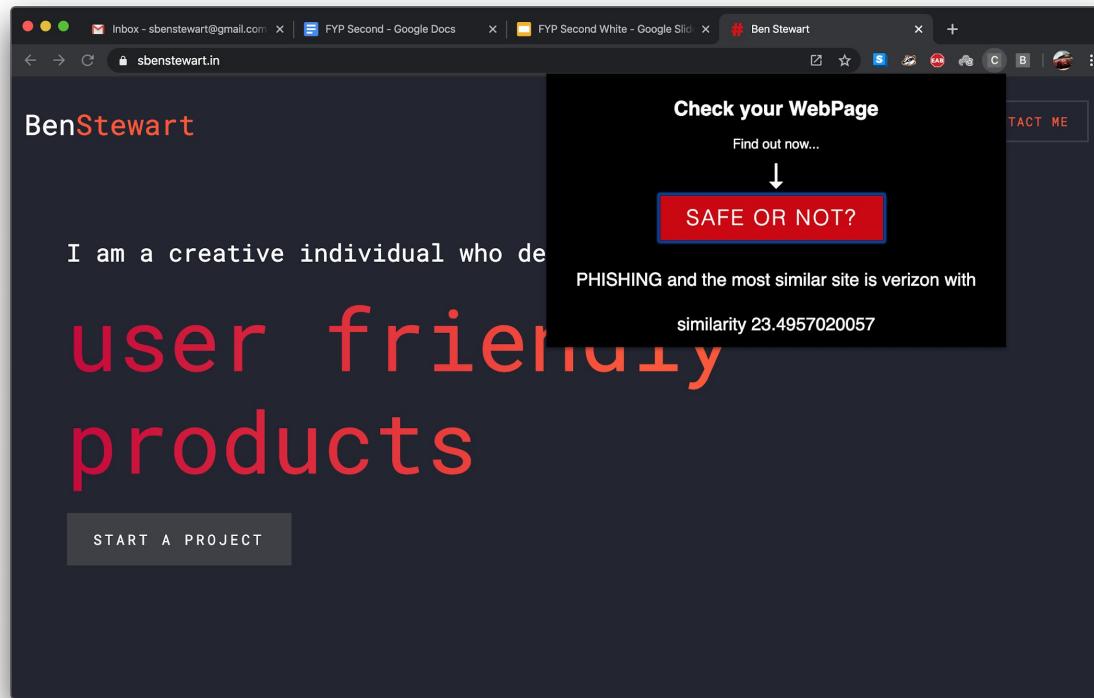
# PHISHING HTML CONTENT



# PHISHING HTML CONTENT



# WRONG HTML CONTENT



# EVALUATION METRICS

1. Phish detection accuracy

Accuracy =  $(TP+TN)/(TP+TN+FP+FN)$

2. Target detection ratio

Detection Ratio =  $(TP)/(TP+TN+FP+FN)$

3. Memory usage profiling

Current total memory usage = Total Memory - (Free + Buffers + Cached)

4. Addon rendering time

Rendering Time = End time - Start time

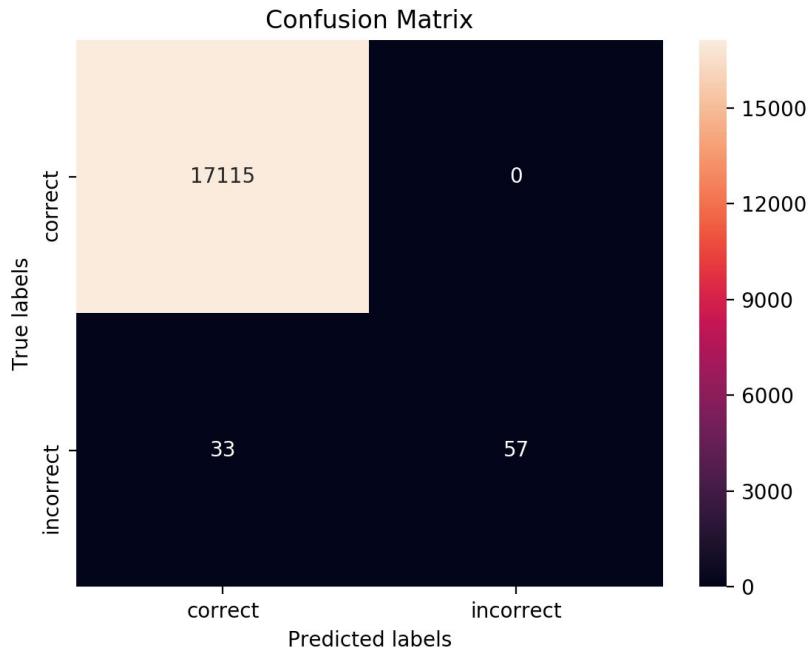
5. Temporal resilience accuracy

Accuracy =  $(TP+TN)/(TP+TN+FP+FN)$

# PHISH DETECTION ACCURACY



# TARGET DETECTION RATIO

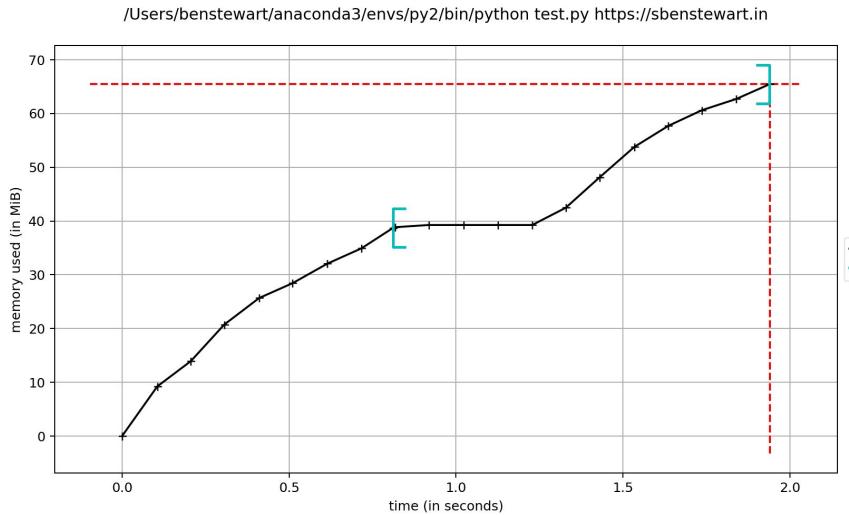


Det. Ratio 0.994  
F1 score 0.944  
Precision 0.955  
Recall 0.933

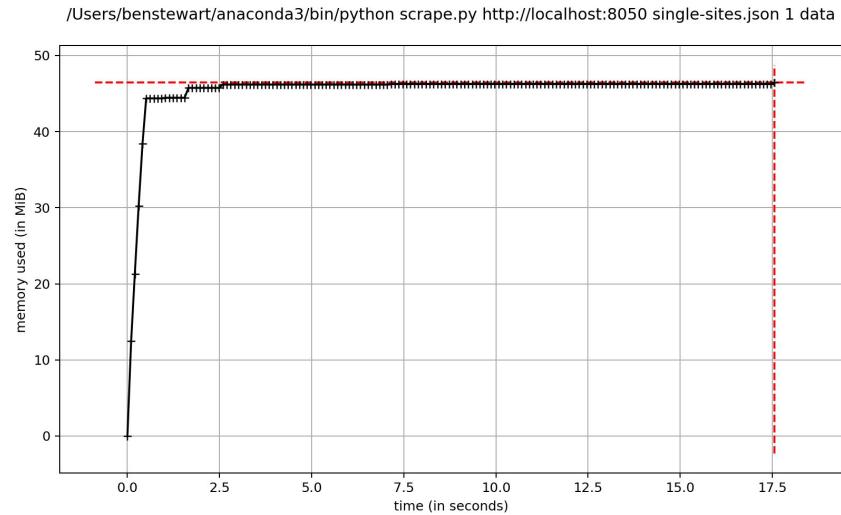
Computed for  
threshold=35

# MEMORY USAGE PROFILING

## PHISH DETECTOR 65 MB

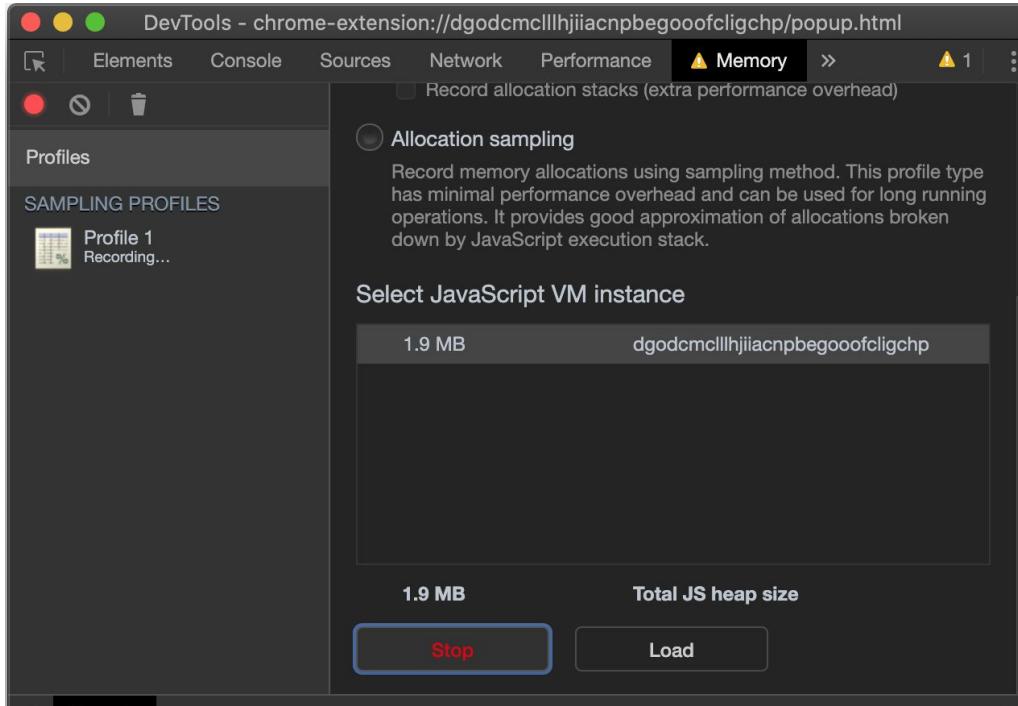


## TARGET IDENTIFIER 45 MB

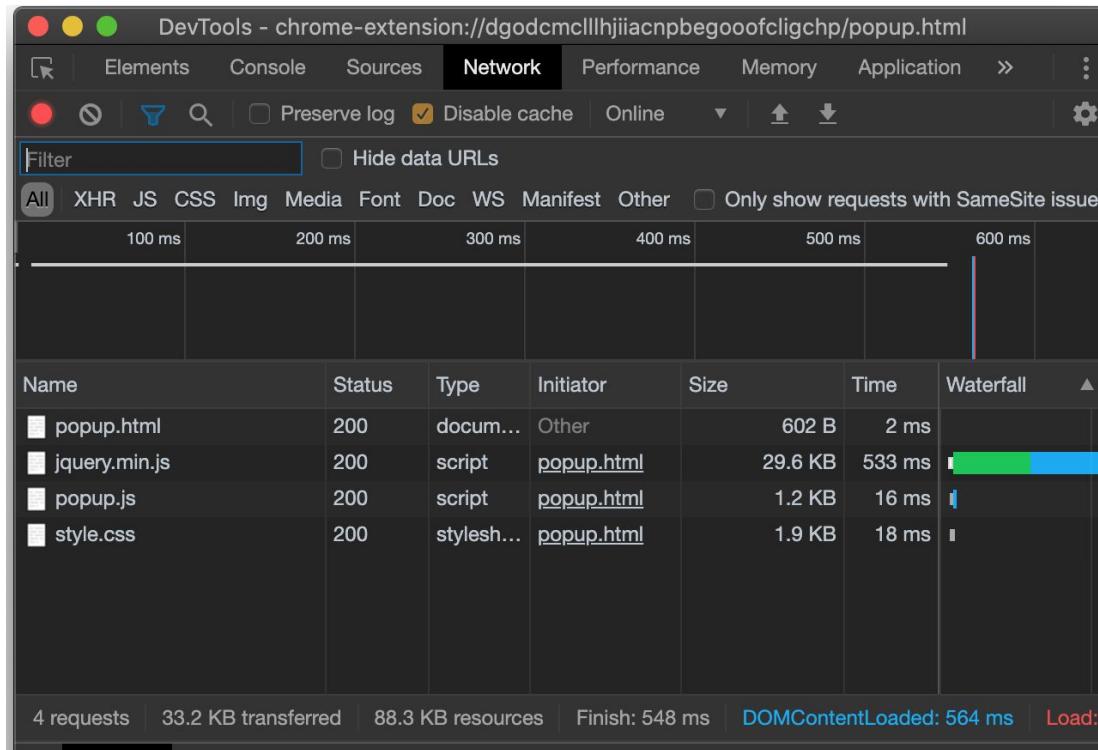


# MEMORY USAGE PROFILING

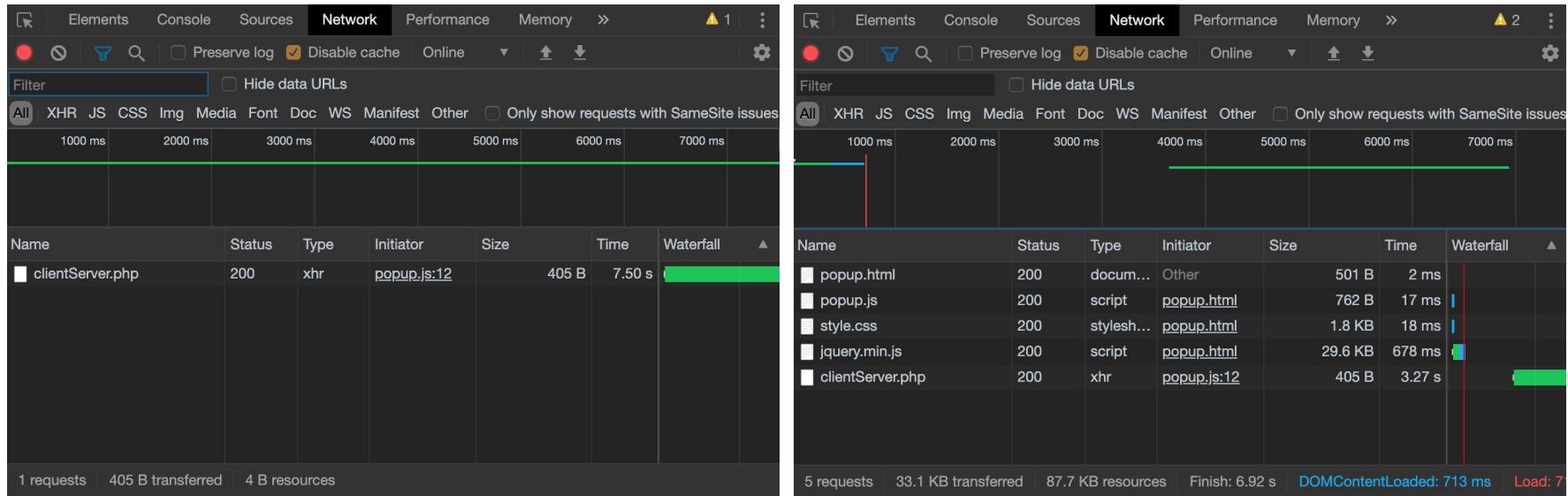
ADDON 1.9 MB



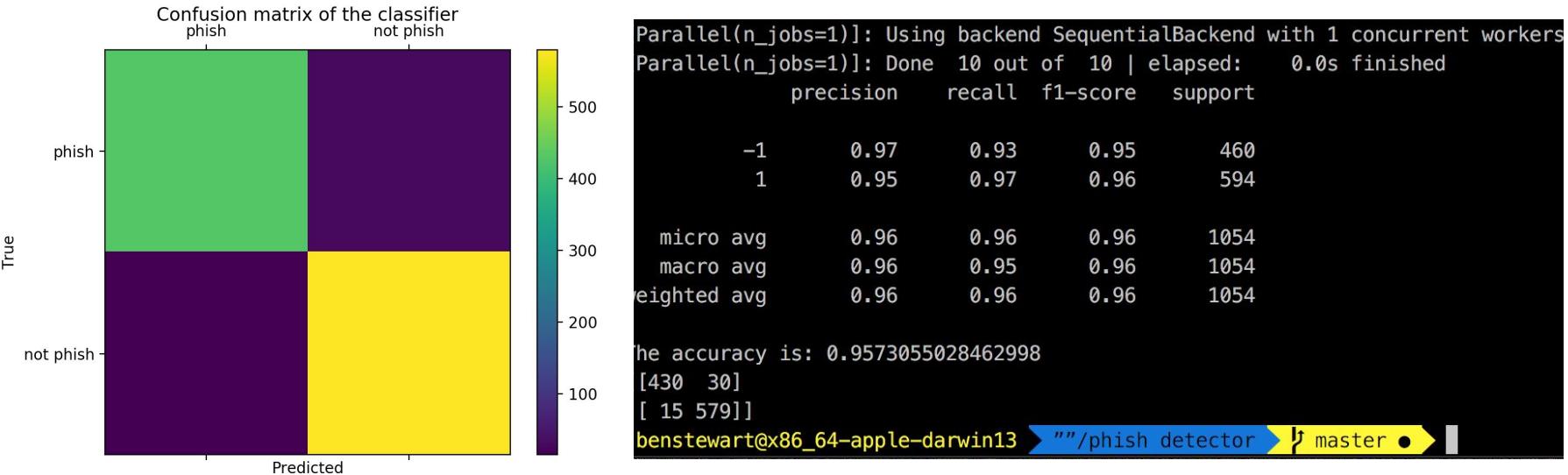
# ADDON RENDERING BASE TIME



# ADDON RENDERING TIME WITHOUT/WITH AUTO UPDATED WHITELIST



# TEMPORAL RESILIENCY ACCURACY



# TEST CASES

ID	MODULE NAME	ASSUMPTIONS	INPUT	EXPECTED OUTPUT
1_01	Background Script	The user loads a web page using the browser	The URL to the web page	<ul style="list-style-type: none"><li>• Start a background listener</li><li>• Make the listener data available for the content script</li></ul>
2_01	Content Script	The webpage is fully loaded	The user clicks on the “Safe or not” button	<ul style="list-style-type: none"><li>• Get background script data</li><li>• Get the web page content</li><li>• Send to dispatcher</li></ul>
3_01	Dispatcher	The content script dispatches content	The content from the content script	<ul style="list-style-type: none"><li>• Search for URL in whitelist</li><li>• If yes, send safe message to the output UI</li></ul>
3_02	Dispatcher	The content script dispatches content	The content from the content script	<ul style="list-style-type: none"><li>• Search for URL in whitelist</li><li>• If no, send the content to the phish detector</li></ul>

# TEST CASES - Contd.

ID	MODULE NAME	ASSUMPTIONS	INPUT	EXPECTED OUTPUT
3_03	Dispatcher	The phish detector is sent the content	Phish output from phish detector	<ul style="list-style-type: none"><li>Send the content to the target identifier</li></ul>
3_04	Dispatcher	The phish detector is sent the content	Safe output from phish detector	<ul style="list-style-type: none"><li>Update the whitelist</li><li>Send safe message to the output UI</li></ul>
3_05	Dispatcher	The target identifier is sent the content	Target output from target identifier	<ul style="list-style-type: none"><li>Send warning message along with the target web page to the output UI</li></ul>

# TEST CASES - Contd.

ID	MODULE NAME	ASSUMPTIONS	INPUT	EXPECTED OUTPUT
4_01	Phish Detector	The dispatcher sends the content	The content from the dispatcher	<ul style="list-style-type: none"><li>• Run the model</li><li>• Publish the result to the dispatcher</li></ul>
5_01	Target Identifier	The dispatcher sends the content	The content from the dispatcher	<ul style="list-style-type: none"><li>• Run the model</li><li>• Publish the target web page to the dispatcher</li></ul>
6_01	Output UI	The Output UI gets the content from the dispatcher	Safe message from the dispatcher	<ul style="list-style-type: none"><li>• Set the safe message on the user interface</li></ul>
6_02	Output UI	The Output UI gets the content from the dispatcher	Warning message from the dispatcher	<ul style="list-style-type: none"><li>• Set the warning message on the user interface</li><li>• Display the target web page</li></ul>

# REFERENCES

1. Mahdieh Zabihimayvan and Derek Doran, "Fuzzy Rough Set Feature Selection to Enhance Phishing Attack Detection", IEEE International Conference on Fuzzy Systems, June 2019.
2. S. Marchal, G. Armano, T. Gröndahl, K. Saari, N. Singh, N. Asokan, "Off-the-hook: An efficient and usable client-side phishing prevention application", IEEE Trans. Comput., vol. 66, no. 10, pp. 1717-1733, Oct. 2017.
3. A. K. Jain, B. B. Gupta, "A novel approach to protect against phishing attacks at client side using auto-updated white-list", EURASIP J. Inf. Secur., vol. 2016, no. 1, Dec. 2016.
4. G. Xiang, J. Hong, C. P. Rosé, L. Cranor, "CANTINA: A feature-rich machine learning framework for detecting phishing Web sites", ACM Trans. Inf. Syst. Secur., vol. 14, no. 2, 2011.
5. Implementation for the Usage of Google Safe Browsing APIs (v4), 2019, [online] Available: <https://github.com/google/safebrowsing>.
6. C. Whittaker, B. Ryner, and M. Nazif, "Large-scale automatic classification of phishing pages," in Proc. Netw. Distrib. Syst. Security Symp., 2010, pp. 1–14.