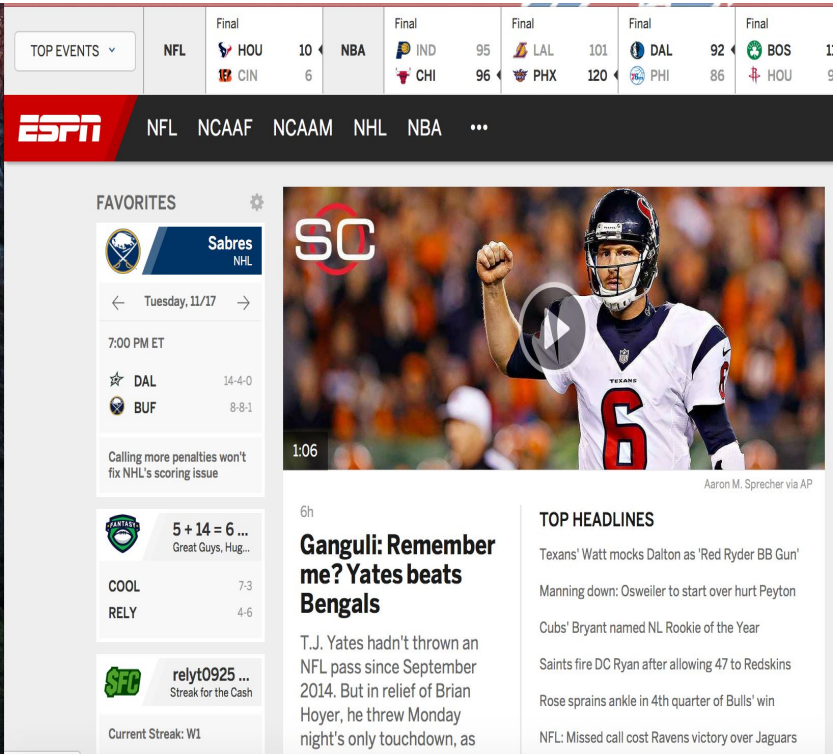
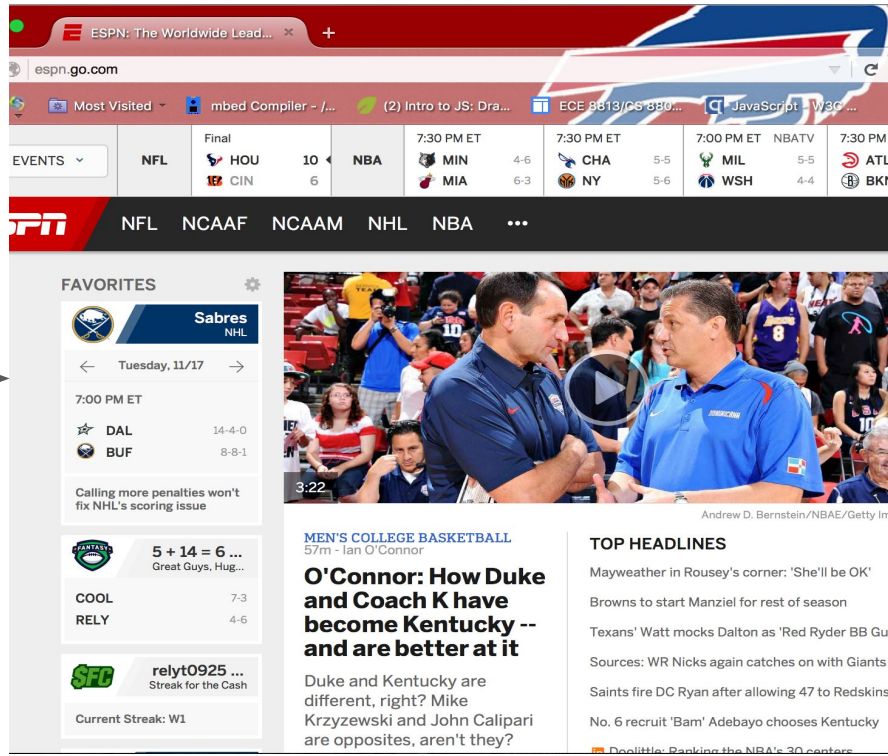
A Dynamic Internet: Categorizing Webpage Content Changes Over Time

Tyler Lisowski
Kim Yie
Yakshdeep Kaul
Judah Okeleye
Zongwan Cao

Web Content Is Constantly Changing



2 hours



Problem

Will this version of the webpage result in malicious activity?

- Web content changing over time
  - News- hourly
  - Banking website- weekly maybe monthly
  - Social media- minutes, possibly even seconds
- Change transitions state of web page
  - Benign site → Benign change → Benign site
  - Benign site → Malicious change → Malicious site
  - Malicious site → Benign change/cleanup → Benign site
  - Malicious site → Failed cleanup and/or changes → Malicious site

Focus on historical changes versus analyzing entire webpage!

Solution: Web Content Change Analysis

- Web Crawler
  - Gets web content from 1-5000 & 150000-155000 of Alexa Top 1 Million domains
  - Also gets IP Whois data from the resolved IP address for each domain
  - Crawls every 6 hours from October 28th to November 16th
- HTML Differ
  - Takes same webpage from 2 different time periods and finds changes
    - Additions, deletions, and modification of nodes
- Classifier
  - Trained on past benign and malicious changes
    - 650 benign changes
    - 74 malicious changes
  - Aims to classify a change as malicious or benign
- Envisioned Use Cases
  - Monitoring by web administrator
  - Client-side browser add on

Web Crawler

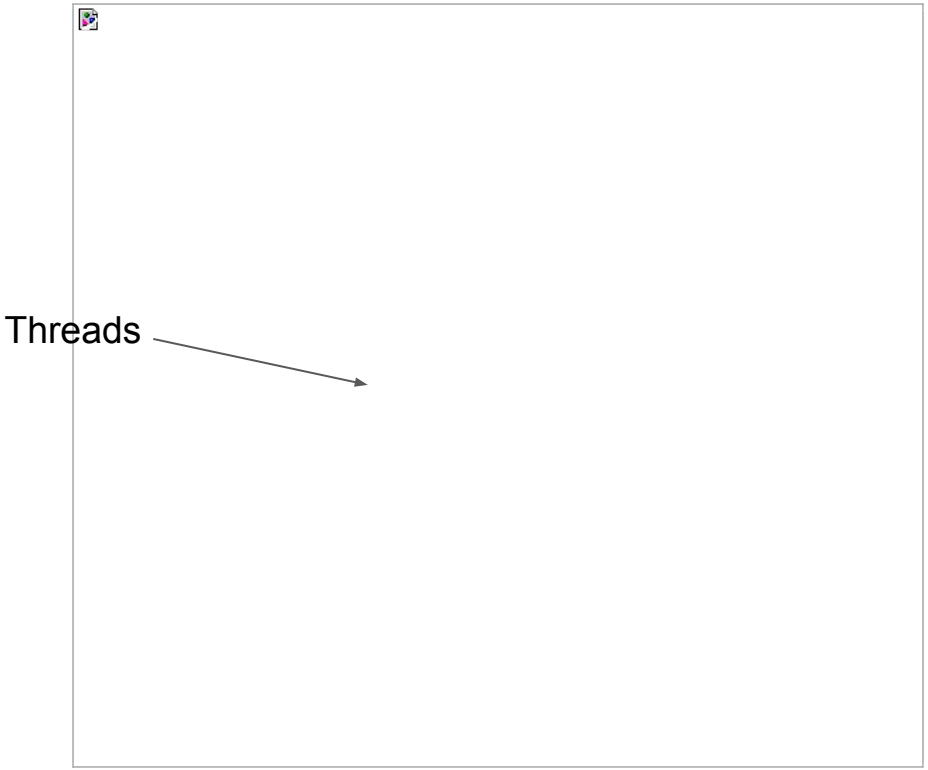


Figure 1: Web Crawler Implementation

1. Read Alexa Top 1 Million File

2. Check to see if active thread threshold reached
   a. If yes: wait till spot opens
   b. If no: spin off thread to fetch data for domain

3. At every checkpoint, serialize fetched data

- **Average page content fetching rate: 2500 pages/minute on VM**
  - Slower with IP Whois data fetching (100 pages/min)
  - Load Balancer + Hadoop can mitigate network bottleneck

Web Crawler Stats

- Memory= 900 MB per 10000 petched pages
  - Database Size of All fetched content: 63 GBs
  - Note some timeout errors AND serialization errors due to libxml
- Max Bandwidth Seen: 500kbps
  - Manually watched over successful fetching of 2557 webpages
  - Done with nethogs
  - Normal bandwidth appeared ~40 kbps
- Latency:
    - Done by calling time.time() before and time.time() right after request and taking difference
    - NOTE: Does include thread eviction time due to context switching
  - IP Whois Data
    - Average= 3.88 seconds, Max= 48.09 seconds
    - Over 3186 web pages fetched
  - Web Content Data
    - Average= 8.78 seconds, Max=155.56097 seconds
    - Over 3061 web pages fetched
- Rate
  - 133 pages/ minute
  - Measured from start of program to successfully fetching 8537 pages
    - Found this by checking serialized files (look at file count and then check amount of time PID has been running)

IP Whois Data


```
{"asn_registry": "arin",
"asn_date": "2007-03-13",
"asn_country_code": "US",
"resolved_IP": "74.125.21.113",
"raw": null,
 "asn_cidr": "74.125.21.0/24",
"raw_referral": null,
"query": "74.125.21.113",
"referral": null,
"nets": [{"updated": "2012-02-24T00:00:00", "handle":
"NET-74-125-0-0-1", "description": "Google Inc.", "tech_emails":
"arin-contact@google.com", "abuse_emails": "arin-contact@google.com",
"postal_code": "94043", "address": "1600 Amphitheatre Parkway",
"cidr": "74.125.0.0/16", "city": "Mountain View", "name": "GOOGLE",
"created": "2007-03-13T00:00:00", "country": "US", "state": "CA",
"range": "74.125.0.0 - 74.125.255.255", "misc_emails": null}],
"asn": "15169"}
```


Figure 2: IP Whois data for 74.125.21.113

Provides insight into changes of resolved IPs!

Google Safe Browsing Lookup API

- Google uncovers 9,500 new malicious websites everyday
- Classifies web pages into suspected
  - Malware category
  - Phishing category
  - Unwanted category
- Used as one of the features for classifier

Client's request URL:
https://sb-ssl.google.com/safebrowsing/api/lookup?client=demo&key=123&appver=1.0&pver=3.0&url=www.avtobanka.ru

Server's response body:
malware

HTML Differ Intuition

- HTML has tree structure
- Detect differences
  - Existing differs
    - Text based
    - No API
- Solution
  - Custom HTML tree diffing algorithm

HTML Differ

1. Key generation

```
<html>
        <body>
                <script>  ……. </script>
                <script>  ……. </script>
        </body>
</html>
```

Key for first script: [document]/!!/html/!!/body/!/**script**

Key for second script: [document]/!!/html/!!/body/!/**script_1**

HTML Differ

2. Find Similar Nodes

- Find all nodes in new file that has same path with original node A
- Pick up one, node B, with highest similarity score
- B is the same of A (score == 1)
  or is the modification of A (score < 1)

Original Version

```
<html>
    <body>
        <script>a</script>    .../!/script
        <script>b</script>    .../!/script_1
    </body>
</html>
```

New Version

```
<html>
    <body>
        <script>d</script>    .../!/script_2
        <script>a</script>    .../!/script
        <script>b</script>    .../!/script_1
    </body>
</html>
```

HTML Differ

Added Node

Original Version

```
<html>
    <body>
        <script>a</script>
        <script>b</script>
    </body>
</html>
```

New Version

```
<html>
    <body>
        <script>d</script>
        <script>a</script>
        <script>b</script>
    </body>
</html>
```

Modified Node

Original Version

```
<html>
    <body>
        <script>a</script>
    </body>
</html>
```

New Version

```
<html>
    <body>
        <script>e</script>
    </body>
</html>
```

Deleted Node

Original Version

```
<html>
    <body>
        <script>a</script>
        <script>b</script>
    </body>
</html>
```

New Version

```
<html>
    <body>
        <script>a</script>
    </body>
</html>
```

HTML Differ

4. Result output

- Modified node
  - Text change (only for leaf node) : <script> **TEXT** </script>
    - Insert
    - Delete
    - Replace
  - Attribute change: <script **src=”www.google.com”**> … </script>
    - Attribute value change
    - Attribute added
    - Attribute deleted
- Added node
- Deleted node

HTML Differ

Sample Output

```
{'afterText': 'cookies/src/jquery.cookie.js',
 'elementType': 'script',
 'fullText': '<del>/js/</del>cookies/src/jquery.cookie.js',
 'op': 'del',
 'otherInfo': 'ATTRIBUTE VALUE CHANGE',
 'rawChange': u'/js/'},
```

BEFORE:<script src="/js/cookies/src/jquery.cookie.js"></script>
AFTER: <script src= "cookies/src/jquery.cookie.js"></script>

```
['afterAttribute': {'src': 'www.google.com'},
 'afterText': None,
 'elementType': 'script',
 'fullText': '',
 'op': '',
 'otherInfo': 'ADDED NODE',
 'rawAttributeChange': {'src': 'www.google.com'},
 'rawTextChange': None}}
```

Add a new leaf node:
<script src= "www.google.com"></script>

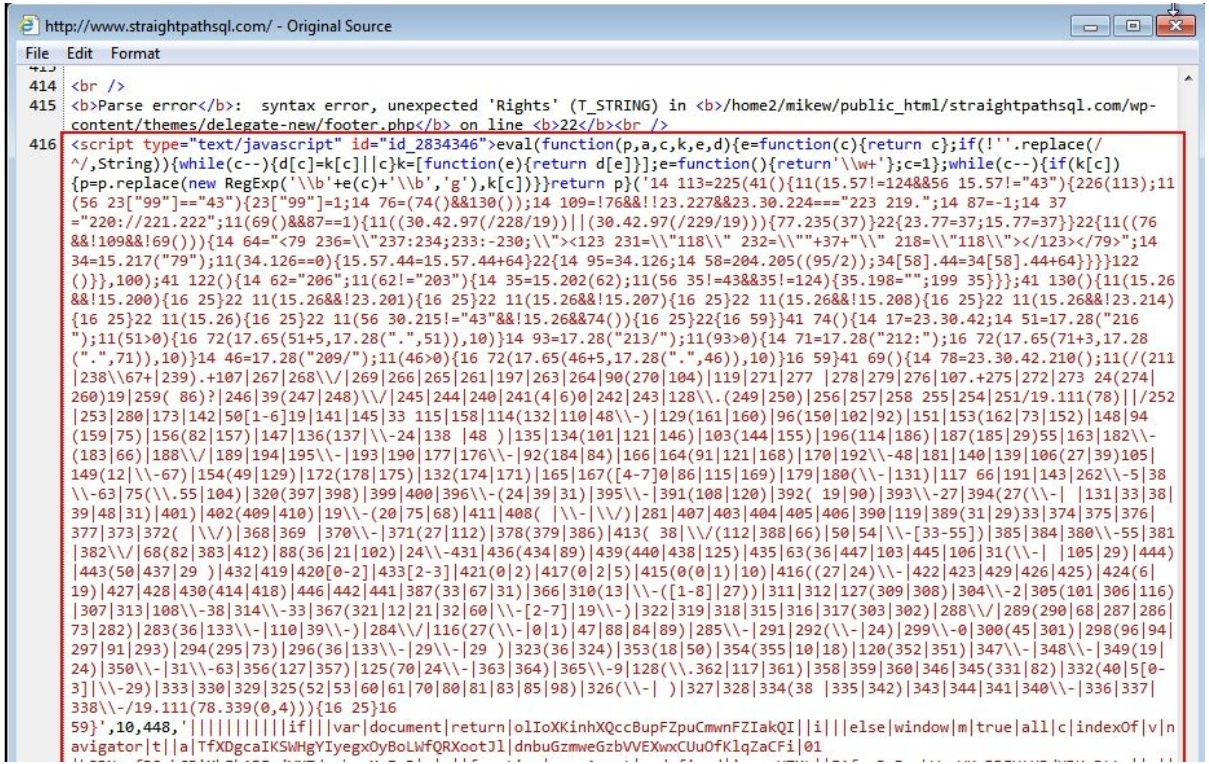
Machine Learning Classifier

- Intuition:
  - Use information from the diffing engine and static content analysis characteristics to train a model to determine whether a change to a website is malicious or benign

Malicious Ground Truth

- malware-traffic-analysis.net
  - Blog that flags malicious drive-by exploits
  - Create set of files that recreate the injection made by an attacker to the website

Malicious Ground Truth



Benign Ground Truth

- Use a news websites that are updated daily, with low chance of being compromised
  - cnn.com

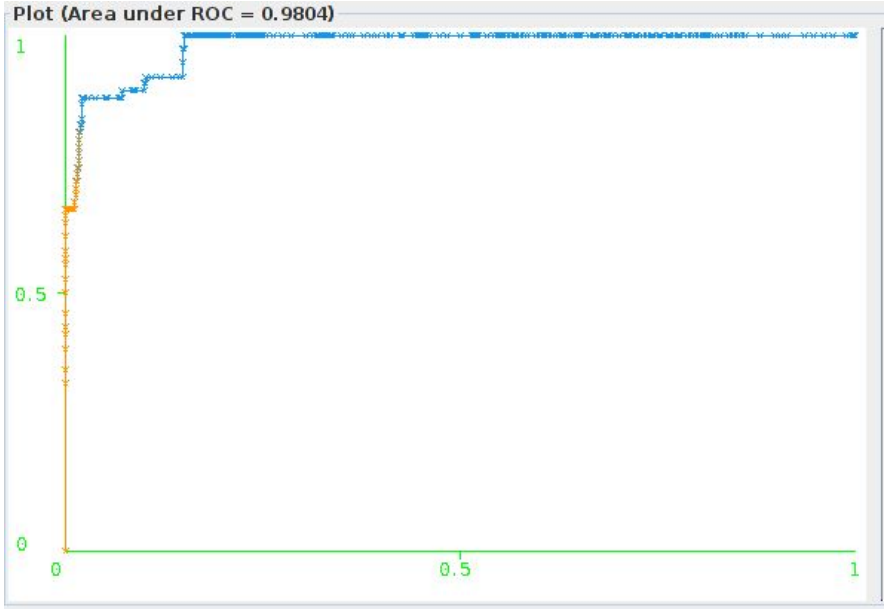
- Total Ground Truth collection:
  - 74 malicious changes
  - 650 benign changes

Feature Set

- elementType
  - tag of the HTML change
    - script, iframe, div, etc
- editType
  - Added or removed or modified
- scriptLen
  - malicious JavaScript can be up to kB in length
- specialCharRatio
  - malicious JavaScript can be obfuscated with many special characters
- GSB
  - is the src attribute of any node blacklisted by google?
- jsEval
  - count number of function calls used for dynamic unpacking of many JavaScript payloads

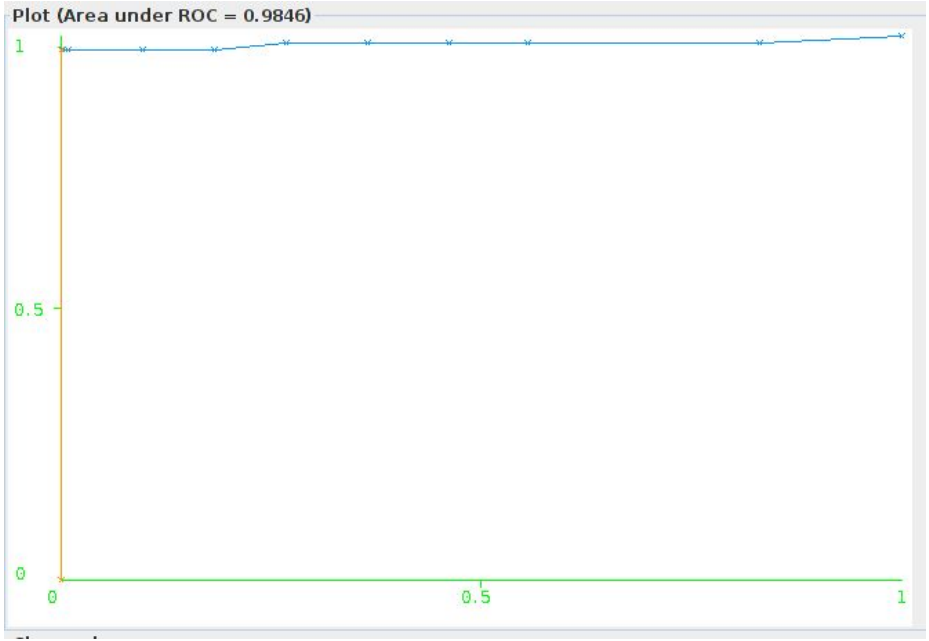
Results

- ROC curve of the Naive Bayes classifier
  - 81.1% TP, 1.8% FP with 10-fold cross validation



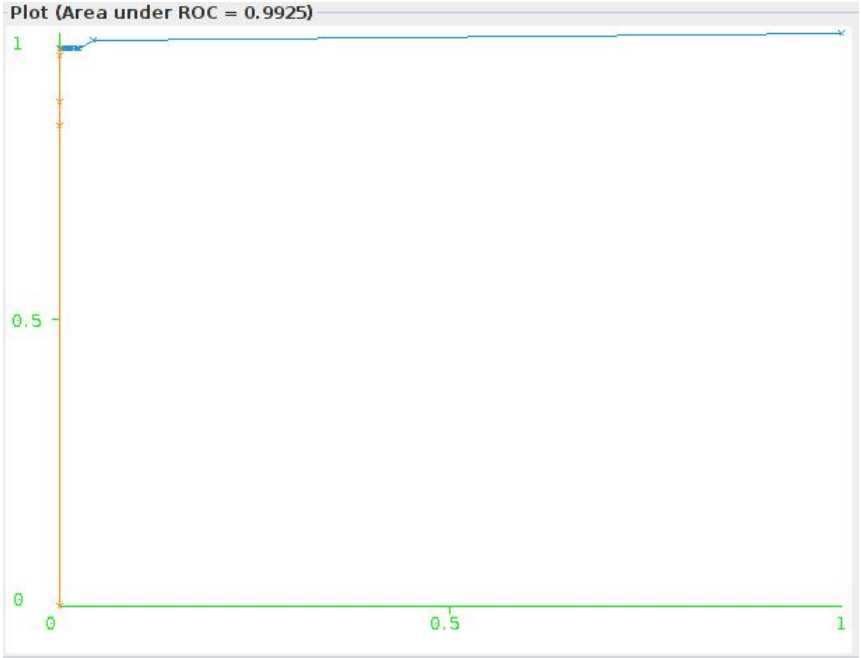
Results

- ROC curve of the J48 classifier
  - 97.3% TP, 0% FP with 10-fold cross validation



Results

- ROC curve of the Random Forest classifier
  - 97.3% TP, 0% FP with 10-fold cross validation



Feature Ranking

- Feature rank using the InfoGain attribute scores

Attribute Name	Ranked Value
editType	.3319
scriptLen	.3033
specialCharRatio	.2987
GSB	.2629
elementType	.2542
jsEval	.0399

Results

- J48 and Random Forest classifiers perform similarly with 10-fold cross validation
  - 97.3% TP Rate, 0% FP Rate
- Using Feature Ranking, we can try to reduce feature set by removing least scoring jsEval and elementType attributes
  - Classifier trained with jsEval removed performs at 97.3% TP, 0% FP
  - Classifier trained with jsEval and elementType removed performs at 87.8% TP, 0% FP
    - can remove jsEval from the full feature set with negligible performance hit

Evasion

- Algorithm, Dataset, and Features are public:
  - Attackers can avoid features we extract from the webpage
    - Lookup webpage in GSB to see if it's flagged (GSB feature)
    - Create shorter malicious payloads (scriptLen feature)
    - Use less special characters in their obfuscation (specialCharRatio feature)
  - Tradeoff:
    - A side-effect of simplifying malicious payloads may be that the malicious JavaScript is easier to be flagged by simple signature generation
  - Features that can not be evaded which are intrinsic to changes returned by the diff engine:
    - elementType
    - editType

Conclusion

- Developed a system that detects changes in webpages over a period of time
- Unique diffing algorithm
- Trained a classifier that performs with **high true positive rate (97.3%) & no false positives!**

Future Improvements

- Improve run-time performance of differ
- Gather malicious ground truth from multiple sources
  - Performance of the classifier seems ‘too good to be true’ for our basic feature set
  - malicious-traffic-analysis.net payloads have a lot of similar characteristics, need more diverse ground truth
- Include dynamic execution features into classifier
  - Run extracted javascript change in a JavaScript Engine, then create features based on executed or unpacked code