ParkedGuard: An Efficient and Accurate Parked Domain Detection System Using Graphical Locality Analysis and Coarse-To-Fine Strategy

Chia-Min Lai, Wan-Ching Lin, Hahn-Ming Lee, Ching-Hao Mao

Abstract—As world wild internet has non-stop developments, making profit by lending registered domain names emerges as a new business in recent years. Unfortunately, the larger the market scale of domain lending service becomes, the riskier that there exist malicious behaviors or malwares hiding behind parked domains will be. Also, previous work for differentiating parked domain suffers two main defects: 1) too much data-collecting effort and CPU latency needed for features engineering and 2) ineffectiveness when detecting parked domains containing external links that are usually abused by hackers, e.g., drive-by-download attack. Aiming for alleviating above defects without sacrificing practical usability, this paper proposes ParkedGuard as an efficient and accurate parked domain detector. Several scripting behavioral features were analyzed, while those with special statistical significance are adopted in ParkedGuard to make feature engineering much more cost-efficient. On the other hand, finding memberships between external links and parked domains was elaborately designed by leverage the graphical locality such that ParkedGuard outperforms the state-of-the-art in terms of both recall and precision rates.

Keywords—Coarse-to-fine strategy, domain parking service, graphical locality analysis, parked domain.

I. INTRODUCTION

Due to that a new type of advertising appearing on the web which is Pay-Per-Click (PPC) (also called as Cost-Per-Click (CPC)) in recent years, people, who own lots of registered domain name, realize that they could make better use of their domains rather than just waiting for someone buying them. Currently unused domain names could be lend to others who 'park'(put) their banners or PPC advertisings in these parked domains, such that the domain owners will get some remuneration as pay back. Therefore, the intermediate service that links domain owner and advisement syndicator helping them finding each other as well as setting up corresponding websites is called domain parking service.

However, when business scale of domain parking service grows larger and larger, it is unavoidable that some cyber-security-related issues will accompany as side effects, such as parked domain monetization, abuses, and illicit activities [1]-[3], starting to influence internet users and causing lots of damage. Researchers were aware and put effort to detect parked domains to avoid users exposed to the threatens in the realm of domain parking services [4]-[7]. Among all the related works, for the purpose of early high-throughput screening, techniques tring to detect parked domain among millions of domain names using machine learning approaches [4] are the most major trend. Yet, to our best knowledge so far, the state-of-the-art detection method, Parking Sensors [4], suffers two primary defects. One is lacking efficiency as features generating in terms of CPU time to calculate and efforts to collect essential information. Besides, parked domains containing external links usually cannot be effectively detected by the previous work, whereas this kind of “targeted parked domains” are most likely used to perform malicious activities, such as Drive-by Download attacks.

In this paper, to ameliorate the previous works defects without sacrificing practical usability, e.g., dropping the effectiveness, an efficient and accurate parked-domain detection system, named as ParkedGuard adopting coarse-to-fine strategy [8] was proposed as followings. At the first stage, to alleviate long latency of the aforementioned work, the proposed ParkedGuard considers several script behavioral indicators that describes the characteristics or the working semantics of domain web pages as features for parked domain prediction. Then, a statistical feature selection mechanism determines the most differentiable features according to their uni-variate significance. By means of significant feature subset selection from original feature set, ParkedGuard successfully reduces the CPU latency for calculating the values of indicators used in parked domain prediction, as well as saves the efforts to collect essential information generating those inputs for feature engineering. These low-cost but high-descriptive feature subset was then used by random forest decision to pre-label our domain set, which is early-stage decided as parked or non-parked domain candidates. On the other hand, the fine-tune part of proposed strategy is designed for not only compensating the resulted accuracy decreasing due to only a subset of original features was picked up, but also improving the prediction ability for important targeted parked domains. In the fine-tune process, one novel relational graph correlating both pre-labeled domain candidates and external links were constructed. And locality of each pre-labeled candidates was statistically analyzed and...
leverage to re-check the predictions of other candidates around its neighborhood using nearest-neighbor based voting mechanism. In the final stage, the proposed ParkedGuard using only a few computational-efficient features produces significantly improved predictions to targeted parked domains, and comparable results for general cases, respectively.

Real-data evaluations show that 1) Comparing to using full features, the computational time needed for feature extracting decreases dramatically to only half of that used to be. In addition, efforts to collect essential information for generating full features could also be saved. 2) the proposed ParkedGuard considering graphical relationships between domains and their external links as well as locality among its neighborhood, successfully outputs precise predictions and especially emphasizes on targeted parked domains. The numerical performance matrices of ParkedGuard include accuracy, recall, precision, and f-measure are 90.4%, 93.1%, 85.4%, and 89.1%, respectively, which all outperform those metrics of previous work. Some real case studies demonstrated that ParkedGuard successfully identified several targeted parked domains whereas previous work failed to differentiate those targets from non-parked ones. As a result, when the scale of domains being checked becomes larger, our system not only maintains the low computation cost but also keeps the effectiveness about detecting parked domain.

II. RELATED WORKS

A. Domain Parking Service

Domain parking services refer to a registered domain name without being associated with any services such as a website or a mail server. The domain owners then temporarily run it as an advertisement web portal to make a profit from the traffic the domain receives. In order to achieve this aim, the domain owner typically chooses to park the domain with a domain parking service, an intermediary between the owner and various monetization options. In a domain parking service, there are four important roles within it, the domain owner, the service provider, the advertisement syndicators, and the advertisers. The architecture of a domain parking service showed in Fig. 1.

Domain owners usually register an account with a domain parking service and park their domain on this service. After the domain being parked, parking service providers supply a site for domain owners to manage their parked domains [9], and they also cooperate with advertisement syndicators to provide advertisements on the parked page. The advertisement syndicators provide a platform for service providers and advertisers. They usually provide Javascript code to automatically generate advertisements on the parked page. So, the parked page will have advertisements to be clicked. However, this platform seldom detects the advertisement’s content. Therefore, some malicious advertisers use this way to provide malicious advertisements on the parked page. This way might influence the users in the domain parking services.

B. Parked Domain Monetization Options

A domain parking service provides many monetization options to domain owners. The most popular ones are search advertising, direct-navigation [10] monetization (Pay-Per-Redirect). Search advertising possibly made through Pay-Per-Click (PPC) [11]-[13]. A parking service provider submits a search query for certain keywords (relevant to the domain names, in the case of parked domains) and receives relevant ads in the XML format, which also include the price per ad from the advertisers. The service providers then display a set of ads on the parked page. Once a user clicks on an ad, the click traffic is bounced through a number of hosts such as click servers before reaching the parked page. This click is paid for by the advertiser and the revenue generated in this way is shared between the domain owners, the service providers, and the ad networks. Another monetization method is direct navigation traffic, which is generated when the web user enters a domain name as a query and expects to be redirected to a related domain. For example, one may type in “findcheaphotels.com” in the address bar and land to mytravelguide.com. This is caused by a direct-navigation-traffic purchase that the owner of mytravelguide.com purchases through a direct navigation system the traffic related to keywords “travel” and/or “hotels.” Parked domains can serve such a direct navigation system by redirecting type-in traffic to traffic buyers like mytravelguide.com. This monetization option is called Pay-Per-Redirect (PPR) or zero-click [14], [15].

C. Illicit Activities in Domain Parking Services

Illicit activities in domain parking service can be discussed into two part. First part is the illicit monetization of parking service, such as click fraud, traffic spam, traffic stealing, Malware distribution, etc. Click fraud is a type of fraud that occurs on the Internet in pay-per-click (PPC) online advertising [16]-[18]. Fraud occurs when a person, automated script or computer program imitates a legitimate user of a web browser, clicking on the ads in order to earn money without having the actual interest in the target of the ad’s link [19], [20]. A parking service may collaborate with traffic monetization platforms, which monetize different types of traffic such as parking traffic, error traffic (i.e. 404 not found
Digital Open Science Index, Computer and Information Engineering Vol:11, No:6, 2017

proposed methods to evaluate whether the blacklists have the

Another part of illicit activities is the abuse of parking

D. Parked Domain Analysis/Detection and Its Current

In 2010, Almishari et al. [28] developed a classifier

III. METHOD AND SYSTEM ARCHITECTURE

A. Scripting Behavior Features Extracting

The efficiency about calculating the high latency features:
- This experiment analyzed the efficiency about calculating
  the high latency features. It took off the high latency
  features from the Parking Sensors and recorded the features extracting time with high latency features and
  without high latency features. The result has shown in
  Fig. 3 that the calculation time about features decreases
  to half of the time when using the $n = 300$ samples.
  With the number of $n$ being larger, we can save more
time in feature extraction.
The effectiveness about the high latency features: This experiment which is designed to detect parking domains with all features, Parking Sensor provide, and with just low latency features verified whether the high latency features are important in the Parking Sensors. It ran in Orange3 which is a component-based data mining software [31], then executed with 300 samples implemented with 10 cross-validation. The result is in Table I. It showed that the precision and recall between the feature set without frame features and the feature set with all features were very nearly.

After analyzing the sample set for the feature extraction time but also maintain the effectiveness of detection. It came into Parked domain scripting behavior feature extracting module for selecting features which have the characteristics of domain web page as we shown in Fig. 4. The input is a domain list, and the output is a set of features. First, high latency features were excluded. Second, it executed with 2000 samples implemented 10 cross-validation in the experiment, and randomly select 50 feature sets from 50 samples. Last, as the results of each feature were not a normal distribution, Kruskal-Wallis test [32] is considered as the method of our feature selection. The Kruskal-Wallis test is a non-parametric method for testing whether samples originate from the same distribution. The results of Kuskal-Wallis test is presented in Table II. 8 features were selected from 22 features, Parking Sensors proposed by judging with Kuskal-Wallis test where their p-value are smaller than 0.05.

The selected features were average source length, external source ratio, link-to-global text ratio, maximum link length, amount of meta refreshes, amount of non-link characters, text-to-HTML ratio and amount of window location. Detailed explanations list in the following paragraphs:

**TABLE I**

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Kruskal-Wallis Statistic</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amount of Meta Refreshes</td>
<td>21.711</td>
<td>0</td>
</tr>
<tr>
<td>Link-to-Global Text Ratio</td>
<td>11.449</td>
<td>0.001</td>
</tr>
<tr>
<td>Amount of Non-Link Characters</td>
<td>10.905</td>
<td>0.001</td>
</tr>
<tr>
<td>Amount of Window Location</td>
<td>9.646</td>
<td>0.002</td>
</tr>
<tr>
<td>Average Source Length</td>
<td>9.041</td>
<td>0.003</td>
</tr>
<tr>
<td>Text-to-HTML Ratio</td>
<td>9.139</td>
<td>0.003</td>
</tr>
<tr>
<td>External Source Ratio</td>
<td>8.004</td>
<td>0.005</td>
</tr>
<tr>
<td>Maximum Link Length</td>
<td>6.271</td>
<td>0.012</td>
</tr>
</tbody>
</table>

**TABLE II**

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Kruskal-Wallis Statistic</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>85.7%</td>
<td>80.6%</td>
</tr>
<tr>
<td>Recall</td>
<td>83.0%</td>
<td>84.8%</td>
</tr>
</tbody>
</table>
• **Amount of Meta Refreshes**
Parked domains usually use redirection mechanisms to lead visitors to other pages or domains. Meta Refreshes is one of a redirection mechanisms in HTML. Although non-parked domains might have redirection mechanisms, it still can assist the classification when combining other features. It can be calculated by looking for \texttt{http-equiv="refresh"} in the HTML files to extract this feature.

• **Amount of Window Location**
Window location is another redirection mechanism of JavaScript redirection code. It can be calculated by searching for \texttt{window.location} in the HTML file to calculate the amount of the presence words.

• **Link-to-Global Text Ratio**
Many parked domains have less text that is not the part of the links. On a typical parked page, text is either part of an ad or part of the "Related Links." To this reason, this feature extracts all text from the HTML pages with Python’s Natural Language Toolkit [33], which return the text without HTML tags. Then, it calculated the ratio of the amount of text that in the links (i.e. \texttt{<a>} element and its child nodes) and the global amount of text display on the page.

• **Amount of Non-Link Characters**
The feature counts the actual amount of characters not belonging to any link element, instead of exclusively relying on the ratio.

• **Maximum Link Length**
Owing to the advertisement links, which are the major component on parked domain web pages, pass more and longer parameters along with the link in order to track the click on the PPC ad. This feature count the number of \texttt{<a>} elements display on the page and measure the string length of the destination addresses. Using these numbers, we can calculate the maximum link length of the page.

• **Average Source Length**
Similar to the previous feature, source addresses for banners and other advertisement media, tend to pass parameters of campaigns, image dimensions, etc. The non-parked web pages were expected to have more static media sources and thus shorter address lengths.

• **Text-to-HTML Ratio**
It can be measured by the ratio of text to the total amount of characters in the HTML file. This feature focuses more on the dynamic generation of content.

• **External Source Ratio**
An external source can be defined as one with an address pointing to another domain. Links and media generated by third-party advertisement syndicators will usually display on domains of that syndicator. The non-parked web pages were expected to have a lower ratio of external links because they commonly also have links to pages and media hosted on the same domain.

---

**B. Parked Domain Candidate Labeling**

In this module, it take the output of previous module, a set of feature vector, as input shown in Fig. 5. Due to aiming for high interpretability of our learning algorithm, it is important to comprehend the prediction of learning algorithm of the module for further improvement and adaptability. Therefore, Random forest algorithm was implemented, as it combines the strength of the ensemble learning with the interpretable qualities of decision trees. Moreover, it also improves the decision tree that tends to be robust with regard to outliers, the ensemble method of Random Forest protects the model against overfitting. Besides, after the tree have been constructed in the learning phase, the classification is usually very quick in the predicting phase. After the ensemble learning is done, we put the candidate domain in the decision tree to help us label it. Finally, we get the candidate domain that can be pre-labeled as parked or non-parked.

After the ensemble learning is done, we put the candidate domain in the decision tree to help us label it. Finally, we get the candidate domain that can be pre-labeled as parked or non-parked.

**C. Domain URL Relation Graph Generation**
The input in this module is pre-labeled candidate domain list and the output is a domain URL relation graph as shown in Fig. 6. To compensate for efficiency of computational time trade-off, ParkedGuard adopt graphical locality analysis, constructing the domain URL relation graph for each candidate domain, as shown in Fig. 7. A domain URL relation graph is defined as below:

![Image](image-url)

Let \( D \) is a set of candidate domain shown in (1) and \( E \) is a set of external URL links of \( D_m \) shown in (2),

\[
D = \{ D_m \mid D_m \text{ is a candidate domain} \} \quad (1)
\]

\[
L_m = \{ L^n_m \mid L^n_m \text{ is an external URL link of } D_m \} \quad (2)
\]

Let \( G = (V, E) \) is a domain URL relation Graph where \( V \) is a set of vertices shown in (3) and \( E \) is a set of edges shown in (4),

\[
V = \{ D, L_m \} \quad (3)
\]
The external URL link is defined as a hyperlink that points at any second level domain other than the second level domain where the link exists, as shown in Fig. 8. Finally, we combine all domain URL relation graphs for each candidate domain \( D_m \) and then get the last graph showed in Fig. 7. This figure demonstrates that different domains might share the same external URL link. We use the output from the Parked Domain Candidate Labeling in Section III-B (i.e. pre-labeled candidate domain) to draw the graph in Fig. 10. The node of false positive is the targeted parked domain which without significant scripting behavior characteristics of the parked domain but still can be observed by their relationships with other domains. We use the relationships in the domain URL relation graph shown in the following to detect the parked domains, especially the targeted parked domains.

**D. Parked Domain URL Neighbors Detection**

In this module, the input is the domain URL relation graph which output from the Domain URL Relation Graph Generation in Section III-C as shown in Fig. 9. We proposed an algorithm called *Parked Domain URL Neighbors Detection* as shown in Algorithm 1. The algorithm uses the relationship between each \( D_m \) and \( L^n_m \) to label the parked domain as the following steps:

1) Labeling the isolated domain that without any domain neighbors by the output label in Section III-B.
2) Sorting each \( L^n_m \) by their degrees from high to low due to the concept that the high degree external URL link has more confidence in which its neighbors are the same class of domain. Each \( L^n_m \) would have a set of \( D_m \).
3) Sorting each \( D_m \) in the set of each \( L^n_m \) by the degree of \( D_m \) from low to high due to the previous concept that high degree \( L^n_m \) have the same class of domain. That is, if a \( D_m \) only has a neighbor that is a high confidence \( L^n_m \), then the \( D_m \) also be high confidence.

4) Using 2, 3 to traverse the domain URL relation graph, and calculating the two scores of each \( D_m \), parked domain score and non-parked domain score. In this part, we calculate the score of each \( D_m \)'s one stage neighbor \( L^n_m \). If \( L^n_m \)'s one stage neighbor exclude \( D_m \) is a parked domain, then add one point to the parked domain score.
On the contrary, if \( L_m \)'s one stage neighbor exclude \( D_m \) is a non-parked domain, then add one point to the non-parked domain score.

5) Labeling \( D_m \) by comparing the parked domain score and non-parked domain score which is higher.

**Algorithm 1 Parked Domain URL Neighbors Detection**

**Require:** \( G = (V,E) \)

\( V = \{ D, L_m \} \)

\( D = \{ D_m | D_m \) is a candidate domain \}

\( L_m = \{ L_m \mid L_m \) is an external URL link of \( D_m \} \)

**Ensure:** \( D = \{ D_m \mid (D_m = \text{Parked Domain}) \lor (D_m = \text{Non} – \text{Parked Domain}) \}\)

1: for all \( D_m \in G.node() \) do
2: if \( D_m \) is aPark Domain then
3: \( D_m = \text{label of } D_m \)
4: end if
5: end for
6: \( \text{Sorted}L = \) a list of \( L_m \) sorted by degree from high to low
7: for all \( i \) in \( \text{Sorted}L \) do
8: \( \text{Sorted}D = \) a list of \( i \)'s neighbor \( D_m \) sorted by degree from low to high
9: \( \text{Score}PD = 0 \)
10: \( \text{ScoreNPD} = 0 \)
11: for all \( j \) in \( \text{Sorted}D \) do
12: \( \text{OneStageNeighborOf}j = \) a list of \( j \)'s one stage neighbor
13: for all \( k \) in \( \text{OneStageNeighborOf}j \) do
14: \( \text{OneStageNeighborOf}k = \) a list of \( k \)'s one stage neighbor stage neighbor
15: for all \( l \) in \( \text{OneStageNeighborOf}k \) do
16: if \( l = \) Parked Domain then
17: \( \text{ScoreNPD} = \text{ScoreNPD} + 1 \)
18: else
19: \( \text{ScoreNPD} = \text{ScoreNPD} + 1 \)
20: end if
21: end for
22: if \( \text{ScorePD} > \text{ScoreNPD} \) then
23: \( D_m = \text{Parked Domain} \)
24: else
25: \( D_m = \text{Non} – \text{Parked Domain} \)
26: end if
27: end for
28: end for
29: end for

IV. Experiments and Results

The goal of experiments intends to measure the effectiveness of detecting parked domains against the abuse of the domain parking service ecosystem in real world. They are designed for 1) the efficiency of ParkedGuard which deliberately get rid of the high execution time features and 2) the effectiveness of ParkedGuard using graphic locality to compensate for the removed features.

A. Description of Dataset

The dataset of our experiment was collected by searching the records of the DNS Census dataset [34], which contains about 2.5 billion DNS records gathered in 2012 and 2013. All domains matched the DNS configurations listed in Table III which Thomas et al [4], collected are extracted then queried domain’s DNS records to confirm whether they were still parked with that particular parking service. There are totally 11,406,099 parked domains from 15 observed parking services and 223,705,447 normal domains. Since the DNS census is outdated, this indicates that at the time of this writing there exist at least 11 million domains whose is to serve ads while they are visited.

**Training and testing data.** We random select 2000 parked domain and 2000 non-parked domain as our training data. Next, we also random select 921 parked domain and 664 non-parked domain as our testing data. Table IV shows the description of our dataset.

B. Evaluation Metrics

ParkedGuard regards each domain as an instance and identifies parked domains as positive instances whereas non-parked domains as negative. Confusion matrix and its derivations provided a set of deliberately metrics in the following evaluation. \( TP \) (true positive) is the number of parked domains correctly detected; \( FN \) (false negative) is the number of parked domain misclassified as normal ones. \( TN \) (true negative) is the number of the non-parked domains that are correctly classified; \( FP \) (false positive) is the number of the non-parked domains that are wrongly classified as parked domains. Following description illustrates the meaning of derivations from confusion matrix.

- The **Accuracy** is defined by
  \[
  \text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (5)
  \]

- The **Recall (True Positive) rate** is defined by
  \[
  \text{Recall rate} = \frac{TP}{TP + FN} \quad (6)
  \]

- The **Precision** is defined by
  \[
  \text{Precision} = \frac{TP}{TP + FP} \quad (7)
  \]
Parking Sensor. The experiments in this section are about to ParkerGuard saves much more time to execute the detection Sensor. As the quantities of domains needed to detect increase, is time-saving detector in comparison with ParkerGuard process.

The proportion of parked domains and non-parked domains were put into from 300, 600, 900, 1200 and 1500 sequentially. The results shown in Fig. 11 express that the execution time of ParkerGuard was roughly half of the one of Parking Sensor proposed and appended graphic locality ParkerGuard is capable of saving. For ParkerGuard can detect the parked domain especially targeted cost down the time but also maintain the effectiveness about number of samples becomes larger, our system can not only detect parking domain.

Even taking graphic locality into account, the domain which is isolated and has no relationship with each other, like http://sexlinkje.com in Fig. 12, was misclassified as a normal one.

The targeted parked domain which without significant scripting behavior characteristics of parked domain could be observed by their relationships of links as ParkerGuard adopted the relationship between domain and external URL links to find the targeted parked domain as we shown in Fig. 12.

The evaluation results show that ParkerGuard can detect the parked domain especially targeted parked domain using the relationship between domain and its external URL links. The evaluation results show that ParkerGuard can detect the parked domain especially targeted parked domain using the relationship between domain and its external URL links. The evaluation results show that ParkerGuard can detect the parked domain especially targeted parked domain using the relationship between domain and its external URL links. The evaluation results show that ParkerGuard can detect the parked domain especially targeted parked domain using the relationship between domain and its external URL links. The evaluation results show that ParkerGuard can detect the parked domain especially targeted parked domain using the relationship between domain and its external URL links. The evaluation results show that ParkerGuard can detect the parked domain especially targeted parked domain using the relationship between domain and its external URL links. The evaluation results show that ParkerGuard can detect the parked domain especially targeted parked domain using the relationship between domain and its external URL links. The evaluation results show that ParkerGuard can detect the parked domain especially targeted parked domain using the relationship between domain and its external URL links. The evaluation results show that ParkerGuard can detect the parked domain especially targeted parked domain using the relationship between domain and its external URL links. The evaluation results show that ParkerGuard can detect the parked domain especially targeted parked domain using the relationship between domain and its external URL links. The evaluation results show that ParkerGuard can detect the parked domain especially targeted parked domain using the relationship between domain and its external URL links. The evaluation results show that ParkerGuard can detect the parked domain especially targeted parked domain using the relationship between domain and its external URL links. The evaluation results show that ParkerGuard can detect the parked domain especially targeted parked domain using the relationship between domain and its external URL links. The evaluation results show that ParkerGuard can detect the parked domain especially targeted parked domain using the relationship between domain and its external URL links. The evaluation results show that ParkerGuard can detect the parked domain especially targeted parked domain using the relationship between domain and its external URL links. The evaluation results show that ParkerGuard can detect the parked domain especially targeted parked domain using the relationship between domain and its external URL links. The evaluation results show that ParkerGuard can detect the parked domain especially targeted parked domain using the relationship between domain and its external URL links. The evaluation results show that ParkerGuard can detect the parked domain especially targeted parked domain using the relationship between domain and its external URL links. The evaluation results show that ParkerGuard can detect the parked domain especially targeted parked domain using the relationship between domain and its external URL links. The evaluation results show that ParkerGuard can detect the parked domain especially targeted parked domain using the relationship between domain and its external URL links. The evaluation results show that ParkerGuard can detect the parked domain especially targeted parked domain using the relationship between domain and its external URL links. The evaluation results show that ParkerGuard can detect the parked domain especially targeted parked domain using the relationship between domain and its external URL links. The evaluation results show that ParkerGuard can detect the parked domain especially targeted parked domain using the relationship between domain and its external URL links. The evaluation results show that ParkerGuard can detect the parked domain especially targeted parked domain using the relationship between domain and its external URL links. The evaluation results show that ParkerGuard can detect the parked domain especially targeted parked domain using the relationship between domain and its external URL links. The evaluation results show that ParkerGuard can detect the parked domain especially targeted parked domain using the relationship between domain and its external URL links. The evaluation results show that ParkerGuard can detect the parked domain especially targeted parked domain using the relationship between domain and its external URL links. The evaluation results show that ParkerGuard can detect the parked domain especially targeted parked domain using the relationship between domain and its external URL links. The evaluation results show that ParkerGuard can detect the parked domain especially targeted parked domain using the relationship between domain and its external URL links. The evaluation results show that ParkerGuard can detect the parked domain especially targeted parked domain using the relationship between domain and its external URL links. The evaluation results show that ParkerGuard can detect the parked domain especially targeted parked domain using the relationship between domain and its external URL links. The evaluation results show that ParkerGuard can detect the parked domain especially targeted parked domain using the relationship between domain and its external URL links. The evaluation results show that ParkerGuard can detect the parked domain especially targeted parked domain using the relationship between domain and its external URL links. The evaluation results show that ParkerGuard can detect the parked domain especially targeted parked domain using the relationship between domain and its external URL links. The evaluation results show that ParkerGuard can detect the parked domain especially targeted parked domain using the relationship between domain and its external URL links. The evaluation results show that ParkerGuard can detect the parked domain especially targeted parked domain using the relationship between domain and its external URL links. The evaluation results show that ParkerGuard can detect the parked domain especially targeted parked domain using the relationship between domain and its external URL links. The evaluation results show that ParkerGuard can ded...
Fig. 12 The diamond is the targeted parked domain which ParkedGuard can detect.

Fig. 13 The diamond is the domain http://sexlinkje.com. It is isolated and has no linkage with another domain.
approach to detect the parked domain, especially the "targeted parked domain" which without significant scripting behavior characteristics of parked domain but still can be observed by their relationships of links.

**ParkedGuard** has the following properties. Effectiveness: It is effective, that is, it is able to distinguish parked domain and non-parked domain in the network. Scalability: It achieves up to 90.4% percentage points in accuracy. It is scalable, that is, it is linear in the size of the problem (i.e., the number of domains in the input list). Efficiency: It does not need to generate too many features, especially the long latency features: frame features and have higher accuracy than the Parking Sensors that need to calculate more features.

Furthermore, **ParkedGuard** also makes the following contributions:

- Improving the high latency problem in calculating scripting behavior features.
- Proposing a Domain URL Relation Graph Generator to detect targeted parked domain.
- Proposing an algorithm called Parked Domain URL Neighbors Detection to detect parked domain, especially the targeted parked domain.
- Developing a system to detect the parked domain, especially the targeted parked domain.

Although this paper focused on the parked domain, especially the targeted parked domain. **ParkedGuard** combined the signature-based behavior features and the relationship between the domain and its external URL links in the relation graph. Because some domain has no relationship with others, it made graphic locality ineffective. Therefore, with more efficient perspective, the next step we need to survey is to define the isolated domain on the domain URL relation graph.

**ACKNOWLEDGMENT**

This research is supported in part by the Ministry of Science and Technology of Taiwan under grants number MOST-105-2221-E-011-085-MY3. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the contributions of this paper.

**REFERENCES**


Chia-Min Lai is a Ph.D. candidate at the Department of Computer Science and Information Engineering (CSIE) at National Taiwan University of Science and Technology (NTUST), Taipei, Taiwan. His advisor is Prof. Hahn-Ming Lee and he is a part of Intelligent System Laboratory. He is also a researcher at the Institute for Information Industry (III), Taipei, Taiwan. His research interests include network security and malware analysis.

Wan-Ching Lin received the MS degrees from the Department of Computer Science and Information Engineering (CSIE) at National Taiwan University of Science and Technology (NTUST), Taipei, Taiwan, in 2016. She is currently an engineer at the Institute for Information Industry (III), CyberTrust Technology Institute, Taipei, Taiwan. Her research interests is information security, cyber security, log analysis and text mining.

Hahn-Ming Lee received the Ph.D. degrees from the Department of Computer Science and Information Engineering, National Taiwan, Taipei, Taiwan, in 1991. He is currently a Distinguished Professor in the Department of Computer Science and Information Engineering (CSIE) at National Taiwan University of Science and Technology (NTUST), Taipei, and also a Research Fellow, Institute of Information Science (IIS), Academia Sinica, Taiwan. His research interests include Information Security, Web-Based Intelligent Systems, Cloud Computing, Artificial Intelligence, Science and Technology Policy. Dr. Lee is a Supervisor of Artificial Intelligence, Taiwanese Association (TAAI) and Chinese Cryptology and Information Security Association, and also a member of Director in National Information Infrastructure Enterprise Promotion Association (NII), Andes Technology Corporation, and President, Consumer Magazine Publication.