Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2020) 28: 1859 – 1870
© TÜBİTAK
doi:10.3906/elk-1908-150

ZEKI: unsupervised *z*ero-day *e*xploit *k*it *i*ntelligence

Emre SUREN*
Department of Information Systems, Informatics Institute, Middle East Technical University, Ankara, Turkey

Received: 27.08.2019 • **Accepted/Published Online:** 06.03.2020 • **Final Version:** 29.07.2020

Abstract: Over the last few years, exploit kits (EKs) have become the de facto medium for large-scale spread of malware. Drive-by download is the leading method that is widely used by EK flavors to exploit web-based client-side vulnerabilities. Their principal goal is to infect the victim's system with a malware. In addition, EK families evolve quickly, where they port zero-day exploits for brand new vulnerabilities that were never seen before and for which no patch exists. In this paper, we propose a novel approach for categorizing malware infection incidents conducted through EKs by leveraging the inherent "overall URL patterns" in the HTTP traffic chain. The proposed approach is based on the key finding that EKs infect victim systems using a specially designed chain, where EKs lead the web browser to download a malicious payload by issuing several HTTP requests to more than one malicious domain addresses. This practice in use enables the development of a system that is capable of clustering the responsible EK instances. The method has been evaluated with a popular and publicly available dataset that contains 240 different real-world infection cases involving over 2250 URLs, the incidents being linked with the 4 major EK flavors that occurred throughout the year 2016. The system achieves up to 93.7% clustering accuracy with the estimators experimented.

Key words: Exploit kit, web malware, drive-by download, URL analysis, unsupervised machine learning, cybercrime

1. Introduction

The past decade has witnessed the introduction of the *"exploit kits"* philosophy by the online criminal world in order to make new exploits easy to adapt for attacks as new vulnerabilities are found. Reportedly, an EK deployment does not require hacking expertise anymore. The adversary only needs to learn the infection business logic and the EK service handles all other technical details. The EK architectures have a standardized interface that makes application of attacks programmable, where the EK APIs incorporate multiple exploits and malware in their repository that are seamless to extend and configure. In recent years, the propagation of financial attacks on end users have mostly been caused by agile development of EKs, which has remained a relatively untouched area in academic works. To this end, investing on EK research has the potential to be beneficial for a broad audience which is the real motivation of why this research is conducted. The global proliferation of EKs and recent advances in EK development are serious problems and without awareness of the contemporary hacking techniques it is not feasible to detect the *zero-day* intrusions caused by these.

In our previous studies, we analyzed the network traffics of Exploit Kits where 4 key findings were discovered [1]: a) All EK families have a similar workflow for malware delivery, b) an EK infection chain contains 5 elements, c) although URLs seem to be randomized, there is a hidden "auto-URL-generation" logic which follows certain patterns, d) while the patterns of any single URL in EK infection chain is not insightful yet, the "overall URL patterns" expose the EK family. In this context; a novel and efficient technique was

*Correspondence: emre.suren@metu.edu.tr

developed [2] which 1) is lightweight by favoring only URL analysis rather than content inspection, 2) is quick by prefering machine learning over signature search, 3) achieves higher accuracy via supervised models. The presented study adds a fourth and last powerful capability to our novel categorization technique which is the detection of unknowns via unsupervised models.

Our original categorization technique, "overall URL patters" was designed to operate with supervised methods [2]. This approach requires labeled dataset where firstly we manually analyze samples and label them [1]. A while ago, we realized that, if we adapt an unsupervised mechanism to existing algorithm, we avoid a quite workload relating to labeling. In this work, we present our unsupervised approach. The advantage of this method is that learning the number of Exploit Kit families and their occurrences in the dataset without manual analysis assistance, in other words automatically estimating the number of clusters and their approximate size. Moreover, we can compare the similarity indexes of the new samples with the previous clusters and identify completely new clusters, namely new Exploit Kit families.

The main objective of this research is to categorize unfamiliar web-based client-side attacks, which are conducted through EKs, with a high accuracy and speed. We have developed unsupervised learning models to be able to mark completely new EK incidents as unknown for more elaborate manual technical analysis. Without relying on previous knowledge, the clustering models are built to group similar EK infections. Experiments with real-world incidents demonstrate that the proposed lightweight model is highly effective in categorizing EK families. The assessment shows that the stable clusterer, *ZEKI*, achieves 87.5% precision at minimum performing very fast. As a result, while the framework design of EKs fortify the malware distribution business and makes the Internet life harder, "ironically the advertised strengths are their actual weakness".

The contribution of the research is that gaining capability of recognizing even minor updates of EK families or brand new EK flavors in an automated fashion via novel *overall URL patterns technique* with unusual features operated on unsupervised machine learning algorithms. It is expected that the developed tool will provide fast zero-day EK intelligence with a high accuracy to help security analysts and will impact the business of the cyber criminals by early disclosure of the evolution in the ecosystem.

2. Related work

The first studies on EKs focused on analysis of the source code of EK families, in which researchers installed EKs from sources to their lab environment for inspection. Grier et al. [3] conducted a study where their dataset contained 77,000 malicious URLs taken from Google Safe Browsing and a blacklist provider. Over 10,000 unique executable files were delivered and dynamic analysis of those binaries led to 32 families of malware. Kotov and Massacci analyzed the source code of 30 (partly inactive) different EK types [4]. The preliminary analysis indicated that the major functionalities of EKs are managing exploits, evading detection mechanisms, and command and control. Allodi et al. performed experiments (MalwareLab) with the source code of 10 EKs [5]. They deployed EKs in a controlled sandbox environment and found that some EK frameworks support the latest exploits, where cyber criminals achieve a higher infection rate in a small amount of time at the expense of short appearance on the market. On the other hand, some EK families prefer to serve more stable exploits, where attackers get a lower but steadier infection pace over time. De Maio et al. executed an analysis, PExy, on the source code of over 50 EKs in 37 families [6]. They also worked with EKs in off-line mode in their laboratories and via automated static source code analysis, where they produced all combinations of HTTP request parameters (in particular URL and user-agents) that cause an EK to trigger an infection. In this way, they retrieved 279 exploit samples including variants.

The following list of studies involve machine learning or statistics to detect EK traffics and our study also focuses on EK families from this perspective. Eshete and Venkatakrishnan [7] analyzed samples of 38 EKs, WebWinnow, and identified content and structural features to model a set of classifiers. They locally installed EKs in a controlled setting and partly supported the dataset with 11 live EKs that were reported by the URL query[1] service.

Taylor et al. developed a method to categorize EK flavors by detecting structural patterns in HTTP traffic [8]. In the detection process, their model builds a candidate tree from the request-response pairs of new infections and finds similar EK products with the Weighted Jaccard Index. They built their own dataset by capturing 3800 hours of real-world traffic via a honeyclient, which includes 28 EK instances. Their approach reduced false positive rates by four orders of magnitude when compared to the state of the art. Stock et al. offered a prevention mechanism, *Kizzle*, which was specifically designed to identify four major EKs *(Angler, Rig, Nuclear, SweetOrange)* and produce signatures that can be applied to antivirus engines or plug-ins of a web browser [9]. The main objective is to auto-generate host-based structural signatures by the *DBSCAN* machine learning algorithm within hours for detecting the superficial but frequent changes. The evaluation showed that the false negative rates are under 5%, while false positive rates are under 0.03%.

Jayasinghe et al. [10] detected drive-by download attacks at runtime using lightweight dynamic analysis of the bytecode stream generated by a web browser during page content execution. They utilized *naive Bayes, support vector machines (SVM) and decision tree* as binary classifiers and *SVM* achieved the best score with almost 95% accuracy. Nappa et al. [11] identified drive-by download attacks by clustering exploit servers belonging to 2 different EKs based on 7 features related to the served exploits and distributed malware. Sood et al. [12] conducted a comparative study for 10 EKs and found 3 victim profiling methods, which were based on *user-agent* fingerprinting, HTML *document object model (DOM)*-based fingerprinting, and IP-based geolocation tagging. There were 4 JavaScript-based attack techniques for drive-by download, which were obfuscation, redirection, content injection on-the-fly, and domain address generation algorithm *(DGA)*. Takata et al. [13] proposed a method, *mine spider*, which analyzes JavaScript code relevant to browser fingerprinting and redirection functionality, then reveals URLs in the webpage by executing the extracted redirection code with the Rhino JavaScript interpreter. Aldwairi et al. [14] tested 23 machine learning classifiers using a dataset of 5435 webpages containing drive-by download attacks and based on the detection accuracy they selected the top five to build the detection model. They extracted 26 content features without executing the webpage and reduced the feature vector size to 15. The Bagged Trees binary classifier achieved the highest accuracy with 90%. Jagannatha [15] proposed a two-layer detection scheme for EKs and processed a *Bro-IDS* HTTP log of 1000 samples generated by a third party in 2012. *Naive Bayes* was applied for binary classification and then *k-means* was utilized for clustering EK families. The 36 features were reduced to 6 attributes and achieved 99% supervised and 75% unsupervised accuracy for 400 reserved samples. Sandnes [16] extracted the URL addresses from the output of an IDS for EK activity detection. The system can detect the sample as either benign or malicious rather than detecting the EK family. The *SVM, random forest, and naive Bayes* classifiers were utilized with 9 features, where the *random forest* model achieved the best accuracy with 97%.

In our first study, *Know Your EK* [1], the webpage contents of EK families were explored. In [1], a context-aware content analysis has been conducted on the current EK families where webpage content characteristics were revealed. Our second work, *I see EK (IsEK)* [2], focused on supervised models that

[1]urlquery.net

leverage URL components of EKs. In [2], supervised models leverage URL components of EKs where models achieved 98.9% classification accuracy. Our two works are similar to the latest three studies [7–9], which try to distinguish between EK types using HTTP traffic. The approach in *Kizzle* [9] is closer to our present study, where unsupervised methods are employed and the EK families appearing in their evaluation significantly overlap with our EK set. On the other hand, their feature set is only based on page content and they report that their clustering approach inherently requires large amounts of data. Some aspects of the *WebWinnow* [7], such as the use of URL features are also similar to our work. The honeyclient technology usage in *WebWinnow* breaks scalability. However, we base our methodology on lightweight analysis with machine learning and utilize simple mathematical calculations and avoid using regular expressions while extracting URL features. Moreover, our method relies on multi-family classification, which is more informative when compared to their favored binary classification. In a nutshell, the proposed technique performs faster and is scalable via customized machine learning algorithms and does not require massive data.

3. Methodology

The *context-aware content analysis* of EK cases stored in pcap files in [1] resulted in two key discoveries about EK characteristics: Firstly, all EK families have a similar workflow for malware delivery. More precisely, infections contain 5 elements that are *campaign, gate, landing page, exploit, and malware.* Secondly, each component in an infection chain follows particular templates. For instance, the length of URLs fall within specific boundaries, URLs contain a peculiar number of query keys, and their values have tailored formats.

The novelty in this study is leveraging the *overall URL patterns* embedded in HTTP interactions between EK servers and victim machines. Specifically, instead of analyzing each URL independently, the goal is to inspect all URLs, which are posted automatically after one click and without any user consent, together. The structures in the workflow allow to characterize EK flavors to a certain extent. After evaluating the statistical differences on the URLs of entire infection chains, we identified the *auto-URL-generation* logic and with the help of our novel technique, we were able to design distinguishing features that cover each EK family. Conclusively, the approach takes advantage of machine learning methods where unsupervised models are built for the discrimination of network traffics that belong to EK-based infections. In this sense, the proposed method differs from two similar studies: the system in [7] that combines both URL and content features with binary classification methods and the work of [9] that only analyzes the web contents individually.

As this work proposes a new technique to reveal *zero-day* EK families, an unsupervised machine learning method is taken on board for the solution. As the *zero-day* paradigm refers to a previously unknown fact by definition, we utilize an exploratory data analysis technique to get an intuition about the structure of the dataset at first. Clustering is defined as identifying certain subgroups in the dataset, where the samples in the same cluster are very similar and this approach is executed to group EK flavors based on URL features.

3.1. Data sources

Access to real-world EK data is usually restricted to government agencies and research institutions that do not allow public access. On the other hand, the advantages of using a community-driven data corpus over generating our own are that it enables proof of the study quality, provides acceptability by a larger audience, opens doors for future researchers to compare their own results, and offers high quality in the data utilized. However it is not easy to find such a priceless data for free. Fortunately, Bradly Duncan[2] is known as one of the top contributors of open-source EK research data which is the primary data source of this study.

[2]malware-traffic-analysis.net

An evaluation of the utilized machine learning algorithms is performed over this public data source, where all the incidents are recorded in network packet captures that have resulted in malware infection after an exploitation via an EK. This differs from the works of other researchers, who used datasets from anonymous sources that are not available to any entity. It is crucial that all the samples were generated during 2016, hence this study totally represents one year, which is also another exclusive aspect when compared to other researches. Since the EKs exhibit a significant evolution in a longer period of time, which makes detection difficult. The year 2016 is the richest year among the last three years in terms of data size and variety of EKs. The total number of incidents is 189 containing over 2250 URLs where 45 pcap files are broken for certain reasons, the remaining set contains 144 infections from Rig, RigV, Angler, and Neutrino EK families that correspond to 1456 URLs. Finally, the network traffics were sniffed while intentionally visiting the compromised webpage that causes malware infection through an EK at the end. The communication between the victim system and EK infrastructure is provided via real operating system and real browser personalities, contrary to the mentioned related work that usually relies on honeyclients. The details of the data source, challenges and how we process the dataset is presented extensively in our previous study *IsEK* [2].

3.2. Feature engineering

With respect to quantifying the patterns in URLs, firstly we measure the path length, count the path tokens, and calculate the maximum, minimum and average of those tokens. Basically, in this way, a 20-character path that has one token is discriminated from a 20-character path that has five tokens. Secondly, we apply the same logic to the query part, but the key-value pairs are computed separately. Likewise, in order to differentiate EKs more reliably, counting the particular special characters, dash and underscore, is also taken into account to recognize the minor changes of EK families.

Based on our understanding of *auto-URL-generation* logic, we design distinguishing features in two categories. **External URL patterns:** 1) An EK infection starts with a campaign page, where the URL address does not contain path or query parts. 2) Landing page, exploit and malware files are redirected from the same domain address which contains several key-value pairs in query part. 3) Command and control (C&C) activity is established via a third domain address that contains just a path in the URL without a query field. **Internal URL patterns:** 1) Landing page, exploit and malware URLs are relatively long. 2) There are particular number of key-value pairs in query segment. 3) The length of query keys are very similar and the length of query values are very diverse among different incidents. Then, we code discovered patterns into numerical values for machine learning. **External features:** Counts of URLs, unique domain addresses, path tokens, query key tokens, query value tokens, and special characters. Total lengths of path and query. **Internal features.** Minimum lengths of path, query key, and query value. Maximum lengths of path, query key, query value. Mean lengths of path, query key, and query value. Sum of lengths of path, query key, and query value.

3.3. Preprocessing features

In order to build accurate machine learning models, the raw dataset was purified, as in the first try, the algorithms could not perform well. It is considered that transforming actual values of features into an explicit representation could improve machine learning estimators. In this scope, four common scaling methods were evaluated, which are *maximum and minimum scaler, standard scaler, standard normalizer, and binarizer.* Experiments showed that the standard scaler performs best on the training dataset.

3.4. Models

Environment and instruments. Using the features extracted on the sanitized dataset, the *scikit-learn* machine learning API [17] is adopted to build clustering models. Several clustering algorithms have been experimented with, however some algorithms *(e.g., mean shift, spectral clustering, affinity propagation, birch etc.)* are not well-executed. In addition, as the *DBSCAN, OPTICS, and one-class SVM* algorithms were primarily developed for outlier detection and *feature agglomeration* is offered for merging features rather than samples, they are not taken into consideration. In this study, we keep our focus on EK detection rather than the individual successes of machine learning algorithms, as replacing machine learning algorithms is quite easier than designing a method for detection. Therefore, we have selected 2 algorithms known for their high performance in terms of accuracy and execution time at preelimination stage, which are *k-means and agglomerative clustering.*

K-means. Initially, the EK clustering problem is assumed as "*expectation-maximization*" where a centroid-based algorithm, *k-means*, is a good candidate for the solution. *K-means* assigns samples to a cluster, where the sum of the squared distances between the samples and the centroid is kept at the minimum. Less variation within clusters means they contain more homogeneous samples.

Agglomerative clustering. Secondly, hierarchical clustering builds nested clusters by merging or splitting them successively, where the hierarchy of clusters is represented as a tree. This does not require to specify the number of clusters and can determine the number from the dataset. It also allows to select what number of clusters provides the best fit for the data. Therefore, choosing *agglomerative* as the full unsupervised algorithm is a sensible option. The linkage criteria is the metric used for the merge strategy and the algorithm is not sensitive to the type of distance metric, where all work equally well whereas the choice of the distance metric is critical for other clustering algorithms.

Feature compression. The clustering algorithms are adversely affected from similar or worthless features, as we experienced in our experiments. A data dimensionality reduction technique, *principal component analysis (PCA)*, is utilized to decompose our features into a set of independent and uncorrelated components that explain a maximum amount of the variance. The 20-feature dataset is transformed into a compressed form, and an empirical evaluation shows that 5 dimensions together explain %85 of the variance. The contributions of each feature to each component is given in Table 1, where the absolute magnitude threshold was taken as 0.276 experimentally.

Table 1. The 20-feature dataset is transformed into 5 dimensions for increasing performance via PCA. The table shows which features are contributed to which principal components.

Component	**Most valuable features**
PC-1	PathLen, QryLen, CPTokens, PSum, QKeyMax, QKeyAvg, QValMax, QValAvg, QValSum
PC-2	CQKeyTokens, QKeySum, CQValTokens
PC-3	PMax, PAvg, CSpecChar, CUnqDomain
PC-4	CPTokens, PMin, PAvg, QKeySum, CURL, CUnqDomain
PC-5	PMax, PAvg, QKeyMin, QKeySum, QValMin, CUnqDomain

4. Evaluation

Unlike simply calculating the precision and recall of a supervised classification, evaluating the performance of a clustering algorithm is quite tricky. Several metrics are employed, which primarily measure the similarity of samples belonging to the same class or similarity of the true clustering and the predicted one. In clustering methods, the notion of similarity is perceived as the closeness of a sample to the centroid of the cluster. Hence, uniformity of the classes within the dataset could be evaluated according to a similarity measure *(e.g., euclidean distance, cosine distance, Manhattan distance or correlation-based distance)*. In principal, clustering is considered an unsupervised learning model contrary to supervised learning, since the ground truth is not available to compare the predictions to the true labels to evaluate its accuracy. For a scientific study, providing performance evaluations is inevitable to present the success of the work. Therefore, a labeled dataset is utilized to learn how clustering algorithms fit and a separate dataset is utilized to understand how they are used for prediction.

4.1. Performance results

The calculated metrics to highlight how well the model performs are explained in Table 2. *Adjusted random index* measures the similarity of two samples by ignoring permutations and with chance normalization. *Adjusted/normalized mutual information scores* measures the agreement of the two samples by ignoring permutations. For both metrics, the perfect index is 1.

By using conditional entropy analysis, the following 3 metrics are calculated, where the perfect score is 1. *Homogeneity* is the rate of each cluster containing only members of a single class. *Completeness* is the rate of all members of a certain group being predicted as in the same cluster. *V-measure* is the harmonic mean of *homogeneity and completeness* and is also equivalent to *normalized mutual information score*.

Fowlkes-mallows scores calculate the geometric mean of the pairwise precision and recall. Where *true positive* is the number of pair of samples that belong to the same clusters in both the true labels and the predicted labels, *false positive* is the number of pair of samples that belong to the same clusters in the true labels and not in the predicted labels, and *false negative* is the number of pair of samples that belong in the same clusters in the predicted labels and not in the true labels. A score closer to 1 indicates a good similarity.

Table 2. The quality of designed unsupervised two models are measured by 10 similarity metrics. The 8 metrics out of 10 promising significant results and relatively low scores from remaining silhouette and Calinski-Harabaz metrics are perceived that the dataset covering discrete samples.

Metric	K-means	Agglomerative
Adjusted random index	0.873	0.777
Adjusted mutual index	0.857	0.771
Normalized mutual index	0.865	0.786
Mutual index	1.166	1.052
Homogeneity	0.861	0.776
Completeness	0.869	0.795
V measure	0.865	0.786
Fowlkes-mallows	0.906	0.837
Silhouette	0.340	0.338
Calinski-Harabaz	65.49	63.08

When the ground truth labels are not available, the inputs and predicted labels are used to calculate some consistency metrics. *Silhouette score* is the mean distance between a sample and all other points in the same class, and the mean distance between a sample and all other points in the next nearest cluster together compose the silhouette value. It determines the degree of separation between clusters. When the coefficients are close to 1, the sample is far away from the other clusters. When the silhouette average score is greater than 0.5 and the size of clusters have higher than the average score, the number is interpreted as good. This metric does not produce promising results, it is usually interpreted as the dataset contains challenging samples and problem is not much straightforward.

Calinski-Harabasz index is the ratio of the between-clusters dispersion mean and the within-cluster dispersion.

Contingency matrix reports the intersection cardinality for every true and predicted cluster pair, where the samples are independent and identically distributed and one does not need to account for some instances not being clustered.

K-means handles the shape of the dataset smoothly and performs better with 94% average accuracy as shown in Table 3. In addition, *k-means* inherently forces a cluster to contain only closer samples, it causes far-away samples to be a new cluster. Therefore, it allows to expose completely new EK families.

On the other hand, our dataset has also a partially hierarchical structure. We understand this from the *agglomerative* performance, where it can recover this formation with 88% average accuracy as shown in Table 4, while most of the other clustering algorithms cannot achieve it. However, the accuracy is not as good as *k-means* and the execution time of the *agglomerative* is computationally expensive due to time complexity when providing such an advantage, unlike the linear complexity of *k-means*.

Table 3. Accuracy scores for k-means model

	precision	recall	f1-score
Angler	0.79	0.97	0.87
Neutrino	0.97	1.00	0.99
Rig	1.00	0.98	0.99
RigV	1.00	0.75	0.86
micro avg	0.94	0.94	0.94
macro avg	0.94	0.92	0.93
weighted avg	0.95	0.94	0.94

Table 4. Accuracy scores for agglomerative model

precision	recall	f1-score
0.68	0.81	0.74
0.85	1.00	0.92
1.00	0.98	0.99
1.00	0.61	0.76
0.88	0.88	0.88
0.88	0.85	0.85
0.89	0.88	0.87

4.2. Error analysis

The model based on *k-means* misclustered 9 samples shown in Table 5 where 7 RigV samples were predicted as Angler, 1 Rig sample anticipated as Angler, and 1 Angler sample called as Neutrino.

When the cluster sizes are not balanced, *k-means* performance worsens. By giving more weight to the larger clusters, it tries to prevent variance per cluster, which causes to allow samples being away from the centroid. In this case, smaller size clusters are embedded into bigger clusters and are totally lost. On the other hand, in order to create for the stated number of clusters, it partitions bigger clusters wrongly. Therefore, if the number of infection cases belonging to specific EKs increases in an unbalanced manner, the detection rate dramatically reduces both in terms of false positive and false negative. This possibility is always valid, since some EK families sweep competitors and become prominent. Identifying new small clusters and assigning more weight to them could be a solution, but that is not a trivial process. Secondly, *k-means* is sensitive to outliers and this domain naturally has outlier samples.

The model based on *agglomerative* misclustered 18 samples as shown in Table 6, where 11 RigV samples were predicted as Angler, 1 Rig sample anticipated as Angler, and 6 Angler samples called as Neutrino. Consistently, the error pairs are the same as those of *k-means*.

Table 5. Confusion matrix for k-means model

True Label	KMeans				
	Angler	30	1	0	0
	Neutrino	0	33	0	0
	Rig	1	0	51	0
	RigV	7	0	0	21
		Angler	Neutrino	Rig	Rigv
		Predicted Label			

Table 6. Confusion matrix for Agglomerative model

True Label	Agglomerative				
	Angler	25	6	0	0
	Neutrino	0	33	0	0
	Rig	1	0	51	0
	RigV	11	0	0	17
		Angler	Neutrino	Rig	Rigv
		Predicted Label			

4.3. Comparison

Although EKs have been researched in the past years, studies dedicated to EK detection are quite limited. Moreover, while our study utilizes machine learning for detection, other works mainly apply custom techniques. The results of the current analysis and the literature are compared in Table 7 to give an overall idea. WebWinnow [7] evaluated 5 binary classifiers and J48 performed better. When compared to our lightweight study, that method is quite time-consuming due to the examination of page contents rather than solely utilizing URL addresses. Taylor et al. [8] employed the weighted Jaccard index and also inspected both URLs and page content. While Kizzle [9] utilized DBSCAN for clustering web content, particularly JavaScript code blocks, the number of features is not a valid criterion for their model and only false negative rate (FNR) was reported. Jagannatha [15] only tried naive Bayes in combination with k-means and IsEK performs better in terms of accuracy. Sandnes [16] experimented with 3 classifiers and random forest achieved the best score, and the model was only able to detect samples being malicious or not. On the other hand, our proposed method discriminates particular EK families with a high accuracy.

Table 7. Comparison of the proposed model with the related work.

Study	Accuracy	Features	Algorithms
[7]	TPR: 99.9% FPR: 0.001% FNR: N/A	30 Page content	J48 decision tree
[8]	TPR: 95% FPR: N/A FNR: N/A	8 Page content and URL	Weighted Jaccard index
[9]	TPR: N/A FPR: 0.03 FNR: %5	Page content	DBSCAN
[15]	TPR: 75%-85% FPR: N/A FNR: N/A	6 URL	naive Bayes and k-means
[16]	TPR: 97% FPR: N/A FNR: N/A	9 URL	Random forest
[2]	TPR: 98.9% FPR: N/A FNR: N/A	20 URL	Gradient boosting classifier
Our new	TPR: 93.7% FPR: N/A FNR: N/A	20 URL	k-means

4.4. Discussion

Our previous study [2] presented a novel technique for EK detection and showed practical usage via supervised machine learning. However this approach requires significant need, in order to conduct such experiments, large amount of records in the dataset were labeled via manual analysis to build a baseline model. Back when, I understood that it is better to replace the application of such a time consuming and error-prone manual analysis process with an intelligent solution. Therefore, this present research aims to eliminate the heavy workload which is labeling operation. This new research presents the details of the implemented unsupervised mechanism and the experiments shows significant results.

The preferred usage scenario of the two methods is that unsupervised model is operated for labeling in an environment where the number of new data is not many i.e. a honeypot. In here, if the number of EK families is known before the analysis, k-means is a perfect solution since it performs better than agglomerative. On the other hand, if the analysis is not conducted in a controlled environment, it is not easy to know the number of clusters beforehand, in this situation the alternative is agglomerative which does not require any previous knowledge, but having lower success rate. We run supervised models in real environment i.e. a company network, where the number of new data is more, since it produces better accurate results. In here, we have two different options. If the labeled dataset via unsupervised models contains only known EK families, we can test our supervised models performance for comparison and identify the errors in both sides and improve both techniques. If the labels are belong to unknown EK families, the supervised methods are trained with the produced labeled data corpus and models learn new EK families. In this manner, the present research complement the second study rather than being full-fledged alternative.

There are some certain challenges while working with unsupervised learning methods. Some algorithms are semisupervised, requiring human feedback, where unsupervised methods assign random labels and start clustering randomly in each execution. The techniques to overcome these are explained in this part. The first challenge is that, some *unsupervised methods* cannot learn the number of clusters from the dataset *(e.g., k-means)* and require assistance. The best number of groups could be found by experimenting with the *elbow* method. The method determines the alternative cluster numbers based on the error sum of the squared distance (SSE) between samples and their assigned centroid (arithmetic mean of all the samples assigned to that cluster). The values when the SSE curve starts to forge an elbow are interpreted as promising cluster numbers. Our dataset is utilized to evaluate the SSE across different values of cluster numbers and the graph in Figure shows that candidates are *2, 3 and 4*. As a result, although *k-means* does not learn the number of clusters from the dataset, it exposes SSE values, which is usable to determine cluster number and has worked pretty well for our EK cases.

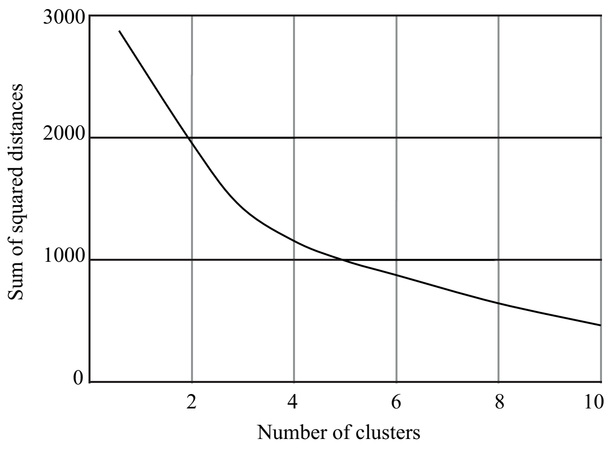

Figure . The k-means semisupervised method is not capable of estimating the number of clusters in the dataset. Elbow curve is utilized to assist k-means model to determine the best number of clusters. The Elbow line starts to bend at point 2 and ends with 4. For this dataset the middle number 3 has worked well.

Another challenge is that unsupervised methods use random labels, so we convert these notions carefully to actual labels. Clustering algorithms randomly start operation, where the fitted labels (clustering results) change in different executions of the algorithm. We forced the algorithms to process samples in a particular order while clustering to make experiments repeatable and consistent. Some algorithms do not allow such options and some custom ways are required to circumvent that, which is why we model 2 algorithms for this study.

5. Conclusion

The exploit kit phenomenon remains a serious threat for the web residents due to the fact that they are able to quickly adapt to changing conditions and further, turn them into an advantage. Whenever a vulnerability is disclosed publicly, EK owners develop corresponding exploits and integrate in their arsenal. Beyond that, they are frequently faster than the web users, who need to patch the application. Even worse, exceptional EK authors could also exploit vulnerabilities before vendors release a patch or discover zero-day vulnerabilities. Ultimately, EK products serve all those capabilities with a user-friendly interface for the threat actors. Overall, in this cat and mouse game, the threat actors will always have the advantage, since they make the opening gambit and the window of malware distribution is wide open until the campaign is revealed.

This research proposes a lightweight discrimination system for the network traffics of exploit kit families. By using *only* the URL characteristics of a complete infection chain, our novel *overall URL patterns* technique reasons about the likelihood of a sequence of HTTP interactions belonging to a specific EK. Our implementation is evaluated on a real-world dataset collected by a pioneer researcher on EK. In particular, our empirical results show that the unsupervised model *ZEKI* clusters a set of unknown EK-based infection traffics quickly between 87.5% – 93.7% precision. Although unsupervised methods provide high precision, the results could be confirmed with our supervised methods [2], since their results is better and nearly flawless. If there is too much deviation between supervised and unsupervised methods, manual analysis could be conducted to learn the root cause. In addition, both researches also complement each other seamlessly. It is conjectured that such an agile solution will help security analysts, who work with bulk data collected by honeypots, by providing early threat intelligence feed *(e.g., evolved attack techniques)*, discovery of *zero-day* attacks, creating obstacles for cyber criminals, and increasing the workload of EK engineers.

The major advantage of the EK infrastructures is its framework design which allows large scale malware propagation. However, our research revealed that this fabricated logic is also their weakness. Since *auto-URL-generation* logic follows templates which is a sitting target for models based on machine learning techniques. On the other hand, if EK authors agree on not using full URL addresses with patters, namely if they use only domain addresses, detection will be easier even for the traditional signature-based systems. As a result, some advantages and drawbacks makes it a trade-off. It is not easy to bypass our both unsupervised and supervised methods together, unless adversarial machine learning techniques are applied. Since next-generation prevention systems will likely rely on artificial intelligence, attacks that poison machine learning models are expected to be in the scene in the near future. Exploit kit for mobile and exploit kit for IoT are also expected to become more prevalent.

References

[1] Suren E, Angin P. Know your EK: A content and workflow analysis approach for exploit kits. Journal of Internet Services and Information Security 2019; 9 (1): 24-47.

[2] Suren E, Angin P, Baykal N. I see EK: A lightweight technique to reveal exploit kit family by overall URL patterns of infection chains. Turkish Journal of Electrical Engineering & Computer Sciences 2019; 27 (5): 3713-3728.

[3] Grier C, Pitsillidis A, Provos N, Rafique MZ, Rajab MA et al. Manufacturing compromise: The emergence of exploit-as-a-service. In: Proceedings of the 19th ACM Conference on Computer and Communications Security; Raleigh, NC, USA; 2012. pp. 821-832.

[4] Kotov V, Massacci F. Anatomy of exploit kits: preliminary analysis of exploit kits as software artefacts. In: Proceedings of the 5th International Symposium on Engineering Secure Software and Systems; Paris, France; 2013. pp. 181-196.

[5] Allodi L, Kotov V, Massacci F. MalwareLab: Experimentation with cybercrime attack tools. In: Proceedings of the 6th Usenix Workshop on Cyber Security Experimentation and Test; Washington, DC, USA; 2013. pp. 1-8.

[6] De Maio G, Kapravelos A, Shoshitaishvili Y, Kruegel C, Vigna G. PExy: The other side of exploit kits. In: Proceedings of the 11th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment; Egham, UK; 2014. pp. 132-151.

[7] Eshete B, Venkatakrishnan VN. WebWinnow: Leveraging exploit kit workflows to detect malicious urls. In: Proceedings of the 4th ACM Conference on Data and Application Security and Privacy; San Antonio, TX, USA; 2014. pp. 305-312.

[8] Taylor T, Hu X, Wang T, Jang J, Stoecklin MP et al. Detecting malicious exploit kits using tree-based similarity searches. In: Proceedings of the 6th ACM Conference on Data and Application Security and Privacy; San Antonio, TX, USA; 2016. pp. 255-266.

[9] Stock B, Livshits B, Zorn B. Kizzle: A signature compiler for detecting exploit kits. In: Proceedings of the 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks; Toulouse, France; 2013. pp. 455-466.

[10] Jayasinghe GK, Culpepper JS, Bertok P. Efficient and effective realtime prediction of drive-by download attacks. Journal of Network Computer Application 2014; 38: 135-49.

[11] Nappa A, Rafique MZ, Caballero J. The MALICIA dataset: Identification and analysis of drive-by download operations. International Journal of Information Security 2015; 14 (1): 15-33.

[12] Sood AK, Zeadally S. Drive-by download attacks: a comparative study. IT Professional 2016; 18 (5): 18-25.

[13] Takata Y, Akiyama M, Yagi T, Hariu T, Goto S. MineSpider: Extracting hidden URLs behind evasive drive-by download attacks. IEICE Transactions on Information and Systems 2016; 99 (4): 860-872.

[14] Aldwairi M, Hasan M, Balbahaith Z. Detection of drive-by download attacks using machine learning approach. International Journal of Information Security and Privacy 2017; 11 (4): 16-28.

[15] Jagannatha P. Detecting exploit kits using machine learning. MSc, University of Twente, Twente, the Netherlands, 2016.

[16] Sandnes J. Applying machine learning for detecting exploit kit traffic. MSc, University of Oslo, Oslo, Norway, 2017.

[17] Pedregosa F, Varoquaux G. Scikit-learn: Machine Learning in Python. 2011.