# Detecting malicious URLs. A semi-supervised machine learning system approach.

Anton Dan Gabriel
*"Al.I. Cuza" University - Faculty of Computer Science*
*Bitdefender Laboratory*
*Iaşi, România*
*Email: danton@bitdefender.com*

Dragoş Teodor Gavriluţ
*"Al. I. Cuza" University - Faculty of Computer Science*
*Bitdefender Laboratory*
*Iaşi, România*
*Email: dgavrilut@bitdefender.com*

Băetu Ioan Alexandru
*"Al. I. Cuza" University - Faculty of Computer Science*
*Bitdefender Laboratory*
*Iaşi, România*
*Email: abaetu@bitdefender.com*

Popescu Adrian Ştefan
*"Al. I. Cuza" University - Faculty of Computer Science*
*Bitdefender Laboratory*
*Iaşi, România*
*Email: apopescu@bitdefender.com*

***Abstract*—As malware industry grows, so does the means of infecting a computer or device evolve. One of the most common infection vector is to use the Internet as an entry point. Not only that this method is easy to use, but due to the fact that URLs come in different forms and shapes, it is really difficult to distinguish a malicious URL from a benign one. Furthermore, every system that tries to classify or detect URLs must work on a real time stream and needs to provide a fast response for every URL that is submitted for analysis (in our context a fast response means less than 300-400 milliseconds/URL). From a malware creator point of view, it is really easy to change such URLs multiple times in one day. As a general observation, malicious URLs tend to have a short life (they appear, serve malicious content for several hours and then they are shut down usually by the ISP where they reside in).**

**This paper aims to present a system that analyzes URLs in network traffic that is also capable of adjusting its detection models to adapt to new malicious content. Every correctly classified URL is reused as part of a new dataset that acts as the backbone for new detection models. The system also uses different clustering techniques in order to identify the lack of features on malicious URLs, thus creating a way to improve detection for this kind of threats.**

**Keywords: malicious URLs, semi-supervised learning, big data, data streams**

## I. Introduction

We have entered an age when terms like malicious software, cyber-attacks or targeted infections started to be heard of more and more often. Most malicious software, shortly named malware, or attacks are propagated most of the time using the Internet. With more than a half of the world population having Internet access, delivering malicious content on the web has become a usual technique for bad actors.

Whether the potential victims are searching for information in a browser or opening an attachment in an email message, it all comes down to first accessing an URL (Uniform Resource Locator), commonly and informally termed as a web address.

The flexibility and convenience of having content immediately available using the Internet is great for the regular user, but also for an attacker with bad intentions. The begin with, using a URL as an entry point to the actual bad-intended payload, it is very easy for an attacker who has control of the malware hosting server to simply change the content which the URL points to once the payload is identified as malicious. It is also very common that in order to avoid detection or make it more difficult for security researchers, bad actors deliver different malicious content depending on the location the users access the URL from. Another practice that malware creators are often using is server-side polymorphism. This way, after every URL access, a different content is generated and delivered to the unsuspicious user.

Moreover, given the fact that one can generate a different URL and make it available very easily and very fast, most malicious URLs or their malicious content are available only for a limited period of time. Thus, a traditional detection model which only tries to classify previously known URLs would have difficulties in keeping up with the detection accuracy.

To tackle some of the problems described above and to prevent a bad actor to infect a users system, a network based filter can be used. Every URL accessed on the system passes through this filter, which classifies the URL as either benign or malicious and determines whether the URL can still be accessed by the user. Practically, the model has to work with a real-time stream of URLs and, in order not have a big performance hit, must respond in terms of milliseconds. However, if the detection logic behind the model is based only on recognizing previously seen URLs or domain host names extracted from the URLs, such a model would have limited efficiency.

We propose in this paper a system based on semi-supervised machine learning which tries to classify URLs and overcome and use in our best interest techniques used

2470-881X/16 $31.00 © 2016 IEEE
DOI 10.1109/SYNASC.2016.39

by bad actors in order to avoid detection. The network based filter will intercept and send the URL to a model which will be hosted in the cloud, in order to have greater flexibility and instant response time. The system which can be named URL-Blocker would try to extract features from the URL and use them in order to train a model. The model would then try to classify streams of URLs and in the same time use them as entries for a future training of the model after being validated by an external system which determines the correct label (benign or malicious) of them. This technique overcomes a situation when a bad actor tests locally if the malicious URL he tries to publish is detected by a security product. The bad actor would generate a new URL and test it, sending it to our model in the cloud, and hope it would not be detected. As our model tries to learn and adapt from the stream of URLs received and also validate the labels of the classified URLs, it would make the evasion of detection more difficult.

## II. Related work

This paper is a continuation of [9], a work of three authors, two of them being authors of the current paper. To prevent attacks that are using the Internet as an infection vector, researchers discovered various solutions. One of them is to use signature based detection for identifying malicious content or even URLs that point to it. As the authors showed in [15], blacklisting [13] and rule-based or heuristic detection [14] are good ideas, but they suffer from the same problem: the difficulty in adapting to new malicious URLs and URL patterns. Another idea is to use machine learning algorithms. Features can be extracted from the lexical form of the URL - schema, hostname, top-level domain, primary domain name, path or query. Also learning algorithms can use host-based features such as IP address, WHOIS properties, domain name properties, geographic properties or connection speed. Features extracted using this external information were used in many papers such as [8] and [3]. In [7], [4], [11] and [3] the authors went forward and correlated the URL to the downloaded resource content.

In [9], Popescu et. all researched the way different offline supervised and unsupervised learning methods can be used in order to detect malicious URLs with respect to the memory footprint. Extracted features are based only on the URL string and are not using external information such as the resource indicated by the URL. The machine learning algorithm tested in [9] is the One-Side Class (OSC) Perceptron [5] which offers good detection rates with zero false positives. FASV, Fingerprint algorithm - voting system [10] - was used for generating clusters and classifying URLs. The results show that these algorithms can be used for detecting malicious URLs that have a short life span or the ones that are used in targeted attacks.

In [1], the authors extracted features based on the URL string, network information, web content, DNS information and link popularity. Two algorithms were tested in order to detect malicious URLs: Support Vector Machine and Naive Bayes. The authors assert that the latter shows better accuracy.

Ma et al. (2009) [8] explored the possibility of using online supervised learning in order to detect malicious Web sites using features extracted from both the lexical form of the URL and external information. The authors tested different online learner algorithms including the perceptron and confidence-weighted (CW) algorithm. Using a continuous training technique, the underlying model is updated using a real-time URL stream. Ma et al. showed the importance of retraining algorithms with new features in order to adapt the model to the evolving URL live feed. They obtained good detection rates, but without giving details on the false positives.

Blum et al. [2] also researched the way confidence-weighted classification can be used for detecting phishing related URLs. The difference from [8] is that the feature set is not using host-based information. The only information used is the URL string. The results are comparable to the ones from [8], but also without giving too much details on the number of false positives.

As Gavrilut et al. stated in [5], when using a perceptron algorithm or other linear classifier in detecting malware, one must be very careful about the false positive rate, because in real life situations false positives are more dangerous than false negatives. The authors proposed in [5] a modified version of the perceptron, the one-side class (OSC), which offers good detection rates with a small number or even zero false positives.

In [6], Lee et al. presented a real-time software system that is able to detect suspicious URLs for spam, phishing and malware distribution in the Twitter live stream. The L2-regularized logistic regression algorithm is used along with a sliding window technique.

In [15], Zhao et al. present a framework of cost-sensitive online active learning (CSOAL) which address the problem of detecting real-world online malicious URLs which queries only a part of the URLs for labeling. Features were extracted from both the URL string and host-based information. In real-life situations the live stream of URLs accessed by users is highly imbalanced regarding the ratio between malicious and benign elements. Instead of maximizing the online accuracy, the authors decided to optimize another performance metric: the sum of weighted sensitivity and specificity. That is because the number of malicious URLs is significantly smaller than the one of benign URLs and a learner that classifies all the URLs as benign could obtain a very high accuracy.

The rest of the paper is organized as follows: Section 3 describes the architecture of a system that is capable of analyzing a stream of URLs and provide real-time detection. Section 4 presents our results and the steps we had to take

for reaching our current solution. Finally, we point out some conclusions and improvements that we consider for this system.

## III. System architecture

In order to solve the issues described in Section I, we propose a semi-supervised architecture to identify and block suspicious URLs. Our goal is to create a system that uses a detection model which can be easily trained and adapted in order to identify new and prevalent malicious URLs. At the same time, we focus on letting our model learn by itself and also intervene when needed in order to maximize our goals (high detection rate and low false positives).
We divide our system into *Detection System* and *URL Analysis System* with two flows as presented in Fig III.

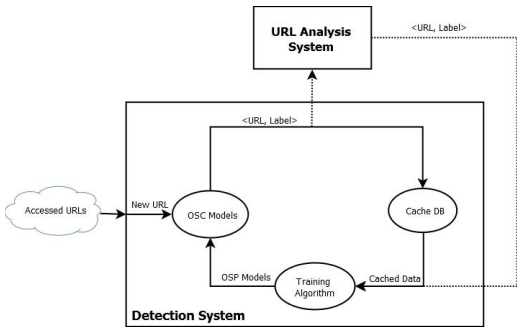


Figure 1. Overview of the detection system and the work-flows between components

Let us consider a newly accessed URL. At first it will be tested by the already trained model and receive a label: malicious or benign. At this moment, from the user point of view, if the URL is considered as being malicious it will be blocked. Otherwise the access to it will be permitted. All the URLs and computed labels will be stored in the Cache DB and will also be sent to the URL Analysis System in order to validate the model output. The URL stream will influence in a specific way the next model trained by the OSC perceptron. In order to better understand the workflow, we will further present each of them.

### A. Detection System

The Detection System consists of three components: the *Training algorithm*, the *Model* and the *Cache Database*:

*1) Training Algorithm:* In order to classify an URL the detection system uses a model trained using the One Side Class perceptron algorithm[5]. Because the detection system will run in a real-life situation, the number of false positives must be as low as possible, even if the detection rate is affected. That is why the model was trained to produce zero false positives.

The training algorithm is using a number of 123 boolean features that are extracted from all the components of an URL (schema, hostname, primary domain, TLD [1], path and query). Some of these boolean features are generated using external services (most accessed URLs, etc.).

We extracted only lexical features, using information from the URL string. Our focus was to eliminate the download of resources indicated by URLs, in order to have a fast response whenever an URL is tested. If we would take into consideration to download the content, the response will be bounded by server availability, content size, number of URLs tested in the same time, etc. Also, the system is not using any time-based or statistical features due to the fact that all features must be extracted at any time from an URLs.

In Table I we present a small set of extracted boolean features:

| URL | Feature name | Value |
|---|---|---|
| http://62.210.6.3/74n66.exe | Executable ext | True |
| http://62.210.6.3/74n66.exe | Has parameters | False |
| http://62.210.6.3/74n66.exe | Domain is IP | True |
| http://62.210.6.3/74n66.exe | Port is standard | True |
| http://22.216.21.73:7101/tp.css?a=1 | Executable ext | False |
| http://22.216.21.73:7101/tp.css?a=1 | Has parameters | True |
| http://22.216.21.73:7101/tp.css?a=1 | Domain is IP | True |
| http://22.216.21.73:7101/tp.css?a=1 | Port is standard | False |
| http://pki.google.com/GIAG2.crl | Executable ext | False |
| http://pki.google.com/GIAG2.crl | Has parameters | False |
| http://pki.google.com/GIAG2.crl | Domain is IP | False |
| http://pki.google.com/GIAG2.crl | Port is standard | False |
| http://pki.google.com/GIAG2.crl | Domain is known | True |
| http://pki.google.com/GIAG2.crl | Domain is whitelisted | True |

Table I
EXAMPLES OF EXTRACTED FEATURES FROM DIFFERENT MALICIOUS URLS (THE FIRST TWO) AND BENIGN URLS (THE LAST ONE). THE LAST TWO FEATURES ARE USING EXTERNAL DATABASES (DOMAINS CONSIDERED BENIGN AFTER AUTOMATED ANALYSIS AND DOMAINS ACCESSED BY A HIGH NUMBER OF USERS)

*2) Model:* The system is using three models - all based on OSC Perceptron ([5]):

- a detection model. This model is used to detect malicious URLs as they are pulled out of the stream. It is also automatically adjusted every hour to compensate for new malicious data that may appear.
- two OSC perceptron models needed for error correction. In section Result will are explaining how we end up using this model.

*3) Cached DB:* This is a database used to collect URLs for a limited period of time (in our case one hour). Its main purpose is to provide the system with a way to retrain the model based on the last received URLs. The system will clean up after each hour - meaning that every adjustment that we make will be limited to data received in the previous hour. While this limit can be modified, we chose to use one hour to better detect URL variations specific to the time zone.

[1]Top Level Domain - RFC 1591, March 1994

### B. URL Analysis systems

The URL Analysis system is developed independently[10] having the purpose of classifying URLs that cannot be labeled with a high certainty. We will also use the system in order to validate our data and analyse the detection and false positive rate.

It is important to mention that such a system cannot be used in-line due to the fact that a response from such a system would produce a high delay time in a real life scenario. Also we also need to take into consideration that only a single user is accessing multiple URLs when accessing a single website (due to the advertisements, iframes, etc.), so there is a huge amount of URLs accessed by millions of users in the same time.

More information about the system will be offered in the next section.

## IV. Results

As previously stated, dealing with a stream of data instead of analyzing its content off-line comes with some advantages and some problems as well. Obviously, testing the stream will be performed in real time. This means that if the model used for analysis is to be adjusted according to new threats, we will have to use the feedback from a secondary system that will analyze all of the URLs from the stream in an off-line mode. The second thing that needs to be pointed out is that in practice it's quite difficult to adjust a model after each URL that comes into the system. It is easier to collect all URLs for a small period of time and adjust a new model based on them. For this paper we have used time frames of one hour to collect URLs and analyze them.

For the purpose of this research we have used an OSC algorithm ([5]) to train and test the models used for detection the malicious URLs.

Our stream of data consists in URL collected over 1000 hours starting from 1st of May 2016. After the stream analysis all of the URLs were analyzed by secondary systems and labeled (this help us check if our stream based detection methods provide a proper protection). Duplicate URLs were removed from the stream as they would not reflect the change of the content. The total number of URLs that we've analyzed is 2057091 benign URLs and 562424 malicious ones.

Initially we created a model based on URLs collected during 10 hours before we started the test. That model was the start up point for the rest of our detection systems. Secondly we needed to validate how proactive such a model is. Figure 2 shows how the detection rate varies over the 1000 time frames of one hour for our model. We will further refer to this method as M1 method.

As it turns out, the detection rate varies a lot, from almost 1-2 % to more than 99%. It also appears that from time to time, a specific type of malicious URLs is reused by malware creators. It's difficult to say why but we assume that it might

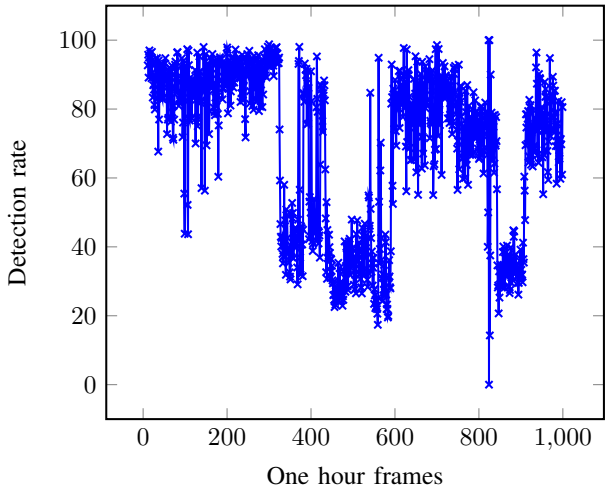


Figure 2. Detection rate for each of the 1000 one hour frames of data

be related to spam campaigns and the fact that in many cases the security measures used for malicious URL detection are based on blacklisting - which means that one can create two similar URLs that are located on different domains, and from a black list point of view, only the ones known will be detected.

Furthermore, let's analyze Figure 3. It presents the variation of the detection rate over 80 frames staring from frame 300 and ending up at frame 380. As it can be observed the detection maintains a steady 90% - 95% detection rate for the first 30. Than in about 5 to 10 frames it drops to 40% where it remains for almost 40 frames. We suspect that during these 40 frames (2 days of URLs) a new campaign of spam was started with a very specific set of malicious URLs that cannot be fully detected by our model. However, the most interesting thing is that drop of detection is not immediate but over a period of 5 frames (5 hours). This means that if can detect such moment, we can adjust o model and be better prepared for the types of URLs that are to come.

But how can we pin point such a moment? First of all, we work on a stream level. This means that we don't know any of the labels of the URLs that we find. We can only know the label that our model assign to an URL but this is not an error proof approach. And if we try to analyze all of the stream data, we will probably find such negative peaks, but only after they occur. The use of another system to help us correctly label a sample is a good idea but not for the number of samples that we need to analyze, which means that we need to find a way to send to this system only the samples that are likely to be misclassified by our model.

For this we developed the following method. Given the fact that we use an OSC based algorithm, instead of creating only one model, we created two models: one that can classify

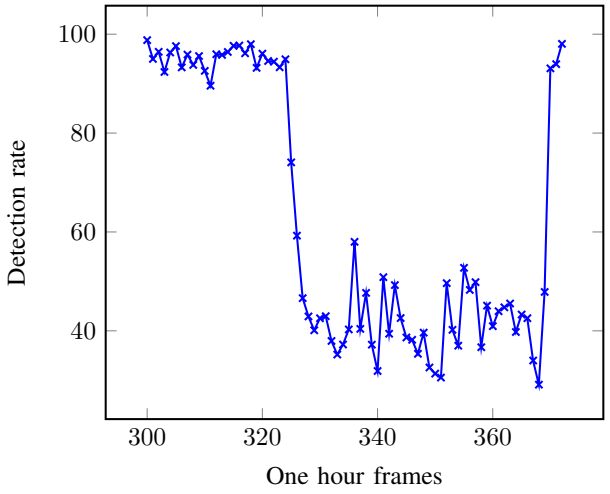


Figure 3. Detection rate for each of the 1000 one hour frames of data, starting with frame 300 and ending with frame 380

benign URLs and one that can classify malicious URLs. Figure 4 shows such two models. The idea behind them is that most likely each model would correctly classify the class for which it was trained for. Model B (blue line) will correctly classify squares (if an object is labeled as a square and its position is on top of the blue line). Model A (red line) will correctly classify stars (if an object is labeled as a star and its position is on bottom of the red line). This means that the elements that are located between those two lines are likely to be the ones that produce a misclassification error.

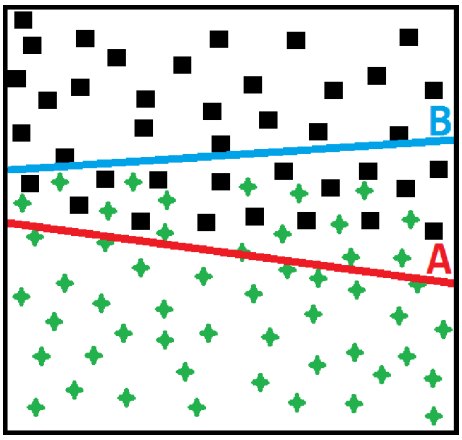


Figure 4. Two OSC models. The elements between the two hyper-planes are elements that are likely to produce a classification error

We conducted the following experiment. For each time frame from our set of 1000 frames, we built two OSC models similar to the way presented in Figure 4. After this, we computed the percentage of elements that are located between

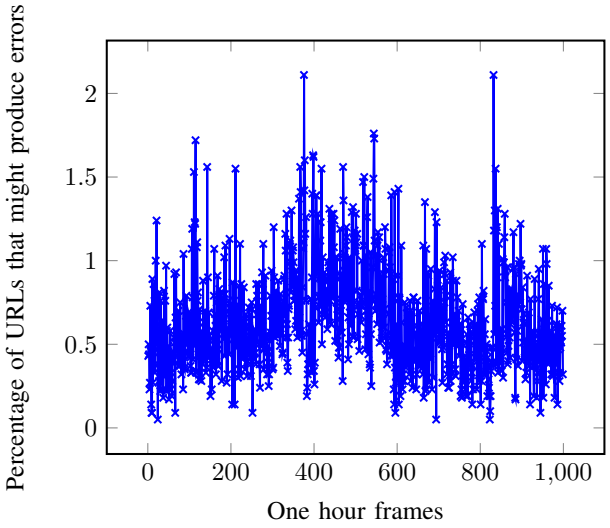


Figure 5. Percentage of URLs between the planes A and B for each of the 1000 one hour frames

those two planes. Figure 5 shows how this value varies for each time frame. The average percentage of elements that are likely to produce errors is 0.64%. This value is small enough so we can in fact use the URL Analysis system in order to analyze some samples but only use it for 0.64% of the samples from each frame. This reduces the number of URL samples that need to be processed to an acceptable value that allows us to have a proper labeling system by the time the next frame starts.

Based on this observation we propose algorithm 1 for an semi-supervised method (denoted from now on as (MS) ), where:

- **F** is a collection of URLs collected during a time frame of one hour.
- **m** is the last model we have used for detecting malicious URLs.
- $osc_m$ is the last OSC model capable of correctly classify malicious URLs
- $osc_c$ is the last OSC model capable of correctly classify benign URLs
- $Train$, $TrainOSCMalware$ and $TrainOSCClean$ are functions used to train a new model (with specific characteristics - for example $TrainOSCMalware$ will create a model that will correctly classify malicious URLs and reduce the number of false positives - all benign samples will reside on one side of the hyper-plane).

After applying this method for the training model on stream we obtained the results from Figure 6. As expected, some negative peaks (frames with lower detection rate) will still be present. Even if the two OSC models act as a way to reduce the number of labeling errors on a stream, they

**Algorithm 1** Stream model training (MS)

**function** $TrainNewModel(\ F,\ m,\ osc_m,\ osc_c)$
  $Posible_m \leftarrow \emptyset$;
  $Posible_c \leftarrow \emptyset$;
  **for** $i = 1 \rightarrow |F|$ **do**
    **if** $F_i$ *is classified by* $osc_c$ *as malicious* **then**
      $Posible_m \leftarrow Posible_m \cup \{F_i\}$;
    **end if**
    **if** $F_i$ *is classified by* $osc_m$ *as benign* **then**
      $Posible_c \leftarrow Posible_c \cup \{F_i\}$;
    **end if**
  **end for**
  $Errors \leftarrow Posible_m \cap Posible_c$;
  **for** $i = 1 \rightarrow |F|$ **do**
    **if** $F_i \subset Errors$ **then**
      *Label* $F_i$ *acording to secondary system*;
    **else**
      *Label* $F_i$ *acording to* $m$;
    **end if**
  **end for**
  $m \leftarrow Train(F)$;
  $osc_m \leftarrow TrainOSCMalware(F)$;
  $osc_c \leftarrow TrainOSCClean(F)$;
  **return** $(\ m,\ osc_m,\ osc_c)$
**end function**

are still bound to have errors of themselves. This reflects in some of the negative peaks that are likely to appear because of some benign samples that were labeled as malicious or vice-versa in the previous one-hour time frame. However, the overall detection rate has improved to 82.9% as seen in Table II. The number of false positives has indeed increased, but if we consider an average of 2057 benign URLs for each time frame, then 15.42 means 0.7% false positives.

| Method | Average Detection rate (%) | Average FPs |
|---|---|---|
| M1 | 68.64 | 7.15 |
| MS | 82.89 | 15.42 |

Table II
AVERAGE FALSE POSITIVE AND DETECTION RATE FOR ONE-MODEL (M1) AND STREAM MODEL (MS) TESTS

## V. CONCLUSION

Using results on about 40 days (1000 hours), we managed to create a relatively stable algorithm for classifying benign and malware URLs, which needs from time to time some external input in order to improve the accuracy and detection rate of the model. If we keep the number of elements that are likely to produce errors low enough in order to be able to be validated by an external system, either Virtual Machine farms or human analysis, we can considerably boost the detection rate of the model with the cost of a small increase of the number of false positives.

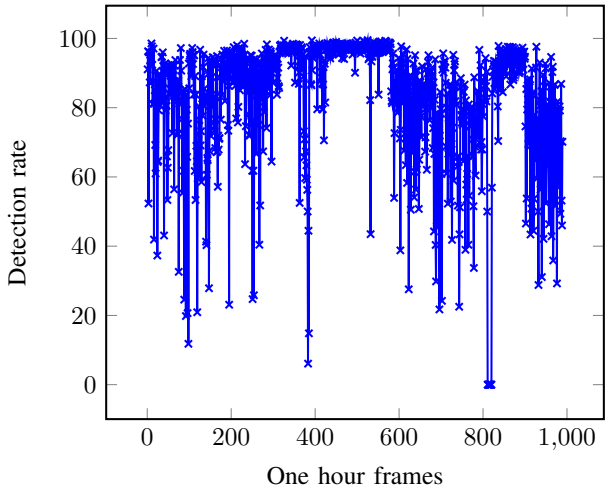


Figure 6. Detection rate for each of the 1000 one hour frames of data using Stream training method (MS)

For future work, we plan to improve our model by trying algorithms other than the OSC Perceptron, creating new better features and also trying a training model using mapped features.

## REFERENCES


[1] Arati M. Dixit Anjali B. Sayamber. Malicious url detection and identification. *International Journal of Computer Applications (0975 8887) Volume 99 No.17, August 2014*, 2014.

[2] Aaron Blum, Brad Wardman, Thamar Solorio, and Gary Warner. Lexical feature based phishing URL detection using online learning. In *Proceedings of the 3rd ACM Workshop on Security and Artificial Intelligence, AISec 2010, Chicago, Illinois, USA, October 8, 2010*, pages 54–60, 2010.

[3] Hyunsang Choi, Bin B. Zhu, and Heejo Lee. Detecting malicious web links and identifying their attack types. In *2nd USENIX Conference on Web Application Development, WebApps'11, Portland, Oregon, USA, June 15-16, 2011*, 2011.

[4] Marco Cova, Christopher Krügel, and Giovanni Vigna. Detection and analysis of drive-by-download attacks and malicious javascript code. In *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, pages 281–290, 2010.

[5] Dragos Gavrilut, Razvan Benchea, and Cristina Vatamanu. Optimized zero false positives perceptron training for malware detection. In *14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2012, Timisoara, Romania, September 26-29, 2012*, pages 247–253, 2012.

[6] Sangho Lee and Jong Kim. Warningbird: Detecting suspicious urls in twitter stream. In *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012*, 2012.

[7] Long Lu, Vinod Yegneswaran, Phillip A. Porras, and Wenke Lee. BLADE: an attack-agnostic approach for preventing drive-by malware infections. In *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*, pages 440–450, 2010.

[8] Justin Ma, Lawrence K. Saul, Stefan Savage, and Geoffrey M. Voelker. Identifying suspicious urls: an application of large-scale online learning. In *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*, pages 681–688, 2009.

[9] Adrian-Stefan Popescu, Dumitru-Bogdan Prelipcean, and Dragos Teodor Gavrilut. A study on techniques for proactively identifying malicious urls. In *17th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2015, Timisoara, Romania, September 21-24, 2015*, pages 204–211, 2015.

[10] Adrian-Stefan Popescu, Gavrilut-Dragos Teodor, and Daniel-Ionut Irimia. A practical approach for clustering large data flows of malicious urls. *Journal of Computer Virology and Hacking Techniques*, pages 1–11, 2015.

[11] Moheeb Abu Rajab, Lucas Ballard, Noe Lutz, Panayiotis Mavrommatis, and Niels Provos. CAMP: content-agnostic malware protection. In *20th Annual Network and Distributed System Security Symposium, NDSS 2013, San Diego, California, USA, February 24-27, 2013*, 2013.

[12] Konstantin Salomatin Jaime Carbonell Siddharth Gopal, Yiming Yang. Statistical learning for file-type identification. *ICMLA12*, 2012.

[13] Jian Zhang, Phillip A. Porras, and Johannes Ullrich. Highly predictive blacklisting. In *Proceedings of the 17th USENIX Security Symposium, July 28-August 1, 2008, San Jose, CA, USA*, pages 107–122, 2008.

[14] Yue Zhang, Jason I. Hong, and Lorrie Faith Cranor. Cantina: a content-based approach to detecting phishing web sites. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pages 639–648, 2007.

[15] Peilin Zhao and Steven C. H. Hoi. Cost-sensitive online active learning with application to malicious URL detection. In *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013*, pages 919–927, 2013.