Real--time Classification of Malicious URLs on Twitter using Machine Activity Data

Conference Paper · August 2015

DOI: 10.1145/2808797.2809281

CITATIONS
19

READS
88

4 authors:

Amir Javed
Cardiff University
13 PUBLICATIONS **68** CITATIONS
SEE PROFILE

Pete Burnap
Cardiff University
95 PUBLICATIONS **2,057** CITATIONS
SEE PROFILE

Omer F. Rana
Cardiff University
565 PUBLICATIONS **6,871** CITATIONS
SEE PROFILE

Malik Shahzad Kaleem Awan
University of Plymouth
23 PUBLICATIONS **146** CITATIONS
SEE PROFILE

2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining

Real-time Classification of Malicious URLs on Twitter using Machine Activity Data

Pete Burnap, Amir Javed, Omer F. Rana, Malik S. Awan
School of Computer Science and Informatics
Cardiff University
Cardiff, UK
Email: burnapp@cardiff.ac.uk

***Abstract*—Massive online social networks with hundreds of millions of active users are increasingly being used by Cyber criminals to spread malicious software (malware) to exploit vulnerabilities on the machines of users for personal gain. Twitter is particularly susceptible to such activity as, with its 140 character limit, it is common for people to include URLs in their tweets to link to more detailed information, evidence, news reports and so on. URLs are often shortened so the endpoint is not obvious before a person clicks the link. Cyber criminals can exploit this to propagate malicious URLs on Twitter, for which the endpoint is a malicious server that performs unwanted actions on the person's machine. This is known as a drive-by-download. In this paper we develop a machine classification system to distinguish between malicious and benign URLs within seconds of the URL being clicked (i.e. 'real-time'). We train the classifier using machine activity logs created while interacting with URLs extracted from Twitter data collected during a large global event – the Superbowl – and test it using data from another large sporting event – the Cricket World Cup. The results show that machine activity logs produce precision performances of up to 0.975 on training data from the first event and 0.747 on a test data from a second event. Furthermore, we examine the properties of the learned model to explain the relationship between machine activity and malicious software behaviour, and build a learning curve for the classifier to illustrate that very small samples of training data can be used with only a small detriment to performance.**

I. Introduction

Online social networks (OSNs) (e.g. Twitter, Facebook, Tumblr) are inherently vulnerable to the risk of collective contagion and propagation of malicious viral material such as spreading rumour [18] and antagonistic content [3] following widely publicized emotive events. Another misuse case includes the spread of malicious software (malware) via URLs, for which the endpoint is a Web page that contains a malicious script. When the client device executes the script, it attempts to exploit a vulnerability in the browser or a plugin to perform malicious activity on the client device. This action is commonly referred to as a drive-by-download [19]. The 2013 Security Intelligence Report from Microsoft reported that malicious Web sites are now the top threat to enterprise security [15].

Possibly the most prominent example of the injection of malicious URLs into OSNs is the *Koobface* worm [25]. Koobface initially spread by using an infected machine to send messages to Facebook 'friends' of the infected user, which included a link to a third-party website that infected the machine of the user visiting it by installing malicious software. The worm was effectively executed on a number of OSNs due to the highly interconnected nature of the their users. Thomas and Nicol subsequently analysed Koobface to identify the social network accounts used to distribute malicious URLs and notably identified that current defences flagged only 27% of threats and took 4 days to respond. During this period, 81% of vulnerable users clicked on Koobface links [25]. This highlights the ineffectiveness of existing drive-by-download detection methods and motivates further research into ways in which malware on OSNs can be identified, and its propagation monitored and managed.

Being an open, real-time, and highly interconnected micro-blogging platform, Twitter is one of the most widely used OSNs. It is becoming a crowd-sourced news reporting platform [8] and go-to source for the latest information and reaction following large-scale events, such as natural disasters [20], political elections [26] and terrorist attacks [3]. Due to the 140 character limit on posts imposed by Twitter, users often post URLs to provide additional sources of information to back up claims, give a more complete report of events, or supply evidence for a statement. Twitter therefore has the potential to become an environment within which Cyber criminals can *piggyback* on large-scale events and lure information seeking users to malicious endpoints. Identifying which events are most popular is made easier due to the openly accessible 'trending topics' list that can be produced by analysing the stream of Tweets over time, and which summarises the most frequently used terms and hashtags [10].

To date the majority of research into malicious content posted to Twitter has investigated the social network properties of user accounts, such as friends and followers, to identify malicious accounts. In this paper we explore whether the behavioural analysis of malware can lead to the identification of machine activity-related features to support the near real-time classification of malicious URLs. The aim of this paper is (i) to develop a real-time machine classification system to distinguish between malicious and benign URLs within seconds of the URL being clicked, and thus possibly before the full extent of the planned malicious activity can be successfully executed, and (ii) to examine and interpret the learned model to explicate the relationship between machine activity and malicious behaviour exhibited by drive-by-downloads on Twitter. We achieve this by training several machine classification models using machine activity logs generated while interacting with URLs extracted from Twitter data collected during two large sporting events – the Superbowl and the Cricket World

ASONAM '15, August 25-28, 2015, Paris, France
ACM ISBN 978-1-4503-3854-7/15/08
DOI: http://dx.doi.org/10.1145/2808797.2809281

Cup.

First, we present a log-based machine classifier for URLs posted to Twitter, which can distinguish between malicious and benign URLs within seconds of the interaction beginning (i.e. 'real-time'). This can be used as an early warning tool to help identify and neutralise a drive-by-download during large scale events, when it is known people are highly likely to search for information and click on URLs. As far as we are aware, this is the first study that provides a comparative analysis of different types of modelling methods for the run-time classification of URLs posted to Twitter. Second, the volume of Twitter data generated around such events makes it impossible to manually achieve this task, creating data sampling, storage and analysis challenges. We refine the set of measurable machine activity metrics to identify the most predictive features by examining how they are used to make decisions in the machine classification model. We explain these features and furthermore, build a learning-curve chart that demonstrates the sample required to train a model is much smaller than 100%, suggesting in fact that using 1% of the training data only has a small detrimental effect on performance, reducing the overhead on data collection and mitigating sampling concerns such as 'how much data is enough' for this problem.

II. Related Work

A. Malware Propagation and Social Networks

It has been demonstrated that OSNs play a crucial role in the spread of information between people, and the social network structure can have a control effect on the speed of propagation [13]. The literature contains a strong focus on social network account features, such as number of friends and followers. [21] analysed malware on Twitter and studied the importance of social network connectivity between users, also incorporating theory on epidemiology. They found that, even with a low degree of connectivity and a low probability of clicking links, the propagation of malware was high. [6] also used infection models from the study of contagious diseases to simulate user interactions and model the propagation of malware, finding that highly clustered networks and interactions between 'friends' rather than strangers slowed the propagation. These findings suggest that users seeking information from strangers on Twitter following a large-scale event would be an ideal scenario for the spread of malicious content.

[29] analysed the "cyber criminal ecosystem" on Twitter using graph metrics including local clustering coefficient, betweenness centrality, and bi-directional links ratio, and found that malicious accounts tend to be socially connected forming small networks, and such accounts are also more likely to be connected via a 'friendship' relationship.

B. Classifying Malicious Web Pages

To determine whether a Web page was likely to perform a malicious activity, [4] used static analysis of scripts embedded within a Web page to classify pages as malicious or benign. [5] also studied static code to identify drive-by-downloads, using a range of code features including sequences of method calls, code obfuscation mechanisms and redirection techniques. [7] provided an enhancement to static code analysis to handle evasive malware that detects the presence of dynamic analysis systems and tries to avoid detection. The enhancements include a code similarity detection tool and an evasive behaviour detector. [17] used client honeypots to interact with potentially malicious servers and analyse the code using anti-virus and HTML decomposition. [12] used URL redirect chains across a corpus of Tweets to identify malicious web site reuse.

[16] developed a more interactive dynamic behaviour based tool to identify malicious web pages, arguing that code-based analysis is more manually intensive, and instead monitored the run-time interactions between a web page and the client machine for several minutes. If there were persistent-state changes on the client machine, such as new processes created, registry files modified or executables created, this behaviour was used to signal a malicious exploit. The study actually identified a zero-day exploit of an unpatched java vulnerability that was operating behind 25 malicious URLs, making a strong case for dynamic behaviour modelling over static code analysis. [9] subsequently reported that analysing run-time behaviours has better performance in term of detection accuracy over static analysis. The authors propose a two-step process, using static analysis to provide a first-pass analysis, before forwarding potentially malicious web pages for run-time analysis. Of course, this approach has the potential to overlook zero-day exploits where new code-based attack signatures are as yet unknown, but the two-step approach reduces the overhead of running run-time analysis on all URLs.

To summarise, while the majority of research into the propagation of malicious content on OSNs has investigated the social network properties of user accounts, such as friend and followers to identify malicious accounts, we are more concerned with the active analysis of malware behaviour on Twitter, and understanding the features of machine behaviour that enable the 'real time' classification of malicious URLs posted to Twitter, while reducing false positives. Thus, we are interested in identifying malware via its behaviour, which is logged within 5 seconds of the first interaction with the URL, and at regular intervals for a 5 minute period of interaction. As far as we are aware, this is the first study on the run-time classification of URLs posted to Twitter surrounding large-scale events.

III. Data Collection and Annotation

A. Data Collection

One of the key concerns with data-driven models is whether the results can be generalized to other datasets (and events). To build a model of malicious behaviour from machine log data, and avoid a model built solely on the data characteristics from a single event we collected data from two large-scale events. Data from one event was used to train a classifier and build a model, with data from another event being used to test the model's generalisability beyond a single event. Sporting events have been reported to generate large volumes of Twitter traffic so we collected data at the time of two world events - the American Football Superbowl, and the cricket World Cup. Data for our study were collected from Twitter using its programmatically accessible streaming API using Tweepy (tweepy.org). We chose Twitter as the OSN for the study because it supports the collection of 1% of all daily tweets,

which was assumed to be of sufficient bandwidth to collect all Tweets explicitly mentioning the two events. Large sporting events were chosen as it has been reported that they have produced the largest ever volume of Twitter traffic[1].

Super Bowl and cricket World Cup tweets were collected using event related search hashtags (i.e. #superbowlXLIX, #CWC15). Tweets were also required to contain a URL to be relevant to our study. On 1st February 2015, the day of the Super Bowl final, we collected 122,542 unique URLs posted in tweets containing the event-specific hashtags. As the cricket World Cup final generated less traffic we also collected at the time of the semi-finals. In total 7,961 URLs were collected from the two semi-finals (24th and 26th March) and the final (29th March).

B. Identifying Malicious URLs

The second stage of the experiment involved building up a system to analyse the activity of all URLs collected from the Twitter stream and determine whether to annotate the URL as malicious or benign. To carry out stage two we utilised a client-side honeypot system[19]. Client honeypots are active defence systems that visit potentially malicious URLs and log the system state during the interaction. They can perform static analysis on Web page code, looking for evidence of malicious scripts (e.g. [5] [7]. These are generally referred to as *low interaction* honeypots. Alternatively, they can perform dynamic analysis of the interaction behaviour between the Web page and the client system, looking for evidence of malicious actions (e.g. [16] [9]). These are known as *high interaction* honeypots.

Given our intention to build a model of malicious behaviour and to incorporate potential zero day attacks for which the code may be previously unseen, but the behaviour is clearly malicious, we chose to use a high interaction client honeypot. The Capture HPC toolkit [22] is an open source high interaction client honeypot that can be used to undertake run-time behavioural analysis of malicious pages [1]. With a general purpose operating system, it is a challenging task to deduce behaviour from a log file as a large number of events can be generated (and many when the system is idle). CaptureHPC on the other hand restricts logging of events of particular interest to a user. Once a server has been initiated, it can start up multiple clients, each within its own virtual machine (via VMWare) enabling multiple concurrent clients to co-exist in their own virtual machine. Capture-HPC uses an exclusion list of malicious behaviours and analyses the log files of a client honeypot after visiting suspicious URLs. Changes to machine state such as registry, file system or running processes can lead to the URL being flagged as malicious [19]. Exclusion lists have limitations in that they require constant updating as malicious behaviour changes over time. New vulnerabilities are identified and subsequently attack vectors change. However, there are rigorous methods for determining the exclusion lists as discussed in [19].

The methodology for linking streamed Twitter data to a high interaction honeypot is as follows: (1) connect to the Twitter Streaming API and send the search term to collect on (#superbowlXLIX) and specifying that only posts containing URLs should be returned. This returns Tweets as they are posted. Write the details of the Tweets to a database; (2) expand shortened URLs and remove duplicates; (3) for every 500 (new) URLs, upload a text file using a (clean) Capture HPC virtual machine; (4) Capture HPC iterates through the list, visiting each URL, and keeping the connection open for 5 minutes, logging machine activity on the client machine and identifying whether the URL is malicious or benign via reference to its exclusion list. The 5 minute interval is currently a heuristic to ensure that a large number of sites can be visited – it makes the *significant* assumption that any malicious activity will be triggered within the first 5 minutes of the visit.

C. Architecture for suspicious URL annotation

The Capture HPC client honeypot system is designed to operate in a Virtual Machine (VM) environment, to ensure any side effects due to interacting with suspicious URLs and logging interactions do not carry over to other activities. Once the orchestration machine was bootstrapped, the Capture HPC clients established a communication channel between the machine on which it was running and the endpoint to which it was directed by the next suspicious URL in the list. The client honeypot then logged changes to the state of the system, using file, registry and process monitors running on the client machine. Based on the logged activity, Capture HPC determines whether the URL should be classified as benign or malicious.

Whether visiting a URL, opening an application, or being idle, the operating system generates hundreds of system events, many of which would be benign. These events should be omitted from the log files generated by various monitors. To implement exceptions, exclusion lists were created that can be populated based on probable malicious behaviour, and are portable to different operating systems. By default Capture HPC client monitors logs all activity – however, this might result in a large volume of data being recorded, much of which may not be relevant to flag any malware-based activity. Therefore, a user can either specify their own omission or an inclusion rule to the existing list to enable recording of events that are likely to be of interest in the context of particular malware analysis, additional detail can be found in Seifert et al. [22] and Puttaroo et al. [19].

Capture HPC relies on a malicious activity being executed for it to assign the 'malicious' label to a URL. However, the activity log is collected from the point of first interaction with the endpoint. This poses the question as to whether the activities that precede a malicious event contain 'signals' that can be identified as being linked to an eventual exploit, and can be used to flag this risk. Our paper essentially answers this question by using the log data activities as predictive features in the URL classification system.

D. Sampling and Feature Identification

From the pre-processed collection of tweets we randomly sampled 2000 Tweets from each event. The training data, sampled at regular intervals throughout the Super Bowl data collection period, contained 1000 URLs identified by Capture HPC as being malicious, and 1000 benign. This provided a training set of 2000 tweets. The test set collected around the time of the cricket World Cup consisted of 891 malicious

[1] http://mashable.com/2014/07/09/brazil-germany-world-cup-most-tweeted/

URLs and 1,100 benign (sampled 80% from the day of final and 10% from each of the semi finals). To identify features that were predictive of malicious behaviour we collected machine log data during the period Capture HPC was interacting with the URL. The metrics we measured were: i) CPU usage, ii) Connection established/listening (yes/no), iii) Port Number, iv) Process ID (number), v) Remote IP (established or not), vi) Network Interface (type e.g. Wifi, Eth0), vii) Bytes Sent, viii) Bytes received, ix) Packets Sent, x) Packets Received, and xi) Time since start of interaction. In total there were 5.5 million observations recorded from interacting with 2000 tweets (1000 malicious and 1000 benign). Each observation represents a feature vector containing metrics and whether the URL was annotated by Capture HPC as malicious or benign.

IV. Data Modeling

A. Baseline Model Selection

Our data modelling activity was intended: (i) to identify a model suitable for extracting features from machine activity logs that would be predictive of malicious behaviour occurring during an interaction with a URL endpoint, and (ii) to explicate the relationship between machine activity and malicious behaviour. There are a number of methodological considerations to take into account. First was whether to use a generative or discriminative model. Our data contains logs of machine activity, which occurs even when the system is idle, so it is likely that any log will contain a great deal of 'noise' as well as malicious behaviour. Table I presents a comparison between the training and testing datasets with respect to the mean and standard deviation of recorded machine activity. It illustrates the high variance in the mean recorded values of CPU usage, byes/packets sent/received, and ports used between the two datasets, which suggests identifying similar measurements between datasets for prediction purposes would be challenging. The standard deviation in both datasets is very similar, which suggests the variance is common to both datasets, but the deviation is high, which suggests a large amount of 'noise' in the data.

TABLE I. Descriptive Statistics for Train and Test Datasets at t=60

Attribute	Mean		Std. Dev	
	Train	Test	Train	Test
cpu	1.255354	6.26	2.144828	2.31
connection	.86	.88	.34	.32
portnumber	49478.7	38284.41	18023.01	23180.74
processid	3343.329	3480.35	1947.882	1624
remoteip	.86	.88	.34	.32
network	4	4	2	2
bytessent	1.01e+08	3.59e+08	2.06e+08	9.50e+08
bytesrecd	2.87e+08	3.12e+08	8.47e+08	8.90e+08
packetssent	470821.5	2442275	1472258	6659166
packetsrecd	539358	2849133	1843365	7742467

In addition to the 'noise' in the data – although the training and testing datasets contain a well balanced number of malicious and benign activity logs - the behaviours in both logs are largely benign, creating a large skew in log activity towards the benign type. The noise and skew could have an impact on the effectiveness of a discriminative classifier in identifying decision boundaries in the space of inputs (i.e. the inputs may not be linearly separable, which could cause problem using a perceptron-type classifier) even after a large number of iterations (for instance if using a multilayer perceptron developed using multiple layers of logistic regression). It could be argued that for more complex relationships, such as multiple sequential activities leading to a malicious machine exploit, a generative model would be more appropriate to generate a full probabilistic model of all variables (possible behaviours) given a training dataset of machine logs. For example, a Bayesian approach could be effective at capturing dependencies between variables over time. Or a Naive approach to Bayesian modelling may be more suitable by assuming there are no dependencies, but that the probabilistic value of individual variables will be enough to determine likely behaviour.

The first phase of data modelling was therefore to conduct a number of baseline experiments to determine which model would be most appropriate based on prediction accuracy. We used the Weka toolkit to compare the predictive accuracy of (i) generative models that consider conditional dependencies in the dataset (BayesNet) or assume conditional independence (Naive Bayes), and (ii) discriminative models that aim to maximise information gain (J48 Decision Tree) and build multiple models to map input to output via a number of connected nodes, even if the feature space is hard to linearly separate (Multi-layer Perceptron).

TABLE II. BayesNet Results

BayesNet	Precision		Recall		F-Measure	
Time	Train	Test	Train	Test	Train	Test
0	0.829	0.660	0.825	0.663	0.819	0.657
30	0.916	0.642	0.915	0.640	0.916	0.641
60	0.916	0.636	0.916	0.636	0.916	0.636
90	0.915	0.667	0.915	0.657	0.915	0.655
120	0.910	0.671	0.910	0.665	0.910	0.664
150	0.903	0.678	0.903	0.664	0.903	0.661
180	0.903	0.661	0.903	0.651	0.903	0.645
210	0.913	0.670	0.910	0.666	0.912	0.664
240	0.860	0.687	0.840	0.680	0.848	0.678
270	0.865	0.684	0.857	0.681	0.858	0.68

TABLE III. Naive Bayes Results

Naive	Precision		Recall		F-Measure	
Time	Train	Test	Train	Test	Train	Test
0	0.506	0.503	0.510	0.448	0.422	0.373
30	0.592	0.513	0.617	0.500	0.590	0.486
60	0.595	0.545	0.619	0.524	0.594	0.498
90	0.591	0.595	0.616	0.542	0.589	0.490
120	0.585	0.605	0.615	0.560	0.573	0.526
150	0.575	0.620	0.610	0.565	0.559	0.528
180	0.584	0.624	0.615	0.556	0.566	0.503
210	0.574	0.632	0.593	0.579	0.539	0.537
240	0.533	0.504	0.547	0.506	0.470	0.449
270	0.554	0.575	0.558	0.564	0.494	0.552

TABLE IV. J48 Decision Tree Results

J48	Precision		Recall		F-Measure	
Time	Train	Test	Train	Test	Train	Test
0	0.833	0.721	0.830	0.633	0.830	0.617
30	0.975	0.681	0.975	0.680	0.975	0.680
60	0.973	0.670	0.973	0.670	0.973	0.670
90	0.971	0.686	0.971	0.685	0.971	0.685
120	0.969	0.673	0.968	0.671	0.968	0.671
150	0.964	0.692	0.964	0.688	0.964	0.687
180	0.962	0.700	0.962	0.697	0.962	0.696
210	0.956	0.698	0.956	0.696	0.956	0.696
240	0.872	0.693	0.870	0.683	0.870	0.678
270	0.872	0.695	0.870	0.685	0.870	0.552

TABLE V. MULTI LAYER PERCEPTRON RESULTS

Multi	Precision		Recall		F-Measure	
Time	Train	Test	Train	Test	Train	Test
0	0.710	0.655	0.709	0.648	0.709	0.620
30	0.786	0.720	0.788	0.690	0.853	0.680
60	0.971	0.690	0.971	0.668	0.971	0.662
90	0.816	0.726	0.815	0.704	0.809	0.699
120	0.920	0.745	0.900	0.723	0.920	0.719
150	0.950	0.747	0.915	0.725	0.950	0.721
180	0.925	0.743	0.902	0.720	0.925	0.716
210	0.893	0.746	0.873	0.727	0.863	0.723
240	0.912	0.737	0.892	0.721	0.912	0.717
270	0.850	0.747	0.825	0.729	0.850	0.724

B. Baseline Model Results

Results are provided using standard classification metrics, Precision (a measure of false positives), Recall (a measure of false negatives) and F-measure (a harmonized mean of P and R). We present results tables for two classifiers belonging to each type of model (generative and discriminative), and each table presents the results when training and testing the model using machine activity logs split into incremental time windows (0-270 seconds) with aggregated log results. Note that t=0 is not actually 0 seconds but 5 seconds after the connection is opened. The classifiers used are BayesNet (BN), Naive Bayes (NB), J48 Decision Tree (DT) and Multi Layer Perceptron (MLP).

We can see from the training performance data (see Tables 2-5) that each model exhibits optimal performance between t=30 and t=60, with very little improvement after this time. The low error rates at t=60 in models that consider conditional dependencies in the training phase suggests three things (i) the features we are using to build the models are predictive of malicious behaviour, (ii) malicious activity is probably occuring within the first 60 seconds of the interaction, (iii) there are conditional dependencies between the measured variables. This is a logical result as we would expect certain machine activity to have some conditional dependencies, for example CPU usage and Bytes sent/received.

With reference to Figure 1, a chart of the correctly classified instances over time for the test dataset, the first point of note is that the model that does not consider dependencies between input variables (the NB model) performs much worse than the other models. The model improves over time, but takes until t=210 to reach peak performance, while other models begin to plateau or decrease in performance around t=120 (see Table III). The second point is that the discriminative models outperform the generative models, suggesting that there are distinct malicious activities that are linearly separable from benign behaviour. This means that over time and, most importantly, across different events, we can monitor and measure specific machine behaviours that can be indicative of malicious activity when clicking URLs. We cannot make strong claims about this given that our optimal F-Measure performance was only 0.724 at time t=270 using the MLP model (see Table V). However, it is encouraging to see that the MLP model exhibited a precision performance of 0.720, only slightly below its optimum level, at time t=30. This demonstrates the model's ability to reduce false positives fairly early on in the interaction (i.e. in real time), but still takes some time to improve the false negatives and missed instance of malicious behaviour.

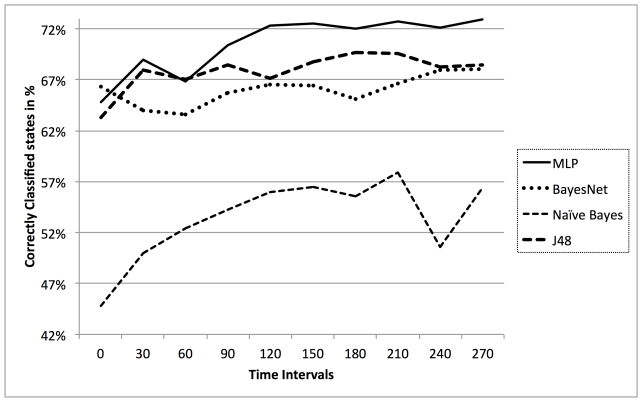

Fig. 1. Classifier performance over time

V. RESULTS

A. Model Analysis

TABLE VI. NODE WEIGHTS BY CLASS

Inputs	Weights (Benign)	Weights (Malicious)
Threshold	-9.13	9.13
Node 1	-9.49	9.49
Node 2	9.61	-9.61
Node 3	17.66	-17.66
Node 4	26.81	-26.81
Node 5	9.45	-9.45
Node 6	1.96	-1.96
Node 7	8.73	-8.73
Node 8	4.34	-4.34
Node 9	-15.85	15.85

Table VII presents weightings assigned to each attribute used as a predictive feature in the MLP model. These results are extracted from the best performing iteration of the MLP model during the training phase (t=60) to examine how the model is representing a learned weighting between features. The model produced 9 hidden nodes and Table VI shows the weighting given to each node for each class (malicious or benign), with 6 of the 9 nodes having values above the threshold value, and nodes 3, 4 and 9 having high weightings towards a particular class. Node 9 stands out as the most discriminative positive weighted node for malicious URLs. If we look further into this, see Table VII, we can identify that Bytes Received variable is the highest weighting. If we look at Node 3 in comparison, which is more heavily weighted towards the benign class, we can see that Bytes sent/received has a similar weighting but that the Packets sent/received is negatively weighted in Node 3 and positively weighted in Node 9. This is an interesting finding as Web endpoints will almost always send data to the machine visiting them. This model demonstrates that there are measurable 'norms' for the inflow of packets from Web pages and that there are measurable deviations from this that can act as predictors of malicious behaviour, as is happening in Nodes 3 and 9.

CPU has weightings higher than most other variables and the threshold for the Node in Nodes 5 and 6, which suggests this is also a predictive feature. It is interesting to note that CPU weighting is at its highest when the network traffic weights are at their lowest, suggesting that CPU is used as a secondary feature when network traffic is not as useful in making a classification decision. Furthermore, ProcessID,

TABLE VII. MLP ANALYSIS

Attribute	Node 1	Node 2	Node 3	Node 4	Node 5	Node 6	Node 7	Node 8	Node 9
Threshold	85.07	42.21	-5.97	1.20	1.39	2.96	-109.11	50.20	12.76
CPU	0.46	0.86	1.18	0.33	3.85	123.46	0.32	-9.13	1.26
Connection	78.79	46.15	-0.22	0.01	-0.02	-14.29	-17.17	11.51	-0.13
PortNumber	5.17	0.23	0.28	0.31	0.41	0.68	0.14	-0.56	-0.52
ProcessID	-1.10	-0.23	-0.25	-0.18	-0.73	33.42	-0.27	10.33	0.56
RemoteIP	78.77	46.12	-0.17	0.03	-0.03	-14.28	-17.13	11.44	-0.08
NIC=1	-8.72	-14.07	5.97	1.55	-1.40	-11.06	78.26	-30.34	-13.12
NIC=2	-73.87	-35.44	6.54	-1.31	-0.96	6.28	68.90	-48.84	-13.17
NIC=3	-74.14	-35.31	6.13	-1.23	-1.31	6.68	68.95	-49.39	-13.12
NIC=4	-7.43	-13.27	5.86	1.72	-1.56	-10.79	77.76	-29.36	-12.96
NIC=5	-180.28	-69.52	-6.72	-7.22	0.78	-1.13	105.03	-14.35	15.05
NIC=6	-6.48	-13.41	5.93	1.61	-1.46	-11.12	77.80	-29.73	-13.06
NIC=7	-74.16	-30.01	6.26	-1.36	-1.18	6.31	69.09	-49.04	-13.42
BytesSent	894.02	516.07	27.17	35.32	75.38	-13.97	-866.89	392.79	13.09
BytesReceived	-29.16	-68.52	125.22	-4.11	150.17	-2.69	98.62	-47.79	88.93
PacketsSent	-52.06	-49.15	-11.49	-2.73	32.62	-3.06	104.37	-50.02	18.38
PacketsReceived	-59.21	-45.36	-10.36	-1.34	20.60	-2.07	106.07	-49.84	14.90

which records the maximum process ID on the machine, has its highest weighting in the same Node as CPU (Node 6). This can be interpreted as malware creating more processes on the machine and pushing up CPU usage. The Connection attribute, which measure the presence of a remote endpoint connection is highly weighted in Node 1, which is weighted towards the malicious class. At the same time, the identification of a remote endpoint address (RemoteIP) is at its highest, and the BytesSent attribute is extremely high, suggestive of an attack based on data exfiltration (possibly advanced persistent threat) across the ethernet network interface (NIC=5).

B. Sampled Learning

Finally, we investigate how much training data is required to train an MLP model. Storing Twitter data around ongoing real-world events is an issue given that events can last several weeks. Furthermore, if the system were deployed it could run on a daily basis to monitor malware and retrain learned models. Less data means less storage space and less computational time required to extract model features and run models. In addition, interacting with URLs is a time-intensive process. Questions could be asked around whether the training set is missing a significant proportion of malicious activity given that not all URLs can be visited in real-time with the relatively low level of compute resources available to academic researchers. Malicious endpoints are frequently taken down and are not accessible after short periods of time following their deployment in drive-by-downloads. Demonstrating that a small training sample achieves similar performance to the full samples alleviates this to some degree as it demonstrates the most explanatory features are present in the smaller sample. We retained the full test dataset and sampled (using no replacement) from the training data at 1%, 5%, 10% and increments of 10% up to 100. Figure 2 illustrates the percentage of correctly classified instances with a 100% sample, down to 1% and shows that a sample of 20%, 30% and 40% yields a performance of 68%, only 1% lower than the optimal performance of 69%. The performance using a 1% sample is 63% - a performance drop of only 5% on the optimal performance with a complete sample. These results are based on the mean of two runs for each sample, but it is worth noting that both runs yielded almost identical results (e.g. for the 1% sample, run 1 was 63.3% and run 2 was 63.4%)

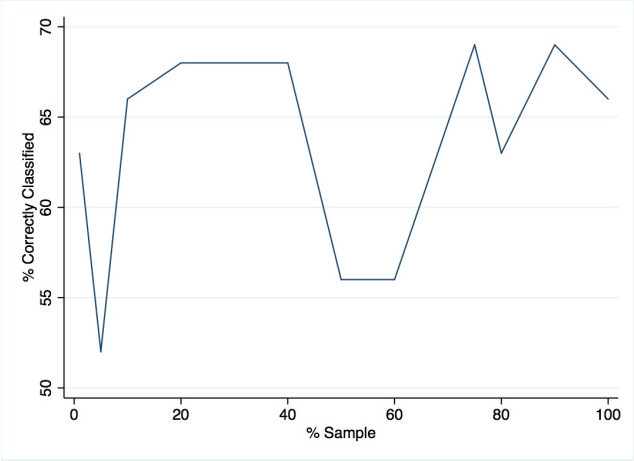

Fig. 2. Correctly classified instances with sampled training data

VI. DISCUSSION AND CONCLUSIONS

We collect data from Twitter, using event-specific hashtags to sample tweets posted at the time of two real-world sporting events. Sporting events have been shown to produce large volumes of microblog posts and within these it is common for people to include URLs that point to additional information, such as news articles, images and other sources of evidence to support the short (max 140 character) post. Given the limited space in a tweet, Twitter automatically shortens the URL so the endpoint is not clear from the text. This makes tweets particularly susceptible to drive-by-downloads – Cyber attacks that direct users to a malicious endpoint that proceeds to attack users' machines. Given the surreptitious nature of these attacks, the information seeking nature of people at the time of large events, and the growth of Twitter as a news reporting and citizen journalism platform, we sought to develop a machine classifier that could classify a URL as malicious or benign in 'real-time', i.e. within seconds of clicking the link. We attempted to achieve this using machine activity log data, such as CPU usage, network traffic and network connection statistics, as features that could discriminate between malicious and benign URLs. We also aimed to explicate the relationship between machine activity and malicious system behaviour surrounding Twitter event coverage. Finally, given the large volumes of tweets surrounding events and potential issues with

sampling from all malicious sites due to their short lifespan and the time intensiveness of interacting with all of them, we aimed to understand the impact on classification performance when using much smaller samples.

We built a number of machine classifiers and identified that a Bayesian model, a Decision Tree (DT) and a Multi Layer Perceptron (MLP) approach all worked extremely well during training, achieving over 90% accuracy, up to 97% for the DT and MLP. However, discriminative models performed better than generative models in testing, and the MLP model performed best overall with accuracy of up to 72% using previously unseen data. The Bayesian approach performed best in the early stages of the interaction (within 5 seconds of clicking the URL), achieving 66% accuracy when the model had the least information available. The high training scores suggests the features used are indeed predictive of malicious behaviour for a single event. The drop in performance on a new event suggests attack vectors are slightly different across events, but with a reasonably high degree of accuracy we can claim some independence between predictive features and events, though this should be tested in future with additional events beyond sports and within everyday 'mundane' blog posts to add further weight to this claim.

Upon inspecting the decision-making process within the MLP model, we found evidence to suggest that the key predictive machine activity metric was network activity - particularly packets sent and received. CPU use and process IDs also had a clear raised and correlated weighting in the model, as did the bytes sent from the network when correlating with new connections to remote endpoints, suggesting data exfiltration exercises can be distinguished from general data transfer.

A learning curve produced using a range of samples from the training dataset, while still testing on the full testing dataset, revealed only a small drop in classification performance when compared to using the full training sample. This suggests machine log data can be predictive of malicious system behaviour even with very small samples, alleviating some concerns over appropriate sampling mechanisms, lack of a complete log of all Twitter URL activity, and the requirement for large amounts of data storage. However, this was a binary classification task and if we aimed to further classify malware into different types or families based on its behaviour, which we intend to in future, we anticipate that the sampling requirements may be different.

Twitter have recently introduced new policies to protect their users against harm and it would appear there is a need for an automated and reliable approach for an alert or warning system to help detecting malicious URLs in the waves of micro posts that surround real-world events. This work presents such an approach and provides some insight into using machine activity data to predict malicious behaviour within seconds of interacting with a URL.

Acknowledgment

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) Global Uncertainties Consortia for Exploratory Research in Security (CEReS) programme, grant number: EP/K03345X/1.

References

[1] Y. Alosefer and O. Rana. Honeyware: a web-based low interaction client honeypot. In *Software Testing, Verification, and Validation Workshops (ICSTW), 2010 Third International Conference on*, pages 410–417. IEEE, 2010.

[2] P. Burnap, N. J. Avis, and O. F. Rana. Making sense of self-reported socially significant data using computational methods. *International Journal of Social Research Methodology*, 16(3):215–230, 2013.

[3] P. Burnap, M. L. Williams, L. Sloan, O. Rana, W. Housley, A. Edwards, V. Knight, R. Procter, and A. Voss. Tweeting the terror: modelling the social media reaction to the woolwich terrorist attack. *Social Network Analysis and Mining*, 4(1):1–14, 2014.

[4] D. Canali, M. Cova, G. Vigna, and C. Kruegel. Prophiler: a fast filter for the large-scale detection of malicious web pages. In *Proceedings of the 20th international conference on World wide web*, pages 197–206. ACM, 2011.

[5] M. Cova, C. Kruegel, and G. 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 '10, pages 281–290, New York, NY, USA, 2010. ACM.

[6] M. Faghani and H. Saidi. Malware propagation in online social networks. In *Malicious and Unwanted Software (MALWARE), 2009 4th International Conference on*, pages 8–14, Oct 2009.

[7] A. Kapravelos, Y. Shoshitaishvili, M. Cova, C. Kruegel, and G. Vigna. Revolver: An automated approach to the detection of evasive web-based malware. In *USENIX Security*, pages 637–652, 2013.

[8] H. Kwak, C. Lee, H. Park, and S. Moon. What is twitter, a social network or a news media? In *Proceedings of the 19th international conference on World wide web*, pages 591–600. ACM, 2010.

[9] V. L. Le, I. Welch, X. Gao, and P. Komisarczuk. Two-stage classification model to detect malicious web pages. In *Advanced Information Networking and Applications (AINA), 2011 IEEE International Conference on*, pages 113–120, March 2011.

[10] C.-H. Lee. Unsupervised and supervised learning to evaluate event relatedness based on content mining from social-media streams. *Expert Systems with Applications*, 39(18):13338 – 13356, 2012.

[11] K. Lee, J. Caverlee, and S. Webb. Uncovering social spammers: Social honeypots + machine learning. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, pages 435–442, New York, NY, USA, 2010. ACM.

[12] S. Lee and J. Kim. Warningbird: Detecting suspicious urls in twitter stream. In *NDSS*, 2012.

[13] K. Lerman and R. Ghosh. Information contagion: an empirical study of the spread of news on digg and twitter social networks. *CoRR*, abs/1003.2664, 2010.

[14] J. Martinez-Romo and L. Araujo. Detecting malicious tweets in trending topics using a statistical analysis of language. *Expert Systems with Applications*, 40(8):2992 – 3000, 2013.

[15] Microsoft. Security intelligence report. December 2013.

[16] Y. min Wang, D. Beck, X. Jiang, R. Roussev, C. Verbowski, S. Chen, and S. King. Automated web patrol with strider honeymonkeys: Finding web sites that exploit browser vulnerabilities. In *In NDSS*, 2006.

[17] J. Nazario. Phoneyc: A virtual client honeypot. In *Proceedings of the 2Nd USENIX Conference on Large-scale Exploits and Emergent Threats: Botnets, Spyware, Worms, and More*, LEET'09, pages 6–6, Berkeley, CA, USA, 2009. USENIX Association.

[18] R. Procter, F. Vis, and A. Voss. Reading the riots on twitter: methodological innovation for the analysis of big data. *International journal of social research methodology*, 16(3):197–214, 2013.

[19] M. Puttaroo, P. Komisarczuk, and R. Cordeiro De Amorim. Challenges in developing capture-hpc exclusion lists. In *Security of Information and Networks (SIN), 2014 7th International Conference on*, September 2014.

[20] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, pages 851–860. ACM, 2010.

[21] A. Sanzgiri, J. Joyce, and S. Upadhyaya. The early (tweet-ing) bird spreads the worm: An assessment of twitter for malware propagation.

Procedia Computer Science, 10(0):705 – 712, 2012. {ANT} 2012 and MobiWIS 2012.

[22] C. Seifert and R. Steenson. Capture-hpc. *Internet: https://projects. honeynet. org/capture-hpc*, 2008.

[23] G. Stringhini, M. Egele, C. Kruegel, and G. Vigna. Poultry markets: on the underground economy of twitter followers. In *Proceedings of the 2012 ACM workshop on Workshop on online social networks*, pages 1–6. ACM, 2012.

[24] G. Stringhini, C. Kruegel, and G. Vigna. Detecting spammers on social networks. In *Proceedings of the 26th Annual Computer Security Applications Conference*, ACSAC '10, pages 1–9, New York, NY, USA, 2010. ACM.

[25] K. Thomas and D. Nicol. The koobface botnet and the rise of social malware. In *Malicious and Unwanted Software (MALWARE), 2010 5th International Conference on*, pages 63–70, Oct 2010.

[26] A. Tumasjan, T. O. Sprenger, P. G. Sandner, and I. M. Welpe. Predicting elections with twitter: What 140 characters reveal about political sentiment. *ICWSM*, 10:178–185, 2010.

[27] A. Wang. Machine learning for the detection of spam in twitter networks. In M. Obaidat, G. Tsihrintzis, and J. Filipe, editors, *e-Business and Telecommunications*, volume 222 of *Communications in Computer and Information Science*, pages 319–333. Springer Berlin Heidelberg, 2012.

[28] W. Webberley, S. Allen, and R. Whitaker. Retweeting: A study of message-forwarding in twitter. In *Mobile and Online Social Networks (MOSN), 2011 Workshop on*, pages 13–18. IEEE, 2011.

[29] C. Yang, R. Harkreader, J. Zhang, S. Shin, and G. Gu. Analyzing spammers' social networks for fun and profit: A case study of cyber criminal ecosystem on twitter. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 71–80, New York, NY, USA, 2012. ACM.

[30] S. Yardi, D. Romero, G. Schoenebeck, and danah boyd. Detecting spam in a twitter network. *First Monday*, 15(1), 2009.