The Early Bird Gets the Botnet: A Markov Chain Based Early Warning System for Botnet Attacks

Zainab Abaid*†, Dilip Sarkar‡, Mohamed Ali Kaafar† and Sanjay Jha*
*School of Computer Science and Engineering, University of New South Wales, Australia
{zainaba,sanjay}@cse.unsw.edu.au
†CSIRO Data61, Australia
dali.kaafar@data61.csiro.au
‡Computer Science Department University of Miami, USA
sarkar@miami.edu

***Abstract*—Botnet threats include a plethora of possible attacks ranging from distributed denial of service (DDoS), to drive-by-download malware distribution and spam.**

While for over two decades, techniques have been proposed for either improving accuracy or speeding up the detection of attacks, much of the damage is done by the time attacks are contained. In this work we take a new direction which aims to predict forthcoming attacks (i.e. before they occur), providing early warnings to network administrators who can then prepare to contain them as soon as they manifest or simply quarantine hosts. Our approach is based on modelling the Botnet infection sequence as a Markov chain with the objective of identifying behaviour that is likely to lead to attacks. We present the results of applying a Markov model to real world Botnets data, and show that with this approach we are successfully able to predict more than 98% of attacks from a variety of botnet families with a very low false alarm rate.

I. Introduction

Botnet generated attacks such as high volume DDoS can cause network outages resulting in financial losses to companies. Likewise, information stealing and phishing can compromise users' privacy and leak valuable financial information. The potent danger of these attacks is such that it is not sufficient to merely detect them; as by the time an attack has been detected and mitigated, much of the damage may have been done. Instead if these attacks can be predicted before they occur, preventative measures can be taken so that they never materialise. If a host is known to be at high risk of engaging in an attack soon, it can be closely monitored, quarantined or taken off the network.

Existing work deals with the problem of detection and speedy response to attacks. Predicting attacks however has received little attention so far. In this work we propose a Markov chain based approach for predicting attacks before they occur. We postulate and verify that Botnet infections show distinguishable malicious behaviour other than just attacks. Earlier studies have shown that infections often begin with a social engineering attack or an exploit on a victim host, leading to the download of a malicious binary, via a drive-by download or other mechanism. If executed, the binary may then engage in an attack, communicate with a C&C server or download further updates. Given that attacks occur within this context, it is then possible to fit a model to the behavioural stages of an infected host before and after it engages in an attack. Once we have such a model, we can then predict when attacks will occur within a hosts' behavioural sequence, allowing us to generate *early attack warnings*.

We identify the typical stages of a Botnet behaviour and model the stages as states of a Markov chain. We train the model to identify the likelihood of transitions between the states and propose a methodology to predict future states based on currently observed behaviour. Using real network traces from a variety of Botnet families, we predict predict attacks with 98% accuracy. Perhaps more importantly, we show that our methodology can even be extended to predict other behaviour of interest, such as a bot's "phone-home" communication with its control server. The key novelty of our work is that we move beyond merely identifying an infection; we are able to identify and predict the specific behavioural stage that an infection has reached, and at any given time calculate the probability that it will start to attack other entities.

The remainder of this paper is organised as follows. Section II outlines related work in this domain. In Section III we define some key terminology and properties of Markov chains. In Section IV we describe our model, instantiate it with Botnet data and present the theoretical and empirical verification that it represents a valid Markov chain. Section V presents our attack prediction approach and experimental results. In Section VI we outline the limitations of this work and future directions. Section VII concludes the paper.

II. Related Work

While many solutions have been proposed for identifying specific kinds of Botnet attacks, a general solution that is not restricted to particular attacks remains elusive. In developing such a solution we draw inspiration from two classes of existing work: generic, behaviour-based approaches to detecting Botnet infections, and the use of Markov chains and Hidden Markov Models (HMMs) in intrusion detection.

Traditional attack detection methods monitor the characteristics of network traffic during an attack, a classic example being port scan detection based on features such as the number of failed outgoing connections [13], [19]. In similar vein is detection of outgoing DDoS attacks based on the

entropy of destination addresses at network routers [21] or of incoming DoS attacks based on the arrival pattern of incoming packets [22]. SPOT [6] uses a content filter to examine the sequence of outgoing messages from each host to identify spam generation. As all of these methods aim to identify traffic that is generated during attacks, they cannot be used for attack prediction. While there is little work in predicting attacks, one recent scheme [11] uses the increase in traceroute packets in the network prior to a target link DDoS attack to predict the attack itself, essentially detecting the preparation stage before the attack happens. This approach however is restricted to a specific attack; to the best of our knowledge, no general attack prediction solution exists.

In our quest for generality, we draw on the idea that Botnet infections follow a typical behavioural sequence. Several works have presented an analysis of this sequence, often termed as the bot lifecycle, and proposed approaches to identify individual stages within this lifecycle – hosts are classified as infected based on the number or sequence of stages they show [8]–[10], [14]. For example, BotHunter [9] is an industry-standard Botnet detection tool that employs various heuristics to detect lifecycle stages and performs rule-based analysis to decide whether they constitute a Botnet infection. While this approach forms an inspiration for our work, we take it a step further and use the identification of behavioural stages to predict future malicious behaviour (for example, attacks) rather than simply identify an infection.

Markov models have been used in intrusion and anomaly detection research in the past. An example of a Markov chain based intrusion detection system (IDS) is found in [24] which uses a stream of audit events to train a Markov chain; event streams generated by hosts are tested for similarity with the model and are classified as attack streams if they are similar to the model. Such approaches allow attack identification but not prediction, and to the best of our knowledge, Markov chains have not been used in predicting future behaviour of known infected hosts.

HMMs have been applied often in Botnet detection research; the general idea is to represent some aspect of Botnet behaviour as a sequence of symbols, and build a model in which the states are assumed to emit those symbols [7]. For example [16] maps values of inter-packet delay to a language of four symbols to generate a four-state HMM, which is trained on C&C communication traces of a Botnet. The similarity of observed data to the model is computed to identify Botnet communication. A similar approach has used SNMP (Simple Network Management Protocol) variables as the symbol sequence [23].

Possibly the closest inspiration for our research is another HMM approach [15] which uses the set of bot lifecycle stages as the symbol sequence combined with a "clean" symbol indicating absence of infection. The HMM is used to compute the most likely state of the host given the observed sequence. However the objective of this work is to identify infection rather than to predict attack or any other behaviour, and such prediction ability is not investigated in [15]. In fact, to the best of our knowledge there is no existing approach, whether using Markov chains or otherwise, that addresses the problem of providing an early detection and prediction of Botnet attacks and that is general in nature, rather than solutions restricted to specific attacks.

III. Some Terminology for Markov Chains

Markov chains are used to model systems that randomly move among a set of states. A Markov chain is represented as a state transition diagram, with the paths between the states weighted by the probability of moving from one state to another. In this work we use a discrete time Markov chain, where the assumption is that transitions from one state to another happen at discrete time steps; each transition from a state S_i to a state S_j has a probability $t_{i,j}$ of occurrence.

We briefly define some terminology that we later refer to in developing and verifying our model. An *irreducible* Markov chain is one in which all states can communicate - i.e. for each pair of states S_i and S_j, there exists a path from S_i to S_j and vice versa. A state is *periodic* if it can only be visited at time values that are multiples of an integer greater than 1, for example at time $t = 3, 6, 9$ and so on. States not exhibiting this property, such as those with a self-transition, are *aperiodic*, and a chain for which all states are aperiodic is itself called an aperiodic chain. For an irreducible Markov chain, it is possible to define a *stationary distribution* that defines the long-term probabilities of being in each of the states, i.e. $P(S_i)$ for all states S_i as time grows large. This distribution should satisfy $P_j = \sum_{i \in S} P_i t_{i,j}$ and $\sum_{i \in S} P_i = 1$, where S is the set of states in the model, P_i is the stationary probability of being in state S_i and $t_{i,j}$ is the probability of going from state i to state j. If the chain is also aperiodic, then this stationary distribution is also the *limiting distribution* for the Markov chain. Finally, a *reversible* Markov chain has to satisfy for all pairs of states S_i and S_j the property $P_i t_{i,j} = P_j t_{j,i} \forall i, j$ where P_i and $t_{i,j}$ are as defined above.

IV. Modelling Botnet Infections as a Markov Chain

We present our model to capture the states of the Botnet infection cycle, and introduce datasets in use to instantiate and validate our model.

A. Proposed Model

We now develop a Markov chain model to describe Botnet infections. Prior studies [4], [9], [20] have suggested that bots follow a well-defined sequence, beginning with a social engineering attack or an incoming exploit, which if successful will result in a malicious binary download, followed by communication with a C&C server and then launch of an outgoing attack, such as outbound propagation, spam generation, port scanning or DoS attacks. Outgoing attacks therefore occur within a context that can be modelled; we capture this context as a state transition diagram, shown in Fig. 1, with each state representing a different stage of infection.

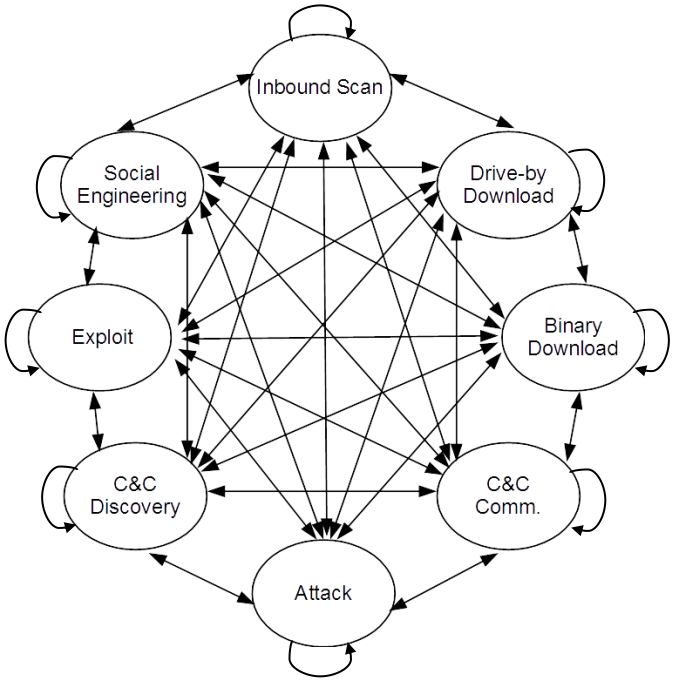

Fig. 1: State transition diagram modelling botnet infection stages.

We have based our state model on the various models presented in earlier studies and on the typical behavioural stages exhibited by a wide range of modern Botnets, including mobile Botnets. Table I describes the behaviour each state in our model represents. We also exemplify each of the behavioural stages with a variant of a recent banking trojan called Dridex, mapped to the relevant states of our model. We derive this information from an analysis of a Dridex variant (`Dridex 120`) published by CISCO [3]. We believe that our model provides comprehensive coverage of behaviour exhibited by a typical botnet infection.

Note that unlike previous work, in which states are linked in a logical flow that the infection should proceed in, we link the states in a full mesh. This is to avoid pre-defining what the sequence of steps in a bot infection should be. In the Botnet traces used in our work (Section IV-C), we have seen many instances of Botnet stages appearing out of sequence, for example, an exploit occurring again after C&C communication, rather than only once in the beginning. We design our model to also account for all such cases of unusual behaviour.

B. Current Model Limitations

Applying the comprehensive model of Fig. 1 entails being able to detect activities falling within each state of the model. To the best of our knowledge, this kind of detection is not possible with existing tools. Most detection tools only provide the user with a final detection decision. Some rule-based tools, for example Suricata [2], Bro [17] and Snort [18], are capable of providing a break-down of the specific malicious behaviour detected, but at present there is no ruleset publicly available with these tools that contains detection logic for all the behavioural stages in our model.

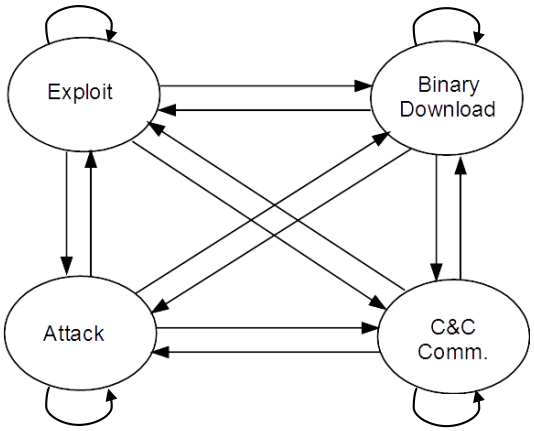

Fig. 2: Limited state transition model used in our current work.

Designing such a comprehensive detection tool is beyond the scope of this work. Therefore, in this work we use a more limited model whose states can be detected using currently available tools, and leave the more comprehensive model as future work. We use Snort as our detection tool, with Botnet detection rulesets from Emerging Threats as well as rules from BotHunter, an industry-standard Botnet detection IDS, a combination that allows us to detect a greater variety of malicious activities than any other existing tool. Based on our limited detection ability, we redefine our state model as shown in Fig. 2. We remove the drive-by download, social engineering and C&C discovery states as these are beyond our current detection capability; likewise we remove the inbound scan as it is not applicable to our current dataset. For the remainder of this paper, we work with this limited model. However, all our methodology is applicable as it is to the full model, including the calculations and the proofs of validity that follow in the next section; with an enhancement in the detection capability of the IDS used, our approach can therefore be applied as it is.

C. Datasets

We use the following two datasets to instantiate our model.

1) The SysNet Dataset: The SysNet trace has been generated by the SysNet lab [1] in July 2013 and contains traffic from ten infected hosts. It was generated by running ten bot binaries in separate virtual machines: four variants of Pushdo, two variants of Sality, Kolabc, Virut, Dorkbot, and Bobax. This covers HTTP, IRC, and P2P-based bots that engage in a range of attacks including sending spam and outbound scanning. Full packet traces from the virtual machines were captured for 24 hours using Wireshark on host machines.

2) The ISCX Botnet Dataset: This trace was collected by researchers at the ISCX Laboratory at UNB, Canada and made available publicly. It contains traces of 30 infected machines, nearly half of which are infected by IRC Botnets and the remainder by a variety of P2P and HTTP Botnets, including variants of Zeus, Virut, NSIS, Storm, and Zero Access among others. Interested readers are referred to [5] for details of the

TABLE I: Description of states of the Markov Chain and mapping `Dridex 120` behaviour to each state (where applicable).

State	Description of Behaviour	Example of Dridex Behaviour
Inbound Scan	An incoming port scan on any port of the host, characterised by the same port being scanned on many hosts (horizontal scan) or many ports being scanned on one host (vertical scan).	N/A
Social Engineering	Human users are tricked into compromising on usual security practices, e.g. clicking on an intriguing (but malicious) spam link in an email message.	Victim receives email with a malicious Microsoft Word or Excel attachment.
Drive-by Download	Malicious executables are downloaded without the user's consent during a web browsing session.	N/A
Exploit	Any attack targeting application or OS vulnerabilities for gaining "backdoor" access or remote execution privileges on the victim machine, or web browser.	N/A
Binary Download	The actual download of a malicious piece of code or an executable, which may be disguised as legitimate software or an image or document.	User opens the attachment, triggering execution of embedded macro; intermediate dropper downloaded which then downloads and runs the actual binary.
C&C Discovery	When the downloaded binary is run and attempts to contact the botnet's C&C server. Discovery behaviour may include contacting a list of servers one by one over HTTP, opening multiple P2P connections, or a large number of DNS queries for a list of possible domains.	Sends an HTTP Post request to a specific hard-coded IP address.
C&C Communication	Exchanges with the discovered its C&C server, over IRC, HTTP, or in some cases, custom protocols.	If HTTP POST is successful, bot receives configuration and instructions regarding which websites to target for redirection attacks.
Outgoing Attack	Either self-propagation by the botnet e.g. through port scanning or mass-emailing itself, or other attacks such as information stealing, phishing, DoS or spam.	When victim visits a banking website included in the configuration file, the bot can perform redirection to a phishing website and credential stealing.

trace. The durations of the traces range from a few minutes up to 3 days.

Both datasets are in the form of full packet traces. For training and testing the Markov chain, the traces first had to be mapped to state sequences comprising the states in our model, i.e. exploit, binary download, C&C communication and attack. We currently relied on Snort for detecting the behaviour represented by these states, using Botnet detection rulesets from Emerging Threats and BotHunter. Snort performs deep packet inspection over the traces and generates alerts when a rule is triggered. We post-processed the Snort logs to classify the alerts as belonging to one of the states in our model. We then generated a timestamped sequence of states observed in each host's network traffic. A snapshot of the data used to instantiate our model is shown in Table II. The timestamp represents minutes from start of the trace; for example, `"2:CNC"` indicates that a C&C Communication event was detected at time 2 minutes.

D. Instantiating the Markov chain

1) Calculating Transition and Limiting Probabilities: For deriving the limiting distribution, we use the flow balance condition of a Markov model, i.e. the probability of leaving a state must equal the probability of entering it. This allows us to formulate a balance equation for each state of the model. In general for each state i that has incoming transitions from a set of n states k and has outgoing transitions to a set of m states j:

$$\sum_{k=1}^{n} P_k t_{k,i} = \sum_{j=1}^{m} P_i t_{i,j} \tag{1}$$

Based on this we define the following flow balance equations for each of the states in our model from S_1 to S_4 respectively:

$$P_2 t_{2,1} + P_3 t_{3,1} + P_4 t_{4,1} = P_1 t_{1,2} + P_1 t_{1,3} + P_1 t_{1,4} \tag{2}$$

$$P_1 t_{1,2} + P_3 t_{3,2} + P_4 t_{4,2} = P_2 t_{2,1} + P_2 t_{2,3} + P_2 t_{2,4} \tag{3}$$

$$P_1 t_{1,3} + P_2 t_{2,3} + P_4 t_{4,3} = P_3 t_{3,1} + P_3 t_{3,2} + P_3 t_{3,4} \tag{4}$$

$$P_1 t_{1,4} + P_2 t_{2,4} + P_3 t_{3,4} = P_4 t_{4,1} + P_4 t_{4,2} + P_4 t_{4,3} \tag{5}$$

where $P_i \; \forall \; i_{1..4}$ is the limiting probability of being in each state, and $t_{i,j}$ is the transition rate from state i to state j. In order to calculate the values P_i for each state, we first need transition probabilities $t_{i,j}$. Therefore we first generate a transition probability matrix T from the datasets with each cell $[i, j]$ representing the transition rate between states S_i and S_j. A common approach to calculate the transition probability matrix is to build a training sequence of states by observing the system for some length of time, and then generate a probability for $T_{i,j}$, i.e. the transition from each state i to each state j as follows:

$$\frac{\sum T_{i,j}}{\sum\limits_{k \in S} T_{i,k}} \tag{6}$$

TABLE II: Example of event sequences generated for Markov chain.

IP	Event Sequence
a.a.a.a	2:CNC, 3:CNC, 4:ATTACK, 7:EXPLOIT, 9:CNC...
b.b.b.b	0:EXPLOIT, 6:EXPLOIT, 7:BINARY, 19:ATTACK...
c.c.c.c	1:ATTACK, 2:ATTACK, 6:EXPLOIT, 7:BINARY...
d.d.d.d	1:CNC, 2:ATTACK, 6:CNC, 7:EXPLOIT, 15:CNC...

TABLE III: Stationary probability distribution for dataset.

State	Probability
Exploit	$P_1 = 0.0025$
Binary Download	$P_2 = 0.0015$
CNC Communication	$P_3 = 0.5739$
Attack	$P_4 = 0.4222$

where $\sum T_{i,j}$ represents the total number of transitions observed from state S_i to S_j. For our model, T is therefore a 4×4 matrix and contains the following values:

$$T = \begin{pmatrix} 0.682 & 0.030 & 0.033 & 0.254 \\ 0.035 & 0.426 & 0.527 & 0.012 \\ 0.0001 & 0.001 & 0.926 & 0.073 \\ 0.001 & 0.00001 & 0.099 & 0.899 \end{pmatrix}$$

The states represented by each row and column are Exploit, Binary Download, C&C Communication and Attack, respectively. Using the values from the matrix T, and replacing Equation 5 above with the conservation of total probability equation ($\sum_{i \in S} P_i = 1$) allows us to solve for P_i.

For our dataset we obtain the stationary distribution shown in Table III.

2) Verifying Markov Chain Properties: We now discuss and verify the properties of irreducibility, aperiodicity and reversibility of the Markov chain, and then empirically verify that the stationary probability distribution that we have obtained also fulfils the conditions for a Markov stationary distribution.

Showing our Markov chain to be irreducible requires no proof. As discussed earlier, irreducibility simply refers to a chain where all states can communicate with each other. Fig. 1 shows that the states in our model are connected in a full mesh so this is indeed the case. Also from the figure it is apparent that all states in our model have self-transitions; therefore the Markov chain is aperiodic. Note that this property is important as we label the stationary distribution as the limiting distribution for this chain, which can only be valid if the chain is aperiodic. Reversibility requires that for all states S_i, S_j, $P_i t_{i,j} = P_j t_{j,i}$. Using the calculated limiting probabilities and the matrix T, we verify empirically that this holds for all states in the model. Finally, we verify empirically that the stationary probability distribution holds true to both conditions specified in Section III, i.e. the conservation of total probability and the condition $P_i = \sum_{j \in S} P_j t_{j,i}$. The calculations are straightforward and we do not show them here.

V. Empirical Analysis and Experimental Evaluation

We now investigate the accuracy with which we are able to predict attacks using the Markov chain[1]. Our basic methodology is as follows. To make predictions, we first learn a transition probability matrix from training data, similar to the matrix shown in Section IV-D. Then, given a set of states S, the current state S_i, and a set of transition probabilities $t_{i,j} \forall j \in S$, we predict the next state to be the state S_j such that the transition probability $P(S_i \rightarrow S_j) = \max_{j \in S} t_{i,j}$. This means that we predict the attack state whenever it is the destination of the most likely outgoing transition from the current state. We make this prediction at every time-step. Before we further detail the experiment methodology and present the results, we discuss an important issue: whether or not to consider self-transitions in the data.

A. An Empirical Analysis of Self-Transitions in the Dataset

We first discuss why self-transitions arise in the data. As the event stream is generated from an IDS, multiple consecutive alerts for the same state may occur. For example, a host continuously engaged in C&C communication with a blacklisted server will generate a stream of alerts as it repeatedly makes connections to the server; we interpret this as a sequence of self-transitions from the C&C Communication state, and observe that such cases occur frequently in our dataset.

Therefore, we find that when we build a transition matrix from the original dataset, as shown in Section IV-D, the self-transition probabilities for the C&C Communication and Attack states overwhelm the probabilities of transitions leading to other states. C&C Communication has a self-transition probability of 0.926 and Attack has a self-transition probability of 0.90. This has a negative impact on prediction; as we predict the most likely transition from the C&C state for example, we end up always predicting the self-transition. In fact, the only time we can possibly predict attack is when the current state is the attack state. For practical purposes this is not useful, as we need to know when a non-attack state will transition to attack, not whether the attack state will remain in itself. Thus we had to consider whether dropping self-transitions altogether, by considering each set of consecutive alerts as a single alert, is a viable option.

We argue that for the attack state, it does not matter whether or how long it self-transitions. Firstly, we are interested in predicting the first entry to the attack state from a different state, not subsequent self-transitions within the attack state. Secondly, we are only concerned with predicting the *occurrence* of attack events and not the *duration*. Once the host has engaged in an attack, it no longer matters how long it will remain in the attack state. Therefore we choose to ignore self-transitions of the attack state in our data.

The next question is whether to ignore self-transitions in other states too. For example, if the current state is the C&C Communication state, we can predict whether the next *state change* is likely to be to the attack state. However should we be

[1]In the scope of this work, "attack" can refer to sending spam emails or performing outbound scans on other machines

TABLE IV: Statistical properties of duration (in minutes) of self-transitions for each state.

State	Min.	Max.	Mode	Mean	Stdev.
Exploit	< 1	6	< 1	1.3	2.2
Binary Download	< 1	14	< 1	0.75	3.12
CNC Comm.	< 1	1437	< 1	2.7	47.1
Attack	< 1	1420	< 1	5.6	63.2

TABLE V: Statistical properties of the number of self-transitions for each state.

State	Min	Max	Mode	Mean	Stdev
Exploit	2	70	3	16.8	24.1
Binary Download	2	16	2	4.6	4.6
CNC Comm.	2	4266	2	18.1	93.2
Attack	2	23003	2	37.4	694.7

TABLE VI: Accuracy analysis of predicting the attack state in the test data.

True Positives	98.3%
False Positives	1.3%
True Negatives	98.7%
False Negatives	1.7%
Overall Accuracy	98.5%

TABLE VII: Accuracy analysis of predicting the C&C Communication state in the test data.

True Positives	99.8%
False Positives	1.7%
True Negatives	98.3%
False Negatives	0.02%
Overall Accuracy	99%

concerned about how long the C&C Communication state will self-transition before it goes to the attack state? We conclude that this value is indeed important if there is a certain pattern in non-attack states' self-transitions before they go to attack; for example, we may find that the C&C Communication state mostly remains in self-transitions for some x minutes before it transitions to the attack state. In this case we would find it useful to generate a warning that an attack is likely to happen x minutes later whenever we see the C&C Communication state. To see whether such a pattern exists, we perform an empirical analysis of the durations of the self-transitions of each state in the data.

Table IV and V show the statistical properties of the durations (in terms of number of minutes) and number (i.e. number of consecutive alerts) respectively of self-transitions of each state. We find that there is no predictable pattern in either duration or number of transitions. The tables show that the standard deviation of both the duration and number of self-transitions is very high compared to the mean for almost all states, and the difference between the minimum and maximum is also generally very large. We do a similar analysis for each individual host, not shown here owing to space constraints, but find that even within individual hosts infected by a single bot, the pattern is unpredictable, with values of both duration and number widely scattered.

Table IV shows that the minimum duration of each state is less than 1 minute, and this is also the modal value for all states. This, combined with the fact that the actual duration is unpredictable owing to its high variance, leads us to conclude that whenever we predict an attack following another state, we should conservatively always predict that it is likely to follow in less than a minute. The actual duration after which it will follow then becomes irrelevant. The downside of this approach is that when an attack does not in reality follow within a few minutes, but rather after many hours, valuable resources will be wasted in monitoring the suspected host for a long time, or its communications will be unnecessarily restricted. However we find this acceptable because long durations of self-transitions are uncommon. Out of a total of 1233 occurrences of the attack state in our dataset, only 27 times the duration is over 2 minutes in length. Similarly only 170 out of 3659 occurrences of the C&C Communication state last longer than 2 minutes. Therefore, although self-transitions are part of our original model, we now decide to ignore all self-transitions in the data. In the training stage, we build our transition matrix from a dataset where we collapse all self transition sequences into a single state, i.e. a number of consecutive alerts for a state will be considered a single alert. In the testing stage, when we make a prediction from a state, we ignore further alerts for the same state, until there is a state change, at which point we make another prediction.

We now obtain the following transition probability matrix, which we call T':

$$T' = \begin{pmatrix} 0 & 0.0938 & 0.1042 & 0.8021 \\ 0.0619 & 0 & 0.9175 & 0.0206 \\ 0.0018 & 0.0178 & 0 & 0.9804 \\ 0.0154 & 0.0002 & 0.9844 & 0 \end{pmatrix}$$

As before, the states represented by each row and column are Exploit, Binary Download, C&C Communication and Attack, respectively. The values along the diagonal are now equal to 0, as self-transitions have been removed. However, it can be easily verified that this model still satisfies all Markov properties discussed in Section III. We do not show the proof here.

B. Results: Predicting Attacks With the Markov Chain

We first divided the dataset temporally into training and testing data - i.e. for each host, we considered the first 1.5 hours of data as training, and the remainder as testing data. We chose this rather short training interval as the data duration varies considerably across traces, with some traces lasting only 2 to 3 hours. Therefore, using longer training intervals leaves fewer hosts' traces available for testing the model. We then ran the Markov training process to obtain a probability transition matrix from the training data, similar to matrix T' in Section V-A, but with different values as it was built on less

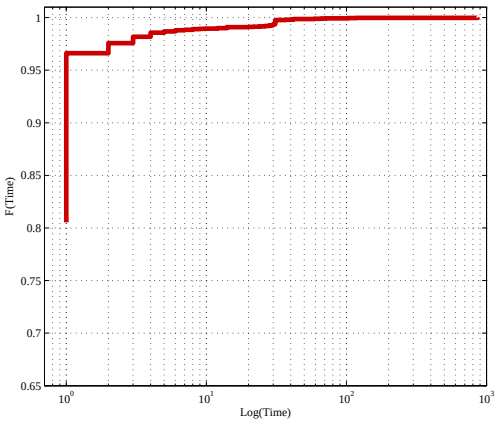

Fig. 3: CDF of the amount of time (in minutes) between an attack prediction and the attack itself.

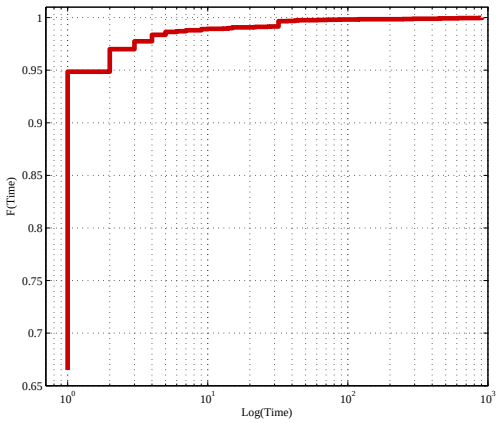

Fig. 4: CDF of the amount of time (in minutes) between a prediction of the C&C Communication state and an attack.

data. The testing phase worked as follows. We iterated over the testing data, treating it as a real-time, previously unseen stream of events. After observing each state, we predicted whether the next state to follow was likely to be the attack state. As described earlier, we made this decision based on whether the transition to the attack state has the highest probability out of all possible transitions from the current state.

The accuracy analysis of the experiment is shown in Table VI. We achieve a very high percentage (98.3%) of true positive predictions with a very low false alarm rate of 1.3%; i.e. only 1.3% of the time when we made an attack prediction, the attack did not occur as the immediate next transition. In this experiment, we achieve an overall accuracy of 98.5%.

Finally we show the frequency distribution of the time that elapsed after a correct attack prediction until the attack state actually occurred. Fig. 3 presents the CDF of these results. The x-axis is log-scaled to magnify the variation in values smaller than 10 minutes. The figure shows that in 81% of cases, the warning time is less than 1 minute, i.e. after the majority of predictions, less than one minute elapses before an attack is seen. In the remaining 19% of the cases, the warning time ranges from one minute and up to 865 minutes.

In a fully automated deployment even a minute is sufficient time to direct defensive measures towards hosts predicted to engage in attacks. However, a manual analysis of the data shows that in some cases there may only be a fraction of a second between the attack prediction and the actual attack. In such worst-case scenarios, our attack prediction is not meeting the desired objective of having ample early warning to allow for selective defensive measures. Hence, we investigate the possibility of increasing the warning time by making the attack prediction even earlier.

C. Increasing Warning Time: Predicting the C&C Communication State

We now investigate whether it is possible to generate attack warnings significantly earlier by predicting states that are very likely to precede the attack state. The probabilities in the transition matrix of Section IV-D are overwhelmed by the effect of self-transitions; we instead use the transition matrix T' from Section V-A, which we calculated based on the model with self transitions removed, and determine that the C&C Communication State, represented by the third row, has the maximum likelihood of leading to attacks. We hypothesise that predicting the C&C Communication state instead of directly predicting the attack state would allow for earlier yet reliable warning of attacks.

In the remainder of this section, we investigate three main questions: (a) how accurately we can predict the C&C Communication state, (b) how often the predicted C&C Communication state is actually preceding the attack state, i.e. the accuracy of using C&C Communication as an indicator of attack, and (c) how much of an earlier warning we can get for attacks when predicting C&C Communication instead of predicting the attack state directly. We run the CNC prediction experiment exactly like the attack prediction: from each state, we predict the next state to be C&C Communication if it is the destination of the most likely outgoing transition.

Table VII shows that we are able to predict the C&C Communication state with 99% accuracy, with 99.8% true positive predictions and only 1.7% false positive prediction. Further analysis of the true positives shows that 99% of the time the C&C Communication state was predicted, it was indeed followed by the attack state, demonstrating that it is a good indicator of future attacks. Finally, Fig. 4 plots the distribution of the time difference between predicting the C&C Communication state and the occurrence of the attack state. Even with the earlier prediction capability, in the majority of the cases we still have under a minute of warning. However, the proportion of these cases reduces from 81% when we were only predicting the attack state to 66% now that we are able to predict the C&C Communication state. While we acknowledge that a further improvement in this capability is needed, the results do validate our hypothesis that as we move further back in the state sequence, we should be able to increase the time difference between an attack warning and the actual attack. We hypothesise that for even earlier warning, we can predict the state that most likely leads to the C&C Communication state, which in turn most likely leads to the attack state. However,

owing to data limitations we leave a formal investigation of this approach as future work. In our current dataset, we only see 102 occurrences of the Binary Download state and 96 of the Exploit state, compared to nearly 5000 occurrences of both the attack and CNC states. Clearly this dataset is insufficient to investigate the temporal advantage of predicting these states.

VI. Limitations and Future Work

In this work we have presented and validated the idea of using Markov chains for attack prediction; however we do acknowledge that one of the key limitations of this work is the limited scope of the state transition model we use. Having validated the approach, we would in future like to extend the model to cover a larger range of malicious behaviour. In general we would like to replicate our experiments with the complete state model of Fig. 1. Specifically, we believe that adding a drive-by download state is crucial; as one of the most common infection vectors in recent Botnets, it represents the starting point of many infections. It is also very important to broaden the types of attack we are able to capture, as at present we are only able to detect portscans and spam. Secondly, in the current work, the type of attack has no bearing on the prediction; both spam and portscan are mapped to the same state in the model. With a greater variety of attacks in the dataset, we would like to map different attack types to different states in the model and investigate whether each kind of attack is preceded by unique behavioural patterns and whether this knowledge can further improve our prediction capability.

Using a limited model is necessitated by the limited detection capability of existing IDS tools, since a capability to detect and label each state of the model in the training data is essential if we are to use an extended model. For this work, we are limited to using states that can be detected using an existing IDS (Snort) with existing rulesets; therefore, another important future direction for us is to build our own detection tool that can detect a wider range of Botnet behaviour.

Finally, we want to investigate the effect of using variations of the simple Markov chain that we use in this work. One possible direction is higher-order Markov models [12] where future states depend on multiple past states rather than only the current state; we believe that incorporating more history in the decision will result in better prediction accuracy.

VII. Conclusion

In this work we investigated the intuition that we should be able to predict Botnet attacks based on the malicious behaviour observed in an infected host's traffic before it launches an attack. We designed a Markov chain model that predicted likely future behaviour based on the currently observed behaviour, using this capability to predict the probability of an attack. In addition we demonstrated the novel capability of predicting, in theory, any behavioural stage of an infection. Our results showed this to be a promising approach to generating early warnings of attacks, with over 98% of attacks predicted accurately and a worst-case false alarm rate of 1.7%.

References

[1] Systems and Networks Research Lab. https://sysnet.lums.edu.pk/. [Online; accessed 2-May-2016].

[2] Suricata IDS. https://suricata-ids.org/, 2010. [Online; accessed 20-April-2016].

[3] Dridex Attacks Target Corporate Accounting. http://blogs.cisco.com/security/dridex-attacks-target-corporate-accounting, 2015. [Online; accessed 20-April-2016].

[4] M. Abu Rajab, J. Zarfoss, F. Monrose, and A. Terzis. A multifaceted approach to understanding the botnet phenomenon. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 41–52. ACM, 2006.

[5] E. B. Beigi, H. H. Jazi, N. Stakhanova, and A. A. Ghorbani. Towards effective feature selection in machine learning-based botnet detection approaches. In *Communications and Network Security (CNS), 2014 IEEE Conference on*, pages 247–255. IEEE, 2014.

[6] Z. Duan, P. Chen, F. Sanchez, Y. Dong, M. Stephenson, and J. M. Barker. Detecting spam zombies by monitoring outgoing messages. *Dependable and Secure Computing, IEEE Transactions on*, 9(2):198–210, 2012.

[7] W. Gobel. Detecting botnets using hidden markov models on network traces.

[8] G. Gu, R. Perdisci, J. Zhang, W. Lee, et al. Botminer: Clustering analysis of network traffic for protocol-and structure-independent botnet detection. In *USENIX Security Symposium*, pages 139–154, 2008.

[9] G. Gu, P. A. Porras, V. Yegneswaran, M. W. Fong, and W. Lee. Bothunter: Detecting malware infection through IDS-driven dialog correlation. In *USENIX Security*, volume 7, pages 1–16, 2007.

[10] O. Haq, Z. Abaid, N. Bhatti, Z. Ahmed, and A. Syed. Sdn-inspired, real-time botnet detection and flow-blocking at isp and enterprise-level. In *Communications (ICC), 2015 IEEE International Conference on*, pages 5278–5283. IEEE, 2015.

[11] T. Hirayama, K. Toyoda, and I. Sasase. Fast target link flooding attack detection scheme by analyzing traceroute packets flow. In *Information Forensics and Security (WIFS), 2015 IEEE International Workshop on*, pages 1–6, Nov 2015.

[12] W.-H. Ju and Y. Vardi. A hybrid high-order markov chain model for computer intrusion detection. *Journal of Computational and Graphical Statistics*, 10(2):277–295, 2001.

[13] J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan. Fast portscan detection using sequential hypothesis testing. In *Security and Privacy, 2004. Proceedings. 2004 IEEE Symposium on*, pages 211–225. IEEE, 2004.

[14] S. Khattak, Z. Ahmed, A. Syed, and S. A. Khayam. Poster: Botflex: a community-driven tool for botnetdetection. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 1387–1390. ACM, 2013.

[15] E. Kidmose. Botnet detection using Hidden Markov Models. Master's thesis, Aalborg University, Denmark, 2014.

[16] C. Lu and R. Brooks. Botnet traffic detection using hidden markov models. In *Proceedings of the Seventh Annual Workshop on Cyber Security and Information Intelligence Research*, page 31. ACM, 2011.

[17] V. Paxson. Bro: a system for detecting network intruders in real-time. *Computer networks*, 31(23):2435–2463, 1999.

[18] M. Roesch et al. Snort: Lightweight intrusion detection for networks. In *LISA*, volume 99, pages 229–238, 1999.

[19] S. E. Schechter, J. Jung, and A. W. Berger. Fast detection of scanning worm infections. In *Recent Advances in Intrusion Detection*, pages 59–81. Springer, 2004.

[20] S. S. Silva, R. M. Silva, R. C. Pinto, and R. M. Salles. Botnets: A survey. *Computer Networks*, 57(2):378–403, 2013.

[21] Y. Tao and S. Yu. Ddos attack detection at local area networks using information theoretical metrics. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2013 12th IEEE International Conference on*, pages 233–240, July 2013.

[22] T. Thapngam, S. Yu, W. Zhou, and S. K. Makki. Distributed denial of service (ddos) detection by traffic pattern analysis. *Peer-to-peer networking and applications*, 7(4):346–358, 2014.

[23] G. K. Venkatesh, V. Srihari, R. Veeramani, R. Karthikeyan, and R. Anitha. Http botnet detection using hidden semi-markov model with snmp mib variables. *International Journal of Electronic Security and Digital Forensics*, 5(3-4):188–200, 2013.

[24] N. Ye et al. A markov chain model of temporal behavior for anomaly detection. In *Proceedings of the 2000 IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop*, volume 166, page 169. West Point, NY, 2000.