Scalable Framework for Cyber Threat Situational Awareness Based on Domain Name Systems Data Analysis

R. Vinayakumar, Prabaharan Poornachandran and K. P. Soman

Abstract There are myriad of security solutions that have been developed to tackle the Cyber Security attacks and malicious activities in digital world. They are firewalls, intrusion detection and prevention systems, anti-virus systems, honeypots etc. Despite employing these detection measures and protection mechanisms, the number of successful attacks and the level of sophistication of these attacks keep increasing day-by-day. Also, with the advent of Internet-of-Things, the number of devices connected to Internet has risen dramatically. The inability to detect attacks on these devices are due to (1) the lack of computational power for detecting attacks, (2) the lack of interfaces that could potentially indicate a compromise on this devices and (3) the lack of the ability to interact with the system to execute diagnostic tools. This warrants newer approaches such as Tier-1 Internet Service Provider level view of attack patterns to provide situational awareness of Cyber Security threats. We investigate and explore the event data generated by the Internet protocol Domain Name Systems (DNS) for the purpose of Cyber threat situational awareness. Traditional methods such as Static and Binary analysis of Malware are sometimes inadequate to address the proliferation of Malware due to the time taken to obtain and process the individual binaries in order to generate signatures. By the time the Anti-Malware signature is available, there is a chance that a significant

R. Vinayakumar (✉) · K. P. Soman
Amrita School of Engineering, Coimbatore, Centre for Computational Engineering and Networking (CEN), Amrita Vishwa Vidyapeetham, Amrita University, Coimbatore, India
e-mail: r_vinayakumar@cb.amrita.edu; vinayakumarr77@gmail.com

K. P. Soman
e-mail: kp_soman@amrita.edu

P. Poornachandran
Amrita School of Engineering, Centre for Cyber Security Systems and Networks, Amrita Vishwa Vidyapeetham, Amrita University, Amritapuri, Coimbatore, India
e-mail: prabasuja@gmail.com

S. S. Roy et al. (eds.), *Big Data in Engineering Applications*,
Studies in Big Data 44, https://doi.org/10.1007/978-981-10-8476-8_6

amount of damage might have happened. The traditional Anti-Malware systems may not identify malicious activities. However, it may be detected faster through DNS protocol by analyzing the generated event data in a timely manner. As DNS was not designed with security in mind (or suffers from vulnerabilities), we explore how the vast amount of event data generated by these systems can be leveraged to create Cyber threat situational awareness. The main contributions of the book chapter are two-fold: (1). A scalable framework that can perform web scale analysis in near real-time that provide situational awareness. (2). Detect early warning signals before large scale attacks or malware propagation occurs. We employ deep learning approach to classify and correlate malicious events that are perceived from the protocol usage. To our knowledge this is the first time, a framework that can analyze and correlate the DNS usage information at continent scale or multiple Tier-1 Internet Service Provider scale has been studied and analyzed in real-time to provide situational awareness. Merely using a commodity hardware server, the developed framework is capable of analyzing more than 2 Million events per second and it could detect the malicious activities within them in near real-time. The developed framework can be scaled out to analyze even larger volumes of network event data by adding additional computing resources. The scalability and real-time detection of malicious activities from early warning signals makes the developed framework stand out from any system of similar kind.

Keywords DNS log analysis · Big data analytics · Machine learning
Deep learning

1 Introduction

Nowadays, Internet has become the largest critical global communication medium and infrastructure. It connects several billions of nodes enabling them to communicate with each other. At its heart, Internet uses one important protocol—Domain Name System (DNS). The DNS protocol translates, difficult to remember Internet addresses to human readable names and vice versa. With the increasing dependency and usage of the Internet by users and systems, all the malicious activities that used to occur in the physical world moved to the connected digital world of Internet. The vectors of malicious activities mainly include Virus, Worms, Trojans, Cyberattacks, Phishing, Spam and Cyber-intrusions. With the exponential growth of hosts connected to the Internet, the usage of the DNS protocol by the systems and networks is progressively increasing to a very high level. This in turn produces very large volume of event data amounting to trillions of events per minute at the Tier-1 Internet Service Provider (ISP).

2 Related Work

This section discusses the selected research works previously done on cyber threat analysis and detection. In order to analyze the cyber threat or to provide cyber threat situational awareness, we investigated the event data and status information generated by Internet protocols—Domain Name Systems (DNS). The Internet community is facing serious threats from everywhere. Nowadays the behavior and operation of such threats are undergoing a crucial transformation and evolution. One of the widely used attacking strategies is to use botnets. Botnets are generally used for large-scale Cyber Security attacks that include Distributed Denial of Service (DDoS) attacks, large-scale spam campaign, identity theft etc. Botnets try to increase the number of affected hosts by incorporating executable programs that are present in several hundred compromised hosts, which receives its instructions or commands from Command and Control (C&C) server that is operated by the malicious actors known as bot-master [1]. One of the security measures to protect against this threat is to blacklist the C&C server with whom the botnets tries to contact for getting instructions. However, to evade being blacklisted by organizations, the attackers use DNS agility or fluxing approach by changing their domain names or IP addresses frequently in rapid succession [2]. This technique is known as Fast Flux and several families of Botnets use fast flux to hide their criminal activities and evading detection by changing the compromised hosts that acts as proxies for malware distribution. Fast Flux networks can be of two types, they are IP-Flux and Domain-Flux respectively [3]. IP-Flux can be further classified as Single-Flux, and Double-Flux. Constant registering and de-registering of IP addresses for a given malware domain are the main characteristics of Single Flux. In Double-Flux, the SRecord or the name server record for a DNS zone of a malicious site is also changed in addition to 'A' records. These techniques provide additional redundancy for botnet. Unlike IP-flux, Domain-Flux assigns several fully qualified domain names to a single IP address or C&C infrastructure. Usually hundreds of thousands of domain names are employed in a Domain-Flux. Bot-masters achieved this by using Domain Generated Algorithms (DGA) that generates domain names randomly on a large scale for registration. Examples for such botnets are Conflicker [4], Torpig [5], Kraken [6], Murofet [7]. 'Kwyjibo tool' is a special malicious program which uses a sophisticated domain generation algorithm for generating domain names that are similar to English Dictionary words [8]. These generated words cannot be used in botnets because there could be an ambiguity with existing domain names. However, botnets try to contact their C&C server with these randomly generated domain name using hit and trial method. For example, out of several thousand DGA generated domain names, the bot master will procure only a handful of domain names. As a result, lots of Non-Existent (NX) response queries get generated. This process makes the security strategy very expensive for the defender and very economical for the attacker. In this scenario, the attacker has to buy just a handful of domain names, but at the same time, the defender will have to procure/block/sinkhole millions of domains that makes the

effort very expensive. A Non-Existent (NX) response or Name Error response are those domain queries for which no IP addresses or any record exists [9]. Pleiades [10] was the first system that was able to detect DGA based domains without reverse engineering the bot malware. The core of Pleiades consists of two modules —DGA discovery module and DGA classification and C&C detection module. DGA discovery module detects and clusters the botnet queries, given all NX domain features. For clustering botnet queries, extracted features from the observed domain names such as n-gram, entropy based features, structural domain features are used with X-means clustering algorithm. The latter module used Alternating Decision Tree learning algorithm for the identification and comparison of NX-Domain clusters, whereas C&C detection module used Hidden Markov Model. It is reported that detection rate of Pleiades' DGA classifier is 99.7% while C&C detection system was said to have detection rate greater than 91% for five out of six tested botnets and 22.67% for the remaining bot. Another work on the detection of botnets is done by Schiavoni et al. [11] and they did an elaborate survey on existing domain flux related C&C server detection systems. They propose an unsupervised algorithm which does not require any prior knowledge of the DGAs and no reverse engineering of the malware samples. In [12], J. Raghuram et al. proposed another unsupervised way for detecting malicious domain names. They used linguistic features and Expectation-Maximization for solving the problem. One of the main reasons for generating the non-pronounceable domain name is to avoid collusion that might occur due to the presence of an existing domain name. While IP-Flux networks make it time consuming and expensive to disrupt, Domain-Flux makes the process of purchasing the malware domains expensive for the purpose of sinkhole, as it will require the purchase or sinkhole several Millions of domain names. In [13], Thomas et al. analyzes DNS traffic from several authoritative name servers to identify strongly connected DGA based domains. Fast fluxing takes the advantage of DNS based load balancing by masking its own activities as it is done in the case of Content Delivery Networks (CDN). Fast Fluxing uses several IP addresses that are hidden behind a single domain name just as large CDNs and Antivirus providers architect their systems. The IP addresses of domains involved in Fast Fluxing changes with extreme frequency in round-robin fashion with very short Time-To-Live (TTL) for each DNS Resource Record (RR).

A dictionary based approach with Smith-Waterman algorithm was proposed for predicting the malicious domain detection [14]. Moreover, the experiments were done with the data captures from real-time systems. Zdrnja et al. [15] proposed a passive DNS anomaly detection project based on data captured at the University of Auckland Internet gateway with a view to detect and correlate domains used for botnet controls using a passive DNS monitor. A passive data capturing sensor was deployed at the network edge and extracted query name, resource record type, resource record data, TTL and first seen time stamp from the DNS data. This information can further be used for analyzing the historical behavior of certain DNS records and how they are linked with each other. Ramachandran and Feamster [16] studied the network-level behavior of spammers such as range of IP addresses which send spams, common bot modes, persistent time of each spam

host, spamming bot characteristics etc. From the analysis, they could identify the region of IP addresses from which maximum bot attacks being sent. Anderson et al. [17] proposed a scam-hosting infrastructure in which patterns in emails are identified to track the spam servers. Image shingling method is then applied to find the scams whose webpages are graphically similar to a cluster of spam servers. All these approaches propose botnet detection based on DNS dictionary lookup. These approaches will fail in botnet detection when the number of incoming queries is very high in real-time. In other words, these approaches are not scalable and storage requirements and time requirement for decision making are also very high.

Major drawbacks of this approach are (1) computational complexity is very high with large volume of data. (2) Over fitting may occur with huge data. An old concept of artificial intelligence called as neural network (in recent times typically termed as deep learning) has achieved a significant result in various multitudinous fields namely natural language processing, image processing, speech recognition and many others [18]. Deep learning mechanisms itself facilitated to extract features by taking massive amount of raw data set as input. This provides significant contribution towards big data analytics. Deep learning algorithms are mainly categorized into two types (1) Convolution neural network (CNN) (2) recurrent neural network (RNN); CNN are largely used in the field of image recognition mainly due to the fact that, CNN applies a set of filters on rectangular region to extract complex features by travelling through layer by layer. The complex features are composed from a set of lower level features that forms a hierarchical feature representation. As a result CNN captures the features in various level of abstraction. CNN primarily use a pair of convolution, pooling operations and a non-linear activation functions. [19] adopted CNN in character level towards large scale text classification including several languages. The effectiveness of CNN model is compared with the other traditional mechanisms such as bag of words, n-grams and tf-idf. RNNs are initially proposed for time-series data-modeling [20] and later this has been applied in sequence data modeling tasks in the field of speech processing, natural language processing. As the research goes, [21] found the vanishing and exploding gradient issue when dealing with large sequences. On alleviating the vanishing and exploding gradient issue, researchers worked in the 3 directions; (1) enhancement of optimization methods for example Hessian-free optimization [22], (2) proposing a new architecture; LSTM [23] and its variants GRU [24], (3) appropriate weight initializations for example I-RNN [25, 26]. Moreover, researchers have also used hybrid network, in which the first layer is used as CNN and the CNN outputs are given as input to other recurrent network layers [27].

The effectiveness of various classical and deep learning approaches is studied for various Cyber Security tasks like Android malware detection [28, 29], DGA analysis [30, 31], traffic analysis [32, 33], malicious URL detection [34], intrusion detection [35, 36], anomaly detection [37], ransomware detection [38], encrypted text categorization [39], network traffic prediction [40].

3 Background

3.1 Domain Name System (DNS)

Domain Name System (DNS) is considered as the core Internet protocol. The hierarchical level of DNS is shown in Fig. 1. The present work is based on the analysis and detection of attacks that can be detected by observing the behavior of the DNS protocol and studying the events that are produced when the DNS protocol is used by the systems and networks. The section below gives an overview about the DNS protocol.

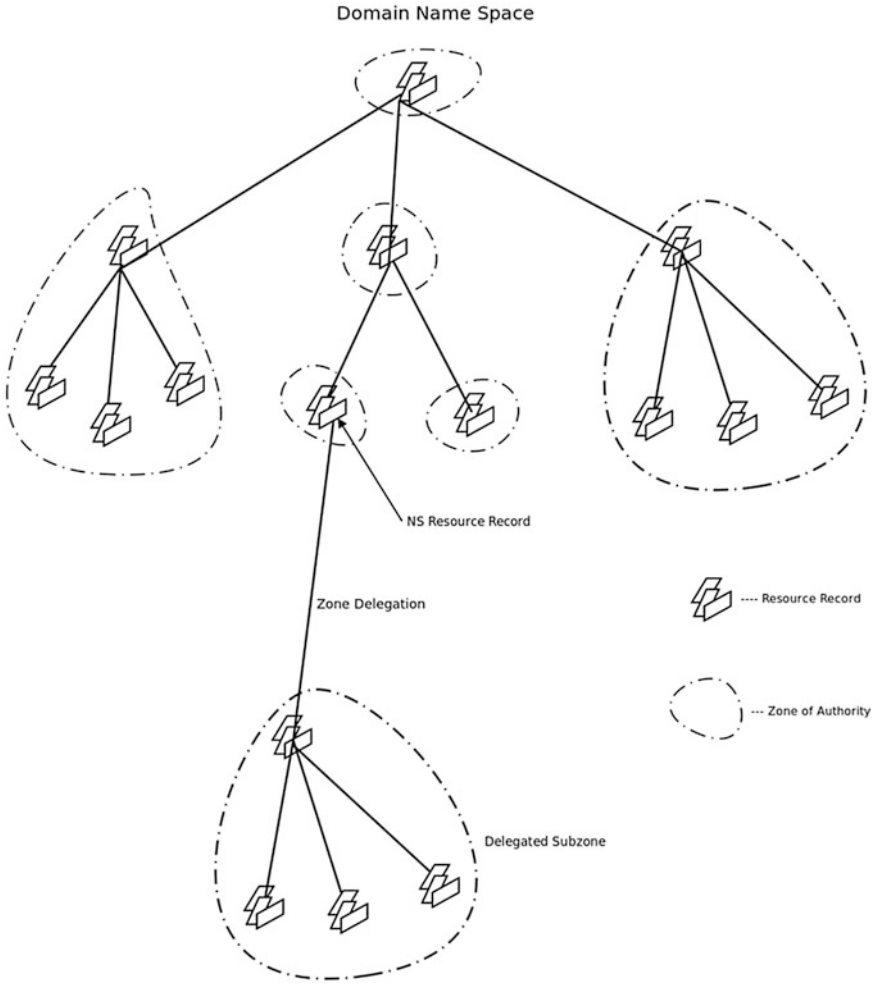

Fig. 1 Hierarchical domain name system

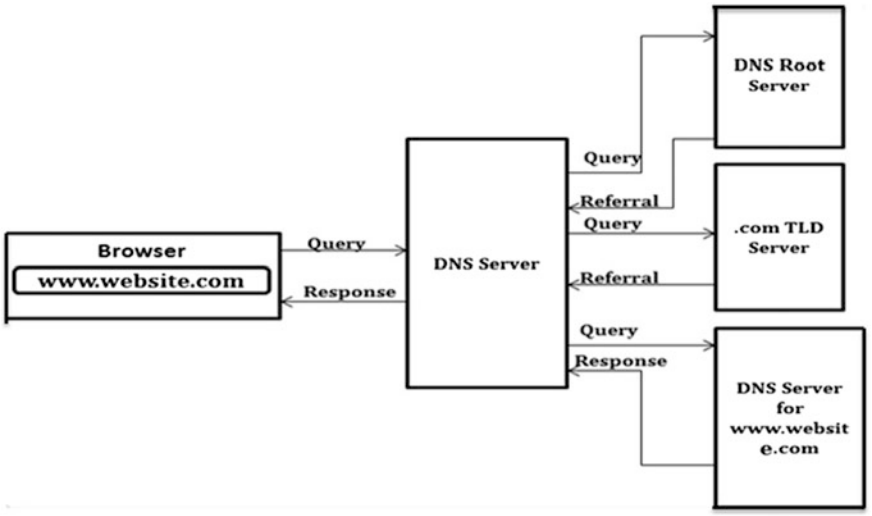

Fig. 2 Recursive DNS query

End users usually access Internet by using a browser that renders the web pages and portals. This is done by typing a domain name in the address bar of the browser. When this is done in a right fashion, Internet helps the users in information exchange and transactions. DNS servers can be classified into two types: Non-recursive/Iterative servers and Recursive servers. Non-recursive DNS servers act as the Start of Authority (SOA). They answer queries inside the governed domains without querying other DNS servers even if they cannot provide the requested answer. Whereas, Recursive DNS servers (an example is shown in Fig. 2) respond to queries of all types of domains by querying other servers and passes the response back to the client. Distributed Denial of Service (DDoS) attacks, DNS cache poisoning, unauthorized use of resources, root name server performance degradation, etc. are some of the major attacks suffered by Recursive DNS servers. DDoS is an attack which makes a network resource unavailable to the legitimate intended users by flooding it with fraudulent or forged requests continuously to the targeted DNS server that overwhelms the system. Cache poisoning/DNS spoofing attacks are the fraudulent activity of inserting a fake address record for an Internet domain into the DNS. The cache is considered as poisoned, if the server accepts this fake record and after which, subsequent requests for the domain will be answered by attacker's server. This cached entry will remain there till it's Time-To-Live (TTL) expires and during this period the subscriber's browser will go to the address provided by the compromised DNS server. This is an attack that diverts Internet traffic away from legitimate servers to fake servers by the malicious actors. Since it spreads across DNS servers, it is considered as a dangerous attack. For example, on 11th October, 2013 Google's Malaysian domains google.com.my and google.my were hijacked and redirected users to a web page that announced that the attack was perpetrated by a Pakistani group called Madleets

[41]. This attack was a DNS cache poisoning attack that lasted a few hours and while it was active, it redirected users to a website hosted in Canada. Hijacking of a server is simple in recursive DNS query enabled servers as attackers can poison the response while it is coming back from the root servers [2]. DNS servers that are not configured correctly are more vulnerable to these types of attacks. In several cases, many organizations do not even know or does not have situational awareness system that can quantify the number of compromised systems in their networks. These organizations could be Internet Service Providers, or small, medium and large enterprise networks [27].

In the case of enterprise networks, the visibility and situational awareness is limited to the anti-virus systems that are installed. And in the case of Internet Service Providers, over 63% of surveyed respondents do not know the proportion of the devices in their network which are compromised and involved in botnet or other malicious activities [27]. One of the sophisticated and popular types of malware that wreaks havoc on the Internet is botnets. When a bot infects a system or host, it can be commandeered to do various automated malicious activities that include sending malwares, stealing private and sensitive information, key stroke logging, and participation in C&C based DDoS attacks. In addition to the above-mentioned attacks, it might affect the local system with various malwares like Clickfraud, Adwares, Spywares, etc. Clickfraud is an automated fraud computer program which will click an advertisement without the knowledge of the actual user with intent to give false clicks to an advertisement in order for the advertiser to make more financial gains illegitimately. A pictorial diagram of the various activities of a Botnet is shown in Fig. 3. Fast flux is termed as one of the malicious activities that are being done using botnets. Fast flux service networks use many IP addresses that are mapped to a single domain name.

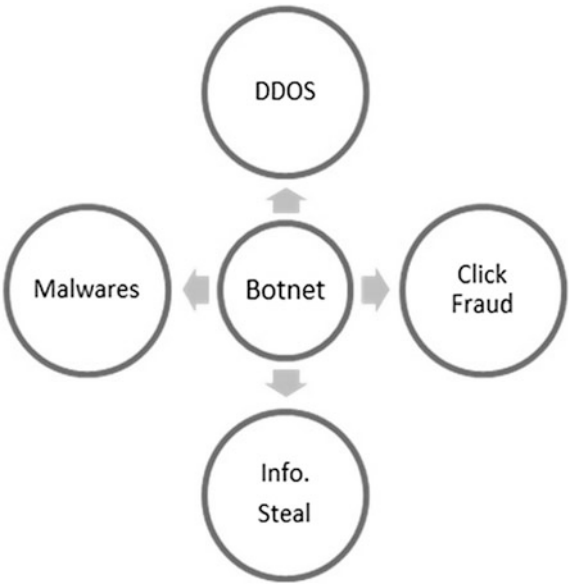

Fig. 3 Botnet activities

3.2 Scalable Algorithms

In real-time, it is required to process extremely large volume of data that result from the system and network events on the use of DNS system. The algorithms that are computationally efficient and that can be distributed across multiple systems have been studied in the current research work. We use deep learning algorithms for this research work to detect the attacks. The deep learning algorithm contains billions of parameters. In order to train them in the context of detecting the dga generated domain, we use distributed TensorFlow framework. This facilitates parallelism of deep learning models in two aspects; one is within a machine through multi-threading and second one is across machines through message passing. Moreover, the framework also supports the data parallelism where multiple replicas of deep learning models are used to achieve a single task.

4 System Architecture

4.1 Scalable Architecture

Since, DNS and BGP together produce several Billions of data events per minute, a highly scalable framework has been developed that can collect and process the data in real-time. The framework consists of DNS and BGP sensors that collect the data in a distributed manner. These sensors receive data directly from the DNS servers and BGP enabled routers. To ensure that, the introduction of this system does not impact the functionality of the DNS and BGP systems, they are designed to collect the passive information in the form of out-of-band mode. The collected data is parsed and aggregated and then sent to real-time and non-real-time analysis engine that runs highly scalable distributed deep-learning algorithms. The query-router (standalone) services that controls the communication between different modules. The query-router also provides an interface with the message broker. A subsystem known as Front-End Message Router controls the communication between the Interactive Visualization and analysis engine on the processed data. The Figs. 4 and 5 depict the standalone and scalable architecture of the framework respectively.

The present research work proposes to deploy the standalone architecture in individual Tier-1 ISPs. Each standalone system is capable of handling few millions of DNS and BGP data per second without any stability issues. Hence, Terabytes (TB) of data were able to collect within a day. However, monitoring a single ISP might not be enough to get an overall situational awareness of a malware propagating through a zone or country, thus resulting in monitoring and correlating the network activity of several Tier-1 ISPs. The proposed scalable architecture in this research work employs distributed and parallel algorithms with various optimization techniques that make it capable of handling huge volume of data. The scalable

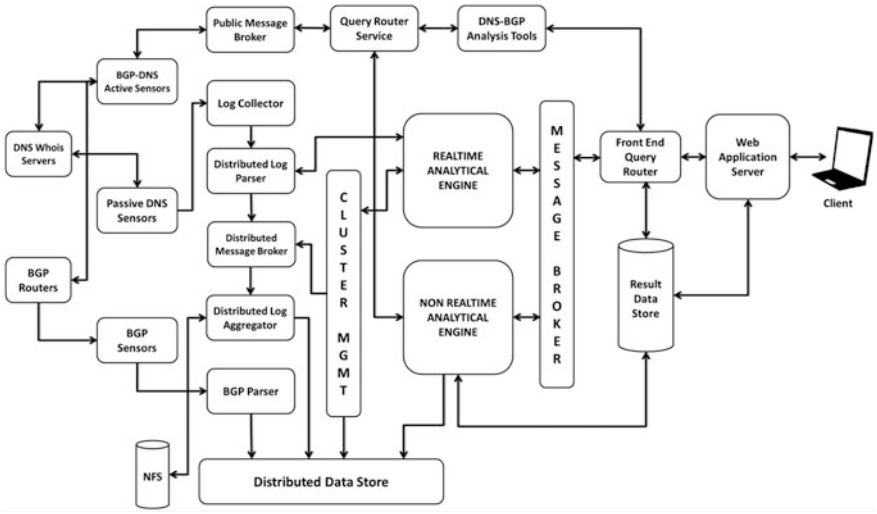

Fig. 4 Architecture of data collection framework

architecture also leverages the processing capability of the General Purpose Graphical Processing Unit (GPGPU) cores for faster and parallel analysis of DNS data. The framework architecture contains two types of analytic engines—real-time and non-real-time analytic engines. The purpose of analytic engine is to detect malicious activities thereby generate an alert in case of threat.

4.2 Supporting Services

The important supporting services needed for this project are explained below.

1. Passive Sensor: The Passive Sensor collects DNS Query/Response from the DNS Servers (Any DNS Server). The passive sensor captures Network Traffic from DNS Servers and passes it to an application. The parser inspects the DNS Response Packet, converts it into human readable format and forwards it to DNS Log Collector. The Passive Sensor could be installed inside the DNS Server itself or any mirrored traffic could be sent to the sensors. Each Sensor could process the data from multiple DNS Servers if needed. It collects data passively from DNS Servers without affecting the DNS Server.
2. Active Sensor: This sub-system performs an active DNS Query related to the given sources, to collect data and perform various analyses. Its main sources are DNS server, WHOIS server, application programming interfaces such as Google Safe browsing Application programming interface (API), and other DNS databases.

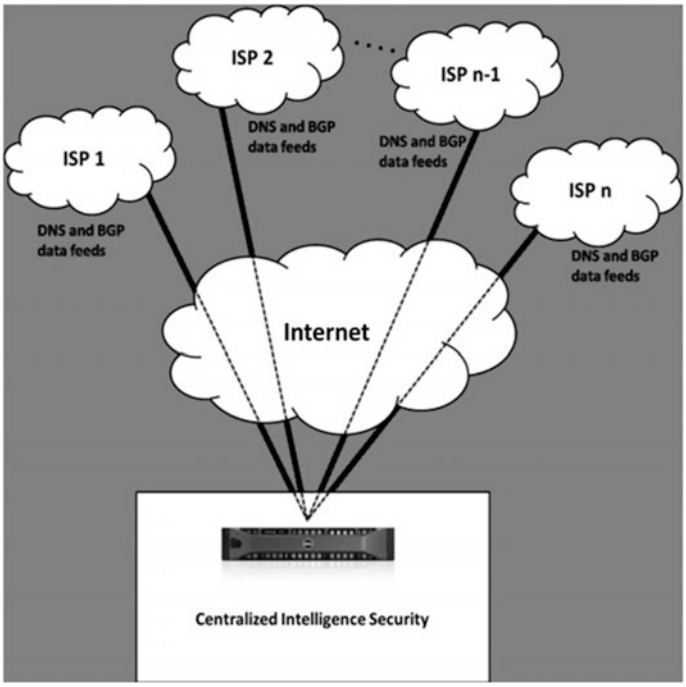

Fig. 5 High-level architecture correlating data from multiple ISPs

3. WHOIS Server: This will provide information associated with a Domain or IP Address. Queries will be forwarded to corresponding WHOIS server for that particular Domain/IP address. The information related to domain and IP are explained below: For a domain, query is sent to retrieve the following information:

- Registrant name, address, email, phone.
- Administrator name, address, email, phone.
- Domain registered date, expiration date, authoritative name server etc.

For an IP address the fields are:

- Owner
- Prefix (Network)
- ASN
- Location
- Expiration Date

4. Online API's: Some of the analysis results could be correlated with the online external systems for validation and cross-correlation.
5. Log Collector and Parser: This subsystem collects the parsed DNS Responses from distributed sensors and forwards it to the collector. The collector looks up the Geo-Location and details of ASN of each IP address (Client IP, DNS Server IP, and A Records in Resource records) in the DNS Responses. The parser uses the Geo IP database to find Geo-location (city, country, latitude, longitude) of an IP Address. ASN database is used for finding details of ASN (AS Number, AS Name) of an IP Address. This data will be appended with original data coming from sensors and publish to queue for real-time and batch analysis.
6. Query Router Service: It is a standalone service, which controls the communication between different modules. It also interfaces with public message broker.
7. Front-End Message Router: As a standalone service, this subsystem controls all the communication to-and-from the front-end UI. It also interfaces with the respective back end subsystems.
8. Internal Router: Internal router software is peered with the BGP Router to get BGP Updates in a configurable time interval such as 5 min etc.
9. BGP Monitor: The BGP monitor subsystem is responsible for collecting BGP Update-messages in real-time. It gets updates from the TCP port as a stream and stores them in an XML file format. The parsers will produce BGP update messages in one single format, after extracting the data from two different sources. A Distributed Log Aggregator collects this parsed data to store it in a distributed database for further analysis.

4.3 Data Collection

DNS data are collected in a passive manner by reading from the mirrored traffic using promiscuous mode on DNS communication information between the DNS server and the DNS clients. The data consists of DNS queries and the corresponding DNS answer made by the DNS client and DNS server respectively. The extracted data is analyzed for malicious events. The BGP data is collected by adding a read-only peer to a BGP speaking router. The read-only peer collects the data that occurs in the form of BGP updates, announcements, neighbor information etc. The BGP data consists of the route and prefix information. To identify the malicious announcements, malware propagation and activities, the prefix announcements, route announcements and updates information can be used.

5 A Sub System for Detecting DNS Anomalies Based on Deep Learning and GPGPU

5.1 Introduction

Recursive DNS servers, responds to queries of all types of domains, by querying other servers and passes the response back to the client as shown in Fig. 6.

DeepBot primarily focuses on identifying botnets using the Domain Flux service. In Domain Flux, the Botmaster changes the domain name that has to be mapped with the IP address of the C&C server frequently. For this, they use Domain Generated Algorithms (DGA) to generate domain names randomly on a large scale for registration. Botnets use different DGA's for domain name generation. For example: Conflicker [4], Torpig [5], Kraken [6], Murofet [7] etc. Kwyjibo [8] uses a sophisticated domain generation algorithm for generating domain names that is similar to English Dictionary words. In Domain Flux, the botnets try to contact their C&C server with the randomly generated domain name using hit and trial method. As a result, in most of the cases, a lot of Non-Existent (NX) response queries get generated. A Non-Existent (NX) response or Name Error response are those domain queries for which no IP addresses or any record exists [9].

The DeepBot framework is based on an assumption that the botnets sitting in an infected system generate large sets of NX queries before actually getting resolved (if successful). Also, since botnets using the same DGA's generate similar domain names, comparing the patterns in NX-Domains and the resolved domain names of a

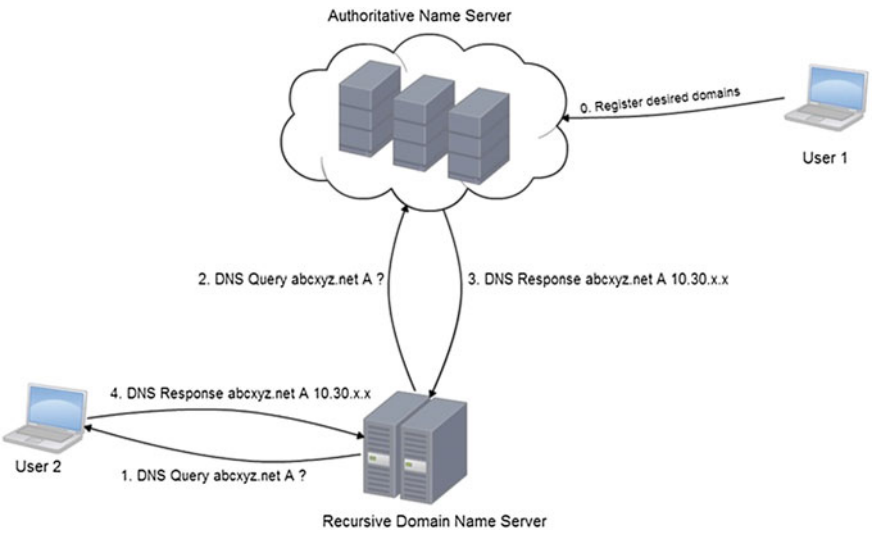

Fig. 6 Working flow of a legitimate DNS query

particular host, one could identify the malicious C&C server. NX-Domains could also get generated due to human errors like spelling mistakes while querying a domain name. Hence, to distinguish between a human error and DGA generated NX-Domains, the framework employ deep learning algorithm. This implicitly obtains optimal feature representations to distinguish the domain name as benign or DGA generated. Once the framework classifies a particular NX-Domain as a DGA domain, the framework assumes that the host is infected and analyses its resolved domain list for finding the corresponding C&C server. Thus by analyzing the NX-Domain queries, the Deep Bot framework is not only able to find out the compromised hosts infected by botnets but also the malicious C&C server.

5.2 System Architecture

This section describes the overall architecture and working of the analysis, along with the methods used for data collection. Figure 7 shows the high level architecture diagram of Domain Flux analysis. The analysis consists of three main modules: Identifying DGA Infected Hosts module, C&C Detection module and finally Time Analysis. The system consists of mainly 2 modes: Training mode and Testing mode.

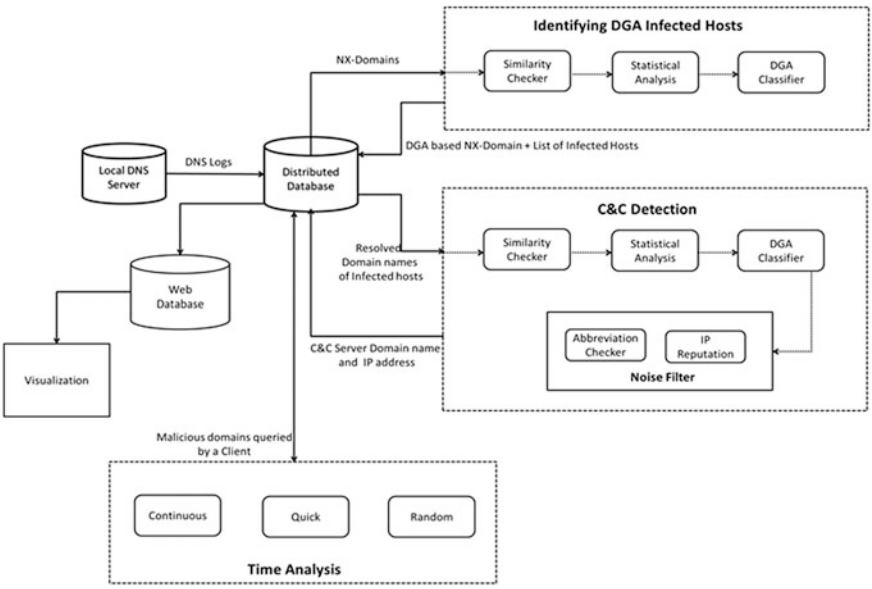

Fig. 7 High level architecture diagram of domain flux analysis

1. Training Mode: The system is trained using the white list and malicious datasets. In the white-list dataset, there are top 1 million domains provided by Alexa [42] whereas approximately 50 million malicious domains were collected from April, 2012 to 2016. The system is trained using these features in an on-line mode. In due course, thresholds were set for various analyses performed, which are being used for testing.
2. Testing Mode: In this mode, the system analyses DNS log streams which were acquired from the local DNS server. The data coming from the sensors are first stored in a distributed database. The data is fetched from the distributed database periodically with a time interval. The implementation of this module is given bellow.

As mentioned earlier the system has mainly three main modules. Throughout this chapter, for convenience, some terms were used. Suppose a domain d, where d = "abc.example.com". Here, the term "com" is termed as the TLD i.e. the Top Level Domain. Likely, "example.com" is called as 2LD (Second Level Domain) and "abc.example.com" is called as 3LD (Third Level Domain).

5.3 Details of Implementation

This section explains in detail the steps employed to identify the hosts that are already infected by a DGA based bot. The analysis is based on an assumption that an infected host on the process of communicating to a C&C server generates many NX queries. Thus, by analyzing the malicious NX-Domains it could easily be understood that whether the host is infected or not.

However, not all of the NX queries made could be termed as malicious. For example, human typing mistakes also lead to NX queries. Hence, the challenge lies in classifying the human mistake and the malicious queries into different sets. From the distributed database, a set of data were taken which is in the form of a tuple $T(t_s, h_i, d_i, s)$. The tuple t has four values namely, t_s-Timestamp, h_i-host IP address, d_i-domain queried, s-query status. Here, since the focus is only for the NX queries, the status is kept as NX. So NX queries made by all the hosts were recursively analyzed within a time period of 10 min. Once all the 10-min logs were collected, send it to this module in the tuple format mentioned above. Thereafter the domain name is split and only its 2LD label is extracted. The 2LD name thus extracted is sent for Similarity Checker Analysis.

1. Similarity Checker: The objective of this analysis is to find whether any legitimate domains are present which are quite similar. The 2LD domain label is compared with all of the domains in the 1 million Alexa set with the help of Approximate String matching algorithm [43]. Using the Damerau-Levenshtein [44] Edit Distance algorithm the distance between two strings is calculated. If the distance is more than a threshold value then it is considered that the

particular 2LD domain name is not a typing mistake and it is sent for Statistical Analysis. The advantage is that domain queries like "ggoole" which is caused by typing mistake could be excluded easily.

2. Statistical analysis: The system was developed and deployed with an aim to detect DGA domains automatically in real-time. After detecting the DGA domain, the system monitors its activities in an interval of 5 seconds on a regular basis. Figure 10 presents the architectural diagram of the implemented system which consists of three modules (1) Data Collection (2) Deep learning for detecting DGA (3) Dynamic reputation.

The data for the system is collected by deploying passive sensors across the four geographically distributed University campuses comprising of more than 30,000 unique users. The Passive Sensor collects DNS Query/Response from the deployed DNS Servers (Any DNS Server). Hence in this work data were collected from four DNS servers. The sensor captures the network traffic from DNS servers and passes it to an application, which takes only the traffic that is originated from DNS Server (DNS Response Traffic). It will then dissect the DNS Response Packet, converts it into human readable format and forward it to DNS Log Collector. The Passive Sensor could be installed inside the DNS Server itself (for Linux/Unix Platform) or mirror the traffic to a server which is dedicated for DNS Passive Sensors. The sensors get the data by port mirroring the traffic from the DNS Servers present in the deployed network. The logs received from the 4 campus sensors are received by a distributed log parser. The parser finds Geo Location and ASN Details of each IP address (Client IP, DNS Server IP, and A Records in Resource records) in the DNS Responses. The parser uses the Maxmind Geo IP database to find Geo location (city, country, latitude, longitude) of an IP Address. Maxmind ASN database is used for finding ASN details (AS Number, As Name) of an IP Address. A Query Router Service is then used to control the communication between different modules. The Front End Message Router controls all the communication to and from the front end UI. It also interfaces with the respective back end modules. Dynamic reputation subsystem analyses the DNS traffic in a network and detect and alert the presence of suspicious domains receiving anomalous hits. The architectural details of the developed framework are provided in Fig. 8.

1. Domain names encoding in character level: In recent days, deep learning approaches have achieved a significant performance in various tasks such as language modeling, text classification and many others in the area of natural language processing (NLP) [18]. They have an ability to learn appropriate feature representations by considering the input as in the form of raw data. A primary task in NLP is how to represent the text into numeric vectors. Primarily 2 views of representation are used by researchers (1) Texts are represented as a stream of characters (2) Texts are represented as a sequence of words. Domain name representation is called as domain name encoding. Domain name encoding consists of 2 steps. In first step raw domain names are preprocessed and tokenized to characters. Preprocessing involved in removal of

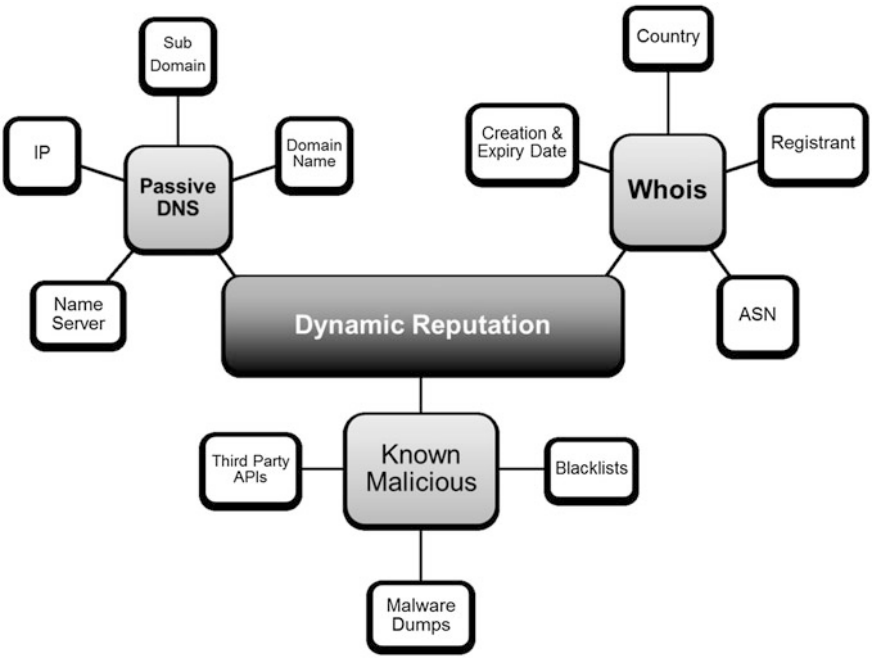

Fig. 8 Ensemble based dynamic reputation system for DNS

top-level domain followed by transforming the characters to lower case, otherwise, results in a regularization issue [19]. In second step, a vocabulary is formed by assigning a unique id to each character. The unknown characters are assigned to default id 0. These unique ids of vectors are passed batch-size of 64 to embedding layer. An embedding layer facilitates to learn the semantics and contextual similarity structures of domain names by coordinating with the other layers in the deep network during optimizing in the backpropogation process. The high-dimensional vectors of embedding layer are passed to t-SNE [45] for visualizing the character clustering. The Fig. 9 showed that the similar characters are clustered together. Most importantly, the special characters and numbers are appeared in a separate cluster. Thus, the embedding layer has learnt the semantic and contextual similarity of domain names. Finally, the embedding layer output is passed to the other layers such as (1) RNN (2) LSTM (3) GRU (4) I-RNN (5) CNN (6) CNN-LSTM. Based on the parameter tuning, the number of units/memory blocks is set to 128 for recurrent hidden layers, 64 filters with filter length 3 for CNN. These layer obtains the optimal feature representation for classifying the domain name as either benign or DGA generated. Finally, the various RNN layers output is passed to the dropout layer. This facilitates to avoid the state of over fitting by randomly removing the neurons with its connections to other neurons.

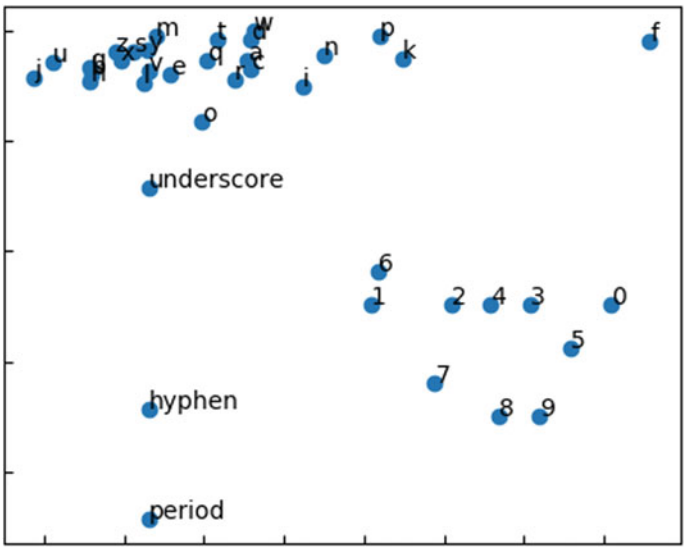

Fig. 9 Embedded character vectors learned by LSTM model is represented using 2-dimensional linear projection (PCA) with t-SNE

2. Classification: The dropout layer output is passed to the dense layer. A dense layer is a fully-connected layer and it composed of two layers. One is dense with unit 1 and followed by an activation layer i.e. *sigmoid* with loss function as binary cross-entropy, as shown below.

$$loss(pr, ex) = -\frac{1}{N}\sum_{j=1}^{N}\left[ex_j \log pr_j + (1 - ex_i)\log\left(1 - pr_j\right)\right]$$

Here *ex* is a vector of expected class label, *pr* is a vector of predicted probability for all domain names in testing data set.

3. Evaluation results: All the deep learning architectures are trained using the most recent software framework Google's open source data flow engine, TensorFlow [46]. TensorFlow allows programmers to build numerical systems as unified data flow graphs. The data flow graph has nodes and edges that represent mathematical operations and the tensors respectively. In addition, programmers can also deploy computations on heterogeneous platforms: one or more CPUs, GPU, or mobile devices. To accelerate the gradient descent computations all experiments are run on GPU enabled TensorFlow in single NVidia GK110BGL Tesla k40. The performance of the trained model was evaluated on the testing data set. LSTM and CNN-LSTM have followed improvement in accuracy till

Table 1 Test results for detecting DGA

Algorithm	Accuracy	Precision	Recall	F-score	Loss
LSTM	0.999	0.999	0.998	0.999	0.00
GRU	0.998	0.991	0.996	0.993	0.01
CNN-LSTM	0.997	0.985	0.996	0.990	0.01
RNN	0.968	0.795	0.943	0.863	0.09
I-RNN	0.979	0.866	0.966	0.913	0.06
CNN	0.965	0.777	0.936	0.849	0.10
Bigram-LR	0.937	0.577	0.892	0.701	0.16
Hand-crafted features					
RF	0.926	0.483	0.858	0.618	
DT	0.908	0.564	0.966	0.712	
MT	0.892	0.483	0.951	0.641	
AB	0.924	0.664	0.936	0.777	
NB	0.909	0.564	0.967	0.712	

epochs 700. After, accuracy has seen a sudden decrease due to over fitting. IRNN has performed well till epochs 400. The performance of both CNN and GRU is good till epochs 250 epochs. RNN has started to over fitting once it reaches epochs 800. This infers that each deep layer has required different number of epochs to attain the best performance in classifying the domain name as benign or DGA generated. To compare the performance of deep learning models with the traditional machine learning classifiers [47, 48], we followed the feature engineering approach. The detailed performance of traditional machine learning and deep learning algorithms performance is reported in Table 1. Deep learning models performed well in comparison to the traditional machine learning classifiers. Moreover, LSTM has performed well in comparison to the other deep layer (Fig. 10).

To verify the accuracy of the built model, the binary cross-entropy is calculated. The binary cross-entropy is a standard measure used to identify the inefficiency of the classifier to predict the data based on the premise truth. In other words, binary cross-entropy is a cost function used to measure the inaccurateness of the built model. To minimize binary cross-entropy, the model employs gradient descent algorithm with a learning rate of 0.01. Gradient descent is a first order optimization algorithm, where the model shifts each variable a little bit in the direction that reduces the cost. The classifier is trained with both malicious and non-malicious datasets. If the classifier suggests the domain d_i to be randomly generated, a flag is assigned to that domain d_i as a DGA domain name and mark the host h_i as infected. The domain d_i and the host IP along with the timestamp t_i is then stored in the distributed database for further analysis. Figure 11 illustrates the detailed implementation of the above process.

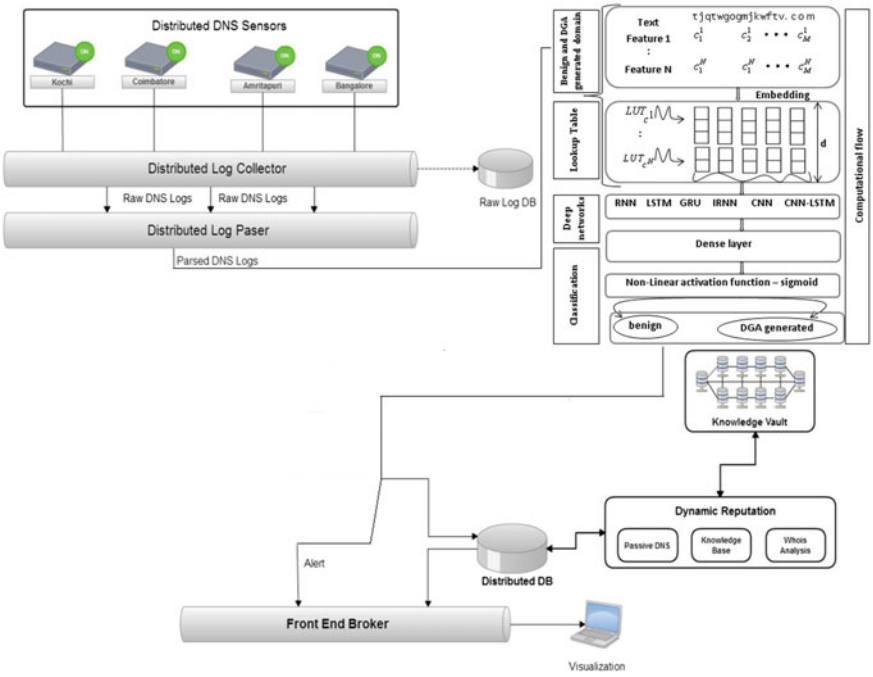

Fig. 10 An intuitive overview of employed deep learning architectures

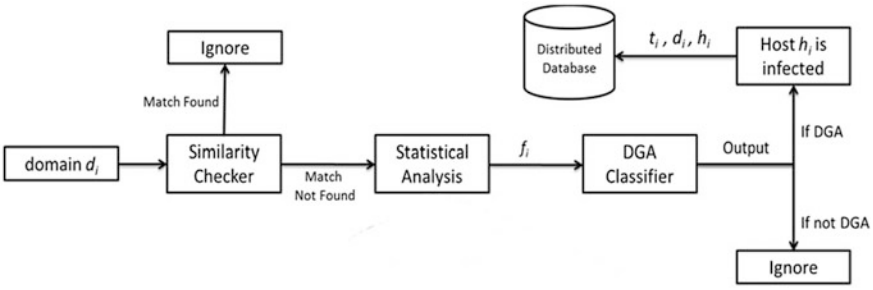

Fig. 11 Detailed diagram of identifying DGA infected hosts module

5.4 C&C Detection

The Deep Bot framework goes a step further by analyzing the response queries of the infected hosts h_i. The idea behind is to find domain names with similar patterns as DGA domains and getting the IP address it actually resolves to. The entire domains that got resolved when queried by an infected host are analyzed. For this analysis, the same procedure of passing it through similarity checker then statistical

analyzer followed by the DGA classifier is followed. However, the output of this analysis also includes some false positive domains that could be also termed as noise. The presence of these noisy domains reduces the efficiency of the entire system. To avoid this, a noise filter is employed. In noise filtering two analyzes are performed namely: Abbreviation Checker and 3-Way Scoring, which is explained in detail bellow:

(a) Abbreviation Checker: Here the presences of any abbreviations are checked inside a domain name. Generally, an abbreviation is a short form of a committee or organization. In this analysis, not only check the presence of an abbreviation term but also perform three statistical analyses mentioned bellow:

- Length ratio of the abbreviation to the domain name.
- Position of the abbreviation.
- Presence of any date before or after the abbreviation.

(b) 3-Level Reputation Scoring: Given a domain name or an IP address, a reputation score was annotated to it accordingly. The reputation score is based on a 3-level check which involves spanning through the Passive DNS Intelligence, Malware Knowledge Base and WHOIS Data Base.

1. Malware Knowledge Base: A knowledge base is created by continuous crawling of public block lists, blacklists, online malware dumps and information related to known botnet IPs and domains from open Internet.
2. Passive DNS Intelligence: Passive DNS or pDNS is the methodology of constructing the duplicate copies of zone data from the captured name server responses. For the collection of pDNS data, the preliminary condition is to capture the inter-server DNS message by multiple sensors and then to forward all the data to collection point for analysis. Once the analysis on the captured data is over, every individual DNS record is stored in a database from where data lookup could be performed for any analysis.
3. WHOIS Database: WHOIS provides information associated with a Domain or IP Address. Queries will be forwarded to corresponding WHOIS server for that particular Domain/IP address. The information related to domain and IP are explained below: For a domain, query is sent to retrieve the following information:

- Registrant name, address, email, phone.
- Registrar name, address, email, phone.
- Administrator name, address, email, phone.
- Domain registered date, expiration date, authoritative name server etc.

For an IP address the fields are:

- Owner
- Prefix (Network)
- ASN
- Location
- Expiration Date etc.

Higher the reputation score of a particular domain, lesser are its chances of being malicious. Based on these measure it was understood that the history of this IP and have a say whether it could be malicious or not. Using the noise filter a lot of false positives could be reduced significantly. The final output is the malicious domain name of the C&C server along with the IP address it resolved to. Both these details along the timestamp containing the query time is then stored back to the distributed database.

5.5 Time Based Analysis

Deep Bot additionally performs time based to analyze the behavioral patterns of the C&C server. For this Deep Bot analyzes the timestamps of the entire DGA requests made by an infected host. Based on the query patterns with which a host queries, Deep Bot classify it into three groups namely,

1. Continuous: When the host is generating queries at an equal interval of time.
2. Quick: When a lot of queries are generated in a small period of time.
3. Random: Host queries DGA requests at random time periods.

By classifying the host into these three categories, the nature of the C&C server can be understood. Also one could know the number of hosts belonging in a particular category. Now analyses the domains (both NX and resolved ones) queried by different host residing in the same category. If any similar patterns are found in the domains queried by different hosts, it can be concluded that both the hosts are infected by the same DGA and thus controlled by the same bot-master. All such hosts and the domains are clustered accordingly.

5.6 Case Studies

In this section, three case studies of DGA based domains and its resolved server details as per the DNS data collected on 16th May, 2016 are discussed.

1. Case Study 1: IP—xxx.xxx.213.30 The network activity of all domains contacting a sinkhole server was monitored for a day. Sinkhole servers are generally operated by multiple security organizations whose tasks are to monitor and

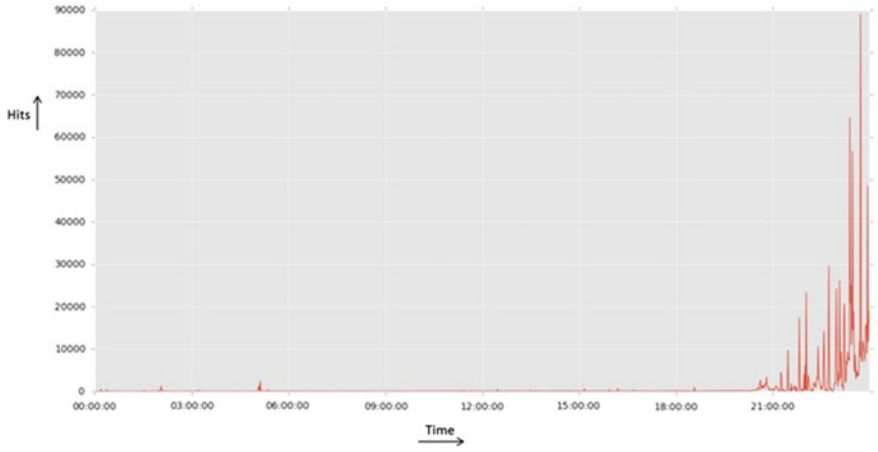

Fig. 12 Hits gained by IP xxx.xxx.213.30 on 16th May, 2016

mitigate the malware infections propagating in a network [49]. This is achieved by redirecting the malicious traffic to trusted and reputed hosts and analyzing them scrupulously. On 16th May, 2016 this sinkhole IP received approximately 1.1 million hits from more than 9 k unique clients. The timeline of the hits received is illustrated in Fig. 12.

There was a sudden surge in the hits received after 9 PM. Figure 13 shows DGA network queries made by a client to this sinkhole IP address. The client also queried a multiple DGA domains, which resolved to a Portugal IP xxx.xxx.26.248. On analysis the DeepBot framework was able to identify the similar patterns between queries made by this client. A vast majority of the domains started with a prefix HLD 'five' followed by a one or two digit number and ended with either '.ru' or '.com' TLD. Some of the DGA domain patterns observed is illustrated in Tables 2 and 3 respectively.

2. Case Study 2: Infected Host IP—xxx.xxx.153.192 The DGA like domains queried by host xxx.xxx.153.192 were monitored. Figure 14 illustrates the DGA domains and its resolved IP addresses identified by DeepBot framework.

The IP address xxx.8.69.25 is a conflicker sinkhole IP hosted in China. Some conflicker domains resolving to this IP address on 16th May, 2016 is provided in Table 4.

3. Case Study 3: Domain Name—hzmksreiuojy.biz: DeepBot framework also found out another conflicker set of domains with the prefix 'hzmksreiuojy' on May 16, 2016. Details of these domains are provided in Fig. 15 and are tabulated in Table 5. DeepBot went also found out many suspicious domains connected to xxx.xxx.249.128. Some of them are shown in Table 6

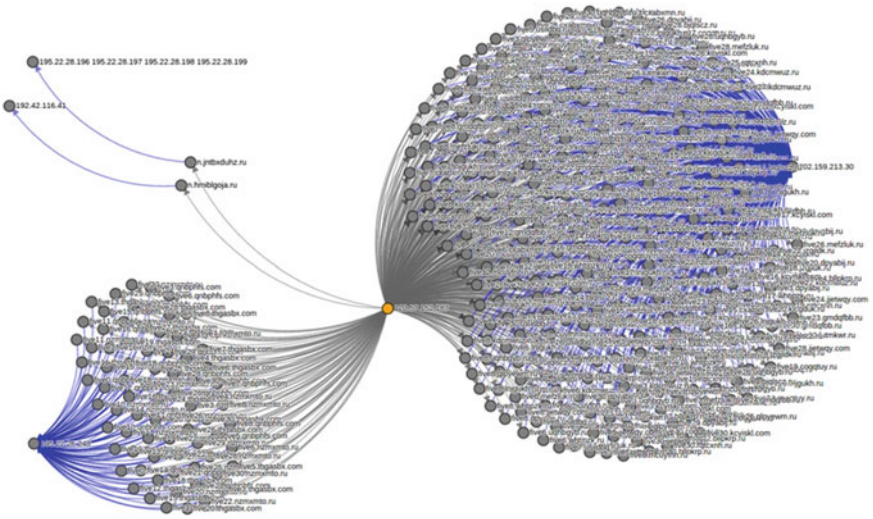

Fig. 13 DGA queries made by a client to the IP xxx.xxx.213.30

Table 2 DGA domains starting with prefix 'five' resolving to 202.159.213.30

five30.lnjgukh.ru five30.jzgjldk.ru five30.mcuyfnh.ru five30.usildbq.ru
five25.cxabxmn.ru five25.bjqlscz.ru five25.jzgjldk.ru five25.qlpyewm.ru
five3.klcgduk.ru five3.whtjpzk.ru five3.jjetwqy.com five3.coqqtuy.ru
five12.usildbq.ru five12.kcyiskl.com five12.gmdqfbb.ru five12.cxabxmn.ru
five6.jjetwqy.com five6.kcyiskl.com five6.uqhbgyb.ru five6.bjqlscz.ru
five28.dpyabij.ru five28.whtjpzk.ru five28.kdcmwuz.ru five28.mefzluk.ru

Table 3 DGA domains starting with prefix 'green' resolving to 202.159.213.30

green32.qfmtsvxp.ru green28.qfmtsvxp.ru green5.qfmtsvxp.ru green13.qfmtsvxp.ru
green32.ztcgyuyh.ru green28.ztcgyuyh.ru green5.ztcgyuyh.ru green13.ztcgyuyh.ru
green32.entggmuq.ru green28.entggmuq.ru green5.entggmuq.ru green13.entggmuq.ru
green32.rsfdhhez.ru green28.rsfdhhez.ru green5.rsfdhhez.ru green13.rsfdhhez.ru
green32.czectdfl.ru green28.czectdfl.ru green5.czectdfl.ru green13.czectdfl.ru

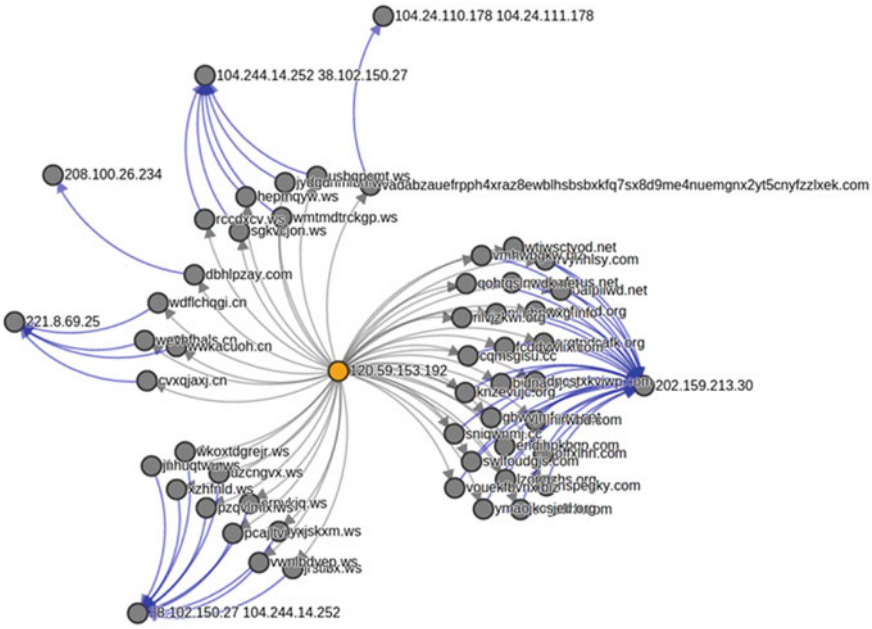

Fig. 14 DGA domains queried by infected host xxx.xxx.153.192

Table 4 Conflicker domains observed on 16th, May 2016

Domain name	Resolved IP address	Name server
abhumk.cn	xxx.xxx.69.25	ns.conflicker-sinkhole.cn
aidmj.cn		
akkbzhqnfv.cn		
aapvv.cn		
adphpkx.cn		
afxddfgcmqq.cn		
bmiusjs.cn		
achrszijz.cn		
agwjrlfycyf.cn		
aidjgjsr.cn		
bdvuzzi.cn		
atdja.cn		
bdgnsjdqcdo.cn		

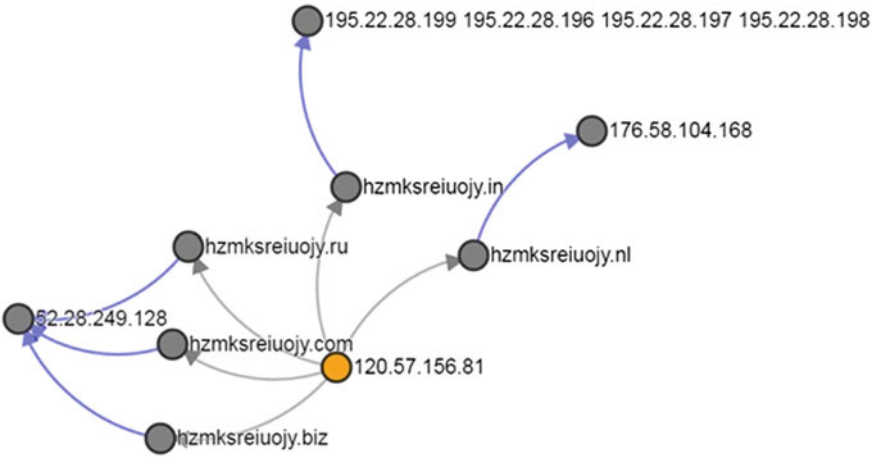

Fig. 15 Conflicker domains with prefix 'hzmksreiuojy'

Table 5 DNS Information of domains with prefix 'hzmksreiuojy' as on 16th May, 2016

Host IP	Domain name	Resolved IP	Name server	Blacklisted
120.57.156.81	hzmksreiuojy.biz	52.28.249.128	ns.conficker-sinkhole.com ns.conficker-sinkhole.net ns.conficker-sinkhole.org	Yes
	hzmksreiuojy.ru		ns1.101domain.com ns2.101domain.com ns5.101domain.com	No
	hzmksreiuojy.com		ns1.dynadot.com ns1.dynadot.com	Yes
	hzmksreiuojy.in	195.22.28.196 195.22.28.197 195.22.28.198 195.22.28.199	a0.in.afilias-nst.in a1.in.afilias-nst.info a2.in.afilias-nst.info b0.in.afilias-nst.org b1.in.afilias-nst.in b2.in.afilias-nst.org c0.in.afilias-nst.info ns1.csof.net ns4.csof.net	Yes
	hzmksreiuojy.nl	176.58.104.168	sinkhole.sidnlabs.nl proteus.sidnlabs.nl	Yes

Table 6 Domains resolving to IP 52.28.249.128 observed on 16th May, 2016

0-0-1-4-5-5-2-5-7-8-0-7-4-2-2-0-2-7-8-0-2-8-4-4-3-4-0-3-8-2-2- .0-0-0-0-0-0-0-0-0-0-0-0-0-49-0-0-0-0-0-0-0-0-0-0-0-0.info
0-0-2-4-3-5-4-7-3-4-3-7-6-1-5-4-4-3-7-0-8-1-4-2-8-8-7-8-0-5-7- .0-0-0-0-0-0-0-0-0-0-0-0-0-49-0-0-0-0-0-0-0-0-0-0-0-0.info
0-0-5-6-4-3-0-5-0-6-8-6-2-8-2-7-7-0-2-2-1-6-1-6-5-7-8-2-5-1-4- .0-0-0-0-0-0-0-0-0-0-0-0-0-49-0-0-0-0-0-0-0-0-0-0-0-0.info
0-0-7-2-1-0-5-7-5-6-2-3-1-5-7-2-2-8-8-7-3-5-5-3-5-7-0-6-7-1-5- .0-0-0-0-0-0-0-0-0-0-0-0-0-53-0-0-0-0-0-0-0-0-0-0-0-0.info
0-0-8-7-5-7-0-7-8-6-8-3-8-4-4-6-0-0-6-4-0-1-1-3-6-7-7-7-7-3-8- .0-0-0-0-0-0-0-0-0-0-0-0-0-49-0-0-0-0-0-0-0-0-0-0-0-0.info
0-1-1-6-4-6-0-0-0-1-1-8-3-2-8-2-4-5-0-7-5-8-4-1-8-5-3-1-8-8-6- .0-0-0-0-0-0-0-0-0-0-0-0-0-53-0-0-0-0-0-0-0-0-0-0-0-0.info
0-1-5-8-7-7-7-4-1-7-1-0-2-7-4-7-6-3-4-5-8-2-8-0-1-5-6-3-8-5-5- .0-0-0-0-0-0-0-0-0-0-0-0-0-49-0-0-0-0-0-0-0-0-0-0-0-0.info
0-1-6-7-6-4-0-0-4-4-4-6-2-5-8-2-5-6-5-4-6-2-3-2-5-7-5-4-5-3-2- .0-0-0-0-0-0-0-0-0-0-0-0-0-49-0-0-0-0-0-0-0-0-0-0-0-0.info
0-1-7-1-6-2-4-7-7-1-7-7-6-6-6-4-2-4-7-3-5-1-7-7-7-7-2-3-0-3-6- .0-0-0-0-0-0-0-0-0-0-0-0-0-53-0-0-0-0-0-0-0-0-0-0-0-0.info
0-1-7-3-7-4-5-6-6-6-7-4-8-1-7-7-2-4-1-4-5-1-1-1-8-5-3-7-3-4-2- .0-0-0-0-0-0-0-0-0-0-0-0-0-49-0-0-0-0-0-0-0-0-0-0-0-0.info
0-2-8-1-4-1-3-5-8-3-2-4-5-8-0-5-5-8-6-3-1-2-1-7-3-7-7-8-0-6-4- .0-0-0-0-0-0-0-0-0-0-0-0-0-53-0-0-0-0-0-0-0-0-0-0-0-0.info
0-2-8-1-8-4-6-0-6-1-5-2-4-6-8-2-3-5-4-7-4-8-1-0-5-2-2-5-4-7-2- .0-0-0-0-0-0-0-0-0-0-0-0-0-53-0-0-0-0-0-0-0-0-0-0-0-0.info
0-3-0-1-7-4-3-0-8-1-2-5-6-7-5-2-2-5-8-8-6-7-7-4-3-4-3-2-3-4-1- .0-0-0-0-0-0-0-0-0-0-0-0-0-49-0-0-0-0-0-0-0-0-0-0-0-0.info
0-3-0-7-4-1-7-6-5-7-3-0-8-8-6-3-2-6-3-5-8-2-8-1-6-8-7-6-4-8-8- .0-0-0-0-0-0-0-0-0-0-0-0-0-49-0-0-0-0-0-0-0-0-0-0-0-0.info
0-3-4-6-1-8-8-3-0-7-2-7-5-4-6-1-3-3-3-3-3-0-6-8-4-0-8-8-5-1-5- .0-0-0-0-0-0-0-0-0-0-0-0-0-53-0-0-0-0-0-0-0-0-0-0-0-0.info
0-4-1-4-4-1-1-5-3-8-3-2-6-1-8-5-5-0-5-3-4-0-5-3-2-1-2-8-7-8-0- .0-0-0-0-0-0-0-0-0-0-0-0-0-49-0-0-0-0-0-0-0-0-0-0-0-0.info

6 Conclusion, Future Work and Limitations

In the current research work, situational awareness on Cyber Security threats have been thoroughly studied and created a highly scalable framework for the same. This framework analyses DNS event data as its input without making any significant architectural changes to the networking infrastructure. This distributed framework is highly scalable to combine and correlate the attack information from several Tier-1 service provider networks. This is the first of its kind framework to employ deep learning techniques for handling very large scale data to be processed in near real-time. This framework has been compared with other commercially available

solutions and found that this framework outperforms (in terms of features and performance) all the existing solutions. Due to the confidential nature of the research, details of the comparison cannot be disclosed. To the best of our knowledge, this is the only framework that works across several Internet Service Providers as a single unified system providing situational awareness across the country.

The current research does not include malware binary analysis that provides detailed information on the structure and behavior of the malware. Since the framework does not have access to the end hosts, the damage done to the end systems cannot be measured. Also, the framework will not be able to detect the malicious communications that avoids DNS by using direct IP addresses for their communication. This limitation can be mitigated by the inclusion Net Flow. As a future direction, the framework will be enhanced by adding multi-lingual Internationalized Domain Names (IDN) domain name support as it does not perform the analysis of IDN based domain names. Though we developed a distributed platform for collecting and correlating the BGP events, the analysis is not done.

Acknowledgements This research was supported in part by Paramount Computer Systems and Ministry of Electronics and Information Technology (MeitY), Government of India. We are also grateful to NVIDIA India, for the GPU hardware support to research grant. We are grateful to Computational Engineering and Networking (CEN) department for encouraging the research.

References

1. Abu Rajab, M., Zarfoss, J., Monrose, F., & Terzis, A. (2006). A multifaceted approach to understanding the botnet phenomenon. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement* (pp. 41–52). ACM.
2. Antonakakis, M., Perdisci, R., Dagon, D., Lee, W., & Feamster, N. (2010). Building a dynamic reputation system for DNS. In *USENIX Security Symposium* (pp. 273–290).
3. Ollmann, G. (2009). Botnet communication topologies. Retrieved September 30, 2009.
4. Foster, K. (2010). *The conicker worm and variants*.
5. Torpig. (2016). Retrieved January 11, 2016 from http://en.wikipedia.org/wiki/Torpig.
6. Royal, P. (2008). Analysis of the kraken botnet. *Damballa*, Apr 9.
7. Looking back at murofet, a zeusbot variant's active history. (2015). Wikipedia: The Free Encyclopedia. Wikimedia Foundation, Inc. Retrieved August 1, 2014 from https://blog.dambella.com/archives/1008.
8. Crawford, H., & Aycock, J. (2008). Kwyjibo: Automatic domain name generation. *Software: Practice and Experience*, *38*(14), 1561–1567.
9. Antonakakis, M., Perdisci, R., Nadji, Y., Vasiloglou, N., Abu-Nimeh, S., Lee, W., & Dagon, D. (2012). From throw-away traffic to bots: Detecting the rise of dga-based malware. In *Presented as part of the 21st USENIX Security Symposium (USENIX Security 12)* (pp. 491–506).
10. Will, C. (2014) Botnet detection with dns monitoring. *Network*, *25*.
11. Schiavoni, S., Maggi, F., Cavallaro, L., & Zanero, S. (2014). Phoenix: Dga-based botnet tracking and intelligence. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment* (pp. 192–211). Springer.

12. Raghuram, J., Miller, D. J., & Kesidis, G. (2014). Unsupervised, low latency anomaly detection of algorithmically generated domain names by generative probabilistic modeling. *Journal of Advanced Research, 5*(4), 423433.
13. Thomas, M., & Mohaisen, A. (2014). Kindred domains: detecting and clustering botnet domains using DNS traffic. In *Proceedings of the 23rd International Conference on World Wide Web* (pp. 707–712). ACM.
14. Ashwini, B., Menon, V. K., & Soman, K. P. (2016). Prediction of malicious domains using smith waterman algorithm. In *International Symposium on Security in Computing and Communication* (pp. 369–376). Singapore: Springer.
15. Zdrnja, B., Brownlee, N., & Wessels, D. (2007). Passive monitoring of dns anomalies. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment* (pp. 129–139). Springer.
16. Ramachandran, A., & Feamster, N. (2006). Understanding the network-level behavior of spammers. In *ACM SIGCOMM Computer Communication Review* (vol. 36, no. 4, pp. 291–302). ACM.
17. Anderson, D. S., Fleizach, C., Savage, S., & Voelker, G. M. (2007). Spamscatter: Characterizing internet scam hosting infrastructure. In *Usenix Security* (pp. 1–14).
18. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436–444.
19. Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. *Advances in Neural Information Processing Systems*.
20. Elman, J. L. (1990). Finding structure in time. *Cognitive Science, 14*(2), 179211.
21. Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks, 5*(2), 157166.
22. Martens, J. (2010). Deep learning via hessian-free optimization. In *Proceedings of 27th International Conference on Machine Learning*.
23. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780.
24. Cho, K., van Merrienboer, B., Gulcehre, C., Bougares, F., Schwenk, H., & Bengio, Y. (2014). *Learning phrase representations using rnn encoderdecoder for statistical machine translation*. arXiv:1406.1078, http://arxiv.org/abs/1406.1078.
25. Le, Q. V., Jaitly, N., & Hinton, G. E. (2015). *A simple way to initialize recurrent networks of rectified linear units*. arXiv:1504.00941 (2015).
26. Talathi, S. S., & Vartak, A. (2015). *Improving performance of recurrent neural network with relu nonlinearity*. arXiv:1511.03771.
27. Anstee Darren, C. F. C. P. B., & Sockrider, G. (2015). *Worldwide infrastructure security report*.
28. Vinayakumar, R., Soman, K. P., Poornachandran, P., & Sachin Kumar, S. Detecting android malware using long short-term memory-LSTM. *Journal of Intelligent and Fuzzy Systems*, IOS Press [In press].
29. Vinayakumar, R., Soman, K. P., & Poornachandran, P. (2017). Deep android malware detection and classification. In *International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2017* (pp. 1677–1683). IEEE.
30. Vinayakumar, R., Soman, K. P., Poornachandran, P., & Sachin Kumar, S. Evaluating deep learning approaches to characterize and classify the DGAs at scale. *Journal of Intelligent and Fuzzy Systems*, IOS Press [In press].
31. Vinayakumar, R., Soman, K. P., & Poornachandran, P. Detecting malicious domain names using deep learning approaches at scale. *Journal of Intelligent and Fuzzy Systems*, IOS Press [In press].
32. Vinayakumar, R., Soman, K. P., & Poornachandran, P. (2017). Evaluating shallow and deep networks for secure shell (ssh) traffic analysis. In *International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2017* (pp. 266–274). IEEE.
33. Vinayakumar, R., Soman, K. P., & Poornachandran, P. (2017). Secure shell (ssh) traffic analysis with flow based features using shallow and deep networks. In *International*

Conference on Advances in Computing, Communications and Informatics (ICACCI), 2017 (pp. 2026–2032). IEEE.
34. Vinayakumar, R., Soman, K. P., & Poornachandran, P. Evaluating deep learning approaches to characterize, signalize and classify malicious URLs. *Journal of Intelligent and Fuzzy Systems*, IOS Press [In press].
35. Vinayakumar, R., Soman, K. P., & Poornachandran, P. (2017). Applying convolutional neural network for network intrusion detection. In *International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2017* (pp. 1222–1228). IEEE.
36. Vinayakumar, R., Soman, K. P., & Poornachandran, P. (2017). Evaluating effectiveness of shallow and deep networks to intrusion detection system. In *International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2017* (pp. 1282–1289). IEEE.
37. Vinayakumar, R., Soman, K. P., & Poornachandran, P. (2017). Long short-term memory based operation log anomaly detection. In *International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2017* (pp. 236–242). IEEE.
38. Vinayakumar, R., Soman, K. P., Velan, K. S., & Ganorkar, S. (2017). Evaluating shallow and deep networks for ransomware detection and classification. In *International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2017* (pp. 259–265). IEEE.
39. Vinayakumar, R., Soman, K. P., & Poornachandran, P. (2017). Deep encrypted text categorization. In *International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2017* (pp. 364–370). IEEE.
40. Vinayakumar, R., Soman, K. P., & Poornachandran, P. (2017). Applying deep learning approaches for network traffic prediction. In *International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2017* (pp. 2353–2358). IEEE.
41. Tripwire, google's malaysian domains hit with DNS cache poisoning attack. (2013). Retrieved October, 2013 from http://www.tripwire.com/state-of-security/top-security-stories/googlesmalaysian-domainshit-dns-cache-poisoning-attack/.
42. Alexa-the top 500 sites on the web. (2014). Retrieved October 10, 2014 from http://www.alexa.com/topsites.
43. Hall, P. A., & Dowling, G. R. (1980). Approximate string matching. *ACM Computing Surveys (CSUR), 12*(4), 381–402.
44. Dameraulevenshtein distance. (2014). Retrieved December 12, 2014 from http://en.wikipedia.org/wiki/DamerauLevenshtein.
45. Van der Maaten, L., & Hinton, G. (2008). Visualizing data using T-Sne. *Journal of Machine Learning Research, 9*(2579–2605), 85.
46. Abadi, M., et al. (2016). TensorFlow: A system for large-scale machine learning. In *OSDI* (Vol. 16).
47. Soman, K. P., Loganathan, R., & Ajay, V. (2009). *Machine learning with SVM and other kernel methods*. Ltd: PHI Learning Pvt.
48. Soman, K. P., Diwakar, S., & Ajay, V. (2006). *Data mining: Theory and practice [WITH CD]*. Ltd: PHI Learning Pvt.
49. Kuhrer, M., Rossow, C., & Holz, T. (2014). Paint it black: Evaluating the effectiveness of malware blacklists. In *International Workshop on Recent Advances in Intrusion Detection* (pp. 1–21). Springer.