Anomaly Behavior Analysis of DNS Protocol

Pratik Satam[1]*, Hamid Alipour[2], Youssif Al-Nashif[3], and Salim Hariri[1]
[1]Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ 85721 USA
pratiksatam@email.arizona.edu, hariri@ece.arizona.edu
[2]Cloud Identity Services and Security Division, Microsoft, Redmond, WA 98052 USA
hra@email.arizona.edu
[3]Department of Computer Engineering, Florida Polytechnic University, Lakeland, Fl-33805
yalnashif@flpoly.org

Abstract

DNS protocol is critically important for secure network operations. All networked applications request DNS protocol to translate the network domain names to correct IP addresses. The DNS protocol is prone to attacks like cache poisoning attacks and DNS hijacking attacks that can lead to compromising user's accounts and stored information. In this paper, we present an anomaly based Intrusion Detection System (IDS) for the DNS protocol (DNS-IDS) that models the normal operations of the DNS protocol and accurately detects any abnormal behavior or exploitation of the protocol. The DNS-IDS system operates in two phases, the training phase and the operational phase. In the training phase, the normal behavior of the DNS protocol is modeled as a finite state machine where we derive the temporal statistics of normal DNS traffic. Then we develop an anomaly metric for the DNS protocol that is a function of the temporal statistics for both the normal and abnormal transitions of the DNS protocol. During the operational phase, the anomaly metric is used to detect DNS attacks (both known and novel attacks). We have evaluated our approach against a wide range of DNS attacks (DNS hijacking, Kaminsky attack, amplification attack, Birthday attack, DNS Rebinding attack). Our results show attack detection rate of 97% with very low false positive alarm rate (0.01397%), and round 3% false negatives.

Keywords: DNS, Intrusion detection system, Anomaly detection, Machine learning, Data mining, Supervised training.

1 Introduction

Domain Name System (DNS) is used to associate a domain name with an IP address. It is one of the mostprominently used protocols over the internet, used by for domain name resolution by users during user to user communication and server to server communication. With the current trends in the growth of Internet of Everything (IOE), wherein all user appliances ranging from cars to microwave ovens will be connected to the internet, the DNS protocol is set to play an even more important role.

The DNS protocol is a query/reply based protocol where the authenticity of the reply is not confirmed or confirmed by approaches that can be thwarted easily. One of the distinguishing features of the DNS protocol is caching of the DNS replies in local DNS servers, so that future look up of these domains can be achieved with minimum delay. This feature of the DNS protocol is exploited by attacks like the Kaminsky[6] attack shown by Dan Kaminsky. The Kaminsky attack showed that cache poisoning could be used to hijack not only a domain but also a complete zone. Attacks on the DNS protocol are not limited to cache poisoning. For example, DNS amplification[8, 14] attack does not involve cache poisoning

*Corresponding author: Tel: +1-520-621-9915 but can be used to attack DNS servers. Attacks like the DNS tunneling attack could be used to setup a secret communication link over the internet.

DNS protocol has many revisions. Some of the important modifications to the protocol were the randomization of the source port and transaction id as a means to thwart the Kaminsky attack. Also secured versions of the DNS protocol (e.g. DNSSEC [2]) have also been introduced. As a part of the DNSSEC, the extensions added to the resolver the ability to perform origin authentication and data integrity on the received replies. But the use of this secured protocol comes with problems like, broken validations which may result into a denial of service[10]. Moreover the adoption of the DNSSEC protocol causes a significant increase in DNS traffic over the network. This increase in the traffic is not justified by the small amount of security improvement that the protocol provides[7].

In this paper we present an anomaly based intrusion detection system to protect the operations of the DNS protocol against cyber attacks. The approach uses supervised machine learning to understand the normal behavior of the protocol and then uses this understanding of the normal behavior of the protocol to detect attacks. Our approach can detect existing as well as new or modified DNS attacks with low false positive alarm rates (less than 0.01397%). The paper is organized into the following sections: Section 2 reviews briefly the DNS protocol, Section 3 discusses the related work, Section 4 presents our DNS IDS approach, Section 5 presents our experimental results, and Section 6 presents conclusion and future research direction.

2 DNS Protocol

DNS is a distributed naming system that utilizes a set of hierarchically connected DNS servers while maintaining a client server model. It is used to translate the human readable domain names to IP addresses. The DNS passes the queries hierarchically over a tree like domain space network and the query for a particular domain travels up the network till it reaches the authoritative server for the requested domain. Then the server sends a reply down the tree to the requesting source with a DNS reply. This tree network is subdivided into smaller networks called zones beginning at the root zone. Each zone has at least one authoritative name server that provides authoritative replies for that zone. A zone may contain multiple domains. An authoritative server may sometimes delegate its task of answering queries for a domain in its zone to some other DNS server that is authoritative to a sub zone. Generally the host of a network has a list of root authoritative servers. These servers are used as a query initiation point. These servers then pass the query up the tree until it reaches the correct authoritative DNS server which then generates a reply to the query and sends it back to the local DNS server which in turn sends the reply back to the requesting machine's resolver. Generally depending on the Time to Live(TTL) of the DNS reply and the type of the DNS server involved, the reply that a server receives is cached by the DNS server for future query replies. The TTL of the cached records is reduced with time. This cached record of DNS query-reply is flushed out when the TTL of that particular record reduces to zero. A DNS server can be configured to query recursively. It can also be configured to just forward the queries that it receives. Generally a DNS query is of a particular type (example Type A- 32 bit IP address). A list of the general queries in a network is given in Table 1.

Table 1: Common DNS message types

Type	Description
A	A 32-bit IPv4 address, most commonly used to map host-names to the IP address of the host
AAA	A 128-bit IPv6 address, most commonly used to map host-names to the IP address of the host
NS	Specifies the authoritative name servers for related zones.
CNAME	Alias of one name to another canonical name. The DNS lookup will continue by retrying the lookup with the new canonical name
MX	Maps a domain name to a list of message exchange agents for that domain.
SOA	Specifies authoritative information about a DNS zone, including the primary name server, the email of the domain administrator, the domain serial number, and several timers relating to refreshing the zone.
TXT	Originally for arbitrary human-readable text in a DNS record.
PTR	Pointer to a canonical name. Unlike a CNAME, in PTR DNS processing does not proceed, just the name is returned. The most common use is for implementing reverse DNS lookups.

The reply to a query can be of any size. If the reply to a query is more than the size of the MTU then the reply is fragmented into multiple packets. At the user end, a DNS resolver is responsible for resolving the DNS information for a particular system. A resolver may be implemented by an operating system or locally by a web browser. A resolver is generally configured to cache the replies to the queries that it sends over the network for future use till the TTL for the reply is greater than zero.
Typically, the DNS protocol operates over the UDP. The DNS packet has a header as shown in Figure 1.

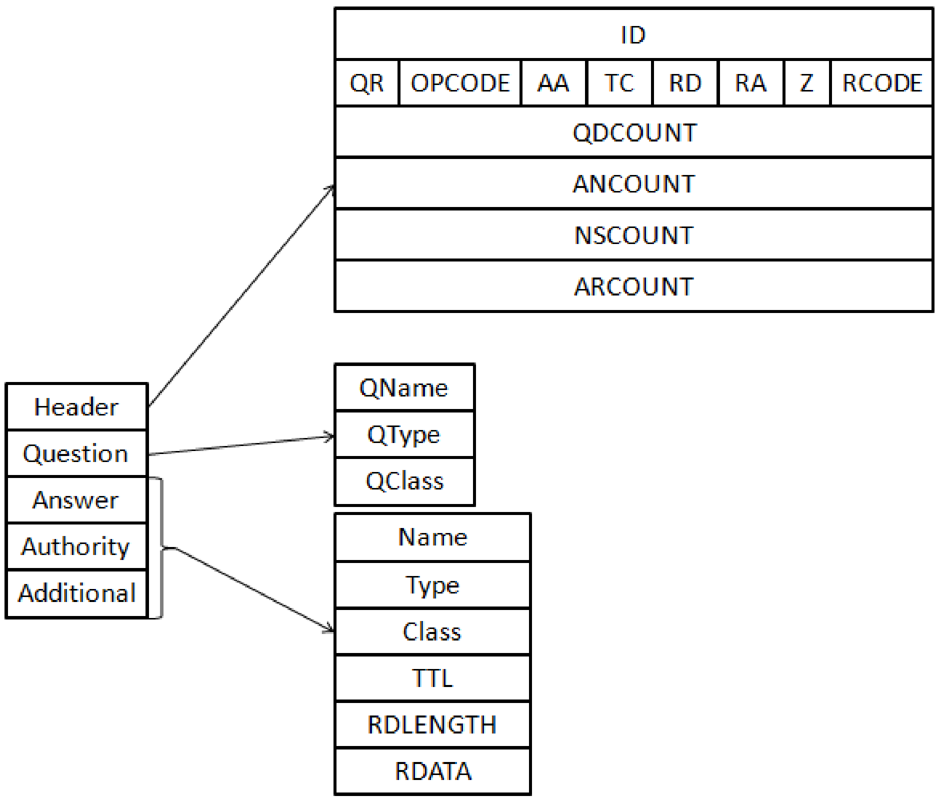

Figure 1: DNS message Format

3 Related Work

In general there are two main approaches to protect the DNS protocol against cyber attacks. The first approach is the use of DNSSEC[7] in which the DNS protocol has been significantly changed by adding security features The second approach is to use an intrusion detection system(IDS) to detect attacks that are launched over the network[17, 16, 18]. The use of DNSSEC to secure the DNS protocol has certain problems like broken validations which give results similar to self DoS[10], and the system generates large number of DNS responses for a small increase in the DNS protocol security[7].

The goal of the IDS methods is to analyze normal and abnormal behavior of the DNS traffic to detect threats against the DNS protocol [17, 16, 18, 9]. Since our approach focuses on detecting attacks against the DNS protocol, we will focus our analysis at different levels of the Domain Name hierarchy during a period of time in order to model the temporal transitions of normal DNS traffic. Other techniques focused on developing statistical models for the DNS traffic without any detection approach. But there are other methods [17, 16, 9] which have proposed anomaly detection mechanisms for DNS protocol. In what follow, we highlight the main approach used by these techniques.

In [17], normal traffic of DNS protocol is monitored and some statistical thresholds for each protocol parameter are calculated during the training phase. Then during the detection phase, values of these parameters will be compared with the specified thresholds to detect abnormal traffic. This approach will be heavily influenced by the traffic context. For example considering the time and the location of the deployment of the system the traffic of the system will change. Consequently it is difficult to come up with a fixed threshold for the system and that results in high false positive alarm rate.

In [16], the authors describe an approach in which they monitor the header information in order to evaluate the statistical properties of various header parameters with respect to an n-dimensional normal distribution. The covariance of the parameters is compared with a threshold value to measure the anomalous behavior in case of an attack. They also perform payload scanning and analysis to detect attacks that are exploiting protocol vulnerabilities. As discussed previously this approach will be affected by the traffic context. Moreover the value of the threshold that is difficult to determine the optimal value.

In [9] the authors presented an approach in which the flow of DNS traffic between the source and the destination DNS server is used to detect attacks. As a part of their approach, the authors present different models that can detect attacks on the DNS protocol. For instance to detect cache poisoning attacks they use the flow based approach to separate the queries and requests in a flow and calculate if the count is below a threshold. To detect tunneling attacks using DNS they measure the traffic statistics at standard DNS ports which is indicative of tunneling activity. Also they present Cross-Entropy Anomaly detection model which they use to detect anomalies in DNS packet sizes. Where they use cross entropy to detect changes in the sizes of packets given the normal distribution of packets. They perform decision fusion of these different models to give one conclusive result of the alerts received from the model.

In [5] the authors present methods to monitor DNS traffic in large networks. They use the concept of standard flow and extended flow to detect DNS attacks on large networks. They present the use of access control lists in combination with the standard knowledge presented in a DNS flow to detect malware infected devices that operate rouge DNS resolvers. They also present a statistical approach to detect changes in networks DNS behavior patterns.

4 DNS IDS Approach

The proposed system is developed based on the assumption that DNS follows a stateful approach in which generally consecutive DNS queries and replies from a source are dependent on each other and the protocol moves through a well defined state machine to perform its request and reply messages. The DNS state machine can be built dynamically by observing and analyzing the DNS messages as in Figure 2.

The state transitions depend on the Q(TYP) and R(TYP) of the received or sent DNS packets. Generally the type (TYP) field has 70 different types. Thus due to these different transition values, each transitions can accept 70 different values. It is manually impossible to go over all these combinations. Consequently, we use the n-gram approach to model the DNS state transitions[4].

A ***dnsgram***[1, 12] is a data structure the captures important properties of consecutive DNS queries and replies. Thus a ***dnsgram*** of size 5 is used to characterize the properties of 5 consecutive DNS queries and replies from a source. The ***dnsgram*** collection acts like a sliding window over the DNS traffic. Hence when the system is operating normally, the transitions of protocol will be close to the transitions of the state machine shown in figure 2. But when the DNS protocol is attacked, it will no longer follow the protocol state machine and hence its ***dnsgrams*** will be significantly different from the normal ***dnsgrams***. To detect slow attacks and fast attacks, the state transitions of the protocol are observed over a configurable time window of *t* seconds. The group of ***dnsgrams*** that are observed in this time window of *t* seconds is

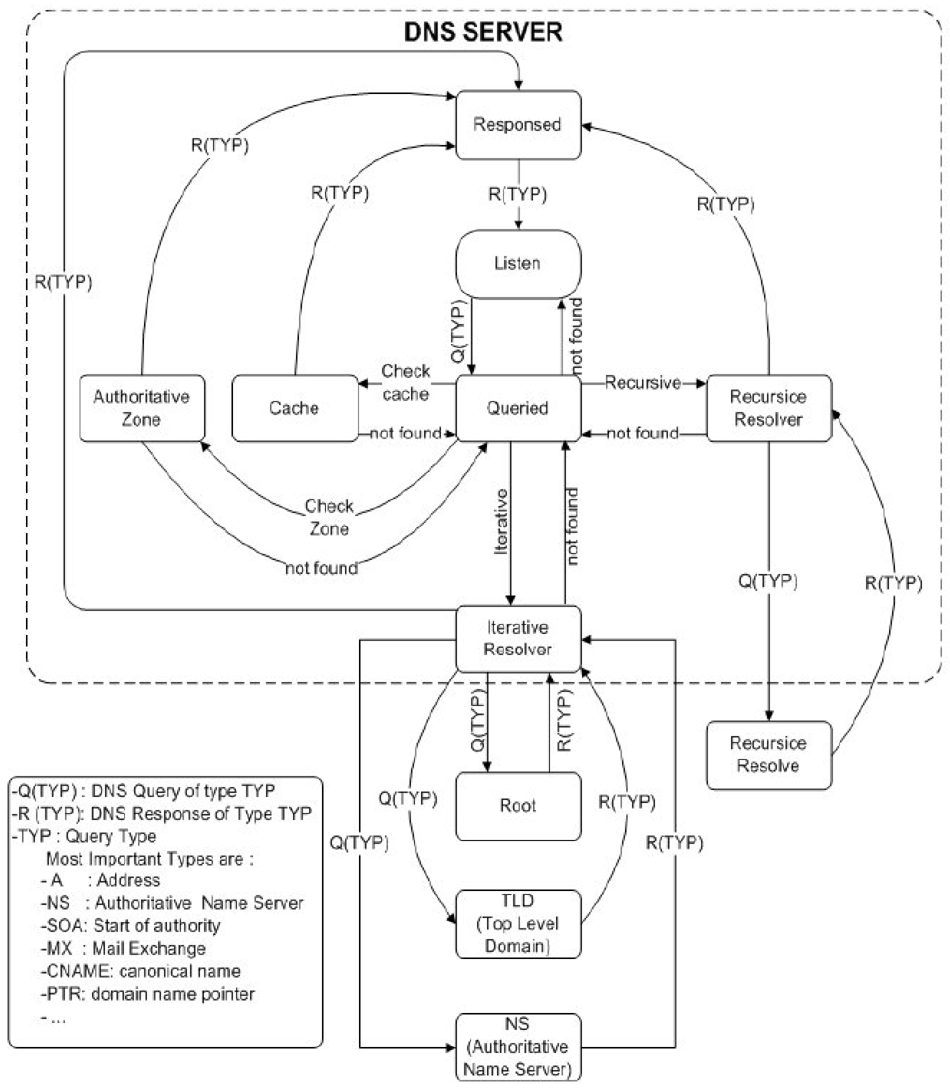

Figure 2: DNS protocol state diagram

called a ***dnsflow***[1, 12].

The architecture of the DNS-IDS system is shown in Figure 3.

During the training phase of the system, supervised training is used to observe the state transitions of the DNS protocol. The training phase operates in two stages.

4.1 Stage 1:

During this stage, the state transitions of the protocol are observed as ***dnsgrams***. The ***dnsgrams*** are periodically organized into ***dnsflows***. Then the statistical properties of the ***dnsgram*** with respect to the ***dnsflow*** are evaluated. One metric we compute is the number of occurrences of a particular ***dnsgram*** in a ***dnsflow***. The metric values are stored in a Counting Bloom Filter(CBF)[3]. This results in the generation of two Counting Bloom Filters at the end of the Stage 1. The first CBF is associated with the DNS

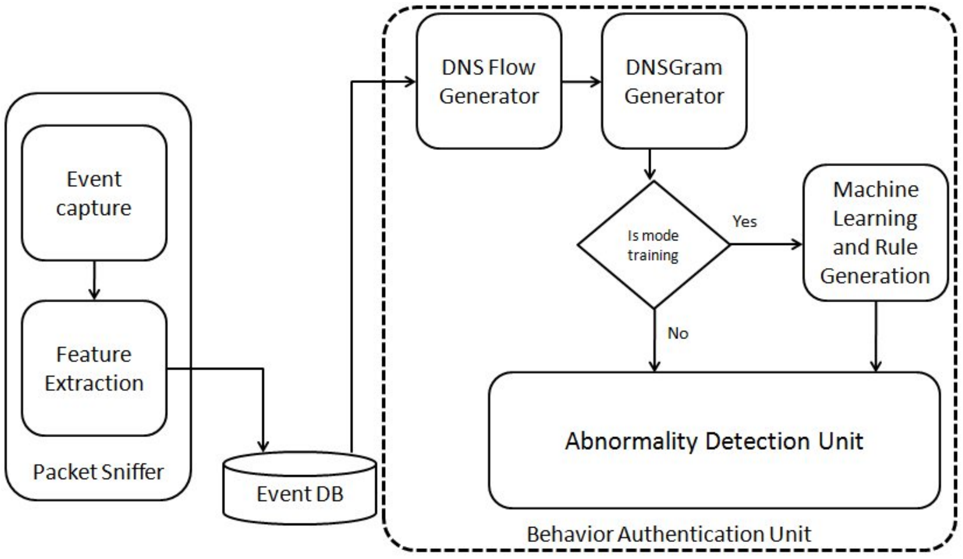

Figure 3: System Architecture

normal traffic and the second CBF is associated with the DNS traffic when it is under a known attack.

4.2 Stage 2:

During this stage of the training, the CBFs obtained in Stage 1 is used to obtain the statistical properties of the observed ***dnsflows***. The properties obtained during this stage are computed by the equations shown in Table2.

Table 2: Property extraction equations

$n(dnsgram_i)_{normal} = min(count(dnsgram_i), cbf_{normal}(dnsgram_i))$	(1)
$n(dnsgram_i)_{abnormal} = min(count(dnsgram_i), cbf_{abnormal}(dnsgram_i))$	(2)
$normalqueryindex(flow) = \sum n(dnsgram_i)_{normal}$	(3)
$abnormalqueryindex(flow) = \sum n(dnsgram_i)_{abnormal}$	(4)

The two query indexes obtained called the ***normalqueryindex*** and the ***abnormalqueryindex*** are stored as a 2-tuple data structure that will be used by the Bagging classification algorithm [11]. The use of the bagging algorithm produces a condition tree that is encoded as if statements that will be used to classify the normality and the abnormality of the flow.

In the operational mode, the network is monitored at run time for the DNS traffic. The collected DNS traffic is grouped into ***dnsgrams***. This collection of ***dnsgrams*** is then grouped into ***dnsflows***. Statistical properties of these ***dnsflows*** are extracted in timely manner so that the ***normalqueryindex*** and ***abnormalqueryindex*** is obtained for a particular flow. These indexes are then compared with the values that are obtained for normal DNS operations. This comparison then results in the classification of the ***dnsflows*** as normal or abnormal at run time. Our experimental results to be discussed later show that DNS attacks can be detected with low false positive and negative alarm rates.

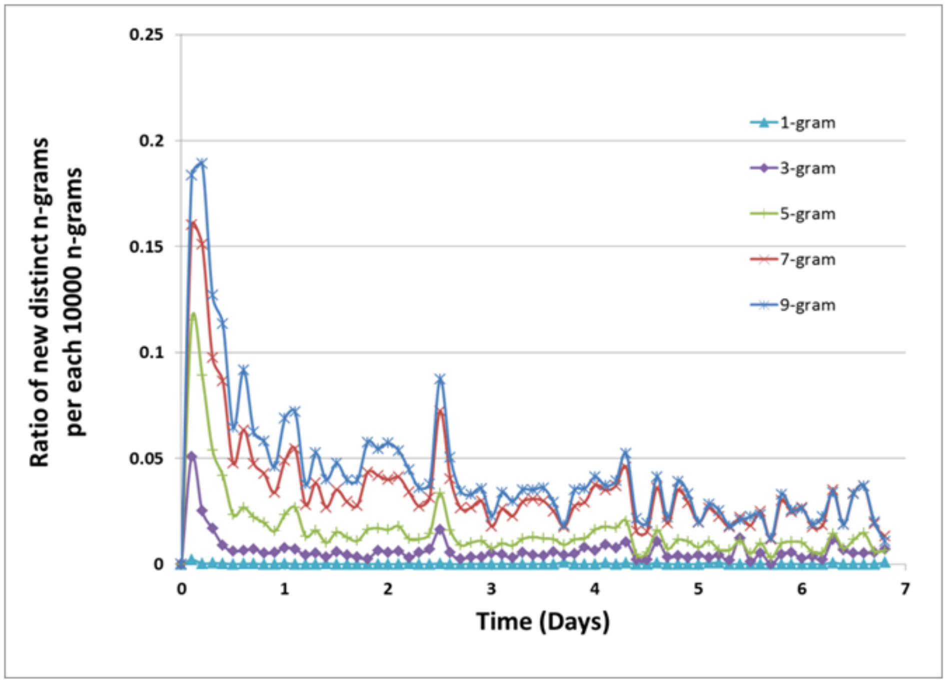

Figure 4: Ratio of new distinct dnsgrams during training phase

5 Experimental Results

5.1 Training Phase :

For a fixed window size of 10 seconds, the system was trained on the DNS traffic of the ACL's networks. This traffic, on average, consisted of normal internet traffic of the ACL's users which consist of 16 desktop computers, a number of mobile devices connected to the ACL's wireless network, and the traffic of the ACL cloud servers (at any time, at least 20 virtual machines are connected to the network). During the training phase, the size of the ***dnsgram*** was varied from values of 1 to 10 to determine the optimum size of the ***dnsgram*** that provides the best results for the selected time window, i.e., 10 seconds. The time window of 10 seconds was chosen because it allowed the detection of both fast attacks as well as slow attacks. As shown in Figure 4, it was observed that for larger ***dnsgram*** sizes the number of new ***dnsgrams*** observed decreases to a near zero after a period of 4 days.

We used Bagging classification algorithm to obtain the rules for the conditions for normal and abnormal ***dnsgram*** transitions.

5.2 Operational phase :

During this phase, the rules obtained from the training phase are used to detect anomalous DNS behavior that might have been triggered by cyber attacks. The DNS-IDS's performance was evaluated by performing various experiments for a ***dnsgram*** size of 5.

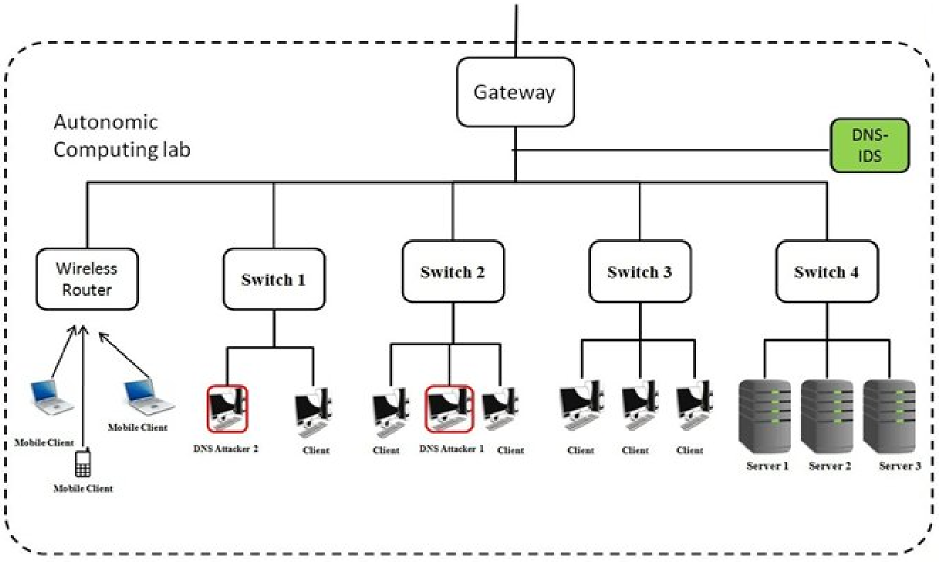

Figure 5: Test Bed

5.2.1 Experiment 1: False Positive Calculation

For this experiment, the system was setup on the test bed as shown in Figure 5. The system ran for two weeks where we observed the DNS traffic and tagged the traffic as normal or abnormal. During these two weeks, the system did not encounter any known attacks over the experiment period, where a total number of 82242 ***dnsgrams*** of which 1149 ***dnsgrams*** were observed to be abnormal. It was observed that the system gave a false positive rate of 0.01397%.

5.2.2 Experiment 2- Birthday Attack

As a part of this experiment, a Birthday attack[13] was launched in the network. In this attack, the attacker sends a large number of queries to the local DNS server followed by a large number of replies to these DNS queries. This attack takes advantage of the birthday paradox to successfully execute the attack. The intensity of the birthday attack was varied from 30 queries to 1000 queries. The attack was successfully detected for all these cases.

5.2.3 Experiment 3- DNS Amplification Attack

In this experiment, a DNS amplification attack[14] was launched in the network. The DNS amplification is a DDoS attack in which the objective of the attacker is to flood the system being attacked with replies from various DNS servers, which results into a denial of service attack. The DNS amplification attack intensity varied from 10 queries to 10000 queries. The attack was successfully detected in each case.

5.2.4 Experiment 4- DNS Hijacking

Under this experiment, a DNS hijacking attack[13] was launched in the network. In this attack, the attacker listens onto the local network and whenever it sees the local DNS server send out a query, it immediately replies to that query with a predetermined answer, resulting in the domain being hijacked. Thus when the authentic reply arrives, it is ignored by the local DNS server. This attack was performed on the network a number of times and has had a 100% detection rate for this attack.

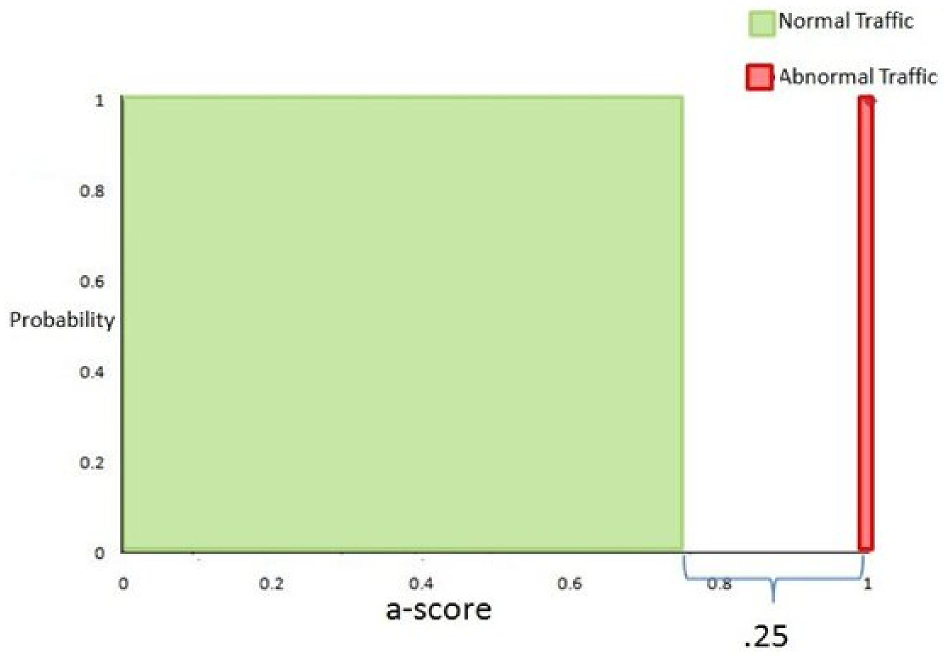

Figure 6: Normal Traffic vs Abnormal Traffic score

5.2.5 Experiment 5- DNS Tunneling

As a part of this experiment, a DNS tunneling attack[15] was executed in the network. It was observed that the attack was detected in each case.

Figure 6 shows a graph that maps the a-score value to the probability of the traffic being normal or abnormal. The large gap between the a-score of the observed attack traffic and that of the normal traffic indicates that our approach can detect attacks accurately with low positive false rates.

5.2.6 Experiment 6- Random Subdomain Attack

In this experiment, a Random Subdomain Attack was executed on the network, where a local DNS server was flooded with requests for random subdomains. This results in a denial of service attack. This attack was successfully detected in each case.

6 Conclusion

In this paper we presented a new anomaly behavior analysis method to detect attacks against the DNS protocol. Unlike previous works which have only focused on cache poisoning attacks our approach is general and can be applied to detect any type of DNS attacks (known or unknown). In our approach the normal DNS traffic is modeled as n consecutive transitions of ***dnsgram*** patterns for a dnsflow that covers a time window t. The statistical properties of the ***dnsflow*** are then evaluated and compared with those obtained during the training phase. Our experimental results show that a ***dnsgram*** of size 5 can be used to detect accurately known DNS attacks such as DNS cache poisoning, DNS amplification and DNS tunneling with less than 0.01397% false positive and a detection rate of 97%.

Acknowledgement

This work is partly supported by the Air Force Office of Scientific Research (AFOSR) Dynamic Data-Driven Application Systems (DDDAS) award number FA95550-12-1-0241, National Science Foundation research projects NSF IIP-0758579, NCS-0855087 and IIP-1127873, and Thomson Reuters in the framework of the Partner University Fund (PUF) project (PUF is a program of the French Embassy in the United States and the FACE)

References

[1] H. Alipour, Y. Al-Nashif, P. Satam, and S. Hariri. Wireless anomaly detection based on ieee 802.11 behavior analysis. *IEEE Transactions on Information Forensics and Security*, 10(10):2158–2170, 2015.

[2] G. Ateniese and S. Mangard. A new approach to dns security (dnssec). In *Proc. of the 8th ACM Conference on Computer and Communications Security (CCS'01), Philadelphia, Pennsylvania, USA*, pages 86–95. ACM, 2001.

[3] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[4] M. Bramer, M. Bramer, and M. Bramer. *Principles of data mining*, volume 131. Springer, 2007.

[5] M. Čermák, P. Čeleda, and J. Vykopal. Detection of dns traffic anomalies in large networks. In *Revised Selected Papers of the 20th EUNICE/IFIP EG 6.2, 6.6 International Workshop on Advances in Communication Networking, Rennes, France, LNCS*, volume 8846, pages 215–226. Springer, December 2014.

[6] S. Friedl. An illustrated guide to the kaminsky dns vulnerability. Unixwiz. net Tech Tips, August, August 2008. `http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html` [Online; Accessed in November 2015].

[7] A. Herzberg and H. Shulman. Retrofitting security into network protocols: The case of dnssec. *IEEE Internet Computing*, 18(1):66–71, 2014.

[8] G. Kambourakis, T. Moschos, D. Geneiatakis, and S. Gritzalis. A fair solution to dns amplification attacks. In *Proc. of the 2nd International Workshop on Digital Forensics and Incident Analysis (WDFIA'07), Samos, Greece*, pages 38–47. IEEE, August 2007.

[9] A. Karasaridis, K. Meier-Hellstern, and D. Hoeflin. Nis04-2: Detection of dns anomalies using flow data analysis. In *Proc. of the 2006 IEEE Global Telecommunications Conference (GLOBECOM'06), San Francisco, CA, USA*, pages 1–6. IEEE, November-December 2006.

[10] W. Lian, E. Rescorla, H. Shacham, and S. Savage. Measuring the practical impact of dnssec deployment. In *Proc. of the 22nd USENIX conference on Security (SEC'13), Washington, D.C., USA*, pages 573–588. USENIX Security, August 2013.

[11] T. U. of Waikato Team. Ensemble learning. `http://www.cs.waikato.ac.nz/ml/weka/mooc/dataminingwithweka/transcripts/Transcript4-6.txt` [Online; Accessed in November 2015].

[12] P. Satam, H. Alipour, Y. Al-Nashif, and S. Hariri. Dns-ids: Securing dns in the cloud era. In *Proc. of the 2015 International Conference on Cloud and Autonomic Computing (ICCAC'15), Boston, MA, USA*, pages 296–301. IEEE, September 2015.

[13] S. Son and V. Shmatikov. The hitchhiker's guide to dns cache poisoning. In *Proc. of the 6th Iternational ICST Conference on Security and Privacy in Communication Networks (SecureComm'10), Singapore*, volume 50 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 466–483. Springer Berlin Heidelberg, September 2010.

[14] United States Computer Emergency Readiness Team. Dns amplification attacks. Alert (TA13-088A), July 2013. https://www.us-cert.gov/ncas/alerts/TA13-088A.

[15] T. van Leijenhorst, K.-W. Chin, and D. Lowe. On the viability and performance of dns tunneling. In *Proc. of the 11th International Conference on Information Technology and Applications (ICITA'08), Cairns, Queensland, Australia*, pages 560–566, June 2008.

[16] Y. Wang, M.-z. Hu, B. Li, and B.-r. Yan. Tracking anomalous behaviors of name servers by mining dns traffic. In *Proc. of the ISPA 2006 Workshops, Sorrento, Italy, LNCS*, volume 4331, pages 351–357. Springer Berlin Heidelberg, December 2006.

[17] A. Yamada, Y. Miyake, M. Terabe, K. Hashimoto, and N. Kato. Anomaly detection for dns servers using frequent host selection. In *Proc. of the 23rd International Conference on on Advanced Information Networking and Applications (AINA'09). Bradford, UK*, pages 853–860. IEEE, May 2009.

[18] B. Zdrnja, N. Brownlee, and D. Wessels. Passive monitoring of dns anomalies. In *Proc. of the 4th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA'07), Lucerne, Switzerland, LNCS*, volume 4579, pages 129–139. Springer Berlin Heidelberg, July 2007.

Author Biography

Pratik Satam is a M.S. student at the Department of Electrical and Computer Engineering at University of Arizona since August 2013. He received his B.E in Electronics and Telecommunication engineering from University of Mumbai in July 2013. His research interests include Cyber security, computer networks, secure critical infrastructure, Autonomic computing and Data analysis.

Hamid Alipour received the B.S. degree in computer engineering from University of Isfahan, Iran, in 2005 and the M.S. degree in computer engineering from Shahid Beheshti University, Tehran, Iran, in 2009. He received his Ph.D. degree in computer engineering from Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ, in 2013. He worked as a research assistant with the Autonomic Computing Laboratory and the NSF center for Cloud and Autonomic Computing in University of Arizona from 2010 to 2013. He is currently with the Cloud Identity Service and Security Division, Microsoft, Redmond, WA. His research interests include Cybersecurity, Cloud computing and Cloud security, Cyber trust, Biometrics and Multi-factor authentication, Behavior Analysis, Data mining, Big data analysis, Machine learning, and Interne of things.

Youssif Al-Nashif is an Assistant Professor in the Department of Electrical and Computer Engineering at the Old Dominion University since July 2014. He received his Ph.D. in Computer Engineering from The University of Arizona in 2008. He received his B.Sc. Electrical Engineering and M.Sc. Communication and Electronics Engineering form Jordan University of Science and Technology in 1999 and 2000 respectively. He worked as an Assistant Research Professor at the University of Arizona from 2009 to 2014. He is the first director for the Old Dominion Center for Cybersecurity Education and Research. He was also the director of the Secure and Trusted Technologies (STT) lab. He is now with the Department of Computer Engineering, Florida Polytechnic University, Lakeland, Fl-33805. His research interests include Cybersecurity, Cyber Resilience, Secure Critical Infrastructures, Cyber Trust, Cloud Security, Autonomic Computing, Data Analysis, Behavior Modeling, and Power and Performance Management.

Salim Hariri is a Professor in the Department of Electrical and Computer Engineering at The University of Arizona. He received his Ph.D. in Computer Engineering from University of Southern California in 1986, and an MSc from The Ohio State University in 1982. He is the UA site director of NSF Center for Cloud and Autonomic Computing and he is the Editor-In-Chief for the CLUSTER COMPUTING JOURNAL (Springer, http://clus.edmgr.com) that presents research techniques and results in the area of high speed networks, parallel and distributed computing, software tools, and network-centric applications. He is the Founder of the IEEE/ACM International Symposium on High Performance Distributed Computing (HPDC) and the co-founder of the IEEE/ACM International Conference on Cloud and Autonomic Computing. He is co-author/editor of four books on Autonomic computing, parallel and distributed computing: Autonomic Computing,: Concepts, Infrastructure, and Applications (CRC Press, 2007), Tools and Environments for Parallel and Distributed Computing (Wiley, 2004), Virtual Computing: Concept, Design and Evaluation (Kluwer, 2001), and Active Middleware Services (Kluwer, 2000). His research interests include autonomic cyber security, big data analytics, resilient cloud services, critical infrastructure protections, and autonomic programming, and resilient Dynamic Data Driven Application Systems (rDDDAS).