Contents lists available at ScienceDirect

Franklin Open

journal homepage: www.elsevier.com/locate/fraope

DNS Intrusion Detection (DID) — A SNORT-based solution to detect DNS Amplification and DNS Tunneling attacks

Sanjay Adiwal [a,*], Balaji Rajendran [a], Pushparaj Shetty D. [b], Sithu D. Sudarsan [a]

[a] Centre for Development of Advanced Computing (C-DAC), Bangalore, Karnataka, India
[b] Department of Mathematical and Computational Sciences, National Institute of Technology Karnataka Surathkal, India

ARTICLE INFO

Keywords:
IDS
DNS intrusion
Amplification
Tunneling
DNS vulnerabilities
DoS

ABSTRACT

Domain Name System (DNS) plays a critical role in the Internet ecosystem, translating numerical IP addresses to memorable domain names and vice versa. The malicious user targets DNS by taking advantage of vulnerabilities in DNS. The most complex attacks in the DNS attacks vector include Distributed Denial of Service (DDoS) based DNS amplification attacks and sophisticated DNS tunneling attacks. An Intrusion Detection System (IDS) is a solution available to monitor the traffic for intrusion in the network but not exclusively for DNS intrusions. In this research paper, we present – DNS Intrusion Detection (DID), a system integrated into SNORT – a prominent open-source IDS, to detect major DNS-related attacks. We developed novel IDS signatures for various tools used in the tunneling, amplification, and DoS attacks and added them to the existing ruleset file of IDS to detect DNS-based intrusions. Our approach successfully identifies empirical DNS attacks carried out by various known tools available over the Internet. Evaluation of DID showed a high detection rate and a very low false-positive rate.

1. Introduction

With more than 4.66 billion users, the Internet is a vital part of everyday life and a crucial means of communicating in the modern world, and now it has become a key component for driving business. As a result, cybercriminals utilize the Internet to their financial advantage by targeting organizations with malware, data theft, and other cyber-attacks. Adopting cyber security measures is, therefore, necessary to protect networks, software, and hardware from various cyberattacks. Multiple layers of security, including firewall systems, IDS/IPS, and unified threat management systems are implemented among various elements of network infrastructures to prevent cyber attacks. Among these listed security solutions, IDS is the most economical and adopted widely by Internet users.

The ever-growing Internet has led the Internet Standards organization's DNS Extension Working Group (DNSEXT) to define DNS as the "Critical Infrastructure" with its tremendous usage. The DNS infrastructure is used by almost all distributed applications, including web services, email services, FTP, remote login, etc., to resolve addresses.

Due to the importance of the DNS application, it has been abused by malevolent actors interested in doing harm and gaining a personal advantage. As a result, DNS is vulnerable to a variety of security risks, including DNS spoofing, cache poisoning, DNS hijacking, NX-DOMAIN/NSNX, phantom domain attacks, botnet-based DDoS, DNS floods, tunneling, and amplification attacks [1]. DNS attacks can bring a web/application service down resulting the Internet inaccessible to a large number of users or even divert users to illegitimate or duplicate websites for the extraction of personal and sensitive financial information. DNS attacks can make DNS servers participate in command and control (C2) communication and data exfiltration. Among various DNS attacks, the following should be addressed immediately for a resilient DNS infrastructure: DoS/DDoS attacks through BOTNET, DNS reflection & amplification, and DNS tunneling.

Over the years, many security solutions and DNS security add-ons have evolved, and the DNS protocol also has been revised many times, furthermore many extensions to DNS (e.g. DNSSEC and TSIG) are introduced to address some DNS attacks. DNSSEC is a suite of protocols developed by the IETF community to respond to some of the threats to the DNS infrastructure. DNSSEC assures the authenticity of the origin of the response and provides integrity for the data received from DNS servers [2]. DNSSEC uses asymmetric cryptography to prevent DNS spoofing by ensuring that all responses received are digitally signed

Abbreviations: IDS, Intrusion Detection System; IPS, Intrusion Prevention System; ISP, Internet Service Provider; FTP, File Transfer Protocol; NXDOMAIN, Non-existent Domain; DNSSEC, Domain Name System Security Extensions; DS, Delegation Signer; NS-2, Network Simulator Version 2; OSINT, Open Source Intelligence; C&C, Command and Control

* Corresponding author.

E-mail addresses: sanjayadiwal@cdac.in (S. Adiwal), balaji@cdac.in (B. Rajendran), prajshetty@nitk.edu.in (Pushparaj Shetty D.), sds@cdac.in (S.D. Sudarsan).

https://doi.org/10.1016/j.fraope.2023.100010
Received 15 July 2022; Received in revised form 21 December 2022; Accepted 27 January 2023
Available online 6 February 2023

by the DNS zone administrators and can easily be verified using the public key. Therefore DNSSEC protects from content corruption attacks like cache poisoning and assures the origin of data. While DNSSEC does not resolve flood-based attacks, it can reinforce the DNS against a form of DoS attack. DNSSEC suffers from implementation complexity owing to key management issues — as there is a Zone Signing Key (ZSK) and a Key Signing Key (KSK), and the latter is seeded at the immediate top-level domain as DS records [3]. DNSSEC also leads to overheads in response size. The whole DNS resolution can go down if a key mismatch happens, requiring cautious and correct implementation of DNSSEC.

Transaction Signatures (TSIG) ensures secure communication between primary and secondary DNS servers using symmetric keys and cryptographic hash functions [4]. It also ensures that the data received in zone transfer is authentic and not modified during transit and provides for the authenticity of the DNS response. TSIG is a mechanism used to address IP spoofing during a Resource Record (RR) updating operation between master and slave servers. It is used to provide a means of authenticating updates to a DNS database. TSIG only protects from "spoofing master” attack, and provides a secure and authentic zone transfer between DNS servers.

A DNS firewall is another way to ensure a secure DNS infrastructure. A DNS firewall is a security appliance that filters bad DNS traffic and protects against various DNS attacks by blocking certain DNS queries to known bad domains. Most DNS firewall has the feature of malware protection, and Response Policy Zones (RPZ) functionality [5]. It can detect most DNS attacks based on attack signatures and provide defense by dropping packets if signatures match. It can monitor live DNS traffic and block specific domains.

A significant amount of research has been conducted, and many security solutions and protocols have been evolved & developed for securing DNS infrastructure. However, none of them thwarted all the DNS-related attacks, and still, many security issues remain unresolved related to DNS. This research paper fills this gap by proposing a technique embedded into a popular open-source IDS SNORT to addresses all possible DNS-related attacks. In this work, we propose a snort-based intrusion detection system designed particularly for detecting DNS protocol anomalies, we also created the attack signatures for DNS tunneling, DNS amplification, and DoS attacks.

2. Literature survey

Anna Drozdova’s thesis work [6] focused on developing and testing a system to safeguard DNS servers in a lab environment by installing an IDS (SNORT) in proximity to the DNS server and implementing some snort rules to block malicious website lookups. The test lab was set up in a LAN environment with three systems: a victim system, an attacker system, and a SNORT IDS system. The SNORT system was being investigated for DNS security, although DNS server protection is neither complete nor comprehensive. However, this work aims to demonstrate the behavior of the DNS server protection system in a real network and investigate the capabilities of the SNORT system and its configuration settings for DNS server security.

Hock and Kortis [7] came up with an idea of a firewall system for DNS protection, which deals with DNS security mechanisms applicable to transport and network layers. The proposed protection technique is based on traffic shaping, flow filtering, and prioritization. The authors mentioned the instructions for the creation of rules for traffic shaping and prioritizing. They demonstrated how a firewall might alter traffic dynamically to ensure that packets arrive while the DNS server is under attack.

An anomaly-based IDS was presented by Pratick Satam et al. [8,9], which operates in two different phases, i.e., training and operational. In their research work, the normal behavior of the DNS protocol was modeled as a finite state machine in the training phase, and temporal statistics were stored in a database. Abnormal DNS traffic transitions were noted in a separate database. Based on these two statistics, an anomaly metric was calculated and used in the operational phase to detect various DNS attacks. The evaluation of their work was done by conducting a wide range of known DNS attacks. However, they did not consider DoS or DDoS attacks on DNS during the evaluation process.

Steven Cheung [10] presented an explicit intrusion detection method for safeguarding DNS service by using formal specifications, formal modeling, and proofs to boost the assurance of the IDS for DNS. The DNS wrappers were used for monitoring malicious DNS traffic. DNS traffic was investigated against their specifications, and deviations were noted as DNS anomalies. The authors also mentioned that if the monitored traffic diverges from the authoritative answer, this event was flagged as a possible attack.

Rastegari et al. [11] proposed a machine learning-based IDS for DNS DoS attacks, which seeks to use the learning power of neural networks to identify DNS DoS attacks. The proposed IDS was evaluated on three different neural networks, i.e., backpropagation, radial basis function, and self-organizing maps. The entire experiment was simulated on the NS-2 simulator. The findings show that a BP neural network outperforms other types, with a solution accuracy of 99 percent for attack detection and a low false alarm rate.

The author of [12] used various approaches, including visualization, machine learning techniques, and statistical analysis, to provide a reliable and robust network-based solution as a protection system against DNS-based threats, particularly DNS tunneling attacks. The suggested approach functions as a DNS server, detecting and preventing DNS tunneling attacks from various attack tools.

The authors of [13] presented botnet-based DNS tunneling detection by examining various strategies for identifying C&C systems and employing signature matching methodology to detect DNS-tunneled SSH handshakes between bots and C&C systems. The authors used the Domain Generation Algorithm, Fast Flux, and DNS tunneling to offer a multi-staged approach for detecting botnet communications. The Bro network security monitoring solution was used and rules were created based on network anomalies and signatures to detect such behavior.

The authors of [14] designed a DNS firewall based on the open-source DNS response policy zones technology. The proposed system was designed to consider the following requirements: domain blocking, user notification, event logging, and domain blacklist sharing. However, the study has shown that the DNS firewall can indeed be used to block access to malicious domains, which can be used in DNS tunneling attacks for C2 and data exfiltration, but it cannot detect or protect against most of the DNS attacks.

In [15], the authors suggest a DoH traffic analysis-based machine learning approach for identifying malicious DNS tunnel tools. The suggested solution focuses only on DoH traffic analysis and is based on hierarchical machine learning categorization. The assessment findings show that their suggested method can identify all six malicious DNS tunnel tools (dns2tcp, dnscat2, iodine, dnstt, TCP-over-DNS, and tuns) with a classification rate of 98.02%. However, the solution only detects tunneling attacks.

To identify DNS queries and filter them depending on their likelihood of being malicious or not, Claudio Marques et al. [16] suggested a firewall solution based on machine learning. OSINT sources and analytical techniques were used to augment the DNS dataset. The resultant dataset was then presented to several supervised ML algorithms to accurately and rapidly determine whether a domain request is malicious or not. Exploratory analysis and data preparation processes have been completed. The suggested approach does not, however, identify DNS system intrusion.

Many studies have been conducted to secure DNS from tunneling, amplification, and DNS denial-of-service attacks. However, no study has been done to ensure that DNS is protected from all types of possible attacks. Furthermore, the traditional security solutions such as firewalls and IDS/IPS are not purpose-built DNS security systems and do not have an in-depth understanding of the DNS protocol and therefore are not adapted to protect DNS infrastructure, including root, top-level

Table 1
List of DNS tunneling attack tools.

Tool Name	Programming Language used	Resource Record Used	Encoding Technique Used	Platform Supported	Availability of Encryption
Iodine	C	NULL	Base32/64	Linux, Unix, MacOS X and Windows	No
DNSCAT2	C and Ruby	CNAME MX TXT	HEX and NetBIOS	Linux, Unix, MacOS X and Windows	Yes
DNS2TCP	C	TXT KEY	BASE64	Linux	No
ThunderDNS	Python, Bash, Powershell	TXT	Base64	Windows Unix	No
OzmanDNS	Perl	TXT	Base32	Linux	No

domain's DNS servers, second-level domain's DNS servers, recursive resolvers, and even DNS clients. As a result, it becomes necessary to develop a DNS Intrusion Detection that protects from all possible DNS attacks by developing appropriate attack signatures against all possible attack surfaces.

3. IDS signatures for DNS intrusion detection

An intrusion into a computer system or computer network is carried out to compromise the system or network and violate the CIA security principles of confidentiality, integrity, and authentication. The confidentiality property ensures that only authenticated and authorized receivers can receive the intended information. The integrity property ensures that the data or the information is received as it is sent but does not guarantee that it has not been modified during transmission. Authentication is the process of identifying or proving that something is true, genuine, or valid against a claim [17]. Intrusion detection is the process of monitoring the events occurring in a computer system or network and analyzing them for signs of possible incidents of threats and violations of computer security practices, acceptable usage policies, or standard security policies. An Intrusion Detection System (IDS) is a hardware appliance or software application that monitors the events occurring in a computer system or network and alerts on events that have signs of possible incidents of violations of security policies. An IDS uses a predefined collection of signatures to detect network intrusions. Snort is an open-source and one of the most widely used IDS.

DNS is one of the most significant elements of the Internet and one of the most targeted applications, with various ways to misuse it, each with its own set of toolkits. Because each toolkit has its own set of specifications, it is difficult to create signatures that work across all toolkits for a given attack. Additionally, attackers can also deploy DNS attacks with previously unseen signatures to bypass IDS/IPS appliances. As a result, most IDS are unable to identify all DNS-based attacks.

In this research work, we used a popular IDS, i.e., SNORT, and created novel signatures to detect DNS amplification, tunneling, and DoS attacks. We used publicly available applications and tools for DNS exploitation and captured their patterns as signatures. We then added these novel signatures to the existing DNS rule file of SNORT to detect various DNS attacks. The resultant SNORT-based IDS solution is called DNS Intrusion Detection — DID.

This section of the paper presents various novel IDS signatures for detecting DNS tunneling, DNS amplification, and DoS attacks for specific attack tools used for performing a particular attacks. We have also used different utilities for performing DNS tunneling, amplification, and DoS attacks.

3.1. IDS signature for DNS tunneling

Unlike web traffic, DNS traffic is generally unmonitored in most organizations, enabling malicious users to exploit all the vulnerabilities of the DNS protocol. DNS tunneling, also known as DNS exfiltration, is the malicious use of DNS infrastructure used for command and control (C2C or C2) and data exfiltration. An attacker creates a DNS tunnel from his system to the targeted system, bypassing the firewall system on the targeted system's network. The tunnel created by the attacker can then be used for various malicious purposes, like data exfiltration, C2C, and even tunneling IP traffic. The DNS protocol is not intended for a command channel or general-purpose tunneling. Therefore, it needs to be noted that upon setup of a successful DNS tunnel, the total IP traffic can be passed through the DNS tunnel, leading to control by a remote system, data exfiltration, and the routing of any Internet traffic through the DNS tunnel.

DNS tunneling is a method to encapsulate any form of binary data within DNS queries and responses and tunnel it to a remote system via the Internet. Currently, various applications are available on the Internet for tunneling over DNS; the most common intent of these applications is to enable free internet browsing over WiFi networks. Nevertheless, such tools can exploit the DNS infrastructure, such as C2C, providing complete remote access through the DNS tunnel. Further, DNS tunneling can communicate with malware such as the Feederbot and Moto.

The DNS Protocol is defined in RFCs 1034 and 1035, but around 265 other RFCs are related to the DNS protocol [18]. DNS has over 65 resource records (RR) [19], and the most commonly used RRs are A, AAAA, MX, PTR, NS, and CNAME. However, the attackers will use other RRs like TXT to perform DNS tunneling attacks. Many tools are available over the Internet for enabling DNS tunneling and have different characteristics, as shown in Table 1. The common characteristics of all the listed tools of this DNS abuse are: encoded and encrypted queries and responses rather than plain text; a high volume of lengthy queries and responses, and different resource record distributions [20].

As shown in Fig. 1, two main components are required for a DNS tunneling attack: a controlled, authoritative DNS server and a malware-infected client. The attacker uses data encoded in the DNS payload to create a tunnel. In DNS tunneling, the IPv4 network packets are tunneled through DNS by using the hostname to send data using a DNS query and a record type, e.g., NULL, TXT, or any other record type for transporting the response, meaning it will have DNS queries in a format like <encoded data>.abc.com for the upstream and the DNS response for the downstream. There are many utilities available today on the Internet for DNS tunneling, with their features and bugs. The following are the most popular and effective tools that are easily available: Iodine, DNSCAT2, DNS2TCP, ThunderDNS, and OzmanDNS. We have created the IDS signatures for all these tools.

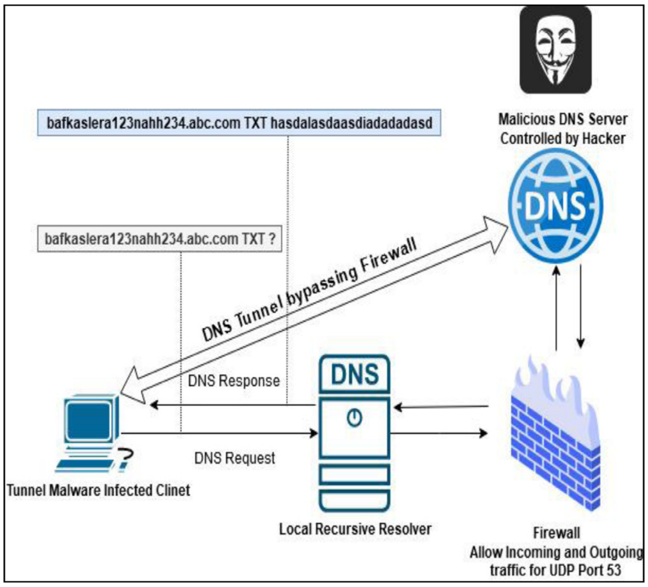

bafkaslera123nahh234.abc.com TXT hasdalasdaasdiadadadasd
Malicious DNS Server Controlled by Hacker
DNS
bafkaslera123nahh234.abc.com TXT ?
DNS Tunnel bypassing Firewall
DNS Response
DNS
DNS Request
Tunnel Malware Infected Clinet
Local Recursive Resolver
Firewall
Allow Incoming and Outgoing traffic for UDP Port 53

Fig. 1. DNS tunneling attack.

3.1.1. *IDS signature for Iodine*

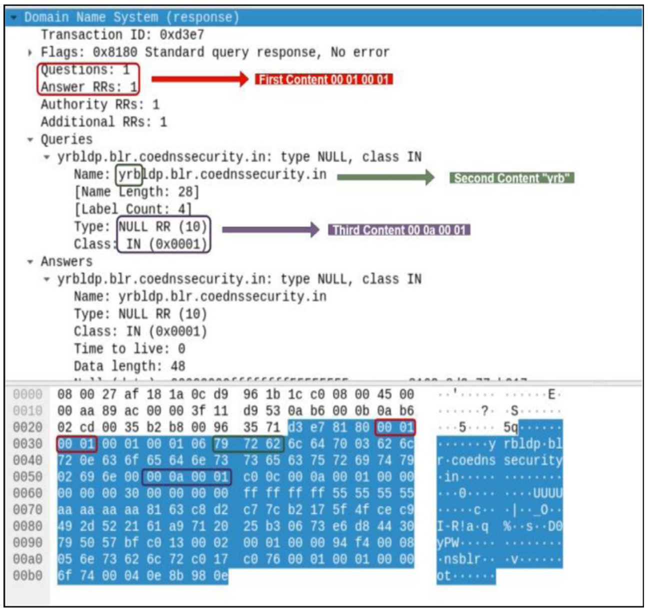
Iodine [21] was written by Erik Ekman & Bjorn Andersson and can be used to tunnel the entire IPv4 traffic using the DNS protocol. This tool uses a TUN or TAP interface on the endpoint, enabling a network interface on the connected client, which connects the server and client as both share the same network. This makes Iodine a unique tool that enables endpoints to access any protocol encapsulated by an IP header. Iodine tunnel always uses the NULL resource record type because the NULL RR type is only used for testing and is now obsolete. Thus, in DNS query/response, if NULL has been identified, it could be tunnel traffic. So the following signature is created to determine the Iodine tunnel on the recursive resolver:

```
Alert udp any 53 -> any any (msg:" iodine DNS Tunneling
Request"; content:"|00 01 00 01|", offset 4, depth 4;
content:"yrb", distance 4, within 4; content:"|00 0a 00
01|", within 255; threshold: type threshold, track by_src,
count 2, seconds 15; sid: 806254370; rev:1;)
```

This signature has three content matches. The hex bytes “00 01 00 01” correspond to the question section and answer RR section in the header of the response from the DNS server for the initial client Iodine tunnel request. As an initial client request, the domain is prefixed with “yrb” in the DNS query. The reply also contains the same prefix, which we will check in the next content match. The last content, hex bytes “00 0a 00 01”, represents a NULL type request and IN class. We observe that this pattern should match twice in fifteen seconds to generate an alert for Iodine tunnel traffic.

The client tries to auto-detect the DNS query type, and if it gets NOTIMP as a reply, which means the server does not support this kind of request, then CNAME is used as the default query type, and the domain is prefixed with "yrb”. The default snort signature will not be able to detect this. The Fig. 2 shows a typical network packet captured for Iodine tunnel traffic. Based on the characteristics and traffic captured of Iodine application we have developed the following IDS novel signature to recognize Iodine tunnel:

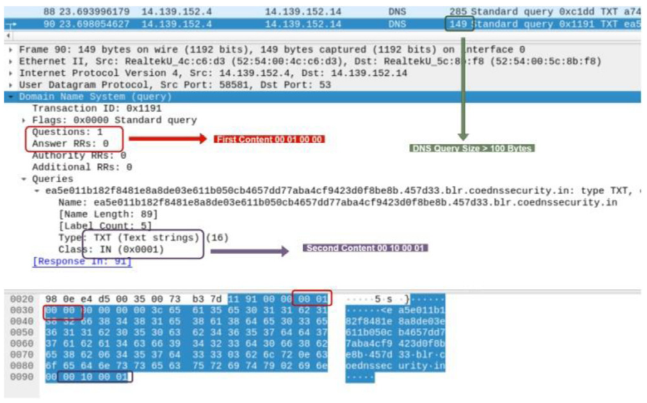
```
alert udp any 53 -> any any (msg:" iodine DNS Tunneling
Request"; content:"|00 01 00 01|", offset 4, depth 4;
content:"yrb", distance 4, within 4; content:"|00 01 00
01|", within 255; threshold: type threshold, track by_src,
count 2, seconds 15; sid: 806254371; rev:1;)
```



```
▾ Domain Name System (response)
    Transaction ID: 0xd3e7
  ▸ Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 1
    Additional RRs: 1
  ▾ Queries
    ▾ yrbldp.blr.coednssecurity.in: type NULL, class IN
        Name: yrbldp.blr.coednssecurity.in
        [Name Length: 28]
        [Label Count: 4]
        Type: NULL RR (10)
        Class: IN (0x0001)
  ▾ Answers
    ▾ yrbldp.blr.coednssecurity.in: type NULL, class IN
        Name: yrbldp.blr.coednssecurity.in
        Type: NULL RR (10)
        Class: IN (0x0001)
        Time to live: 0
        Data length: 48

0000  08 00 27 af 18 1a 0c d9  96 1b 1c c0 08 00 45 00   ··'·····  ······E·
0010  00 aa 89 ac 00 00 3f 11  d9 53 0a b6 00 0b 0a b6   ······?·  ·S······
0020  02 cd 00 35 b2 b8 00 96  35 71 d3 e7 81 80 00 01   ···5····  5q······
0030  00 01 00 01 00 01 06 79  72 62 6c 64 70 03 62 6c   ·······y  rbldp·bl
0040  72 0e 63 6f 65 64 6e 73  73 65 63 75 72 69 74 79   r·coedns  security
0050  02 69 6e 00 00 0a 00 01  c0 0c 00 0a 00 01 00 00   ·in·····  ········
0060  00 00 00 30 00 00 00 00  ff ff ff ff 55 55 55 55   ···0····  ····UUUU
0070  aa aa aa aa 81 63 c8 d2  c7 7c b2 17 5f 4f ce c9   ·····c··  ·|··_O··
0080  49 2d 52 21 61 a9 71 20  25 b3 06 73 e6 d8 44 30   I-R!a·q   %··s··D0
0090  79 50 57 bf c0 13 00 02  00 01 00 00 94 f4 00 08   yPW·····  ········
00a0  05 6e 73 62 6c 72 c0 17  c0 76 00 01 00 01 00 00   ·nsblr··  ·v······
00b0  6f 74 00 04 0e 8b 98 0e                            ot······
```

Fig. 2. A network packet capture for Iodine traffic.



```
88 23.693996179  14.139.152.4   14.139.152.14   DNS   285 Standard query 0xc1dd TXT a74
90 23.698054627  14.139.152.4   14.139.152.14   DNS   149 Standard query 0x1191 TXT ea5

▸ Frame 90: 149 bytes on wire (1192 bits), 149 bytes captured (1192 bits) on interface 0
▸ Ethernet II, Src: RealtekU_4c:c6:d3 (52:54:00:4c:c6:d3), Dst: RealtekU_5c:8b:f8 (52:54:00:5c:8b:f8)
▸ Internet Protocol Version 4, Src: 14.139.152.4, Dst: 14.139.152.14
▸ User Datagram Protocol, Src Port: 58581, Dst Port: 53
▾ Domain Name System (query)
    Transaction ID: 0x1191
  ▸ Flags: 0x0000 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ▾ Queries
    ▾ ea5e011b182f8481e8a8de03e611b050cb4657dd77aba4cf9423d0f8be8b.457d33.blr.coednssecurity.in: type TXT,
        Name: ea5e011b182f8481e8a8de03e611b050cb4657dd77aba4cf9423d0f8be8b.457d33.blr.coednssecurity.in
        [Name Length: 89]
        [Label Count: 5]
        Type: TXT (Text strings) (16)
        Class: IN (0x0001)
    [Response In: 91]

0020  98 0e e4 d5 00 35 00 73  b3 7d 11 91 00 00 00 01   ·····5·s  ·}······
0030  00 00 00 00 00 00 3c 65  61 35 65 30 31 31 62 31   ······<e  a5e011b1
0040  38 32 66 38 34 38 31 65  38 61 38 64 65 30 33 65   82f8481e  8a8de03e
0050  36 31 31 62 30 35 30 63  62 34 36 35 37 64 64 37   611b050c  b4657dd7
0060  37 61 62 61 34 63 66 39  34 32 33 64 30 66 38 62   7aba4cf9  423d0f8b
0070  65 38 62 06 34 35 37 64  33 33 03 62 6c 72 0e 63   e8b·457d  33·blr·c
0080  6f 65 64 6e 73 73 65 63  75 72 69 74 79 02 69 6e   oednssec  urity·in
0090  00 00 10 00 01                                     ·····
```

First Content 00 01 00 00
DNS Query Size > 100 Bytes
Second Content 00 10 00 01

Fig. 3. A network packet capture for DNSCAT2 traffic.

With the aforementioned signature, our custom-built IDS monitored full-load peak traffic, and we observed no false positives. This proves that this is a specific signature for a DNS tunneling attack through Iodine.

3.1.2. *IDS signature for DNSCAT2*

Ronn Bowes developed DNSCAT2 [22] in 2010, supporting almost all major OS, including Mac OS-X. The main purpose of this tool is to create a tunnel for a C2C channel between a compromised client and a controlled authoritative DNS server, which means that it is capable of making a raw connection over DNS. The DNS tunnel by DNSCAT2 uses hex encoding as well as NetBIOS encoding. The main highlights of DNSCAT2 are that it encrypts the tunnel by default and that it can work without having an authoritative public server for a domain, which is required for almost all DNS tunnel tools to function. By default, it uses CNAME, MX, and TXT records, allowing an attacker to work in stealth mode.

DNSCAT2 traffic can be easily monitored if it uses an unencrypted channel, as all queries and responses have “dnscat” prefixed to them. The following default SNORT rule can be used to identify such traffic:

```
alert udp any any -> any 53 (msg: “DNSCAT2-Tunneling Attempt”; content:"|00 01 00 00|",offset 4,depth 4; content: “dnscat”, distance 28, within 255; sid: 806254374; rev:1)
```

But it will be very hard to detect if it uses encrypted communications between the client and server. However, as every packet has an unencrypted 5-byte header and an unencrypted 16-bit nonce, and the DNS query/reply packets are always greater than 100 bytes. A example network packet for DNSCAT2 tunnel communication is shown in Fig. 3. We have created the novel signatures below to identify DNSCAT2 tunnel based on the features and traffic gathered by DNSCAT2 application:

```
alert udp any any -> any 53 (msg: "Possible-DNSCat-Tunnel-CNAME-Traffic"; content:"|00 01 00 00|",offset 4,depth 4; content: "|00 05 00 01|",distance 6, within 255; dsize:>100;detection_filter: track by_src, count 2, seconds 10;sid:806254375;rev:1;)
```

```
alert udp any any -> any 53 (msg: "Possible-DNSCat-Tunnel-MX-Traffic"; content:"|00 01 00 00|",offset 4,depth 4; content: "|00 0f 00 01|",distance 6, within 255; dsize:>100;detection_filter: track by_src, count 2, seconds 10;sid:806254376;rev:1;)
```

```
alert udp any any -> any 53 (msg: "Possible-DNSCat-Tunnel-TXT-Traffic"; content:"|00 01 00 00|",offset 4,depth 4; content: "|00 10 00 01|",distance 6, within 255; dsize:>100;detection_filter: track by_src, count 2, seconds 10;sid:806254377;rev:1;)
```

3.1.3. *IDS signature for DNS2TCP*

DNS2TCP [23], written by Olivier Dembour and Nicolas Collignon, is among the data exfiltration applications that enable POP, SMTP, SSH, and other TCP connections over the DNS protocol. It is written in the C programming language and is intended to relay TCP connections as DNS queries to form a DNS tunnel without encryption. The main highlight of DNS2TCP is that it does not require admin privileges to run this program. It supports KEY and TXT resource records types in DNS queries and responses. To establish TCP handshakes, it uses the fixed subdomains, i.e., =auth, =connect. It uses a session tag for tracking and maintaining the same session and uses the first 4 bytes of the subdomain. The behavior for creating a signature to detect a connection for DNS2TCP tunneling is as follows:

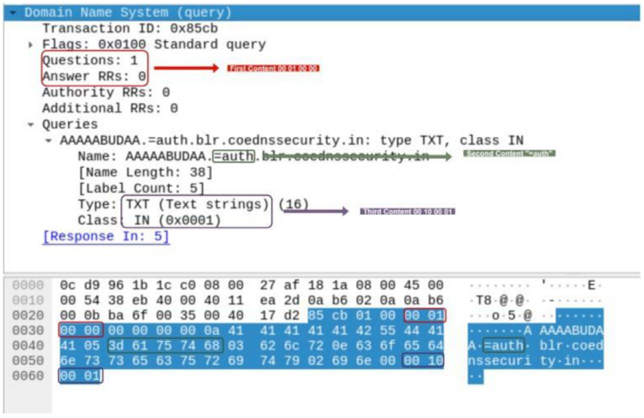
1. DNS header question and answer field - content “00 01 00 00” follows immediately after query identification and Flags
2. TCP handshake string “=auth”
3. TXT record in IN class, content “00100001”.

Fig. 4 shows a typical network packet captured for DNSCAT2 tunnel traffic. Based on the behavior, we created an IDS signature to detect the connection handshake for DNS2TCP as:

```
alert udp any any -> any 53 (msg: "DNS2TCP-TCP Handshake"; content:"|00 01 00 00|", offset 4, depth 4; content:"=auth", distance 4; content:"|00 10 00 01|", distance 2, within 255; sid:806254378; rev:1;)
```

3.1.4. *IDS signature for ThunderDNS*

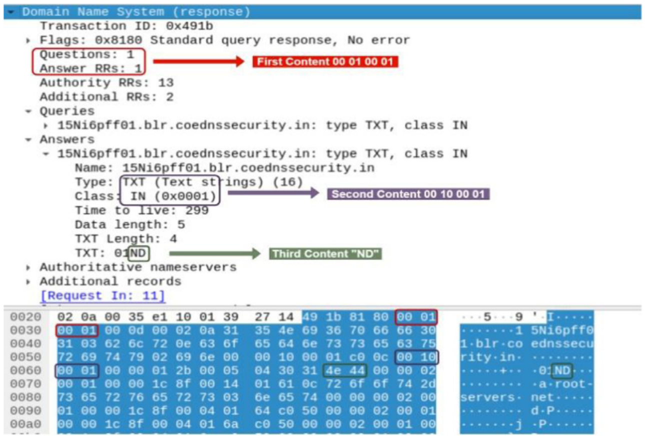
A ThunderDNS [24] tunnel can be used to create a tunnel through which TCP traffic is forwarded over the DNS protocol. The program for the tunnel server is written in Python and is even available for Docker. The client program is available as a Linux shell, a Windows power shell, and a PHP web program. Most of the IDS do not have an inbuilt signature for the ThunderDNS tunnel. It uses a fairly simple protocol for communication between the server and the client. The characteristics of communication between client and server are as follows:



```
▾ Domain Name System (query)
    Transaction ID: 0x85cb
  ▸ Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ▾ Queries
    ▾ AAAAABUDAA.=auth.blr.coednssecurity.in: type TXT, class IN
        Name: AAAAABUDAA.=auth.blr.coednssecurity.in
        [Name Length: 38]
        [Label Count: 5]
        Type: TXT (Text strings) (16)
        Class: IN (0x0001)
    [Response In: 5]

0000  0c d9 96 1b 1c c0 08 00  27 af 18 1a 08 00 45 00   ········ '·····E·
0010  00 54 38 eb 40 00 40 11  ea 2d 0a b6 02 0a 0a b6   ·T8·@·@· -·······
0020  00 0b ba 6f 00 35 00 40  17 d2 85 cb 01 00 00 01   ···o·5·@ ········
0030  00 00 00 00 00 00 0a 41  41 41 41 41 42 55 44 41   ·······A AAAABUDA
0040  41 05 3d 61 75 74 68 03  62 6c 72 0e 63 6f 65 64   A·=auth· blr·coed
0050  6e 73 73 65 63 75 72 69  74 79 02 69 6e 00 00 10   nssecuri ty·in···
0060  00 01                                              ··
```

Fig. 4. A network packet capture for DNS2TCP traffic.



```
▾ Domain Name System (response)
    Transaction ID: 0x491b
  ▸ Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 13
    Additional RRs: 2
  ▾ Queries
    ▸ 15Ni6pff01.blr.coednssecurity.in: type TXT, class IN
  ▾ Answers
    ▾ 15Ni6pff01.blr.coednssecurity.in: type TXT, class IN
        Name: 15Ni6pff01.blr.coednssecurity.in
        Type: TXT (Text strings) (16)
        Class: IN (0x0001)
        Time to live: 299
        Data length: 5
        TXT Length: 4
        TXT: 01ND
  ▸ Authoritative nameservers
  ▸ Additional records
    [Request In: 11]

0020  02 0a 00 35 e1 10 01 39  27 14 49 1b 81 80 00 01   ···5···9 '·I·····
0030  00 01 00 0d 00 02 0a 31  35 4e 69 36 70 66 66 30   ·······1 5Ni6pff0
0040  31 03 62 6c 72 0e 63 6f  65 64 6e 73 73 65 63 75   1·blr·co ednssecu
0050  72 69 74 79 02 69 6e 00  00 10 00 01 c0 0c 00 10   rity·in· ········
0060  00 01 00 00 01 2b 00 05  04 30 31 4e 44 00 00 02   ·····+·· ·01ND···
0070  00 01 00 00 1c 8f 00 14  01 61 0c 72 6f 6f 74 2d   ········ ·a·root-
0080  73 65 72 76 65 72 73 03  6e 65 74 00 00 00 02 00   servers· net·····
0090  01 00 00 1c 8f 00 04 01  64 c0 50 00 00 02 00 01   ········ d·P·····
00a0  00 00 1c 8f 00 04 01 6a  c0 50 00 00 02 00 01 00   ·······j ·P······
```

Fig. 5. A network packet capture for ThunderDNS traffic.

The client registers the tunnel with the server by requesting a TXT record as per the following format:

0<random string with 7 characters><client ID>.<domain name>.

Once the initial negotiation is completed, the client pools data from the server as follows:

1<random string with 7 characters><client ID>.<domain name>.

If new data is available for the client, then the server replies with TXT records with the data; else, it will return as <id>ND.

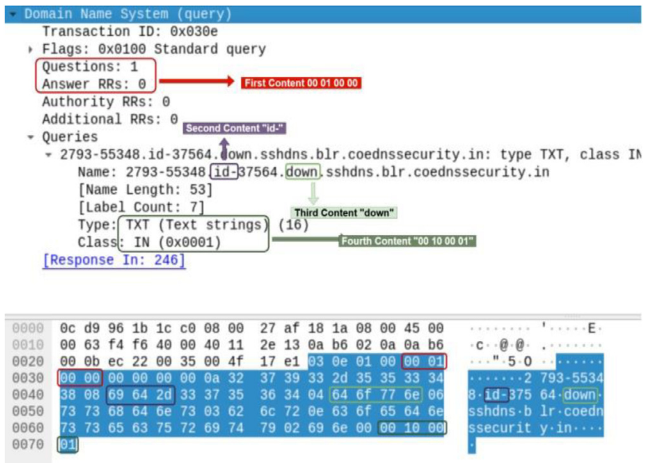
Therefore, the following signature captures this replay and confirms the ThunderDNS tunnel traffic:

```
alert udp any 53 -> any any (msg:"ThunderDNS Tunnel Traffic"; content:"|00 01 00 01|", offset 4, depth 4; content:"|00 10 00 01|", distance 4; content:"ND", distance 4, within 255; threshold: type threshold, track by_src, count 15, seconds 2;sid:806254380;rev:1;)
```

We captured network traffic for the ThunderDNS attack as shown in Fig. 5. Here the first content corresponds to the standard DNS response from the server to the client, while the second content represents a TXT reply with internet class (IN), and the last content matches “ND”, which reveals the ThunderDNS tunnel traffic.

3.1.5. *IDS signature for OzymanDNS*

The OzymanDNS [25] tool was written by Dan Kaminsky in 2004 to create a DNS tunnel through SSH. It is written in Perl with MIME::Base32 and Net::DNS modules. The tool uses TXT records with base32-encoded requests and base64-encoded responses. The Perl script uses



```
▾ Domain Name System (query)
    Transaction ID: 0x030e
  ▸ Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ▾ Queries
    ▾ 2793-55348.id-37564.down.sshdns.blr.coednssecurity.in: type TXT, class IN
        Name: 2793-55348.id-37564.down.sshdns.blr.coednssecurity.in
        [Name Length: 53]
        [Label Count: 7]
        Type: TXT (Text strings) (16)
        Class: IN (0x0001)
    [Response In: 246]

0000  0c d9 96 1b 1c c0 08 00  27 af 18 1a 08 00 45 00   ········ '·····E·
0010  00 63 f4 f6 40 00 40 11  2e 13 0a b6 02 0a 0a b6   ·c··@·@· .·······
0020  00 0b ec 22 00 35 00 4f  17 e1 03 0e 01 00 00 01   ···"·5·O ········
0030  00 00 00 00 00 00 0a 32  37 39 33 2d 35 35 33 34   ·······2 793-5534
0040  38 08 69 64 2d 33 37 35  36 34 04 64 6f 77 6e 06   8·id-375 64·down·
0050  73 73 68 64 6e 73 03 62  6c 72 0e 63 6f 65 64 6e   sshdns·b lr·coedn
0060  73 73 65 63 75 72 69 74  79 02 69 6e 00 00 10 00   ssecurit y·in····
0070  01                                                  ·
```

Fig. 6. A network packet capture for OzymanDNS traffic.

either one of the keywords "up" or "down" to indicate whether the traffic is upstream or downstream.

We captured the OzymanDNS traffic as shown in Fig. 6. The signatures to detect the OzymanDNS tunnel are as follows:

```
alert udp any any -> any 53 (msg:"OzymanDNS Client
Down Request"; content:"|00 01 00 00|",offset 4,depth 4;
content:"id-",distance 4; content:"down"; within:10;
content:"|00 10 00 01|"; distance:5; within:255;
threshold: type threshold, track by_src, count 20, seconds
5;sid: 806254381;rev:0;)
```

The above signature generates an alert message when a client sends a DNS query wherein the DNS header question is set to one and the answer RR is zero (content: "| 00 01 00 00|"), and the query contains "id-" and "down" keywords. The client always asks for a text record in internet class (content: "| 00 10 00 01|").

```
alert udp any any -> any 53 (msg:"OzymanDNS Client up
Request"; content:"|00 01 00 00|";offset:4;depth:4;
content:"-0"; distance:16; content:"id-
";distance:1;within:3; content:"up"; within:8; content:"|00
01 00 01|"; distance:5; within:255; threshold: type
threshold, track by_src, count 20, seconds 5;sid:
806254382;rev:0;)
```

The above signature will result in an alert when the client sends a DNS query that contains "-0", "id-" and "up" keywords asking for an A record (content: "| 00 01 00 01|").

3.2. IDS signature for Reflection/Amplification attack

In a DNS amplification attack, an attacker uses globally accessible public recursive resolvers to flood a victim's system with huge DNS response traffic that eventually leads to a DDoS attack on the victim. Here, the victim's IP address is spoofed in DNS queries sent to public recursive resolvers by the attacker. The recursive resolvers then send the DNS responses back to the target system, as shown in Fig. 7. The attacker uses "any" in the DNS query to amplify the attack vector and send as much zone information as possible. The open DNS servers will then return all known information about a DNS zone in a response, amplifying the attack. The attackers can send large DNS packets using the EDNS(0) extension to amplify the DNS attack further. The attacker can also use the DNSSEC protocol, which would increase the DNS packet size. Using these amplification techniques, 37 bytes of DNS query packet can be configured to elicit a response of over 3256 bytes to the victim's system, resulting in an 88x amplification factor. If a botnet is used to produce huge spoofed queries, the attacker can generate immense traffic with very little effort. The victim's system looks at DNS responses as valid DNS traffic from legitimate DNS servers. This makes these attacks extremely difficult to prevent.

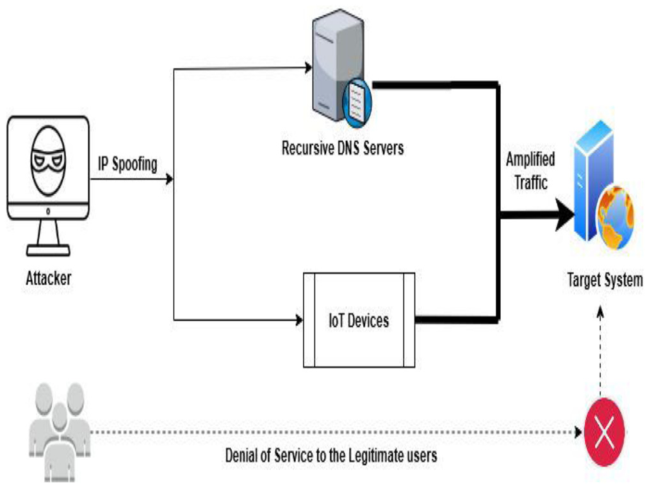
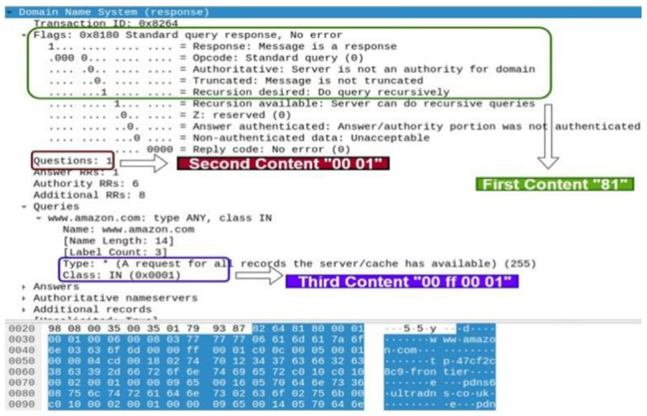


Fig. 7. DNS reflected amplification attack.



```
▾ Domain Name System (response)
    Transaction ID: 0x8264
  ▾ Flags: 0x8180 Standard query response, No error
      1... .... .... .... = Response: Message is a response
      .000 0... .... .... = Opcode: Standard query (0)
      .... .0.. .... .... = Authoritative: Server is not an authority for domain
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... 1... .... = Recursion available: Server can do recursive queries
      .... .... .0.. .... = Z: reserved (0)
      .... .... ..0. .... = Answer authenticated: Answer/authority portion was not authenticated
      .... .... ...0 .... = Non-authenticated data: Unacceptable
      .... .... .... 0000 = Reply code: No error (0)
    Questions: 1
    Answer RRs: 1
    Authority RRs: 6
    Additional RRs: 8
  ▾ Queries
    ▾ www.amazon.com: type ANY, class IN
        Name: www.amazon.com
        [Name Length: 14]
        [Label Count: 3]
        Type: * (A request for all records the server/cache has available) (255)
        Class: IN (0x0001)
  ▸ Answers
  ▸ Authoritative nameservers
  ▸ Additional records

0020  98 08 00 35 00 35 01 79  93 87 82 64 81 80 00 01   ···5·5·y ···d····
0030  00 01 00 06 00 08 03 77  77 77 06 61 6d 61 7a 6f   ·······w ww·amazo
0040  6e 03 63 6f 6d 00 00 ff  00 01 c0 0c 00 05 00 01   n·com··· ········
0050  00 00 04 cd 00 18 02 74  70 12 34 37 63 66 32 63   ·······t p·47cf2c
0060  38 63 39 2d 66 72 6f 6e  74 69 65 72 c0 10 c0 10   8c9-fron tier····
0070  00 02 00 01 00 00 09 65  00 16 05 70 64 6e 73 36   ·······e ···pdns6
0080  08 75 6c 74 72 61 64 6e  73 02 63 6f 02 75 6b 00   ·ultradn s·co·uk·
0090  c0 10 00 02 00 01 00 00  09 65 00 14 05 70 64 6e   ········ ·e···pdn
```

Fig. 8. Packet capture for EthanWillnor traffic.

Various applications/tools are available on the Internet for DNS amplification attack, and some popular ones are listed in Table 2. We have created an IDS signature for these listed tools. Most amplification attack tools require a list of open recursive servers, the target system's IP address, and a domain or list of domains for querying all records of those domains using the ANY option in the DNS query.

3.2.1. IDS signature for Ethanwilloner DNS amplification

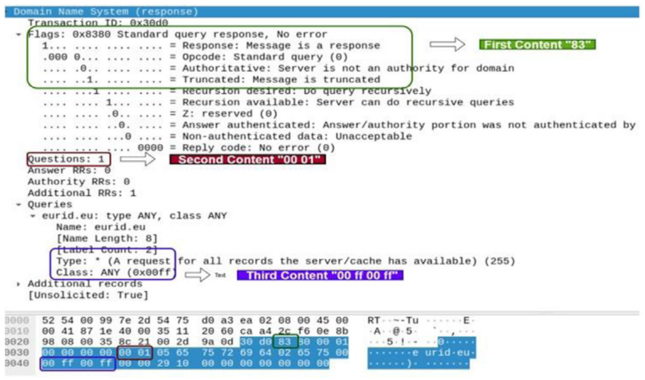
Ethan Willoner [26] developed a tool that can provide a proof of concept for DNS amplification attacks. The tool is written in the "C" programming language, which sends UDP DNS queries to a list of open DNS servers mentioned in the program with the target's IP address spoofed as the source. Since the code needs to access raw sockets, it requires root privileges to run the program.

We captured the attack traffic as shown in Fig. 8 and analyzed the traffic to create an IDS signature that identifies this attack pattern. The main parameters to detect this attack on the victim are:

(a) Standard Query Response — The first content field is for standard query response. The content "81" corresponds to a response with recursion desired, or "83" corresponds to a response with truncated and recursion desired.

Table 2
List of DNS amplification tools.

Tool Name	Query Type	Program	Class	EDNS Support
Ethanwilloner	ANY	C	IN	No
Saddam	ANY	Python	ANY	No
DNSREFLECT-DNS Flooder v1.1	ANY	C	IN	Yes
DNSDRDOS	A	C	IN	No



```
Domain Name System (response)
    Transaction ID: 0x30d0
  - Flags: 0x8380 Standard query response, No error
      1... .... .... .... = Response: Message is a response
      .000 0... .... .... = Opcode: Standard query (0)
      .... .0.. .... .... = Authoritative: Server is not an authority for domain
      .... ..1. .... .... = Truncated: Message is truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... 1... .... = Recursion available: Server can do recursive queries
      .... .... .0.. .... = Z: reserved (0)
      .... .... ..0. .... = Answer authenticated: Answer/authority portion was not authenticated by
      .... .... ...0 .... = Non-authenticated data: Unacceptable
      .... .... .... 0000 = Reply code: No error (0)
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 1
  - Queries
    - eurid.eu: type ANY, class ANY
        Name: eurid.eu
        [Name Length: 8]
        [Label Count: 2]
        Type: * (A request for all records the server/cache has available) (255)
        Class: ANY (0x00ff)
  ▸ Additional records
    [Unsolicited: True]

0000  52 54 00 99 7e 2d 54 75  d0 a3 ea 02 08 00 45 00   RT··~-Tu ······E·
0010  00 41 87 1e 40 00 35 11  20 60 ca a4 2c f6 0e 8b   ·A··@·5·  `··,···
0020  98 08 00 35 8c 21 00 2d  9a 0d 30 d0 83 80 00 01   ···5·!·- ··0·····
0030  00 00 00 00 00 01 05 65  75 72 69 64 02 65 75 00   ·······e urid·eu·
0040  00 ff 00 ff 00 00 29 10  00 00 00 00 00 00 00      ······)· ·······
```

Fig. 9. Packet capture for offensive-python saddam tooltik traffic.

(b) Question Section — The question section corresponds to the content in the DNS header: "| 00 01|".
(c) Query Type — This tool uses "ANY" in the query type with the Internet class "IN", which is represented by content: "| 00 ff 00 01|".
(d) DNS packet size: We generate the alert if the DNS packet size is greater than 100
(e) Threshold value — We notice that around 10,000 packets of the aforementioned pattern arrive at the client side every ten seconds while capturing the EthanWillnor amplification attack traffic, so we generate alerts if 10,000 packets match the signature in ten seconds.

Based on the above parameters we have created the following IDS signatures to detect this attack:

```
alert udp any 53 -> any any (msg:"EthanWillonr-DNS Amplification Attack"; content:"|81|",offset 2,depth 2; content:"|00 01|", distance 1, within 2;content:"|00 ff 00 01|",distance 6; dsize:>100; threshold: type threshold, track by_src, count 10000, seconds 10; sid:806254383;rev:1;)
```

```
alert udp any 53 -> any any (msg:"EthanWillonr-DNS Amplification Attack"; content:"|83|",offset 2,depth 2; content:"|00 01|", distance 1, within 2;content:"|00 ff 00 01|",distance 6; dsize:>100; threshold: type threshold, track by_src, count 10000, seconds 10; sid:806254384;rev:1;)
```

3.2.2. IDS signature for OffensivePython saddam amplification

It is a Python program that does various amplification attacks, including DNS amplification. It supports benchmarking, which reveals the amplification factor for a given list of open recursive resolvers. Apart from DNS amplification, it also supports NTP, SNMP, and SSDP amplification [27]. As shown in Fig. 9, We captured the attack pattern of this tool and created the following signatures that generate an alert if this tool is used for an amplification attack:

```
alert udp any 53 -> any any (msg:"Saddam-DNS Amplification Attack"; content:"|81|", offset 2, depth 2; content:"|00 01|", distance 1, within 2;content:"|00 ff 00 ff|", distance 8; threshold: type threshold, track by_src, count 10000, seconds 10; sid:806254385;rev:1;)
```

```
alert udp any 53 -> any any (msg:"Saddam-DNS Amplification Attack"; content:"|83|", offset 2, depth 2; content:"|00 01|", distance 1, within 2;content:"|00 ff 00 ff|", distance 8; threshold: type threshold, track by_src, count 10000, seconds 10; sid:806254386;rev:1;)
```

The parameters are the same as for the Ethanwilloner tool except for the query type. Here, ANY query type will be used with ANY Internet class which is represented by content: "| 00 ff 00 ff|".

3.2.3. IDS signature for DNSREFLECT-DNS Flooder v1.1

The DNSREFLECT-DNS Flooder-1.1 [28] can amplify the DNS request by a factor of 50 or more. The tool is written in the "C" programming language, allowing attackers to customize their DNS resource records. The DNS flooder uses ANY request with a spoofed IP address to perform a reflected amplification attack. It also enables the feature to send a DNS query, which ensures the largest possible response from the recursive resolver by enabling EDNS(0) in the DNS request, which allows the recursive resolver to send a huge response to the target without truncation or retransmissions. We captured DNS Flooder-1.1 attack traffic as shown in Fig. 10, and identified the following signature to detect DNS flooder traffic:

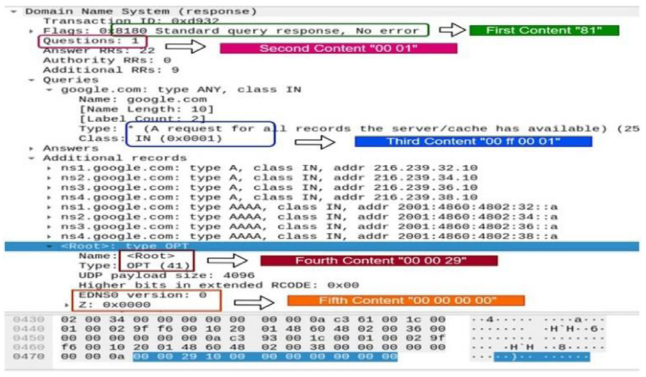
```
alert udp any 53 -> any any (msg:"DDoS Attack attempt using DNS flooder 1.1";content:"|81|",offset 2,depth 2; content:"|00 01|",distance 1, within 2; content:"|00 ff 00 01|",distance 8; content: "|00 00 29|",distance 50;content:"|00 00 00|",distance 2, within 4;dsize:>200;threshold: type threshold, track by_src, count 10000, seconds 10; sid: 806254387;rev:1;)
```

```
alert udp any 53 -> any any (msg:"DDoS Attack attempt using DNS flooder 1.1";content:"|83|",offset 2,depth 2; content:"|00 01|",distance 1, within 2; content:"|00 ff 00 01|",distance 8; content: "|00 00 29|",distance 50;content:"|00 00 00|",distance 2, within 4;dsize:>200;threshold: type threshold, track by_src, count 10000, seconds 10; sid: 806254388;rev:1;)
```

The content "| 00 ff 00 01|" is related to query type (ANY) and class (IN). Regarding EDNS(0), the additional section in the response from the server contains an OPT record with EDNS(0) version set to 0 and Z bits to be clear. The content "| 00 00 29|" will look for the OPT query type in the additional section of the response, and the content "| 00 00 00 00|" corresponds to Z bits for ENDS(0).

3.2.4. IDS signature for DNSDRDOS by noptirx

Noptrix [29] from "Nullsecurity" developed a "C" program for demonstrating DoS through DNS reflection and amplification. The program needs to be compiled before execution, and it requires a list of public recursive DNS servers stored in the text file, the IP address of the victim system that needs to be spoofed, the domain name used to send DNS queries, and the loop count. A malicious user can effectively launch a DDoS attack from many hosts, using many available name servers, and completely flood the victim with unwanted network traffic. By analyzing Wireshark traffic shown in Fig. 11, the ensuing signatures were created to detect DNSDRDOS traffic at the victim as:

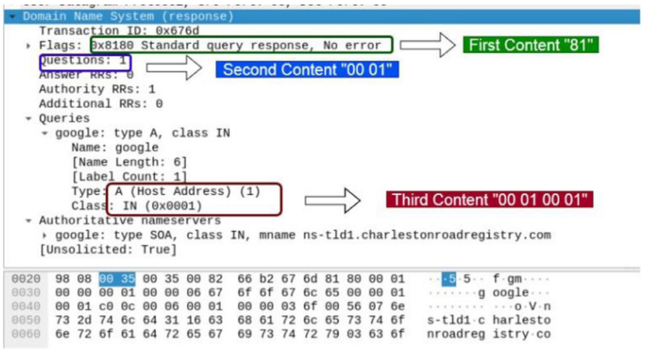


```
▾ Domain Name System (response)
    Transaction ID: 0xd932
  ▸ Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 22
    Authority RRs: 0
    Additional RRs: 9
  ▾ Queries
    ▾ google.com: type ANY, class IN
        Name: google.com
        [Name Length: 10]
        [Label Count: 2]
        Type: * (A request for all records the server/cache has available) (25
        Class: IN (0x0001)
  ▸ Answers
  ▾ Additional records
    ▸ ns1.google.com: type A, class IN, addr 216.239.32.10
    ▸ ns2.google.com: type A, class IN, addr 216.239.34.10
    ▸ ns3.google.com: type A, class IN, addr 216.239.36.10
    ▸ ns4.google.com: type A, class IN, addr 216.239.38.10
    ▸ ns1.google.com: type AAAA, class IN, addr 2001:4860:4802:32::a
    ▸ ns2.google.com: type AAAA, class IN, addr 2001:4860:4802:34::a
    ▸ ns3.google.com: type AAAA, class IN, addr 2001:4860:4802:36::a
    ▸ ns4.google.com: type AAAA, class IN, addr 2001:4860:4802:38::a
    ▾ <Root>: type OPT
        Name: <Root>
        Type: OPT (41)
        UDP payload size: 4096
        Higher bits in extended RCODE: 0x00
        EDNS0 version: 0
      ▸ Z: 0x0000

0430  02 00 34 00 00 00 00 00  00 00 0a c3 61 00 1c 00   ··4····· ····a···
0440  01 00 02 9f f6 00 10 20  01 48 60 48 02 00 36 00   ········ ·H`H··6·
0450  00 00 00 00 00 00 0a c3  93 00 1c 00 01 00 02 9f   ········ ········
0460  f6 00 10 20 01 48 60 48  02 00 38 00 00 00 00 00   ··· ·H`H ··8·····
0470  00 00 0a 00 00 29 10 00  00 00 00 00 00 00         ···  ··)·· ······
```

First Content "81"
Second Content "00 01"
Third Content "00 ff 00 01"
Fourth Content "00 00 29"
Fifth Content "00 00 00 00"

Fig. 10. Packet capture for DNS Flooder-v1.1 traffic.



```
▾ Domain Name System (response)
    Transaction ID: 0x676d
  ▸ Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 0
    Authority RRs: 1
    Additional RRs: 0
  ▾ Queries
    ▾ google: type A, class IN
        Name: google
        [Name Length: 6]
        [Label Count: 1]
        Type: A (Host Address) (1)
        Class: IN (0x0001)
  ▾ Authoritative nameservers
    ▸ google: type SOA, class IN, mname ns-tld1.charlestonroadregistry.com
    [Unsolicited: True]

0020  98 08 00 35 00 35 00 82  66 b2 67 6d 81 80 00 01   ···5·5··  f·gm····
0030  00 00 00 01 00 00 06 67  6f 6f 67 6c 65 00 00 01   ·······g oogle···
0040  00 01 c0 0c 00 06 00 01  00 00 03 6f 00 56 07 6e   ········ ···o·V·n
0050  73 2d 74 6c 64 31 16 63  68 61 72 6c 65 73 74 6f   s-tld1·c harlesto
0060  6e 72 6f 61 64 72 65 67  69 73 74 72 79 03 63 6f   nroadreg istry·co
```

Fig. 11. Packet capture for DNSDRDOS traffic.

```
alert udp any 53 -> any any (msg:"DDoS Attack attempt using DNDDRDOS.C Program"; content:"|81|",offset 2,depth 1; content:"|00 01|", distance 1, within 2;content:"|00 01 00 01|",distance 6; dsize:>100; threshold: type threshold, track by_src, count 10000, seconds 10; sid:806254388;rev:1;)
```

```
alert udp any 53 -> any any (msg:"DDoS Attack attempt using DNDDRDOS.C Program"; content:"|83|",offset 2,depth 1; content:"|00 01|", distance 1, within 2;content:"|00 01 00 01|",distance 6; dsize:>100; threshold: type threshold, track by_src, count 10000, seconds 10; sid:806254388;rev:1;)
```

This attack's characteristic is a general reflection attack. Here the DNS query type is an A record in the internet class which corresponds to the content: "| 00 01 00 01|".

3.3. IDS signature for DoS attack on DNS servers

A DoS on DNS servers is also called a DNS flood, which leverages UDP-a networking protocol that is session-less, where a malicious user floods an authoritative DNS server or a recursive resolver with DNS queries in an attempt to disturb the DNS resolution and can ultimately lead to inaccessibility of the services and applications through domain names. DNS flood attacks should be differentiated from DNS amplification attacks, as the central idea behind DNS amplification attacks is to exhaust the bandwidth resources and congest the network. The DNS flood attack aims to harm the server itself and make the service unavailable. A DNS flood is a type of DoS attack in which one or more DNS servers belonging to a given zone are attacked by the attacker, attempting to obstruct the resolution of resource records for that zone and its sub-zones. Here, the attacker attempts to overbear a particular DNS server with seemingly legitimate traffic, overwhelming server resources and impeding the ability of the servers to direct legitimate requests to zone resources. The DNS flood is also treated as a DDoS attack when multiple controlled systems called BOT flood the victim's DNS server with DNS queries, making it inaccessible to legitimate users for DNS resolution, as shown in Fig. 12.

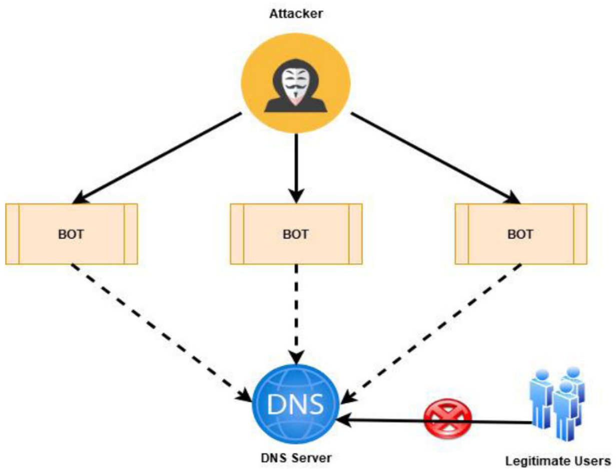


Fig. 12. DoS/DDoS attack on DNS server.

Various applications are available to craft standard DNS queries with spoofed IP addresses and send them to targeted DNS servers for DoS attacks. Google's DNS-flood [30] and Python SCAPY are among the most used DNS-flooding tools.

The signature for detecting a DoS attack is as given below:

```
alert udp any any -> any 53 (msg:"DNS Flooder-DoS Attack"; content:"|01 00 00 01|", offset 2, depth 4; content:"|00 01|", distance 10, within 255;threshold: type threshold, track by_src, count 10000, seconds 10; sid:806254390;rev:1;)
```

The above rule will generate an alert message "DNS Flooder-DoS Attack" when the server receives more than 10,000 DNS queries every 10 seconds from the same source.

4. Experiment

SNORT is one of the most popular and robust IDS, and its signatures detect potentially malicious traffic in the network, so we used SNORT for our experimental setup. As of this writing there are more than 50,000 signatures in SNORT IDS for detection of the various categories of attacks as follows: TCP and UDP-based DoS and DDoS attacks, Attacks on various applications like FTP, TELNET, SSH, and DNS, Attacks on DBMS, Web application attacks, and attacks on email-related protocols like POP, IMAP, and SMTP.

SNORT has some predefined signatures to protect DNS from various attackers. The snort can detect DNS-based attacks at the preprocessor level, which is disabled by default. This check involves an overflow in the DNS Client's RDATA and the detection of experimental and obsolete record types. The current signatures for DNS attacks protect against "an attempt on DNS name and its version" and "an attempt on zone transfer". The DNS-related rules are collected in two files, i.e., "dns.rules" and "protocol-dns, which contains around 85 signatures". The signatures for DNS tunneling and amplification for some of the tools listed in Section 3 are not available in the SNORT signature set.

In our experiment, we have removed all the default signatures from SNORT, and along with "dns.rules" and "protocol-dns", we have added

Table 3
Experimental setup configuration.

System name	CPU	Memory	OS used	IP address
Victim DNS	8 Core Xeon Processor	16 GB	Centos 8.1	14.139.152.14
DID	8 Core Xeon Processor	16 GB	Centos 8.1	14.139.152.5
SNORT-IDS	8 Core Xeon Processor	16 GB	Centos 8.1	14.139.152.6
Attacker System	8 Core Xeon Processor	32 GB	Kali 2020	223.31.121.172

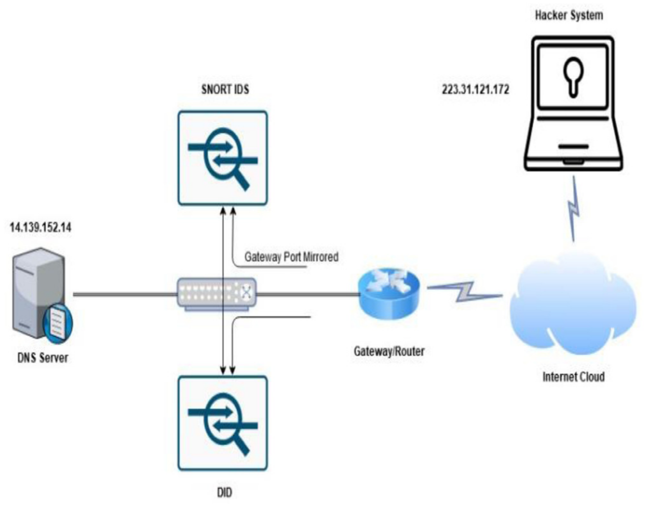


Fig. 13. Experiment setup of DID evaluation.

the novel signatures created in Section 3 for amplification, tunneling, and DoS attacks, and this resultant SNORT-based solution is called DNS Intrusion Detection — DID.

For our experiment, we used a physical server with 128 GB of memory and a 16-core, two-socket CPU. This server is allocated for virtualization with the commonly used hypervisor “KVM” on CentOS-8.1. In this experiment, we installed DID on CentOS-8.1 virtual machines. The victim DNS server is installed on another CentOS-8.1 virtual machine. The Kali Linux virtual machine is configured on another physical server located on a different physical network than the victim system for performing DNS attacks. The experiment is conducted in a real but controlled network environment with Internet-exposed public IP addresses assigned to the victim’s DNS server virtual machine. The Attacker virtual machine is also assigned a public IP on the different networks. The whole network setup is described in Fig. 13.

We installed and configured the following virtual machines for our experiment:

Victim DNS Server Virtual Machine — This system receives an enormous request from attacker’s systems for various DNS attacks.

Attacker Virtual Machine — This virtual machine is used to perform various attacks, e.g., DNS tunneling, amplification, and DNS DoS attacks using all the tools listed in Section 3.

DID Virtual Machine — SNORT version 2.9 is installed and configured on this system. All the signatures are removed except the DNS rules, and the novel signatures created in Section 3 are added to the existing DNS rules. The proposed solution DID is configured and placed in monitor mode, and all the traffic of the victim DNS server is mirrored to the DID.

SNORT Virtual Machine — SNORT version 2.9 is installed and configured with registered signatures. The signature count of snort is approximately 50,000, and snort will get the same traffic as DID gets.

The configuration of each system in the experimental setup is listed in Table 3.

Table 4
Confusion matrix.

Actual	Predicted	
	Legitimate	Intrusion
Legitimate	True Negative	False Positive
Intrusion	False Negative	True Positive

5. Comparative evaluation of DID and SNORT concerning DNS Attacks

The matrix for the evaluation of the performance of IDS is called a confusion matrix, which represents the result of classification as true or false [31]. The possibilities for classifying events shown in Table 4 are as follows:

True Positive: Intrusions that are identified effectively by the IDS

True Negative: Legitimate traffic is identified as legitimate by the IDS.

False Positive: Legitimate traffic is wrongly classified as intrusive traffic by the IDS.

False Negative: Intrusions that are missed by the IDS and classified as legitimate traffic.

We embrace standard measures for the assessment models; the standard performance measures for evaluating IDS are as follows:

1. Detection Rate or Sensitivity: The ratio between predicted attacks and the total number of attacks. It is extremely rare for an IDS that all the possible attacks are successfully detected, and this ratio becomes 1. This is also called the “True positive rate” and can be expressed mathematically as :

$$\boldsymbol{True\ Positive\ Rate\ (TPR)} = \frac{\boldsymbol{TP}}{\boldsymbol{TP+FN}}$$

2. False Positive Rate: The ratio between the number of legitimate instances wrongly classified as intrusion and the total number of legitimate instances.

$$\boldsymbol{False\ Positive\ Rate\ (FPR)} = \frac{\boldsymbol{FP}}{\boldsymbol{FP+TN}}$$

3. Accuracy or Classification Rate: The accuracy is calculated as the ratio of correctly classified instances and the total number of instances. It assesses the IDS’s ability to detect legitimate or malicious traffic behavior.

$$\boldsymbol{Accuracy\ (CR)} = \frac{\boldsymbol{TP+TN}}{\boldsymbol{TP+TN+FP+FN}}$$

To the best of our knowledge and at the time of writing this paper, there are no datasets available on the Internet that emulate DNS-based attacks. Therefore, the experiment setup in Section 4 was running for a day, where we observed the DNS traffic and tagged the traffic as legitimate or intrusive. The performance of DID was evaluated by performing various experimental attacks as follows:

Experiment-1: DNS amplification attack

The DNS amplification attacks were performed using various tools listed in Section 3 by the Kali Linux system connected to the different ISP network. For a day, the attacker system launched the attack at different instances using the DNS amplification tools discussed in Section 3.

Table 5
Evaluation of SNORT and DID signatures.

Attack type	Attack tool name	Number of instances	Number of alerts	
			SNORT	DID
DNS tunnel	Iodine	4	0	4
	DNSCAT2	2572	0	3610
	DNS2TCP	8	0	8
	Thunderdns	11	0	11
	Ozmandns	277	0	277
DNS Amplification	Ethanwilloner	1070000	0	107
	Saddam	18150000	0	1815
	Dns flooderv1.1	140000	0	14
	Dnsdrdos	21030000	0	2103
DoS	Flood DNS [28]	890000	0	89
Legitimate Traffic		5408922		

Table 6
Efficiency of DID signatures.

Tool name	True positive	False positive	False negative
Iodine	4	0	0
DNSCAT2	3610	1038	0
DNS2TCP	8	0	0
Thunderdns	11	0	0
Ozmandns	277	0	0
Ethanwilloner	107	0	0
Saddam	1815	0	0
Dns Flooderv1.1	14	0	0
Dnsdrdos	2103	0	0
Flood DNS	89	0	0
Total	**8038**	**1038**	**0**

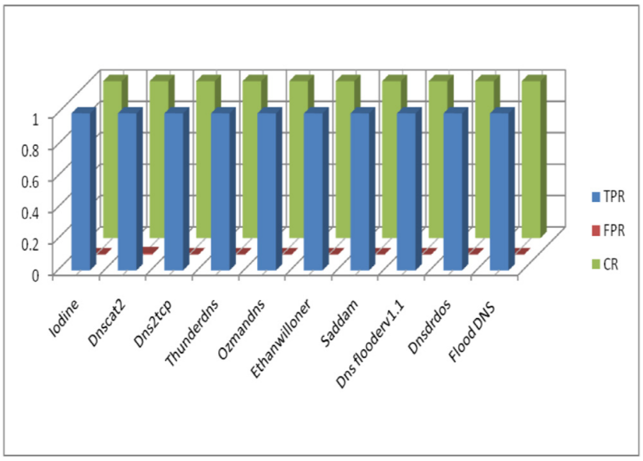


Fig. 14. Performance analysis of DID.

Experiment-2: DNS Tunneling Attack
As a part of this experiment, the tunneling attacks were launched using DNS tunneling attack tools discussed in Section 3. The attack was launched in different instances over a period of a day.

Experiment-3: DoS attack on DNS server
The DoS attack was performed using various tools like DNS-Flood and Hping at different instances over a period of a day.

Tables 5 and 6 show the effectiveness of the proposed signatures to identify various DNS-based attacks. With the help of the tools listed in Section 3, we launched DNS tunneling, amplification, and DoS attacks, and the results are summarized in Table 5 as the efficacy of SNORT and DID signatures, revealing that DID outperforms then SNORT with respect to DNS attacks.

The efficiency of DID signatures is depicted in Table 6. We received true positive results for every signature developed in Section 3 which reveals that all the attack instances were successfully captured by DID. We also observed false positive results for the only DNSCAT2 signature, where 1038 instances of legitimate traffic are identified as DNSCAT2 attacks. However, we did not observe any false negative results for any of the signatures of DID ensuring no intrusive traffic is treated as legitimate. Our result shows that the detection rate for identifying DNS attacks has improved drastically. As opposed to the classic rules for the DNS defense mechanism of SNORT, the proposed new rules can accurately detect DNS amplification, DNS tunneling, and DNS-based DoS attacks. Fig. 14 shows the TPR, FPR, and CR for signatures created in Section 3; except for signatures of DNSCAT2, the TPR and CR is 1 for all other signatures.

6. Conclusion and future scope of work

In this research paper, we have suggested a snort-based intrusion detection system specifically for DNS protocol anomalies and created an attack signature for various tools of DNS tunneling attacks, DNS amplification attacks, and DoS attacks on DNS. We have assessed DID against a wide range of DNS attacks using various attack tools available on the Internet. The simulation experiments on standard SNORT IDS deployment with inbuilt signatures showed that the proposed solution “DID” is suitable for attack classification in NIDS for DNS-based attacks. Our findings show a 100% attack detection rate, a 0.01918% false-positive alert rate, and 99.98% accuracy.

The future scope of work includes a machine learning-based approach to identifying attack patterns of DNS amplification attack tools & DNS tunneling attack tools and creating appropriate signatures dynamically.

CRediT authorship contribution statement

Sanjay Adiwal: Conceptualization, Methodology, Writing – original draft, Data curation, Visualization, Experiment evaluation. **Balaji Rajendran:** Writing – review & editing. **Pushparaj Shetty D.:** Supervision, Validation, Investigation. **Sithu D. Sudarsan:** Supervision, Validation, Investigation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Adam Ali Zare Hudaib, E.A.Z. Hudaib, DNS advanced attacks and analysis, Int. J. Comput. Sci. Secur. (IJCSS) 8 (2) (2014) 63.
[2] Herzberg Amir, Haya Shulman, Retrofitting security into network protocols: The case of dnssec, IEEE Internet Comput. 18 (1) (2014) 66–71, http://dx.doi.org/10.1109/MIC.2014.14.

[3] Shaikh Asadullah, Bhavika Pardeshi, Faraz Dalvi, Overcoming threats and vulnerabilities in DNS, in: Proceedings of the 3rd International Conference on Advances in Science & Technology, ICAST, 2020, http://dx.doi.org/10.2139/ssrn.3568728.

[4] G. Alexis, DNSSEC operational impact and performance, in: 2006 International Multi-Conference on Computing in the Global Information Technology, ICCGI'06, IEEE, 2006, p. 63, http://dx.doi.org/10.1109/ICCGI.2006.27.

[5] N. Wilde, L. Jones, R. Lopez, T. Vaughn, A DNS RPZ firewall and current American DNS practice, in: International Conference on Information Science and Applications, Springer, Singapore, 2018, pp. 259–265, http://dx.doi.org/10.1007/978-981-13-1056-0_27.

[6] A. Drozdova, Securing a DNS server with snort IDS: severen-telecom case, 2015.

[7] F. Hock, P. Kortis, Design implementation and monitoring of the firewall system for a DNS server protection, in: 2016 International Conference on Emerging ELearning Technologies and Applications, ICETA, IEEE, 2016, pp. 91–96, http://dx.doi.org/10.1109/ICETA.2016.7802040.

[8] P. Satam, H. Alipour, Y. Al-Nashif, S. Hariri, Dns-ids: Securing dns in the cloud era, in: 2015 International Conference on Cloud and Autonomic Computing, IEEE, 2015, pp. 296–301, http://dx.doi.org/10.1109/ICCAC.2015.46.

[9] P. Satam, H.R. Alipour, Y.B. Al-Nashif, S. Hariri, Anomaly behavior analysis of DNS protocol, J. Internet Serv. Inf. Secur. 5 (4) (2015) 85–97.

[10] S. Cheung, K.N. Levitt, A formal-specification based approach for protecting the domain name system, in: Proceeding International Conference on Dependable Systems and Networks, IEEE, 2000, pp. 641–651, http://dx.doi.org/10.1109/ICDSN.2000.857602.

[11] S. Rastegari, M.I. Saripan, Rasid, Detection of denial of service attacks against domain name system using neural networks, Int. J. Comput. Sci. Issues 6 (2009).

[12] Y.F. Mohammed, Network-Based Detection and Prevention System Against DNS-Based Attacks, University of Arkansas, 2021.

[13] T. Ghosh, E. El-Sheikh, W. Jammal, A multi-stage detection technique for DNS-tunneled botnets, in: CATA, 2019, pp. 137–143.

[14] S. Spacek, M. Lastovicka, M. Horak, T. Plesnik, Current issues of malicious domains blocking, in: 2019 IFIP/IEEE Symposium on Integrated Network and Service Management, IM, IEEE, 2019, pp. 551–556.

[15] R. Mitsuhashi, Y. Jin, K. Iida, T. Shinagawa, Y. Takai, Malicious DNS tunnel tool recognition using persistent DoH traffic analysis, IEEE Trans. Netw. Serv. Manag. (2022) http://dx.doi.org/10.1109/TNSM.2022.3215681.

[16] C. Marques, S. Malta, J. Magalhães, DNS firewall based on machine learning, Future Internet 13 (12) (2021) 309, http://dx.doi.org/10.3390/fi13120309.

[17] Y. Ni, Z. Guo, Y. Mo, L. Shi, On the performance analysis of reset attack in cyber–physical systems, IEEE Trans. Automat. Control 65 (1) (2019) 419–425, http://dx.doi.org/10.1109/TAC.2019.2914655.

[18] StatDNS, DNS related RFCs, An up to date list of domain name system RFCs, 2022, https://www.statdns.com/rfc/. (Accessed 25 October 2022).

[19] IANA, domain name system (DNS) parameters, 2022, https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml. (Accessed 25 October 2022).

[20] A. Nadler, A. Aminov, A. Shabtai, Detection of malicious and low throughput data exfiltration over the DNS protocol, Comput. Secur. 80 (2019) 36–53, http://dx.doi.org/10.1016/j.cose.2018.09.006.

[21] Erik Ekman, IODINE, GitHub repository, 2014, https://github.com/yarrick/iodine. (Accessed 25 October 2022).

[22] Ronn Bowes, DNSCAT2, GitHub repository, 2010, https://github.com/iagox86/dnscat2. (Accessed 25 October 2022).

[23] Olivier Dembour, DNS2TCP, GitHub repository, 2017, https://github.com/alex-sector/dns2tcp. (Accessed 25 October 2022).

[24] FBK CyberSecurity research laboratory, Thunderdns, GitHub repository, 2019, https://github.com/fbkcs/ThunderDNS. (Accessed 25 October 2022).

[25] DNStunnel.de, OzymanDNS Tunnel tool, 2004, https://dnstunnel.de/. (Accessed 25 October 2022).

[26] Ethan Willoner, DNS amplification, GitHub repository, 2013, https://github.com/ethanwilloner/DNS-Amplification-Attack. (Accessed 25 October 2022).

[27] OffensivePython, saddam DNS amplification, GitHub repository, 2015, https://github.com/OffensivePython/Saddam. (Accessed 26 October 2022).

[28] Plxsertr, dnsreflect - DNS Flooder v1.1, GitHub repository, 2014, https://github.com/plxsertr/dnsreflect. (Accessed 26 October 2022).

[29] Noptrix, nullsecurity, 2015, https://nullsecurity.net/tools/dos.html. (Accessed 26 October 2022).

[30] Google code archive, DNS-flood, 2012, https://code.google.com/archive/p/dns-flood/. (Accessed 26 October 2022).

[31] B.S. Khater, A.W. Wahab, M.Y.I. Idris, M.A. Hussain, A.A. Ibrahim, M.A. Amin, H.A. Shehadeh, Classifier performance evaluation for lightweight IDS using fog computing in IoT security, Electronics 10 (14) (2021) 1633, http://dx.doi.org/10.3390/electronics10141633.