A Closer Look into DHCP Starvation Attack in Wireless Networks

Article *in* Computers & Security · October 2016

DOI: 10.1016/j.cose.2016.10.002

CITATIONS
28

READS
6,347

2 authors, including:

Nikhil Tripathi
Indian Institute of Technology Indore
17 PUBLICATIONS **284** CITATIONS

SEE PROFILE

A Closer Look into DHCP Starvation Attack in Wireless Networks

Neminath Hubballi and Nikhil Tripathi
Discipline of Computer Science and Engineering, School of Engineering
Indian Institute of Technology Indore
{neminath, phd1401101002}@iiti.ac.in

***Abstract*—Dynamic Host Configuration Protocol (DHCP) is used by clients in a network to configure their interface with IP address and other network configuration parameters such as Default Gateway and DNS server IP addresses. This protocol is vulnerable to a Denial of Service (DoS) attack popularly known as classic DHCP starvation attack. In this paper, we make threefold contribution. First, we highlight the practical difficulty in generating classic DHCP starvation attack in wireless networks. Secondly, we propose a stealth starvation attack which is effective in wireless networks, easier to launch, requires fewer number of messages to be transmitted and difficult to detect by known detection methods. We also show a structurally similar attack in IPv6 networks which can affect address configuration protocols such as DHCPv6 and *StateLess* Address Autoconfiguration (SLAAC). Subsequently, we also describe an anomaly detection method to detect the proposed attack. We design and generate the attacks in a real network setup and report the results. The proposed detection method use the Hellinger distance between two probability distributions generated from training and testing data to detect starvation.**

***Index Terms*—DHCP, DHCPv6, SLAAC, DHCP Starvation Attack, ARP Poisoning, 802.11 Wi-Fi Network, Hellinger Distance.**

I. Introduction

Dynamic Host Configuration Protocol (DHCP) is used for automatic IP address allocation to DHCP enabled clients. It also enables DHCP clients to configure themselves with other network parameters like subnet mask, default gateway IP address, etc. This protocol is vulnerable to a Denial of Service (DoS) attack popularly known as DHCP starvation attack. In this case, a malicious insider can launch a DoS attack preventing other clients from acquiring an IP address. There are open source tools like Gobbler [1], DHCPIG [2] to launch this attack. These tools use MAC address spoofing to generate large number of IP requests and for every request, a new IP address is released by a DHCP server and this exhausts the address pool. However this method is not effective in a wireless network as described in Section II-B.

In this paper, we propose a new DHCP starvation attack termed as *Induced DHCP Starvation* that does not require a malicious client to inject large number of IP request messages using random MAC addresses to exhaust IP address pool. Instead, the attack is intended to exploit a loophole present in DHCP client-side IP address conflict detection scheme. According to RFC 2131 [3], a DHCP enabled client should probe an allocated IP address to check if IP address is already in use. To launch the proposed attack, a malicious client injects a fake reply in response to probe request sent by the client. Once the DHCP client receives the corresponding reply, it declines the IP address offered and informs DHCP server by sending a decline message. When DHCP server receives this decline message, it marks the offered IP address as unavailable for the length of the lease period. These set of operations are repeated for every probe request sent to detect IP address conflict, thereby, preventing a DHCP client from acquiring an IP address which is a case of DoS. This attack is equally effective in both wired and wireless networks. We show that, the proposed attack is effective in DHCPv6 also. We also propose a statistical abnormality measurement technique that uses the Hellinger distance between two probability distributions generated from training and testing data to detect starvation. We evaluate proposed detection method to detect *Induced DHCP Starvation* attack presented in this paper and the attack we proposed in one of our previous work [4].

Rest of the paper is organized as follows. We provide an overview of DHCP working and two variants of classic DHCP starvation attacks in Section II. In Section III, we describe our proposed DHCP starvation attack. Experimental studies are presented in Section IV. Relevance of proposed starvation attack in IPv6 networks is discussed in Section V. We provide the overview of other DHCP starvation detection methods in Section VI. We also describe a method for detecting proposed attack in Section VII. Finally, the paper is concluded in Section VIII.

II. Background

A. DHCP Working

In a network, a DHCP server is configured with a pool of IP addresses and other network parameters [3]. To get an IP address, four messages are exchanged between a DHCP client and the DHCP server as shown in Figure 1. The complete process of automatic IP address allocation is known as *DORA* process where 'D','O','R' and 'A' stands for 'Discover', 'Offer', 'Request' and 'Acknowledgment' respectively. RFC 2131 [3] describes few other DHCP messages that are exchanged during unsuccessful IP address configuration. DHCPDECLINE is one such message that is sent by a client to DHCP server if client finds that the assigned IP address is already in use by any

other client in the network. DHCP clients usually detect such IP address conflicts by sending probe requests destined to the IP address allotted.

B. Classic DHCP Starvation Attacks

Due to the absence of an in-built authentication in DHCP, it is vulnerable to an attack popularly known as DHCP starvation attack. In this subsection, we describe two popular methods of generating DHCP starvation attack along with few limitations of these methods.

1) Using Similar (Random) MAC Address in Ethernet Header and DHCP Header (Type-1): In this case, a malicious client sends large number of DHCPDISCOVER messages using spoofed source MAC addresses to a DHCP server. Same MAC address (spoofed) is used in the CHADDR field of DHCP header and Ethernet header. The server allots IP address to each of these requests using *DORA* process. This results in the exhaustion of IP address pool present at DHCP server. As a result, new legitimate clients will be denied from obtaining an IP address from the DHCP server. In wireless networks this attack can not be so easily generated. IEEE 802.11 standard has an association phase of a wireless client with Access Point (AP) and also session key generation phase (WPA 2) which uses client's MAC address. The sequence of operations done while a client connects to AP in WPA2 is shown in Figure 2 [5]. AP drops packets sent or received with random MAC addresses as these are not associated with it. As a result, this type of attack does not work in wireless networks unless the malicious client precedes an association with AP for each spoofed address. However, considering the association phase and the session key exchange using 4-way handshake shown in Figure 2, it is not easy for a malicious client to complete these steps for every spoofed MAC address. Typically, AP has restriction of allowing only limited number of clients connecting to it. This also prevents address pool exhaustion by any malicious client if the number of addresses in the pool is greater than the associations permitted by AP.

2) Using Dissimilar MAC Address in Ethernet Header and DHCP Header (Type-2): In this type of starvation, a malicious client uses dissimilar MAC address in Ethernet header and CHADDR field of DHCP header. While sending a DHCPDISCOVER message, a malicious client uses its own MAC address in the Ethernet header as a source MAC address but a different random MAC address (spoofed) in the CHADDR field of DHCP header [3]. In a wireless network, AP accepts these DHCPDISCOVER messages and forward it to DHCP server because the source MAC address (i.e. malicious client's MAC) is associated to the AP. When the server receives this DHCPDISCOVER message, it uses the MAC address from the CHADDR field of DHCP header to send a unicast DHCPOFFER message. As this address is spoofed by malicious client and there was no association of this address with AP, it drops DHCPOFFER packet. Hence *DORA* process is not completed and malicious client fails to exhaust address pool to create DHCP starvation.

TABLE I
NOTATIONS

MAC_VICTIM	MAC address of victim client
MAC_MALICIOUS	MAC address of malicious client
ARP_REQ	ARP Request
ICMP_REQ	ICMP Request
ARP_REPLY	ARP Reply
ICMP_REPLY	ICMP Reply
SrcIP	Source IP address
TarIP	Target IP address
SrcMAC	Source MAC address
DestMAC	Destination MAC address
IP_SERVER	IP Address of DHCP Server
MAC_SERVER	MAC Address of DHCP Server
IP_SELECTED	IP Address selected for offering by DHCP Server

III. INDUCED DHCP STARVATION ATTACK

In this section, we describe a method of generating *Induced DHCP Starvation* attack in both wired and wireless networks. This attack requires fewer messages and is light weight on malicious client as compared to classic starvation attack.

Consider a network setup as shown in Figure 3. In the figure, there are 4 entities namely Access Point (AP), a victim client which fails to acquire IP configuration, a malicious client and DHCP server. Both malicious client and victim client are connected to AP through wi-fi link and DHCP server is connected to AP through a wired connection. We use the notations shown in Table I to describe proposed attack.

The sequence of messages exchanged to launch this type of starvation attack is as follows.

1) DHCPDISCOVER by victim: When the victim client comes up, it associates itself with AP by a handshake mechanism. It then sends a DHCPDISCOVER message to find the available DHCP server(s) which can lease an IP address.
2) DHCPOFFER by server: DHCP server(s) reply with a DHCPOFFER message in response to victim's DHCPDISCOVER.
3) DHCPREQUEST by victim: Victim requests the IP address offered in previous stage. This message has the MAC address of victim i.e. MAC_VICTIM. As this message is a broadcast message, the malicious client also receives this message and it takes note of MAC_VICTIM.
4) DHCPACK by server: The server sends an acknowledgement confirming the allocation of address to victim client.
5) ARP_REQ by victim: Victim client send an ARP_REQ broadcast packet to check if any other client in the network is using the IP address offered by server. The source IP address of this ARP request is set to "0.0.0.0" [6]. This address is used as the client does not have a configured IP address yet.
6) ARP_REPLY by malicious client: Malicious client sniffs ARP_REQ sent by victim with a filter set on source IP address "0.0.0.0". Malicious client replies with a false ARP_REPLY with the IP address allocated to

victim as source IP address and its own MAC address MAC_MALICIOUS as source MAC address. The target IP address of this message is set to "0.0.0.0" and target MAC address to MAC_VICTIM.

7) DHCPDECLINE by victim: Once the victim client receives spoofed ARP_REPLY, it interprets this message as some other client is already using the IP address offered to it by server [7]. In order to avoid IP conflict, victim client sends a DHCPDECLINE message to server denying the use of IP address offered to it earlier.

Once DHCP server receives this DHCPDECLINE message, it marks the IP address as unavailable for the length of lease period [3]. Victim client re-initiates the process of acquiring a new IP address again and set of operations are repeated; effectively denying the victim client from acquiring an address. This ensures the victim client is starved. As the server marks every declined IP address unavailable for the length of the lease, it will run out of the addresses in the address pool after few iterations.

Figure 4 and 5 shows the sequence of operations in *Induced DHCP Starvation* attack from victim client's and malicious client's perspective respectively. The notations used in the flow charts are in sync with the ones used in Table I.

To increase readability of this paper, we describe our previously proposed attack [4] also below.

A. Exploiting Server-side IP Address Conflict Detection Scheme:

RFC 2131 mandates a DHCP server to check if IP address selected from the DHCP pool is already in use by some other client in the network. Thus, DHCP server probes the selected IP address before offering it to any DHCP client. A malicious insider sends fake reply in response to the probes sent by DHCP server to launch the attack. The sequence of events to launch this attack is listed below.

1) DHCPDISCOVER message by victim: As soon as victim client connects to the network, it sends a DHCPDISCOVER message to find the available DHCP server(s) in the network.
2) Probe Request by DHCP server: Once DHCP server receives DHCPDISCOVER message sent by victim client, it selects one IP address from its pool, say IP_SELECTED, and sends a probe request to check if this IP address is in use by some other client. Depending on the DHCP's software implementation, server uses either ARP_REQ or ICMP_REQ for the probing purpose. However, it should be noted that even if ICMP_REQ probe is used, ARP_REQ is automatically generated by the system to get the MAC address of the client which is using IP_SELECTED.
   (a) ARP_REQ as a probe: If server selects ARP_REQ as probe; the source IP, source MAC, target IP and destination MAC addresses of this probe are set to IP_SERVER, MAC_SERVER, IP_SELECTED and *"ff:ff:ff:ff:ff:ff"* respectively.
   (b) ICMP_REQ as a probe: If server selects ICMP_REQ as probe; it crafts an ICMP_REQ with source IP and destination IP as IP_SERVER and IP_SELECTED respectively.
3) Fake Probe Reply by malicious client: Malicious client sniffs probe sent by server with a filter set on source IP address, IP_SERVER, and then injects corresponding fake probe reply.
   (a) ARP_REPLY as a probe reply: Corresponding to ARP_REQ sent by DHCP server as probe, the malicious client sends a fake ARP_REPLY by setting the source IP, source MAC, target IP and destination MAC addresses to IP_SELECTED, MAC_MALICIOUS, IP_SERVER and MAC_SERVER respectively.
   (b) ICMP_REPLY as a probe reply: If the malicious client receives the ICMP_REQ, it replies back with a fake ICMP_REPLY having its source IP and destination IP as IP_SELECTED and IP_SERVER respectively.

Once DHCP server receives ARP_REPLY/ICMP_REPLY, it interprets that some other client is already using IP_SELECTED. As a result, server marks this IP address unavailable for the length of lease and does not offer it to the victim client. Victim client re-initiates the process of acquiring a new IP address again and set of operations are repeated effectively denying the victim client from acquiring an address.

Figure 6 and 7[1] shows the sequence of operations in previously proposed attack from victim client's and malicious client's perspective respectively. The notations used in the flow charts are in sync with the ones used in Table I.

B. Comparing Induced DHCP Starvation Attack with Type-1 and Type-2 Attacks

In this subsection, we discuss how proposed *Induced DHCP Starvation* attack is different from previously known Type-1 and Type-2 DHCP starvation attacks. We also describe how it is more effective and easier to launch as compared to Type-1 and Type-2 DHCP starvation attacks. The differences are as follows:

- Attack Launching Technique: On the one hand where classic DHCP starvation attacks require sending of multiple DHCPDISCOVER and DHCPREQUEST messages to consume IP address pool, *Induced DHCP Starvation* attack requires sending of spoofed ARP replies in response to ARP requests sent by victim client. The attack proposed in [4] also requires sending of spoofed ARP/ICMP replies but it is sent in reponse to ARP/ICMP probes sent by DHCP server to detect address conflict.
- Attack's Effectiveness in Wired and Wireless Networks: As discussed earlier, though classic DHCP starvation attacks are effective in wired networks, they are ineffective in case of wireless networks unless the malicious client precedes an association with AP for each spoofed MAC

[1]For simpler representation, we show the case in which DHCP server selects ARP_REQ as probe.

address which is quite tedious. Even if fake associations are preceded, classic starvation attacks again fail to exhaust IP address pool in wireless networks due to restrictions on number of such associations enforced by AP. On the other hand, since *Induced DHCP Starvation* attack and attack proposed in [4] neither require spoofed MAC addresses nor do they require fake associations of these MAC addresses with AP, they are quite easy and effective in both wired and wireless networks.

- Traffic Overhead due to Attack: In case of classic DHCP starvation attack, two DHCP messages (one DHCPDISCOVER and one DHCPREQUEST) are required to consume one IP address from the pool. The number of messages which attack proposed in [4] requires depends upon the type of requests DHCP server is sending in order to detect address conflict. For example, in [4], we used ISC DHCP server which uses ICMP requests for IP address conflict detection. In that case, the malicious client required two messages (one ARP reply followed by one ICMP reply) to consume one IP address. If we consider most of the built-in DHCP softwares available with different wireless access points, they use ARP request for the conflict detection purpose. So in this case, the malicious client just needs to send one ARP reply message only. The *Induced DHCP Starvation* attack, however, requires only one message (ARP reply) to consume one IP address because victim clients use only ARP for address conflict detection. We must notice that clients can not use protocols like ICMP for address conflict detection as other protocols require an IP address to be already configured[2] on the interface for communication.

Table II shows a brief summary of different types of DHCP starvation attacks.

C. Comparing Induced DHCP Starvation Attack with DNS Poisoning Attack

In this subsection, we discuss how proposed attack compares with DNS [8] poisoning attack which is another popular Denial of Service attack at the application layer [9]. We also describe how it is easier to launch the proposed attack as compared to DNS poisoning. The differences are as follows:

- Entity that poisons Address Pool: In case of DNS poisoning attack, it is the malicious client that needs to poison DNS cache so that victim clients receive an invalid response for the query sent by them. However, the proposed *Induced DHCP Starvation* attack does not require a malicious client to poison the data store (DHCP server's IP address pool in our case). Instead, it is the victim client that finally poisons the address pool available with DHCP server by sending DHCPDECLINE message. Malicious client just needs to send spoofed ARP Reply in response to the ARP Request sent by a victim client. This ARP request is sent by victim client to check if any other client is already using the IP address offered by DHCP server. As soon as victim client receives the spoofed ARP Reply, it interprets that the offered IP address is already in use by some other client (malicious client in this case). In order to avoid IP address conflict, victim client sends a DHCPDECLINE message to DHCP server denying the use of IP address offered to it earlier. Once DHCP server receives this DHCPDECLINE message, it marks the IP address as unavailable for the length of lease period. This causes the poisoning of DHCP server's IP address pool. Thus, we see it is the DHCPDECLINE message (sent by victim client) but not the spoofed ARP reply (sent by malicious client) that poisons the address pool.
- Race Condition: DNS poisoning requires a malicious client to beat legitimate authoritative DNS servers by sending a fake query response with the hope that their fake response reach at the caching nameserver earlier. These race conditions make DNS poisoning difficult for the malicious client to succeed. However, a malicious client never requires to face a race condition to launch *Induced DHCP Starvation* attack. Since the IP address offered to victim client by DHCP server is not allotted to any other client, there is no candidate eligible to send the corresponding ARP Reply except malicious client who intentionally responds with the fake ARP reply. As a result, malicious client does not face any race condition to launch *Induced DHCP Starvation* attack which makes this attack easier to launch.
- Bandwidth Requirement: In order to win the race situation discussed above to launch DNS poisoning, malicious client needs to send a flood of fake query responses towards caching nameserver. This requires a lot of malicious client's bandwidth. However, as discussed earlier, *Induced DHCP Starvation* attack does not require any race condition. As a result, the malicious client's bandwidth required to launch *Induced DHCP Starvation* attack is also minimal.
- Impact Coverage: No doubt DNS poisoning can affect internet services on a much wider scale, however, *Induced DHCP Starvation* can also affect internet access for a particular group of clients. The impact of *Induced DHCP Starvation* depends on the size of LAN in terms of number of clients connected to it. Since clients can not acquire an IP address due to presence of *Induced DHCP Starvation* attack in LAN, they can not communicate with their default gateway either. As a result, victim clients do not have any idea of where to forward their traffic due to which they can not access internet services too, resulting into a global impact.

IV. Experiments and Discussions

In order to demonstrate *Induced DHCP Starvation* attack, we executed starvation attacks in one of our departmental network. This network has a switch (Cisco WS-C2950T) and also a wireless access point (Netgear N150 Wireless Router) allowing clients to connect to it. We had setup a DHCP server

[2]According to RFC 2131, a client should not configure its interface with an IP address before performing address conflict detection.

TABLE II
COMPARISON OF DIFFERENT DHCP STARVATION ATTACKS

Type of starvation attack	Attack launching technique	Requirement of spoofed MAC addresses	Attack launching difficulty in wired/wireless networks	Number of messages required to consume 1 IP address
Type-1	Using multiple DHCPDISCOVER messages each with same MAC address in CHADRR field and Ethernet header	Yes	Easy/Difficult	2
Type-2	Using multiple DHCPDISCOVER messages each with different MAC address in CHADRR field and Ethernet header	Yes	Easy/Difficult	2
Attack proposed in [4]	Using spoofed replies in response to DHCP server's probe requests	No	Easy/Easy	2 or 1 depending on probe's protocol
Induced DHCP Starvation attack	Using spoofed replies in response to victim client's probe requests	No	Easy/Easy	1

on the wired part of network behind a switch as shown in Figure 3. This server was configured to offer 256 IP addresses[3] in the range of 10.200.1.0 to 10.200.1.255. Malicious client was running Kali Linux (version 1.1.0a) while other clients were running Windows 7 (service pack 1). A C program was written to sniff ARP requests with source IP address set to "0.0.0.0" and also to send spoofed ARP replies. These two functions were implemented as two threads. First thread did the sniffing part and other thread generated spoofed ARP replies. We generated *Induced DHCP Starvation* attack in both wired and wireless networks[4]. In each case, we created two scenarios and in both the scenarios, a malicious host was injecting spurious ARP replies to create starvation and it was the first machine to be connected to network with an IP address from the pool. Subsequently, we manually triggered other machines one after the other to use DHCP to get an address from the server.

A. In Wireless Networks

In this experiment, we connected all the clients to access point and created a wireless LAN. We tested the AP's permitted number of clients connecting to it by connecting one machine at a time to AP and we found that it allows 34 clients to associate. The 35th association request was rejected and client displayed a *"Can't connect to this network"* message. Subsequently, we used this setup of 34 machines (including malicious and victim clients) to generate *Induced DHCP Starvation* attack. We created two scenarios, one with only one victim and second with multiple victims as explained below.

I. First Scenario: In this scenario, there was a designated victim and only this machine was prevented from acquiring an IP address. This was done by sending spurious ARP replies only to this client's verification probes. Remaining computers acted as genuine hosts requesting for IP address in between and they did so successfully. Figure 8 shows the number of IP addresses (blue line) released and the number of spurious ARP replies (black line) that malicious client sent over time. The left y-axis value is the number of addresses released and the right y-axis value is the number of fake ARP replies sent in a time interval of 200 seconds. The numbers in blue and black boxes show total number of IP addresses released[5] and total number of ARP replies sent by malicious client up to that interval (cumulative) respectively. We can notice that, in 2600 seconds, all the IP addresses were released. We can also notice that roughly around 2000 seconds these two lines (blue and black) merge. This is because until this point, other 32 clients came and joined the network and had already acquired IP addresses successfully as malicious client did not reply to these clients' verification probes. Thus there is a difference of 32 between the number in blue and black box at this point. However victim client had not got the IP address yet and it continued its attempt to acquire an IP address and all those attempts were prevented by malicious client with a fake ARP reply. Hence beyond this point, IP addresses were only marked as unusable (but released) and was equal to number of ARP replies sent by malicious client.

II. Second Scenario: In the second scenario, malicious host was preventing every other host from acquiring an IP address (every machine was a victim). Figure 9 shows the number of IP addresses released (blue line) and the number of spurious ARP replies (black line) that malicious client sent over time. We can notice that in 350 seconds, all the IP addresses were released. Compared to the first case, IP address pool was exhausted much quickly. This was because all the hosts were simultaneously trying to acquire IP address and every attempt failed and DHCP server marked one IP address unusable. We can also notice that the number of IP addresses released and number of ARP replies sent by malicious client are equal

[3]Among 256 , one each was allotted to malicious client and server itself. Other two IP addresses, 10.200.1.0 and 10.200.1.255 were Network and Broadcast address respectively and were not used.

[4]Although the focus is made on wireless networks, we executed *Induced DHCP Starvation* attack in wired networks also to effectively compare it with classic starvation attack

[5]Number of IP addresses released is sum of victim client's attempt to acquire an IP address and other clients' successful attempt

throughout. This is because, the malicious client was sending spurious ARP replies for each and every client's verification probes and there were no successful attempts of IP address acquisition.

B. In Wired Networks

For this experiment, we had 43 clients (including malicious and victim clients) and the DHCP server connected to the network switch thus creating a wired network. In this case also we experimented with one victim and multiple victim scenarios as in the previous case. Figure 10 and Figure 11 shows the number of IP addresses released/number of ARP replies sent over time. We can notice that the rate of allocation is almost similar to previous case. We can also notice that it took around 2600 seconds and 300 seconds to exhaust the address pool in one victim and multiple victims scenarios respectively.

C. Rate of IP Address Pool Exhaustion

In case of *Induced DHCP Starvation* attack, the rate of pool exhaustion depends upon the rate with which new clients come and join the network. The reason is malicious client injects spoofed ARP replies only when it receives corresponding ARP requests with source IP address "0.0.0.0". Such ARP requests are generated only after completion of *DORA* processes by DHCP clients. If the rate of clients' entry increases, the rate of *DORA* processes completion and ARP requests generation will also increase that will lead to injection of more number of fake ARP replies. This eventually causes rapid pool exhaustion.

D. Traffic Overhead Comparison

In order to compare the overhead on malicious client to generate induced and classic starvation attacks, we calculated the number of messages sent by a malicious client in case of a single victim and multiple victims scenarios in wired network only. We do not present the overhead comparison in wireless network because of the difficulty of creating classic starvation attack in wireless networks. Thus, wired network provides us a common base for studying the traffic behaviour and overhead comparison of induced and classic starvation attacks.

1) Single Victim: In single victim scenario, total number of messages sent by malicious client to exhaust 256 IP addresses and create starvation was 211 in case of *Induced DHCP Starvation* attack, while it was 422 in case of classic DHCP starvation attack. This is because in case of *Induced DHCP Starvation* attack, malicious client has to send only one fake ARP reply compared to two DHCP messages in classic DHCP starvation attack.

2) Multiple Victims: Similar analysis was performed for multiple victims scenario too. The total number of messages sent by malicious client to exhaust 256 IP addresses was 252 in *Induced DHCP Starvation* attack and 504 in case of classic DHCP starvation attack.

We can notice that *Induced DHCP Starvation* attack is light weight to create as compared to classic starvation attack from a malicious client's perspective.

V. DHCP Starvation Attack in IPv6 Networks

IPv6 facilitates IP address configuration in two modes as *stateless* and *stateful* address autoconfiguration. *StateLess* Address AutoConfiguration (SLAAC) is based on the address prefixes advertised by IPv6 enabled routers in Router Advertisement (RA) messages. IPv6 clients use these prefixes to configure their interfaces with site-local unicast IPv6 addresses which are equivalent to private IP addresses $(10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16)$ in IPv4 networks. On the other hand, *stateful* address autoconfiguration is based on protocols such as DHCPv6 to obtain IPv6 address and other network configuration parameters. *Stateful* address autoconfiguration requires presence of one or more DHCP servers on LAN which can allot site-local unicast IPv6 addresses to clients. *Stateful* address autoconfiguration is used either if an IPv6 enabled client receives RA messages with no prefix options or if there are no routers present on the local link. Even if routers advertise prefixes in their RA messages, DHCPv6 servers are still required in a LAN because prefixes in RA messages help IPv6 enabled clients to configure their interfaces with IPv6 addresses only. Other network configuration parameters such as DNS server's IP address, Subnet Prefix Length and Default Gateway's IP address can be configured using DHCPv6 only. Thus, for an IPv6 client to communicate with colleague devices, DHCPv6 server is an essential entity.

IPv6 allows various types of unicast IPv6 addresses that can be configured on an interface at a time. In this paper, we consider only two of them - *link-local unicast IPv6 addresses* and *site-local unicast IPv6 addresses*. For the details of different types of IPv6 addreses please consult [10]. Link-local unicast IPv6 addresses are equivalent to IPv4 addresses that are automatically configured on the interfaces without intervention of a DHCP server. For example, windows based operating systems configure a client's interfaces automatically with IPv4 addresses which are selected from the range $169.254.0.0/16$. Similarly, an IPv6 enabled client can configure its interfaces automatically with link-local unicast IPv6 addresses. The scope of these addresses is the local link and an IPv6 enabled router does not forward link-local traffic beyond the link. This address is always automatically configured even in the absence of other unicast IPv6 addresses. On the other hand, as discussed earlier, site-local unicast IPv6 addresses are equivalent to private IP addresses in IPv4 networks. The scope of these addresses is the site. A site can be an organisational network or a part of it. Site-local unicast IP addresses can not be reached from internet and an IPv6 router must not forward site-local traffic outside the site scope. Site-local unicast IPv6 addresses can not be configured automatically. These addresses are derived either with the help of prefixes advertised by IPv6 routers (*Stateless* address autoconfiguration) or using DHCPv6 (*Stateful* address autoconfiguration).

Similar to ARP in case of IPv4 networks, IPv6 also facilitates a Neighbour Discovery Protocol (NDP) that is used to resolve an IP address into corresponding MAC address. Like IPv4 clients, an IPv6 enabled client also performs a

precautionary check to verify the uniqueness of an IPv6 address that it is going to use for communication purpose. This precautionary check is known as Duplicate Address Detection (DAD). IPv6 clients perform this DAD check independent of the methods (*stateless* and *stateful*) used for address autoconfiguration. To perform DAD, a client sends a Neighbour Solicitation (NS) message with the target address set to IP address for which this check is being performed. If a Neighbour Advertisement (NA) message sent in response to the NS message is received, the client interprets that the tentative IP address is already being used by some other client and thus, starts trying to configure its interface with another IP address. The sequence of messages exchanged during execution of starvation attack is as follows:

1) Derivation of Link-local Unicast IP Address by Victim Client: As soon as victim client joins a network, it automatically derives link-local unicast IP address that can be used to communicate with on-link devices. In case the victim client needs DHCPv6 to obtain site-local unicast IPv6 address, it must configure its interface with link-local unicast IP address first because using this address only, it can communicate with a DHCPv6 server. Thus to obtain a site-local unicast IP address using *stateful* address autoconfiguration, victim client must possess a link-local unicast IP address on its interface.
2) DAD Check for Link-local Unicast IP Address by Victim Client: Before configuring this address on its interface, victim client sends a NS message with Target Address set to derived link-local unicast address to check if any other client is already using this IP address. The source and destination IP address of this NS message is set to unspecified address (::) and solicited node multicast address respectively.
3) Spoofed NA Message by Malicious Client: As soon as malicious client receives NS message sent by victim client in previous step, it sends a spoofed NA message whose source IP address is set to link-local IP address that is derived by victim client in previous step. The destination IP of NA message is set to link-local scope all-nodes multicast IP address ($ff02::1$). This address is equivalent to broadcast address in IPv4 networks.

Once victim client receives the spoofed NA message, it interprets this message as some other client is already using the derived link-local unicast IP address. In order to avoid IP address conflict, victim client derives new link-local unicast address by re-initiating the address autoconfiguration process. The malicious client repeats the steps discussed above to effectively deny the victim client from ever acquiring an IP address, resulting into starvation. Since victim client can not configure its interface with a link-local unicast IPv6 address, it can not communicate with DHCPv6 server(s) present on LAN. As a result, when DHCP Starvation attack is launched, victim client can not receive a site-local unicast IP address too using *stateful* address autoconfiguration. However, in our experiments, we observed that even in the presence of attack, though victim client can not configure a link-local unicast IP address, it can still configure its interface with site-local unicast IP address if it uses *stateless* address autoconfiguration. The reason is IPv6 allows routers to send unsolicited RA messages periodically without waiting to receive Router Solicitation (RS) messages from IPv6 clients. As a result, even when victim client can not send RS message due to absence of a link-local unicast IP address on its interface, it can still receive RA messages advertised by IPv6 routers periodically. These periodic RA messages are sent to link-local scope all-nodes multicast IP address. Thus, all IPv6 clients receive these messages even if they do not send corresponding RS message. After receiving RA message, victim client extracts prefix information from it and derives a site-local unicast IP address. Before configuring its interface with this IP address, victim client performs DAD check for derived site-local unicast IP address by sending a NS message as discussed earlier. This gives a chance to malicious client to launch the attack by sending spoofed corresponding NA message. This message forces victim client to interpret that the address is already in use. As a result, victim client becomes unable to configure its interface with a site-local unicast IPv6 address using *stateless* address autoconfiguration.

A. Experimental Results

We created a real IPv6 network testbed to evaluate the effectiveness of proposed attack in IPv6 networks. This testbed was somewhat similar to the testbed of IPv4 network shown in Figure 3 with total of 10 clients. The only difference was Layer 2 switch used earlier was replaced by a HP Aruba 2920 Layer 3 Managed switch and the DHCP server was replaced by Dibbler [11] DHCPv6 server. In our testbed, we designated 2 clients as victim and 1 client as malicious. The victim clients and malicious client were running Windows 7 (service pack 1) and Ubuntu 14.04 respectively. We conducted our experiments in two phases. In first phase, we evaluated the attack when clients configure IPv6 addresses through *stateful* address autoconfiguration using a DHCPv6 server. In second phase, we evaluated the attack when clients configure IPv6 addresses through *stateless* address autoconfiguration using prefixes advertised by the managed switch. These two phases are discussed below:

1) Attack's Effectiveness against Stateful Address Autoconfiguration: In this phase, we disabled the IPv6 protocol on the managed switch so that clients could obtain an IP address through *stateful* address autoconfiguration using DHCPv6 only. The malicious client was already connected to the network and continuously sniffing for NS messages sent by victim clients for DAD checking. As soon as victim clients joined the network, they sent NS messages in order to verify uniqueness of the automatically configured link-local unicast IPv6 addresses. Once malicious client sniffed these messages, it immediately sent spoofed corresponding NA messages. Thus, victim clients could not obtain a link-local unicast IPv6 address. This further made them unable to communicate with DHCPv6 server due to absence of a link-local unicast IPv6 address on their interface. In this phase, we

ran the experiment for about 1000 seconds. In this time period, we observed that each victim client performed exactly 10 trials to configure its interface with a link-local unicast IPv6 address. Thus, it sends 10 NS messages for DAD checking with a gap of 0.5 seconds in between two consecutive messages. Overall, after 5 seconds, victim client finally gave up without acquiring any link-local unicast IP address. Thus, proposed attack is effective in creating starvation scenario when DHCPv6 is used for address autoconfiguration.

2) Attack's Effectiveness against Stateless Address Autoconfiguration: In this phase, we disconnected DHCPv6 server from the switch so that clients could obtain an IP address through *stateless* address autoconfiguration using prefixes advertised by managed switch only. As discussed earlier, IPv6 allows routers to send unsolicited RA messages even in the absence of corresponding RS messages. Due to this reason, victim clients were able to obtain site-local unicast IP address even when they were unable to configure their interface with link-local unicast IP address. Thus, in this phase, malicious client needs to send spoofed NA messages in response to NS messages sent for DAD checking of both link-local as well as site-local unicast IPv6 addresses. We ran the experiment for about 1000 seconds and observed that each victim client first performed 2 trials to configure its interface with a link-local unicast IPv6 address and then 2 trials for configuring a site-local unicast IP address. After this, victim client gave up without acquiring any site-local unicast IP address. However, it performed 6 more trials to acquire a link-local unicast IP address. After a period of about 3.5 seconds, victim client finally stopped performing any more trial to acquire either link-local or site-local unicast IP address. Thus, proposed attack is effective in creating starvation scenario when SLAAC is used for address autoconfiguration.

VI. Known Detection/Mitigation Techniques

Several techniques are proposed in the literature to detect and mitigate DHCP starvation attacks. In this section, we describe some popular countermeasures and their capability to detect classic DHCP starvation attacks, attack proposed in [4] and *Induced DHCP Starvation* attack.

A. Cryptographic and Certificate based Techniques

DHCP starvation attacks are possible because of no in-built authentication in DHCP protocol. Various cryptographic techniques [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23] are proposed to enforce authentication in DHCP to mitigate starvation attacks. Since these techniques can prevent identity spoofing based attacks, all types of DHCP starvation attacks can be mitigated using these techniques. However, these techniques have several issues and limitations. Firstly, all of these add complexity for their implementation. Some require intervention of network administrator to distribute the certificates and shared keys beforehand, using an out of band mechanism. The requirement of human intervention straightaway conflicts the purpose for which DHCP came into existence. Another issue is the addition of significant complexity and traffic load due to the involvement of third party modules like Authentication Server, Confirmation Server and Detection Server. Also, the authors in [24] pointed that the maximum DHCP message options length is too short to encapsulate a large amount of data like a Digital certificate. Due to such complex and non deployable infrastructural requirement, the cryptographic techniques are rarely implemented in the networks.

B. Security Features in Switches

In this subsection, we discuss the capability of few popular security features available in modern network switches to detect and mitigate various DHCP starvation attacks. We assume a network setup similar to the one shown in Figure 3 to evaluate these security features since this is the most common LAN architecture followed in different institutions and organizations.

1. Port Security [25]: In [22], authors claimed that DHCP starvation attacks can be mitigated by limiting the number of permissible MAC addresses per port in a network switch. Port Security is one such measure that is implemented in modern switches. Port security feature limits the number of MAC addresses allowed per port of a network switch. Once this limit is reached, the port is automatically disabled and an SNMP trap is sent. Since a malicious client generates multiple random MAC addresses in order to launch classic DHCP starvation attacks[6], port security can immediately disable the port as soon as more than one MAC is observed at the port to which malicious client is connected. Thus, port security can easily mitigate Type-1 attack. However, Type-2 attack can not be mitigated using port security feature because malicious client uses its real MAC address in Ethernet header of all DHCPDISCOVER messages and random MAC address is present only in DHCP header. Since port security does not analyze application layer protocols' headers, it is unable to mitigate Type-2 classic DHCP starvation attack. In case of attack proposed in [4], the spoofed ICMP replies traverse the switch in order to reach DHCP server. However, since the malicious client is communicating with DHCP server using its own MAC address, port security can see only one MAC address on the port and thus, the attack can not be detected. In case of *Induced DHCP Starvation* attack, spoofed ARP replies remain local to AP and does not traverse network switch. Thus, *Induced DHCP Starvation* can not be detected using this security feature.

2. DHCP Snooping [26]: This feature available with network switches can notice the difference in MAC address present in the Ethernet header and CHADDR field of a DHCPDISCOVER message. Thus, it can easily mitigate Type-2 classic DHCP starvation attacks. However, since this feature does not limit the number of permissible MAC addresses per port in a network switch, it can not mitigate Type-1 DHCP starvation attacks. In *Induced DHCP Starvation* attack and

[6]Though it is difficult, we assume that malicious client precedes an association with AP for each spoofed address to launch the attack in a wireless network.

the attack proposed in [4], only a spoofed ARP/ICMP reply is sent in response and there are no difference in CHADDR field MAC address and Ethernet MAC address. This security feature does not validate ARP/ICMP messages hence it can't detect *Induced DHCP Starvation* attack and attack proposed in [4] as well.

3. Dynamic ARP Inspection (DAI) [27] with DHCP Snooping: It determines the validity of an ARP message based on valid IP-MAC bindings stored in a trusted DHCP Snooping binding database. Since Type-1 and Type-2 starvation attacks do not require sending of spoofed ARP/ICMP replies, they can not be detected using DAI. In case of attack proposed in [4], the fake ICMP replies sent by malicious client have invalid IP-MAC bindings that are not stored in DHCP Snooping database. Moreover, since these spoofed replies traverse the network switch in order to reach DHCP server, DAI can detect this attack unless DHCP server itself is configured on a wireless device and connected to same AP. As a result, DAI enabled switch prevents such fake ICMP replies from being forwarded. Thus, this feature mitigates attack proposed in [4]. In case of *Induced DHCP Starvation* attack, the fake ARP replies sent by malicious client does not traverse the switch and remain local to AP. As a result, the *Induced DHCP Starvation* attack is not detected using this feature also.

C. Using DHCP Relay Agent Information Option

In [28], authors used the DHCP Relay Agent Information option to mitigate DHCP starvation attacks. The relay agent includes the port number and the switch ID in the Relay Agent Information option before forwarding the DHCP messages to the DHCP server. Using this information, the DHCP server verifies that the client has not exceeded the maximum allowed number of assigned IP addresses, thereby, preventing DHCP starvation attack. Thus, it can detect Type-1 and Type-2 DHCP starvation attacks since they involve sending of large number of DHCPDISCOVER messages. However, *Induced DHCP Starvation* attack and attack proposed in [4] need not send any DHCP message at all, so they go undetected. Moreover during attack, since it is the victim client that sends multiple DHCPDISCOVER messages while trying again and again to acquire an IP address, this scheme interprets victim client itself as the malicious one.

D. DHCP Traffic Threshold based Mechanism

In [29], authors proposed a scheme to count the number of DHCPREQUEST packets within a fixed period of time. If this count exceeds a predetermined threshold, the detection system will alert the network administrator about the ongoing DHCP starvation attack. In classic DHCP starvation attacks, a malicious client sends a flood of DHCP messages to complete several DORA processes in order to exhaust address pool. Thus, this detection scheme can easily detect Type-1 and Type-2 DHCP starvation attacks. Induced DHCP starvation attack and the attack proposed in [4] also causes a victim client to complete several DORA processes due to which DHCPREQUEST messages are generated. However, the number of DHCPREQUEST messages generated during these attacks is quite less as compared to classic DHCP starvation attack. As a result, it is difficult to detect induced DHCP starvation attack and the attack proposed in [4] using this detction scheme.

E. Fair Allocation based Mitigation Technique

Another technique proposed in [30] estimates the number of contending users at each port of a switch and thereafter proposed a way to fairly allocate IP addresses for each port. However, malicious client can simply connect to AP and consume fair share of IP addresses allotted at the port of switch to which AP is attached, thereby, causing starvation for upcoming legitimate clients which are going to connect with AP the malicious client is already connected with. Thus, it can not mitigate any of the DHCP starvation attacks.

Table III shows a comparison of detection of DHCP starvation attacks using different detection techniques. We assume a network setup similar to one shown in Figure 3.

VII. Proposed Detection Technique

In this section, we describe a method to detect the proposed *Induced DHCP Starvation* attack, the attack discussed in [4] and DHCPv6 starvation attack in IPv6 networks. We describe a statistical abnormality measurement technique to detect these attacks. In particular, we treat the normal behavior of DHCP operation as a distribution comprised of various events. This detection method is motivated by the fact that various DHCP messages have strong correlation between them and a similar observation made in the context of SIP [31]. For example every DHCPDISCOVER message is most likely followed by DHCPOFFER message and subsequently with DHCPREQUEST and DHCPACK messages. This balance will be disturbed either due to increased number of DECLINE messages in case of *Induced DHCP Starvation* attack or due to absence of all types of DHCP messages except DHCPDISCOVER in case of attack proposed in [4]. We exploit this change in observation to detect the proposed starvation attacks.

Our detection method has two phases of operation as training phase and testing phase. In the training phase we collect and create normal behavior profile of DHCP operation (distribution graph) and in the testing phase, we compare the profile of current DHCP operation with previously generated profile to detect starvation attacks. For this comparison, we opted for methods which compute distances between two probability distributions. Some of the popular distance metrics are Bhattacharyya Distance [32], Total Variation Distance [33], Mahalanobis Distance [34], Kullback-Leibler Divergence [35] and Hellinger Distance [36] which are given in Equation 1 2, 3, 4 and 5 respectively. In these equations $\mu*$ is the mean of corresponding vectors, $\sigma*$ is the standard deviation of corresponding vectors and C^{-1} is the co-variance matrix generated from training intervals (each interval is a probability distribution). Here $\mathcal{P}$ and $\mathcal{Q}$ are N dimensional vectors with each component P_* of the vector representing the probability of an attribute of vector $\mathcal{P}$.

TABLE III
DETECTION/MITIGATION OF DHCP STARVATION ATTACKS USING DIFFERENT DETECTION TECHNIQUES

Type	Cryptography based	Port Security	DHCP Snooping	DAI	Using Relay Agent [28]	DHCP Traffic Threshold [29]	Fair Allocation [30]
Type I	Yes	Yes	No	No	Yes	Yes	No
Type II	Yes	No	Yes	No	Yes	Yes	No
Attack proposed in [4]	Yes	No	No	Yes	No	No	No
Induced DHCP Starvation	Yes	No	No	No	No	No	No

$$d_B(\mathcal{P},\mathcal{Q}) = \frac{1}{4}ln(\frac{1}{4}(\frac{\sigma_p^2}{\sigma_q^2}+\frac{\sigma_q^2}{\sigma_p^2})) + \frac{1}{4}(\frac{(\mu_p-\mu_q)^2}{\sigma_p^2+\sigma_q^2}) \quad (1)$$

$$d_T(\mathcal{P},\mathcal{Q}) = \frac{1}{2} \parallel \mathcal{P}-\mathcal{Q} \parallel \quad (2)$$

$$d_M(\mathcal{P},\mathcal{Q}) = \sqrt{(\mathcal{Q}-\mu_p)^t C^{-1}(\mathcal{Q}-\mu_p)} \quad (3)$$

$$d_K(\mathcal{P},\mathcal{Q}) = \sum_i (P_i \ log \ \frac{P_i}{Q_i}) \quad (4)$$

$$d_H = (\frac{1}{2}\Sigma_{i=1}^{N}(\sqrt{P_i}-\sqrt{Q_i})^2)^{\frac{1}{2}} \quad (5)$$

Among these distance metrics, we chose Hellinger Distance between the training and testing probability distributions to detect *Induced DHCP Starvation* attack. In the next subsection, we discuss few theoretical reasons due to which we chose Hellinger distance over others. In subsection VII-C, we present some experimental details also to prove that Hellinger Distance is the most appropriate distance metric for our purpose.

A. Choosing Appropriate Distance Metric

Due to following theoretical reasons, we chose Hellinger Distance to detect induced DHCP starvation attack and the attack proposed in [4].

- Lightweight Computation: Unlike Mahalnobis distance measure, measuring Hellinger Distance between two probability distributions does not involve computationally expensive calculations like matrix inverse or covariance. Thus, Hellinger Distance is a lightweight alternative for IDS/IPSs to detect and mitigate proposed starvation attack.
- Natural Lower and Upper Bounds: It is worth noting that the value of d_H will be always in between 0 and 1 with 0 representing perfect similarity and 1 representing maximum dissimilarity between $\mathcal{P}$ and $\mathcal{Q}$ therefore Hellinger distance has natural lower and upper bounds of 0 and 1 which other distance measuring methods do not have.
- Yielding Finite Distance Value: Unlike Kullback-Leibler Divergence which is defined only if $Q_i = 0 \Rightarrow P_i = 0$, $\forall i$, Hellinger Distance does not require any such dependency between two probability distributions. When the attack proposed in [4] is launched, it causes the probability (Q) of occurrence of DHCPOFFER, DHCPREQUEST, DHCPACK and DHCPDECLINE messages to become 0. In that case, Kullback-Leibler Divergence gives infinite value while on the other hand, Hellinger Distance yields appropriate finite distance value ranging from 0 to 1.

B. Adapting Hellinger Distance for DHCP Starvation Detection

The described detection system has following two components.

- *Probabilistic Distribution of Training Data*: In order to create a profile of normal DHCP operation we observe and generate a distribution profile using DHCP messages over a period of n observation intervals each of duration ΔT. The profile generated in the training phase $\mathcal{P}$ has 5 attributes as bellow. $\mathcal{P}$ = $\{P_{DISC}, P_{OFFER}, P_{REQ}, P_{ACK}, P_{DECL}\}$ where $P_{DISC}, P_{OFFER}, P_{REQ}, P_{ACK}, P_{DECL}$ represent the probability of occurrence of DISCOVER, OFFER, REQUEST, ACKNOWLEDGEMENT and DECLINE DHCP messages respectively. Probability of a DHCP message type P_i is estimated using Equation 6 where N_i denote the count of number of events of type i during the entire training period ($n * \Delta T$) and N_{total} represent the count of events of all 5 types in the same duration.

$$P_i = \frac{N_i}{N_{total}} \quad (6)$$

- *Probabilistic Distribution of Testing Data*: Once the system is trained and $\mathcal{P}$ is generated we use it to detect starvation attacks from $(n+1)^{th}$ interval of duration ΔT. We generate a probability distribution $\mathcal{Q}$ every ΔT interval using Equation 6. In this case N_i represent the count of events of type i in the interval of consideration and N_{total} represent sum of all 5 types of events in the same interval. $\mathcal{Q}$ generated thus is compared with $\mathcal{P}$ using Hellinger distance. If the distance between the two distributions is greater than a threshold δ is detected as starvation attack.

C. Experimental Reasons to Choose Hellinger Distance as Distance Metric

We performed few experiments also which proved Hellinger Distance is a better metric as compared to other methods

to compute distance between two probability distributions. We considered each of the above discussed distance metics to detect proposed starvation attack and the attack proposed in [4]. Table IV shows different distances measured between two probability distributions using different methods. Based on these readings, we omitted Mahalanobis Distance and Kullback-Leibler Divergence because the former one did not have any natural bound and the latter one yielded undefined distance value if probability of occurrence of one or more events in testing period becomes 0. After this omission, we chose the distance value that is maximum among all 18 values measured by testing 18 normal intervals. Finally, we selected that method which was reflecting the largest difference between maximum distance and minimum distance measured while testing normal intervals and distance measured while testing proposed attack's interval. For example, as we can see in Table IV, the maximum distance measured using Bhattacharyya Distance while testing normal intervals was 0.002132 and distance measured while testing proposed attack's interval was 0.0936. So, Bhattacharyya Distance reflected a difference of 0.091468. Similarly, Total Variation Distance and Hellinger Distance reflected differences of 0.129128 and 0.280757 respectively. Since Hellinger Distance reflects largest difference between distances measured by testing normal and attack interval, we selected it as the most appropriate distance metric. This selection criteria will help network administrators to define a threshold distance easily. Moreover, it reduces the chances of false positives too.

TABLE IV
DISTANCES MEASURED BETWEEN TWO PROBABILITY DISTRIBUTIONS USING DIFFERENT METHODS

Method	Minimum distance (Normal)	Maximum distance (Normal)	Distance measured (Attack)	Distance measured (Attack [4])
Bhattacharyya Distance	0.0000197	0.002132	0.0936	0.6859
Total Variation Distance	0.00615	0.059072	0.1882	0.7464
Mahalanobis Distance	0.2432	0.5922	1.9912	7.7056
Kullback-Leibler Divergence	0.0000758	0.008459	0.1872	Undefined
Hellinger Distance	0.004381	0.046143	0.3269	0.7045

D. Experimental Evaluation of DHCP Starvation Detection in IPv4 Networks

To evaluate the detection capability of our proposed technique, we collected 3 days of normal DHCP traffic from our departmental network. This network has 43 clients connecting to the DHCP server for IP address. First to establish the fact that there is a correlation between the various message types of DHCP we plot a graph of all 5 chosen DHCP messages timings as shown in Figure 12 from the data of normal DHCP operation intervals. As we can see from this figure, 4 out of 5 message types (except DECLINE) are overlapping with each other confirming that they have strong correlation between them. However there were no DECLINE messages seen in the entire training dataset hence its line overlaps with X axis. We can also observe that, for few time intervals, all five messages are zero. This is because no new host joined the network in those time intervals and these intervals correspond to after office hours of the day.

For detection purpose, we used the first two days' traffic for training purpose i.e to create distribution profile $\mathcal{P}$. We set the interval duration ΔT to 30 minutes for our experiments. Thus, there were a total of 96 such intervals from two days of training data. We generated the probability distribution graph using these 96 interval data which represent the normal system behavior of DHCP. Figure 13 shows the distribution graph generated from this data. As observed previously all 4 types of messages appear in almost equal number and hence their probabilities are also similar (around 0.25) and DECLINE messages do not appear and its probability being 0.

We used the third day's normal DHCP traffic for testing purpose. Similar to training phase, we used intervals of $\Delta T = 30$ minutes in testing phase too. As a result, there were a total of 48 such intervals from third day's normal DHCP traffic. As mentioned previously we measure the distance between probability distributions of normal and testing interval. In the third days's traffic also we had few intervals where no messages were seen which makes probability estimation difficult. In order to address this case we considered only those intervals in which the total number of occurrences of all five types of events is at least equal to half of mean number of occurrences of all five event types from the training phase. Using this as a filter we report the calculated Hellinger distance between testing interval and training profile. There were 18 such intervals from the third day's data meeting this criteria. The estimated Hellinger distances and the clock time of such 18 intervals are shown in Table V. During the time intervals $4:00-4:30pm$ and $4:30-5:00pm$, we launched proposed *Induced DHCP Starvation* attack and the attack proposed in [4] respectively. The Hellinger distances calculated between these intervals and training profile are also shown in Table V with red colors. We can clearly notice that the estimated Hellinger distances in case of starvation scenarios is quite high as compared to the Hellinger distances estimated in case of normal DHCP traffic. The sample probability distributions generated from normal testing interval, in case of *Induced DHCP Starvation* attack and attack proposed in [4] are shown in Figure 14, Figure 15 and Figure 16 respectively. As again we can see that Figure 14 is more similar to the one generated in training phase and results in smaller distance value.

It is worth noting that evading this detection is possible only when an attacker generates DHCP messages such that distribution profile is similar to the one generated in training period. As we noticed DECLINE messages are not seen or very rarely seen (when someone inadvertently uses an address in the pool without knowledge of DHCP server) but in *Induced DHCP Starvation* each attempt generates a DECLINE message. These

TABLE V
DETECTION OF NORMAL AND STARVATION SCENARIOS

Scenario	Interval	Hellinger Distance	Detection result	Detection Rate
Normal	4:00pm-4:30pm	0.0044	Normal	100%
	4:30pm-5:00pm	0.0333	Normal	100%
	5:00pm-5:30pm	0.0192	Normal	100%
	5:30pm-6:00pm	0.0044	Normal	100%
	6:00pm-6:30pm	0.0257	Normal	100%
	9:00am-9:30am	0.0044	Normal	100%
	9:30am-10:00am	0.0044	Normal	100%
	10:00am-10:30am	0.0044	Normal	100%
	10:30am-11:00am	0.0388	Normal	100%
	11:00am-11:30am	0.0044	Normal	100%
	11:30am-12:00pm	0.0044	Normal	100%
	12:00pm-12:30pm	0.0181	Normal	100%
	12:30pm-1:00pm	0.0044	Normal	100%
	1:00pm-1:30pm	0.0044	Normal	100%
	1:30pm-2:00pm	0.0044	Normal	100%
	2:30pm-3:00pm	0.0461	Normal	100%
	3:00pm-3:30pm	0.0167	Normal	100%
	3:30pm-4:00pm	0.0229	Normal	100%
Induced DHCP Starvation	4:00pm-4:30pm	0.3269	Anomaly	100%
Attack proposed in [4]	4:30pm-5:00pm	0.7045	Anomaly	100%

messages not only make the DECLINE message probability go higher but also causes other four message probability to reduce and this makes evading detection difficult.

E. Experimental Evaluation of DHCP Starvation Detection in IPv6 Networks

To evaluate the detection capability of proposed scheme in IPv6 networks, we first collected 3 days of DHCPv6 traffic from the testbed of 10 clients we created with IPv6 enabled. Form the same network, we then collected 3 more days of IPv6 traffic where the clients used SLAAC to obtain site-local unicast IPv6 adderss. From first set of 3 days traffic, we used first two days' traffic to train proposed detection scheme with DHCPv6 traffic. Similarly, from second set of 3 days traffic, we used first two days' traffic to train proposed detection scheme with SLAAC traffic. During both the training periods, we set the interval duration ΔT to 30 minutes for our experiments. Thus, there were a total of 96 such intervals from each of the training sets. We generated the probability distribution graphs using these 96 interval data which represent the normal system behavior of DHCP. Figure 17 and Figure 18 show the distribution graph for DHCPv6 and SLAAC traffic respectively. Similar to IPv4 networks, we considered five different events for IPv6 networks too. These were: 1) NS message to perform DAD check for link-local unicast IP address, 2) DHCP SOLICIT message (similar to DHCPDISCVOER in IPv4 networks), 3) DHCP ADVERTISE message (similar to DHCPOFFER in IPv4 networks), 4) DHCP REQUEST message (similar to DHCPREQUEST in IPv4 networks) and 5) DHCP REPLY message (similar to DHCPACK in IPv4 networks).

From both sets of 3 days traffic, we used the third day's traffic for testing purpose. Similar to training phase, we used intervals of $\Delta T = 30$ minutes in testing phase too. Thus, there were a total of 48 such intervals to be tested. Since in third days's traffic of both the data sets, we had few intervals where no messages were seen so we considered only those intervals in which the total number of occurrences of all five types of events is at least equal to half of mean number of occurrences of all five event types from the training phase. After adapting this filter, we finally obtain 6 eligible intervals from the third day's data of both the data sets.

TABLE VI
DETECTION OF NORMAL AND STARVATION SCENARIOS WHILE USING DHCPV6

Scenario	Interval	Hellinger Distance	Detection result	Detection Rate
Normal	9:00pm-9:30am	0.0630	Normal	100%
	9:30am-10:00am	0.1323	Normal	100%
	10:00am-10:30am	0.0443	Normal	100%
	10:30am-11:00am	0.1323	Normal	100%
	11:00am-11:30am	0.0443	Normal	100%
	11:30am-12:00pm	0.0630	Normal	100%
DHCPv6 Starvation	12:00pm-12:30pm	0.7534	Anomaly	100%

For DHCPv6 traffic, the estimated Hellinger distances and the clock time of such 6 intervals are shown in Table VI. During the time interval $12:00-12:30pm$, we launched proposed induced starvation attack. The Hellinger distance calculated between attack interval and training profile is also shown in Table VI with red color. We can clearly notice that the estimated Hellinger distance in case of starvation scenario are quite high as compared to the Hellinger distances estimated in case of normal DHCPv6 traffic. The sample probability distributions generated from testing normal interval and in case of starvation attack are shown in Figure 19 and Figure 20 respectively. Once again, we can see that Figure 19 is more similar to the one generated in training phase with DHCPv6 traffic and thus, results in smaller distance value.

For SLAAC traffic, the estimated Hellinger distances and the clock time of such 6 intervals are shown in Table VII. During the time interval $04:30-5:00pm$, we launched proposed induced starvation attack. The Hellinger distance calculated between attack interval and training profile is also shown in Table VII with red color. We can clearly notice that the estimated Hellinger distances in case of starvation scenario is quite high as compared to the Hellinger distances estimated in case of normal SLAAC traffic. The sample probability distributions generated from testing normal interval and in case of induced starvation attack are shown in Figure 21 and Figure 22 respectively. Once again, we can see that Figure 21 is more similar to the one generated in training phase with SLAAC traffic and thus, results in smaller distance value.

F. Other Possible Countermeasures

As a first line of defense, one can also employ the below techniques to safeguard against *Induced DHCP Starvation* attack.

1. Threshold on Number of IP Address Conflict: In *Induced DHCP Starvation* attack a client fails to acquire IP

TABLE VII
DETECTION OF NORMAL AND STARVATION SCENARIOS WHILE USING SLAAC

Scenario	Interval	Hellinger Distance	Detection result	Detection Rate
Normal	1:00pm-1:30pm	0	Normal	100%
	1:30pm-2:00pm	0.1138	Normal	100%
	2:00pm-2:30pm	0.3660	Normal	100%
	2:30pm-3:00pm	0.0649	Normal	100%
	3:00pm-3:30pm	0.0587	Normal	100%
	3:30pm-4:00pm	0	Normal	100%
DHCPv6 Starvation	04:30pm-5:00pm	0.7071	Anomaly	100%

addresses repeatedly. A simple way for a client to protect itself from repeated failures in acquiring an IP address is to monitor the rate of IP conflicts. Beyond a threshold number of failures, it can generate an alert to user. It can also take note of malicious client's MAC address in successive fake ARP replies as malicious client uses its own MAC address to make fake IP address ownership.

2. At Access Points: Access Point can forward all ARP replies to a DAI (Dynamic ARP Inspection) enabled network switch. This switch can then validate the received IP-MAC binding with the bindings available in DHCP snooping trusted database to detect *Induced DHCP Starvation*.

VIII. CONCLUSION

DHCP starvation attack is a popular DoS attack to prevent clients from acquiring an IP address. In this paper, we highlighted few experimental difficulties in generating these attacks in wireless networks. Subsequently, we demonstrated an easier to create and more stealth attack in wireless networks using ARP spoofing. Comparison with state of the art detection and mitigation methods is also made, highlighting state of the art detection methods fails to detect it. We also described the scenario with IPv6 where structurally similar attacks are possible causing the same effect of *Induced DHCP Starvation* attack. We also proposed an anomaly based detection method to detect *Induced DHCP Starvation* attack and similar attacks. We consider that the attack presented in the paper will motivate research community in the area for future research in securing DHCP and DHCPv6.

REFERENCES

[1] Gobbler. http://gobbler.sourceforge.net/.
[2] DHCPIG. https://github.com/kamorin/dhcpig.
[3] R. Droms. Dynamic Host Configuration Protocol. RFC 2131, 1997.
[4] N. Tripathi and N. Hubballi. Exploiting DHCP Server-side IP Address Conflict Detection: A DHCP Starvation Attack. In *International Conference on Advanced Networks and Telecommunications Systems*, pages 19–21, 2015.
[5] X. Xing, E. Shakshuki, D. Benoit, and T. Sheltami. Security Analysis and Authentication Improvement for IEEE 802.11i Specification. In *IEEE Global Telecommunications Conference*, pages 1–5, 2008.
[6] D. C. Plummer. An Ethernet Address Resolution Protocol. RFC826, 1982.
[7] S. Cheshire. IPv4 Address Conflict Detection. RFC5227, 2008.
[8] P. Mockapetris. Domain Names - Implementations and Specifications. RFC 1035, 1987.
[9] Y. Musashi, M. Kumagai, S. Kubota, and K. Sugitani. Detection of kaminsky dns cache poisoning attack. In *ICINIS '11: Proceedings of the 2011 4th International Conference on Intelligent Networks and Intelligent Systems*, pages 121–124, 2011.
[10] S. Deering and R. Hinden. Internet Protocol, Version 6. RFC 2460, 1998.
[11] DIBBLER. http://klub.com.pl/dhcpv6/.
[12] B. Issac. Secure ARP and Secure DHCP Protocols to Mitigate Security Attacks. *International Journal of Network Security*, 8(2):107–118, 2009.
[13] R. Droms and W. Arbaugh. Authentication for DHCP messages. RFC3118, 2001.
[14] Y. I. Jerschow, C. Lochert, B. Scheuermann, and M. Mauve. CLL: A Cryptographic Link Layer for Local Area Networks. In *International Conference on Security and Cryptography for Networks*, pages 21–38, 2008.
[15] G. Glazer, C. Hussey, and R. Shea. Certificate-based Authentication for DHCP, 2003.
[16] J. Demerjian and A. Serhrouchni. DHCP Authentication using Certificates. In *Security and Protection in Information Processing Systems*, pages 456–472. 2004.
[17] K. de Graaf, J. Liddy, P. Raison, J.C. Scano, and S. Wadhwa. Dynamic Host Configuration Protocol (DHCP) Authentication using Challenge Handshake Authentication Protocol (CHAP) Challenge, 2013. US Patent 8,555,347.
[18] H. Ju and J. Han. DHCP Message Authentication with an Effective Key Management. *World Academy of Science, Engineering and Technology*, 8:570–574, 2007.
[19] S. Shen, X. Lee, Z. Sun, and S. Jiang. Enhance IPv6 Dynamic Host Configuration with Cryptographically Generated Addresses. In *International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pages 487–490, 2011.
[20] Z. Su, H. Ma, X. Zhang, and B. Zhang. Secure DHCPv6 that uses RSA Authentication Integrated with Self-Certified Address. In *International Workshop on Cyberspace Safety and Security*, pages 39–44, 2011.
[21] S. Duangphasuk, S. Kungpisdan, and S. Hankla. Design and Implementation of Improved Security Protocols for DHCP using Digital Certificates. In *International Conference on Networks*, pages 287–292, 2011.
[22] S. Majumdar, D. Kulkarni, and C. V. Ravishankar. DHCP Origin Traceback. In *International Conference on Distributed Computing and Networking*, pages 394–406, 2011.
[23] T. Aura, M. Roe, and S.J. Murdoch. Securing Network Location Awareness with Authenticated DHCP. In *International Conference on Security and Privacy in Communications Networks and the Workshops*, pages 391–402, 2007.
[24] D.D. Dinu and M. Togan. DHCP Server Authentication using Digital Certificates. In *International Conference on Communications*, pages 1–6, 2014.
[25] Port Security. http://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst6500/ios/12-2sx/configuration/guide/book/port_sec.html.
[26] DHCP Snooping. http://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst6500/ios/12-2sx/configuration/guide/book/snoodhcp.html.
[27] Dynamic ARP Inspection. http://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst6500/ios/12-2sx/configuration/guide/book/dynarp.html.
[28] M. Patrick. DHCP Relay Agent Information Option. RFC 3046, 2001.
[29] T. OConnor. Detecting and Responding to Data Link Layer Attacks. 2010.
[30] H. Mukhtar, K. Salah, and Y. Iraqi. Mitigation of DHCP Starvation Attack. *Computers and Electrical Engineering*, 38(5):1115–1128, 2012.
[31] H. Sengar, H. Wang, D. Wijesekera, and S. Jajodia. Detecting voip floods using the hellinger distance. *IEEE Transactions on Parallel and Distributed Systems*, 19(6):794–805, 2008.
[32] A. Bhattacharyya. On A Measure of Divergence between Two Statistical Populations Defined by Their Probability Distribution. *Bulletin of the Calcutta Mathematical Society*, 35:99–110, 1943.
[33] Distances and Affinities Between Measures. www.stat.yale.edu/∼pollard/books/asymptopia/metrics.pdf.
[34] Mahalanobis Metric. https://www.cs.princeton.edu/courses/archive/fall08/cos436/duda/pr_mahal/m_metric.htm.
[35] Kullback Leibler Distance (KL). http://www.csse.monash.edu.au/∼lloyd/tildemml/kl/.
[36] X. Cao. Model Selection Based on Expected Squared Hellinger Distance, 2007. Ph.D Thesis, Colorado State University.

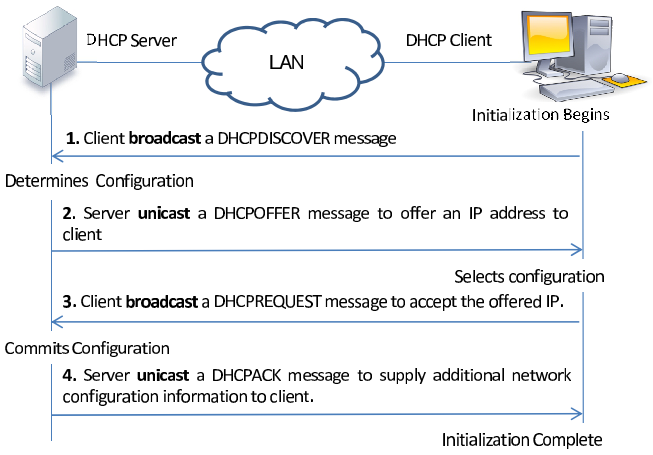


Fig. 1. Exchange of messages in DHCP operation

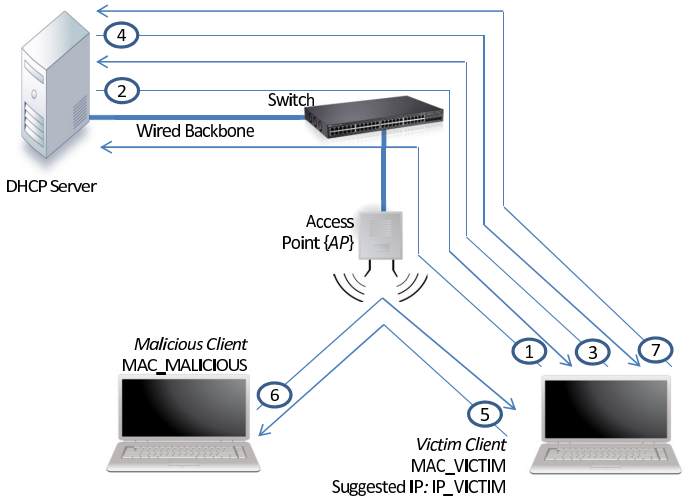


Fig. 3. Induced DHCP Starvation Attack

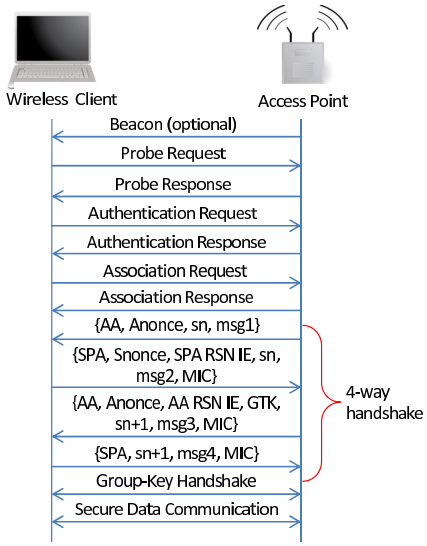


Fig. 2. Association of Client with AP

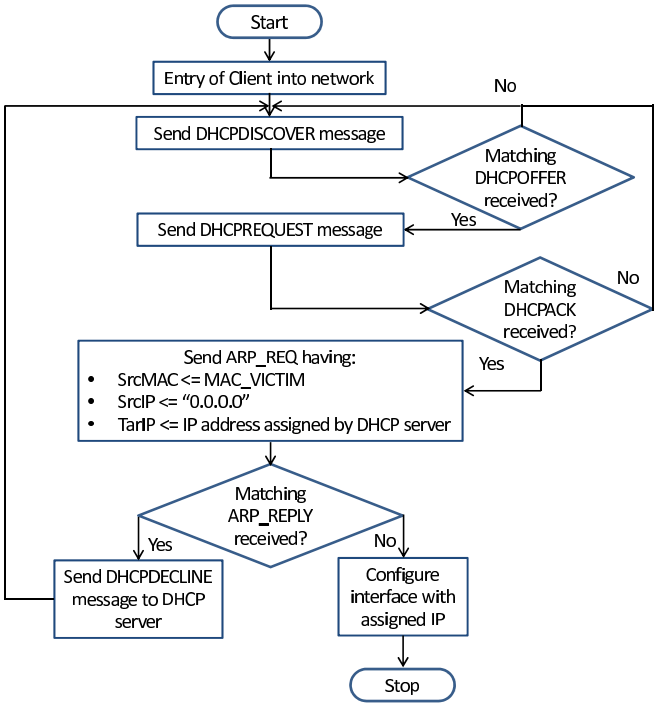


Fig. 4. Induced DHCP Starvation Attack from Victim Client's Perspective

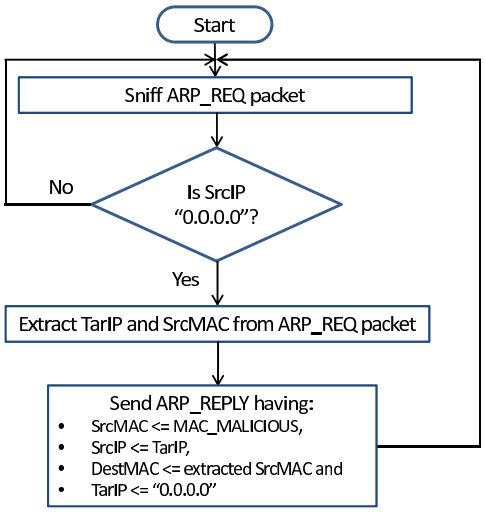


Fig. 5. Induced DHCP Starvation Attack from Malicious Client's Perspective

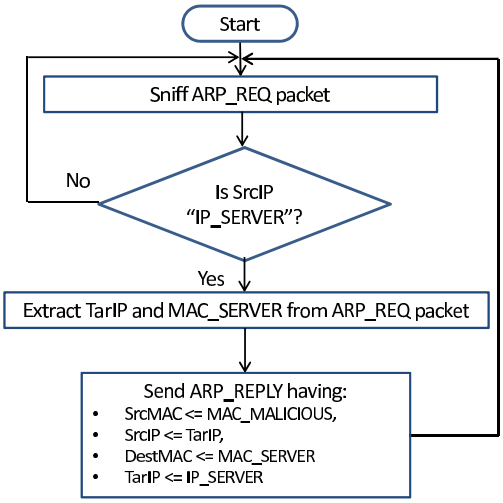


Fig. 7. Server-side IP Address Conflict Detection Exploitation based Attack from Malicious Client's Perspective

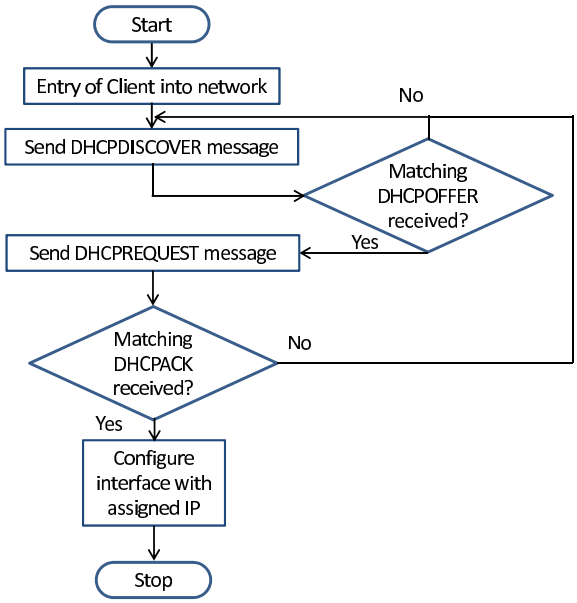


Fig. 6. Server-side IP Address Conflict Detection Exploitation based Attack from Victim Client's Perspective

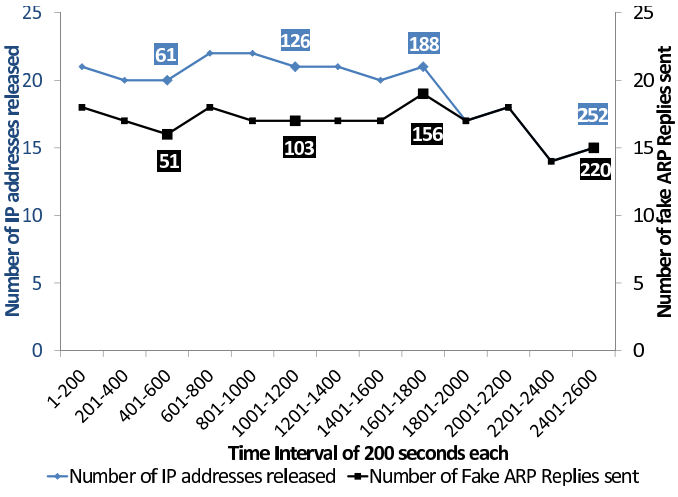


Fig. 8. IP Address Allocation and Fake ARP Replies Sent Over Time with Single Victim in Wireless Network

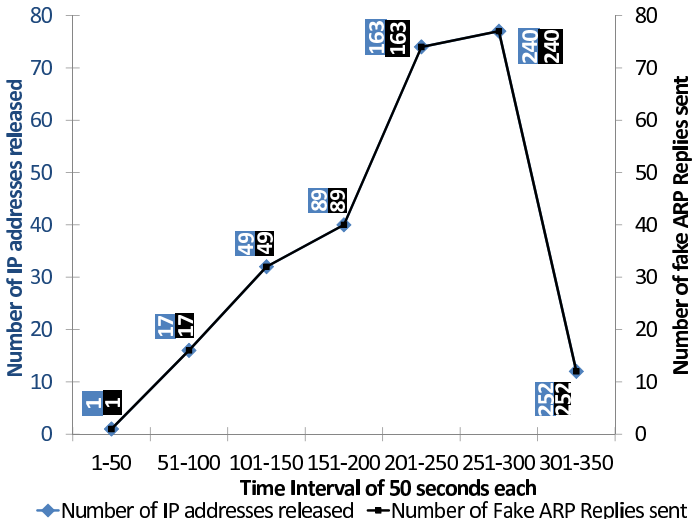


Fig. 9. IP Address Allocation and Fake ARP Replies Sent Over Time with Multiple Victims in Wireless Network

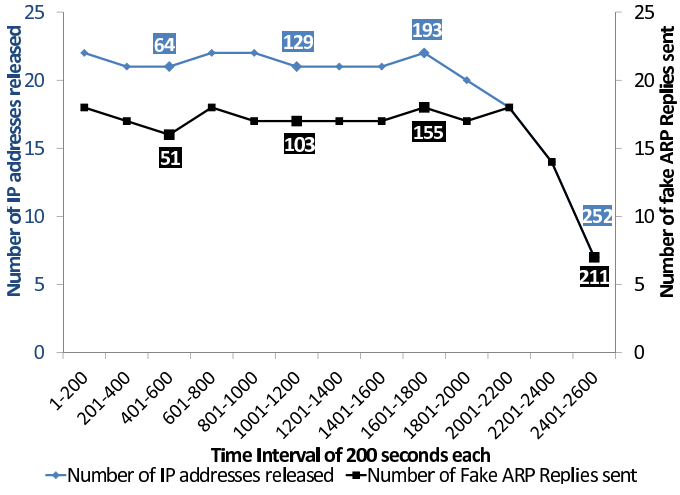


Fig. 10. IP Address Allocation and Fake ARP Replies Sent Over Time with Single Victim in Wired Network

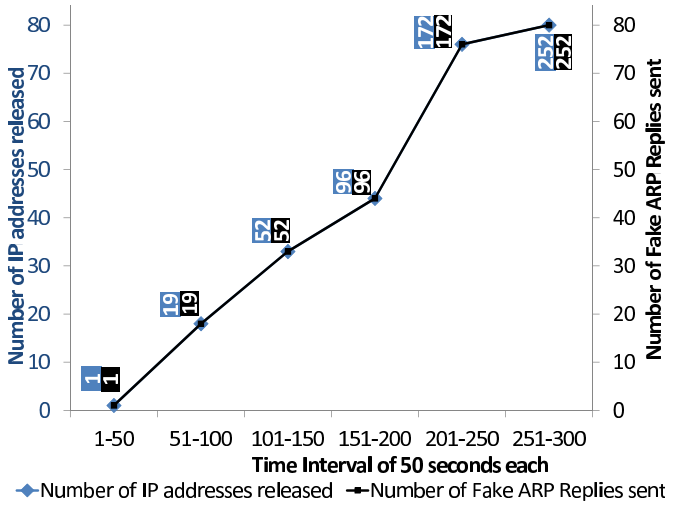


Fig. 11. IP Address Allocation and Fake ARP Replies Sent Over Time with Multiple Victims in Wired Network

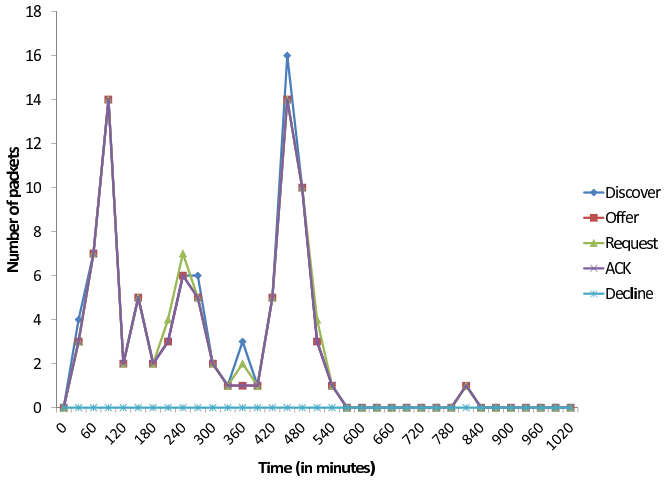


Fig. 12. DHCP Events Occurrences in Different Time Intervals

Fig. 13. Probability Distribution from Training Data

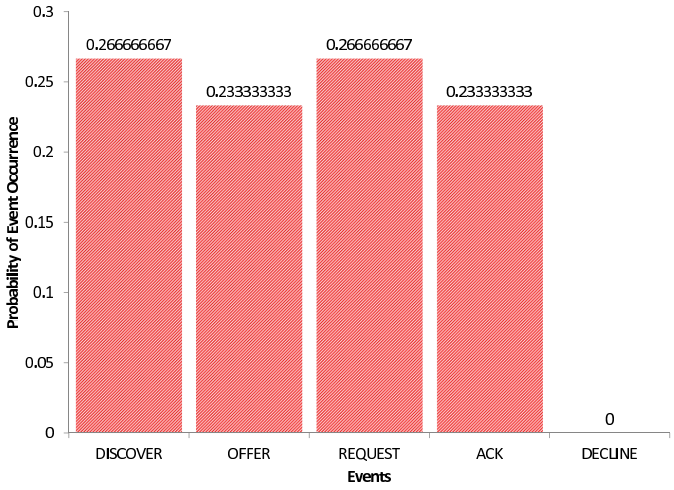


Fig. 14. Probability Distribution Generated from Testing Normal Interval

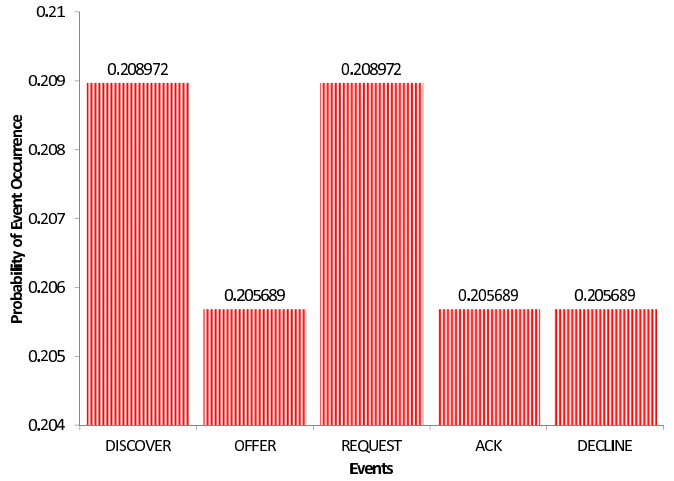


Fig. 15. Probability Distribution Generated from *Induced DHCP Starvation* Attack Interval

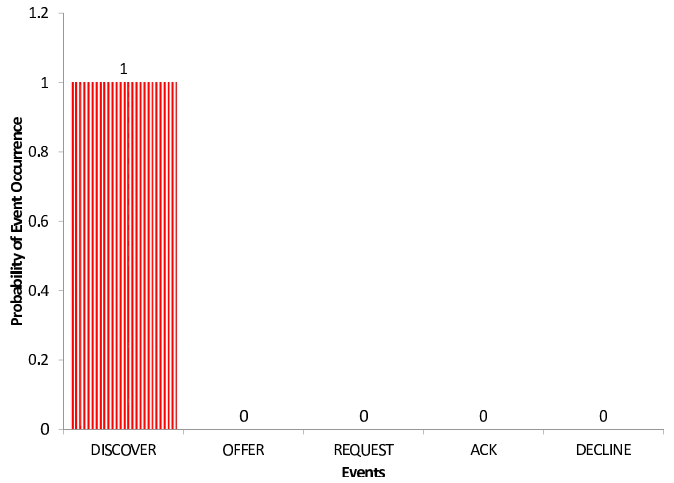
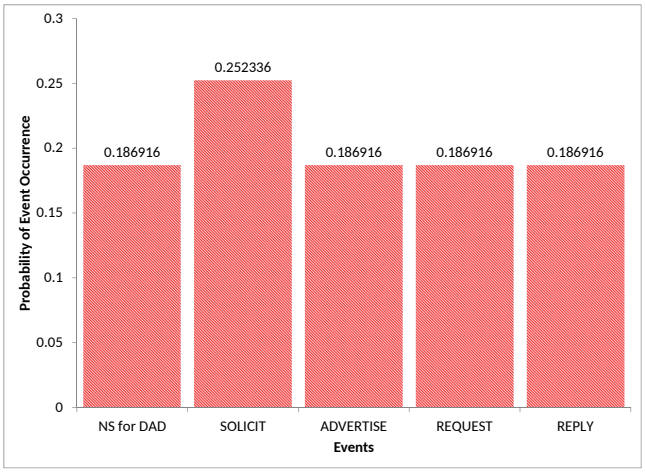
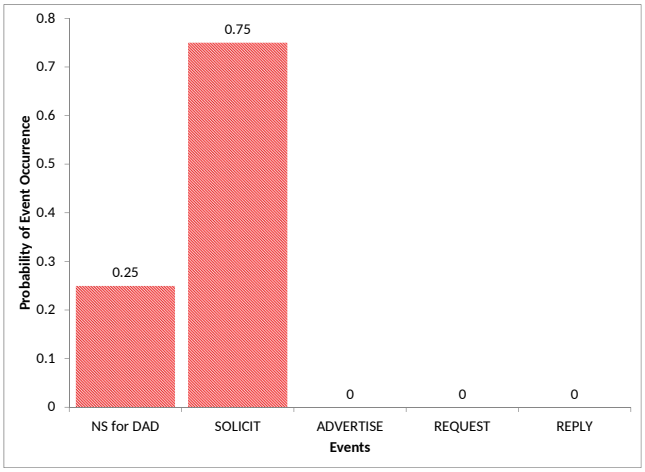


Fig. 16. Probability Distribution Generated from Server-side Exploitation based Attack Interval



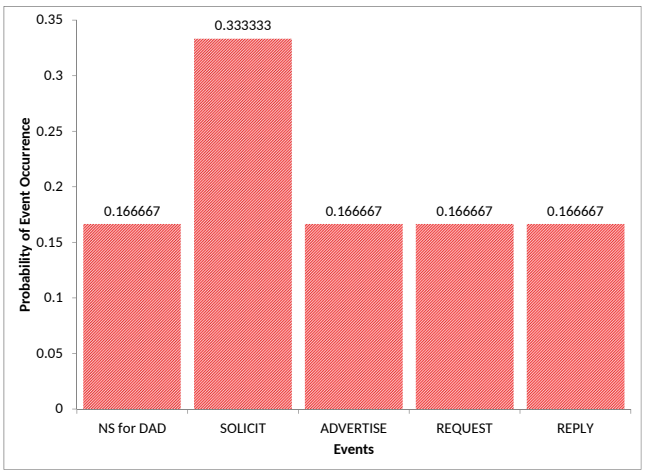
2

Fig. 17. Probability Distribution from Training Data while using DHCPv6



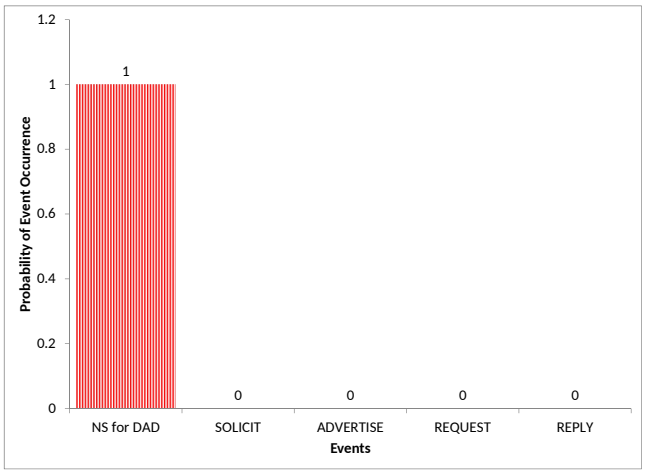
2

Fig. 18. Probability Distribution from Training Data while using SLAAC



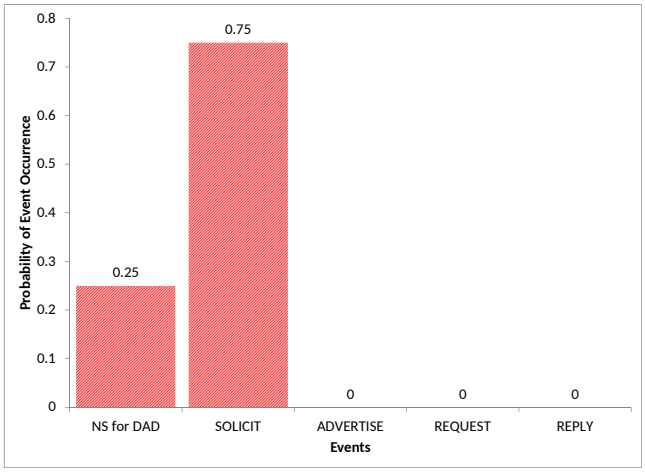
2

Fig. 19. Probability Distribution from Testing Normal Interval while using DHCPv6



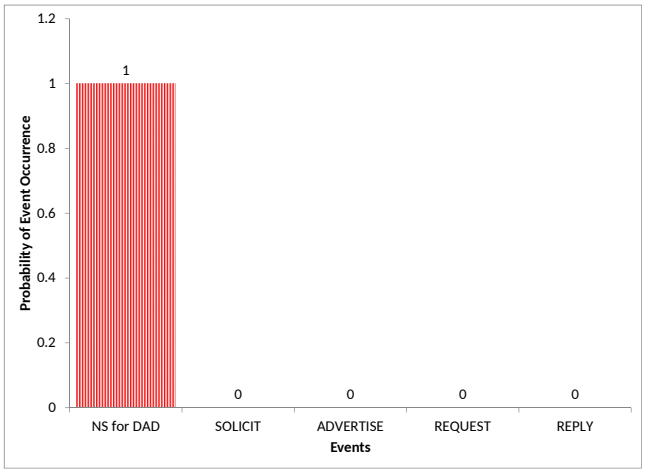
2

Fig. 20. Probability Distribution from Testing DHCP Starvation Attack while using DHCPv6



2

Fig. 21. Probability Distribution from Testing Normal Interval while using SLAAC



2

Fig. 22. Probability Distribution from Testing DHCP Starvation Attack while using SLAAC