Detecting Stealth DHCP Starvation Attack using Machine Learning Approach

Article *in* Journal of Computer Virology and Hacking Techniques · August 2018

DOI: 10.1007/s11416-017-0310-x

2 authors, including:

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

SEE PROFILE

Noname manuscript No.
(will be inserted by the editor)

Detecting Stealth DHCP Starvation Attack using Machine Learning Approach

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

Received: date / Accepted: date

Abstract Dynamic Host Configuration Protocol (DHCP) is used to automatically configure clients with IP address and other network configuration parameters. Due to absence of any in-built authentication, the protocol is vulnerable to a class of Denial-of-Service (DoS) attacks, popularly known as DHCP starvation attacks. However, known DHCP starvation attacks are either ineffective in wireless networks or not stealthy in some of the network topologies. In this paper, we first propose a stealth DHCP starvation attack which is effective in both wired and wireless networks and can not be detected by known detection mechanisms. We test the effectiveness of proposed attack in both IPv4 and IPv6 networks and show that it can successfully prevent other clients from obtaining IP address, thereby, causing DoS scenario. In order to detect the proposed attack, we also propose a Machine Learning (ML) based anomaly detection framework. In particular, we use some popular one-class classifiers for the detection purpose. We capture IPv4 and IPv6 traffic from a real network with thousands of devices and evaluate the detection capability of different machine learning algorithms. Our experiments show that the machine learning algorithms can detect the attack with high accuracy in both IPv4 and IPv6 networks.

Keywords Anomaly Detection · One-class Classifiers · DHCP · DHCPv6 · DHCP Starvation Attack

1 Introduction

Dynamic Host Configuration Protocol (DHCP) [2] is used to obtain network configuration parameters including IP address from a DHCP server. This protocol is vulnerable to a class of Denial-of-Service (DoS) attacks popularly known as classical DHCP starvation attacks. Classical DHCP starvation attacks [4], [5] require a malicious client to inject a large number of IP requests using spoofed MAC addresses. For every such request received, a new IP address is released by a DHCP server. Thus, eventually DHCP server runs out of the IP addresses. However, it is not easy to launch classical DHCP starvation attacks using spoofed MAC addresses in wireless networks as Access Point (AP) drops all the packets having source or destination MAC address previously not associated with it. The only way to create a starvation attack is to precede and maintain association with AP for each spoofed MAC address. However, considering the computational complexity involved in association and key exchange phase in WPA2 wireless networks, it is not feasible to perform multiple manual associations [11]. Moreover, various security features like port security [25] implemented on network switches can easily mitigate this attack by disabling the suspicious port on which multiple MAC addresses are seen at a time. On the other hand, Induced DHCP starvation attacks [17], [6], though effective in wireless networks, can be mitigated by features like Dynamic ARP Inspection (DAI) [1] in wired networks as discussed in Section 3.3.

In this paper, we propose a new stealth DHCP starvation attack that is effective in both IPv4 and IPv6 networks. This attack exploits IP address conflict detection scheme implemented on all DHCP clients. This attack is highly stealth as various popular security features in modern network switches can not detect the attack. Moreover, other proposed methods belonging to either of three categories, viz., *Encryption* [14], *Threshold based* [23] and *Fair IP address allocation* [27] have various drawbacks such as implementation complexity

Address(es) of author(s) should be given

issues in encryption based techniques, high misclassification rate and cumbersome process of deciding threshold in threshold based mechanisms and infeasibility of IP address allocation to each port in wireless networks. Motivated from this fact, we also propose an anomaly detection framework that uses one-class classification techniques to detect the proposed attack. To develop efficient and robust Intrusion Detection/Prevention Systems (IDS/IPS), machine learning algorithms are already used in the literature [8], [9]. Authors in [7] presented recent comprehensive study on use of machine learning techniques to detect intrusion in the networks. Our proposed detection framework learns the normal traffic (DHCP and ARP in IPv4 and DHCPv6 and NS in IPv6) behaviour and once trained, can be put into use to detect anomalies in the network traffic behaviour. In particular, we make following specific contributions:

- We propose a stealth DHCP starvation attack that is easier to launch and difficult to detect by known security measures.
- We test the effectiveness of proposed DHCP starvation attack in both IPv4 and IPv6 networks and report the results.
- We propose an anomaly detection framework in which we use various one-class classifiers to detect anomalous traffic in different time intervals.

Rest of the paper is organized as follows. We provide a brief overview of DHCP working in Section 2. In Section 3, we describe our proposed DHCP starvation attack. Experiments performed to test effectiveness of proposed attack are presented in Section 4. We discuss the proposed detection framework's working and experimental results in Section 5 and Section 6 respectively. We discuss the prior works available in the literature to detect starvation attacks in Section 7. Finally the paper is concluded in Section 8.

2 Background

In this section, we briefly discuss DHCP and DHCPv6 operations in IPv4 and IPv6 networks respectively.

2.1 DHCP in IPv4 Networks

DHCP servers are implemented in local networks to automate the IP address allocation and thus, to reduce the IP address conflicts. DHCP provides ability to define TCP/IP configurations from a central location and thus, it can handle any network changes very efficiently. It is quite cumbersome to allot IP address and other network configuration parameters (e.g., default gateway and DNS server IP address) to each individual device manually within the network. Moreover, it may lead to IP address conflicts also as same IP address can be mistakenly allotted to two or more different devices. Due to these reasons, almost every other network today is equipped with one or more DHCP servers.

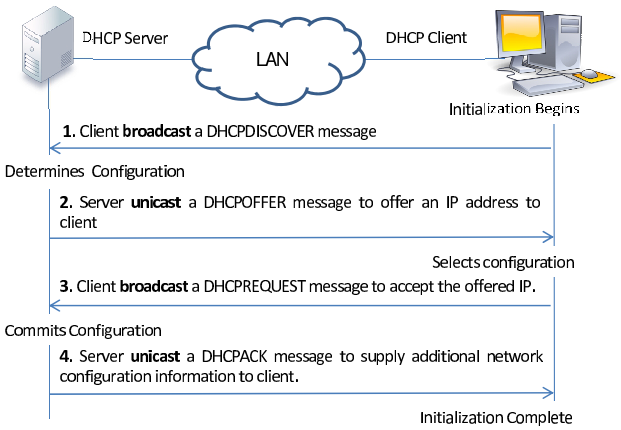

Fig. 1: Exchange of messages in DHCP operation

DHCP server, implemented within a network, is configured with a pool of IP addresses and other network configuration parameters. As soon as a client joins the network, it tries to configure its interface with an IP address by exchanging four messages with the DHCP server as shown in Figure 1. These messages are **D**ISCOVER, **O**FFER, **R**EQUEST and **A**CK and thus, the complete process of IP address allocation is known as *DORA* process. Also, there are various other message types which are exchanged in case of unsuccessful IP address configuration. DHCPDECLINE is one such message that is transmitted by client to DHCP server if client finds that the allotted IP address is already assigned to some other client. A DHCP enabled client finds out such IP address conflicts using ARP request probes destined to the IP address in question.

2.2 DHCPv6 in IPv6 Networks

In an IPv6 network, *Stateful* mode of IP addressing uses DHCPv6 [3] server to allocate site-local unicast IPv6 addresses to clients. Similar to DHCP server in IPv4 networks, DHCPv6 server is also configured with a pool of IP addresses and other network configuration parameters. As soon as a client joins the network, it tries to configure its interface with a site-local unicast IP address by exchanging four messages with the DHCP server. These messages are SOLICIT, ADVERTISE, REQUEST and REPLY and the purpose of these mes-

sages is similar to DHCPDISCOVER, DHCPOFFER, DHCPREQUEST and DHCPACK respectively in IPv4 networks. Various other DHCPv6 message types and their purposes are described in RFC 3315 [3]. DECLINE is one of the message types that is sent by client to DHCP server if it finds that the allotted IP address is already in use by some other client. The client finds such IP address conflicts using NS probe messages destined to the allotted IP address.

3 Proposed Stealth DHCP Starvation Attack

In this section, we discuss the working of proposed stealth DHCP starvation attack in IPv4 and IPv6 networks and subsequently compare it with previously known starvation attacks.

3.1 Stealth DHCP Starvation Attack in IPv4 Networks

Consider a network topology similar to the one shown in Figure 2. In this topology, there are four entities namely DHCP server, malicious client, victim client and a switch. All entities are connected to switch using wired connection.

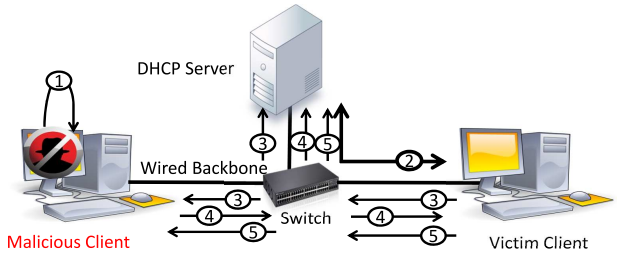

Fig. 2: Proposed Stealth DHCP Starvation Attack Event Sequence

The sequence of events occurred while launching stealth DHCP starvation attack in an IPv4 network is as follows:

1. Manual IP Address Configuration by Malicious Client: As soon as malicious client joins the network, it disables the DHCP daemon running in background and manually configures its interface with an IP address and other configuration parameters. The malicious client does so because in such a way, DHCP server does not allot the IP address to malicious client and thus, it does not possess any record of IP address binded to malicious client's MAC address in its DHCP database. This step helps malicious client to bypass DAI security feature which is dependent on trusted DHCP snooping database to check inconsistencies in IP-MAC bindings.
2. Address Allocation to Victim Client using *DORA* Process: Victim client joins the network and acquires IP address from DHCP server using *DORA* process. After successful allocation, DHCP server adds an entry for a binding between offered IP and victim client's MAC address in its database.
3. ARP Request Broadcast by Victim Client: The victim client broadcasts an ARP request to check if the offered IP address is already in use. The source IP address of this ARP request is "0.0.0.0" as victim client does not configure its network interface with the offered IP prior to performing conflict check.
4. ARP Request Broadcast by Malicious Client: As soon as malicious client receives ARP probe request sent in step 3, it also broadcasts an ARP probe request. The source IP address of this probe is set to "0.0.0.0", target IP address is set to IP address in question, target MAC address is set to "00:00:00:00:00:00", destination MAC in ethernet header is set to "ff:ff:ff:ff:ff:ff" while the source MAC address in ethernet header is set to malicious client's interface MAC address.
5. DHCPDECLINE Message Broadcast by Victim Client: Once the victim client receives the ARP probe request sent in step 4, it declines the assigned IP address by broadcasting a DHCPDECLINE message. DHCP server, on receiving this message, marks the IP address in question as unavailable for the length of lease period. Victim client reinitiates the process of acquiring a new IP address and set of operations are repeated; effectively preventing the victim client from ever acquiring an address. This ultimately leads to DoS scenario. As DHCP server marks every declined IP addresses unavailable for the length of lease period, it runs out of IP addresses available in pool after few iterations.

3.2 Stealth DHCP Starvation Attack in IPv6 Networks

The sequence of events occurred while launching proposed attack in an IPv6 network is as follows:

1. Manually Deriving Link-Local and Site-Local Unicast IP Address by Malicious Client: As soon as malicious client joins the network, it disables the DHCPv6 daemon running in background and man-

ually configures its interface with site-local unicast IPv6 address.

2. Deriving Link-Local and Site-Local Unicast IP Address by Victim Client: As soon as victim client joins the network, it first automatically derives a link-local unicast IP address without the intervention of DHCP server and configures its interface with this IP. After deriving link-local unicast IP address, victim client tries to obtain a routable site-local unicast IP address. To do so, it exchanges SOLICIT, ADVERTISE, REQUEST and REPLY messages with DHCPv6 server. Site-local unicast IP address are equivalent to private IP addresses in IPv4 networks.
3. Duplicate Address Detection (DAD) Checking for Site-local Unicast IP Address by Victim Client: Before using the site-local unicast IP address offered by DHCPv6 server in Step 2, victim client checks if the address is already in use by some other client. To perform this check, it broadcasts a NS probe with target IP address set to offered site-local unicast address. The source and destination IP address of this probe is set to unspecified address (::) and solicited node multicast address respectively.
4. Similar NS Message by Malicious Client: As soon as malicious client receives NS probe sent by victim client in Step 3, it broadcast a similar NS probe with the target, source and destination IP address of the probe set to offered site-local unicast IP address, unspecified address (::) and solicited node multicast address respectively.
5. DECLINE Message by Victim Client: When victim client receives the NS probe sent by malicious client in previous step, it broadcast a DECLINE message to DHCPv6 server informing that the offered address is already in use by some other client. DHCP server, on receiving DECLINE message, marks the IP address in question as unavailable for the lease time. Victim client then reinitiates the process of acquiring a new IP address and set of operations are repeated; effectively preventing the victim client from ever acquiring an address. This results into DoS scenario.

3.3 Stealth DHCP Starvation Attack v/s Classical and Induced DHCP Starvation Attacks

The proposed attack is easier to launch and stealthier as compared to previously known starvation attacks in the following way:

– Attacks' Effectiveness in Wired and Wireless Networks: Since the proposed attack does not require MAC address spoofing, it can easily be launched in wireless networks. Thus, it is equally effective in both wired and wireless networks similar to Induced DHCP starvation attack. However, as discussed earlier, classical DHCP starvation attack is not feasible to launch in wireless networks due to MAC spoofing and also, it fails to exhaust IP address pool in wireless networks even if a fake association is preceded due to restrictions on number of MAC address associations enforced by AP.
– Attacks' Stealthiness: Various security features available in modern network switches can not detect proposed attack as discussed in Section 7. However, classical DHCP starvation attack can easily be mitigated using port security [25] while Induced DHCP starvation attack can be mitigated using Dynamic ARP Inspection (DAI) [1] assuming a network topology similar to the one shown in Figure 3. We can notice that the DAI enabled switch drops the fake ARP reply sent by malicious client since the trusted DHCP snooping database does not have any record of IP-MAC binding present in the fake ARP reply.

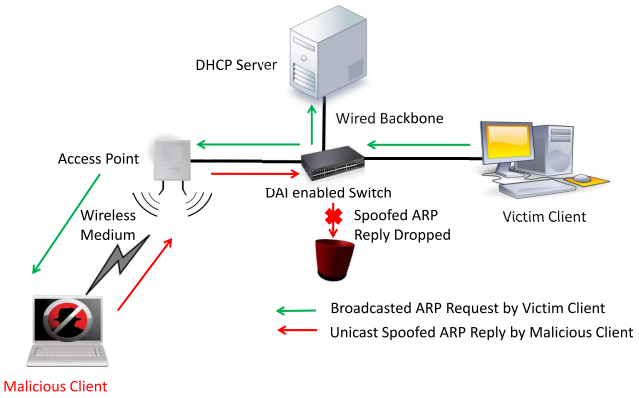

Fig. 3: Ineffectiveness of Induced DHCP Starvation Attack

4 Experimental Setup and Results

In order to test the effectiveness of proposed attack in an IPv4 network, we created a testbed consisting of four computers and one ISC DHCP server having 252 available IP addresses in its pool. One computer was designated as malicious client while other three as victim clients. Malicious and victim clients were using Kali Linux 2.0 and Windows 7 operating systems respectively. All these machines were connected using a switch. We deployed a python script on malicious client to sniff ARP probes and send corresponding ARP

probes in order to mislead victim clients. Using this setup, we launched the attack. While attack was going on, victim clients were trigerred to get an IP address from DHCP server. After getting IP address, victim clients started probing to check for any address conflict. Each time malicious client received an ARP probe from victim clients, it sent similar ARP probes. As a result, victim clients could not acquire IP addresses even after several attempts. Figure 4 shows the number of IP addresses released (equal to number of ARP probes sent by victim clients) by DHCP server over time. The proposed attack took around 970 seconds to exhaust IP pool.

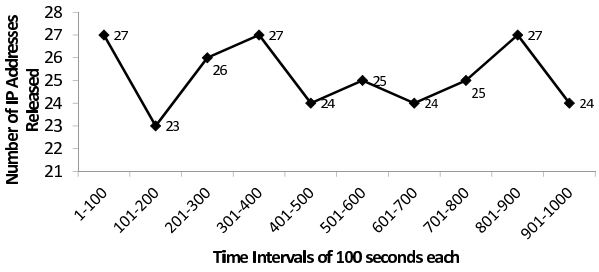

Fig. 4: Number of IP Addresses released by DHCP Server Over Time

To test proposed attack in an IPv6 network, we created a similar network setup. Instead of ISC DHCP server, we used Dibbler DHCPv6 server. A python script on malicious client was used to sniff NS probes and send corresponding NS probes to mislead victim clients. Using this setup, we launched the attack for 1000 seconds. During this time period, we observed that each victim client continuously tried to configure its interface with a link-local unicast IPv6 address and each time, malicious client sent NS probe in response to the probe sent by victim client for conflict detection purpose. As a result, victim clients could not acquire site-local unicast IP addresses even after several attempts. Thus, proposed attack is effective in creating starvation scenario in IPv6 network also. Figure 5 shows the number of IP addresses released (equal to number of NS probes sent by victim clients) by DHCPv6 server over duration of 1000 seconds. Since the pool size of DHCPv6 server was very large, it would take a long time to consume the whole pool. Nevertheless, attack remained effective as victim clients were not able to obtain IP address due to conflict generated by malicious client.

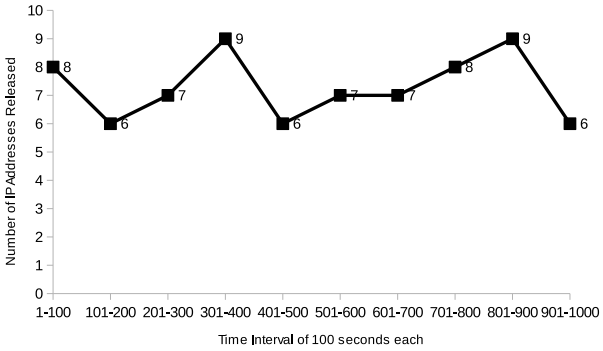

Fig. 5: Number of IP Addresses released by DHCPv6 Server in 1000 seconds

5 Proposed Anomaly Detection Framework

In this section, we present our proposed anomaly detection framework that uses machine learning approach to detect the stealth DHCP starvation attack. In particular, we use six different one-class classification algorithms for detection purpose - Gaussian Density based one-class classifier [18], Naive Bayes one-class classifier [20], K-Nearest Neighbour (KNN) algorithm [19], K-means clustering algorithm [18], Principal Component Analysis (PCA) [18] and Support Vector Data Description (SVDD) [21]. The reason behind choosing different one-class classifiers is to help network administrators to select a suitable classification technique based on the network characteristics. One-class classification algorithms are found to be highly useful in the scenarios when only information of one of the classes, the target class, is available. The task of classification involves defining such boundary around target class so as to cover as much of the target objects as possible and at the same time to minimize the chance of accepting outlier objects.

The proposed framework, shown in Figure 6, consists of two phases: the pre-processing phase and the classification phase. In subsequent subsections, we discuss these two phases.

5.1 Pre-processing Phase

In this phase, DHCP and ARP (with source IP "0.0.0.0") traffic is collected using a packet sniffer to extract packet information like ARP header, UDP header and DHCP header from each packet. We use JNetPcap [12] java library to extract packet information. After that, the packet's information is partitioned by considering different message types and formed into a record by aggregating information after every ΔT time

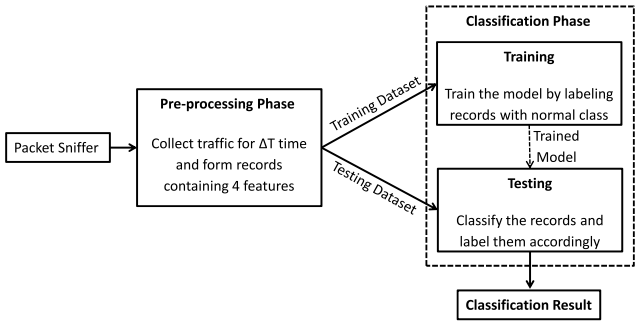

Fig. 6: Proposed Anomaly Detection Framework

Table 1: Candidate Features to Detect Attacks in IPv4 Networks

Features	Description	Type
Feature-1	Number of DHCPDISCOVER messages in ΔT	Numeric
Feature-2	Number of DHCPRREQUEST messages in ΔT	Numeric
Feature-3	Number of DHCPDECLINE messages in ΔT	Numeric
Feature-4	Number of ARP probe requests with source IP "0.0.0.0" in ΔT	Numeric

Table 2: Candidate Features to Detect Attacks in DHCPv6 equipped IPv6 Networks

Features	Description	Type
Feature-1	Number of SOLICIT messages in ΔT	Numeric
Feature-2	Number of REQUEST messages in ΔT	Numeric
Feature-3	Number of DECLINE messages in ΔT	Numeric
Feature-4	Number of NS probe requests having link-local unicast source IP and site-local unicast target IP in ΔT	Numeric

duration. Each record consists of key signature features representing the characteristics of network traffic during ΔT. After extensive experiments, we find 4 essential features to detect any anomaly in both IPv4 and IPv6 networks. These features along with their types are shown in Tables 1 and 2 for IPv4 and IPv6 networks respectively. We consider counts of only those messages as key features which are broadcasted within the network. Messages like DHCPOFFER, DHCPACK, ARP reply in IPv4 networks and ADVERTISE, REPLY, NA in IPv6 networks are unicast and we do not consider count of these messages as features. Due to this reason, even if proposed anomaly detection framework can not receive unicast communication, it can still detect anomaly.

Few examples of data records obtained after pre-processing phase in IPv4 network are shown in Table 3 where network traffic in time intervals T_1, T_2 and T_3 belong to normal class.

Table 3: Example of Data Records from Pre-processing Phase in IPv4 Network

Interval	Data of 4 features	Class
T_1	69, 250, 1, 1286	Normal
T_2	49, 260, 0, 1435	Normal
T_3	58, 127, 0, 1150	Normal

5.2 Classification Phase

In the classification phase, data records obtained from the pre-processing phase is classified as normal or attack data. The classification phase is further divided into training and testing period. In training period, we train the selected one-class classifier as a detection model by using pre-processed data records belonging to normal traffic (answer class) only collected over a period of n time intervals, each of size ΔT. Thus, all data records during training period are labeled as normal category. Once the detection model is trained, we test the model with new or unknown dataset during testing period where each record was captured in a real-time environment as shown in classification phase in Figure 6.

5.3 Consistency-based Model Selection for Different One-class Classifiers

In order to select model with optimized parameter, we use consistency-based model selection [28] with different classifiers. This model selection works based on 2-sigma bound around error of classification model. It determines a threshold error, err_{thr}, as shown in Equation 1 and runs different classifiers with their parameters till it yields less error than err_{thr}. We use consistency based model selection in order to select optimal parameter so that classifiers can create a proper boundary.

$$\begin{aligned} err_{thr} =& (M * fracrej + sigma\_thr * sqrt(fracrej * \\ & (1 - fracrej) * M))/M \\ =& fracrej + sigma\_thr * sqrt(fracrej * \\ & (1 - fracrej)/M) \end{aligned} \tag{1}$$

Here, $M = (N/f)$ such that f denotes number of folds, M denotes number of samples in validation set, $sigma\_thr$ = Required threshold for estimating decision boundary during model selection, $M * fracrej$

denotes estimated number of objects to be rejected, $sqrt(fracrej * (1 - fracrej) * M)$ is the standard deviation and $M * fracrej + sigma\_thr * sqrt(fracrej * (1 - fracrej) * M$ is the maximum allowed number of rejected target objects. For our experiments, we took $fracrej = 5\%$ of most erroneous data.

6 Experiments and Performance Evaluation

In this section, we describe the experiments conducted to evaluate the detection performance of proposed framework. In next few subsections, we discuss testbed setup, datasets for training and testing purpose and detection performance of proposed framework in IPv4 and IPv6 networks.

6.1 Testbed Setup for IPv4 Network

To collect normal DHCP and ARP traffic for training period, we used our institute network which connects approximately 3000 heterogeneous clients like mobile phones, laptops and desktops as shown in Figure 7. There were five types of entities in the network as DHCP server, malicious client, switch, packet sniffer and genuine clients. The DHCP server used in the institute network was ISC DHCP server. This server was configured to offer 65,536 IP addresses[1] in the range of 10.100.0.0 to 10.100.255.255. We installed tcpdump, a packet sniffer, on one computer to capture only broadcast DHCP traffic and ARP request messages having source IP "0.0.0.0". This computer was having 16 GB of physical memory and Core i5 quad core processor and using Ubuntu 16.04LTS operating system. The genuine clients joined and left the network to access different resources as usual without any intervention by us. In this way, we collected normal DHCP and ARP traffic from IPv4 network to evaluate the detection performance of proposed framework.

In order to collect mixed (normal and anomalous) traffic for testing period, we configured malicious client to launch proposed stealth DHCP starvation attack against only one victim client such that remaining genuine clients were unaffected by this attack and able to communicate within the network as usual. The malicious client's computer was having 4 GB of physical memory and Core i5 quad core processor and using Kali 2.0 operating system. We wrote a python script using scapy library [13] and executed it on malicious client in order to sniff ARP requests having source IP "0.0.0.0" from genuine clients and send ARP requests accordingly. These two functions were implemented as two threads such that first thread executed the sniffing module while second thread generated ARP requests with source IP "0.0.0.0". In this way, we collected mixed DHCP and ARP traffic from IPv4 network to evaluate the detection performance of proposed framework.

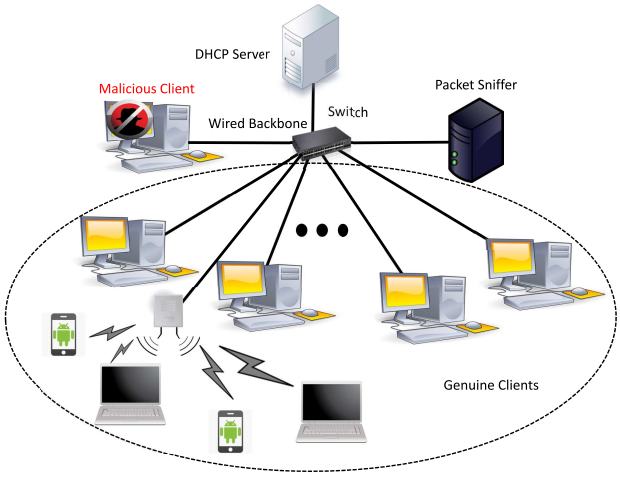

Fig. 7: Testbed for Data Collection

6.2 Testbed Setup for IPv6 Network

To collect normal DHCPv6 and NS traffic for training period, we configured a Dibbler DHCPv6 server in the same institute network. Since clients nowadays come with IPv4/IPv6 dual stack with IPv6 enabled by default, they start configuring their interfaces with IPv6 address as soon as they receive ADVERTISE message from DHCPv6 server in response to SOLICIT message sent by them. So, we did not need to configure each client to use IPv6 along with IPv4. We set the filtering parameters of tcpdump accordingly so as to capture only DHCPv6 traffic and NS messages having link-local unicast source IP and site-local unicast target IP. Using this setup, we collected normal DHCPv6 and NS traffic to evaluate the detection performance of proposed framework in IPv6 network.

In order to collect mixed (normal and anomalous) traffic for testing period, we configured malicious client to launch the attack against only one victim client such that remaining genuine clients were unaffected by this attack. To launch the attack in IPv6 network, we modified the python script executed on malicious client to sniff DHCPv6 traffic and NS messages having link-local

[1] Among 65536, one each was allotted to malicious client and server itself. Other two IP addresses, 10.100.0.0 and 10.100.255.255 were Network and Broadcast address respectively and were not used.

unicast source IP and site-local unicast target IP and to send corresponding NS messages. In this way, we collected mixed DHCPv6 and NS traffic from IPv6 network.

6.3 Training and Testing Dataset in IPv4 Network

We collected 3 days of normal DHCP and ARP traffic from the packet sniffer and partitioned the whole collected traffic in several smaller time intervals $\Delta T = 10$ minutes. As a result, we obtained 432 smaller time intervals. The traffic in these time intervals was then pre-processed in the pre-processing phase which resulted into 432 normal records with each record having 4 features similar to the example shown in Table 3. Out of these 432 normal records, we used 144 normal records (i.e. 1 day normal traffic) for training purpose. The remaining 288 normal records (i.e. 2 days normal traffic) was then used for testing purpose. We further validated proposed framework by rotating the training and testing data so as to use different training and testing data in each iteration. In this way, we performed $3-$fold validation. We also collected 3 days of mixed DHCP and ARP traffic generated by targeting only one victim client. This traffic was also partitioned in time intervals of size $\Delta T = 10$ minutes. As a result, we obtained 432 smaller time intervals which was then pre-processed to form 432 attack records. As a result, we obtained a total of 720 records for testing purpose in which there were 288 normal records and 432 attack records.

6.4 Detection Performance in IPv4 Network

Table 4 shows various one-class classifiers and the associated parameters along with their range. For few of the classifiers, we selected range which was further used by consistency-based model selection to choose an optimal parameter while few classifiers used the range chosen by DD toolbox [29] itself. Table 5 shows mean of various detection performance metrics along with deviations for different one-class classifiers. These results were obtained after $3-$fold cross validation. We compared the performance of various one-class classifiers integrated in DD toolbox available for matlab. We used following performance metrics to evaluate detection performance of proposed framework:

$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$

$Recall = \frac{TP}{TP+FN}$

$Precision = \frac{TP}{TP+FP}$

Table 4: One-Class Classifiers along with Parameters and their Range

Classifer	Parameter Name	Parameter Range
Gaussian Density based Classifier	Regularization parameter	$[10^{-6}, 10^{6}]$
Naive Bayes Classifier	k-smoothing parameter	Default range selected by DD toolbox itself
KNN Classifier	Number of Nearest Neighbours	[1-20]
K-means Clustering	Number of Clusters	[1-20]
PCA	Fraction of Accepted Variance	Default value selected by DD toolbox itself
SVDD	Width parameter in the RBF kernel	Twenty values are selected between minimum and maximum pair wise squared Euclidean distance among training records

Here TP is True Positive, FP is False Positive, TN is True Negative and FN is False Negative. A TP is the case when normal record is correctly detected as normal record by framework while FP is the case when attack record is incorrectly detected as normal one. Similarly, TN is the case when attack record is correctly detected as attack record by framework while FN is the case when normal record is incorrectly detected as attack one.

We can notice from Table 5 that K-means clustering gave best detection results while SVDD performed worst for detection purpose. Observed *Precision* for all the classifiers was 100%. We can also notice that the time taken to build the model was highest in case of Naive Bayes classifier while it was lowest for KNN classifier. Thus, based on these performance metrics, administrators need to keep a trade-off between the time taken to build the model and detection accuracy and then decide which classifier is appropriate in a particular network to detect proposed attack.

6.5 Training and Testing Dataset in IPv6 network

We collected 3 days of normal DHCPv6 and NS traffic from the packet sniffer and partitioned the whole traffic in several smaller time intervals again of size $\Delta T =$ 10 minutes. As a result, we obtained 432 smaller time intervals. On pre-processing the traffic in these time intervals, we obtained 432 normal records. Out of these

Table 5: Detection Performance of Various One-Class Classifiers in case of IPv4 Network

Classifer	Accuracy	Recall	Precision	Time taken to build model (in seconds)
Gaussian Density based Classifier	96.02±2.93	90.05±7.34	100±0	0.0548±0.0476
Naive Bayes Classifier	94.68±2.64	86.69±6.61	100±0	1.9342±3.0651
KNN Classifier	96.20±2.26	90.51±5.66	100±0	0.0427±0.0332
K-means Clustering	96.44±1.90	91.09±4.74	100±0	0.6236±0.5640
PCA	96.20±1.52	90.51±3.81	100±0	0.3706±0.4077
SVDD	85.93±9.83	64.81±24.59	100±0	0.1809±0.1297

432 normal records, we used 144 normal records (i.e. 1 day normal traffic) for training purpose. The remaining 288 normal records (i.e. 2 days normal traffic) was then used for testing purpose. We further validated proposed framework by rotating the training and testing data so as to use different training and testing data in each iteration. In this way, we performed 3−fold validation. We also collected 3 days of mixed DHCPv6 and NS traffic generated by targeting one victim client only. This traffic was also partitioned in time intervals of size $\Delta T = 10$ minutes. As a result, we obtained 432 smaller time intervals which was then pre-processed to form 432 attack records. As a result, we obtained a total of 720 records for testing purpose in which there were 288 normal and 432 attack records.

6.6 Detection Performance in IPv6 Network

The range of parameters of various classifiers to evaluate detection performance of framework in IPv6 network is shown in Table 4. Tables 6 shows mean of various detection performance metrics along with deviations for different one-class classifiers in case of IPv6 network. These results were obtained after 3−fold cross validation. We can notice from Table 6 that K-means clustering and Gaussian Density based classifier gave best results while SVDD again performed worst for detection purpose. Observed *Precision* for all the classifiers was 100%. We can also notice that the time taken to build the model was highest in case of Naive Bayes classifier while it was lowest for KNN classifier.

7 Prior Work and Comparison

Different techniques have been proposed in the literature for the detection and mitigation of DHCP starvation attacks. In this section, we first mention few well-known schemes and then discuss their abilities to detect/mitigate proposed stealth DHCP starvation attack.

7.1 Detection and Mitigation Techniques

Port Security: Port security [25] feature available in modern switches puts a restriction on number of MAC addresses attached to a port of the network switch. Once this limit is crossed, the port is disabled automatically and an SNMP trap is generated. Since the proposed attack does not require MAC address spoofing and the malicious client uses its own MAC address for communication, port security can see only one MAC address at the port. Thus, the proposed attack goes undetected by port security.

DHCP Snooping: DHCP Snooping [10] notices the difference in MAC address present in the Ethernet header and CHADDR field of a DHCPDISCOVER message which helps in mitigating classical DHCP starvation attack. However, proposed stealth DHCP starvation attack does not manipulate the header of DHCP messages and thus, it goes undetected by DHCP snooping also.

DAI with DHCP Snooping: Dynamic ARP Inspection [1] determines the validity of an ARP message by checking valid IP-MAC bindings stored in DHCP snooping database. Since malicious client allocates IP manually to its interface without the intervention of DHCP server, snooping database does not contain an IP-MAC binding corresponding to malicious client. Moreover, the fake ARP requests sent by malicious client are having source IP "0.0.0.0" which is seen as a client has just joined the network and it is sending ARP probes to check address conflict. Thus, the proposed attack is not detected by DAI also.

DHCP Traffic Threshold based Mechanism: Authors in [23] proposed a scheme that counts the number of DHCPREQUEST packets within a period of time. If this count crosses a predefined threshold, the detection system raises the alarm. The proposed stealth DHCP starvation attack also forces victim client for multiple *DORA* processes due to which several DHCPREQUEST messages are generated. However, this number is quite less as compared to classical DHCP starvation attack. As a result, it is difficult to detect proposed attack using this detection scheme.

Table 6: Detection Performance of Various One-Class Classifiers in case of IPv6 Network

Classifer	Accuracy	Recall	Precision	Time taken to build model (in seconds)
Gaussian Density based Classifier	97.92±2.02	94.79±5.04	100±0	0.0553±0.0488
Naive Bayes Classifier	97.08±2.50	92.71±6.25	100±0	0.2954±0.2715
KNN Classifier	97.22±2.65	93.06±6.62	100±0	0.0431±0.0337
K-means Clustering	97.92±2.02	94.79±5.04	100±0	0.0474±0.0370
PCA	94.17±2.47	85.42±6.17	100±0	0.1853±0.1544
SVDD	84.49±11.75	61.23±29.38	100±0	0.1858±0.1284

Limiting Number of IP Addresses on a Switch Port: The mitigation approaches proposed in [26] and [27] involve limiting the number of IP addresses that can be assigned on a switch port. However, since the proposed attack does not require consuming IP addresses by completing several DORA process, it can not be prevented by these mitigation approaches.

Cryptographic Techniques: Due to absence of any in-built authentication scheme in DHCP protocol, starvation attacks are very easy to launch. To mitigate the attacks, various cryptography-based works [14–16, 22, 24] are proposed in the literature to secure the DHCP communication using authentication and encryption. These techniques can easily mitigate identity spoofing based attacks including proposed stealth DHCP starvation attack. However, implementation of these techniques in real network is a very cumbersome process. Some of these techniques which involve certificate and key distribution, require intervention of network administrators. This straightaway conflicts the purpose of using DHCP in the network. Moreover, other third party modules like Authentication Server, Confirmation Server and Detection Server add extra complexity and significant traffic load. Due to these limitations, cryptographic techniques are rarely implemented in local networks.

7.2 Comparison

In this subsection, we evaluate the detection/mitigation performance of different countermeasures which can be deployed to detect/mitigate proposed stealth DHCP starvation attack.

Testing Various Security Features: To test port security, DHCP snooping and DAI, we created a testbed setup similar to the one shown in Figure 8. This setup was having a HP Aruba 2920 layer-3 managed switch and two computers. One computer was designated as malicious client while other one as victim client. Malicious client was using Ubuntu 16.04LTS while victim client was using Windows 7 operating system. Both malicious and victim clients were having 4 GB of physical memory and Core i5 quad core processor. The switch's in-built DHCP server was enabled and configured with a pool of 256 IP addresses. Using this setup, we tested the security features and the experimental observations are presented below.

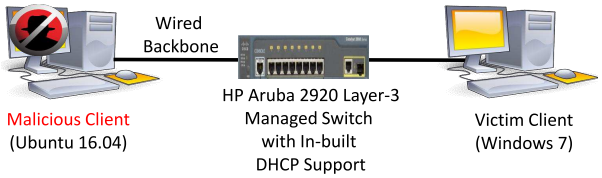

Fig. 8: Testbed Setup for testing Security Features

Testing Port Security: We enabled port security on the switch port to which malicious client was connected by limiting the number of MAC addresses allowed at that port. We also chose the security parameter that if security violation occurred, an SNMP trap must be generated and port must be disabled immediately. After this, we released and then renewed the IP address lease of victim client by executing "ipconfig /release" and "ipconfig /renew" commands on victim client respectively. As soon as victim client performed conflict checking after obtaining an IP address from in-built DHCP server of switch, malicious client sent fake ARP request. This ARP request was not dropped by switch due to which attack was successful and victim client was not able to configure its interface with IP address even after several attempts. Thus, port security was not able to mitigate the proposed attack.

Testing DHCP Snooping: We enabled DHCP snooping globally on the switch and designated the in-built DHCP server of switch as an authorized DHCP server. After this, we released and renewed the IP address lease of victim client. This lead to conflict checking by victim client. Again, malicious client sent fake ARP request in response to the ARP request sent by victim client. Since DHCP snooping did not drop the fake ARP request sent by malicious client, proposed attack was successful due to which victim client was not able to configure its interface with an IP address.

Testing DAI with DHCP Snooping: We further tested the DAI security feature on the switch by first configuring switch ports as trusted because ARP mes-

sages received on untrusted ports were not forwarded in the network due to which victim client could not perform IP conflict checking. DHCP snooping was also enabled along with DAI as DAI used the trusted DHCP snooping database to check if IP-MAC bindings present in ARP messages were genuine. We then released and renewed the IP address lease of victim client and launched the attack from malicious client by sending fake ARP request in response to ARP request sent by victim client for conflict checking. Again, the attack was found to be successful as DAI did not drop the fake ARP request sent by malicious client. Thus, proposed stealth DHCP starvation attack was found to be effective even if DAI was enabled on the switch.

Detection Performance of DHCPREQUEST Threshold based Mechanism [23]: We evaluated the detection performance of the DHCPREQUEST threshold based mechanism [23] in both IPv4 and IPv6 networks. The obtained results are given below.

Detection Performance in IPv4 Network: To evaluate the detection performance of DHCPREQUEST threshold based mechanism in IPv4 network, we used the same 3 days of normal DHCP and ARP dataset (discussed in Section 6.3) which was used to evaluate the detection performance of our proposed detection scheme. From this dataset, we calculated the mean number of DHCPREQUEST messages received in different time intervals of a day where each time interval was of 10 minutes. After this, we compared the number of DHCPREQUEST messages received in each time interval of each day with mean number of DHCPREQUEST messages received in that time interval. For example, number of DHCPREQUEST messages, n_d, received in time interval 12:00am-12:10am of a day d was compared with the mean number of DHCPREQUEST messages, n_m received in time interval 12:00am-12:10am. If the difference $n_m - n_d$ crossed a predefined threshold, anomaly was detected in that interval. From our experiments, we noticed that, in presence of stealth DHCP starvation attack, victim client tried 53 times in a time interval of 10 minutes to acquire an IP address. This resulted into generation of 53 extra DHCPREQUEST messages. If the number of victim clients which were to be targeted increased, the generated number of DHCPREQUEST messages also increased. Thus, we considered 53 as the threshold value for the difference $n_m - n_d$. Since we took time interval size of 10 minutes, there were 144 time intervals in one day. The dataset was having 3 days DHCP traffic thus there were total $144 * 3 = 432$ intervals that were to be tested. Out of these 432 intervals, we injected attack traffic in 258 time intervals by targeting only one victim client. Thus, remaining 174 intervals contained only normal DHCP traffic. First row of Table 7 shows the obtained detection accuracy of this approach while detecting the proposed stealth DHCP starvation attack in IPv4 network. We can notice that putting a threshold limit on a DHCP message type is not a sound approach as we achieved only 65.28% *Accuracy*.

Detection Performance in IPv6 Network: To evaluate the detection performance of DHCPREQUEST threshold based mechanism in IPv6 network, we used the same 3 days of normal DHCPv6 and NS dataset (discussed in Section 6.5) which was used to evaluate the detection performance of our proposed detection scheme. In IPv6 network also, we calculated the mean number of REQUEST messages received in different time intervals of a day where each time interval was of 10 minutes. After this, we compared the number of REQUEST messages received in each time interval of each day with mean number of REQUEST messages received in that time interval. In case of IPv6 network, we considered 15 as the threshold value for the difference $n_m - n_d$ as victim client tried 15 times in a time interval of 10 minutes to acquire an IP address from a DHCPv6 server. In case of IPv6 network also, we injected attack traffic in 258 time intervals out of 432 intervals by targeting only one victim client whereas remaining 174 intervals contained only normal DHCPv6 traffic. Second row of Table 7 shows the obtained detection accuracy of this approach while detecting the proposed stealth DHCP starvation attack in IPv6 network. We can notice that this approach could detect the proposed attack with only 69.44% *Accuracy*.

Mitigation approaches which involve limiting number of IP addresses on a switch port can not mitigate the proposed attack as it does not require consuming IP addresses by completing several DORA process. Cryptographic techniques can mitigate all identity spoofing based attacks including the proposed stealth DHCP starvation attack. However, due to their well-known issues like high implementation and computational complexity and requirement of third party modules like Authentication Servers, we did not test these techniques in our experiments.

8 Conclusion

In this paper, we proposed a stealth DHCP starvation attack that is difficult to detect by known detection mechanisms. The proposed attack is effective and stealthier in both IPv4 and IPv6 networks independent of wired and/or wireless topology. We showed that the attack can easily bypass known detection/mitigation mechanisms and prevent clients from acquiring IP address. We also proposed a ML based anomaly detection

Table 7: Detection Performance of DHCPREQUEST Threshold based Mechanism

Network	Accuracy	Recall	Precision
IPv4	65.28	54.26	81.40
IPv6	69.44	56.20	88.42

framework to detect the proposed attack. The framework uses one-class classifiers in order to classify the traffic within different time intervals. We tested the detection performance of proposed framework in both IPv4 and IPv6 networks using traffic captured from a real network that connected thousands of heterogeneous devices and showed that the framework can detect the proposed attack with very high accuracy.

We believe that this work will motivate researchers in the networking community to further assess DHCP protocol as it may lead to finding new vulnerabilities and thus, mitigating any resulting threat vectors with robust detection/mitigation mechanisms.

References

1. Dynamic ARP Inspection. http://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst6500/ios/12-2SX/configuration/guide/book/dynarp.html. Accessed: 2017-09-23
2. Droms, R.: RFC2131: Dynamic Host Configuration Protocol. Internet Engineering Task Force (1997).
3. Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., Carney, M.: RFC3315: Dynamic Host Configuration Protocol for IPv6 (DHCPv6). Internet Engineering Task Force (2003).
4. Gobbler. http://gobbler.sourceforge.net/. Accessed: 2017-09-23
5. DHCPIG. https://github.com/kamorin/DHCPig. Accessed: 2017-09-23
6. Tripathi, N., Hubballi, N.: Exploiting DHCP Server-side IP Address Conflict Detection: A DHCP Starvation Attack. In: International Conference on Advanced Networks and Telecommunication Systems (ANTS), pp. 1-3, (2015).
7. Aburomman, A.A., Reaz, M.B.I.: A Survey of Intrusion Detection Systems based on Ensemble and Hybrid Classifiers, In Computers & Security, Volume 65, Pages 135-152, (2017).
8. Al-Yaseen, W.L., Othman, Z.A., Nazri, Z.A.: Multi-level Hybrid Support Vector Machine and Extreme Learning Machine based on Modified K-means for Intrusion Detection System, In Expert Systems with Applications, Volume 67, Pages 296-303, (2017).
9. Liu, L., Zuo, W.L., Peng, T.: Detecting Outlier Pairs in Complex Network based on Link Structure and Semantic Relationship, In Expert Systems with Applications, Volume 69, Pages 40-49, (2017).
10. DHCP Snooping. http://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst6500/ios/12-2SX/configuration/guide/book/snoodhcp.html. Accessed: 2017-09-23
11. Xing, X., Shakshuki, E., Benoit, D., Sheltami, T.: Security Analysis and Authentication Improvement for IEEE 802.11i Specification. In: Global Telecommunications Conference (GLOBECOM), pp. 1-5, (2008).
12. JNetPcap. http://jnetpcap.com/docs/javadocs/jnetpcap-1.4/index.html. Accessed: 2017-09-23
13. Scapy. http://www.secdev.org/projects/scapy/. Accessed: 2017-09-23
14. Issac, B.: Secure ARP and Secure DHCP Protocols to Mitigate Security Attacks. International Journal of Network Security, 8(2), pp. 107-118, (2009).
15. Droms, R., Arbaugh, W.: RFC3118: Authentication for DHCP Messages. Internet Engineering Task Force (2001).
16. Jerschow, Y. I., Lochert, C., Scheuermann, B., Mauve, M.: CLL: A Cryptographic Link Layer for Local Area Networks. In: International Conference on Security and Cryptography for Networks (SCN), pp. 21-38, (2008).
17. Hubballi, N., Tripathi, N.: A Closer Look into DHCP Starvation Attack in Wireless Networks, In Computers & Security, Volume 65, Pages 387-404, (2017).
18. Bishop, C., M.: Neural Networks for Pattern Recognition. Oxford University Press, Inc., (1995).
19. Chien, Y.: Pattern Classification and Scene Analysis, In IEEE Transactions on Automatic Control, 19(4), pp. 462-463, (1974)
20. Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian Network Classifiers. Machine learning, 29(2-3), 131-163, (1997).
21. Martinus, D., Tax, J.: One-class Classification: Concept-learning in the Absence of Counterexamples, PhD Thesis, Delft University of Technology, (2001).
22. Demerjian, J., Serhrouchni, A.: DHCP Authentication using Certificates. In: Security and Protection in Information Processing Systems, Springer US, pp. 456-472, (2004).
23. OConnor, T.: Detecting and Responding to Data Link Layer Attacks. http://www.sans.org/reading-room/whitepapers/intrusion/detecting-responding-data-link-layer-attacks-33513. Accessed: 2017-09-23.
24. de Graaf, K., Liddy, J., Raison, P., Scano, J., Wadhwa, S.: Dynamic Host Configuration Protocol (DHCP) Authentication using Challenge Handshake Authentication Protocol (CHAP) Challenge. US Patent 8,555,347, (2013).
25. Port Security. http://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst6500/ios/12-2SX/configuration/guide/book/port_sec.html. Accessed: 2017-09-23
26. Patrick, M.: RFC3046: DHCP Relay Agent Information Option. Internet Engineering Task Force (2001).
27. Mukhtar, H., Salah, K., Iraqi, Y.: Mitigation of DHCP Starvation Attack. Computers & Electrical Engineering, 38(5), pp. 1115-1128, (2012).
28. Tax, D.M.J., Muller, K.R.: A Consistency-based Model Selection for One-class Classification," In: International Conference on Pattern Recognition (ICPR), pp. 363-366, (2004).
29. Tax, D.M.J.: DDtools, the Data Description Toolbox for Matlab, version 2.1.2, (2015).