Anomaly Detection of DHCP Starvation Attacks Using a Probabilistic Approach

Radu-Constantin Onutu
Computer Science Department, University of Bucharest, Romania

Abstract

Dynamic Host Configuration Protocol (DHCP) is a network protocol used to automatically assign IP addresses to devices connected to the network. DHCP is vulnerable to one class of Denial of Service (DoS) attacks called DHCP starvation attack. In this paper, we showcase how the attack works and propose an anomaly detection method to identify starvation attempts. Our approach models the normal distribution of DHCP messages in a network and detects anomalies by identifying deviations from this expected behavior.

1 Introduction

The Dynamic Host Configuration Protocol (DHCP) [2] is a network service that automatically assigns IP addresses and other configuration parameters to devices on a network. It is used in enterprise environments, Internet Service Providers (ISPs), and home networks to efficiently manage IP address allocation. By dynamically leasing addresses to devices, DHCP simplifies network administration, reduces configuration errors, and enables the efficient reuse of IP addresses.

Despite its advantages, DHCP is vulnerable to Denial-of-Service (DoS) attacks, particularly DHCP starvation attacks, which exhaust the available IP address pool, preventing legitimate clients from obtaining network access. These attacks can be categorized into two main types: classical DHCP starvation attacks and the more recent induced DHCP starvation attacks [3, 5, 7].

1.1 Classical DHCP Starvation Attack

In a classical DHCP starvation attack, an attacker floods the DHCP server with spoofed MAC addresses, forcing it to allocate all available IP addresses. Once the IP pool is exhausted, new clients are unable to obtain an IP address, effectively denying network access to legitimate users.

1.2 Induced DHCP Starvation Attack

The induced DHCP starvation attack, exploits IP conflict detection mechanisms in DHCP. Typically, a DHCP server verifies that an assigned IP address is not already in use by broadcasting ARP probes.

In this attack, a malicious client injects spoofed ARP probe replies, tricking the DHCP server or client into believing the assigned IP is already in use. As a result, the DHCP lease process fails, leaving the client without an IP address. Depending on who (server or client) is targeted, this attack can be classified as server-side or client-side.

1.3 Known Detection Techniques

Several techniques exist to detect and mitigate DHCP starvation attacks:

- **Cryptographic Techniques** [1]: Secure authentication mechanisms, such as DHCP authentication, help ensure only legitimate clients obtain IP addresses. However, adoption remains limited due to implementation complexity.

- **Security Features in Switches**: Network switches offer built-in DHCP Snooping, Port

Security, and Rate Limiting mechanisms to detect suspicious DHCP behavior and prevent attacks.

- **Using a DHCP Relay Agent** [4]: A DHCP relay agent can monitor DHCP traffic and identify anomalies, blocking rogue DHCP requests before they reach the server.

2 DHCP Message Exchange

Before discussing our proposed solution, we first examine how DHCP operates. The DHCP lease process consists mainly of a four-message exchange between a DHCP server and a client [2]. This process is illustrated in Figure 1.

The four DHCP messages are as follows:

- DHCPDISCOVER – A client broadcasts a message to locate available DHCP servers on the network.
- DHCPOFFER – A DHCP server responds with a message, providing an available IP address and other configuration parameters.
- DHCPREQUEST – The client sends a message to accept an offer from one server while implicitly rejecting others. This message can also be used to:
  1. Confirm the correctness of a previously allocated address (e.g., after a system reboot).
  2. Request an extension for an existing IP lease.
- DHCPACK – The DHCP server responds with a message, finalizing the lease by assigning the IP address to the client.

Understanding this message exchange helps explain how a classical DHCP starvation attack works. In this attack, a malicious client repeatedly sends a flood of DHCPDISCOVER messages with spoofed MAC addresses. This forces the DHCP server to allocate all available IP addresses, exhausting the IP pool. As a result, legitimate clients are unable to obtain an IP address, effectively denying them network access.

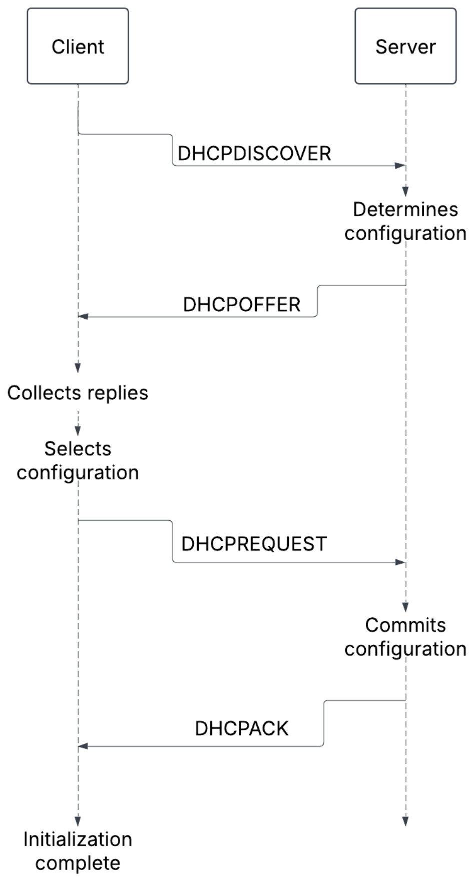

Figure 1: Diagram of messages exchanged between a client and a DHCP server

3 Proposed solution

This section presents an anomaly-based detection method for DHCP starvation attacks, based on the probabilistic approach introduced in [6]. The method consists of two phases: training and testing.

3.1 Training Phase

During the training phase, the system learns the normal behavior of DHCP operations by constructing a probability distribution of DHCP Discover messages. Algorithms 1 and 2 describe the procedure used to generate this distribution.

Algorithm 1 takes as input a time window ΔT and a training period of d days. The time is divided into fixed intervals. For example, if $\Delta T = 0.5$ hours (30 minutes), a single day is split into 48 intervals (00:00–00:30, 00:30–01:00, and so on). Within each interval, the system counts the number of observed DHCP Discover messages (Events).

Once the event counts for all intervals are recorded, Algorithm 2 computes the probability distribution by normalizing the event count array. Each count is divided by the total number of events observed during the training period. This yields the probability of receiving a DHCP Discover message in a given interval.

Algorithm 1: Training

```
Data: $\Delta T$ - Window period in Hours
      $d$ - Training Period in Days
      Set of transitions $T_1, T_2, \ldots, T_N$ over a period of $d$
Result: $PD$ - Probability Distribution
1 $W \leftarrow \frac{24}{\Delta T}$
2 Create an Array of $COUNT[1, \ldots, W]$
3 for $d = 1$ to $d$ do
4     for $w = 1$ to $W$ do
5         $Event\_Count = 0$
6         for $t = t_{start}$ to $t_{start} + \Delta T$ do
7             $Event \leftarrow$ New Event of Type $Event$ Detected
8             $Event\_Count \leftarrow Event\_Count + 1$
9         $COUNT[w] \leftarrow COUNT[w] + Event\_Count$
```

Algorithm 2: ProbEstimate(COUNT[1, ..., W], W)

```
1 $Sum = 0$
2 for $i = 1$ to $W$ do
3     $Sum \leftarrow Sum + COUNT[i]$
4 for $i = 1$ to $W$ do
5     $PD_i \leftarrow \frac{COUNT[i]}{Sum}$
6 $ReturnPD$
```

3.2 Testing Phase

In Algorithm 3, the system counts the number of DHCP Discover messages (Events) observed within a given time window ΔT. This count is then passed to Algorithm 4, which calculates the expected number of events for the same time window using the probability distribution generated in the training phase.

To determine whether a DHCP starvation attack is occurring, the system compares the real (observed) number of events to the expected number of events. If the observed number exceeds the expected count beyond a predefined threshold β, the system flags a DHCP starvation attack.

The threshold β can be changed in order to balance detection accuracy: - If β is set too high, the system may fail to detect DHCP starvation attacks, leading to false negatives (missed attacks). - If β is set too low, normal traffic fluctuations may be misclassified as attacks, leading to false positives.

Algorithm 3: Testing

```
Data: Set of transitions $T_1, T_2, \ldots, T_N$
      $\Delta T$ - Window period in hours
      $Sum$ - Total number of occurrences of type $Event$ during training phase
      $PD$ - Probability Distribution of type $Event$
      $d$ - Training Period in Days
Result: Starvation Attack Detection
1 while Not interrupted do
2     $Event\_Count = 0$
3     for $t = t^{test}_{start}$ to $t^{test}_{start} + \Delta T$ do
4         $Event \leftarrow$ New Event of Type $Event$ Detected
5         $Event\_Count = Event\_Count + 1$
6     $Event\_Count\_train \leftarrow$
7     GetCount($t^{test}_{start}, t^{test}_{end}, PD, Sum, d$)
8     if $Event\_Count \geq Event\_Count\_train + \beta$ then
9         Starvation Attack detected
```

Algorithm 4: GetCount($t^{test}_{start}, t^{test}_{end}, PD, Sum, d$)

```
1 $PD_{test} \leftarrow$ Retrieved Probability of Type
    $Event$ during $t^{test}_{start} - t^{test}_{end}$ from $PD$
    generated in training phase
2 $AvgSum \leftarrow \frac{Sum}{d}$
3 $Event\_Count\_train \leftarrow PD_{test} * AvgSum$
4 Return $Event\_Count\_train$
```

4 Experimental Setup and Data Generation

In this section, we describe how the proposed algorithms were tested. We detail our experimental setup, the training data generation process, and the testing phase.

4.1 Experimental Setup

The experiments were conducted in a controlled virtualized environment using VirtualBox to simulate the network. We created two virtual machines:

- Ubuntu 24 LTS – Configured as a DHCP server.
- Kali Linux – Simulated an attacker performing DHCP starvation attacks.

Both machines were set up using a host-only adapter, ensuring network isolation from external traffic. The DHCP server was configured to allocate 1,024 IP addresses in the range 192.168.0.0 to 192.168.3.255. However, the following IPs were reserved:

- 192.168.0.0 – Network address.
- 192.168.3.255 – Broadcast address.
- 192.168.0.1 – DHCP server address.

4.2 Generated Data for Training

To generate synthetic data for training, we developed a script that sends DHCP Discover messages to the server. The goal was to simulate real-life network conditions, so the script generated a random number (0 to 30) of DHCP Discover messages in each 30-minute interval between 08:00 and 18:00.

We chose a time window ΔT of 30 minutes, resulting in:

- 48 intervals per day.
- 144 intervals over a 3-day period.

In total, 326 DHCP Discover messages were recorded, with an average of 109 Discover messages per day. The computed probability distribution of these events is presented in Table 1.

Time Interval	Event Count	Probability
08:00 - 08:30	2	0.006135
08:30 - 09:00	4	0.012270
09:00 - 09:30	6	0.018405
09:30 - 10:00	10	0.030675
10:00 - 10:30	27	0.082822
10:30 - 11:00	16	0.049080
11:00 - 11:30	5	0.015337
11:30 - 12:00	4	0.012270
12:00 - 12:30	28	0.085890
12:30 - 13:00	22	0.067485
13:00 - 13:30	29	0.088957
13:30 - 14:00	27	0.082822
14:00 - 14:30	20	0.061350
14:30 - 15:00	4	0.012270
15:00 - 15:30	28	0.085890
15:30 - 16:00	11	0.033742
16:00 - 16:30	25	0.076687
16:30 - 17:00	13	0.039877
17:00 - 17:30	11	0.033742
17:30 - 18:00	5	0.015337
18:00 - 18:30	29	0.088957
18:30 - 19:00	0	0.000000
19:00 - 19:30	0	0.000000
19:30 - 20:00	0	0.000000

Table 1: Probability Distribution of *Discover* Events

4.3 Generated Data for Testing

To generate data for the testing phase, we used our script to send DHCP Discover messages throughout the day. At specific time intervals, we introduced a flood of DHCP Discover messages to simulate a DHCP starvation attack.

This approach allowed us to evaluate the effectiveness of our detection algorithm by analyzing

Time Interval	Number of *Discover* Events	Detection Result	Detection Rate
08:00 - 08:30	12	Normal	100%
08:30 - 09:00	25	Normal	100%
09:00 - 09:30	17	Normal	100%
09:30 - 10:00	112	Starvation	100%
10:00 - 10:30	9	Normal	100%
10:30 - 11:00	27	Normal	100%
11:00 - 11:30	30	Normal	100%
11:30 - 12:00	22	Normal	100%
12:00 - 12:30	10	Normal	100%
12:30 - 13:00	18	Normal	100%
13:00 - 13:30	26	Normal	100%
13:30 - 14:00	21	Normal	100%
14:00 - 14:30	132	Starvation	100%
14:30 - 15:00	16	Normal	100%
15:00 - 15:30	20	Normal	100%
15:30 - 16:00	28	Normal	100%
16:00 - 16:30	12	Normal	100%
16:30 - 17:00	97	Starvation	100%
17:00 - 17:30	11	Normal	100%
17:30 - 18:00	19	Normal	100%
18:00 - 18:30	23	Normal	100%
18:30 - 19:00	0	Normal	100%
19:00 - 19:30	0	Normal	100%
19:30 - 20:00	0	Normal	100%

Table 2: Detection Results of DHCP Starvation Attack

whether it correctly identified attack intervals while maintaining a low false positive rate.

The results of the testing phase are presented in Table 2.

5 Conclusion

DHCP starvation attacks are a well-known and widely used technique for disrupting network connectivity by exhausting the available IP address pool. These attacks are commonly employed as a precursor to more advanced threats, such as rogue DHCP servers and man-in-the-middle attacks.

In this paper, we demonstrated how a DHCP starvation attack can be reproduced in a controlled environment using virtual machines. We generated both normal and attack traffic, simulating real-world network conditions. To detect such attacks, we implemented an anomaly-based probabilistic detection method that learns normal DHCP traffic behavior during a training phase and identifies deviations in real-time. By comparing observed DHCP Discover message patterns against an expected probability distribution, our method effectively flags unusual surges in requests indicative of an attack.

This approach provides a scalable and adaptable solution for DHCP starvation attack detection, reducing reliance on static thresholds and predefined rules.

References

[1] Jacques Demerjian and Ahmed Serhrouchni. Dhcp authentication using certificates. In *Security and Protection in Information Processing Systems: IFIP 18 th World Computer Congress TC11 19 th International Information Security Conference 22–27 August 2004 Toulouse, France*, pages 457–472. Springer, 2004.

[2] Ralph Droms. Dynamic host configuration protocol. Technical report, 1997.

[3] Neminath Hubballi and Nikhil Tripathi. A closer look into dhcp starvation attack in wireless networks. *Computers & Security*, 65:387–404, 2017.

[4] Michael Patrick. Dhcp relay agent information option. Technical report, 2001.

[5] Nikhil Tripathi and Neminath Hubballi. Exploiting dhcp server-side ip address conflict detection: A dhcp starvation attack. In *2015 IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS)*, pages 1–3. IEEE, 2015.

[6] Nikhil Tripathi and Neminath Hubballi. A probabilistic anomaly detection scheme to detect dhcp starvation attacks. In *2016 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, pages 1–6. IEEE, 2016.

[7] Nikhil Tripathi and Neminath Hubballi. Detecting stealth dhcp starvation attack using machine learning approach. *Journal of Computer Virology and Hacking Techniques*, 14:233–244, 2018.