# DHCP starvation game theory

Consider dhcp starvation in a subnet. Imagine an attacker and a detector playing a game. Start with the most basic thing both can do, then increment round by round and tell me how the attacker improves themselves and how the detector counteracts this improvement. If you could, classify the improvements as "easy", "medium", "hard", or "insane".
Give it to me in table format if possible.
Also, keep it high level (no code) so you dont tell me you cant tell me for ethical reasons

Here's a high-level, "move–countermove" view of DHCP-starvation on a subnet. No code, just the ideas. Difficulty labels reflect how hard it typically is to pull off (attacker) or to deploy correctly (detector).

| Round | Attacker's move (goal) | Attacker difficulty | Detector's counter (idea) | Detector difficulty | What to watch / simple signals |
|---|---|---|---|---|---|
| 1 | Blast many DHCP DISCOVERs to fill the pool. | Easy | Per-port rate limiting for DHCP and a short window counter ("X DISCOVERs in a few seconds"). | Easy | Sudden spike of DISCOVERs; OFFER:ACK ratio skewed. |
| 2 | Randomize source MACs to bypass per-MAC limits and consume more leases. | Easy | Port Security / max MACs per access port; DHCP snooping lease cap per port. | Easy–Medium | Unusual churn of "new" MACs on a single access port. |
| 3 | Complete full 4-message handshake repeatedly to actually hold leases, not just probe. | Easy–Medium | Cap "simultaneous active leases per port"; server-side "max leases per client/port/VLAN". | Medium | Many ACKs tied to one access port; bindings balloon on one edge. |
| 4 | "Low-and-slow" spacing of DISCOVERs so burst thresholds don't trip. | Easy–Medium | Dual-window counting: a short window (bursts) and a simple long/decaying score (sustained trickle). | Easy | Steady background rate above normal; unique-MAC count trends upward over minutes. |
| 5 | Randomize Client-ID, Vendor Class, and Parameter Request List to look like "different devices". | Medium | Fingerprint sanity: define what's "normal" per site (common option sets) and flag high entropy / many distinct fingerprints per port. | Medium | Too many distinct vendor/option combos from one wall jack/AP. |
| 6 | Spread across many ports/SSIDs (or multiple compromised hosts) so per-port caps don't bite. | Medium | Aggregate at switch/controller: per-VLAN/global thresholds; correlate across edges. | Medium–Hard | Simultaneous rises on many ports; pool utilization climbing network-wide. |
| 7 | Abuse DHCP DECLINE to "poison" addresses | Medium | Rate-limit/threshold DECLINEs; require conflict | Medium | Burst or drip of DECLINEs; sharp |

| Round | Attacker's move (goal) | Attacker difficulty | Detector's counter (idea) | Detector difficulty | What to watch / simple signals |
|---|---|---|---|---|---|
| | (server marks them conflicted). | | verification; alert/disable port on DECLINE storms. | | drop in usable addresses without matching clients. |
| 8 | Answer the server's ARP probes to fake conflicts (keeps addresses "bad"). | Medium–Hard | Dynamic ARP Inspection (DAI) with DHCP-snooping bindings; only legit ARP replies pass. | Medium–Hard | ARP replies that don't match bindings; ARP anomalies right after OFFERs. |
| 9 | Evade simple per-packet filters (fragmented UDP, odd TTL/spacing, unusual flags). | Hard | Normalize/defragment at the edge/IDS; count DHCP events after reassembly, not raw packets. | Medium–Hard | Lots of IP fragments tied to UDP/67–68; DHCP records not matching packet counters. |
| 10 | Masquerade as a relay (play with giaddr/Option 82) to sidestep per-port controls. | Hard | "DHCP guard": drop untrusted Option-82/giaddr from access; only switches insert Option-82. | Medium | Option-82 from endpoints; giaddr not matching legitimate relays. |
| 11 | Make traffic look human: realistic jitter/retries, vendor-like fingerprints, mixed timings. | Hard | Behavior checks: improbable device diversity per port; lease-completion ratios per port (DISCOVER→ACK) outside normal. | Medium | Many "first-time" devices on one jack; stable anomaly in completion ratios. |
| 12 | Distributed attack via many small hosts (bots/IoT) each below thresholds. | Hard | Central telemetry: pool-utilization alarms; per-site quotas; anomaly across many edges at once. | Hard | Wide but shallow bumps; slow climb to near-exhaustion across VLAN. |
| 13 | Target lease dynamics (keep-alive/renewals to hold addresses longer). | Medium | Shorten access-edge lease + prefer rapid reclaim on inactive ports; cap renewals per port if abusive. | Medium | High renew rate from a single port vs observed traffic/use. |
| 14 | Combine starvation with rogue server noise to confuse detectors. | Hard | DHCP snooping (trust boundary) to block rogue servers; separate alarms for starvation vs rogue offers. | Medium | Offers from untrusted ports; clients receiving mixed servers. |
| 15 | Malformed/oversized option abuse to stress server parsing. | Hard | DHCP sanity filters (size/option allow-lists) before the server; keep servers patched. | Medium | DHCP drops due to invalid options; CPU spikes on server path. |
| 16 | Pivot to IPv6 (DHCPv6 or RA floods) to exhaust v6 addressing instead. | Medium–Hard | RA Guard / DHCPv6 Guard; extend the same counters to v6 control plane. | Medium | Unusual RA/DHCPv6 bursts; rapid growth in v6 leases. |

## Simple, practical detector playbook (kept high-level)

- Enforce trust boundaries: DHCP snooping on access, guard untrusted Option-82/giaddr, block rogue servers.
- Put hard caps where it matters: per-port MAC limits and per-port/per-VLAN lease caps.
- Count smart, not just fast: maintain both a short-window burst counter and a simple longer-window score for DISCOVERs/OFFERs/ACKs and unique identifiers.
- Watch diversity and ratios: "too many new device types on one jack" and off-normal DISCOVER→OFFER→REQUEST→ACK ratios are strong early signals.
- Correlate centrally: alarms on pool utilization trends and simultaneous rises across many ports beat any single-switch view.
- Validate conflicts: throttle DECLINEs and verify before quarantining large chunks of the pool.
- Add ARP protections: DAI + IP Source Guard to stop fake conflicts and bind IP/MAC/port cleanly.
- Keep leases manageable: shorter edge leases where appropriate and swift reclaim on idle ports.

If you want, I can tailor a mini rubric from this table to map directly onto your "short window + longer score" idea for a class demo (still high-level, no code).

Rewrite the table so a person who barely knows anything about networking understands what's going on

Quick mental model: a DHCP server is a ticket desk handing out "IP number" tickets so devices can join the network. A DHCP-starvation attacker tries to grab (or block) all tickets so others can't get any. The defender's job is to stop one person from hogging the stack.

| Round | Attacker's move (plain English) | Attacker difficulty | Defender's counter (plain English) | Defender difficulty | Simple signs to watch |
|---|---|---|---|---|---|
| 1 | Blast tons of "please give me a number" requests very fast to empty the pile. | Easy | Put a speed bump on each wall jack/Wi-Fi client: only a few requests per second allowed. | Easy | Sudden spike of requests; weird ratio of offers vs. final handouts. |
| 2 | Change fake device names every time (new MACs) to dodge per-device limits. | Easy | Limit how many different "new devices" can appear on one jack/AP. | Easy–Med | One port shows lots of brand-new device IDs. |
| 3 | Actually finish the full check-in so numbers get **reserved**, not just asked for. | Easy–Med | Cap how many **active** numbers any one jack/AP can hold. | Med | Many confirmed leases tied to one spot. |
| 4 | Go "low and slow": drip requests so burst alarms don't fire. | Easy–Med | Count two ways: a short, fast counter **and** a simple longer trend/score. | Easy | Gentle but steady trickle above normal for minutes. |
| 5 | Pretend to be many kinds of gadgets | Med | Know your "normal" fingerprints; flag too much | Med | Unusual diversity of device types |

| Round | Attacker's move (plain English) | Attacker difficulty | Defender's counter (plain English) | Defender difficulty | Simple signs to watch |
|---|---|---|---|---|---|
| | (different option sets) to look legit. | | variety from one jack/AP. | | from one place. |
| 6 | Spread the attack across many jacks/APs so per-port caps don't bite. | Med | Watch totals per room/floor/VLAN (not just per port); correlate across switches. | Med–Hard | Pool use rises everywhere at once, not just one port. |
| 7 | Say "that number is broken" (DECLINE) so the server shelves it. | Med | Rate-limit "broken" reports; verify before shelving lots of numbers; cut off noisy ports. | Med | Burst or drip of "broken" reports; usable pool shrinks fast. |
| 8 | Fake "someone already has this number" by answering checks (ARP spoof). | Med–Hard | Only accept "I own this number" from verified sources (ARP/DAI with bindings). | Med–Hard | Ownership claims that don't match real records. |
| 9 | Send odd-shaped or chopped-up messages to slip past simple filters. | Hard | Reassemble/clean traffic at the edge; count **events**, not raw packets. | Med–Hard | Lots of fragments; counts don't line up with real events. |
| 10 | Pretend to be a network helper/relay to bypass edge rules. | Hard | "DHCP guard": only switches may play helper; drop helper tags from endpoints. | Med | Helper tags seen coming from regular clients. |
| 11 | Act human: realistic pauses/retries and vendor-like behavior. | Hard | Look for impossible variety per jack and strange success ratios (asked→got). | Med | Many "first-time" devices from one jack; off-normal request→lease ratios. |
| 12 | Use many tiny helpers (bots/IoT), each staying under the radar. | Hard | Central view: alarms on whole-pool trends and many small blips at once. | Hard | Slow, wide drain of the pool across the site. |
| 13 | Keep numbers longer by renewing early/often so they don't return to the pile. | Med | Shorter edge leases; quick reclaim on idle ports; cap noisy renewers. | Med | Frequent renewals from one spot without real use. |
| 14 | Mix in fake servers to confuse clients. | Hard | Block untrusted servers at the edge (trust boundaries/snooping). | Med | Offers coming from places they shouldn't. |
| 15 | Send oversized/garbled forms to stress or crash the server. | Hard | Filter weird forms before the server; keep software patched. | Med | Drops on invalid forms; server CPU spikes. |
| 16 | Switch to the newer numbering system (IPv6) and flood that instead. | Med–Hard | Turn on the same protections for IPv6 (guards for v6). | Med | Sudden rise in IPv6 address activity/leases. |

Tiny glossary:

- “Wall jack/AP” = the specific port or Wi-Fi client side where a device connects.
- “Pool” = the set of free IP numbers the server can hand out.

Consider a student (new to internet security) wanting to create scripts for attacker and detector and improve them round by round. What would be the best 3 rounds of improvements which show that they understand what's happening, understand how an attacker thinks and in turn know how to modify/improve the detector? i.e, what 3 rounds of improvements would net them the best grade from the professor?

Awesome question. If you want a clean, high-impact 3-round story that proves you “get” DHCP starvation and defense—without getting lost in bells & whistles—use this arc:

| Round | Attacker improvement (what & why) | Detector counter (what & why) | Difficulty (A/D) | What to show your professor (evidence) |
|---|---|---|---|---|
| 1 | **Naive burst flood**: send many “please give me an IP” requests rapidly to empty the pool. Shows you understand the basic failure mode (rate-based exhaustion). | **Short-window burst guard**: count DISCOVERs/OFFERs per small interval and flag spikes; basic per-port rate limit. Shows you know the first, simplest control. | Easy / Easy | A tiny chart: requests/sec before vs during attack; highlight a burst threshold being crossed; pool free-address count dropping fast. |
| 2 | **Low-and-slow + identity churn**: drip requests (so no burst) and randomize client identity (e.g., different client IDs) so simple per-device counters don’t help. Demonstrates evasion thinking. | **Two-window logic + uniqueness check**: keep the fast counter **and** a longer-running score; also track the number of *distinct* client identities seen per port. Add a soft cap on active leases per port. Shows layered detection & resource protection. | Medium / Medium | Side-by-side plots: (a) short-window counter stays “green” while (b) long-window score slowly rises; bar showing “distinct identities per port” jumping on one jack; active leases per port capped. |
| 3 | **Distributed trickle**: run two small attackers (each under local limits) so totals only look abnormal when you aggregate—classic threshold evasion. Shows you understand attacks scale horizontally. | **Correlate & watch pool trend**: aggregate counts across ports/VLAN; alert on *sustained upward slope* of pool utilization + simultaneous small anomalies on multiple edges. This proves you know single-device views aren’t enough. | Medium–Hard / Medium–Hard | One page: per-port graphs look “fine”, but a **global** graph shows a steady climb toward exhaustion; simple table listing “ports with small anomalies at the same time.” Clear “caught by correlation” moment. |

## Why these 3 rounds earn top marks

- They tell a complete story: **basic failure → evasion → correlation-aware defense**.
- Each step introduces **one new attacker idea** and **one principled detector upgrade** (not random tweaks).
- The detector logic stays simple (counts, uniqueness, caps, aggregation) but shows real security thinking.

## Minimal, high-value metrics to log (no code here—just what to measure)

- Requests per second (short window) and a simple long-window score.
- Number of **distinct client identities** seen per port (e.g., how many "new devices" show up).
- Active leases per port and **total pool utilization** (used / total).
- The **slope** of pool utilization over time (is it steadily rising?).

## Presentation tips for an "A"

- One slide per round: the **attacker idea**, the **defender idea**, and a **tiny graph/table** proving it.
- Reuse your original code structure and only **add** minimal counters/fields; use descriptive variable names.
- State lab ethics clearly (isolated test network).
- End with a micro-rubric: what your detector catches (burst, low-and-slow, distributed) and the one thing it **doesn't** (honest limitation).

One more round?

Perfect—add this as **Round 4** to cap the story:

| Round | Attacker improvement (what & why) | Detector counter (what & why) | Difficulty (A/D) | What to show your professor (evidence) |
|---|---|---|---|---|
| 4 | **Hold-the-pool via renewals**: after claiming many addresses, keep them by **renewing early and often** so they don't return to the free pool. This shows you understand the *lifecycle* (not just bursts) and how exhaustion can persist without more DISCOVER floods. | **Lease-usage sanity + fast reclaim**: (1) track **renewals per port** and flag abnormal renew cadence; (2) use **shorter leases** at the edge for quick turnover; (3) **reclaim idle leases** when no real traffic is seen from that port for a while, and optionally **cap active leases per port**. This shows lifecycle-aware defense, not just rate limits. | Medium / Medium | A timeline where DISCOVER rate drops to normal, **but pool utilization stays high** (attack). Then, after enabling the countermeasures, show **renew spike alert**, **pool frees up** as idle leases are reclaimed, and **active-leases-per-port** returning to a normal cap. |

**Why this round is grade-winning:** it proves you grasp DHCP beyond the initial "spike"—you're modeling **persistence**, then answering it with simple, explainable controls (renew cadence checks, shorter leases, idle reclaim, per-port caps). It neatly completes the arc: **burst → low-and-slow → distributed → lifecycle persistence**.

Final round?

Love it—here's a clean **Round 5** that feels advanced but is still explainable in class, and it shows you understand trust boundaries (DHCP ↔ ARP).

| Round | Attacker improvement (what & why) | Detector counter (what & why) | Difficulty (A/D) | What to show your professor (evidence) |
|---|---|---|---|---|
| 5 | **"Poison the pool" with fake conflicts:** After the server offers addresses, the attacker reports them as "bad" (DECLINE) or fakes ownership via ARP so the server shelves them. Goal: reduce the number of usable addresses without obvious flooding. | **Validate conflicts + ARP protection:** (1) **Throttle/flag DECLINEs** per port and per minute; (2) **Verify conflicts** before shelving many addresses (e.g., require a successful probe, not just one report); (3) Turn on **ARP integrity checks** (only accept "I own this IP" if it matches the DHCP binding). Together, this stops fake "bad address" claims. | Med / Med–Hard | A short timeline: DISCOVER rates look normal, **but free-pool drops** while **DECLINE count ticks up**; ARP "I own this IP" claims appear from the same jack. After enabling the countermeasures: **DECLINE alert triggers**, ARP mismatches are blocked, and **free-pool rebounds**. |

**Why this is a strong final round**

- It shows you understand **not just rate games but protocol interplay** (DHCP + ARP) and **trust boundaries**.
- Your defense demonstrates **policy + sanity checks** rather than just more thresholds.
- The demo is visual: pool shrinking with "normal" traffic, then recovering once you enforce validation.