BrooskieHub
Cybersecurity Internship Report

Task 6 :DNS Spoofing Attack Simulation.

Submitted by: Namita Rana

Role: Cybersecurity Intern

Company: BrooskieHub

Date of Submission: 25 September 2025

Table of Contents

1. Introduction

1.1 Background of DNS

The Domain Name System (DNS) is a distributed, hierarchical naming service that resolves user-friendly domain names into numerical IP addresses required for routing traffic on the Internet. At its core are authoritative name servers that answer queries about specific zones and recursive resolvers that perform lookups on behalf of clients; caching is used aggressively to reduce latency and DNS traffic. Because DNS responses are trusted by clients and often accepted without cryptographic validation on most networks, DNS becomes a critical trust anchor — and therefore a high-value target — for attackers seeking to redirect or intercept traffic.

DNS resolution involves several steps — from client query to recursive resolver, to root and TLD servers, and finally to the authoritative server for a domain — and responses are cached at each stage according to TTL values. This caching improves performance but also creates windows where forged responses can persist and impact many users for the TTL duration. Understanding this architecture, the roles of different DNS components, and how caching works is essential to understanding why and how DNS spoofing attacks succeed and how their effects propagate.

1.2 Evolution of DNS Threats

DNS threats have progressed from straightforward misconfigurations and zone-transfer abuses to sophisticated attacks that exploit protocol quirks, resolver behavior, and trust assumptions. Early attacks often relied on default or weak administrative configurations; later techniques evolved to include cache-poisoning by predicting transaction IDs, blind spoofing, and the use of compromised recursive resolvers. In recent years attackers have combined DNS manipulation with social engineering, supply-chain compromise, and abuse of third-party DNS providers, creating complex multi-stage campaigns.

The introduction of privacy-enhancing technologies like DNS-over-HTTPS (DoH) and DNS-over-TLS (DoT) changed the operational landscape. While they protect client privacy and prevent eavesdropping, these protocols can reduce network visibility for defenders and complicate enterprise DNS monitoring, leading to different trade-offs in detection and mitigation. Consequently, defenders must consider both classical DNS hardening (e.g., rate limiting, source port randomization, DNSSEC) and how modern encrypted transport mechanisms affect detection and response.

1.3 Relevance of DNS Spoofing in Cybersecurity

DNS spoofing remains a high-impact, low-cost tactic in the attacker's toolkit because it can redirect legitimate user traffic to attacker-controlled infrastructure without needing to compromise the target's host or webserver. By hijacking DNS resolution, attackers can stage phishing pages, intercept credentials, inject malicious payloads, or route traffic through

command-and-control servers. For defenders, DNS-based attacks are difficult to detect if they mimic legitimate traffic patterns or exploit trusted internal resolvers.

Beyond immediate user-level redirection, DNS spoofing can be leveraged as part of larger offensive campaigns — enabling lateral movement, persistence, or exfiltration. For example, an attacker who controls DNS responses may redirect security updates to malicious repositories or intercept communications to cloud services. Therefore, learning to simulate and detect DNS spoofing is essential for incident responders, red teams, and security architects who need to create resilient DNS ecosystems and detection rules.

1.4 Real-World Examples of DNS Spoofing Attacks

Historically, DNS cache poisoning incidents have caused major outages and enabled large-scale phishing campaigns. Examples include targeted redirections against financial institutions that routed customers to counterfeit login portals, and state-level DNS manipulations used for censorship or surveillance. In some incidents attackers exploited compromised recursive resolvers or abused domain registrar access to effect large-scale redirection that affected millions of users.

Studying these real-world cases reveals common root causes — inadequate cryptographic validation, lax administrative controls, and the availability of vulnerable third-party services — and demonstrates the range of attacker goals from theft to censorship. These cases also highlight remedial practices adopted by defenders, such as moving authoritative DNS to well-controlled providers, deploying DNSSEC, and improving monitoring to detect anomalous TTLs or unexpected authoritative nameservers.

---

2. Objective

2.1 Primary Aim of the Simulation

The primary objective of the lab exercise is to reproduce, in a safe and controlled manner, a local-network DNS spoofing attack so learners can observe the attack lifecycle and its operational signatures. The lab illustrates how ARP spoofing or man-in-the-middle (MITM) positioning enables the attacker to intercept DNS queries and respond with forged answers, resulting in traffic redirection to a controlled webserver. Emphasis is placed on reproducibility, instrumentation (pcap capture, log correlation), and evidence collection.

An additional aim is to use the experiment as a vehicle for testing detection and mitigation strategies: implement monitoring rules, measure TTL effects, and see how caches retain poisoned entries. The simulation therefore serves both red-team (attack technique) and blue-team (detection & remediation) learning outcomes in a single controlled workflow.

2.2 Why Controlled Lab Environments Matter

Running DNS spoofing experiments on live networks can cause collateral damage, violate acceptable-use policies, and expose the experimenter to legal risks. An isolated virtual lab (host-only networks, internal VirtualBox networks) prevents spoofed traffic from leaking and ensures that other systems are neither disrupted nor inadvertently compromised. Labs also provide repeatable conditions for controlled comparisons — for example, changing TTLs or source port randomization to measure attack success rates.

Controlled labs enable the safe capture of forensic artifacts (full pcap files, application logs) and precise measurement of attacker and victim behaviour. By snapshotting VM states, experimenters can revert environments and test different defense configurations without cumulative contamination or persistent side effects.

2.3 Scope and Limitations of the Study

This study concentrates on local network-based approaches where the attacker can operate on the same L2 segment as the victim (or can otherwise intercept traffic), using ARP cache poisoning and DNS proxying to inject forged responses. It intentionally excludes attacks that require compromise of authoritative name servers, global BGP hijacking, or large-scale abuse of public DNS infrastructure. The focus is educational and bounded to lab-contained operations.

Limitations include the artificial homogeneity of VM environments versus heterogeneous production networks; certain real-world behaviors (e.g., ISP-level caching policies, enterprise DoH configurations, and mixed resolver chains) might not be fully represented. Despite that, the controlled experiments offer high pedagogical value for understanding the attack primitives and designing defenses.

---

3. Tools and Environment Setup

3.1 Overview of Tools

The core toolkit for this simulation typically includes: **Ettercap** (to perform ARP poisoning and passively/actively intercept traffic), **DNSChef** (a configurable DNS proxy to serve forged DNS responses), a lightweight **HTTP server** (to host a benign demonstration page), and **VirtualBox** (to create and isolate attacker and victim VMs). Supplementary tools include **tcpdump / Wireshark** for packet capture, **dig / nslookup** for query testing, and text editors for configuration.

Each tool plays a specialized role: Ettercap provides network-layer interception capabilities; DNSChef manipulates application-layer DNS responses; the HTTP server proves the end-to-end redirection; and the capture tools provide the raw evidence required for analysis. Alternative toolchains (e.g., arpspoof + socat, mitmproxy with DNS plugins) are available and can be used to explore different techniques.

3.2 Installing VirtualBox (and host prep)

Install VirtualBox using the official binaries for Windows, macOS, or Linux and, if necessary, add the Extension Pack for additional networking features. Ensure that virtualization is enabled in BIOS/UEFI. Prepare the host by ensuring sufficient disk space and memory for multiple VMs; create snapshots or templates to avoid redoing base setups.

When creating VMs, allocate network adapters configured as host-only or internal networks rather than bridged adapters. This prevents the lab traffic from touching the host's external network. Optionally, use a dedicated test host or an isolated VLAN on lab hardware for higher-fidelity experiments.

3.3 Configuring an Isolated Network (host-only / NAT)

Create an internal network in VirtualBox (e.g., host-only adapter) and assign static IPs or run a small DHCP server within the lab. Static addressing simplifies packet correlation and documentation. Ensure no NAT or bridging that might forward traffic to the internet unless explicitly required and authorized.

Document the IP addressing scheme, DNS resolver IPs, and network topology. Include details such as subnet masks, gateway (if any), and the attacker/victim IP addresses to make reproducing the experiment straightforward.

3.4 Setting up the Attacker Machine (packages & hardening)

Deploy a Linux-based security distro (Kali, Parrot, or Ubuntu) for the attacker VM. Update packages and install Ettercap, DNSChef (or equivalent), tcpdump, and a simple HTTP server (e.g., Python's http.server or Nginx). Configure the attacker VM to listen only on the lab network and disable unnecessary services to reduce noise in captures.

Create directories for storing pcaps, logs, and configuration files. Enable IP forwarding only when needed, and consider restricting outbound connections at the hypervisor level. Record exact package versions and command arguments to ensure reproducibility and traceability.

3.5 Setting up the Victim Machine (baseline config)

Use a standard desktop VM (Windows, Ubuntu, etc.) to simulate a user environment. Configure a default browser, accept default DNS settings, and verify that normal browsing behaviour works within the lab environment (i.e., resolving and loading hosted pages that simulate public web content). Capture a baseline pcap and DNS logs from the victim to record normal resolution behaviour.

Include steps for clearing DNS cache and browser cache to test the impact of caching on spoofing persistence. Save baseline captures in an organized file-naming scheme prefixed with “baseline” and date/time stamps.

3.6 Security Considerations and Logging

Implement strict lab containment: disable host-level bridging, use firewall rules to block traffic to public networks, and consider ephemeral lab networks that are destroyed after tests. Centralize logs and pcaps into a secure location; apply access controls so that only authorized personnel can view sensitive artifacts. If sensitive data ever appears in captures, follow data-protection and sanitization policies.

Use consistent logging formats and include metadata (test ID, operator, datetime, VM snapshots used) to aid future audits or reproductions. Securely store any credentials used solely for the lab and destroy them when no longer needed.

---

4. Attack Procedure

4.1 Initial Lab Verification & Baseline Capture

Before performing any attack actions, document and capture the normal state of the network: run dig from the victim to targeted test domains, capture DNS responses, and verify that the HTTP server returns expected content when accessed legitimately. Collect baseline pcaps and configuration snapshots so you can compare pre- and post-attack behavior. Record timestamps and environment variables (e.g., TTLs observed in responses).

Perform smoke tests for all tools (Ettercap can run in sniff-only mode) and ensure packet capture is functioning at both attacker and victim endpoints. Confirm that time synchronization between VMs is accurate to make cross-log correlation simpler.

4.2 Launching ARP Spoofing with Ettercap (mechanics)

Use Ettercap to poison the ARP cache of the victim and the gateway (or the resolver) to position the attacker in the data path. The attacker will then forward traffic (so as not to disrupt connectivity) while inspecting and selectively manipulating DNS queries and replies. Carefully monitor ARP tables on both victim and attacker to confirm successful poisoning, and log the exact commands and filters used.

Be mindful of the disruptive nature of ARP poisoning: if multiple hosts share the same segment, the attack could affect more machines than intended. This reinforces the need for an isolated lab. Document the sequence in which ARP entries changed and the duration of poisoning for forensic completeness.

4.3 Configuring DNSChef for Fake Responses (mapping logic)

DNSChef should be configured with a mapping file (or CLI mappings) that instructs the proxy to return a specific IP for the targeted hostname(s) while allowing all other domains to be resolved normally (forwarded). Test the mapping locally on the attacker machine using dig @127.0.0.1 <target> to validate DNSChef behaviour prior to launching the full attack. When mapping, pay attention to TTL values and DNS transaction IDs to observe how caches are affected.

Log DNSChef output to a file so each query and response pair is recorded. This logging is crucial for demonstrating the exact moment when forged responses were injected and the query/response pair that led to victim redirection.

4.4 Redirecting to a Malicious Webpage (web hosting & response)

Host an instructional, non-malicious demo webpage on the attacker machine that clearly indicates it is a test (“Demo — Spoofed Site”). Configure the HTTP server to respond to requests for the domain(s) being spoofed. When the victim requests the target domain, the

forged DNS response will direct the browser to the attacker-hosted page, demonstrating end-to-end redirection.

Take care to avoid any content that could be mistaken for real phishing or malware. Use conspicuous banners and text indicating the site is for demonstration only. Capture browser network logs and screenshots showing the loaded page and the URL bar to form unambiguous evidence.

4.5 Monitoring Traffic and Logs (pcap analysis workflow)

Continuously capture packets on the attacker and victim interfaces using tcpdump/Wireshark; label captures with timestamps and metadata. After the test, analyze the pcap for DNS query/response pairs, ARP poisoning frames, and subsequent TCP sessions to the hosted webserver. Highlight artifacts such as mismatched transaction IDs, forged source IPs, and deviations in expected TTLs.

Create annotated extracts (screenshots or exported packet views) that show the precise fields proving the injection (e.g., DNS answer section showing the attacker IP, ARP reply frames with spoofed MACs). Correlate these artifacts with application logs and timestamps to build a credible timeline.

4.6 Verifying the Attack from Victim's Perspective (browser, cache)

On the victim VM, perform the user-facing validation: visit the targeted domain, observe redirection to the demo page, capture screenshots of the browser, and collect DNS cache entries to show the poisoned records. Clear browser and resolver caches and repeat queries to demonstrate whether the spoofed state persists or if normal resolution is restored, thereby illustrating caching effects and TTL behavior.

If the victim's resolver caches the forged entry, document the TTL and time-to-expiration to show persistence. Repeat the experiment with different TTLs on the forged replies to measure practical impacts and observe how resolver behavior differs across OSes or resolver implementations.

---

5. Hints and Best Practices

5.1 Importance of Isolation and Containment

Always segregate lab networks from operational networks. Use host-only interfaces and explicit firewall rules to enforce containment. If experiments must reach the internet (rarely required), perform them only with explicitly authorized test infrastructure and narrow scopes. Clean up artifacts and destroy test networks when finished to avoid accidental reuse.

Containment reduces legal, reputational, and operational risk and simplifies incident response if something behaves unexpectedly. It also makes analysis simpler because the traffic is limited to known hosts.

5.2 Ethical Considerations and Legal Compliance

Be explicit about scope, authorization, and purpose before conducting any offensive-security exercise. Obtain written approvals from supervisors or lab owners and ensure participants understand boundaries and emergency stop procedures. Keep experiments strictly within educational or authorized research contexts and never engage with live external targets.

Respect data privacy by avoiding the capture of real user data and sanitizing any captures that contain sensitive information prior to sharing. Maintain an ethical mindset: research to improve defenses, not to exploit vulnerabilities.

5.3 Avoiding Real-World Harm (test domains, sanitization)

Use non-routable or explicitly designated test domains (for example, subdomains of example.com reserved for documentation) to avoid interfering with legitimate services. Verify that DNS responses cannot leak to public resolvers and review captures for accidental egress. If any artifacts contain real-world data, follow your organization’s data-handling and sanitization policies before distribution.

Consider adding explicit “demo” watermarks on hosted pages used in the lab so that even accidental exposure is less likely to be mistaken for malicious activity.

5.4 Documentation Best Practices & Versioning

Document both the technical steps and the rationale behind them: why a particular TTL was chosen, what assumptions about resolver behavior were made, and what detection signatures were expected. Use version control for scripts and configs and maintain a changelog for experiments. Label captures with test IDs and store them in secure, access-controlled storage.

Good documentation shortens onboarding time for new analysts and provides the institutional memory needed to replicate or audit experiments months later.

---

6. Outcome and Learning

6.1 Hands-On Skills Gained and Competency Goals

Participants will learn to provision isolated labs, operate ARP poisoning tools, configure DNS proxies for targeted domain spoofing, host benign demo content, and collect and interpret forensic packet captures. These hands-on skills translate to better incident response, stronger red-team capabilities, and deeper understanding of how DNS weaknesses are exploited and detected.

Additionally, learners will gain procedural discipline: documenting and packaging artifacts, following ethical constraints, and communicating technical findings to non-technical stakeholders.

6.2 Understanding DNS Attack Vectors & Limitations

The lab clarifies multiple vectors and constraints: local ARP spoofing is highly effective on permissive LANs but requires L2 access; cache poisoning success depends on resolver implementations, transaction ID randomness, and TTL management. Participants will understand both the power and limitations of DNS attacks and will be better equipped to evaluate attacker feasibility in different environments.

This knowledge helps prioritize defensive measures: which vulnerabilities can be mitigated with configuration changes, and which require architectural changes (e.g., DNSSEC adoption or removing untrusted recursive resolvers).

6.3 Defensive Measures & Detection Opportunities

Defensive takeaways include the importance of DNSSEC for data origin authentication, limiting the use of open recursive resolvers, enforcing proper resolver configuration (source port randomization, query rate limiting), and instrumenting DNS telemetry for anomaly detection. From a network perspective, host-based protections and ARP table monitoring can detect suspicious MITM attempts; network segmentation limits attacker reach.

Participants should practice writing detection rules for SIEMs and IDS/IPS systems using the forensic artifacts produced — for example, detecting abrupt changes in authoritative answers, surges in NXDOMAIN responses, or the presence of internal hosts returning external authoritative records.

6.4 Long-Term Relevance, Maintenance and Testing Cadence

DNS remains a foundational Internet service; although tools evolve, the trust model and caching mechanics persist. This lab therefore provides durable learning outcomes. Organizations should establish periodic testing cadences — quarterly lab exercises, red-team campaigns, and automated DNS checks — to ensure that defenses keep pace with evolving attack techniques and to validate configuration drift does not reintroduce vulnerabilities.

Maintain a playbook for DNS incidents that is updated with lessons from each exercise, including detection signatures, containment steps, and communication templates for stakeholders.

---

7. Conclusion

7.1 Summary of Findings

The controlled DNS spoofing simulation demonstrates how local network manipulation combined with a configurable DNS proxy can redirect users to attacker-controlled destinations with minimal complexity. The experiment shows clear signals in ARP frames, DNS query/response pairs, and browser behaviour that can be used both to validate attacks and to construct detection/response playbooks.

While the attack is straightforward in permissive environments, its success and persistence depend on resolver behaviour and caching, underscoring the need for layered defenses and active monitoring.

7.2 Tactical & Strategic Recommendations for Practitioners

Tactically, enable DNSSEC where possible, restrict access to resolvers, enforce strong resolver configuration (randomized source ports, strict TTLs), and deploy DNS telemetry for anomaly detection. Strategically, invest in training and recurring lab exercises, maintain an incident response playbook that includes DNS-specific containment steps, and periodically audit third-party DNS services for misconfiguration or compromise risk.

Also, include business stakeholders in tabletop exercises so the organization can plan legal, operational, and customer-communication aspects of a DNS-impacting incident.

7.3 Future Research Directions and Appendices to Consider

Future work could explore the interaction between DoH/DoT deployments and enterprise detection capabilities, simulate attacks on diverse resolver implementations to understand real-world variability, and evaluate machine-learning approaches for detecting anomalous DNS patterns. Appendices that can add value to the report include: a reproducible script set, sanitized pcap samples with annotations, command-line examples for Ettercap/DNSChef, and a checklist for safe lab teardown.
