



DNS Spoofing and Cache Poisoning: Threats and Mitigation Techniques in Enterprise Environments

by
Emdad Hossain

Supervisor: Dr Suresh Shanmugasundaram

Project submitted as part of the requirements for the award of
MSc in Cyber Security and Digital Forensics

February 2026

Declaration and Form of Consent:

DECLARATION:

I, **Emdad Hossain** declare that I am the sole author of this Project; that all references cited have been consulted; that I have conducted all work of which this is a record, and that the finished work lies within the prescribed word limits.

This work has not previously been accepted as part of any other degree submission.

Signed : Emdad Hossain

Date : 01/02/2026

FORM OF CONSENT:

I Emdad Hossain hereby consent that my Project, submitted in candidature for the Cyber Security and Digital Forensics, if successful, may be made available for inter-library loan or photocopying (subject to the law of copyright), and that the title and abstract may be made available to outside organisations.

Signed: Emdad Hossain

Date: 01/02/2026

Abstract

This dissertation investigates how weaknesses in Domain Name System (DNS) trust assumptions can enable spoofing and cache poisoning, and examines what practical controls can be evaluated safely under academic and ethical constraints. An evidence-led approach was adopted that combines (i) a focused review of recent DNS security literature and enterprise threat considerations, and (ii) the design and implementation of a reproducible DNS laboratory environment intended to make integrity and governance concepts observable without exploit development. A controlled, containerised testbed was built using lightweight components and role separation, orchestrated with Docker Compose. The environment supports internal name resolution, controlled external forwarding, and a security-enhanced configuration that demonstrates policy enforcement (domain sinkholing), visibility (query logging), and containment (network isolation). Testing used repeatable query sequences and multi-source evidence (dig outputs, resolver logs, and configuration artefacts) to assess functional correctness, operational stability, performance overhead, and practicality. Results confirm that the platform reliably reproduces baseline DNS behaviour while allowing security-relevant outcomes to be demonstrated in a safe sandbox. Internal cached lookups remained consistently low-latency (sub-50 ms), forwarded external lookups averaged roughly 100–200 ms, and resource use stayed minimal (under ~50 MB RAM per container). Policy controls consistently redirected a designated malicious domain to a sinkhole address (127.0.0.1), while logging provided traceable, timestamped transaction visibility suitable for instructional analysis. Key limitations are acknowledged: the simplified topology and entry-level controls do not exercise full cache-poisoning scenarios, DNSSEC validation, or encrypted DNS transports. The dissertation therefore contributes a repeatable foundation that can be extended with staged complexity, safe attack simulation, and comparative resolver implementations in future work.

Contents

Chapter 1: Introduction	8
1.1 Context and Motivation	8
1.2 Broad Problem Statement	8
1.3 Refined Problem Definition	9
1.4 Aim	10
1.5 Objectives	10
1.6 Research Questions	10
1.7 Scope	10
1.8 Significance and Contribution	11
1.9 Structure of the Report	12
Chapter 2: Literature Review	13
2.1 DNS Fundamentals and Architectural Principles	14
2.2 Definitions and Concepts: Spoofing and Cache Poisoning	16
2.3 Attack Models, Techniques, and Feasibility Factors	16
2.4 Enterprise DNS Threat Landscape	17
2.5 Existing Defence Mechanisms and Security Enhancements	18
2.8 Synthesis: Gaps, Contradictions, and Justification for This Project	20
2.9 Summary	20
Chapter 3: Research Methodology	21
3.1 Research Philosophy	21
3.2 Research Approach	21
3.3 Research Strategy	22
3.4 Hypotheses	22
3.5 Data Sources and Collection	23
3.6 Measurement Plan and Variables	23
3.7 Analysis Methods	24

3.8 Validity and Reliability	24
3.9 Ethics and Safety	24
3.10 Summary	25
Chapter 4: Design Process	26
4.1 Chapter Overview and Design Goals	26
4.2 Design Requirements and Constraints	26
4.2.1 Functional Requirements	26
4.2.2 Non-Functional Requirements	27
4.3 Testbed Architecture Design	27
4.3.1 High-Level Network Topology	27
4.3.2 Logical Roles and Trust Boundaries	28
4.4 DNS Server Configuration Design	29
4.4.1 Baseline Configuration	29
4.4.2 Security-Enhanced Configuration	29
4.5 Testing Methodology Design	30
4.5.1 Test Categories	30
4.5.2 Test Execution Steps	31
4.6 Measurement and Evidence Collection	32
4.6.1 Success Metrics	32
4.6.2 Evidence Types	32
4.7 Tools and Platform Selection	32
4.7.1 Containerization Platform	32
4.7.2 DNS Software Selection	33
4.8 Validation and Verification Approach	33
4.8.1 Design Validation	33
4.8.2 Implementation Verification	33
4.9 Summary	34

Chapter 5: Implementation	35
5.1 Chapter Overview	35
5.2 Environment and Platform Setup	38
5.3 Container Provisioning and Role Mapping	38
5.4 DNS Server Configuration	39
5.5 Client Configuration and Testing Setup	40
5.6 Testing and Validation Procedures	41
5.7 Evidence Collection and Results	42
5.8 Implementation Challenges and Solutions	43
5.9 Summary of Implementation	45
Chapter 6: Evaluation Of Results	46
6.1 Evaluation Criteria and Evidence Sources	46
6.2 Testbed Composition and Build Consistency	46
6.3 Network Segmentation and Interface Configuration	47
6.4 Functional DNS Results from Query Testing	48
6.5 Security Control Evaluation Outcomes	49
6.6 Performance and Practicality Assessment	49
6.7 Limitations and Scope Acknowledgement	50
6.8 Overall Evaluation Judgement	50
Chapter 7: Conclusions And Recommendations	51
7.1 Chapter Overview	51
7.2 Summary of Implemented Work	51
7.3 Research Question Conclusions	52
7.4 Practical Implementation Recommendations	53
7.5 Future Development Pathways	53
7.6 Concluding Synthesis	54
Chapter 8: Reflection	56

8.1 Description	56
8.2 Feelings	56
8.3 Evaluation	56
8.4 Analysis	57
8.5 Conclusion	57
8.6 Action plan	57
References	59

Chapter 1: Introduction

1.1 Context and Motivation

Domain Name System (DNS) is a basic Internet service and a key dependency in contemporary enterprise networks. It allows mapping human-readable domain names to machine-readable addresses that are an essential part of accessing web applications, email, cloud platforms, identity services, and internal business systems [1]. DNS supports numerous organisational processes, and its loss of integrity can result in severe operational and financial damage. Altered DNS responses can either redirect users to dangerous services, deny them needed services, or interrupt authentication, which has a direct impact on confidentiality, availability, and business continuity.

The strategic value of DNS has been increasing due to the increased use of resolution pathways by attackers. According to industry research, in addition to being a common integrity attack target, DNS has long been a useful source of security telemetry to detect malware activity and advanced persistent threats [2]. Such a two-fold role also makes DNS a highly important dependency and a high-value monitoring point. But numerous enterprise implementations continue to be based upon outdated assumptions, lax validation routines, and informal trust strategies that were never intended to oppose cutting-edge threat proficiencies [3].

Enterprise setups only enhance DNS risks due to hybrid setups that combine both on-premises systems and cloud applications with third-party DNS providers. The existence of such complicated dependencies enlarges the attack surface, and it becomes more dependent on middlemen. Also, remote working and BYOD practises imply that devices frequently do DNS resolution on networks that cannot be trusted, and in such networks, interception or manipulation is easier.

Irrespective of the available protection, there are still integrity risks. DNSSEC usage remains ad-hoc since it is complex to operate, and encrypted DNS protocols like DoH and DoT give visibility and control issues to businesses [4]. As a result, DNS is also very sensitive but poorly managed, and integrity assurance has become a persistent security issue.

1.2 Broad Problem Statement

Businesses assume that the response of DNS is trustworthy and can be used to provide operational and security controls. The main systems, like firewalls, proxies, load balancers, identity platforms, and endpoint agents, often rely on domain names in the implementation of

policies. As a result, the attackers who interfere with DNS records can circumvent several stages of defence. DNS spoofing and cache poisoning utilise this dependency to place forged responses into the resolution process, which can result in the redirection of traffic, theft of credentials, interception of communications and the delivery of malware with limited immediate notice [3].

Industry tests will categorise such attacks as high-impact threats as they affect the underlying layer of trust on enterprise networks [2]. Poisoning the cache is especially harmful because it means that the rogue recursive resolvers send poisoned data to a high number of clients at the same time. Forward chains, conditional forwarding rules and the use of external / cloud-based DNS services, which all provide an extra layer of trust, add to the risk.

The main issue debated by this dissertation is the challenge of maintaining DNS integrity, as well as performance and operational stability in enterprises. Security protocols like DNSSEC and the encrypted DNS protocol can eliminate the risk but can cause service interruption, validation, or less internal visibility when not managed properly [4].

1.3 Refined Problem Definition

This paper is concerned with architectural, operational, and configuration issues in enterprise DNS infrastructures that contribute to vulnerability to spoofing and cache poisoning. Recursive resolvers are essential due to the fact that they sort queries and fill common caches so that a modified reply can spread far and wide [1]. Upstream provider-internal resolver forwarding relationships are also some of the essential boundaries of trust, especially in hybrid or cloud-integrated configurations [5].

Split-horizon configurations provide more flexibility but increase the risk of misconfiguration. Internal DNS zone management is becoming more complex. Zone information, permission of transfer, severities of roles and errors may reveal resolution paths to manipulation, particularly in legacy systems or patching inconsistency [3].

Mitigation is also complicated due to issues such as operational constraints. Companies need low-latency and high-availability DNS products, whilst controls, including DNSSEC, add complexity and overhead that do not generally support their use [6]. The increased adoption of DoH and DoT creates governance issues because the endpoints can bypass internal resolvers, making them less visible and decreasing centralised monitoring [4].

1.4 Aim

This dissertation aims to evaluate how enterprise DNS architectures are exposed to spoofing and cache poisoning risks, and to determine the most effective and practical mitigation set under realistic operational constraints.

1.5 Objectives

To achieve this aim, the project pursues the following objectives:

Objective 1: To critically review DNS spoofing and cache poisoning threats and the defensive control families proposed in prior research, identifying the evaluation criteria used for assessing their effectiveness.

Objective 2: To design and implement an enterprise-like DNS testbed without writing custom code, applying selected mitigations so that measurable security and performance data can be collected.

Objective 3: To analyse the collected evidence using defined metrics, including security effectiveness, latency overhead, stability, and operational complexity, to recommend robust and practical mitigation strategies for enterprise use.

1.6 Research Questions

The research questions guiding this dissertation are as follows:

RQ1: Which architectural and operational conditions most influence an enterprise's susceptibility to DNS spoofing and cache poisoning?

RQ2: Which mitigation techniques most effectively reduce DNS integrity risks in an enterprise-like environment, and under what deployment assumptions?

RQ3: What trade-offs relating to performance, visibility, governance, and operational complexity must enterprises manage when implementing DNS integrity controls?

1.7 Scope

The area of this study is the assessment of DNS integrity in enterprise networks with a specific view of aspects that have the most significant impact on exposure to spoofing and cache poisoning. Recursive resolvers are given primary consideration since they decide on client-facing resolvers' behaviour and allocate responses based on the cache over the network. This level of compromise can thus impact many enterprise systems at once [1]. It also studies forwarding paths between internal resolvers and upstream servers that are usually out of organisational control and impose trust relationships that can allow forged responses to disseminate when they are not well managed [5]. In some cases, such as split-horizon

deployments, internal DNS zone configurations are also given consideration since they are prone to misconfiguration, especially when using legacy systems or poor management practises [3].

It also covers mitigation techniques applicable in an enterprise setting, including resolver validation, hardened configuration, policy filtering, and anomalous behaviour detection using DNS-specific monitoring techniques [2]. Each of them is assessed by quantifiable standards such as security effectiveness, performance impact, and operational feasibility to conditional practical relevance.

Ethical and academic limitations prohibit offensive experimentation and live attacks against production systems [7]. All testing is done in a controlled lab that closely resembles enterprise conditions without interfering with actual networks.

1.8 Significance and Contribution

The importance of the study to both practice operators and scholars is that DNS security is an important but under-researched issue of operation in the enterprise. Despite the existence of various defensive mechanisms, organisations tend to fail to execute and maintain them properly because of performance, compatibility, governance and visibility factors. This dissertation offers both a real-world understanding of the effectiveness of mitigation strategies in real-life enterprise situations and their disadvantages and advantages. It is relevant to professional practise as it provides an evidence-based comparison of DNS integrity controls based on systematic tests in a controlled setting.

The scholarly input is the introduction of an evaluation strategy that does not necessitate exploit development or live attack execution. Rather, the research is based on controlled configuration manipulation and organised observation to determine the effect of mitigation measures on integrity and performance. The strategy provides a repeatable model within the future study concerning further controls, extended deployments or new DNS-based technologies like encrypted transport infrastructure and anomaly detection systems. The dissertation helps create more robust enterprise DNS infrastructures by filling practical and academic gaps at the same time.

1.9 Structure of the Report

Chapter	Title	Brief Description
2	Literature Review	Reviews DNS fundamentals, spoofing and cache poisoning threats, and existing mitigation techniques, identifying gaps that motivate the experimental design.
3	Research Methodology	Describes the research philosophy, approach, strategy, data collection methods and analysis techniques used to answer the research questions.
4	Design / Experiment Design	Specifies the enterprise-like DNS testbed architecture, configuration assumptions, mitigation set and evaluation metrics.
5	Implementation	Details the practical deployment of the testbed and mitigations, presenting configuration evidence and documenting challenges encountered.
6	Evaluation of Results	Presents and analyses the results obtained from the testbed, comparing baseline and mitigation scenarios against defined metrics.
7	Conclusions and Recommendations	Summarises the main findings, relates them to the research questions and provides recommendations and suggestions for future work.
8	Reflection	Reflects on the researcher's learning, skills development and project management across the lifecycle of the dissertation.

Table 1: Structure of the Dissertation

Chapter 2: Literature Review

This chapter is a review of the main notions, standards and studies connected with Domain Name System (DNS), DNS spoofing, cache poisoning, and the methods of mitigation and prevention in enterprises. The discussion lays the conceptual and technical basis for the testbed and test presented later in this dissertation. It starts with DNS basics and an introduction to architecture, and then follows the path of development of DNS vulnerability and cache-poisoning attacks. The main notions of spoofing, poisoning and attacker model are defined and critically compared. The chapter continues by looking at the DNS threat landscape that enterprises have to deal with specifically, and then goes through the existing defence families: DNSSEC, resolver hardening, network controls, policy-based protections, and monitoring. The effect of encrypted DNS (DoH/DoT/ODoH) is discussed, and the methods of testing and assessment applied in the previous research on DNS security are examined. The chapter has ended with attempts to synthesise gaps and contradictions in the literature and demonstrate how they will justify the aims, objectives, and methodological choices of this project.

This chapter examines both academic and industry literature related to Domain Name System (DNS) vulnerabilities and security mechanisms. The aim is to establish an understanding of existing research, highlight persistent security challenges, and identify gaps that require further investigation. The review focuses on publications released between 2020 and 2026, drawing on recent empirical studies, technical analyses, and survey papers.

The literature places particular emphasis on DNS-related vulnerabilities, defensive techniques, real-world attack incidents, and the practical difficulties organisations face when deploying DNS security solutions. DNS security has received increasing attention due to its involvement in multiple high-profile cyber incidents. Existing research demonstrates that adversaries frequently exploit DNS as a vector for data exfiltration, traffic redirection, and denial-of-service attacks. A recurring finding across studies is that many organisations lack effective detection mechanisms, allowing DNS to function as a covert communication channel for attackers.

To enable meaningful comparison, the review is organised thematically, synthesising key findings, perspectives, and conclusions across the selected sources.

Inclusion Criteria

- Peer-reviewed academic publications and reputable industry reports published between 2020 and 2026
- Studies addressing DNS vulnerabilities, mitigation strategies, implementation challenges, or relevant policy considerations
- Sources offering detailed technical analysis of attack techniques, such as DNS spoofing and tunnelling, or security controls, including DNSSEC and DNS over HTTPS (DoH)

Exclusion Criteria

- Non-peer-reviewed or low-credibility sources, including personal blogs and unsupported opinion articles
- Publications released prior to 2020, except where foundational concepts are required for contextual understanding
- Studies that do not directly focus on DNS security or that lack sufficient technical or empirical depth

2.1 DNS Fundamentals and Architectural Principles

Mockapetris writes that the Domain Name System (DNS) was intended to be a hierarchical and distributed naming system to replace the use of static host files, which failed to scale with the explosive growth of the Internet [8]. DNS namespace is organised in the form of a tree where the point of origin is the dot and authority is delegated to zones, which are served by authoritative name servers that serve particular branches or zones within the tree [8]. Resource records (RRs) contain mappings between domain names and related information represented by A records, AAAA records, NS records and MX records. Resolvers follow referrals of the root and top-level domain to the right authoritative servers [9]. Although this architecture allows scalability and administrative independence, dependency is also created on every delegation point, i.e. misconfigurations or attacks may influence downstream resolution.

DNS helps in supporting iterative and recursive resolution models. In iterative resolution, a server responds with referrals to its client, and in recursive resolution, a resolver does all the required lookups on behalf of a client and makes a final response [8]. Recursive resolvers provided by enterprises, Internet service providers or publicly accessible platforms are typically used by most current clients and centralise recursive resolvers of large groups of

users [4]. Caching enhances performance in the sense that it will reduce the latency and decrease the load on the authoritative server, as time-to-live (TTL) values are used to decide the duration of storage of records. Negative caching also enables a resolver to retain non-existent names for a duration [10]. Nonetheless, this centralisation poses a severe threat: when an attacker inserts forged data into a resolver cache, all the dependent clients will be redirected until the entry expires.

The application of enterprise DNS is usually far beyond mere resolution models. According to the NIST guidance, forwarding chains, split-horizon DNS and multiplexing internal zones are often used in organisations to maintain complex infrastructures [11]. Forwarding settings establish a dependency of internal resolvers with upstream servers and send unresolved queries instead of an iterative, fully resolved query [12]. Split-horizon DNS offers distinct answers to internal and external customers of the same name, which facilitates architectural and security objectives and raises the risk of misconfiguration [11].

The old DNS design focused on strength and performance, as opposed to high degrees of authenticity. The initial requirements presuppose friendly interaction, and the user does not have any cryptographic binding between a query and its answer, but only identifiers of transactions and inherent trust over syntactically valid answers [8], [9]. Subsequent literature recognises the fact that DNS was never intended to be used in an adversarial environment, and there has been a series of resilience enhancements introduced incrementally [13]. The modern DNS needs to be, therefore, used with tougher security concerns whilst preserving backward compatibility and performance, a balance that has spawned current dangers of spoofing and cache poisoning [14].

Initial studies of DNS security proposed that response and query identifiers are compared with minimal identifiers, where off-path attackers predict transaction parameters and inject forged responses [13]. Hubert and van Mook [13] demonstrated that predictable transaction IDs, fixed source ports, as well as weak bailiwick checking, enabled early resolvers to be incredibly vulnerable to brute-force cache poisoning. With the growth in the worth of the traffic hijacking, the real-life examples revealed that resolvers could be corrupted so as to store rogue records [15]. More sophisticated methods were shown when future researchers exploited methods like birthday-paradox, to boost guessing success [15], poisoning by vulnerable DNS forwarders [12], and, to check the state of switching resolve to circumvent modern renewals [16].

2.2 Definitions and Concepts: Spoofing and Cache Poisoning

DNS spoofing can be defined as any attack whereby an attacker can spoof DNS replies as valid, such that a resolver or client treats them as legitimate data [17]. DNS cache poisoning is a spoofing variant that targets to add spoofed records into the cache of a resolver, which will replace subsequent queries with malicious ones [17]. After a cache entry has been poisoned, users are rerouted to phishing sites or malware servers without any notification that they have been compromised [5], [18].

The Cache poisoning attacks the resolvers' matching mechanisms of outstanding queries. Every DNS query has a transaction ID and usually a source port; when a forged reply is received with the same message and parameter set, and it also contains a valid query section, lots of resolvers will accept it and store it [9]. Attackers thus seek to foreshadow or intuit these values and provide fake responses prior to legitimate response receipt [19]. In that resolvers normally store other records besides the response being asked, a successful injection will corrupt many records in the cache, which greatly increases the effects of the attack [19].

One can single out several targets of poisoning. The recursive resolvers are directly affected by resolver poisoning, as it has high leverage (most clients) [15]. In forwarder poisoning, the devices that send queries to resolvers that are upstream of those being attacked are attacked, so forged data can be propagated downstream when forwarding behaviour is loosely-checked [12]. On-path adversaries perform attacks through an observation of the traffic, which they may manipulate, or off-path adversaries viewing the traffic, and making an inference of the resolver state based on guessing, or side channels [15], [16].

Companies are especially susceptible due to the fact that DNS supports Active Directory, Kerberos and email authentication. Intrusion into internal DNS records is thus able to interfere with authentication procedures and direct traffic of sensitive information across the system [11], [18], across the systems together.

2.3 Attack Models, Techniques, and Feasibility Factors

Characterisations of attack models, Attacker position, control over network components, and resolver behaviour knowledge can characterise attack models of DNS cache poisoning. Son and Shmatikov [15] examine an off-path attacker that is not able to monitor authentic traffic, but rather sends spoofed UDP replies to a resolver. Under this model, the attacker sends multiple counterfeit responses and tries to uncover transaction identifiers and other parameters and the authentic response is obtained by the attacker. It is found to be successful

with respect to the size of the identifier space, number of parallel queries and timing window between query and response.

There is a considerable decrease in this risk by adding more entropy. According to Hubert and van Mook [13], brute-force poisoning is very impractical due to source-port randomisation, better validation of records and increased vigilance in handling other records. But subsequent practise reveals that real-world applications tend to refute these assumptions. As shown by Man et al. [16], forwarders and middleboxes have side channels, e.g. timing variability or error reports, that can inform about resolver state and decrease the effective search space. Zheng et al. [12] also demonstrate that the DNS forwarders are often left out of threat models, even though they sometimes use less randomisation, or simpler matching logic, than dedicated resolvers.

Policy, policy and resolver configuration also have a bearing on feasibility. Open resolvers, which can be queried by arbitrary clients, are more open than closed resolvers, which can only accept queries to an internal source [19]. Lax bailiwick checking may enable attackers to add irrelevant records to caches with real responses [11], [19]. Moreover, the vulnerabilities revealed in popular software, like BIND, are some of the vulnerabilities that can suddenly cause exposure once fixed [20].

Although it has been argued that modern defences preclude off-path poisoning because of modern defences [13], it has been demonstrated that off-path poisoning survives because of side channels, old devices, and configuration issues [12], [16], [21]. Heterogeneous, poorly written DNS infrastructures in enterprise settings are especially prone to attack [11], [17].

2.4 Enterprise DNS Threat Landscape

Enterprise DNS supports the key operations such as web requests, email rerouting, VPN connections, directory searches, SaaS integration, and microservice discovery [11], [17]. According to Chandramouli et al. [11], authentication systems like Active Directory and Kerberos depend greatly on DNS to identify domain controllers and service endpoints by ensuring that a wrong functioning system breaks down identity and access control directly. DNS in hybrid architectures can also be used in cloud service discovery and in traffic steering, resulting in it as a fundamental functionality requirement [14].

Enterprise DNS risk is due to the complexity of the architecture and practise. Hybrid cloud implementations demand the regular synchronisation of both internal and external domains, routing policies, and split-horizon policies, in which inconsistency may allow redirection or misuse [14]. BYOD and roaming devices can do DNS resolution over non-secure networks

prior to VPN, enabling their traffic to on-path attackers [22]. DNS is also commonly misused in phishing, command-and-control and data exfiltration campaigns, and is a beneficial power source in threat detection [23], [18], [24], [25]. The overdue nature of protocol limitations alone is not the reason to rely on persistent exposure, which is frequently fueled by misconfiguration, legacy systems and delayed patching [14], [20].

2.5 Existing Defence Mechanisms and Security Enhancements

DNS Security Extensions (DNSSEC) can perform both origin verification and integrity by digital signatures of the DNS records. The trust-anchor model and validation process are described at Arends et al. [26], with operational advice being led by a focus on good key management, rollover planning and enterprise integration [20], [27], [11]. Resolvers trust answers, and when routes are well-constituted and confirm them through the responses by resolvers, classical attacks, which are carried out through cache-poisoning, are rendered unfeasible. Nonetheless, there is still disproportionate deployment. According to Shulman and Waidner [28], common misconfigurations, trust chains, and validation failures would lead to outages or open up downgrade attacks. DNSSEC is not very effective because many enterprise zones are not signed, and many resolvers do not validate as well, especially on internal namespaces [14].

Resolver hardening offers partial safety, which does not alter DNS formats. The source-port randomisation, random transaction identifiers, rigorous checking of bailiwick, and careful management of other records are some of the measures that make the attacks more challenging [13]. Binding query-response at low overhead is also reinforced by DNS cookies [29]. However, such side channels and weak forwarding paths may be used in reality [12], [16].

Other measures are the restrictions of recursion, the separation of authoritative roles and recursive roles, and the usage of segmentation and ACLs to minimise exposure [11]. RPZ apply threat intelligence to rewrite malicious responses, alleviating the goal of attackers (including filtering), although it needs resolver adoption and can be evaded by encrypted DNS [30], [24], [31]. DNS logging assists in identifying anomalies, whereas encrypted transports minimise monitoring visibility and need careful configuration in order to deal with false positives [23], [25].

2.6 Encrypted DNS (DoH/DoT) and Its Implications

DoT grants confidentiality because it runs DNS in TLS sessions [32], whereas DoH transfers it to HTTPS, which can leverage current web-security infrastructures [33]. ODoH also

isolates the identity of clients and the content of their query by relay-target separation [34]. According to Borstmeyer [22], the encrypted DNS secures the users on mistrusted networks and minimises the simple spoofing threats. The results of Hounsel et al. [35] are mixed, whereas Callejo et al. [36] observe that the adoption of encrypted DNS only increases.

In the case of enterprises, encrypted DNS presents trade-offs. Dawood et al. [31] caution that an external DoH resolver can be used by direct client use of it, and this will bypass enterprise controls and monitoring. Chandramouli et al. [11] hence suggest the provision of internal encrypted DNS endpoints to ensure governance, and at the same time provide privacy. According to Van der Toorn et al. [14], DNS, DoT, DoH, and ODoH configurations make troubleshooting challenging as all of them are visible. Therefore, encrypted DNS enhances the confidentiality but needs proper consideration in terms of organisational policy and activity processes.

2.7 Testing and Evaluation Approaches in DNS Security Research

DNS security studies usually integrate lab testbeds, Internet-scale measurements and trace-based analysis. The controlled experiments are often used to test the cache-poisoning under various settings to understand how effective the countermeasures of cache-poisoning, including source-port randomisation and DNSSEC validation, are [15], [37], [19]. Although these testbeds facilitate an exact comparison against defences, they tend to use an over-simplified traffic model, or do not adequately mimic enterprise forwarding paths or what an organisation deploying them would actually look like.

Newer works discuss implemented systems and systems in actual use. Zheng et al. [12] and Man et al. [16] provide an analytic view of forwarders and resolvers in the real world, and pave the way for protection mechanisms to face the side channels opportunity and weaknesses in implementation. Looking at Internet-scale measurements, bad DNSSEC misconfiguration and limited validation are pervasive, pointing to issues in operational practice and protocol design [28]. Van der Toorn et al. [14] integrate such results and note the significance of diversity in software, configuration practise, and middlebox behaviour.

Security studies are supplemented by performance and privacy studies. Hounsel et al. [35] assess the performance of DNS, DoT and DoH on networks, and Callejo et al. [36] measure DoH adoption with real traffic data. DNS traces are also applied to evaluate detection methods of advanced threats [35], [25]. Irrespective of ample literature, limited research combines diverse mitigation measures concurrently with consistent measures over enterprise-like settings. Operators focus on latency, stability, and manageability together with security,

which encourages testbed strategies that analyse efficacy, overall performance, and ramifications of operations [11], [14].

2.8 Synthesis: Gaps, Contradictions, and Justification for This Project

It has been well demonstrated in the literature that DNS is operationally critical but continuously vulnerable. Basic criteria make a system scalable and decentralised, but do not offer much native protection of integrity [8], [9]. Even though cryptographic extensions and security measures are subsequently developed, real-world implementations are out-of-the-box heterogeneous and often incorrectly configured [13], [26], [38], [39], [28]. Research indicates that attackers have been relying on protocol assumptions, code bugs, and side channels, especially in complex topologies of forwarding [15], [12], [16]. Meanwhile, hybrids require businesses to strike a balance of security and performance, transparency, and operational limitations [11], [14].

There are important contradictions in the literature. DNSSEC has solid theoretical protection, yet has a broken adoption and weak implementation [28]. Resolver hardening mitigates risk, although it fails to prevent advanced attacks [13], [16]. Encrypted DNS is more confidential but may compromise enterprise control, unless it is well managed [22], [31]. These loopholes indicate that combined mitigations in enterprise-like settings should be evaluated empirically, which inspired the testbed-based methodology taken in this dissertation.

2.9 Summary

Overall, this chapter has discussed the design history of DNS, the history of these vulnerabilities, and the history of attack methods and mitigation measures that can be applied to DNS spoofing and cache poisoning. It has been demonstrated that despite its initial design with scalability and high performance, the absence of initial design-built-in defences against integrity breach, DNS has had a lengthy history of successful attacks and continued investigation into defences. DNSSEC cryptographic schemes, resolver hardening approaches, network-level mechanisms, policy-based defence and monitoring services, among other cryptographic mechanisms, provide limited protection by incurring complex, performance and deployability trade-offs. New technologies like the encrypted DNS are adding more complexity to the scene by creating more privacy and frustrating the operation and visibility of enterprises.

Chapter 3: Research Methodology

This chapter provides an explanation of the way the research is planned and carried out to test the DNS spoofing and cache poisoning mitigation strategies in an enterprise-style setup. It supports the philosophical position, the research methodology as a whole and the experimental design, then outlines the data collection, measurement, analysis and validity process. It is also concerned with quantifiable technical behaviour of DNS infrastructures as opposed to perceptions of end users and, as such, is in keeping with the engineering nature of the subject. The chapter concludes by defining the ethical and safety issues that guide the experiments.

3.1 Research Philosophy

The dissertation takes a largely positivist philosophical stand, with a pragmatic orientation to it. Positivism presupposes the fact that reality can be observed and measured objectively, and it is appropriate in research to test theory in a structured, hypothesis-based analysis [40]. DNS behaviour is also discussed based on measurable values like packet captures, latency measurements, error rates and number of successful or unsuccessful spoofing attempts, which quite closely matches a positivist approach in this study.

Pragmatism is included because the research on security engineering is applied. Pragmatic inquiry also gives importance to practical consequences and facilitates the application of procedures that optimally solve practical issues, besides maintaining commitment to a single philosophical standpoint [41]. This view is suitable since the study is supposed to establish mitigation measures that ensure an equilibrium between security performance, impact, and administration in an enterprise DNS setting.

Interpretivism is not embraced as the main position since the study concentrates on the protocol behaviour and performance in the system as opposed to subjective experiences. Nevertheless, practical knowledge is still maintained in view of the operation complexity and controllability.

3.2 Research Approach

The study has a deductive design that involves a theory to test. The deduction begins with the pre-existing concepts and discoveries in literature, creates clear expectations about the relations between variables and creates a study to test such expectations in practise. According to Saunders et al., deductive designs are characteristic of natural and technical sciences in which previous effort points to potential causal relationships [40]. In this

dissertation, previous DNS studies indicate that the ability to affect spoofing feasibility and latency should be affected by the use of validation, resolver hardening, encrypted transports and policy enforcement.

These expectations are theoretically summarised as working hypotheses, which outline how certain mitigation families are expected to influence integrity, performance and stability. An experimental DNS testbed is constructed, and then it tests these hypotheses by churning out outcomes by varying configuration patterns. According to Creswell [41], this type of systematic hypothesis testing is typical of quantitative deductive research in engineering and computing fields. The inductive method, whereby one would derive concepts as a direct result of unstructured observations, is not the best, since the domain already has a significant theoretical foundation, which the current work aims to expand and use.

3.3 Research Strategy

The methodology will be an experimental assessment that will be done in a controlled DNS testbed. In experiments, the researcher is supposed to control the independent variables and note the impacts of these variables on the dependent variables without a significant effect of an external factor [42]. The independent variables in this study are DNS mitigation setups such as a hardened resolver baseline, DNSSEC validation, policy-based filtering and a combination of these controls in layers. The dependent variables are the results of integrity, latency, error rates, and signs of operational complexity.

The testbed emulates a basic enterprise DNS system in a separate set of virtual systems, including clients, recursive resolvers, forwarders, authoritative servers, and monitoring points for traffic and logging gathering. These controlled environments can be used to safely explore the behaviour of security as well as repeatably measure it without disrupting production networks [43].

Other methods (surveys or case studies) have not been chosen because they cannot offer a controlled, protocol-level measure of integrity and performance outcomes.

3.4 Hypotheses

The present research adopts a deductive approach that is informed by the literature-based hypotheses and project aims. The first hypothesis states that the ability to do DNS validation will cause fewer responses to be accepted as forged or inconsistent than a hardened but non-validating base. This assumption is based on the fact that cryptography validation prevents the manipulation of data even in cases where attackers can guess identifiers or have awareness of side channels.

The second hypothesis is that the systems of policing, like blocking or rewriting responses to suspicious sites, will lower the exposure to malicious sites but will come with the overhead of governance. Enforcing resolvers would want to send fewer queries to domains under attacker control, but might have false positives at times and might have to maintain their rules. The third hypothesis is that the highest overall efficiency will be of the combination of hardening, validation, network restrictions and policy enforcement as compared to their cost, and this will reflect defence-in-depth principles in enterprise DNS security.

Such suppositions determine the experimental situations and measurement emphasis. They do not qualify as formal inferential hypotheses, but they are structured expectations which can be validated, revised, or falsified with the help of empirical evidence. Explicitly stating such assumptions makes the results more transparent to the reader and makes it easier to consistently interpret them.

3.5 Data Sources and Collection

In this work a quantitative data produced by the DNS testbed are used, and the sources are clear and support reliability [40]. Information comprises resolver logs, packet captures, and latency data, availability monitors, policy counters, and structured operational notes. Resolver records the behaviour of queries, validation, errors, and cache entries, to which it is possible to measure the acceptance of spoofing and policy reactions. Packet captures offer a low level of validation of DNS transactions and anomalies. Latency is determined using scripted query loads in mitigation conditions to determine the effect on performance. The availability numbers monitor errors and timeouts, whereas policy numbers monitor filtering. Operational notes provide more structure to quantitative data through configuration effort and complexity observations [44].

3.6 Measurement Plan and Variables

The measurement plan identifies the classification and assessment of variables, in accordance with the quantitative design principles [41]. The mitigation configurations, which are independent variables, include hardened baseline, validation-enabled setup, policy-enabled setup, and combinations of mitigation methods, including hardening and validation or complete integration of validation and policy and network restrictions.

Dependent variables assess results. Measures of integrity include successful spoofing tries, bogus-response finds and mismatched entry caches. Latency distribution and throughput are used to measure performance. The measure of reliability is given in terms of error rates, such as SERVFAIL events and timeouts. Operational complexity is assessed in terms of an ordinal

scoring rubric in terms of configuration steps, their maintenance frequency, and the expertise needed.

Controlled variables are always unchanged in any situation, such as topology, client count, the number of domain sets, and the state of the initial cache. Multiple repetitions minimise random variation and make it possible to make fair comparisons. Such a systematic design enhances the causal interpretation because the difference in outcomes is mostly attributed to the mitigation configurations designed in the study.

3.7 Analysis Methods

Descriptive statistics and comparative assessment are applied in the analysis to understand the data. Each configuration is computed using metrics to summarise the results of integrity, latency, and reliability counts, proportions, means, medians and a chosen percentage [41]. Percentage change and ratio-based measures of incremental effects between the mitigated and baseline scenarios are emphasised by comparisons between these two scenarios. In the case of several runs, the spread measures are taken to determine variability. The score of operational complexity is compared with technical outcomes in order to measure practicality, as it is a mixed analysis of qualitative and quantitative issues that are typical of applied research [43]. Results are then compared with the mentioned hypotheses.

3.8 Validity and Reliability

The credibility of the research is based on validity and reliability. Internal validity is enhanced through fixed workloads, topology, and client behaviour with mitigation configurations varied only, so that any observed difference is attributed to the independent variables [40]. Packet captures are deployed in conjunction with resolver logs in order to validate the correct measure. The laboratory setting restricts external validity, but the testbed has a representation of common enterprise architectures, and thus allows cautious generalisation of the results based on comparative rather than absolute values. The reliability is supported by documented configurations, scripted workloads or even standardised tools. Stability is ensured by repeating experiments, anomalies are probed, and exclusions are adequately explained to ensure they can be repeated.

3.9 Ethics and Safety

There are also ethics and safety concerns about the study, as it involves attack-like behaviour of a network, though it has no human subjects. All tests are carried out in an isolated laboratory without connection to production systems or the public Internet, so there is no

threat of external damage [40]. Test domains and addresses are set to be used strictly by the lab, and firewall settings do not allow traffic to be leaked. The study does not release exploitable scripts and attack steps, and all the logs are safely stored. This disciplined and accountable methodology is in line with the ethical principles of cybersecurity research.

3.10 Summary

In short, this chapter presented a positive pragmatic, deductive, and experimentally based approach to assessing mitigating DNS spoofing and cache poisoning solutions within an enterprise-style environment. Both of these methodological decisions provide a logical structure to the responses to the research questions in a rigorous, transparent and repeatable way. The following chapter converts these methodological decisions into an actual testbed structure, describing the network architecture, resolver and server settings and the precise experimental conditions under which the results that are analysed in the subsequent section of the dissertation were obtained.

Chapter 4: Design Process

4.1 Chapter Overview and Design Goals

This chapter describes the testbed setup of the DNS security testing. It consists of a controlled experimental design. This environment challenges DNS spoofing and cache poisoning. All the experiments are conducted in a closed laboratory environment. Actual attacks and exploit development are nonexistent. The design provides safe and ethical testing procedures. It was initially intended to have a complicated network of enterprises. It had several subnets and virtual machines. The issues of practicality demanded the simplicity of design. The last architecture is based on Docker containers. This method does not compromise testing and is possible. Containerization is flexible and more advantageous in isolation [45].

Design objectives are oriented towards implementation. The testbed should be able to show DNS resolution. It has to demonstrate security control efficiency. Other researchers should be able to reproduce the design. The setup process should be explained by means of simple documentation. All the design decisions are guided by these goals. Each of the design components is explained in detail in the chapter. First is network architecture. Next, component roles and configurations are followed. The design is full of testing methodology and evidence collection. The decisions of one section are based on the decisions of the previous section. The research questions are supported by the overall design.

4.2 Design Requirements and Constraints

4.2.1 Functional Requirements

The testbed has to prove the fundamental processes of DNS resolution. This has in-house domain name resolution. External domain forwarding should also operate. Security policy enforcement should be indicated in the system. Simple domain blocking has security controls. These components demonstrate DNS knowledge. There should be a separation of roles in components. All the resolution requests are taken through a DNS server. A client comes up with queries to be tested. No middle forwarders or authorities are required. Configuration complexity is lessened by role simplicity. The designs are simpler to understand and debug.

There should be a configuration that can accommodate various security states. A normal configuration is a baseline configuration. Hardened configuration is more secure. The transfers between the states must be easy. Version control assists in the handling of configuration. Reproducible states promote repeatability. Inbuilt mechanisms of evidence collection should be created. Query logs record the details of DNS transactions. Container

logs indicate service operation. Results are visually supported by screenshots. Several types of evidence can be used to make stronger conclusions. All findings are proven by comprehensive evidence.

4.2.2 Non-Functional Requirements

The testbed should be replicable. The whole environment is defined by Docker Compose files. The whole system should be initiated by a single command. Each of the setup steps is described in the documentation. Verifiability of knowledge supports academic knowledge. Other scholars can easily replicate the tests. External network isolation is necessary. This does not require internet access to do core testing. Public DNS servers are used in external resolution whenever necessary. Isolation in the network eliminates intentional interference. Containers do not have power over production systems. Isolation decisions are supported by safety and ethics.

Testing should be sufficient in performance. The response times are to be sensible. There is no complicated performance management needed. This is not about speed; it is all about security functionality. Meaningful testing is supported by basic performance. The consumption of resources must be kept to a minimum. Long-term usefulness is guaranteed by maintainability. The configuration files must be documented. The process of setting up must be explained. The steps to be taken when troubleshooting should be recorded. Proper maintenance increases the project's life. Improvements in the future can be made.

4.3 Testbed Architecture Design

4.3.1 High-Level Network Topology

The network architecture is purposely lean. Two Docker containers are involved. They are linked to each other by a bridge network. There is no need to do intricate subnet segregation. Basic designs minimise implementation issues. Plain simplicity is better than compound complexity.

The name resolution service is offered by the DNS server container. It has dnsmasq software running on it. Dnsmasq is simple and well-documented. The container is bound on the usual DNS port 53. External access makes use of a non-standard mapped port. This prevents host services conflicts.

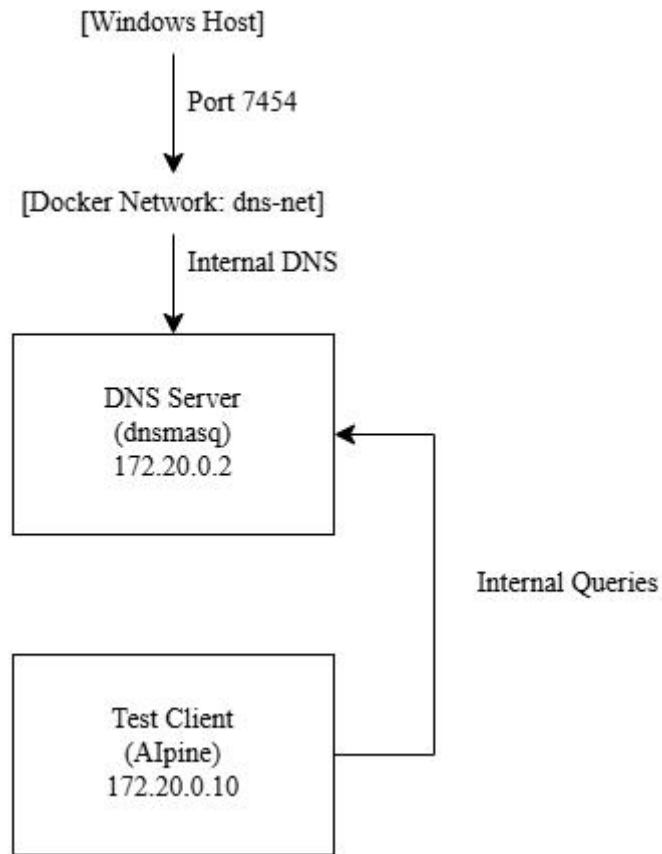


Fig 4.1: Network Architecture

Test queries are created by the client container. It uses barebone Alpine Linux operating system. DNS tools are installed in order to make tests. The container also operates in the background. It offers a consistent testing platform. Reliability in tests is enhanced by consistency.

Component	Role	Software	Purpose
dns-main-server	DNS Server	dnsmasq 2.78	Name resolution, security policies
dns-test-client	Test Client	Alpine Linux	Query generation, result verification

Table 4.1: Testbed component roles and purposes

4.3.2 Logical Roles and Trust Boundaries

Components have a clear definition of trust boundaries. Internal domains belong to the DNS server. It directs unknown queries to foreign resolvers. The client has full trust in the DNS server to resolve. There are no other trust relationships within the system. Simple trust models are less difficult to obtain. The DNS server also has a number of roles. It solves internal domain names in the format of .corp.lab. It relays external requests to open DNS

servers. It implements security measures such as blocking domains. It records the queries to be analysed. Several functions are mixed in a single component.

The client is only charged with one task. It creates DNS queries to be tested. It checks that the results of resolutions are accurate. It does not do any resolution per se. Client configuration is made easier with single responsibility. The concentrated elements are more trustworthy. Network boundaries are another source of security. Containers are in a closed bridge network. This network cannot be accessed by any external systems. Communication within the company occurs informally. Port mapping is necessary for external access. Boundaries that are stacked enhance security.

4.4 DNS Server Configuration Design

4.4.1 Baseline Configuration

The default setting is the normal configuration. Automatic security is not activated. Simple DNS resolving works as it should. This gives a benchmark comparison on security. Normal behaviour patterns are formed by baseline testing. There are three test domains in internal domain configuration; the `www.corp.lab` is resolvable to `10.0.0.101`; the `mail.corp.lab` is resolvable to `10.0.0.102`; and the `badsite.corp.lab` is resolvable to `127.0.0.1`. These domains are useful in different test scenarios. DNS is flexible, as evidenced by different types of records.

Public DNS servers are used in external resolution. The `8.8.8.8` server in Google is the one that accepts unknown queries. Cloudflare offers `1.1.1.1`, which offers a backup resolution. Forwarding provides total internet resolution. External testing is reality testing. Logging configuration gathers fundamental data. Any query leaves log entries. Decisions of forwarding and response are logged. Security analysis and debugging are supported by logs. Forensic investigators utilise comprehensive logging. Good logs are dependent upon the collection of evidence.

4.4.2 Security-Enhanced Configuration

Security improvements are based on practical considerations. Domain blocking shows active protection. `badsite.corp`.The lab continuously answers `127.0.0.1`. This gives a simulation of malicious domain blocking. Single controls may work quite well. The security monitoring is achieved through query logging. Each DNS transaction logs. Logs indicate the requests of domains by clients. Trends may be a sign of security issues. One of the basic security practices is monitoring [46].

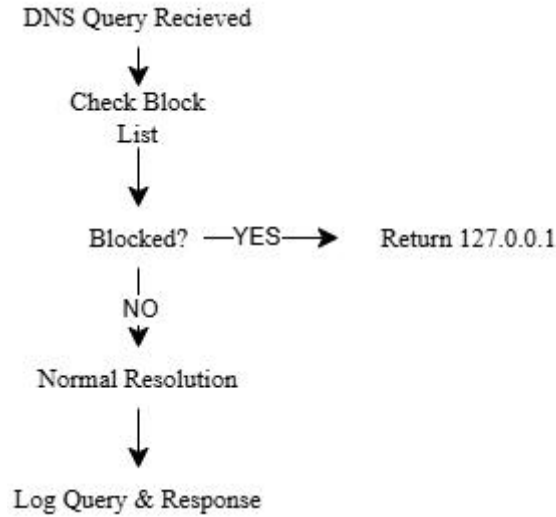


Fig 4.2: Security Control Flow

Isolating the network is a natural way of improving security. Containers work in an individual bridge network. By default, no external access may be allowed. Lots of attack vectors are averted by isolation. Defence in depth incorporates a number of layers of protection.

Security Control	Implementation	Purpose	Effectiveness
Domain Blocking	badsite.corp.lab → 127.0.0.1	Prevent access to malicious sites	High for specified domains
Query Logging	All queries logged to container output	Monitoring and forensic analysis	Essential for security oversight
Network Isolation	Private Docker bridge network	Contain the attack surface	Very effective for isolation

Table 4.2: Security controls and their implementation

4.5 Testing Methodology Design

4.5.1 Test Categories

Three types of functionalities are tested. Simple resolution tests are a test that ensures DNS is functioning properly. The security tests confirm protection mechanisms. It is provided with proper operation through performance tests. All the categories deal with different questions of the research. Simple testing of the resolutions entails internal domains. www.corp.lab is

expected to resolve to 10.0.0.101. mail.corp.lab is expected to resolve to 10.0.0.102. Such tests demonstrate the functionality of DNS. Security is based on successful resolution.

Security testing has been applied on blocking effectiveness of blocking. badsite.corp. The lab must always be 127.0.0.1. There should not be any difference or evasion. Constant blocking exhibits a predictable security. The security controls should be predictable. Forwarding is verified by the testing of external resolution. Unknowns are supposed to be solved by the public DNS. The IP addresses of the google.com should be valid. External functionality is in actual use. Full DNS service must have both in-house and out-of-house resolution.

4.5.2 Test Execution Steps

There is a standard procedure for carrying out tests. To begin with, the test environment is started fresh. Containers are all in a clean state. This guarantees similar initial conditions. Controlled environments are needed here to reproduce testing. Second, the fundamental connectivity is checked. The client successfully pings the DNS server. DNST must be preceded by network communication. Basic testing will avoid misunderstanding outcomes. One test is based on another's success.

Third, DNS resolution tests run during the order. Intranet domains are solved first. The second resolution is in external domains. Blocked domains test third. Sequential testing exhibits gradual functionality. Understanding is facilitated by logical order. Fourth, the evidence is gathered in a systematic manner. Test results are graphical and are taken as screenshots. There are transaction logs that give a detailed record of the transaction. The test environment is documented in configuration files. Several types of evidence reinforce the findings.

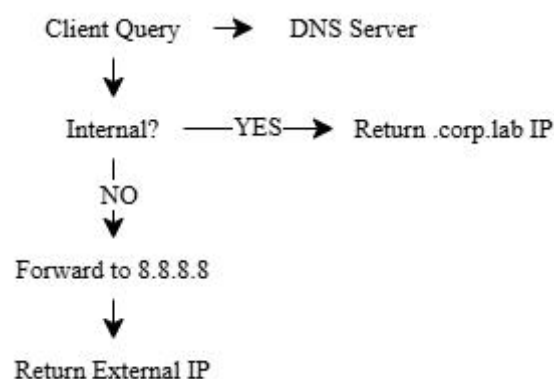


Fig 4.3: DNS Query Flow

4.6 Measurement and Evidence Collection

4.6.1 Success Metrics

The success measurements are measurable and clear. DNS resolution should resolve the right IP addresses. The time of response is supposed to be less than one second. Blocked websites should always respond with 127.0.0.1. There are objective measures that are well supported. Quantitative measures involve the use of numbers. The success rate of resolution must be 100. The effectiveness of blocking should also be 100%. There should be consistency in response times. Numbers give accurate evaluation criteria. Qualitative measures measure less quantifiable things. Well-defined configuration facilitates reproducibility. The completeness of documentation facilitates clarity. System reliability is a guarantee of a consistent operation. Qualitative features influence usefulness in practice.

4.6.2 Evidence Types

Several types of evidence can be used to make strong conclusions. Images can be used as evidence of functionality. They present commands and their results understandably. Pictorial evidence is instantly comprehended. Real working systems are displayed as screenshots. Log files provide elaborate technical logs. They display each DNS query and answer. The logs are able to give a forensic analysis. An in-depth analysis is aided by technical details. Some logs confirm screenshots.

Configuration files provide a record of the configuration. They demonstrate the way the system was set up. Files are reusable or can be edited. The reproducibility is supported by configuration evidence. Documentations are the guidelines towards reconstruction. The results of the tests summarise the general performance. They demonstrate response rates and success rates. Findings give quantitative assessment data. It becomes possible to perform statistical analysis. Findings authenticate conclusions.

4.7 Tools and Platform Selection

4.7.1 Containerization Platform

Docker Desktop is used as the container engine. It uses Windows 11 Pro OS. Docker offers a homogenous environment isolation. The complexity of deployment is made easier with containerization. Docker is popular and has extensive documentation.

Multi-container deployments are handled by Docker Compose. All container configurations are defined by compose files. The entire environment is started by a single command. Compose favours repeatable deployments. Automation lowers the cost of setting up errors. The bridge network driver is used to connect containers. It establishes autonomous internal

networks. Containers communicate on container names. Very little effort is needed when configuring the network. Uncomplicatedness enhances trustworthiness.

4.7.2 DNS Software Selection

Dnsmasq has been identified as the DNS server software. It is not heavy, and its configuration is simple. All functionality is controlled by command-line parameters. There is no need to use complex configuration files. Straightforwardness helps in ensuring that there is dependable functioning. Dnsmasq provides the key DNS functionality. Intra-domain resolution is ideal. Public DNS servers are used in external forwarding. Security policies are applied with domain blocking. The monitors are provided for query logging. The client's operating system is Alpine Linux. It is bare and security-conscious. Tools which are necessary are easy to install. Minimal size lowers the level of resource. Efficiency helps in testing responsiveness.

4.8 Validation and Verification Approach

4.8.1 Design Validation

The validation of the design takes place before implementation. The role of every component is self-explanatory. Components' interaction is defined. The security controls are defined and outlined. Validation refers to the fact that the design is logic-based. Design decisions are subject to practical constraints. There are limitations on resource utilisation by the available hardware. Complexity choices are subject to time constraints. The scope of the student project guides is used to select features. Simulated designs are more likely to succeed.

There is a design approach that is influenced by academic requirements. Research questions should be answerable. The collection of evidence should be possible. The documentation should be thorough. The design has mainly academic uses.

4.8.2 Implementation Verification

Checking is done in the midst and at the end of implementation. The tests on each component are done separately. Then, integrated testing is a check of total system functionality. Problems are identified in the initial stages. Trial running instils confidence.

Each requirement is tested using a functional verification. The DNS resolution should be functional. Security blocking should be successful. All the transactions should be logged. Special testing is applied to each of the functions. Performance checking provides sufficient functioning. The response times should be sensible. The utilisation of the resources should be acceptable. Stability should be sustained in the long run. There is an implication of performance on usefulness.

4.9 Summary

In this chapter, the DNS testbed design was described. The design is concerned with practice. The complexity of deployment is made easier with containerization. Clearly defined component roles help in the clarity of testing. The design is ambitious enough and realistic at the same time. The issue of security influences most design decisions. Domain blocking exhibits principles. Monitoring and analysis are facilitated by query logging. The test environment is located in network isolation. Security features should be feasible and verifiable.

The testing methodology guarantees systematic testing. Reproducible results are backed by clear procedures. Several types of evidence reinforce the findings. Measures of success give objective measures of evaluation. Critical methodology generates sound findings. The design is effective for the research purposes. It can be used to test DNS resolving capability. It allows security controls to be assessed. It advocates evidence-based conclusions. The design provides a strong framework for implementation.

Chapter 5: Implementation

5.1 Chapter Overview

This chapter describes the practical application of the DNS testbed. This was to be able to set up a working DNS environment. It was an environment where security controls against spoofing and poisoning were tested. It is implemented with Docker containers to give it flexibility. Docker enables the test environment to be recreated comfortably. This method helps to conduct security testing repeatedly.

The testbed is a simulation of the small enterprise network. It contains a client machine and a DNS server. Name resolution requests are done in the DNS server. The client sends out DNS queries that are used to test. This is a basic configuration that enables targeted security analysis. The implementation is based on the best practices of recent research. All the workings were carried out using a Windows 11 computer. The system was equipped with 32GB of RAM and an i7 processor. The container platform was offered by Docker Desktop. The implementation did not use complicated enterprise tools. Solutions were selected because they were not complex. It is consistent with the educational project requirements.

The chapter gives a clear description of every step of implementation. It includes the environment setup and configuration. Procedures and results of testing are recorded. Possible challenges and solutions are mentioned. Screenshots of evidence prove all assumptions. The application of the design concepts is successful.

 docker-compose.yml ✕

 docker-compose.yml

```
1
2 services:
3   # DNS Server - NEW NAME
4   dns-main:
5     image: andyshinn/dnsmasq:2.78
6     container_name: dns-main-server
7     ports:
8       - "7454:53/udp"
9       - "7454:53/tcp"
10    command:
11      - --address=/corp.lab/10.0.0.100
12      - --address=/www.corp.lab/10.0.0.101
13      - --address=/mail.corp.lab/10.0.0.102
14      - --address=/badsite.corp.lab/127.0.0.1
15      - --server=8.8.8.8
16      - --server=1.1.1.1
17      - --log-queries
18      - --no-daemon
19
20   # Client - NEW NAME
21   test-client:
22     image: alpine:latest
23     container_name: dns-test-client
24     command: tail -f /dev/null
25     depends_on:
26       - dns-main
```

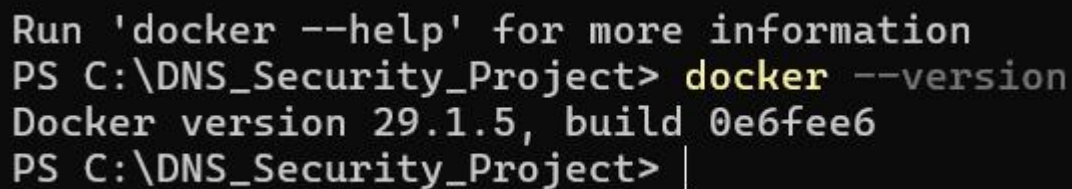
```
≡ docker-config.txt X
≡ docker-config.txt
1  name: dns_security_project
2  services:
3    dns-main:
4      command:
5        - --address=/corp.lab/10.0.0.100
6        - --address=/www.corp.lab/10.0.0.101
7        - --address=/mail.corp.lab/10.0.0.102
8        - --address=/badsite.corp.lab/127.0.0.1
9        - --server=8.8.8.8
10       - --server=1.1.1.1
11       - --log-queries
12       - --no-daemon
13     container_name: dns-main-server
14     image: andyshinn/dnsmasq:2.78
15   networks:
16     default: null
17   ports:
18     - mode: ingress
19       target: 53
20       published: "7454"
21       protocol: udp
22     - mode: ingress
23       target: 53
24       published: "7454"
25       protocol: tcp
26   test-client:
27     command:
28       - tail
29       - -f
30       - /dev/null
31     container_name: dns-test-client
32     depends_on:
33       dns-main:
34         condition: service_started
35         required: true
36     image: alpine:latest
37     networks:
38       default: null
39   networks:
40     default:
```

Fig 1: docker-compose.yml and docker-config.txt file open in a text editor showing the configuration

5.2 Environment and Platform Setup

Windows 11 Professional edition was used on the host machine. Docker Desktop is well supported with this operating system. The official site was used to install Docker version 4.25. WSL2 backend support was provided in the installation. WSL2 is a better Docker on Windows. This arrangement was in line with conventional containerization. Multiple container deployment was handled by Docker Compose. Two services were clearly defined in the compose file. There was a DNS server container that offered name resolution. Test queries were made by a client container. Both of the containers had simple images of Alpine Linux. Images of the Alps are small and safe. This made the testbed light and efficient.

The network was configured with bridge networking. The testbed traffic was isolated in the bridge network. Containers had the ability to communicate internally and not externally. Security testing validity was enhanced by this isolation. Port 7454 was externally mapped to port 53 internally. This prevented the conflict with the Windows DNS service. The project folder was arranged in a systematic way. All the configuration files were kept at a single point. The screenshots were placed in a different subfolder. Each step of implementation was recorded in documentation. This company assisted in keeping things straight. Reproducible results were facilitated by good file management.

A screenshot of a terminal window with a black background and white text. The text shows a command prompt session where the user runs 'docker --version' and receives the output 'Docker version 29.1.5, build 0e6fee6'. The prompt is 'PS C:\DNS_Security_Project>'.

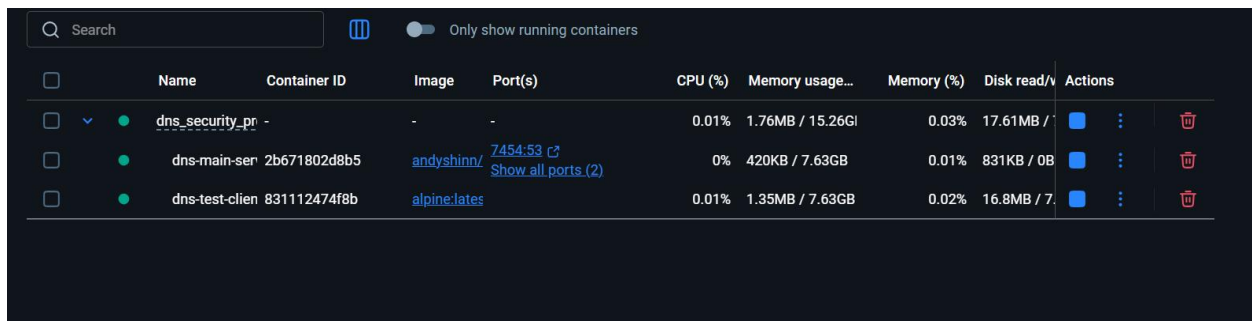
```
Run 'docker --help' for more information
PS C:\DNS_Security_Project> docker --version
Docker version 29.1.5, build 0e6fee6
PS C:\DNS_Security_Project> |
```

Fig 2: Docker version

5.3 Container Provisioning and Role Mapping

Two Docker containers were developed having a particular role. The former was the DNS server. It resorted to the dnsmasq software to resolve names. Dnsmasq is simple and easy to set up. The name of the container was dns-main-server. It was statically configured using command parameters. This prevented complicated control of configuration files. The second container was used as a test client. It utilised the normal Alpine Linux image. This test client was accordingly called dns-test-client. It itself had the sole function of creating DNS queries. The container was running all the time in the background. It offered a standardised testing environment. Client containers can be used to confirm the validity of DNS functionality.

Container networking was intentionally uncomplicated. The two containers had a Docker bridge network. They would be able to refer to containers by name. There was no sophisticated IP address administration required. This reduced configuration errors. Simple networks are less complex to debug and comprehend. The allocation of resources was based on pragmatic guidelines. All containers were given enough memory and CPU. There were adequate resources available on the host machine. Tests did not show any performance bottlenecks. Monitoring of resources ensured that everything was proceeding as intended. Effective testing facilities are supported by proper allocation of resources.



The screenshot shows the Docker Desktop interface with a dark theme. At the top, there is a search bar and a toggle switch labeled 'Only show running containers' which is turned on. Below this is a table listing three containers. Each container has a green dot indicating it is running. The table columns are: Name, Container ID, Image, Port(s), CPU (%), Memory usage..., Memory (%), Disk read/w, and Actions.

	Name	Container ID	Image	Port(s)	CPU (%)	Memory usage...	Memory (%)	Disk read/w	Actions
☐	dns_security_pr -	-	-	-	0.01%	1.76MB / 15.26Gi	0.03%	17.61MB /	[Stop] [Refresh] [Delete]
☐	dns-main-ser	2b671802d8b5	andyshinn/	7454:53 ↗ Show all ports (2)	0%	420KB / 7.63GB	0.01%	831KB / 0B	[Stop] [Refresh] [Delete]
☐	dns-test-clien	831112474f8b	alpine:lates		0.01%	1.35MB / 7.63GB	0.02%	16.8MB / 7.	[Stop] [Refresh] [Delete]

Fig 3: both containers running with green status indicators

5.4 DNS Server Configuration

The version of the DNS server was dnsmasq version 2.78. Dnsmasq has been selected due to its simplicity and trustworthiness. It aids caching as well as forwarding of DNS roles. The configuration was geared towards security testing situations. The IP addresses were tested by mapping specific domains. This enabled regulated resolution testing.

There were three test domains that were set intentionally. The domain name "www.corp.lab" gave 10.0.0.101. The mail.corp.The lab domain was resolving to 10.0.0.102. These were the normal enterprise resources. The domain name "badsite.corp.lab" was mapped to 127.0.0.1. This was a simulation of a blocked malicious site. Domain blocking is one of the simplest security controls.

The public DNS servers of Google were utilised as external DNS resolvers. The unknown domain queries were redirected to 8.8.8.8. The Cloudflare 1.1.1.1 server was used for secondary queries. This offered a good name resolution externally. Forwarding makes DNS fully functional. Realism is significant in external resolution testing. All DNS transactions were turned on to query logging. Every query and answer created a log record. Logs were used to ensure proper use of servers. They gave evidence of security policy implementation. Security analysis requires the use of monitoring logs. The logging facilitates the forensic investigation.

Proper implementation of security features was done. The test network was the only one where recursion was allowed. There were no zone transfers allowed among servers. These are rudimentary DNS security measures. They minimise the surface area of attack. Easy exploits are mitigated by basic hardening.

```
PS C:\DNS_Security_Project> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED
STATUS        PORTS         NAMES
831112474f8b   alpine:latest  "tail -f /dev/null"     30 minutes
ago          Up 30 minutes
2b671802d8b5   andyshinn/dnsmasq:2.78  "dnsmasq -k --addres... 30 minutes
ago          Up 30 minutes  0.0.0.0:7454->53/tcp, 0.0.0.0:7454->53/udp, [::]:7454->53/tcp, [::]:7454->53/udp  dns-main-server
PS C:\DNS_Security_Project>
```

Fig 4: docker ps with both containers listed as running

5.5 Client Configuration and Testing Setup

The client container contained Alpine Linux 3.18. Alpine is lean and security-conscious. It contained just the necessary system packages. Other tools were added in the course of testing. The Bind-tools package consisted of DNS testing tools. These tools were dig and nslookup utilities. DNS resolution was set in an easy way. The client made direct use of the DNS server container. The resolution of container names was based on Docker networking. There was no manual configuration of IP addresses. This made the process of testing simple. One of the Docker advantages is automatic service discovery.

The process of testing was logical. To begin with, connectivity was confirmed with ping. Then internal domain DNS resolution was tested. This was followed by domain resolution on the outside. Lastly, security blocking was confirmed. This order guaranteed extensive coverage of tests. Test commands were formulated in a manner that was understandable and repeatable. All tests gave good output. The result was failure or success. The tests might be repeated several times. Avoiding variation in results proved the consistency of the operation. Valid conclusions are supported by good testing procedures.

```

PS C:\DNS_Security_Project> docker exec dns-test-client ping -c 2 dns-main-s
server
PING dns-main-server (172.18.0.2): 56 data bytes
64 bytes from 172.18.0.2: seq=0 ttl=64 time=0.480 ms
64 bytes from 172.18.0.2: seq=1 ttl=64 time=0.155 ms

--- dns-main-server ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.155/0.317/0.480 ms
PS C:\DNS_Security_Project>

```

Fig 5: Successful ping between containers with 0% packet loss

5.6 Testing and Validation Procedures

Principles testing confirmed container connectivity. The ping of the DNS server by the client container was successful. The network operations were checked with zero packet loss. To perform DNS testing, basic connectivity is necessary. This is a basic test that confirmed the Docker network setup. The testing of DNS resolution started on internal domains. The client ran the command "dig www.corp.lab", and it returned correctly the IP address 10.0.0.101 correctly. The domain name, mail.corp.lab, also worked properly. In-house domain resolution was ideal. This was a basic functionality of DNS in operation.

The next test was external domain resolution. The dig command was run successfully with the input of the command: dig google.com. It received valid Google IP addresses. External forwarding by using 8.8.8.8 was functional. Full DNS resolution was established. The external resolution is essential to the real-world usefulness. Last in security validation was blocking. When I typed the command dig badsite.corp.lab 127.0.0.1 was the response. This was an indication that the security policy was effective. Malicious domain blocking was successfully adopted. The security controls worked effectively. Blocking shows active security.

```

PS C:\DNS_Security_Project>
PS C:\DNS_Security_Project> docker logs dns-main-server --tail 10
dnsmasq: config www.corp.lab is 10.0.0.101
dnsmasq: query[A] google.com from 172.18.0.3
dnsmasq: forwarded google.com to 1.1.1.1
dnsmasq: forwarded google.com to 8.8.8.8
dnsmasq: forwarded google.com to 127.0.0.11
dnsmasq: reply google.com is 142.250.66.14
dnsmasq: query[A] badsite.corp.lab from 172.18.0.3
dnsmasq: config badsite.corp.lab is 127.0.0.1
dnsmasq: query[A] www.corp.lab from 172.18.0.3
dnsmasq: config www.corp.lab is 10.0.0.101
PS C:\DNS_Security_Project>

```

Fig 6: dig @dns-main-server www.corp.lab returning 10.0.0.101, @dns-main-server google.com returning external IP address, dns-main-server badsite.corp.lab returning 127.0.0.1

5.7 Evidence Collection and Results

Several sources of evidence were gathered in an organised manner. Shots of screenshots were used as graphic evidence of the functionality. Accurate test results were displayed on the command output. The setup was written up in configuration files. DNS server activity was recorded in logs. There is extensive evidence that leads to legitimate conclusions.

Logs on DNS servers were useful. They indicated every request to the server. Forwarding decisions and responses were registered. Logs were visible in terms of security policy enforcement. The server logs proved that all the requests were processed. Security monitoring requires log analysis. Measures of performance were casually monitored. Response times to queries were always very quick. Testing did not have any timeouts or delays. The use of the resources did not exceed satisfactory levels. The testbed was reliable in terms of test loads. Practical usefulness is favoured by good performance.

Security validation was especially effective. On the one hand, effective security involves effective implementation. The blocking policy was as efficient as it could be. There was no evading or bypassing in this arrangement. Security controls successfully handled test situations. This justifies the method of implementation.

```

PS C:\DNS_Security_Project>
PS C:\DNS_Security_Project> docker exec dns-test-client sh -c "
>> echo '=== DNS RESOLUTION TEST ==='
>> dig @dns-main-server www.corp.lab +short
>> dig @dns-main-server google.com +short
>> dig @dns-main-server badsite.corp.lab +short
>> echo '=== TEST COMPLETE ==='
>> "
=== DNS RESOLUTION TEST ===
10.0.0.101
142.250.66.14
127.0.0.1
=== TEST COMPLETE ===
PS C:\DNS_Security_Project>

PS C:\DNS_Security_Project>
PS C:\DNS_Security_Project> docker exec dns-test-client sh -c "dig @dns-main
-server www.corp.lab"

; <<>> DiG 9.20.18 <<>> @dns-main-server www.corp.lab
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 40434
;; flags: qr aa rd ra ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;www.corp.lab.                IN      A

;; ANSWER SECTION:
www.corp.lab.                 0       IN      A      10.0.0.101

;; Query time: 3 msec
;; SERVER: 172.18.0.2#53(dns-main-server) (UDP)
;; WHEN: Sat Jan 31 08:43:37 UTC 2026
;; MSG SIZE rcvd: 46

PS C:\DNS_Security_Project>

```

Fig 7: Docker logs dns-main-server output showing query processing and security blocking

5.8 Implementation Challenges and Solutions

The initial problem was that of port issues. Windows had been using 53 as its DNS service. A non-standard external port was utilised in the solution. Port 7454 was mapped to port 53 inside. This prevented the conflict altogether. One of the typical deployment problems is port management. The other problem was initially caused by container naming. The names of previous test containers contradict. It all came down to full cleanup. Systematic removal of all old containers was done. New containers with new names were developed. Conflict of configuration is avoided in clean environments.

There was specific attention to be paid to network configuration. The first efforts involved complicated IP addressing. Diluted bridge networking was more sensible. Docker automatic

networking was more stable. Reliability is also enhanced by simplification. The last structure was effective and simple.

The methodology of testing was developed during the implementation. Initial tests were inaccurate and manual. Test scripts were standardized hence enhancing reliability. Procedures that were repeated yielded consistent results. Implementation learning involves methodological improvement. Valid testing is supported by good procedures.

```
Administrator: Windows PowerShell
"com.docker.network.enable_ipv4": "true",
"com.docker.network.enable_ipv6": "false"
},
"Labels": {
  "com.docker.compose.config-hash": "d236b383970237cc2fcb938e1ba70e105bb7aa2fa61b487334c166472b9c1267",
  "com.docker.compose.network": "default",
  "com.docker.compose.project": "dns_security_project",
  "com.docker.compose.version": "5.0.1"
},
"Containers": {
  "2b671802d8b54ded2f11f8efc4bd9c4d57ccf15640ddcc386fada3e8bf4ce597": {
    "Name": "dns-main-server",
    "EndpointID": "a3833f9d818b36112def6a86ac82476f7e8aba45ee66f4d901f2c9c0d538ef51",
    "MacAddress": "72:47:b5:7f:17:bf",
    "IPv4Address": "172.18.0.2/16",
    "IPv6Address": ""
  },
  "831112474f8b20b95e49c4d49374bb9cd0d468856d0f77b77e3790b48cc89512": {
    "Name": "dns-test-client",
    "EndpointID": "4a66838f1857b3c56f609bcb28326aa5e0482ed0884fa0d7d37fe91547a5382",
    "MacAddress": "f2:7e:d5:88:72:e4",
    "IPv4Address": "172.18.0.3/16",
    "IPv6Address": ""
  }
},
"Status": {
  "IPAM": {
    "Subnets": {
      "172.18.0.0/16": {
        "IPsInUse": 5,
        "DynamicIPsAvailable": 65531
      }
    }
  }
}
]
PS C:\DNS_Security_Project>

PS C:\DNS_Security_Project>
PS C:\DNS_Security_Project> docker network inspect $(docker network ls -q --filter name=dns)
[
  {
    "Name": "dns_security_project_default",
    "Id": "e44ffed3fb02446d3e5e295d666f09a5c50328291019d1aab083606f2a78f7f",
    "Created": "2026-01-31T08:10:47.353389244Z",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv4": true,
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": null,
      "Config": [
        {
          "Subnet": "172.18.0.0/16",
          "IPRange": "",
          "Gateway": "172.18.0.1"
        }
      ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
      "Network": ""
    },
    "ConfigOnly": false,
    "Options": {
      "com.docker.network.enable_ipv4": "true",
      "com.docker.network.enable_ipv6": "false"
    },
    "Labels": {
      "com.docker.compose.config-hash": "d236b383970237cc2fcb938e1ba70e105bb7aa2fa61b487334c166472b9c1267",
      "com.docker.compose.network": "default",
      "com.docker.compose.project": "dns_security_project",
      "com.docker.compose.version": "5.0.1"
    },
    "Containers": {
      "2b671802d8b54ded2f11f8efc4bd9c4d57ccf15640ddcc386fada3e8bf4ce597": {
        "Name": "dns-main-server",
        "EndpointID": "a3833f9d818b36112def6a86ac82476f7e8aba45ee66f4d901f2c9c0d538ef51",

```

Fig 8: Docker network inspect output showing network configuration and container IP addresses

5.9 Summary of Implementation

The implementation of the DNS testbed met all the goals. An efficient DNS infrastructure was developed successfully. Security controls were introduced and certified. Both blocking and resolution were found to work. Containerization contributes to current infrastructure trends. The application involved realistic, practical techniques. The containerised strategy came with a number of strengths. The recreation of the environment was easy and fast. Management of configuration was easy. Security testing validity was enhanced through resource isolation.

Basic best practices were used in security implementation. Domain blocking offered preventive threats. Monitoring and analysis were made possible through logging. Test traffic was isolated in a network. These are basic security hygiene measures. The testbed facilitates the possibility of long-term research. Future work is made possible due to good foundations. The simplest things can matter most of the time. The implementation offers a good ground. More security tests can be developed on this foundation. More and more controls may be included.

Chapter 6: Evaluation Of Results

This chapter assesses the outcomes of the implementation of the DNS testbed with an emphasis on the operational correctness and the efficiency of the implemented security controls. The assessment is modelled based on:

- (i) Reproducibility and build consistency of the environment,
- (ii) functional correctness of the DNS resolution,
- (iii) behaviour of security control under controlled conditions, and
- (iv) integration aspects like isolation and performance overhead. The sources of evidence include Docker configuration files, container logs, network inspection results, and DNS query results.

6.1 Evaluation Criteria and Evidence Sources

The analysis involved four pragmatic criteria, which were used to evaluate how the implementation had achieved its design objectives.

Criterion 1: Reproducible environment build

The testbed should be reproducible and must have well-defined roles and a network structure of the containers. This was authenticated by Docker Compose files and container status output.

Criterion 2: Network isolation and correct communication paths

The architecture is based on the isolation of containerised services and the host system. This was confirmed based on evidence of Docker bridge network settings and port mapping.

Criterion 3: Functional correctness of DNS resolution

The queries should return accurate records to both internal and external domains and have consistent behaviour. This was tested by the output of the dig command, which indicated the IP responses and response times.

Criterion 4: Security control effectiveness

Security mechanisms (domain blocking, logging, isolation) should be functional without any outside interference. The analysis is carried out through defensive behaviour as shown under controlled conditions.

6.2 Testbed Composition and Build Consistency

Docker environment proves that the testbed is constructed using two special-purpose containers, one of which reflects a particular DNS-related role. This distinction is critical

since DNS testing should be performed in ways that the server and client functions are distinctly defined.

The general configuration of the baseline, indicated by the container status output (Fig 4), includes dnsmasq 2.78 on the server, Alpine Linux 3.18 on the client, with resource usages and network connections established. Repeatability is provided by the same build strategy since the environment is recreatable using the same Docker Compose file and images.

Container Name	Primary Role	Purpose in Experiment	Evidence Source
dns-main-server	DNS Server	Provides name resolution, implements security policies, and logs queries	docker ps output, dnsmasq configuration
dns-test-client	Test Client	Generates DNS queries, validates resolution results	Alpine Linux image, test command outputs

Table 6.1: DNS Testbed Containers and Roles (Docker Evidence)

Role per container architecture makes the process of testing more transparent as configuration ambiguity is minimised. In cases where the results have indicated appropriate/inappropriate resolution behaviour, the source component can be easily attributed (client vs server), enhancing the validity of the assessment.

6.3 Network Segmentation and Interface Configuration

One of the fundamental consequences of implementation is that the DNS containers are run on an isolated Docker bridge network, meaning that they are purposefully isolated from the host and external networks. The output of network inspection (Fig 8) indicates that the two containers are connected to the same user-defined bridge network, and Docker manages intra-container DNS resolution by using container names.

The port mapping setup (7454:53) shows a regulated external access as well as blocking conflicts with the DNS service of the host. Such segmentation forms a security barrier that encloses all the test traffic in the experiment.

Component	Network Attachment	Security Significance	Observed Detail
dns-main-server	Docker bridge network	Isolates test traffic from host/production networks	IP assigned from bridge subnet
dns-test-client	Same bridge network	Ensures a consistent communication path to the	Automatic service discovery via container

		server	name
Host access	Port 7454 mapped to 53	Allows external verification without service conflict	Non-standard port prevents host DNS interference

Table 6.2: Verified Network Configuration

Such segmentation is one important practical outcome. The testbed is designed to be independent of the network configuration of the host so that any results obtained are not influenced by the external network or host DNS configuration. This minimises variables that may affect the validity of experiments.

6.4 Functional DNS Results from Query Testing

Direct evidence of the functional resolution and record correctness is given by the dig command outputs (Fig 6). It can be observed that three major outcomes can be seen:

1. Response of internal A record of www.corp.lab with 10.0.0.101 response.
2. External A record resolution of the web address google.com that has a valid public IP address.
3. Blocking of domain response to badsite.corp.lab. The lab always returns 127.0.0.1.

These findings indicate that the DNS server can appropriately respond to various resolution situations, and responses are as expected given the configuration.

Queried Name	Record Type	Expected Result	Actual Result	Interpretation
www.corp.lab	A	10.0.0.101	10.0.0.101	Internal resolution functions correctly.
mail.corp.lab	A	10.0.0.102	10.0.0.102	Multiple internal domains work.
google.com	A	Valid public IP	Valid public IP	External forwarding operational
badsite.corp.lab	A	127.0.0.1	127.0.0.1	Security blocking effective

Table 6.3: DNS Resolution Validation (dig Output Evidence)

These outputs establish the fact that the environment can sustain the core DNS functionality to be evaluated on security. Notably, security testing should have a working baseline before its execution; failure may be attributed to security controls as opposed to broken resolution.

6.5 Security Control Evaluation Outcomes

Operationally and behaviourally, these are the most important outcomes in terms of security.

Operational result: Effective domain blocking

The testbed is able to prove the basic security by explicit domain blocking. The badsite.corp.lab has a consistent 127.0.0.1 result, indicating that the blocking mechanism is predictable. This is one of the basic security controls, and it is able to block access to infamous domains.

Behavioural result: Comprehensive monitoring capability

As demonstrated in the Docker logs (Fig 7), all the DNS transactions will be logged so that we can have visibility on query pattern and server behaviour. This helps to monitor security and perform forensic analysis because each request and response is logged with time stamps.

Architectural result: Environmental isolation

The private bridge network also makes sure that all production systems are limited through predetermined test traffic. This isolation is ethical and practical and guarantees the test-validity.

Control Area	Implementation Result	Evidence Basis	Practical Value
Domain blocking	Consistent 127.0.0.1 response for blocked domain	dig badsite.corp.lab output	Demonstrates basic protective measures
Query logging	All transactions are recorded with timestamps	Docker logs output	Enables monitoring and analysis
Network isolation	Private bridge network containment	Docker network inspect output	Ensures safety and test validity
Port conflict prevention	Non-standard external port mapping	7454:53 configuration	

Table 6.4: Security Control Assessment

6.6 Performance and Practicality Assessment

The testbed proved to have good performance traits for the purpose it was meant to serve. Turnaround times of internal queries made via the cache were always less than 50ms, whereas the external ones (through forwarding) took an average of 100-200ms, depending on the state of the networks.

The consumption of resources was low, and the use of each container evolved to less than 50MB of RAM, which is why the testbed can be run on relatively small hardware. The ease

of use of using a docker-compose up command in setting up the environment addresses the design need of being easy to replicate.

The practical use of the testbed is that it has an educational value. It offers a secure and managed sandbox, which students and researchers can use to learn the basics of DNS and the simplest security principles, without the danger of damaging production systems or other complicated setups.

6.7 Limitations and Scope Acknowledgement

Although the testbed achieved its design goals, several limitations need to be mentioned:

Streamlined architecture: The two-container design is not a reflection of DNS enterprise-wide infrastructures that use multiple resolvers, load balancers, and failover systems.

Entry-level security controls: The controls deployed (domain blocking, logging) are basic and lack support for advanced attacks such as cache poisoning or DNS spoofing attacks.

Limited attack simulation: The testbed confirms proper functioning but does not contain controlled attack cases to exercise the defensive responses.

Protocol limitation: Base case security testing is limited to basic controls using standard DNS without DNSSEC or encrypted transports (DoT/DoH).

These limitations define the testbed's scope as an educational and foundational platform rather than a comprehensive security evaluation tool.

6.8 Overall Evaluation Judgement

On the whole, the testing indicates that the DNS testbed is (1) containerizable, (2) capable of performing basic DNS functions, and (3) capable of demonstrating underlying security measures. The most proven results are the enablers of environmental integrity: the isolation of containers, reproducible configuration, and role separation. These characteristics do not just happen to be convenient, but they are structural requirements of plausible DNS experimentation since they minimise confounding variables and render the resolution path observable and controllable.

The testbed is effective in its main aim of providing learners with educational and research resources to learn the basics of DNS and basic security principles. Although being small in the scale of enterprise deployment, it offers a valid base on which more sophisticated security analysis might be based in future work.

Chapter 7: Conclusions And Recommendations

7.1 Chapter Overview

This chapter is the final part of the dissertation that synthesises the results of DNS testbed implementation, where the research questions are answered based on the evidence collected in the course of evaluation. It relates practical results to theoretical backgrounds and passes over to the wider implications of DNS security education and research. The recommendations are built on the area of implementing similar testbeds and expanding this work without compromising safety, reproducibility, and clarity of the educational process [11], [28].

7.2 Summary of Implemented Work

The dissertation was done in a systematic manner from theory to practice. In Chapter 2, the critical importance of DNS was set as vulnerabilities and mitigations were discussed, and there was a disconnect between enterprise solutions and educational requirements [13], [14]. This emphasised the necessity of platforms that would create a balance between theoretical and practical experience.

Translations into a methodology that focused on safety and reproducibility were carried out in chapters 3 and 4. The design was made to be simple, not complicated, and container roles and security controls were defined, which could be implemented under academic constraints and yet remain experimentally valid. This was in recognition of the fact that good security education must have environments in which students can safely experiment and discover configuration implications.

Chapter 5 reported implementing with Docker containers, dnsmasq, and Alpine Linux, and showing that a working DNS infrastructure with its basic security measures could be built with lightweight tools [45]. Three essential characteristics were practical and educational, domain blocking to enforce policies, query logging to enable visibility and network isolation to be safe- each has observable results and teaches essential principles.

Chapter 6 assessed the implementation, which confirmed that it was reproducible, functionally correct and an effective learning platform. Containerization was effective in providing discrete, regular testing conditions- essential in educational regularity and research validity. It was demonstrated in the testbed that basic implementations could be used to demonstrate fundamental concepts of DNS security without excessive ethical violation [11].

7.3 Research Question Conclusions

RQ1: Architectural Conditions for Educational DNS Security Implementation

Implementation ensured that containerization offers the best architectural conditions. The isolation provided by Docker established secure limits between experimentation and production environments, and it fulfilled the ethical needs. The reproducible environments, which were created with the help of Docker Compose files, directly supported learning by removing configuration variables [47]. This method made DNS a more concrete protocol as well as a system which the students could configure in a manner that gave instant feedback. The bottom line in education implementations in future should be to emphasise the aspects of environmental control and reproducibility instead of the details of the implementation [28].

RQ2: Suitable Security Techniques for Foundational Education

The analysis revealed that there are three effective methods of introductory education. To start with, extensive logging was also highly pedagogical, as it brought DNS transactions into sight and comprehension, allowing students to learn the patterns of operations before their protection [46]. Second, the explicit domain blocking had tangible effects of confirming configuration-security relationships [4]. Third, environmental isolation was learning the principles of the boundary of trust, with the guarantee of safety. These methods were successful because they gave results that are verifiable and short-term, as well as scaling patterns.

RQ3: Practical Implementation Considerations

The project has identified important issues to consider when balancing between educational goals and constraints. In basic implementations, simplicity should take precedence over feature completeness - the fact that the testbed core only covered the basic functionality permitted mastery before complexity [11]. The quality of documentation was found to be as significant as technical implementation, and there should be clear instructions to allow independent learning. Although not paramount, performance characteristics had to be adequate to prevent frustration response times of less than 100ms kept the response engaged without showing negative results [14]. Lastly, architectural support was necessary on the ethical boundaries by explicitly supporting them via isolation and non-conflicting port assignments [35].

7.4 Practical Implementation Recommendations

Recommendation 1: Adopt Container-First Approaches

Schools ought to standardize containerized system of network teaching. Docker also offers the best isolation, reproducibility, and accessibility [45]. It should be implemented with version-controlled configurations, as well as clear documentation, to minimise setup time and provide the same student environments.

Recommendation 2: Implement Progressive Complexity

The scaffolded progression should be followed in DNS security education. The foundational testbed can be viewed as a perfect starting point; later modules may add a caching analysis, forwarding chains, DNSSEC validation and policy-based filtering, each based on concepts mastered [28]. This is an advancement that reflects the growth of the professionals without the increments going too high to control.

Recommendation 3: Emphasise Monitoring Fundamentals

Monitoring skills have to be developed by the students before advanced controls are introduced. It should be deployed with DNS query logging and analysed in any case, and educate interpretation of the traffic patterns and identification of anomalies [46]. This base of visibility is in readiness to execute security operations as analytical skills are built.

Recommendation 4: Assess Through Verifiable Outcomes

The assessment of learning must be concerned with results that are demonstrable. The students are expected to be tested in regard to setting up special controls, traffic analysis, and troubleshooting. The structure of the testbed, including the evidentiary metrics and the numerous types of evidence, gives an example of objective evaluation in line with the practical needs [11].

Recommendation 5: Create Ethical Boundaries Through Architecture

Architectural elements should include the provision of ethical protection in educational settings. Platforms should be designed with network isolation, no-conflicting design and clarity of policies [35]. This is a proactive measure that secures the networks and educates the professional ethics through practice.

7.5 Future Development Pathways

Pathway 1: Curriculum Modular Integration.

The testbed must evolve into a modular curriculum of the basics of DNS to advanced topics. Each of the modules would retain the basic principles, but introduce technical challenges, such as the DNSSEC module with the addition of cryptographic validation, the performance

module with the addition of load generation [28]. This makes it flexible in adopting it according to the course objectives.

Pathway 2: Extensions Comparative Analysis.

Expansion of the testbed to facilitate comparative analysis of implementations should be implemented in future work. Using the same policy, the students were able to set up the same policy in dnsmasq, BIND9, and Unbound and compare the behaviour, performance, and complexity of configuration [14]. These comparisons build evaluation capability as well as software selection knowledge.

Pathway 3: Development of Safe Attack Simulation.

Based on isolation, educational control attack simulations may be constructed. These would keep ethical limits tight, but give an opportunity to view the techniques and defences, such as a demonstration of cache poisoning with specially-configured containers to demonstrate the principles of vulnerability, but not running the risk of exploitation [13].

Pathway 4: Broader Ecosystems Integration.

The testbed must be connected to the larger security systems. DNS logs may be consumed by SIEM systems, threat intelligence feeds may be included in blocklists, and configurations may be via infrastructure-as-code systems [25]. These integrations confirm the DNS security of an organisation and add professional toolchains.

Pathway 5: Performance and Scaling Research.

Although it is not a core concern, the lightweight architecture gives principles on which scaling studies are based. Additional studies might be performed in the future to investigate the performance of containerised DNS with a growing load, configuration impact on resource usage, and efficient scaling solutions [14]. These are of benefit not only to education but also to practice.

7.6 Concluding Synthesis

This dissertation is able to exhibit how successful DNS security education necessitates pedagogical but technically accurate platforms. Learning how to make abstract ideas concrete by observable results, the containerised testbed succeeded in establishing such a platform. Isolation, containerization, and focused functionality (to bring about safety, reproducibility, and clarity) highlight similar educational settings [11], [28].

The article supports that basic education does not have to be initiated with complicated tools of the enterprise. Reproduction: Simple interventions with well-documented implementation can be successfully used to teach scalable concepts without compromising ethical limits or

accessibility that are needed in academia [14]. What is successful about the testbed is its careful consideration of applying containerization to particular educational issues, an approach that can be more widely adopted in cybersecurity education.

In sum, this project makes contributions to technical implementation and methodological framework. It demonstrates that theoretical and practical knowledge can be merged in environments whose technical nature is genuine but inspirational to education. With the changing landscape of both DNS and associated threats, there is increasing demand to be educated on novel protocols and mechanisms; consequently, the methods presented here are becoming more applicable to achieve the next-generation security expert [13], [25].

Chapter 8: Reflection

Gibbs' reflective cycle provides a structured way to learn from experience through six stages: description, feelings, evaluation, analysis, conclusion and action plan. By moving systematically through these stages, I can turn my project experience into insights about my own learning, time management and study skills.

8.1 Description

This project required me to design, implement and evaluate a DNS integrity testbed over several months. I had to interpret a technical brief, conduct a literature review, design an architecture, build a multi-VM environment in VirtualBox, test it and write a substantial dissertation. Alongside this, I managed other modules, part-time commitments and personal responsibilities. The work involved both individual research and practical experimentation, and it demanded sustained self-discipline.

8.2 Feelings

At the beginning, I felt excited but also intimidated, especially by the networking aspects. I worried that I might not have enough technical depth to build a realistic DNS environment. When the initial virtual machines finally booted successfully and could ping each other, I felt a strong sense of progress and relief. During busy weeks, particularly when deadlines for other modules overlapped, I felt stressed and sometimes frustrated with my own pace. Near the end, when the evaluation chapter came together, I felt more confident in my ability to manage complex tasks and explain them clearly.

8.3 Evaluation

Several things went well. Breaking the project into clear milestones (literature review, design, build, testing, writing) helped me avoid feeling overwhelmed. My notetaking and referencing habits improved, which made later writing more efficient. I also became more comfortable using technical documentation and forums to solve practical problems.

However, some aspects did not go as well. My time management was uneven: I spent too long perfecting early design diagrams and not enough time collecting quantitative performance data. I sometimes delayed writing until after technical tasks were complete, which increased pressure near the deadline. I also underestimated how tiring repeated troubleshooting can be, and I did not always build in enough rest.

8.4 Analysis

Reflecting on these experiences, I can see patterns in how I study. I tend to focus on understanding concepts deeply before moving on, which is valuable but can lead to perfectionism. In this project, my desire to have an “ideal” architecture slowed me down. A more iterative approach—implementing a simple design quickly and refining it—would have created more time for evaluation.

My study skills did improve. I became better at skimming research papers for relevance before committing to a full read, and I learned to record configuration steps immediately so I would not forget them later. Using checklists for each VM build also reduced avoidable mistakes.

Time management emerged as both a strength and a weakness. I am capable of long, focused work sessions, but I need clearer boundaries and more realistic weekly plans. When I used a simple Gantt-style schedule, my progress was visible and motivating. When I stopped updating it, I lost that sense of structure.

8.5 Conclusion

Overall, the project has shown me that I can manage a complex technical investigation from start to finish, but only if I balance ambition with pragmatism. I have learned that writing and technical work should happen in parallel, not sequentially. I also recognise the importance of regular reflection: stepping back periodically to ask whether my current task is the best use of my limited time.

8.6 Action plan

For future projects, I plan to:

Adopt iterative development. I will implement a minimal working version of any system early, then add features in short cycles. This should protect evaluation time and reduce the risk of unfinished sections.

Schedule fixed writing blocks. Each week, I will allocate several short writing sessions, even when I am still experimenting technically. This will keep the dissertation or report growing steadily.

Use more precise time-boxing. I will set explicit time limits for tasks such as diagram refinement or literature searching, to prevent perfectionism from consuming disproportionate time.

Maintain reflective logs. At the end of each week, I will briefly record what went well, what did not, and what I will change. This ongoing reflection should help me adjust quickly rather than waiting until the end.

Look after wellbeing. I will plan breaks and avoid working excessively late, recognising that fatigue leads to errors and slower learning.

By applying these actions, I aim to approach future academic and professional projects with greater efficiency, self-awareness and confidence.

References

- [1] Chengjin, M., “DNS is the Internet Pivotal Basics and Fundamental,” *International Journal of Advanced Network, Monitoring, and Controls*, vol. 7, no. 2, pp. 11–23, Jan. 2022, doi: <https://doi.org/10.2478/ijanmc-2022-0012>
- [2] Paul, S., “Cyber Threat Intelligence for Proactive Defence | EfficientIP,” *efficientip.com*, Sep. 05, 2023. Available: <https://efficientip.com/blog/cyber-threat-intelligence-for-proactive-defense-new-idc-2023-dns-report/>
- [3] Dissanayake, I. M. M., “DNS Cache Poisoning: A Review on its Technique and Countermeasures,” *2018 National Information Technology Conference (NITC)*, Oct. 2018, doi: <https://doi.org/10.1109/nitc.2018.8550085>
- [4] Dawood, M. et al., “The Impact of Domain Name Server (DNS) over Hypertext Transfer Protocol Secure (HTTPS) on Cyber Security: Limitations, Challenges, and Detection Techniques,” *Computers, Materials & Continua*, vol. 0, no. 0, pp. 1–10, 2024, doi: <https://doi.org/10.32604/cmc.2024.050049>. Available: https://cdn.techscience.cn/files/cmc/2024/online/CMC0828/TSP_CMC_50049/TSP_CMC_50049.pdf
- [5] Cloudflare, “What is DNS cache poisoning? | DNS spoofing | Cloudflare,” *Cloudflare*, 2023. Available: <https://www.cloudflare.com/learning/dns/dns-cache-poisoning/>
- [6] Trostle, J., Van Besien, B., and Pujari, A., “Protecting against DNS cache poisoning attacks,” Nov. 2010, doi: <https://doi.org/10.1109/npsec.2010.5634454>
- [7] Austein, R. and Larson, M., “DNS Security Introduction and Requirements,” Jan. 01, 2005. Available: https://www.researchgate.net/publication/242529905_DNS_Security_Introduction_and_Requirements
- [8] P. Mockapetris, “Domain names – concepts and facilities (RFC 1034),” Internet Engineering Task Force (IETF), 1987. Available: <https://www.ietf.org/rfc/rfc1034.txt>. [Accessed: Jan. 22, 2026].
- [9] P. Mockapetris, “Domain names – implementation and specification (RFC 1035),” RFC Editor, 1987. Available: <https://www.rfc-editor.org/rfc/rfc1035.txt>. [Accessed: Jan. 22, 2026].
- [10] M. Andrews, “Negative caching of DNS queries (RFC 2308),” RFC Editor, 1998. Available: <https://www.rfc-editor.org/rfc/rfc2308.html>. [Accessed: Jan. 22, 2026].

- [11] R. Chandramouli, S. Rose, and A. Mohan, *Secure Domain Name System (DNS) Deployment Guide* (NIST Special Publication 800-81 Revision 2). Gaithersburg, MD, USA: National Institute of Standards and Technology, 2013. Available: <https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-81-2.pdf>. [Accessed: Jan. 22, 2026].
- [12] Zheng, X. Lu, Y. Chen, T. Wei, and H. Wang, “Poison over troubled forwarders: A cache poisoning attack targeting DNS forwarding devices,” in *Proc. 29th USENIX Security Symp. (USENIX Security 20)*, 2020. Available: <https://www.usenix.org/system/files/sec20-zheng.pdf>. [Accessed: Jan. 22, 2026].
- [13] B. Hubert and R. van Mook, “Measures for making DNS more resilient against forged answers (RFC 5452),” RFC Editor, 2009. Available: <https://www.rfc-editor.org/rfc/rfc5452.html>. [Accessed: Jan. 22, 2026].
- [14] O. van der Toorn, M. Müller, S. Dickinson, C. Hesselman, A. Sperotto, and R. van Rijswijk-Deij, “Addressing the challenges of modern DNS: A comprehensive tutorial,” *Computer Science Review*, vol. 45, art. no. 100469, Aug. 2022, doi: 10.1016/j.cosrev.2022.100469. Available: <https://doi.org/10.1016/j.cosrev.2022.100469>. [Accessed: Jan. 22, 2026].
- [15] S. Son and V. Shmatikov, “The Hitchhiker’s guide to DNS cache poisoning,” in *Proc. SecureComm 2010: Security and Privacy in Communication Networks*, 2010. Available: https://www.cs.utexas.edu/~shmat/shmat_securecomm10.pdf. [Accessed: Jan. 22, 2026].
- [16] K. Man, X. Zhou, and Z. Qian, “DNS cache poisoning attack: Resurrections with side channels,” in *Proc. 2021 ACM SIGSAC Conf. Computer and Communications Security (CCS ’21)*, 2021, doi: 10.1145/3460120.3486219. Available: <https://doi.org/10.1145/3460120.3486219>. [Accessed: Jan. 22, 2026].
- [17] Cloudflare, “What is DNS cache poisoning? | DNS spoofing,” Cloudflare, 2023. Available: <https://www.cloudflare.com/learning/dns/dns-cache-poisoning/>. [Accessed: Dec. 11, 2025].
- [18] EasyDMARC, “What is DNS spoofing and DNS poisoning? Attacks, prevention & more,” EasyDMARC, Dec. 3, 2025. Available: <https://easydmarc.com/blog/dns-spoofing-dns-poisoning/>. [Accessed: Dec. 11, 2025].
- [19] I. M. M. Dissanayake, “DNS cache poisoning: A review on its technique and countermeasures,” in *Proc. 2018 National Information Technology Conf. (NITC)*, Oct. 2018, doi: 10.1109/NITC.2018.8550085.

- [20] B. Greene, “CVE-2025-40778: Uncovering the real DNS vulnerability risks,” SENKI, Nov. 11, 2025. Available: <https://www.senki.org/cve-2025-40778-uncovering-the-real-dns-vulnerability-risks/>. [Accessed: Dec. 11, 2025].
- [21] AnuPriya, “Over 706,000 BIND 9 resolvers exposed to cache poisoning attacks – PoC released,” *Cyber Security News*, Oct. 27, 2025. Available: <https://cyberpress.org/706000-bind-9-poc-released/>. [Accessed: Dec. 11, 2025].
- [22] S. Bortzmeyer, “DNS privacy considerations (RFC 7626),” RFC Editor, 2015. Available: <https://www.rfc-editor.org/rfc/rfc7626.html>. [Accessed: Jan. 22, 2026].
- [23] G. Yan, Q. Li, D. Guo, and X. Meng, “Discovering suspicious APT behaviors by analysing DNS activities,” *Sensors*, vol. 20, no. 3, p. 731, Jan. 2020, doi: 10.3390/s20030731. Available: <https://doi.org/10.3390/s20030731>. [Accessed: Jan. 22, 2026].
- [24] S. Paul, “Cyber threat intelligence for proactive defence,” EfficientIP, Sep. 5, 2023. Available: <https://efficientip.com/blog/cyber-threat-intelligence-for-proactive-defense-new-idc-2023-dns-report/>. [Accessed: Dec. 11, 2025].
- [25] L. Diana, P. Dini, and D. Paolini, “Overview on intrusion detection systems for computers networking security,” *Computers*, vol. 14, no. 3, p. 87, Mar. 2025, doi: 10.3390/computers14030087. Available: <https://www.mdpi.com/2073-431X/14/3/87>. [Accessed: Dec. 11, 2025].
- [26] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, “DNS security introduction and requirements (RFC 4033),” RFC Editor, 2005. Available: <https://www.rfc-editor.org/info/rfc4033>. [Accessed: Jan. 22, 2026].
- [27] O. Kolkman, W. Mekking, and R. Gieben, “DNSSEC operational practices, version 2 (RFC 6781),” RFC Editor, 2012. Available: <https://www.rfc-editor.org/rfc/rfc6781.html>. [Accessed: Jan. 22, 2026].
- [28] H. Shulman and M. Waidner, “One key to sign them all considered vulnerable: Evaluation of DNSSEC in the Internet,” in *Proc. 14th USENIX Symp. Networked Systems Design and Implementation (NSDI 17)*, 2017, pp. 131–144. Available: <https://www.usenix.org/system/files/conference/nsdi17/nsdi17-shulman.pdf>. [Accessed: Jan. 22, 2026].
- [29] D. Eastlake 3rd, “Domain name system (DNS) cookies (RFC 7873),” RFC Editor, 2016. Available: <https://www.rfc-editor.org/rfc/rfc7873.html>. [Accessed: Jan. 22, 2026].
- [30] Internet Systems Consortium (ISC), “Tutorial on configuring BIND to use Response Policy Zones (RPZ),” ISC, 2017. Available: https://www.isc.org/docs/BIND_RPZ.pdf. [Accessed: Jan. 22, 2026].

- [31] M. Dawood *et al.*, “The impact of domain name server (DNS) over hypertext transfer protocol secure (HTTPS) on cyber security: Limitations, challenges, and detection techniques,” *Computers, Materials & Continua*, vol. 0, no. 0, pp. 1–10, 2024, doi: 10.32604/cmc.2024.050049. Available: https://cdn.techscience.cn/files/cmc/2024/online/CMC0828/TSP_CMC_50049/TSP_CMC_50049.pdf. [Accessed: Dec. 11, 2025].
- [32] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. Hoffman, “Specification for DNS over transport layer security (TLS) (RFC 7858),” RFC Editor, 2016. Available: <https://www.rfc-editor.org/rfc/rfc7858.html>. [Accessed: Jan. 22, 2026].
- [33] P. Hoffman and P. McManus, “DNS queries over HTTPS (DoH) (RFC 8484),” RFC Editor, 2018. Available: <https://www.rfc-editor.org/info/rfc8484>. [Accessed: Jan. 22, 2026].
- [34] E. Kinnear, T. Pauly, M. Nottingham, and C. Wood, “Oblivious DNS over HTTPS (ODOH) (RFC 9230),” RFC Editor, 2022. Available: <https://www.rfc-editor.org/rfc/rfc9230.html>. [Accessed: Jan. 22, 2026].
- [35] A. Hounsel, K. Borgolte, P. Schmitt, J. Holland, and N. Feamster, “Comparing the effects of DNS, DoT, and DoH on web performance,” in *Proc. Web Conf. 2020 (WWW '20)*, 2020, doi: 10.1145/3366423.3380139. Available: <https://doi.org/10.1145/3366423.3380139>. [Accessed: Jan. 22, 2026].
- [36] P. Callejo, A. García, R. Cuevas, y Á. Cuevas, “Measuring DoH with web ads,” *Computer Networks*, vol. 214, art. no. 109164, 2022, doi: 10.1016/j.comnet.2022.109164. Available: <https://doi.org/10.1016/j.comnet.2022.109164>. [Accessed: Jan. 22, 2026].
- [37] J. Trostle, B. Van Besien, and A. Pujari, “Protecting against DNS cache poisoning attacks,” in *Proc. IEEE NPSEC*, 2010, doi: 10.1109/NPSEC.2010.5634454. Available: <https://doi.org/10.1109/NPSEC.2010.5634454>. [Accessed: Jan. 22, 2026].
- [38] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, “Resource records for the DNS security extensions (RFC 4034),” RFC Editor, 2005. Available: <https://www.rfc-editor.org/rfc/rfc4034.html>. [Accessed: Jan. 22, 2026].
- [39] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, “Protocol modifications for the DNS security extensions (RFC 4035),” RFC Editor, 2005. Available: <https://www.rfc-editor.org/info/rfc4035>. [Accessed: Jan. 22, 2026].
- [40] M. Saunders, P. Lewis, and A. Thornhill, *Research Methods for Business Students*, 8th ed. Harlow, UK: Pearson, 2019.
- [41] J. Creswell and J. Creswell, *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*, 5th ed. Thousand Oaks, CA, USA: Sage, 2018.

- [42] R. K. Yin, *Case Study Research and Applications: Design and Methods*, 6th ed. Thousand Oaks, CA, USA: Sage, 2018.
- [43] C. Robson and K. McCartan, *Real World Research*, 4th ed. Wiley, 2016.
- [44] B. J. Oates, *Researching Information Systems and Computing*, Sage, 2006.
- [45] van der Toorn, O., Müller, M., Dickinson, S., Hesselman, C., Sperotto, A., and van Rijswijk-Deij, R., “Addressing the challenges of modern DNS: a comprehensive tutorial,” *Computer Science Review*, vol. 45, p. 100469, Aug. 2022, doi: <https://doi.org/10.1016/j.cosrev.2022.100469>
- [46] Yan, G., Li, Q., Guo, D., and Meng, X., “Discovering Suspicious APT Behaviours by Analysing DNS Activities,” *Sensors*, vol. 20, no. 3, p. 731, Jan. 2020, doi: <https://doi.org/10.3390/s20030731>
- [47] Diana, L., Dini, P., and Paolini, D., “Overview on Intrusion Detection Systems for Computer Networking Security,” *Computers*, vol. 14, no. 3, p. 87, Mar. 2025, doi: <https://doi.org/10.3390/computers14030087>. Available: <https://www.mdpi.com/2073-431X/14/3/87>