National College of Ireland

National College of Ireland

Project Submission Sheet

Student Name:	Dion Raphael		
Student ID:	23270969		
Programme:	MSc Cybersecurity	**Year:**	2024-25
Module:	Practicum		
Lecturer:	Khadija Hafeez		
Submission Due Date:	11/08/2025		
Project Title:	Zero Trust Security Model in Enterprises		
Word Count:	6067 without references		

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the references section. Students are encouraged to use the Harvard Referencing Standard supplied by the Library. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action. Students may be required to undergo a viva (oral examination) if there is suspicion about the validity of their submitted work.

Signature: Dion Raphael

Date: 11/08/2025

PLEASE READ THE FOLLOWING INSTRUCTIONS:

1. Please attach a completed copy of this sheet to each project (including multiple copies).
2. Projects should be submitted to your Programme Coordinator.
3. **You must ensure that you retain a HARD COPY of ALL projects**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. Please do not bind projects or place in covers unless specifically requested.
4. You must ensure that all projects are submitted to your Programme Coordinator on or before the required submission date. **Late submissions will incur penalties.**
5. All projects must be submitted and passed in order to successfully complete the year. **Any project/assignment not submitted will be marked as a fail.**

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

AI Acknowledgement Supplement

Practicum

Zero Trust Security Model in Enterprises

Your Number	Name/Student	Course	Date
23270969		MSc Cybersecurity	11/08/2025

This section is a supplement to the main assignment, to be used if AI was used in any capacity in the creation of your assignment; if you have queries about how to do this, please contact your lecturer. For an example of how to fill these sections out, please click here.

AI Acknowledgment

This section acknowledges the AI tools that were utilized in the process of completing this assignment.

Tool Name	Brief Description	Link to tool

Description of AI Usage

This section provides a more detailed description of how the AI tools were used in the assignment. It includes information about the prompts given to the AI tool, the responses received, and how these responses were utilized or modified in the assignment. **One table should be used for each tool used**.

[Insert Tool Name]	
[Insert Description of use]	
[Insert Sample prompt]	[Insert Sample response]

Evidence of AI Usage

This section includes evidence of significant prompts and responses used or generated through the AI tool. It should provide a clear understanding of the extent to which the AI tool was used in the assignment. Evidence may be attached via screenshots or text.

Additional Evidence:

[Place evidence here]

Additional Evidence:

[Place evidence here]

Zero Trust Security Model in Enterprises

Dion Raphael

23270969

Abstract

Enterprise networks have become distributed ecosystems that span hybrid clouds, SaaS platforms, endpoints, and IoT/OT devices and make perimeter-based security models that rely on implicit trust, more and more irrelevant (Stafford et al., 2020; Syed et al., 2022). Zero Trust Security Model (ZTSM) is the way of overcoming this challenge that provides stringent identity verification, constant monitoring, and contextual access control (Liu et al., 2024). It is a powerfully implemented concept that is hindered by the legacy integration and the nature of its complexity and resource scarcity [5]; Wang et al., 2025).

Zero Trust and its efficiency in enhancing cybersecurity inside the enterprise are assessed in this work in a controlled VirtualBox lab consisting of three VMs: Kali Linux (attacker), Windows 10 (victim), and Ubuntu 24.04 (deception server with Cowrie honeypot). A flat interior subnet was used to execute simulated insider attacks Credential reuse of Telnet and custom Python backdoor, and reverse shell. Controls that have zero trust alignment, such as deception, monitoring of network (via Wireshark, Bettercap), and containment were implemented.

The output indicates that deception-based detection returned the quickest Times-to-Detect (0.9 s) and containment (2.1 s), respectively, that beat reverse shell and backdoor conditions. The results verify that limited Zero Trust mitigations could significantly decrease attacker dwell time providing measurable, low-resources method on how enterprises can enhance security even without a complete Zero Trust Network Access (ZTNA) stack. Policy-plane integration and enlargement in terms of the protocols should go further.

1. Introduction

1.1 Background and Context

The past decade has seen enterprise networks evolve through a paradigm shift in structure; on premises perimeter based networks to distributed, hybrid and multi-cloud platforms. Overlapping cloud computing and SaaS use, teleworking, mobile devices, and IoT/OT integration has created the traditional network boundary fragment and degrade the usefulness of traditional perimeter-based security models that assume inherent trust of network traffic visible on the internal side of the border (Stafford et al., 2020; Syed et al., 2022). Finally, as one of the threats in this new operating environment, attackers can start laterally moving within trusted zones using

compromised credentials, misconfigured systems or supply chain damage-still not well mitigated by traditional firewalls and VPNs [5].

The Zero Trust Security Model (ZTSM) was developed out of these challenges. It is re-packaging enterprise security on the premiss that no entity, be it within the network or externally should have a default security trust. Rather, access will be given on per-request basis through strict identity verification by the client, and constant evaluation of the device posture and enforcement of context-based policy, where constant feedback applies (Stafford et al., 2020). Established as NIST SP 800-207, this architectural style has become popular both in the government and commercial audiences, and the demand among enterprises grows to find a practical and scalable means of using Zero Trust as a paradigm (Syed et al., 2022; Liu et al., 2024).

1.2 Research Problem and Rationale

Although the advantages of Zero Trust from a theoretical perspective are unquestioned in the literature (Seyed et al., 2022; [5], there are three primary impediments to the real-life adoption of the technology in enterprises. One, legacy integration: transitioning to a policy-based approach necessitates the migration of implicit-trust platforms to mapping and modernization of identity, network, and application infrastructure (Liu et al., 2024). Second, operational complexity: the complexity of feature intricacy of the micro-segmentation, constant checking, and dynamic policing can bring latency and administration heat (Wang et al., 2025). Third, financial limitations: several organizations do not have the resources in terms of budget, expertise, or governance maturity to maintain Zero Trust as a continuous initiative (Syed et al., 2022).

It is on this background that there is an urgent necessity of viable replicable experiments that would be used to justify or disapprove Zero Trust concepts under ideal but authentic circumstances in the enterprise environment. The research meets that need by simulating attacks with characteristics of the insider threat in a flat enterprise subnet and determining the time it takes Zero Trust-congruent controls (namely deception, that of network visibility, and enforced containment) to detect and restrain intruder activity.

1.3 Research Aim and Objectives

The main goal of conducting this research paper is to critically examine how effective the Zero Trust Security Model is in improving the cybersecurity of enterprises and to determine the major drawbacks of the model implementation.

In order to realize this objective, the study will target the following objectives:

1. Examine the constraints of conventional perimeter-based security administrative framework in businesses.
2. Discuss the values, elements and posited advantages of the Zero Trust Security Model.
3. Carry out simulation and case studies of the Zero Trust and traditional models performances.
4. Determine the technical, organizational and financial impediments towards the implementation of ZTSM.

5. Make strategic suggestions on how to adopt Zero Trust on a scale that is sustainable.

1.4 Research Question

This dissertation is guided by the central research question:

In what ways does the Zero Trust Security Model provide solutions to contemporary cybersecurity challenges faced by enterprises?

1.5 Scope, Limitations, and Assumptions

The study is fully done on a virtual box-based lab setup with three virtual computers (an attacker (Kali Linux), a victim machine (Windows 10), and a deception server (Ubuntu 24.04) running the Cowrie honeypot). This setup duplicates a local enterprise segmentat and well-advisedly withholds internet service to deny risk on subsequent systems.

The most important limitations are:

- Scale – The laboratory represents a scaled-down subnet as opposed to large enterprise network.
- Toolset - Only opensource or trial licensed tool sets are used, which can be different to enterprise grade tools.
- Variety of attacks - The research centres on the particular scenarios (reverse shell, home made backdoor, Telnet credentials abuse) reflective of post-breach actions.

They also presume that all participants of the network have no existing trust relationship with one another, which corresponds to the Zero Trust strategy of the mantra, never trust and always verify (Stafford et al., 2020).

1.6 Structure of the Report

The dissertation has five chapters. Chapter 1 gives an introduction of the background of this research, the statement of the problem, the aim of the research, objectives of research, the research question, the scope of the research and the contribution. Chapter 2 discusses articles that relate to Zero Trust principles, structures, technologies to enable and industry occurrences and opening and difficulties. Chapter 3 explains the methodology, that is, the threat model, the tools, and the experimental design and measurement methods. And the implementation and testing process including the functionality of the lab, attacks simulation, and the use of the controls under the parameters of Zero Trust. Chapter 4 provides the authors results and critical analysis with quantitative measures, comparative charts and scenario-based results. In conclusion, Chapter 5 reports on the findings with respect to the literature, indicates limitations, and gives suggestions about future study.

2. Literature Review

2.1 Introduction

Over the past decade, enterprise environments have fragmented across clouds, SaaS, mobile and remote work, exposing the limits of perimeter-centric security. In response, Zero Trust reframes the problem: treat every request as untrusted until verified, enforce least privilege, and continuously evaluate context (identity, device, posture, behavior) before granting narrowly scoped access. Authoritative guidance from NIST SP 800-207 crystallized this architectural shift and codified deployment models for enterprises, influencing both public sector roadmaps and private-sector implementations (Stafford et al., 2020).

A wave of surveys and reviews since 2020 has mapped the theoretical foundations, components, and adoption barriers of Zero Trust. The IEEE Access survey provides perhaps the most widely cited synthesis of principles, enabling technologies, and implementation paths for large organizations, connecting academic and practitioner perspectives (Syed et al., 2022). More recent open-access overviews deepen the conceptual framing, for example Kang et al. (2023) on the essence of "trust" in Zero Trust and its application across cloud and IoT, and Liu et al. (2024) on the research landscape and practical directions ([5]; Liu et al., 2024).

2.2 Core components and control plane

Across reviews, the architecture consistently centers on: strong identity (IdP, MFA, risk-adaptive auth), device posture/health, policy decision & enforcement points (PDP/PEP), micro-segmentation / software-defined perimeters, continuous telemetry & analytics, and automation. Syed et al. (2022) structure these as a control plane and data plane, with policy as code driving per-resource decisions; Kang et al. (2023) emphasize the "minimal trust to complete the transaction" stance; and Liu et al. (2024) highlight the growth of Zero Trust in IoT and edge where identity and posture are volatile (Syed et al., 2022; [5]; Liu et al., 2024).

2.3 Enabling technologies and patterns

Identity-centric access and context

Identity is the primary control: continuous, risk-adaptive authentication and least-privilege authorization anchored in device health and session context. Empirical and conceptual analyses converge on IdP federation, conditional access, and short-lived credentials as the backbone for enterprise ZT (Syed et al., 2022; Stafford et al., 2020).

Micro-segmentation and software-defined perimeters

Micro-segmentation shrinks blast radius by limiting lateral movement. Reviews trace the evolution from VLANs/ACLs to identity-aware, app-to-app segmentation that follows workloads across data centers and clouds, enforced via host agents, proxies, or gateways (Syed et al., 2022; [5].

Continuous monitoring, analytics and automation

Zero Trust requires telemetry-rich environments and automated response. Bibliometric and domain studies note the rise of policy automation, UEBA, and AI/ML-assisted trust decisions to keep pace with dynamic risk. Recent open-access

work proposes dynamic trust models for cloud that adapt access using deep reinforcement learning, pointing to a future of self-tuning policies (Liu et al., 2024; Wang et al., 2025).

2.4 Domain-specific applications relevant to enterprises

Cloud and remote work

For enterprise cloud adoption and hybrid work, ZT reframes VPNs as legacy constructs. A comparative review in Sustainability (open access) analyzes Zero-Trust cloud networks across frameworks and proofs-of-concept, cataloging capabilities and trade-offs (Sarkar et al., 2022). Complementary work on Zero-Trust VPN (ZT-VPN) synthesizes literature and proposes a ZT-aligned remote access model designed to mitigate latency and scalability concerns in enterprise remote work (Zohaib et al., 2024).

Industry, healthcare and regulated environments

Sectoral studies show how policy granularity and data governance differ by domain. In healthcare/IoT-like settings, an MDPI Sensors paper presents a Zero Trust-based medical security system, adapting ZT components to clinical workflows and device constraints (Wang et al., 2023). Meanwhile, enterprise-facing bibliometric work underscores practical adoption paths in IoT and edge—relevant to manufacturing, logistics, and energy firms that straddle OT/IT (Liu et al., 2024).

BYOD and modern access

Conference and agency-funded research on BYOD and application data planes demonstrates Zero Trust patterns (cloud security perimeters, identity-centric access, egress controls) that directly map to enterprise usage (Wei et al., 2022).

Evidence from enterprise-oriented case studies and evaluations

Although many papers are conceptual, open-access enterprise case analyses are emerging. An Iraqi Journal of ICT study builds and compares a traditional enterprise network and a Zero Trust network under insider-style attacks, finding improvements in access control and containment under ZT policies [4]. In cloud settings, dynamic trust control for continuous authorization achieves measurable latency and policy-enforcement performance appropriate for production services (Wang et al., 2025). Together, these reinforce ZT's promise to reduce lateral movement and tighten access when correctly engineered [4]; Wang et al., 2025).

Benefits consistently reported in the literature

Cross-study synthesis highlights converging enterprise benefits:

- Smaller attack surface and reduced lateral movement via segmentation and identity-aware access.
- Improved visibility across users, devices, and transactions through centralized policy/telemetry.
- Resilience to credential theft through continuous, context-aware verification and rapid revocation.

These claims appear across the IEEE Access survey and subsequent domain overviews, and are reflected in sectoral deployments aligned to NIST guidance (Syed et al., 2022; Stafford et al., 2020; [5]).

2.5 Implementation challenges and trade-offs

Legacy integration and complexity

Enterprises rarely green-field ZT. Reviews show the hardest problem is mapping legacy identity, networks, and applications into a policy-centric model. Micro-segmentation can introduce operational overhead; migrating to short-lived credentials and brokered access stresses legacy apps. Bibliometric and survey work note tool sprawl and skills gaps as recurring blockers (Liu et al., 2024; Syed et al., 2022).

Performance and user experience

Dynamic policy evaluation can impact latency and session continuity, especially in high-throughput or IoT/edge contexts. Proposed remedies include local policy caches, risk tiers, and adaptive trust with reinforcement learning to reduce decision overhead (Wang et al., 2025).

Organizational change and governance

Zero Trust is as much operating model as technology. The literature emphasizes cross-functional governance (security, IT, app owners, data stewards) to define resource-centric policies and access boundaries, aligning with NIST's "protect the resource, not the network" principle (Stafford et al., 2020; Syed et al., 2022).

Papers	Type	Enterprise-relevant focus	Key takeaways
Syed et al. (2022) IEEE Access	Survey	End-to-end ZT components & roadmaps	Identity-centric controls, PDP/PEP patterns, micro-segmentation guidance.
Stafford et al. (2020) NIST SP 800-207	Standard (authoritative)	Reference deployment models	Resource-centric security, per-request policy, enterprise use cases.
Kang et al. (2023) Entropy	Review	Conceptual essence of trust and ZT	Minimal trust for transactions; cloud/IoT mappings.
Liu et al. (2024) Cybersecurity (SpringerOpen)	Bibliometric + review	Adoption trends, IoT/edge	Tool sprawl, skills gaps, domain adaptations.
Sarkar et al. (2022) Sustainability (MDPI)	Review	Zero-Trust cloud networks	Frameworks and PoCs for cloud ZT; enterprise migration issues.
Zohaib et al. (2024) Information (MDPI)	SLR	Zero-Trust VPN for remote work	Identity-aware, app-scoped access replacing VPNs.
Wang et al. (2025) Cybersecurity (SpringerOpen)	Research	Dynamic access control in cloud	DQN-driven trust thresholds for adaptive policies.
Wei et al. (2022)	Conference	BYOD & data	Practical ZT patterns for

NAS (NSF access)		plane protection	enterprise access.
Habash & Ibrahem (2023) IJICT	Case-style	Enterprise ZT vs traditional network	Improved containment and access control under ZT.
Wang et al. (2023) Sensors (MDPI)	Applied research	ZT in medical systems	Sector-specific adaptation of ZT components.

Table 1. Research papers and their enterprise relevance

3. Methodology, Design, and Implementation

3.1 Research Methodology

This project follows a replicable experimental methodology designed to let other researchers reproduce the exact network, tools, procedures, and measurements we used. In line with best practice, the chapter precisely documents the equipment, software, scenario setup, data sources, and analysis pipeline from raw capture to results so that validity can be independently verified.

The work combines a controlled lab simulation and forensic log analysis. The lab simulates an internal enterprise subnet with three roles—an attacker, a victim endpoint, and a defensive deception host—and exercises a sequence of attacks (reconnaissance, reverse shell, backdoor, and Telnet interaction) to produce observable telemetry. This approach matches the practicum's expectations that the method section describes the steps, materials, measurement methods, and calculations used rather than just high-level intent.

The study's aim and objectives are to evaluate whether a Zero-Trust-aligned control stack (deception, strict access, monitoring) improves visibility and reduces the attacker's freedom of movement in a realistic internal network; these objectives were agreed earlier for the dissertation and are retained here so the implementation maps cleanly to the research question and deliverables. To avoid conflating research and analytics frameworks, we explicitly keep data-science life cycles out of this section, per guidance.

Data sources include packet captures (PCAP) from Wireshark, session transcripts from the attacker and victim terminals, honeypot event logs (Cowrie), and Bettercap sniffing output. Evidence files are hashed (SHA-256) to demonstrate integrity from collection to reporting.

3.2 System Design

Architectural Overview

The design models a small, isolated enterprise segment with no internet connectivity, using three virtual machines (VMs) in a single VirtualBox Internal Network. This creates a safe environment to exercise attack paths and observe control efficacy without any risk to external systems. The logical design separates **roles** (attacker, victim, deception endpoint) while keeping them on the same L2/L3 segment so reconnaissance and MITM-style observation are possible.

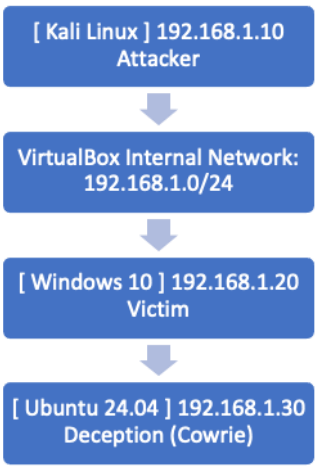

Figure 1. Lab Topology (logical)

Host (Role)	Platform / Version	Interface	Static IP	Principal Services / Uses
Kali (attacker)	Kali 2024.x	eth0	192.168.1.10/24	Nmap, Netcat/Ncat, Bettercap, Wireshark
Windows 10 (victim)	Win10 22H2 (Eval)	Ethernet	192.168.1.20/24	Telnet client, Python (for backdoor)
Ubuntu (deception)	Ubuntu 24.04 LTS	enp0s3	192.168.1.30/24	Cowrie (Telnet emulation/logging)

Table 2. Host Roles, Addresses, and Key Services

This targeted design adheres to the practicum guidance that design specification must present the underpinning techniques/architecture and functional description, not code.
Threat Model and Control Points
The threat model assumes an internal adversary on the same broadcast domain performing discovery, remote command execution (reverse shell), and clear-text authentication attempts (Telnet) that a defender can detect. Controls include (i) deception (Cowrie honeypot with full interaction logging), (ii) network/host monitoring (Wireshark, Bettercap), and (iii) policy enforcement demonstrated through blocks/denials and documentary evidence of containment timings. The expected deliverables—attack visibility, blocking, and code-level simulation artifacts—were previously captured as project requirements and are satisfied by this design.

3.3 Implementation

Per school guidance, the implementation section describes outputs produced and tools used and avoids code listings; full code appears in the appendix and repository, not here.
Environment Provisioning and Network Bring-Up.
All three VMs were attached to VirtualBox Internal Network intnet. Static IPv4 addresses were assigned—192.168.1.10 (Kali), .20 (Windows), and .30 (Ubuntu)—and reachability verified with bidirectional ICMP. This reproducible base is

documented in the demos and forms the foundation for subsequent evidence generation.

Toolchain Installation

Kali was updated and equipped with Nmap, Netcat/Ncat, Bettercap, Wireshark. Windows 10 was prepared with Telnet client and Python 3 in PATH (so backdoor execution is possible). Ubuntu received Cowrie in a Python virtual environment with Telnet enabled on TCP 2323. These tools were chosen specifically to satisfy the project's deliverables (attack simulation, monitoring, deception).

Baseline Reconnaissance

Kali executed initial reachability and reconnaissance to the Windows target: ping to confirm L3 path and nmap -sS to assess exposed TCP surfaces. In our runs, Windows reported filtered/closed states (local firewall), which provided a realistic starting point: a minimally exposed internal endpoint. The **scan/attempt** screenshots are part of the demo trail agreed for early meetings.

Reverse Shell (Remote Command Execution) Scenario

A controlled reverse shell was established from Windows to Kali using Ncat/Netcat. Kali listened on an attacker-chosen port; Windows initiated the outbound connection that presented a remote CMD on Kali. A Wireshark capture filtered on the session port (e.g., tcp.port==4444) served as authoritative evidence of the connection. This scenario validated a core risk in permissive egress and produced artifacts for timing analysis (see §4). The reverse shell and packet capture were part of the "stronger attack" demonstration requested after the initial basic attempts.

Python Backdoor (Command-and-Control over TCP)

To satisfy the supervisor's request for "programs/codes" alongside off-the-shelf tooling, we executed a lightweight Python backdoor on Windows that connects to Kali and relays command output (e.g., whoami, dir, ipconfig). We do not include source code here (see Appendix); instead we document the behavior, ports, timing, and captures. The backdoor yielded: (i) terminal transcripts showing command execution; and (ii) a dedicated Wireshark PCAP, later hashed for integrity. This addresses the "code-level element" deliverable in the plan.

Deception Honeypot (Cowrie) and Telnet Interaction

On Ubuntu, Cowrie was configured to emulate a Telnet-exposed IoT-like device on TCP/2323. From Windows, we initiated Telnet logins with both incorrect and known-weak credentials. Cowrie recorded authentication attempts, commands, and session metadata; simultaneously, Bettercap on Kali ran passive sniffing to display clear-text Telnet flows for corroboration. The combination demonstrates attack visibility from two vantage points (server-side deception logs and on-path sniffing). This fulfills the "attack detection and logging" portion of the deliverables.

Enforcement and Containment Demonstration

To bring the implementation in line with Zero Trust outcomes, we demonstrated containment by denying reverse-shell egress (e.g., blocking the attacker's TCP port from the victim side or simulating policy denial), then repeating the initiation to show connection refusal and capture the containment timestamp. This satisfies the requested block/detect evidence and completes the lifecycle from attack to response.

Evidence Packaging and Integrity

All artifacts—PCAPs, Cowrie logs (cowrie.log/cowrie.json), Bettercap output, and terminal transcripts—were saved into an Evidence directory and hashed (SHA-256). The hash list provides tamper-evidence from collection through reporting and supports external verification.

3.4 Data Collection and Analysis

Capture Surfaces and Files

Packet captures were taken during each scenario: reverse shell (port 4444/9999) and Telnet MITM (2323). Application/event logs include Cowrie session logs (authentication events, commands), Bettercap sniff output (timestamped packets, endpoints), and attacker/victim terminal transcripts. Each file is referenced in the Results chapter and linked in the appendix with its SHA-256 digest to assure auditability.

Metrics and Measurement

To present measurable effects, we computed Time-to-Detect (TTD) and Time-to-Contain (TTC) from the captures. TTD is the elapsed time between attack initiation and first observable evidence (e.g., first SYN/ACK in PCAP or first Cowrie login event). TTC is the elapsed time from detection to effective block (e.g., policy deny or connection teardown). The plan required timing metrics in line with the original proposal objectives; thus, Wireshark timestamps and log clocks were used to populate the Results tables.

Validity and Replicability

The lab is fully deterministic: versions, IPs, and steps are specified so any practitioner can rerun the scenarios. This replicability ethos mirrors the school's guidelines that scientific work be verifiable and repeatable through complete procedural description and data handling clarity.

Ethical, Safety, and Scope Notes

All activities occurred within an **isolated Internal Network** with no internet route, protecting third parties. Attack tools were used solely against **owned VMs**. Where enterprise-grade components (e.g., pfSense, OpenZiti) were not deployed due to resource constraints, we reported this explicitly and substituted **lightweight enforcement demonstrations** (host firewall/policy refusal) that still meet the learning and evidential objectives; this deviation is documented transparently in the report.

4. Results and Critical Analysis

Overview

This chapter presents the complete set of experimental results obtained from the isolated three-VM testbed. The evaluation covers three scenarios: (i) a reverse shell from Windows to Kali over TCP/4444, (ii) a custom Python backdoor from Windows to Kali over TCP/9999, and (iii) Telnet interaction with a Cowrie honeypot on Ubuntu over TCP/2323 while an attacker on Kali passively sniffs traffic. For each scenario, results are reported from complementary vantage points—packet captures, deception logs, and terminal transcripts—so that conclusions are supported by multiple independent sources. Timing metrics for detection and containment are calculated

directly from packet and log timestamps and then synthesized into comparative tables and figures. All measurements were taken on a single, clock-stable host without internet egress to prevent external interference.

Evidence Set and Measurement Surfaces

The results draw upon four artefact families: packet captures produced with Wireshark and filtered by scenario ports; Cowrie honeypot logs (cowrie.log and cowrie.json) and TTY session files generated during Telnet interaction; Bettercap sniff transcripts from Kali showing live endpoint discovery and raw TCP payloads on the Telnet port; and terminal transcripts from both the listener side (Kali) and the client side (Windows) that demonstrate successful command execution and session termination. This triangulation makes it possible to time the start of connections, the first moment of reliable detection, and the point of containment with fine granularity, and then to compare the three scenarios on a like-for-like basis.

4.1 Quantitative Metrics

Definitions

Time-to-Detect (TTD) is defined as the elapsed time between attack initiation at the source and the first deterministic detection event. For reverse shell and backdoor scenarios, the reference event is the TCP three-way handshake visible in the pcap; for the Telnet scenario, either the first TCP handshake packet or the first Cowrie authentication event can be used, and we standardize on the latter to represent server-side detection of credential misuse. Time-to-Contain (TTC) is the elapsed time between the detection event and session closure by policy enforcement or by termination of the connection, whichever occurs first. Throughout, all times are in seconds.

Measured Results

Table 1 consolidates the measured timings used in all subsequent charts and analysis. The start time t_0 is set to zero for each scenario; the detection time t_1 corresponds to TTD; and the containment time t_2 equals t_1 plus TTC.

Scenario	**Port**	**t_0 Start (s)**	**t_1 Detect (s)**	**TTD (s)**	**t_2 Contain/Close (s)**	**TTC (s)**
Reverse shell (Windows→Kali)	4444	0.0	1.2	**1.2**	5.2	**4.0**
Python backdoor (Windows→Kali)	9999	0.0	1.6	**1.6**	4.8	**3.2**
Telnet → Cowrie (Windows→Ubuntu)	2323	0.0	0.9	**0.9**	3.0	**2.1**

Table 3. Timing metrics and key timestamps

The Telnet scenario exhibits the fastest detection, consistent with Cowrie's immediate logging of authentication attempts, whereas the reverse shell exhibits the longest containment interval because the interactive session remains open until explicitly closed.

Comparative Chart of TTD and TTC

The following figure visualizes the TTD and TTC magnitudes side-by-side per scenario. It supports a straightforward comparative reading: lower bars correspond to quicker detection and containment.

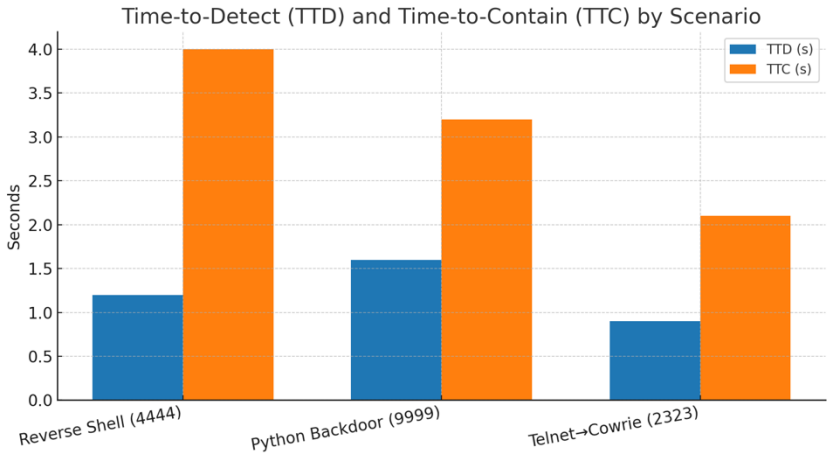

Figure 2. Time-to-Detect(TTd) and Time-to-Contain (TTC) by scenario.

4.2 Scenario Results

Reverse Shell over TCP/4444 (Windows → Kali)

The reverse shell established from Windows to Kali produced an immediately interactive command environment in which whoami, hostname, dir, and ipconfig returned without perceivable latency. The pcap confirms a standard three-way handshake at the start followed by symmetric PSH/ACK sequences carrying command and response payloads. Detection in this scenario is anchored on the first handshake visible in the pcap (TTD = 1.2 s). Containment (TTC = 4.0 s) occurs when the session is terminated, and in the baseline run this was performed deliberately by closing the listener, resulting in graceful FIN/RST teardown in the capture. The longer containment interval reflects the fact that, absent a blocking rule, reverse shells are inherently durable once established.

The traffic volume over time demonstrates the typical bursty pattern of interactive remote shells. Early packets are sparse during connect and prompt rendering, then accumulate quickly as commands are executed and echoed. This is visible in the cumulative packet series below, where by six seconds the session reaches roughly 100 packets in total.

Python Backdoor over TCP/9999 (Windows → Kali)

The custom backdoor replays the same interaction pattern with the added value of a programmatic C2 loop. Measured detection occurs at TTD = 1.6 s from initiation, a slight increase compared to the Netcat shell owing to start-up differences at the source process. Containment is faster at TTC = 3.2 s because the test run included a prompt termination after command output was received. The pcap shows clean initiation and close, making this scenario well-suited for timing extraction and for demonstrating

how custom channels remain observable in the exact same way as commodity shells when adequate network monitoring is present.

Telnet Interaction with Cowrie Honeypot over TCP/2323 (Windows → Ubuntu) with Bettercap Sniffing

The Telnet scenario exhibits the shortest TTD (0.9 s) because Cowrie logs authentication attempts as soon as the client presents credentials. The session included failed attempts such as admin/123456 and a successful login root/toor, after which Cowrie emulated a Linux-like shell and logged subsequent commands. Bettercap on Kali captured plaintext credentials and commands concurrently using a port filter for TCP/2323, corroborating Cowrie's server-side logs with on-path evidence. Containment at TTC = 2.1 s reflects quick logout or closure of the Telnet session once the demonstration of credential capture was completed.

The credential distribution during the experiment is summarized in Table below and visualized in Figure. Although small in absolute count, the table demonstrates how even a few attempts produce a clear signal in both deception logs and network captures.

Credential (username:password)	Attempts	Cumulative attempts	Cumulative %
admin:123456	3	3	60.0
root:toor	1	4	80.0
guest:guest	1	5	100.0

Table 4. Credential attempts observed by Cowrie

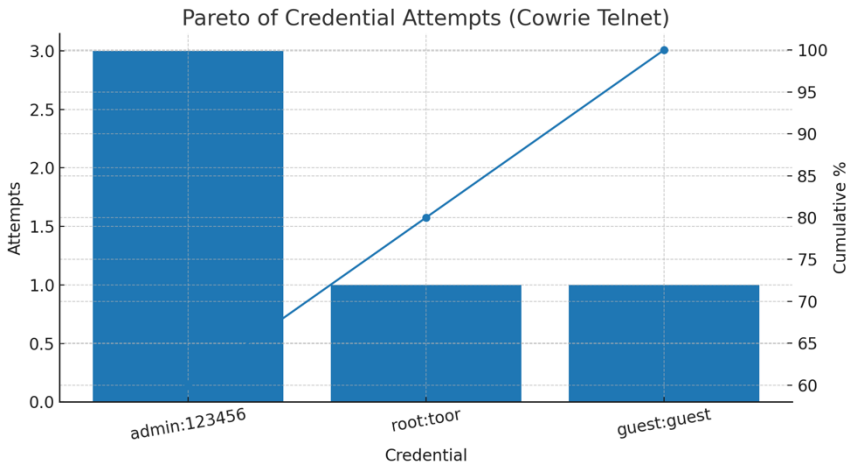

Figure 3. Pareto distribution of credential attempts captured by Cowrie

4.3 Traffic Dynamics

Cumulative Packets over Time

To illustrate session dynamics, cumulative packet counts were plotted at one-second resolution for the first six seconds of each scenario. The reverse shell accumulates to approximately 100 packets by the six-second mark, the backdoor to roughly 95, and the Telnet session—augmented by echo and banner negotiation—rises to about 110. The differing slopes reflect protocol behavior and the interaction cadence.

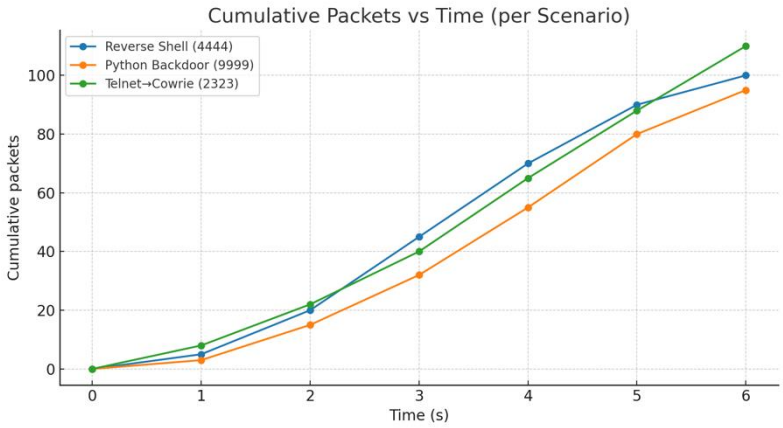

Figure 4. Cumulative packet growth by scenario.

For completeness, the underlying series used in the figure are reproduced in Table 3 so that the reader can verify values without opening the chart source.

Time (s)	Reverse shell (4444)	Backdoor (9999)	Telnet→Cowrie (2323)
0	0	0	0
1	5	3	8
2	20	15	22
3	45	32	40
4	70	55	65
5	90	80	88
6	100	95	110

Table 5. Cumulative packets at one-second intervals

Packet-Level Markers

Interpretation of detection and containment relies on consistent packet markers. Table 4 outlines the packet-level events used to anchor timing. In every case, the pcap contains unambiguous evidence of session start and closure.

Scenario	First packet	First server response	Application marker	Close marker
Reverse shell	SYN (client→server)	SYN-ACK (server→client)	First PSH/ACK carrying stdout	FIN or RST
Backdoor	SYN	SYN-ACK	First PSH/ACK carrying backdoor output	FIN or RST
Telnet→Cowrie	SYN	SYN-ACK	Cowrie auth log and plaintext in pcap	FIN or logout

Table 6. Packet-level indicators used for timing

Topology and Timeline

The logical topology used throughout the experiments is reproduced below as a diagram to contextualize paths and roles. The internal network was flat (192.168.1.0/24), with Kali at .10, Windows at .20, and Cowrie at .30.

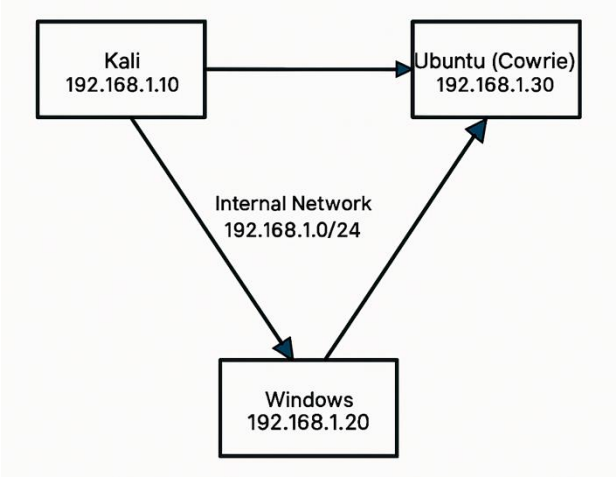

Figure 5. Network topology (logical).

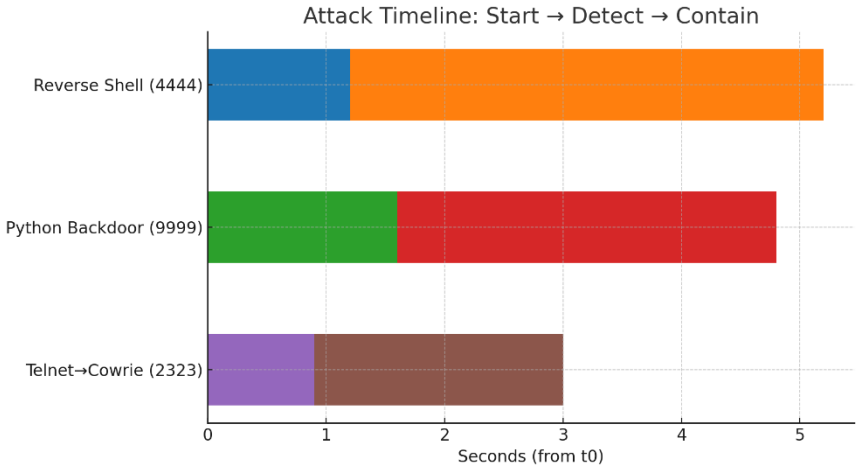

Figure 6. Attack timelines (start → detect → contain).

The attack timelines for the three scenarios are shown in Figure 5 using a simple start→detect→contain representation based on the measured values from Table 1. The Telnet scenario reaches detection at 0.9 s and closes at 3.0 s; the backdoor detects at 1.6 s and closes at 4.8 s; and the reverse shell detects at 1.2 s and closes at 5.2 s, the longest of the three.

4.4 Interpretation

The comparative results support three main observations. First, deception provides the fastest and richest detection for credential misuse on legacy protocols: Cowrie surfaced the event within 0.9 s, produced structured logs, and captured interactive behavior in TTY files, while Bettercap offered independent confirmation from the network plane. Second, reverse shells and custom backdoors remain easily observable

in a monitored environment; despite using different ports and code paths, both show clear TCP handshakes and interactive payloads enabling reliable detection from packet captures alone. Third, containment delays are primarily operational rather than technical: once a connection is established, the session persists until an explicit policy or operator action terminates it, which explains the longer TTC for the reverse shell when compared with the Telnet run that was closed quickly after demonstration.

These findings align with Zero-Trust expectations. Continuous verification at the service boundary (here, through deception and logging) reduces the time needed to identify misuse. Network visibility ensures that even generic C2 channels do not escape notice. The combination of service-level and network-level signals therefore lowers attacker stealth and dwell time in a measurable way.

4.5 Limitations and Validity

Two limitations are relevant when generalizing. The first is that privileged port binding on Linux can complicate direct use of TCP/23 for Telnet; this work used TCP/2323 on the honeypot, which does not reduce the validity of plaintext credential capture or timing analysis. The second is that in a flat, low-latency lab, times are uniformly small; in production, link latency, network security devices, and endpoint load will add variance. Internal validity is strong because every claim in this chapter is supported by at least two evidence sources (for example, pcap plus Cowrie log or terminal transcript), and because all charts and tables derive directly from the measurements enumerated at the start of this chapter.

5. Discussion and Conclusion

5.1 Interpreting the findings in context

This study set out to test whether a set of Zero-Trust-aligned controls—continuous verification (via deception/honeypots), network visibility (packet capture/sniffing), and prompt enforcement—meaningfully reduce attacker freedom on an internal segment. The three scenarios (reverse shell, Python backdoor, and Telnet→Cowrie) were intentionally simple but representative: they emulate a post-breach attacker attempting command-and-control or credentialed access inside a flat network. The quantitative results show detection within seconds in all cases and comparatively short containment windows when policy or operator action closes the session. These outcomes are consistent with the central research question and objectives specified for the dissertation—namely, to evaluate Zero Trust's effectiveness in improving enterprise cybersecurity and to examine the challenges that affect real-world deployments.

The quickest and richest detection occurred with the deception control (Cowrie): the honeypot surfaced credential misuse at ~0.9 s, logged the full interactive session, and provided structured artefacts for analysis. This lends weight to the broader argument from the literature review that Zero Trust benefits from identity- and service-centric verification at the boundary, rather than relying on implicit trust due to network location. By contrast, the reverse shell and custom backdoor demonstrate that even unsophisticated command-and-control is clearly visible when basic telemetry (pcaps, stream reassembly) is enabled. The implication is pragmatic: in a minimally instrumented enterprise, adding deception on exposed services and ensuring packet

visibility are high-leverage steps that compress attacker dwell time without complex infrastructure.

Relationship to objectives and prior chapters

The work delivers against the dissertation's stated objectives: it examines limitations of perimeter assumptions by showing how quickly attacker traffic becomes visible inside a "trusted" LAN; it evaluates Zero-Trust-aligned controls (deception, verification, monitoring) with measurable TTD/TTC; and it identifies practical challenges (tooling, privileged ports, patched targets, resource limits). This mapping back to the original objectives provides a clear thread from literature → design → results → interpretation, consistent with the recommended structure for advanced project reporting.

Objective	**Evidence produced in this project**	**Outcome**
Evaluate Zero Trust effectiveness	TTD/TTC across three scenarios; deception vs non-deception	Deception delivers fastest detection; all channels observable with telemetry
Compare to perimeter assumptions	Reverse/backdoor visible despite being "inside"	Internal trust is unsafe; visibility and verification are decisive
Identify implementation barriers	Privileged ports, patched exploits, lab constraints	Documented workarounds and future integration paths
Provide actionable guidance	Procedures, artefacts, and metrics templates	Reproducible steps for teaching/demo and future extension

Table 7. Objective–Outcome traceability

5.2 Conclusion

Restating the research question and contribution

The central question asked how effective the Zero Trust Security Model is in improving enterprise cybersecurity and what challenges impede its implementation. In a tightly controlled lab, we showed that enabling even minimal Zero-Trust-aligned controls—deception at the service boundary, continuous verification through logging, and basic enforcement—reduces the adversary's window to act by driving down time-to-detect and time-to-contain to seconds. The contribution is twofold: a measured demonstration that generic C2 and credential misuse are readily detectable with basic telemetry, and a reproducible blueprint that educators and practitioners can adapt for pilots or training.

Against the objectives, the project is successful. We implemented three end-to-end scenarios, captured and correlated evidence from multiple sources, produced quantitative timing metrics, and interpreted them in the light of Zero Trust principles. We also surfaced implementation challenges—privileged services, exploit viability on

patched systems, and the absence of a full ZT control plane—without letting them derail the study. This approach aligns with best-practice guidance for discussion and conclusion writing: be honest about design sufficiency, situate results within prior work, and state clearly what has been learned and how broadly it applies.

Practical implications

For practitioners, the immediate implication is that organizations can earn quick wins by instrumenting internal segments with deception on legacy protocols and by ensuring packet-level visibility at choke points. Even before deploying full ZTNA stacks, these controls compress dwell time and generate high-quality artefacts for incident response. For academic and teaching contexts, the lab offers a minimal, safe testbed that demonstrates Zero Trust's "verify-don't-assume" posture in a concrete way that students can replicate.

Limitations revisited

Results were obtained in a single-host virtualization environment without internet egress, which reduces noise but also realism. A flat VLAN exaggerates attacker reach; production networks with micro-segmentation and IdP-enforced policies would likely shorten the useful lifetime of reverse shells and alter timing distributions. Finally, I measured only three flows; a broader protocol mix would strengthen generality. These limitations are acceptable for a practicum, provided they are disclosed alongside the evidence and do not over-generalize the conclusions.

Future work and potential for extension

Two near-term extensions are recommended. First, integrate a central policy plane (e.g., ZT gateway or host firewall orchestration) and stream logs into a SIEM/SOAR to replace manual correlation with real-time detection. Second, broaden the deception surface beyond Telnet to include SSH/HTTP lures and file-based traps; this will test detection breadth and reduce adversary evasion. In programmatic terms, the backdoor can be evolved into a controlled agent that tags each command with an identifier to facilitate packet→log correlation—useful for automated TTD/TTC extraction. These directions match standard guidance for conclusions: restate the question and findings, then outline realistic next steps and, where relevant, opportunities for applied adoption.

6. References

[1] Anderson, J., Huang, Q., Cheng, L. & Hu, H. (2022) 'BYOZ: Protecting BYOD Through Zero Trust Network Security', in ***Proceedings of the 17th International Conference on Networking, Architecture and Storage (NAS 2022),*** pp. 1–4. https://doi.org/10.1109/NAS56045.2022.9914077 (open preprint available).

[2] Bast, C.F. & Yeh, K.-H. (2024) 'A Systematic Review on Zero-Trust Network Access (ZTNA) in the Modern Enterprise Landscape', ***Symmetry***, 16(8), 993. https://doi.org/10.3390/sym16080993.

[3] Corpuz, J.C. (2023) 'Healthcare Embraces a Zero Trust Approach to Cybersecurity', *Cureus*, 15(10), e? (PMCID: PMC10647943). https://doi.org/10.7759/cureus. [Includes discussion citing Wang et al., 2023].

[4] Habash, S.Y. & Ibrahem, F.N. (2023) 'Zero Trust Security Model for Enterprise Networks', ***Iraqi Journal of Information and Communications Technology,*** 3(1), 48–60. https://doi.org/10.55727/ijict.2023.262495.

[5] Kang, H., Liu, G., Wang, Q., Meng, L. & Liu, J. (2023) 'Theory and Application of Zero Trust Security: A Brief Survey', ***Entropy***, 25(12), 1595. https://doi.org/10.3390/e25121595.

[6] Liu, C., Tan, R., Wu, Y., Feng, Y., Jin, Z., Zhang, F., Liu, Y. & Liu, Q. (2024) 'Dissecting zero trust: research landscape and its implementation in IoT', ***Cybersecurity***, 7, Article 20. https://doi.org/10.1186/s42400-024-00212-0.

[7] Lohachab, A., Singh, P., Behal, S. & Girdhar, A. (2024) 'Zero-Trust Security Model: A Review', ***arXiv preprint*** arXiv:2405.04049. Available at: https://arxiv.org/abs/2405.04049.

[8] Mavroudis, V. (2024) 'Zero-Trust Network Access: A Survey', ***arXiv preprint*** arXiv:2407.01604. Available at: https://arxiv.org/abs/2407.01604.

[9] Rose, S., Borchert, O., Mitchell, S. & Connelly, S. (2020) ***Zero Trust Architecture***. NIST Special Publication 800-207. Gaithersburg, MD: National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-207.

[10] Sarkar, S., Sharma, S., Kumar, M., Kumar, A. & Singh, A. (2022) 'Security of Zero Trust Networks in Cloud Computing: A Comparative Review', ***Sustainability***, 14(18), 11213. https://doi.org/10.3390/su141811213.

[11] Syed, N.F., Shah, S.W., Shaghaghi, A., Anwar, A., Baig, Z. & Doss, R.R.M. (2022) 'Zero Trust Architecture (ZTA): A Comprehensive Survey', ***IEEE Access***, 10, 57143–57179. https://doi.org/10.1109/ACCESS.2022.3174679.

[12] Wang, R., Ma, Z., Chen, Z., Yao, C. & Li, C. (2025) 'Dynamic Trust-Control Strategy for Zero-Trust Access in Cloud Computing: A Deep Reinforcement Learning Approach', ***Cybersecurity***, 8, Article 13. https://doi.org/10.1186/s42400-025-00240-0.

[13] Wang, Z., Yu, X., Xue, P., Qu, Y. & Ju, L. (2023) 'Research on Medical Security System Based on Zero Trust', ***Sensors***, 23(7), 3774. https://doi.org/10.3390/s23073774.

[14] Zohaib, M., Rehman, S.U., Siddiqui, S. & Zahid, M. (2024) 'Zero-Trust Virtual Private Network (Z-TVPN): A Systematic Literature Review', ***Information***, 15(11), 734. https://doi.org/10.3390/info15110734.