1 Introduction

Authentication attacks represent one of the most pervasive and damaging threats in today's cybersecurity landscape. With compromised accounts and credentials accounting for nearly one-third of all cybercrime incidents in Australia, and Microsoft reporting over 600 million identity attacks globally per day—99% of which are password-based—the need for effective defensive strategies has never been more critical.

This report provides a comprehensive analysis of authentication threat vectors and defensive countermeasures through both theoretical examination and practical implementation. The research evaluates prominent attack tools including Hashcat, Hydra, John the Ripper, and NetExec, alongside defensive solutions such as Fail2ban, Wazuh, pfSense, and Suricata. Through systematic tool comparison and selection, this analysis identifies the most relevant technologies for focused security research.

The practical component implements a realistic scenario involving a small web development business under attack, demonstrating real-world authentication vulnerabilities and defensive responses. Using Hydra for credential-based attacks against SSH and MariaDB services, and Fail2ban for automated threat mitigation, this research examines the effectiveness of common defensive measures against systematic password attacks.

The findings contribute to understanding authentication security challenges facing small-to-medium enterprises and provide practical insights into defensive tool deployment and effectiveness. This research aligns with MITRE ATT&CK framework methodologies, offering both academic rigor and practical applicability for cybersecurity professionals.

2 Threat Analysis

2.1 Authentication

According to the Annual Cyber Threat Report(ACTR) 2023-2024, the most frequently reported threat in Australia were Authentication attacks in the form of "compromised accounts or credentials"[1], accounting for nearly one-third of all cybercrime incidents nationwide. This prevalence underscores the critical importance of understanding and mitigating authentication-related threats in the current cybersecurity landscape.

Authentication breaches encompass a wide array of attack categories, each exploiting different weaknesses in user and system behavior. The most prominent include **phishing**, which leverages psychological manipulation to trick users into divulging credentials, and **brute force attacks**, where adversaries systematically guess passwords using automated tools. **Password spraying** represents a "low and slow" approach, targeting many accounts with common passwords to evade detection. **Credential stuffing** utilizes previously breached username-password pairs, often sourced from large-scale data leaks, to gain unauthorized access through automation. These categories highlight the multifaceted nature of authentication threats, demonstrating why password-based attacks remain a critical focus for security research and organizational defense.

2.2 Threats in the Authentication Landscape

2.2.1 Current Threat Landscape

Authentication threats represent a critical challenge in today's cybersecurity landscape, both in Australia and globally. The Albanese Government's commitment of $15–$20 billion through 2033–34 to strengthen cyber domain capabilities underscores the national urgency to address these issues[1]. In the 2023–24 financial year, the Australian Signals Directorate received over 36,700 calls to its Cyber Security Hotline 12% increase from the previous year, reflecting the growing prevalence of cyber incidents[1]. Notably, compromised accounts or credentials accounted for **32% of all cybercrime**, making it the leading contributor to reported incidents. On a global scale, Microsoft Entra data reveals that of more than **600 million identity attacks per day**, over **99% are password-based**[2]. These statistics highlight the widespread and persistent nature of authentication-related threats, demonstrating why robust authentication mechanisms are essential for organizational and national security.

2.2.2 Typical Adversary Types

Authentication attacks are executed through several distinct methods, each exploiting specific vulnerabilities in user behavior and system design.

- **Phishing** is a psychological manipulation technique where adversaries impersonate trusted entities, often via email or messaging platforms, to deceive users into revealing their credentials. This method remains highly effective due to its reliance on social engineering rather than technical flaws.

- **Brute force attacks** involve systematically generating and attempting password combinations until a valid login is achieved. Attackers leverage automated tools to accelerate this process, targeting accounts with weak or commonly used passwords.
- **Password spraying** refines brute force tactics by distributing login attempts across many accounts using a shortlist of popular passwords, thereby evading detection mechanisms that monitor for rapid, repeated failures. A notable example is the 2024 Midnight Blizzard incident, where attackers employed password spraying against Microsoft's infrastructure, prompting significant defensive actions[3].
- **Credential stuffing** utilizes credentials harvested from previous data breaches, exploiting the widespread practice of password reuse across multiple services. The 2022 Optus breach illustrated the impact of this technique, as compromised credentials were repurposed to target other organizations[4]. These methods collectively demonstrate the evolving sophistication and persistence of authentication threats in the modern cyber landscape.

2.2.3 Impact

This section starts with how phishing and credential stuffing can be used as good starts

The organizational impact of authentication breaches is starkly illustrated by the 2022 Optus incident, which exposed sensitive data of nearly 10 million Australians and triggered widespread concerns over identity theft and fraud [4]. More recently, the Qantas breach in July 2025 affected 5.7 million customers, with varying degrees of personal information compromised [5]. Both cases highlight the persistent threat posed by credential stuffing, where attackers leverage previously leaked credentials to infiltrate systems at scale. The consequences include substantial financial losses, operational disruption, and enduring reputational harm, emphasizing the critical need for organizations to strengthen authentication controls and proactively address credential-based threats.

2.3 Threat Choice Justification

The selection of authentication attacks as the focal threat for this analysis is driven by their overwhelming prevalence and critical impact on both Australian and global organizations. Recent reports indicate that compromised credentials account for nearly one-third of all cybercrime incidents in Australia, with high-profile breaches such as Optus and Qantas underscoring the real-world consequences of these attacks[1], [4], [5]. The universal reliance on password-based authentication across industries makes this threat highly relevant and widely applicable. From a technical perspective, authentication attacks such as brute force, password spraying, and credential stuffing offer a rich landscape for analysis, enabling clear success and failure metrics and supporting laboratory-based experimentation.

While phishing remains a significant aspect of authentication security due to its effectiveness and prevalence, it is not easily testable in a controlled lab environment and will not be the focus of practical testing in this assignment. Nevertheless, its role in the broader authentication threat landscape will be acknowledged. Furthermore, the threat aligns strongly with established frameworks like MITRE ATT&CK (T1110 family)[6], providing a robust foundation for evaluating defensive strategies and mapping adversary techniques.

This combination of prevalence, technical depth, and framework integration makes authentication attacks an ideal subject for comprehensive security analysis. It will also be interesting to see how easy it is to break into hypothetical systems and gain a practical understanding of how secure my own personal passwords really are.

3 Overview of Attacker Tools

This section examines four prominent authentication attack tools to understand their capabilities, limitations, and practical applications in cybersecurity research. The analysis evaluates both offline password recovery tools (Hashcat, John the Ripper) and online network-based attack tools (Hydra, NetExec), providing a comprehensive overview of the current authentication threat landscape. By comparing these tools across different attack methodologies—from hash cracking to live network authentication testing—this research establishes the foundation for selecting the most appropriate tool for subsequent defensive analysis and laboratoryd

3.1 Comparison Table

3.2 Hashcat

Hashcat is a specialized tool for password-based attacks, focusing on scenarios where the attacker has access to known hash or encrypted data and aims to recover the original password or passphrase. It employs brute force,

Tool	Attack Type	Primary Use Case	Key Strengths
Hashcat	Offline hash cracking	Password recovery from hashes/encrypted files	450+ algorithms, GPU acceleration, high performance
Hydra	Online network attacks	Live service authentication testing	50+ protocols, real-time feedback, flexible attack modes
John the Ripper	Offline hash cracking	Educational/legacy password auditing	Rule engine, single mode, broad compatibility
NetExec	Network reconnaissance	Enterprise Active Directory assessment	Multi-protocol, BloodHound integration, lateral movement

Table 1: Attack Tools Comparison

dictionary, and hybrid attack methods to achieve this goal. Hashcat excels at cracking password hashes across more than 450 supported algorithms[7], recovering passwords for encrypted file formats, and deciphering offline authentication tokens.

Hashcat is limited to offline password recovery and cannot break mathematically sound encryption without access to passwords, attack live network services, or exploit flaws in cryptographic implementations. Its strengths lie in unmatched password recovery capabilities, broad algorithm support, and GPU-accelerated performance, making it a specialized tool for cracking password hashes and encrypted files rather than attacking network protocols or cryptographic weaknesses.

Rather than replacing network-based tools such as Hydra. Hashcat focuses on offline attacks against captured password hashes or encrypted files, while Hydra targets live authentication services by attempting to guess credentials in real time. Hashcat operates independently of network access, working in the dark to recover passwords from stored data. Once successful, the recovered credentials can be used to access systems without further brute force attempts.

3.2.1 Use Cases

- Cracking password hashes extracted from sources such as Windows "Ntds.dit" or "SAM" files, Linux "/etc/shadow", or other stored hash values supported by Hashcat's extensive algorithm coverage.
- Recovering lost passwords for encrypted files or authentication tokens

3.3 Hydra

Hydra is a versatile tool for conducting password-based attacks against live network services. Unlike Hashcat, which focuses on offline password recovery from hashes or encrypted files, Hydra targets active authentication endpoints by systematically testing username and password combinations in real time. It supports more than 50 protocols, including web services (HTTP, HTTPS), remote access (SSH, Telnet, RDP), file sharing (FTP, SMB), databases (MySQL, PostgreSQL), and email (IMAP, SMTP)[8]. Hydra is commonly used for brute force, password spraying, and credential stuffing attacks, providing immediate feedback on successful logins.

For this research, Hydra was chosen over Medusa[9] due to Medusa's lack of recent git commits and lower comparative popularity on GitHub, indicating less active development and community support. Medusa offers functionality comparable to Hydra, with a modular architecture that enables consistent command usage across various protocols. However, its lower popularity and less active development made it a secondary choice for this research.

3.3.1 Use Cases

- Testing the strength of passwords for web applications, remote access, and file sharing services
- Simulating password spraying attacks to evaluate organizational defenses
- Assessing exposure to credential stuffing

3.4 John the Ripper

John the Ripper is a foundational password cracking and security auditing tool that established many baseline techniques in the field. As the "original" password cracker[10], it operates primarily through offline hash cracking similar to Hashcat, while also offering limited online attack capabilities. John the Ripper exists in two main

variants: the classic mode with traditional password cracking algorithms, and the community-enhanced "Jumbo" version that supports over 200 hash formats including Unix/Linux systems, Windows environments, applications, databases, and cryptocurrency wallets[11].

John the Ripper distinguishes itself through its sophisticated rule engine for password transformations and unique "single mode" that leverages account information to generate targeted password candidates. While it shares offline hash cracking capabilities with Hashcat, John the Ripper's historical significance and educational value make it valuable for understanding classical password cracking techniques, though it operates at significantly slower speeds on modern GPU hardware.

3.4.1 Use Cases

- Educational environments for learning password cracking fundamentals
- Legacy system auditing where modern tools may not be compatible
- Specialized rule-based attacks leveraging account information
- Cryptocurrency wallet password recovery for older formats
- Cross-platform password auditing on resource-constrained systems

3.5 NetExec

NetExec is a network penetration testing tool designed to identify security vulnerabilities within enterprise networks by systematically testing credentials across multiple services and protocols. Operating under the philosophy that "you're only as strong as your weakest point," NetExec performs comprehensive credential validation across network infrastructure to locate authentication weaknesses[12]. The tool primarily targets Windows-specific protocols commonly used in enterprise environments, making it particularly effective for Active Directory assessments and post-exploitation activities.

NetExec distinguishes itself from single-protocol tools like Hydra through its multi-protocol approach and specialized Windows/Active Directory focus. While Hydra targets individual services, NetExec provides a comprehensive network-wide assessment capability, automatically testing credentials against discovered services and integrating with tools like BloodHound for attack path visualization[13]. This makes it valuable for both initial network reconnaissance and post-compromise lateral movement scenarios.

3.5.1 Use Cases

- Enterprise Active Directory assessment for automating credential validation and security posture evaluation across large Windows networks
- Post-compromise lateral movement through advanced credential attacks including pass-the-hash and Kerberos abuse
- Comprehensive domain security validation including delegation detection and Certificate Services enumeration
- Automated BloodHound data collection for attack path visualization and privilege escalation planning

---

3.6 Attack Tool Selection and Summary

After evaluating the four authentication attack tools, **Hydra** emerged as the most suitable choice for this research assignment. This selection was critical to establish before proceeding to defensive tool analysis, as it provides the foundation for understanding what threats need to be mitigated.

Hydra offers the ideal balance of capabilities for this assignment's scope. Its support for over 50 network protocols, real-time authentication testing, and immediate feedback mechanisms make it particularly valuable for understanding live network attack scenarios. The tool's ability to perform brute force, password spraying, and credential stuffing attacks directly aligns with the authentication threat landscape identified in the threat analysis section.

4 Overview of Defender Tools

4.1 Comparison Table

Tool	Attack Type Defended	Primary Use Case	Key Strengths
Fail2ban	Brute force, password spraying	Automated banning of IPs after failed logins	Real-time response, lightweight, highly configurable, persistent bans
Wazuh	Distributed brute force, credential stuffing, coordinated attacks	Centralized security event correlation and automated response	Unified XDR/SIEM, behavioral analysis, threat intelligence, scalable, open source
pfSense	Network-level brute force, high-volume attacks, external threats	Perimeter firewall, traffic filtering, threat intelligence integration	Network-wide protection, geographic/reputation IP blocking, real-time analysis, open source
Suricata	Automated credential attacks, rapid connection attempts, protocol abuse	Deep packet inspection, real-time threat detection	High-performance, passive/active modes, SIEM integration, rapid response

Table 2: Defender Tools Comparison

4.2 Fail2ban

Fail2ban is a UNIX based system (sorry windows), that operates by scanning log files and systemd journals using specified regular expressions (filter-rules) to detect authentication failures and other suspicious activities[14]. When failures exceed configured thresholds, fail2ban executes actions to ban the offending sources, typically by updating system firewall rules to reject new connections from those IP addresses for a configurable duration. The system operates through "jails" - configuration units that define which log files to monitor, what patterns constitute failures, and how many attempts within a specified time window trigger bans.

Fail2ban comes pre-configured to monitor standard log files for services like SSH and Apache, but can be easily customized to read any log file and detect any error pattern.

4.2.1 Hydra Defense

Fail2ban could effectively counter Hydra's authentication attacks by monitoring system logs for failed login patterns and automatically blocking source IP addresses after exceeding configured thresholds. When Hydra conducts brute force or password spraying attacks against SSH, HTTP forms, FTP, or other services, fail2ban detects the repeated authentication failures and implements immediate firewall blocks that terminate the attack session.

4.3 Wazuh

Wazuh is a free and open-source platform that unifies XDR (Extended Detection and Response) and SIEM (Security Information and Event Management) capabilities for protecting workloads across on-premises, virtualized, containerized, and cloud-based environments[15].

Wazuh's centralized architecture enables enterprise-wide correlation of security events, detecting coordinated attacks that may be missed by isolated systems. By integrating MITRE ATT&CK mapping and threat intelligence, Wazuh identifies complex attack patterns through behavioral analysis and cross-system event relationships. Detected threats trigger automated responses such as firewall updates, security alerts, compliance reporting and other various sets of functionality.

4.3.1 Hydra Defense

Wazuh can help defend against Hydra's authentication attacks by monitoring security events from many systems at once. This means it can spot brute force attacks that use multiple computers to avoid detection. Wazuh looks for patterns like repeated failed logins or attempts to guess passwords quickly. When it finds suspicious activity, it can automatically alert security staff, block the attacker's access, or record the incident for review.

4.4 pfSense

pfSense is a FreeBSD-based open-source firewall and router platform designed for network perimeter security[16]. Acting as a gateway, it proactively filters traffic and blocks threats before they reach internal systems. With integrated threat intelligence, geographic IP filtering, and reputation databases, pfSense can preemptively block known malicious sources. Its connection rate limiting and automated rule enforcement help prevent brute force and high-volume attacks at the network boundary, providing organization-wide protection through real-time traffic analysis.

4.4.1 Hydra Defense

pfSense can help protect against Hydra attacks by blocking traffic from suspicious locations and known bad IP addresses before it reaches your network. Its ability to limit the number of connection attempts and automatically block sources that try to log in too quickly makes it effective at stopping brute force attacks.

4.5 Suricata

Suricata is a network security tool that inspects data packets in real time to detect threats and suspicious activity. It uses predefined rules and signatures to spot attacks, unusual protocol behavior, and malicious content as traffic passes through the network. Suricata can work passively to monitor and log network activity for later analysis, or actively block threats when deployed inline. Its focus on analyzing network traffic at the packet level allows for quick detection and detailed investigation of attacks that may not show up in standard system logs.

4.5.1 Hydra Defense

Suricata could potentially detect Hydra attacks through deep packet inspection that identifies rapid TCP connection sequences and systematic authentication patterns characteristic of automated credential testing tools. When deployed in IPS mode, the platform might actively block detected attack traffic in real-time, though effectiveness could be limited against encrypted authentication protocols and distributed attacks that blend with legitimate network traffic.

4.6 Defender Tool Selection and Summary

Fail2ban was selected as the defensive tool for this assignment because it provides simple, automated protection against brute force authentication attacks like those performed by Hydra. It is lightweight, easy to configure, and works directly with standard Linux logs, making it ideal for small business or single-server environments. Unlike more complex or resource-intensive solutions, Fail2ban offers effective real-time blocking without requiring advanced expertise or additional hardware.

5 Scenario Design

5.1 Defender

A small, up-and-coming web development team fresh out of Swinburne operates with a single Linux server to keep costs low and offer competitive pricing. Without dedicated IT staff, the team members each have varying levels of technical expertise and share basic system administration duties. Their focus on affordability has resulted in limited security measures, exposing the business to risks while they strive to maintain essential services for customer orders and revenue.

5.1.1 Key Points

- Single Linux server directly exposed to the internet via a residential ISP using a generic home modem/router, no DMZ or dedicated firewall
- Router port forwarding: 22 (SSH), 80 (HTTP), 3306 (MariaDB) directly to server
- Flat network topology with server on same subnet as personal devices (192.168.1.0/24)
- Default service configurations with weak password policies
- Shared credentials and password reuse across team members No centralized logging or security monitoring capabilities

5.2 Attacker

A former classmate and friend of the students, feeling betrayed after discovering that the team is using his original business idea without credit, becomes determined to take revenge. Motivated by anger and a sense of injustice, he leverages his technical knowledge to conduct reconnaissance against his former friends' business. After seeing their company advertised on social media, he uses the website's domain name to identify their server's IP address through DNS lookups. Port scanning reveals multiple exposed services, confirming his suspicions about their poor security practices from their time together at university.

5.2.1 Key Points

- Target identified via company website and social media
- Server IP address discovered through DNS lookup
- Port scanning exposed SSH, HTTP, and MariaDB services
- Weak passwords and shared credentials increase risk of unauthorized access
- Personal knowledge of team members aids password guessing
- Valuable customer data and business disruption motivate attack

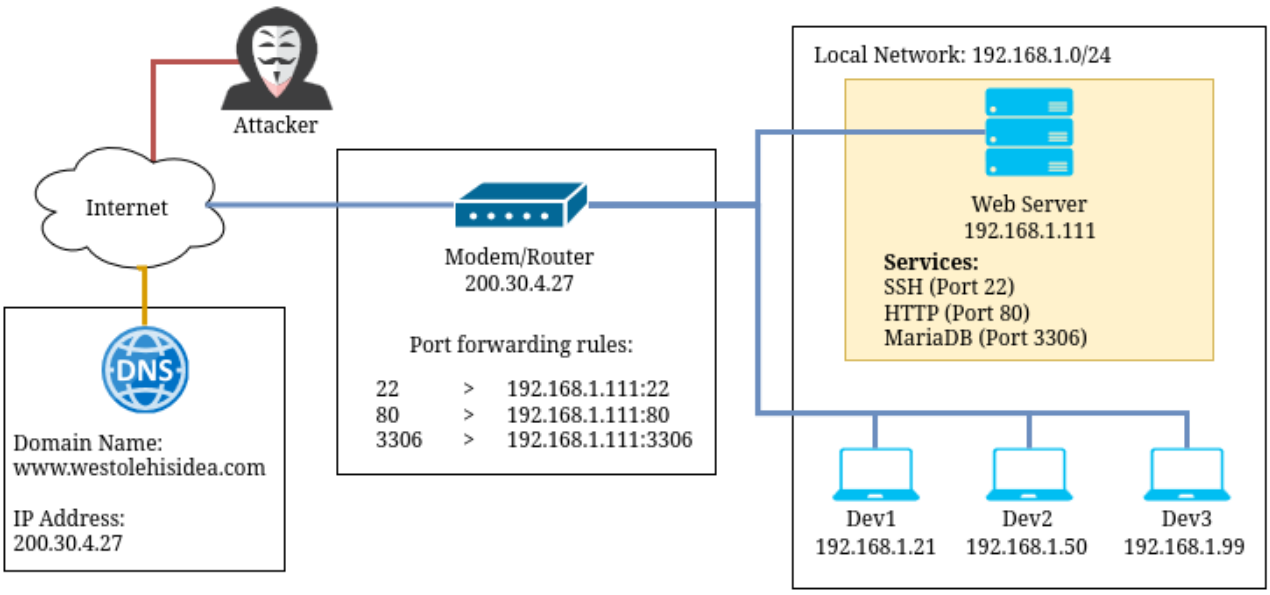

Figure 1: Visual representation of scenario

5.3 Success and Failure Criteria

5.3.1 Attack Success

- **Success:** Discovery of valid username/password combinations and successful authentication to target services
- **Failure:** No valid credentials discovered or all attempts blocked by defensive measures

5.3.2 Defense Success

- **Success:** Prevention of unauthorized access through automated IP blocking within configured thresholds
- **Failure:** Attackers successfully discover credentials and gain system access

5.3.3 Measurement

Success will be evaluated through credential discovery rates, fail2ban logs, and verification of actual system access versus blocked attempts.

6 Enviroment Setup

6.1 Technical Details

6.1.1 Passwords

- A random selection of 500 passwords were sourced.
- Kali Linux's `rockyou.txt` password list was used. It can be accessed via the `wordlists` command.
- Using an existing password list was done to maintain focus on tool usage rather than password list generation, which easily has enough depth to have a whole seprate report written about it.
- In real-world scenarios, attackers rarely have a guaranteed password list, but for testing purposes, a known list ensures tool functionality.
- I used the `shuf` tool to randomly select passwords for the user profiles. I then created the users "bob, jim and harold" with the passwords.
- Database passwords were made similar to SSH passwords to highlight typical credential reuse, making it easier for attackers to guess passwords across services.
- It is assumed that the attacker knows the usernames, which is plausible within they were friends in the context of this exercise.

Service	Username	Password
mariadb	root	neyo24
mariadb	bob	all4mine
mariadb	jim	ilovepico
mariadb	harold	angelica
ssh	root	neyo25
ssh	bob	all5mine
ssh	jim	ilovepicodepuppy
ssh	harold	angelcha

Table 3: Generated Password List

6.1.2 Permissions

- root privileges had to be given to be given to bob's user account to progress past "step 1" of the scenario.
- bob, jim and harold have been given `SELECT, INSERT and UPDATE` permisions.
- root can only be accessed from a root account, so to access the account the root user password will first need to be discovered.

6.1.3 Virtual Network

- In the scenario, the attacker discovered the server's IP address by resolving the domain name, which was publicly shared in a social media post. This could have been virtually implemented but was not required for the scope of the scenario.
- Both machines were placed on the same virtual network to meet the assignment's requirement of a non-internet-facing environment. A diagram that references this can be seen in Figure 2.

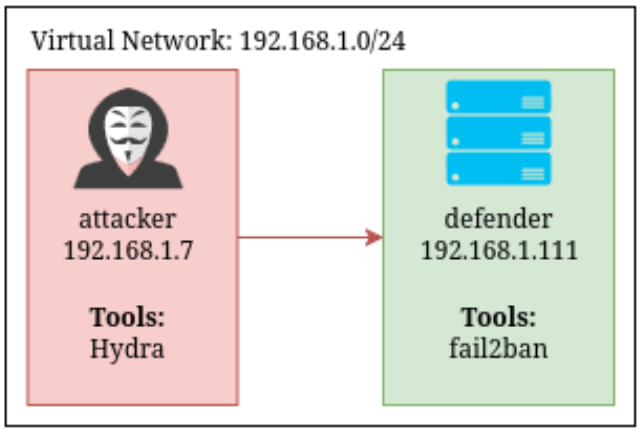

Figure 2: Visual representation of scenario

6.1.4 Attacker Machine

The attacker machine was chosen for its comprehensive suite of pre-installed security tools, which suited the requirements of this assignment. The only additional configuration involved updating all tools to their latest versions and manually assigning an IP address, as there was no DHCP server available within the virtual network. This setup provided a reliable and ready-to-use environment for conducting the required tests.

6.1.5 Defender Machine

I selected Arch Linux for the defender machine because it is the Linux distribution I am most familiar with. Its lightweight nature allowed me to efficiently run the assignment on my laptop, and my experience with its package manager made installing and configuring the necessary tools straightforward. Additionally, using Arch Linux ensured that no defense tools or background services would be operating without my knowledge, giving me full control over the environment. This familiarity ensured a smooth setup process and minimized time spent troubleshooting environment issues.

6.1.6 Proof of Virtual Network

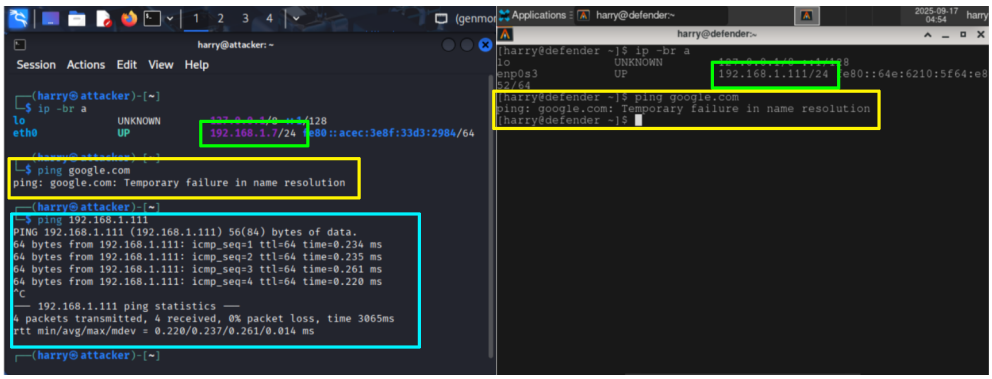

Figure 3: Green: IP Addresses, Yellow: Unable to ping google.com, Blue: Attacker pinging the defender machine

7 Scenario Breakdown

7.1 Step 1: The Initial Attack

7.1.1 Step 1.1

- Began by targeting the MariaDB service (port 3306) using Hydra, with a password list created by sampling "rockyou.txt" and reinserting the captured passwords in random order.

  ```
  hydra -L usernames.txt -P ./mysql_attack_passwords.txt 192.168.1.111 mysql
  ```

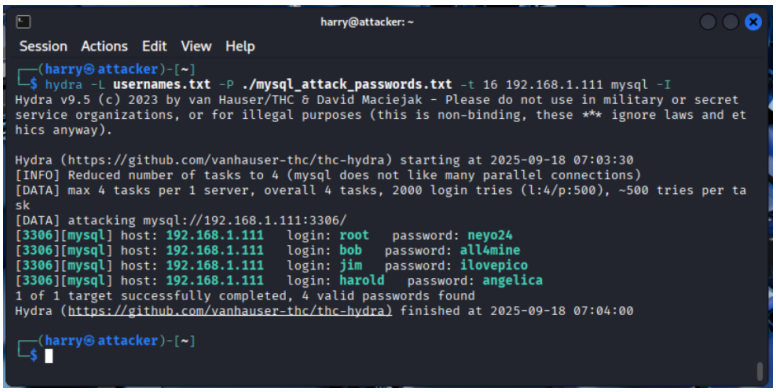

Figure 4: Hydra successfully accessing all 4 mariadb passwords

7.1.2 Step 1.2

- Used a quick and dirty approach by asking an AI chatbot to generate alternative passwords, speeding up the creation of a realistic password list for the attack.

7.1.3 Step 1.3

- Execute Hydra brute force attack against SSH service (port 22) on defender machine (192.168.1.111) using the new password list.

  ```
  hydra -l bob -P ./bob_passwords_50.txt -t 1 -30 192.168.1.111 ssh
  ```

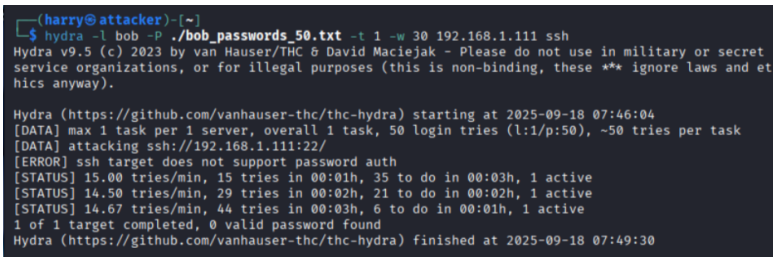

Figure 5: Failed brute force attempt, even with access to the correct password and a small list of 50 passwords.

7.1.4 Step 1.4

- To facilitate testing, the SSH configuration on the defender machine was temporarily modified to allow a higher number of authentication attempts and concurrent sessions:

  ```
  #UsePAM yes                  # Commented out to disable, enabled in Arch by default

  MaxAuthTries 1000            # Allow many attempts per connection
  MaxSessions 100              # Allow multiple concurrent sessions
  MaxStartups 100:30:200       # Increase connection startup limits
  LoginGraceTime 0             # Disable login timeout (or set to 3600)
  ClientAliveInterval 0        # Disable client keepalive checks
  ClientAliveCountMax 0        # Disable automatic disconnections
  ```

7.1.5 Step 1.5

- Once the SSH configuration was altered on the defender machine finding the passwords was a breeze.

7.2 Step 2: The Retaliation

7.2.1 Step 2.1

- SSH logging was enabled by default, but when Mariadb was installed it weirdly was not. To do this I found the following solution[17].

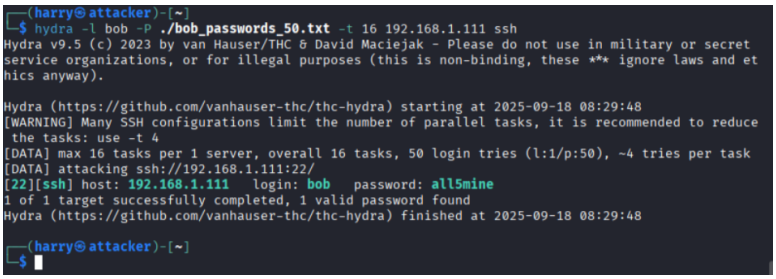

Figure 6: Hydra successfully finding all bob's ssh password

```
sudo nvim /etc/my.cnf.d/logging.cnf          # Created logging config

[mysqld]                                     # Entered this config
log-error = /var/log/mariadb/mariadb.log
log-warnings = 2
general_log = 1
general_log_file = /var/log/mariadb/general.log

sudo mkdir -p /var/log/mariadb               # Created log directory

sudo chown mysql:mysql /var/log/mariadb      # Ensured mysql's permissions

sudo systemctl restart mariadb               # Restart the service
```

7.2.2 Step 2.2

- When setting up the rules for fail2ban I went for a pretty common config of 3 failed attempts within 10 minutes, and ban duration (10 minutes) to automatically block attacking IP addresses via iptables rules.
- Here is how it was done:

```
[DEFAULT]
bantime  = 600 #10 minutes in seconds (10 x 60sec)
findtime = 600
maxretry = 3

[sshd]
enabled = true
port    = ssh
filter  = sshd
logpath = %(sshd_log)s
backend = %(sshd_backend)s

[mysqld-auth]
enabled = true
port    = 3306
filter  = mysqld-auth
logpath = /var/log/mariadb/mariadb.log  #Newly created log path
```

- After the service had been restarted I went straight into testing. I used some of the commands below to construct a script that would reset the fail2ban enviroment.

```
sudo fail2ban-client unban --all          # Clears the jails
sudo fail2ban-client status sshd          # See the status of ssh attempts
sudo fail2ban-client status mysqld-auth   # See the status of mariadb attempts
```

7.2.3 Step 2.3

- After my initial test, I encountered an unexpected outcome as seen in figure 7.
  - For the blue-highlighted content, fail2ban worked as intended—Hydra was unable to discover the password because the attacking IP was promptly blocked.

  - However, for the green-highlighted content, although fail2ban did eventually ban the IP, Hydra managed to find the password before the ban took effect. This suggests that the fail2ban response was not fast enough to prevent successful brute force in this scenario.
- To rectify this result I went back and altered the ssh config to it's original config.
- Refer to Figure 8 for the result after the SSH configuration was restored.

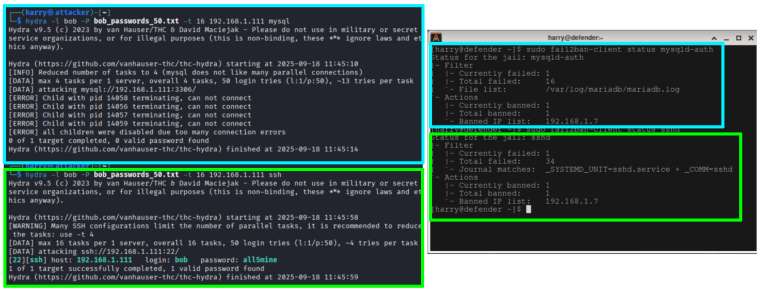

Figure 7: Unexpected result from test

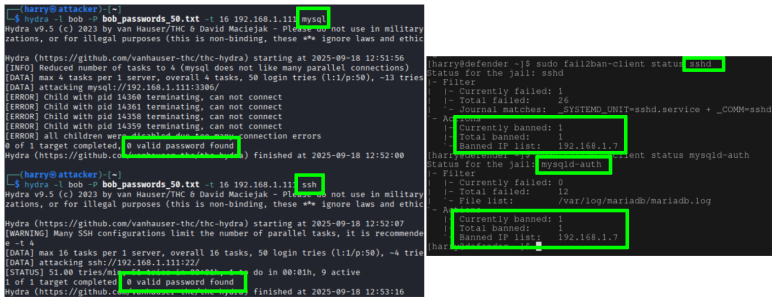

Figure 8: A successful hydra attempt after the original config had been restored.

7.3 Step 3: Spray Attack

This step demonstrates that bypassing fail2ban is possible, though highly impractical. To prove this, I configured fail2ban with a `bantime` and `findtime` of 10 seconds each, while Hydra was throttled to one attempt every 25 seconds, well outside fail2ban's detection window.

- Green: fail2ban banned attempts as expected, but Hydra's slow pace avoided triggering further bans.
- Yellow: Hydra never exceeded one failed attempt per account, staying below fail2ban's threshold.
- Blue: Hydra successfully discovered a password from a list of 10, but the process was inefficient.

This confirms that, under controlled conditions, fail2ban can be bypassed, but the effort required makes it an unlikely attack vector.

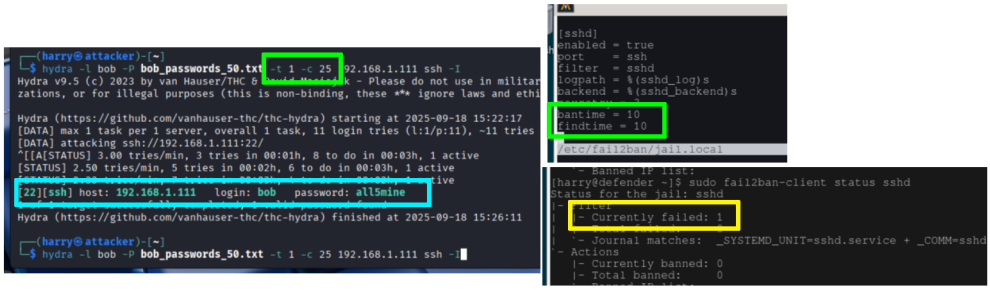

Figure 9: Screen capture of successful brute force in very controlled conditions

Step	Action	Tool/Method	Target	Result
1.1	MariaDB Password Attack	Hydra with mysql password list	MariaDB service (port 3306)	Success: All 4 database passwords discovered
1.2	Password Pattern Analysis	AI-generated similar password variations	Captured MariaDB credentials	50-password lists created per user
1.3	SSH Brute Force Attack	Hydra with bob password list	SSH service (port 22)	Failure: Maximum authentication attempts exceeded
1.4	SSH Configuration Modification	Manual config changes	Defender SSH service	SSH limits increased to allow testing
1.5	SSH Attack Retry	Hydra with modified SSH config	SSH service (port 22)	Success: Access gained with correct credentials

Table 4: Step 1: Initial Attack Phase Summary

Step	Action	Tool/Method	Target	Result
2.1	MariaDB Logging Setup	Manual configuration and service restart	MariaDB logging system	Success: Authentication failure logging enabled
2.2	fail2ban Configuration	Manual jail configuration with 3/10min thresholds	SSH and MariaDB services	Success: Monitoring jails activated
2.3a	Initial Defense Test	fail2ban vs Hydra attacks	SSH and MariaDB services	Mixed: Some passwords found before ban triggered
2.3b	SSH Config Restoration	Restore original SSH security settings	SSH service configuration	Success: Original security posture restored
2.3c	Final Defense Test	fail2ban vs Hydra with proper SSH config	SSH and MariaDB services	Success: All attacks blocked effectively

Table 5: Step 2: Defense Implementation Phase Summary

Step	Action	Tool/Method	Target	Result
3.1	fail2ban Reconfiguration	Reduced detection windows for testing	fail2ban detection parameters	Success: 10-second bantime/findtime configured
3.2	Spray Attack Setup	Hydra with 25-second delays	Password list preparation	Success: Attack timing configured to evade detection
3.3	Password Spray Execution	Hydra with controlled timing vs fail2ban	SSH authentication service	Success: Password discovered while evading bans
3.4	Evasion Analysis	Manual assessment of bypass effectiveness	Attack efficiency and practicality	Mixed: Bypass successful but highly impractical
3.5	Defense Evaluation	Compare practical vs theoretical bypass	fail2ban effectiveness assessment	Success: Defense proven effective against realistic attacks

Table 6: Step 3: Password Spray Attack Phase Summary

8 Results Analysis

References

[1] Australian Cyber Security Centre, "Annual cyber threat report 2023-2024," Australian Signals Directorate, 2024. Available: https://www.cyber.gov.au/about-us/view-all-content/reports-and-statistics/annual-cyber-threat-report-2023-2024

[2] Microsoft Corporation, "Microsoft digital defense report 2024," Microsoft Corporation, Technical Report, 2024. Available: https://www.microsoft.com/en-us/security/security-insider/threat-landscape/microsoft-digital-defense-report-2024

[3] Microsoft Security Response Center, "Microsoft actions following attack by nation state actor midnight blizzard." 2024. Available: https://msrc.microsoft.com/blog/2024/01/microsoft-actions-following-attack-by-nation-state-actor-midnight-blizzard/

[4] D. Maguire, "What's happening with the optus data breach? What we know about the alleged hacker's ransom, data release and apology," *ABC News*, 2022, Available: https://www.abc.net.au/news/2022-09-27/optus-data-breach-cyber-attack-hacker-ransom-sorry/101476316

[5] S. Chalmers, "Qantas confirms 5.7 million customers were impacted in cyber attack," *ABC News*, 2025, Available: https://www.abc.net.au/news/2025-07-09/qantas-confirms-number-of-customers-impacted-in-cyber-attack/105510654

[6] MITRE Corporation, "MITRE ATT&CK technique T1110: Brute force." 2024. Available: https://attack.mitre.org/techniques/T1110/

[7] J. Steube, "Hashcat: Advanced password recovery." 2025. Available: https://hashcat.net/hashcat/

[8] van Hauser, "THC-hydra: Online password cracking tool." 2025. Available: https://github.com/vanhauser-thc/thc-hydra

[9] F. Networks, "Medusa: Parallel network login auditor." 2025. Available: http://foofus.net/?page_id=51

[10] S. Designer, "John the ripper: Password cracker." 2025. Available: https://github.com/openwall/john

[11] KeychainX, "How to recover lost bitcoin passwords." 2021. Available: https://keychainx.medium.com/how-to-recover-lost-bitcoin-passwords-c34c42ee6f17

[12] NetExec Team, "NetExec wiki." 2024. Available: https://www.netexec.wiki/

[13] SpecterOps, "BloodHound legacy." 2024. Available: https://github.com/SpecterOps/BloodHound-Legacy

[14] fail2ban Project, "How fail2ban works." 2024. Available: https://github.com/fail2ban/fail2ban/wiki/How-fail2ban-works

[15] W. Project, "Wazuh: Security detection, visibility, and compliance." 2025. Available: https://github.com/wazuh/wazuh

[16] Netgate, "pfSense: Open source firewall and router." 2025. Available: https://www.pfsense.org/

[17] M. Corporation, "MariaDB error log documentation." [Online]. Available: https://mariadb.com/docs/server/server-management/server-monitoring-logs/error-log