114 年公務人員特種考試國家安全局國家安全情報人員考試試題

考試別：國家安全情報人員考試
等　別：三等考試
類　科：資訊組（選試英文）
科　目：網路應用與安全

曹勝老師解題

一、為何許多舊有協定如 Telnet、FTP 被視為不安全？（10 分）若分別以 SSH 取代 Telnet、以 SFTP 或 FTPS 取代 FTP，請說明其運作方式為何能改進安全問題？（10 分）

【解題關鍵】
1.《考題難易》★★★
2.《破題關鍵》掌握明文傳輸、弱身份驗證、無加密導致不安全以及 SSH/SFTP 用加密通道、強驗證；FTPS 用 SSL/TLS 加密、證書驗證，解決竊聽與篡改問題。

【擬答】

㈠為何 Telnet 和 FTP 被視為不安全？

Telnet 和 FTP 是早期設計的網路協定，當時網路環境相對簡單，安全問題尚未被廣泛重視。這導致它們在設計上存在以下安全缺陷：

1. 明文傳輸：Telnet 和 FTP 傳輸資料（包括使用者名稱、密碼和檔案內容）時使用明文（未加密）。這意味著任何能夠攔截網路流量的第三方（如中間人攻擊者）都可以輕易讀取這些敏感資訊。例如，在 Telnet 中，當用戶輸入登錄憑證時，這些資料會以純文字形式傳輸，攻擊者可通過封包嗅探（packet sniffing）工具（如 Wireshark）直接獲取。
2. 缺乏身份驗證和完整性檢查：這兩個協定不提供強大的身份驗證機制，容易遭受偽造或未授權存取。沒有資料完整性檢查，無法確保傳輸的資料未被篡改。
3. 無加密保護：Telnet 和 FTP 不支援傳輸層加密，無法防止資料在傳輸過程中被竊聽或修改。這在現代網路環境中尤其危險，因為公開 Wi-Fi 或不受信任的網路環境增加了資料被攔截的風險。
4. 易受攻擊：Telnet 和 FTP 容易受到中間人攻擊（MITM）、會話劫持（session hijacking）或密碼破解等攻擊。FTP 的「被動模式」和「主動模式」涉及動態開啟的連接埠，增加了被惡意利用的風險。

㈡ SSH 取代 Telnet 的安全改進：

SSH（Secure Shell）是一種加密的遠端存取協定，專為取代 Telnet 等不安全的協定而設計。以下是 SSH 如何改進 Telnet 的安全問題：

1. 加密傳輸：SSH 使用強大的加密技術（如 AES）對所有傳輸的資料進行加密，包括登錄憑證和會話內容。即使攻擊者攔截了資料，也無法直接讀取，因為資料已被加密為不可讀的格式。
2. 身份驗證機制：SSH 支援多種身份驗證方式，這些方法大幅降低了未授權存取的風險。包括：
   ⑴密碼驗證：用戶輸入密碼，密碼在傳輸時被加密。
   ⑵公鑰/私鑰驗證：使用非對稱加密（如 RSA 或 ECDSA），客戶端持有私鑰，伺服器持有公鑰，確保更高的安全性。

⑶多因子驗證（MFA）：某些 SSH 實作支援結合密碼和硬體金鑰等機制。

3. 資料完整性保護：SSH 使用訊息完整性檢查（MAC，Message Authentication Code）來驗證資料在傳輸過程中未被篡改。這確保即使攻擊者試圖修改傳輸的資料，接收端也能檢測到篡改。

4. 安全通道：SSH 通過建立加密的「通道」（tunnel）來保護通信，防止中間人攻擊和會話劫持。它還支援動態埠轉發（Dynamic Port Forwarding）和本地/遠端埠轉發，允許安全地傳輸其他協定的流量。

SSH 運作方式：

1. SSH 使用客戶端-伺服器模型，基於 TCP（通常使用埠 22）。

2. 連線建立時，客戶端與伺服器進行握手（handshake），協商加密演算法和金鑰。

3. 伺服器會提供其公鑰，客戶端驗證其真實性（通常通過比較已知的伺服器指紋）。

4. 連線建立後，所有資料都通過加密通道傳輸。

㈢ SFTP 和 FTPS 取代 FTP 的安全改進：

SFTP（SSH File Transfer Protocol）和 FTPS（FTP Secure）是 FTP 的安全替代方案，兩者都通過加密技術解決了 FTP 的安全問題，但運作方式不同。

1. SFTP（基於 SSH 的檔案傳輸）：

SFTP 是在 SSH 協議上運行的檔案傳輸協定，完全取代了 FTP 的明文傳輸。SFTP 運行在 SSH 協議之上，通過 SSH 建立安全的加密通道。客戶端發起連線後，與伺服器進行 SSH 握手，協商加密參數。檔案傳輸（上傳、下載、刪除等）全部在加密通道內進行，支援斷點續傳和目錄操作。完全解決了 FTP 明文傳輸和弱身份驗證的問題，並簡化了網路配置，適用於需要高安全性的場景。安全改進：

⑴加密傳輸：SFTP 利用 SSH 的加密通道傳輸檔案和命令，確保檔案內容、使用者憑證等敏感資訊不會以明文形式傳輸。

⑵身份驗證：與 SSH 相同，SFTP 支援密碼和公鑰/私鑰驗證，確保只有授權用戶可以存取檔案。

⑶資料完整性：SFTP 使用 SSH 的 MAC 機制，確保檔案在傳輸過程中未被篡改。

⑷單一連接埠：SFTP 僅使用 SSH 的預設埠（通常為 22），不像 FTP 需要額外的資料通道埠，降低了防火牆配置的複雜性和安全風險。

2. FTPS（FTP over SSL/TLS）：

FTPS 是基於 SSL/TLS 協議的 FTP 擴展，通過加密層保護 FTP 的傳輸。FTPS 在 FTP 協議上添加了 SSL/TLS 層。連線時，客戶端與伺服器進行 SSL/TLS 握手，協商加密演算法和金鑰。控制通道（用於命令）和資料通道（用於檔案傳輸）均可加密，確保安全傳輸。FTPS 解決了 FTP 明文傳輸的問題，並提供與 HTTPS 類似的安全保證，適合需要與傳統 FTP 環境相容的場景。安全改進：

⑴加密傳輸：FTPS 使用 SSL/TLS（通常在埠 990 或 21）對控制通道和資料通道進行加密，防止資料被竊聽。支援顯式（Explicit）模式（客戶端主動請求加密）和隱式（Implicit）模式（連線一開始即加密）。

⑵身份驗證：FTPS 支援基於證書的身份驗證，伺服器和客戶端可交換數位證書以驗證身份，防止偽造。支援用戶名/密碼驗證，這些憑證在傳輸時被加密。

⑶資料完整性：SSL/TLS 提供資料完整性檢查，確保傳輸的檔案未被篡改。

⑷靈活性：FTPS 相容於傳統 FTP 客戶端和伺服器，只需增加 SSL/TLS 支援即可升級。

二、某平台導入 Google Authenticator 作為登入系統的第二層驗證機制。請說明此類基於 TOTP（Time-based One-Time Password）的驗證方式原理？（10 分）另其安全性相較傳統密碼的優勢為何？（10 分）

【解題關鍵】

1.《考題難易》★★★

2.《破題關鍵》TOTP 基於共享密鑰與時間生成一次性密碼，30 秒失效。相較傳統密碼，TOTP 提供雙因素驗證、動態密碼，防竊聽與重放攻擊。

【擬答】

㈠ **TOTP（Time-based One-Time Password, 基於時間的一次性密碼）驗證原理：**

是一種基於時間同步的雙因素驗證（2FA）機制，廣泛用於像 Google Authenticator 這樣的應用。以下是其運作原理的詳細說明：

1. 共享密鑰（Shared Secret）：在啟用 TOTP 時，伺服器會生成一個唯一的秘密金鑰（通常以 QR 碼形式提供），並與用戶的設備（如 Google Authenticator 應用）共享。這個秘密金鑰儲存在伺服器和用戶設備上，作為生成一次性密碼的基礎。
2. 時間同步：TOTP 使用當前時間作為輸入參數。通常以 Unix 時間戳（從 1970 年 1 月 1 日起計算的秒數）為基礎，將時間分成固定長度的時間窗口（通常為 30 秒）。例如，假設當前時間戳為 1634567890 秒，TOTP 會將其除以 30，得到一個時間計數器（Counter = floor（1634567890 / 30））。
3. 密碼生成：TOTP 使用 HMAC-SHA（通常是 HMAC-SHA1）演算法，將共享密鑰和時間計數器作為輸入，生成一個雜湊值。該雜湊值經過截斷（truncation）和模運算（通常模 10^6 或 10^8），生成一個 6 或 8 位數的數字密碼（即一次性密碼，OTP）。公式簡化為：OTP = HMAC-SHA1（Shared Secret, Time Counter）mod 10^d，其中 d 是密碼位數。
4. 驗證過程：用戶在登入時輸入帳號密碼（第一因素）後，開啟 Google Authenticator 應用，取得當前 30 秒窗口內的 OTP。用戶將 OTP 提交給伺服器，伺服器使用相同的共享密鑰和當前時間計算 OTP，並與用戶提供的 OTP 比對。為容忍輕微的時間偏差，伺服器通常會檢查當前窗口以及前後一個窗口的 OTP（即允許 ±30 秒的時間誤差）。
5. 密碼有效期：每個 OTP 在其 30 秒時間窗口內有效，窗口結束後會生成新的 OTP，確保密碼的「一次性」特性。
6. 技術細節：TOTP 遵循 RFC 6238 標準，密碼生成不依賴網路，僅需設備與伺服器的時間同步（通常通過 NTP 協議或設備本地時鐘）。Google Authenticator 等應用會在設備上安全儲存共享密鑰，通常加密保護以防止未授權存取。

㈡ **TOTP 相較傳統密碼的安全性優勢：**

傳統密碼（僅依賴用戶名和密碼的驗證）在現代網路環境中存在多種安全風險，而 TOTP 作為雙因素驗證的一部分，顯著提升了安全性。以下是 TOTP 相較傳統密碼的優勢：

1. 雙因素驗證（2FA）：傳統密碼僅依賴「知識因素」（something you know），即密碼。若密碼洩露（如通過釣魚、鍵盤記錄器或資料庫洩露），帳戶立即面臨風險。TOTP 要求「知識因素」（密碼）和「擁有因素」（something you have），即生成 OTP 的設備（如手機）。即使密碼洩露，攻擊者仍需物理存取用戶的設備或其共享密鑰才能登入。
2. 動態密碼：傳統密碼是靜態密碼長期不變，易被猜測、重用或通過暴力破解攻擊。TOTP 每 30 秒生成一個新 OTP，密碼短暫有效，無法重用。即使攻擊者攔截了一個 OTP，該密碼在下一個時間窗口即失效，降低了重放攻擊（replay attack）的風險。

3. 抗中間人攻擊（MITM）：傳統密碼在不安全的網路（如公開 Wi-Fi）中，密碼可能被攔截，尤其在使用未加密的 HTTP 連線時。TOTP 的 OTP 由用戶設備本地生成，不通過網路傳輸生成過程。即使攻擊者攔截了用戶輸入的 OTP，其有效期極短，且無法推導出共享密鑰或下一個 OTP。
4. 抗釣魚攻擊：傳統密碼的用戶可能在釣魚網站輸入密碼，導致洩露。釣魚網站難以模擬 TOTP 驗證，因為 OTP 由用戶設備生成，且釣魚網站無法獲取共享密鑰。即使攻擊者騙取了 OTP，由於其短暫有效性，攻擊者需在 30 秒內使用，增加了攻擊難度。
5. 無需網路連線：傳統密碼需要傳輸到伺服器驗證，存在被攔截的風險。TOTP 的 OTP 在用戶設備本地生成，無需依賴網路傳輸生成過程，降低了中間環節的風險。
6. 抗密碼重用：傳統密碼的用戶常在多個平台使用相同密碼，若一個平台洩露，所有使用該密碼的帳戶都可能受影響。TOTP 在每個平台的共享密鑰獨立，OTP 僅適用於特定帳戶。即使一個平台的密鑰洩露，其他平台不受影響。

三、某機關部署傳統防火牆僅開放必要的 443（HTTPS）與 80（HTTP）port，且限定 IP 白名單，但仍遭遇資料外洩事件。經分析發現，攻擊者透過 Web 表單注入惡意指令成功竊取機敏資料。請問：（每小題 10 分，共 20 分）

㈠為何傳統以 port 和 IP 為基礎的防火牆無法阻擋這類攻擊？

㈡請說明應用層防火牆（WAF）在此案例中可提供的保護機制與優勢？

【解題關鍵】

1.《考題難易》★★★

2.《破題關鍵》傳統防火牆不檢查應用層內容，無法偵測 Web 表單注入。 WAF 提供深度封包檢查、規則比對、防 SQL 注入與 XSS，保護應用層安全。

【擬答】

㈠為何傳統以 port 和 IP 為基礎的防火牆無法阻擋這類攻擊？

傳統防火牆（也稱為網路層防火牆或包過濾防火牆）主要基於網路層（Layer 3）和傳輸層（Layer 4）的資訊進行過濾，例如 IP 地址、埠號和協議類型。在本案例中，機關部署的傳統防火牆僅開放 80（HTTP）和 443（HTTPS）埠，並限定 IP 白名單，但仍無法阻止透過 Web 表單注入惡意指令的攻擊，原因如下：

1. 僅檢查網路層和傳輸層資訊：傳統防火牆專注於檢查封包的來源/目的 IP 地址、埠號和協議（如 TCP/UDP），不分析應用層（Layer 7）的內容。Web 表單注入攻擊（如 SQL 注入、跨站腳本攻擊 XSS 或命令注入）發生在應用層，隱藏在正常的 HTTP/HTTPS 流量（80 或 443 埠）中。這些流量從網路層看來是合法的，因此傳統防火牆無法辨識其中的惡意內容。
2. 無法識別惡意應用層內容：攻擊者通過合法的 IP 白名單地址發送惡意指令，這些指令嵌入在 HTTP/HTTPS 請求的表單數據（如 POST 請求的參數）中。傳統防火牆缺乏對 HTTP 請求內容（如 URL、表單數據、頭部資訊）的解析能力，無法檢測惡意指令或異常行為。
3. HTTPS 加密流量限制：HTTPS（443 埠）流量使用 TLS/SSL 加密，傳統防火牆無法解密並檢查封包內容。即使攻擊者通過 HTTPS 傳送惡意指令，防火牆也無法識別，因為它只看到加密後的數據流。
4. IP 白名單的局限性：IP 白名單僅限制來源 IP，但攻擊者可能通過合法 IP（如受感染的內部設備、被盜用的白名單 IP 或代理伺服器）發送惡意請求。此外，若攻擊者利用

已認證用戶的會話（如透過釣魚或會話劫持），即使 IP 在白名單內，攻擊仍可繞過防火牆。

5. 缺乏上下文和行為分析：傳統防火牆不具備應用層上下文分析能力，無法判斷請求是否符合正常的應用行為（如表單輸入是否包含惡意 SQL 語句或腳本）。Web 表單注入攻擊利用應用程式漏洞，這些漏洞通常與後端程式碼處理不當有關，傳統防火牆無法檢測或修補這類漏洞。

㈡應用層防火牆（Web Application Firewall, WAF）在此案例中的保護機制與優勢：

WAF 專為保護應用層（Layer 7）設計，特別針對 Web 應用程式（如 HTTP/HTTPS 流量）提供安全防護。WAF 能夠分析和過濾 Web 請求的內容，對於防止 Web 表單注入攻擊（如 SQL 注入、XSS、命令注入等）具有顯著優勢。以下是 WAF 在此案例中提供的保護機制：

1. 深度封包檢查（Deep Packet Inspection）：WAF 能夠解析 HTTP/HTTPS 請求的內容，包括 URL、表單數據、HTTP 頭部和主體，檢查是否存在惡意模式或異常行為。例如，WAF 可以檢測表單輸入中是否包含 SQL 注入語句（如 SELECT * FROM users WHERE 1=1）或 XSS 腳本（如 <script>alert('hack')</script>），並阻止這些請求。
2. 規則和簽名比對：WAF 使用預定義的規則集（如 OWASP Top 10 攻擊模式）來識別常見的 Web 攻擊，例如 SQL 注入、XSS、命令注入等。這些規則會檢查請求中的關鍵字、語法或異常模式，並在檢測到可疑行為時阻斷請求。
3. HTTPS 流量解密與檢查：WAF 支援 TLS/SSL 終止（termination），可以在伺服器端解密 HTTPS 流量，檢查其內容後再重新加密傳送到後端伺服器。這使得 WAF 能夠檢測隱藏在加密流量中的惡意指令，解決傳統防火牆無法檢查 HTTPS 內容的問題。
4. 行為分析與異常檢測：WAF 能夠根據應用程式的正常行為建立基線，檢測異常的請求模式。例如，若某表單通常只接受數字輸入，但收到包含 SQL 語句的輸入，WAF 可標記為異常並阻斷。進階 WAF 還可能使用機器學習來識別未知攻擊模式（零日攻擊）。
5. 防護應用層漏洞：Web 表單注入攻擊通常利用應用程式漏洞（如未正確過濾用戶輸入）。WAF 通過過濾惡意輸入，為應用程式提供額外的保護層，即使後端程式碼存在漏洞也能降低風險。
6. 虛擬修補（Virtual Patching）：WAF 可以快速部署規則來阻止特定攻擊，而無需修改應用程式程式碼。這對於緊急應對已知漏洞（如等待開發者修補程式碼期間）特別有用。
7. 日誌與監控：WAF 記錄所有 Web 請求的詳細資訊（包括來源 IP、請求內容、觸發的規則等），幫助安全團隊分析攻擊事件並追蹤攻擊者。

㈢應用層防火牆（WAF）在此案例中的優勢：

1. 針對性防護：相較於傳統防火牆，WAF 專注於應用層安全，特別針對 Web 應用程式攻擊（如表單注入）提供精細的保護，傳統防火牆無法做到這一點。
2. 加密流量檢查：WAF 能夠解密並檢查 HTTPS 流量，解決傳統防火牆對加密流量的盲點問題。
3. 快速響應：WAF 的規則更新速度快，能迅速應對新發現的攻擊模式，而傳統防火牆需要手動調整 IP 或埠規則，靈活性較低。
4. 應用層漏洞保護：WAF 通過過濾惡意輸入，直接應對應用層漏洞（如 SQL 注入、XSS），為應用程式提供即時保護，而傳統防火牆無法干預應用層內容。
5. 靈活的部署方式：WAF 可以部署為反向代理、雲端服務或嵌入式模組，適應不同規模的機關需求，傳統防火牆的硬體部署方式相對固定。

6. 降低誤報風險：WAF 的上下文分析能力能更精確地區分正常和惡意流量，減少誤報阻斷合法用戶的風險，而傳統防火牆的簡單規則可能導致過度阻斷。

四、某公司員工外出開會時，不慎將使用公司帳號登入的手機遺失。手機未設定密碼鎖，也未啟用遠端清除功能。請問：（每小題 10 分，共 20 分）

㈠此事件可能導致那些行動裝置相關的資安風險？

㈡若企業部署行動裝置管理機制（Mobile Device Management, MDM），可以採取那些對應手段來降低風險？

【解題關鍵】

1.《考題難易》★★★

2.《破題關鍵》遺失手機可能導致未授權存取、資料外洩、惡意軟體植入。 MDM 提供遠端擦除、鎖定、強制密碼、應用管理，降低資料洩露風險。

【擬答】

㈠某公司員工遺失手機可能導致的行動裝置相關資安風險：

當公司員工遺失未設密碼鎖且未啟用遠端清除功能的手機，且該手機已使用公司帳號登入，可能引發以下行動裝置相關的資安風險：

1. 未授權存取公司帳號與資料：

手機未設密碼鎖，任何拾獲手機的人都可以直接解鎖並存取公司帳號（如電子郵件、企業應用程式或雲端服務）。

攻擊者可能存取敏感資料（如公司郵件、客戶資料、內部文件）或利用帳號進行進一步攻擊，例如發送釣魚郵件或竊取機密資訊。

2. 應用程式漏洞利用：

若手機上安裝的應用程式（如公司內部 App 或第三方 App）存在漏洞，攻擊者可能利用這些漏洞提權（privilege escalation）或竊取儲存在應用程式中的資料（如認證令牌、緩存數據）。

已登入的公司帳號可能儲存了未過期的會話令牌（session token），允許攻擊者直接存取企業系統。

3. 網路釣魚與社交工程：

攻擊者可利用手機上的通訊錄、郵件或訊息應用程式，冒充員工發送釣魚訊息，誘騙其他員工或客戶提供敏感資訊。

例如，攻擊者可能假冒員工向 IT 部門要求重設密碼或存取其他系統。

4. 惡意軟體植入風險：

攻擊者可能在手機上安裝惡意軟體（如鍵盤記錄器或間諜軟體），進一步監控員工的活動或竊取其他帳號的憑證。

若手機連接到公司網路，惡意軟體可能成為內網攻擊的跳板。

5. 資料外洩：

手機可能儲存敏感的離線資料（如下載的附件、備忘錄或螢幕截圖），這些資料可能被未授權人員直接存取。

若手機連接到雲端服務（如 OneDrive、Google Drive），攻擊者可能通過已登入的帳號下載大量公司資料。

6. 身份冒充與財務損失：

攻擊者可能利用手機上的公司帳號進行未授權操作，例如轉移資金、修改資料或執行詐

騙行為。

若手機存有公司相關的支付應用或銀行帳戶資訊，風險更大。

7. 缺乏遠端清除功能的後果：

由於未啟用遠端清除功能，公司無法遠端擦除手機資料，無法即時阻止資料外洩或未授權存取。

這使得攻擊者有更多時間分析手機內容並利用其中的資訊。

㈡企業部署行動裝置管理機制（MDM）可採取的對應手段：

行動裝置管理（Mobile Device Management, MDM）是一種企業級解決方案，旨在管理和保護員工使用的行動裝置。針對上述資安風險，MDM 可採取以下對應手段來降低風險：

1. 強制設備安全策略：

MDM 可強制要求所有註冊的行動裝置設定密碼鎖、PIN 碼或生物辨識（如指紋、臉部辨識）。可設定密碼複雜度要求（如最少 8 位，包含字母、數字和符號）並定期強制更新。限制多次錯誤嘗試後自動鎖定或擦除設備。即使手機遺失，未授權人員也無法輕易解鎖，降低未授權存取公司帳號和資料的風險。

2. 遠端擦除與鎖定：

MDM 允許 IT 管理員在設備遺失或被盜時遠端鎖定或擦除設備上的所有資料（包括公司帳號和應用程式數據）。可選擇性擦除僅與公司相關的資料（例如企業應用程式和郵件），保留個人資料。即使手機未設密碼鎖，遠端擦除功能可快速清除敏感資料，防止資料外洩或未授權存取。

3. 應用程式管理與限制：

MDM 可限制哪些應用程式可以安裝在設備上，防止未經授權或不安全的第三方應用程式。將公司應用程式與個人應用程式隔離（如使用容器化技術），確保公司資料儲存在安全的沙箱環境中。禁止在未受信任的 Wi-Fi 網路上存取企業應用程式。防止攻擊者利用應用程式漏洞或惡意軟體竊取資料，同時限制公司資料的存取範圍。

4. 強制加密與資料保護：

MDM 可要求設備啟用儲存加密，確保手機上的資料（包括緩存和下載的文件）以加密形式儲存。強制使用安全的 VPN 連線，保護公司應用程式與企業伺服器之間的通訊。即使手機被物理存取，加密資料也難以被破解，降低資料外洩風險。

5. 遠端登出與帳號管理：

MDM 可遠端登出公司帳號，終止所有活躍的會話（例如電子郵件或企業應用程式的會話令牌）。配合單一簽入（SSO）或身份管理系統，快速停用遺失設備上的帳號存取權限。防止攻擊者利用已登入的帳號進行未授權操作或資料存取。

6. 設備合規性檢查：

MDM 可定期檢查設備是否符合企業安全策略（如作業系統版本是否更新、是否越獄/Root）。若設備不符合要求（如未設密碼鎖或安裝了不安全應用程式），MDM 可限制其存取企業資源。確保所有設備保持安全狀態，減少因設備配置不當導致的漏洞。

7. 日誌與監控：

MDM 提供設備活動的詳細日誌，包括存取的應用程式、連線的網路和異常行為。可設定即時警報，當設備遺失或檢測到可疑活動時通知 IT 管理員。幫助企業快速發現並回應潛在的資安事件，縮短攻擊者的操作時間。

8. 備份與恢復：

MDM 可強制定期備份公司資料到安全的企業伺服器，並在設備遺失後協助員工在新設

備上快速恢復工作環境。確保備份資料加密並限制存取權限。即使設備遺失，企業資料可安全恢復，減少業務中斷和資料損失。

五、某開發人員使用 AI 程式碼產生工具（如 GitHub Copilot）來加速專案開發，未經詳細審查便直接部署至內網服務，後續被發現程式引入一個過時版本的加密函式庫與硬編碼的帳號密碼。請問：（每小題 10 分，共 20 分）

㈠使用 AI 輔助寫程式可能引入那些資訊安全風險？

㈡可採取那些方法來減少上述風險？

【解題關鍵】

1.《考題難易》★★★

2.《破題關鍵》AI 可能引入過時函式庫、硬編碼憑證、不安全程式碼模式。採取程式碼審查、SAST 工具、依賴項檢查、安全憑證管理降低風險。

【擬答】

㈠使用 AI 程式碼產生工具可能引入的資訊安全風險：

AI 程式碼產生工具（如 GitHub Copilot）能夠根據提示生成程式碼片段，顯著提高開發效率。然而，若未經詳細審查便直接部署至內網服務，可能引入以下資訊安全風險，特別是在本案例中發現的過時加密函式庫和硬編碼帳號密碼問題：

1. 引入過時或不安全的函式庫：

AI 工具可能基於其訓練數據（通常來自公開的程式碼倉庫）生成依賴過時函式庫的程式碼，例如本案例中的過時加密函式庫。這些函式庫可能包含已知的漏洞（如弱加密演算法或未修補的安全問題），容易被攻擊者利用。過時加密函式庫可能使用不安全的演算法（如 MD5 或 DES），導致資料加密保護不足，增加資料洩露風險。

2. 硬編碼敏感資訊：

AI 工具可能生成包含硬編碼（hardcoded）敏感資訊（如帳號、密碼、API 金鑰）的程式碼，這是因為訓練數據中常見此類不良實作。硬編碼的憑證容易被攻擊者從程式碼中提取，尤其在程式碼未經混淆或儲存在公開倉庫時。硬編碼的帳號密碼可能被內網中的惡意行為者或外部攻擊者（若程式碼洩露）直接存取，導致未授權存取系統。

3. 生成不安全的程式碼模式：

AI 工具可能生成不符合安全編碼最佳實務的程式碼，例如未正確處理用戶輸入（導致 SQL 注入或跨站腳本攻擊 XSS）、缺乏錯誤處理或未實施安全的資料驗證。若程式碼未經審查，這些不安全模式可能引入漏洞，讓攻擊者得以利用應用程式缺陷。

4. 缺乏上下文理解：

AI 工具對專案的具體安全需求（如合規性要求或內網環境限制）缺乏深入理解，可能生成不適用或不安全的程式碼。例如，AI 可能忽略企業要求的特定加密標準（如 FIPS 140-2）。生成的程式碼可能不符合內網服務的安全政策，增加合規風險。

5. 依賴不透明的訓練數據：

AI 工具的訓練數據來自公開程式碼倉庫，可能包含惡意程式碼、錯誤實作或過時技術。開發者無法完全了解生成程式碼的來源，增加了引入潛在風險的可能性。過時加密函式庫可能源自訓練數據中的舊程式碼，AI 工具未考慮其安全性。

6. 程式碼品質與可維護性問題：

AI 生成的程式碼可能缺乏清晰的結構或註釋，增加未來維護的難度。若安全問題未在初始部署時發現，可能在後續更新中被忽視。硬編碼密碼或過時函式庫可能在未經充分

測試的情況下長期存在，延遲漏洞修補。

7. 誤導開發者信任：

開發者可能過分依賴 AI 工具的輸出，認為其生成程式碼安全可靠，從而忽略必要的審查流程。這種「自動化偏見」可能導致漏洞直接進入生產環境。開發者未審查程式碼，直接部署到內網，導致安全問題未被及時發現。

㈡可採取的方法來減少上述風險：

為降低使用 AI 程式碼產生工具引入的資訊安全風險，企業和開發者可以採取以下方法，結合本案例中的問題進行針對性改進：

1. 實施嚴格的程式碼審查流程：

要求所有 AI 生成的程式碼必須經過人工審查（peer review）或自動化靜態程式碼分析工具（如 SonarQube、Checkmarx）檢查。審查重點包括檢查硬編碼憑證、過時函式庫和不安全的程式碼模式。建立程式碼審查清單，明確要求檢查加密函式庫版本、敏感資訊儲存方式和輸入驗證。審查流程可識別硬編碼的帳號密碼和過時加密函式庫，防止其進入內網服務。

2. 使用靜態應用程式安全測試（SAST）：

部署 SAST 工具（如 Fortify、CodeQL）在開發階段掃描程式碼，檢測硬編碼憑證、過時函式庫和常見漏洞（如弱加密演算法）。將 SAST 整合到 CI/CD 管道，確保程式碼在提交或部署前自動檢查。SAST 工具可自動標記硬編碼密碼和過時加密函式庫，提醒開發者修復問題。

3. 管理依賴項與函式庫：

使用依賴項檢查工具（如 Dependabot、Snyk、OWASP Dependency-Check）監控程式碼中使用的函式庫，確保其為最新版本且無已知漏洞。建立企業內部批准的函式庫清單，限制使用未經審核的第三方函式庫。定期更新專案依賴項，修補已知的安全漏洞。依賴項檢查工具可檢測過時加密函式庫，建議升級到安全版本（如從 DES 升級到 AES）。

4. 安全儲存敏感資訊：

禁止硬編碼憑證，使用安全的憑證管理解決方案（如 AWS Secrets Manager、HashiCorp Vault 或環境變數）儲存帳號密碼和 API 金鑰。教育開發者遵循安全編碼實務，避免將敏感資訊直接寫入程式碼。將硬編碼的帳號密碼替換為安全的憑證管理機制，降低憑證洩露風險。

5. 提供安全編碼培訓：

為開發者提供定期安全編碼培訓，涵蓋 AI 工具的使用風險、常見漏洞（如 OWASP Top 10）以及安全最佳實務。強調 AI 生成程式碼的局限性，鼓勵開發者保持批判性思維。培訓可提高開發者對硬編碼和過時函式庫的警覺性，減少依賴 AI 工具的不當使用。

6. 實施 DevSecOps 實務：

將安全性整合到開發流程中（DevSecOps），在 CI/CD 管道中加入安全檢查，如 SAST、DAST（動態應用程式安全測試）和 SCA（軟體成分分析）。設定自動化閘道（gates），確保只有通過安全檢查的程式碼才能部署到內網。DevSecOps 流程可在部署前攔截包含過時函式庫或硬編碼憑證的程式碼，防止其進入內網服務。

7. 限制 AI 工具的訓練數據範圍：

若企業使用自訂的 AI 程式碼生成工具，可限制其訓練數據僅來自內部安全審核過的程式碼倉庫，減少引入不安全實作的可能性。選擇支援企業級安全功能的 AI 工具（如 GitHub Copilot Enterprise），允許自訂生成規則以符合企業安全標準。限制訓練數據可

減少生成過時加密函式庫或硬編碼憑證的程式碼。

8. 記錄與監控程式碼來源：

要求開發者在提交程式碼時標記 AI 生成的部分，方便後續審查和追蹤。使用版本控制系統（如 Git）記錄程式碼變更歷史，確保可追溯問題來源。標記 AI 生成程式碼有助於審查者聚焦檢查高風險區域，快速發現硬編碼或過時函式庫問題。

9. 執行滲透測試與漏洞掃描：

定期對內網服務進行滲透測試和漏洞掃描，模擬攻擊者利用硬編碼憑證或過時函式庫的場景。使用紅隊演練（Red Team Exercises）識別潛在的安全弱點。滲透測試可發現部署後的漏洞，確保即使初始審查遺漏問題，也能在後續檢測中修補。