情報処理安全確保支援士への道(10)：令和７年 春 午後問題 問3を解いてみる(設問1(2)~(4)編)

ルチルMike
2025年8月10日 23:54

問題冊子・配点割合・解答例・採点講評（2025年度、令和7年度） | 試験情報 | IPA 独立行政法人 情...

情報処理推進機構（IPA）の「問題冊子・配点割合・解答例・採点講評（2025年度、令和7年度）」に関する情報です。

www.ipa.go.jp

目次

設問１（３）

ステップ1：攻撃者の手持ちの情報を確認する

ステップ2：ダウンロード対象のファイル名を特定する方法を探す

ステップ3：ファイルをダウンロードする具体的な手順を組み立てる

すべて表示

設問１（２）


(2) 表 1 中の下線②について，アクセスキーを取得する方法を，具体的に答えよ。

この問題は、脆弱性2「Cサービスのアクセスキーの保護に不備がある」状況で、攻撃者がどのようにして「平文のアクセスキー」をFアプリから取得するのか、その具体的な方法を答える問題です。

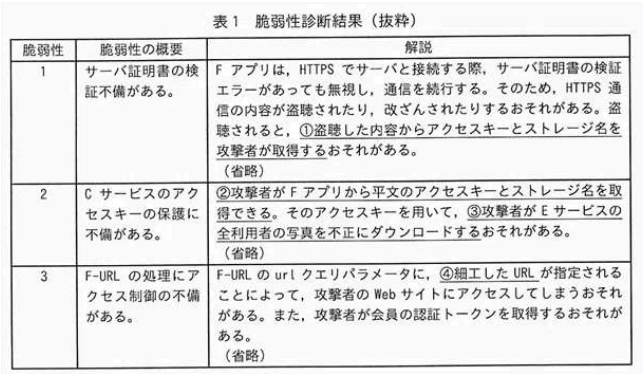
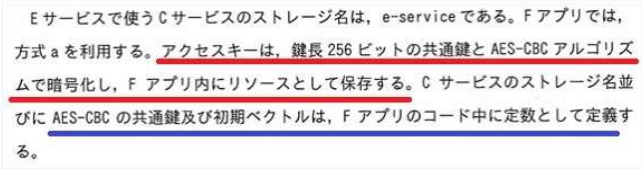


表 1 脆弱性診断結果（抜粋）

脆弱性	脆弱性の概要	解説
1	サーバ証明書の検証不備がある。	F アプリは，HTTPS でサーバと接続する際，サーバ証明書の検証エラーがあっても無視し，通信を続行する。そのため，HTTPS 通信の内容が盗聴されたり，改ざんされたりするおそれがある。盗聴されると，①盗聴した内容からアクセスキーとストレージ名を攻撃者が取得するおそれがある。 （省略）
2	C サービスのアクセスキーの保護に不備がある。	②攻撃者が F アプリから平文のアクセスキーとストレージ名を取得できる。そのアクセスキーを用いて，③攻撃者が E サービスの全利用者の写真を不正にダウンロードするおそれがある。 （省略）
3	F-URL の処理にアクセス制御の不備がある。	F-URL の url クエリパラメータに，④細工した URL が指定されることによって，攻撃者の Web サイトにアクセスしてしまうおそれがある。また，攻撃者が会員の認証トークンを取得するおそれがある。 （省略）

ステップ1：脆弱性の核心を理解する

まず、アクセスキーがFアプリ内でどのように保護されているのか、その設計上の不備を本文から正確に把握します。


E サービスで使う C サービスのストレージ名は，e-service である。F アプリでは，方式 a を利用する。アクセスキーは，鍵長 256 ビットの共通鍵と AES-CBC アルゴリズムで暗号化し，F アプリ内にリソースとして保存する。C サービスのストレージ名並びに AES-CBC の共通鍵及び初期ベクトルは，F アプリのコード中に定数として定義する。

- **保管方法:** アクセスキー自体はAES-CBC方式で暗号化され、アプリ内のリソースとして保存されています 。
- **最大の問題点:** その暗号化を解くための「**共通鍵**」と「**初期ベクトル**」が、**Fアプリのコード中に定数として直接書き込まれて（ハードコードされて）います** 。

これは、金庫（暗号化されたアクセスキー）は用意したものの、その金庫を開けるための鍵（共通鍵）を金庫のすぐ横に貼り付けているような状態です。

ステップ2：可能な攻撃手法を特定する

この「アプリ内に鍵と金庫が揃っている」という状況を悪用できる攻撃手法を考えます。

- **攻撃手法: リバースエンジニアリング (Reverse Engineering)**
- **攻撃の概要:** アプリのプログラムファイル（apkファイルなど）を、専用のツールを使って解析し、ソースコードや内部のロジック、保存されているデータなどを抜き出す攻撃です。

ステップ3：具体的な攻撃手順を組み立てる

リバースエンジニアリングによってアクセスキーを盗むまでの具体的な手順を組み立てます。

1. **アプリの解析（逆コンパイル）:** 攻撃者は、Fアプリのプログラムファイルを**PC**上で逆コンパイルツールにかけ、人間が読める形式のソースコードに戻します。
2. **復号鍵の特定:** 攻撃者は、解析したソースコードの中から、暗号化に使われている**AES**の「**共通鍵**」と「**初期ベクトル**」を見つけ出します。これらは定数としてコード内に書かれているため、比較的容易に発見できます。
3. **暗号化データの特定:** 同時に、アプリ内のリソースファイルとして保存されている「**暗号化されたアクセスキー**」のデータも特定します。
4. **復号の実行:** 攻撃者は、ステップ2で手に入れた「共通鍵」「初期ベクトル」と、ステップ3で手に入れた「暗号化されたアクセスキー」を使って、自身の**PC**上で復号処理を実行します。これにより、元の平文のアクセスキーを完全に取得できます。

ステップ4：解答をまとめる

以上の手順を、解答として具体的にまとめます。

解答例: Fアプリをリバースエンジニアリングして、コード中にハードコードされた復号用の共通鍵と初期ベクトルを特定する。それらを使い、アプリ内にリソースとして保存されている暗号化されたアクセスキーを復号する。

IPA解答例：設問１（２）

(2)	F アプリを解析し，暗号化されたアクセスキー並びに共通鍵及び初期ベクトルを入手し，暗号化されたアクセスキーを AES-CBC で復号する。

設問１（３）

この問題は、脆弱性2の状況で、攻撃者がどのようにして「Eサービスの全利用者の写真を不正にダウンロードする」のか、その具体的な方法を答える問題です。

(3)　表１中の下線③について，ダウンロードする方法を，具体的に答えよ。

表１　脆弱性診断結果（抜粋）

脆弱性	脆弱性の概要	解説
1	サーバ証明書の検証不備がある。	F アプリは，HTTPS でサーバと接続する際，サーバ証明書の検証エラーがあっても無視し，通信を続行する。そのため，HTTPS 通信の内容が盗聴されたり，改ざんされたりするおそれがある。盗聴されると，①盗聴した内容からアクセスキーとストレージ名を攻撃者が取得するおそれがある。 （省略）
2	C サービスのアクセスキーの保護に不備がある。	②攻撃者が F アプリから平文のアクセスキーとストレージ名を取得できる。そのアクセスキーを用いて，③攻撃者が E サービスの全利用者の写真を不正にダウンロードするおそれがある。 （省略）
3	F-URL の処理にアクセス制御の不備がある。	F-URL の url クエリパラメータに，④細工した URL が指定されることによって，攻撃者の Web サイトにアクセスしてしまうおそれがある。また，攻撃者が会員の認証トークンを取得するおそれがある。 （省略）

ステップ1：攻撃者の手持ちの情報を確認する

まず、この攻撃を開始する時点での攻撃者の状況を確認します。前の設問(2)の解答から、攻撃者はすでに以下の2つの重要な情報を手に入れています。

- **Cサービスのアクセスキー**（平文）
- **Cサービスのストレージ名**（e-service）

これらが攻撃の元手となります。

ステップ2：ダウンロード対象のファイル名を特定する方法を探す

次に、攻撃者が全利用者の写真をダウンロードするためには、個々のファイル名を知る必要があります。その命名規則に関する情報を本文から探します。

- **ヒント:** 図4「フォトグッズの注文処理の流れ」とその注記に重要な記述があります 。

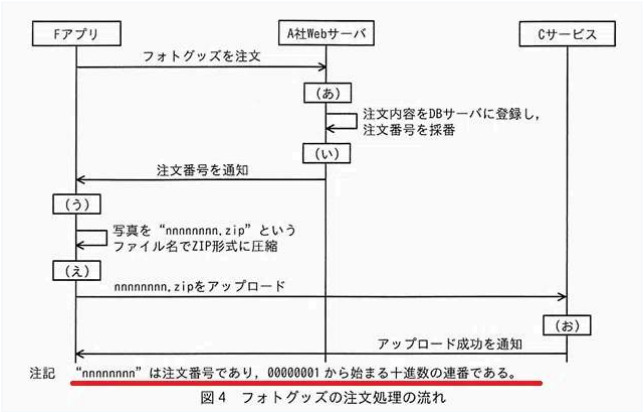


図４　フォトグッズの注文処理の流れ

- **該当箇所:** 「注記 “nnnnnnnn” は注文番号であり、**00000001から始まる十進数の連番**である。」
- **ファイル形式:** 「写真を “nnnnnnnn.zip” というファイル名で**ZIP形式に圧縮**」

ここから、ファイル名が 00000001.zip, 00000002.zip, 00000003.zip, ... という**推測しやすい連番**になっていることが分かります。

ステップ3：ファイルをダウンロードする具体的な手順を組み立てる

攻撃者は「アクセスキー」と「推測可能なファイル名」を持っています。これらを使ってファイルをダウンロードする方法を組み立てます。

1. **ダウンロードリクエストの作成:** 攻撃者は、ステップ1で得たアクセスキーを使います。そして、ステップ2で判明した連番のファイル名（例: 00000001.zip）を指定して、Cサービスに対するダウンロード要求（HTTP GETリクエスト）を作成します。このとき、リクエストのAuthorizationヘッダーにアクセスキーを設定します 。
2. **攻撃の自動化と実行:** 攻撃者は、このダウンロード要求を自動的に、かつ繰り返し実行するスクリプトを作成します。スクリプトはファイル名の数字を 00000001 から1ずつ増やしながら、ファイルが存在しなくなるまで（例: サーバから404 Not Foundエラーが返されるまで）ダウンロード要求を送り続けます。

この手順により、サーバに保存されている全利用者の写真を網羅的にダウンロードすることが可能になります。

ステップ4：解答をまとめる

以上の手順を、解答として具体的にまとめます。

解答例: 取得したアクセスキーを使い、00000001から始まる連番の注文番号をファイル名として指定し、Cサービスにファイルのダウンロード要求を繰り返し実行する。

IPA解答例：設問１（３）

(3)	ファイル名に含まれる注文番号を 00000001 から順に増やしてダウンロードを試行する。

設問１（4）

この問題は、脆弱性3「F-URLの処理の不備」を利用して、攻撃者のWebサイト k-sha.co.jp にアクセスさせ、かつ認証トークンも窃取できるような、細工したURLの例を答える問題です。

(4) 表１中の下線④について，攻撃者の Web サイトにアクセスさせることができるように細工した URL の例を，攻撃者が取得したドメイン名を k-sha.co.jp とした場合で答えよ。

表１ 脆弱性診断結果（抜粋）

脆弱性	脆弱性の概要	解説
1	サーバ証明書の検証不備がある。	F アプリは，HTTPS でサーバと接続する際，サーバ証明書の検証エラーがあっても無視し，通信を続行する。そのため，HTTPS 通信の内容が盗聴されたり，改ざんされたりするおそれがある。盗聴されると，①盗聴した内容からアクセスキーとストレージ名を攻撃者が取得するおそれがある。 （省略）
2	C サービスのアクセスキーの保護に不備がある。	②攻撃者が F アプリから平文のアクセスキーとストレージ名を取得できる。そのアクセスキーを用いて，③攻撃者が E サービスの全利用者の写真を不正にダウンロードするおそれがある。 （省略）
3	F-URL の処理にアクセス制御の不備がある。	F-URL の url クエリパラメータに，④細工した URL が指定されることによって，攻撃者の Web サイトにアクセスしてしまうおそれがある。また，攻撃者が会員の認証トークンを取得するおそれがある。 （省略）

ステップ1：攻撃のゴールと、それを阻む壁を理解する

- **攻撃者のゴール:**
  1. アプリのWebViewに、自身のサイト(k-sha.co.jp)を表示させる。
  2. 表示させた自身のサイトから、アプリの getToken 関数を呼び出し、**認証トークンを盗む**。
- **攻撃を阻む壁:** 図7で示されている getToken 関数のセキュリティチェックです 。この関数は、呼び出し元のWebページのURLが「https://www.a-sha.co.jp」で始まっていないと、認証トークンを返してくれません 。

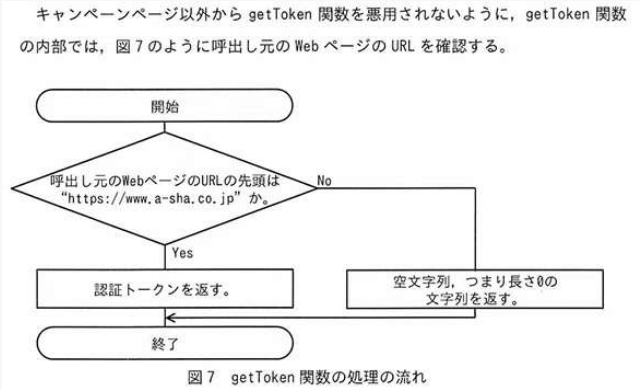


図 7　getToken 関数の処理の流れ

単純に f-app://campaign?url=https://k-sha.co.jp のようなURLを使っても、サイトは表示できますが、このチェックに引っかかり、肝心の認証トークンが盗めません。

ステップ2：セキュリティチェックの「穴」を見つける

攻撃者は、このセキュリティチェックをかいくぐる方法を考えます。

- **チェック方法:** URLの「先頭文字列」が一致するかどうかしか見ていません 。
- **穴:** URLの構造を悪用すれば、「先頭は https://www.a-sha.co.jp で始まる」という条件を満たしつつ、**実際にアクセスするドメインは別の場所**、というURLを作れる可能性があります。

ステップ3：攻撃用URLを組み立てる

ステップ2の「穴」を突くための具体的なURLを組み立てます。URLの仕様には、ホスト名の前に @ を使ってユーザー情報を含めることができる、というものがあります（例: https://user:pass@host）。これを悪用します。

1. **正規ドメインをユーザー情報に見せかける:** https://www.a-sha.co.jp@... のように記述します。 こうすると、URLの先頭は https://www.a-sha.co.jp となり、ステップ1のチェックをパスできます。

2. **本当の接続先を指定する:** @ の後ろに、攻撃者のドメイン k-sha.co.jp を記述します。
   https://www.a-sha.co.jp@k-sha.co.jp
    1. 多くのブラウザやWebViewは、このように記述されたURLを「**k-sha.co.jp というホストに、www.a-sha.co.jp というユーザー名でアクセスする**」と解釈します。結果として、WebViewは攻撃者のサイト k-sha.co.jp にアクセスします。
3. **F-URL全体を完成させる:** これをF-URLの形式に当てはめます。
   https://www.a-sha.co.jp@k-sha.co.jp

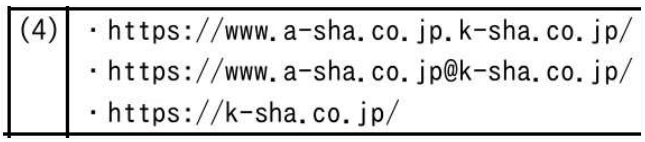
ステップ4：解答をまとめる

以上の手順で組み立てたURLが、この設問の解答となります。

解答例: https://www.a-sha.co.jp@k-sha.co.jp

このURLを使えば、getToken 関数のチェックを回避しつつ、攻撃者のサイトをWebViewに表示させ、認証トークンを窃取することが可能になります。

IPA解答例：設問１（４）



(4)	・https://www.a-sha.co.jp.k-sha.co.jp/ ・https://www.a-sha.co.jp@k-sha.co.jp/ ・https://k-sha.co.jp/

さいごに

IPAの解答例のうち、**最初の2つは脆弱性を完全に悪用するために有効な解答例**だと思います。理由はどちらも「先頭文字列しか見ていない」というチェックの甘さを突くテクニックを示しているからです３つめは認証トークンを取得できないので不完全な解答に思えますが。。。