問2　Web アプリケーション開発におけるセキュリティ対策に関する次の記述を読んで，設問 1～3 に答えよ。

A 社は，Web マーケティングを支援する従業員数 40 名の企業である。A 社では，現在，Web マーケティング分析システム（以下，W システムという）を Web アプリケーションとして独自に開発し，クラウドサービス事業者 S 社のクラウドサービス上で顧客にサービスとして提供している。W システムでは，A 社独自の分析アルゴリズムを用いて，顧客の Web サイトを分析する。顧客の評価は高く，販売も好調である。

〔W システムの概要〕

W システムの開発は，役員 T 氏とシステム担当の K 氏が行っている。T 氏がアルゴリズムなどの基本的な仕様を決め，K 氏が，Java，サーブレットや開発フレームワークなどを用いて開発している。また，ブラウザ側での機能性向上のために JavaScript も用いている。開発は K 氏が管理する PC（以下，開発用 PC という）上で行い，完成したプログラムをシステム管理者の F 氏が本番システムにデプロイし，運用している。

W システムの画面構成は，次のようになっている。

・総ページ数は 8 である。
・“ログインページ”で利用者を認証し，認証が成功すると，“ダッシュボードページ”へ遷移する。
・“ダッシュボードページ”からは，“分析キーワード入力ページ”などへ遷移できる。

W システムのシンプルなページの構成は，顧客にも好評である。

〔W システムの実装に関する脆弱(ぜい)性〕

A 社では，W システムのセキュリティ検査を，あらかじめ定められた手順に従い K 氏がブラウザを用いて手動で実施している。しかし，昨今の他社のセキュリティインシデント増加を受け，念のために，先月実施した社内のセキュリティ検査とは別に，今月になり初めて，外部の専門家にセキュリティ検査を依頼することにした。

そこで，セキュリティ専門会社 L 社の情報処理安全確保支援士である N 氏が，W システムのセキュリティ検査を担当することになった。

まず，N 氏が脆弱性検査ツールを使って本番システムを検査したところ，図 1 の Java コードで書かれたサーブレット SearchServlet に，SQL インジェクション脆弱性及びクロスサイトスクリプティング（以下，XSS という）脆弱性があることが判明した。図 2 は，図 1 の SearchServlet を呼び出す HTML コードである。

続いて N 氏が本番システムのソースコードを確認し，次の指摘をした。

・SQL インジェクション脆弱性への対処としては，図 1 の［　ア　］行目から［　イ　］行目までを①適切なコードに置き換える必要がある。

・XSS 脆弱性への対処としては，XSS 脆弱性を招く可能性の否定できない箇所について，HTML 形式での出力時に処置するよう改修することとする。

今回指摘された脆弱性は，検査すべき項目の中に含まれており，K 氏によるブラウザを用いた手動のセキュリティ検査によって発見され，開発用 PC 上のコードでは先月に修正されていた。しかし，K 氏からファイルを受け取った F 氏が，本番システムに修正版をデプロイし忘れたので，本番システムに脆弱性が残存したままとなっていた。

```
   (省略)   //package, import宣言など
 1:public class SearchServlet extends HttpServlet {
     (省略)   //変数やメソッドの定義など
 2:
 3:  protected void doPost(HttpServletRequest request, HttpServletResponse response) thr
  ows ServletException, IOException {
 4:    Connection conn = null;
 5:    String cname = request.getParameter("cname");
 6:    response.setContentType("text/html; charset=UTF-8");
 7:    PrintWriter out = response.getWriter();
 8:    try {
         (省略) //データベースにアクセスするためにconnを初期化など
 9:      String sql = "SELECT * FROM companylist WHERE cname = '" + cname + "'";
10:      Statement stmt = conn.createStatement();
11:      ResultSet rs = stmt.executeQuery(sql);
12:      if (cname != null && rs != null) {
13:        out.println("<html>");
14:        out.println("<head>");
15:        out.println("<title>分析結果</title>");
           (省略) //その他，JavaScriptの読込みなど
16:        out.println("</head>");
17:        out.println("<body>");
```

図 1　分析結果を表示するページを出力する Java コード（抜粋）

```
18:         out.println("<table border=1>");
19:         out.println("<tr><th>キーワード</th><th>アクセス</th><th>売上げ</th></tr>");
20:         while (rs.next()) {
21:           out.println("<tr><td>" + rs.getString(1));
22:           out.println("</td><td>" + rs.getString(2));
23:           out.println("</td><td>" + rs.getString(3));
24:           out.println("</td></tr>");
25:         }
26:         out.println("</table>");
            (省略) //その他
27:         out.println("</body>");
28:         out.println("</html>");
29:       }
30:     } catch (SQLException e) {
          (省略) //例外処理
31:     } finally {
          (省略) //データベースへのアクセスを終了する処理など
32:     }
        (省略) //その他，エラー処理など
33:   }
34:
      (省略)    //その他のメソッドの定義など
35:}
```

図 1　分析結果を表示するページを出力する Java コード（抜粋）（続き）

```
1:<form action="SearchServlet" method="post">
2:  分析キーワードを入力してください：<input type="text" name="cname" />
3:  <input type="submit" value="検索" />
4:</form>
```

図 2　SearchServlet を呼び出す HTML コード（抜粋）

〔W システムの設計に関する脆弱性〕

以前の W システムは，ログイン状態で 30 分間操作しないと，サーバ側で自動的にログアウトする仕様であった。自動的にログアウトした場合，ログアウト直前のページを閲覧するには，再度ログインをして，ダッシュボードページから所望のページを選択する必要があるので，不便だとの指摘が顧客からあった。これを改善するため，自動的にログアウトした場合に，ログアウト直前のページの閲覧を試みると，一旦はログインページに遷移するが，認証が成功すると，閲覧を試みたページへリダイレクトする機能（以下，リダイレクタ機能という）を導入した。例えば，W システムの URL である https://w-system.a-sha.jp/ で動作する Web アプリケーションにおいて，W システムにログインしていない状態で，https://w-system.a-sha.jp/dashboard.jsp という URL にアクセスすると，図 3 のように生成された URL へリダイレクトされる。

```
https://w-system.a-sha.jp/LoginServlet?redirect_url=https://w-system.a-sha.jp/dashboard.jsp
```

図3　リダイレクタ機能によって生成されたURL

認証用サーブレット LoginServlet で認証が成功すると，https://w-system.a-sha.jp/dashboard.jsp へリダイレクトされる。

リダイレクタ機能を含めて W システムの設計に関して検査を実施したところ，N 氏から幾つかの指摘があった。主な指摘は次の3点であった。

指摘1：W システムの認証後のリダイレクタ機能は，オープンリダイレクタの問題を招く。修正する必要がある。

指摘2：W システムからの Cookie 発行の際，TLS 通信時だけ Cookie をブラウザから送信する［　a　］属性を設定する必要がある。

指摘3：JavaScript から Cookie を操作できないようにする［　b　］属性を設定する方がよい。

K 氏は，指摘 1 に対しては，W システムの特性を考慮して，②ホワイトリスト方式によるリダイレクタ機能を採用することにした。指摘 2 と指摘 3 については，N 氏の指摘どおりに改修することにした。

〔脆弱性対策の強化〕

脆弱性が残存した原因も踏まえ，脆弱性対策の強化として，次のとおり提案と指摘を N 氏は行った。

・変更管理プロセスを改善すべきである。

・③脆弱性検査手順を改善すべきである。

・K 氏によるブラウザを用いた検査には問題がある。W システムに XSS 脆弱性があったとして，適当な検査用文字列を入力しても，④一部のブラウザでは XSS 攻撃の試みを完遂できないことがある。

・S 社のクラウド型 WAF サービス（以下，S サービスという）を導入することを推奨する。S サービスでは，Web システムに脆弱性が発見された際，短時間で適切なシグネチャが WAF に追加される。したがって，S サービスを利用していれば，WAF を利用せずに W システムを改修するケースと比較して，⑤Web アプリケーションサーバへのリスクの低減を期待できる。

A社では，N氏のこれらの提案と指摘を検討し，適切に対処することにした。

設問1　〔Wシステムの実装に関する脆弱性〕について，(1)～(3)に答えよ。

(1)　本文中の［　ア　］，［　イ　］に入れる，適切な行番号を答えよ。また，本文中の下線①について，必要な全てのコードを解答群の中から選び，適切な順序に並べ替えて解答欄の左から順に記号で答えよ。

解答群

ア　PreparedStatement pstmt = conn.prepareStatement(sql);

イ　pstmt.setString(1, cname);

ウ　ResultSet rs = pstmt.executeQuery();

エ　ResultSet rs = pstmt.executeQuery(sql);

オ　String sql = "SELECT * FROM companylist WHERE cname = '" + cname + "'";

カ　String sql = "SELECT * FROM companylist WHERE cname = ?";

(2)　図1には，XSS脆弱性を招く可能性を否定できないコードがある。N氏の指摘に従い，改修すべき行番号を全て答えよ。

(3)　XSS脆弱性に対する修正を，N氏の指摘に従い，解答群の中から選び，記号で答えよ。

解答群

ア　CSRF対策トークンを使う。

イ　Refererヘッダの値のURLのドメイン名がWシステムのものであることを確認する。

ウ　出力時に<，>，&，"，'の各文字をエスケープする。

エ　同一生成元ポリシを適用する。

オ　バインド機能を使う。

設問２　〔W システムの設計に関する脆弱性〕について，(1)，(2) に答えよ。

(1)　本文中の［　a　］，［　b　］に入れる適切な字句を，解答群の中から選び，記号で答えよ。

解答群

ア　Expires　　イ　HttpOnly　　ウ　Max-Age

エ　Secure　　オ　Secured

(2)　本文中の下線②について，W システムで採用するホワイトリスト方式の適切な仕様を，40 字以内で具体的に述べよ。

設問３　〔脆弱性対策の強化〕について，(1)～(3) に答えよ。

(1)　本文中の下線③について，どのように改善すべきか。改善された検査手順を 30 字以内で述べよ。

(2)　本文中の下線④について，XSS 攻撃の試みを完遂できないことがある理由を 30 字以内で述べよ。

(3)　本文中の下線⑤について，低減できるリスクを 50 字以内で述べよ。