問3　アプリケーション開発時の脆弱性対策に関する次の記述を読んで，設問1，2に答えよ。

C社は，一般消費者向けにファックスや電話による，生活用品の通信販売を行っている，従業員数50名の会社である。C社の企画開発課は申込方法の多様化を目的として，インターネットで申込みを受けるためのXシステム（図1）を開発してきた。Xシステムでは，Webアプリケーション（以下，Webアプリという）のログイン画面で，利用者IDとパスワードによって利用者認証を行う。C社の開発ポリシには，サービス開始前に第三者によるセキュリティ検査を実施することが規定されている。そこで，企画開発課のF課長は，部下のG君に指示してWebアプリを対象とした疑似侵入テストと，データベース（以下，DBという）サーバを対象とした脆弱性診断を，セキュリティ専門会社のD社に依頼した。

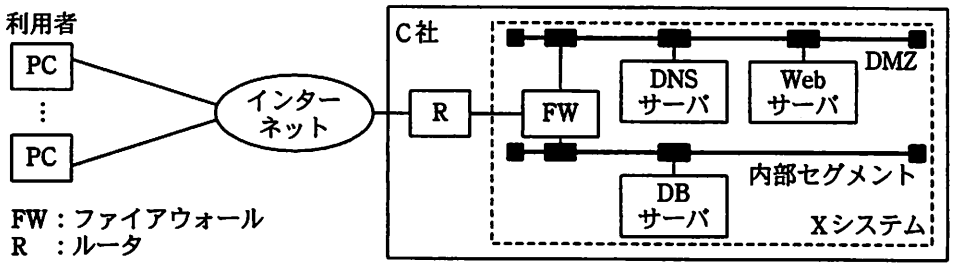

図1　Xシステムの構成

〔Webアプリのセキュリティに関する問題点〕

D社のH氏から疑似侵入テストと脆弱性診断の結果が報告された。次は，疑似侵入テストで見つかった脆弱性に関する会話の一部である。

H氏　：まず，各ログインセッションを識別するセッション識別子は，ログイン日と会員番号を組み合わせた文字列として構成され，図2のようにcookieにセットされますね。このままでは，悪意をもつ者に他人のセッション識別子を推定され，①容易になりすましアクセスができてしまいます。影響と対策については報告書に記述しましたので参照してください。

G君　：分かりました。セッション識別子の生成部分を修正します。

session_id = 20081223309804

注　会員番号 309804 の利用者が 2008 年 12 月 23 日にログインしたときの cookie の値を示している

図 2　cookie にセットされたセッション識別子の例

H 氏　：次は，クロスサイトスクリプティング（以下，XSS という）についてです。Web アプリでは，XSS の脆弱性が多数見つかりました。

G 君　：XSS 対策については，図 3 のプログラムのように，利用者からの入力データに<script や<iframe などの文字列が含まれていた場合は，入力チェックで，処理を中止するようにしていました。

H 氏　：入力チェックだけで完全な XSS 対策を行うのは困難です。まず HTML 出力の際に “<”，“>”，“"”，“'”，“&” をそれぞれ “<”，“>”，“"”，“'”，“&” に置換することが基本です。その上で，プログラムが HTML タグの②ある特定の属性に値を出力する場合は，個別の対処が必要です。

G 君　：（報告書を見ながら）なるほど，そのように対応します。

```
 1 #!/usr/bin/perl
 2 use CGI;                              # 汎用 CGI 処理用モジュール CGI の使用を宣言
 3 use DBI;                              # 汎用 DB 処理用モジュール DBI の使用を宣言
 4 $cgi = new CGI;                                  # CGI クラスのオブジェクトを生成
 5 $uname = $cgi->param('uname');                   # uname パラメタを取得
 6 die if (chkstr($uname));                         # uname に対する入力チェック
（省略）
11 $dbh = DBI->connect('DBI（省略）') or die;       # DB に接続
（省略）                                            # uname を基に SQL 文を組み立て，実行
18 $dbh->disconnect or die;                         # DB から切断
（省略）
24 print "<img src=\"http://www.c-sha-△△.com/img/a0.png\" alt=\"（省略）\">";
（省略）
29 print "<img src=\"../img/a1.png\" alt=\"（省略）\"></div>";
30 print "<form action=\"../cgi-bin/shori-a.cgi\" method=\"post\">";
31 print "<p>利用者名：<input name=\"uname\" type=\"text\" value=\"$uname\">";
（省略）
50 print "<img src=\"https://www.c-sha-△△.com/img/a9.png\" alt=\"（省略）\">";
51 print "</div></body></html>";                    # HTML 終了部分を出力
52
53 sub chkstr {                                    # 引数内に特定キーワードが存在した場合
54   my $str = shift;                              # true を返す
55   if ($str=~/(<script|<iframe|（省略）)/) {return(true)};
56 }
```

注　（省略）は，記述を省略していることを意味する。
www.c-sha-△△.com は，X システムが利用しているドメイン名である。
△△は，特定の文字列を表す。

図 3　プログラム（抜粋）

H氏　：次に，図3のプログラムはHTTPSでアクセスする画面を出力するものですが，その画面を御社で推奨されているブラウザで見たときに，③HTTPSでの正常なアクセスを意味する鍵マークが表示されなくなっています。

F課長：これは注意が足りませんでした。ほかのプログラムも含めて確認し，修正します。

〔DBセキュリティに関する問題点〕

脆弱性診断によって，DBセキュリティに関する問題点も報告された。次は，その問題点に関する会話である。

H氏　：DBサーバには，利用者の個人情報が平文で格納される仕様になっています。万が一，不正アクセスによって個人情報が流出した場合に備えて，Webアプリで個人情報を暗号化して格納することをお勧めします。

G君　：実は，DBMS側で自動的に暗号化／復号する透過的データ暗号化機能（以下，透過的暗号化という）を利用する予定です。

H氏　：残念ながら④透過的暗号化の効果は限定的です。

H氏は透過的暗号化について説明した。

F課長：防止できる脅威を考えると，Webアプリによる暗号化を採用すべきだな。個人情報の登録と参照をしている箇所を洗い出して，実装してくれ。

G君　：承知しました。ざっと見積もって1週間以上は掛かると予想されます。

F課長：仕方ない。よろしく頼む。

H氏　：今後システム開発をするときにはもっと早い段階で暗号化について検討し，実装すべきですね。次に，Xシステムのログイン認証用のパスワード情報について確認します。DBに格納されているパスワード情報はハッシュ値のように見えますが，どのような方法で算出しているのですか。

G君　：パスワードをMD5関数の引数にして算出しています。

H氏　：パスワードだけからハッシュ値を算出すると，悪意ある第三者にハッシュ値が知られた場合，検索エンジンや専用ツールを使って，元の値であるパスワ

ードを突き止められてしまう可能性があります。

F課長：パスワードを突き止められることで，X システムに不正ログインされる可能性があるのは理解できますが，ほかに何か問題があるのですか。

H氏　：利用者の多くは，ほかのサイトでも同じパスワードを使用しているようです。攻撃者は不正に入手した利用者 ID，パスワードを使ってほかのサイトに不正ログインを試みるでしょう。したがって，万が一，情報流出した際には，御社に対する非難の声が大きくなりかねませんし，利用者にほかのサイトのパスワードを変更する負担を強いることにもなります。そういう最悪の事態にならないようにするためにも，⑤より工夫してパスワードのハッシュ値を算出した方がよいでしょう。

F課長：なるほど，当社だけの問題ではないのですね。ハッシュ値の算出方法を変更します。

その後，H 氏から指摘された脆弱性に関する対策を完了し，C 社は正式にサービスを開始した。

設問1　〔Webアプリのセキュリティに関する問題点〕について，(1)～(3)に答えよ。

(1) 本文中の下線①について，悪意をもつ者が，どのようにして他人になりすますことができるのか。その方法を50字以内で具体的に述べよ。

(2) 本文中の下線③の原因となっている図3のプログラムの部分を，その行番号で答えよ。また，どのように修正すればよいかを25字以内で述べよ。ここで，a0.png～a9.pngの画像ファイルは同じフォルダに配置されているものとする。

(3) 本文中の下線②の属性に該当するものを二つ挙げ，それぞれ15字以内で答えよ。

設問2　〔DBセキュリティに関する問題点〕について，(1)，(2)に答えよ。

(1) 本文中の下線④について，DBに格納されている情報に対する不正閲覧のうち，透過的暗号化によって防止できるものを解答群の中から一つ選び，記号で答えよ。

解答群

ア　SQLインジェクションによる不正閲覧

イ　サーバ管理者によるSQL文を利用した不正閲覧

ウ　図1中の内部セグメントにおける通信傍受による不正閲覧

エ　バックアップ媒体を入手した第三者による不正閲覧

(2) 本文中の下線⑤について，よく使われる文字列をパスワードにしていた場合でも，ハッシュ値から元のパスワードを突き止められないようにするためのハッシュ値の算出方法を，50字以内で述べよ。