Klausur Websecurity WS23/24

1. Klausurtermin am 05.03.2024

Name:

Matrikelnummer:

Hinweise

- Bitte schreiben Sie Ihren **Namen und Matrikelnummer** auf das Deckblatt.
- **KEINESFALLS** die Klammerung auflösen.
- Nur Dokumentenechte Stifte benutzen.
  Keinen roten oder grünen Stift nutzen.
  Keinen Bleistift, Tintenkiller oder TippEx benutzen.
- Es sind keinerlei Hilfsmittel erlaubt.
- Schreiben Sie leserlich. Unlesbare Antworten werden nicht gewertet.
- Falls Sie mehr Platz benötigen, fragen Sie nach Papier.
- Die Bearbeitungszeit beträgt **90 Minuten**.

Dieser Teil ist zur Auswertung bestimmt und
darf von den Teilnehmenden der Klausur **nicht** ausgefüllt werden!

Aufgabe	**A1** 5P	**A2** 6P	**A3** 4P	**A4** 8P	**A5** 8P	**A6** 8P	**A7** 5P	**A8** 6P	**Gesamt** 50P	**Note**
erreichte Punkte										

Aufgabe 1: Multiple Choice **(5 Punkte)**

Beurteilen Sie die folgenden Aussagen, indem Sie die richtigen ankreuzen und die falschen leer lassen! Ein Punkt für jede Teilaufgabe wird nur dann vergeben, wenn Sie in dieser alle richtigen und keine falschen Kreuze gesetzt haben.
Hinweis: Es können alle, keine oder mehrere Aussagen richtig sein.

(a) (1 Punkt) HTTP Status codes

Der HTTP Status-code 400 zeigt an, dass beim Verarbeiten der HTTP Anfrage auf dem Server ein Fehler aufgetreten ist.	◯
Ein HTTP Status-code beginnend mit 3 deutet auf eine Weiterleitung hin.	◯

(b) (1 Punkt) Cookies

Cookies ermöglichen es einem Server mehrere HTTP-Anfragen einem Nutzer zuzuordnen.	◯
Das `httpOnly` Cookie-Flag verhindert, dass ein Browser das Cookie über einen verschlüsselten Kanal überträgt.	◯

(c) (1 Punkt) SOP & CORS

Ähnlich zur Same-Origin Policy existiert eine Same-Destination Policy.	◯
Aufgrund des Cross Origin Resource Sharing wird die Same Origin Policy heutzutage nicht mehr benötigt.	◯

(d) (1 Punkt) URL

`http://foo:bar@tu.berlin:80/sect/` ist nach URL-Schema gültig.	◯
`https://0x8D172338/sect/` ist nach URL-Schema gültig.	◯

(e) (1 Punkt) DNS

Das Domain Name System (DNS) löst Domains zu IP Adressen auf.	◯
DNS unterstützt nur IPv4 Adressen.	◯

Punkte	

Aufgabe 2: SOP, Cookies, CORS **(6 Punkte)**

(a) (1 Punkt) Nennen Sie die drei Kriterien, anhand derer die Same-Origin-Policy den Zugriff von Seite A auf Seite B überprüft.

-
-
-

(b) (1 Punkt) Erklären Sie anhand eines einfachen Beispiels, welchen Nutzen die Same-Origin-Policy hat(te).

(c) (2 Punkte) Ein Server setzt die folgenden 3 Cookies:

(a) `Set-Cookie: user=admin; domain=admin.vuln.com; Secure`

(b) `Set-Cookie: group=root; domain=admin.vuln.com; path=/groups`

(c) `Set-Cookie: session=12345; domain=.vuln.com; HttpOnly`

Geben Sie zu jeder URL an, welche Cookies ein Browser mitschicken würde. Es genügt, wenn Sie den Buchstaben des jeweiligen Cookies neben die URL schreiben (bspw. a, b, c).

- `http://admin.vuln.com/`
- `https://admin.vuln.com/groups/admin`
- `http://user.vuln.com/groups`
- `https://user.vuln.com/profile`

(d) (2 Punkte) Die Stadt Berlin stellt im Rahmen einer 'Open data' Initiative verschiedene Datensätze öffentlich zur Verfügung. Beispielsweise findet sich unter der URL[1] eine Liste der häufigsten Vornamen in Pankow für 2023. Beim Abruf dieser URL finden Sie folgenden Header in der HTTP Antwort: `access-control-allow-origin: *`
Entscheiden und begründen Sie, ob dieser CORS Header in dieser Situation ein Sicherheitsrisiko darstellt, und falls ja, wie man dieses behebt.

[1] `https://raw.githubusercontent.com/berlinonline/haeufige-vornamen-berlin/master/data/cleaned/2023/pankow.csv`

Punkte	

Aufgabe 3: Client-Side Validation **(4 Punkte)**

Das Webinterface eines smarten Geräts zeigt unter "/" eine Login-Maske zur Abfrage des Administrator-Passworts. Sie schauen sich den Quelltext dieser Seite im Browser an und finden folgende Zeilen[2]:

```
1 <script>
2 if(prompt("Passwort bitte:") == atob("YWRtaW4xMjM=")) {
3         document.location = '/admin/';
4 } else {
5         alert("Falsches Passwort");
6 }
7 </script>
```

(a) (2 Punkte) Welche zwei sicherheitsbezogene Probleme fallen Ihnen bei dieser Implementierung auf? Beschreiben Sie diese jeweils in einem Satz!

(b) (1 Punkt) Erklären Sie in 1-2 Sätzen, weshalb Client-Side Validation im Allgemeinen nicht sinnvoll ist.

(c) (1 Punkt) Geben Sie mindestens 2 Argumente an, weshalb dennoch in manchen Branchen (bspw. Kopierschutz von Spielen / Filmen) auf dieses Prinzip gesetzt wird!

[2] `prompt` - Zeigt ein Eingabefeld an und gibt die Eingabe zurück. `atob` - Dekodiert eine Base64-kodierte Zeichenkette

Punkte	

Aufgabe 4: Cross-Site Scripting **(8 Punkte)**

Gegeben sind Ihnen die folgenden 3 Qutelltextausschnitte aus einer PHP-basierten Webapplikation:

(1) Quelltext 1:

```
1 <h1>Hi <?php echo $_GET['name']; ?></h1>
```

(2) Quelltext 2[3]:

```
1  <?php
2  include "db.php";
3  // Nutzer-Kommentare aus der Datenbank holen.
4  $comments  = $db->getComments(); ?>
5  <ul>
6  <?php foreach($comments as $c) {
7   echo "<li author='". str_replace(["<",">"], "", $c['author']) ."'>";
8   echo htmlentities($c['text']);
9   echo "</li>";
10 } ?>
11 </ul>
```

(3) Quelltext 3[4]:

```
1 <div id='backdiv'>Zur&uuml;ck</div>
2 <script>
3  var divElem  = document.getElementById("backdiv");
4  divElem.innerHTML =  '<a href="' + document.referer + '">Zurueck</a>';
5 </script>
```

(a) (3 Punkte) Geben Sie für jeden Quelltext an, um welche Art von Cross-Site Scripting Lücke es sich handelt und wie ein Angreifer ein `alert(1)` ausführen könnte.

(a)

(b)

(c)

[3] `foreach` ist eine For-Schleife über alle Elemente einer Liste.
`str_replace($search, $replace, $subject)` ersetzt Zeichen aus `$search` mit `$replace` in `$subject`.

[4] `getElementById` sucht und gibt das HTML-Element mit dem entsprechenden `id` Attribut zurück.

Punkte	

(b) (2 Punkte) Beschreiben Sie anhand von zwei unterschiedlichen Angriffsszenarien, wozu ein Angreifer Cross-Site Scripting nutzen kann!

(c) (1 Punkt) Der Senior-Entwickler schlägt zur Behebung der Lücke in Quelltext 3 folgende server-seitige Lösung vor:

```
1 <?php
2 // Referer der Anfrage
3 $referer = htmlentities($_SERVER['HTTP_REFERER']);
4 echo '<a href="' . $referer . '">Zur&uuml;ck</a>';
5 ?>
```

Zeigen Sie anhand eines Beispiel-Exploits, weshalb weiterhin eine Cross-Site Scripting Lücke besteht!

(d) (1 Punkt) Beschreiben Sie, wie man das Cross-Site Scripting Problem im Quelltext 3 tatsächlich beheben könnte.

(e) (1 Punkt) Wägen Sie begründet ab, welche Cross-Site Scripting Art für einen Angreifer generell am interessantesten ist.

Punkte	

Aufgabe 5: Misconfiguration **(8 Punkte)**

(a) (3 Punkte) Geben Sie an, welchen HTTP-Security Header **und** Wert die Seite A setzen muss, damit ...

(a) ... Seite B diese nicht in einem Iframe einbinden kann?

(b) ... der Browser *keine* Javascript-Dateien von externen Seiten lädt?

(c) ... der Browser die Webseite zukünftig *nicht* über HTTP abruft?

(b) (1 Punkt) Geben Sie an, welche Klasse von Sicherheitslücken effektiv durch den `Content-Security-Policy` Header vermieden werden kann:

(c) (2 Punkte) Laut Google sind 96% des WWW mit SSL/TLS Verbindungen gesichert. Nennen Sie 2 Risiken und mindestens 2 daraus resultierende Angriffsmöglichkeiten, wenn unverschlüsselte Verbindungen genutzt werden.

Punkte	

(d) (1 Punkt) Sie bekommen mit, wie ein Administrator eine neue Webapplikation per `git clone https://<unternehmens-repository>/website.git` im `DocumentRoot` des Webservers installiert. Erklären Sie, weshalb dies ein potentielles Sicherheitsproblem darstellt!

(e) (1 Punkt) Darauf angesprochen antwortet der Administrator: "Wir haben eine WebApplication-Firewall, also kann nichts passieren!". Beziehen Sie Stellung zu der Aussage! Teilen Sie Meinung des Administrators?

Punkte	

Aufgabe 6: SQL Injection **(8 Punkte)**

In einer Webapplikation finden Sie den folgenden Quellcode[5] in `login.php`:

```
1  <?php
2  include "db.php"; // Verbindung aufbauen
3  $uid = $_POST['uid'];
4  $result = $db->query("SELECT * FROM users WHERE uid = $uid");
5  if(isset($result['uname'])) {
6   echo "Hello " . htmlentities($result['uname']) . "!";
7   if($result['upw'] === md5($_POST['pw'])) {
8    echo "Login erfolgreich!";
9   } else { echo "Login fehlgeschlagen"; }
10 } ?>
```

Zudem ist Ihnen die Datenbankstruktur gegeben. Die Notation der angedeuteten Passwort-Hashes können Sie ebenfalls übernehmen, falls notwendig.

users		
uid	uname	upw
1	admin	8ef013[...]
2	user1	7abe83[...]
...	...	...

posts			
pid	title	content	public
1	First post	Hello world	1
2	Secret post	Very secret content	0
...	...	...	...

(a) (2 Punkte) Geben Sie einen SQL Injection Exploit an, mit dem ein Angreifer den geheimen Post aus der Tabelle *posts* abrufen kann.

(b) (2 Punkte) Der Administrator hat ein sehr starkes und zufälliges Passwort gewählt, welches nicht brute-forcebar ist. Dennoch wurden Zugriffe auf seinen Account festgestellt. Beurteilen Sie, ob es einen Zusammenhang mit der vorliegenden Sicherheitslücke gibt und falls ja, geben Sie an, wie der Exploit ausgesehen haben könnte.

[5] === vergleicht den Wert **und** den Typ der Operanden.

Punkte	

(c) (2 Punkte) Beschreiben Sie die Funktionsweise von Boolean-based SQL Injection! Gehen Sie dabei auf zwei unterschiedliche Nachteile gegenüber der "klassischen" SQL Injection ein!

(d) (1 Punkt) Ein Entwickler schlägt folgende Änderung[6] der Zeile 4 im o.g. Quelltext vor. Begründen Sie, ob damit die SQL Injection verhindert wird. Falls nein, geben Sie einen Exploit an!

```
4 $db->query("SELECT * FROM users WHERE uid = " . intval($uid));
```

(e) (1 Punkt) Erklären Sie weshalb *Prepared Statements* stets SQL-Queries mit String-Konkatenation vorgezogen werden sollten!

[6] `intval` wandelt die Eingabe in eine Zahl um.

Punkte	

Aufgabe 7: Path Traversal **(5 Punkte)**

In einer Python-basierten Webapplikation finden Sie den folgenden Code:

```
1  def isallowed(p):
2   if "/storage/documents/" in p:
3    return True
4   return False
5
6  def securepath(p):
7   return p.replace("../", "")
8
9  @app.route("/download")
10 def download():
11  path = request.args.get("path")
12  if isallowed(path):
13   return open(securepath(path)).read()
14  else:
15   return "Not found"
```

(a) (2 Punkte) Beurteilen Sie, ob die Datei-Download Funktionalität gegen Path Traversal anfällig ist und ein Angreifer an die Datei `/etc/passwd` gelangen kann? Falls ja, geben Sie einen entsprechenden Exploit an!

Punkte	

(b) (2 Punkte) Ein anderer Entwickler behauptet, der folgende PHP Code[7] wäre sicher(er), da er keine Umgehungsmöglichkeit kenne. Begründen Sie, weshalb der Entwickler falsch liegt, und beschreiben Sie einen möglichen Bypass!

```
1 $p = $_GET['path'];
2 if (!str_contains($p, "/storage/documents/")) {die("Invalid path");}
3 $sp = str_replace(".", "", $p);
4 echo file_get_contents($sp);
```

(c) (1 Punkt) Beschreiben Sie in eigenen Worten oder Pseudo-Code wie sie sicherstellen würden, dass nur Dateien aus dem Ordner `/storage/documents/` abgerufen werden können!

[7] `str_contains` prüft, ob Argument 2 in Argument 1 enthalten ist.
`str_replace` ersetzt **alle** Vorkommnisse 1. Arguments in dem 3. Argument mit dem 2. Argument.
`die` zeigt den Text an und beendet die PHP-Ausführung.

Punkte	

Aufgabe 8: Authentication & Authorization **(6 Punkte)**

(a) (1 Punkt) Grenzen Sie die Begriffe Authentifizierung & Authorisierung voneinander ab.

(b) (1 Punkt) Grenzen Sie die Begriffe horizontale & vertikale Rechteausweitung voneinander ab.

(c) (2 Punkte) Nennen und beschreiben Sie jeweils in einem Satz die **zwei** Authentifizierungsfaktoren, die als 2. Faktor angesehen werden!

(d) (2 Punkte) Ein Entwickler behauptet stolz, er habe das "Passwort-vergessen"-Problem seiner Nutzer gelöst! Statt einer komplexen Passwort-Policy sind alle Passwörter auf 5 Ziffern begrenzt. "Das macht es den Nutzern einfach, sich diese zu merken!" sagt er. Die Passwörter würden von der Webapplikation mit *salt* und geheimen *pepper* ausreichend verstärkt. Zudem seien Brute-Force Risiken aufgrund der Nutzung des bcrypt-Algorithmus ausgeschlossen, so der Entwickler.
Beurteilen Sie den Lösungsansatz des Entwicklers! Stimmen Sie ihm zu, oder sehen Sie Schwächen in diesem Ansatz?

Punkte	