AH

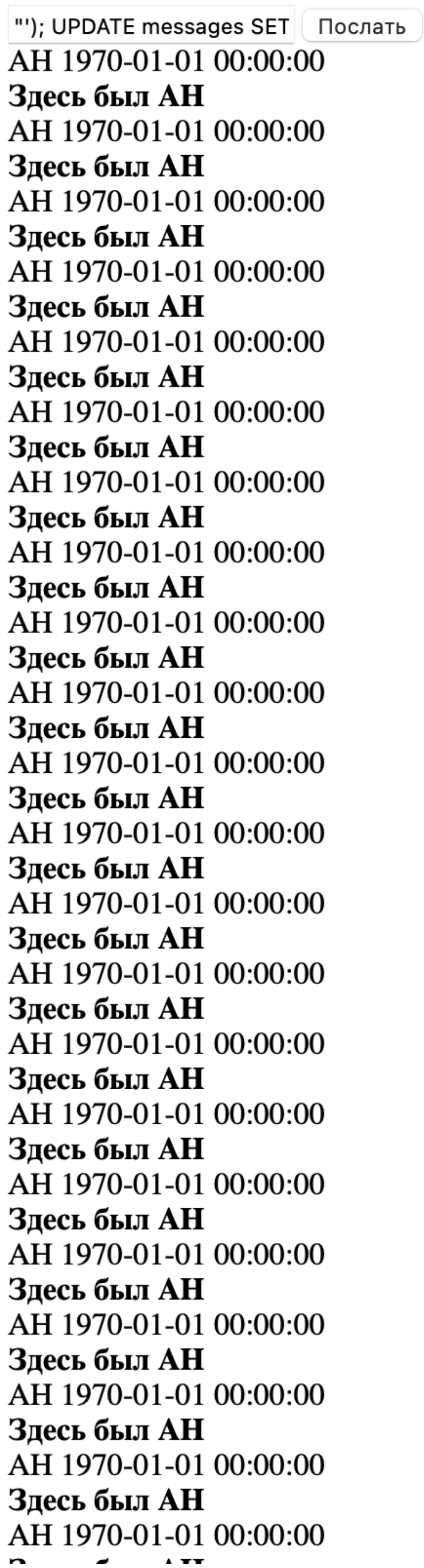
1) SQL Injection:
Вставляем в поле ввода сообщения:

```
"'); UPDATE messages SET UserName = 'AH', Message = 'Здесь был
AH', Timestamp = '1970-01-01 00:00:00'; --
```

Мы закрываем) и добавляем свою SQL-команду:

```
$sql = "INSERT INTO messages(UserName, Message)
                VALUES('$user','$message')";
        $cursor = $conn->multi_query($sql);
        $conn->close();
```

1. Подготовленные выражения: PDO или MySQLi с подготовленными выражениями.
2. Экранирование ввода: `mysqli_real_escape_string()` для экранирования данных.
3. Минимизация прав.



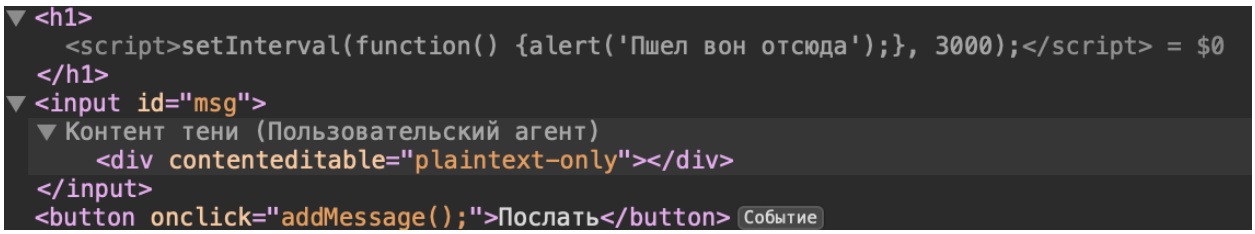
2) JS Injection:
При авторизации в логин вставить строку:

```
<script>setInterval(function() {alert('Пшел вон отсюда');},
3000);</script>
```

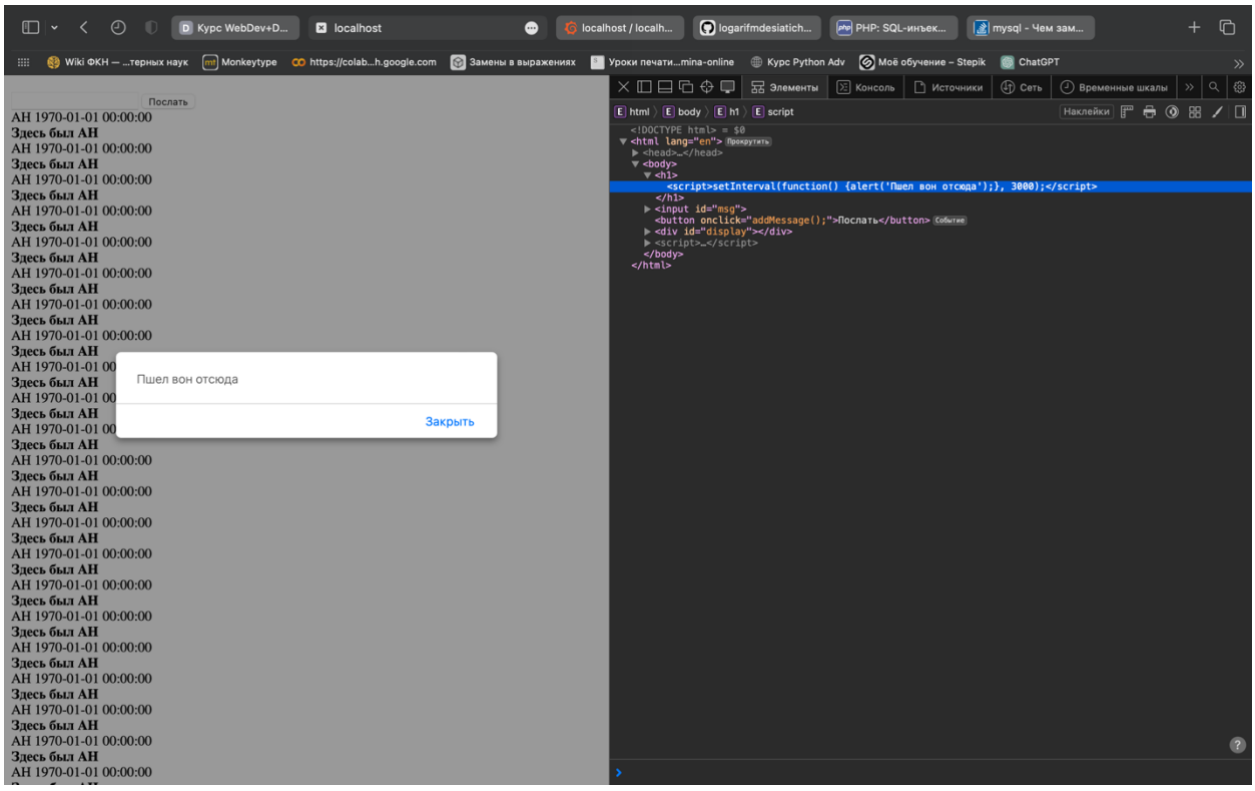
Тк значение логина подставляется без экранирования вставляются в chat.php:

```
<body>
    <h1><?=$_SESSION["user"]?></h1>
    <input id="msg">
    <button onclick="addMessage();">Послать</button>
    <div id="display"></div>
</body>
```

, то мы получаем:



И наблюдаем каждые 3 секунды надоедливое окно:



Защита от JavaScript-инъекций:

1. Экранирование вывода: Использовать `htmlspecialchars()` для экранирования специальных символов.
2. Content Security Policy (CSP): Настроить заголовки CSP для ограничения выполнения скриптов.
3. Не вставлять пользовательский ввод непосредственно в HTML.

+ ограничение количества попыток входа или таймаут между неудачными попытками для защиты от брутфорса.