Лабораторная работа №5

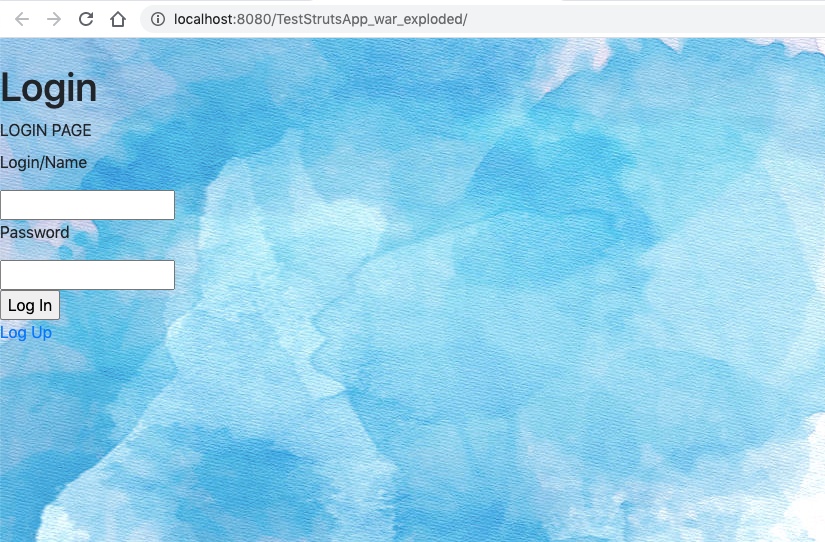
Карпов Алексей

12 группа кафедра ИСУ

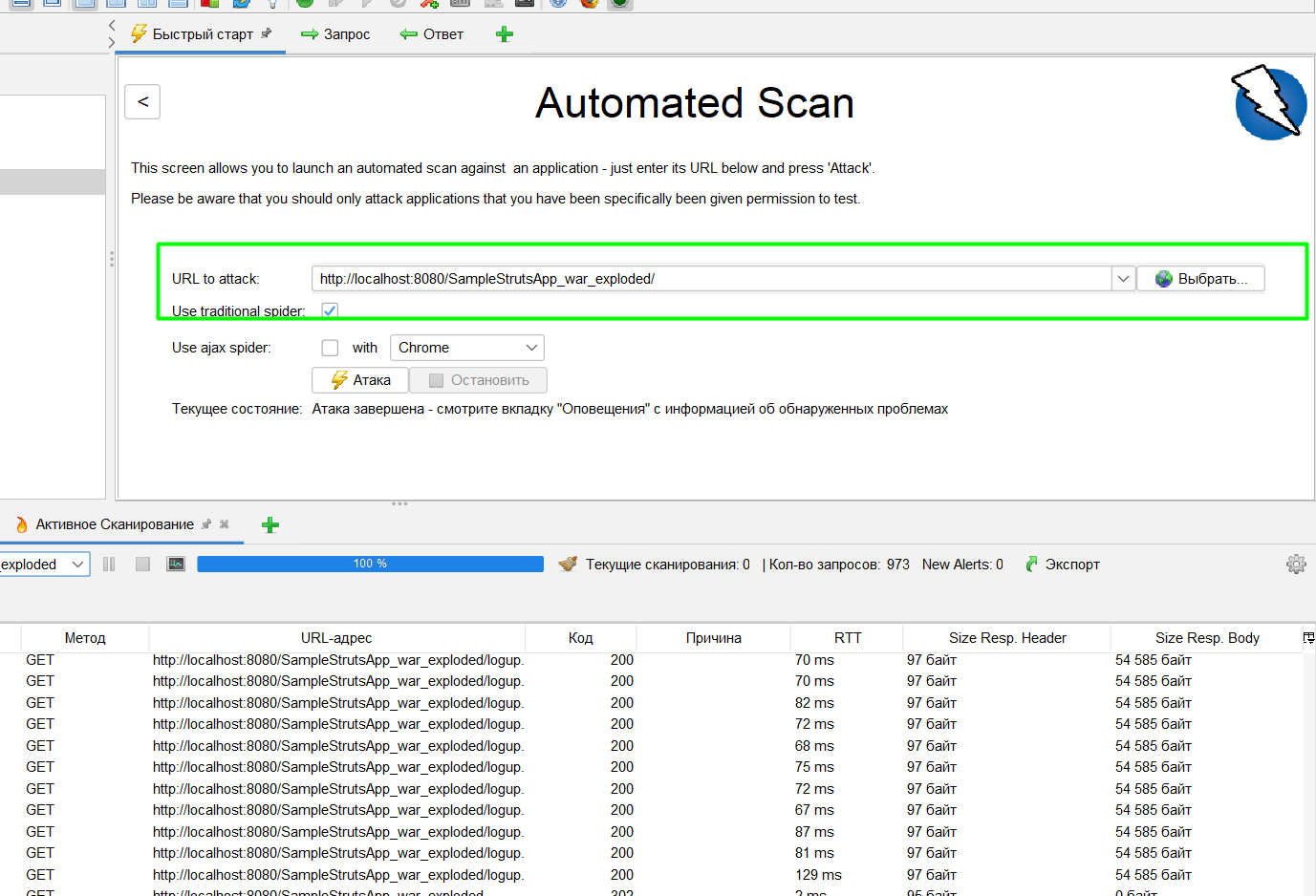
В данной лабораторной работе было принято рассмотреть систему собственного проекта под названием Flexinder (https://github.com/Seal72/Flexinder), являющуюся аналогом сайт-знакомств.

Для начала создадим артефакт самого приложения и запустим томкат сервер, положив артефакт в его директорию.

Получим следующий результат после разворачивания:



После того, как развернули нашу систему, копируем адрес и вставляем в программу OWASP ZAP для того, чтобы протестировать её на уязвимости, а потом их исправить, чтобы злоумышленники ими не воспользовались:

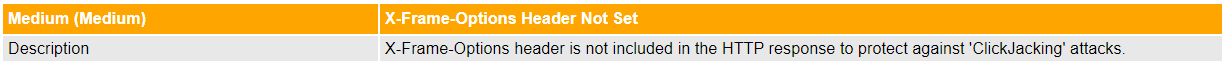
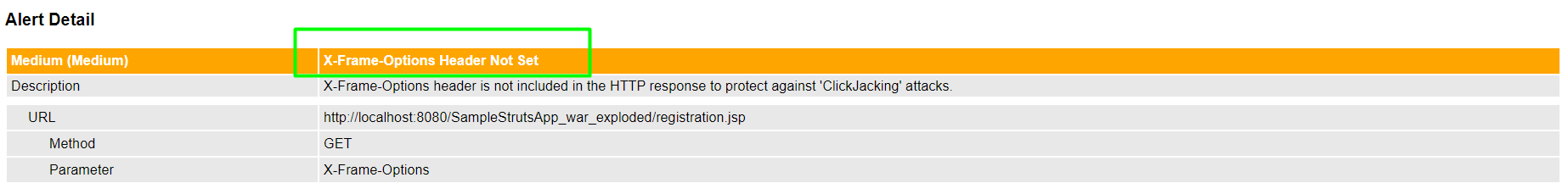


После этого вставляем наш URL в поле для атаки и жмём кнопку Attack.

Таким образом, проверяются все возможные места на сайте, которые могут быть уязвимыми. То есть таким же образом они могут быть обнаружены злоумышленником, который получит несанкционированный доступ к системе.

Получили отчет о сканировании веб приложения.

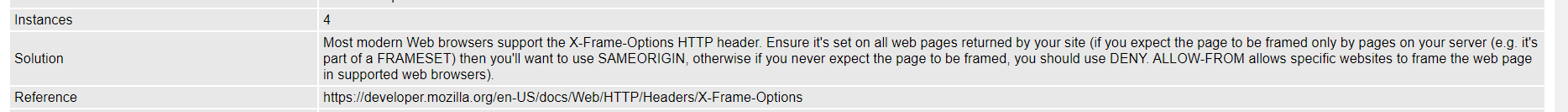
Рассмотрим одну из уязвимостей:



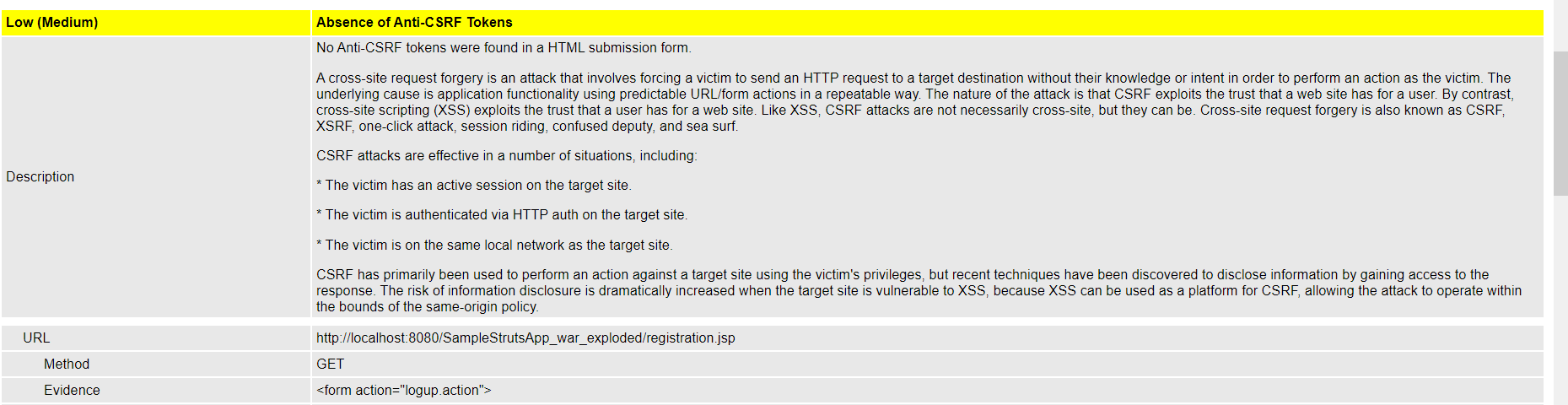
Как видно на скриншоте данная проблема может быть использована для атаки Кликджэкингом, то есть пользователю будет предоставлена ссылка на вредоносный контент с помощью, которого появится возможность получить доступ к аккаунту пользователя, так как он будет считать страницу внешне безопасной.

Это можно сделать так: поверх ссылки вредоносная страница размещает прозрачный <iframe> с src с нашего сайта, таким образом, что кнопка находится прямо над этой ссылкой. Обычно это делается с помощью z-index в CSS. При попытке клика на эту ссылку посетитель на самом деле нажимает на кнопку и совершает действие нужное взломщику, например, одним из примеров является Password manager attacks. Этот тип атаки направлен на то, чтобы обмануть менеджеры паролей, чтобы воспользоваться их функцией автозаполнения.

Поэтому избавимся от данной уязвимости путём добавления **HTTP-заголовка X-Frame-Options.**

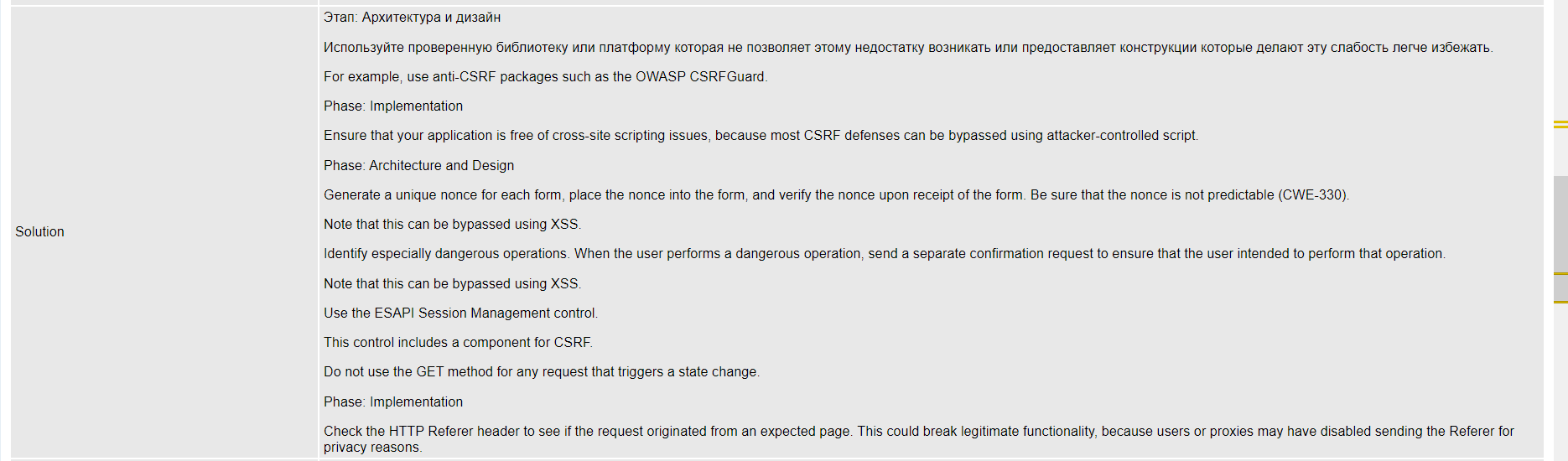


Ещё одна из проблем это CSRF (Cross-site request forgery), то есть подделка межсайтовых запросов.

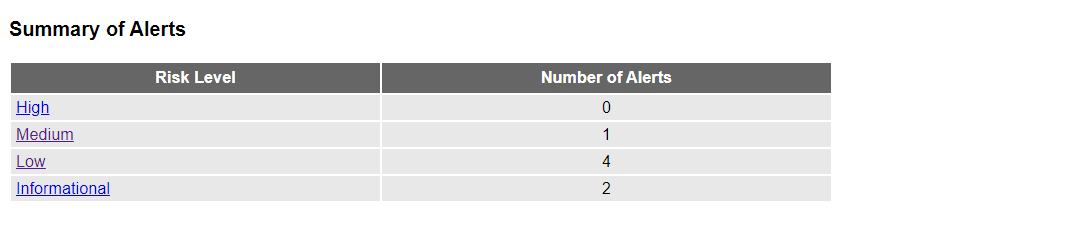


Так как браузер будет отправлять токены аутентификации с каждым новым запросом приходящим на наш сайт. Если жертва заходит на сайт, созданный злоумышленником, от её лица тайно отправляется запрос на другой сервер, осуществляющий некую вредоносную операцию.

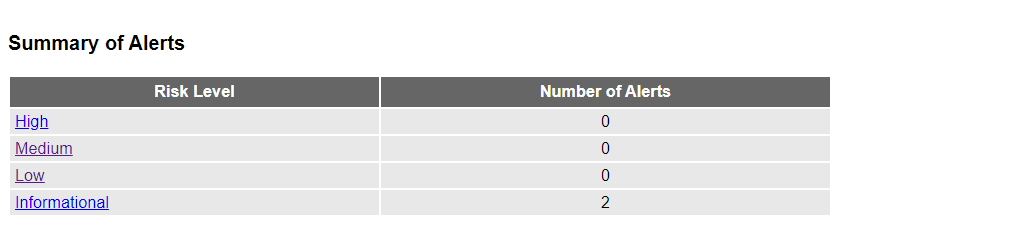
Поэтому данную уязвимость исправим путём добавления CSRF токена на запросы POST, PUT, DELETE к серверу.



Результат до работы над уязвимостями:



Результат после работы над уязвимостями:



Осталась лишь информационные недочёты, которые по своей сущности не влияют на безопасность приложения.