**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение высшего образования «Казанский национальный исследовательский технический университет им. А.Н. Туполева-КАИ»

Институт компьютерных технологий и защиты информации

### КАФЕДРА СИСТЕМ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

**МЕТОДИЧЕСКИЕ УКАЗАНИЯ**

### к выполнению Лабораторной работы № 3

на тему

**«**Типовая методика анализа уязвимостей в веб- приложениях, использующих веб-сервисы**»**

Казань 2021

**ЛАБОРАТОРНАЯ РАБОТА № 3**

**Название работы:**

Типовая методика анализа уязвимостей в веб-приложениях, использующих веб-сервисы

**Цель работы:**

Ознакомиться на практике с методиками анализа уязвимостей в веб-приложениях.

**Теоретический материал:**

**ОБЩИЕ СВЕДЕНИЯ**

1. **Методы обнаружения уязвимостей веб-приложений.**

На сегодняшний день согласно исследованиям OWASP наиболее эффективным способом обнаружения уязвимостей веб-приложений является экспертный анализ исходных кодов веб-приложения (code review). Этот способ весьма трудоемок, требует высокой квалификации эксперта и не защищен от ошибок эксперта. По этому активно развиваются методы автоматического обнаружения уязвимостей веб-приложений. Методы автоматического обнаружения уязвимостей веб-приложений можно разделить на две основные группы:

1) методы, анализирующие работу развернутого на стенде веб-приложения без обращения к исходным кодам веб-приложения;

2) методы, анализирующие исходные коды веб-приложения и конфигурационные настройки.

Первая группа методов рассматривает веб-приложение с точки зрения внешнего пользователя, то есть потенциального злоумышленника. В эту группу входят следующие методы:

Метод получения идентифицирующей информации о веб-приложении.

Метод тестирования на проникновение.

Вторая группа состоит из следующих методов:

Метод статического анализа исходных кодов веб-приложения.

Метод динамического анализа исходных кодов веб-приложения.

Для разработки веб-приложений применяется большое количество разнообразных инструментальных средств, использующих различные языки программирования. Возможности, предоставляемые этими средствами, существенно различаются, ограничивая применение тех или иных методов анализа. В данной работе рассмотрены методы автоматического обнаружения уязвимостей и описаны области применимости и ограничения каждого из методов.

1. **Получение идентифицирующей информации о веб-приложении**

В сети Интернет накоплены огромные объёмы данных по уязвимостям программных продуктов с указанием версий программ, влиянием уязвимостей, методами реализации атак. Бюллетени безопасности Bugtraq и Security Focus ежедневно публикуют 3 отчёты о новых найденных уязвимостях. Эта информация предназначена для администраторов и специалистов в области информационной безопасности. Этот метод получил широкое распространение для проведения атак на операционные системы, вызвав широко известные эпидемии Интернет-червей. При разработке веб-приложений широко используются готовые компоненты, как входящие в состав технологии разработки веб-приложения, так и внешние, например, форумы, доски объявлений и т.п. Получение идентифицирующей информации о веб-приложении основывается на анализе ответов веб-сервера, которые могут содержать идентифицирующую информацию в заголовках HTTP ответов и в текстах HTML-страниц. Для определения типа и версии веб-сервера используется техника fingerprint, которая основана на том, что каждый веб-сервер по-своему обрабатывает HTTP-протокол, в результате чего можно с высокой степенью вероятности определить тип и даже версию веб-сервера путем посылки серверу набора корректных и некорректных запросов и анализа соответствующих ответов. Данный метод получил широкое практическое применение при проведении атак ввиду своей простоты и доступности. Данный метод требует тонкой настройки шаблонов, по которым производится поиск идентифицирующей информации. Эти шаблоны в большинстве случаев специфичны для конкретной технологии разработки веб-приложений.

1. **Тестирование на проникновение**

Метод тестирования на проникновение (penetration testing) рассматривает веб-приложение с точки зрения внешнего пользователя, то есть потенциального злоумышленника. При этом считается, что злоумышленник обладает такими же возможностями, как и обычный пользователь, т.е. не имеет доступа к исходным кодам веб-приложения и не имеет доступа к серверу, на котором работает веб-приложение. Метод предусматривает тестирование работающего на стенде веб-приложения путем посылки запросов, которые эмулируют пользовательскую активность, включающую «корректные» запросы, соответствующие нормальным действиям пользователя и некорректные запросы, соответствующие действиям злоумышленника. При поиске уязвимостей в веб-приложении методом тестирования на проникновение возникают три основные задачи:

1) задача получения и анализа структуры веб-приложения;

2) задача построения набора тестовых HTTP-запросов на основе построенной структуры веб-приложения;

3) задача прогона тестового набора с анализом ответов веб-приложения для выявления уязвимостей.

Задача получения и анализа структуры веб-приложения состоит в том, чтобы построить полный список URI веб-приложения, методов доступа к ним и списков их параметров, выделить URI, защищённые аутентификацией. Задача построения тестового набора запросов к веб-приложению состоит в том, чтобы по исходным данным (список URI приложения, методы доступа к каждому URI и набор параметров к данному URI) подобрать запросы так, чтобы было обнаружено как можно больше уязвимостей. Существующие способы построения таких запросов:

Запрос ресурсов по базе имён подразумевает, что существует база ресурсов, состоящая из имён файлов, которые потенциально могут встретиться в структуре веб-приложения. Наличие в ней того или иного ресурса свидетельствует об уязвимости, связанной с возможным доступом к этому ресурсу. Сканирование происходит по всей иерархии веб-приложения. В каждом каталоге по очереди запрашиваются по имени все ресурсы, представленные в базе данных. Данный способ полностью автоматический и опирается на накопленную информацию о характерных уязвимостях веб-приложений.

Генерация запросов по шаблону с типизированными параметрами заключается в том, что для каждого URI задаётся шаблон, параметры которого типизированы. Далее происходит автоматическая генерация запросов по заданному шаблону со случайным выбором значений конкретных параметров. Значения параметров могут задаваться регулярными выражениями, что позволяет эмулировать атаки XSS и SQL injection на параметрах запроса. Описанным способом также реализуются переборы паролей. При этом данный способ не зависит от технологии, на которой разработано веб-приложение, так как работает только в терминах протокола HTTP.

Анализ настроек каталогов Веб-приложения заключается в том, что по иерархии веб-приложения проверяются настройки веб-сервера, которые могут выявить уязвимости веб-приложения, связанные с неправильным конфигурированием веб-приложения и веб-сервера. Сюда относятся проверка возможности автоматического построения индекса каталога, выполнения HTTP-методов PUT и DELETE, возможность обращения к ресурсам из областей аутентификации напрямую, возможность получения исходных кодов веб-приложения. Задача прогона тестового набора и анализа ответов сервера ставит перед собой цель сделать правильный вывод о том, демонстрирует ли данный HTTP-запрос наличие уязвимости в веб-приложении или нет. Данная задача тесно связана с задачей построения тестового набора, а основная проблема состоит в определении критериев наличия уязвимости. В настоящее время для решения этой задачи применяется метод, в котором распознавание осуществляется 7 регулярными выражениями, задаваемыми экспертом.

Метод тестирования на проникновение существенно меньше зависит от технологии разработки веб-приложения, чем другие методы. Данный метод позволяет накапливать знания эксперта в виде набора правил построения запросов и набора шаблонов для анализа HTTP-ответов. Этот метод получил широкое распространение для грубого выявления уязвимостей разработанных веб-приложений, когда необходимо оценить наличие хотя бы грубых ошибок при разработке и настройке веб-приложения. Достоинством данного метода является то, что метод позволяет оценивать развернутое и настроенное веб-приложение, выявляя не только ошибки кодирования, но и ошибки конфигурирования веб-сервера и веб-приложения.

1. **Статический анализ исходных кодов**

Метод статического анализа исходных кодов веб-приложения не предусматривает реального выполнения веб-приложения. Вместо этого производится построение графов программ (графа управления и графа зависимостей по данным) и обнаружение уязвимостей посредством анализа этих графов. Для обнаружения уязвимостей используется два основных подхода: анализ типов безопасности и анализ потоков данных. При анализе типов безопасности вводится набор типов безопасности (t1, t2, …, tn). Над типами вводится отношение частичного порядка ≤, так моделируется иерархия от безопасного типа к самому опасному. Каждой переменной программы ставится в соответствие её тип безопасности. Существуют два способа определения типов безопасности всех переменных – ручной и автоматический. Ручной способ предполагает, что при каждом объявлении переменной программист должен явно задать её тип безопасности. Автоматический способ предполагает, что даны: разметка переменных, разметка функций, осуществляющих пользовательский ввод правила вывода типов безопасности ещё неразмеченных переменных. Таким образом, при анализе программы последовательно от начала до конца все неразмеченные переменные получают свой тип безопасности автоматически. Следующий пример иллюстрирует данный подход:

Типы безопасности: {untainted, tainted}

Начальная аннотация: int main (int argc, tainted char \*argv[])

Сама программа:

void LogMessage(char \*fmt, …){

… fprintf(fd, fmt, arg);

}

int main (int argc, tainted char \*argv[]){

char \*arg2 = argv[2]; …

char \*Mess = new [strlen(arg2) + strlen(“arg2 = ”) + 1];

strcat(strcpy(Mess, “arg2 =”), arg2);

LogMessage(Mess);

}

Программа имеет уязвимость форматной строки. Методом анализа уровней безопасности данная уязвимость будет обнаружена, если функция fprintf будет аннотирована, как принимающая второй аргумент только untainted уровня безопасности.

Анализ типов безопасности имеет существенный недостаток – постоянную привязку типа безопасности к переменной. В результате тип безопасности определяется без учета того, из какого источника данные попали в переменную, что приводит к большому числу ошибок первого рода. Обнаружение уязвимостей веб-приложений методом статического анализа предусматривает решение следующих основных задач:

Определение конструкций языка и библиотечных функций, которые возвращают данные, потенциально контролируемые злоумышленником. Разработка аксиом распространения метки tainted между переменными приложения.

Определение конструкций языка, которые не должны принимать переменные, содержащие tainted информацию. В этот список должны входить функции для работы с системным окружением, с СУБД, с почтовыми сервисами и т.д.

Метод статического анализа исходных кодов веб-приложения позволяет обнаруживать уязвимости, связанные с неустойчивостью веб-приложения к некорректным входным данным. Другие классы уязвимостей данный метод обнаруживать не позволяет. Метод специфичен для каждой технологии создания веб-приложений и требует от программиста аннотирования функций проверки корректности входных данных.

**5. Динамический анализ**

Метод динамического анализа исходного кода веб-приложения аналогичен методу статического анализа исходных кодов за исключением того, что анализ производится в процессе выполнения веб-приложения. Метод динамического анализа позволяет осуществлять обнаружение уязвимостей для широко применяемых при создании веб-приложений динамических языков, как, например, Perl, PHP, Python, Ruby. Основной проблемой метода динамического анализа является проблема снятия метки tainted с информации. Метод динамического анализа в отличие от статического анализа позволяет осуществлять обнаружение уязвимостей в генерируемом налету коде. В тоже время, метод динамического анализа анализирует лишь один из возможных графов выполнения программы и не позволяет сделать вывод об отсутствии уязвимостей во всей программе. Метод динамического анализа, аналогично статическому анализу специфичен для каждой технологии создания веб-приложений.

**Порядок выполнения работы:**

1. **Изучить теоретический материал.**

**2. Выполнить сравнительный анализ трех методик анализа уязвимостей веб-приложений. Результат анализа представить в виде таблицы:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название методики | Плюсы | Минусы | Источник |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |

**3. Для каждой методики разработать блок схему алгоритма поиска уязвимости. Провести анализ. Выделить идентичные блоки алгоритмов. Выявить «узкие места» методик.**

**Результат представить описательно.**

**4. Разработать проект совершенствования современных методик анализа уязвимостей веб-приложений. Результат представить описательно.**

**5. Ответить на контрольные вопросы.**

Контрольные вопросы:

1. Какие существуют группы методов автоматического обнаружения уязвимостей веб-приложений?

2. Опишите метод получения идентифицирующей информации о веб-приложении.

3. Опишите метод тестирования на проникновение.

4. Опишите метод статического анализа исходных кодов веб-приложения.

5. Опишите метод динамического анализа исходных кодов веб-приложения.

6. В чем заключается общая методика анализа уязвимостей в веб-приложениях?

7. Перечислите преимущества и недостатки метода получения идентифицирующей информации о веб-приложении.

8. Перечислите преимущества и недостатки метода тестирования на проникновение.

9. Перечислите преимущества и недостатки метода статического анализа исходных кодов веб-приложения.

10. Перечислите преимущества и недостатки динамического анализа исходных кодов веб-приложения.