УДК 004.052.42
https://doi.org/10.47533/2020.1606-146X.153

М. Е. БАЙМУРЗИНОВ*, А. А. ДУПИК, А. Т БЕКТЕМЕСОВ

Казахстанско-Британский технический университет, Алматы, Казахстан
Университет «Туран», Алматы, Казахстан
Казахстанский инженерно-технологический университет, Алматы, Казахстан

ИСПОЛЬЗОВАНИЕ МЕТОДА MODEL CHECKING ДЛЯ ВЫЯВЛЕНИЯ УЯЗВИМОСТЕЙ В ВЕБ-ПРИЛОЖЕНИЯХ

На сегодняшний день имеется очень много веб-приложений, которые содержат ошибку в логике работы. Большинство данных уязвимостей могут быть устранены при автоматизированном тестировании, кроме тех, что заключены в логике приложения.

Для выявления данных ошибок логики следует использовать формальные методы проверки. В данном случае в этой работе для проверки и выявление уязвимости в приложении был использован метод автоматической формальной верификации параллельных систем с конечным числом состояний. Для достижения этих задач была построена модель в утилите Spin, произведена верификация и получены результаты. Тем самым была произведена попытка демонстрации одной из уязвимостей в алгоритме веб – приложения.

Ключевые слова*: верификация, проверка алгоритма, веб-приложение, безопасность приложения, логические ошибки, параллельные процессы, утилита Spin, model checking.*

Введение. В эпоху стремительного развития информационных технологий появляются все более новые технологии, методы и инструменты разработки программного обеспечения, утилит, приложения, которые основаны на быстрой и эффективной разработке, но в этом случае может пострадать корректность работы приложения. Неправильная логика работы и ошибки могут повлиять на безопасность, тем самым создав уязвимость в приложении. Поэтому тестирование и проверка приложений на наличие ошибок является важным этапом перед выпуском программного обеспечения. Но автоматизированное тестирование не гарантирует, что все ошибки будут выявлены, а ручной подход поиска ошибок может занять большое количество времени и ресурсов. В данном случае нам может помочь метод верификаций под названием Model Checking. Метод Model Checking позволяет нам построить модель программы и проверить ее на выполнение той или иной логики. Одним из преимуществ данного метода является то, что он может быть полностью автоматизирован [1]. Рассмотрим данный метод относительно веб-приложений, поскольку на сегодняшний день веб-приложения являются одними из привлекательнейших целей для кибератак. По данным Symantec, более 200 атак совершаются каждый день на веб-сайты, и часть этих сайтов имеют ряд незалатанных уязвимостей [2]. Уязвимости в веб-приложениях, кроме логики приложения, могут касаться серверной и клиентской части. Если рассматривать литературу по выявлению и предотвращению уязвимостей, можно рассмотреть следующих авторов: Wang [3] рассмотрел различные виды атак на веб-приложения, такие как Cross Site Request Forgery, PHP Injection. И предлагает методы поиска уязвимостей веб-приложений с помощью использования сканирования угроз.

* E-mail корреспондирующего автора: muhambet20111@gmail.com

Следующий автор – Divyaniyadav Gupta – предлагает обеспечить безопасность коммерческих веб-приложений и базы данных от SQL+ инъекций и Cross Site Scripting путем шифрования, контроля доступа и ошибок [4]. Если рассматривать защиту локального хранилища браузера, то автор предлагает использовать аналитику кода и валидацию входных данных, полученных из IndexedDB, для защиты от межсайтового скриптинга [5].

Кроме этого, автором Takamatsu предлагаются различные методы защиты от захвата клика, один из них – использование заголовков X-Frame-Options со стороны сервера, который может разрешать или запрещать отображение страницы внутри фрейма [6].

Однако у данных авторов не затрагиваются проблемы и методы поиска ошибок и уязвимостей относительно логики и алгоритма веб-приложения. Для решения проблем в логике приложения можно рассмотреть работы в области верификаций. Одной из таких работ являются исследования и применение метода Model Checking в работе Карпова Ю.Г. [7]. В данной работе можно заметить доказательства некорректности через контрпримеры в логике многих алгоритмах. Кроме того, автор использует контрпримеры для нахождения искомого пути в решении задач бизнес-процессов, логических задач и задач криптографий. Но автором не затрагивается использование данного метода в области веб - приложений. Из этого можно сделать вывод, что уделяется мало внимания именно верификации веб- приложений. Таким образом, нам необходимо проверить, выполняется ли на нашей построенной формальной модели веб-приложения заданное условие, а затем проанализировать вывод программы на наличие ошибок, некорректностей, приводящих к уязвимости.

Метод исследования. Нашей задачей в этом исследовании является нахождение некорректной работы веб-приложения, связанное с логикой. Для нахождения таких ошибок и некорректностей используются различные подходы. Проанализировав различные методы и подходы, можно выделить несколько. Первым делом рассмотрим вариант тестирования на безопасность, этот вид тестирования позволяет выявить уязвимости от различных информационных угроз, приводящих к изменению логики приложения, но данный вид тестирования подходит только для выявления уязвимостей уже по известному списку угроз. Следующий вид тестирования – это функциональное тестирование, для которого могут использоваться инструменты ручного и автоматизированного тестирования. Данный вид тестирования позволяет имитировать фактическое использование системы, выявить ошибки и сравнить фактический и расчетный ожидаемый результат. Хотя в данном случае этот подход также не походит, поскольку он рассматривает внешнее поведение системы. Тем самым упускается возможность выявления логических ошибок. Также можно рассмотреть статический анализ, который позволяет выявить очень много различных дефектов. Однако этот вид тестирования слаб в нахождении ошибок в параллельных программных системах или продуктах, и при этом выдается большое количество ложноположительных срабатываний. Конечным итогом был выбран метод верификаций model checking, который позволит нам проверить формальную модель работы приложения.

Преимуществом этого метода перед вышеизложенными методами и подходами является то, что данный метод ориентирован на нахождение ошибок и несоответ-

ствий в логике приложений, а также отлично походит для проверки параллельных систем.

Для проведения исследования был выбран инструмент верификации SPIN, это утилита использует язык мета процессов, что позволяет автоматизировать процесс проверки.

Суть исследования заключается в том, что у нас имеется веб-приложение, а именно Интернет - магазин для продажи определённого товара. Поскольку это веб-приложение, посетители получают доступ к приложению через запрос к веб-серверу. Кроме продажи товаров, в этом Интернет - магазине имеется возможность активации купона на снижение стоимости товара.

Рассмотрим шаги алгоритма работы.

Покупатель заходит на сайт

Вводит код купона

Идет запрос на веб-сервер

Проверка баланса купона

Если проверка пройдена, с текущей цены товара вычитается значение купона

Купон обнуляется

Мы будем проверять данный алгоритм работы на поиск недочетов путем верификаций методом Model Checking. Построим нашу формальную модель в Spin для верификаций.

Формальный модель на языке Promela:

```
int product_cost = 600 // стоимость товара
int coupon_balance = 100; // купон

proctype request_1(){
if ::(coupon_balance == 100) ->
                product_cost = product_cost - coupon_balance;
                coupon_balance == coupon_balance - 100;
     ::else -> skip;
fi }
proctype request_2(){
if ::(coupon_balance == 100) ->
                product_cost = product_cost - coupon_balance;
                coupon_balance == coupon_balance - 100;
     ::else -> skip;
fi }
proctype request_3() {
if ::(coupon_balance == 100) ->
                product_cost = product_cost - coupon_balance;
                coupon_balance == coupon_balance - 100;
     ::else -> skip;
fi }
```

```
init {
run request_1();
run request_2();
run request_3();
}
```

После построения модели, необходимо задать с помощью операторов LTL следующие условия: среди всех возможных сценарий не существует такого пути, который не снизится стоимость товара на 300 при балансе 100. Если такой сценарий возможен, то Spin выдаст контрпример, который и является искомым способом снизить стоимость.

Результат. После верификаций выдан результат, что существует такой путь выполнения, по которому стоимость товара снизится в 3 раза, используя лишь один купон. Проанализировав выданный отчет программы, можно сделать вывод, что уязвимость заключается в некорректной обработке параллельных запросов. Разберем некорректность алгоритма более подробно.

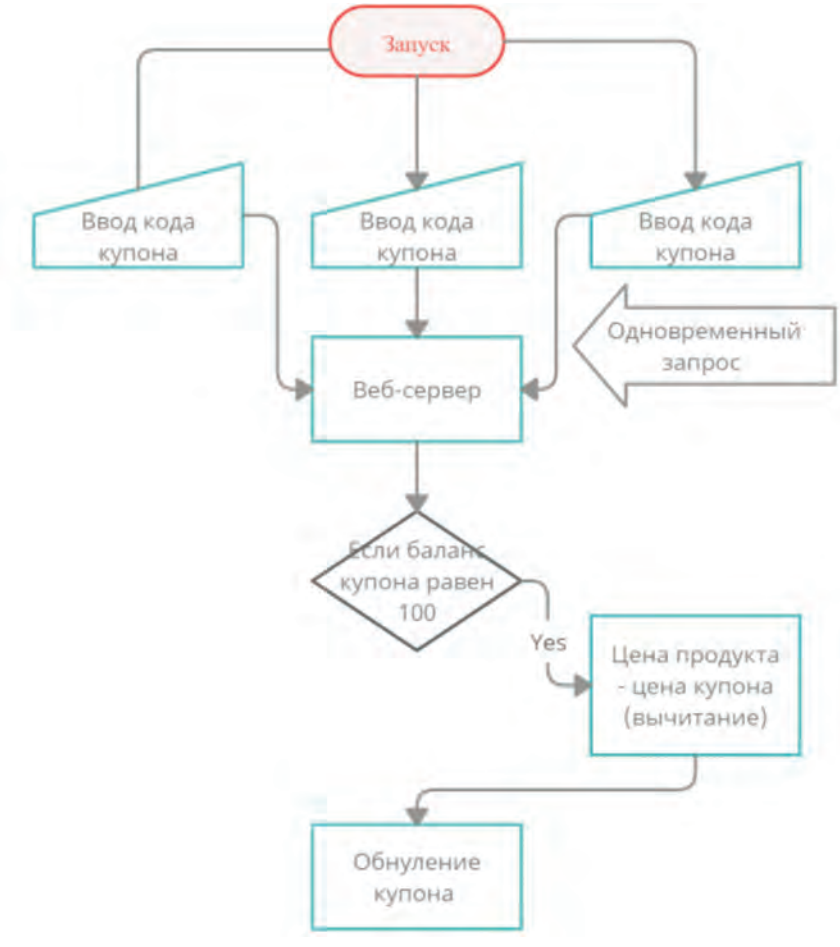

Рисунок 1 – Блок-схема работы алгоритма

Представим, что злоумышленник зашел на сайт Интернет - магазина, который обслуживается сервером Dell PowerEdge R330 с установленном на нем одной из самых используемых сети веб-серверов Apache HTTP-сервер. Веб-сервер обрабатывает запросы последовательно, а также может иметь расширение многопроцессорных модулей (MPM), которые позволяют создавать дочерние потоки для обработки нескольких запросов.

Но при этом злоумышленник, не имея информаций о веб-сервере, может успешно эксплуатировать уязвимость путем некоторых попыток. Для этого он использует программное обеспечение, которое позволит создать несколько запросов одновременно, в нашем случае – три запроса на активацию одного и того же купона, поскольку из-за специфики протоколов верхнего уровня запросы не всегда последовательны, и временной диапазон обработки может занимать несколько миллисекунд, а наличие многопроцессорного модуля позволяет выполнять многопотоковые запросы.

Таким образом, эти запросы могут обрабатываться серверной стороной почти параллельно с разницей в миллисекунды. В таком случае все три запроса могут выполнить условия проверки одновременно. Следовательно, в это время баланс в трех запросах будет равен 100, что позволит выполниться условию во всех трех запросах в соответствие с рисунком №1. Как следствие, купон снизит стоимость товара в три раза, что не было предусмотрено изначально в алгоритме. Исходя из вышеизложенного, можно сделать вывод, что метод Model Checking позволил нам найти непредвиденный сценарий работы алгоритма, который может привести к эксплуатации злоумышленником.

Для решения вышеприведенной проблемы может быть использовано свойство под названием «взаимное исключение», суть которой заключается в том, что доступ к объекту может быть только у одного запроса. Обеспечить такой механизм защиты возможно при помощи использования возможностей серверного языка программирования.

Заключение. Подводя итоги исследования и анализа, можно отметить то, что существуют различные методы и подходы проверки программного обеспечения и приложений. Каждые из них имеют свои преимущества и недостатки. В нашем случае мы рассмотрели метод верификаций Model Checking, который позволил нам выявить недостаток в логике работы веб-приложения. Кроме этого, был предложен вариант решения данной проблемы. В дальнейшем метод Model Checking хорошо применим для поиска ошибок в различных модулях веб-серверов и клиентов, поскольку сфера веб расширяется и дополняется все более новыми технологиями, использующими многопоточность.

ЛИТЕРАТУРА

1 Э. М. Кларк, О. Грамберг, Д. Пелед Верификация моделей программ: Model Checking, Издательство: Московский центр непрерывного математического образования. 2002 г.

2 Huang, Hsiu Chuan Zhang, Zhi Kai Cheng, Hao Wen Shieh, Shiuhpyng Winston, “Web Application Security: Threats, Countermeasures, and Pitfalls”, Computer, vol. 50, October 2017.

3 Wang, Bin Liu, Lu Li, Feng Zhang, Jianye Chen, Tao Zou, Zhenwan, "Research on Web Application Security Vulnerability Scanning Technology", Proceedings of 2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference, IAEAC 2019, July 2019.

4 Divyaniyadav Gupta, Deeksha Singh, Dhananjay Kumar, Devendra Sharma, Upasana, "Vulnerabilities and security of web applications", 2018 4th International Conference on Computing Communication and Automation, ICCCA 2018, July 2018.

5 Kimak, Stefan Ellman, Jeremy Laing, Christopher, "An Investigation into Possible Attacks on HTML5 Index

Telecommunications, Networking and Broadcasting, Liverpool, UK, July 2012.

6 Takamatsu, Yusuke Kono, Kenji, "Clickjuggler: Checking for incomplete defenses against clickjacking", 2014 12th Annual Conference on Privacy, Security and Trust, PST 2014, July 2014.

7 Юрий Карпов: Model Checking. Верификация параллельных и распределенных программных. Издательство: BHV, 2010 г.

REFERENCES

1 E. M. Klark, O. Gramberg, D. Peled Verifikaciya modelej programm: Model Checking, Izdatel'stvo: Moskovskij centr nepreryvnogo matematicheskogo obrazovaniya. 2002 g.

2 Huang, Hsiu Chuan Zhang, Zhi Kai Cheng, Hao Wen Shieh, Shiuhpyng Winston, "Web Application Security: Threats, Countermeasures, and Pitfalls", Computer, vol. 50, October 2017.

3 Wang, Bin Liu, Lu Li, Feng Zhang, Jianye Chen, Tao Zou, Zhenwan, "Research on Web Application Security Vulnerability Scanning Technology", Proceedings of 2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference, IAEAC 2019, July 2019.

4 Divyaniyadav Gupta, Deeksha Singh, Dhananjay Kumar, Devendra Sharma, Upasana, "Vulnerabilities and security of web applications", 2018 4th International Conference on Computing Communication and Automation, ICCCA 2018, July 2018.

5 Kimak, Stefan Ellman, Jeremy Laing, Christopher, "An Investigation into Possible Attacks on HTML5 Index

Telecommunications, Networking and Broadcasting, Liverpool, UK, July 2012.

6 Takamatsu, Yusuke Kono, Kenji, "Clickjuggler: Checking for incomplete defenses against clickjacking", 2014 12th Annual Conference on Privacy, Security and Trust, PST 2014, July 2014.

7 YUrij Karpov: Model Shecking. Verifikaciya parallel'nyh i raspredelennyh programmnyh Izdatel'stvo: BHV, 2010 g.

М. Е. БАЙМУРЗИНОВ, А.А. ДУПИК, А.Т БЕКТЕМЕСОВ

Қазақстан-Британ техникалық университеті, Алматы, Қазақстан
Тұран университеті, Алматы, Қазақстан
Қазақстан инженерлік-технологиялық университеті, Алматы, Қазақстан

ВЕБ-ҚОСЫМШАЛАРДАҒЫ ОСАЛДЫҚТАРДЫ АНЫҚТАУ ҮШІН MODEL CHECKING ӘДІСІН ҚОЛДАНУ

Бүгінгі таңда жұмыс логикасында қатесімен көптеген веб-қосымшалар бар. Бұл осалдықтардың көпшілігін қосымшаның логикасында көрсетілгеннен басқа автоматтандырылған тестілеу арқылы алдын алуға болады.

Логикалық қателерді анықтау үшін формальды тексеру әдістерін қолдану керек. Осы жұмыста осалдықты тексеріп анықтау үшін соңғы күйлері бар параллель жүйелерді автоматты формальды түрде тексеру әдісі қолданылды. Осы мақсаттарға қол жеткізу үшін Spin утилитасында модель құрылды, верификация жүргізілді және нәтижелер алынды. Осылайша, веб-қосымшаның алгоритміндегі осалдықтардың бірін көрсетуге әрекет жасалды.

***Түйін сөздер**: верификация, алгоритмді тексеру, веб қосымшасы, қосымшаның қауіпсіздігі, логикалық қателер, параллель процестер, Spin утилитасы, Model Checking.*

M.E. BAIMURZINOV, A.A. DUPIK, A.T. BEKTEMESOV

Kazakh-British Technical University, Almaty, Kazakhstan,
Turan University, Almaty, Kazakhstan,
Kazakhstan Engineering Technological University, Almaty, Kazakhstan

USING THE MODEL CHECKING METHOD TO IDENTIFY VULNERABILITIES IN WEB APPLICATIONS

To date, there are a lot of web applications that contain an error in the logic of work. Most of these vulnerabilities can be eliminated with automated testing, except for those that are enclosed in the application logic.

Formal verification methods should be used to identify these logic errors. In this article, was used the method of automatic formal verification of parallel systems with a finite number of states to check and identify vulnerabilities in the application. To accomplish these tasks, a model was built in the Spin utility, verification was performed and results were obtained. Thus, an attempt was made to demonstrate one's of the vulnerabilities in the algorithm of the web application.

***Keywords**: verification, algorithm check, web application, application security, logical errors, parallel processes, Spin utility, Model Checking.*