УДК 004.416.6 DOI 10.17223/2226308X/9/35

ОБ ИДЕНТИФИКАЦИИ ЗАЩИТНЫХ ЭКРАНОВ ВЕБ-ПРИЛОЖЕНИЙ В МОДЕЛИ MitB

Д. Н. Колегов, П. А. Линейцев

Рассматриваются существующие способы идентификации защитных экранов веб-приложений и их программная реализация в рамках модели нарушителя «Man in the Browser».

Ключевые слова: *безопасность приложений, защитные экраны веб-приложений, идентификация.*

Защитные экраны уровня приложения (Web Application Firewalls, WAF) применяются для обнаружения и предотвращения атак на веб-приложения. Применение защитных экранов не гарантирует безопасность веб-приложения, так как они сами могут содержать уязвимости реализации и конфигурирования. Одной из актуальных задач анализа защищённости веб-приложений является разработка программных средств идентификации защитных экранов на клиентской стороне веб-приложений в рамках модели «Man in the Browser» (MitB) [1]. В данной работе исследованы возможности идентификации экранов приложений в рамках модели MitB и реализован программный модуль для фрэймворка BeEF [2], реализующий данную функциональность.

Идентификация защитных экранов, как правило, реализуется на основе сигнатурного анализа следующих веб-сущностей: идентификаторы сессий, HTTP-заголовки, cookie, коды ответов и т. д. Известными средствами идентификации защитных экранов являются WAFW00F [3] и sqlmap [4]. Приведём пример модуля, идентифицирующего экран «Wallarm» в sqlmap на основе сигнатуры HTTP-заголовка «Server»:

```
retval = False
for vector in WAF_ATTACK_VECTORS:
    _, headers, _ = get_page(get=vector)
    retval = re.search(r"nginx-wallarm",
                       headers.get(HTTP_HEADER.SERVER, ""),
                       re.I) is not None
    if retval:
        break
```

Вместе с тем данные средства ограничены в применении и не могут быть использованы в модели MitB, являющейся основной с точки зрения анализа клиентской части веб-приложения. Для обеспечения возможности идентификации защитных экранов в модели MitB реализован программный модуль для фрэймворка BeEF (Browser Explotation Framework) с использованием известных баз сигнатур.

Разработанный модуль функционирует следующим образом: он отправляет стандартный HTTP-запрос, затем анализирует ответ, полученный от веб-приложения. Если это не дало никаких результатов в части идентификации экрана, то отправляется серия HTTP-запросов с различными векторами атак, что приводит к реакции защитного экрана при его наличии. Экран может добавить специальный заголовок с сигнатурой или выставить идентификатор сессии. Если защитный экран не удалось идентифицировать с помощью HTTP-запросов, то анализируются коды ответов веб-приложения.

Для проведения сигнатурного анализа необходима коллекция сигнатур защитных экранов. Для её составления использовались базы сигнатур средств WAFW00F [3] и

sqlmap [4], некоторые сигнатуры добавлены авторами. Сигнатуры хранятся в формате JSON. Пример сигнатуры рализованного модуля BeEF:

```
1 {
2   "name":"F5 BIG-IP ASM",
3   "cookie": ["TS[a-zA-Z0-9]{3,8}"],
4   "headers": []
5 }
```

ЛИТЕРАТУРА

1. *Alkorn W., Frichot C., and Orru M.* The Browser Hacker's Handbook. Indianapolis, John & Wiley Sons, 2014. 648 p.
2. The Browser Exploitation Framework Project. `http://beefproject.com/`
3. The WAFW00F project. `https://github.com/EnableSecurity/wafw00f`
4. The sqlmap project. `http://sqlmap.org/`

УДК 004.94 DOI 10.17223/2226308X/9/36

ЛЕГКОВЕСНАЯ РЕАЛИЗАЦИЯ МЕХАНИЗМА АТРИБУТНОГО УПРАВЛЕНИЯ ДОСТУПОМ ДЛЯ СУБД НА УРОВНЕ ЗАЩИТНОГО ЭКРАНА

Д. Н. Колегов, Н. О. Ткаченко

Рассматривается легковесная реализация механизмов атрибутного управления доступом для систем управления базами данных на уровне защитных экранов приложений (*Web Application Firewall*, *Database Firewall*), функционирующих в режиме прокси-сервера. Предлагается механизм проекции ролей для добавления элементов ролевого управления доступом.

Ключевые слова: *управление доступом, ABAC, RBAC, защитный экран, СУБД.*

В настоящее время политики безопасности для приложений являются, как правило, ролевыми (*RBAC*), атрибутными (*ABAC*) или гибридными, сочетающими в себе как ролевое, так и атрибутное управление доступом. Для каждого управления доступом в отдельности имеются или создаются детально проработанные стандарты [1, 2], но они ориентированы на реализацию в конечных системах, а не в защитных экранах приложений, а потому не могут быть использованы без существенной адаптации как к условиям функционирования защитных экранов в режиме прокси-сервера, так и к самим защищаемым системам управления базами данных (СУБД).

Задача ставится следующим образом. Имеется СУБД и заданная политика безопасности, которая не может быть реализована встроенными механизмами управления доступом этой СУБД. Необходимо реализовать эту политику безопасности без изменения конфигурации, исходного кода и данных на СУБД. Такая реализация называется неинвазивной и была предложена авторами в работах [3, 4]. Для решения поставленной задачи строится легковесный механизм атрибутно-ролевого управления доступом — *ABAC-lite*.

Реализуемый механизм состоит из следующих структурных элементов в терминологии *NIST ABAC* [1]. *Policy Enforcement Point* (*PEP*) отвечает за получение и парсинг *SQL*-запросов от клиента к серверу СУБД и формирование запросов средствами