Данный метод обеспечивает целостность значений cookie, требует меньшего количества служебных cookie, но накладывает следующие существенные ограничения на веб-приложение:

— невозможность выставления cookie c с атрибутами path и domain, отличными от пути и домена HTTP-запроса, в котором выставляется cookie c;
— невозможность пересчёта значения $hmac$ по истечении времени жизни или при удалении cookie;
— инициализация новой сессии при любом некорректном запросе.

Реализация данных методов не требует изменения исходного кода веб-приложения и может быть выполнена на уровне гибридных WAF (например, ModSecurity), модульных фрэймворков (например, Django, Ruby on Rails) и сетевых WAF (например, F5 BIG-IP ASM). Прототип системы, реализующей эти методы, разработан на базе Django Middleware [8].

ЛИТЕРАТУРА

1. *Barnett R.* The Web Application Defender's Handbook: Battling Hackers and Protecting Users. Indianapolis: John Wiley & Sons, 2013. 522 p.
2. Reducing Web Application Attack Surface. `http://blog.spiderlabs.com/2012/07/reducing-web-apps-attack-surface.html`
3. ModSecurity Advanced Topic of the Week: HMAC Token Protection. `http://blog.spiderlabs.com/2014/01/modsecurity-advanced-topic-of-the-week-hmac-token-protection.html`
4. *Колегов Д. Н.* Общий метод аутентификации HTTP-сообщений в веб-приложениях на основе хеш-функций // Прикладная дискретная математика. Приложение. 2014. № 7. С. 85–89.
5. *Fu K., Sit E., Smith K., and Feamster N.* Dos and Don'ts of client authentication on the Web // Proc. 10th USENIX Security Symp., Washington, 2001. P. 251–268.
6. *Liu A., Kovacs J., Huang C., and Gouda M.* A secure cookie protocol // Proc. 14th Intern. Conf. Computer Communications and Networks, 2005. P. 333–338.
7. *Murdoch S.* Hardened Stateless Session Cookies. `http://www.cl.cam.ac.uk/~sjm217/papers/protocols08cookies.pdf`
8. Прототип модуля неинвазивного контроля целостности cookie на базе Django. `https://github.com/tsu-iscd/django-HTTPauth`

УДК 004.94 DOI 10.17223/2226308X/8/33

НЕИНВАЗИВНАЯ РЕАЛИЗАЦИЯ МАНДАТНОГО УПРАВЛЕНИЯ ДОСТУПОМ В ВЕБ-ПРИЛОЖЕНИЯХ НА УРОВНЕ СУБД

Д. Н. Колегов, Н. О. Ткаченко

Предлагается неинвазивный метод (метод, не изменяющий исходный код самого приложения) устранения уязвимостей в механизмах логического управления доступом и информационными потоками в веб-приложениях на уровне СУБД. Задача ставится следующим образом: имеется многоуровневое веб-приложение, в котором реализована подсистема базового управление доступом (как правило, ролевого), возможно, имеющая некоторое множество уязвимостей. Необходимо устранить как можно более широкий класс данных уязвимостей без изменения исходного кода самого приложения или обеспечить реализацию новой политики мандатного управления доступом. Метод включает выполнение следующих

шагов: идентификация субъектов веб-приложения с помощью специального интегрируемого модуля и передачи полученных идентификаторов в SQL-запросах; обработка SQL-запросов с учётом формальной политики мандатного управления доступом на уровне прокси-сервера; переписывание SQL-запросов для предотвращения несанкционированного доступа к данным.

Ключевые слова: *управление доступом, веб-приложения, безопасность СУБД.*

Управление доступом в системах управления базами данных (СУБД) в большинстве случаев реализуется на уровне ядра системы. Однако такой подход имеет ряд недостатков. Во-первых, в подавляющем большинстве случаев реализуется управление доступом на основе дискреционной политики. Исключением являются специализированные защищённые СУБД, такие, как RUBIX [1], Линтер [2] или расширения для СУБД общего назначения, например пакет Oracle Label Security для СУБД Oracle [3], где реализуется политика мандатного управления доступом типа MLS. Во-вторых, в реализованных механизмах управления доступом, как и в любом другом программном обеспечении, могут содержаться ошибки. К примеру, сравнительно недавно в популярной СУБД MySQL была найдена уязвимость, позволяющая обойти процедуру аутентификации [4]. При этом ошибки могут быть не только в реализации системы, но и в её модели безопасности [5], особенно если речь идёт о достаточо сложном мандатном или ролевом управлении доступом.

В то же время с точки зрения управления доступом СУБД, как правило, является отдельной информационной системой и функционирует независимо от приложения пользователя, которое её использует. В качестве примера можно рассмотреть типовое веб-приложение, которое содержит множество пользователей на уровне веб-сервера, но все они выполняют операции в СУБД от имени одной её учётной записи, соответствующей этому веб-приложению. Так как все операции на уровне СУБД выполняются от имени одной учётной записи, можно считать, что управление доступом на уровне СУБД отсутствует и в случае наличия ошибок в реализации управления доступом в коде веб-приложения злоумышленник может получить доступ к данным любого пользователя приложения, хранящимся в СУБД.

В работе предлагается неинвазивный метод реализации управления доступом в веб-приложении на уровне СУБД MySQL, основанный на формальных моделях безопасности для СУБД MySQL [6, 7] и реализации монитора безопасности на уровне прокси-сервера для SQL-запросов [8]. Метод позволяет получить информацию об идентификаторах пользователей веб-приложения и передать их (прозрачно для веб-приложения) в SQL-запросах, а затем на уровне SQL-прокси реализовать заданную политику управления доступом. Процедура идентификации субъектов осуществляется с помощью технологии тэгирования — добавления к SQL-запросу идентификатора пользователя в форме комментария. После того как мы идентифицировали субъектов веб-приложения, необходимо идентифицировать сущности СУБД. В СУБД MySQL минимальной единицей доступа являются столбцы, поэтому стандартными средствами возможно реализовать управление доступом лишь до уровня столбцов, что недостаточно для современных защищённых веб-приложений. Для обеспечения возможности идентификации строк таблиц СУБД предложен метод определения принадлежности записи из защищаемой таблицы СУБД пользователю веб-приложения.

Рассмотрим последний метод более детально. В некоторых СУБД, в том числе в СУБД MySQL и в порождённых от нее, например, в СУБД MariaDB, управление доступом не позволяет задать право доступа к строке таблицы. Наименьшим контей-

нером, к которому право доступа может быть задано, выступает столбец. Типовые веб-приложение часто используют таблицу для хранения однотипных записей всех пользователей. В этом случае отсутствие проверок допустимых значений входных данных пользователяей веб-приложения, влияющих на генерируемый SQL-запрос, может привести к несанкционированному доступу к данным. Предлагаемый метод включает анализ генерируемого SQL-запроса и добавление в него дополнительных условий. Анализ запроса состоит в проверке наличия защищаемой таблицы в списке таблиц, к которым осуществляется доступ. *Защищаемой таблицей* называется таблица, для столбца которой активирован описываемый механизм. Если такая таблица присутствует в запросе, то он модифицируется путём добавления условия в раздел WHERE или HAVING исходного SQL-запроса. Добавляемое условие формируется в момент анализа запроса из шаблона, описанного администратором в конфигурационном файле. Шаблон представляет собой кортеж из трёх объектов:

— полного пути до защищаемого столбца в формате «БАЗА_ДАННЫХ.ТАБЛИЦА.СТОЛБЕЦ»;
— регулярного выражения, поддерживаемого СУБД MySQL и соответствующего уникальному значению идентификатора пользователя;
— пути, связывающего столбец с идентификаторами пользователей и столбец из защищаемой таблицы, если такой может быть построен.

Второй и третий объекты решают задачу определения принадлежности записи конкретному пользователю. При этом возможны следующие варианты. Если защищаемая таблица содержит столбец с идентификаторами пользователей, то третий объект не обязателен для заполнения и может быть построено регулярное выражение с использованием предопределённых параметров, таких, как идентификатор пользователя, определяющий принадлежность записи. Если же защищаемая таблица содержит столбец, по которому возможно определить принадлежность записи пользователю через связи с другими таблицами, то третий объект может быть использован для описания этих связей. Например, если в защищаемой таблице хранятся зарплатные ведомости и первичным ключом является инкрементируемое числовое значение, то для связи владельца заработной платы и её значения может быть построен путь от первичного ключа защищаемой таблицы через промежуточную таблицу, реализующую связь многие ко многим, до столбца с идентификаторами пользователей.

Основой прототипа является MySQL-proxy, реализующий политику мандатного управления доступом типа MLS и TE на основе разработанной ранее формальной ДП-модели [7]. Механизмы, позволяющие определить принадлежность записи из защищаемой таблицы СУБД пользователю веб-приложения, и идентификация пользователей веб-приложения реализованы в виде модулей веб-фреймворка Django. Общая схема полученного прототипа изображена на рис. 1.

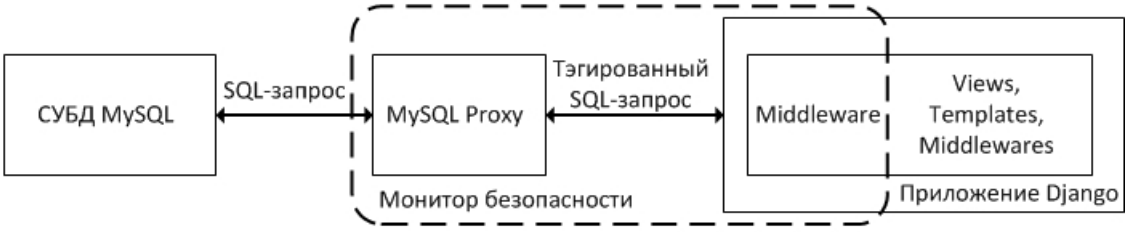

Рис. 1. Общая схема прототипа системы

Рассмотрим процесс прохождения запроса пользователя веб-приложения к СУБД. Аутентифицированный пользователь веб-приложения отправляет HTTP-запрос на выполнение какого-либо действия. Это приводит к генерации SQL-запроса к СУБД MySQL. Перед отправкой запрос тэгируется идентификатором пользователя, инициировавшим его выполнение, с использованием модуля Django фреймворка. Основная функция модуля состоит в модификации некоторых методов класса-обёртки CursorWrapper, который используется для перехвата некоторых исключений класса Cursor. В результате модификации SQL-запросы пользователя, генерируемые слоем Object-Relational Mapping фреймворка Django или написанные пользователем вручную, будут перехвачены и обработаны. Далее запрос, содержащий идентификатор пользователя, поступает на MySQL-proxy. Если механизм контроля записей сконфигурирован, то осуществляется анализ SQL-запроса. В результате анализа определяется возможность получения пользователем записей, ему не принадлежащих; если это возможно, к запросу добавляется дополнительное условие, гарантирующее получение пользователем только его записей. Затем запрос отправляется на обработку в СУБД MySQL.

ЛИТЕРАТУРА

1. Trusted DBMS Rubix. `http://rubix.com/cms`
2. СУБД Линтер. `http://linter.ru`
3. Oracle Database. Oracle Label Security. `http://www.oracle.com/technetwork/database/options/label-security/index.html`
4. CVE-2012-2122. `https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2012-2122`
5. *Девянин П. Н., Захаренков П. С.* Способ реализации информационного потока по времени в операционных системах с мандатным управлением доступом через clipboard // Методы и технические средства обеспечения безопасности информации: материалы Юбилейной 20-й науч.-технич. конф. 27 июня–01 июля 2011 г. СПб.: Изд-во Политехн. ун-та, 2011. С. 76–77.
6. *Колегов Д. Н., Ткаченко Н. О., Чернов Д. В.* Разработка и реализация мандатных механизмов управления доступом в СУБД MySQL // Прикладная дискретная математика. Приложение. 2013. № 6. С. 62–67.
7. *Колегов Д. Н., Ткаченко Н. О., Чернов Д. В.* Основные элементы разработки механизма мандатного управления доступом в СУБД MySQL на основе ДП-моделей // Безопасность информационных техологий. 2014. № 3. С. 102–107.
8. *Ткаченко Н. О.* Реализация монитора безопасности СУБД MySQL в DBF/DAM-системах // Прикладная дискретная математика. Приложение. 2014. № 7. С. 99–101.

УДК 004.056.5 DOI 10.17223/2226308X/8/34

РЕАЛИЗАЦИЯ АТАКИ DNS REBINDING

Т. И. Милованов

Исследована акутальность атаки DNS Rebinding в современных браузерах. Атака направлена на обход концепции одинакового источника (Same Origin Policy). Цель работы — исследование применимости атаки для доступа к узлам локальной сети пользователя. Составлен список браузеров, наиболее подверженных атаке. В инструмент для тестов на проникновение BeEF встроено расширение, позволяющее реализовать атаку на практике. Сформулированы условия, при которых данная атака успешно реализуется, и рекомендации по защите.

Ключевые слова: *HTTP, pentesting, веб-приложения.*