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.*

Существует важная концепция веб-безопасности, которой руководствуются современные браузеры при исполнении сценариев — концепция одинакового источника (Same Origin Policy) [1]. Сценарий — это программа, исполняемая на стороне пользователя. Пользователь получает сценарий при обращении к некоторому веб-серверу, который будем называть источником. Обычно сценарии написаны на языке JavaScript и предназначены для оформления веб-страниц или выполнения на стороне пользователя обработки перед отправкой данных на сервер. Концепция разрешает сценарию доступ к данным источника, если и только если сценарий получен с этого источника. Источник характеризуется тремя признаками: доменным именем, портом, протоколом. Два источника считаются одинаковыми, если у них совпадают все три признака.

При запросе к веб-серверу браузер пользуется системой DNS, преобразующей доменное имя веб-сервера в его сетевой адрес (IP-адрес). DNS реализуется распределённой системой серверов, выполняющих данное преобразование. Владелец веб-сервера может иметь собственный DNS-сервер, отвечающий за доменное имя веб-сервера.

Основная идея атаки DNS Rebinding в том, что в концепции одинакового источника в характеристики источника не входит сетевой адрес. Злоумышленник может выполнить свой сценарий с данными другого источника, будем называть его целевым, если доменному имени веб-сервера злоумышленника будет соответствовать не один, а два сетевых адреса — адрес веб-сервера злоумышленника и адрес целевого веб-сервера. Заметим, что система DNS позволяет такое неоднозначное соответствие. В данной работе рассматривается возможность доступа к целевым веб-серверам, находящимся в локальной сети пользователя. Таковыми могут быть, например, роутеры, практически все из которых имеют веб-интерфейс. Приведём общую схему атаки.

1) Пользователь обращается к веб-серверу злоумышленника с помощью доменного имени.
2) Браузер пользователя формирует запрос к DNS-серверу злоумышленника с целью получить сетевой адрес, соответствующий доменному имени. DNS-сервер отвечает парой сетевых адресов.
3) Браузер пользователя обращается по первому сетевому адресу, который считает основным, и получает веб-страницу злоумышленника вместе со сценарием.
4) Злоумышленник блокирует дальнейшие обращения пользователя к своему веб-серверу (например, с помощью межсетевого экрана).
5) Сценарий в браузере пользователя инициирует повторное обращение к веб-серверу злоумышленника с помощью доменного имени, но вследствие блокировки веб-сервера не получает ответа по первому сетевому адресу и обращается по второму, который является адресом целевого веб-сервера в локальной сети пользователя. Браузер пользователя разрешает это обращение, поскольку не изменился ни один из трёх признаков источника.

В современных браузерах существует два типа реакций на полученный от DNS-сервера ответ, содержащий локальный IP-адрес. Некоторые браузеры считают его основным, некоторые — второстепенным. Первый тип реакции более безопасен, он предотвращает приведённую выше схему атаки, поскольку исключает получение сценария с веб-сервера злоумышленника. Протестированы реакции современных веб-браузеров на получение подобных запросов:

Браузер	Версия	Тип реакции
Opera	27.0	Второй
Android Browser	4.2	Второй
Google Chrome	40	Второй
Google Chrome	41	Первый
Firefox	35.0.1	Первый

Атака реализована в инструменте для тестов на проникновение — BeEF (The Browser Exploitation Framework). Это проект с открытым исходным кодом. Похожая реализация атаки была сделана на языке C в виде отдельного приложения [2].

BeEF состоит из двух основных частей. Первая часть — серверная и устанавливается на компьютере тестировщика на проникновение, будем называть его исследователем. Вторая часть — JavaScript-сценарий на компьютере пользователя, локальная сеть которого является объектом исследования. Для реализации атаки написаны модуль (module) и расширение (extension). Это предусмотренные разработчиками BeEF средства для добавления новой функциональности в проект. Модуль отвечает за то, как действует JavaScript в браузере пользователя (вторая составная часть BeEF). Расширение отвечает за то, как реагирует на эти действия BeEF (первая составная часть BeEF). Написанные части вместе позволяют исследователю взаимодействовать с целевым веб-сервером из локальной сети пользователя так же, как если бы исследователь имел прямой доступ к серверу. Написанное расширение состоит из двух частей — веб-сервера и прокси-сервера. Веб-сервер при обращении к нему возвращает сценарий пользователю и блокирует дальнейшие его обращения, то есть реализует п. 3 и 4 приведённой схемы атаки. Прокси-сервер необходим для интерактивной двусторонней связи исследователя со сценарием, выполняющимся в браузере пользователя. С помощью этого сервера исследователь может отправлять или получать данные с целевого веб-сервера в локальной сети пользователя.

Главная часть модуля — это JavaScript-сценарий. Он состоит из трёх асинхронных запросов. Первый запрос регулярно обращается к прокси-серверу за очередным запросом исследователя к целевому серверу. Второй отправляет полученный запрос к целевому серверу и обрабатывает ответ. Третий отправляет результат обратно прокси-серверу. Вторая задача модуля — это занесение в DNS-сервер исследователя соответствия «доменное имя веб-сервера исследователя — сетевые адреса», необходимого для реализации п. 2 атаки.

Для того чтобы проведение исследования было возможным, должны выполняться следующие условия:

1) браузер пользователя не должен содержать открытых соединений с целевым веб-сервером;
2) браузер пользователя не должен содержать TCP-соединений с состоянием TIME WAIT с сетевым адресом целевого веб-сервера;
3) браузер пользователя не должен содержать в DNS-кэше других сетевых адресов, связанных с доменом веб-сервера исследователя.

При невыполнении хотя бы одного из этих условий не будет выполнен п. 3 схемы атаки. Браузер пользователя будет считать сетевой адрес целевого веб-сервера основным и не сможет получить сценарий.

Для защиты от данной атаки необходимо правильно настроить веб-сервер, который может оказаться атакуемым. Для того чтобы нарушить п. 5 схемы атаки, веб-сервер не должен отвечать на запросы, у которых заголовок Host содержит произвольный сетевой адрес.

ЛИТЕРАТУРА

1. `https://tools.ietf.org/html/rfc6454` — The Web Origin Concept.
2. `http://code.google.com/p/rebind/` — DNS Rebinding Tool.

УДК 004.65, 004.056.52 DOI 10.17223/2226308X/8/35

АТРИБУТНОЕ УПРАВЛЕНИЕ ДОСТУПОМ К ХРАНИЛИЩУ ДАННЫХ ТИПА «КЛЮЧ — ЗНАЧЕНИЕ»

С. В. Овсянников, В. Н. Тренькаев

Предлагается способ разграничения доступа пользователей к хранилищу данных типа «ключ — значение», когда право на доступ вычисляется в зависимости от параметров запроса (тип операции, идентификатор данных, пароль). Данный способ апробирован при разработке NoSQL СУБД с сервером управления доступом и удалённым хранилищем данных.

Ключевые слова: *атрибутное управление доступом, хранилище данных типа «ключ — значение», NoSQL база данных.*

В современных СУБД нередко отсутствует реализация так называемого мелко гранулированного управления доступом к данным (fine-grained access control), когда требуется ограничить доступ пользователей к отдельным строкам таблицы (как в случае реляционной БД) или к отдельной паре «ключ — значение» (как в случае NoSQL-хранилища данных). В данной работе для этих целей предлагается использовать подход, который можно отнести к атрибутной модели управления доступом [1], когда субъект имеет право доступа к сущности, если истинен предикат, вычисленный от атрибутов субъекта и/или сущности. При этом рассмотрена ситуация, когда имеется возможность пользователям самим настраивать политику безопасности СУБД, определяя правила задания разграничительной политики доступа к ресурсам БД.

Далее будем иметь дело с хранилищем данных типа «ключ — значение» (key — value), т. е. когда база данных представляет собой набор записей, идентифицируемых по ключу. Формально ключ будем рассматривать как слово в заданном алфавите. В простейшем случае каждый ключ ставится в соответствие значению в виде произвольных данных, в усложнённом варианте значение связано с определённым типом данных (целые, строки, списки, множества). Хранилище пар «ключ — значение» отличается упрощённой моделью запросов, используется малый набор операций: установка (set), получение (get), удаление (delete) значений по ключу.

Предлагается задавать политику безопасности хранилища данных с помощью функции управления доступом $C : K \times O \times P \to \{Allow, Deny, Pass\}$, где K — множество префиксов ключей; $O = \{set, get, delete, access\}$ — множество операций, P — множество парольных слов. Значение функции C, равное $Allow$, изначально определяется не менее чем для одной тройки $(k, access, p) \in K \times O \times P$, и тот, кто знает пару (k, p), может условно считаться администратором хранилища данных. Кроме запросов к хранилищу на обработку данных по ключу $(set, get, delete)$, возможны также запросы на изменение политики безопасности $(access)$, т. е. на задание (изменение) значений функции C.

Пусть запрос к хранилищу данных имеет следующие параметры: key (идентификатор данных), $value$ (значение данных), op (операция), pas (пароль). С точки зрения управления доступом запрос обрабатывается, следуя двум правилам.