АНАЛИЗ ИСПОЛЬЗОВАНИЯ СОВРЕМЕННЫХ АЛГОРИТМОВ ХЕШИРОВАНИЯ, ПРИМЕНЯЕМЫХ ДЛЯ АУТЕНТИФИКАЦИИ ПОЛЬЗОВАТЕЛЕЙ В ВЕБ-ПРИЛОЖЕНИЯХ

Москаленко Дарья Владимировна

студент Дальневосточного федерального университета, РФ, г. Владивосток

Множество проблем безопасности веб-сервисов связано с получением root-прав. Злоумышленник, используя атаки на хеш-функции или атаки перебором, в большом числе случаев достигает цели. В данной статье будет рассказано о том, как правильно использовать хеши для аутентификации пользователей сервиса.

Hash = хеш-функция — функция или свертка однозначного отображения строки длины n на конечное множество, где n может принимать любые значения. Сам хеш — это результат вычисления хеш-функции над данными. Существуют различные, как криптографические так и некриптографические хеш-функци [2].

Криптографические хеш-функции отличаются следующими условиями от остальных хеш-функций:

- необратимость: для установленного значения хеш-функции A должно быть невозможно в реальном времени найти блок данных X, хеш-функция для которого F(X)=A;
- стойкость к коллизиям 1-го рода: для какого-либо сообщения P должно быть невозможно в реальном времени подобрать другое какое-либо сообщение Q, для которого хеш-функция F(P)=F(Q);
- стойкость к коллизиям 2-го рода: должно быть невозможно, в реальном времени, подобрать такую пару сообщений (P, P'), хеш для которых одинаков [3].

Не будем вникать в подробности криптографии различных реализаций хеш-функций, достаточно запомнить и знать какие функции можно ещё использовать, а для каких уже нашли коллизии, и они скомпрометированы. К примеру, MD5 уже нельзя, на протяжении 3-х лет. А bcrypt/scrypt пока ещё держатся, их можно.

Хеш-функции в веб-приложениях используются в большинстве случаев для безопасного хранения паролей в базе данных. Только хэш-функция является последним звеном защиты, если злоумышленник свёл атаку к локальному нападению на систему аутентификации. Также возможно проводить атаку брут-форсом, перебирать пароли HTTP запросами, но данный вид атаки занимает слишком много времени.

Перечислю основные требования, которым должен удовлетворять хеш, хранимый в базе данных:

- наличие устойчивости к атакам перебора, как прямым, так и по словарю;
- невозможность нахождения одинаковых паролей различных пользователей по хешу.

Для того чтобы выполнить первое требование, необходимо использовать актуальные и криптостойкие функции хеширования.

Для выполнения второго требования необходимо использовать «соль» — строку случайных символов, которая должна подаваться на вход функции хеширования вместе с исходными данными. Таким образом, у пользователей с одинаковым паролем «12345» будут разные соли

«$salt1» и «$salt2», а это значит, что хеш-функции от «12345$salt1» и «12345$salt2» в базе тоже будут разные [1].

Но просто добавив соль, мы не особо усложним жизнь злоумышленникам. Необходимо ещё грамотно организовать хранение пароля и соли в базе. Есть большая вероятность того, что получив доступ к СУБД, хакер получит сразу и хеши и соли.

Чтобы увеличить сложность атаки перебором нужно дописывать соль к паролю, а никак иначе, потому что, согласно требованиям поточности алгоритма хеш-функция, зачастую, вычисляется построчно, и злоумышленнику будет проще, когда хеш от выражения будет начинаться с соли, а не с пароля. Ему будет известна длина соли, и сама соль, и соответственно будет гораздо проще вычислить сначала хеш от «соли», а потом уже вычислять отдельно хеш от «пароля». Сложность вычислений уменьшится и сведётся к простому вычислению хешей от всех паролей, которые он будет перебирать.

Есть способ ещё более усложнить жизнь злоумышленнику — использовать какой-либо локальный параметр, присущий именно той системе, на которой происходит аутентификация. К примеру, ID какого-либо устройства, процессора или жёсткого диска. Но тут сразу видны очевидные минусы этого способа — при поломке или смене оборудования необходимо будет обновлять итоговые хеши у пользователей.

С другой стороны, локальный параметр — это некая вторая «соль», которая дописывается ко всем паролям, и одинакова для всех хешей в базе. Но насколько этот метод эффективен, дописывать везде одну и ту же строку? В чем же трюк? В том, что локального параметра в базе нет. Это некая константа системы, которая находится в памяти приложения. В память она попадает любым доступным способом, только не из базы. Данная мера проста в исполнении и позволяет исключить атаку перебором, поскольку без знания данного параметра любой перебор хешей из базы будет бесполезен.

Со слов Артема Петросова, специалиста в ИБ компании Onsec, они ломали хеши с локальным параметром, разработав при этом алгоритм атаки: регистрировались в приложении, затем получали из базы свой хеш, соль, и, зная свой пароль, перебирали локальный параметр. Как оказалось, это было бесполезно и дорого по железу с длинами 16 и более байт для современных хеш-функций. В конце концов, после неудачной атаки на хеши, проще оказалось скомпрометировать систему аутентификации [4].

Учитывая всё выше сказанное, можно порекомендовать использовать в своих проектах «соль» в виде локального параметра, и хеш-функции scrypt/bcrypt. А если использование scrypt/bcrypt невозможно, то хотя бы внедрять локальный параметр даже на слабых MD5, он поможет повысить информационную безопасность вашей базы.

Если вы задумались о переходе с одного типа хеширования на другой, то непременно задумайтесь не только о необходимых трудозатратах, но и о производительности того или иного хеша. Как известно, более стойкие алгоритмы хеширования более требовательны к ресурсам. В части применения хешей для аутентификации пользователей важно не перегнуть палку, тестируйте нагрузку на скорость входа пользователей в секунду. Значение не должно превышать разумных пределов, иначе могут посыпаться вопросы от пользователей, которые вас не обрадуют. Но, как показывает опыт, для большинства проектов смена хеша на scrypt оказывается безболезненной.

Конечно, для каждой конкретной ситуации выбор идеального хеша может быть разным. К примеру, многие банки, для дистанционного банковского обслуживания, всё чаще выбирают решения, основанные на отдельном устройстве, которое генерирует хеши уже с заданной скоростью.

В заключении хотелось бы привести некоторые значения скорости перебора хешей (в мегахешах в секунду) которые были получены на видеокарте AMD Radeon 7870, которую можно приобрести за довольно адекватные деньги, около 350 $:

- MD5: 13200 M/s;

- SHA-1: 4800 M/s;
- SHA256: 1580 M/s;
- SHA512: 170 M/s;
- NTLM: 26300 M/s;
- bcrypt: 0,0075 M/s.

Список литературы:

1. Вирт Н. Алгоритмы и структуры данных. — М.: Мир, 1989.
2. Кнут Д. Искусство программирования, том 3. Сортировка и поиск = The Art of Computer Programming, vol.3. Sorting and Searching. — 2-е изд. — М.: «Вильямс», 2007.
3. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. — М.: Триумф, 2002.
4. Хеширование паролей — [Электронный ресурс] — Режим доступа. — URL: http://xxsybreedxx.blogspot.ru (дата обращения 10.03.14).