**Bruteforce как средство передачи информации**

Это очень похоже на обычные бытовые ситуации, когда один человек не говорит о том, о чем у него спрашивают, отвечая однозначно "да" или "нет", а отвечает всегда "нет", и лишь в один из удачных моментов отвечает "да".

Технология передачи информации за счет перебора представляет собой передачу относительно небольшого объема информации, возможно, дающей ключ к более широким перспективам. Например, мы можем себе представить перебор пароля на архив. Если мы его узнаем, то, собственно говоря, мы сделаем именно то, что хотели, а заодно и узнали те несколько байт, которые были для нас преградой - пароль!

Односторонний перебор.

Перебор архива является примером одностороннего перебора, т.е. мы единожды подобрав пароль теряем весь интерес к дальнейшему перебору или перебору с самого начала. Такой перебор дает нам ограниченную информацию размером вряд ли более 16 байт, а скорее около 2-8. Подобный перебор не представляет особого интереса.

Двусторонний перебор.

Наша с вами задача - получать информацию посредством перебора. Двусторонний перебор подразумевает под собой знание того же изначального пароля и как следствие - его использование.

Например, представим себе интернет-сайт с доступом к нему только после аутентификации. Аутентификационный пароль может состоять только из одного байта, так что при использовании латинского алфавита мы имеем. 66 комбинаций (a-zA-Z0-9). Итак, мы сможем за 65 запросов однозначно узнать пароль, после чего можем установить свой.

Вот тут включается вторая сторона и производит аналогичные действия: перебирает пароль и максимум через 65 попыток его узнает, после чего опять же устанавливает свой.

В это время сам пароль будет носителем информации. Т.е. если наш перебор остановился на 'j', значит, мы получили 6 6ит информации. Какой информации - можно установить по заранее изготовленным таблицам либо алгоритмам.

Самый простой вариант - a=1, b=2, c=3, ...

Применение данному способу можно найти следующее.

Существует внутренний корпоративный сайт с ограниченным доступом в интернет. У него существует собственная страница, которая доступна как со стороны внутренней сети, так и со стороны интернет. Сайт позволяет производить аутентификацию для просмотра некоторых данных. Имея Web интерфейс для изменения пароля доступа к станице и право на это, можно собственно и устроить передачу информации через форму ввода пароля.

Подобные ситуации возникают нередко. Например, провайдер может предоставлять бесплатный доступ к странице просмотра статистики, при этом ничего кроме этой страницы не работает. Такой же фокус используют некоторые предприятия, допуская всех корпоративных пользователей лишь к единственной странице.

Данный способ наиболее универсален, однако создает значительно более узкий канал, нежели иные способы передачи данных через Web формы (да и любые интерактивные элементы, позволяющие хранить любую информацию на стороне сервера, либо каких-либо его посредниках).

Представим себе следующую ситуацию: вы диалапник, у вашего провайдера имеется страница с доступом к статистике и ее некоторым настройкам, хранящимся на сервере, например, пункт о периодичности рассылки с состоянием вашего счета. Для передачи информации вам уже нет необходимости осуществлять перебор для просмотра этого параметра, т.к. он высвечивается как элемент ваших личных настроек. Благодаря этому ваша скорость по передаче данных значительно увеличивается.

Что значит синхронизация передачи данных? Подумайте, как вы будете передавать информацию, если у вас нет непосредственной связи с вашим скриптом. Информацию надо через ЧТО-ТО передавать. Вот тут мы и рассмотрим механизмы синхронизации передачи данных. У нас есть серверная часть (со стороны интернет) и клиентская (со стороны корпоративной сети). Серверная часть выполняет наши запросы, команды, а клиентская их задает. В обязанности обоих сторон входит отслеживание состояния. Возьмем наш метод передачи данных через поле пароля.

Рассмотрим следующий вариант. Условимся, что пароль

"1" будет означать дефолтовую ситуацию, когда ничего не передается

"2" - начало передачи данных, запрос.

"3" - подтверждение передачи, успешная передача.

Cо стороны сети устанавливаем пароль "2" (по умолчанию был установлен пароль "1") и периодически проверяем пароль.

Сервер, увидя, что пароль стал "2", в свою очередь подтверждает это установкой пароля "3". Это значит, что он принял наш сигнал и готов к получению запроса.

Мы проверяем это событие проверкой пароля, и если пароль равен "3", то теперь настала очередь передачи данных. За один раз мы передадим 1 бит, а запрос может быть, например, из 20 байт. Мы устанавливаем пароль "a".

Со стороны сервера происходит следующее. Сервер производит проверку, установлен ли пароль "a" или "b" (если ни один из них не подходит, то проверяется пароль "1", что означает конец передачи данных). Исходя из значения пароля, делается вывод, пришел бит 1 или 0 - т.е. мы передали серверу 1 бит.

После перебора сервер устанавливает пароль "2" - сигнал, что нужно передать следующий бит.

Передается следующий бит.

По окончании передачи клиент устанавливает пароль "1" и теперь начинает работать в режиме сервера.

В это время сервер, получив запрос, выполняет его (например, скачивает указанный УРЛ).

После скачивания данных сервер выставляет пароль "2", а клиент ждет этого пароля.

Получив пароль "2", клиент устанавливает пароль "3".

Сервер, определив, что уже установлен пароль "3", начинает побитовую передачу. А мы подтверждаем каждый бит установкой пароля "3".

Существует очень много возможностей реализовать данный трюк, однако все эти возможности настолько разнообразны, что написать какой-то универсальный движок будет достаточно сложно. Например, для одного из местных провайдеров существует способ передачи к себе информации по каналу шириной в 2 бита за 1 запрос, а в обратную сторону можно использовать достаточно широкий канал шириной около 24 байта.

Предыстория нахождения подобного извращенного метода передачи информации достаточно невеселая. Вечерами и ночами я работал в городе Норильске, обслуживая одну из местных контор, план работ давно закончил, а делать было нечего, вот и вспомнил про провайдеров далекого Урала.

К данной технике пытался написать движок на Perl, но времени не хватило, однако кое-что осталось.

При передаче информации таким путем необходимо:

Шифровать информацию

Упаковывать

Конечно же использовать rar для такого не будешь, и поэтому мне помог друг, проведя тесты архиваторов.

Результат оказался предсказуем:

ASS GZ 48

ASS BZ2 51

ASS LZH 54

ASS RAR 98

ASS\_ ZIP 118 // winrar zip

ASS 7Z 119

ASS ARJ 121

ASS ZIP 130 // 7z zip

ASS 512

Содержимое файла ass:

asasasasasasasasasasasasasasasasa

asasasasasasasasasasasasasasasasa

asasasasasasasasasasasasasasasasa

asasasasasasasasasasasasasasasasa

asasasasasasasasasasasasasasasasa

asasasasasasasasasasasasasasasasa

asasasasasasasasasasasasasasasasa

asasasasasasasasasasasasasasasasa

asasasasasasasasasasasasasasasasa

asasasasasasasasasasasasasasasasa

asasasasasasasasasasasasasasasasa

asasasasasasasasasasasasasasasasa

asasasasasasasasasasasasasasasasa

asasasasasasasasasasasasasasasasa

asasasasasasasasasasas

Режим движка "Ожидание запроса".

Алгоритм следующий:

Запускается модуль, производящий ожидание запроса (все зависит от типа пароля либо метода передачи (не исключена возможность передачи средствами XSS атаки, как это реализуемо у одного из Уральских провайдеров)).

При получении запроса происходят операции, аналогичные 4-8.

Выполняется запрос в Интернет (да, эти пункты скорее будут в движке, т.к. они производятся обычным образом, например, с помощью библиотеки LWP).

Принимается ответ из Интернета.

Упаковываются данные.

Зашифровываются данные.

Запускается модуль отправки результата, который и принимает ответные данные.

Библиотеки, которые я вам советую для разработки: LWP, Compressed::ZLib, MIME::Base64.

Пожалуй на этом можно остановиться. Хочется добавить лишь примеры извращенных методов передачи:

передача информации за счет счетчиков посещений (требуется контроль за правильностью передачи, ведь не исключены заходы других юзеров);

анализ задержек ответа сервера (можно искуственно создавать незначительные задержки с любой стороны различными методами);

в некоторых случаях передачу данных можно осуществлять за счет сессий (номера которых обычно сохраняются у клиента в куках).