АТАКИ НА СЕТЬ ЧЕРЕЗ ПЕРЕПОЛНЕНИЕ БУФЕРА – ТЕХНОЛОГИИ И СПОСОБЫ БОРЬБЫ

Материал подготовлен специалистами компаний Rainbow Technologies и WatchGuard Technologies на основании публикаций Rik Farrow, специалиста по компьютерной безопасности.

Сегодня перед специалистами, отвечающими за информационную безопасность корпоративных ресурсов, очень остро стоит проблема атак на сеть путем переполнения буфера. Практически еженедельно, а иногда и несколько раз в течение недели консультанты LiveSecurity (служба сервиса компании WatchGuard) предупреждают о новых уязвимостях, связанных с переполнением буфера.

Переполнения буфера встречаются везде, даже в коде, в котором, по заверениям разработчиков, все потенциальные возможности для этого исследованы и устранены.

Возможно, вы слышали о таких вариантах переполнения буфера, как формат строки или атаки на хип. В данной статье, используя аналогии из повседневной жизни, мы попытаемся объяснить, как работают эти атаки. Позаимствуем идею из книги Брюса Шнеера *«Секреты и ложь» (Bruce Schneier «Secrets and Lies»)*, хотя разовьем ее и обобщим.

ГЛУПЫЙ ПРОДАВЕЦ

Шнеер объясняет переполнение буфера, сравнивая компьютерную память с перекидным ежедневником, содержащим инструкции для продавца круглосуточного магазина. На каждой странице написана одна инструкция, например: «Поприветствовать посетителя», или «Выбить чек», или «Принять деньги». Предположим, что продавец глуп и может работать, только дословно выполняя инструкции.

Всё это делает наш магазин уязвимым для простой атаки. Атакующий подходит к стойке и, пока продавец роется в инструкциях, вставляет в них листок, на котором написано: «Взять все деньги из кассы и передать их покупателю». Единственное, на что в подобном случае можно надеяться, так это на то, что если продавец точно выполняет все инструкции, то он должен был их запомнить и, наверное, заметит, что здесь что-то неладно.

ПО СТРАНИЦАМ МОЕЙ ПАМЯТИ...

В отличие от продавцов компьютеры точно выполняют инструкции и при этом вообще лишены чувств. Если атакующий сможет подсунуть компьютеру дополнительные инструкции, он выполнит их один в один. Это является основой для нападения, связанного с переполнением буфера.

Существует три разновидности атак, основанных на переполнении буфера: атаки на стек, формат строки или хип. Эти разновидности одинаковы по сути, но каждая направлена на разные части памяти компьютера. Чтобы понять различия между этими атаками, вкратце обрисуем, как работает компьютерная память.

Когда программа начинает выполняться, операционная система выделяет для нее виртуальную память большого размера. Можно рассмотреть эту память как разграфленную тетрадь, в которой написана программа, начиная со страницы, где хранятся инструкции, и заканчивая данными программы. После того, как программа записана, остается много чистых страниц. Пустые страницы, находящиеся сразу же за данными программы, называются хипом (heap — куча), а те, которые находятся в самом конце тетради, называются стеком. Как если бы вы использовали тетрадь с двух сторон: хип будет расти в сторону конца тетради, а стек — в сторону начала. А тетрадь (то есть виртуальная память) настолько велика, что хип никогда не достигнет стека и наоборот.

Позиции внутри этой виртуальной памяти (страницы в тетради) в программе, стеке, хипе или между ними задаются адресами, выраженными в шестнадцатеричном формате. Например, старший адрес в памяти размером 2 Гбайт будет равным 8FFFFFFF. Небольшие области этого адресного пространства служат для ввода данных в программу. Эти области, которые могут иметь адреса в стеке или хипе, называются буферами. Таким образом, мы нашли первую разгадку на пути того, почему атаки называются «переполнения буфера». Если вы услышите, как кто-то говорит: «Это переполнение буфера в стеке», или «Это нападение на стек», или «Это переполнение буфера хипа», — значит, он хочет указать на проблему с памятью, которая выделена программе.

АТАКИ НА СТЕК

В стеке хранятся временные данные. Вновь мы можем проиллюстрировать это примером из книги Шнеера. Предположим, что вы пишете заметки по проекту, над которым работаете, ко-

гда звонит телефон. Звонящий сообщает некую информацию, которую вы у него запрашивали, следовательно, вы берете новый листок бумаги, кладете его поверх первого и записываете эти данные. Прежде чем вы успеваете завершить разговор, в комнату входит начальник, привлекает ваше внимание и просит вас сделать кое-что по окончании звонка. Вы берете еще один листок бумаги и записываете на него просьбу начальника. Теперь у вас есть небольшая стопка (стек) листков бумаги с написанными на них инструкциями и данными. Как только вы выполняете очередное задание, вы сминаете листок и бросаете его в мусорную корзину. Вы используете стек таким же образом, как и в случае с атакой, направленной на переполнение буфера.

Если мы будем говорить о компьютерах, то там, конечно, нет листков бумаги, а есть просто память (RAM). Данные действительно добавляются в стек сверху и потом извлекаются. При атаке «переполнение буфера», направленной на стек, нарушитель добавляет в него больше информации, чем предусмотрено, при этом лишняя часть перезаписывается поверх данных, для которых разработчик программы не предусмотрел такой вариант. Например, предположим, что при исполнении программы она дошла то такой стадии, на которой необходимо использовать почтовый индекс из web-формы, заполненной пользователем. Величина даже самого длинного почтового индекса не превышает 12 символов. Но в нашем примере web-форму заполняет нарушитель. Вместо введения почтового кода он 256 раз вводит букву «А», а за ней пишет определенные команды. После того, как программа получает эту сверхдлинную строку, бессмысленные данные переполняют буфер, выделенный для почтового индекса (как вы помните, буфер — это область памяти, зарезервированная для ввода данных), и команды атакующего попадают в стек. Как и в случае с вором, подсовывающим инструкцию: «Отдай мне все деньги», — атака типа «переполнение буфера» подкладывает инструкции, которые программа в обычных условиях не должна выполнять. Будучи дословным исполнителем, компьютер не сможет выполнить неверные инструкции — программа завершится аварийно. Если же инструкции точны, программа слепо выполнит команды атакующего.

В идеале программисты защищаются от атак, связанных с переполнением буфера, путем проверки размерности всех данных, поступающих в программу, и того, что они не превысят тот размер памяти, который для них предусмотрен. (В приведенном выше примере с почтовым индексом программа должна быть написана так, чтобы не вводить больше 12 символов.) На практике же программисты часто забывают о том, что программу могут атаковать или что данные могут поступать из «ненадежных» источников. Чем сложнее становится программа, тем больше вероятность того, что произойдет атака.

АТАКИ НА ФОРМАТ СТРОКИ

Атаки на формат строки также используют стек, но требуют гораздо меньше изменений, чем переполнение буфера, которое мы обсуждали ранее. Форматирование означает подготовку каких-либо данных к отображению или печати. Однако инструкции форматирования так гибки, что некоторые нарушители нашли способы использовать их для записи в память.

Атаки на формат строки обычно добавляют в память адрес, указывающий на другую ссылку, по которой нарушитель добавляет свои исполнимые инструкции. Используя нашу аналогию с глупым продавцом, предположим, что его книга с инструкциями содержит 25 страниц. Предположим также, что после страницы с инструкцией, гласящей: «Возьми у покупателя деньги и открой кассу», — вор вставил инструкцию: «Перейди на страницу 26». Вор мог подготовить несколько страниц с инструкциями типа: «Отдай покупателю все деньги», «Дай ему уйти и не поднимай тревоги» — и поместить их в конец книги. Если глупый продавец последует этим указаниям, это будет аналогично программе, которая перешла по указанному адресу в памяти и выполнила все найденные там инструкции.

КУЧИ ПРОБЛЕМ

Атаки на хип совершенно не затрагивают стек. Вспомните аналогию с заметками (проект, телефон, начальник) — стек использует временную память. В противоположность этому хип — это название, данное программистами памяти, которая не является временной, а должна быть готовой к использованию в течение работы программы. Страницы хипа могут быть считаны или записаны, и этим удачно пользуются хакеры. Они пишут инструкции атаки на страницах хипа и затем заставляют компьютер выполнять их. Технически это не отличается от атаки на стек.

ЗАЩИТА: О ЖУКАХ И КАНАРЕЙКАХ

Мы уже знаем, что является наилучшей защитой от подобного вида атак. Это грамотное программирование. В идеале каждое поле в каждой программе должно позволять вводить только заданное число символов (концепция, известная как «проверка границ») и только определенного типа (почему, например, программа должна позволять вводить буквы или метасимволы типа % для телефонного номера?). Также мы знаем, что программы несовершенны, они содержат ошибки, некоторые из которых позволяют проводить атаки. В связи с этим программисты придумали схемы защиты от атак, связанных с переполнением буфера.

Простейшая схема основана на том, что в стеке и хипе должны быть только данные, компьютер никогда не должен выполнять найденные там инструкции. Этот подход прекрасно работает во многих UNIX-системах, однако не может использоваться в Windows. Более того, эта схема заставляет UNIX-администраторов изменять параметры конфигурации на каждом сервере. Легче всего это сделать на неинтеловских процессорах (например, Sun Microsystem's Sparc).

Другая популярная схема защищает от переполнения буфер, но только связанный со стеком. Эта защита основана на использовании «канареек». Помните рассказы о шахтерах, которые брали с собой в угольные шахты канареек? В шахтах часто выделяется опасный газ —

метан, который не имеет запаха и ядовит. Если шахтеры будут углублять шахту в ту сторону, где выделяется метан, канарейки умрут первыми, чем дадут людям шанс покинуть опасную зону.

«Канарейка» защищает стек, будучи помещенной в критические места памяти (около адресов возврата) и указывая компьютеру, какие команды выполнять после завершения текущей функции. Перед использованием адресов возврата программа проверяет, в порядке ли «канарейка». Если она уничтожена, программа завершает работу, сообщая об ошибке.

Идея использования «канареек» принадлежит группе разработчиков Linux, создавших версию Linux (Immunix. com), использующую Stackguard для встраивания «канареек» в операционную систему и прилагающиеся программы. Новый компилятор Microsoft для среды Visual C тоже имеет возможность добавлять «канареек» в стек.

«Канарейки», конечно, помогают, но не могут полностью защитить от атак на хип. Эти атаки совершенно не затрагивают стек и обходят «канареек». Таким образом, программисты должны создавать такой код, который позволяет копировать в буфер только то количество данных, на которое он был рассчитан (или, другими словами, писать программы правильно). Этот способ является наиболее эффективной защитой.

ЧТО ВЫ МОЖЕТЕ СДЕЛАТЬ С ПЕРЕПОЛНЕНИЕМ БУФЕРА

Проблему переполнения буфера сегодня можно попытаться решить, используя специализированные аппаратные или программные средства. Довольно хорошо с подобными проблемами справляются современные межсетевые экраны, в том числе включенные в UTM-устройства WatchGuard Firebox. Пользователи этих устройств имеют дополнительный рубеж обороны, который заключается в следующем. Когда вы настраиваете межсетевой экран на использование служб прокси, это программное обеспечение отслеживает использование чрезвычайно длинных входных данных для защищаемых сервисов: электронной почты, HTTP, FTP и DNS. Не являясь идеальной защитой, прокси, тем не менее, могут остановить многие атаки, направленные на переполнение буфера. При использовании пакетных фильтров (даже с динамическим анализом) вы лишаетесь этого преимущества.

Хочется надеяться на то, что эта статья помогла вам понять, что такое атаки, направленные на переполнение буфера, каковы разновидности этих атак и применяемые контрмеры, а также почему даже эти контрмеры не всегда работают. Но помните, что вы не беззащитны. Если ваш межсетевой экран поддерживает шлюзы приложений или прокси, используйте их. Когда служба информирования LiveSecurity предупреждает вас о критичных уязвимостях, связанных с переполнением буфера, применяйте заплатки для приложений. Используйте эти меры, ваши новые знания о переполнениях буфера, и всё будет в порядке.