захват чужих WEB-аккаунтов через дыры в FS

крис касперски ака мыщъх, no-email

многопользовательские операционные системы (к которым относятся Linux и xBSD) содержат огромное количество ошибок, приводящих к утечкам данных и утрате конфиденциальности, что особенно актуально для "промышленных" серверов, обслуживающих тысячи пользователей, имеющих право на установку своих собственных программ (например, PHP-скриптов). в ходе широкомасштабного исследования, проведенного мыщъх'ом, выяснилось, что никсы содержат фундаментальные уязвимости, позволяющие злоумышленникам получать доступ к удаленным данным других пользователей, среди которых помимо мусора зачастую находится очень много интересных вещей!

введение

Во времена MS-DOS для восстановления ошибочно удаленных файлов помимо энуреза использовался следующий трюк. Создавался файл, открытый на запись, делался seek до конца диска, после чего файл закрывался, вбирая в себя все свободное пространство, над которым, конечно, еще предстояло поработать, но не фрагментированные текстовые файлы "вытягивались" без проблем.

Linux и BSD таких вольностей уже не допускают и заботливо "подчищают" выделяемое дисковое пространство, забивая файл нулями, предотвращая захват удаленных данных, принадлежащих другим пользователям. На этом можно было бы поставить жирную точку и закончить статью (как говориться, на нет и суда нет), но механизм затирания реализован криво, с большим количеством ошибок, то исправляемых, то вновь появляющихся в новых ядрах, а потому, представляет интерес раскурить эту тему, погрузившись в клубы благородного дыма, испускаемого медленно тлеющими распечатками исходных текстов ядра и сопровождающих его библиотек.

Вот с библиотек мы и начнем.

Рисунок 1 мыщъх в тусклом свете норы своего одиночества, где клавиатура самый близкий друг, намного более отзывчивый чем жена, которая в очередной раз ушла

libc internals

Для работы с файловым в/в большинство программистов использует функции библиотеки libc — fopen(), fseek(), etc, являющиеся достаточно тонкими обертками вокруг системных вызовом open, lseek и т. д. Насколько тонкими? Хороший вопрос!!! Разработчики libc (и ее аналогов, поставляемых вместе с компиляторами типа gcc) никак не встанут на путь единой половой ориентации, бросаясь из крайности в крайность. То они в порыве энтузиазма начинают чистить выделяемое файлу пространство сами, то перекладывают эту заботу на lseek, которая как бы выполняет подчистку еще внутри ядра.

"Как бы" потому что lseek явным образом не гарантирует подчистки выделяемого пространства и выполняет его далеко не везде, не всегда и существует тысяча исключений, при которых подчистка умышленно не выполняется. Почему так, мы расскажем ниже, а пока обратим внимание на то, что подавляющее большинство современных высокоуровневых библиотек (в том числе и входящих в интерпретируемые языки типа Perl, Python, PHP) самостоятельной подчистки _не_ выполняют и потому для реализации атаки спускаться на уровень системных вызовов совершенно необязательно (хотя и желательно для нейтрализации возможных побочных эффектов).

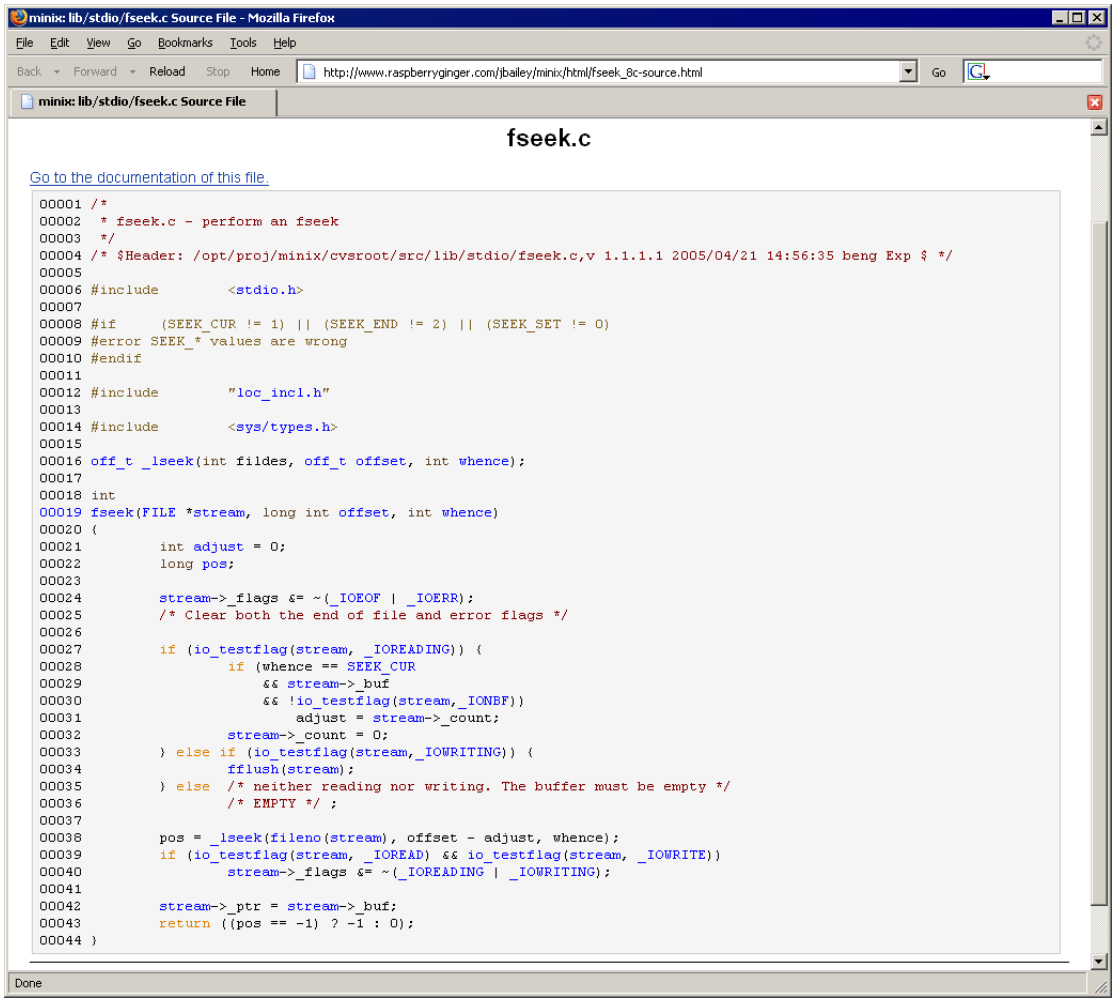

Рисунок 2 пример реализации fseek(), не заботящейся о подчистке данных и молчаливо перекладывающих эту заботу на плечи системного вызова lseek, услугами которого она, собственного говоря, и пользуется

kernel internals

Операционные системы семейства Linux и BSD реализуют унифицированную политику в/в, обеспечивая единый интерфейс взаимодействия как между файлами, так и между (псевдо)устройствами, что существенно упрощает программирование, попутно сокращая

количество системных вызовов, но вместе с тем порождает серьезную проблему, нарушающую стройную концепцию безопасности.

Очевидно, что при работе с устройствами (например, дисковыми томами) функция lseek не может, просто не имеет права, заниматься "подчисткой", поскольку это чревато глобальными разрушениями данных. С другой стороны, lseek должна гарантировать, что при выделении файлу новых кластеров, их содержимое будет "отцензурированно", то есть забито нулями, во избежании попадании конфиденциальных данных в лапы к врагу.

Вот так и разрушается единообразие доступа ко всем файлам и устройствам. Абстрагироваться от природы объекта, над которым выполняется операция позиционирования, уже не получается и приходится фаршировать ядерный код дополнительными проверками, за правильность реализации которых никто и никогда не ручался.

Теоретически, ядро операционной системы при _любых_ операциях позиционирования никогда не должно отдавать пользователю не подчищенные кластеры, содержащие данные, принадлежащие ранее удаленным файлам и _простейшие_ тесты на вшивость (открыл файл, выполнил позиционирование на пару мегабайт, записал несколько байт, закрыл файл) показывают, что политическая ситуация Багдада спокойна как слон в нирване и поводов для беспокойства как будто бы и нет.

Но разве серьезные тесты так выполняются?! Давайте испробуем более хитрые комбинации, вводящие ядро в замешательство и заставляющие его отдавать нам чужие данные, чего по логике вещей происходить не должно, но… все-таки происходит!

Разные ядра имеют разные дыры и в процессе экспериментов (проводимых с удаленными серверами) мыщъх далеко не всегда мог определить с каким именно ядром он имеет дело, а потому приводит лишь голый псевдокод без указания жертвы, на которой он работает. Как уже говорилось выше, конкретные реализации могут использовать либо системные вызовы (если они доступны на том языке, который установлен на удаленном сервере), либо высокоуровневые файловые функции в/в.

В принципе, возможна комбинация, при которой "дыра" в ядре перекрывается библиотечной реализацией fseek, самостоятельно подчищающей выделяемые файлу данные, но… тут уж ничего не поделаешь. Это на локальной машине мы можем выбирать все, что нам заблагорассудится, а вот при атаках на сервера приходится использовать что дают. К счастью, серверов сейчас намного больше одного и если не ставить перед собой цель подломать какой-то конкретный сервер, а просто хочется чего-нибудь захачить (неважно что), то приятное время провождение обеспечено.

Администраторам же остается только посоветовать применить ниже перечисленные алгоритмы на себе, и при необходимости предпринять защитные меры.

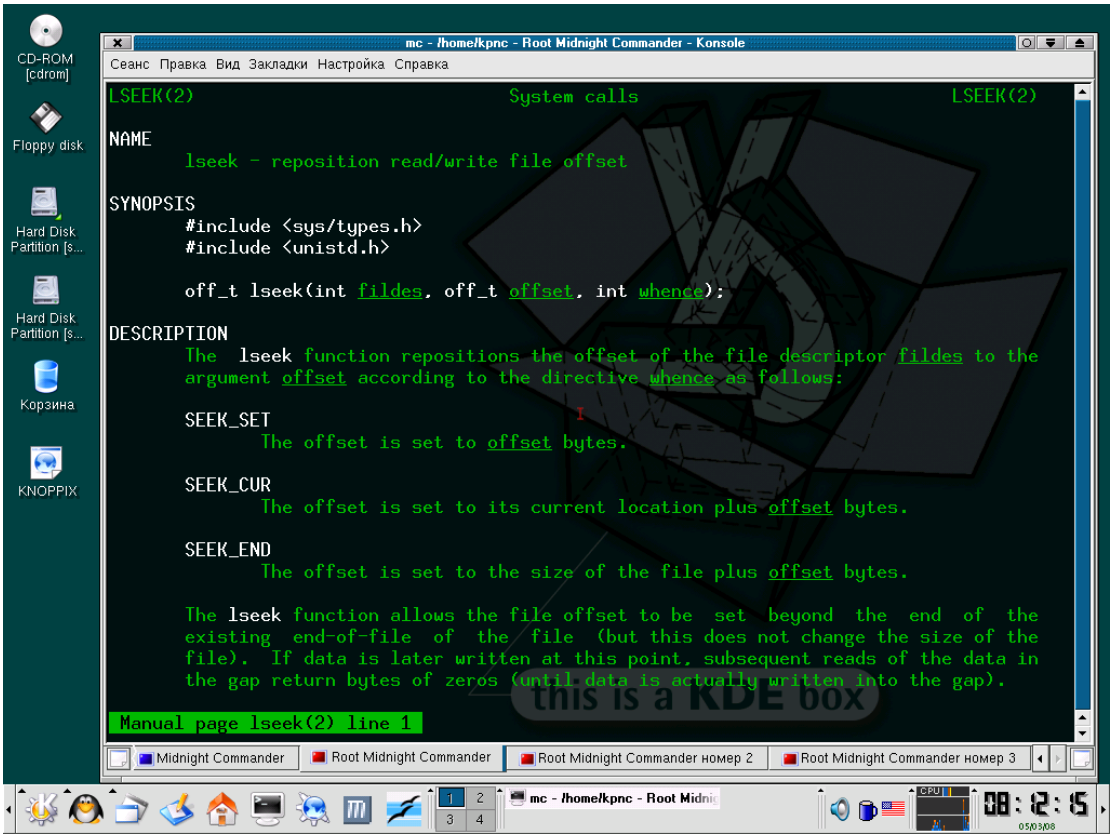

Рисунок 3 man по lseek

six bullets

Выстрел первый. Предупреждающий. В смысле дохляк полный. Но иногда он все-таки срабатывает:

1. открываем файл на запись;
2. делаем fseek/lseek на сколько хватит совести/квоты;
3. записываем в файл несколько байт;
4. закрываем файл;
5. смотрим, что за дичь попала в наши сети (в большинстве случаев — нули);

Выстрел второй. Уже прицельный и работающий на достаточно большом количестве операционных систем и библиотек;

1. открываем файл на запись;
2. делаем fseek/lseek на сколько хватит совести/квоты;
3. записываем в файл _ноль_ байт;
4. закрываем файл;
5. смотрим, что за дичь попала в наши сети (как правило, что-то да попадается);

Выстрел третий. Контрольный. Срабатывает довольно часто, хоть иногда и промахивается.

1. открываем файл на запись/чтение;
2. делаем fseek/lseek на сколько хватит совести/квоты;
3. записываем в файл _ноль_ байт;
4. делаем fseek/lseek на начало файла;
5. читаем пока не встретим EOF (в некоторых случаях EOF встречается сразу, в некоторых — мы имеем нули до позиции последнего seek'a, а в некоторых — захватываем чужие данные из невычещенных кластеров);
6. закрываем файл (впрочем, теперь его можно уже и не закрывать);

Выстрел четвертый. Вариация на предыдущую тему с той лишь разницей, что на шаге N#3 мы записываем не ноль байт, а хотя бы один.

Выстрел пятый. Пуля со смещенным центром тяжести. Довольно эффективна для SuSE Linux и некоторых BSD-подобных операционных систем (предположительно FreeBSD):

1. открываем файл на запись;
2. делаем fseek/lseek на N (мега)байт, относительно начала файла;
3. записываем в файл _ноль_ байт (в некоторых системах — один байт);
4. делаем fseek/lseek на N/K, где N > K, относительно конца файла;
5. записываем в файл _ноль_ байт (в некоторых системах — один байт);
6. закрываем файл — есть шанс, что на отрезке от K до N окажутся захваченные данные;

Выстрел шестой. Последний. Если не поможет и он, то, поцеловав холодную сталь ствола, падем на землю и ждем смерти, согласно композиции the last ammunition (какая к черту холодная сталь, когда мы только что выпустили из пистолета весь наш боезапас?!), но ладно, не будем придираться. В хакерстве, как и в лирике есть свои нюансы.

1. открываем файл на запись;
2. мотаем цикл, последовательно делая fseek/lseek на размер кратный длине кластера (подбирается экспериментально);
3. ждем пока fseek/lseek не вернет ошибку;
4. записываем в файл _ноль_ байт (в некоторых системах один);
5. закрываем файл — некоторые системы некорректно обрабатывают ситуацию с записью в файл после ошибки позиционирования, при условии, что само позиционирование осуществлялось "порциями", равными размеру одного кластера, в результате чего мы сразу же захватываем не вычищенное содержимое всех секторов;

Разумеется, поимо описанных способов, существуют и другие трюки. Их достаточно много, даже слишком много, чтобы перечислять, тем более, что все вертится вокруг одного и того же механизма. Записи нуля байт или одного байта и запутанной схемы позиционирования по файлу в надежде, что ядро "рехнется" и забудет подчистить выделяемые кластеры.

Один тонкий момент. Часть ядер содержат ошибку при которой в закрытом файле оказываются одни нули, но если этот файл читать до его закрытия, то… можно получить доступ к информации, к которой мы доступа иметь не должны. Идея, надеюсь, понятна?!

К сожалению, написать универсальный грабер дискового пространства так и не получилось и вот почему: если мы выбираем неверный прием (обрабатываемый ядром), то данные, принадлежащие ранее удаленным файлам, необратимо затираются нулями и к моменту пока мы подберем "ключ", никаких полезных данных на диске скорее всего уже не останется, однако, сервер не стоит на месте, а продолжает активно работать, пользователи создают и удаляют файлы, так что, после того, как правильный "патрон" найден им можно пользоваться вновь и вновь, беззастенчиво тыря информацию у своих соседей по серверной площадке.

Рисунок 4 шесть пуль в хакерском кармане

monopoly

Имея монопольный доступ к атакуемой машине не ограниченный никакими квотами, мы можем захватить хоть все свободное пространство целиком, а в нем… сказать, что до хвоста всего интересно означает не сказать ничего, поскольку там просто кладезь критических данных, позволяющих нам повысить уровень своих привилегий или просто похулиганить.

Достаточно многие администраторы хранят пароли, назначаемые пользователям, открытым текстом в специальных файлах, доступным только им одним (в самом деле, ситуация в которой пользователь забыл свой пароль — более чем типична, а поскольку Linux/BSD хранят не пароли, а их хэши, то единственный выход — назначить пользователю новый пароль, но тогда он его точно забудет). Даже если файл с паролями ни разу не удалялся, все равно — в процессе открытия/модификации/сохранения, операционная система могла перенеси его на новое место, создать резервную копию, временный файл, автоматически удаляемый после редактирования и т. д. Словом, возможностей много.

Электронная почта — это вообще Клондайк. Чего там только не бывает! А файлы после их прочтения удаляются довольно часто (редко кто хранит всю почтовую базу целиком), а если еще на атакуемой машине расположен почтовый сервер, складывающий все сообщения в папку типа inbox, откуда администратор забирает их по протоколу POP3 это прямой путь к root'у.

Впрочем, монопольный доступ к чужой машине в реальной жизни практически никогда не встречается, особенно, если на этой машине работает администратор.

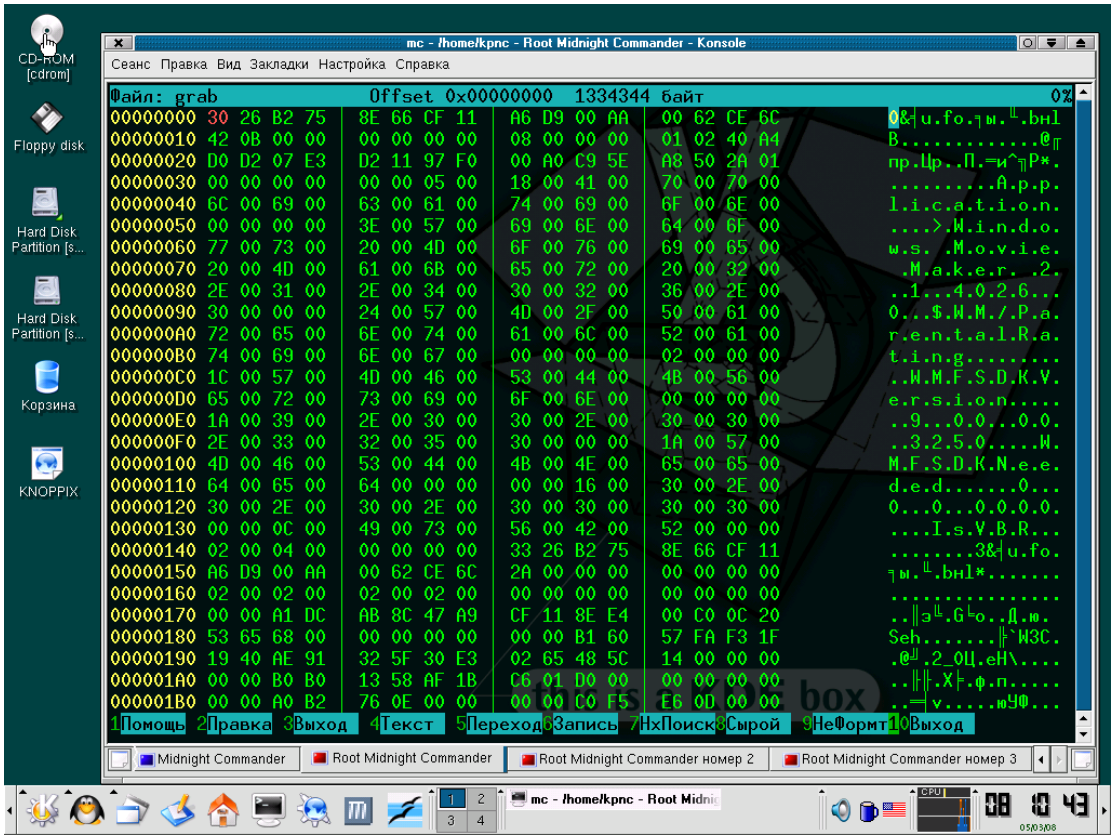

Рисунок 5 мыщъх грабит свою локальную машину

WEB-accounts are under attack

Прежде чем обсуждать аспекты атаки на WEB-аккаунты поговорим о такой неприятной для хакеров штуке как квотирование дискового пространства, где мы уже не можем сделать seek на размер нашей совести, поскольку объем квоты наверняка меньше. Ну и что интересного мы там найдем?!

На самом деле — много чего! Квотирование лишь лимитирует суммарный размер всех файлов, принадлежащих данному пользователю, но не закрепляет за ними какой-то конкретный региона дискового пространства и потому операционная система может выделять нам любой. Ну… или практически любой, как правило, наилучшим образом соответствующий размеру данного файла плюс небольшой "зазор" на вырост. Таким образом, делая seek на различные расстояния, мы каждый раз захватываем разные блоки данных. А если учесть, что карта свободного пространства сильно нагруженной дисковой системы постоянно меняется, то у нас есть все шансы "перепахать" все свободное дисковое пространство, естественно, тут же возвращая захваченные данные назад, во избежании превышения отпущенной нам квоты.

Лучше всего делать это в бесконечном цикле, чтобы овладевать чужими файлами сразу же после их удаления, пока они не будет затерты кем-то еще. Конечно, подобная дисковая активность редко остается незамеченной и администратор запросто может вызывать нас на ковер, а в случае бесплатного WEB-хостинга вообще закрыть аккаунт без всяких предупреждений, но… кто не рискует, тот не пьет шампанского!

Теперь перейдем непосредственно к технике захвата чужих аккаунтов. Что содержится в удаленных данных? В основном — исходные тексты скриптов, зачастую содержащие грубые

ошибки (т. е. дыры), которые остается только найти и заюзать. Кстати, пароли, жестко прошитые в теле скрипта, — вполне обычные дело, встречающее практически повсеместно.

Следом идут данные пользователей, среди которых числятся иногда и сами держатели аккаунта. Что-то типа тестового входа, зачастую с тем же самым паролем, что управляет основным аккаунтом. Именно так мыщъх в короткое время захватил сотни WEB-аккаунтов, правда, не желая никому причинять вреда, тут же вернул их обратно.

Про номера кредитных карт, адреса электронной почты и бла-бла-бла вряд ли стоит говорить. Случается, что их шифруют, но это один случай из миллиона. Практически все их держат открытым текстом. Конечно, использовать такую информацию в общем случае незаконно, но… вот написать пользователям насколько надежен сайт, которому они доверили свои данные — сам Джа велел.

Кстати говоря, наибольший интерес представляют провайдеры, предоставляющие как бесплатный, так и платный WEB-хостинг, что позволяет нам зарегистрировавшись под левым ником атаковать сайты серьезных организаций. Провайдеры, предоставляющие чисто халявный хостинг неинтересны по той простой причине, что серьезные пользователи обходят их стороной, а радоваться как круто ты взломал домашнюю страничку Васи Пупкина — ну какой в этом кайф?! Соответственно, чисто коммерческие провайдеры (не имеющие тестового входа) идут лесом, поскольку регистрация аккаунта не только требует денег, но и в большинстве случаев раскрывает наше инкогнито, а вот это уже реально нехорошо. Спрятать свой IP-адрес можно (и нужно) тысячами способов, но скрыть личность отправителя платежа – довольно проблематично даже с использованием Web-Money или прочих электронных кошельков, которые на самом деле никакие не анонимные. Другой вопрос, что искать хакера, не причинившего реального вреда, никто не будет, но… никаких гарантий у нас нет, впрочем, мы сильно отклонились в сторону от основного разговора, а ведь взлом аккаунтов не всегда проходит легко и приятно.

Рисунок 6 атака на сервер под надзором админа

shit happens

Мелкие провайдеры зачастую держат на одной машине и WEB- и (My)SQL- сервер, и диск у них всего один (в смысле логический диск, а физически это может быть и многодисковый RAID). Базару нет — сломать их сможет даже пионер, правда, если движок

сайта основан на (My)SQL, то трем лыжи салом. Или готовим скипидар. И грызем лед зубами, а все потому, что база данных это довольно сложное сооружение, живущее по своим законам и удаленные записи не возвращаются в пул свободного дискового пространства. Другими словами, до них добраться мы не сможем.

Однако, целиком весь сайт в (My)SQL'е никто не держит и множество скриптов расположено в дисковых файлах и эти файлы редактируются, затачиваясь под себя и под конкретный замысел дизайнера. Именно в этих файлах чаще всего и встречаются жестко прошитые пароли на доступ к базе данных, которая (при знании пароля) доступна не только владельцу аккаунта, но и всем праздно шатающимся хакерам.

Захват аккаунтов по прежнему остается возможным, даже если (My)SQL сервер физически расположен на другой машине, а сама база хранится на диске, к которому у нас нет прямого доступа, да он нам и не нужен!!! Нам вполне достаточно того небольшого количества скриптовых файлов, хранящихся на пользовательских дисках.

А сколько этих дисков?! Зависит от конфигурации сервера, но даже сравнительно небольшие серверы имеют множество жестких дисков – десятки, а то и сотни!!! Если администратор сервера принял решение объединить их в RAID (неважно — программный или нет), то можно считать, что нам крупно повезло, поскольку мы имеет шансы захватить данные _любого_ из пользователей.

Хуже, если диски монтируются на файловую систему, что в каком-то смысле упрощает администрирование и делает систему намного более масштабируемой, однако, в таком случае мы ограничены только тем диском, на котором находятся наши файлы и файлы наших "соседей" по лестничной площадке, хостящиеся на тот же самом физическом диске. Естественно, шансы заполучить интересный контент при этом существенно уменьшаются особенно на BSD системах, которые делят диск на группы цилиндров, группируя файлы, принадлежащие одному пользователю в пределах одной зоны. Количество зон, как правило, невелико и в одной зоне живет множество пользователей, но все-таки… возможности по захвату свободного пространства тают прямо на глазах.

Единственный выход — создавать множество аккаунтов, надеясь на то, что они окажутся расположенными на разных дисках, обеспечивая тем самым покрытие большой площади для захвата данных.

заключение

Безопасность операционных систем и серверов — весьма неустойчивая вещь, подобная снежной лавине. Мелкие ошибки, накапливаясь со временем, постепенно уплотняются и образуют мощные осадочные пласты, готовые прийти в движение, сметая любые преграды на своем пути.

Самые коварные ошибки проектирования — те, которые неоднократно обсуждались, все о них как бы помнят, но в то же время — уже давно забыли. Захват свободного пространства посредством позиционирования — как раз к таким ошибкам и принадлежит. Казалось бы, проблема не стоит выведенного яйца и была решена еще в незапамятные времена, но… это в теории. А на практике в эксплуатации находится огромное количество серверов, работающих на уязвимых операционных системах и допускающих удаленные атаки по одному из сценариев, описанных выше.