Создавая новое поколение - часть 3
Безопасность и файлы
Николас Блэхфорд (Nicholas Blachford), Понедельник, 01 Ноябрь 2004, 00:27

Безопасность

<напыщенность режим="">Мне сильно не нравится тенденция индустрии ПО винить пользователей в проблемах в безопасности: это отмазка, уход от ответственности Проблемы безопасности созданы компьютерной индустрией и ей их и решать. Вирусы, трояны и другие неприятности существуют уже много лет, так как пользователи могут быть виноваты, когда производители не сделали ничего видимого, чтобы их защитить? Владелецу автомобиля не надо знать о внутреннем устройстве его машины для того, чтобы ее водить, почему же мы должны знать о том как работают внутренности компьютера? Безопасность это непростая тема [Security], но это не извиняет обвинений пользователя.</>

Если надо загружать заплатки, система должна, по умолчанию, проверять их наличие ежедневно. Конечно, если система была изначально нормально спроектирована, то много заплаток не понадобится. Microsoft предупреждали о потенциальных проблемах в безопасности годы назад, они что-нибудь сделали для этого? Чья это ошибка? Уже говорилось, что существуют более защищенные ОС, нежели Windows, особенно ОС, основанные на Unix, такие как Linux / *BSD и OS X (несмотря на ловко замаскированныепопытки маркетологов сказать иначе).

В любой новой системе безопасность должна быть встроена с самого начала. Модель безопасности должна предполагать, что все хотят нанести системе ущерб, и уменьшать потенциальные возможности программ делать это. Я не верю в создание стены, так как ее можно пробить, но я верю в то, что множество стен преодолеть гораздо сложнее, и это предоставляет много возможностей по отражению атаки, а значит защищенность должна быть во всех частях системы. Ни на одну часть нельзя полагаться. Все части системы должны быть по умолчанию защищены.

Проверка на вирусы
Сканер вирусов, который должен проверять все, должен быть стандартной частью системы, а не дополнением. Даже если вирусов (или вирей) нет, система должна сканировать приходящие файлы, чтобы проверить, являются ли они или их часть исполняемым кодом.

Все новые файлы - в песочницу
Все файлы, не только присоединенные к электронным письмам или скачанные из веба, таким образом они не смогут нанести никакого вреда. Если исполняемый файл ни разу не запускался, сразу же его помечать таким, чтобы когда его запустят, его можно было сперва поместить в "". FreeBSD умеет это делать через Jail (тюрьмы) [Jail].

Можно пойти дальше и помещать в песочницу все и всегда, активно выискивая что-либо для помещения в песочницу. Например, если приложение пытается скачать исполняемый файл и запустить его в памяти, система должна либо запретить это, либо отреагировать на это помещением в песочницу.

Не позволять программам удалять все файлы

Также, нельзя позволять программам иметь доступ к резервным копиям файлов (смотри секцию "Файловая "). Это не даст вирусу или заблудшей программе удалить все файлы в вашем домашнем каталоге. Если она пытается это сделать, файлы должны перемещаться в резервные копии, а система должна следить за файловой системой, выявляя такое поведение, предупреждая пользователя об этом факте, и предагать ему возможность восстановления файлов или запрещения приложения, или еще какие-нибудь действия для специальных файлов. Удаление резервных копий должно быть привилегией, которую имеет только пользователь - ни одно приложение не должно иметь такой возможности.

Такой сценарий возможен практически во всех операционных системах, и это случилось со мной в Linux, в далеком 2000-м году. Альфа-версия браузера Opera упала и удалила содержимое моего домашнего каталога, не стоит и говорить, что я был не очень счастлив узнать об этом. Я ожидал, что альфа-версия будет нестабильна, без некоторых функций, но я не ожидал (и не ожидаю), что она может вычистить мой домашний каталог. Если приложение было настолько плохо сделано, что смогло сотворить это, подумайте, что может сделать программист со злыми намерениями. Сегодня приложения по умолчанию могут вызывать столько разрушений в вашем домашнем каталоге, сколько им хочется. Система не должна этого допускать.

Автоматически идентифицироват все файлы

Предупреждать пользователя когда идентификация дает сбой, если это текстовый идентификатор - изменить его (но сообщить об этом пользователю). Текстовые идентификаторы файлов могут быть обмануты и обманываются, поэтому на них нельзя более полагаться.

Не запускать сервисы от имени Root

Этого невозможно достичь на большинстве систем, основанных на Unix, из-за архитектуры ядра. А одно из преимуществ микроядра, в том, что нет необходимости запускать сервисы от имени root [Microkernel], что уменьшает возможности сервисов сделать что-либо зловредное.

Ограничить внешний доступ

Несмотря на то, что наша система ориентирована на применение в настольных системах, нам могут понадобиться некоторые серверные возможности, например, возможность контролировать систему извне. Такой доступ должен осуществляться только через зашифрованный канал и лучше, если бы это могло делать только специализированое приложение. Конечно, будут возможны и некоторые нешифрованые соединения, например, web или FTP сервера.

Непрямой запуск программ

Программа, которая запускает другие программы, не должна иметь возможности сделать это напрямую, запуск должен проходить через движок взаимодействия (описан в дальнейшей части этой серии), который запускает программу. Это отдает жесткостью ограничений, но делая запуски программ непрямыми, мы обеспечиваем то, что после завершения они не возвратятся в терминал, а значит атакующий, вызывая обвал программы, не сможет получить к нему доступ удаленно.

Экспортируемые сервисы

Как только я запускаю веб-сервер, он автоматически становится виден всем снаружи, если только он не был закрыт файрволлом. Я предлагаю добавить в нашу систему "" так, чтобы только те сервисы, которые были специально "" через этот туннель, были видны внешнему миру. Это означает, что веб-сервер придется специально экспортировать, прежде чем он будет доступен снаружи. Туннель также сможет вести мониторинг, так что можно будет отслеживать и смотреть на то, что в систему приходит и уходит, и если будет работать что-то не то, можно будет запретить ему экспорт (автоматический экспорт самими программами должен быть запрещен).

Однако, вопрос о том как реализовать туннель по прежнему открыт. Одна из идей - обеспечить два сетевых стека: внутренний стек будет взаимодействоваь с программами в системе, а внешний стек будет работать с внешними сетевыми интерфейсами. Туннель будет сидеть посередине, соединяя два стека. Вы можете запустить маршрутизатор, файрволл, NAT (Network Address Translation) и другие сервисы на внешнем стеке, в то время как внутренний стек будет полностью изолирован от интернета. А когда вы соедините их, вы сможете идти через NAT, который сам по себе добавляет еще один слой безопасности.
[newpage]

Безопасность в файловой системе

Необходимо использовать "" резервное копирование файлов, а доступ к таким файлам должен быть только у системы. Ни одно приложение не должно видеть эти копии. Только пользователю через системные инструменты должно быть позволено ими управлять (например, удалять конфиденциальны данные). Точно так же должна существовать полная резервная копия ОС, тогда, при обмане атакующим механизмов контроля, не допускающих изменения системных файлов, система сможет обнаружить несоответствие и заменить измененные файлы корректными версиями. Файлы невидимой резервной копии ОС не должны показываться никому, даже пользователю.

Безопасность на уровне кода

Еще один метод повышения безопасности - улучшить код системы, провести аудит [Audit] кода для выявления ошибок и просто плохо написанного кода. Лучше код - меньше ошибок, а чем меньше ошибок, тем более стабильна и защищена система.

Другой метод улучшения безопасности - принудительная проверка корректности или на уровне языка, или во время выполнения. На некоторых языках программировани написать небезопасный код очень легко [BadCode], но единственный метод улучшения этой ситуации - добавление дополнительных модулей во время компиляции или выполнения, которые проверяли бы программу на корректность исполнения. Perl уже может это делать в режиме "Taint" [Taint], однако, это также возможно и для языков вроде C [code].

Такие приемы можно расширять и далее - если в библиотеку включены потенциально опасные функции, почему бы их не удалить, не заменить их, так чтобы они перестали быть частью системного API?! Сканировать исполняемые файлы на вызовы сомнительных функций, и при их наличии - отказывать в запуске программы. Все это сделает портирование программ превосходным геморроем, но программы от этого станут только лучше.

Попытка модифицировать и приспособить существующую систему к такому уровню безопасности будет непроста и, скорее всего, поломает работу приложений и расстроит пользователей. Как раз один из плюсов начала с нуля - некого расстраивать. Кстати, будет

интересно посмотреть на реакцию рынка на Service Pack 2 для Windows XP.

Безопасность это непростой вопрос, но так ограничивая настройки по умолчанию и делая параноидальную систему, которая постояннно проверяет саму себя и все, что в нее приходит, мы делаем жизнь атакующих гораздо сложнее. Закрытие кода системы от незначительных изменений и сведение количества ошибок к минимуму означает, что защищенность встроена прямо в систему, а не добавлена после, в результате чего она уже не никогда не сможет быть также эффективна. Это тоже работает, OpenBSD [OpenBSD] использует эту философию и некоторые из этих техник уже годами.

Конечно, продвинутым пользователям не понравится такая система, но вполне возможно добавить инструменты для того, чтобы дать разработчикам и продвинутым пользователям больше доступа и контроля над системой.

[b]Файловая система[/b]
Естественно, что для нашей новой платформы мы должны использовать лучшую файловую систему, которую мы можем создать. В идеале она должна иметь некоторые из этих характерстик / функциональностй:

64-битная
Журналируемая
Не фрагментирующаяся
С автоматическим анализом и идентификацией
Произвольные метаданные для файлов с индексацией для обеспечения высокоскоростноо
Поддержка "" поисков, она должна отслеживать изменения в файловой системе и сообщать заинтересованны в этом приложениям об изменения

Может показаться, что это значительно усложненная файловая система, но я только что описал файловую систему "BFS", которая была представлена этому миру в BeOS. Сейчас очень много шума вокруг OS X "Tiger", которая добавляет технологию поиска и Windows Longhorn, которая будет использовать базу данных ([i]статья написана в июле 2004-го, когда еще не было известно, что Microsoft не включит WinFS в Longhorn - примечание переводчика[/i]) в качестве файловой системы (подход, который пыталась использовать Be, но позже оставила его из-за низкой производительноти). Это еще один пример того как существующая технология распространяется по индустрии [Indexing], несмотря на то, что она и является старой новостью для некоторых из нас!

Однако, я не думаю, что Be или пользователи BeOS когда-либо полностью использовали возможности BFS и, учитывая это, поддержку ее возможностей можно будет лучше интегрировать в систему. Похоже, что очень похожая технология будет в OS X Tiger, но также гораздо лучше интегрированная и, поэтому, более полезная для конечного пользователя.

Gnome Storage [Storage] хочет пойти дальше и переводить файлы при сохранении так, чтобы можно было читать их различными программами без необходимости поддержки ими различных форматов. Сходная функциональност была доступна через Datatypes в Amiga и Translators в BeOS, хотя эти технологии использовались приложениями напрямую. Я не уверен, что мне нравится идея встраивания этого в файловую систему, так как это будет

влиять на производительноть при каждом чтении / записи, а использование XML означает преобразования в / из нелинейных структур, компьютеры в принципе плохо с этим справляются (вбрасывание на эту проблему более скоростных ЦП не поможет, так как это проблема латентности памяти).

[b]Поиск: искать лучше[/b]
Я только что перенес картинки с USB флэшки и хочу отредактироватьих. Мне придется или искать их и потом открывать редактором, или открыть редактор и искать самому через меню открытия файла. Можно упростить эту задачу, добавив несколько новых кнопок использующих функции поиска в это меню, нажмите "последние ", и они появятся. Вам больше никогда не придется отправляться на поиск файлов.

Конечно, это будет работать не всегда, для того, чтобы метаданные работали как надо, их необходимо добавлять к файлам и существуют пределы автоматизации этого процесса. Добавление метаданных будет необходимо делать, хотя бы частично, самостоятельно, а это становится проблемой пользовательскоо интерфейса. Никто не будет добавлять метаданные для каждого файла, так что должно появляться окно с набором параметров, которое поможет этому процессу, а для выбора этих параметров будет важен контекст и тип файла. Также система может проанализироват предыдущие действия и составить список параметров на их основе.

Этот инструмент не так сложен, как может показаться, если я загружаю изображение, он может мне дать на выбор "Астрономию", "", "", "", "", "" и так далее. А если он действительно умен, то он сможет посмотреть на веб-сайт, откуда пришло это изображение и попробовать догадаться сам, например, все с сайта NASA скорее всего попадет в категорию "Астрономия" (это потребует от браузера сохранения URL как метаданных вместе с сохраненным изображением).

Потенциально метаданные также можно будет использовать для некоторых не столь очевидных вещей. Если вы запустите вторую ОС, что позволяет эта система, она будет ожидать увидеть файлы на положенных им местах. Но стуктура вашей файловой системы может отличаться от той, что хочет видеть альтернативная ОС, а вам, например, не хочется добавлять новый раздел на жесткий диск. Тогда вы сможете использовать метаданные для создания виртуальной файловой системы, записать желаемый путь как метаданные, но положить файл в папку с файлами для второй ОС. Когда вторая ОС захочет получить доступ к файлу, файловая система отдаст файл из этой папки, найдя его по этим метаданным.

Это также несет в себе интересные возможности в области безопасности, так как вторая ОС не сможет напрямую прочитать каталоги, то даже если она попытается попросить что-либо другое, она никогда никуда не доберется, просто потому что других файлов с корректными метаданными не существует.

Также можно будет создавать вид файловой системы из метаданных, проделывая описанное выше в обратном порядке. Мы сможем сделать то же самое с обычными метаданными. Возвращаясь к моему примеру с загрузкой, давайте предположим, что я сохранил несколько забавных и астрономическихизображений в своей папке для скачиваемых файлов. Одной командой можно будет переместить их в правильное место, например, <tt>home/nick/pictures/humour</tt> или <tt>home/nick/pictures/astronomy</tt>.

Я бы хотел увидеть и другие функции в файловой системе, такие как автоматическое резервное копирование, технология, используемая "большим " уже годами. Опять же, метаданные тут могут помочь - хранить резервные копии в других местах с изначальным путем в метаданных. Это предотвратит хранение в каталогах нескольких копий одного файла, которые быстро перестают читаться.

Поиск также может быть улучшен хранением частей файлов в метаданных. Система сможет искать файлы, содержащие необходимую строку, но для ускорения процесса сначала смотреть в метаданных (именно это делает Apple в Tiger)
[newpage]
[b]Управление файлами[/b]
Безусловно, всеми этими файлами по прежнему необходимо управлять. В те дни, когда я использовал Amiga, одним из самых полезных приложений был двухпанельный файловый менеджер. Их существовало множество, самые известные - SID или Directory Opus v4 [DOpus4] (примечание: не более поздние версии, в который вид изменился). Я еще не видел эквивалентов на любой другой платформе, несмотря на то, что кое-что есть для Unix/Linux [LinOpus]. Вы также можете симулировать двухпанельный менеджер, расположив 2 окна Finder рядом друг с другом в OS X 10.3, тогда можно будет использовать перетаскивание мышкой вместо кнопки копирования (похожим образом у меня постоянно настроены два окна Finder'a).

Двухпанельный подход очень прост в использовании и, при хорошей реализации, он позволяет вам ходить по системе с хорошей скоростью. Я никогда не видел лучшего подхода к управлению файлами, хотя вы, конечно, можете использовать навигаторы (пространственны или браузерные) или даже командную строку, но я не вижу других способов, которые работали бы также хорошо или также быстро.

Конечно, теперь мы можем улучшить его, добавив поддержку перетаскивания мышкой и метаданных. Можно будет сделать кнопку "", по нажатию которой будут отображены все музыкальные файлы, в независимости от того, где они находятся. То же самое можно сделать и с картинками или другими форматами данных. Действия можно будет расширить, двойной щелчок на файле будет открывать его просмотрщиком / проигрывателем, двойной щелчок на каталоге будет перенесет вас к местоположению файлов, перетаскивание каталога на другую панель откроет ее там.

Исторические и иерархические списки позволят вам ходить вперед-назад по предыдущим и другим местам. Добавление возможности навигации через меню позволит быстрее перепрыгивать в другие части системы.

Сложно описать, насколько мощна, но и проста, будет такая программа. DOpus и его последователи очень хороши для своих задач, замечу, что я никогда не использовал браузеры или командную строку. В комбинации с поиском по метаданным эти инструменты станут еще лучше.

Загрузка и сохранение файлов может также использовать некоторые подобные технологии (но не двойные панели). Стандартный диалог запроса файлов позволит быструю навигацию, сможет осуществлять поиск по метаданным и предлагать лучшее расположение для

открытия или сохранения файла.

[b]Другие случайные идеи[/b]
При создании новой ОС можно также опробовать некоторые новые концепции, вот несколько из них:

[b]Большие размеры страниц[/b]
Существующие операционные системы до сих пор страдают пережитками той аппаратуры, для которой они изначально создавались, например, управление памятью, как правило, осуществляется страницами по 4 Кб. На сегодняшний день с нынешней аппаратурой, с ее сотнями мегабайт памяти для пользовательски настольных машин и гигабайтами для рабочих станций, это выглядит слишком маленьким размером. Я бы предложил использование гибкого размера страницы, начиная с 32 Кб, этот подход имеет минусы использования больших блоков памяти, но их можно уменьшить располагая небольшие запросы на выделение памяти в уже размещенных блоках. Это можно реализовать встраиванием маленького менеджера памяти в приложения прямо во время их выполнения. Когда приложению понадобится память, менеджер памяти активируется и проверит есть ли достаточное количество памяти в его выделенных блоках, а если нет - попросит ОС выделить еще.

Это не только сделает использование памяти более эффективным, но также уменьшит количество вызовов ОС и переключений контекстов. Использование больших страниц также уменьшит размеры каталога страниц, так что б[b]о[/b]льшая его часть сможет разместиться в ЦП, а поиски по каталогу страниц смогут производиться без таких частых обращений к основной памяти. В реальности, если отключить пейджинг на диск (вполне возможно при сегодняшних объемах памяти), то станет возможным вместить весь каталог страниц в ЦП, что даст прирост производительноти, так как он никогда не будет сбрасываться.

[b]Grid[/b]
Media kit из BeOS позволяет вам свободно перенаправлять вывод из медийных программ, это, например, дает возможность добавить аудио фильтр или эффект к MP3 плееру, даже если MP3 плеер не поддерживает аудио эффектов.

Grid - это то же самое, но расширенное на всю систему или даже среди нескольких систем. Если я захочу сохранить файл на другом компьютере, как правило для этого мне необходима специальная поддержка этого в файловой системе. Grid позволит этому механизму быть прозрачным, еще один диск появится в вашем окошке выбора места сохраняемого файла, вы сможете выбрать диск и нажать "", как обычно. Файловая система сохранит файл как обычно, а Grid пошлет данные на другой компьютер.

Grid также сможет расширить существующий media kit, позволяя переместить часть обработки на другую систему. В принципе, можно будет даже распределять программы по нескольким системам. Сделайте приложение или сервис ОС, который будет знать, как общаться через Grid, а система возьмет на себя заботу о передаче сообщений. Grid позволит обходиться без передачи данных по всему сетевому стеку, так как Grid сам по себе работает как простая сеть.

Это опять же возвращает нас к идее простоты, вместо наличия множества приложений, дающих функциональност, она располагается в ОС, так чтобы все могли ее использовать.

Похожий принцип используется в последователе Unix, "Plan 9" [Plan9], Grid является хорошим воплощением концепции Plumbing [Plumbing] из Plan9.

[b]Заключение[/b]
Обдумывая новую систему всегда приходится находить различные компромиссы, все мы хотим иметь хорошую производительноть, но можем ли мы ею пожертвовать для лучшей безопасности и стабильности? Для сегодняшних систем этот вопрос задавался годами или даже десятилетиями назад и ответ был "". Сегодняшние процессоры обеспечивают 99.9% людей большей мощностью, чем они могут использовать, а с приходом многоядерных процессоров на настольные машины у нас будет еще гораздо больше мощи. Сегодня, с учетом все возрастающего числа вирусов и атак, я бы пожертвовал производительнотью, вряд ли кто-либо, кроме фанатов тестов, заметит разницу. Я готов в одиночестве использовать систему, которую практически невозможно взломать и практически невозможно обрушить.

Описываемая мною система не так отличается новыми идеями, сколько тем, что берет лучшие идеи, как старые, так и новые и сочетает их вместе, чтобы сделать что-то лучше всего, что было до нее. Создание новой системы позволяет нам это, но так как она дает возможность одновременно запускать другие ОС, мы получаем комбинацию, которая дает нам все лучшее из обоих миров. Однако, мы можем поэкспериментирвать и за границами архитектуры ОС, мы можем исследовать новые области, улучшать и их. Достаточно сказать хотя бы, что даже сегодня очень немногие, если вообще кто-либо, по-настоящему поняли что такое удобство использования. Это непростой вопрос, больше вопрос мнений, а не абсолютных критериев. Mac'и известны тем, что они легки в использовании, и как раз по этому поводу...

[b]Исправление:[/b]
В первой части, в моем описании Macintosh'a, я не упомянул удобство его использования как заслугу Джефа Раскина (Jef Raskin), что было главным в его первой версии этой машины. Позже это было исправлено, прошу прощения за ошибку.

37 лет назад Джеф Раскин был первым, кто представил миру концепцию удобства использования ([i]usability[/i]), а миру потребовалось много времени, чтобы ее понять. В четвертой части я опишу, как мы можем сделать нашу новую ОС более удобной.

[b]Ссылки / Подробная информация[/b]

[Security] [link=http://cbbrowne.com/info/security.html]Безопасность[/link]

[Jail] FreeBSD
[link=http://www.freebsd.org/cgi/man.cgi?query=jail&apropos=0&sektion=0&manpath=FreeBSD+5.2-RELEASE+and+Ports&format=html]Jail[/link]

[Microkernel] [link=http://cbbrowne.com/info/microkernel.html]Микроядра[/link] имеют преимущества в безопасности и стабильности.

[Audit] Разработчики OpenBSD используют
[link=http://www.openbsd.org/security.html#process]метод аудита кода[/link] для того, чтобы сделать свою систему более защищенной.

[BadCode]
[link=http://acmqueue.com/modules.php?name=Content&pa=showpage&pid=160&page=1]Встраивание[/link] защищенности в языки программировани.

[Taint] В Perl есть режим
[link=http://searchsecurity.techtarget.com/tip/1,289483,sid14_gci892449,00.html]Taint[/link].

[Code]
[link=http://acmqueue.com/modules.php?name=Content&pa=showpage&pid=160&page=3]Предложения[/link] по улучшению безопасности встраиванием модулей во время исполнения.

[OpenBSD] [link=http://www.openbsd.org/]OpenBSD[/link] имеет репутацию безопасной системы.

[Indexing] Описания [link=http://www.gnome.org/%7Eseth/blog/document-indexing]различных систем[/link] индексирования документов.

[Storage] Новое в [link=http://www.gnome.org/%7Eseth/storage/features.html]Gnome Storage[/link]

[DOpus4] [link=http://www.uae.rzeszow.net/ekrany/dopus.png]Скриншот[/link] Directory Opus 4.

[LinOpus] [link=http://www.boomerangsworld.de/worker/]Worker[/link] - двухпанельный файловый менеджер для *nix.
[link=http://www.boomerangsworld.de/worker/pics/worker-shot11.png]Скриншот[/link].
Еще один называется [link=http://www.obsession.se/gentoo/]Gentoo (не дистрибутив Linux)[/link], также для *nix. [link=http://www.obsession.se/gentoo/screenshots/main.png]Скриншот[/link].

[Plan9] [link=http://www.cs.bell-labs.com/wiki/plan9/overview/index.html]Обзор[/link]
[link=http://www.cs.bell-labs.com/wiki/plan9/1/index.html]Plan9[/link], развития идей Unix.

[Plumbing] Концепция [link=http://plan9.bell-labs.com/sys/doc/plumb.html]Plumbing[/link] из Plan9.

[b]Об авторе:[/b]
[link=http://www.blachford.info/]Николас Блэхфорд[/link] (Nicholas Blachford) - 33-х летний эмигрант из Британии, ныне живущий в Париже, но не говорящий по-французски (еще). Интересуется различными причудливыми темами (Аппаратное обеспечение, программное обеспечение, фотография) и всякими другими вещами, особенно теми, в которых используются новые технологии. На сегодняшний день безработный.