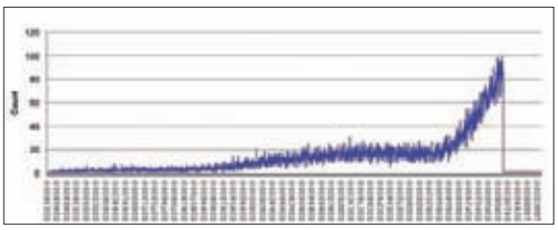
ПОРОЧНОЕ НАСЛЕДИЕ WINDOWS

Концептуальные методы взлома Linux через флешку и защита от них

Всем известна истерия виндузятников по поводу флешек с вирусами. Линуксоиды же смотрят на все это с ухмылкой и чувством полного иммунитета. Но как выяснилось, рано расслаблять булки: и в линуксах существует не меньше косяков по этому поводу.

Autorun.inf мертв, да здравствует .autorun!

Полагаю, все в курсе, что в системах Windows существует возможность автоматически запускать программы после втыкания usb-флешки и прочих внешних накопителей. За эту функциональность ответственен файл autorun.inf, с помощью которого юные вирусописатели наломали немало дров, прежде чем мелкомягкие отключили автоматический запуск. Теперь, начиная с Windows 7, этот функционал отключен по умолчанию. Но хитрые хакеры не унывали и находили новые способы натягивать пользователей. Об этом свидетельствуют нашумевшие уязвимости при обработке ярлыков (эту брешь эксплуатировал червь Stuxnet) и разнообразных библиотек создания эскизов, которые вступают в действие при простом просмотре содержимого флешки в «Проводнике». Было бы наивно предполагать, что в Линуксе подобной функциональности нет. Спецификации freedesktop.org, которым следуют такие окружения рабочего стола, как GNOME и KDE, предполагают наличие специальных файлов автозапуска на съемном накопителе: .autorun, autorun или autorun.sh. Если на флешке присутствуют два или все три из перечисленных типов файлов, то обрабатываться будет только один. Приоритет будет отдан файлу .autorun, а далее — в порядке, перечисленном выше.



После 40 960 запусков evince-thumbnailer получился такой график распределения адресов загрузки libc

Краткий ликбез по средствам безопасности в Ubuntu

1. AppArmor — модуль для ядра Linux, который реализует принудительный контроль доступа для приложений. С каждым приложением может быть ассоциирован специальный профиль, который ограничивает его телодвижения. В Ubuntu некоторые программы при инсталляции также устанавливают свой профиль для AppArmor. Дополнительные профили содержатся в пакете apparmor-profiles. Часть из них устанавливается по пути /usr/share/doc/apparmor-profiles/extras, поэтому может потребоваться перенос в /etc/apparmor.d.

2. ASLR (Address Space Layout Randomization) — технология рандомизации адресного пространства. С ее помощью загрузчик ELF для каждого нового процесса будет задавать разные адреса стека, кучи, подгружаемых библиотек и так далее. Значение /proc/sys/kernel/randomize_va_space соответствует включенности (1 или 2) или выключенности (0) ASLR. С 2005 года (ядро 2.6.12) Linux включает в себя простую реализацию ASLR. Различные патчи безопасности (PaX, ExecShield и другие) реализуют более сложные и полные варианты ASLR. В дистрибутивах, содержащих в названии «Hardened», а также в современных версиях Ubuntu сильные варианты включены по умолчанию.

3. PIE (Position Independent Executables) — специальные флаги сборки приложений «-fPIE –pie». Собранные с такими флагами приложения могут использовать все преимущества ASLR и будут каждый раз загружаться по разным адресам. Не рекомендуется использовать на 32-битных системах, так как наблюдается весьма заметное снижение производительности (до 10%).

4. NX бит (No eXecute Bit) — один бит в процессорном регистре, который обозначает, что находящимся в отдельных зонах памяти данным исполняться запрещено. Данная технология может работать только при соблюдении следующих условий:

- используется процессор, поддерживающий технологию на аппаратном уровне (начиная с Intel Pentium 4 серии 6xx и всех модификаций AMD Athlon 64);
- используется PAE или архитектура x86-64 (в этих режимах доступен бит запрета исполнения в таблице страниц).

По сути, это некий аналог autorun.inf в Windows, только несколько урезанный в возможностях. Здесь можно только прописать путь к исполняемому файлу, но нельзя — сразу к нескольким, а также нельзя выходить за пределы файловой системы флешки с помощью ссылки на вышестоящий каталог. Кроме того, для неисполняемых файлов (например, документа pdf) тоже существует возможность автооткрытия. Нужно создать файл .autoopen или autoopen в корневой директории съемного устройства и прописать туда путь до нужного файла, причем корнем является не корень системы, а корень съемного устройства. Правда, в том же Nautilus эта функциональность пока (намеренно?) не реализована. Единственная омрачающая картину автозапуска вещь — пользователю необходимо подтвердить это действие. Но такой ли это минус? Слепая уверенность линуксоида в своей безопасности от малвари может сыграть с ним злую шутку.

Количество опубликованных уязвимостей в Evince заставляет задуматься

Копаем глубже

Надо признать, что фокусами с автозапуском нынче мало кого удивишь, поэтому взглянем на предметную область несколько шире. После втыкания флешки (или любого другого устройства) исполняется большое количество кода:

- модули подсистем USB, eSATA, FireWire, PCMCIA;
- модули файловой системы режима ядра (ext3, ext4 и другие);
- модули файловой системы пользовательского режима (ntfs-3g);
- библиотеки создания эскизов (через файловый менеджер).

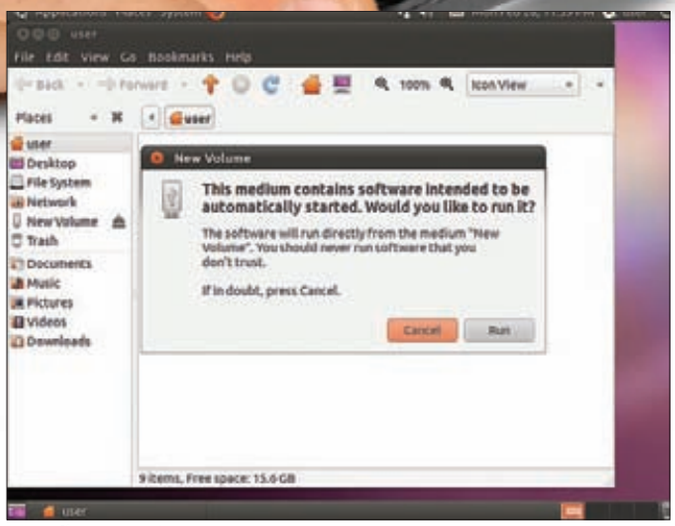
Ошибки могут существовать на любом из этих уровней. Например, в 2009 году был найден баг в драйвере VoIP телефона Auerswald (CVE-2009-4067). Он заключался в неправильной обработке USB-дескрипторов и приводил к классическому переполнению буфера. В результате эксплуатации этой бреши атакующий мог выполнить произвольный код на уровне ядра. Для более удобного поиска ошибок в USB-драйверах Тобиас Мюллер создал фаззер, основанный на QEMU и позволяющий эмулировать USB-устройства. В конце 2009 была исправлена серьезная уязвимость в модуле файловой системы ext4, в функции ext4_decode_error(), которая приводила к разыменованию NULL-указателя и выполнению произвольного кода при монтировании специально скомпонованного образа ФС. Уязвимости в драйверах файловых систем используются при помощи специально сформированного образа этой файловой системы, залитого на флешку. Успешно проэксплуатированная уязвимость дает атакующему полный рутовый доступ к системе, так как драйверы файловых систем исполняются в режиме ядра. Эксплуатация уязвимостей в драйверах файловой системы пользовательского режима (через FUSE) дает нам возможность исполнять код с привилегиями того пользователя, от чьего имени был подмонтирован раздел.

links

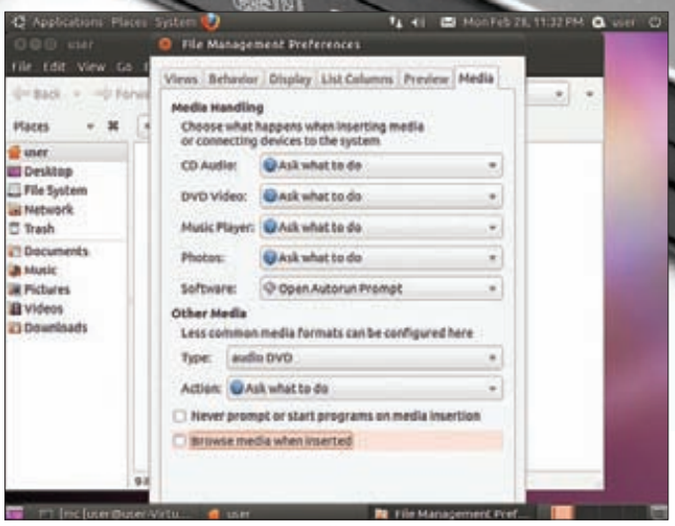
- Спецификации, в которых кто-то додумался описать автозапуск с флешек: goo.gl/2wIIA;
- база уязвимостей, на которую чаще всего ссылаются эксперты: cve.mitre.org/cve/cve.html;
- видеозапись выступления Джона Ларимера: youtube.com/watch?v=ovfYBa1EHm4;
- полный список средств защиты, используемых в Ubuntu: wiki.ubuntu.com/Security/Features.

dvd

На прилагаемом к журналу DVD ты найдешь видеозапись и слайды с выступления Джона Ларимера на конференции ShmooCon 2011, которое, собственно, и подняло всю эту панику.



Nautilus как бы говорит, что ответственности за наши действия не несет



Отключаем автоматический просмотр содержимого свежеподключенных разделов

Как искать уязвимости в системных драйверах? Здесь существует несколько путей:

1. Ручной анализ кода. Особое внимание следует уделять тому, как парсятся структуры файловой системы.

2. Автоматический анализ кода при помощи специализированного софта (lint, clang static analyzer и другие).

3. Фаззинг. В Линуксе любое блочное устройство (в том числе файл) может быть смонтировано как раздел. Таким образом, можно легко написать программу, модифицирующую файл, смонтировать его, провести некоторые операции, а потом размонтировать и повторить процедуру заново. Модифицируя файл не абы как, а с учетом структур файловой системы, можно быстрее прийти к успеху (так называемый smart fuzzing).

Наследник Adobe Acrobat Reader

Evince — весьма распространенный просмотрщик PDF-документов. Также поддерживает такие форматы, как PostScript, TIFF, DVI, DjVu. Он является стандартным компонентом рабочей среды GNOME. Зайдя на сайт Common Vulnerabilities and Exposures, можно увидеть, что для Evince открыто девять уязвимостей, причем четыре из них — достаточно новые и датируются июлем 2010 года. Одна из этих уязвимостей (CVE-2010-2640) была продемонстрирована Джоном Ларимером, им же был описан процесс создания эксплойта под нее, который исполнял произвольный код в системе. В данном случае была продемонстрирована возможность «отпирания» залоченной скринсейвером системы Ubuntu 10.10 после простого втыкания флешки (без ведома пользователя и системы исполнялся находившийся на флешке простой скрипт kill.sh с командой «killall gnome-screensaver»). Сама уязвимость заключается в неправильной обработке шрифтов в DVI-файлах. В этих файлах могут подключаться внешние шрифты, путь до которых задается в абсолютном виде (/media/NNN). Вообще говоря, эта уязвимость никоим образом не связана с флешками и может использоваться сама по себе, просто дурная привычка Nautilus открывать окна с новыми подмонтированными системами пришлась очень кстати.

Nautilus под микроскопом

GNOME Nautilus — файловый менеджер, который используется по умолчанию в дистрибутивах Ubuntu, а также в среде рабочего стола GNOME. Он поддерживает большинство спецификаций freedesktop.org и автоматически монтирует известные ему файловые системы на USB-дисках по умолчанию. Чтобы получать информацию о новых подключенных съемных накопителях, Nautilus использует GVFS — виртуальную файловую систему, которая дает возможность монтировать разделы без рутовых привилегий. Смонтированный раздел находится по пути /media/NNN, где NNN — название раздела. Также Nautilus автоматически открывает окно с содержимым смонтированного раздела и генерирует эскизы для каждого файла, находящегося в корневой директории раздела. Причем это происходит даже при работающем скринсейвере и заблокированной системе! Nautilus умеет генерировать эскизы для изображений, видеофайлов, текстовых документов и некоторых других файлов. Иконки для изображений генерируются с помощью стандартной гномьей библиотеки GdkPixBuf, которая в своей работе вызывает функции из других библиотек для различных изображений, таких как libpng, libtiff, libjpeg. Во всех трех библиотеках существуют общеизвестные уязвимости. В начале 2011 года была опубликована уязвимость в библиотеке libpng версии < 1.5.0 (CVE-2011-0408), которая присутствует в функциях png_do_expand_palette() и png_do_rgb_to_gray(). Их реализацию можно найти в исходном файле pngrtran.c. С помощью специально сформированных PNG-, MNG- или JNG-файлов можно аварийно завершить работу использующего эту библиотеку приложения и при некоторых условиях выполнить произвольный код на целевой системе, что и было продемонстрировано Джоном Ларимером на конференции ShmooCon. К сожалению, PoC-эксплойт пока не представлен широкой общественности, зато давно доступен закрывающий этот баг апдейт библиотеки до версии 1.5.1. Летом 2010 года было обнаружено переполнение буфера в библиотеке LibTIFF 3.x, которое приводит к выполнению произвольного кода при попытке обработки TIFF-изображения со специально оформленным блоком тэгов SubjectDistance. Проблема устранена в версии 3.9.4. В ноябре прошлого года была найдена уязвимость в библиотеке для рендеринга шрифтов FreeType версии < 2.4.3, позволяющая выполнить произвольный код в системе при обработке специально сформированного шрифта TrueType GX. Баг в виде переполнения буфера существовал в функции ft_var_readpackedpoints().

Все эти и многие другие нераскрытые уязвимости свидетельствуют о том, что нынешняя реализация библиотек под Linux далека от совершенства в плане безопасности.

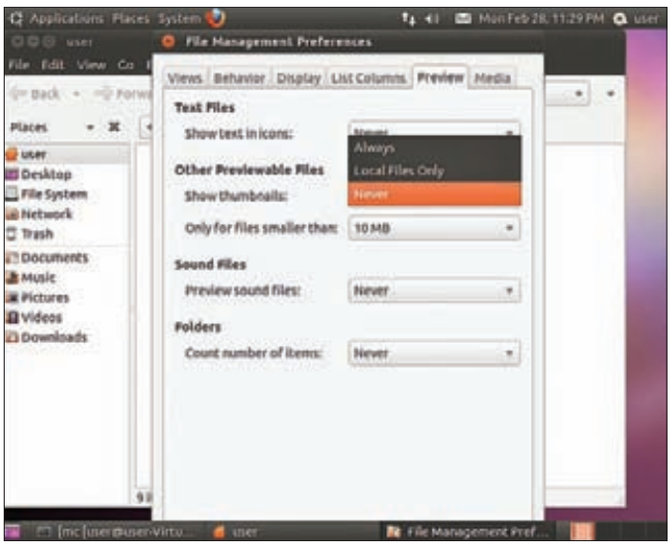
Для формирования эскизов других файлов используются сторонние утилиты:

- evince-thumbnailer — для документов pdf;
- totem-video-thumbnailer — для видео- и аудиофайлов;
- gnome-thumbnail-font — для файлов со шрифтами.

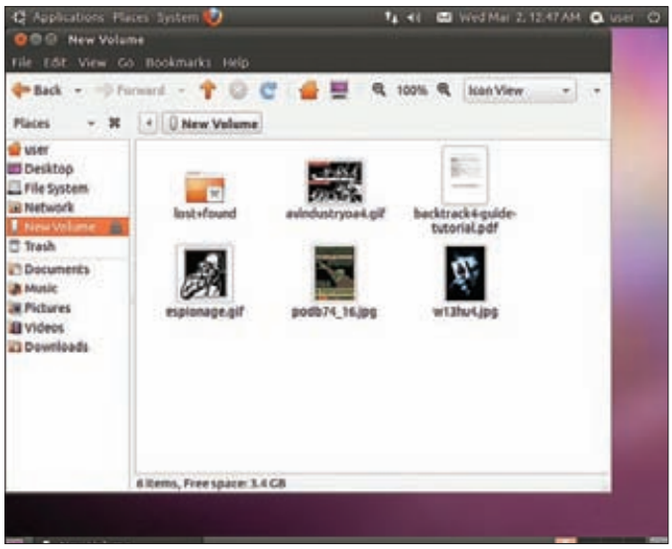
У всех этих утилит схожий синтаксис — например, вызов evince-thumbnailer выглядит так:

```
$ evince-thumbnailer -s 100 /home/user/doc.pdf \
   /home/user/thumb.png
```

Здесь аргумент '-s' задает размер миниатюры в пикселях по горизонтали, следующий параметр — исходный PDF-документ, последний параметр — файл с получившейся миниатюрой. Таким образом Nautilus запускает соответствующую утилиту для каждого файла в просматри-



Отключаем создание эскизов при просмотре содержимого файловой системы



Создание эскизов в Nautilus включено по умолчанию

ваемой директории. Посмотреть список используемых построителей эскизов и ассоциированных с ними типов файлов можно так:

```
$ gconftool -R /desktop/gnome/thumbnailers
```

Стоит заметить, что уязвимости в генераторах эскизов могут использоваться и без участия флешки: достаточно скачать файл из интернета и просмотреть содержимое соответствующей папки в файловом менеджере. Что примечательно, не все генераторы эскизов защищены AppArmor в Ubuntu 10.10, среди незащищенных — totem-video-thumbnailer и gnome-thumbnail-font. Будем надеяться, в будущих релизах Ubuntu это исправят. Тем не менее, вышеописанные утилиты рано или поздно попадут под разнос хакеров, что, судя по последним событиям, уже начинается. Но в той же Ubuntu существует множество встроенных средств защиты, призванных обезопасить пользователей от всех напастей. Посмотрим, везде ли они смогут помочь.

Обход встроенных средств защиты

Дистрибутив Ubuntu по праву признан одним из самых защищенных: тут тебе и AppArmor, и ASLR, и PIE, и использование NX-бита. Последний уже считается неэффективным ввиду появления таких техник атак, как возврат в библиотеку (ret2lib) и возвратно-ориентированное программирование (ROP). В своем выступлении на последней конференции ShmooCon 2011 Джон Лаример показал слабости механизма ASLR/PIE в 32-битном Linux. Он проанализировал адреса, по которым загружается библиотека libc (но это справедливо и для любой другой), и пришел к выводу, что возможно всего около 3 000 вариантов, а в определенных условиях — еще меньше. Причем вероятности нахождения библиотеки по одному из этих адресов не равны, график распределения вероятностей можешь посмотреть на картинке (по горизонтальной шкале разместились адреса, а по вертикальной — количество попаданий в конкретный адрес). Таким образом, можно просто подобрать адрес нужной библиотеки — например, с помощью создания множества pdf-документов, эксплуатирующих баг в evince-thumbnailer. Это становится возможным, так как Nautilus запускает для каждого файла отдельный процесс evince-thumbnailer. Другой интересный и уже почти стандартный механизм защиты — AppArmor — на самом деле защищает лишь настолько, насколько способны его профили, расположенные в /etc/apparmor.d. Например, в Ubuntu 10.10 профиль для evince-thumbnailer позволяет запись в ~/.config/autostart — место, которое может быть использовано вирусописателями для автоматической загрузки их творений (или даже произвольных скриптов) при входе пользователя в систему. От некоторых вещей AppArmor не способен защитить в принципе, поскольку такая защита может нарушить стабильность работы системы:

- вызовы библиотеки X11 (может быть нарушен доступ к сети);
- завершение процесса скринсейвера, перехват нажатых клавиш, эмуляция нажатий клавиш и так далее.

Принимаем меры

Защититься от всех угроз нам в любом случае никогда не удастся, но могу дать ряд полезных советов, которые существенно снизят риски:

1. Своевременно ставь обновления системы. В Linux патчи выходят гораздо оперативней, чем в Windows, где публичные баги могут существовать месяцами. Этим ты обезопасишь себя хотя бы от известных уязвимостей.

2. Отключи автомонтирование съемных накопителей или (более простой вариант) открытие окна Nautilus при автомонтировании. Для этого в Nautilus необходимо зайти в меню «Edit -> Preferences -> Media» и снять галку с опции «Browse media when inserted».

3. Отключи генерацию эскизов в своем файловом менеджере. Это не обязательно должен быть Nautilus, все графические файловые менеджеры используют для этого сходные компоненты. В Nautilus создание эскизов отключается в меню «Edit → Preferences → Preview».

4. Используй AppArmor с расширенным набором профилей, которые можно найти в интернете. Особое внимание следует уделять всяким проприетарным приложениям типа Skype, поскольку патчи к таким приложениям обычно выходят не так быстро, как хотелось бы.

5. Поставь патч к ядру PaX, который не позволяет коду исполняться в стеке, а также не допускает возможности записи в область кода программы — таким образом предотвращается эксплуатация уязвимостей переполнения буфера. К тому же PaX увеличивает количество бит энтропии для ASLR, делая перебор адресов загрузки библиотек практически невозможным.

6. Используй 64-битное ядро, в нем ASLR лишен слабостей, описанных выше. Кроме того, переход на 64 разряда даст твоей системе прирост производительности, во всех современных процессорах поддержка набора команд x86-64 имеется.

7. С осторожностью используй дистрибутив Ubuntu (а лучше вообще не используй), так как он самый распространенный среди линуксов, и новые зловреды, скорее всего, будут затачиваться под него.

Заключение

Как видишь, на деле Linux оказался не таким безопасным, как о нем привыкли думать, особенно это касается настольных и ориентированных на пользователя дистрибутивов типа Ubuntu. Другое дело, что ими пока не особо интересуются вирусописатели: извлечь материальную выгоду достаточно затруднительно из-за небольшого распространения Linux по сравнению с Windows.