# TOP10 ошибок администраторов Server 2003

крис касперски, ака мыщъх, a.k.a nezumi, no-email

*О Windows, сын ошибок трудных!*
народное

**легкость установки и управления MS Server'ом создает обманчивую иллюзию, что администрировать его проще простого. никакой командной строки! и даже обновления устанавливаются автоматически! настоящий рай, да и только! почему же, тогда его так часто ломают?! составив TOP10 ошибок управления Server 2003, и подробно прокомментировав каждую из них, мы надеемся помочь начинающим администраторам (хотя некоторые факты наверняка будут интересны и матерым волкам)**

## *#1 – заплатки и обновления*

Достаточно многие администраторы вообще не устанавливают никаких заплаток, считая, что никто их атаковать не будет, но это всего лишь распространенное заблуждение. Основная угроза исходит от червей, сканирующих IP адреса и выявляющих не залатанные машины, даже если это домашний сервер, на котором нет никаких конфиденциальной информации.

Обновляться все-таки надо, причем обновлять не только операционную систему, но и все используемые приложения: MS Office, Adobe Photoshop и… даже WinRAR, в которых так же обнаруживаются критические ошибки, естественно, не устраняемые обновлениями от Microsoft. Следовательно, нужно составить список используемого программного обеспечения и регулярно посещать сайты производителей на предмет поиска новых заплаток.

Кстати говоря, заплатки от Microsoft содержат одну очень неприятную особенность, граничащую с ошибкой, а именно — они не проверяют номера версий замещаемых исполняемых файлов/динамических библиотек. Допустим, у нас есть две заплатки A и B, исправляющие ошибки в KERNEL32.DLL (например). Поскольку, установщик не отслеживает последовательность обновлений, то установив заплатки в обратном порядке (инсталлятор при этом даже не пикнет!), мы закроем дырку A, но откроем дырку B, поскольку, KERNEL32.DLL (как и любая другая динамическая библиотека) всегда замещается целиком, а не частями!

При автоматическом обновлении никаких проблем не возникает, т. к. заплатки ставятся в том порядке, в котором они выпускаются, но вот если мы сохраняем заплатки на локальный диск или качаем их вручную, то необходимо в "свойствах" файла найти внутреннюю версию и дату его создания, составив "план" последовательности установки заплаток. Впрочем, тут не обходится без подводных камней. Иногда Microsoft выпускает одни и те же заплатки по несколько раз. Допустим, сначала выходит A, исправляющая ошибки E1, E2, E3, после чего выходит B, исправляющая E4, E5, E6, а затем… выходит обновленная заплатка A', исправляющая все те же E1, E2, E3, а обновленной заплатки B — нет. Установка A' поверх уже установленной B открывает дыры E4, E5, E6. Так что с заплатками нужно быть очень внимательными и всегда читать бюллетени безопасности, чего практически никто делать не собирается.

В целях экономии трафика ряд Интернет-провайдеров устанавливает свои собственные сервера обновлений, предлагая клиентам прописать в настройках Windows-Update их адреса. Соблазн на самом деле очень велик, но угроза быть атакованным еще выше! Microsoft прилагает огромные усилия для защиты своих серверов, вкладывая в безопасность нехилые деньги. Что же касается провайдеров, то… атаковать их порядка на три проще (и ведь их атакуют, подсовывая троянизированные обновления).

Выход: скачивайте заплатки _только_ у самих поставщиков, при этом, чтобы избежать вероятности "подмятия" доменного имени с перенаправлением на другой узел, не используйте DNS-сервер провайдера. Установите свой собственный DNS, напрямую обращающийся к корневым доменным серверам по TCP-протоколу и заблокируйте 53-UDP порт на брандмауэре для отсечения подложных DNS-ответов.

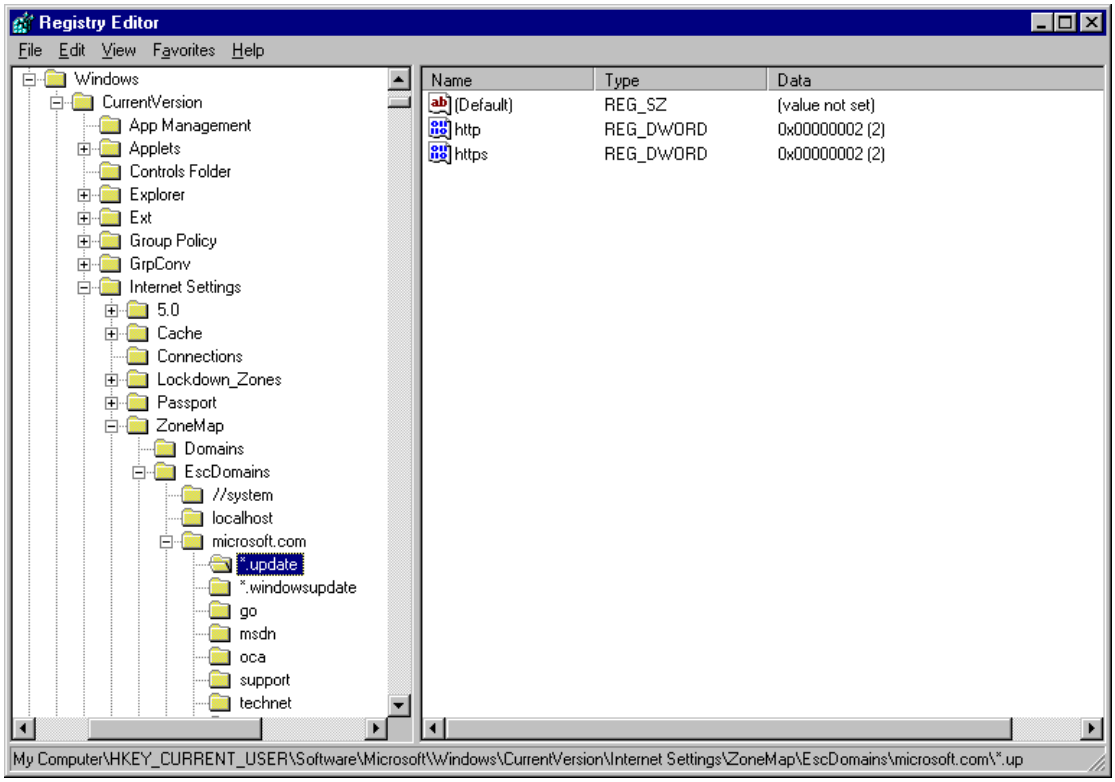


**Рисунок 1 ветки реестра, ответственные за установку адреса сервера автоматических обновлений**

## #2 – баги в процессорах

Ругая Windows (и отчасти Linux) за то, что программное обеспечение наших дней дыряво как ведро без дна, мы почему-то забываем об аппаратной оснастке, считая "железо" совершенно непогрешимым. Увы, процессоры не святые. Самая громкая ошибка в Pentium была обнаружена в 1995 году и продемонстрирована на следующем примере: x - (x/y)*y, результат которого (если только y != 0) должен быть равен нулю, однако, при определенных значениях x и y (x = 4195835, y = 3145727), процессор выдавал… 256! Потрясающая точность, не правда ли?!

Журналисты подхватили сенсацию, вынудив Intel пойти на замену процессоров, чего она изначально делать не хотела, доказывая, что людям, далеким от математики, точные вычисления не нужны, а вероятность проявления ошибки на произвольном (а не умышленно подготовленном) наборе данных близка у нулю.

С тех пор, сообщений об ошибках в ЦП как будто бы не отмечалось. И потому заявление Theo de Raadt'a (ведущего разработчика Open-BSD), что Core2Duo содержит огромное количество ошибок, многие из которых допускают удаленный захват управления стало очередной сенсацией года ("*Intel understates the impact of these erraata [sic!] very ignificantly. Almost all operating systems will run into these bugs*" – http://marc.info/?l=openbsd-misc&m=118296441702631).

Часто ошибок может быть исправлена программным путем (и разработчики Open-BSD сделали это, чего нельзя сказать о лагере NT-подобных систем), часть — обновлением микрокода процессора (для чего, в свою очередь, необходимо обновить версию BIOS, если только разработчики прошивки включили в нее обновленный микрокод), но все эти меры лишь уменьшают вероятность атаки, а оставшиеся ошибки исправляются исключительно сменой процессора на более новый (кстати говоря, так же содержащий ошибки, перечисленные в секции "errata" обновленной спецификации от Intel, распространяемой на свободной основе).

Выход: почаще обновлять прошивку BIOS, в критических случаях использовать Open-BSD, разработчики которой прилагают все усилия для исправления ошибок ЦП, а еще лучше не использовать Core2Duo, поскольку, это все-таки "бытовой" процессор, не отвечающий жестким требованиям серверной индустрии.

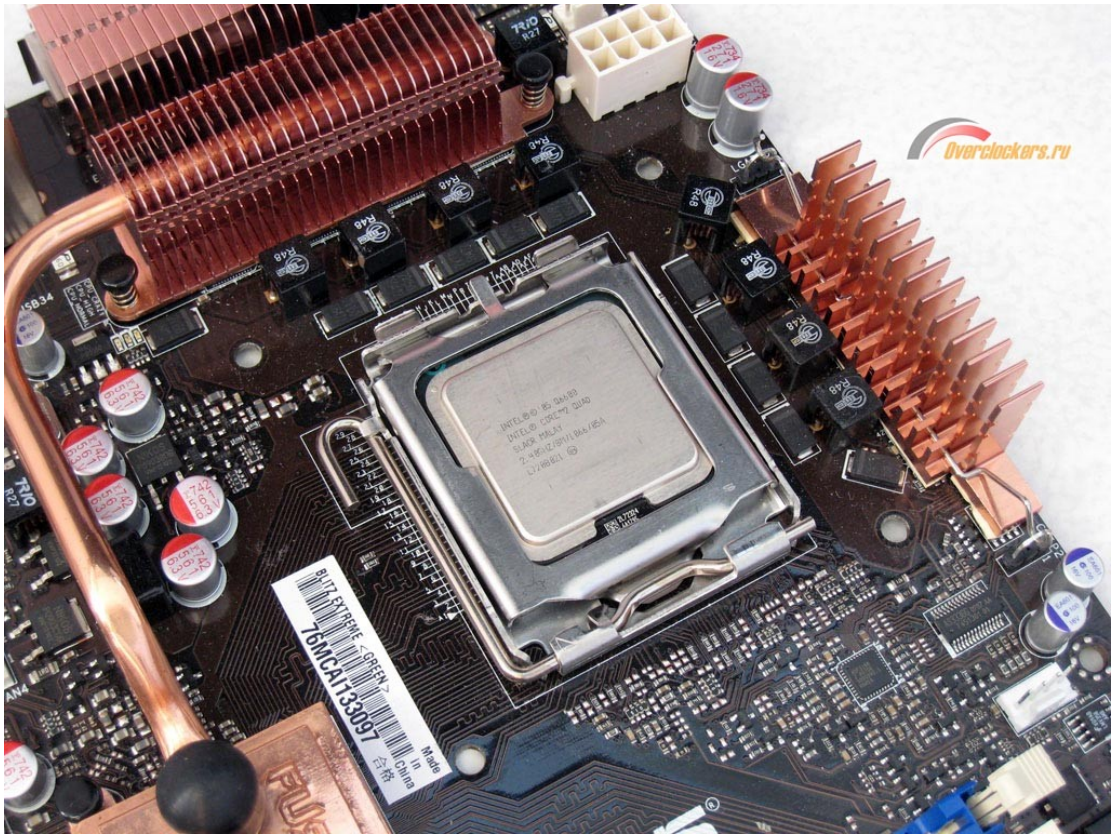


**Рисунок 2 Core2Duo – процессор, в котором обнаружено рекордное число критических ошибок**

## #3 – излишняя сложность

Чем сложнее система, тем выше вероятность внезапных отказов и тем проще атаковать ее, найдя слабейшее звено в линии обороны. Если администратор не пользуется удаленным доступном к реестру, зачем оставлять эту службу включенной?

IIS – бесспорно, могучая вещь, однако, так ли он необходим небольшой организации, обслуживающей сотни (ну, пускай даже тысячи) подключений в день? Сайт с движком на PHP – отличная штука, это современно и круто! А ошибки в скриптах (или самом PHP-интерпретаторе?). Почему бы не попробовать установить что-нибудь наподобие SMALL HTTP сервера? Бесплатный, поддерживает практически все функции, которые только могут быть нужны, обладает приятным интерфейсом, не требователен к системным ресурсам…

А ситуация, когда начинающий администратор вместо простой одно-ранговой сети, обслуживающей с полсотни сотрудников, разворачивает контроллер домена, работающий по принципу "сейчас опять все развалиться" — вообще живая классика.

Вывод: система должна быть предельно простой и не содержать ничего лишнего (тем не менее, стоит помнить, что иная простота хуже воровства).

## #4 – режем "в живую" без наркоза

Практически все администраторы устанавливают заплатки непосредственно на "продакшен" сервере (а в случае активной службы Windows-Update это происходит автоматически), даже не задумываясь какому они себя подвергают риску, не говоря уже про обновление прошивки BIOS.

В лучшем случае, после установки очередной порции заплаток возникают разные мелкие конфликты. Неприятные, но вполне совместимые с жизнью. Гораздо хуже, если система вообще откажется грузится или "забастуют" критические приложения сторонних разработчиков, что, кстати говоря, более чем вероятно и очень часто после выпуска заплатки, Microsoft следом за ней выпускает кучу инструкций по устранению конфликтов. Естественно, это относится только к популярным программным комплексам хорошо известным на западе. А как быть, если приложение складского учета, упакованное разработчики неимоверно крутым протектором

(чтобы злые хакеры не взломали) скручивает дулю и выдает голубой экран смерти или "всего лишь" аварийно завершает свою работу сразу же после запуска?

Выход: создать точную копию основного сервера, устанавливая заплатки/обновления сначала на ней и, если после более или менее полного цикла тестирования, никаких побочных эффектов не выявляется — переносить заплатки на основной сервер. Конечно, это требует дополнительных расходов и при "тугом" бюджете сервер-клон можно организовать и на виртуальной машине, не забывая, что она работает с виртуальным железом и потому потенциально не способна выявить ряд конфликтов. Однако, это все же лучше, чем совсем ничего.

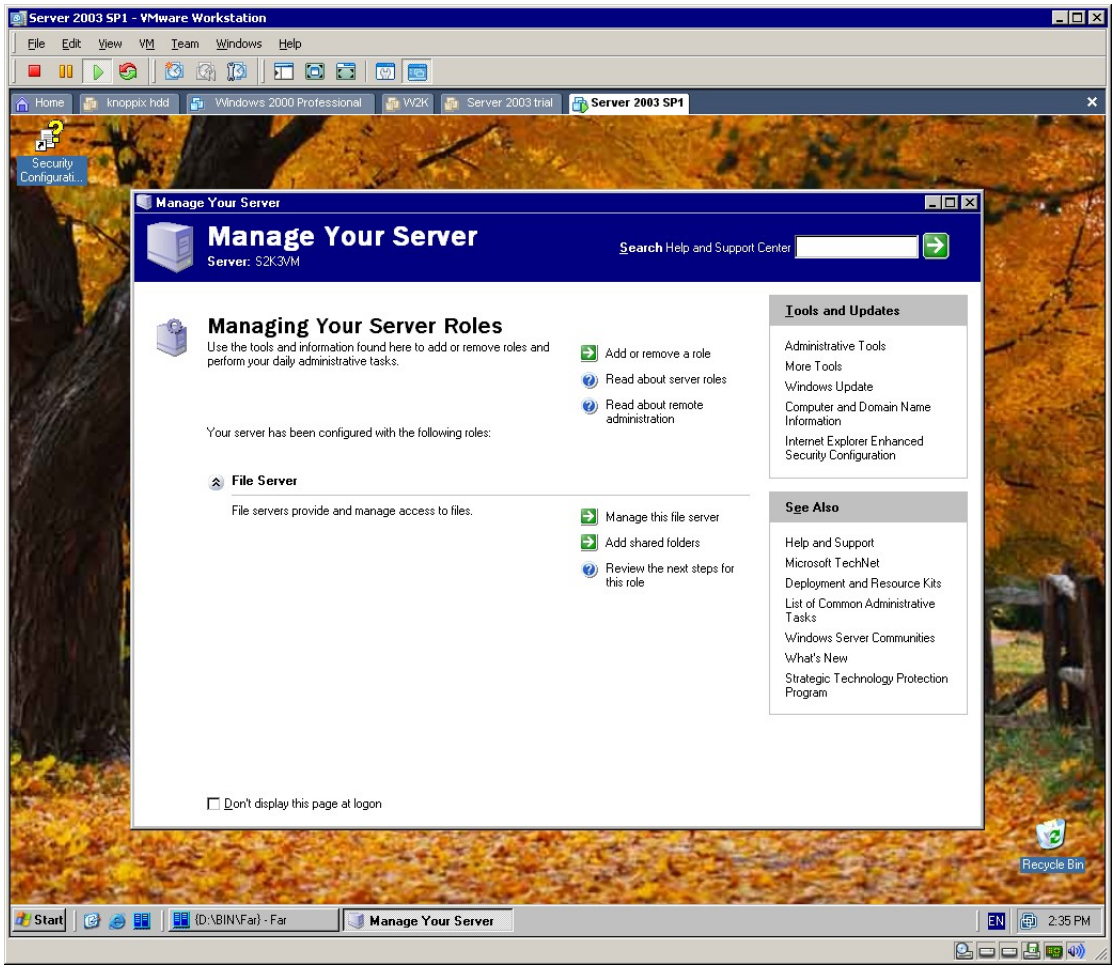


**Рисунок 3 экспериментальный Server 2003, воздвигнутый под VM Ware**

## *#5 – пакеты обновления и дистрибутивы*

Популярный способ "поднятия" упавшего сервера — установка операционной системы поверх уже имеющейся. Прием не то, чтобы хороший или красивый, однако, в жестких временных рамках и отсутствии резервной копии — это единственно возможное решение.

Проблема в том, что после установки Service Pack'ов, "родной" дистрибутив системы отказывается устанавливаться поверх более новой версии, предлагая либо вообще отказаться от инсталляции, либо удалить старую систему и поставить новую с нуля, переустанавливая все остальные программы, на что может уйти несколько рабочих дней (и бессонных ночей!)

К счастью, пакеты обновления могут быть интегрированы непосредственно в сам дистрибутив (чему посвящено огромное количество статей, так что мыщъх не будет повторяться, тем более, что описать процесс интеграции в двух словах все равно не получится). Желательно обновлять дистрибутивный диск при каждой установке Service Pack'а, чтобы потом лихорадочно не интегрировать его впопыхах, рискуя окончательно угробить систему, которой только полная переустановка и поможет.

Как вариант, можно заблаговременно создать образ системы с помощью штатной утилиты ntbackup.exe, а затем восстановить его с помощью все того же ntbackup.exe. Однако, следует помнить, программы, установленные после создания образа, при этом перестанут работать, а изменения настроек системы так же окажутся утерянными. Так что резервируйтесь почаще!!!

Если же система вообще отказывается запускаться, то… не беда. Распаковываем содержимое архивного файла (что можно сделать с помощью все того же ntbackup.exe) и, запустив консоль аварийного восстановления, перезаписываем файлы вручную.

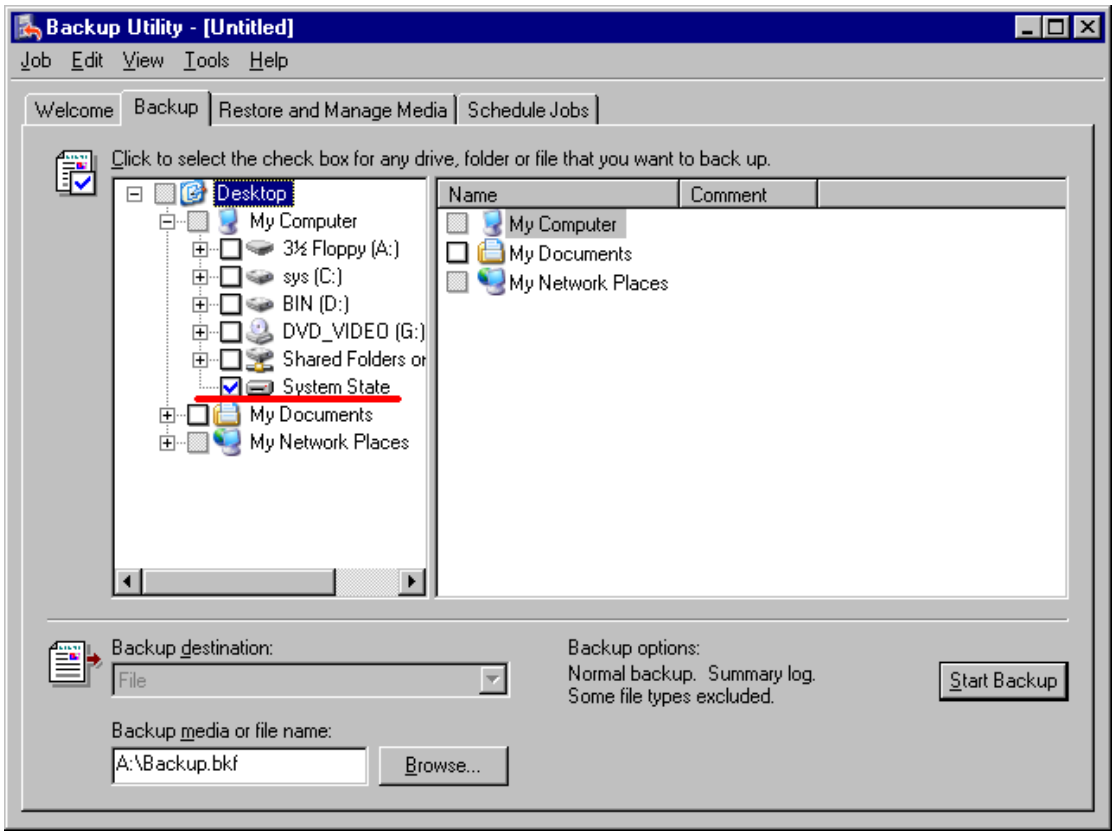


**Рисунок 4 сохранение состояния системы при помощи штатной утилиты ntbackup.exe**

## *#6 – он слишком много доверял…*

Практически все WEB/FTP/MAIL сервера по умолчанию устанавливают себя с привилегиями администратора, а то и системы (system), получая доступ ко всем файлам, которые только есть. Как следствие — любая ошибка конфигурации сервера, любой дефект PHP/Perl-скрипта, любая дыра самого сервера позволяет атакующему получать доступ к секретной информации или уничтожать данные.

Как защититься от этого? Очень просто! Достаточно запускать сервер из-под ограниченного аккаунта, имеющего доступ к тем и только тем файлам, которые ему необходимы. Что это за файлы?! Во-первых, файлы самого сервера, во-вторых, публичные файлы, раздаваемые пользователям. Конечно, если в сервере или скриптах имеется дыра, то злоумышленник по-прежнему сможет изменять конфигурацию сервера, а так же получать несанкционированный доступ к файлам других пользователей, не предназначенных для всяких "левых" лиц.

Тем не менее, разделение привилегий на уровне файловой системы, существенно ограничивает потенциальный ущерб, наносимый злоумышленником. Кстати говоря, не следует забывать, что многие файлы по умолчанию доступны всем и потому, администратору следует тщательно проверить атрибуты секретности, явно обозначив круг лиц, имеющих право на чтение/запись каждого более или менее значимого файла.

Может ли злоумышленник обойти ограничения доступа, налагаемые файловой системой? Безусловно. Но для этого ему придется найти дыру, дающую привилегии ядра или

позволяющую повышать права до уровня администратора. Такие дыры, действительно, есть, но их сравнительно немного и Microsoft их быстро затыкает.

## #7 – DEP u ASLR

Начиная с Server 2003 SP1 появилась поддержка неисполняемого стека и кучи, известная под аббревиатурой DEP (Data Execution Prevention), которая по умолчанию распространяется на все процессы (в XP же, по умолчанию DEP включен только для системных компонентов).

Насколько эффективна такая защита?! Во-первых, она требует обязательной поддержки со стороны процессора, позволяющего выставлять биты NX/XD не только для целых селекторов, как это было ранее, но и на уровне отдельных страниц. Без аппаратной поддержки, DEP вообще никак не работает. Во-вторых, DEP представляет собой довольно конфликтную штуку, препятствующую функционированию многих честных программ. В-третьих, вся эта защита элементарно обходится атакой типа return2libc, позволяющей атакующему вызывать API-функций, присваивающие стеку атрибуты исполняемого. Активный DEP отсекает лишь "пионерские" exploit'ы, протестированные на XP, но ни разу не нюхавшие Server 2003 SP1 и выше.

Для предотвращения атаки необходимо задействовать рандомизацию адресного пространства (Address Space Layout Randomization или, сокращенно, ASLR), реализованную в Server 2008, а так же в защитных пакетах независимых производителей, работающих хоть на Windows 2000 и не требующих аппаратной поддержки NX/XD битов. Одним из таких пакетов является Buffer-Shield, представляющий собой коммерческий порт известного проекта PaX, реализованного на большинстве UNIX-систем. Скачать бесплатную (урезанную) или полнофункциональную триальную версию можно с официально сайта: http://www.sys-manage.com/PRODUCTS/BufferShield/tabid/61/Default.aspx.

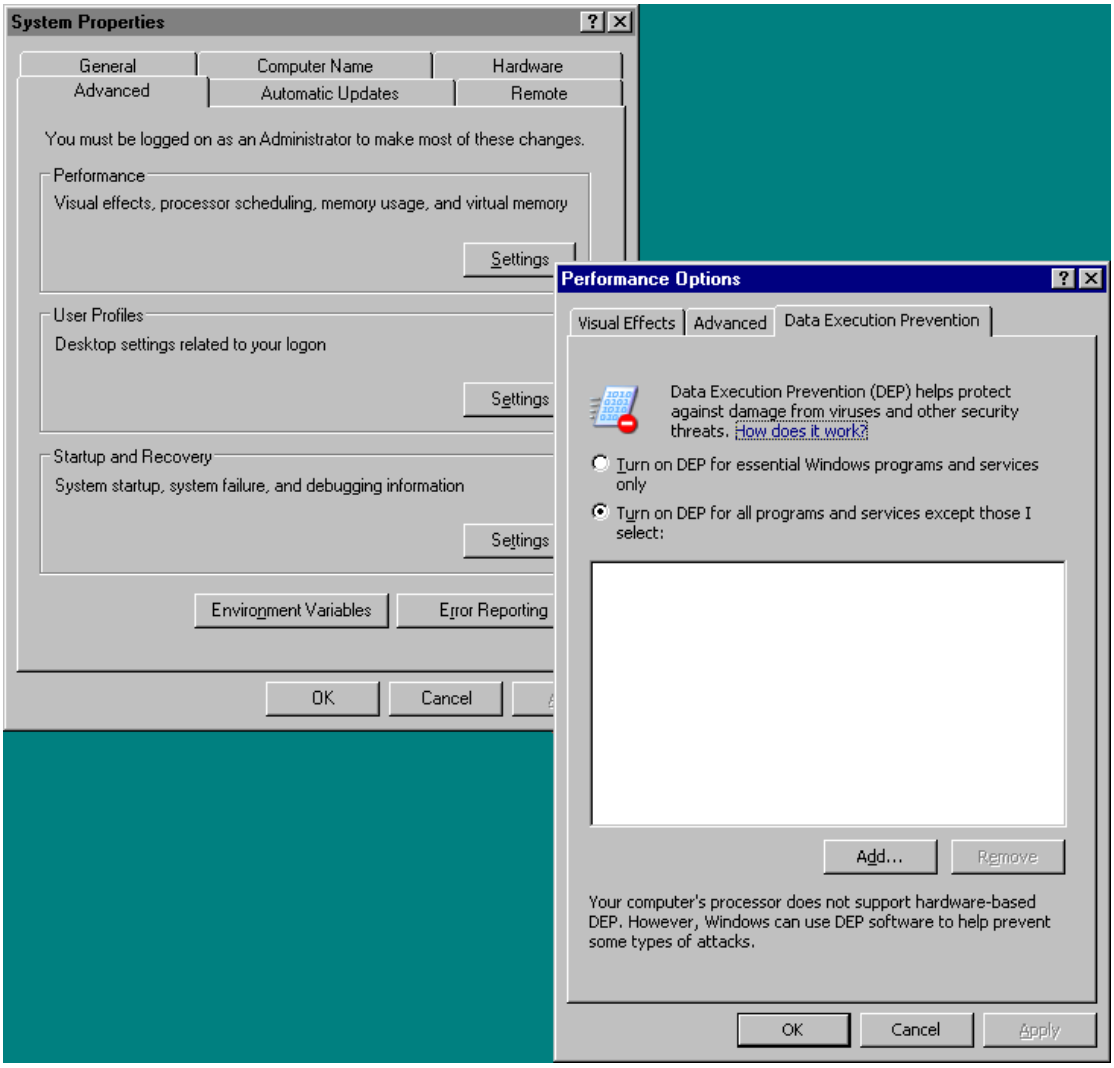


**Рисунок 5 активный DEP на Server 2003 это дикая головная боль на почве (не)совместимости и… никакой _реальной_ защиты**

## *#8 – предсказуемость конфигурации системы*

Успешность большинства атак объясняется высокой предсказуемостью конфигурации системы в установке по умолчанию. Это в мире открытых исходных текстов администратор может (и должен!) перекомпилировать все и вся, чтобы никакой хакер ни за что не догадался по каким адресам лежат интересующие его функции.

С Windows в этом плане ситуация намного сложнее, но не полностью безнадежно. Переименование ядра — эффективный способ борьбы с rootkit'ами, определяющими адреса функций путем вызова функции LoadLibrary("ntoskrnl.exe") без проверки реального имени ядра, задаваемого через ключ "/kernel=" файла boot.ini. Рекомендуется переименовать ядро во что-то другое, например, в souriz.exe, а вместо ntoskrnl.exe положить ядро от другой версии системы, чтобы адреса экспортируемых функций отличались.

Те rootkit'ы, что правят файл непосредственно на диске, уйдут лесом, не достигнув желаемой цели (ведь ntoskrnl.exe уже никак не используется), те же, rootkit'ы, что осуществляют перехват в оперативной памяти, залезут совсем не в ту степь и вызовут BSOD, что хоть и неприятно, но успешное внедрение rootkit'а было бы еще хуже.

Естественно, после переименования ядра, его необходимо обновить в кэше SFC, иначе она немедленно его восстановит, а так же перед установкой пакетов обновлений выполнить откат назад, поскольку, пакеты обновления (как и rootkit'ы) не проверяют реального имени ядра.

Установка системы на диск, отличный от C: так же уменьшает вероятность успешной атаки — большинство зловредных программ слишком тупы, а их создатели слишком ленивы,

чтобы проверить переменные окружения, вот они и используют фиксированные абсолютные пути.

## #9 – ботовой журнал капитана Немо

Реестр — крайне вредное изобретение, порождающее множество трудноразрешимых проблем. Текстовые конфигурационные файлы (традиционные для UNIX-систем) удобны тем, что в них можно оставлять комментарии и, в процессе внесения изменений, блокировать старые параметры символом комментария, существенно упрощая откат в случае неудачи.

А реестр?! Хорошо, если систему обслуживает всего один администратор, худо бедно помнящий какие параметры он менял и зачем. Когда же администраторов несколько и все они вносят изменения в реестр, работа превращается в сплошной разбор полетов "кто трогал реестр и весь его вытрогал?"

Чтобы этого не происходило, необходимо вести журнал (предпочтительнее всего на бумаге, поскольку, журнал — это документ), описывающий _каждое_ изменение конфигурации системы с указанием причин и сохранением предыдущих значений, заверенных подписью администратора. Тогда, если система начнет "выпендриваться" и вести себя нестабильно или же на ней обнаружатся черви, ломанувшиеся в широко открытые двери — всегда можно установить кто именно их открыл и чем он руководствовался при этом.

Кстати, в нормальных фирмах, у администратора есть инструкция, внятно объясняющая какие действия он вправе выполнять, а какие — нет. Эксперименты с системой на продакшен машинах (без предварительного согласования с руководством) в общем случае строго запрещены. И это правильно!

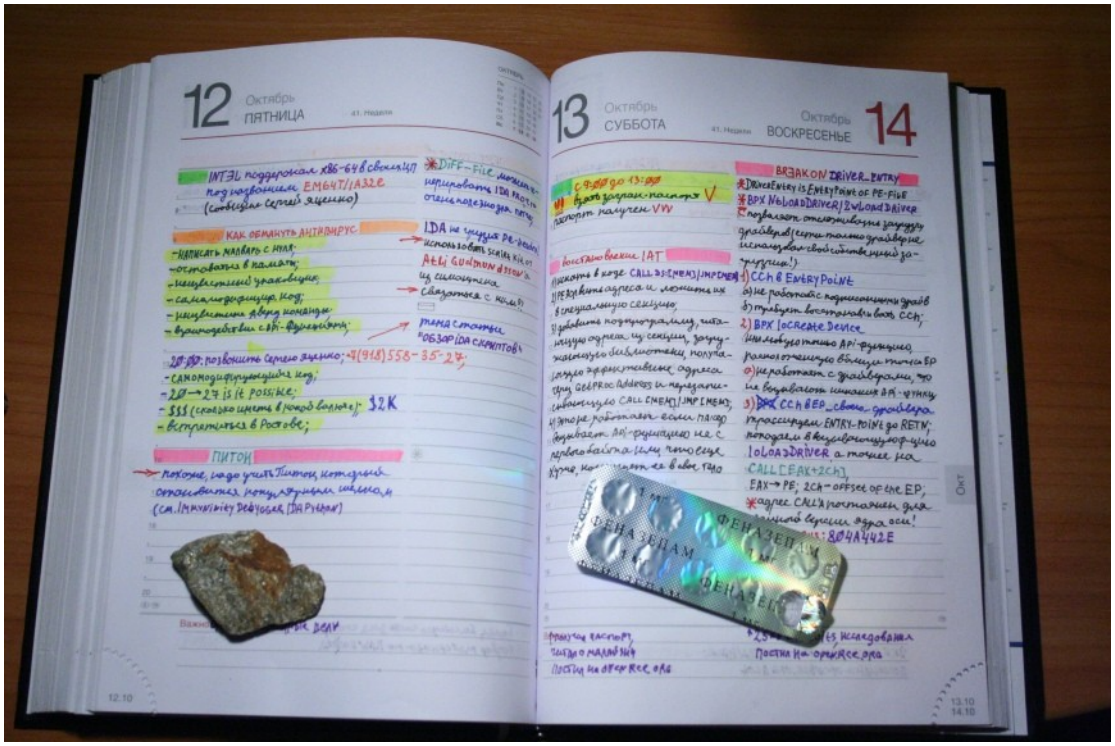


**Рисунок 6 бумага — великая вещь!**

## #10 – расплата за бездумность

Никакие защитные комплексы не дают 100% гарантии и от риска быть атакованным никуда не деться, увы. А потому, необходимо _заранее_ выработать четкий, хорошо продуманный и отлаженный план действий выхода из ситуации. Обнаружив на компьютере постороннюю зловредную программу, мало удалить ее из системы. Необходимо как минимум определить, что она успела натворить за время своей жизнедеятельности.

Помимо традиционных охранных комплексов, сервер должен быть оснащен снифферами и прочими шпионами всех мастей, протоколирующих максимум возможных действий и сохраняющих результат своей деятельности на носителях однократной записи

(CD/DVD-R), уничтожить которые никакой хакер не в состоянии и которые позволяет полностью реконструировать последовательность событий, прямо или косвенно связанных с атакой.

Отсутствие подобных средств — существенно ослабляет безопасность системы, поскольку, вспоминая слова Жеглова можно сказать, что степень защиты определяется не стойкостью сервера к атакам, а скоростью и успешностью раскрытия всяких несанкционированных действий.