exploit review
(5'й выпуск)

крис касперски ака мыщъх, no-email

Microsoft Windows SMB: удаленное выполнение кода

brief: в службе общих файлов и принтеров обнаружилась ошибка, о которой рапортовали сразу два человека: Gerardo Richarte из Core Security Technologies и Matthew Amdur из VMWare, а так же исследовательские группы NS Focus и Fortinet. послав специальным образом сконструированный пакет, любой неавторизованный злоумышленник, может вызвать основательное разрушение внутренних структур драйвера SRV.SYS, что приводит либо к отказу в обслуживании, либо к захвату управления на уровне нулевого кольца, что дает неограниченную власть над компьютером, образец пакета-разрушителя содержится в exploit'e, написанным хакером cocoruder'ом (frankruder#hotmail.com) и опубликованным Senthil Velan'ом; подробности об уязвимости содержатся в бюллетене безопасности Microsoft, зарегистрированном под номером MS06-063: http://www.microsoft.com/technet/security/Bulletin/MS06-063.mspx;

targets: уязвимости подвержены практически все Windows-системы: w2k SP4, XP SP1/SP2, XP x86-64, Server 2003/Server 2003 SP1/Server 2003 x86-64/Server 2003 IA64 (Itanium);

exploit: исходный код exploit'a, вызывающего отказ в обслуживании (но _не_ удаленное выполнение кода!) лежит на сервере security-focus'a: http://downloads.securityfocus.com/vulnerabilities/exploits/MS06_035-aug222006.c;

solution Microsoft уже выпустила заплатки для всех своих систем, которые можно скачать через службу Windows Update, однако, никаких гарантий, что дыра залатана полностью у нас нет (и знака ОТК на заплатках что-то не наблюдается), поэтому для надежности рекомендуется закрыть следующие порты на брандмауэре: UDP: 135, 137, 138 и 445. TCP: 135, 139 и 445 (при этом, естественно, общий доступ к файлам и принтерам работать не будет).

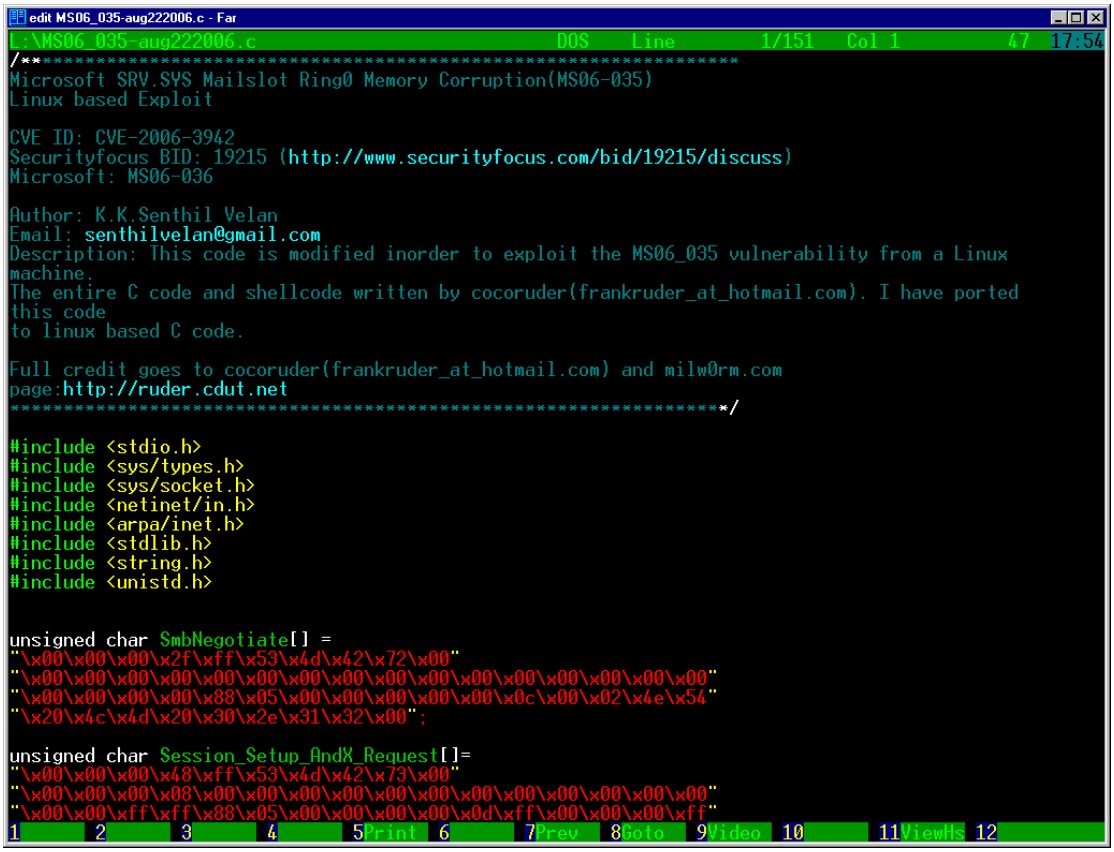

Рисунок 1 исходный код exploit'a, созданного cocoruder'ом

Google Earth: удаленное выполнение кода

brief: в первых числах октября коллектив исследователей JAAScois Security Team обнаружил огромную дыру в приложении Google Earth (beta), выпущенном 13 сентября 2006 (вот что значит релизить продукт по несчастливым числам!), допускающую возможность засылки shell-кода с последующим захватом управления и привилегий запустившего его пользователя (а большинство пользователей, как известно, постоянно сидят под администратором). это типичное переполнение буфера, в обработчике kml/kmz-файлов: программа выделяет блок памяти фиксированного размера и копирует туда принятые данные, забыв перед этим проверить их фактическую длину;

target: дыра содержится в версии 4.0.2091, как и принято у Googl'a, носящий статус "beta", про более ранние версии пока ничего не известно; при запуске приложения под управлением Windows Vista (вплоть до RC1), работающей на процессорах с аппаратной поддержкой DEP (т.е. битов NX/XD), вероятность успешной атаки составляет 1/256 за счет частичной рандомизации адресного пространства, а в остальных случаях уязвимая бета рушится с воплем о критической ошибке;

exploit: исходный код exploit'a, демонстрирующего переполнение, но _не_ содержащего shell-кода, можно скачать как с сервера security-focus'a: downloads.securityfocus.com/vulnerabilities/exploits/google-earth-v4.0.2091beta-bufferoverflow.c, так и с сервера самого коллектива JAAScois Security Team: http://www.jaascois.com;

solution поскольку, компания Google предпочла действовать методом страуса, рекомендуется прекратить использование Google Earth (beta) вплоть по изменения ситуации (выхода обновленной беты или по крайней мере, заплатки);

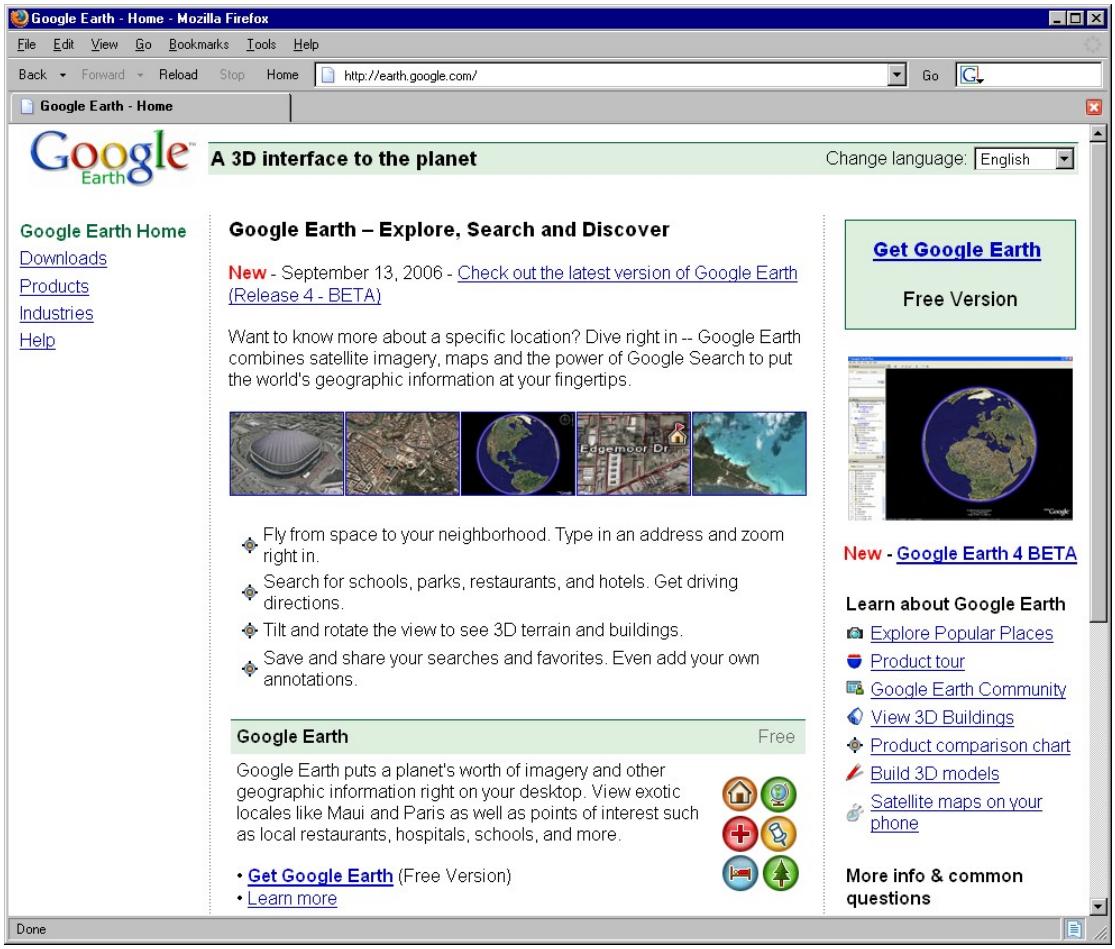

Рисунок 2 внешний вид сайта Google Earth, продолжающего раздавать уязвимую программу

Toshiba Bluetooth-стек: удаленное выполнение кода

brief: фирма Toshiba выпускает чипы беспроводных Bluetooth-устройств, используемые многими производителями материнских плат и другого оборудования. она же пишет под них драйвера, реализующие сетевой Bluetooth-стек (не путать с обычным стеком). 11 октября 2006 года David Maynor из SecureWorks, Inc. и независимый исследователь Jon Ellch опубликовали сообщение о дыре в драйвере TOSRFBD.SYS, приводящей к разрушению памяти с отказом в обслуживании типа перезагрузки системы или экрана голубой смерти, а при удачном стечении обстоятельств — и захвату управления компьютером на ядерном уровне привилегий (при этом хакер должен находится в радиусе действия bluetooth-устройства или вооружится направленной антенной, поражающей его на расстоянии прямой видимости). технические подробности можно найти на http://trifinite.org/trifinite_advisory_toshiba.html;

targets: уязвимости подвержены все версии драйверов от 3 до 4.00.35 включительно, в том числе и Toshiba Bluetooth Stack 4 SP2. эти драйвера поставляются вместе с оборудованием, выпущенным ASUS, Dell, Sony и другими компаниями. определить подвержено ли угрозе конкретно взятое устройство или нет, можно, во-первых, по наличию драйвера TOSRFBD.SYS, а, во-вторых, по имени производителя (Toshiba) в свойствах беспроводного устройства.

exploit: для реализации атаки exploit'a не требуется, достаточно как следует "пропинговать" жертву, обрушив на нее шквал L2CAP-echo запросов, что можно осуществить с помощью linux-утилиты l2ping. атаки данного типа довольно широко распространены и называются BlueSmack. подробнее о них можно прочитать на http://trifinite.org/trifinite_stuff_bluesmack.html

solution фирма Toshiba выпустила заплатки для своих драйверов, доступных для скачки по http://aps.toshiba-tro.de/bluetooth/redirect.php?page=pages/download.php, однако, следует помнить, что многие производители аппаратного обеспечения модифицировали оригинальные драйвера и в этом случае обновления необходимо брать с их сайтов;

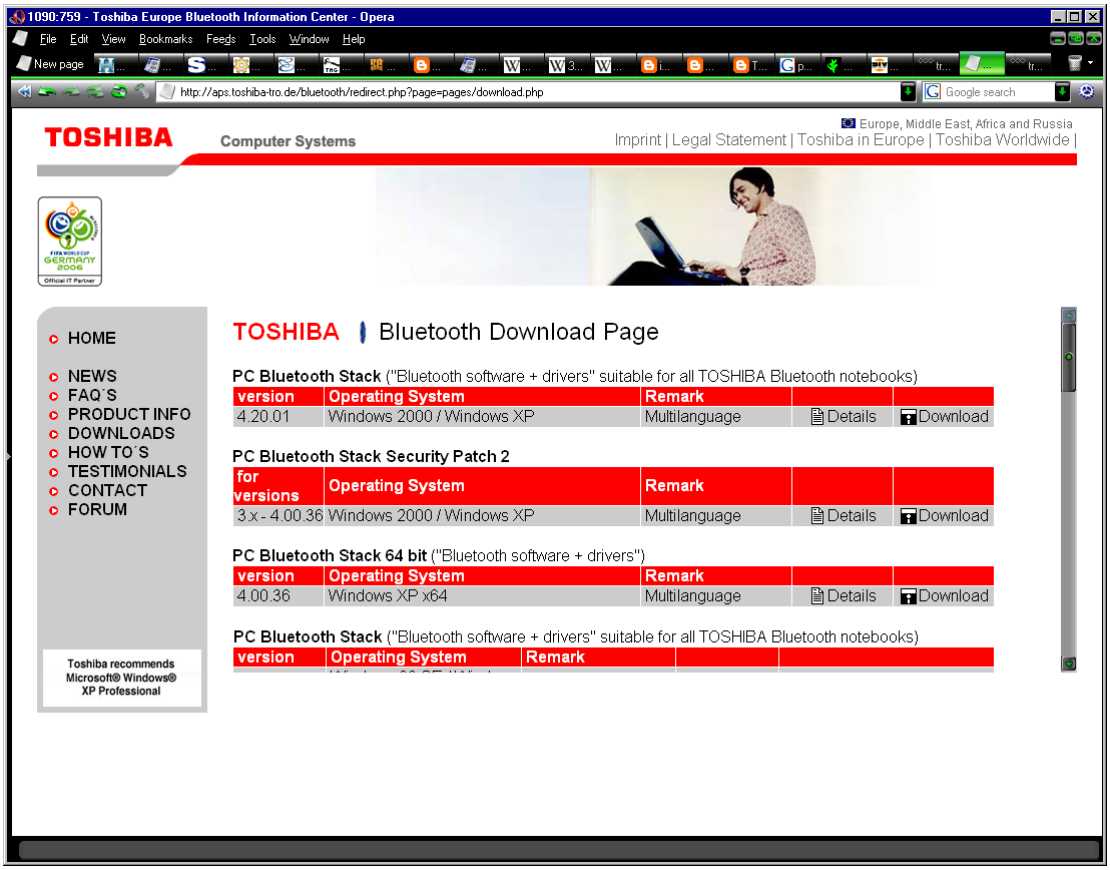

Рисунок 3 отсюда можно слить заплатки для дырявых драйверов TOSHIB'ы

full disclose
переполнение буферов в суровых условиях висты

brief: у мыщх'а две новости. хорошая и плохая. начну с хорошей. в Windows Vista/Server Longhorn полностью переписан сетевой стек и реализован IPv6, причем реализован _очень_ коряво, ранние беты повторяли практически все ошибки, допущенные в старом сетевом стеке, вылизываемом годами. и хотя основные дыры в Vista RC1 уже залатаны, сама новизна IPv6 протокола вкупе с не протестированным сетевым стеком выглядит весьма сексапильно и открывает огромные перспективы для всевозможных атак. для хакеров это настоящий клад. теперь плохая новость. Microsoft предприняла целых комплекс противохакерских мер, затрудняющих засылку shell-кода и реализацию большинства атак, поэтому, прежде чем виста станет доминирующей системой (а случится это скорее всего через год-полтора) злоумышленникам требуется "проапгрейдить" арсенал своего оружия, адаптировавшись к новым условиям. что же это за защитные механизмы и можно ли их обойти? вот об этом мы сейчас и поговорим.

intro: основные оборонительные сооружения расставлены вокруг переполняющихся буферов, однако, эта стратегия обречена на поражение, поскольку "брачный период" атак данного типа уже позади. достигнув пика своей популярности в ~2005 году, переполняющиеся буфера стали клониться к закату. хакерская мысль не стоит на месте! набирают силу атаки, основанные на ошибках синхронизации драйверов, примерами которых являются дыры в драйверах Intel Centrino и Toshiba Bluetooth, дающие захватчикам ядерные привилегии нулевого кольца! но все-таки вернемся к

переполняющимся буферам. было бы глупо сдаваться без боя, тем более, что оборону Microsoft'а сравнительно легко прорвать.

DEP: доля процессоров с аппаратной поддержкой DEP обречена на неуклонное возрастание и хотя на x86-платформах DEP по умолчанию включен только для некоторых системных служб, поскольку все еще существует множество приложений, нуждающихся в исполняемом стеке (в особенности это относится к упаковщикам, протекторам и прочим защитным механизмам), разработчики уже отреагировали на эту инициативу. новые версии (в своей массе) больше не трактуют атрибут PAGE_READ как PAGE_EXECUTABLE, и перед выполнением кода в области памяти отличной от секции .text, атрибут PAGE_EXECUTABLE присваивается явно, чтобы программа могла работать при включенном DEP. кстати говоря, это обязательное требование, предъявляемое Microsoft'ом при выдаче логотипа Windows Compatible. несложно спрогнозировать, что через несколько лет программ, конфликтующих с DEP, практически не останется совсем и в очередном Service Packe'e для висты Microsoft запросто сможет изменить политику "DEP выключен по умолчанию для несистемных приложений" на "по умолчанию DEP включен для всех приложений". сам по себе DEP не сильно осложняет жизнь хакерам и легко обходится атаками типа return-to-libc (подробно описанных в моей статье "судьба shell-кода на системах с неисполняемым стеком или атака на DEP", которую можно скачать с ftp://nezumi.org.ru), однако, в комбинации с другими защитными механизмами, DEP превращается в мощную броню, существование которой приходится принимать в расчет, как и тот факт, что эта броня прожигается кумулятивными снарядами, продвигаемыми на рынок самой же Microsoft под видом платформы .NET, базирующейся на интерпретируемом языке C#. несмотря на то, что C# намного меньше подвержен ошибкам переполнения, чем Си, защитные свойства DEP'а на него не распространяются, поскольку интерпретируемый код с точки зрения процессора представляет данные, не нуждающиеся в атрибуте PAGE_EXECUTABLE. на самом деле, C# не совсем интерпретируемый, а компилируемый в память, причем откомпилированный код помешается в область памяти, доступную как на запись, так и на исполнение. так что внедрение процессоров с поддержкой аппаратного DEP'а компенсируется победоносным шествием .NET'а, на котором удобно писать интерфейс, но реализовывать приложение целиком… нет уж, увольте. во-первых, C# стоит намного ближе к Visual Basic'у, чем к Си, а какая у Basic'а производительность? во-вторых, на Си/Си++ написано огромное количество библиотек, которые никто переписывать на .NET не собирается и ближайшие 3-5 лет большинство программ продолжат стоится по гибридному принципу — Си/Си++ плюс C#, что позволит эксплуатировать ошибки переполнения, характерные для Си/Си++ через C#! если хакерская активность и снизится, то несильно.

rands: классический сценарий атаки на переполняющиеся буфера предполагает подмену адреса возврата из функции на адрес машинной команды jmp esp (опкод FFh E4h), находящейся где-то в оперативной памяти и таких команд там предостаточно — такие двухбайтовые последовательности в изобилии встречаются как в системных библиотеках, так и в самом атакуемом приложении. jmp esp передает управление на вершину стека, в которой расположен переполненный буфер, содержащий shell-код и все идет по плану, по такому хорошему конопляному плану, выращенному под жарким южном солнцем. но при включенном DEP такой план ни хвоста не торкает, поскольку выполнение команд в стеке категорически запрещено и приходится совершать ряд дополнительных действий, по устранению этих досадных неприятностей. самое простое, что можно сделалась – это занести в стек адрес API-функции VirtualProtect, позволяющей манипулировать атрибутами доступа к памяти, а вслед за ней — адрес API-функции MoveMemory, копирующей shell-код из стека в область памяти, "обработанную" функцией VirtualProtect, присвоившей ей атрибут "исполняемый". остается только передать сюда управление, для чего в стек кладется адрес новой дислокации shell-кода и наступает полный приход.

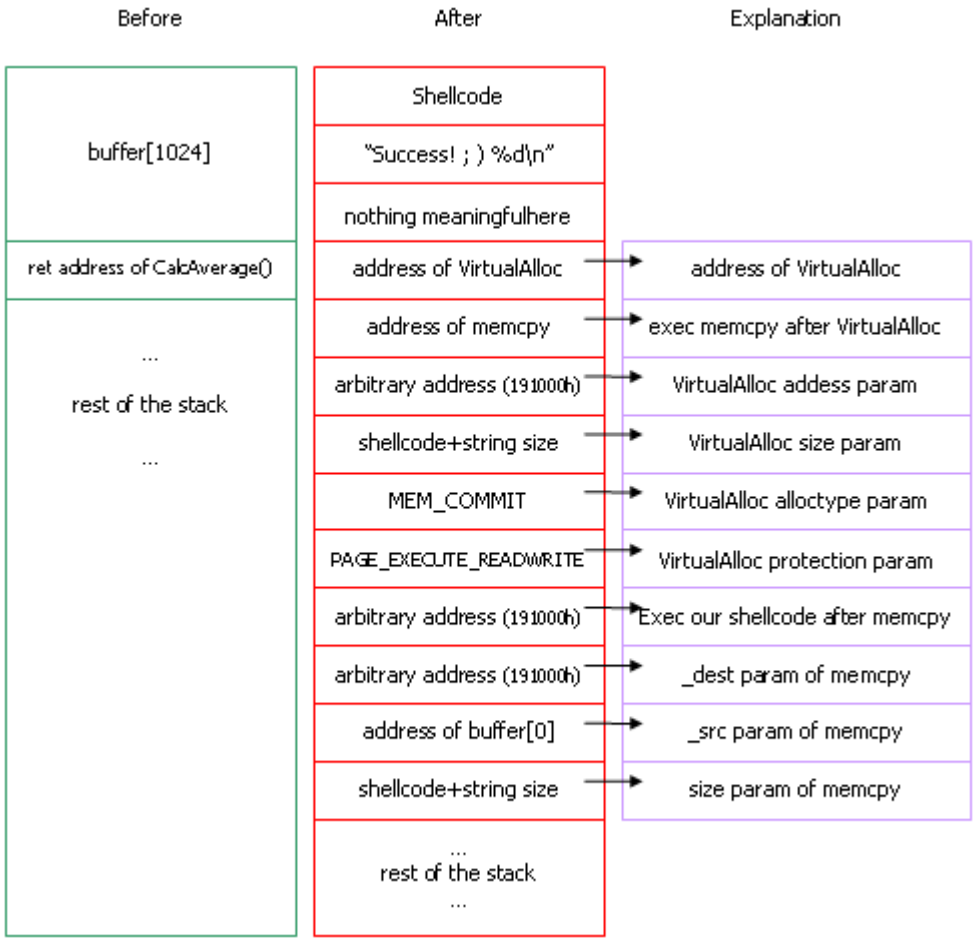

Рисунок 4 подготовка стека к атаке на DEP

как видно, независимо от активности DEP, мы должны знать содержимое адресного пространства атакуемой жертвы, чтобы найти в памяти инструкцию jmp esp или адреса API-функций. на XP и Server 2003 это не представляет никаких проблем, поскольку и сам исполняемый файл, и системные библиотеки всегда грузятся по одним и тем же адресам. конечно, эти адреса постоянны только в рамках данной версии Windows и меняются с каждым Service Pack'ом, что вынуждает атакующего либо определять версию косвенным путем, либо просто действовать наугад. если ему повезет, он будет кайфовать, в противном случае у жертвы случится крэш и она словит сообщение о критической ошибке (ни слова о критических днях, мужики), а вместе с ним бэд трип.
постоянным и не зависящим от версии остается только базовый адрес загрузки самого уязвимого приложения, однако, это не слишком большая зацепка, поскольку при смене версии меняется и его содержимое. то есть, написать универсальную атакующую программу, поражающую все системы не так-то просто, а в общем случае — вообще невозможно, но это обстоятельство не ломает кайф и не высаживает на измену. количество версий Windows (вместе с версиями уязвимого приложения) хоть велико, но все-таки конечно. если настойчиво долбить жертву, то рано или поздно она скажет "пых", shell-код получит управление и все будут очень довольны, ну или почти все.
что изменилось в висте? (дети, не ставьте себе висту, такая дристня нападет!) системные библиотеки теперь грузятся по одному из 256 возможных базовых адресов, выбираемом случайным образом. в заголовке исполняемых файлов появился специальный бит, указывающий должен ли он загружаться по фиксированному или наугад взятому адресу. тоже самое относится к несистемным динамическим библиотекам. это значит, что в настоящий момент реально радномизируются _только_ системные библиотеки и приложения, входящие в комплект штатной поставки висты. все ранее написанные приложения грузятся по одному и тому же адресу, и будут грузится по нему еще долго, поскольку сторонние разработчики крайне прохладно относятся к новой инициативе Microsoft и даже когда появится линкеры,

поддерживающие флаг рандомизации, вовсе не факт, что каждый поспешит им воспользоваться, поскольку загрузка по произвольному адресу требует наличия таблицы перемещаемых элементов, увеличивает потребности в оперативной памяти (особенно при запуске нескольких копий приложения) и облегчает взлом, поскольку существующие упаковщики/протекторы не поддерживают рандомизацию и распаковывают файл по фиксированным адресам. отсюда следует, что часть адресного пространства уязвимого приложения, относящаяся к исполняемому файлу и его "личным" динамическим библиотекам по-прежнему остается предсказуемой, что позволяет атакующему свободно передавать управление на shell-код через jmp esp, а так же вызывать любые функции содержащиеся или импортируемые уязвимым приложением! проблема в том, что функция VirtualProtect не относится к числу особо популярных и в таблице импорта конкретно взятого приложения ее вполне может и не быть. какой облом! или все-таки не облом, а приход?! да еще какой конкретный приход! на самом деле ситуация практически никак не изменилась. вероятность успешной атаки (с учетом кучи версий Windows) и раньше не составляла 100%, теперь же, она сократилась в приблизительно в ~256 раз. как говориться, ну не умер, так сдох. и это они еще называют надежной защитой?! ну-ну… тот, кто очень хочет, своего добьется. если настойчиво долбиться, рано или поздно, хакер угадает базовый адрес KERNEL32.DLL, содержащую все необходимые ему функции и тогда жертва падает. то есть, нет. как раз наоборот, тогда произойдет захват управления удаленной машиной, а во всех остальных случаях — ее падение. администратор запариться поднимать сервер (рабочую станцию), не понимая – то ли это его атакуют, то ли что-то конкретно глючит (например, сбоит память). правда, заподозрив атаку, он сможет быстренько скачать все заплатки (при условии, что они есть), заткнув дыры. но! это опять-таки- теоретически. при настойчивой долбежке компьютер будет падать так часто, что ничего скачать не удастся! но все-таки, падения — это нехорошо. зачем привлекать лишнее внимание? чтобы подавить сообщения о критических ошибках необходимо перезаписать указатель на текущий обработчик структурных исключений, с которым связана еще одна оборонная инициатива Microsoft — в висте он перемещен из легко затираемого стека в секцию .pdata, доступную только на чтение. какой обл… то есть, какой дуболом проектировал защиту?! все это относится только к статическим обработчикам структурных исключений, адрес которых известен еще на стадии компиляции. это справедливо для простых Си-программ, но в Си++ достаточно большое количество обработчиков устанавливаются динамически. к тому же, установкой обработчиков ведает компилятор, а все существующие компиляторы размешают указатели на обработчики в стеке! так что появление висты само по себе ничего не меняет. как минимум требуется дождаться появления обновленных версий компиляторов, а до этого можно смело перезаписывать переназначать обработчики структурных исключений на свои собственные. правда, начиная с Server 2003, система выполняет дополнительную проверку, препятствующую размещению кода обработчика в стеке, но… можно "подсунуть" адрес одного из обработчиков уязвимого приложения, который бы не завершал работу приложения и не выдавал бы никаких сообщений на экран, а тем или иным образом, "обработав" исключение, продолжал бы работу в обычном режиме. в большинстве случаев для этого достаточно прыгнуть сразу в середину обработчика, поближе к API-функции Continue и… можно смело долбить переполняющийся буфер, перебирая все возможные варианты один за другим.