ХАРЬКОВСКИЙ НАЦИОНАЛЬНЫЙ ЭКОНОМИЧЕСКИЙ УНИВЕРСИТЕТ

ЛЕКЦИЯ 10

Защита приложений

доцент кафедры информационных систем

к.т.н., с.н.с. Евсеев Сергей Петрович

ПРАВИЛА БОЯ

Любое ПО, которое устанавливается на компьютер, особенно подключенный к Интернету, необходимо защищать. Имеется в виду то, что код приложения становится постоянно — 24 часа в сутки, 7 дней в неделю — доступным для атаки из любой точки земного шара.

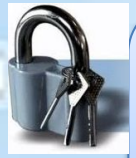
Поэтому он должен противостоять атакам, чтобы не стать причиной компрометации, повреждения, удаления или перлюстрации злоумышленником защищаемых системой ресурсов.

Правило №1: защищающемуся приходится охранять все слабые места, а нападающему достаточно выбрать одно из них

Нападающему хватит единственной бреши в защите вашего приложения, а вы обязаны «закрыть» все слабые места.

Правило №2: защищающийся готовится отразить известные атаки, а нападающий может разработать новые методы взлома

Единственный способ защититься от атак заранее неизвестного типа — отключить все возможности программы, которые явно не востребованы пользователем.



Правило №3: оборону следует держать постоянно, удар же возможен когда угодно

Разработчики должны предусмотреть в ПО средства противостояния атакам и инструменты для мониторинга системы, помогающие пользователям выявить атаку.

ПРАВИЛА БОЯ

Правило №4: защищающему приходится соблюдать правила, а нападающему не возбраняется вести «грязную игру»

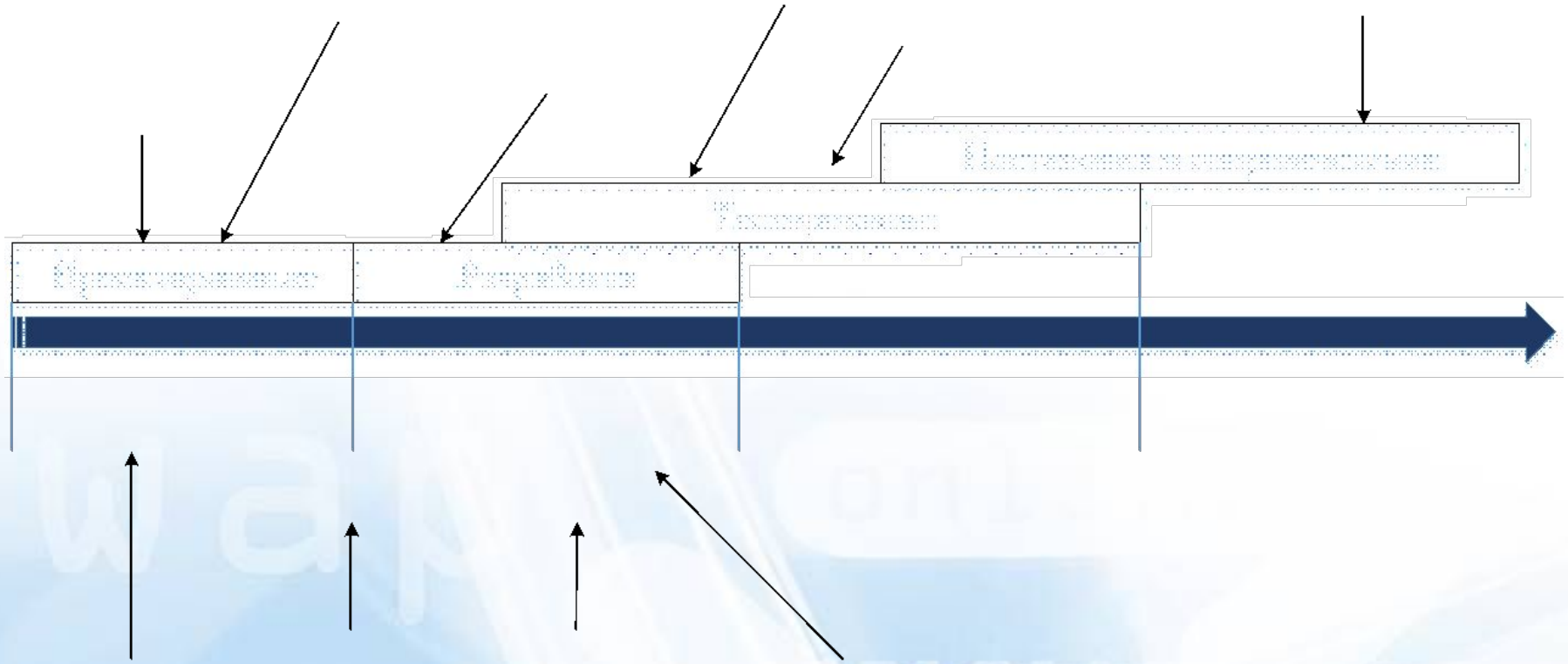
В распоряжении защищающегося масса хорошо изученных средств, разработанных «белыми хакерами» для защиты системы и обнаружения атак.

Нападающему же ничто не мешает прибегнуть к любому доступному ему средству проникновения в систему и обнаружения слабых мест в защите. И в этом случае преимущество на его стороне.

ISO 17799 “Information Technology – Code of practice for information security management” (“Информационная технология – свод правил по управлению информационной безопасностью”) – международный стандарт, определяющий организационные, физические, коммуникационные и системные политики безопасности при разработке ПО, – во введении и подразделе 10.1.1 раздела 10.1 “Securityrequirements of systems” (“Требования к безопасности систем”):

Требования по безопасности и процедуры по защите должны соответствовать бизнес-ценности информационных активов и возможному ущербу для бизнеса, возникающему по причине нарушения или отсутствия защиты.

УКРЕПЛЕНИЕ ЗАЩИТЫ В ПРОЦЕССЕ РАЗРАБОТКИ ПО НА КАЖДОМ ЭТАПЕ



Лучший пример итеративного первого шага в процессе разработки безопасного ПО - образование.

ПРИНЦИПЫ БЕЗОПАСНОСТИ

ПРИНЦИП SD^3: БЕЗОПАСНО СОГЛАСНО ПРОЕКТУ, ПО УМОЛЧАНИЮ И ПРИ РАЗВЕРТЫВАНИИ

безопасность приложения должна обеспечиваться на стадии разработки проекта, в конфигурации по умолчанию и при развертывании

Безопасно согласно проекту

- Назначьте человека, ответственного за обеспечение безопасности продукта
- К моменту завершения фазы проектирования подготовьте модели опасностей, грозящих системе
- Придерживайтесь рекомендаций по безопасному проектированию и программированию
- Как можно раньше устраняйте все ошибки, которые возникают из-за несоблюдения рекомендаций
- Упростите код и модель защиты
- Перед началом продаж ПО проведите «тест на выживание»

Безопасно по умолчанию

- Не «включайте» в конфигурации по умолчанию все функции и возможности: выбирайте только те, которые потребуются большинству ваших пользовать ей, и позаботьтесь о простом механизме активизации остальных функций
- Приложение должно работать в условиях минимально возможных привилегий
- Обеспечьте адекватную защиту ресурсов. Конфиденциальные данные и критически важные ресурсы обязательно защитите от атак.

Безопасно при развертывании

- Позаботьтесь о создании механизма администрирования безопасности ПО
- Максимально оперативно выпускайте качественные пакеты исправлений подсистемы безопасности
- Проинформируйте пользователей, как ***безопасно*** работать с системой

ПРИНЦИПЫ БЕЗОПАСНОСТИ

УЧИТЕСЬ НА ОШИБКАХ

История — это огромная система раннего предупреждения.
Норман Козине

в Microsoft организована такая важная процедура, как «посмертное вскрытие» ошибок защиты, которые были зафиксированы в Центре безопасности Microsoft (Microsoft Security Response Center)

В документе содержатся следующие данные:
- название продукта;
- версия продукта;
- имя контактного лица:
- код ошибки в базе данных;
- описание бреши;
- возможные последствия взлома защиты в этом месте;
- проявляется ли эта ошибка при установке по умолчанию;
- что могут сделать проектировщики, разработчики или тестировщики, для устранения недоработки

ПРИНЦИПЫ БЕЗОПАСНОСТИ

УМЕНЬШАЙТЕ «ПЛОЩАДЬ» ПРИЛОЖЕНИЯ, УЯЗВИМУЮ ДЛЯ НАПАДЕНИЙ

Мучительнее извлечения уроков из опыта только их неизвлечение.
Арчибальд Маклейш

Очень важно минимизировать их число, а пользователей заставить активизировать дополнительные возможности только по мере необходимости

Следует учитывать число:

- открытых сокетов (TCP и UDP);
- открытых именованных каналов (named pipes);
- открытых конечных точек удаленного вызова процедур (RFC endpoints);
- служб;
- служб, запускаемых по умолчанию;
- служб, обладающих высокими полномочиями:
- фильтров и приложений ISAPI;
- динамических Web-страниц;
- учетных записей в группе администраторов;
- файлов, каталогов и параметров реестра с не обеспечивающими должной защиты списками управления доступом (Access Control List, ACL).

ПРИНЦИПЫ БЕЗОПАСНОСТИ

НАЗНАЧАЙТЕ БЕЗОПАСНЫЕ ПАРАМЕТРЫ В КОНФИГУРАЦИИ ПО УМОЛЧАНИЮ

Не помнящий прошлого обречен на его повторение.
Джордж Сантаяна

Для уменьшения открытой «площади» приложения необходимо, в том числе, назначать безопасные параметры для конфигурации по умолчанию.

ЗАЩИЩАЙТЕ ВСЕ УРОВНИ

Для уменьшения открытой «площади» приложения необходимо, в том числе, назначать безопасные параметры для конфигурации по умолчанию.

ИСПОЛЬЗУЙТЕ НАИМЕНЬШИЕ ПРИВИЛЕГИИ

Позаботьтесь, чтобы все приложения выполнялись с минимальным набором привилегий, достаточным для выполнения работы, и не более того

ПРИНЦИПЫ БЕЗОПАСНОСТИ

БУДЬТЕ ГОТОВЫ К ПРОБЛЕМАМ С ОБРАТНОЙ СОВМЕСТИМОСТЬЮ

Решаясь на внесение в продукт существенных изменений, связанных с безопасностью, будьте готовы к появлению вороха проблем с обратной совместимостью

ПРИНИМАЙТЕ КАК АКСИОМУ, ЧТО ВНЕШНИЕ СИСТЕМЫ НЕ ЗАЩИЩЕНЫ ПО ОПРЕДЕЛЕНИЮ

Предположение о незащищенности внешних систем связано с принципом защиты на всех уровнях: одна только осторожность при взаимодействии с такими системами — уже неплохое средство безопасности. Любые данные, поступающие из систем, над которыми вы не властны, по определению признаются небезопасными, а сами системы — потенциальным источником атак.

ПРИНЦИПЫ БЕЗОПАСНОСТИ

РАЗРАБОТАЙТЕ ПЛАН ДЕЙСТВИЙ НА СЛУЧАЙ СБОЕВ И ОТКАЗОВ

золотое правило безопасного сбоя: запрещать все по умолчанию, а разрешать только то, что соответствует всем условиям

ПОМНИТЕ, ЧТО ВОЗМОЖНОСТИ ПОДСИСТЕМЫ БЕЗОПАСНОСТИ — ЭТО НЕ ТО ЖЕ САМОЕ, ЧТО БЕЗОПАСНЫЕ ВОЗМОЖНОСТИ СИСТЕМЫ

Чтобы гарантировать защиту от атак, мы должны быть уверены в том, что включаем корректные функции и адекватно их используем

НЕ СТРОЙТЕ СИСТЕМУ ЗАЩИТЫ НА ОГРАНИЧЕНИИ ИНФОРМАЦИИ О ПРИЛОЖЕНИИ

Всегда предполагайте, что нападающий знает ровно столько же, сколько вы, в том числе знаком со всеми исходными кодами и проектной документацией.

ПРИНЦИПЫ БЕЗОПАСНОСТИ

РАЗДЕЛЯЙТЕ КОД И ДАННЫЕ

Если в вашем приложении предусматривается смешивание кода и данных, мы рекомендуем по умолчанию запрещать выполнение кода и предоставить пользователю самостоятельно определять политику

КОРРЕКТНО ИСПРАВЛЯЙТЕ ОШИБКИ В ЗАЩИТЕ

Обнаружив ошибку защиты или архитектурный просчет, устраните его и немедленно проверьте остальные части приложения

МОДЕЛИРОВАНИЕ ОПАСНОСТЕЙ

ПРАВИЛА СОЗДАНИЯ:

- все потоки данных начинаются и заканчиваются на процессах;
- хранилища данных соединяются с процессами через потоки данных;
- хранилища данных никогда не соединяются друг с другом напрямую — только через процессы;

Основополагающий принцип ***диаграммы потоков данных*** (data flow diagrams, DFD):

приложение или систему можно разбить на подсистемы, а те, в свою очередь, — на более мелкие подсистемы следующего уровня. Подобный итерационный подход делает DFD-диаграммы удобным средством декомпозиции приложений

КЛАССИФИКАЦИЯ ОПАСНОСТЕЙ: МЕТОДИКА STRIDE

РАЗГЛАШЕНИЕ ИНФОРМАЦИИ (INFORMATION DISCLOSURE)

Подразумевается раскрытие информации лицам, доступ к которой им запрещен.

ОТКАЗ В ОБСЛУЖИВАНИИ (DENIAL OF SERVICE)

В атаках такого типа взломщик пытается лишить доступа к сервису правомочных пользователей

DoS-атаки трудно предотвратить, так как они относительно просты в реализации, причем атакующий остается неизвестным

ПОВЫШЕНИЕ ПРИВИЛЕГИЙ (ELEVATION OF PRIVILEGE)

В данном случае непривилегированный пользователь получает привилегированный доступ, позволяющий ему «взломать» или даже уничтожить систему.

В уязвимой системе ничто не запретит злоумышленнику поместить на диск файл, который выполняется автоматически при входе пользователя в систему, и дождаться вход в систему полномочного пользователя.

DREAD — МЕТОДИКА ОЦЕНКИ РИСКА

ПОТЕНЦИАЛЬНЫЙ УЩЕРБ (DAMAGE POTENTIAL)

Мера реального ущерба от успешной атаки.

Наивысшая степень (10) опасности означает практически беспрепятственный взлом средств защиты и выполнение практически любых операций

ВОСПРОИЗВОДИМОСТЬ (REPRODUCIBIHTY)

мера возможности реализации опасности.

ПОДВЕРЖЕННОСТЬ ВЗЛОМУ (EXPLOITABILITY)

мера усилий и квалификации, необходимых для атаки.

КРУГ ПОЛЬЗОВАТЕЛЕЙ, ПОПАДАЮЩИХ ПОД УДАР (AFFECTED USERS)

доля пользователей, работа которых нарушается из-за успешной атаки. Оценка выполняется на основе процентной доли: 100% всех пользователей соответствует оценка 10, а 10% — 1 балл.

ВЕРОЯТНОСТЬ ОБНАРУЖЕНИЯ (DISCOVERABILITY)

любая опасность поддается реализации, поэтому выставляется всем по 10 баллов, а при ранжировании опасностей учитываю другие показатели.

МЕТОДЫ БЕЗОПАСНОГО КОДИРОВАНИЯ

Параметр /GS предотвращает эксплуатацию простого переполнения буфера

Разрушение стека (stack smashing) - стандартный метод переполнения буфера для изменения адреса возврата функции. Пресекается «на корню» параметром/GS.

Перенаправление указателя (pointer subterfuge) — перезапись локального указателя с целью поместить данные в нужное место. Параметр */GS* не в состоянии предотвратить атаку, если это место — не адрес возврата.

Атака на регистр (register attack) — перезапись значения, хранимого в регистре для получения управления. Иногда удается предотвратить

Захват VTable (VTable hijacking) — изменение локальной ссылки на объект так, чтобы вызов VTable приводил к запуску' нужной функции. Как правило, */GS* здесь не помогает.

Захламление обработчиков исключений (exception handler clobbering) — изменение кода обработки исключения, заставляющее систему выполнить подставленный взломщиком код. Параметр/GS здесь также не помогает.

Выход индекса за границы диапазона (index out of range) — использование индекса массива, который не проверяется на соответствие разрешенному диапазону. Параметр */GS* здесь также не помощник, за исключением случая изменения адреса возврата

МЕТОДЫ БЕЗОПАСНОГО КОДИРОВАНИЯ

Переполнения кучи (heap overflow) — принуждение диспетчера кучи к выполнению злой воли хакера. От этого /GS тоже не спасет.

Как утверждает Грег Хогланд (Greg Hoglund) в своих сообщениях на сайте NTBUGTRAQ, нельзя расслабляться, просто установив /GS.

Запрет на вызовы небезопасных функций — неплохой способ, но программисты все равно найдут возможность напортачить

Проверка кода — еще один хороший метод выявления ошибок, но проверяющий, как и автор кода, тоже человек, а значит, может ошибаться. Качество проверки кода напрямую зависит от опыта проверяющего, а также от того, насколько он свеж и бодр. Везение тоже не стоит сбрасывать со счетов.

Тщательное тестирование — еще один мощный инструмент, но кто из нас претендует на обладание идеальным планом тестирования?

Средства анализа кода — эта область пока находится в зачаточном состоянии. Их преимущество в том, что они всегда начеку и быстро анализируют миллионы строк кода.

ВЫБОР МЕХАНИЗМА УПРАВЛЕНИЯ ДОСТУПОМ

Стандартное и наименее знакомое большинству пользователей средство — *списки управления доступом* (Access Control List, ACL).

ОС Windows поддерживают ACL двух типов: *избирательную* (Discretionary Access Control List, DACL) и *системную* (System Access Control List, SACL) *таблицы управления доступом.* Первая управляет доступом к защищенным ресурсам, а вторая — аудитом защищенных ресурсов.

примеры ресурсов, доступ к которым управляется таблицами DACL, а аудит — таблицами SACL:

- файлы и каталоги;
- общие файлы (например \\BlakesLaptop\BabyPictures);
- разделы реестра;
- общая память;
- объекты-задания;
- мьютексы (mutex);
- именованные каналы (named pipes);
- принтеры;
- семафоры;
- объекты каталога Active Directory.

ВЫБОР МЕХАНИЗМА УПРАВЛЕНИЯ ДОСТУПОМ

Каждая DACL содержит несколько *записей управления доступом* (Access Control Entry, ACE), хотя она может быть и пустой.

- ***DACL, равная NULL*** – обеспечение полного доступа, что означает, что к ресурсу не применяются никакие механизмы управления доступом.
- ***«Нулевая» DACL*** — это очень плохо, и ее не следует применять, потому что злоумышленнику ничего не стоит установить свою политику доступа к объекту.
- ***ACE*** состоит из двух основных компонентов: учетной записи, представленной *идентификатором безопасности* (security ID, SID) и перечнем разрешенных действий над ресурсом.
- ***SID*** представляет учетную запись пользователя, группы или компьютера.

Выяснить, соответствует ли ACL целям приложения, просто:

1. определите ресурсы, которые нужны приложению;
2. выясните, какие требования по управлению доступом к ресурсам предписывают особенности бизнеса;
3. выберите подходящую технологию управления доступом;
4. реализуйте требования по управлению ресурсами в виде технологии управления доступом.

ОПАСНЫЕ ТИПЫ ACE

Everyone *(WRITE_DAC)*

WRITE_DAC — право изменять DACL в дескрипторе безопасности объекта. Получив возможность изменять ACL, злонамеренный пользователь может назначить себе ничем не ограниченный доступ к объекту и закрыть его для остальных

Everyone (WRITE_OWNER)

WRITE_OWNER — право изменять владельца в дескрипторе безопасности объекта. По определению, владелец объекта может делать с ним все, что угодно. Недобросовестный пользователь, присвоивший владение объектом, получает безграничный доступ и возможность закрыть объект для доступа других пользователей.

Everyone (FILE_ADD_FILE)

ACE с Everyone (FILE_ADD_FILE) особенно опасна, потому что позволяет ненадежным пользователям добавлять в файловую систему свои исполняемые программы. Опасность в том, что атакующий может разместить опасный файл в нужном каталоге и дождаться, когда администратор запустит программу на исполнение.

ОПАСНЫЕ ТИПЫ ACE

Everyone (DELETE)

ACE с таким разрешением позволяет удалить объект вашего приложения любому, а этого разрешать нельзя, особенно пользователям, которым вы не особенно доверяете

Everyone (FILE_DELETE_CHILD)

Это разрешение отображается в пользовательском интерфейсе Windows как *Delete subfolclers and files* (Удаление подпапок и файлов) и позволяет пользователю удалять дочерний объект, например файл, даже если у него нет к нему доступа. Обладая разрешением ILE_DELETE_CHILD на родительском объекте, вы вправе удалить любой дочерний объект независимо от разрешений последнего.

Everyone (GENERIC_ALL)

Разрешение *GENER1C_ALL,* или Full Control (Полный доступ), столь же опасно, как и *NULL DACL.* Лучше его не применять.

ТРИГГЕРЫ И РАЗРЕШЕНИЯ СЕРВЕРА SQL SERVER

Триггеры SQL Server позволяют разработчику размещать правила произвольной сложности в таблицах базы данных. Ядро базы данных автоматически вызывает триггеры при добавлении, удалении или изменении данных в таблицах.

Разрешения SQL Server — это аналоги списков ACL в Windows, а механизм их действия можно выразить простой фразой: «субъекту разрешается (или запрещается) выполнять определенные операции над объектом».

Обратите внимание, что триггеры не вызываются при записи данных в журнал аудита, так как в соответствии с бизнес-правилами запись разрешается всем. Однако такое решение не лишено недостатков: читать информацию журнала аудита могут все, так как триггеры не реагируют на чтение. В этом случае логично применить к таблице разрешение, например: «обычным пользователям (public) разрешается только запись в журнал». «Обычные пользователи» — это то же, что и группа Everyone в Windows.

В данном сценарии разрешения таблицы и триггеры работают в паре, надежно закрывая журнал от злоумышленников (даже если администратор случайно удалит разрешение из контрольной таблицы), обеспечивая защиту «в глубину».

В отсутствие качественной реализации шифрования и управления ключами списки ACL становятся последним форпостом защиты, способным остановить прорвавшего остальные заслоны взломщика

ОПРЕДЕЛЕНИЕ ОПТИМАЛЬНОГО НАБОРА ПРИВИЛЕГИЙ

Принцип минимальных привилегий предполагает сокращение времени работы с повышенными полномочиями — это уменьшает интервал, когда злоумышленник может воспользоваться недостатками системы. В Windows дополнительные права разрешается назначить прямо перед выполнением задачи, а после снова отозвать их.

Этап 1: выясните, какие ресурсы нужны приложению

Следует инвентаризовать ресурсы, необходимые для работы приложения: файлы, разделы реестра, сокеты, данные Active Directory, именованные каналы и т. п., а также определить, какой уровень доступа требуется для того или иного ресурса.

Этап 2: выясните, какими системными API-функциями пользуется приложение

Создайте список всех функций, для работы которых необходимы привилегии.

Этап 3: определите, какая требуется учетная запись

Опишите учетную запись, необходимую для нормальной работы приложения.

Этап 4: исследуйте содержимое маркера

Выясните, какие SID и привилегии содержатся в маркере учетной записи, определенной на предыдущем этапе. Для этого или войдите в систему под этой учетной записью, или используйте команду RunAs для запуска нового экземпляра командного процессора.

Этап 5: выясните необходимость всех привилегий и SID-идентификаторов

Совместно с представителями групп проектирования, разработки и тестирования проанализируйте каждый SID и привилегию, пытаясь выяснить, без каких вы обойдетесь.

Этап 6: внесите изменения в маркер

Ограничение возможностей маркера: • разрешить выполнение приложения под другими, менее привилегированными, учетными записями; • использовать *ограниченные маркеры* (restricted tokens); • радикально удалить ненужные привилегии.

СЛУЧАЙНЫЕ ЧИСЛА КРИПТОГРАФИЧЕСКОГО КАЧЕСТВА В WIN32

Никогда не вызывайте rand, пользуйтесь более надежными источниками случайных данных в Windows, например ***CryptGenRandom,*** которая обладает двумя свойствами хорошего генератора случайных чисел — непредсказуемостью и равномерным распределением.

Функция CryptGenRandom опирается на случайность [ее также называют *системной энтропией* (system entropy)], которая создается на основе многих источников, в том числе с использованием:

- идентификатора текущего процесса *(GetCurrentProcessID);*
- идентификатора текущего потока *(GetCurrentThreadID);*
- числа тактов процессора с момента загрузки *(GetTickCount);*
- текущего времени *(GetLocalTime).*
- показаний различных высокоточных счетчиков производительности (QueryPerformanceCountef);
- MD4-xenia блока пользовательского окружения, куда входит имя пользователя, имя компьютера и путь поиска;
- показаний высокоточных внутренних счетчиков процессора, таких как RDMSR, RDPMC;
- низкоуровневой системной информации, показаний счетчиков производительности: Idle Process Time, Io Read Transfer Count и т.д.;
- информации о системных исключениях, в том числе показаний счетчиком Alignment Fix Up Count, Exception Dispatch Count, Floating Emulation Count и Byte Word Emulation Count;
- информации, хранимой системой, в том числе показаний счетчиков: Current Depth, Maximum Depth, Total Allocates, Allocate Misses. Total Frees, Free Misses Type, Tag и Size;
- информации о системных прерываниях, в том числе показаний счетчиков Context Switches, Deferred Procedure Call Count, Deferred Procedure Call Rate, Time Increment, Deferred Procedure Call Bypass Count и Asynchronous Procedure Call Bypass Count;
- системной информации о процессах, в том числе показаний счетчиков: Net Entry Offset, Number Of Threads, и .т.д.

СЛУЧАЙНЫЕ ЧИСЛА КРИПТОГРАФИЧЕСКОГО КАЧЕСТВА НА WEB-СТРАНИЦАХ

В приложениях ASP.NET очень легко получить качественные случайные числа, просто вызвав управляемые классы.

ОЦЕНКА ЭФФЕКТИВНОЙ ДЛИНЫ ПАРОЛЯ

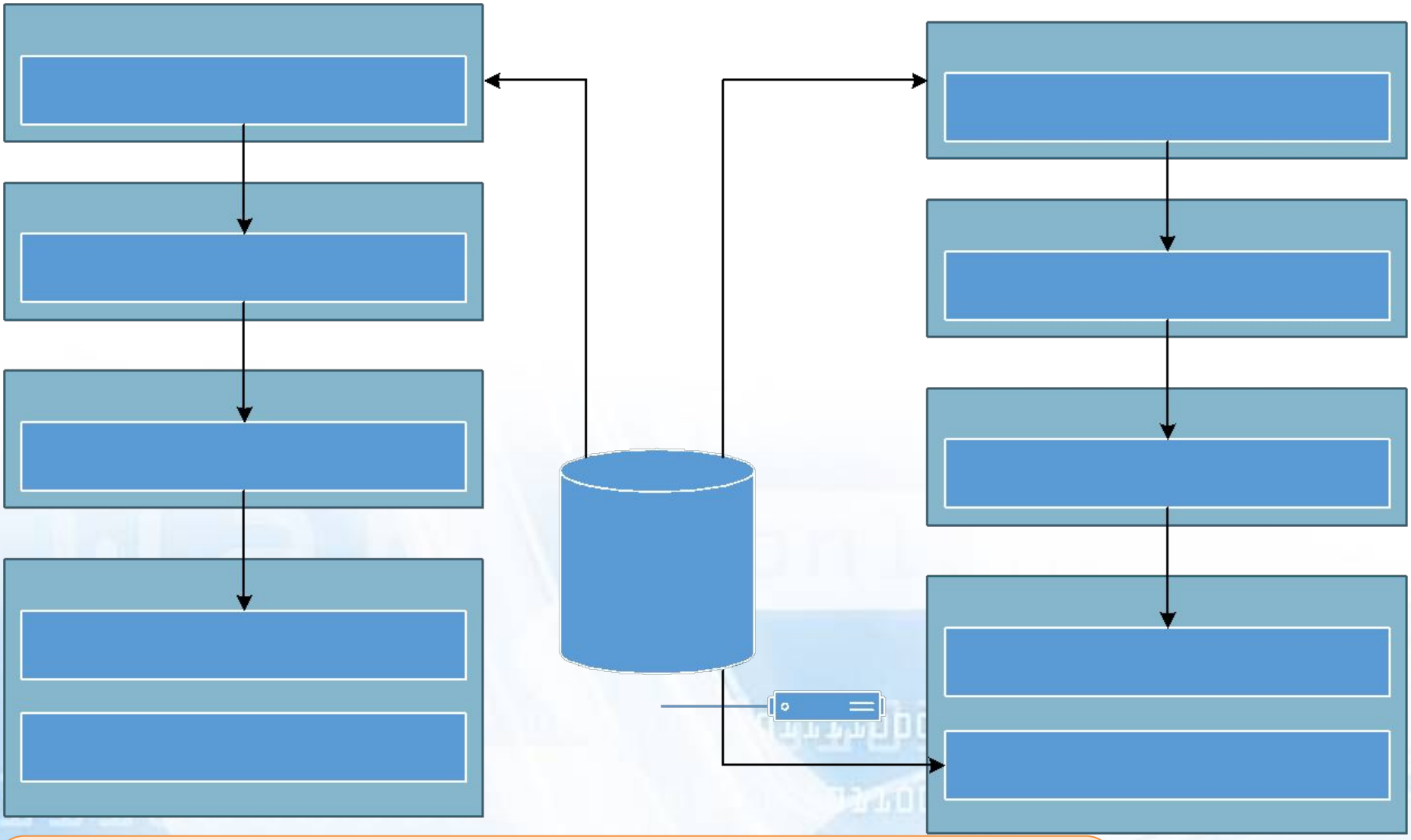
число «информативных» бит в случайно взятом пароле составляет $\mathbf{\log_2(n^m)}$, где n — размер множества допустимых символов, а m — длина пароля.

Число полезных бит в пароле очень важно при вычислении его надежности, но также следует принимать во внимание то, насколько легко его угадать.

МНОЖЕСТВА ДОСТУПНЫХ СИМВОЛОВ И ДЛИНЫ ПАРОЛЕЙ ДЛЯ КЛЮЧЕЙ РАЗНОЙ ДЛИНЫ

Вариант	Доступные символы	Необходимая длина для 56-го ключа	Необходимая длина для 128-го ключа
Числовой PIN-код	10 (0 – 9)	17	40
Буквы без учета регистра	26 (A – Z, или a – z)	12	28
Буквы с учетом регистра	52 (A – Z и a – z)	10	23
Буквы с учетом регистра и цифры	52 (A – Z, a – z и 0 - 9)	10	22
Буквы с учетом регистра и цифры, знаки пунктуации	93 (A – Z, a – z и 0 – 9, знаки пунктуации)	9	20

КЛЮЧИ, «ПУТЕШЕСТВУЮЩИЕ» ПО ВСЕМУ ПРИЛОЖЕНИЮ



Секретные данные, в том числе пароли, больше подвержены компрометации, если передаются между компонентами приложения, а не хранятся централизованно и не обрабатываются локально.

Хеш с модификатором данных

Чтобы усложнить взломщику задачу, часто добавляют модификатор хеша. ***Модификатор (salt)*** — это случайное число, которое добавляется к хешируемым данным и позволяет предотвратить «взлом по словарю», сильно усложняя задачу расшифровки сообщения. ***Взломом по словарю (dictionary attack)*** называется такая атака, когда пытаются расшифровать текст, подставляя известные слова и комбинации символов из предварительно созданного массива, или словаря.

Создавать хеш с модификатором или простой верификатор с помощью функций ***CryptoAPI*** *очень просто.*

Функция ***CryptGetHashParam*** *интерфейса Windows API добавляет данные к хешу и выполняет повторное хеширование, эта операция так же эффективна, как* ***CryptoAPI***

Применение PKCS #5 (Public-Key Cryptography Standard)

В ***PKCS#5*** пароль с модификатором хешируется несколько сотен или тысяч раз подряд. Или, если быть совсем точным, так работает алгоритм ***PBKDF1*** (***Passwoid Based Key Derivation Function #1***) — наиболее популярный вариант PKCS #5. Другой вариант, ***PBKDF2***, немного отличается: в нем применяется генератор псевдослучайных чисел.

Основное назначение PKCS #5 — защита от атак по словарю, так как при переборе паролей на обсчет каждого уходит много процессорного времени. Во многих программах хранят только хеш пароля, а при аутентификации хеш введенного пользователем пароля просто сравнивается с имеющимся в системе. Вы значительно затрудните задачу хакеру, храня не хеш, а значение, вычисленное по методу PKCS #5.

Стандартные криптопровайдеры ***CryptoAPI*** *в Windows не поддерживают напрямую PKCS #5, однако есть функция* ***CryptDeriveKey****, которая обеспечивает примерно такой же уровень защиты.*

ЗАЩИТА СЕКРЕТНЫХ ДАННЫХ

Защита секретов в Windows

Для сохранения секретных данных пользователя в Windows 2000 применяют функции ***CryptProtectData*** и ***CryptUnprotectData*** API-интерфейса DPAPI (***Data Protection API***). DPAPI поддерживает два способа хранения секретов: когда доступ к данным предоставляется только их владельцу и когда данные доступны любому пользователю компьютера.

Защищая данные путем установки флага CRYPTPRQTECT_LOCAL_MACHINE, обязательно делайте резервную копию шифрованного текста. В противном случае при критическом сбое компьютера, сопровождающимся разрушением операционной системы, ключ потеряется и доступ к данным станет невозможным

Управление секретами в памяти

Последовательность работы с секретными данными в памяти такова:

- получение секретных данных;
- обработка и использование секретных данных;
- отбрасывание секретных данных;
- очистка памяти.

Время между получением секретных данных и очисткой памяти должно быть *минимальным*, что позволит избежать сброса секретных данных в ***страничный файл***. Впрочем, угроза воровства секретных данных из страничного файла довольно мала.

ЗАЩИТА СЕКРЕТНЫХ ДАННЫХ

Управление секретами в памяти

Закончив работу с секретами, перезапишите буфер посторонней информацией (или просто обнулите его) вызовам ***memset*** или ***ZeroMemory***.

Шифрование секретных данных в памяти

В Windows .NET Server 2003-2008 имеются две API-функции, ***CryptProtect-Memory*** и ***CryptUnprotectMemory***, они выдержаны в идеологии DPAPI, но защищают данные в оперативной памяти.

Основной ключ шифрования данных обновляется при каждой загрузке компьютера, материал для ключей определяется флажками, передаваемыми в функции. Когда применяются эти функции, приложению не за чем «видеть» ключ шифрования.

Блокировка памяти для предотвращения выгрузки секретной информации на диск

Предотвратить выгрузку секретной информации в страничный файл можно, заблокировав данные в памяти. Однако делать этого настоятельно не рекомендуется, так как блокировка не дает ОС эффективно управлять памятью. Блокировать память (вызовом таких функций, как ***Allocate User Physical Pages*** и ***Virtual Lock***) следует очень осмотрительно и применительно только к очень секретным данным.

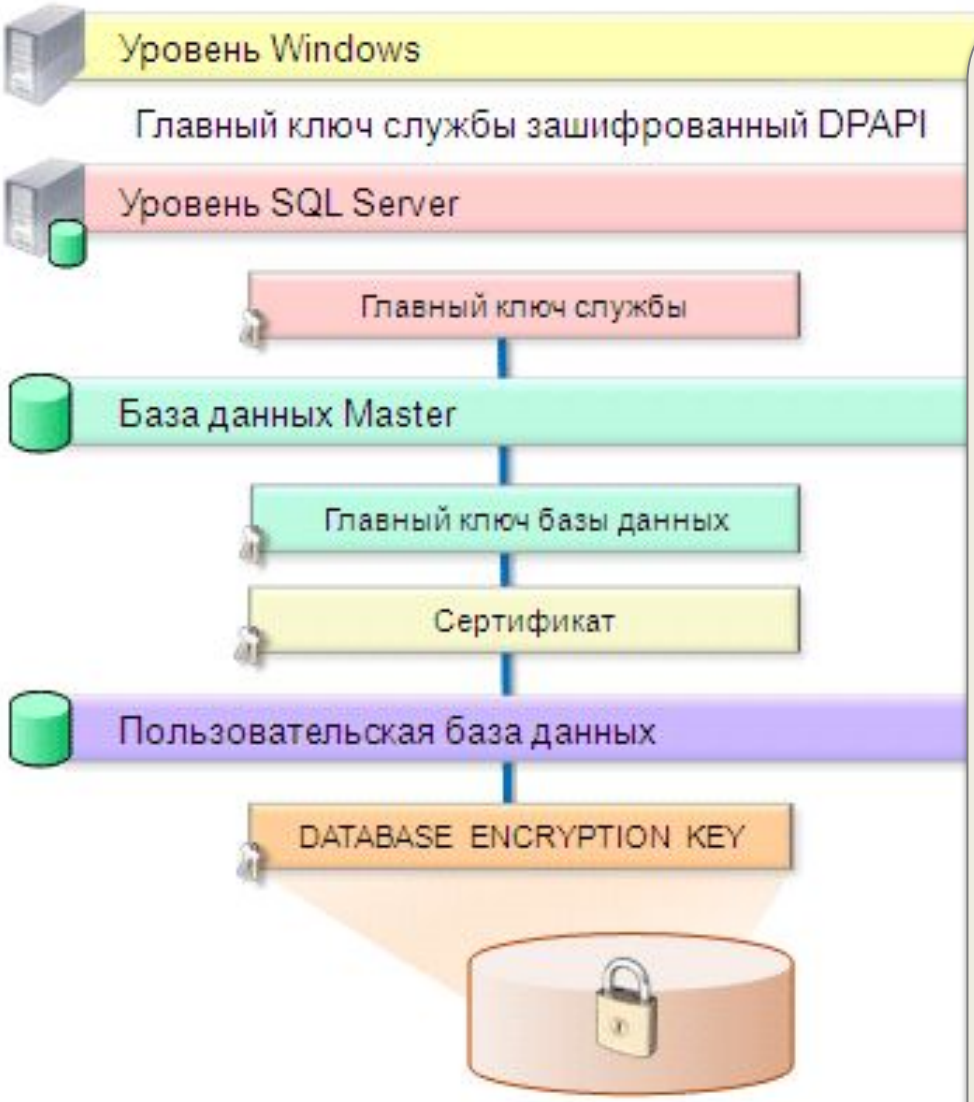
ЗАЩИТА СЕКРЕТНЫХ ДАННЫХ В СУБД

В Microsoft SQL Server 2008 появилось новое решение для обозначенных выше проблем – это прозрачное шифрование БД (***Transparent Data Encryption*** или TDE). TDE позволяет шифровать базы данных целиком. Когда страница данных сбрасывается из оперативной памяти на диск, она шифруется. Когда страница загружается обратно в оперативную память, она расшифровывается.

Таким образом, база данных на диске оказывается полностью зашифрованной, а в оперативной памяти – нет.

Основным преимуществом TDE является то, что шифрование и расшифровка выполняются абсолютно прозрачно для приложений. Следовательно, получить преимущества от использования TDE может любое приложение, использующее для хранения своих данных Microsoft SQL Server 2008. При этом модификации или доработки приложения не потребуется.

ЗАЩИТА СЕКРЕТНЫХ ДАННЫХ В СУБД



В контексте Transparent Data Encryption (TDE) она строится следующим образом:

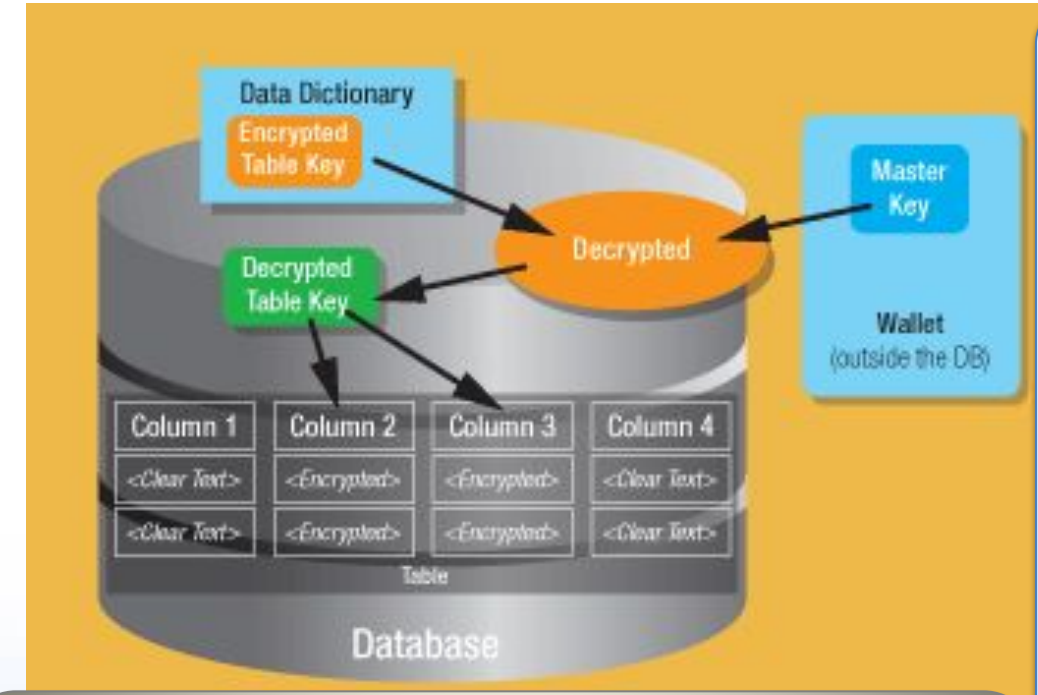
- Для каждой базы данных, которая шифруется с помощью Transparent Data Encryption (TDE) создается специальный ключ – Database Encryption Key (DEK). Этот ключ используется для шифрования данных.
- Database Encryption Key (DEK) шифруется сертификатом, который должен быть создан в БД master.

Далее стандартно:

- Этот сертификат шифруется главным ключом БД master.
- Главный ключ БД master шифруется главным ключом службы (Service Master Key или SMK).
- Главный ключ службы (SMK) шифруется с помощью DPAPI.

Если злоумышленник смог получить доступ к защищаемым данным через SQL Server, то Transparent Data Encryption (TDE) оказывается абсолютно бесполезным.

ОСНОВЫ ШИФРОВАНИЯ В СЕРВЕРЕ ORACLE



Data Dictionary – словарь данных;
Encrypted Table Key – зашифрованный ключ таблицы;
Master Key – главный ключ; Decrypted – расшифрованный; Wallet (outside the DB) – бумажник (за пределами базы данных);
Decrypted Table Key – расшифрованный ключ таблицы;
Column – столбец; Clear Text – обычный текст;
Encrypted – зашифрованный; Table – таблица;
Database – база данных.

По умолчанию для шифрования используется алгоритм AES с 192-битовым ключом.

- определить столбец, который будет шифроваться, и сервер Oracle Database 10g создаст криптографически стойкий ключ шифрования для таблицы, содержащей этот столбец,
- зашифрует данные обычного текста в этом столбце, используя указанный вами алгоритм шифрования.
- сервер Oracle Database 10g шифрует его, используя главный ключ, который хранится в безопасном месте, называемом бумажником (wallet), который может быть файлом сервера базы данных.
- Зашифрованные ключи таблиц размещаются в словаре данных.
- Когда пользователь вставляет данные в столбец, определенный как зашифрованный, сервер Oracle Database 10g извлекает из бумажника главный ключ, расшифровывает ключ шифрования для этой таблицы, находящийся в словаре данных, использует этот ключ для шифрования входного значения и сохраняет зашифрованные данные в