Посвящается светлой памяти
Нэнси Симоне Мак-Гроу
(1939–2003)

Предисловие

В начале июля 2003 года мне позвонил Дэвид Дилл (David Dill) — профессор в области информационных технологий Стэнфордского университета. Дилл рассказал мне, что в сети Internet был опубликован исходный код для электронной машины голосования, которая была выпущена одним из крупнейших поставщиков подобного оборудования, компанией Diebold Electron Systems. Он сказал, что стоит проверить этот исходный код на наличие уязвимых мест. Такая возможность выпадает нечасто, поскольку производители систем голосования обычно тщательно охраняют разработанное ими программное обеспечение. Нас потрясло то, что мы обнаружили: ошибок в программном коде, включая и те, которые имели отношение к безопасности, было настолько много, что хакер мог бы просто растеряться, какую атаку из всего доступного диапазона средств для взлома использовать первой (мы *не* рекомендуем использовать такой метод, чтобы приводить хакеров в замешательство). Мы исследовали большие, сложные фрагменты кода, к которому не давалось никаких пояснений. В этом коде использовался только один статический ключ для шифрования результатов голосования. Использовались небезопасные генераторы псевдослучайных цифр и контрольные суммы без шифрования. Исследование журналов CVS (Concurrent Versions System — система управления параллельными версиями) показало, что применялся собственный, "нестандартный" метод управления разработкой программного кода.

Является ли пример с машиной голосования компании Diebold единственным примером плохого контроля за качеством программного обеспечения? Мне так не кажется. Многие компании наподобие Diebold стремятся выбросить на рынок свои продукты раньше конкурентов. Побеждает компания с лучшей и наиболее богатой с точки зрения функциональных возможностей системой. Согласно модели стимулирования, шансы на победу выше у той компании, которая первой представит на рынке свой продукт, в котором будет максимум возможностей, а не той, программа которой будет более безопасна в использовании. Создать грамотную систему безопасности довольно сложно, а результат не всегда оправдывается в материальном смысле. Компании Diebold просто не повезло: программный код ее продукта был обсужден в открытом форуме, что показало возможность его полного взлома. Большинство компаний чувствуют себя в относительной безопасности при условии, что независимые аналитики получают возможность просмотра их кода при жестких ограничениях неразглашения. Только когда они попадают под огонь критики, такие компании уделяют внимание разрекламированными ими ранее средствам обеспечения безопасности. Программный код для машины голосования компании Diebold был не первой исследованной мной сложной системой с массой ошибок, "затрагивающих" безопасность программы. Почему же так трудно создать безопасное программное обеспечение?

Ответ прост. Все дело в *сложности*. Любой программист знает, что для решения одной и той же задачи с помощью средств программирования существует множество приемлемых вариантов. Важным вариантом выбора является язык программирования. Нужно ли вам удобство арифметических указателей и предоставляемых ими возможностей для оптимизации производительности программ, или требуется "экономный" язык, который не допускает переполнения буферов, но лишает программиста некоторых возможностей? Для решения каждой задачи существует множество

доступных алгоритмов, параметров и структур данных. Для каждого блока кода можно выбирать разные имена переменных, способ комментария и даже метод его добавления к основному тексту программы. Каждый программист — уникальная личность, и каждый делает свой уникальный выбор. Большие проекты создаются командами программистов, и различные программисты должны понимать и изменять код других программистов. Достаточно трудно отредактировать свой собственный код, не говоря уж о программном обеспечении другого программиста. Очень трудно избежать ошибок в средствах безопасности в программах, состоящих из сотен строк кода, а для программ с миллионами строк кода, например, в современных операционных системах, это практически невозможно.

Однако сложные системы должны создаваться, и мы не можем просто сдаться и сказать, что безошибочное программирование подобных систем невозможно. Мак-Гроу и Хогланд провели огромную работу, чтобы объяснить, почему программное обеспечение можно взламывать, рассказали, как именно работают программы атаки и как избежать создания уязвимых программ. Вы можете удивиться: для чего демонстрировать, как работают программы взлома? Действительно, существует компромисс, который должны учитывать профессионалы в области безопасности, между сохранением в тайне и опубликованием программ атаки. В этой книге принята здравая позиция, согласно которой единственный способ минимизировать количество ошибок — это понять, почему появляются уязвимые места и как их используют хакеры. Поэтому эту книгу стоит прочесть любому человеку, который принимает участие в создании сетевого приложения или операционной системы.

Взлом программного обеспечения: руководство начинающего хакера — это лучшая книга по теме обеспечения защиты приложений и уязвимых мест в программном обеспечении, которую я когда-либо читал. Гари Мак-Гроу и Грег Хогланд давно изучают эту науку. Первая книга Мак-Гроу, *Java Security* (Безопасность в Java), стала основополагающим трудом по проблемам безопасности в среде выполнения Java-приложений и концепции Novel по запуску непроверенного переносимого кода в пределах надежного браузера. Следующая книга Мак-Гроу, *Building Secure Software (Создание безопасных приложений)*, стала классическим трудом, в котором продемонстрированы принципы построения приложений без уязвимых мест, многие из которых описаны в этой книге. Хогланд имеет очень большой опыт разработки приложений и практической реализации защиты от программ атаки.

После чтения этой книги вы будете удивляться не тому, как много систем было взломано, а тому, сколько систем еще не взломано. Проведенный нами анализ программного кода электронной машины для голосования показал, что уязвимое программное обеспечение присутствует повсюду. Тот факт, что многие системы еще не скомпрометированы, объясняется тем, что злоумышленники часто удовлетворяются более легкими целями. Лично мне теперь будет не совсем приятно в следующий раз идти на выборы, которые обслуживаются с помощью электронной машины голосования, работающей под управлением Windows. Возможно, я просто использую открепительный талон для голосования, по крайней мере, в этом случае ошибка не будет вызвана недостатками программного обеспечения.

Авиэль Д. Рубин (Aviel D.Rubin)
Адъюнкт-профессор компьютерных наук, технический руководитель института по информационной безопасности при университете Джона Хопкинса