Моей потрясающей семье.
Ничто не может сравниться с ощущением,
которое испытываешь, когда приходишь домой
и в ответ на вопрос «Кто дома, ребятки?» слышишь,
как два голоска хором кричат: «Папа!»
– Майкл

Моему отцу, который объяснил мне,
почему надо постоянно учиться
и принимать новые вызовы.
– Дэвид

Маме. Она привила мне интеллектуальное
любопытство и всегда была со мной рядом.
– Джон

Содержание

Об авторах

Майкл Ховард работает старшим менеджером по безопасности программного обеспечения в группе по обеспечению безопасности в Microsoft Corp. Является соавтором удостоенной различных наград книги «Writing Secure Code» (Разработка безопасного кода). Он также совместно с коллегами ведет колонку «Basic Training» в журнале «IEEE Security & Privacy Magazine» и является одним из авторов документа «Processes to Produce Secure Software» («Процессы в производстве безопасного программного обеспечения»), выпущенного организацией National Cyber Security Partnership для Министерства национальной безопасности (Department of Homeland Security). Будучи архитектором «Жизненного цикла разработки безопасного программного обеспечения» в Microsoft, Майкл посвящает большую часть времени выработке и внедрению передового опыта создания безопасных программ, которыми в конечном итоге будут пользоваться обычные люди.

Дэвид Лебланк, доктор философии, в настоящее время работает главным архитектором программ в компании Webroot Software. До этого он занимал должность архитектора подсистемы безопасности в подразделении Microsoft, занимающемся разработкой Microsoft Office, стоял у истоков инициативы Trustworthy Computing и работал «белым хакером» в группе безопасности сетей в Microsoft. Дэвид является соавтором книг «Writing Secure Code» и «Assessing Network Security» («Оценка безопасности сети»), а также многочисленных статей. В погожие дни он любит конные прогулки вместе со своей женой Дженнифер.

Джон Виега первым дал описание 19 серьезных просчетов при написании программ. Этот труд привлек внимание средств массовой информации и лег в основу настоящей книги. Джон является основателем и техническим директором компании Secure Software (www.securesoftware.com). Он один из авторов первой книги по безопасности программного обеспечения «Building Secure Software» («Создание безопасного программного обеспечения»), а также книг «Network Security and Cryptography with OpenSSL» («Безопасность и криптографические методы в сетях. Подход на основе библиотеки OpenSSL») и «Secure Programming Cookbook» («Рецепты для написания безопасных программ»). Он является основным автором процесса CLASP, призванного включить элементы безопасности в цикл разработки программ. Джон написал и сопровождает несколько относящихся к безопасности программ с открытыми исходными текстами. Раньше Джон занимал должности адъюнкт-профессора в техническом колледже штата Вирджиния и старшего научного сотрудника в Институте стратегии кибепространства (Cyberspace Policy Institute). Джон хорошо известен своими работами в области безопасности программ и криптографии, а в настоящее время он трудится над стандартами безопасности для сетей и программ.

О научных редакторах

Алан Крассовски работает главным инженером по безопасности программного обеспечения в компании Symantec Corporation. Он возглавляет группу по безопасности продуктов, в задачу которой входит оказание помощи другим группам разработчиков в плане внедрения безопасных технологий, которые сокращают риски и способствуют завоеванию доверия со стороны клиентов. За последние 20 лет Алан работал над многими коммерческими программными проектами. До присоединения к Symantec он руководил разработками, был инженером-программистом и оказывал консультативные услуги многим компаниям, занимающим лидирующее положение в отрасли, в частности Microsoft, IBM, Tektronix, Step Technologies. Screenplay Systems, Quark и Continental Insurance. Он получил научную степень бакалавра в области вычислительной техники в Рочестерском технологическом институте, штат Нью-Йорк.

Дэвид А. Уилер много лет занимается совершествованием практических методов разработки программ для систем с повышенным риском, в том числе особо крупных и нуждающихся в высокой степени безопасности. Он соавтор и соредактор книги «Software Inspection: An Industry Best Practice» («Инспекция программ: передовой опыт»), а также книг «Ada95: The Lovelace Tutorial» и «Secure Programming for Linux and UNIX HOWTO» («Рецепты безопасного программирования для Linux и UNIX»). Проживает в Северной Вирджинии.

Предисловие

В основе теории компьютеров лежит предположение о детерминированном поведении машин. Обычно мы ожидаем, что компьютер будет вести себя так, как мы его запрограммировали. На самом деле это лишь приближенное допущение. Современные компьютеры общего назначения и их программное обеспечение стали настолько сложными, что между щелчком по кнопке мыши и видимым результатом лежит множество программных слоев. И мы вынуждены полагаться на то, что все они работают правильно.

Любой слой программного обеспечения может содержать ошибки, из-за которых оно работает не так, как хотел автор, или, по крайней мере, не соответствует ожиданиям пользователя. Эти ошибки вносят в систему неопределенность, что может приводить к серьезным последствиям с точки зрения безопасности. Проявляться они могут по-разному: от простого краха системы, и тогда ошибку можно использовать, чтобы вызвать отказ от обслуживания, до переполнения буфера, позволяющего противнику выполнить в системе произвольный код.

Коль скоро поведение программных систем недетерминировано из-за ошибок, то самые лучшие идеи по их защите – не более чем гипотезы. Мы можем воздвигать межсетевые экраны, реализовывать технологии защиты от переполнения буфера на уровне ОС, применять самые разнообразные методики, но все это никоим образом не изменит фундаментальную парадигму безопасности. И лишь за счет радикального улучшения качества программ и сокращения числа ошибок мы можем надеяться на успешность попыток обеспечить безопасность программного обеспечения.

Устранение всех рисков, относящихся к безопасности, – нереальная задача при современном уровне развития систем разработки. У этой проблемы так много аспектов, что, даже для того чтобы просто оставаться в курсе дел, нужно посвящать этому все свое время. Что уж говорить о владении предметом в совершенстве!

Если мы хотим добиться прогресса в битве против ошибок, связанных с безопасностью, то должны облегчить процесс их идентификации и устранения организациям, занимающимся разработкой, и при этом учесть реальные ограничения. О безопасности программного обеспечения написано немало отличных книг, в том числе и авторами настоящего издания. Но я полагаю необходимым не углубляться в разного рода сложности, а предложить разработчикам небольшой набор критически важных советов, следуя которым они смогут повысить качество своих программ с минимальными усилиями. Идея в том, чтобы осветить наиболее типичные проблемы, которые нетрудно устранить, а не ставить нереалистичную задачу достижения полной безопасности.

В бытность начальником отдела в Министерстве национальной безопасности я попросил Джона Виегу составить перечень 19 «грехов» программиста. Первоначальный список был призван поставить корпоративный мир в известность о тех ошибках, которые чаще всего угрожают безопасности, но он не был составлен в форме рецептов. А эта книга именно такова. В ней приводится список проблем, от которых организации-разработчики должны защищаться в первую очередь, и даются рекомендации, как не допустить самого возникновения этих проблем. В книге также показано, как выявить подобные ошибки: посредством анализа кода или тестирования. Описание приемов и методик краткое и точное, авторы четко формулируют, что надо, а чего никогда не надо делать. Авторы проделали огромную работу, чтобы представить вашему вниманию список наиболее распространенных дефектов, от которых страдает безопасность современных программ. Надеюсь, что сообщество разработчиков оценит эту книгу и воспользуется ей для устранения недетерминизма и рисков, с которыми мы постоянно сталкиваемся.

Амит Йоран,
бывший начальник
отдела национальной кибербезопасности
Министерства национальной безопасности
Грейт Фоллс, Вирджиния,
21 мая 2005 г.

Благодарности

Эта книга – косвенный результат дальновидности Амита Йорана. Мы благодарны ему за то, что во время работы в Министерстве национальной безопасности (и позже) он делал все возможное, чтобы привлечь внимание к проблемам безопасности программного обеспечения. Мы также выражаем признательность следующим специалистам в области безопасности за усердие, с которым они рецензировали черновики отдельных глав, за их мудрость и за откровенные комментарии: Дэвиду Рафаэлю (David Raphael), Марку Кэрфи (Mark Curphy), Рудольфу Араю (Rudolph Arauj), Алану Крассовски (Alan Krassowski), Дэвиду Уилеру (David Wheeler) и Биллу Хильфу (Bill Hilf). Эта книга не состоялась бы без настойчивости сотрудников издательства McGraw-Hill. Большое спасибо трем «Дж»: Джейн Браунлоу (Jane Brownlow), Дженнифер Хауш (Jennifer Housh) и Джоди Маккензи (Jody McKenzie).

Введение

В 2004 году Амит Йоран, тогда начальник отдела национальной кибербезопасности Министерства национальной безопасности США, объявил, что около 95% всех дефектов программ, относящихся к безопасности, проистекают из 19 типичных ошибок, природа которых вполне понятна. Мы не станем подвергать сомнению ваши интеллектуальные способности и объяснять важность безопасного программного обеспечения в современном взаимосвязанном мире, вы и так все понимаете, но приведем основные принципы поиска и исправления наиболее распространенных ошибок в вашем собственном коде.

Неприятная особенность ошибок, касающихся безопасности, состоит в том, что допустить их очень легко, а результаты одной неправильно написанной строчки могут быть поистине катастрофическими. Червь Blaster смог распространиться из-за ошибки всего в двух строках кода.

Если попытаться выразить весь накопленный опыт одной фразой, то, наверное, она звучала бы так: «Никакой язык программирования, никакая платформа не способны сделать программу безопасной, это можете сделать только вы». Существует масса литературы о том, как создавать безопасное программное обеспечение, да и авторы настоящей книги написали на эту тему немало текстов, к которым прислушиваются. И все же есть потребность в небольшой, простой и прагматической книге, в которой рассматривались бы все основные проблемы.

Работая над этой книгой, мы старались придерживаться следующих правил, которые не позволили бы оторваться от земли.

- *Простота*. Мы не тратили место на пустую болтовню. Здесь вы не найдете ни репортажей с поля боя, ни забавных анекдотов – только голые факты. Скорее всего, вы просто хотите сделать свою работу качественно и в кратчайшие сроки. Поэтому мы стремились к тому, чтобы найти нужную информацию можно было просто и быстро.
- *Краткость*. Это следствие предыдущего правила: сосредоточившись исключительно на фактах, мы смогли сделать книгу небольшой по объему. Это введение тоже не будет многословным.
- *Кроссплатформенность*. Интернет – это среда, связывающая между собой мириады вычислительных устройств, работающих под управлением разных операционных систем и программ, написанных на разных языках. Мы хотели, чтобы эта книга была полезна всем разработчикам, поэтому представленные примеры относятся к большинству имеющихся операционных систем.
- *Многоязычие*. Следствие предыдущего правила: мы приводим примеры ошибок в программах, которые составлены на разных языках.

Структура книги

В каждой главе описывается один «смертный грех». Вообще-то они никак не упорядочены, но самые гнусные мы разместили в начале книги. Главы разбиты на разделы:

- **«В чем состоит грех»** – краткое введение, в котором объясняется, почему данное деяние считается грехом;
- **«Как происходит грехопадение»** – описывается суть проблемы; принципиальная ошибка, которая доводит до греха;
- **«Подверженные греху языки»** – перечень языков, подверженных данному греху;
- **«Примеры ошибочного кода»** – конкретные примеры ошибок в программах, написанных на разных языках и работающих на разных платформах;
- **«Где искать ошибку»** – на что нужно прежде всего обращать внимание при поиске в программе подобных ошибок;
- **«Выявление ошибки на этапе анализа кода»** – тут все понятно: как найти грехи в своем коде. Мы понимаем, что разработчики – люди занятые, поэтому старались писать этот раздел коротко и по делу;
- **«Тестирование»** – описываются инструменты и методики тестирования, которые позволят обнаружить признаки рассматриваемого греха;
- **«Примеры из реальной жизни»** – реальные примеры данного греха, взятые из базы данных типичных уязвимостей и брешей (Common Vulnerabilities and Exposures – CVE) (www.cve.mitre.org), с сайта BugTraq (www.securityfocus.com) или базы данных уязвимостей в программах с открытыми исходными текстами (Open Source Vulnerability Database) (www.osvdb.org). В каждом случае мы приводим свои комментарии. Примечание: пока мы работали над этой книгой, рассматривался вопрос об отказе с 15 октября 2005 года от номеров CAN в базе данных CVE и переходе исключительно на номера CVE. Если это случится, то все ссылки на номер ошибки «CAN...» следует заменить ссылкой на соответствующий номер CVE. Например, если вы не сможете найти статью CAN-2004-0029 (ошибка Lotus Notes для Linux), попробуйте поискать CVE-2004-0029;
- **«Искупление греха»** – как исправить ошибку, чтобы избавиться от греха. И в этом случае мы демонстрируем варианты для разных языков;
- **«Дополнительные защитные меры»** – другие меры, которые можно предпринять. Они не исправляют ошибку, но мешают противнику воспользоваться потенциальным дефектом, если вы ее все-таки допустите;
- **«Другие ресурсы»** – это небольшая книжка, поэтому мы даем ссылки на другие источники информации: главы книг, статьи и сайты;
- **«Резюме»** – это неотъемлемая часть главы, предполагается, что вы будете к ней часто обращаться. Здесь приводятся списки рекомендуемых, нерекомендуемых и возможных действий при написании нового или анализе существующего кода. Не следует недооценивать важность этого раздела! Содержание всех Резюме сведено воедино в Приложении В.