

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 .....	18
О научных редакторах .....	19
Предисловие .....	20
Благодарности .....	22
Введение .....	23
Структура книги .....	24
Кому предназначена эта книга .....	25
Какие главы следует прочитать .....	25
<b>Грех 1. Переполнение буфера .....</b>	<b>26</b>
В чем состоит грех .....	26
Подверженные греху языки .....	27
Как происходит грехопадение .....	27
Греховность C/C++ .....	31
Родственные грехи .....	33
Где искать ошибку .....	33
Выявление ошибки на этапе анализа кода .....	33
Тестирование .....	34
Примеры из реальной жизни .....	35
CVE-1999-0042 .....	35
CVE-2000-0389 – CVE-2000-0392 .....	35
CVE-2002-0842, CVE-2003-0095, CAN-2003-0096 .....	35
CAN-2003-0352 .....	36
Искупление греха .....	37
Замена опасных функций работы со строками .....	37
Следите за выделениями памяти .....	37
Проверьте циклы и доступ к массивам .....	37
Пользуйтесь строками в стиле C++, а не C .....	37
Пользуйтесь STL-контейнерами вместо статических массивов .....	38

Пользуйтесь инструментами анализа .....	38
Дополнительные защитные меры .....	38
Защита стека .....	39
Запрет исполнения в стеке и куче .....	39
Другие ресурсы .....	39
Резюме .....	40
<b>Грех 2. Ошибки, связанные с форматной строкой .....</b>	<b>42</b>
В чем состоит грех .....	42
Подверженные греху языки .....	42
Как происходит грехопадение .....	43
Греховность C/C++ .....	45
Родственные грехи .....	45
Где искать ошибку .....	46
Выявление ошибки на этапе анализа кода .....	46
Тестирование .....	46
Примеры из реальной жизни .....	47
CVE-2000-0573 .....	47
CVE-2000-0844 .....	47
Искупление греха .....	47
Искупление греха в C/C++ .....	48
Дополнительные защитные меры .....	48
Другие ресурсы .....	48
Резюме .....	48
<b>Грех 3. Переполнение целых чисел .....</b>	<b>49</b>
В чем состоит грех .....	49
Подверженные греху языки .....	49
Как происходит грехопадение .....	49
Греховность C и C++ .....	50
Поразрядные операции .....	55
Греховность C# .....	55
Греховность Visual Basic и Visual Basic .NET .....	56
Греховность Java .....	57
Греховность Perl .....	58
Где искать ошибку .....	59
Выявление ошибки на этапе анализа кода .....	59
C/C++ .....	59
C# .....	61
Java .....	62
Visual Basic и Visual Basic .NET .....	62

Perl .....	62
Тестирование .....	62
Примеры из реальной жизни .....	62
Ошибка в интерпретаторе Windows Script позволяет выполнить произвольный код .....	63
Переполнение целого в конструкторе объекта SOAPParameter .....	63
Переполнение кучи в HTR-документе, передаваемом поблочно, может скомпрометировать Web-сервер .....	63
Искупление греха .....	64
Дополнительные защитные меры .....	66
Другие ресурсы .....	66
Резюме .....	66
Не рекомендуется .....	66
<b>Грех 4. Внедрение SQL-команд</b> .....	67
В чем состоит грех .....	67
Подверженные греху языки .....	67
Как происходит грехопадение .....	68
Греховность C# .....	68
Греховность PHP .....	69
Греховность Perl/CGI .....	69
Греховность Java .....	70
Греховность SQL .....	71
Родственные грехи .....	72
Где искать ошибку .....	72
Выявление ошибки на этапе анализа кода .....	72
Тестирование .....	73
Примеры из реальной жизни .....	75
CAN-2004-0348 .....	75
CAN-2002-0554 .....	75
Искупление греха .....	75
Проверяйте все входные данные .....	76
Никогда не применяйте конкатенацию для построения SQL-предложений .....	76
Дополнительные защитные меры .....	79
Другие ресурсы .....	79
Резюме .....	80
<b>Грех 5. Внедрение команд</b> .....	82
В чем состоит грех .....	82

Подверженные греху языки .....	82
Как происходит грехопадение .....	82
Родственные грехи .....	84
Где искать ошибку .....	84
Выявление ошибки на этапе анализа кода .....	84
Тестирование .....	86
Примеры из реальной жизни .....	86
CAN-2001-1187 .....	86
CAN-2002-0652 .....	87
Искупление греха .....	87
Контроль данных .....	87
Если проверка не проходит .....	90
Дополнительные защитные меры .....	90
Другие ресурсы .....	91
Резюме .....	91
<b>Грех 6. Пренебрежение обработкой ошибок .....</b>	<b>92</b>
В чем состоит грех .....	92
Подверженные греху языки .....	92
Как происходит грехопадение .....	92
Раскрытие излишней информации .....	92
Игнорирование ошибок .....	93
Неправильная интерпретация ошибок .....	93
Бесполезные возвращаемые значения .....	94
Обработка не тех исключений, что нужно .....	94
Обработка всех исключений .....	94
Греховность C/C++ .....	94
Греховность C/C++ в Windows .....	95
Греховность C++ .....	96
Греховность C#, VB.NET и Java .....	96
Родственные грехи .....	97
Где искать ошибку .....	97
Выявление ошибки на этапе анализа кода .....	97
Тестирование .....	97
Примеры из реальной жизни .....	98
CAN-2004-0077 do_mremar в ядре Linux .....	98
Искупление греха .....	98
Искупление греха в C/C++ .....	98
Искупление греха в C#, VB.NET и Java .....	99
Другие ресурсы .....	99
Резюме .....	100

<b>Грех 7. Кросс-сайтовые сценарии</b> .....	101
В чем состоит грех .....	101
Подверженные греху языки .....	101
Как происходит грехопадение .....	101
Греховное ISAPI-расширение или фильтр на C/C++ .....	102
Греховность ASP .....	103
Греховность форм ASP.NET .....	103
Греховность JSP .....	103
Греховность PHP .....	103
Греховность Perl-модуля CGI.pm .....	103
Греховность mod-perl .....	104
Где искать ошибку .....	104
Выявление ошибки на этапе анализа кода .....	104
Тестирование .....	105
Примеры из реальной жизни .....	106
Уязвимость IBM Lotus Domino для атаки с кросс-сайтовым сценарием и внедрением HTML .....	106
Ошибка при контроле входных данных в сценарии isqlplus, входящем в состав Oracle HTTP Server, позволяет удаленному пользователю провести атаку с кросс-сайтовым сценарием .....	106
CVE-2002-0840 .....	107
Искупление греха .....	107
Искупление греха в ISAPI-расширениях и фильтрах на C/C++ .....	107
Искупление греха в ASP .....	108
Искупление греха в ASP.NET .....	108
Искупление греха в JSP .....	108
Искупление греха в PHP .....	110
Искупление греха в Perl/CGI .....	110
Искупление греха в mod-perl .....	111
Замечание по поводу HTML-кодирования .....	111
Дополнительные защитные меры .....	112
Другие ресурсы .....	112
Резюме .....	113
<b>Грех 8. Пренебрежение защитой сетевого трафика</b> .....	114
В чем состоит грех .....	114
Подверженные греху языки .....	114
Как происходит грехопадение .....	115
Родственные грехи .....	117

Где искать ошибку .....	117
Выявление ошибки на этапе анализа кода .....	118
Тестирование .....	121
Примеры из реальной жизни .....	121
TCP/IP .....	121
Протоколы электронной почты .....	122
Протокол E*Trade .....	122
Искупление греха .....	122
Рекомендации низкого уровня .....	123
Дополнительные защитные меры .....	126
Другие ресурсы .....	126
Резюме .....	126
<b>Грех 9. Применение загадочных URL и скрытых полей форм .....</b>	<b>128</b>
В чем состоит грех .....	128
Подверженные греху языки .....	128
Как происходит грехопадение .....	128
Загадочные URL .....	128
Скрытые поля формы .....	129
Родственные грехи .....	129
Где искать ошибку .....	130
Выявление ошибки на этапе анализа кода .....	130
Тестирование .....	131
Примеры из реальной жизни .....	131
CAN-2000-1001 .....	132
Модификация скрытого поля формы в программе MaxWebPortal .....	132
Искупление греха .....	132
Противник просматривает данные .....	132
Противник воспроизводит данные .....	133
Противник предсказывает данные .....	135
Противник изменяет данные .....	136
Дополнительные защитные меры .....	137
Другие ресурсы .....	137
Резюме .....	137
<b>Грех 10. Неправильное применение SSL и TLS .....</b>	<b>138</b>
В чем состоит грех .....	138
Подверженные греху языки .....	138
Как происходит грехопадение .....	139

Родственные грехи .....	142
Где искать ошибку .....	142
Выявление ошибки на этапе анализа кода .....	143
Тестирование .....	144
Примеры из реальной жизни .....	145
Почтовые клиенты .....	145
Web-браузер Safari .....	146
SSL-прокси Stunnel .....	146
Искупление греха .....	147
Выбор версии протокола .....	147
Выбор семейства шифров .....	148
Проверка сертификата .....	149
Проверка имени хоста .....	150
Проверка отзыва сертификата .....	151
Дополнительные защитные меры .....	153
Другие ресурсы .....	153
Резюме .....	154

## **Грех 11. Использование слабых систем**

<b>на основе паролей .....</b>	<b>155</b>
В чем состоит грех .....	155
Подверженные греху языки .....	155
Как происходит грехопадение .....	155
Родственные грехи .....	158
Где искать ошибку .....	158
Выявление ошибки на этапе анализа кода .....	158
Политика управления сложностью пароля .....	158
Смена и переустановка пароля .....	159
Протоколы проверки паролей .....	159
Ввод и хранение паролей .....	160
Тестирование .....	160
Примеры из реальной жизни .....	161
CVE-2005-1505 .....	161
CVE-2005-0432 .....	162
Ошибка в TENEX .....	162
Кража у Пэрис Хилтон .....	163
Искупление греха .....	163
Многофакторная аутентификация .....	163
Хранение и проверка паролей .....	164
Рекомендации по выбору протокола .....	167
Рекомендации по переустановке паролей .....	168



Рекомендации по выбору пароля .....	169
Прочие рекомендации .....	170
Дополнительные защитные меры .....	170
Другие ресурсы .....	170
Резюме .....	170
Не рекомендуется .....	171
Стоит подумать .....	171
<b>Грех 12. Пренебрежение безопасным хранением и защитой данных .....</b>	<b>172</b>
В чем состоит грех .....	172
Подверженные греху языки .....	172
Как происходит грехопадение .....	172
Слабый контроль доступа к секретным данным .....	172
Греховность элементов управления доступом .....	174
Встраивание секретных данных в код .....	176
Родственные грехи .....	177
Где искать ошибку .....	177
Выявление ошибки на этапе анализа кода .....	178
Тестирование .....	178
Примеры из реальной жизни .....	181
CVE-2000-0100 .....	181
CAN-2002-1590 .....	181
CVE-1999-0886 .....	181
CAN-2004-0311 .....	182
CAN-2004-0391 .....	182
Искупление греха .....	182
Использование технологий защиты, предоставляемых операционной системой .....	183
Искупление греха в C/C++ для Windows 2000 и последующих версий .....	183
Искупление греха в ASP.NET версии 1.1 и старше .....	185
Искупление греха в C# на платформе .NET Framework 2.0 ...	185
Искупление греха в C/C++ для Mac OS X версии v10.2 и старше .....	186
Искупление греха без помощи операционной системы (или «храните секреты от греха подальше») .....	186
Замечание по поводу Java и Java KeyStore .....	188
Дополнительные защитные меры .....	189
Другие ресурсы .....	190
Резюме .....	191

<b>Грех 13. Утечка информации</b> .....	192
В чем состоит грех .....	192
Подверженные греху языки .....	192
Как происходит грехопадение .....	193
Побочные каналы .....	193
Слишком много информации! .....	194
Модель безопасности информационного потока .....	196
Греховность C# (и других языков) .....	198
Родственные грехи .....	198
Где искать ошибку .....	199
Выявление ошибки на этапе анализа кода .....	199
Тестирование .....	200
Имитация кражи ноутбука .....	200
Примеры из реальной жизни .....	200
Атака с хронометражем Дэна Бернштейна на шифр AES .....	201
CAN-2005-1411 .....	201
CAN-2005-1133 .....	201
Искупление греха .....	202
Искупление греха в C# (и других языках) .....	203
Учет локальности .....	203
Дополнительные защитные меры .....	203
Другие ресурсы .....	204
Резюме .....	204
<b>Грех 14. Некорректный доступ к файлам</b> .....	206
В чем состоит грех .....	206
Подверженные греху языки .....	206
Как происходит грехопадение .....	207
Греховность C/C++ в Windows .....	207
Греховность C/C++ .....	208
Греховность Perl .....	208
Греховность Python .....	208
Родственные грехи .....	209
Где искать ошибку .....	209
Выявление ошибки на этапе анализа кода .....	209
Тестирование .....	210
Примеры из реальной жизни .....	210
CAN-2005-0004 .....	210
CAN-2005-0799 .....	211
CAN-2004-0452 и CAN-2004-0448 .....	211
CVE-2004-0115 Microsoft Virtual PC для Macintosh .....	211

Искупление греха .....	211
Искупление греха в Perl .....	212
Искупление греха в C/C++ для Unix .....	212
Искупление греха в C/C++ для Windows .....	213
Получение места нахождения временного каталога пользователя .....	213
Искупление греха в .NET .....	213
Дополнительные защитные меры .....	214
Другие ресурсы .....	214
Резюме .....	214

<b>Грех 15. Излишнее доверие к системе разрешения сетевых имен .....</b>	<b>215</b>
В чем состоит грех .....	215
Подверженные греху языки .....	215
Как происходит грехопадение .....	215
Греховные приложения .....	218
Родственные грехи .....	218
Где искать ошибку .....	219
Выявление ошибки на этапе анализа кода .....	219
Тестирование .....	220
Примеры из реальной жизни .....	220
CVE-2002-0676 .....	220
CVE-1999-0024 .....	221
Искупление греха .....	221
Другие ресурсы .....	222
Резюме .....	223

<b>Грех 16. Гонки .....</b>	<b>224</b>
В чем состоит грех .....	224
Подверженные греху языки .....	224
Как происходит грехопадение .....	224
Греховность кода .....	226
Родственные грехи .....	227
Где искать ошибку .....	227
Выявление ошибки на этапе анализа кода .....	228
Тестирование .....	229
Примеры из реальной жизни .....	229
CVE-2001-1349 .....	229
CAN-2003-1073 .....	230
CVE-2004-0849 .....	230

Искупление греха .....	230
Дополнительные защитные меры .....	232
Другие ресурсы .....	232
Резюме .....	233
<b>Грех 17. Неаутентифицированный обмен ключами .....</b>	<b>234</b>
В чем состоит грех .....	234
Подверженные греху языки .....	234
Как происходит грехопадение .....	234
Родственные грехи .....	236
Где искать ошибку .....	236
Выявление ошибки на этапе анализа кода .....	236
Тестирование .....	237
Примеры из реальной жизни .....	237
Атака с «человеком посередине» на Novell Netware .....	237
CAN-2004-0155 .....	238
Искупление греха .....	238
Дополнительные защитные меры .....	239
Другие ресурсы .....	239
Резюме .....	239
<b>Грех 18. Случайные числа криптографического качества .....</b>	<b>240</b>
В чем состоит грех .....	240
Подверженные греху языки .....	240
Как происходит грехопадение .....	240
Греховность некриптографических генераторов .....	241
Греховность криптографических генераторов .....	242
Греховность генераторов истинно случайных чисел .....	242
Родственные грехи .....	243
Где искать ошибку .....	243
Выявление ошибки на этапе анализа кода .....	244
Когда следует использовать случайные числа .....	244
Выявление мест, где применяются PRNG-генераторы .....	244
Правильно ли затравлен CRNG-генератор .....	245
Тестирование .....	245
Примеры из реальной жизни .....	246
Браузер Netscape .....	246
Проблемы в OpenSSL .....	246
Искупление греха .....	247

Windows .....	247
Код для .NET .....	248
Unix .....	248
Java .....	249
Повторное воспроизведение потока случайных чисел .....	250
Дополнительные защитные меры .....	250
Другие ресурсы .....	250
Резюме .....	251
Стоит подумать .....	251
<b>Грех 19. Неудобный интерфейс .....</b>	<b>252</b>
В чем состоит грех .....	252
Подверженные греху языки .....	252
Как происходит грехопадение .....	252
Каков круг ваших пользователей? .....	253
Минное поле: показ пользователям информации о безопасности .....	254
Родственные грехи .....	254
Где искать ошибку .....	255
Выявление ошибки на этапе анализа кода .....	255
Тестирование .....	255
Примеры из реальной жизни .....	256
Аутентификация сертификата в протоколе SSL/TLS .....	256
Установка корневого сертификата в Internet Explorer 4.0 .....	257
Искупление греха .....	257
Делайте интерфейс пользователя простым и понятным .....	258
Принимайте за пользователей решения, касающиеся безопасности .....	258
Упрощайте избирательное ослабление политики безопасности .....	259
Ясно описывайте последствия .....	260
Помогайте пользователю предпринять действия .....	262
Предусматривайте централизованное управление .....	263
Другие ресурсы .....	263
Резюме .....	264
<b>Приложение А. Соответствие между 19 смертными грехами и «10 ошибками» OWASP .....</b>	<b>265</b>
<b>Приложение В. Сводка рекомендаций .....</b>	<b>266</b>
<b>Предметный указатель .....</b>	<b>276</b>



## 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 Дженифер.

Джон Виера 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](http://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 киберпространства (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 совершенствованием 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 двигать 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 строки 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](http://www.cve.mitre.org)), с сайта BugTraq ([www.securityfocus.com](http://www.securityfocus.com)) или базы данных уязвимостей в программах с открытыми исходными текстами (Open Source Vulnerability Database) ([www.osvdb.org](http://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, нерекондуемых и возможных действий при написании нового или анализе существующего кода. Не следует недооценивать важность этого раздела! Содержание всех Резюме сведено воедино в Приложении В.

Конец ознакомительного фрагмента.  
Приобрести книгу можно  
в интернет-магазине  
«Электронный универс»  
[e-Univers.ru](http://e-Univers.ru)