A Look at Open Source Security

By Ivan Ristic

This talk is about *open source*, *software security* and the *challenges of developing secure open source software;* from experience and a very personal perspective.

I will also discuss how to assess the security of an open source project with only a modest time investment.

What do I mean by Open Source?

Open Source means different things to different people.

Today I will assume “open source” refers to products whose source code is freely available or products that are written by a community. Often both.

Commercial versus open source security comparisons are meaningless. Generalisation does not work.

At best, you can determine which project was more secure in a period of time. Maybe.

It is true, however,
that open source projects have a
potential to be more secure.

Source code access gives you the ability to: 1) assess security, 2) fix problems, and 3) compile programs yourself.

It also keeps honest developers honest by making it easier for others to find flaws.

Easier for others to find flaws? Doesn’t that help the bad guys too?

Kerckhoff’s principle: A secure military system must not depend on secrecy.

Open Source Security Myths

Myth: It is easy to
plant a backdoor
into an open source project.

In November 2003, an attempt to create a backdoor in Linux was discovered.

Myth: Given enough eyeballs all bugs are shallow.

In January 2009, an OpenSSL signature verification API misuse problem was reported; it was there for more than 10 years.

A line of code removed from
Debian made it vulnerable from
September 2006 until May 2008.

Myth: Open source developers care about security (*and vendors don't*).

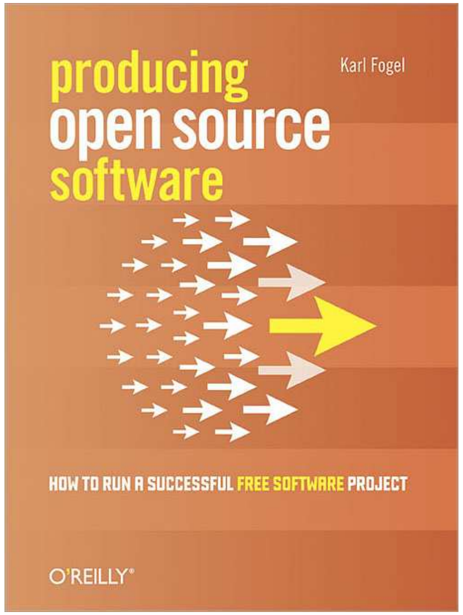
Open Source Development Challenges

Open source development is about one or more of these: freedom, passion, money, fame and software commons.

(Not necessarily in that order.)

Starting and running an open source project is a job. In fact, it is like starting a business, *but without the money*.

Most open source projects start as one-man efforts.


Karl Fogel
producing open source software
HOW TO RUN A SUCCESSFUL FREE SOFTWARE PROJECT
O'REILLY

Producing Open Source Software: 300 pages of hard work.

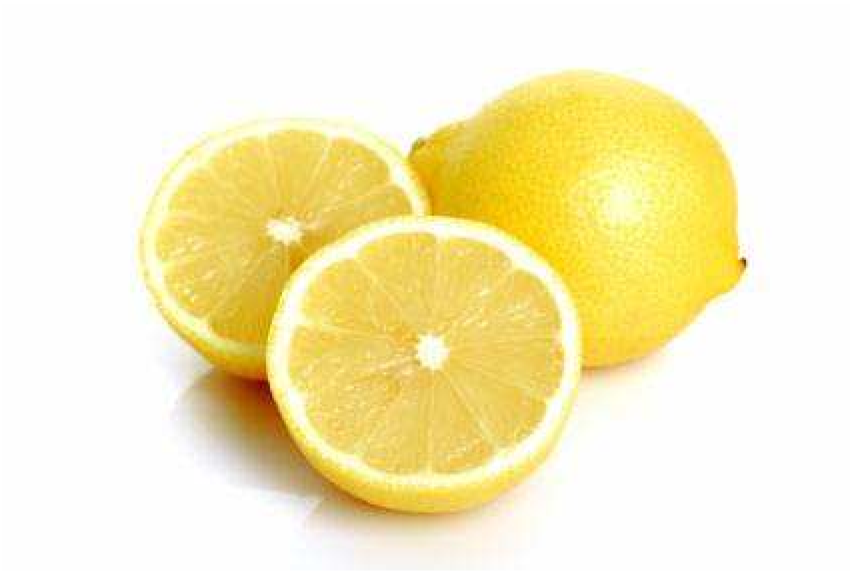
(http://producingoss.com)

Why is software so insecure?

The security of a product
largely depends on the people
who build it and
the people who use it.

To build secure software
you need awareness,
motivation, expertise
and resources.

You also need secure technology, but, sadly, we don't always have a choice.

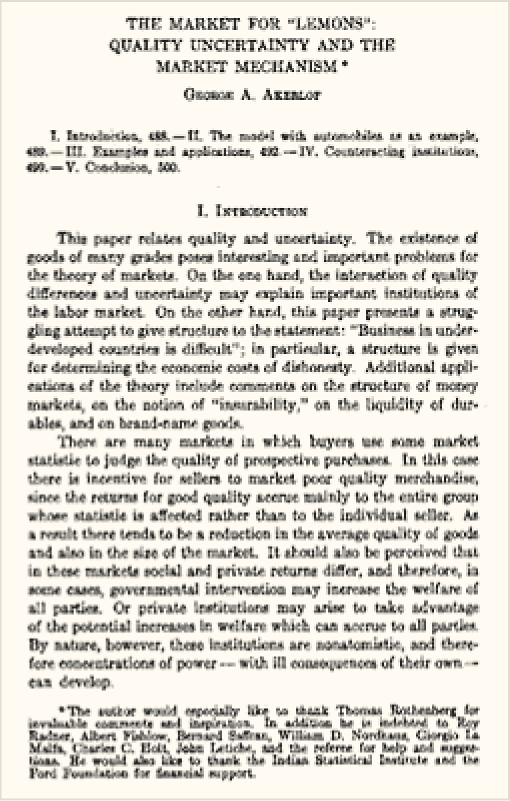


Software is a

market for lemons.

George A. Akerlof

The Market for "Lemons": Quality Uncertainty and the Market Mechanism



THE MARKET FOR "LEMONS":
QUALITY UNCERTAINTY AND THE
MARKET MECHANISM *

GEORGE A. AKERLOF

I. Introduction, 488. — II. The model with automobiles as an example, 489. — III. Examples and applications, 492. — IV. Counteracting institutions, 499. — V. Conclusion, 500.

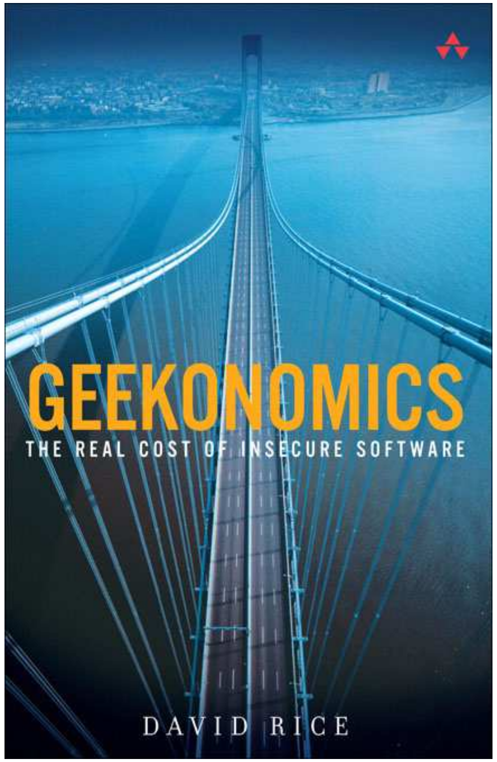
I. INTRODUCTION

This paper relates quality and uncertainty. The existence of goods of many grades poses interesting and important problems for the theory of markets. On the one hand, the interaction of quality differences and uncertainty may explain important institutions of the labor market. On the other hand, this paper presents a struggling attempt to give structure to the statement: "Business in underdeveloped countries is difficult"; in particular, a structure is given for determining the economic costs of dishonesty. Additional applications of the theory include comments on the structure of money markets, on the notion of "insurability," on the liquidity of durables, and on brand-name goods.

There are many markets in which buyers use some market statistic to judge the quality of prospective purchases. In this case there is incentive for sellers to market poor quality merchandise, since the returns for good quality accrue mainly to the entire group whose statistic is affected rather than to the individual seller. As a result there tends to be a reduction in the average quality of goods and also in the size of the market. It should also be perceived that in these markets social and private returns differ, and therefore, in some cases, governmental intervention may increase the welfare of all parties. Or private institutions may arise to take advantage of the potential increases in welfare which can accrue to all parties. By nature, however, these institutions are nonatomistic, and therefore concentrations of power — with ill consequences of their own — can develop.

* The author would especially like to thank Thomas Rothenberg for invaluable comments and inspiration. In addition he is indebted to Roy Radner, Albert Fishlow, Bernard Saffran, William D. Nordhaus, Giorgio La Malfa, Charles C. Holt, John Letiche, and the referee for help and suggestions. He would also like to thank the Indian Statistical Institute and the Ford Foundation for financial support.

“[…] the presence of people who wish to pawn bad wares as good wares tends to drive out the legitimate business”.



Geekonomics: The Real Cost of Insecure Software

(interesting book; terrible name)

The result? Not only is software not secure, but the adoption of (more) secure programming languages and platforms is slow.

Common issues with software are in the areas of: usability, safety, security and appearance.

When a security bug in djbdns was discovered, D.J. Bernstein paid the researcher $1000 and apologised to his users.

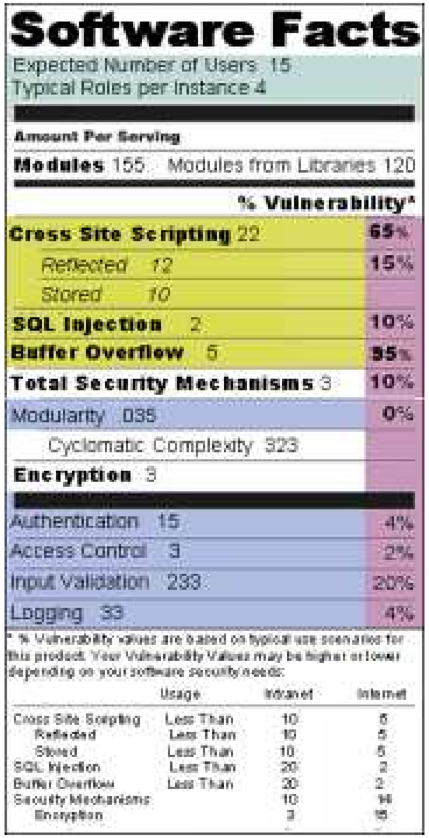
What if we make software publishers liable?

Self-certification,

on the other hand,

seems quite feasible.

(The Software Facts label taken from Jeff Williams's talk at AppSec Europe 2005.)



Software Facts

Expected Number of Users 15
Typical Roles per Instance 4

Amount Per Serving

Modules 155 Modules from Libraries 120

	% Vulnerability*
Cross Site Scripting 22	**65%**
Reflected 12	**15%**
Stored 10	
SQL Injection 2	**10%**
Buffer Overflow 5	**95%**
Total Security Mechanisms 3	**10%**
Modularity .035	**0%**
Cyclomatic Complexity 323	
Encryption 3	
Authentication 15	4%
Access Control 3	2%
Input Validation 233	20%
Logging 33	4%

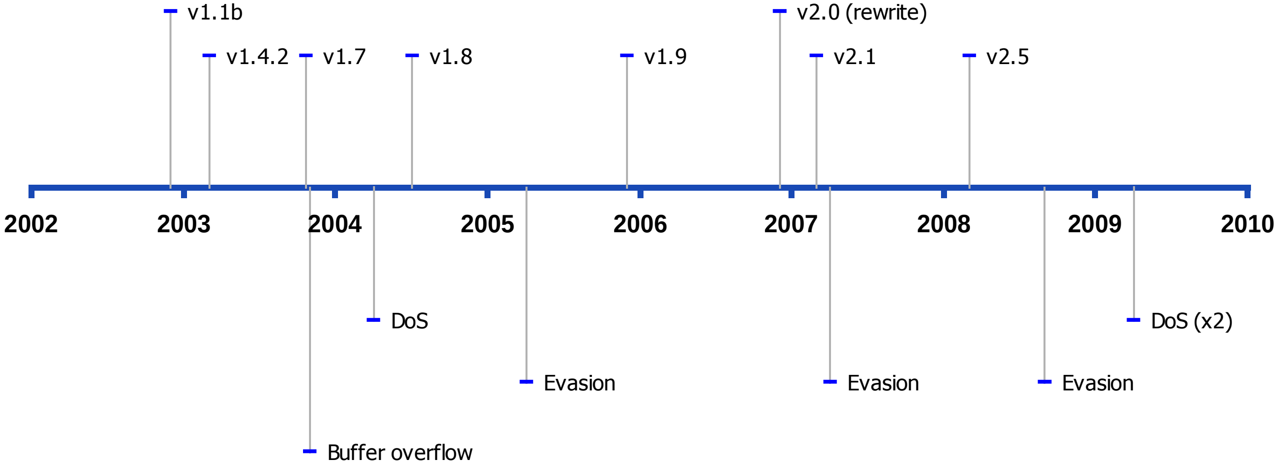
* % Vulnerability values are based on typical use scenarios for this product. Your Vulnerability Values may be higher or lower depending on your software security needs:

	Usage	Intranet	Internet
Cross Site Scripting	Less Than	10	5
Reflected	Less Than	10	5
Stored	Less Than	10	5
SQL Injection	Less Than	20	2
Buffer Overflow	Less Than	20	2
Security Mechanisms		10	14
Encryption		3	15

Case study: ModSecurity

http://www.modsecurity.org

Security issues in ModSecurity



Assessing Open Source Project Security

Project Status

1

1) Project age, release frequency and status
2) Web site & publicly available information
3) Popularity (community) - blog posts, articles, talks, books, tools, add-ons, mailing list activity
4) Developer attitude / tone
5) Commercial activities

Development Practices

2

1) Size of team; skill and experience
2) Secure development methodology
3) Source code repository
4) Issue tracking
5) Regression testing
6) Source code quality
7) Quality of documentation

Treatment of Security

3

1) Clear security statement

2) Security page lists all known issues

3) Security email address

4) History of security problems

5) Response times

6) Own advisories

Thank you!

Download this talk from
http://www.ivanristic.com