

Transparente Software - eine Voraussetzung für datenschutzfreundliche Technologien

1 Allgemeines

1.1 Datenschutzfreundliche Technologien

Die Datenschutzbeauftragten des Bundes und der Länder setzen sich seit geraumer Zeit für die Nutzung datenschutzfreundlicher Technologien ein. In ihrer Entschließung vom 25./26. März 1999 fordern sie Hersteller von Informations- und Kommunikationstechnik auf, Software so zu entwickeln und herzustellen, dass Anwender und unabhängige Dritte sich jederzeit von der Wirksamkeit von Sicherheitsvorkehrungen überzeugen können.

Zur Prüfung und Revision ist es unter anderem notwendig, Software so zu entwickeln und zu publizieren, dass unabhängige Fachleute die Funktionsweise lückenlos nachvollziehen und fehlerhaft arbeitende Teilkomponenten finden können.

Derartige Transparenz von Software kann gewährleistet werden, wenn Quelltexte von Programmen nicht geheim gehalten werden, sondern für Prüf- und Revisionszwecke zugänglich sind. Solche Kontrollmechanismen sind unerlässlich, weil komplexe Software praktisch nie fehlerfrei ist und die Qualitätskontrollen der Hersteller in der Regel nicht ausreichen.

1.2 Transparente Software

Ein vielversprechender Ansatz für Transparenz als eine Form datenschutzfreundlicher Technologien stellt das Entwicklungsmodell „Open Source“ dar. Der Begriff „Open Source“ steht für offenen Quelltext von Programmen. Die uneingeschränkte Veröffentlichung der Quelltexte ist ein entscheidendes Merkmal dieser Software. Insbesondere Fachleuten nützt diese Form der Transparenz, weil sie dadurch die oben genannten Prüf- und Revisionsaufgaben wahrnehmen zu können (siehe dazu Abschnitt 2).

Ein ergänzender Ansatz, ohne den auch das Open-Source-Modell kein berechtigtes Vertrauen erzeugen könnte, stellt die Evaluierung und Zertifizierung von Software durch unabhängige Fachleute dar. Dabei müssen die Quellen nicht allgemein offen gelegt, aber einem eingeschränkten Kreis von Spezialisten preisgegeben werden. Auch dieses Verfahren garantiert den Anwendern ein hohes Maß an Vertrauenswürdigkeit, und ermöglicht im Sinne einer transparenten Verarbeitung (personenbezogener Daten) eine hinreichend verlässliche Einschätzung datenschutzrelevanter Programmfunktionen (siehe dazu Abschnitt 3).

2 Entwicklungsmodell „Open Source“

2.1 „Open Source“

Unter Open-Source-Software (OSS) versteht man Software, deren Quelltext (source code) offengelegt und für jeden frei verfügbar ist. Dadurch kann jedermann den Quelltext lesen, mit ihm arbeiten, ihn verändern und solche Änderungen uneingeschränkt publizieren. Jeder kann also prinzipiell prüfen, ob ein Programm tatsächlich die angegebenen Funktionen realisiert und darüber hinaus keine Programmteile enthält, die unerwartete und meist unerwünschte Funktionen ausführen (z. B. sog. Trojanische Pferde) oder die spätere Eindringmöglichkeiten in das System eröffnen (sog. Hintertüren). Offengelegte Software erleichtert außerdem eine schnelle, bedarfsgerechte Anpassung oder Weiterentwicklung wesentlich.

Das Lesen des Quelltextes reicht allerdings in der Regel nicht aus, um die Funktionsweise von Programmen vollständig zu überblicken. Hierfür sind auch Kommentare und Programmdokumentationen erforderlich. Die nachfolgend beschriebenen Vorteile des Open-Source-Modells können deshalb nur dann zum Tragen kommen, wenn verständliche und aussagekräftige Dokumentationen vorhanden sind und gemeinsam mit den Quellen veröffentlicht werden.

Bei den meisten kommerziell vertriebenen Softwareprodukten werden sowohl der Quellcode als auch detaillierte Dokumentationen geheimgehalten, so dass der oben beschriebene Umgang mit der Software nicht erlaubt bzw. unmöglich ist.

2.2 Bedeutung des Quellcodes

Computerprogramme sind Folgen von Anweisungen, die zunächst in einer bestimmten Programmiersprache als Quelltext vorliegen. Die besondere Bedeutung des Quellcodes resultiert aus der Tatsache, dass die Funktionsweise von Software selbst für Fachleute nur durch Lesen der Quellen nachvollziehbar ist. Mindestvoraussetzung für eine erfolgreiche und tiefgehende Prüfung und Revision von Software ist also immer der Zugang zum Quellcode. Wie unter 2.1 bereits dargestellt, wird es darüber hinaus in der Regel unerlässlich sein, vollständigen Einblick in die Programmdokumentation zu erhalten.

Damit die Anweisungen von Rechnern ausgeführt werden können, wird der Quellcode durch spezielle Programme (Compiler, Interpreter) in den zur Ausführung erforderlichen Maschinencode übersetzt. Die Rückübersetzung ist zwar möglich (mit Hilfe von Decompilern), aber die dabei entstehenden Programme sind nur schwer lesbar, da die Compiler die Programme maschinenabhängig optimieren und dabei oft stark verändern. Außerdem werden dokumentierende Informationen, wie zum Beispiel Variablennamen und Anmerkungen, die die Lesbarkeit des Quelltextes erleichtern, nicht in den Maschinencode übernommen. Vor der Rückübersetzung ist ohnehin zunächst zu prüfen, ob die Rückübersetzung lizenzrechtlich gestattet ist.

Sowohl beim Lesen und Prüfen als auch bei der Änderung und Anpassung von Programmen kommen die Vorteile des Open-Source-Konzeptes zum Tragen. Die drei typischen Phasen der Softwareentwicklung (Entwurf und Realisierung, Prüfung der ordnungsgemäßen Funktion, Änderungen) müssen bei der Entwicklung und Anpassung jeder Software normalerweise mehrmals durchlaufen werden. Liegt der Quelltext mit all seinen dokumentierenden Elementen (z. B. Namen, Anmerkungen) offen vor, kann die Dauer der Zyklen unter Umständen verkürzt werden. Denn in Open-Source-Projekten wird nicht nur ein kleiner Kreis firmengebun-

dener Mitarbeiter tätig, sondern viele unabhängige Entwickler haben weltweit die Möglichkeit, beispielsweise eine im Internet publizierte Software zu bearbeiten.

2.3 Kriterien und Rahmenbedingungen für Open-Source-Software

Die Open-Source-Definition, die aus den Richtlinien für die Software Debian GNU/Linux hervorgegangen ist, definiert die folgenden Kriterien für Open-Source-Software (OSD, siehe <http://www.opensource.org>):

1. Freie Weiterverbreitung – keine Einschränkungen, keine Lizenzgebühren für die Verbreitung;
2. Quellcode – Quelltext und Kompilat müssen verbreitet werden dürfen, der Quelltext muss verfügbar sein und grundlegenden Qualitätsanforderungen genügen;
3. Auf dem Programm basierende Werke – Veränderung und Ableitung von Werken und Verbreitung der so entstehenden Werke muss unter den selben Bedingungen erlaubt sein;
4. Unversehrtheit des Originalcodes – es kann verlangt werden, dass die Verbreitung modifizierten Quelltextes unter Umständen in Form sogenannter Patch-Dateien stattfinden muss und dass abgeleitete Werke andere Namen oder Versionsnummern tragen müssen;
5. Keine Diskriminierung von einzelnen Personen oder Gruppen;
6. Keine Einschränkungen für bestimmte Anwendungsbereiche;
7. Verbreitung der Lizenz – die zum Programm gehörigen Rechte gelten für jeden, der das Programm erhalten hat, ohne dass eine weitere Lizenz beachtet werden muss;
8. Die Lizenz darf nicht für ein bestimmtes Produkt gelten – die zum Programm gehörigen Rechte dürfen nicht davon abhängen, dass das Programm Teil einer bestimmten Distribution ist;
9. Die Lizenz darf andere Software nicht beeinträchtigen – uneingeschränkte Möglichkeit der Verbreitung anderer Software zusammen mit der lizenzierten Software.

Beispiel für eine Lizenzvereinbarung, die dieser Definition genügt, ist die GNU General Public Licence (GPL, siehe <http://www.gnu.org>).

Die oben erwähnten Kriterien und die bei der Open-Source-Software-Entwicklung vorhandenen Strukturen haben aber auch die nachfolgend genannten Rahmenbedingungen zur Folge, die zu Einschränkungen beim Einsatz derartiger Software führen können:

1. Entwickler bieten in der Regel keine Gewährleistung oder Haftung;
2. kein automatischer Support;
3. die Verifizierung von OSS durch unabhängige Fachleute ist nicht automatisch sichergestellt;
4. eine zeitnahe Fehlerbehebung wird nicht garantiert;
5. auch die Beteiligung vieler unabhängiger Programmierer garantiert keine fehlerfreie Weiterentwicklung bzw. Freiheit von unerwünschten Seiteneffekten;
6. die Gesamtfunktionalität bei Detailänderungen oder Weiterentwicklungen wird nicht immer geprüft;
7. Qualitätssicherungsmaßnahmen sind mitunter stark eingeschränkt;
8. die Übereinstimmung der Quellen mit der Binärversion wird nicht garantiert;
9. oftmals keine ausreichende Dokumentation;
10. noch keine Entscheidung für eine einheitliche Benutzeroberfläche (z. B. KDE, Gnome);
11. Sicherheitslöcher sind für potentielle Angreifer leichter auffindbar;

12. aktuelle Service-Releases, Korrekturen und Updates werden nicht automatisch implementiert;
13. kein Zwang zur Berücksichtigung nationaler Gesetzgebung bzw. Interessen;
14. nur stark eingeschränkte Sanktionsmöglichkeit bei Verletzung der OSD-Regeln;
15. die OSD-Regeln verlangen keine formale Zertifizierung;
17. die überwiegende Zahl der Anwender (auch der öffentlichen Stellen) verfügen über kein im Sinne von OSD qualifiziertes Personal.

2.4 Bedeutung von Open-Source-Software

Zu den erfolgreichsten Entwicklungen nach dem Open-Source-Ansatz gehören das Betriebssystem Linux und der Webserver Apache. Auch wenn diese Produkte zur Zeit nur eine relativ kleine Nische des gesamten Marktes besetzen, beweisen sie doch, dass die weltweite, freiwillige Kooperation nach dieser Entwicklungsmethode komplexe und stabile Programme hervorbringen kann. Mittlerweile befassen sich auch namhafte Hersteller mit diesen Produkten – möglicherweise auch deshalb, um einen beginnenden Trend nicht zu verpassen und keine finanziellen Einbußen zu erleiden.

3 Auswirkungen auf Datenschutz und Datensicherheit

3.1 Transparenz

Open-Source-Produkte sind transparent und erfüllen damit eine Basisforderung datenschutzfreundlicher Technologien. Anhand des Quelltextes ist eine Prüfung durch unabhängige Experten prinzipiell uneingeschränkt möglich. Dies erhöht die Revisionsfähigkeit der Software maßgeblich.

Allerdings ist nicht garantiert, dass die Prüfung auch tatsächlich stattfindet und dass der Quelltext verständlich ist. Generell gilt daher, dass auch Programme, die unter Open-Source-Bedingungen veröffentlicht werden, Fehler und Hintertüren enthalten können, von denen möglicherweise lange Zeit niemand Notiz nimmt. Vor diesem Hintergrund ist es besonders wichtig, dass formalisierte Verfahren für Prüfung und Evaluierung genutzt werden. In der Regel ist hierfür zusätzlich die Dokumentation oder sogar die Spezifikation der Software nötig. Allerdings verlangt die Open-Source-Definition nicht, dem Quellcode eine Dokumentation oder gar eine Spezifikation beizulegen. Beides ist aber zur Evaluation zwingend erforderlich. Die OSD verbietet lediglich absichtlich verwirrend geschriebenen Code.

Soll auch ausgeschlossen werden, dass manipulierte oder fehlerbehaftete Entwicklungswerkzeuge zum Einsatz kommen, reicht die Prüfung des Quelltextes der produzierten Software allein nicht aus. Um eine solche umfassende Sicherheitsaussage machen zu können, müssten zusätzlich alle Werkzeuge geprüft werden, die zur Erstellung des Maschinencodes verwendet werden. Da dazu neben den oben erwähnten Compilern auch das Betriebssystem und die Hardware sowie die gesamte Einsatzumgebung gehören, ist allerdings fraglich, ob dieser Aufwand tatsächlich leistbar ist. Voraussetzung hierfür wäre jedenfalls, dass auch dort ausschließlich Open Source Produkte eingesetzt werden.

Prinzipiell ist auch überprüfbar, ob ein System auf ausgereiften Sicherheitsmechanismen basiert, etwa auf anerkannt sicheren, kryptographischen Verfahren. Im Bereich der Kryptographie gilt die Offenlegung von Algorithmen bei vielen Experten schon seit langer Zeit als

wichtige Voraussetzung für deren Sicherheit. Allerdings lehnt das Bundesamt für Sicherheit in der Informationstechnik die Offenlegung von Kryptoalgorithmen insbesondere bei Anwendungen für den Geheimschutzbereich ab, weil befürchtet wird, dass potentiellen Gegnern gerade dadurch bessere Angriffsmöglichkeiten eröffnet werden.

Open-Source-Produkte haben den Vorteil, dass sich Anwender weitgehend selbst gegenüber Sicherheitslücken schützen können, sofern sie sich regelmäßig die erforderlichen Informationen – normalerweise aus dem Internet – beschaffen. Open-Source-Entwickler unterstützen diese Anwender, indem Fehler oft in öffentlich zugänglichen Datenbanken publiziert werden und in der Regel auch entsprechende Schutzmaßnahmen bekanntgegeben werden. Dies führt zwar unter Umständen dazu, dass auch potentielle Angreifer diese Informationen ausnutzen können. Die aktive Beteiligung vieler IT-Experten an der Fehlerbeseitigung kann jedoch dazu beitragen, dass derartige Sicherheitslücken wesentlich schneller geschlossen werden, als das bei herkömmlicher Software möglich ist.

Allerdings wird dem „normalen“ Anwender selbst der offengelegte Quellcode in der Regel wenig nützen. Für diesen Nutzer wäre es sicher auch im Sinne transparenter Software viel hilfreicher, wenn ihm eine verständliche Programmbeschreibung vorliegen würde und er erkennen könnte, wie Sicherheitsfunktionen aktiviert werden können.

3.2 Qualitätssicherung

Open Source ermöglicht, dass Fehler schnell behoben werden können, da die Verbreitung von Korrekturen (im Open-Source-Umfeld als Patch bezeichnet) problemlos funktioniert; es ist erlaubt und erwünscht, diesbezügliche Informationen auszutauschen. Um weitgehend auszuschließen zu können, dass diese Mechanismen zu neuen Fehlern und sogar zu trojanischen Pferden führen, müssen auch diese Patches vor dem Einsatz intensiv getestet werden. Als zusätzliche qualitätssichernde Maßnahme ist denkbar, unbefugte Änderungen der Produktionsumgebung und der Patches erkennbar zu machen, indem digitale Signaturen verwendet werden.

Wird zertifizierte Software gepatcht, verliert die Zertifizierung ihre Gültigkeit. Deshalb müssen nach jedem Patch zumindest Teile der Zertifizierung wiederholt werden. Dies ist jedoch langwierig und teuer. Aus diesem Grunde sollten nur solche Programmversionen zertifiziert werden, die sich als stabil erwiesen haben. Vorzugsweise sollten erst solche Versionen erneut zertifiziert werden, die nach einem Re-Design wieder fehlerfrei laufen. Die Häufigkeit der Versionswechsel muss dabei auf ein solches Tempo beschränkt werden, dass Zertifizierungsinstanzen noch Schritt halten können.

Auch durch die persönliche Motivation der Entwickler, die zum großen Teil selbst mit ihrem Produkt arbeiten, wird die Qualitätssicherung angetrieben; enge Terminsetzungen etwa durch das Marketing entfallen in der Regel. Unter dieser Konstellation hat in der Vergangenheit jedoch oft die Gestaltung der Benutzeroberfläche sowie die Bereitstellung von Installationsprozeduren und aussagekräftigen Dokumentationen gelitten, da für den Entwickler das Funktionieren des Programms im Vordergrund steht. Hier leisten jedoch Distributoren wichtige Arbeit. Von ihnen werden zunehmend softwareergonomische Benutzeroberflächen gefordert und die Integration verschiedener Programme einschließlich deren Installationsroutinen sowie Wartung und Support angeboten, so dass künftig auch von Open Source Produkten mehr Kundenorientiertheit zu erwarten ist.

Anwender mit entsprechenden Programmierkenntnissen können sich in bestimmten Grenzen selbst an der Qualitätssicherung und der Weiterentwicklung von Open-Source-Programmen beteiligen. Derartige Aktivitäten dürfen natürlich nicht dem Wildwuchs überlassen werden. Unterliegen Weiterentwicklung und Qualitätssicherung nicht einem kontrollierten Prozess, wäre das „Endprodukt“ kaum noch überschaubar und sicherheitstechnisch nicht mehr bewertbar. Gut organisierte Open-Source-Software verfügt deshalb über ein zentrales Repository (z. B. das Open-Source-Projekt Concurrent Versioning System – CVS), in dem Quellcode und Änderungen in einer Verwaltungshistorie geführt werden. Lesezugriff zu CVS-Repositories sind in der Regel leicht zu bekommen, während Schreibzugriffe erst nach dem Nachweis entsprechender Fachkompetenz erteilt werden. Mitunter werden auch nur auf bestimmte Teile von Quellen Schreibzugriffe erteilt. Obwohl beispielsweise die Kerne der verschiedenen LINUX-Distributionen als Quellen zum Lesen vollständig bereit stehen, werden Schreibzugriffe darauf nur sehr restriktiv erteilt.

3.3 *Wartung*

Als Nutzer von Open-Source-Programmen ist man einerseits nicht auf einen einzelnen Hersteller angewiesen, hat andererseits aber dadurch zunächst keinen Vertragspartner, der für die Eigenschaften des Produktes haftet und die Gewährleistung übernimmt. Denn in der Regel lehnt der Autor von vornherein jeden Anspruch ab. Die Gewährleistung durch den Autor verliert jedoch in dem Maß an Bedeutung, wie eine Reihe von Dienstleistungen wie Support und Schulungen durch Dritte angeboten werden. Wie bereits oben ausgeführt, finden sich in zunehmendem Maße Distributoren, die neben den oben genannten Dienstleistungen auch die Authentizität von Quelltext, Programmen und Patches sicherstellen helfen.

Unabhängig davon bleibt auch bei Open-Source-Produkten der Anwender für seine Datenverarbeitung verantwortlich und muss sich darum kümmern, dass die Sicherheit seines Systems erhalten bleibt. Wie bei proprietärer Software auch (Closed Source Software) erfordert dies von den Systemadministratoren die ständige Beschäftigung mit Sicherheitsfragen und eine regelmäßige Wartung der Systeme.

3.4 *Zertifizierung und Evaluierung*

Seit langer Zeit wird unter dem Aspekt des „Software-Engineering“ versucht, die Erstellung von Software so zu verbessern und zu verfeinern, dass Funktionalität und Qualität ebenso wie Vertrauenswürdigkeit und Sicherheit gewährleistet sind. Dazu gehören die Verwendung „höherer“ Programmiersprachen, eine strukturierte Programmierung, eine computerunterstützte Softwaregenerierung (CASE-Tools), die Entwicklung nach Lifecycle-Modellen und umfangreiche Dokumentationen und Tests.

Daneben wurden Standards und Prüfkriterien entwickelt, mit denen die Funktionalität, Qualität, Vertrauenswürdigkeit und Sicherheit vorgelegter Produkte gewährleistet werden kann. Hierzu gehören insbesondere die DIN/ISO 9000 als Norm für die Qualitätssicherung bei den Herstellern, die europäischen „Information Technology Security Evaluation Criteria“ (ITSEC) als europäischer Standard und schließlich die „Common Criteria“ (CC) als internationaler Standard für die Prüfung und Bewertung der Sicherheit der Informationstechnik, die die europäischen ITSEC, die US-Amerikanischen „Trusted Computer System Evaluation Criteria“ (TCSEC, Orange Book) und die kanadischen „Canadian Trusted Computer Produkt Evaluati-

on Criteria (CTCPEC) ersetzen sollten. Diese weltweit gültigen Kriterien sollen die gegenseitige Anerkennung von Evaluationsergebnissen (z. B. Zertifikaten) ermöglichen.

Die Sicherheit der Software soll in erster Linie durch Prüfung der Funktionalität und der Schadfunktionsfreiheit garantiert werden. Dabei wird eine Qualitätssicherung beim Hersteller vorausgesetzt. Für tiefergehende Prüfungen wird auch hier die Bereitstellung von Quellen verlangt. Die Funktionalität, Qualität, Vertrauenswürdigkeit und Sicherheit der Software wird in einem umfangreichen Evaluationsprozess festgestellt und in einer abschließenden Bewertung durch ein Zertifikat bestätigt.

Für die Anwender, insbesondere auch in der öffentlichen Verwaltung, ist die Bereitstellung evaluierter und zertifizierter Software ein geeigneter Weg, das Vertrauen in die Qualität und Sicherheit der Produkte zu stärken. Auch wenn die Evaluation und die Zertifizierung gerade bei umfangreichen Produkten zeitaufwendig und teuer sind, sind mittlerweile auch viele Hersteller daran interessiert, ihre Produkte auf diese Weise prüfen zu lassen, da Zertifikate nach ITSEC oder CC in zunehmendem Maße einen Wettbewerbsfaktor darstellen.

4 Fazit

Sowohl das Open-Source-Entwicklungsmodell als auch die Evaluierung und die Zertifizierung sind geeignete Verfahren, um die Transparenz von Software zu erhöhen. Beide Ansätze ermöglichen die Prüfung und Revision durch unabhängige Fachleute und sind somit geeignet, das Vertrauen der Anwender in die von ihnen genutzte Software zu stärken.

Die im Open-Source-Konzept beschriebene vollständige Offenlegung des Quelltextes von Programmen stellt für die IT-Sicherheit und den Datenschutz eine große Chance dar, insbesondere um verdeckte Programmfunktionen besser erkennen zu können. Das ist allerdings nur dann möglich, wenn der Quelltext auch tatsächlich von kompetenten Personen untersucht wird.

Der Einsatz von Open-Source-Software im Bereich der öffentlichen Verwaltung ist also dann sinnvoll, wenn

- die Quellen tatsächlich von vertrauenswürdigen Stellen überprüft (ggf. zertifiziert) werden, und daraus die lauffähige Binärversion erzeugt wird,
- durch digitale Signatur versiegelte Quellen und Binärversionen verteilt oder zum Abruf bereitgestellt werden und
- Betreuung, Fehlerbearbeitung und Weiterentwicklung in angemessener Weise sichergestellt und die Ergebnisse wie oben beschrieben vertrauenswürdig überprüft und bereitgestellt werden.

Für die überwiegende Zahl der Anwender, die sich mangels spezieller Programmierkenntnisse nicht aktiv an Open-Source-Aktivitäten beteiligen können und deren Aufgabenspektrum derartige Tätigkeiten ohnehin nicht vorsieht, ist besonders wichtig, verständliche Beschreibungen der von ihnen eingesetzten Software zu haben und beispielsweise zu wissen, welche Daten sie mit der Software verarbeitet, wohin diese Daten übertragen werden und wie Sicherheitsfunktionen aktiviert werden. Transparenz in diesem Zusammenhang bedeutet, dass die Nutzung der Informationstechnik für den Anwender transparent ist. Voraussetzung hierfür ist aller-

dings, dass er ein berechtigtes Vertrauen in die ordnungsgemäße Funktion der Programme hat.

Das kann nur erreicht werden, wenn die Software beispielsweise nach ITSEC oder Common Criteria evaluiert und zertifiziert wird. Für tiefgreifende Prüfungen ist es auch hier notwendig, dass der Quellcode und die vollständige Dokumentation den evaluierenden Stellen vorliegt.

5 Literatur

Bahr, R. E., Reiländer, R., Troles, E.: Open Source Software in der Bundesverwaltung. – KBSt-Brief Nr. 2/2000, <http://linux.kbst.bund.de/brief2-2000.html>

Feuerbach, H. T., Schmitz, P.: Freiheitskämpfer – Entwickler freier Software gegen Patentierung. – In: c't 16/1999.

Köhntopp, M.: Open Source. – In: Bäumlner, H., Breinlinger, A., Schrader, H.-H. (Hg.): Datenschutz von A-Z. – Neuwied, Kriftel: Luchterhand, 1999.

Köhntopp, K., Köhntopp, M., Pfitzmann, A.: Sicherheit durch Open Source? Chancen und Grenzen. – In: DuD 9/2000, [Ort]

Neumann, P. G.: Robust Nonproprietary Software. – IEEE Symposium on Security and Privacy, Oakland CA, May 15-17, 2000; <http://www.csl.sri.com/neumann/ieee00+.pdf>

Schneier, B.: Crypto-Gram September 15, 1999. – <http://www.counterpane.com/crypto-gram-9909.html>