DuD
Datenschutz und Datensicherheit

Risikoexposition bei Einsatz von Open-Source und proprietären Browsern

Ein Sicherheitsvergleich am Beispiel von Mozilla Firefox und MS Internet Explorer

Stefan Hunziker, Simon Rihs

Die Debatte um Unterschiede im Sicherheitsniveau von Open-Source- und proprietärer Software[1] wird nicht zuletzt aus nahe liegenden wirtschaftlichen Gründen immer wieder angefacht – und z.T. erbittert geführt. Der vorliegende Beitrag bemüht sich um eine unabhängige Gegenüberstellung – am Beispiel der beiden am weitesten verbreiteten Internet-Browser.

MScBA
Stefan Hunziker

Wissenschaftlicher Mitarbeiter am Institut für Wirtschaftsinformatik Bern

E-Mail: stefan.hunziker@iwi.unibe.ch

lic.rer.pol
Simon Rihs

Wissenschaftlicher Assistent am Institut für Wirtschaftsinformatik Bern

E-Mail: simon.rihs@iwi.unibe.ch

[1] Siehe auch Oppliger, DuD 11/2003, S. 669-675.

Einleitung

Durch die zunehmende Interaktion von Computern in lokalen Netzwerken sowie durch das Internet entstehen neue Angriffsflächen und Sicherheitslücken, welche wegen unangemessener Risikobeurteilung und unzureichenden Sicherheitsvorkehrungen zu erheblichen wirtschaftlichen Einbussen führen können.

Die zunehmende Zahl von Sicherheitslücken ist vielfach auf Fehler in der Entwicklung von Software zurückzuführen. Ein viel diskutiertes Thema in diesem Zusammenhang ist die Evaluierung der Software-Sicherheit von Open-Source- und proprietärer Software. Ein Vergleich anhand zweier Browser aus beiden Kategorien liefert dem Anwender wichtige Hinweise über die Sicherheit beim Einsatz der entsprechenden Software.

Die Auswertung der Daten über die Anzahl Sicherheitslücken in Mozilla Firefox und MS Internet Explorer und die dadurch bedingte Risiko-Exposition bis zur Installation eines entsprechenden Patches ergibt im Gesamturteil Sicherheitsvorteile für den Mozilla Firefox.

1 Open-Source- vs. proprietäre Software

Seit Beginn der sechziger Jahre ist der Begriff der Freien Software, im Englischen als Free(Soft)ware bezeichnet, im Software-Umfeld bekannt. Das Schlagwort Open-Source-Software (OSS) wurde erst 1998 eingeführt. Anlass dazu war die Gründung der Open-Source-Initiative (OSI), die auf ein Zusammentreffen von Protagonisten der Freien Software-Bewegung zurückzuführen ist. In der Literatur wird Open-Source häufig mit Freier Software gleichgesetzt, was nicht korrekt ist.[2] Freie Software ist zwar immer OSS, aber OSS muss nicht unbedingt Freie Software sein, da Veränderungen am Quellcode nicht unter den gleichen Lizenzbedingungen veröffentlicht werden müssen, unter denen die Original-Software gestanden hat. OSS ist daher eine weniger restriktive Lizenz als Freie Software.[3] Wesentliche Merkmale von OSS sind:

- Es ist keine Beschränkung der Distribution der Software erlaubt.
- Bei der Weitergabe dürfen keine Lizenzkosten erhoben werden.
- Der Quellcode muss mit dem Programm mitgeliefert werden.
- Veränderungen und Ableitungen am Quellcode sind zulässig, müssen aber ersichtlich gemacht werden.[4]

Im Gegensatz zu OSS stellt der Vertrieb von proprietärer Software die restriktivste Form der Software-Lizenzen dar. Proprietäre Software wird meist mit der Absicht der Gewinnerzielung entwickelt. Der Vertrieb erfolgt häufig nur im Binärcode, d.h. der Quellcode steht dem Nutzer nicht zur Verfügung. Die Lizenzbedingungen schließen das Kopieren, Verändern und Weiterverbreiten der Software weitgehend aus, gewähren dem Nutzer also anders als im Open-Source-Lizenzmodell keine speziellen Nutzungsbefugnisse.[5]

[2] Jaeger, T., Metzger, A., Open Source Software – Rechtliche Rahmenbedingungen der Freien Software, München: C.H. Beck 2002, S.1.

[3] http://www.kbst.bund.de/OSS-Kompetenzzentrum/-,318s0/OS-Glosssar.htm?link=kbs_zur_liststandard&link.orderby=titel&link.orderdir=DESC&link.stTitel=

[4] Perens, B., The Open Source Definition, in: DiBona, C., Ockman, S., Stone, M. (Hrsg.), Opensources: Voices from the Open Source Revolution, Sebastopol: O'Reilly 1999, 171-188, S. 171.

[5] Saleck, T., Chefsache Open Source, Wiesbaden: Vieweg 2005, S. 6 f.

1.1 Sicherheit in Open-Source-Software

Befürworter der OSS sehen im Modell der Software-Entwicklung einen Vorteil hinsichtlich der Sicherheit von Open-Source-Produkten. Einen zentralen Aspekt der Sicherheitsphilosophie hat Raymond in seinem Aufsatz „The Cathedral and the Bazaar“ veröffentlicht: „Given enough eyeballs, all bugs are shallow.“[6] Da der Quellcode schon früh im Software-Entwicklungsprozess einsehbar ist und in kurzen Zeitabständen inkrementelle Entwicklungsfortschritte veröffentlicht werden, ergeben sich Anreize, OSS anzuwenden, weil Probleme innerhalb kurzer Zeitabstände gelöst werden. Die Motivation für Software-Entwickler hingegen lässt sich mit dem sofortigen Einbinden ihrer Programmierarbeit in der Software begründen.

Die Offenlegung des Quellcodes ermöglicht Beta-Testern eine vollumfängliche Inspektion des Codes auf Sicherheitslücken. Die Open-Source Bewegung geht demnach davon aus, dass Fehler in der Software relativ schnell entdeckt und beseitigt werden. Da viele Software-Fehler gleichzeitig Sicherheitslücken darstellen, wird vermutet, dass Open-Source-Produkte im Sinne des Konzepts *Security through Transparency* sicherer sind.[7]

1.2 Sicherheit in proprietärer Software

Im Gegensatz zur Postulierung der Sicherheit durch Offenlegung des Quellcodes steht das Konzept *Security by Obscurity*, welches als Sicherheit durch Geheimhaltung umschrieben werden kann. Die Grundidee hinter diesem Konzept ist die Annahme, dass ein Computer-System jeglicher Art so lange sicher ist, als keine Personen ausserhalb des Entwicklerteams Informationen über die Mechanismen und Arbeitsweise des Computer-Systems verfügen. Wird das Konzept auf die Thematik der Sicherheit von Open-Source und proprietärer Software angewendet, so sollen durch die Verhinderung der Veröffentlichung des Quellcodes, wie es bei proprietären Produkten der Regelfall ist, Sicherheitslücken schwerer auffindbar gemacht werden.[8] Ein weiteres Argument der Verfechter der Sicherheit proprietärer Software zielt auf die im Softwareentwicklungsprozess integrierte Qualitätssicherung, die Sicherheitslücken bereits vor Veröffentlichung der Software aufdeckt und ausräumt.

2 Veröffentlichung von Sicherheitslücken

Sicherheitslücken sind Zustände in Computersystemen, die folgende Eigenschaften aufweisen können:

- Befehle können ohne Berechtigung ausgeführt werden.
- Eine Person hat unberechtigterweise Zugriff auf Daten.
- Es besteht die Möglichkeit, sich gegenüber dem System als jemand anderer auszugeben.
- Angreifer können geschützte Informationen sammeln.
- Es liegt ein Schwachpunkt in einem System vor, den ein Angreifer möglicherweise nutzen könnte, um Zugriff auf das System oder Daten zu erhalten.[9]

Eine Sicherheitslücke ist gemäss dem Rating-System der National Vulnerability Database (NVD), einer amerikanische Organisation, die eine Datenbank über Sicherheitslücken unterhält,[10] hoch gefährlich, falls es

- einem Angreifer möglich ist, den Sicherheitsschutz eines Systems zu durchbrechen oder
- einem Angreifer möglich ist, die komplette Kontrolle über ein System zu erlangen.

2.1 Schwachstellen in Browsern

Sicherheitslücken in Web Browsern und ihre Auswirkungen bei Ausnutzung können in fünf Klassen eingeteilt werden:

- *Web Page Spoofing* ist eine Methode, Internet-Adressen zu fälschen. Dem Opfer einer Spoofing-Attacke wird glaubhaft gemacht, dass er sich auf einer vertrauenswürdigen oder sicheren Webseite befindet, obwohl er mit der Webseite des Angreifers kommuniziert.[11]
- Sicherheitslücken durch *ActiveX Controls und Java Applets*: Software-Komponenten, die aktiv Operationen auf dem Rechner des Benutzers ausführen, werden zu bösartigem Zweck missbraucht.
- *Scripting* Sicherheitslücken: Programmiersprachen wie JavaScript und VB-Script können bei der Implementierung in Web Browsern Sicherheitslücken hervorrufen. Die populärste Form ist das Cross-Site Scripting (XSS), bei welcher bösartiger Code in Script-Sprachen, die von Web Browsern interpretiert werden können, ausgeführt wird.[12]
- *Content Type* Sicherheitslücken: Bei dieser Art Ausnutzung von Sicherheitslücken wird der Web Browser dadurch getäuscht, dass er vermeintlich ungefährliche Daten wie Bilder oder Audiodateien öffnet, welche in Wirklichkeit ausführbaren und bösartigen Code beinhalten.[13]
- Bei *Buffer Overflows* können durch Fehler in der Programmierung größere Datenmengen als vorgesehen in einen temporären Speicherbereich (Buffer) eines Programms geschrieben werden, wodurch nachfolgende Informationen im Speicher bösartig überschrieben werden können.[14]

2.2 Veröffentlichungs-Strategien

Die Bekanntmachung von Sicherheitslücken im Internet ist ein umstrittenes Thema. Während Informationen über Sicherheitslücken einerseits den Benutzern Vorsichtsmassnahmen zur Verhinderung von Sicherheitsproblemen ermöglichen, können andererseits Angreifer von diesen Informationen profitieren und Sicherheitslücken besser ausnutzen.[15]

Befürworter von Veröffentlichungen zu Sicherheitslücken propagieren, dass publik

[6] Raymond, S., The Cathedral & The Bazaar, Sebastopol: O'Reilly 1999, S. 41 ff.

[7] Gehring, R., Sicherheit mit Open Source – Die Debatte im Kontext, die Argumente auf dem Prüfstein, in: Gehring, R., Lutterbeck, B. (Hrsg.), Open Source Jahrbuch 2004, Berlin: Lehmanns Media 2004, S. 208-235, S. 212 f.

[8] Buhl, H., Dzienziol, J.: Mehr Sicherheit durch Open Source – Irrweg oder Zielgerade?, in: Wirtschaftsinformatik (2003) Nr. 4, S. 474-482, S. 477.

[9] http://www.cve.mitre.org/

[10] http://nvd.nist.gov/

[11] http://www.newhorizons.com/elevate/Elevate_July2005.pdf

[12] http://su2.info/uni/sosi/xss_paper.pdf

[13] http://www.newhorizons.com/elevate/Elevate_July2005.pdf

[14] http://searchsecurity.techtarget.com/sDefinition/0,sid14_gci549024,00.html

[15] Arora, A., Krishnan, R., Nandkumar, A., Telang, R., Yang, Y., Impact of Vulnerability Disclosure and Patch Availability – An Empirical Analysis, Arbeitsbericht der H. John Heinz III School of Public Policy and Management, Carnegie Mellon University, Pittsburgh 2004, S. 1.

gemachte Informationen einen Anreiz bieten, Patches zur Behebung des Problems bereitzustellen. Unveröffentlichte Sicherheitslücken üben hingegen keinen Druck auf den Anbieter aus, sich um Schwachstellen zu kümmern.[16]

Gegner führen hingegen an, dass Benutzer machtlos gegenüber Hackern werden, sobald detaillierte Informationen über eine Sicherheitslücke veröffentlicht werden. Angreifer sind so in der Lage, Angaben über Sicherheitslücken einzusehen und anschließend Schwachstellen effizient auszunützen.[17] Im Zusammenhang mit der Verbreitung von Informationen über Sicherheitslücken können drei Strategien unterschieden werden:[18]

- *Full-Disclosure-Ansatz*: Die sofortige Veröffentlichung von detaillierten Angaben über eine Sicherheitslücke nach deren Entdeckung, inklusive der Anleitung, wie die Schwachstelle ausgenutzt werden kann.
- *Responsible-Disclosure-Ansatz*: Diese Strategie sieht vor, den Software-Anbieter nach Entdeckung einer Sicherheitslücke vor deren Veröffentlichung zu informieren, um einen Patch bereit zu stellen, der später zeitgleich mit der Publikation der Schwachstelle verfügbar ist.
- *No Disclosure-Ansatz*: Dieser Ansatz sieht grundsätzlich keine Vorteile in der Bekanntmachung von Sicherheitslücken und propagiert, Schwachstellen möglichst geheim zu halten. Es wird davon ausgegangen, dass die Verantwortung für Software-Sicherheit alleine beim Anbieter liegt.

Welcher Ansatz mehr Vor- als Nachteile verspricht, ist umstritten und Gegenstand aktueller Forschungsarbeiten.

3 Unabhängige Studien

Anbieter von proprietärer Software finanzieren Studien und Beiträge, um ihre Position im Software-Markt im Kampf um Marktanteile verteidigen zu können. Dabei spielt der Nachweis der angeblich überlegenen Sicherheit ihrer Produkte eine wichtige Rolle im Marketing. Die Open-Source-Community reagiert nahezu auf alle derartigen Veröffentlichungen mit einer Gegenstudie.[19] Die Problematik, die mit diesen Studien verbunden ist, ist nahe liegend – die Schlussfolgerungen sind nicht unabhängig vom Auftraggeber und demnach in ihrer Objektivität eingeschränkt.

Obwohl eine Vielzahl von vergleichenden Beiträgen zwischen Open-Source-Software und proprietärer Software veröffentlicht wurde, gibt es nur wenige empirische Studien und systematische Untersuchungen aus sicherheitstechnischer Sicht.[20] Nach eingehender Recherche erweisen sich sechs Studien als aussagekräftig. Diese Aufsätze befassen sich einerseits mit Modellen zur Fehlerdynamik in der Entwicklung von Open-Source und proprietärer Software und andererseits mit allgemeinen Sicherheits- und Qualitätsfragen.

Eine im Jahr 1995 durchgeführte Studie von Miller et al.[21] nutzt ein System, das Zufallsdatenketten generiert, welche als Input in verschiedene Programme unterschiedlicher UNIX-Versionen eingegeben werden. Die Autoren verwenden als Maß für die Zuverlässigkeit der Software, ob ein Programm abstürzt oder hängen bleibt, falls zufällige Datenketten an die ausgewählten Programme gesendet werden. Die Studie testet mehr als 80 Tools von neun UNIX-Versionen. Sieben der UNIX-Betriebssysteme sind proprietäre Produkte, zwei stehen unter einer Open-Source-Lizenz. Die Linux Tools sowie die GNU Tools wiesen eine Fehlerquote unter 10 Prozent auf. Die proprietären Tools schnitten mit Fehlerquoten zwischen 15 und 43 Prozent wesentlich schlechter ab. Die Studie impliziert somit eine höhere Ausfallsicherheit für unter dem Open-Source-Modell entwickelte Tools.

In der Studie von McCormack werden 29 Software-Projekte im Hinblick auf verschiedene Ansätze zur Durchführung der Projekte analysiert.[22] Die Untersuchung befasst sich mit der Frage, ob klassische Software-Entwicklungsmodelle, z.B. das Wasserfallmodell, oder neuere, evolutionäre Modelle wie das Spiralmodell oder Prototyping Erfolg versprechender sind. Ziel der Studie ist die Herausarbeitung von Faktoren, die maßgebend zu einem erfolgreichen Abschluss eines Projektes führen. Ausschlaggebend für die Qualität des Endproduktes ist die Geschwindigkeit, mit der Feedbacks auf Design-Änderungen erfolgen. Die durch Tests gewonnenen Feedbacks werden auf täglicher Basis in der Entwicklung des Software-Codes berücksichtigt. Keines der geprüften Software-Projekte mit langen Feedbackzeiten weist überdurchschnittliche Qualität auf. Somit lässt sich sagen, dass die konsequente Veröffentlichung und Verbesserung von Quellcode während der Entwicklung von OSS die Qualität und somit die Sicherheit begünstigt.

Die Untersuchung von Payne beschäftigt sich mit dem Open-Source-Software-Entwicklungsprozess im Hinblick auf die Sicherheit von Betriebssystemen.[23]. Payne definiert vier Dimensionen von Sicherheit: Confidentiality, Integrity, Availability und Audit. Diesen Dimensionen werden Werte zugeordnet, die eine Vergleichbarkeit der Systemeigenschaften zulassen. Drei Betriebssysteme, davon zwei aus der Open-Source-Kategorie und ein proprietäres, werden mit Hilfe der genannten Dimensionen auf Sicherheitseigenschaften und Schwachstellen getestet. Die Gesamtpunktzahl eines geprüften Betriebssystems ergibt sich aus der Kombination der Resultate aus den Tests auf Sicherheitseigenschaften und Schwachstellen. Die beiden Open-Source-Betriebssysteme erzielen bessere Resultate als das proprietäre System, wobei zwischen den beiden Open-Source-Lösungen signifikante Unterschiede bezüglich der Sicherheit bestehen – OpenBSD schneidet deutlich besser ab als das Debian-System. Payne schließt aus seinen Resultaten, dass jede Diskussion um das Pro und Contra von Software-Sicherheit sich nicht nur auf die Offenlegung des Quelltextes stützen darf, sondern insbesondere der Audit-Prozess während der Entwicklung einen maßgebenden Einfluss auf die Sicherheit ausübt.

[16] http://james.bond.edu.au/courses/inft13332@031/slides/fulldisc.pdf

[17] Farrow, R., The Pros and Cons of Posting Vulnerabilities, in: Network Magazine 15 (2000) 10, S. 146.

[18] http://www.blackhat.com/presentations/bh-usa-02/bh-us-02-blake-politics.ppt

[19] Boulanger, A., Open-Source versus Proprietary Software, Is One More Reliable and Secure than The Other?, in: IBM Systems Journal 44 (2005) 2, S. 239-248, S. 240.

[20] Payne, C., On the Security of Open Source Software, in: Information Systems Journal Volume 12 (2002) Nr. 1, S 61-79, S. 72.

[21] Miller, B., Koski, D., Lee, C., Maganty, V., Murthy, R., Natarajan, A., Steidl, J., Fuzz Revisited: A Re-examination of the Reliability of UNIX Utilities and Services, Arbeitsbericht Nr. 53706-1685 des Computer Science Department, University of Wisconsin, Madison 1995, S. 2 ff.

[22] McCormack, A., Product-Development Practices That Work: How Internet Companies Build Software, in: Sloan Management Review Volume 42 (2001) Nr. 2, S. 75-84, 76 ff.

[23] Payne, C., On the Security of Open Source Software, in: Information Systems Journal Volume 12 (2002) Nr. 1, S. 61-79, S. 63 ff.

Die Studie von Reasoning untersucht die Open-Source-Datenbank MySQL und vergleicht sie mit anderen kommerziellen Datenbanken.[24] Bei MySQL findet Reasoning in den 236.000 Zeilen Quelltext mittels automatischer Inspektion 21 Softwarefehler, was einer Dichte von 0.09 Fehlern pro 1000 Zeilen Code entspricht. Die Codequalität der Open-Source-Lösung wird anschließend mit der durchschnittlichen Schadensdichte proprietärer Datenbanksoftware verglichen, welche 0.57 Fehler pro 1000 Zeilen Quellcode aufweist. Daraus wird die Schlussfolgerung gezogen, dass die getestete MySQL Version 4.0.16 eine sechsmal geringere Schadensdichte aufweist als vergleichbare kommerzielle Datenbanken.

Koetzle et al. haben einen differenzierten Ansatz entwickelt, um die Sicherheit von Betriebssystemen zu vergleichen.[25] Sie definieren diverse Messgrössen, um die Sicherheit der getesteten Betriebssysteme zu bestimmen. Zentral sind die Anzahl gefundener Sicherheitslücken und die Gesamtanzahl Tage, die zwischen der Veröffentlichung einer Sicherheitslücke und der Bereitstellung eines Patches durch den Anbieter des Betriebssystems liegen. Dabei wird spezifisch für die Open-Source-Betriebssysteme ein Maß Distributions-Risikotage entwickelt, welches berücksichtigt, dass Linux-Distributionen oft Bündel von Quellcode aus verschiedenen Quellen sind. Daraus ergibt sich möglicherweise eine Lücke zwischen der Veröffentlichung eines Patches für eine bestimmte Komponente und der Einbindung des Patches in eine neue Linux-Distribution.

Microsoft weist mit 25 Tagen die geringste durchschnittliche Anzahl Tage zwischen der Veröffentlichung einer Sicherheitslücke und der Bereitstellung eines Patches auf. MandrakeSoft schneidet sowohl in den Kategorien Anzahl Risikotage (82) als auch Anzahl Distributions-Risikotage (56) am schlechtesten ab. Microsoft hebt sich mit 128 gefundenen und reparierten Sicherheitslücken deutlich von der getesteten Konkurrenz ab. Der relative Anteil gravierender Sicherheitslücken ist jedoch verglichen mit den Linux-Distributionen erheblich höher. Koetzle et al. vergeben den ersten Rang bezüglich Sicherheit den beiden Betriebssystemen der Fir-

[24] http://www.reasoning.com/pdf/MySQL_White_Paper.pdf

[25] http://download.microsoft.com/download/9/c/7/9c793b76-9eec-4081-98eff1d0ebfffe9d/LinuxWindowsSecurity.pdf.

men Red Hat und Microsoft mit der Begründung, dass zwar Red Hat Linux mehr Sicherheitslücken aufweist, diese jedoch seltener als gravierend einzustufen sind.

Challet/Le Du entwickelten ein Modell zur Beschreibung der Fehlerdynamik in der Open-Source- und der proprietären Entwicklungsmethode.[26] Ziel ihrer Modellierung war es, die Entwicklung der Fehlerdynamik unter gewissen Annahmen abzubilden. Das Modell von Challet/Le Du zeigt im Wesentlichen, dass Software-Projekte in Abhängigkeit von der Zeit trotz durchschnittlichen Programmierern und Betreuern zu einem fehlerlosen Status führen können. Der fehlerfreie Status wird von Open-Source-Projekten regelmäßig in kürzerer Zeitdauer erreicht als in der Entwicklung von proprietärer Software. Die Autoren konstatieren weiter, dass OSS hohe Qualität aufweisen kann, obwohl die Programmierer tendenziell weniger Erfahrung und Fachkönnen aufweisen als Mitarbeiter von Unternehmen, die proprietäre Software entwickeln. Unter sonst gleichen Bedingungen werden Open-Source-Projekte regelmäßig schneller beendet als proprietäre Projekte. Jedoch kann proprietäre Software unter günstigen Voraussetzungen, wie einer großen Anzahl exzellenter Programmierer und einer großen Benutzergemeinde, im Hinblick auf die Anzahl Sicherheitslücken besser sein.

Die Auswertung der vorgestellten Studien ermöglicht keine pauschale Empfehlung des Open-Source oder proprietären Software-Modells. Bei der Entscheidungsfindung müssen Punkte in Betracht gezogen werden, die unabhängig vom Entwicklungsmodell sind, so etwa die Fragen, welche Sicherheits-Reviews von welchen Personen durchgeführt wurden, wie schnell Patches von den Anbietern bereitgestellt werden und in welchen Zeitabständen Software-Versionen den Testern zur Verfügung stehen.

4 Vergleich der beiden Browser

Das Ende des Konkurrenzkampfes zwischen Microsoft und Netscape verhalf dem MS Internet Explorer (MSIE) zum scheinbar unantastbaren Marktführer im Browser-Markt. Bis Ende 2004 gab es keine Alternativen, die Microsoft unter Druck setzten.

Seit Veröffentlichung des Mozilla Firefox 1.0 am 9.11.2004 verzeichnet der Open-Source-Web Browser einen steigenden Marktanteil, der aktuell je nach Quelle zwischen fünf und zehn Prozent liegt.[27] Sicherheitsprobleme, negative Schlagzeilen und ein Stillstand der Entwicklung führten dazu, dass der MSIE erstmals in seiner Geschichte signifikant Marktanteile verliert, nämlich an den Mozilla Firefox, einen Web Browser der Mozilla Foundation, der ursprünglich nur als Nebenprojekt der Entwicklergemeinde gedacht war.[28]

Der Mozilla Firefox als quelloffener Web Browser ist derzeit der grösste Herausforderer des MSIE und momentan zweitwichtigstes Produkt innerhalb des Browser-Marktes. Der Mozilla Firefox eignet sich daher für einen Sicherheitsvergleich mit dem seit Jahren marktführenden MSIE.

4.1 Methodisches Vorgehen

Das Zeitfenster für die Datenerhebung beginnt mit der Veröffentlichung der endgültigen Version des Mozilla Firefox 1.0 im November 2004 und erstreckt sich über 12 Monate bis Ende Oktober 2005. Es ist wichtig anzumerken, dass der Mozilla Firefox unter diesem Namen erst seit Februar 2004 existiert, zuvor als Web Browser-Projekt seit September 2002 unter dem Namen Phoenix erarbeitet wurde und seither weiterentwickelt wird. Der getestete Web Browser von Microsoft ist der MSIE 6.0.

Alle Daten über Sicherheitslücken von Mozilla Firefox und MSIE im definierten Zeitfenster, die öffentlich (d.h. online verfügbar) sind, wurden ausgewertet. Als Quellen dienten die Webseiten von Mozilla und Microsoft, Sicherheitslücken veröffentlichende Mailinglisten [29] sowie das Projekt „Common Vulnerabilities and Exposures“ (CVE), ein Verzeichnis mit standardisierten Namen aller publik gemachten Sicherheitslücken.[30] Hauptaufgaben von CVE sind die Erstellung und Pflege einer Liste von publizierten Sicherheitslücken sowie die Prüfung der Authentizität neu gefundener Schwachstellen. Die gefundenen Sicherheitslücken werden nach Gefahrengrad klassifiziert.

Vier quantitative Maße wurden für den Vergleich der beiden Browser definiert:

- *Anzahl Sicherheitslücken*: Summe der Sicherheitslücken, die im definierten Zeitfenster in einem spezifischen Software-Produkt gefunden und veröffentlicht werden.
- *Prozentsatz reparierter Sicherheitslücken*: Entspricht dem relativen Anteil veröffentlichter und innerhalb des definierten Zeitfensters durch einen Patch reparierter Sicherheitslücken im Vergleich zur Gesamtzahl veröffentlichter Sicherheitslücken.
- *Anzahl Tage der Risikoaussetzung*: Durchschnittliche Anzahl von Tagen, die zwischen der Veröffentlichung einer Sicherheitslücke und dem Bereitstellen eines Patches durch den Software-Anbieter vergehen.
- *Prozentsatz hoch gefährlicher Sicherheitslücken*: Relativer Anteil hoch gefährlich eingestufter Sicherheitslücken, die im definierten Zeitfenster im Vergleich zur Gesamtzahl veröffentlicht wurden.

4.2 Resultate

Im Analysezeitraum vom November 2004 bis Ende Oktober 2005 sind insgesamt 30 Sicherheitslücken in MSIE 6.0 identifiziert und veröffentlicht worden. Von den 30 Sicherheitslücken sind 17 von der NVD als hoch gefährlich eingestuft worden, was einem Anteil von 57 Prozent entspricht.

Als mittel und wenig gefährlich wurden neun respektive vier Sicherheitslücken klassifiziert. Sie erlauben keinen Zugang zu wertvollen Informationen und gewähren keine Kontrolle über ein System, vermitteln aber dem Angreifer Wissen, mit dem er möglicherweise andere Sicherheitslücken auffinden und ausnutzen kann. Abbildung 1 fasst diese Erkenntnisse überblicksartig zusammen.

Die Auswertung der Anzahl Schwachstellen, die während des betrachteten Zeitfensters durch Fehlen eines Patches von Microsoft nicht repariert werden konnten, beläuft sich auf 12 Sicherheitslücken, das entspricht einem Anteil von 40 Prozent.

Die Auszählung für den Mozilla Firefox 1.0 bis 1.0.7 ergibt die Publikation von 55 Sicherheitslücken. Von diesen Sicherheits-

[26] Challet, D., Le Du, Y., Microscopic Model of Software Bug Dynamics, Closed Source versus Open Source, in: International Journal of Reliability, Quality and Safety Engineering (2004) Nr. 10., S. 1-13.

[27] http://www.answers.com/topic/usage-share-of-web-browsers

[28] http://www.zdnet.de/itmanager/tech/0,39023442,39127492,00.htm?041108164759

[29] https://lists.grok.org.uk/mailman/listinfo/full-disclosure; http://www.ntbugtraq.com/; http://www.securityfocus.com/archive/1; http://www.vulnwatch.org/.

[30] http://www.cve.mitre.org/

lücken stuft die NVD 14 als hoch gefährlich ein, was einem Anteil von 25 Prozent entspricht. Als mittel gefährlich werden 31 klassifiziert und 10 stellen nur eine geringe Gefahr dar. Die Mozilla Foundation hat alle im Analysezeitraum erfassten Sicherheitslücken durch die Bereitstellung einer neuen Mozilla Firefox-Version im Sinne einer Sicherheitsaktualisierung repariert. Mit der Version 1.0.7 wurden am Ende des Analysezeitraums 100 Prozent der veröffentlichten Sicherheitslücken behoben.

Der Mozilla Firefox weist im Analysezeitraum 25 Sicherheitslücken mehr auf als der MSIE mit insgesamt 30 Schwachstellen. Diese Zahlen müssen jedoch relativiert werden, da der MSIE mit 17 hochgefährlichen Sicherheitslücken in dieser Kategorie drei Schwachstellen mehr als der Mozilla Firefox aufweist. Wird die Anzahl hoch gefährlicher Sicherheitslücken ins Verhältnis zur Gesamtsumme aufgetretener Schwachstellen pro Web Browser gesetzt, schneidet das Software-Produkt von Microsoft mit einem Anteil von 57 Prozent im Gegensatz zum Mozilla Firefox mit 25 Prozent ebenfalls schlechter ab.

Für die Berechnung der Anzahl Risikotage wird für jede Sicherheitslücke die zeitliche Differenz zwischen dem Datum der erstmaligen Veröffentlichung und der Bereitstellung eines Patches oder Updates berechnet. Um die Aussagekraft dieser Kennzahl zu erhöhen, wird einerseits das arithmetische Mittel der Anzahl Risikotage pro Sicherheitslücke und andererseits die kumulierte Anzahl Risikotage pro Web Browser bestimmt.

Die kumulierte Anzahl Risikotage für den MSIE beträgt 327 Tage, falls nur jene Sicherheitslücken berücksichtigt werden, die Microsoft innerhalb des Analysezeitraums behoben hat. Dieses Resultat ist stark nach unten verzerrt und muss um die Anzahl Risikotage der unreparierten Sicherheitslücken bis Ende Oktober 2005 ergänzt werden, was zu einer kumulierten zeitlichen Risikoaussetzung von 3005 Tagen führt. 1277 Tage davon war der MSIE einem hohen Risiko ausgesetzt, was einem Anteil von 42 Prozent entspricht.

Für Mozilla Firefox werden 1780 Tage berechnet, die der Web Browser einem Risiko innerhalb des Analyseraums ausgesetzt war. 473 Tage davon war der Mozilla Firefox durch hoch gefährliche Sicherheitslücken belastet. Dies ergibt einen Anteil von 27 Prozent, der deutlich unter dem errechneten Wert für den MSIE liegt.

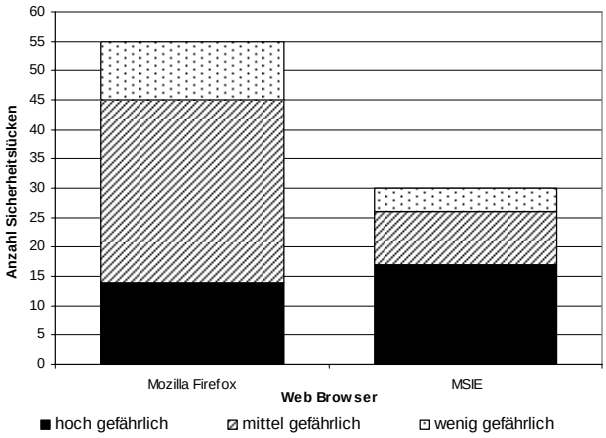

Abb. 1: Anzahl und Schwere der Sicherheitslücken

Eine Sicherheitslücke besteht für den MSIE im Durchschnitt 100 Tage, für den Mozilla Firefox hingegen nur 32 Tage. Dies weist darauf hin, dass die Mozilla Foundation Sicherheitslücken deutlich schneller behebt. Der hohe Durchschnitt für Microsoft resultiert maßgeblich daraus, dass 12 Sicherheitslücken im MSIE bis zum 31. Oktober 2005 nicht eliminiert wurden.

Die Analyse kann weiter verfeinert werden. Dazu wird bestimmt, wie viele Sicherheitslücken Microsoft und die Mozilla-Foundation innerhalb eines bestimmten Zeitraums beheben. In Abbildung 2 werden die in verschiedene Kategorien erfassten Sicherheitslücken hinsichtlich der Anzahl Risikotage unterteilt.

Im MSIE werden 14 Sicherheitslücken innerhalb der ersten 30 Tage nach Publizierung behoben. Dies entspricht einem Anteil von 47 Prozent von allen aufgetretenen Sicherheitslücken im Analysezeitraum. Acht Sicherheitslücken weisen eine Anzahl Risikotage von Null auf, da sie im Rahmen des Responsible Disclosure veröffentlicht wurden. Weiter zeigt Abbildung 2, dass 11 hoch gefährliche Sicherheitslücken innerhalb von 30 Tagen nach deren Veröffentlichung repariert wurden. Fünf Sicherheitslücken stellten ein Sicherheits-Risiko für eine Zeitspanne von einem bis drei Monate dar, wobei eine als mittel gefährlich eingestufte Sicherheitslücke bis zum Ende des Analysezeitraums nicht behoben wurde.

Alle 11 Sicherheitslücken im MSIE, die drei Monate und länger bestehen, wurden bis Ende Oktober 2005 nicht repariert. Drei davon sind von der NVD als hoch gefährlich klassifiziert worden. Eine hoch gefährliche Sicherheitslücke besteht länger als ein Jahr.

Abbildung 3 zeigt, dass 36 von insgesamt 55 Sicherheitslücken im Mozilla Firefox in weniger als 31 Tagen repariert wurden, was einem Anteil von 65 Prozent entspricht. Neun davon waren hoch gefährliche Schwachstellen.

Keine der Sicherheitslücken weist eine Anzahl Risikotage von Null auf, obwohl die Mozilla Foundation das Full Disclosure-Verfahren nur teilweise befürwortet und somit eine möglichst frühe vollumfängliche Offenlegung von Sicherheitslücken nicht fördert.[31] 16 Sicherheitslücken wurden zwischen 31 und 90 Tagen nach deren erstmaliger Veröffentlichung durch ein Update von der Mozilla Foundation behoben. Vier davon waren von der NVD als hoch gefährlich eingestuft worden. Drei Sicherheitslücken bestanden mehr als drei Monate lang, jedoch keine länger als ein Jahr.

Der MSIE wies im Analysezeitraum mit 30 gegenüber 55 deutlich weniger Sicherheitslücken auf als der Mozilla Firefox. Diese Ergebnisse basieren jedoch nicht auf identischen Ausgangslagen, da die Nutzungsanteile der beiden Web Browser nicht mitberücksichtigt werden. Mit knapp 90 Prozent Nutzungsanteilen im Analysezeitraum für den MSIE und knapp 10 Prozent für Mozilla Firefox muss davon ausgegangen werden, dass im MSIE durch die Mehrzahl von Anwendern die Wahrscheinlichkeit für das Auffinden von Sicherheitslücken höher ist. Die Resultate des Vergleichs der Anzahl Sicherheitslücken unter Berücksichtigung der Nutzungsanteile unterstreichen daher zusätzlich den Vorteil vom MSIE gegenüber Mozilla Firefox.

[31] http://www.mozilla.org/projects/security/security-bugs-policy.html

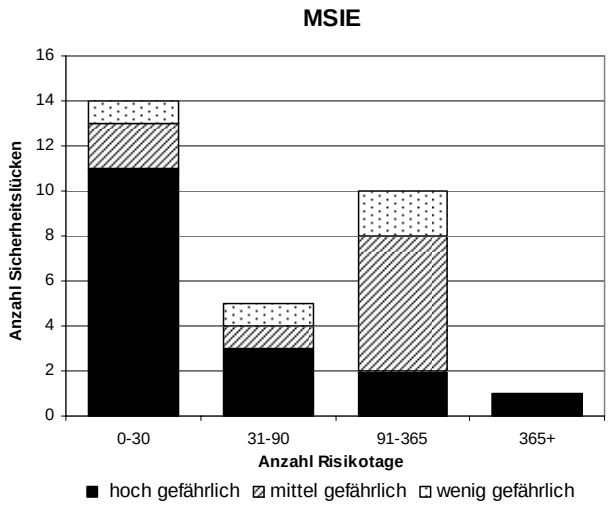

Abb. 2: Anzahl Risikotage für MSIE

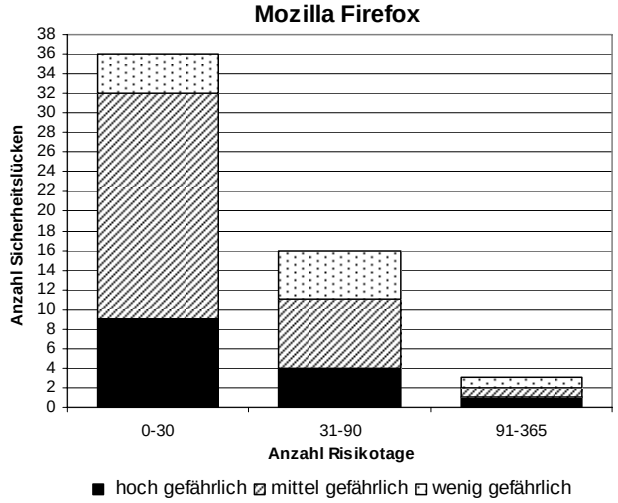

Abb. 3: Anzahl Risikotage für Firefox

57 Prozent aller Sicherheitslücken im MSIE sind als hoch gefährlich einzustufen, in Mozilla Firefox hingegen nur 25 Prozent. Hoch gefährlichen Sicherheitslücken werden im Rahmen des Sicherheitsvergleichs hohe Bedeutung beigemessen. Unter Berücksichtigung der Sicherheitslücken nach Gefahrengrad verringert sich der Vorteil der geringeren Anzahl Sicherheitslücken im MSIE gegenüber Mozilla Firefox.

Die kumulierte Anzahl Tage Risikoaussetzung ist mit 3005 Tagen für den MSIE deutlich höher als für den Mozilla Firefox mit 1780 Tagen. Werden auch diese beiden Kennzahlen bezüglich der Gefahrengrade von Sicherheitslücken unterschieden, so war der MSIE insgesamt 1277 Tage einem hohen Risiko exponiert, der Mozilla Firefox hingegen nur 473 Tage. Berücksichtigt man zusätzlich die drei hoch gefährlichen und bis 31. Oktober 2005 unreparierten Sicherheitslücken im MSIE, so muss bei einer Verlängerung des Analysezeitraums davon ausgegangen werden, dass die tatsächliche Anzahl Tage hoher Risikoaussetzung 1277 übersteigen wird.

Die durchschnittliche Anzahl Tage Risikoaussetzung pro Sicherheitslücke ist ein Maß zur Bestimmung der Reaktionszeit eines Software-Anbieters auf die Veröffentlichung einer Sicherheitslücke. Es drückt die durchschnittliche Zeitdauer in Tagen zwischen der erstmaligen Veröffentlichung bis zur Bereitstellung eines Patches zur Behebung der Sicherheitslücke aus. Für Microsoft errechnet sich ein Durchschnitt von 11 Tagen, falls in der Analyse nur Sicherheitslücken berücksichtigt werden, die innerhalb des Analysezeitraums repariert wurden. Im Gegensatz zu Mozilla Firefox mit 32 durchschnittlichen Tagen liegt dieser Wert deutlich niedriger. Der Grund für diese Diskrepanz ist die von Microsoft verfolgte Veröffentlichungspolitik. Durch Anwendung des Responsible Disclosure-Verfahren weisen acht der insgesamt 18 reparierten Sicherheitslücken null Risikotage auf, da die Veröffentlichung der Informationen über die Sicherheitslücke zeitgleich mit der Bereitstellung eines Patches erfolgte.

Die Auswertung der durchschnittlichen Anzahl Risikotage muss für einen objektiven Vergleich um die bis zum Ende des Analysezeitraums nicht behobenen Sicherheitslücken im MSIE ergänzt werden, was den Durchschnitt von 11 Tagen auf 100 Tage erhöht. Daraus ergibt sich, dass der MSIE dem Risiko durch eine Sicherheitslücke im Durchschnitt mindestens 68 Tage länger ausgesetzt ist als der Mozilla Firefox.

Berücksichtigt man schließlich die Verteilung der Sicherheitslücken auf die Zeitintervalle, wie in den Abbildungen 2 und 3 dargestellt, so fällt der Sicherheitsvergleich zu Gunsten des Mozilla Firefox aus, obwohl der MSIE deutlich weniger Sicherheitslücken aufweist. Entscheidende Sicherheitsvorteile wie die geringere Anzahl Risikotage und Anzahl hoch gefährlicher Sicherheitslücken von Mozilla Firefox überwiegen die reine Auszählung von Sicherheitslücken. Zudem muss beachtet werden, dass 40 Prozent aller im Analysezeitraum veröffentlichten Sicherheitslücken im MSIE bis zum 31. Oktober 2005 nicht behoben wurden.

Fazit

Die Auswertung der quantitativen Daten über die Anzahl Sicherheitslücken und die dadurch bedingte Risiko-Exposition bis zur Installation eines entsprechenden Patches ergibt im Gesamturteil Sicherheitsvorteile für den Mozilla Firefox. Zwar weist der Mozilla Firefox mehr Sicherheitslücken auf als der MSIE, diese bestehen jedoch kürzere Zeit, da die Mozilla Foundation schneller Patches verfügbar macht als Microsoft.

Um die Aussagekraft von Aussagen über die Sicherheit von MS Internet Explorer und Mozilla Firefox zu erhöhen, können in Zukunft weitere Sicherheitsvergleiche durchgeführt werden. Anhand der Auswahl verschiedener Zeitfenster kann möglicherweise ein Trend bestimmt werden, wie sich die beiden analysierten Browser im Hinblick auf ihre Sicherheit entwickeln.