**Univerzita Karlova v Praze**

**Filozofická fakulta**

Ústav informačních studií a knihovnictví

# Bakalářská práce

Dmytro Popov

**Srovnání bezpečnosti open source a proprietárního softwaru**

Comparison of open source and proprietary software security

Praha 2014

Vedoucí práce: PhDr. Mgr. Jan Pokorný, Ph.D.

*Chtěl bych na tomto místě poděkovat PhDr. Mgr. Janu Pokornému, Ph.D. za ochotu a pomoc nejen při realizaci bakalářské práce, ale také při vymýšlení tématu.*

*Prohlašuji, že jsem bakalářskou práci vypracoval samostatně, že jsem řádně citoval všechny použité prameny a literaturu a že práce nebyla využita v rámci jiného vysokoškolského studia či k získání jiného nebo stejného titulu.*

*V Praze, dne 9.5. 2014*

……..

*Podpis*

**Abstrakt:**

Hlavním zaměřením této bakalářské práce je porovnání open source a proprietárního softwaru, a to z hlediska jejich bezpečnosti. Primárním cílem je určit, který z nich poskytuje největší míru ochrany informací, jaké je za daných okolností možné dosáhnout. V úvodu práce jsou vysvětleny a definovány termíny, nastíněna základní historie softwaru a rozvoj oboru počítačové bezpečnosti. V hlavní části se podrobně zabývá otázkou, který z nich poskytuje větší míru bezpečnosti jak pro jednotlivé uživatele, tak pro vládní složky či komerční společnosti, což je doplňováno relevantními statistikami a příklady. Součástí této práce je také zmapování algoritmů používaných pro hodnocení bezpečnosti a databází, jež uchovávají data týkající se bezpečnosti informačních technologií.

**Klíčová slova:**

open source, proprietární software, software, bezpečnost, GNU/Linux, Microsoft

**Abstract:**

The main focus of this bachelor thesis is to compare open source and proprietary software security and to ascertain which software provides the best possible protection for information. The thesis commences with a definition of the frequently used terms in the thesis and is followed by an outline of how software evolved and the history of computer security. The main focus of this work study examines in detail which software type provides better security for end users, government agencies and commercial companies. The theoretical arguments presented are supported by relevant statistics and examples. This thesis also describes the algorithms used to measure and compare the different software security and databases that are used to store information security related data.

**Key words:**

open source, proprietary software, software, security, GNU/Linux, Microsoft

# Obsah

Úvod....6

1. Definice a vymezení pojmů....7

2. Historie vývoje softwaru a softwarové bezpečnosti....10

3. Problémy při srovnání....13

3.1 Kvalita analýz a jejich spolehlivost v čase....13

3.2 Definice bezpečnosti a hledání správného systému měření bezpečnosti....14

4. Rozbor a srovnání....15

4.1 Pozitivní a negativní důsledky zveřejnění zdrojového kódu....15

4.1.1 Teoretické principy softwarové bezpečnosti....15

4.1.2 Nástroje pro zlepšení bezpečnosti....23

4.1.3 Rozdíly v postojích a cílech komerčních firem oproti OSS komunitám....25

4.1.3.1 Finanční aspekty....25

4.1.3.2 Srovnání z hlediska životního cyklu softwarových systémů....27

4.1.3.2.1 Plánování a vývoj....27

4.1.3.2.2 Údržba a aktualizace softwaru....29

4.1.3.3 Rozdíl v přístupu k právům a soukromí uživatelů....33

4.2 Nejčastější hrozby a slabiny open source a proprietárního softwaru....36

4.3 Vzorce pro srovnání a měření bezpečnosti....39

5. Databáze bezpečnosti informačních technologií....41

Závěr....43

# Úvod

Tato práce se zabývá otázkou, zda je z hlediska bezpečnosti kvalitnější software s otevřeným kódem či naopak software proprietární. Vychází ze starších dokumentů a vědeckých článků a snaží se zmapovat a poskytnout plný výčet argumentů pro i proti, které se v nich vyskytují. Argumentace je mapována od nejobecnějších principů až po praktické situace a tam, kde je to možné, je dokládána příklady či relevantními statistikami.

Práce začíná snahou o přesné vymezení pojmů, které se ve zbytku práce budou vyskytovat, a o krátké vysvětlení některých technických výrazů. Dále nabízí jak úvod k historii a vývoji obou druhů softwaru, tak i nejdůležitější informace k historii počítačové bezpečnosti. V poslední z úvodních kapitol jsou rozepsané problémy, na něž výzkum na tomto poli naráží.

Hlavní částí práce je rozbor a porovnání těchto typů softwaru. Její první podkapitola se zabývá obecnými argumenty, kde jsou předloženy hlavní důvody pro a proti zveřejňování zdrojových kódu softwaru. Následující podkapitola pak pokračuje rozborem dalších vlivů, jako jsou například peníze na vývoj bezpečného softwaru. Práce pokračuje srovnáním z hlediska životního cyklu softwaru a porovnává rozdíly mezi vývojem a údržbou. A v poslední řadě je zde přihlédnuto k přístupu k samotným uživatelům a jejich soukromí a bezpečnosti.

V další části jsou prostudovány modely pro analýzu a srovnávání bezpečnosti softwarových systémů. A nakonec jsou v poslední kapitole vypsány nejdůležitější databáze pro sběr a uchováván informací o bezpečnosti v oboru informačních technologií.

# 1. Definice a vymezení pojmů

**Zdrojový kód** počítačového programu je definován jako jakákoliv sbírka počítačových instrukcí napsaná lidem srozumitelným programovacím jazykem, obvykle jako text.

**Open source software**, či zkráceně OSS, česky otevřený software, je software s otevřeným, všem dostupným zdrojovým kódem. Dostupný zdrojový kód však není jediným požadavkem, který musí tento software splňovat. Definice všech požadavků na to, aby se software dal označovat za OSS, se více či méně liší stejně jako různé licence, které z nich vycházejí. Existují dvě nejrozšířenější definice termínu OSS. První z nich je ve smyslu svobodného softwaru[1] tak, jak jej definoval Richard Stallman. Od takovéhoto softwaru (nebo tedy licence pod kterou spadá) se očekává, že nám bude podle definice nadace Free Software Foundation[2] dovolovat:

- spustit program za jakýmkoliv účelem
- studovat, jak program pracuje, a přizpůsobit ho svým potřebám

(předpokladem k výše uvedenému je přístup ke zdrojovému kódu)

- redistribuovat kopie, abyste pomohli vašemu kolegovi
- vylepšovat program a zveřejňovat zlepšení, aby z nich mohla mít prospěch celá komunita

(předpokladem k výše uvedenému je přístup ke zdrojovému kódu) (Free Software Foundation)

Druhá definice je od Open Source Iniciative[3] a liší se hlavně tím, že přesně neurčuje, kdo všechno smí originální zdrojový kód upravovat. Rozdíl v těchto definicích znamená hlavně rozdíl v tom, jakým způsobem se přistupuje k vývoji a opravám softwaru, což se ve výsledku může projevit na míře bezpečnosti. První možností je, že kdokoliv může pozměňovat a přispívat ke zdrojovému kódu. Tomuto modelu vývoje softwaru se říká

[1] Richard Stallman považuje termín open source software za nepřesný a preferuje označení *free software* (česky svobodný software). https://www.gnu.org/philosophy/free-software-for-freedom.en.html

[2] Nezisková organizace založená na podporu svobod a práv uživatelů softwaru.

[3] Organizace založená roku 1998 na podporu open source softwaru.

*bazaar*. Druhou možností je model zvaný *cathedral*, kde je proces vývoje a oprav koordinován osobou přezdívanou *wizard.* (Raymond, 1999, str. 9)

Ačkoliv preferovanou definicí open source softwaru pro tuto práci je ta první definovaná Free Software Foundation, jsou zde použity i informace ze studií, v nichž je k významu termínu open source software přistupováno poněkud volněji. Přesné definice jednotlivých licencí a menší rozdíly mezi nimi však nejsou pro tuto práci příliš důležité a je sem možné zahrnout širší řadu rozličných softwarových licencí, stejně tak jako se postupovalo při definování open source softwaru ve vědeckých článcích, z nichž byly čerpány informace pro tuto práci.

V protikladu ke svobodnému softwaru stojí **software proprietární.** I pro něj lze najít více různých označení a definic. Například: „Proprietární software je takový software, jehož zdrojový kód je dostupný pouze autorům softwaru," (Reinke and Saiedian, 2003) nebo „Proprietární software je jakýkoliv druh softwaru, který podléhá autorskému právu a omezuje jeho užití, distribuci a modifikaci uloženou vydavatelem, prodejcem nebo tvůrcem." (Proprietary Software) Pod takovéto definice může spadat mnoho různých licencí, jež se dají označit za proprietární, jako je například shareware, freeware, soukromý software či komerční software. (Free Software Foundation, Kategorie svobodného a nesvobodného softwaru) Pod proprietární software může spadat dokonce i software, jehož zdrojový kód je přístupný, avšak uživatel nemá právo s ním manipulovat či ho měnit.

**Exploit** je program, který zneužívá nebo nějakým způsobem umožňuje zneužití chyby či zranitelnosti v softwaru.

**Backdoor** či zadní vrátka jsou součástí softwaru, jež umožňuje obejít běžnou autentizaci a získat tak neoprávněný přístup k datům či systému.

**Patch** či záplata nebo aktualizace je proces instalace novější verze softwaru z důvodů, jako jsou bezpečnost, oprava chyb či přidání nových funkcí.

**Debugger** je nástroj používaný k hledání chyb ve zdrojovém kódu určitého softwaru.

**Disassembler** je nástroj převádějící strojový kód do člověku srozumitelné podoby.

**Počítačový červ** je škodlivý program, který je schopen šířit své kopie do dalších počítačů.

**Firewall** je zařízení či program sloužící k řízení a zabezpečení provozu sítě.

**Šifrovací klíč** je informace, pomocí které lze transformovat obyčejný text do šifrovaného či naopak.

**Malware** je druh softwaru, jehož účelem je poškození počítačového systému, neoprávněný zisk informací, umožnění přístupu k systému neautorizovaným osobám apod.

**Rootkit** je typ škodlivého programu, který svojí činností skrývá určité procesy před uživatelem a umožňuje tak většinou fungování dalších škodlivých programů.

# 2. Historie vývoje softwaru a softwarové bezpečnosti

Při distribuci prvních softwarových systémů byly zároveň s nimi šířeny i jejich zdrojové kódy. Tyto programy bylo možné dle libosti přizpůsobovat a dále šířit. Postupně se však situace začala měnit a od poloviny sedmdesátých let již převládal software proprietární. Roku 1969 tehdy největší producent softwarových systémů a hardwaru, společnost IBM, změnil způsob, jímž do té doby prodával své produkty, zrušil dosavadní model, v němž prodával hardwarové systémy zároveň se softwarem, a začal software prodávat individuálně, což vedlo ke specializaci softwarového průmyslu a zároveň i k dalšímu zdroji peněz. (Chronological History of IBM) V osmdesátých letech se objevily snahy o opětovné vytváření softwaru, který by bylo možné volně měnit a redistribuovat. Iniciátorem jednoho z takových hnutí byl Richard Stallman, pod jehož vedením vznikl projekt GNU a nadace Free Software Foundation. Díky nárůstu množství počítačových systémů připojených k Internetu bylo možné, aby se našlo dostatečně velké množství jedinců ochotných se podílet na spolupráci v této oblasti. Poté, co Linus Torvalds vytvořil jádro operačního systému pojmenované Linux, jej bylo možné spojit s již dříve vytvořenými svobodnými programy, aby tak vznikl samostatný čistě open-sourcový operační systém GNU/Linux. (Stallman, *GNU Project*) Od té doby se uživatelská báze OSS značně rozrostla. Zatímco ze začátku to byla pouze skupina technicky zdatných nadšenců, kteří se tomu věnovali převážně ve svém volném čase, nyní je OSS využíván četnými malými i velkými firmami, ve vládních složkách, ale například také technickými laiky na jejich osobních počítačích.

S tím, jak se rozšiřovalo používání počítačů a stále větší množství dat bylo pomocí nich zpracováváno, se zvyšovalo i znepokojení z jejich možného zneužití. Na konci šedesátých let dvacátého století se začaly objevovat první studie počítačové bezpečnosti. Protože k počítačovým systémům mělo díky terminálům a podobnému příslušenství přístup stále větší množství lidí i ze vzdálených míst, bylo zjevné, že fyzická izolace počítače již jako primární bezpečnostní opatření nebyla dostačující. A zároveň tak vyvstala otázka ochrany soukromí a dat jedněch uživatelů před jinými. (Ware, 1979) Potřeba ochránit

citlivá data se nejdříve projevila při použití sálových počítačů ve vojenské sféře a v prostředí vládních počítačů, které uchovávaly osobní data o občanech. Pro takové účely vznikaly operační systémy, jež podporovaly oddělení uživatelských účtů podle jejich oprávnění a další víceúrovňová bezpečnostní opatření. Významné bylo také zavedení šifrovacích standardů, jako byl Data Encryption Standard[4], a jejich výsledné rozšíření i do veřejné sféry.

Když se v osmdesátých letech rozmohlo užití osobních počítačů, důraz na bezpečnost ustoupil do pozadí. Používaly se převážně ve firemním prostředí a ačkoliv se na nich pracovalo s komerčně citlivými daty, nejednalo se již o utajené informace jako ve vojenském či vládním prostředí. Osmdesátá léta však byla také obdobím, kdy se mezi počítači začal šířit malware a hrozícího nebezpečí si začala být vědoma i širší veřejnost.

Rok 1988 byl v tomto ohledu jistým způsobem zlomovým, protože byla poprvé jediným škodlivým programem zasažena značná část počítačů. Počítačový červ, který se jako jeden z prvních dokázal rozšířit mezi velký počet počítačů, byl po svém tvůrci pojmenován Morrisův červ. Zneužíval špatných nastavení unixových programů, jako byl sendmail[5], prolamoval slabá hesla apod., díky čemuž se mu podařilo nakazit údajně 5-10% všech počítačů připojených v té době k Internetu. Tento incident ve výsledku vyústil ve snahu se proti takovýmto hrozbám do budoucna efektivně bránit, a to například vytvořením skupiny CERT popsané v kapitole Databáze.

Od začátku devadesátých let byl přístup k Internetu stále snazší a navíc se objevil World Wide Web (zkráceně web), který uživatelům umožňoval publikovat informace a mít k nim rychlý přístup. S masivním nárůstem lidí připojených ke globální síti se však začalo také zvyšovat množství a rozmanitost útoků. Zpočátku takové útoky z velké části nebyly motivovány snahou o finanční zisky. Ale s nárůstem webových obchodů, internetového bankovnictví, osobních informací, jako jsou čísla kreditních karet uložená na serverech, se změnila i podoba a zaměření nyní již dobře organizovaných skupin, kterým jejich nelegální činnost může přinášet značné zisky.

[4] Symetrická šifra, která vznikla v 70. letech.

[5] Program sloužící k přepravě elektronické pošty.

Stejně jako je těžké předvídat, kterým směrem se bude dále rozvíjet technika, je těžké předpovídat, jaká nebezpečí v budoucnu hrozí. Tudíž není snadné odhadnout, jaká bezpečnostní opatření a jakou ochranu by v sobě měly software, elektronika či rozšířené protokoly zahrnovat. Stejně tak je občas složité tyto bezpečnostní prvky zpětně zahrnout do již zavedených aplikací. (History of Computer Security, 2011)

# 3. Problémy při srovnání

## 3.1 Kvalita analýz a jejich spolehlivost v čase

Snahu o objektivní a přesné srovnání komplikuje řada různých faktorů. Počínaje problémy s vymezením obou kategorií softwaru, jak bylo zmíněno výše. Různí autoři a jejich statistiky a porovnání mohou vycházet z odlišných definic open source softwaru, což se v praxi projeví tím, že do svých prací zahrnou nebo naopak nezahrnou určité programy. Což může způsobit zkreslení při snaze o širší analýzu a srovnání většího množství počítačových programů.

Dalším úskalím je fakt, že informační technologie je takovým odvětvím lidské činnosti, které se neustále a velmi rychle vyvíjí. Postupem času se zvyšují nároky na bezpečnost nejen u velkých firem a státních složek, ale i u jednotlivých uživatelů, což se zákonitě projeví také na kvalitě softwaru. S měnící se kvalitou softwaru a se zlepšujícím se zabezpečením se začíná měnit také podoba a síla útoků a nebezpečí, kterým takový software a jeho uživatelé musí čelit. K rozvoji a nejrůznějším změnám dochází také na straně producentů. Následkem všech těchto faktorů jakékoliv statistiky ztrácejí v průběhu času svou hodnotu a nemusí tak odpovídat současné situaci.

Velkým problémem, který výrazně ovlivňuje diskurz na poli softwarové bezpečnosti, jsou síly, jež stojí v pozadí některých studií, a jejich motivace k ovlivňování výsledků ve svůj prospěch. Rozvoj a prodej softwarových produktů je velmi výdělečným oborem, a proto zde existuje silná konkurence. Komerční firmy, jež se věnují tvorbě a distribuci proprietárního softwaru, se snaží přesvědčit svět o tom, že právě jejich software je ten nejlepší. Za tímto účelem sponzorují jiné organizace, aby vypracovaly studie, které to mají dokázat. Za takových podmínek je však těžké se s jistotou spolehnout na nestrannost a objektivitu těchto studií vytvořených na zakázku. Jako příklad může posloužit jednání jedné z největších nadnárodních softwarových společností Microsoft Corporation, které se začal jevit open source software jako nechtěná konkurence, a tak sponzorovala studie, které měly vyvrátit výhody a silné stránky, jež se open source softwaru přisuzovaly, mezi nimiž

byly i předpoklady k lepšímu zajištění bezpečnosti. Mezi takové studie patří například studie Alexis de Tocqueville Institution (Boulanger, 2005, str. 240) nebo analýza zranitelností open-sourcových webových serverů v srovnání s těmi proprietárními vytvořená firmou Security Innovation. (Messmer, 2005)

Obtíže při objektivním širším srovnání může působit rozdíl ve velikosti a frekvenci používání jednotlivých programů. Navíc je to metodologicky a co se dostupnosti úplných a spolehlivých dat týče velmi složitá činnost. Z toho důvodu se mnoho studií omezuje pouze na komparaci dvou či jiného podobně malého množství programů, které mají stejné zaměření a využití. Kvalitní data a pečlivé analýzy se objevují jen zřídka. (Witten, Landwehr and Caloyannides, 2001)

## 3.2 Definice bezpečnosti a hledání správného systému měření bezpečnosti

Snažit se porovnat bezpečnost softwaru je komplikované, protože je zde nekonečné množství možných rizik. (Curtis, 2009) Existuje větší množství představ o tom, co všechno pojem bezpečnost v prostředí informačních technologií zahrnuje, a tudíž v různých pracích věnujících se tomuto tématu se liší i způsob, jakým se termín bezpečnost definuje. Encyklopedia Britannica definuje počítačovou bezpečnost jako ochranu počítačových systémů a informací proti poškození, krádeži a neoprávněnému použití. (Encyclopaedia Britannica, Computer Security) V této práci se však bude k bezpečnosti přistupovat v užším významu, nicméně tento pojem zůstane relativně volný vzhledem k tomu, že jsou zde informace čerpány z různých studií. Většina prací zabývajících se bezpečností softwaru staví míru bezpečnost softwaru do nepřímé úměrnosti s množstvím chyb, jež se v něm vyskytují. Některé studie vnímají bezpečnost jako souhrnný termín pro řadu různých vlivů, jiné výzkumy zase definují bezpečnost třeba také jako riziko, čímž myslí kombinaci pravděpodobnosti úspěšného útoku na daný systém zároveň s výslednými škodami na majetku. (Hoepman and Jacobs, 2007) Důležitost definice a toho, co všechno tento pojem zahrnuje, se projeví hlavně při snaze bezpečnost softwaru co nejpřesněji a nejefektivněji měřit a porovnávat. Tudíž bude zvlášť důležité, co všechno je pod tímto pojmem zahrnuto, hlavně při vytváření modelů pro kvantitativní analýzu.

# 4. Rozbor a srovnání

## 4.1 Pozitivní a negativní důsledky zveřejnění zdrojového kódu

Jedno z hlavních témat diskuze nad tím, který z těchto dvou typů softwaru je bezpečnější, souvisí s otázkou, zda je lepší, aby byly zdrojové kódy zveřejněné, či nikoliv, a s důsledky, které z toho vyplývají.

### 4.1.1 Teoretické principy softwarové bezpečnosti

Snaha o ochranu a patřičné zabezpečení důležitých informací není něčím, co by se objevilo až s příchodem počítačů a rozvojem Internetu. Již v dávné minulosti se o to lidé pokoušeli zvláště ve vojenském prostředí. „Po mnoho století bylo utajování převládající metodologií obklopující projektování jakéhokoliv bezpečného systému. Například bezpečnost vojenských komunikačních systémů byla založena primárně na skutečnosti, že pouze velmi malá skupina věděla, jak tento systém funguje, a ne na tom, že by daný systém komunikace byl ze své podstaty bezpečný.“ (Hoepman and Jacobs, 2007, str. 80) Představa, že je tento přístup efektivní, zůstává živou i v současné době a přiklání se k ní jak tvůrci dnes již výhradně softwarových systémů, tak jejich uživatelé. Tento princip je dnes v akademických kruzích známý jako *security through obscurity*, což lze chápat jako dosažení bezpečnosti pomocí utajení informací o fungování systému před útočníkem. V opozici k tomuto názoru se však objevily námitky, že takové systémy se za bezpečné považovat nedají.

Ve vojenské sféře přišel Holanďan Auguste Kerckhoffs již roku 1883 s tvrzením, že bezpečnost jakéhokoliv vojenského systému nesmí být ohrožena tím, že se k nepříteli dostanou informace o principu jeho fungování. Tudíž takový systém, jehož bezpečnost se po prozrazení toho, jak funguje, sníží, se nedá považovat za bezpečný. To vešlo ve známost jako Kerckhoffsův princip (Hoepman and Jacobs, 2007, str. 80). Stejný názor zastával také slavný matematik Claude Shannon, jenž se ve své době zabýval mimo jiné šifrovacími

systémy. Podle něj i přesto, že je vytvořen naprosto nový systém, o kterém protivník nemá žádné informace, jejichž znalost by mu usnadnila jeho prolomení, je nutné počítat s tím, že na ně eventuálně přijde. (Shannon, 1949, str. 7) A právě open source, jakožto třída softwaru k jehož struktuře a fungování má přístup kdokoliv, by podle těchto předpokladů měl splňovat tento základní princip bezpečnosti. Naopak proprietární software s uzavřenými zdrojovými kódy z toho vychází jako nevyhovující.

V dnešní době je také v softwarovém prostředí důležité rozlišovat mezi designem systému a jeho implementací. (Hoepman and Jacobs, 2007, str. 80) Ačkoliv snaha, aby byl každý softwarový systém bezpečný díky svému návrhu (*secure by design*), tudíž snaha, aby byla zajištěná bezpečnost a odolnost softwaru před různými hrozbami již při jeho vytváření, je mezi producenty softwaru rozšířenou praktikou, není to úplnou zárukou bezpečnosti. Počítačový program může například používat veřejný, prověřený a bezpečný šifrovací standard AES[6], ale kritická chyba v jeho implementaci snadno ohrozí spolehlivost celého systému a může tak poškodit uživatele.

Právě bezpečnostní chyby v softwaru jsou jádrem mnoha debat na téma, která z těchto dvou softwarových filozofií je lepší. Oponenti open source propagují názor, že otevřený zdrojový kód, který umožňuje přístup komukoliv, nahrává těm, jež se možných zranitelností systému snaží využít. Poté, co objeví v programu kritickou zranitelnost, jim nic nebrání, aby jí zneužili například k získání osobních dat uživatelů nebo díky vytvořenému exploitu získali kontrolu nad postiženými systémy. Naopak pokud jsou veřejnosti dostupné pouze binární kódy systému, nelze to, jak daný systém funguje, detailně prozkoumat a tak případně nalézt a využít možné zranitelnosti ve svůj prospěch. (Brown, 2002)

Tento názor se nicméně zakládá na několika předpokladech, které ovšem nemusejí být vždy platnými. Za prvé počítá s tím, že pokud si daný producent softwaru nepřeje, aby byly zdrojové kódy systémů, jež vytváří, zveřejněné, tak se ani nikdy nedostanou do neoprávněných rukou. Zdrojové kódy se mohou dostat ze střeženého prostředí firmy buď k celé veřejnosti, která má možnost najít a zneužít možné chyby, a nebo pouze do rukou

[6] Šifrovací standard Advanced Encryption Standard, který nahradil již nevyhovující DES.

několika zkušených útočníků, kteří případné zranitelnosti nalezené v kódu zneužijí, aniž by si společnost vlastnící licenční práva k danému softwaru či samotní uživatelé vůbec uvědomovali situaci, která nastala. (Hoepman and Jacobs, 2007, str. 82)

Dobrou ukázkou, že i zdrojové kódy proprietárních systémů se mohou snadno dostat na veřejnost, je incident, při kterém se v červenci roku 2003 na internetu objevily volně dostupné zdrojové kódy volebního softwaru použitého ve volbách roku 2002 ve Spojených státech. Po jejich důkladném prozkoumání bylo nalezeno velké množství bezpečnostních pochybení způsobených nedbalostí a nedostatečnými zkušenostmi programátorů, a to i přesto, že producentem byla dlouho zavedená a velká firma Diebold. Volební software této společnosti, jak se ukázalo, nedostatečně chránil anonymitu voličů. Například využíval zastaralého šifrovacího standardu DES, kde byl navíc použit pouze jeden šifrovací klíč pro všechny volební stroje, který byl sám o sobě dostupný přímo v samotném zdrojovém kódu, čehož se dalo snadno využít. Dále zde pak bylo pouze minimální zabezpečení proti neoprávněnému pozměňování výsledku a řada dalších chyb. Některých z těchto pochybení si byla společnost Diebold dokonce vědoma, přesto nepodnikla žádné kroky k jejich napravení. (Rubin, 2006)

Za druhé skutečnost, že zdrojové kódy daného programu nejsou dostupné, vůbec není zárukou, že programátorské chyby a možné zranitelnosti zůstanou skryty před případnými útočníky. „Nástroje jako jsou debuggery a disassemblery totiž dovolují útočníkům nalézt zranitelnosti v aplikacích bez přístupu k původnímu zdrojovému kódu relativně rychle.“ (Hoepman and Jacobs, 2007, str. 82) Použití takových nástrojů je sice z právního hlediska trestné, protože jsou proprietární softwarové systémy zpravidla chráněny autorskými zákony, jako je například DMCA[7], to však jedince, kteří se v nich pokoušejí nalézt zranitelnosti a bezpečnostní chyby, jichž by následně mohli využít k páchání závažnějších trestních činů, jen těžko odradí. Naopak tato ochrana omezuje zájemce, kteří by si chtěli sami prověřit software, jenž používají, bez toho, aby toho chtěli případně jakkoliv zneužít.

[7] Zákon v USA, který se věnuje autorskému právu a hlavně problematice reprodukce a šíření materiálů podléhajících autorským právům a umožňuje jejich kontrolu.

Pokud by se připustilo, že nedostupnost zdrojového kódu neskryje zranitelnosti, a proto není sama o sobě zárukou bezpečnosti, vyvstává otázka, jak se těchto bezpečnostních pochybení v kódu softwarového systému zbavit. Strategií, jež se uplatňuje u open source softwaru, je umožnit přístup ke zdrojovým kódům úplně všem. To má zaručit, že takový softwarový produkt bude podroben pečlivé prohlídce alespoň částí z jeho uživatelů a případné chyby budou nalezeny a buďto nahlášeny, nebo bude dokonce samotným uživatelem vytvořen patch podle toho, jakému modelu vývoje softwaru daný produkt podléhá. Eric S. Raymond to formuloval do citátu „given enough eyeballs, all bugs are shallow," neboli „je-li dost očí, jsou všechny chyby mělké," (Raymond, 1999, str. 9) čímž je míněno, že pro dostatečně velké množství lidí, kteří by kód kontrolovali, nebude složité chyby najít, čili že chyby, jež jsou pro jednoho složité k nalezení, najde jiný snadno. Tento princip pojmenoval Linusův zákon. Teoreticky tak neleží kvalita bezpečnosti softwarového systému pouze na bedrech programátora/programátorky nebo skupiny programátorů, kteří se na vývoji daného systému podíleli a jejich schopnostech, ale prakticky kdokoliv ji může sám za sebe prověřit.

I toto jsou však pouze předpoklady, které v praxi nemusí fungovat stejně snadno jako v teorii. Přístupnost zdrojových kódu softwaru není sama o sobě zárukou, že se najde dostatečné množství subjektů ochotných věnovat svůj čas jejich detailnímu prozkoumání. „Hlášení Joint Information Systems Committee (JISC) uvádí, že pouze méně než 5% ze 130 000 softwarových projektů dostupných na Sourceforge.net[8] je udržováno a také, že více než 90% z nich má pouze jednoho přispěvatele." (Mansfield-Devine, 2008) Autor však v článku také argumentuje, že se nejedná o software, který by byl frekventovaně užíván, a u softwaru, jako je linuxový operační systém, databázový software či aplikace jako OpenOffice[9], je situace značně odlišná.

Přestože se u zanedbávanějšího druhu softwaru najdou jedinci, kteří by měli zájem se danému kódu dostatečně věnovat, není jisté, zda budou vůbec schopni mu porozumět natolik dobře, aby případné chyby či bezpečnostní rizika odhalili. V některých případech

[8] Projekt, který mimo jiné umožňuje přes webové rozhraní distribuci a získávání open-sourcového softwaru. V současné době je zde registrováno 430 000 projektů a stránku využívá údajně až 41.8 miliónů uživatelů. (Sourceforge.net)

[9] Open-sourcový kancelářský software.

musí být člověk kupříkladu expertem nejen v programování a počítačové vědě obecně, ale zároveň by měl mít znalost kryptologie, a lidé s takovými vědomostmi se v bázi uživatelů daného softwaru budou vyskytovat v mnohem menší míře. „Příkladem výše zmíněného je zranitelnost nalezená v některých implementacích programu Secure Shell (SSH) (...). Nalezení této zranitelnosti vyžadovalo nejen znalost samotného SSH protokolu, ale také mnoha pokročilých problémů vztahujících se ke kryptoanalýze a bezpečné implementaci kryptografického softwaru. Pouze velmi málo „očí“ na to mohlo mít dostatečnou kvalifikaci.“ (Payne, 2002, str. 70) Navíc představa, že bude někdo rozumět kódu lépe než skupina programátorů, jež ho vytvořila v první řadě, nezní pro mnoho lidí příliš pravděpodobně.

Robert L. Glass považuje Linusův zákon v podstatě za naprosto neplatný. Podle něj je za prvé hledání některých chyb komplikovanější než jiných a za druhé zvyšování počtu lidí, kteří kód zkoumají, funguje pouze do určité míry. Uvádí zde testy (Boehm and Basili 2001 cited by Glass, 2003), podle kterých množství nalezených chyb i v těch nejlepších případech nepřesáhne 90% a navýšení množství účastníků kontroly nemá od určitého bodu další význam.

Chyby ohrožující bezpečnost softwaru však nemusí být vůbec dobře skryté nebo ke svému odhalení vyžadovat hluboké znalosti z více oborů, aby zůstaly v OSS dlouho bez toho, aby je někdo odstranil, tvrdí odborníci, kteří nezastávají názor, že by byl open source software oproti tomu proprietárnímu zvlášť bezpečnější. Podle některých se lidé používající svobodný software oddávají mylné představě, že se našlo dost lidí, kteří kódy softwaru zkontrolovali bez toho, aby to udělali oni sami, a mají tak falešný pocit bezpečnosti. To je dokládáno na vícero případech, kdy nebyla potencionálně vážná zranitelnost v systému po dlouhou dobu objevena. Například v e-mailovém programu GNU Mailman nebyly chyby nalezeny po tři roky, v autentizačním protokolu Kerberos[10] nebyl objeven *buffer overflow*[11] po celých 10 let a nakonec SENDMAIL SMTP, který se pravidelně potýkal s bezpečnostními problémy. Jeho bezpečnost se dramaticky zlepšila

[10] Implementace síťového autentizačního protokolu od MIT, jehož prioritou je poskytovat bezpečné přihlašování. http://web.mit.edu/kerberos/

[11] Chyba v programu, která má za následek, že program, který zapisuje data do vyrovnávací paměti, je zapíše i do části paměti, ke které by neměl mít přístup.

poté, co společnost vytvořila a začala vydávat komerční verzi tohoto programu. (Payne, 2002, str. 69) Aktuálním příkladem závažné chyby ve frekventovaně užívaném OSS, kterou nikdo neobjevil po dobu dvou let, je takzvaný Heartbleed Bug v programu OpenSSL. Tato chyba umožňovala potencionálnímu útočníkovi přístup k zašifrovaných zprávám, webovým cookies či dokonce k samotným privátním šifrovacím klíčům. (Slížek, 2014) Tento incident znovu rozdmýchal debatu nad bezpečností softwaru s otevřeným kódem. Ačkoliv byl program často používanou a oblíbenou implementací protokolu SSL, nenašel se nikdo, kdo by program zkontroloval, a chybu, jak předpovídá Linusův zákon, našel. Zranitelnost nedokázaly objevit ani programy pro automatizovanou analýzu kódu. Vzhledem k tomu, že projekt OpenSSL měl pouze jednoho stálého placeného vývojáře, objevily se ohlasy, že spoléhat se na dobrovolnou činnost komunity je již příliš riskantní a je třeba větší množství stálých, placených vývojářů.

Dále pak skutečnost, že se díky zpětné vazbě od uživatelů podaří některé z chyb nebo dokonce naprostou většinu chyb v programu nalézt, stojí v nepoměru k faktu, že potencionálnímu útočníkovi stačí najít jednu jedinou zranitelnost, aby software zkompromitoval. A najít všechny chyby a možné zranitelnosti je vzhledem k velikosti, jaké může dosáhnout například operační systém, nepravděpodobné až nemožné. Backdoor, jenž se může skrývat v pouhých dvou řádcích kódu (Felten, 2013) Linuxového kernelu[12], snadno přehlédne i zkušený programátor, pokud celý zdrojový kód dosahuje rozměrů až patnácti miliónů řádků. (Paul, 2012) Společnost Reasoning se v roce 2003 věnovala analýze implementace internetového protokolu TCP/IP v Linuxovém kernelu a v pěti proprietárních operačních systémech. V testech, kde se poměřovala kvalita kódu a množství chyb, z toho vyšel s nejlepším výsledkem Linux, v jehož implementaci zmíněného protokolu se vyskytovalo 0,1 chyb na 1000 řádků kódu, zatímco proprietární systémy měly 0,55 chyb na 1000 řádků. V rámci svobodné databázové aplikace MySql byly naměřené hodnoty dokonce šestkrát lepší než u jeho proprietárních protějšků. Pouze hodnoty naměřené pro Apache web server[13] se pohybovaly v průměrných hodnotách pro daný typ softwaru.

[12] Základní část operačního systému GNU/Linux, která alokuje prostředky počítače ostatním programům. (Stallman, Linux a systém GNU)

[13] Open-sourcový multiplatformní HTTP server vedený Apache Software Foundation. http://httpd.apache.org/

(Boulanger, 2005, str. 244) Výsledky studie společnosti Coverity, jež zanalyzovala doposud největší množství softwarového kódu (pouze software psaný v programovacím jazyce C/C++[14]) ukazují, že v roce 2013 obsahoval OSS výrazně menší množství chyb než ten proprietární. Navíc průměrné množství chyb, jež se v OSS vyskytuje, od minulého roku pokleslo a podle slov studie tak OSS „...zvyšuje laťku toho, co se dá považovat za kvalitní software pro celý průmysl“. Z jejich průzkumu také vyplynulo, že stále větší množství společností OSS používá a nebo to má do budoucna v plánu. (Coverity, 2014)

Tabulka 1: Srovnání open source a proprietárního C/C++ kódu v roce 2013

| Velikost programu (počet řádků kódu) | Open-sourcový kód | Proprietární kód |
|---|---|---|
| Řádky kódu | 252 010 313 | 684 318 640 |
| Množství projektů | 741 | 493 |
| Průměrná velikost projektu (v počtu řádků kódu) | 340 094 | 1 388 070 |
| Význačné chyby k 12/31/13 | 149 597 | 492 578 |
| Chyby opravené v roce 2013 | 44 641 | 783 799 |
| Hustota chyb | 0,59 | 0,72 |

Zdroj: (Coverity, 2014)

Za předpokladu, že velikost softwaru má přímý vliv na jeho bezpečnost (čím více kódu, tím větší pravděpodobnost výskytu zranitelnosti), mají uživatelé OSS přímou možnost ovlivnit komplexitu jimi užívaného systému odstraněním nepotřebných částí kódu a potencionálně tak zvýšit jeho bezpečnost. (Hoepman and Jacobs, 2007, str. 83) Díky možnosti svobodné programy pozměňovat k vlastní potřebě a dále vydávat tuto

[14] C (a jeho rozšířená novější verze C++) je rozšířený programovací jazyk.

pozměněnou kopii je možné nadbytečné prvky odstranit a vydávat tuto pozměněnou verzi jako alternativu původního produktu. Vznikne tak řada lehce odlišných programů, které se potřebám uživatele přizpůsobí lépe než mnohem méně flexibilní komerční produkt s mnoha funkcemi. Například GNU/Linux je dostupný pro více různých architektur procesoru na rozdíl od operačního systému Windows, jehož portování na méně populární architektury by bylo příliš finančně nákladné. (Prieto) Komerčním firmám by ale na druhou stranu přílišná roztříštěnost jejich programu do mnoha verzí a jejich nekontrolovatelná transformace a šíření mohly působit problémy. Pro firmy by bylo komplikované mít zodpovědnost za různé kopie svého programu v oběhu, jež jiní lidé pozměnili. Funkce, bezpečnost a správa kvality takového programu by již nebyla plně v jejich rukou. (Reinke and Saiedian, 2003)

Je také nutné vzít v potaz, že naprostá většina každodenně používaných OSS programů je dostupná nejen v podobě zdrojových kódu vyžadujících naši námahu s jejich kompilací[15], ale pro praktičtější a rychlejší použití jsou programy dostupné ke stažení i v binární verzi. Dříve bylo stahování zdrojových kódu a vlastní kompilace programu celkem časté buď z důvodů úspory stahovaných dat, nebo kvůli přizpůsobení programu pro lepší výkon na specifické verzi operačního systému. V současné době se ale díky zrychlení internetové sítě, rychlému nárůstu výpočetní síly počítačů a rozšíření uživatelské báze i mimo pokročilé a zkušené uživatele znatelně snížilo procento lidí, které skutečně program získává jinak než stažením jeho dostupné binární verze. Jaká je ale záruka, že dostupná binární verze skutečně odpovídá publikovaným zdrojovým kódům onoho programu a není tam navíc ukrytý backdoor, o kterém nemusí mít ponětí ani sám poskytovatel softwaru, pokud je to například následkem předchozího úspěšného útoku na jeho server a webové stránky, přes něž je program nabízen ke stažení. (Levy, 2000) Dokonce není vůbec nutné, aby musel případný útočník, pokud chce šířit pozměněný software, vynakládat úsilí k získání přístupu k oficiálním stránkám distributora OSS. Veřejná přístupnost zdrojového kódu některých populárních programů a možnost, aby byl kýmkoliv pozměňován, má i negativní efekty. Stále častěji se objevují stránky, z nichž je možné si stáhnout jisté druhy populárních softwarových programů, které ovšem kromě původních funkcí navíc obsahují malware. A ačkoliv je možné tomu předcházet například tím, že uživatel omezí místa, z

[15] Transformace příkazů napsaných ve vyšším programovacím jazyce do strojového kódu.

nichž software získává, pouze na zdroje původního producenta či porovná hash[16] staženého programu s tím, jenž bývá dostupný na stránkách původního producenta, ne všichni uživatelé jsou natolik technicky zkušení či zdravě nedůvěřiví, aby se tím řídili. (Proffitt, 2013)

Ve své práci publikované roku 1984 ukazuje Ken Thompson, jak je možné, aby i software vytvořený z čistých zdrojových kódů obsahoval backdoor. Pokud se vytvoří kompilátor, který bude do jím zkompilovaných programů vkládat backdoor, bude tak možné zkompromitovat i jinak bezpečný software. A navíc pokud by do něj takový kompilátor při kompilaci dalšího stejného programu (kompilátoru) vložil kód, který by rovněž umožňoval vkládání backdoorů, snadno by se šířil a byl by navíc skoro nezjistitelný. (Thompsons, 1984)

I přesto, že je zdrojový kód uzavřený a uživatelům daného softwaru nepřístupný, je možné, aby byl posouzen nezávislou třetí stranou, a to podle obecně přijímané metodologie, jako je například mezinárodní standard pro certifikaci počítačové bezpečnosti Common Criteria. Takové ohodnocení by mělo zaručovat, že testovaný software splňuje jisté zásady pro zaručení bezpečnosti systému a svých uživatelů. Testování softwaru je ale příliš nákladné, a tak společnosti většinou z úsporných důvodů nedávají příliš často své produkty testovat. Tyto testy jsou navíc omezené pouze na určité způsoby použití a parametry jsou nastaveny tak, že nemusí odpovídat prostředí, ve kterém bude systém výsledně používán. Druhou nevýhodou takového ohodnocení je, že se vztahuje pouze na specifickou verzi softwaru a pro novější verze je nutné tuto evaluaci opakovat. (Hoepman and Jacobs, 2007)

### 4.1.2 Nástroje pro zlepšení bezpečnosti

Obecně ale OSS nabízí jednotlivcům i firmám mnohem více možností jak pro ověření, tak i zajištění lepší bezpečnosti softwarových produktů, jež se rozhodnou využívat.

[16] Krátký řetězec znaků vytvořený hašovací funkcí. Pokud budou na softwarové aplikaci provedeny nějaké změny, projeví se to i tím, že hash původního souboru nebude odpovídat tomu ze souboru nového.

Pokud je například dostupný zdrojový kód softwaru, je možné využít k posílení bezpečnosti řady dodatečných nástrojů a programů. Mezi ně patří kupříkladu statické analyzátory kódu, jež prozkoumají kód a ohlásí všechny podezřelé či bezpečnostně slabé části kódu, které by mohly představovat riziko. Na možné zranitelnosti, jež jsou při statické analýze nezjistitelné, je možné využít nástroje na dynamickou analýzu, které kontrolují chování programu při jeho průběhu a pod zátěží. Dají se ale použít i nástroje zmírňující riziko a zlepšující bezpečnost programu rovnou při jeho kompilaci. (Cowan, 2003) Pro testování bezpečnosti a spolehlivosti softwaru je také možné použít nástroje pro takzvanou *fuzzy analýzu*, při které jsou softwarovým systémům místo obvyklých vstupních dat předkládána data náhodně generovaná. Následně je sledováno, zda nenastanou nějaké neočekávané reakce systému.

Na základě rostoucího úspěchu systému GNU/Linux jako svobodného operačního systému se NSA rozhodlo pomoci dále vylepšit jeho bezpečnost a ve spolupráci s dalšími subjekty vytvořila modul SELinux (Security-Enhanced Linux). Jedná se o kernelový modul dovolující mandatorní řízení přístupu, což umožňuje dohlížet nad uživateli a řídit, k jakým procesům mají subjekty využívající systém přístup. NSA to považuje za další krok k tomu, aby se mohl Linux stát populárním a zároveň bezpečným operačním systémem.

Bezpečnost a dostatečně velká možnost kontroly je pro mnoho institucí a společností jedním z rozhodujících faktorů pro výběr softwaru. Oproti proprietárnímu softwaru, který je stále frekventovaně využíván v komerčním sektoru, je použití OSS preferováno převážně na vědeckých pracovištích. Například NASA, která zprvu využívala operační systém Windows, postupně přešla na linuxové operační systémy. Jedním z důvodů byl také zájem o zvýšení bezpečnosti poté, co byl v notebooku jednoho z kosmonautů zavezen na vesmírnou stanici ISS počítačový virus, který se následně rozšířil i na ostatní osobní počítače. (Anthony, 2013) Problémy s funkčností a bezpečností softwaru jsou dostatečným důvodem dokonce i pro knihovny, aby zvážily přechod k OSS. Ačkoliv hlavním důvodem knihoven pro přechod k OSS většinou bývají buď finanční zájmy, nebo snaha o zlepšení poskytovaných služeb. Menší studie přechodu čtyř amerických knihoven na OSS ukázala, že dvě z nich (Howard County Library a University of North Carolina, Chapel Hill) k tomu přiměly přetrvávající problémy s bezpečností. Přechod na OSS vedl k

vyřešení těchto problémů, a žádné další komplikace v tomto směru již nenastávaly (s výjimkou fyzického vandalismu a jednoho incidentu, při němž se uživateli podařilo dostat do příkazové řádky, neměl však příslušná administrátorská práva k tomu, aby se mu podařilo napáchat nějakou škodu). (Houser, 2009)

### 4.1.3 Rozdíly v postojích a cílech komerčních firem oproti OSS komunitám

Na bezpečnost softwarových systémů má neoddiskutovatelně vliv i chování a praktiky jejich tvůrců a producentů, i když je jejich hodnocení komplikovanější, než hledání obecných principů v předchozí kapitole.

#### 4.1.3.1 Finanční aspekty

„Zveřejnění zdrojového kódu systému však není pouze technickou záležitostí, má to i ekonomické a další efekty, které musíme též uvážit.“ (Witten, Landwehr and Caloyannides, 2001) Je nutné mít na paměti, že na předním místě v důvodech k nedostupnosti zdrojových kódů u komerčních softwarových systému stojí vidina peněžního zisku popřípadě ztráty u společností, jež je produkují a distribuují. Například hrubý zisk společnosti Microsoft Corporation za rok 2013 dosahoval hodnoty 77 849 000 000 USD (Helland, 2013) a je tudíž možné předpokládat, že si své zisky bude snažit co nejlépe ochránit. Takové společnosti se domnívají, že pokud by byl možný přístup ke zdrojovým kódům, usnadnilo by to práci kriminálníkům, kteří by jej chtěli nelegálně užívat bez placení, nebo by jejich konkurence mohla využít částí kódu ke svému prospěchu.

Finanční hledisko se ovšem neprojevuje pouze ve snaze uchránit zdrojové kódy, ale i na procesu výroby a udržování softwarového produktu. Snaha získat peníze je hlavní hnací silou komerčních společností, které tomu přizpůsobují i strategie výroby a podpory svých produktů. „Jakmile je trh nasycen počátečním produktem, přesouvá se zájem producenta na vytváření zlepšení. Protože software není konzumován nebo opotřebováván jeho opakovaným používáním, základem pro budoucí odbyt je vydávání nových verzí

původního produktu. V takovém procesu je těžké prodávat bezpečnost, protože oproti funkcím má tendenci být neviditelná při normálním použití a viditelná pouze až se objeví bezpečnostní potíže.“ (Witten, Landwehr and Caloyannides, 2001) Navíc takové množství zaměstnanců věnujících se hledání chyb před vydáním nové verze softwaru jako existuje u populárních open-sourcových projektů by se firmám z finančního hlediska nikdy nemohlo vyplatit. Producenti proprietárního softwaru mají také větší motivaci k tomu, aby se snažili chyby ve svých softwarových systémech skrývat, neboť jejich odhalení sebou přináší i negativní publicitu pro ně a produkty, které prodávají. A negativní publicita může vést ke ztrátě současných i potencionálních zákazníků. Vzhledem k tomu, že je zdrojový kód OSS přístupný a s ním i všechny možné chyby a zranitelnosti, jež obsahuje, nemají je jeho autoři jak skrývat. (Payne, 2002, str. 75)

U starší verze softwaru může taková komerční firma časem přestat vydávat bezpečnostní aktualizace a patche pro zachování bezpečnostní integrity systému a uživatelé si je nebudou moci sami vytvořit, protože nemají přístup ke zdrojovým kódům. Aktuálním příkladem je ukončení podpory operačního systému Windows XP firmou Microsoft Corporation. Pro operační systém, jež je stále používán v mnoha domácnostech, firmách, ale například i v bankomatech, již nebudou nadále poskytovány žádné bezpečnostní aktualizace. A tudíž i přesto, že zde existuje velká skupina lidí, která by měla silný zájem o možnost tento produkt dál spravovat, není to možné, pokud jeho producent Microsoft zdrojové kódy k tomuto operačnímu systému nezveřejní. Pro Microsoft by to však představovalo finanční ztráty ze zákazníků, kteří budou jinak nuceni přejít na novější verze operačního systému Windows. (Microsoft, Support for Windows XP has ended)

Fungování trhu s OSS je komplikovanější. Přestože stála u vzniku OSS hlavně nekomerční motivace a ideály jedinců, v současné době se vytváření a udržování takového softwaru věnuje stále větší množství komerčních firem. Díky takovéto spolupráci se například sníží náklady, které by pro jednu společnost byly příliš vysoké. Nicméně i velké společnosti, jako již zmiňovaný Microsoft, ale i mnoho dalších, se v poslední době aktivně podílejí na tvorbě a finančním sponzorování OSS. Jejich spolupráce a podpora vývojářů OSS má pozitivní vliv jak na OSS produkty samotné, tak i na vzájemnou kompatibilitu s

proprietárním softwarem. (Microsoft, About Microsoft Openness) Výhoda je v tom, že pokud se jedná například o open-sourcové softwarové knihovny[17], může odstranění bezpečnostních chyb v nich vést ke zlepšení bezpečnosti i proprietárních produktů, ve kterých jsou zahrnuty. Dalším důvodem je, že společnosti a jedinci, kteří se podílejí na jeho bezplatném vytváření a distribuci, získávají finanční zdroje prodejem svých služeb spojených s expertní znalostí daných systémů. S tím však souvisí i jednání společností poskytujících služby na poli softwarové bezpečnosti, které se podle některých snaží pouze o to, aby jejich klient maximalizoval výdaje na bezpečnost. (Myth Vs. Reality, 2006) To se však může teoreticky týkat podpory obou druhů softwaru.

Záleží na uvážení každé firmy, zda by výše nákladů spojených se zavedením a užíváním open source softwaru za účelem zesílení schopnosti odolávat úmyslným i neúmyslným hrozbám nepřekročily možné finanční škody, jež by jí taková situace mohla teoreticky způsobit.

#### 4.1.3.2 Srovnání z hlediska životního cyklu softwarových systémů

Nepopiratelný podíl na kvalitě softwaru má způsob jakým se přistupuje k jeho vývoji a zlepšování. A ačkoliv spolu OSS a proprietární software sdílejí relativně podobné postupy, ony rozdíly mohou mít silný vliv na výslednou bezpečnost programů.

##### 4.1.3.2.1 Plánování a vývoj

Proces vytváření nového proprietárního softwarového systému podléhá většinou vodopádovému modelu vývoje. Princip tohoto modelu tkví v postupu přes pět po sobě jdoucích fází s tím, že se projekt může vrátit o jednu nebo více fází dozadu podle toho, jak se budou postupně objevovat nové problémy a požadavky. Na systému pracuje placený tým, který podléhá vnitřní hierarchii firmy a musí plnit například předpokládané termíny

[17] Soubor funkcí a zdrojů s definovaným rozhraním, který může být sdílen a využívám více různými programy.

vydání programu. Zpětná vazba zde existuje pouze od samotných pracovníků týmu. Ačkoliv vývoj OSS jisté podobnosti sdílí, je mnohem méně strukturovaný, postupy práce nejsou natolik formální a hlavně zde navíc existuje zpětná vazba i od spotřebitelů a uživatelů softwaru. A díky tomu, že mají přístup ke zdrojovému kódu, se mohou na tomto procesu svými připomínkami a návrhy podílet přímo. (Boulanger, 2005, str. 245) Ačkoliv je koordinace mezi vývojáři velmi složitá a komplikuje se s nárůstem jejich počtu, u debuggování tento problém nenastává. Je tak možné, aby kontrola kódu probíhala paralelně a nenavyšovala tak dobu trvání a cenu vývoje. Navíc lidé, kteří se na tom podílejí, jsou většinou samotným softwarem hluboce zaujatí a snaží se nalézt a vyřešit možné problémy. Odměnou jim je možnost sledovat rychlý vývoj softwaru, mít potěšení ze svých vlastních příspěvků a v neposlední řadě z možnosti používat takto prověřený software. Nejsou to však lidé, kteří by se k OSS projektům přidali náhodně, většinou daný software po nějakou dobu užívali a nakonec se z rozličných důvodů rozhodli podílet také na jeho vývoji. Aby se nový člen dostal do jakéhosi jádra předních programátorů, musí nejdříve prokázat své schopnosti a spolehlivost tím, že přispívá svou prací k zlepšování softwaru i jeho komunity. Takové budování reputace se podobá modelu akademického výzkumu. (Hansen, Köhntopp and Pfitzmann, 2002)

Existují dva primární modely vývoje OSS, z nichž každý má své výhody. Oba modely sice komukoliv umožňují se podílet na vytváření softwaru, ale jejich metody se rozcházejí v tom, jak se s příspěvky od komunity nakládá. V prvním modelu, který je více centralizovaný, zvaném *cathedral,* se vyskytuje osoba označená jako *wizard,* která rozhoduje o všech změnách v softwaru. Druhý model, jenž více odpovídá filozofii otevřenosti komunit podílejících se na budování OSS, se nazývá *bazaar*. Tento model umožňuje prakticky každému přispět do vývoje, a to i ve velmi raných fázích tvorby softwaru. Eric. S. Raymond uvádí jako nejlepší příklad této strategie Linuxový kernel, k jehož vývoji byla naprosto zásadní velká a dobře informovaná uživatelská báze, jež měla okamžitý přístup k novým pozměněným vydáním kernelu. (Raymond, 1999, str. 9)

Je každopádně důležité, aby byl uživatelům umožněn přístup k softwaru už ve fázi jeho vývoje, a tudíž měli šanci se zapojit do hledání a opravování chyb ještě předtím, něž bude takový softwarový systém uveden do provozu. Objev závažné bezpečnostní chyby v

době, kdy se ještě daný software aktivně nepoužívá, nepřinese potencionálnímu útočníkovi prakticky žádnou výhodu a naopak pomůže předcházet situacím, ve kterých by bylo potřeba rychlého vytváření bezpečnostních patchů na ochranu softwaru, který je již v provozu.

Co znatelně ovlivňuje úroveň bezpečnosti softwarových produktů k horšímu, je soustředění se na účelové části systému a zavádění nových funkcí na úkor dostatečného dohledu nad bezpečností. Jedná se však o pochybení, jehož se dopouštějí jak tvůrci svobodného softwaru, (Messmer, 2013) tak i producenti softwaru proprietárního. (Myth Vs. Reality, 2006)

Skutečnost, že každý člověk může nahlédnout do kódu OSS a vidět, jak kvalitně je napsaný, pobízí vývojářské komunity k větší pečlivosti. „Jinak by společnosti a jednotliví programátoři ztratili respekt a důvěryhodnost. A to jako vedlejší účinek bude stimulovat výzkum a rozvoj nových, vylepšených nástrojů pro vývoj softwaru, testování, hodnocení a možná i jeho ověřování.“ (Hoepman and Jacobs, 2007, str. 83)

#### 4.1.3.2.2 Údržba a aktualizace softwaru

Vytvořením a publikací programu však práce na něm nekončí. Software po svém vytvoření nikdy není dokonalý a často obsahuje množství více či méně závažných chyb, které je potřeba najít a následně opravit. A to samozřejmě vyžaduje někoho, kdo se tomu bude věnovat a software udržovat. „Stoupenci proprietárního softwaru, včetně firmy Microsoft Corporation, tvrdí, že když není nikdo za open source zodpovědný, neexistuje způsob, jak zjistit, zda k opravám skutečně došlo.“ (Prieto) Ve skutečnosti fakt, že za softwarový systém není nikdo zodpovědný, ještě neznamená, že se nenajde nikdo, kdo by potřebnou aktualizaci vytvořil. Pokud by se však nikdo takový nenašel, byl by to pro většinu uživatelů problém, neboť má málokdo z nich potřebné znalosti k tomu, aby chybu sám opravil. Z právnického pohledu na věc zde neexistuje nikdo, kdo by se dal za případné komplikace, jimž by společnost musela čelit z důvodu špatně napsaného softwaru, zažalovat. (Messmer, 2005) Je sice pravda, že většinou není možné právní cestou

vyžadovat od autorů OSS náhradu případných ztrát (pokud se nejedná například o OSS se zaplacenou podporou od firmy, jež ho vydává), ale je pravidlem, že jména autorů programů jsou známá a vzhledem k tomu, že kvalita jimi spravovaného softwaru odráží jejich důvěryhodnost a schopnosti, jsou více motivováni, aby se o něj starali co nejzodpovědněji. Ale i u softwaru, který nemá přímého autora, jenž by se dal kontaktovat, se dá často obecně nalézt pomoc v open-sourcových komunitách dostupných na webu a IRC serverech.

Nalezení bezpečnostní chyby v softwarovém systému je ovšem pouze prvním krokem k vyřešení problému. Dalším krokem je včasné vytvoření patche a v poslední řadě jeho aplikace na již používané instance daného softwaru. Je důležité, aby doba mezi objevením zranitelnosti a veřejným zpřístupněním patche na její opravu byla co nejkratší, protože po celý tento časový úsek je software velmi zranitelný. Záleží samozřejmě i na tom, zda byla chyba objevena jedním či více útočníky, kteří ji už zneužívají, a nebo byla zatím objevena pouze subjektem zainteresovaným spíše na jejím opravení než jejím zneužití. Nicméně to, že si producent či veřejnost není zranitelnosti v softwarovém systému vědoma, neznamená, že si toho nemůže být vědom někdo jiný. Útok, jenž takovéto producentům softwaru doposud neznámé zranitelnosti využívá, se nazývá *zero day attack*. Obzvláště ohrožené jsou pak programy, které poskytují služby přes Internet, jako například e-mailové servery apod., které pro své správné fungování musí být neustále dostupné na veřejné síti. Navíc je útokem na server, který využívá mnoho klientů, možné získat velké množství osobních informací či způsobit provozovateli finanční škody jeho vyřazením. A pokud by vyšlo najevo, že pro jeden z těchto dvou typů softwaru jsou vytvářeny a publikovány kritické bezpečnostní patche rychleji než u jeho konkurence, znamenalo by to, že má tento software značnou bezpečnostní výhodu. Open source software má v tomto ohledu podle argumentů zmiňovaných v předchozích kapitolách tři výhody. Za prvé pokud uživatelé softwaru rozumí zranitelnosti a zdrojovému kódu, jsou schopni nalézt a opravit problém sami. Za druhé existuje teoreticky větší množství lidí, kteří mohou pomoci s hledáním problému a potencionálně zkrátit čas potřebný k jeho nalezení, a nakonec pokud někdo jiný než samotný producent je schopen najít řešení, může jej publikovat buď jako patch, nebo instrukce, jak zdrojový kód opravit. (Reinke and Saiedian, 2003)

Je nutné brát ohled i na to, že každá změna v kódu softwarového systému může potencionálně uvést nové zranitelnosti. Například s tím, jak se počítačový program postupně vyvíjí a přibývají v něm nové funkce. S každou takovou změnou se mohou objevit další chyby, jež je nutné najít, a je pak potřeba vydat patch na jejich opravu. Dokonce i v něm se však mohou objevit další chyby. Žádný software tak není nikdy úplně dokonalý a bezpečný, ale všechna rizika se dají jistě znatelně snížit, pokud je tomu věnováno dostatečné množství zdrojů a pozornosti. (Boulanger, 2005, str. 243)

Díky existenci databází registrujících softwarové chyby, zranitelnosti softwarových systémů, ale i toho, zda byly tyto zranitelnosti opraveny, je možné vytvoření statistik, jež nám při své správné interpretaci mohou pomoci odpovědět, který z těchto dvou druhů softwaru má v tomto směru navrch. Roku 2003 John Reinke a Hossein Saiedian vytvořili a publikovali srovnání OSS a proprietárního softwaru na základě dat získaných z BugTraq mailing list, CERT Coordination Center a Incidents.org. Rozdělili porovnávaný software na vícenásobné implementace[18] (24%), open source software (18%), proprietární software (56%) a na non-software (2%), jež svojí podstatou nespadal do softwarového prostředí. Následně porovnali dobu, kdy se objevilo oznámení o zranitelnosti softwaru, zda těchto zranitelností bylo více v jenom hlášení, zda se tato hlášení musela z nějakého důvodu opakovat a nakonec dobu, za kterou se objevilo řešení pro oznámenou zranitelnost. Kromě srovnání, který typ softwaru poskytuje bezpečnostní aktualizace rychleji, výsledky analýzy také ukázaly, že u proprietárního softwaru se vyskytovalo mnohem větší množství opakovaných hlášení o problému navazujících na hlášení předchozí. Což může znamenat buď to, že u administrátorů takového systému je menší pravděpodobnost, že aktualizaci nainstalují poté, co se objeví, že se tento software často vyskytuje v prostředí, kde je mnoho zranitelností, že jsou tyto zranitelnosti velmi závažné, že OSS není natolik používaný, aby stálo za to tato hlášení opakovat a nebo že administrátoři OSS častěji aktualizují svůj software. Dále vyšlo najevo, že procentuálně se množství hlášení, jež obsahovala více zranitelností, vyskytovalo u OSS, což může znamenat, že producenti OSS čekají delší dobu než nasbírají dostatek oprav na vytvoření nové aktualizace (například z důvodu menších zdrojů, než mají komerční firmy), a nebo že publikují změny přes

[18] Chyba, jenž se vyskytovala ve stejném typu softwaru od různých producentů.

automaticky kompilované verze programu z poslední verze zdrojového kódu v CVS[19] (*nightly builds*). U proprietárního softwaru se objevilo mnohem větší množství hlášení o problému, u nichž se nakonec neobjevilo žádné řešení. Primárním účelem této studie bylo zjistit, u kterého typu softwaru trvá vydání bezpečnostních aktualizací kratší dobu, a v tomto směru měl navrch jednoznačně OSS, jehož bezpečnostní opravy byly na základě získaných dat dostupné o 23 dnů dříve než u jeho protějšku. V této studii se však nebraly ohledy na závažnost bezpečnostních chyb v poměru k době mezi zjištěním chyby a její opravou. (Reinke and Saiedian, 2003)

Tabulka 2: Srovnání výsledků analýzy

| | PSS | OSS | Celkem |
|---|---|---|---|
| Celková hlášení | 54 | 17 | 96 |
| Celková hlášení, bez Prev* a MV** | 40 | 13 | 75 |
| Opravená hlášení | 33 | 12 | 45 |
| Procento opravených | 61% | 71% | 47% |
| Procento opravených, bez Prev* a MV** | 83% | 92% | 60% |
| Průměrný počet potřebných dní pro nalezení řešení | 52,45 | 28,75 | 46.,3 |
| Standardní odchylka dnů potřebných pro nalezení řešení | 30,25 | 23,82 | 30,33 |
| Minimální počet dní potřebných k nalezení řešení | 31 | -33 | -33 |
| Maximální počet dní potřebných k nalezení řešení | 198 | 45 | 198 |

* Opakované hlášení

** Více zranitelností v jednom hlášení

Zdroj: (Reinke and Saiedian, 2003)

[19] Concurrent Version System je program pro kontrolu a uchovávání provedených změn při tvorbě softwaru.

V podobných analýzách je rovněž důležité zvážit, kolika lidmi jsou jednotlivé softwarové systémy využívány a o jak závažné zranitelnosti se jedná. I zde se však objevují analýzy, jež v ohledu rychlosti poskytovaných bezpečnostních oprav hodnotí lépe proprietární software. Ve svém článku Open Source Vs. Windows: Security Debate Rages cituje jeho autor E. Messmer studii firmy Security Inovation, která uvádí, že v roce 2004 se u linuxové distribuce Red Hat vyskytovalo dvojnásobné množství bezpečnostních chyb než u operačního systému Windows Server 2003 a opravit tyto chyby jim také zabralo v průměru dvojnásobek času. I zde však nebyla zjišťována závažnost těchto chyb, a tak mohl inženýr systému Red Hat argumentovat, že z těchto chyb bylo těch závažných menší množství a že firma reagovala co nejdříve primárně na kritické bezpečnostní chyby. (Messmer, 2005)

#### 4.1.3.3 Rozdíl v přístupu k právům a soukromí uživatelů

Velmi důležitou roli v této analýze zaujímá i míra důvěryhodnosti softwaru. Jak daný softwarový systém přistupuje k osobním informacím uživatele a nakládá s nimi, je neméně podstatné než jeho zabezpečení proti vnějším útokům. A důvěryhodnost softwaru může být skutečně ověřena a potvrzena pouze odkrytím jeho zdrojových kódů veřejnosti. Pouze díky tomu, že bude mít jedinec přístup k fungování softwaru, který používá, má záruku ochrany svého soukromí a privátních dat. (Hansen, Köhntopp and Pfitzmann, 2002) Stejně jako můžeme ověřit, zda neobsahuje nějaké vážné bezpečnostní chyby, můžeme u softwaru, u kterého je nám přístupný jeho zdrojový kód, prověřit i to, zda mezi jeho funkce patří jen to, co má, a nezahrnuje i nějaké procesy, jež by mohly narušovat soukromí jeho uživatelů. I zde samozřejmě platí, že aby probíhala nějaká kontrola ze strany uživatelů, je u OSS potřeba zájemců s potřebnými technickými znalostmi, avšak skutečnost, že je to možné, je důležitá sama o sobě. U proprietárního softwaru se můžeme spolehnout pouze na tvrzení jeho producenta, že je jeho software v ohledu zachování soukromí svého uživatele v pořádku.

Existuje mnoho důvodů, proč by měl někdo potřebu snažit se získat přístup k soukromým informacím uživatelů. Jednou ze stran, jež o to již dlouhodobě usilují, jsou různé vládní bezpečnostní organizace. Ačkoliv dříve to byly pouze dohady zakládající se spíše na paranoii a strachu z toho, že taková možnost vzhledem k nedostupným zdrojovým kódům k populárním softwarovým systémům může existovat, teprve v relativně nedávné době za přispění Edwarda Snowdena[20] pronikly na veřejnost dokumenty, které potvrzují snahy vlivné americké agentury NSA (National Security Agency) o domluvě s producenty softwaru na umístění skrytého backdooru v jejich softwarových systémech, a to primárně v těch kryptografických. To vedlo ke všeobecné panice a vzniku silné nedůvěry k amerických softwarovým systémům. (Messmer, 2013) Taková nejistota ze strany uživatelů pak ve výsledku poškodila všechny producenty proprietárních softwarových systémů, včetně těch, jejichž software je skutečně bezpečný a spolehlivý. Samotná firma Microsoft byla z takovéto spolupráce s vládními agenturami dlouho dobu podezírána, ale stále odmítala, že by její software backdoor obsahoval, a tvrdila, že by to nikdy nepřipustila. (No Backdoor in Vista, Microsoft Promises, 2006) Až s odhalením Snowdenových dokumentů vyšlo najevo, že Microsoft skutečně s vládními agenturami spolupracuje (Greenwald et al, 2013) a potvrdila se také skutečnost, že se uživatelé při posuzování bezpečnosti a spolehlivosti softwaru, jenž používají, nemohou spolehnout pouze na čestné prohlášení jeho producenta.

Ale i samotné firmy vytvářející software mohou mít zájem na získání z uživatelova počítače více informací, než by se mu mohlo zamlouvat, i bez vnějšího nátlaku od vládních autorit. A když jsou zájmy společnosti přednější než bezpečnost zákazníků, může to mít špatný dopad. Nechvalně proslulým je případ postupu společnosti Sonny BMG ve snaze zabránit uživatelům vytvářet digitální kopie jejich CD a stažených hudebních alb. Při puštění jimi vydaných hudebních CD nosičů či stažené hudby se do uživatelova počítače bez jeho vědomí zároveň nainstaloval software, díky kterému se při kopírování jejich CD místo hudby objevoval pouze statický šum. Dále pak skenoval běžící procesy na počítači, na němž byl nainstalován, což mělo za následek zpomalení uživatelova počítače. Tento

[20] Systémový administrátor a bývalý zaměstnanec CIA, který pracoval i pro NSA. Neoprávněně zveřejnil utajované praktiky rozsáhlého sledování telefonní a elektronické komunikace americkými bezpečnostními službami.

software také fungoval jako rootkit. Software se snažil utajit svoji přítomnost v počítači a zároveň zamezit svému odstranění tím způsobem, že skrýval všechny soubory, složky nebo procesy, jejichž název začínal na $sys$. Ve výsledku by i jakýkoliv další malware, jenž by se v takto poškozeném operačním systému objevil, byl díky maskování tohoto rootkitu těžce objevitelný. A malware, jenž se snažil tohoto rootkitu využít, se objevil krátce poté, co informace o existenci takovéhoto softwaru na discích od společnosti Sony BMG pronikla na veřejnost. (McGraw, 2006) Incident se prohloubil tím, že program sloužící k odinstalování tohoto rootkitu, jenž Sony BMG následně zveřejnila na svých webových stránkách, nechával po svém použití počítač náchylný k dalšímu napadení skrze tomu uzpůsobené škodlivé webové stránky. (Even when Uninstalled, Sony's Rootkit Still Vexes PCs, 2005)

Rozdíl v použití jednoho či druhého typu softwaru se projeví také na právech a svobodě jedince v užití svého počítače a toho, jak může nakládat se svými dokumenty. Richard Stallman varuje, že se různé korporace pokoušejí skrze software, jenž vytvářejí, přebrat kontrolu nad počítačem a daty uživatelů. Pokud budou proprietární programy zahrnovat digitální šifrování s klíči, jež uživatelům nebudou dostupné, budou takové systémy moci kontrolovat, jaké programy může uživatel spouštět a ke kterým dokumentům a datům bude mít přístup. Přes Internet budou získávat nová pravidla, kterým se bude uživatel muset podřídit, a případné odpojení od sítě bude mít za následek, že některé funkce těchto programů nebudou přístupné. Což se paradoxně týká také programů, které připojení k Internetu ve své podstatě pro správné fungování vůbec nepotřebují. Uživatel tak při použití proprietárního softwaru může ztratit kontrolu nejen nad samotným proprietárním programem, ale i nad daty, se kterými v něm pracuje. V současné době jsou tyto praktiky využívány převážně pro správu digitálních dat (DRM), (Stallman, 2010) ale hrozí i možnost jejich zneužití k přístupu k osobním informacím například vládními režimy porušujícími lidská práva apod.

## 4.2 Nejčastější hrozby a slabiny open source a proprietárního softwaru

Počítačové systémy jsou nuceny čelit řadě různých hrozeb. Za bezpečnější je možné považovat ty softwarové systémy, které se dokáží těmto hrozbám lépe ubránit, u kterých je menší šance, že se stanou jejich terčem, nebo které jsou schopné minimalizovat následná poškození.

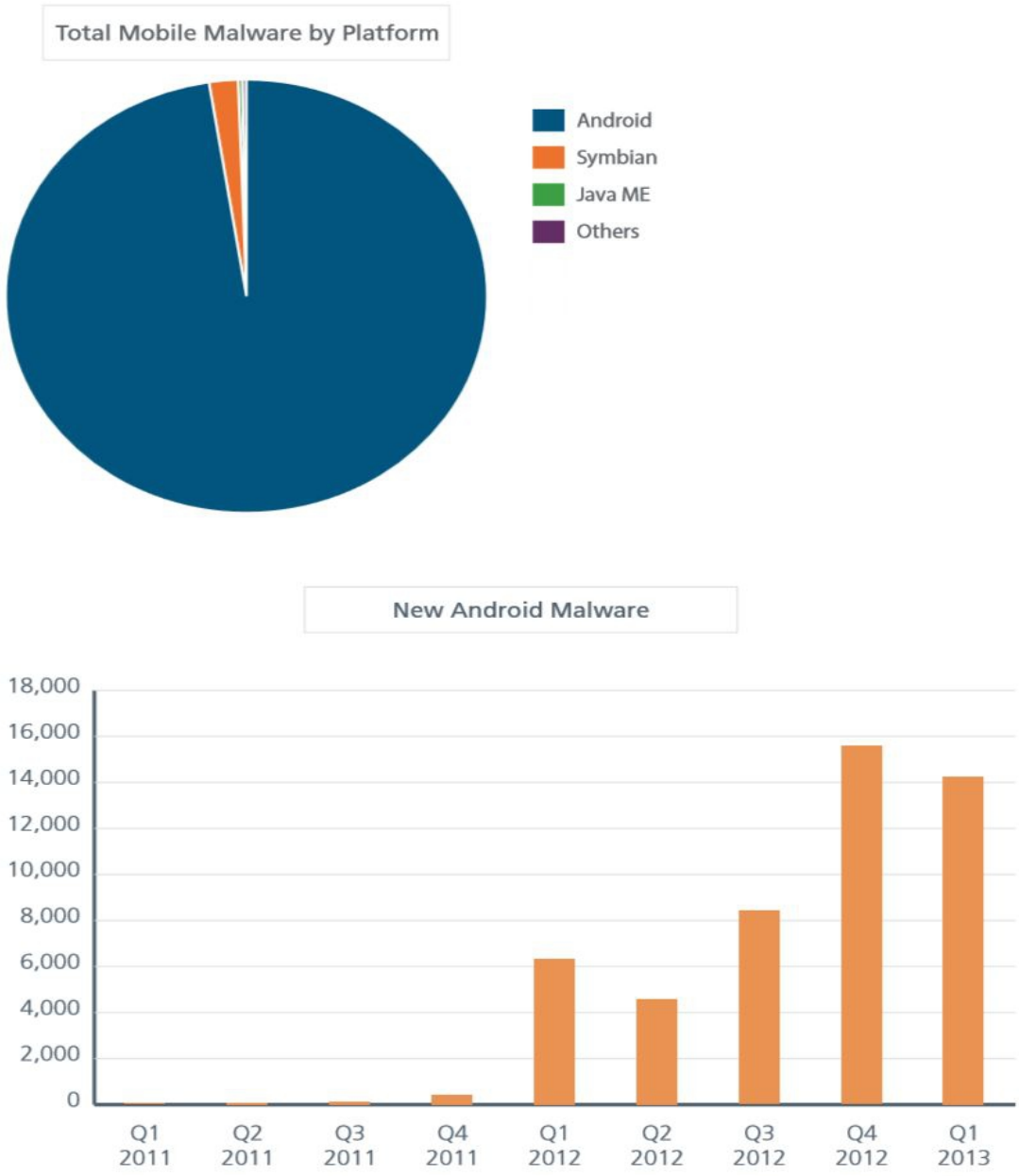
Diskuze na téma který typ softwaru je bezpečnější, zda open source, nebo proprietární, tak kromě množství bezpečnostních chyb a rychlosti a kvality jejich oprav například zahrnuje, jak často se takový software stává terčem útoků. A to jak těch plánovaných, individuálně prováděných, tak i například samovolně se šířících počítačových virů a malwaru. Řada zastánců OSS tvrdí, že open source a zvláště pak operační systém GNU/Linux je oproti komerčním proprietárním operačním systémům bezpečnější, protože není tak tak často terčem malwaru. Většina malwaru se skutečně zaměřuje, přinejmenším v prostředí operačních systémů, na proprietární operační systémy Windows, a to i přesto, že existuje řada antivirových nástrojů a sám Microsoft se proti němu již dlouhou dobu snaží bojovat. (Hunter, 2005) Skutečnost, že neexistuje tak velké množství malwaru, jehož cílem by byl open source a GNU/Linux, je podle některých dána faktem, že je v základu bezpečnější například tím, jak zachází s uživatelskými právy k instalaci softwaru, či díky tomu, že uživatelé mohou získávat software přímo z databáze producenta dané linuxové distribuce. Ne všichni však zastávají tento názor. Podle některých se operační systém GNU/Linux a další svobodný software nestává tak často terčem malwaru jednoduše proto, že jej využívá pouze malé množství uživatelů. Z toho vyplývá, že pokud chce někdo vytvořit škodlivý program, jenž by zasáhl co největší množství počítačových systémů, vyplatí se mu zaměřit spíše na software, který využívá větší množství lidí. Pokud však někdo malware pro OSS, jenž se bude sám od sebe šířit z postiženého počítače na další, přeci jen vytvoří, bude pro tento program složitější se dále šířit, protože mezi všemi dostupnými počítači na síti bude většina pro tento program k jeho fungování a dalšímu šíření nevhodná (za předpokladu, že tento malware není schopen napadení více platforem). Důvodem, proč nejsou OSS tak často terčem útoků může být také to, že již od doby jejich vzniku jsou jejich uživateli převážně zkušení a technicky zdatní jedinci, kteří jsou si

vědomi možných nebezpečí a vyhýbají se tak riskantnímu chování. Ačkoliv se postupem času objevují nové škodlivé programy zaměřující operační systémy typu GNU/Linux jako například *Hand of Thief,* stále zde chybí efektivní způsob, jakým by se mohly šířit. (Vaughan-Nichols, 2013) To ovšem platí pro systémy používané na stolních počítačích či jako servery. Odlišná situace je na scéně vložených (*embedded*) linuxových systémů na zařízeních připojených k Internetu. Routery nebo IP kamery, které využívají staré verze linuxového operačního systému a často zde ani není možné jejich systém aktualizovat, mohou být například napadeny virem, který využije zranitelností a z napadeného zařízení se bude dále šířit po síti. (Goodin, 2013)

Kompletně odlišná situace je také na scéně operačních systémů a softwaru pro chytré mobilní telefony. Zatímco se operační systémy pro osobní počítače a servery založené na GNU/Linux stávají terči útoků v mnohem menší míře než například Microsoft Windows, je naopak systém Android[21] terčem útoků mnohem častěji než jiné mobilní systémy a existuje mnohem větší množství malwaru, které se na tento systém zaměřuje. Někteří tento stav vnímají jako dobrý důkaz, že i open-sourcový operační systém, pokud má dostatečně velkou uživatelskou základnu, bude mnohem častěji zneužíván. Existence takového množství malwaru je přisuzována právě otevřenosti tohoto operačního systému a tomu, kolik práv jeho uživatelé mají oproti například operačnímu systému OS X společnosti Apple (základem tohoto operačního systému je také open-sourcové jádro). Podle jiných, jako je například Richard Stallman, je problém naopak v tom, že Android není dostatečně svobodný a otevřený svým uživatelům. Podle něj zde existuje řada faktorů, které se u klasických linuxových operačních systémů nevyskytují, jež mají na jeho výsledné kvality vliv. (Stallman, Android and Users' Freedom, 2014) Použití operačních systémů v prostředí mobilních telefonů je relativně novou záležitostí a je těžké dopředu říci, jakým způsobem se bude situace dál rozvíjet.

[21] Operační systém pro mobilní telefony a tablety založený na Linuxu.

Obr. 1: Množství existujícího malwaru pro mobilní telefony podle platformy, na níž je zaměřený + graf nárůstu malwaru pro mobilní operační systém Android.




Zdroj: (MCAFEE LABS, *2013)*

## 4.3 Vzorce pro srovnání a měření bezpečnosti

Při snaze o komparaci softwaru však nejsou abstraktní teorie ani důkazy vycházející pouze z několika málo incidentů dostačující. Je potřeba nalézt a použít správný model či postup, jak analyzovat a změřit bezpečnost softwarového systému, případně porovnat větší množství systémů navzájem. Co všechno a jak přesně by se mělo měřit, je však stále otázkou, na níž se v odborných kruzích objevují různé odpovědi a jsou nabízena různá řešení. G. Schryen a R. Kadura (2009) ve své práci Open source vs. closed source software shrnuli problémy, na které výzkum v tomto směru naráží. Podle nich existuje příliš malé množství relevantní literatury, a tudíž i dostatečného množství dobrých modelů pro analýzu. Stejně pak také neexistuje ani dostatečné množství relevantních dat vhodných pro využití v takové analýze a již existující data na druhou stranu často nedosahují příliš vysokých kvalit například co se týče jejich celistvosti nebo uspořádání. Dalším problémem je přistupování k softwarové bezpečnosti jako k nedělitelné jednotce. Je možné ji totiž rozdělit na více separátních částí, jako například důvěryhodnost či spolehlivost softwaru, jejichž oddělený výzkum, měření a srovnání by mohlo přinést přesnější výsledky o celkovém stavu bezpečnosti. Autoři zde navrhují rozdělení na základě zdroje zranitelnosti, tzn. podle toho, zda zranitelnost zapříčinil špatný design softwaru, špatná implementace a nebo je na vině prostředí, ve kterém se software používá. Modely pro analýzu bezpečnosti také často nepočítají s možností, že by bezpečnostní opravy mohly do softwaru uvést nové zranitelnosti. Většina modelů také nebere v úvahu, nakolik jsou nalezené bezpečnostní chyby závažné a jaké tak potencionálně představují nebezpečí. (Schryen and Kadura, 2009)

Pro analýzu bezpečnosti softwaru se objevilo několik různých modelů, které měly výhody, ale zároveň existovaly oblasti, jimž nevěnovaly dostatečné množství pozornosti, nebo aspekty, kterým nepřikládaly takovou důležitost, jaké je potřeba. Jednou z možností jak měřit bezpečnost, je model použitý v pracích Littlewoodem a kol. a Kimurou (2006, cited in Schryen and Kadura, 2009), jenž bere v úvahu celkové množství situací, při nichž došlo k narušení bezpečnosti, a čas, který uběhne než dojde k dalšímu incidentu, jako exponenciálně rozdělený. Tento model však nebere v úvahu, že množství lidí, jež se podílí na hledání chyb v softwaru, nemusí být lineární, ale může se měnit v čase. Software bude

například zkoumán větším množstvím osob těsně po zveřejnění jeho nové verze, než po několika letech jeho používání, a taktéž se může měnit i kvalita schopností těchto lidí. Jonsson a Olovsson (1997, cited in Schryen and Kadura, 2009) vytvořili podobný model, který se zaměřuje na celkový zisk útočící strany. Útoky rozdělili na tři fáze. V první fázi útočník získává informace, poté nasleduje standardní útok a nakonec je inovační útočná fáze. Také Alhazmi a kol. ve své práci rozdělil postup hledání zranitelnosti a jejího zneužití na tři fáze, kde dochází k hledání informací, dále pak samo objevování informací a nakonec přesun k další verzi softwaru. Na základě již získaných dat lze pomocí tohoto modelu i do jisté míry předpovídat zbývající množství neodhalených zranitelností. Objevují se i modely, jež se snaží brát v úvahu úsilí vynaložené na hledání zranitelností. Podle Alhazmi a Malaiya je možné odvodit snahu vynaloženou na hledání zranitelností v softwaru z množství počítačů, na nichž je daný software nainstalován. (cited in Schryen and Kadura, 2009) Existuje i řada dalších modelů, které se od sebe více či méně liší v proměnných, jež používají, v předpokladech, z nichž vycházejí, ale mění se i cíle a údaje měření, jichž se snaží dosáhnout. G. Schryen a R. Kadura ve své práci upozorňují, že většina modelů nebere ohled ani na to, jak závažné nalezené zranitelnosti ve skutečnosti jsou. „Počet zranitelností nemusí nutně odrážet úroveň bezpečnosti programu. Například pokud program A obsahuje pouze jednu zranitelnost, která je snadno nalezitelná a může být systematicky zneužitá a způsobit značné škody, potom A může být považován za méně bezpečný než program B, který sice obsahuje deset zranitelností, ale každá z nich je těžko nalezitelná, může být zneužitá pouze za velmi specifických okolností a nezpůsobí žádnou vážnou škodu.“ (Schryen and Kadura, 2009)

# 5. Databáze bezpečnosti informačních technologií

Existuje řada dobrých důvodů, aby se veškeré informace o věcech vztahujících se k bezpečnosti softwaru zaznamenávaly a ve strukturované formě ukládaly. Informace o možných slabostech systémů, specifických bezpečnostních chybách či o již vydařených útocích mohou sloužit jak pro správce a uživatele ohrožených systémů, tak i pro retrospektivní analýzy a vytváření předpokladů, které v budoucnu pomohou firmám a jedincům ke správným rozhodnutím. Je však bezpodmínečně nutné, aby taková data byla shromažďována v domluveném uniformním formátu, jenž se bude dodržovat, a ani nemohla být jakýmkoliv způsobem ovlivněna zájmovými skupinami, které by byly například proti zveřejňování jistých druhů informací.

Z jednoho z prvních projektů na shromažďování informací o bezpečnosti v počítačové sféře vznikla jedna z dnes nejvýznamnějších databází v tomto oboru – Computer Emergency Response Team Coordination Center (CERT/CC http://www.cert.org/). Je možné zde nalézt velké množství informací a událostí vztahujících se k počítačové bezpečnosti a zároveň jsou tyto informace jasně strukturované. Nevznikla na základě dlouhodobých příprav, ale v rámci pouze několika dní kvůli počítačovému červu zvanému Morrisův červ, jenž v roce 1988 začal napadat počítačové stanice připojené k Internetu. Velmi rychle se šířil státními a akademickými počítačovými systémy využívaje bezpečnostních chyb v softwarových programech, jež na těchto strojích fungovaly. Nakonec byla tato hrozba zažehnána díky spolupráci a výměně informací mezi poškozenými stranami a vyšlo najevo, že je potřeba, aby existovala možnost, jak v případě takových situací zařídit dostatečnou informovanost všech účastníků a snížení velkého množství dezinformací, jež se rychle šířily. Dále je potřeba, aby existovala jedna vedoucí strana, která by tyto snahy řídila. Také se ukázalo, nakolik důležitá byla dostupnost zdrojového kódu, neboť právě ty subjekty, jež k němu měly přístup, byly jako první schopné pochopit podstatu útoku a úspěšně se proti němu bránit. Navíc bylo potřeba nasbírat dostatečné množství informací, aby mohl být incident prostudován a našly se

způsoby, jak by se něčemu podobnému dalo v budoucnosti předcházet. (Reinke and Seiedian, 2003)

Další databází je National Vulnerability Database (NVD http://nvd.nist.gov/home.cfm), což je americká státní databáze fungující pod vedením Národního institutu standardů a technologie, která shromažďuje data o softwarových chybách, produktech, kontrolní seznamy apod. Jsou zde dostupná i standardizovaná rozhraní a protokoly pro správu i měření závažnosti zranitelností. Například její Common Vulnerability Scoring System (CVSS) poskytuje otevřené rozhraní pro komunikaci a hodnocení zranitelností v informačních technologiích. Umožňuje kvantitativně měřit a porovnávat bezpečnost na základě typu zranitelností, čehož je možné využít k odstraňování možných zranitelností v používaném softwaru nebo ohodnocení závažnosti nově nalezených zranitelností. (National Vulnerability Database)

Významným je i BugTraq mailing list, který vznikl roku 1993 jako reakce na nedostatečnou kvalitu již existujících stránek pro nahlašování a diskutování bezpečnostních rizik. Filozofií této stránky je princip *full disclosure,* podle kterého musí bezpečný systém obstát otevřené posouzení na všech úrovních, a tak se snaží o úplné zveřejnění a zpřístupnění informací všem. Ačkoliv si uvědomují, že to může vést i ke zneužití těchto informací, předpokládá se, že pozitiva této strategie vyváží její negativní stránky. Protože je zde možnost různé bezpečnostní problémy nejen nahlašovat, ale také diskutovat, je tento list moderován, aby se omezilo množství informačního šumu. (Security Focus, 2010)

Internet Storm Center se zabývá bezpečnostními riziky, jež ohrožují větší množství systémů zároveň, například systémy používané ve specifických odvětvích nebo regionech. Projekt je sponzorovaný americkou soukromou společností SANS Institute, ale na jeho fungování se podílí i velké množství dobrovolníků, kteří věnují svůj čas analýze bezpečnostních problémů. Byl založen roku 2001 při snaze o zastavení počítačového červa Li0n a v současné době spolupracuje s řadou různých organizací i s poskytovateli internetových služeb. Na tomto projektu se lze snadno podílet přeposíláním záznamů z *firewallu* svého systému a pokud tento systém byl nebo hrozí, že bude napaden, je správci daného zařízení zasláno příslušné varování. (Internet Storm Center)

# Závěr

Cílem této bakalářské práce bylo srovnat míru bezpečnosti, již nám poskytuje open source a proprietární software, a na základě shromážděných argumentů odpovědět na otázku, zda může užívání jednoho z nich být teoreticky bezpečnější a zda tomu na základě statistik a příkladů odpovídají i dosavadní praktické zkušenosti. Větší množství argumentů mluvících ve prospěch OSS v této práci reflektuje jejich vyšší výskyt v nalezených relevantních materiálech a nikoliv autorovu snahu favorizovat tento typ softwaru.

Z údajů prezentovaných v této práci vyplývá, že open source software oproti tomu proprietárnímu poskytuje svým uživatelům možnost vlastní kontroly nad hledáním chyb a jejich případnou opravou. Umožňuje však zároveň zneužití nalezených slabostí útočníkem. Běžnému uživateli většinou chybí schopnosti tyto chyby nalézt a opravit a pokud je komunita vývojářů věnujících se určitému softwarovému systému příliš malá a nebo vůbec neexistuje, nemůže dostatečně rychle a efektivně publikovat potřebné bezpečnostní patche. V dnešní době už je však mnoho open-sourcových projektů financováno nebo udržováno komerčními společnostmi, které z nich sice nemají přímý zisk, ale mohou díky tomu například snížit své výdaje apod. Zda jsou zdrojové kódy zveřejněné není jen záležitostí bezpečnosti, souvisí to také s finanční strategií některých společností, které se snaží minimalizovat své náklady, a tudíž nemají příliš velkou motivaci se bezpečností zabývat. Chtějí omezit šíření svého softwaru mezi ty, kteří za něj nezaplatí, a nebo prostě nechtějí, aby vyšly najevo všechny funkce či vlastnosti jimi publikovaného softwaru. Uživatelům takového softwaru, kteří by měli zájem i schopnosti se na jeho zlepšení podílet, ale kvůli tomu možnost vlastní kontroly chybí.

Otevřenost zdrojového kódu je zásadním krokem pro to, aby bylo možné software podrobovat analýze, hodnotit jej a zlepšovat, stejně jako se děje v akademické sféře s vědeckými články. Protože se stále nedaří předložit jasnou a definitivní odpověď na otázku, který z nich je bezpečnější, i nadále existuje, a pravděpodobně ještě dlouho existovat bude, pře mezi zastánci obou druhů softwaru, kterou rozdmýchá každý nový objev závažnější zranitelnosti v jednom z nich. Avšak i přesto, že neexistuje definitivní a obecně platný

závěr, je možné na základě teorií, zkušeností a statistik předložených v této práci vyvodit, že OSS má nejen lepší předpoklady k tomu, aby poskytoval svým uživatelům větší míru bezpečnosti a ochrany jejich osobních informací, ale odpovídá tomu i současná situace.

Je však důležité si uvědomovat, že slovy Bruce Schneiera: „Bezpečnost není produkt, ale proces“. (Schneier, 2000) Bezpečnostní chyby se budou v softwaru ve větší či menší míře vyskytovat pořád, ale zásadní význam má v tomto směru dobře informovaná veřejnost. Je důležité, aby si lidé plně uvědomovali možné hrozby a jejich následky a věnovali dostatečnou pozornost kvalitě produktů, jež používají, ať už se jedná o ty proprietární či svobodné, a aby pokud je tomu zapotřebí vyžadovali odpovědnost producentů za jejich software i za jejich jednání. Z toho důvodu jsou důležité i podobné studie a analýzy, které mapují stav a vývoj bezpečnosti v prostředí informačních technologií.

# Seznam použitých zdrojů:

About Microsoft Openness. MICROSOFT. [online]. [cit. 2014-04-16]. Dostupné z: http://www.microsoft.com/en-us/openness/default.aspx#home/openstory

ANTHONY, Sebastian. International Space Station switches from Windows to Linux, for improved reliability. *ExtremeTech* [online]. 2013 [cit. 2014-05-03]. Dostupné z: http://www.extremetech.com/extreme/155392-international-space-station-switches-from-windows-to-linux-for-improved-reliability

BOULANGER, A., 2005. Open-Source Versus Proprietary Software: Is One More Reliable and Secure than the Other?. IBM Systems Journal, vol. 44, no. 2, pp. 239-248 ProQuest Computing. ISSN 00188670.

BOULANGER, A., 2005. Open-Source Versus Proprietary Software: Is One More Reliable and Secure than the Other?. IBM Systems Journal, vol. 44, no. 2, pp. 239-248 ProQuest Computing. ISSN 00188670.

BROWN, Kenneth. Opening the Open Source Debate. [online]. 2002, s. 33 [cit. 2014-05-02]. Dostupné z: http://nats-www.informatik.uni-hamburg.de/pub/OSS2004/PaperCollection/AdTIOpensourceWhitepaper.pdf

Computer security. In: *Encyclopaedia Britannica* [online]. [cit. 2014-02-18]. Dostupné z: http://www.britannica.com/EBchecked/topic/130682/computer-security

COVERITY. *Coverity Scan: 2013 Open Source Report* [online]. 2014, 23 s. [cit. 2014-05-07]. Dostupné z: http://softwareintegrity.coverity.com/rs/coverity/images/2013-Coverity-Scan-Report.pdf

COWAN, CRISPIN. Software Security for Open-Source Systems. [online]. 2003 [cit. 2014-04-01]. Dostupné z: www.cs.unibo.it/~sacerdot/doc/papers/SoftwareSecurityforOpenSourceSystems.pdf

CURTIS, Keith. *After the software wars* [online]. Raleigh-Durham, N.C.: Lulu.com, 2009 [cit. 2014-05-04]. ISBN 978-057-8011-899.

Definice svobodného software. FREE SOFTWARE FOUNDATION. *Operační systém GNU* [online]. 2013/07/20 [cit. 2014-02-18]. Dostupné z: http://www.gnu.org/philosophy/free-sw.cs.html

Even when Uninstalled, Sony's Rootkit Still Vexes PCs. Manhasset: United Business Media LLC, Nov 17, 2005 ProQuest Computing.

FELTEN, Ed. The Linux Backdoor Attempt of 2003. Freedom to Tinker [online]. 2013 [cit. 2014-03-16]. Dostupné z: https://freedom-to-tinker.com/blog/felten/the-linux-backdoor-attempt-of-2003/

GLASS, Robert L. Facts and fallacies of software engineering. Boston, MA: Addison-Wesley, 2003, xvi, 195 p. ISBN 03-211-1742-5.

GOODIN, Dan. New Linux worm targets routers, cameras, „Internet of things“ devices. *Ars Technica* [online]. 2013 [cit. 2014-05-03]. Dostupné z: http://arstechnica.com/security/2013/11/new-linux-worm-targets-routers-cameras-internet-of-things-devices/

GREENWALD, Glenn at al. Microsoft handed the NSA access to encrypted messages. *The Guardian* [online]. 2013 [cit. 2014-09-04]. Dostupné z: http://www.theguardian.com/world/2013/jul/11/microsoft-nsa-collaboration-user-data

HANSEN, Marit, Kristian KÖHNTOPP a Andreas PFITZMANN. The Open Source approach — opportunities and limitations with respect to security and privacy. Computers. 2002, vol. 21, issue 5, s. 461-471. DOI: 10.1016/S0167-4048(02)00516-3. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S0167404802005163

HELLAND, Tanner. Where does Microsoft make money?. [online]. 2013 [cit. 2014-03-17]. Dostupné z: http://www.tannerhelland.com/4993/microsoft-money-updated-2013/

*History of Computer Security* [online]. 2011 [cit. 2014-05-04]. Dostupné z: http://media.wiley.com/product_data/excerpt/55/04707411/0470741155-184.pdf

HOEPMAN, Jaap-Henk a Bart JACOBS. INCREASED SECURITY Through Open Source. Communications of the ACM. 2007, roč. 50, č. 1, s. 79-83.

HOUSER, John. Case Studies in Brief. Library Technology Reports. 2009, roč. 45, č. 3, s. 21-24.

HUNTER, Philip. Microsoft tackles viruses and spyware at last, but will it be trusted on security?. Network Security. 2005, vol. 2005, issue 2, s. 16-17. DOI: 10.1016/S1353-4858(05)00201-1. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S1353485805002011

Chronological History of IBM. IBM. [online]. [cit. 2014-04-03]. Dostupné z: http://www-03.ibm.com/ibm/history/history/decade_1960.html

Kategorie svobodného a nesvobodného softwaru. FREE SOFTWARE FOUNDATION. *Operační systém GNU* [online]. [cit. 2014-02-18]. Dostupné z: http://www.gnu.org/philosophy/categories.htm

LEVY, Elias. Wide Open Source. In: SecurityFocus [online]. 2000-04-17 [cit. 2014-03-31]. Dostupné z: http://www.securityfocus.com/news/19

MANSFIELD-DEVINE, Steve. Open source: does transparency lead to security?. Computer Fraud. 2008, vol. 2008, issue 9, s. 11-13. DOI: 10.1016/S1361-3723(08)70137-4. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S1361372308701374

MCGRAW, G. Is Sony BMG Run by Malicious Hackers?. IT Architect, 01, 2006, vol. 21, no. 1. pp. 63 ProQuest Computing. ISSN 15572145.

MESSMER, E. Open Source Vs. Windows: Security Debate Rages. Network World, 2005, Jul 04, vol. 22, no. 26, pp. 26-27 ProQuest Computing. ISSN 08877661.

MESSMER, E. Security of Open-Source Software Scrutinized. Network World. 2013, Mar 25, vol. 30, no. 5, pp. 12-12,14 ProQuest Computing. ISSN 08877661.

MESSMER, E., 2013. NSA Backdoor Fears Creating Crisis of Confidence in U.S. High-Tech Products, Services. Network World (Online), Oct 09 ProQuest Computing. ISSN 08877661.

Myth Vs. Reality. InformationWeek, 2006, Aug 14, no. 1101, pp. 51-52,54 ProQuest Computing. ISSN 87506874.

NIST. *National Vulnerability Database* [online]. [cit. 2014-04-13]. Dostupné z: http://nvd.nist.gov/home.cfm

No Backdoor in Vista, Microsoft Promises. Manhasset: United Business Media LLC, Mar 06, 2006 ProQuest Computing.

PAUL, Ryan. Linux kernel in 2011: 15 million total lines of code and Microsoft is a top contributor. *Ars Technica* [online]. 2012 [cit. 2014-03-16]. Dostupné z: http://arstechnica.com/business/2012/04/linux-kernel-in-2011-15-million-total-lines-of-code-and-microsoft-is-a-top-contributor/

PAYNE, Christian. On the security of open source software. Information Systems Journal [online]. 2002, vol. 12, issue 1, s. 61-78 [cit. 2014-03-31]. DOI: 10.1046/j.1365-2575.2002.00118.x.

PRIETO, OLIVER. Computer Ethics: Open Source Software. [online]. [cit. 2014-04-04]. Dostupné z: http://www.undisciplinedbytes.com/wp-content/uploads/2009/10/OpenSourceSoftware.pdf

PROFFITT, Brian. FLOSS: Accept no substitutes. *ITworld* [online]. 2013 [cit. 2014-05-03]. Dostupné z: http://www.itworld.com/security/182757/floss-accept-no-substitutes

Proprietary Software. In: *Technopedia* [online]. [cit. 2014-02-18]. Dostupné z: http://www.techopedia.com/definition/4333/proprietary-software

RAYMOND, Eric S. The cathedral: musings on Linux and open source by an accidental revolutionary. 1st ed. Cambridge, Mass.: O'Reilly, 1999, xi, 268 p. ISBN 15-659-2724-9.

REINKE, John a Hossein SAIEDIAN. The availability of source code in relation to timely response to security vulnerabilities. Computers. 2003, vol. 22, issue 8, s. 707-724. DOI: 10.1016/S0167-4048(03)00011-7. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S0167404803000117

RUBIN, Aviel D. *Brave new ballot: the battle to safeguard democracy in the age of electronic voting*. 1st ed. New York: Morgan Road Books, c2006, viii, 280 p. ISBN 978-076-7922-104.

SANS INSTITUTE. *Internet Storm Center* [online]. [cit. 2014-04-14]. Dostupné z: https://isc.sans.edu/

SECURITYFOCUS. *SecurityFocus* [online]. 2010 [cit. 2014-04-14]. Dostupné z: http://www.securityfocus.com/

SHANNON, C. E. Communication Theory of Secrecy Systems*. Bell System Technical Journal. 1949, vol. 28, issue 4, s. 656-715. DOI: 10.1002/j.1538-7305.1949.tb00928.x.

SCHNEIER, Bruce. The Process of Security. *Schneier on Security* [online]. 2000 [cit. 2014-04-09]. Dostupné z: https://www.schneier.com/essay-062.html

SCHRYEN, Guido a Rouven KADURA. Open source vs. closed source software. Proceedings of the 2009 ACM symposium on Applied Computing - SAC '09. New York, New York, USA: ACM Press, 2009, s. 2016-. DOI: 10.1145/1529282.1529731. Dostupné z: http://portal.acm.org/citation.cfm?doid=1529282.1529731

SLASHDOT MEDIA. *SourceForge* [online]. [cit. 2014-05-04]. Dostupné z: http://sourceforge.net/

SLÍŽEK, David. Heartbleed Bug: jak se k chybě v OpenSSL staví české i světové firmy. Lupa.cz [online]. 10. 4. 2014 [cit. 2014-04-11]. Dostupné z: http://www.lupa.cz/clanky/heartbleed-bug-jak-se-k-chybe-v-openssl-stavi-ceske-i-svetove-firmy/

STALLMAN, Richard M. a [foreword by Lawrence LESSIG]. Free software, free society: selected essays of Richard Stallman [online]. 2nd ed. Boston, MA: Free Software Foundation, 2010 [cit. 2014-04-09]. ISBN 978-098-3159-209. Dostupné z: www.gnu.org/doc/fsfs-ii-2.pdf

STALLMAN, Richard. Android and Users' Freedom. FREE SOFTWARE FOUNDATION. *GNU Operating system* [online]. 2014 [cit. 2014-05-03]. Dostupné z: http://www.gnu.org/philosophy/android-and-users-freedom.html

STALLMAN, Richard. Linux a systém GNU. FREE SOFTWARE FOUNDATION. *Operační systém GNU* [online]. [cit. 2014-05-04]. Dostupné z: https://www.gnu.org/gnu/linux-and-gnu.html

STALLMAN, Richard. The GNU Project. FREE SOFTWARE FOUNDATION. *GNU Operating System* [online]. [cit. 2014-04-18]. Dostupné z: http://www.gnu.org/gnu/thegnuproject.html.en

Support for Windows XP has ended. MICROSOFT. *Windows* [online]. 2014 [cit. 2014-04-30]. Dostupné z: https://www.microsoft.com/en-us/windows/enterprise/end-of-support.aspx

THOMPSON, Ken. Reflections on trusting trust. *Communications of the ACM* [online]. 1984, vol. 27, issue 8, s. 761-763 [cit. 2014-05-02]. DOI: 10.1145/358198.358210. Dostupné z: http://portal.acm.org/citation.cfm?doid=358198.358210

VAUGHAN-NICHOLS, Steven J. Linux desktop Trojan 'Hand of Thief' steals in. *ZDNet* [online]. Cambridge, MA: CNET Networks, Inc, 2013 [cit. 2014-04-27]. Dostupné z: http://www.zdnet.com/linux-desktop-trojan-hand-of-thief-steals-in-7000019175/

WARE, Willis H. *Security Controls for Computer Systems: Report of Defense Science Board Task Force on Computer Security - RAND Report R-609-1* [online]. 1979 [cit. 2014-05-04]. Dostupné z: http://www.rand.org/pubs/reports/R609-1/index2.html

WITTEN, B., LANDWEHR, C. and CALOYANNIDES, M., 2001. Does Open Source Improve System Security?. IEEE Software, Sep, vol. 18, no. 5, pp. 57-61 ProQuest Computing. ISSN 07407459. DOI http://dx.doi.org/10.1109/52.951496

**Obrazové materiály, grafy a tabulky**:

Obr 1. MCAFEE LABS. *McAfee Threats Report: First Quarter 2013* [online]. 2013, 35 s. [cit. 2014-05-08]. Dostupné z: http://www.mcafee.com/au/resources/reports/rp-quarterly-threat-q1-2013.pdf

Tabulka 1. COVERITY. *Coverity Scan: 2013 Open Source Report* [online]. 2014, 23 s. [cit. 2014-05-07]. Dostupné z: http://softwareintegrity.coverity.com/rs/coverity/images/2013-Coverity-Scan-Report.pdf

Tabulka 2. REINKE, John a Hossein SAIEDIAN. The availability of source code in relation to timely response to security vulnerabilities. Computers. 2003, vol. 22, issue 8, s. 707-724. DOI: 10.1016/S0167-4048(03)00011-7. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S0167404803000117