SEGURANÇA DE SOFTWARE: CÓDIGO ABERTO VERSUS FECHADO FAZ DIFERENÇA QUANDO TRATAMOS DE VULNERABILIDADES EM PROGRAMAS?

12/11/2013

Este post visa esclarecer a importância da segurança de software e como a auditoria de código e melhorias nos processos são itens fundamentais para a qualidade. A auditabilidade de um código é um dos fatores a serem considerados, mas não pode ser ...

Resumo:

Este post visa esclarecer a importância da segurança de software e como a auditoria de código e melhorias nos processos são itens fundamentais para a qualidade. A auditabilidade de um código é um dos fatores a serem considerados, mas não pode ser o único.

Gostaria de começar este post com alguns avisos e agradecimentos. Primeiramente, gostaria de agradecer a 4Linux por me ceder este espaço para discutir um tema que recentemente tem ganhado muito espaço (esclarecerei os motivos para o uso da palavra recentemente, mesmo sendo este um tema bastante antigo). Preferi divulgar este pequeno artigo através da 4Linux simplesmente por eles serem uma empresa séria e já altamente reconhecida na comunidade Open Source, sendo esta um dos meus principais objetivos de atingir. Para as pessoas que preferem partir para perguntas e comentários, antes de sequer estudar (leia-se estudar, não apenas ler) um determinado conteúdo, sugiro que leiam completamente o FAQ que disponibilizei ao final deste artigo. Às pessoas interessadas em saber um pouco mais sobre mim também recomendo que vão para a FAQ. A todos aqueles que são amantes do software livre, peço que leiam o texto pensando no que vocês podem mudar para contribuir com a melhoria da qualidade na segurança do software, já altamente efetiva, que colaboram, utilizam ou promovem. Para todos os leitores: por favor entendam que meu objetivo aqui é desmentir algo que acaba sendo prejudicial à comunidade de software livre, e não atacar o mesmo. Software livre em minha opinião é a melhor opção e deve ser incentivado.

Este artigo foi amplamente motivado pela constante dúvida e discussão em torno da segurança de software. Programas crescem, novos programadores contribuem (independentemente do código ser aberto ou fechado) e diferentes quesitos e necessidades de qualidade surgem dependendo dos objetivos do mesmo (um grande exemplo é o software utilizado em dispositivos médicos, amplamente testado para prover a consistência dos dados, mas não necessariamente protegido contra leituras indevidas [1]). Essa realidade faz com que bugs (não necessariamente de segurança) constantemente sejam inseridos e corrigidos. A dúvida e discussão discorre sobre o que fazer a respeito. Como garantir a segurança no desenvolvimento e a constante melhoria deste quesito específico em uma aplicação? As denúncias recentes em relação a espionagem americana [2] ainda enfatizaram a inserção proposital de vulnerabilidades em aplicações. A discussão a que me refiro surge quando os defensores de software livre utilizam tais denúncias para dizer que o melhor e mais seguro é usar software livre.

Garantir a segurança de um software exige processos de teste e auditoria de código constantes. Obviamente pelo software ser livre, e portanto, seu código amplamente disponível, confunde-se esta facilidade com a automática segurança que a mesma proporcionaria. Infelizmente em uma área de exatas, a facilidade para algo não implica necessariamente que este algo venha a ser feito. E aqui fica minha súplica para as pessoas de software livre: aprendam mais sobre como encontrar vulnerabilidades em programas. Entendam mais sobre quesitos de segurança que de fato dificultam a exploração de sistemas (tais como ativar canary na compilação, fornecer executáveis PIE que proporcionam a efetiva aplicação de randomização, utilização correta de criptografia e seus diversos elementos), implementem e aceitem testes específicos para encontrar problemas de segurança (fuzzers e

análise estática).

O argumento de inserção de falhas propositais em software proprietários deveria deixar de ser utilizado, pois assume que tais falhas não poderiam ser inseridos em software abertos (vejam a influência, por exemplo, da NSA no desenvolvimento do kernel do Linux com o SELinux [3]). Tal argumento também assume que software proprietários não poderiam ser auditados (embora a tarefa seja mais trabalhosa, a mesma não é impossível, ainda mais quando se assume as formas clássicas que aparentemente a maioria das pessoas imagina para inserção de vulnerabilidades em programas).

Aproveitando para falar sobre a inserção proposital de vulnerabilidades, quero enfatizar que a mesma não se trata de modificações visivelmente falhas. Em geral, tais inserções são sutis, misturadas com reais benefícios

para o software e facilmente confundidas com erros não intencionais quando descobertas. Dado isto, como saber se um ou mais contribuidores de software não o fazem de propósito, tanto em empresas de software proprietário quanto em projetos Open Source? O pagamento por vulnerabilidades descobertas e os altos valores negociados em exploits [4] mudam a balança de valores quando tratamos tais assuntos, e a inocência perante este fato não deve ser desconsiderada.

Dado tudo o que foi dito até agora, do que se trata então um software seguro? Trata-se do software corretamente auditado (por um número adequado de programadores que entendam os problemas de segurança - infelizmente não existe um número mágico, nem uma proporção mágica, dado que tudo depende da complexidade do software, interrelação com elementos externos e diferentes quesitos), um software que possui testes constantes de segurança (automatizados na forma de fuzzers e análises estáticas, bem como manuais em novos recursos adicionados).

Um dos argumentos que costumo ouvir e que esta corretíssimo para o software aberto em diversos casos, mas infelizmente não se aplica a questão segurança está no fato de que uma empresa interessada em utilizar determinado sistema e que deseje garantir que o mesmo não possui vulnerabilidades intencionais e possui certa qualidade em termos de segurança, poderia simplesmente contratar um terceiro para efetuar uma auditoria, dada a disponibilidade do código. Este quesito mostra-se errado (na maioria dos casos, mas obviamente em casos menores pode ser totalmente plausível, espero que o leitor entenda tais argumentos e comece a criar uma visão crítica para avaliar cada caso) pois o custo para de fato se realizar uma análise extremamente completa de um projeto de porte mediano depois de desenvolvido seria astronômico (isso sem considerar a análise contínua do mesmo - razão pela qual as empresas de software fechado hoje apanham tanto em relação a segurança de seus programas, afinal, por anos não investiram e agora mesmo os bilhões direcionados a esta finalidade não conseguem resolver o problema) e portanto uma análise amostral teria de ser utilizada. Como explicado, vulnerabilidades intencionais são facilmente disfarçadas como vulnerabilidades normais, e portanto mesmo uma auditoria completa mostraria no máximo algumas vulnerabilidades e não todas de um software. Tal resultado pode ser atingido da mesma forma com análise de aplicativos binários (como já comprovado pelas diversas vulnerabilidades lançadas em diferentes programas de mercado). Ao invés de utilizar o argumento de ser mais seguro por ser aberto, dever-se-ia aproveitar a verdadeira vantagem do código aberto: Por ser aberto, poderia ser tornado mais seguro, caso a empresa estivesse disposta a investir em especialistas para tal projeto. Isso criaria o incentivo

para diversas empresas usuárias de software livre apoiarem os projetos, além de desenvolver um mercado maior para especialistas focados em software aberto e influenciar positivamente a segurança dele. A economia com licenças poderia ser revertida em conhecimento para a empresa (com a contratação de tais especialistas), que auxiliariam não apenas neste projeto, como em diversas outras necessidades de segurança da mesma, gerando economia posterior (danos a imagem e perdas em casos de invasões, bem como na própria contratação de mão de obra qualificada).

Se considerarmos o papel dos governos na melhoria da qualidade de software, ainda mais em face das espionagens denunciadas recentemente, devemos pensar que incentivos na educação e investimentos em sistemas auditáveis (e em processos de auditorias abertos para os mesmos) são provavelmente a única via para prover independência tecnológica. Deixarei para discutir este item em um próximo post, mas fiquem a vontade para começarem a opinar.

A pergunta que fica então, é: como se tornar um pesquisador de vulnerabilidades? Este pergunta me é muitas vezes feita por pessoas que pensam em encontrar vulnerabilidades em software proprietários e entrar no mercado de exploits. Estas pessoas se esquecem que a pessoa que audita códigos (tanto proprietário dentro das empresas, quanto Open Source, seja dentro de empresas ou apenas por diversão) também é um pesquisador de vulnerabilidades. Muitos pesquisadores de vulnerabilidades estão focados na análise binária (engenharia reversa, fuzzing) de um software, mas diversos (eu diria que a maioria) são especializados em análise de código. Como tudo em segurança (ou quase?), a prática leva a perfeição e aqui o software aberto pode contribuir (e com isso receber contribuições) e muito! Olhar as diversas correções efetuadas ao longo dos anos, os problemas encontrados e as melhores práticas adotadas para evitá-los é essencial para o interessado conseguir criar a mentalidade necessária para auditar código. Cursos na área podem agilizar este processo, dando a visão de padrões claramente errados, para que a experiência posterior permita o aprendizado e a extrapolação de casos mais gerais.

FAQ:

1-) Mas, o fato do software ser livre facilita a auditoria

Esse argumento é constantemente usado e apesar de ser válido, ainda deixa em aberto a questão de quem irá realizar tal auditoria e como ela seria realizada. Mesmo projetos de tamanho médio possuem tamanha complexidade que auditorias completas são inviáveis e portanto

amostragens devem ser utilizadas. Em tais casos, um software proprietário pode ser tão ou melhor auditado que um software livre (dado que amostragens podem conter erros de amostra, "sorte" nas escolhas, melhor experiência do analista com determinado quesito do software, etc.)

Tal argumento também esquece o fator mudança de um software livre: os projetos evoluem mais rapidamente do que as empresas conseguem atualizar seus sistemas, o que faz com que diversas vulnerabilidades corrigidas no mainline ainda não estejam com as atualizações presentes nas distribuições e menos ainda nos sistemas em produção. Investir nestes processos é essencial para o aumento da segurança do software livre.

A economia do mercado de vulnerabilidades é outro fator constantemente ignorado por tal argumento. Lembrando que vulnerabilidades inseridas são sutis, e muitas vezes confundidas/misturadas com novos recursos.

2-) Então o software proprietário é vantajoso em termos de segurança?

Mais uma conclusão precipitada, comumente vista em discussões em que as pessoas se apressam a generalizar e concluir coisas antes de as entenderem. NÃO. Software proprietário não é vantajoso em termos de segurança. Nem desvantajoso. Ele possui aspectos favoráveis (como o controle de releases/versões e práticas de testes) e fatores desfavoráveis (as empresas são influenciadas pelo mercado e não por quesitos técnicos e podem ser forçadas por governos a tomar ações duvidosas em relação a segurança).

O software livre pode e deveria ser sempre melhor, também no quesito segurança. O que acontece é que em diversos projetos, tal quesito não é prioritário; ou então é decidido por programadores não especialistas no assunto segurança.

SOFTWARE SEGURO É AQUELE EM QUE DECISÕES DE PROJETO LEVAM EM CONSIDERAÇÃO A SEGURANÇA DAS INFORMAÇÕES E AUDITORIAS CONSTANTES GARANTEM QUE ERROS SEJAM ENCONTRADOS, CORRIJIDOS, DOCUMENTADOS E LIÇÕES SEJAM APRENDIDAS. Em geral, software livre falha (e muito) na parte de documentações [6].

3-) Não há sentido em fazer cursos para auditoria de software, dado que cada software é diferente?

Sempre pensei que cursos podem ajudar caso a experiência prática do professor agilize o aprendizado do aluno. Nunca aceitei e nem aceitarei que um curso substitua a prática ou que sem um curso o aprendizado seria

impossível. No caso de auditoria de programas, dado o crescimento dos mesmos, a complexidade e a quantidade de informações disponíveis atualmente, um curso pode ajudar a alinhar os pensamentos e as pesquisas, provendo informações que contribuam para a extrapolação posterior do conteúdo para casos mais específicos vivenciados na prática.

4-) Você acha software livre mais seguro?

Vejo diversos problemas nos processos atuais de software em geral, seja ele livre ou proprietário. Mas as vantagens do software livre extrapolam quaisquer desvantagens na maioria dos casos e se as discussões fossem mais justas com menos argumentos duvidosos (por ambas as partes, e em geral feitos por pessoas que não são de fato especialistas nos itens em que estão argumentando), creio que ficaria MUITO mais claro para todos as vantagens do software livre.

Sobre o autor

Rodrigo Rubira Branco (BSDaemon), atua como pesquisador sênior no centro de excelência em segurança da Intel . Foi fundador do projeto Dissect || PE de análise de malware e palestrante em diversas conferências nacionais e internacionais, tais como Blackhat, Defcon, Hack in The Box, XCon e Hackito.

Membro do comitê técnico de diversas conferências (Blackhat Brasil, Hackito e Nosuchcon, por exemplo) também foi palestrante principal (keynote) em eventos fora do Brasil e em território nacional. É organizador do evento Hackers to Hackers (H2HC), maior e mais antigo evento de pesquisas em segurança da informação na América Latina. Atuou em diversas empresas, tais como Check Point (como Chief Security Research) e Qualys (como Diretor de Pesquisas de Vulnerabilidades e Malware). Também é conselheiro do Instituto Coaliza.

Em 2011 foi homenageado pela Adobe como um dos contribuidores principais em vulnerabilidades nos produtos da empresa. Brasileiro convicto (apesar de ter morado em Israel, Dubai e atualmente nos Estados Unidos), é membro do Comitê Técnico da RENASIC, ligada ao Centro de Defesa Cibernética (CDCiber) do Departamento de Defesa Brasileiro.

Mantenha contato com o autor e saiba mais sobre suas ideias via Twitter: @bsdaemon

REFERÊNCIAS:

[1] Jack, Barnaby. "IMPLANTABLE MEDICAL DEVICES: HACKING HUMANS". Black Hat Presentation.

[2] Snowden NSA Revelations.

[3] NSA. "Security-Enhanced Linux".
Site: http://www.nsa.gov/research/selinux/. Acessado em: 15/10/2013.

[4] the grugq. "Shopping For Zero-Days: A Price List For Hackers' Secret Software Exploit".
Site:http://www.forbes.com/sites/andygreenberg/2012/03/23/shopping-for-zero-days-an-price-list-for-hackers-secret-software-exploits/. Acessado em: 15/10/2013.

[5] Phrack Magazine. Site: http://www.phrack.org. Acessado em: 15/10/2013.

[6] Torvalds, Linus. "[stable] Linux 2.6.25.10".
Site: https://lkml.org/lkml/2008/7/15/296. Acessado em: 15/10/2013.