PROGRAMAÇÃO SEGURA EM AMBIENTE LINUX

Arnaldo Bispo de Jesus / Josemar Rodrigues Souza*

RESUMO: *Muito se tem discutido sobre segurança da informação. Sobretudo no dinâmico e imprevisível mundo digital, torna-se cada vez mais necessário rever procedimentos e técnicas, visando proteger as pessoas e organizações de ataques maliciosos e toda sorte de intrusões. Ideal seria que os programas e soluções computacionais fossem dotadas de dispositivos de segurança desde sua concepção, evitando transtornos e incidentes indesejáveis para seus futuros clientes e usuários. Este artigo objetiva apresentar uma revisão sobre o tema de segurança no desenvolvimento de software, bem como uma série de técnicas e medidas a serem adotadas por desenvolvedores e projetistas de software em ambiente Linux..*

Palavras-chave: Programação segura; Metodologia KISS; Segurança da informação.

INTRODUÇÃO

O sistema operacional Linux, dado o seu atrativo custo de aquisição e implantação, aliada a sua incontestável superioridade técnica frente a outros sistemas operacionais proprietários, tem atraído a cada dia grandes parcelas do mercado produtor e consumidor de sistemas mediados por computador. Este fato vem ao encontro do anseio da sociedade moderna em equipar, automatizar e tornar disponível sua produção material e intelectual em ambiente eletrônico, sobretudo na internet.

Entretanto, devido a este mesmo crescimento acelerado e desordenado, a web como a conhecemos, tem atraído uma série de eventos e pessoas nem sempre bem intencionadas. Os chamados "ciber-terroristas" têm intensificado seus ataques e estão cada vez mais sofisticados e audaciosos. É necessário, então, que os administradores e mantenedores de sistemas empreendam uma verdadeira guerra aos tais, lançando mão de procedimentos e técnicas que defendam seus interesses, num comportamento que, se criticamente analisados, parecem por vezes beirar a paranóia.

O presente artigo pretende atualizar programadores, estudantes, projetistas de software e público interessado, em geral, acerca das ações no mundo da segurança de desenvolvimento de software no sistema operacional Linux. Acreditamos que, se os softwares forem produzidos de modo a prever e resguardar a segurança de seus usuários, muito contribuirão em desonerar os administradores e mantenedores dos sistemas. Para tanto, na seção 1, apresentaremos os conceitos e definições que envolvem o tema segurança da informação; na seção 2, abordaremos o tema sob a ótica dos sistemas "open source", ressaltando as particularidades do sistema operacional Linux; na seção 3, apresentaremos as principais técnicas a serem observadas pelos programadores em ambiente Linux e, por fim, na seção 4, apresentaremos nossas conclusões e comentários finais sobre o estudo empreendido.

* Professores do Curso de Informática e do Núcleo de Estudos e Desenvolvimento de Aplicações Educacionais Livres – NEDAEL da Universidade Católica do Salvador – UCSal. E-mail: arnaldo@nedael.org; josemar@ucsal.br.

1 - CONCEITOS E DEFINIÇÕES

Princípios básicos da programação segura

Desenvolvedores e mantenedores de sistema de um modo geral, mesmo que de forma inconsciente, lançam mão todos os dias de técnicas e princípios julgados adequados às boas práticas de programação. Via de regra, muito do conteúdo constante dos guias, manuais e relatórios sobre programação segura é visto como ementário dos cursos de graduação em Ciência da Computação e Bacharelado em Informática (SABELFELD, 99).

Entretanto alguns grupos, em todo o mundo, empenham-se em produzir material subjacente e complementar, visando atualizar seus usuários sobre técnicas modernas de ataque e defesa. Uma destas fontes de referência é o IATF (Information Assurance Technical Framework). O IATF essencialmente é um guia operacional para desenvolvedores e projetistas de software que contem informações sobre técnicas, procedimentos e métodos de construção de sistemas de informação seguros (NSA 2004).

As instruções do IATF, em sua versão 3.1, fornecem orientações técnicas que visam a proteção tanto das informações quanto das infra-estruturas de informação. A infra-estrutura de informação pode ser entendida como a parte da organização responsável pelos depósitos e os meios de transmissão das informações, sobretudo aquelas voltadas às operações críticas. O objetivo da adoção destes princípios é o estabelecimento de uma postura "robusta" de segurança e envolve políticas, procedimentos, técnicas, e mecanismos relativos a todas as camadas do circuito de produção e disseminação das informações dentro da organização. O IATF define ainda um ciclo de processamento para desenvolvimento de sistemas capazes de fornecer "garantia de informação" que atendam às exigências de segurança tanto para o hardware quanto componentes de software nestes sistemas, Para tanto, lança mão de um conjunto de elementos chamados de *Defense-in-Depth* (Estratégias de defesa em profundidade*)* são eles:

- Defenda a Rede e Infra-estrutura de informação.
- Defenda o Ambiente de Computação.
- Defenda os elos de ligação da organização com terceiros.
- Defenda as informações que geram procedimentos e regras de negócio.

Além das estratégias, o IATF define ainda uma estrutura de departamentalização ou setorização dos núcleos estratégicos que são: *Department of Defense* – DoD (Departamento de Defesa), *Global Information Grid* - GIG (Grade de Informação Global), *Information Assurance policy* (Política de IA) .

A figura 1 ilustra como as estratégias, e os departamentos se comunicam com a política de IA.

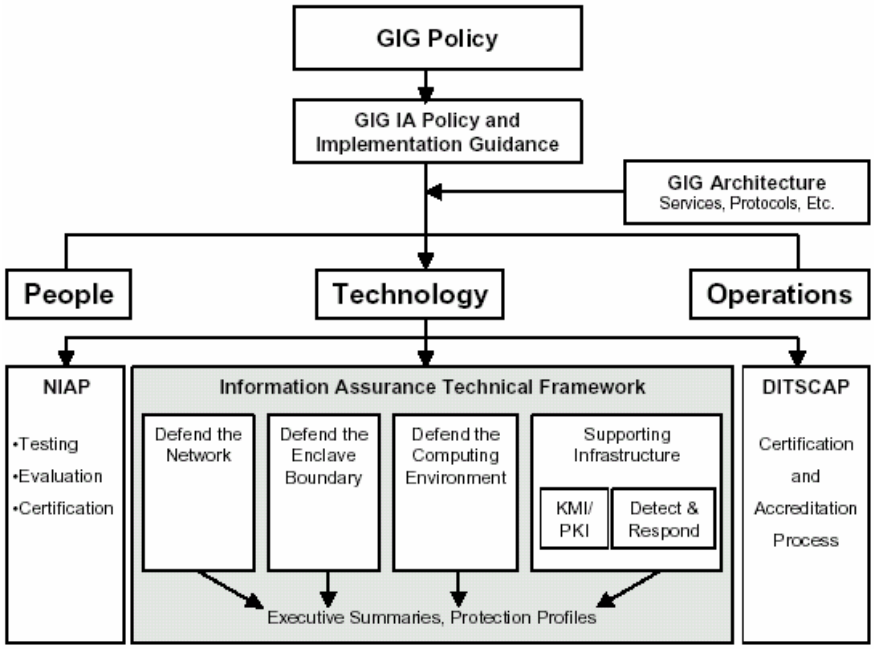

Figura 1: Esquema de relacionamento entre estratégias, departamentos e políticas de segurança – Fonte: IATF FORUM.

Além do IATF, uma série de outros guias, contendo preceitos para desenvolvimento e administração segura de sistemas, podem ser encontrados na Internet;, entre eles, o do *Computer Security Resource Center* (CSRC) mantido pelo NIST (*National Institute of Standards and Technology*) (CSRC 2004). Alguns princípios defendidos pelo CSRC são:

Confidencialidade (confidentiality) - Este é o princípio de que sistemas computacionais e seus recursos só podem ser acessadas por pessoas autorizadas.

Integridade (Integrity) - Os dados só podem ser modificados e apagados por meios, pessoas e/ou outros sistemas autorizados.

Disponibilidade (Availability) - Este é o princípio de que os dados e recursos somente estarão disponíveis para determinadas partes do sistema e em um tempo especificado. A quebra deste princípio é chamada de Negação do Serviço ou *"Denial Of Service"* ou *"DOS Attack"*.

2 - SEGURANÇA EM SISTEMAS "OPEN SOURCE"

Sistemas *Open source* são programas cujas licenças dão aos seus usuários a liberdade de utilização dos programas para qualquer propósito: estudar e modificar o programa, redistribuir cópias do original ou modificar o programa (sem ter que pagar fortunas por isso). Sistemas *Open Source*, ou sistemas de código fonte aberto, recebem esta denominação graças à natureza colaborativa de seu

desenvolvimento e, por este mesmo motivo, muitas vezes tornam-se passíveis de ataques maliciosos (PFLEEGER, 1997), necessitando, portanto, de medidas que visem resguardar sua integridade. Desta forma, podemos definir programação segura em sistemas Linux como um conjunto de técnicas e regras práticas que visam assegurar sistemas e procedimentos de software, no tocante a vulnerabilidades ou ações maliciosas.

Programação segura envolve uma série de medidas, as quais dotarão programas de usos diversos, de certo grau de confiabilidade. Segundo (WHEELER 2000), um programa seguro situa-se na fronteira da segurança, levando-se em conta a interação com outros elementos (usuários e programas) que não possuem as permissões necessárias para tal.

A construção de programas seguros envolve um conjunto de ações, que vão desde a adoção de modelos de criptografia até a implantação de políticas internas de segurança e questões como engenharia social (MCDERMOTT, 96). As vulnerabilidades mais comuns são: *Buffer overflows, links simbólicos, Pathnames, bugs* ocultos, etc. Suas principais características estão descritas em (IEEE 2002) e enumeradas a seguir:

Buffer overflows: Ou estouro de *buffer* é uma das formas mais comuns de ataque. Consiste em executar no programa vítima, uma série de códigos maliciosos que provocam a sobrecarga de blocos de memória estrategicamente instanciados (buffers). Existem várias formas de ataque deste tipo como, por exemplo, stack smashing, que consiste em substituir o endereço de retorno. Quando a função retorna de uma chamada, ao invés de mover para o endereço de retorno esperado, é movida para o endereço especificado pelo atacante. Isso permite ao invasor a possibilidade de executar código malicioso. Uma das formas de defesa para este tipo de ataque é o stackguard um compilador que gera um código binário chamado “monitor" o qual, uma vez incorporado ao programa projetado, prevê ataques deste tipo.

Race conditions: O sistema está “em condição de corrida" quando dois ou mais programas tentam alterar ambientes ou bloquear recursos compartilhados. Este tipo de concorrência pode levar o sistema a panes sucessivas, acarretando problemas como: *deadlock, livelock* e outras chamadas “locking failure conditions”. A primeira coisa a fazer é checar o código fonte. Deve-se assegurar de que não existe nenhum par de operações que possam vir a falhar se um código malicioso se aproveitar de seus intervalos. Um programa seguro é aquele que consegue decidir se um pedido de processamento pode ser concedido sem causar condições de conflito. Neste caso, não deve existir nenhuma brecha para que um usuário não autorizado possa se aproveitar de um estado do sistema antes que o mesmo possa se recompor. Este tipo de “race condition" é o mais comum e é chamado de *TOCTOU* “time of check - time of use".

Erros de parâmetros: Parâmetros são seqüências de caracteres de tipo variado que são passados para os programas ou movidos de uma rotina para outra através das chamadas. Um parâmetro pode ser um dado simples, como um valor a ser usado em um cálculo, ou uma informação extremamente complexa como uma lista de endereços web. Neste sentido, parâmetros podem ser usados como subconjuntos das próprias linguagens de programação. O programador ou projetista descuidado pode, deste modo, estar abrindo uma brecha enorme em seus sistemas para que atacantes que conhecem o formato dos parâmetros possam agir. É necessária a análise minuciosa tanto das chamadas quanto das implicações que os parâmetros têm entre elas.

Ataques e tentativas de invasão em todo o mundo: Um estudo realizado por (WHEELER 2000) apresenta a informação de que, em 1997, o CERT/CC informou 2.134 incidentes de segurança de computador e 311 vulnerabilidades distintas; antes de 2002, tinha subido a 82.094 incidentes e 4.129 vulnerabilidades. O Instituto de Segurança de Computador – *Computer Security Institute* (CSI), a Agência Federal de São Francisco e o FBI inspecionaram 503 grandes corporações e agências de governo em 2003 e concluíram que 92% dos entrevistados informaram ataques. As organizações estudadas identificaram sua conexão de Internet (78%) e seus sistemas internos (36%) como pontos freqüentes de ataque. 75% deles admitem perdas financeiras, embora só 47% conseguissem quantificar as perdas. Estima-se que os prejuízos somados totalizem mais de 200 milhões de dólares.

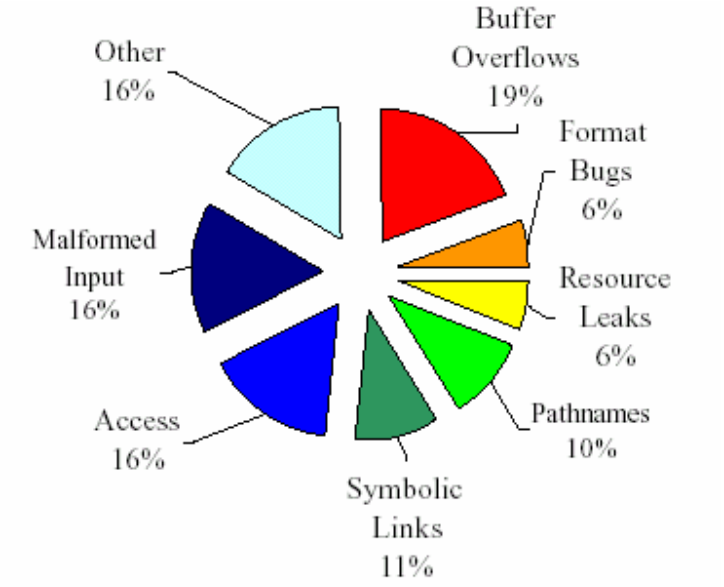

Figura 2: Ataques a sistema vulneráveis ou em estado de falha em Janeiro a setembro de 2001 – Fonte: IEEE

3 - TÉCNICAS DE PROGRAMAÇÃO SEGURA

Adotar boas práticas de programação e seguir alguns princípios clássicos da programação segura podem poupar desenvolvedores de uma série de problemas tidos como comuns na área de segurança computacional. Esta seção apresenta algumas recomendações julgadas importantes:

Metodologia KISS: (Keep It Simple, Stupid) Baseia-se no princípio da simplicidade e da economia de recursos, e o projeto de segurança deve ser o mais compacto e simples possível. A metodologia KISS defende o princípio de que o código fonte deve ser o mais compacto quanto possível, o que facilita a auditoria e a revisão rápida em caso de suspeita de intrusão.

Limitação de privilégios: Normalmente, sistemas de grande porte acessam recursos estratégicos tanto da rede quanto dos sistemas operacionais da organização. Para que estes sistemas executem corretamente suas atividades, os programadores eventualmente embutem privilégios nestes sistemas. Neste caso, recomenda-se que as funções que necessitam de privilégios sejam centralizadas.

Auditoria de sistemas: Um dos mais defendidos princípios da programação segura é a auditoria de sistemas, que prega a prática de que os sistemas devem ser periodicamente auditados a fim de verificar possíveis alterações em seu comportamento.

Consistência de dados de entrada: Deve-se, sempre que possível, dotar os sistemas de mecanismos de validação e consistência dos dados que os alimentam, a fim de evitar que código malicioso seja inserido entre os dados.

Variáveis de ambiente: Alguns sistemas necessitam de configuração especial através de variáveis de ambiente, estas configurações podem variar de simples informações sobre pastas e arquivos até endereços e esquemas de empacotamento na rede. Assim podem esconder esquemas completos de ataque por parte do invasor que conheça sua estrutura e vulnerabilidades.

Mascaramento Aleatório: Consiste em designar aleatoriamente quais algoritmos devem responder às chamadas do sistema. Estes mudam entre si a forma de execução e tratamento dos dados, escondendo a complexidade do processamento e dificultando que seus mecanismos sejam estudados por terceiros.

Componentes de terceiros: Deve-se sempre que possível evitar a utilização de componentes de terceiros, cujo método de implementação e o código fonte sejam desconhecidos do desenvolvedor que está embutindo-os no sistema. Um programador desavisado pode optar por adotar componentes deste tipo que têm acesso a partes específicas do sistema ou dos depósitos de dados, promovendo, desta forma, um meio de acesso por parte de invasores.

4 -CONCLUSÕES

Este artigo apresentou uma revisão do tema: Segurança no desenvolvimento de softwares, sendo o enfoque deste trabalho avaliar como se dá a relação entre os mecanismos de intrusão mais conhecidos com as características do sistema operacional Linux que propiciem quaisquer técnicas de invasão.

Tal preocupação mostra-se relevante devido à natureza aberta e flexível dos programas que rodam sob tal plataforma, uma vez que estes programas podem ser modificados, infectados ou estudados sem que o usuário comum tenha conhecimento prévio do fato.

Espera-se que estudantes de computação, projetistas e desenvolvedores de software, a partir das reflexões propostas neste texto, possam se munir de recursos e conhecimentos que lhes permitam, tornar suas soluções computacionais mais seguras e eficientes, evitando danos às organizações, aos usuários e à sociedade de um modo geral. Dada a relevância e abrangência do tema proposto, técnicas e soluções de segurança computacional, sobretudo no desenvolvimento de software, é um assunto pesquisado em universidades e empresas fabricantes de software em todo o mundo, revelando tendências e perspectivas quanto ao futuro da pesquisa que fundamenta a questão, por exemplo:

- Desenvolvimento de ferramentas de análise de vulnerabilidades nos sistemas (MIND 2004).
- Uso da inteligência artificial para prevenção e detecção de intrusões (DERBI 2004).
- Desenvolvimento de mecanismos que apóiem a justiça no tocante à coibição de crimes na informática (IJURIS 2003).

Os autores pretendem, ainda, que este trabalho desperte, sobretudo nos alunos da graduação dos cursos de informática, sistemas de informação e ciência da computação, o interesse pelo tema, gerando desta forma novas pesquisas, análises e técnicas que contribuam para a melhoria da segurança em sistemas computacionais.

REFERÊNCIAS

(IEEE 2002) Improving Security Using Extensible Lightweight Static Analysis, David Evans and David Larochelle University of Virginia – 2002.

(WHEELER 2000) WHEELER DAVID. Secure Programming for Linux and Unix. Disponível em: < www.dwheeler.com >. Acesso em: 06/07/2004

(PFLEEGER 1997) PFLEEGER, CHARLES P. 1997. Security in Computing. Upper Saddle River, NJ: Prentice-Hall PTR. ISBN 0-13-337486-6.

(NSA 2004) National Security Agency (NSA). Information Assurance Technical Framework (IATF). Disponível em: < http://www.iatf.net > Acesso em 13/08/2004;

(SCRC 2004) Computer Security Resource Center. Disponível em < http://csrc.nist.gov/publications/nistpubs/ > Acesso em: 04/07/2004.

(MIND 2004) Artificial Intelligence: Computer Crime Security. Diponivel em: <http://www.mindingyourebusiness.com > Acesso em: 10/05/2004

(DERBI 2004) DERBI: Diagnosis, Explanation and Recovery from computer Break-Ins. Disponível em: < http://www.ai.sri.com/~derbi/ > Acesso em: 16/05/2004

(SABELFELD 99) SABELFELD, ANDREI e SANDS, DAVIDS. A Per Model of Secure Information Flow in Sequential Programs. European Symposium on Programming, 1999

(MCDERMOTT 96) MCDERMOTT, JOHN e GOLDSCHLAG, DAVID . Towards a Model of Storage Jamming. The 9th Computer Security Foundations Workshop. IEEE Computer Society Press, - 1996

(IJURIS 2003) IJURIS - Instituto de Governo Eletrônico, Inteligência Jurídica e Sistemas. Disponível em: < **http://www.ijuris.org/** > Acesso em 11/11/2003.