Segurança

(Apenas anotações)

Ameaças

Confidencialidade: o proprietário deve ser capaz de especificar quem pode ver o que e o sistema tem de assegurar essas especificações.
Ameaça: Exposição de dados

Integridade: usuários não autorizados não devem ser capazes de modificar dado algum sem a permissão do proprietário.
Ameaça: Manipulação de dados

Disponibilidade: ninguém pode perturbar o sistema para torná- lo inutilizável.
Ameaça: Recusa de serviço

Atacantes

Hacker: Autodidata, cara que entende muito da programação. Detém o conhecimento mas não usa esse conhecimento para invadir sistemas e roubar as informações.

Cracker: “Hacker do mal”

Motivações: roubo, ativismo cibernético, vandalismo, terrorismo, guerra cibernética, espionagem, spam, extorsão, fraude — e ocasionalmente o atacante ainda quer simplesmente se exibir, ou expor a fragilidade da segurança de uma organização.

Atacantes variam de aspirantes a chapéus pretos não muito habilidosos, também conhecidos como script-kiddies, a crackers extremamente habilidosos.

O esforço necessário para a segurança e proteção claramente depende de quem achamos que seja o inimigo.

Segurança de sistemas operacionais

Alguns dos incidentes de segurança mais importantes ocorrem por causa de ataques cibernéticos sofisticados.

Há ataques que tentam roubar informações passivamente e ataques que tentam ativamente fazer com que um programa de computador comporte-se mal.

Criptografar: embaralhar uma mensagem ou arquivo de tal maneira que fique difícil de recuperar os dados originais a não ser que você tenha a chave.

Endurecimento de software: mecanismos de proteção a programas a fim de dificultar a ação de atacantes que buscam fazê-los comportar-se inadequadamente. O problema é que hoje os softwares já tem um grau de complexidade alto, e por causa disso incluir mais funcionalidades para garantir uma segurança extra pode causar um overhead de eficiência.

A única maneira segura de construir um sistema seguro é mantê-lo simples. As funcionalidades são o inimigo da segurança.

Cada sistema confiável possui uma TCB (Trusted Computing Base — Base Computacional Confiável) mínima consistindo no hardware e software necessários para fazer valer todas as regras de segurança. Se a base de computação confiável estiver funcionando de acordo com a especificação, a segurança do sistema não pode ser comprometida, não importa o que mais estiver errado. (não faz isso hoje em dia)

Em geral, a segurança de um sistema é inversamente proporcional ao tamanho da base computacional confiável.

Os sistemas operacionais não são projetados para serem seguros, o principal objetivo é que eles sejam funcionais, é que ofereçam um maior número de recursos possível pros usuários, programas e aplicativos.

Controlando o acesso aos recursos

Direitos ao acesso a informações podem ser modelados como uma grande matriz, com as linhas sendo os domínios (usuários) e as colunas sendo os objetos (por exemplo, arquivos).

Cada célula especifica os direitos de acesso do domínio para o objeto. Tendo em vista que a matriz é esparsa, ela pode ser armazenada por linha, que se torna uma lista de capacidade dizendo o que o domínio pode fazer, ou por coluna, caso em que ela se torna uma lista de controle de acesso dizendo quem pode acessar o objeto e como.

ACL: lista (ordenada) contendo todos os domínios que possam acessar o objeto, e como.

Modelos formais de sistemas seguros

Modelo Bell-LaPadula: modelo de segurança multinível mais amplamente usado, projetado para lidar com segurança militar, mas também é aplicável a outras organizações. Princípio de não vazamento das informações.

Possui regras sobre como a informação pode fluir:

● Propriedade de segurança simples: um processo executando no nível de segurança k pode ler somente objetos nesse nível ou mais baixo.

● Propriedade *: um processo executando no nível de segurança k só pode escrever objetos nesse nível ou mais alto.

Pode ser demonstrado que nenhuma informação pode vazar de um nível de segurança mais alto para um mais baixo.

Problema: processo 2 cria um arquivo no nível 3, processo 3 modifica o arquivo no nível 3, processo 2 sobrescreve arquivo no nível 3. Logo, não garante a integridade dos dados.

Modelo Biba: modelo projetado para garantir a integridade dos dados.

As regras sobre como a informação pode fluir são:

● Propriedade de integridade simples: um processo executando no nível de segurança k pode escrever somente objetos nesse nível ou mais baixo (não acima).

● Propriedade integridade: um processo executando no nível de segurança k só pode ler objetos nesse nível ou mais alto (não abaixo).

Canais ocultos

Usando as técnicas de modelagem formais, o fluxo de informação em um sistema pode ser modelado e limitado. No entanto, às vezes ele ainda pode vazar canais ocultos, como modular o uso da CPU.

Usando um esquema de matriz de proteção conseguimos facilmente garantir que o servidor não possa se comunicar com o colaborador escrevendo um arquivo para o qual o colaborador tenha acesso de leitura. Podemos também provavelmente assegurar que o servidor não possa se comunicar com o colaborador usando o mecanismo de comunicação entre processos do sistema.

Esteganografia

Tipo de canal oculto que pode ser usado para passar informações secretas entre processos, pois esconde a existência de informações (técnica popular para as pessoas que acreditam firmemente na liberdade de expressão).

Ex: Imagem das zebras que contém o texto criptografado da peça de shakespeare

Outro uso da esteganografia é para a inserção de marcas d'água em imagens usadas nas páginas da web para detectar seu roubo e reutilização em outras páginas da web. Música, filmes e outros tipos de materiais também podem ser identificados com marcas d'água dessa maneira.

Criptografia

Uma maneira de manter a informação secreta é criptografá-la e gerenciar as chaves cuidadosamente. Esquemas criptográficos podem ser categorizados como chave secreta ou chave pública.

Chave secreta: exige que as partes se comunicando troquem uma chave secreta antecipadamente, usando algum mecanismo fora de banda.

Chave pública: não exige a troca secreta de chaves antecipadamente, mas seu uso é muito mais lento.
Ex: RSA

Esse sistema tem a propriedade que chaves distintas são usadas para criptografia e decriptação e que, fornecida uma chave criptográfica bem escolhida, é virtualmente impossível descobrir-se a chave de decriptação correspondente. Sob essas circunstâncias, a chave de encriptação pode ser tornada pública e apenas a chave de decriptação mantida em segredo.

Ex: Alguém quer mandar mensagem para mim, vai usar a minha chave pública para cifrar e só eu com minha chave privada posso decifrar.

As chaves simétricas eu consigo ter um nível de segurança maior e é mais rápido do que o algoritmo de chave assimétrica.

Combina as chaves: Usa algoritmos de chave pública para estabelecer a conexão, e define uma chave de sessão simétrica naquela conexão.

Chave de sessão: usa chave pública para estabelecer uma chave secreta a ser usada naquela sessão.

Assinatura Digital

Para provar a autenticidade de uma informação digital, podem ser usados resumos criptográficos, assinaturas digitais e certificados assinados por uma autoridade de certificação confiável.
● MD5Sum, SHA-1, SHA-2

Módulos de plataforma confiável (TPM) (Conceitual)

Toda criptografia exige chaves. Se as chaves são comprometidas, toda a segurança baseada nelas também é comprometida. Portanto, é essencial armazenar as chaves de maneira segura.

O TPM pode realizar operações criptográficas como encriptar blocos de texto puro ou a decriptação de blocos de texto cifrado na memória principal. Também pode verificar assinaturas digitais.

O TPM não torna os computadores mais seguros contra ataques externos. Seu foco realmente é utilizar a criptografia para evitar que os usuários façam qualquer coisa que não seja aprovada direta ou indiretamente por quem quer que controle o TPM.

Autenticação

Visa garantir que a informação está íntegra mas também que a origem dela é confiável.

Em qualquer sistema seguro, usuários precisam ser autenticados. Isso pode ser feito por algo que o usuário conhece, algo que ele tem, ou que ele é (biometria).

A identificação por dois fatores, como uma leitura de íris e uma senha, pode ser usada para incrementar a segurança.

Senha fraca: como lembrar-se de todas as senhas é difícil demais, as pessoas tendem a escolher senhas simples, fracas e reutilizá-las em muitos sites. Uma senha fraca ou padrão capacita os atacantes a colher um grande número de contas, às vezes com todos os direitos do administrador.

Segurança por senhas do UNIX

O programa de login pede ao usuário para digitar seu nome e senha. A senha é imediatamente “encriptada”, usando-a como uma chave para encriptar um bloco fixo de dados.

A vantagem desse esquema é que ninguém, nem mesmo o superusuário, poderá procurar pelas senhas de qualquer usuário, pois elas não estão armazenadas de maneira encriptada em qualquer parte do sistema. Para fins de ilustração, presumimos por ora que a senha encriptada é armazenada no próprio arquivo de senhas.

Se o atacante consegue obter a senha encriptada, o esquema pode ser atacado.

Técnica que torna o ataque quase inútil: associar um número aleatório de n-bits, chamado sal, com cada senha. Ex: dois usuários com senhas iguais, a hash de criptografia seria idêntica.

Senhas de uso único

Trata-se da modificação da senha a cada login.

O algoritmo é baseado em uma função de mão única, isto é, uma função y = f (x) cuja propriedade é: dado x, é fácil de encontrar y, mas dado y, é computacionalmente impossível encontrar x. A entrada e a saída devem ser do mesmo comprimento, por exemplo, 256 bits.

Quando todas as senhas tiverem sido usadas, o servidor é reinicializado com uma nova chave secreta.

Autenticação por resposta a um desafio

A ideia é fazer com que cada novo usuário forneça uma longa lista de perguntas e respostas que são então armazenadas no servidor com segurança. As perguntas devem ser escolhidas de maneira que o usuário não precise anotá-las.

No login, o servidor faz uma das perguntas aleatoriamente e confere a resposta.

No desafio-resposta, o usuário escolhe um algoritmo ao registrar- se como um usuário, por exemplo, x2. Ao fazer o login, o servidor envia a ele um argumento, digamos 7, ao que o usuário digita 49. O algoritmo pode ser diferente de manhã e de tarde, em dias diferentes da semana, e por aí afora.

Hoje, o objeto físico usado é muitas vezes um cartão plástico que é inserido em um leitor associado com o computador. Além de inserir o cartão, é preciso digitar uma senha para evitar que alguém use um cartão perdido ou roubado.

Os cartões de plástico contendo informações podem ser de dois tipos: com uma faixa magnética ou com chip (cartões com valores armazenados e cartões inteligentes).

Autenticação usando biometria

Um sistema de biometria típico tem duas partes: cadastramento e identificação.

A característica escolhida deve ter uma variabilidade suficiente para distinguir entre muitas pessoas sem erro.

Exemplos: análise do comprimento dos dedos, reconhecimento pela íris, análise de assinatura, biometria de voz,

Explorando softwares

Muitos tipos de defeitos no código podem ser explorados para assumir os programas e sistemas:

Transbordamentos de buffer:

Maioria dos SOs escritos em C ou C++
- não faz verificação de limites de vetores

Com endereço de retorno sobrescrito o desvio vai para um endereço qualquer e o programa "dá pau" ou desvia para um endereço especialmente selecionado pelo invasor com o código que ele deseja executar (ex: código de shell)

Escreve na posição de memória fora do seu arranjo

Prevenção de execução de dados (DEP – Data Execution
Prevention)
- bit NX (Não eXecutar)
- Mac OS X, Linux e Windows usam esta técnica

Ataques de reutilização de código
- retorno à libc (System)

- ROP (Return Oriented Program-
ming): junção de rotinas simples

terminando com um return

Usuário consegue escrever em áreas que não poderia, blocos pequenos de código

Randomização de layout endereço-espaço (ASLR – Addres Space Layout Randomization)
- Randomização do endereço de funções e dados a cada execução do programa

A cada vez que um processo do programa vai ser executado, ele é carregado numa posição diferente da memória. Porque o atacante descobre onde existe endereço de retorno, com isso randomizado, o atacante não obterá essa informação facilmente.

• Ataques de desvio de fluxo sem obtenção de controle
- visa apenas obtenção de informação

Ataques por string de formato:
Strings de formato %s (string), %d (inteiro), %n (nada)...

• Ataques por cadeias de caracteres de formato se utilizam de uma funcionalidade do comando printf que permite escrever na memória.

printf(“Ola %nmundo\n”, &i); /* o %n armazena em i */

Ola mundo
i=4

Ataques por ponteiros pendentes:
Alocação dinâmica de memória em C: malloc e free
Ponteiro pendente: acesso à uma área já liberada

- Permite acesso ao heap de outros processos

- técnica de heap feng shui: modificar o nível de autorização no sistema

Ataques de retorno à libc

Ataques por dereferência de ponteiro nulo:

Ataque por dereferência de ponteiro nulo ataca uma falha de versões antigas de SOs

- algumas rotinas do SO (modo núcleo) chaveiam para modo usuário, usando o espaço de endereçamento do processo.

- Ex: Linux 32 bits: 3 G para usuários e 1 G para SO

• Chamada de função cujo endereço ainda não está definido
(ponteiro NULL → página 0)

Como não há código na página zero:
- o programa do usuário (ou o SO) será derrubado.
- mmap permite mapear um código em endereço específico:

Usuário escreve um shell e mapeia para a página 0 acesso ao shell com privilégio de administrador

SHELL com endereço fixo

Ataques por transbordamento de inteiro:

A linguagem C não faz tratamento de erro de transbordamento (overflow)
- Ex: 40000 x 40000 = 4096 (inteiros de 16 bits, sem sinal)
- 40000 x 40000 = 5F5E1000hex =
= 0101 1111 0101 1110 0001 0000 0000 0000

Chamada de funções conhecidas com valores grandes, para alocar (muito) mais memória do que a necessária

- Cria as condições para ataque de transbordamento de buffer

Ataques por injeção de comando:

Utiliza parâmetros do programa para a execução de comandos do sistema

TOCTOUs (corrida crítica)

Exemplos de contramedidas que tentam evitar esses ataques: canários de pilha, prevenção de execução de dados e randomização de layout de espaço de endereçamento.

Primeira coisa que um atacante externo irá tentar fazer: Vai tentar achar quais as portas de serviço que tem num determinado sistema e vai tentar posteriormente acessar essas portas e serviços.

Ataques internos

Executados por programadores e outros empregados da empresa executando o computador a ser protegido ou produzindo um software crítico. Eles diferem dos ataques externos, pois as pessoas de dentro do sistema têm um conhecimento especializado e acesso que as pessoas de fora não têm.

Bomba lógica: fragmento de código escrito por um dos programadores (atualmente empregado) da empresa e secretamente inserido no sistema de produção. Se o programador for subitamente despedido e fisicamente removido do local sem aviso, no dia seguinte (ou semana seguinte) a bomba lógica não será alimentada com sua senha diária e detonada. Detonar poderia envolver limpar o disco, apagar arquivos de forma aleatória, cuidadosamente fazer mudanças difíceis de serem detectadas em programas fundamentais, ou criptografar arquivos essenciais.

Back door (porta dos fundos): problema criado por um código inserido no sistema por um programador para driblar alguma verificação normal. Uma maneira para as empresas evitarem back door é ter revisões de código como uma prática padrão.

Começou com a ideia de um usuário interno desse sistema colocar essa “porta” para poder corrigir uma falha, por exemplo, em sua casa.

Mascaramento de login: técnica em que o atacante é um usuário legítimo que tenta conseguir as senhas de outras pessoas, geralmente empregada em organizações com muitos computadores públicos em uma LAN usados por múltiplos usuários.

Ideia: Desenvolver uma interface igual a de login de um sistema, roubar o login dos usuários para o atacante posteriormente usá-las para um ataque.

Malwares (“Software do mal”)

Cavalo de Troia: qualquer malware escondido no software ou em uma página da web que as pessoas baixam voluntariamente. Quando o programa é inicializado, ele chama uma função que escreve o malware para o disco como um programa executável e o inicializa. Esse tipo de ataque não exige que o autor viole o computador da vítima. A própria vítima faz todo o trabalho!

Intuito: Roubar uma informação, permitir um acesso remoto, pode ser usado as máquinas infectadas para fazer um ataque de negação de serviço, apagar arquivos...

Vírus: um programa que pode reproduzir-se anexando o seu código a outro programa, de maneira análoga à reprodução dos vírus biológicos. Uma vez instalado na máquina da vítima, o vírus fica dormente até o programa infectado ser executado. Uma vez inicializado, ele começa infectando outros programas na máquina e então executando sua carga útil. Tipos: vírus companheiro, de programa executável, de memória, de inicialização, de driver de dispositivo e de código-fonte.

Ex: Vírus de BIOS que apaga informação em memória flash, ela é quem faz a inicialização da máquina. (Vírus conhecido como chernobyl)

Muito difícil fazer um vírus “genérico”, teria que passar por um compilador, gerar o código executável e esse código ser interpretado pelo hipervisor então deve ser feito para cada SO específico.

A efetividade do vírus depende muito da abrangência do SO, a maioria das máquinas do mundo são Windows.

Worm (verme): um programa que se auto replica, explora erros e replica a si mesmo em segundos em toda máquina que ganha acesso.

A diferença entre Vírus e Worm é que o vírus precisa da “ajuda do usuário", ele precisa ser executado de maneira explícita pelo usuário (ex:clicando no link de email e baixando).

Spyware: software que, carregado sorrateiramente no computador sem o conhecimento do dono, executa em segundo plano fazendo coisas sem o consentimento do usuário como recolher informações sobre o usuário e sobre os seus costumes na Internet. Pode infectar o computador por um cavalo de Troia, via contágio por contato ou por controles ActiveX.

Ex de uso: identificar o que o usuário está fazendo e direcionar anúncios comerciais quando o usuário começar a web.

Ele pode mudar informações do seu navegador de internet. Google -> Yahoo
Enviar informação para outro lugar

Rootkit: programa ou conjunto de programas e arquivos que tenta ocultar sua existência, mesmo diante de esforços determinados pelo proprietário da máquina infectada de localizá-lo e removê-lo. Normalmente, o rootkit contém algum malware que está sendo escondido também. É muito difícil ser detectado.

Ex: feito pela Sony que impediu gravar CD

Defesas

Há uma série de maneiras como os sistemas podem defender- se. A melhor estratégia é a defesa em profundidade, usando múltiplas técnicas.

Firewalls: existem os de hardware e os de software. São configurados com regras descrevendo o que é **permitido entrar e o que é permitido sair**. O proprietário do firewall pode mudar as regras, comumente através de uma interface na web.

O Firewall de hardware é mais eficiente do que fazer toda uma inspeção de firewall por software.

Filtragem de pacotes, pode analisar o conteúdo de pacotes e permitir identificar assinaturas de ataques. Em caso de dúvida, destaca o pacote.

Varreduras de vírus: uma vez que um programa antivírus tenha sido instalado na máquina, a primeira coisa que ele faz é varrer todos os arquivos executáveis no disco procurando por qualquer um dos vírus no banco de dados de vírus conhecidos. O antivírus também pode fazer a verificação de integridade e a comportamental.

Ex: antivírus conhece a assinatura digital de um programa, então ele a compara com o programa do seu computador, se for diferente, sabe que não é o programa original

2 fases: Identificar o vírus e identificar a mudança que foi feita e retirar essa mudança.

Assinatura de código: baseada na criptografia de chave pública.
Você pode conferir o hash do arquivo com o do fabricante para ver se é original.

Encarceramento: técnica utilizada para verificar se o software está comportando-se corretamente, pois a assinatura prova somente de onde ele veio, não o que ele faz.

Essa defesa é do próprio SO. Toda chamada do sistema é passada pelo "carcereiro" antes de passar para o SO. Usado para programas novos na máquina.

Sistemas de detecção de intrusão (IDS): há dois tipos básicos, um focado em inspecionar pacotes de rede que chegam e outro que procura por anomalias na CPU. Na detecção de intrusão baseada em modelo estático, o carcereiro tem de conhecer o gráfico de chamada do sistema. Muitos IDSs fazem uso de um conceito chamado chamariz (honeypot), uma

armadilha colocada para atrair e pegar crackers e malwares. Utilizam servidores falsos para deixar ataques acontecer para aprender o padrão de ataque.

Ex: Não deixa passar ataque de negação de serviço, pois identifica o padrão e joga fora os pacotes. (isso no servidor real)

Encapsulamento de código móvel: os métodos mais conhecidos são caixa de areia (confina cada applet a uma faixa limitada de endereços virtuais implementados em tempo de execução) e interpretação (applets são executados interpretativamente de forma que não possam assumir o controle real do hardware).

São executados no cliente. Ex: validar cpf

Segurança em Java: a linguagem Java é tipificada e segura, no sentido de que o compilador rejeitará qualquer tentativa de usar uma variável em uma maneira que não seja compatível com seu tipo.

- Programas de Java são compilados para um código binário intermediário chamado byte code de JVM (Java Virtual Machine).

- Quando um applet chega, ele é executado através de um verificador de byte code de JVM que confere se o applet obedece a determinadas regras. Se o applet passa por todos os testes, ele pode ser seguramente executado sem medo de que ele vá acessar outra memória que não seja a sua.