Unidade 1: Introdução à Segurança da Informação

Prof. Daniel Caetano

Objetivo: introduzir o conceito de Segurança da Informação, indicar as metodologias existentes e em que consiste uma Política de Segurança da Informação.

Bibliografia: FERREIRA, 2003; FERREIRA E ARAÚJO, 2006.

INTRODUÇÃO

Conceitos Chave:

- Problemas:
    * Como proteger a empresa de um ataque de um cracker?
    * Como transmitir dados com segurança?
    * Como evitar infecção por vírus?
- Milhares de sistemas são atacados por ano
    * Bem protegidos => ataque fracassa
    * Medianamente protegidos => ataque de sucesso, atacante identificado
    * Mal protegidos => ataque tem completo sucesso, atacante livre

Com o avanço das tecnologias de informação e transmissão de dados, maior tem sido a preocupação com a proteção dados de uma empresa, sejam eles digitais ou não.

Esta preocupação não é sem motivo: todos os anos são milhares de ataques à informação das mais diversas empresas, mas, graças às estratégias e técnicas da segurança da informação, a maior parte deles não obtém êxito.

A fonte de ataques mais comumente citada pela imprensa são os vírus, *worms* e *trojans*, softwares maliciosos elaboratos por *crackers* para causar danos a um equipamento ou para dar-lhes o controle de um equipamento. Como proteger uma empresa deste tipo de ataque? Como permitir uma operação segura dos sistemas de informação sem prejudicar demasiadamente sua facilidade de operação?

A resposta para estas perguntas é bem menos complexa do que parece, e as alternativas serão analisadas neste curso.

1. SEGURANÇA DA INFORMAÇÃO

Conceitos Chave:

- Informação na Empresa
  - * Ativo
  - * Deve ser protegida
- Razões para proteção
  - * Garantir continuidade dos negócios
  - * Maximizar retorno de investimentos/oportunidades
  - * Minimizar transtornos
- Necessidade
  - * Informação está em constante risco
  - * Originalmente: segurança física
  - * Atualmente: segurança física e lógica
- Objetivos:
  - * Confidencialidade
  - * Integridade
  - * Disponibilidade
- Praticidade x Segurança
  - * P = 1/S
- Custos
  - * Segurança aumenta custos, mas é necessária
  - * Como determinar os critérios de segurança?
    - Que informação proteger?
    - Contra o quê/quem?
    - Quais as ameaças?
    - Relação importância/nível de proteção
    - Recursos Disponíveis (financeiros e pessoais)
    - Expectativas dos clientes
    - Conseqüências das falhas de segurança

- Política de Segurança!

O conceito de "segurança da informação" dentro de uma empresa é relativamente intuitivo: refere-se à proteção das informações daquela empresa. Embora o conceito de informação seja também intuitivo, a razão pela qual ela deve ser protegida nem sempre é clara. O conceito fundamental aqui é: **informação é um ativo** e, portanto, deve ser protegida.

As razões principais para se garantir a segurança da informação são:

- **Garantir continuidade dos negócios**, já que muitas informações são necessárias para que os negócios ocorram.

- **Maximizar retorno de investimentos/oportunidades**, já que, nas informações, é possível identificar aspectos em que um negócio pode ser melhorado.
- **Minimizar transtornos**, uma vez que algumas informações sigilosas não devem ser divulgadas em público.

Mas... existe sentido em tão grande preocupação com a segurança da informação? Sim, já que **a informação está em constante risco**. E a segurança da informação vem se tornando mais complexa ao longo do tempo. Se **no início** havia apenas a necessiade de **segurança física**, pois os documentos eram todos na forma de papéis, **hoje** as informações estão todas em computadores, muitas vezes interligados em redes, cuja **proteção** é bem mais complexa e **envolve aspectos físicos e lógicos**.

O objetivo de toda a segurança da informação é a garantia de:

a) Confidencialidade: só pessoas autorizadas possuem acesso às informações
b) Integridade: as informações se mantém corretas e completas
c) Disponibilidade: as informações estão disponíveis sempre que necessário

Valendo lembrar que, em termos de segurança da informação, vale a regra:

Praticidade = 1/Segurança

Ou seja: quanto mais "prático" é o acesso a uma informação, menos protegida ela estará, e vice versa.

Entretanto, qualquer tipo de **segurança causa custos adicionais**. A segurança da informação não é diferente. Para que seja possível **garantir os objetivos** da segurança da informação de uma maneira fundamentada **e não incorrer em custos adicionais excessivos**, é preciso **responder a algumas questões**.

a) Que informações devem ser protegidas?
b) Contra o quê/quem?
c) Quais são as ameaças?
d) Qual a importância de cada recurso e o nível de proteção desejado?
e) Quais os recursos (financeiros e pessoal) disponíveis?
f) Quais as expectativas dos clientes quanto à segurança?
g) Quais as conseqüências se houver vazamento de informações?

2. Política de Segurança da Informação e Seu Desenvolvimento

Conceitos Chave:

- Empresa deve possuir Políticas, Padrões e Procedimentos
    * Políticas: documento de alto nível, dá uma direção a seguir
    * Padrões: documento detalhado, indica medidas de controle
    * Procedimentos: descrição passo a passo para determinadas funções
- Objetivo: ressaltar regras da segurança corporativa
- Como?
    * Classificação da Informação
        + Classificação: pública x interna x confidencial x secreta
        + Diferenças: Acesso Externo, Interno e Integridade
        + Documentos em papel: classificação impressa (todas as páginas)
        + Dificuldade de classificação => confidencial
    * Responsabilidades e Penalidades aos usuários
        + Dados devem ter proprietários, que devem classificá-los
        + Configurar acesso à informação
    * Classificação de Ameaças
    * Critérios de Segurança (leis externas!)
    * Monitoramento contínuo e auditoria
- Tipos de Políticas
    * Regulatória (lei) x Consultiva (op.básica) x Informativa (penas)
- Conteúdo mínimo de Política:
    * Especificação da política (público alvo e procedimentos)
    * Declaração da Alta Administração (endosso)
    * Autores/Patrocinadores (responsáveis)
    * Referências (indicações a regulamentos internos)
    * Procedimentos de Exceções (como requisitar exceções)
    * Procedimentos de Mudanças (como requisitar mudanças)
    * Punições
    * Datas (publicação, validade, revisões)
- Divulgação
    * Material promocional
    * Treinamentos periódicos
        + Custoso: treinar TODOS apenas no que precisam ser treinados.
- Auditorias contínuas
    * Análise do sistema e suas saídas
        + Verificar se tudo funciona corretamente
    * Objetivo:
        + Identificar irregularidades
        + Evitar grandes problemas

Toda empresa precisa possuir um conjunto de políticas, padrões e procedimentos para garantir seu correto funcionamento. No caso da segurança da informação, o objetivo de cada um destes documentos é:

- Política: Documento de alto nível que direciona a segurança da informação;
- Padrões: Documento detalhado, indicando as medidas necessárias para o controle;
- Procedimentos: Descrição passo-a-passo para atingir a objetivos esperados.

Assim, o objetivo geral destes documentos é especificar claramente as regras de segurança corporativa. Para atingir a este objetivo é necessário definir:

- Classificação da Informação em Níveis de Segurança
- Responsabilidades e penalidades aos usuários da informação
- Classificação das ameaças
- Critérios de segurança, baseados também em leis externas a serem respeitadas
- Monitoramento contínuo e auditoria

Exemplos de Classificação:

Classe 1 - Informação Pública:
- Divulgação externa não prejudicial;
- Integridade da informação não é vital.

Classe 2 - Informação Interna:
- Acesso externo deve ser evitado;
- Integridade da informação importante, mas ainda não vital.
- Acesso interno a usuários selecionados.

Classe 3 - Informação Confidencial:
- Acesso externo proibido, divulgação interna como confidencial;
- Integridade de dados é vital;
- Acesso interno restrito.

Classe 4 - Informação Secreta ("Top Secret"):
- Acesso não autorizado pode ter conseqüências graves;
- Integridade da informação é vital;
- Acesso restrito a poucas pessoas.

Resumo das Atribuições dos Usuários

- **Todo dado deve ter um proprietário**;
- É **obrigação do proprietário classificar** a informação (baseado na política e lei);
- Na **dificuldade de classificar**, a informação deve ser marcada como **confidencial**;
- É **responsabilidade do proprietário configurar o acesso** à informação;
- O **proprietário é responsável por seus dados**, baseado na classificação da info.;
- **Documentos em papel** devem ter sua **classificação impressa** em **todas as páginas**.

Tipos de Políticas (de prioridade mais alta para mais baixa):

- Regulatória - garante conformidade com a norma e a lei;
- Consultiva - fornece conhecimentos básicos das operações dos funcionários;
- Informativa - oferece informações adicionais (com penas).

Conteúdo (mínimo) de uma Política de Segurança:

- **Especificação da política**: define claramente o público alvo e os procedimentos.
- **Declaração da Alta Administração**: endosso do principal executivo da empresa.
- **Autores / patrocinadores da política**: indica os responsáveis pela criação.
- **Referências**: indicações a regulamentos internos referentes a esta política.
- **Procedimentos de exceções**: indica como requisitar exceções à política.
- **Procedimentos de mudanças**: indica como requisitar mudanças na política.
- **Punições**: quais as sanções a quem violar a política.
- **Datas**: publicação, validade e revisões.

Deve sempre ser criado **material promocional**: todos devem tomar conhecimento, seja na data da publicação, seja quando entram novos funcionários. Treinamentos periódicos.

Além disso, é importante ressaltar que o processo de **auditoria** deve ser contínuo: é preciso acompanhamento constante para identificar irregularidades o mais cedo possível, de forma a evitar que os problemas cresçam.

Auditoria é o processo de análise do sistema e de suas saídas, avaliando se ele está funcionando como o esperado e se seus usuários estão agindo dentro das normas definidas pela política vigente. Existem diversos processos de auditoria, cada um deles visando objetivos específicos.

Treinamento: é sempre um processo custoso! Portanto, deve-se limitar o treinamento de cada funcionário aos tópicos que ele, de fato, precisa ter conhecimento. A tabela abaixo é uma sugestão (FERREIRA, 2003):

Matriz de Treinamento

Área de Treinamento	*Noções Básicas Segurança*	*Gestão e Planejamento: Segurança da Informação*	*Políticas e Procedimentos de Segurança*	*Planos de Contingência*	*Gestão Gerenciamento de Mudanças*
Executivos					
Diretores de Informática					
Auditoria e Security Officer					
Gerentes de Desenvolvimento de Sistemas					
Usuários Finais					

Nível / Tipo do treinamento

Conscientização
Política
Implementação
Execução

3. Características e Benefícios de uma Política de Segurança

Conceitos Chave:

- Características Fundamentais
    * Ser verdadeira e coerente
    * Complementada com recursos
    * Válida para todos e com comprometimento da alta-organização
    * Simples
- Benefícios
    * Evidencia:
        + a importância das informações
        + envolvimento da alta administração
        + responsabilidades de funcionários e colaboradores
    * Estabelece padrões para manutenção da segurança
    * Formaliza procedimentos (e obriga a criação de novos)
    * Evita vazamentos e aumenta a segurança nos negócios

Características desejáveis para uma política de segurança da informação:

- **Ser verdadeira**: coerente com objetivos da organização;
- **Ser complementada com recursos**: disponibilidade de equipamentos e pessoal;
- **Ser válida para todos**: do presidente ao estagiário;
- **Ser simples**: qualquer um deve compreender;
- **Comprometimento da alta-organização**: assinada pelo presidente (autoridade).

Benefícios diretos alcançados:

- **Evidencia a importância das informações**;
- **Evidencia o envolvimento da alta-administração**;
- **Evidencia a responsabilidade de funcionários e colaboradores**;
- **Estabelece padrões para a manutenção da segurança**;

- **Formalização de procedimentos** (que se tornam rotina)
- **Implementação de novos procedimentos**
- **Prevenção de vazamento da informação**
- **Maior segurança nos negócios**

4. Metodologias de Segurança da Informação

Conceitos Chave:

- CobiT (Internacional)
- BS7799 - ISO/IEC 17799 - ABNT NBR ISO/IEC 17799 (Inglaterra e Brasil)
- SAS70 (EUA)

Existem diversas metodologias de segurança da informação. O objetivo destas metodologias são orientar o desenvolvimeno de uma **política de segurança da informação**, que são as normas que devem ser seguidas em uma empresa para garantir a segurança da informação. Dentre elas, três das mais importantes metodologias são:

- CobiT (Control Objectives for Information and Related Technology): criada pela ISACA (Information Systems Audit and Control Association), ela define as seguintes necessidades de garantia em uma política de segurança:

- Efetividade
- Eficiência
- Confidencialidade
- Integridade
- Disponibilidade
- Conformidade
- Credibilidade

- BS7799 - ISO/IEC 17799 - ABNT NBR ISO/IEC 17799: criada pelo BSI (Britsh Standard Institute), define as necessidades de garantia em uma política de segurança:

- Confidencialidade
- Integridade
- Disponibilidade
- Confiabilidade

- SAS70: criada pelo AICPA (American Institute of Certified Public Accountants), ela não define/especifica objetivos ou atividades de controle, mas define um relatório de especificação de nível de segurança de serviços.

5. Bibliografia

FERREIRA, F. N. F. **Segurança da Informação**. Rio de Janeiro: Ciência Moderna, 2003.

FERREIRA, F. N. F; ARAÚJO, M. T. **Política de Segurança da Informação: Guia Prático de Elaboração e Implementação**. Rio de Janeiro: Ciência Moderna, 2006.

Unidade 2: Controle de Acesso à Informação

Prof. Daniel Caetano

Objetivo: apresentar os conceitos básicos de segurança lógica e controle de acesso e apresentar os cuidados de instalação e logging necessários para manutenção da segurança lógica.

Bibliografia: FERREIRA, 2003.

INTRODUÇÃO

Conceitos Chave:
- Problemas:
  * Como proteger informações de acesso indevido?
  * Como detectar se um sistema/ambiente foi invadido?
  * Como detectar possíveis responsáveis?
- Controle de Acesso => Registro de Acesso
  * Segurança Física
  * Segurança Lógica

Mesmo as empresas mais bem protegidas estão sujeitas a serem invadidas, seja esta uma invasão de suas instalações físicas, seja esta uma invasão de seus sistemas computacionais (lógicos). E, ainda que uma empresa não tenha sido invadida, ela pode estar sendo alvo de ataques que objetivem uma invasão.

Mas como proteger uma informação, para garantir sua confidencialidade, integridade e disponibilidade? E quando esta proteção falha, como pode o profissional de segurança identificar estas tentativas de invasão ou mesmo os responsáveis por elas?

O primeiro passo, para proteger as informações, é *controlar* quem pode acessar aquela informação. Em outras palavras, deve-se ter um mecanismo que impeça que pessoas não autorizadas acessem uma informação, permitindo àquelas que tiverem acesso. Este tipo de mecanismo é chamado de controle de acesso.

Caso o sistema sofra tentativas de invasão ou mesmo sofra uma invasão de fato, o segredo para que seja possível identificar o invasor e quais foram as alterações realizadas é a correta manutenção de um registro de acesso. O registro de acesso permite que seja possível realizar uma análise, uma *auditoria*, para encontrar as respostas desejadas. Entretanto, um registro de acesso só é útil para o profissional de segurança se todos os acessos forem registrados.

Assim, para garantir que todos os acessos sejam registrados é necessário que este registro seja feito pelo controle de acesso ao sistema, podendo este ser uma barreira física ou lógica, que garanta que todo acesso a um local ou a um sistema seja devidamente registrado.

Quando o controle de acesso é uma barreira física para impedir o acesso físico a equipamentos, documentos e localidades, fala-se em segurança física de um sistema. Quando o controle de acesso é uma barreira lógica para impedir o acesso a dados e informações, fala-se em segurança lógica de um sistema.

Em outras palavras, a segurança física trata do acesso físico ao meio portante dos dados, como os papéis de documentos impressos e aos equipamentos que armazenam dados. A segurança lógica, por outro lado, trata da informação em si, de seu conteúdo.

Ambos os tipos de segurança exigem um adequado controle de acesso; no momento o foco será a segurança lógica, mas futuramente serão apresentados mais detalhes sobre segurança física.

1. CONTROLE DE ACESSO LÓGICO

Conceitos Chave:

- Segurança => Controle de Acesso
- Envolve: usuário + recurso
- Quem pode o quê?
    * Padrão: tudo é proibido, a menos que permitido
- Objetivos:
    * Proteger informações e transações
    * Monitoramento de acesso
- Procedimento:
    * Identificação: saber quem o usuário diz ser
        + UserID
    * Autenticação: comprovar que o usuário é quem diz ser
        + Password, Biometria, Cartão...
        + Algo que o usuário tem + Algo que o usuário sabe
- Permite responsabilização de usuários
- Dificuldades
    * UserIDs podem ser facilmente obtidos
    * Cartões podem ser roubados, mas são preferíveis
    * Senhas: longas, não óbvias, e não usadas em outros sistemas
        + Podem ser, ainda assim, roubadas.
    * Biometria: custo, variações
- Medida adicional: limitar tentativas de logon fracassadas

A segurança da informação, seja em qual nível for, sempre implicará em controle de acesso, seja ele físico (portões, muros etc.) ou lógico (login, logging etc.). O controle de acesso lógico deve sempre englobar o recurso que se pretende proteger e o usuário a quem se pretende dar privilégios de acesso.

De forma geral, um primeiro passo para uma boa política de controle de acesso é: **tudo é proibido a menos que expressamente permitido**. Isso está de acordo com a regra *praticidade = 1/ segurança* que foi mencionada anteriormente.

Assim, claramente o controle de acesso lógico reduz a praticidade do sistema. Mas qual é o objetivo central do controle de acesso? São, basicamente, dois os pontos de interesse:

- Proteger informações e transações de usuários não-autorizados;
- Monitoramento do acesso a recursos críticos para a empresa.

1.1. Procedimento do Controle de Acesso

O controle de acesso é feito, usualmente, através de dois processos: identificação e autenticação. O primeiro deles (identificação) permite ao sistema reconhecer qual usuário pretende acessar a informação, para identificar quais são suas permissões. O segundo (autenticação) permite ao sistema comprovar a identidade do usuário, visando impedir que uma pessoa se passe por outra. O processo completo de identificação e autenticação se chama "*logon*" ou "*login*".

> O termo "*logon*" ou "*login*" significa aproximadamente algo como "criar uma entrada no *log*". *Log* é o nome em inglês para arquivos de registro de acesso. Mais detalhes sobre o arquivo de *log* serão apresentados mais adiante.

Usualmente, a identificação é feita por algum tipo de informação única para cada usuário, como um *número de funcionário* (*userid*, que deve ser único), que pode ser digitado manualmente ou fornecido através de um cartão, por exemplo.

Após a identificação, o sistema deve fazer a autenticação, para ter certeza de que o usuário que realiza o acesso é realmente quem diz ser. Assim, o usuário deve fornecer uma senha (*password*), por exemplo. Outras formas de autenticação são os sistemas de medição biométrica (como timbre de voz, exame de retina, leitura de digital etc.). É comum se afirmar que o *login*, ou seja, a identificação/autenticação, é feita por uma combinação entre algo que o usuário tem e algo que ele sabe, associado à sua identificação de usuário.

> Biometria é uma área do conhecimento que estuda medidas únicas de seres vivos - em especial o ser humano, visando a obtenção de medidas que sejam únicas de indivíduo para indivíduo, de forma que esta medição possa ser usada para identificação e/ou autenticação de sistemas.

1.2. Dificuldades de Alguns Métodos de Identificação/Autenticação

Por permitir que todas as ações do usuário dentro do sistema sejam monitoradas e registradas, o *logon* é muito importante, pois permite que sejam apuradas as responsabilidades em casos de problemas de segurança. Entretanto, o *logon* precisa ser resistente à tentativas de invasão; caso um invasor utilize um *logon* de algum funcionário, a responsabilidade recairá sobre este funcionário.

Neste panorama, algumas tecnologias possuem alguns problemas. Cartões de identificação podem ser perdidos, mas *userid*s que precisem ser digitados podem ser obtidos por outras pessoas ainda mais facilmente. Por esta razão, os cartões de identificação costumam ser a forma mais bem aceita de se identificar num sistema.

Para diminuir os riscos de uma tentativa de invasão, também a autenticação exige algumas providências de segurança. Usualmente são usadas senhas, o que é um bom método de autenticação, caso a senha não seja curta, óbvia e seu dono não a tenha escrito em algum lugar (incluindo aqui não tê-la usado em outros sistemas). Entretanto, senhas podem ser facilmente roubadas, o que tem levado a muitas empresas adotarem senhas conjuntamente com sistemas de medição biométrica para a autenticação de usuário, já que a medição biométrica costuma ser menos passível de falsificação, embora tenha seus problemas com falsos negativos e seu custo elevado.

De qualquer forma, usualmente são tomadas previdências adicionais com relação ao processo de autenticação, para frustar qualquer tentativa de invasão: o número de tentativas incorretas de *logon* costuma (e deve) ser limitado. Usualmente considera-se que três tentativas incorretas é um bom valor para bloquear uma dada conta de usuário.

> Três tentativas é um valor considerado razoável para serviços de importância, que são usados com relativa freqüência e que, em geral, envolvem responsabilidade ou movimentações financeiras. Caso não sejam estas as características da aplicação, é razoável ampliar o número de tentativas para 9 ou 10 antes de bloquear um usuário.

2. GERENCIAMENTO DE USUÁRIOS E CONTROLE DE LOG

Conceitos Chave:
- Auditoria => apuração de falhas e responsabilidades
- Análise de Logs: muito importante
  * Logs precisam registrar TUDO que o usuário fez
  * Logs precisam registrar QUEM fez tudo
- Administradores de Sistema
  * Implementar cadastro/autorizações de usuários
  * Equipe de Administradores... tomar cuidados
    + Comunicação
    + Controle de Alterações
    + UserIDs diferentes para cada administrador
  * Gerenciamento de Logs
    + Sincronia de Relógios da Rede
    + Armazenamento de Logs - protegido dos usuários
    + Rotação de Logs (Backup após análise)
  * Logs: Locais x Remotos x Mistos
  * Auditorias Freqüentes

Normalmente, se a segurança do sistema for bem planejada e for bem administrada, a maioria das falhas de segurança e identificação de responsabilidades são possíveis através de auditoria. Um dos principais instrumentos de auditoria são os arquivos de registro de acesso e uso do sistema, os chamados *logs*.

Em muitos casos, a análise dos *logs* é o único instrumento que possibilita essas identificações e, por esta razão, é de extrema importância que os *logs* fiquem protegidos da ação direta dos usuários. É preciso, também, garantir que os *logs* registrem de fato tudo que foi feito e qual foi o usuário que realizou cada uma das ações.

Em geral, para que isso seja conseguido, existe uma pequena equipe de administradores de sistema que cuidam de implementar o cadastro e as autorizações a usuários selecionados por níveis superiores. Para que esta equipe funcione bem, algumas regras são necessárias:

- Comunicação (registro de alterações feitas, problemas encontrados, etc.)
- Controle de Alterações (registro de autoria das modificações, para contato futuro)
- Um *userid* para cada administrador.

Considerando que o gerenciamento de usuários é adequado, também é função dos administradores cuidar do gerenciamento de logs. O primeiro e mais importante fato é cuidar da sincronia dos relógios das diferentes máquinas, como já foi citado anteriormente, de forma a permitir o rastreamento das ações dos usuários que ocorrem seqüencialmente nas diversas máquinas.

A segunda medida é cuidar do adequado armazenamento dos logs. Uma primeira medida é garantir uma partição especial para os logs locais, para evitar os problemas já citados. Outra medida é a rotação dos logs, transportando logs mais antigos (já revisados) para mídias de armazenamento como DVDs, liberando espaço para logs mais recentes.

Em alguns sistemas, entretanto, a manutenção de logs locais não é adequada, dado que estão sujeitos a serem destruídos. Nestes casos, é interessante usar um equipamento dedicado a armazenar o log de tudo que é feito em outras máquinas. Esta medida tem algumas limitações, como banda de transmissão, problemas para registrar ações do usuário local caso a rede caia etc. Algumas vezes é usado um misto de ambos os sistemas.

É importante ressaltar que os logs devem ser monitorados periodicamente, e a freqüência com que isso é feito deve ser tão maior quanto mais crítico for o acesso ao sistema. Qualquer evento estranho deve ser devidamente investigado, de forma a averiguar sua origem e normalidade.

3. INSTALAÇÃO VISANDO SEGURANÇA DA INFORMAÇÃO

Conceitos Chave:

- Antes da criação de usuários e permissões: Instalação Segura
- Revisar Política de Segurança
    * Quais são os mecanismos de segurança necessários?
        + Backups? RAID? Logs?
    * Quais serviços/softwares são necessários em cada equipamento?
- Primeiro Passo: Particionamento de Disco
    * Negligência => dificuldade de manutenção
    * Garantir espaço para:
        + Arquivos de Sistema (incluindo Swap)
        + Dados do Usuário
        + Log de Acesso e Uso
        + Páginas Web Públicas
        + Páginas Web Privadas / Confidenciais
    * Evita que cracker entulhe disco
        + Garante espaço para o swap e logs
- Segundo Passo: Instalação Mínima
    * Instale só o necessário
        + Desative recursos não usados
        + Instale todas as atualizações ANTES de ligar equipamento à rede
- Terceiro Passo: Evite concentração de recursos
    * Não coloque todos os scanners, impressoras etc. no mesmo micro
- Quarto Passo: Configure a sincronia de relógios da rede
- Quinto Passo: Documente a Instalação

Para que o controle de acesso seja facilitado, é fundamental que algumas providências sejam tomadas muito antes da configuração de usuários e permissões: é preciso que o sistema como um todo seja instalado visando a segurança.

O primeiro passo para isso é revisar a política de segurança da informação da empresa e verificar quais são os mecanismos de segurança necessários. Com isso é possível definir quais são as necessidades de cada departamento e funcionários específicos e, com isso, identificar o objetivo de cada componente (computador) a ser instalado.

Antes de instalar software para atender aos objetivos do sistema, porém, uma medida deve ser avaliada com cuidado: o particionamento de discos. Muitas vezes o particionamento de discos é negligenciado, trazendo grandes dificuldades para a manutenção da segurança do sistema. É adequado particionar o disco para garantir espaço para algumas informações:

- **Arquivos do sistema** (incluindo **swap**, evitando *traps* do sistema);
- **Dados dos usuários** (limitando o espaço disponível para estes tipos de arquivo);
- **Logs de acesso e uso** do equipamento (registrando o que foi acessado na máquina);
- **Páginas Web públicas** (se for um servidor de páginas ineternet);
- **Páginas Web privadas/confidenciais** (se for um servidor de páginas intra/extranet).

Este procedimento evita que um eventual invasor crie, por exemplo, um arquivo de dados tão grande que a atualização de segurança automática do sistema falhe por falta de espaço, ou que suas ações não sejam registradas por falta de espaço para crescimento do arquivo de log.

Finalmente, definidos os objetivos de cada componente do sistema e o particionamento adequado de seu disco, deve-se realizar a instalação mínima para que este sistema cumpra seus objetivos, preferencialmente instalando todas as atualizações de segurança de todos os software instalados antes de conectá-lo a qualquer rede (incluindo aqui os anti-vírus, sobre os quais falaremos em aulas posteriores). Esta instalação mínima também implica na desativação de todos os recursos que não sejam necessários ao usuário.

Evite também concentrar todos os recursos disponíveis na rede (discos compartilhados, impressoras, scanners, plotters etc.) em um único computador; se eles estiverem todos em um único equipamento e este equipamento tiver problemas, tudo sairá do ar ao mesmo tempo.

Lembre-se de que um sistema de sincronia de relógios entre as máquinas é essencial para que a análise de logs destas faça sentido. Se os relógios dos equipamentos não estiverem ajustados de forma idêntica, não é possível acompanhar a progressão de uma eventual invasão (ou tentativa de invasão) de forma a identificar a falha de segurança e sua possível correção.

Por fim, não deve ser esquecida a documentação da instalação, incluindo espaços para registro de modificações. Estas informações podem facilitar bastante a identificação de

problemas, como por exemplo um bug ou falha de segurança que surgiu com a instalação de um novo software ou sua atualização.

4. EXERCÍCIO

Em grupos de 3 ou 4 alunos, respondam às perguntas abaixo:

1. Você é administrador do sistema de uma empresa e, ao se logar na intranet, percebe que ela foi atacada por um 'defacer' (um ataque em que substituem a sua homepage por uma outra página, em geral com protestos políticos). Quais são as ações que você toma?

2. Em um banco os cartões de funcionários são todos identificados através do nome do funcionário (escrito em relevo no cartão); para autorizar operações especiais, entretanto, é preciso de uma senha, além do cartão, sendo que a senha é diferente para cada funcionário. Critique este sistema.

3. Comente sobre a importância da sincronia entre os relógios dos diferentes computadores de uma rede.

4. Qual a melhor estratégia: unificar todos os recursos compartilhados em uma única máquina servidora ou possuir vários servidores diferentes, cada um compartilhando um recurso? Por quê?

5. Sua equipe recebeu os seguintes logs para analisar:

COMP1:
20/01/2008 - 22:17:55 - IP: 200.178.95.16 - ddamasio logged in.
20/01/2008 - 22:19:30 - IP: 200.192.67.112 - jsoldi logged in.
20/01/2008 - 22:54:17 - IP: 200.178.95.16 - ddamasio logged out.
20/01/2008 - 22:55:32 - IP: 200.192.67.112 - jsoldi logged out.
20/01/2008 - 23:20:13 - IP: 163.102.100.17 - cabrahm logged in.
21/01/2008 - 00:10:11 - IP: 163.102.100.17 - cabrahm logged out.

COMP2:
20/01/2008 - 22:07:30 - IP: 200.178.95.16 - printed file c:\work\biometrics.doc
20/01/2008 - 22:53:30 - IP: 200.192.67.112 - deleted file c:\work\biometrics.doc

COMP3:
20/01/2008 - 23:22:17 - IP: 163.102.100.17 - copied c:\work\test.c to c:\shared.

COMP4:
20/01/2008 - 23:25:33 - IP: 163.102.100.17 - copied \\COMP3\shared\test.c to c:\work.
20/01/2008 - 23:27:33 - IP: 163.102.100.17 - opened file c:\work\test.c.
21/01/2008 - 00:08:25 - IP: 163.102.100.17 - saved and closed file c:\work\test.c.

Considerando que estes computadores estão em sincronia, qual usuário fez o quê?

5. BIBLIOGRAFIA

FERREIRA, F. N. F. **Segurança da Informação**. Rio de Janeiro: Ciência Moderna, 2003.

Unidade 3: Integridade Lógica:
Tipos Comuns de Ataque à Informação

Prof. Daniel Caetano

Objetivo: apresentar alguns tipos comuns de ataque à informação e alguns princípios de segurança de rede para evitar ou dificultar alguns destes ataques.

Bibliografia: FERREIRA, 2003

INTRODUÇÃO

Conceitos Chave:
- Problema: Como proteger a rede contra ataques
  * Questão: **quais** ataques?
- Mecanismos de Proteção x Tipos de Ataques

Os ataques à informação são, hoje, bastante comuns. A maioria das pessoas pelo menos já ouviu falar a respeito de invasões de serviços, roubo de senhas ou números de cartão de crédito, dentre outros.

Mas como é possível proteger um sistema contra estes ataques? Qual é o tipo de medida necessária?

Para responder a estas perguntas, é necessário um conhecimento dos tipos mais comuns de ataques, sendo que existem diversos deles. Da mesma forma, existem também alguns recursos que permitem algum grau de segurança contra muitos destes ataques.

Nas próximas seções serão apresentados alguns destes ataques mais comuns e também alguns métodos de prevenção contra ataques lógicos pela rede.

1. TIPOS COMUNS DE ATAQUES À INFORMAÇÃO

Conceitos Chave:

- Controle de acesso: fundamental
    * Não garante segurança sozinho!
        + Senhas roubadas, força bruta etc.
        + Ataques que independem de acesso ao sistema
- Tipos Comuns
    * Engenharia Social
        + "Levar na Conversa"
        + Conhecimento dos procedimentos (e suas falhas)
        + Falhas ao seguir os procedimentos
        + Proteção: treinamento e procedimentos
    * Login por Força Bruta
        + Tentar todas as senhas possíveis, uma a uma
        + Fácil de identificar pelo log e aumento de tráfego
        + Proteção: limitação de tentativa e evitar senhas simples
    * Ataques por E-Mail / Phishing
        + Obtenção de dados (eng. social)
        + Instalar backdoors, propagar vírus...
        + Proteção: instalação de software protetor, treinamento.
    * Ataques por Acesso Remoto
        + Sistemas mal-configurados
        + Proteção: evitar o uso, adequar uso às políticas de segurança
    * Ataques DoS e DDoS
        + Excesso de acessos ao servidor
        + "Ônibus Lotado"
        + Duas formas: Falhas de Sistema ou DDoS
        + Proteção: corretivos de segurança, firewall

Ainda que o controle de acesso seja fundamental para garantir a segurança da informação, tal controle não garante esta segurança sozinho. Mesmo um controle de acesso eficiente quase sempre pode ser enganado (senhas roubadas, descoberta por força bruta etc.). Além disso, caso não sejam usadas proteções especiais, há alguns tipos de ataques (como ataques DoS, ataques por e-mail etc.) que podem ser aplicados independente do invasor conseguir obter acesso ao sistema pelo sistema de logon.

Engenharia Social é o processo que a maioria dos crackers (invasores de sistemas com objetivos prejudiciais) usam para obter acesso a um sistema. Aplicar engenharia social é aproveitar conhecimentos sobre as rotinas administrativas da empresa (e suas falhas) para conseguir acesso ao sistema. Por exemplo, o invasor liga para o suporte da empresa, diz que é um funcionário e fornece o nome de um funcionário que conhece (ou que já foi demitido), e diz que está há muito tempo sem acessar a intranet, e que esqueceu seu login e senha, e solicita que lhe forneçam o login e uma nova senha.

Caso não exista um procedimento previamente determinado para garantir um processo seguro de fornecer a senha ao usuário (como por exemplo, enviar o novo login e senha para a mesa do funcionário na empresa, entregue em mãos), a equipe de suporte pode causar um grande furo na segurança. Este tipo de invasão causa muitos problemas, pois o ataque ficará registrado no nome do funcionário que, possivelmente, não teria culpa alguma. Dependendo do nível de acesso deste usuário, os resultados podem ser catastróficos.

A forma de tentar evitar que este tipo de ataque tenha sucesso é através do estabelecimento de procedimentos de segurança bem definidos e, dentro do possível simples. Adicionalmente, é indispensável que sejam feitos treinamentos constantes dos funcionários, para que estejam sempre utilizando as mais novas práticas de segurança adotadas pela empresa.

Login por Força Bruta quando não é possível usar engenharia social, a maioria dos crackers parte para a tentativa de login por força bruta. O processo é descobrir um nome de usuário existente e testar diferentes senhas até que uma funcione. Este tipo de ataque pode ser detectado checando-se os logs, mas se o invasor não tiver pressa e executar as tentativas espaçadas por um intervalo de tempo, pode ser que seja mais difícil a identificação simplesmente por inspeção dos logs.

Por esta razão, são usados métodos alternativos, como impedir o uso de senhas comuns pelos usuários (nada de usar data de aniversário, por exemplo, que é uma informação que costuma ser facilmente obtida por engenharia social, e é sempre uma das primeiras tentativas de senha em login por força bruta), bloqueamento de acesso após três ou quatro tentativas de login incorretas, uso de firewalls e proxys para limitar quais máquinas de uma rede podem acessar quais recursos etc.

Ataques por E-Mail e Phishing são bastante comuns nos dias de hoje e vão desde ataques para roubar dados do usuário (forma mais recente de engenharia social), até tentativas de instalar *backdoors* (normalmente por cavalos de tróia) nas máquinas, possibilitando acesso ao sistema sem que o sistema de controle de acesso tome conhecimento, vírus (que podem ter os mais variados objetivos) etc.

Apesar de bastar bom senso para evitar este tipo de ataque, infelizmente é difícil garantir que todos os usuários do sistema da empresa tenham o mesmo nível de informação e bom senso. Infelizmente é difícil evitar que alguns usuários cliquem em mensagens como "Fotos da traição" ou "Atualize seus dados do Banco Trouxa S/A".

Ataques por Acesso Remoto (Remote Access) são comuns em sistemas mal configurados. A maioria das empresas instalam utilitários que permitem a configuração e suporte remoto das máquinas. Entretanto, alguns destes sistemas podem não estar (por padrão) subordinados às políticas de acesso ao sistema, criando brechas de segurança que não são admissíveis. O uso de acesso remoto deve ser evitado e, quando necessário, deve ser configurado com extremo cuidado e todas as restrições aplicáveis.

Ataques DoS (Denial of Service - Negação de Serviço) foram mais comuns há algum tempo, mas ainda são usados. Ataques de DoS são ataques que sobrecarregam o subsistema de rede dos servidores e os tornam indisponíveis (fazendo com que os usuários fiquem sem acesso aos recursos providos pela máquina em questão).

> Uma analogia comumente usada é a do ônibus lotado. Um ônibus serve para permitir o transporte de pessoas que precisam de um deslocamento; se alguém lotar este ônibus com "usuários falsos", pessoas que não precisariam estar naquele ônibus, ele ficará indisponivel para os usuários legítimos, que precisam do serviço. Neste caso, ocorrerá com o ônibus uma "negação de serviço".
>
> Da mesma forma, um servidor sob ataque DoS não consegue atender aos usuários legítimos porque há muitos "usuários falsos" ocupando toda sua capacidade. Por isso este ataque chama-se "Negação de Serviço" (Denial of Service, em inglês).

Há basicamente duas formas de conseguir sucesso com um ataque DoS: a primeira é explorando falhas na programação do subsistema de rede do sistema operacional do servidor. A segunda é através de milhares de pessoas conectando em um servidor ao mesmo tempo (DDoS - Distributed Denial of Service).

A primeira pode ser solucionada com a aplicação se corretivos de segurança do sistema com a maior freqüência possível. A segunda pode ser parcialmente solucionada com o uso de firewalls que detectem tentativas de ataque DoS e bloqueiem os usuários atacantes. Entretanto, há situações em que nem os firewalls conseguem bloquear um DoS.

2. VÍRUS, WORMs e TROJANs

Conceitos Chave:

- Vírus
  - * Programas que se multiplicam e podem causar danos
  - * Se anexam a outros programas
  - * Proteção: antivirus
- Worms
  - * Programas individuais que se transmitem e podem causar danos
  - * Proteção: firewall, antivirus, orientação de funcionários
- Trojans
  - * Programas aparentemente "úteis", mas com funções ocultas
  - * Transmitidos por vontade do usuário
  - * Proteção: antivirus e orientação de funcionários

Alguns dos ataques mais comuns nos tempos atuais são aqueles que objetivam espalhar os chamados vírus, worms (vermes) e trojans (cavalos de tróia). Cada um destes tem forma de operação específica e, daí, sua nomenclatura.

2.1. Vírus

Um dos ataques mais comuns nos dias de hoje são os vírus. Vírus são pequenos programas que, quando executados, têm a capacidade de se multiplicar/espalhar e podem ou não causar danos aos dados. É difícil indicar uma razão para a existência da maioria dos vírus já que a grande maioria não tem um propósito específico. Entretanto, existem vírus de propósito especiais, para atacar e/ou roubar informações específicas.

As formas mais comuns de disseminação são anexos em e-mails e cópia de programas infectados, seja por discos ou pela rede. Em geral se anexam a executáveis de outros programas e são executados quando o programa hospedeiro for carregado. Sua função principal é ficar residente na memória à espera de um momento para se reproduzir (se anexando a outros arquivos); a função secundária depende de seu desenvolvedor, mas em geral é apagar arquivos ou prejudicar a operação do equipamento.

As únicas formas minimizar os problemas com vírus são evitar a execução de programas de procedência duvidosa e manter sempre o Anti-Vírus em execução em plano de fundo, mantendo a base de dados atualizada (atualizações semanais, pelo menos).

2.2. Worms

Ao contrário dos vírus, os worms são quase sempre programas individuais, que são auto-executados por terem um nome específico (autorun.inf / xxxx.exe, por exemplo). Em alguns casos eles se auto-executam em uma máquina simplesmente pela conexão de rede, usando algum recurso de acesso remoto.

Os Worms em geral se reproduzem pela rede, seja pelo envio direto usando recurso de acesso remoto, seja pelo envio de mensagens contendo o worm anexado.

Para evitar worms é preciso ter anti-vírus atualizados, proteção da rede por firewall, e treinar os funcionários para que não executem qualquer coisa que venham em seus e-mails.

2.3. Trojans

Os trojans (cavalos de tróia) são similares aos Worms/Vírus, mas em geral eles estão disfarçados dentro de outros programas, aparentemente inofensivos. O usuário usa um programa e, sem querer, coloca em execução o trojan que, em geral, serve para roubar dados do usuário ou fornecer uma *backdoor* para invasores.

Os trojans normalmente se reproduzem pela vontade do usuário, que envia aquele programa "bonitinho" ou "útil" para outras pessoas, enviando o trojan junto.

Para evitar trojans é preciso ter anti-vírus sempre atualizados e treinar os funcionários para que não executem qualquer coisa que venham em seus e-mails.

3. PROTEÇÃO DA REDE

Conceitos Chave:

- Limitação do Acesso
- Endereço de Rede
    * Endereço x Telefone
    * Protocolo de Rede
- TCP/IP
    * IPv4: 32 bits - x,y,z,k
        + Ex.: 143.107.x.y => grande entidade
        + Ex.: 200.168.18.x => pequena entidade
        + Ex.: 200.168.18.55 => máquina
    * Analogia da Empresa
    * Portas: 143.107.254.11:80
        + Port Scanners
        + Portas Padrão
    * IPs Dinâmicos: DHCP
        + Melhor aproveitamento de IPs.
        + Maior proteção
    * Roteamento
        + Gateway e Máscara
        + Papel do Gateway/Roteador
- NetBIOS sobre TCP/IP (SMB / NetBeui)
    * Compartilhamento de Impressoras e Arquivos
    * Nome de Domínio e IP de Escopo
    * \\SERVIDOR\RECURSO
        + Ex.: \\MAFALDA\HP692C
    * Traduzidos em Pacotes TCP/IP
- Proxys
    * Intermediário que protege "identidade" do servidor
    * IPs Reais x IPs Fictícios
    * Redirecionamento por proxy
        + Controle de tráfego
        + Problema: compatibilidade
    * NAT e Socks
- Firewalls
    * No gateway ou nas máquinas
    * Inspeção de pacotes de informação
        + Regras de bloqueio ou permissão
    * Pacotes incluem firewall com outros serviços

A maior parte dos ataques comuns, automatizados ou não, dependem de acesso aos equipamentos através de uma rede, que no caso mais comum é a Internet. Para entender os mecanismos básicos de proteção de redes, é importante que sejam apresentados alguns aspectos sobre o funcionamento das redes.

3.1. Identificação de Máquinas pela Rede

No dia-a-dia de uma pessoa, se ela quiser ir até a casa de alguém, ela precisa do endereço para o qual se deslocar. Quando se pensa em endereço, automaticamente vem à mente dados como "rua", "número", "apartamento", "cidade"...

Embora nem todos pensem nisso, números de telefone também são endereços. O "DDD" indica uma grande região, os 4 primeiros dígitos do número indicam uma pequena região dentro da região do DDD (número da central) e os 4 últimos dígitos do número indicam um ponto de telefone (uma moradia). Em alguns lugares o ponto de telefone pode ser expandido com uma central proprietária, quando então é necessário identificar o ramal (um "cômodo" da moradia).

Da mesma maneira que uma casa tem um endereço e um telefone tem um "endereço", também os equipamentos ligados na rede possuem um endereço: o **endereço de rede**. É este endereço que, que usamos para nos comunicar com a internet, que os atacantes buscam para que possam atacar especificamente nossas máquinas.

Entretanto, ao contrário do que ocorre com o endereçamento de ruas e de telefone, o endereçamento de rede depende do *protocolo de rede* que é usado pelas máquinas que se comunicam naquela rede. Não cabe aqui uma análise de todos os protocolos disponíveis, mas serão analisados os dois protocolos mais comuns atualmente, devido à Internet: o TCP/IP e o NetBIOS sobre TCP/IP.

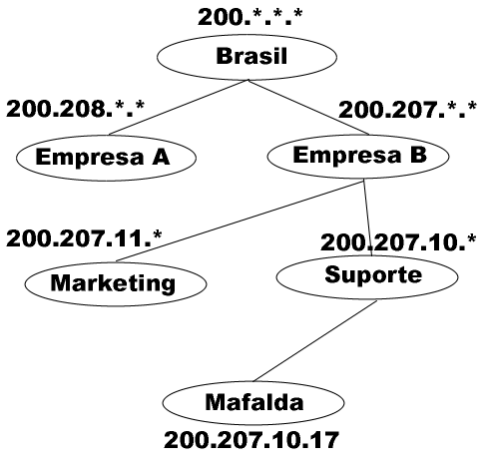
3.1.1. TCP/IP

A identificação de uma máquina no protocolo TCP/IP recebe o nome de "Número IP". Assim como o número de telefone especifica uma série de informações (grande região, central, moradia, ramal...), também o IP é composto por 4 informações, cada uma delas tendo 8 bits (compondo um número de 32 bits), separadas por pontos. Por exemplo: 200.207.10.17.

Simplificadamente, cada um dos números indicam subníveis de rede. Em geral, as grandes empresas e grandes universidades possuem um IP com os dois primeiros valores fixos (200.207.x.y, por exemplo), sendo que podem definir os valores que desejarem para os dois últimos (x.y, no exemplo anterior). Pode-se dizer que esta empresa pode ter até 65536 IPs reais internos, e o IP da empresa é 200.207.*. Este é o caso, por exemplo, da Universidade de São Paulo.

Empresas menores costumam ter uma faixa menor de IPs, sendo os 3 primeiros números do seu IP fixos (200.207.10.x), por exemplo, podendo associar qualquer valor ao último número (x), podendo ter apenas 256 IPs reais internos. Empresas muito pequenas possuem apenas um ou dois IPs reais, normalmente, sendo todos eles fixos (200.207.10.17 e 200.207.10.18, por exemplo).

Para facilitar a compreensão, é possível usar uma analogia como sendo o primeiro número a designação de país, o segundo número designa uma empresa, o terceiro um departamento e o quarto é um computador específico do departamento. Mas note que esta não é uma analogia perfeita: há países com mais de um primeiro número, há primeiros números compartilhados entre vários países, há empresas com mais de um segundo número e a divisão de IPs dentro de uma empresa não precisa obedecer qualquer tipo de ordenação.



Como cada computador pode estar executando vários serviços de rede (web server, ftp, servidor de banco de dados etc.), o protocolo TCP/IP define ainda um quinto número no endereço IP, que identifica o número do serviço. Este número pode ir de 1 a 65535, sendo este o número máximo de serviços que uma máquina pode oferecer no TCP/IP versão 4. Cada um destes números representa o que é chamado de "porta". Por exemplo: o ftp fica na porta 21 e o http na porta 80.

Assim, se o IP da máquina web da USP é 143.107.254.11, o endereço completo para o acesso web é: **143.107.254.11:80** . Note que o número da porta vem após o endereço da máquina, separado por dois pontos. Quando um atacante escolhe uma máquina para atacar, em geral ele usa um programa chamado **Port Scanner** para determinar quais são as portas abertas (serviços disponíveis). Se encontrar algo como um Telnet (porta 23), ele certamente tentará atacar por aí.

Agora, seria exigir muito dos usuários de Internet que eles guardassem centenas de endereços numéricos para cada site que visitam; as pessoas precisariam ter "livros de endereços" no mesmo molde de agendas telefônicas...!

Baseando-se nessa idéia de "agenda telefônica", foi criada uma "agenda telefônica oficial", que foi chamada de "**Serviço de Nome de Domínio**" (Domain Name Service, DNS). O servidor DNS é uma máquina que tem uma função muito simples:

a) Dado um nome textual (www.usp.br) retornar um endereço IP (143.107.254.11)
ou
b) Dado um endereço IP (143.107.254.11) retornar um endereço textual (www.usp.br)

Assim, de acordo com o caso acima, o nome da máquina 143.107.254.11:80 "traduzido" por um DNS é: **www.usp.br:80** . Isso é feito para que não precisemos decorar a seqüência de números 143.107.254.11.

Em geral, também não é necessário identificar o número da porta do serviço, dado que os programas de acesso aos serviços sabem as portas padrão com as quais se comunicar. Por exemplo: um navegador sabe que o servidor web está, por padrão, na porta 80. Caso o servidor web esteja em uma portal alternativa (8080, por exemplo), aí será necessário indicar ao navegador a porta alternativa.

Em geral, o acesso caseiro à Internet é feito sem números IP fixos. Isso significa que, a cada vez que um computador caseiro se conecta a Internet, ele recebe um IP diferente do provedor de acesso, através de um serviço chamado DHCP (*Dynamic Host Configuration Protocol* - Protocolo de Configuração de Provedor Dinâmico).

Esta atribuição dinâmica de IP tem uma função principal: permitir que um número grande de usuários compartilhem um número relativamente pequeno de IPs. Como só pode existir uma única máquina com um IP real, se o IP fosse "estático" e a empresa tivesse só 256 IPs, ela só poderia ter 256 clientes. Como o IP é dinâmico, ela pode ter muito mais clientes; a única limitação é que ela *só pode ter 256 clientes usando a rede simultaneamente*.

O efeito colateral ruim deste processo é que o usuário acaba com um uso limitado de sua conexão: se quiser ter um servidor web em sua máquina, precisará apelar para alguns serviços de terceiros, como o DynDNS, já que o número de sua máquina muda constantemente.

Mas há também um efeito colateral relativamente benéfico ao usuário, que é a proteção. Como o número IP muda a cada conexão, ou seja, a cada vez que o computador é ligado, fica mais difícil para um atacante realizar um ataque direcionado a um usuário. Fazendo uma analogia, seria como se um bandido quisesse espionar os hábitos de uma determinada pessoa, mas cada dia a casa dela estivesse em um lugar diferente da cidade.

Mas como uma máquina encontra a outra na rede?

Toda máquina, além de ser configurada com um número IP de identificação e um número de pelo menos um servidor DNS, ela é também configurada com um "Gateway" (também conhecido como Router), que é uma espécie de "maestro da rede", e uma "máscara de rede", que define quais os IPs de computadores que estão na mesma rede que ela.

Toda a informação que uma máquina deseja enviar para outra máquina, ela verifica, pela máscara de rede, se o dado é para uma outra máquina da própria rede em que ela se encontra. Se for, ela envia o dado diretamente para esta máquina. Caso contrário, se a informação for para uma máquina externa, ela será enviada para o Gateway.

O Gateway verifica se ele sabe para qual outra máquina aquele dado precisa ser enviado. Se ele souber, ele envia para ela. Se não, ele envia para seu próprio Gateway (afinal, ele também é uma máquina na rede!), que repete esse processo, até que a informação seja entregue ao destino correto. A informação vai, assim, caminhando de Gateway em Gateway, até encontrar a rede a que é destinada e ser entregue à máquina em questão.

Da mesma forma, quando um computador precisa ser contactado por uma máquina de fora da rede onde ele se encontra, a solicitação passará primeiramente pelo Gateway, e será posteriormente entregue ao computador a que ela se destina.

Este processo todo é chamado "Roteamento", e é por isso que alguns sistemas dão o nome de Roteadores (Routers) para os Gateways.

3.1.2. NetBIOS sobre TCP/IP (SMB)

O NetBIOS sobre TCP/IP é o que costumeiramente é usado para compartilhar arquivos e impressoras em uma rede. Também conhecido como Protocolo SMB (Samba, no Linux) ou TCP-Beui (embora esta última nomenclatura seja incorreta), este protocolo tem uma denominação um tanto diferente. Em geral o endereço é especificado por um nome de **domínio**, um nome de **máquina**, além do nome de **recurso**. A configuração deve incluir também o nome do escopo de máquina, que no caso será o nome do domínio da rede TCP/IP na qual ela se encontra.

Para que se tenha acesso a outras máquinas, é necessário identificar o domínio da máquina com o mesmo nome do domínio da rede. A partir de então as máquinas são acessíveis através de seus nomes, que podem ser indicados como: **\\computador\recurso** . Por exemplo: \\MAFALDA\HP692C , para acessar a HP692C no computador chamado MAFALDA. Se está máquina fizer parte da rede citada anteriormente, seu endereço na web (se acessível) poderia ser: http://mafalda.suporte.empresab.com.br/ . Esta nomenclatura é, internamente, traduzida para um endereço TCP/IP (por isso o protocolo chama-se NetBIOS sobre TCP/IP), que é usado para acessar a máquina.

Numa rede deste tipo é possível deixar que cada usuário controle o acesso a seus recursos (logon local em cada máquina conectada na rede) ou é possível centralizar o logon em um servidor de rede. Embora esta última alternativa seja mais burocrática, ela é mais adequada por aumentar o controle sobre a informação.

3.2. Mecanismos de Proteção da Rede

Como todo o acesso a uma máquina se dá através de um endereço IP e, em geral, esta informação passa antes por um computador Gateway, os mecanismos de proteção de redes são, em geral, projetados para se beneficiar destas características.

Existem dois tipos de proteção mais utilizados: os proxys e os firewalls.

3.2.1. Proxys

Como apresentado anteriormente, nos dois protocolos de rede mais comuns (TCP/IP e NetBIOS sobre TCP/IP) a identificação fundamental das máquinas é feita através do endereço IP. Sempre que um atacante visa uma máquina, a primeira necessidade é descobrir o IP desta máquina e obter acesso a ela. Assim, uma das maneiras mais comuns de evitar que máquinas servidoras sejam atacadas é não permitir o acesso do atacante ao IP destas máquinas.

A solução para isso pode ser obtida por uma analogia. Imagine que uma pessoa quer realizar um negócio com uma outra, mas não quer encontrá-la. Uma solução possível seria encontrar uma terceira pessoa para *intermediar* o negócio, certo?

No caso dos computadores, é a mesma coisa: a empresa tem um servidor web, mas ela não quer que os usuários tomem conhecimento do endereço deste servidor. A solução é usar um computador intermediário: o usuário acessa o computador intermediário que acessa o computador onde se encontra o servidor web. Tudo que o usuário solicitar, ele solicita para o intermediário e toda as respostas do servidor web são repassadas ao usuário pelo intermediário.

Isso permite que o usuário não tenha acesso ao número da máquina onde está o servidor web de fato: ele vai ter apenas o número da máquina intermediária, também conhecida como *máquina proxy*.

Essa proteção é parcial, dado que se o atacante conseguir o número da máquina onde está o servidor web, ele ainda poderá atacá-la. Mas existe uma solução para isso.

Se o Gateway de uma rede tiver um número IP válido na Internet, mas os números IP dos computadores da rede ligada a este Gateway forem definidos de tal forma que o Gateway **nunca** permita que máquinas externas enviem dados diretamente às internas, fica impossível que uma máquina de outra rede acesse as máquinas da rede local, pois elas possuem números IP que não são acessíveis por elas.

Estes números são chamados "*IPs Fictícios*" e, em geral, são do tipo 10.x.y.z ou 192.168.x.y. Estes números são inválidos para roteamento para fora de uma rede local.

Mas se o IP de uma máquina da rede local é inválido para o mundo exterior, como é que eu posso instalar um servidor web nesta máquina e permitir que ele seja acessado?

É aí que entra a mágica do servidor proxy. Se for instalado um servidor proxy no gateway (ou em uma máquina com um *IP real*, válido na rede externa), é possível configurar alguns redirecionamentos para a rede interna.

Por exemplo, é possível configurar o servidor proxy para que acessos à porta 80 do gateway sejam redirecionados para um computador da rede interna qualquer, 192.168.1.10,

por exemplo. É possível até mesmo redirecionar a porta. Basta, para isso, configurar uma regra no Proxy, que tem a seguinte cara:

Connection	**Redirection**
200.168.10.5:80	192.168.1.10:8081
(válido na internet)	(inválido na internet)

Assim, o usuário usará a máquina 200.168.10.5 para acessar o servidor web que se encontra na máquina 192.168.1.10. Como a máquina 192.168.1.10 é *inacessível* por computadores externos, o servidor web estará mais protegido lá do que se estivesse na máquina 200.168.10.5.

Este tipo de configuração permite controlar melhor quais portas estão disponíveis para o mundo exterior e também limitar o acesso de máquinas externas às máquinas internas. Infelizmente esta configuração também traz alguns inconvenientes: como as máquinas internas não possuem IPs válidos, elas também não conseguem acessar as máquinas externas. Por esta razão, os servidores proxy também costumam funcionar como intermediários para o à rede externa. Os acessos nos computadores externos serão registrados como se tivessem se originado na máquina proxy.

Desta forma, como o proxy intermedia tudo que entra e sai da rede interna para a rede externa (Internet), o uso de proxys também permite monitoramento de tudo que os usuários da rede estão acessando, o que em alguns casos pode ser importante para detectar algum usuário que esteja enviando informações não autorizadas através da rede.

O lado ruim é que nem todos os programas interagem bem com servidores proxy. Assim, algumas "evoluções" foram desenvolvidas para fazer com que programas rodando em máquinas com IPs fictícios possam funcionar como se estivessem em máquinas com IPs reais. Estas "evoluções" escondem a operação do proxy, passando este a funcionar como se fosse um Gateway normal e o IP fictício desta máquina fosse, na verdade, um IP real. Alguns destes mecanismos são o serviço **NAT** (Network Address Translator, ou Tradutor de Endereço de Rede) e o **Socks**, que podem inclusive permitir acessos apenas mediante login e senha etc.

3.2.2. Firewalls

Embora algumas vezes a atuação do firewall pareça similar à do proxy, os mecanismos de funcionamento são bem diferentes. Enquanto o proxy se limita a controlar acesso e redirecionar conexões, os firewalls servem para **inspecionar cada pacote de dados que entra ou sai**.

O Firewall é usualmente instalado no gateway (roteador) da rede e verifica todo o acesso entre a rede interna e a externa, e seu uso é comum mesmo em redes onde os IPs internos são todos reais. Através da **inspeção de todos os dados** trafegados, um firewall é capaz de **detectar ataques e bloqueá-los**, bloquear acesso de determinados IPs caso o

administrador da rede assim deseje, dentre outras características interessantes como, por exemplo, bloquear automaticamente máquinas que estejam tentando causar um DoS.

A maioria dos firewalls recentes podem ser instalados nas máquinas clientes também, permitindo que o usuário **controle quais programas enviam dados** a partir de sua máquina, evitando a atuação de adwares e, em alguns casos, vírus, trojans e worms.

Ao contrário do que muitos acreditam, os firewalls não são usados apenas na conexão da rede interna com o mundo exterior. Muitas vezes **é interessante ter firewalls separando** alguns grupos de máquinas **até mesmo do acesso interno**, se a informação contida nestas máquinas for sensível. Isso minimiza os riscos de sucesso de um ataque interno, realizado por um funcionário, por exemplo.

A grande maioria dos pacotes de firewall atuais trazem consigo também funções de proxy, NAT, socks, VPN e muitas outras. Este fato pode confundir um pouco sobre as funções específicas de um firewall, que são as de filtrar o tráfego que passa através de um ponto da rede, impondo critérios para definir o que pode e o que não pode passar.

4. EXERCÍCIOS

1. A intranet da empresa é, quase sempre, conectada à Internet. Você acha que isso é adequado? Que medidas você adotaria para evitar ataques ao servidor web da intranet, sabendo que existe a necessidade de páginas públicas na web (que não são da intranet)? Proponha uma solução completa.

2. Como você poderia evitar que seus funcionários sejam vulneráveis à engenharia social?

3. Como evitar ataques do tipo DoS?

5. BIBLIOGRAFIA

FERREIRA, F. N. F. **Segurança da Informação**. Rio de Janeiro: Ciência Moderna, 2003.

TANENBAUM, A. S. **Redes de Computadores**. Rio de Janeiro: Campus, 2003

Unidade 4: Integridade Física e Disponibilidade da Informação

Prof. Daniel Caetano

Objetivo: apresentar os conceitos de segurança lógica e física na manutenção da integridade física da informação, métodos de prevenção de perdas de dados e elaboração de planos de contingência.

Bibliografia: FERREIRA, 2003.

INTRODUÇÃO

Conceitos Chave:
- Problemas:
  * Como proteger os equipamentos?
  * Como proteger os dados, caso os equipamentos se danifiquem?
- Garantir a Integridade Física e a Disponibilidade

Uma vez que um profissional é promovido a um cargo de gerência de segurança da informação ou de sistemas, ele precisa ter em mente que são de sua responsabilidade não apenas a segurança contra hackers que ameaçam a informação através da rede de computadores, mas também envolve a segurança dos equipamentos, num sentido mais físico.

Cabe então a este profissional responder a algumas perguntas: "como proteger meus equipamentos?" ou "caso meus equipamentos sejam perdidos, como assegurar a recuperação dos dados?"

Nos próximos tópicos serão abordados alguns aspectos teóricos e práticos envolvidos na manutenção da integridade física e disponibilidade da informação.

1. SEGURANÇA LÓGICA E FÍSICA

Conceitos Chave:

- Aspectos Lógicos x Físicos
  - * Dado em Si x Meio Portante
  - * Integridade Lógica x Integridade Física
- Segurança Lógica
  - * Processo lógico do controle de acesso
    - + Identificação e Autenticação
    - + Registro em logs
    - + Controle de Permissões de Acesso
  - * Evitar que a informação seja alterada/apagada indevidamente
  - * Evitar que estranhos entrem na sala do servidor (controle da porta, p.ex.)
- Segurança Física
  - * Aspectos físicos do controle de acesso
  - * Dão suporte aos mecanismos lógicos
    - + Se a identificação/autenticação for falha, quem barra o sujeito?
      - = A porta
      - = O muro
      - = A grade
  - * Ligada apenas à integridade física
    - + Filosofia: "Destruição" Física implica "Destruição" Lógica

Sempre que se fala em informação, há dois aspectos a serem considerados: os lógicos e os físicos. Pode-se falar em manter a integridade lógica, ou seja, da informação em si ou em manter a integridade física da informação, ou seja, do meio portante (equipamentos e mídias, por exemplo).

Assim, cabe apresentar os conceitos de segurança lógica e segurança física. A segurança lógica é aquela que tem sido apresentada com mais ênfase no momento, relativa ao processo lógico do controle de acesso (como login e senha, log de acesso etc.) e é usada para garantir tanto a integridade lógica (mudança do conteúdo dos documentos) quanto a integridade física (destruição física dos equipamentos e/ou documentos).

A segurança física, por outro lado, lida com os aspectos físicos do controle de acesso (como muros, portas, câmeras, sistemas de fechadura eletrônicos etc.). Embora seja possível dizer que a destruição ou roubo de um documento provoque um dano ao seu conteúdo lógico (que pode ter deixado de existir), não é comum fazê-lo. Em geral se faz ligação da segurança física apenas à integridade física da informação (ou seja, do equipamentos e mídias).

1.1. Segurança Lógica na Integridade Física

Conceitos Chave:

- Papel Chave no Controle de Acesso Físico
  - * "Não há equipamento seguro se o intruso tiver acesso físico ao eqpto."
  - * 72% dos ataques são causados por funcionários (FBI)
    - + 20% são causados por terceiros autorizados pela empresa
    - + 8% são causados por agentes externos (pessoas sem permissões)
- Limitar e Registrar o acesso físico
  - * Quem é, o que Possui/Sabe
- Controle de Acesso Físico Automatizado => Segurança Lógica
  - * Software que faz o controle de acesso
  - * Características desejáveis de um sistema deste tipo
    - + Proteção contra ataques forçados *(força bruta/corte de energia)*
    - + Atualização do Sistema *(número de usuários e privilégios)*
    - + Registro de Acessos *(data, hora, usuário, local, no. de tentativas)*
    - + Autenticação por Senha *(smartcard + senha)*
    - + Bloqueio de Múltiplos Acessos *(entrar duas vezes sem sair?)*
    - + Flexibilidade *(controle centralizado de acesso)*
    - + Monitoração *(log de uso indevido, falhas...)*
    - + Sistema de Backup *(sistemas sobressalentes, base de dados...)*
    - + Proteção dos Equipamentos *(segurança máxima!)*

Na área de segurança lógica, além da usual preocupação com o acesso aos dados através de redes, existe grande preocupação também com o controle de acesso físico (obviamente, em combinação com os aspectos da segurança lógica).

Alguns autores chegam a dizer que "não há equipamento seguro se o intruso tiver acesso físico ao equipamento". Segundo o FBI, 72% dos ataques (fraudes, roubos, sabotagens, etc.) são causados por funcionários da própria empresa. Cerca de 20% são causados por terceiros autorizados pela empresa a manipular as informações e apenas cerca de 8% dos ataques são causados por agentes externos (por redes ou por pessoas sem permissões de acesso).

Assim, além do usual papel de bloquear acessos indesejáveis aos dados através de terminais ou rede, o controle de acesso tem também a função de limitar o acesso físico. Assim como no controle de acesso mencionado quando a Integridade Lógica foi apresentada, mais uma vez surgem os elementos: "quem é o indivíduo", "o que ele possui" e/ou "o que ele sabe". É importante relembrar que os sistemas de controle de acesso devem se basear ao menos em dois destes elementos.

Mas se estamos falando de acesso físico, porque "Segurança Lógica"? Porque a maioria do acesso físico de segurança é, hoje, automatizado. E existe um software por trás da

automação destes dispositivos físicos de controle de acesso. Ou seja: este software e seu funcionamento são os aspectos lógicos envolvidos.

Os sistemas de controles de acesso físico automatizado devem ter as seguintes características:

- **Proteção contra ataques forçados** (força bruta, corte de energia etc.).
- **Atualização do sistema** (maior número de usuários, mudanças de privilégios etc.).
- **Registro de acessos** (log de data, hora, local, usuário, tentativas frustradas).
- **Autenticação por senha** (preferencia um sistema de smartcard + senha)
- **Bloqueio de múltiplos acessos** (quem já está dentro não pode entrar novamente).
- **Flexibilidade** (controle centralizado das permissões de acesso de cada elemento das instalações (elevadores, portões etc.)).
- **Monitoração** (monitoração, com geração de log, de uso indevido das instalações, violações, mal-funcionamento, falta de energia).
- **Sistema Backup** (sistemas sobressalentes para situações críticas de falha... cuidados com a base de dados!).
- **Proteção dos Equipamentos** (segurança máxima para o sistema que administra o controle de acesso físico).

Muitas destas características são similares às já comentadas quando o assunto era o controle de acesso para manutenção da integridade lógica, mas algumas são distintas. É importante ressaltar que nem todas as empresas precisam de sistemas tão complexos, sendo necessária uma avaliação caso a caso.

1.2. Segurança Física

Conceitos Chave:
- Segurança Física sem Segurança Lógica => Cadeado => Ok
- Segurança Lógica sem Segurança Física => Pula Muro => Não Ok
- Investimento => Segurança Física
- Controle de Acesso Físico:
    * Grades, muros, portas
    * Guardas
    * Crachás
    * Sistemas de Portas Duplas
- Segurança Física é só Controle de Acesso?
    * Proteção contra agentes naturais ou criminosos!
    * Dificultar espionagem

De nada adianta um segurança lógica perfeita em um sistema de controle de acesso se a segurança física for falha. Por exemplo: se o sistema de identificação e autenticação for perfeito mas o funcionário puder entrar em uma dada sala ignorando solenemente o tal sistema, ele terá falhado completamente.

Assim, uma boa parte do investimento do controle de acesso está na segurança física. Alguns dos tipos do controle de acesso físico são:

- **Grades, muros, portas** (limites das regiões restritas, monitoradas)
- **Guardas** (garantir o controle de acesso em pontos estratégicos - como as entradas - checagem de visitantes).
- **Crachás** (identificação dos funcionários e visitantes. Quanto menos informações, melhor).
- **Sistemas de Portas Duplas** (obrigar a identificação, evitando que intrusos "entrem junto" com pessoas autorizadas).

Entretanto, o aspecto de segurança física engloba mais que o controle de acesso. É preciso que as informações estejam protegidas fisicamente contra agentes naturais ou criminosos (fogo, explosões, inundações, corte de energia etc.).

Além disso, as instalações onde serão armazenadas informações sensíveis devem ser projetadas de forma a dificultar espionagem, até mesmo com isolamento, no caso de informações especiais.

2. PLANO DE CONTINGÊNCIA E CONTINUIDADE DE NEGÓCIOS

Conceitos Chave:
- Negócios => Altamente dependentes das informações
  * Continuidade de negócios!
- Objetivo: continuidade das operações até que sistema volte a operar
- Metas:
  * Segurança de empregados e visitantes
  * Minimizar perdas e danos imediatos
  * Restauração das atividades (instalações e eqptos) rapidamente
  * Rápida reativação dos processos críticos de negócio
  * Conscientização e treinamento dos responsáveis pela execução do plano

Como a maioria dos negócios hoje é altamente dependente das informações, além da preocupação com a preservação da mesma, existe também uma preocupação com a continuidade dos negócios quando da ocorrência de uma catástrofe (com possível perda de informações, ainda que temporária).

O objetivo é que, no caso de uma catástrofe, seja possível dar continuidade às operações da empresa até que o sistema volte a operar normalmente. Para que esta continuidade seja possível, é preciso *antever* as possíveis catástrofes e planejar uma solução de operação quando da ocorrência das mesmas.

As principais metas de um plano de continuidade de negócios são:

- Segurança dos empregados e visitantes;
- Minimizar perdas e danos imediatos;
- Restauração das atividades, instalações e equipamentos de forma rápida;
- Rápida reativação dos processos críticos de negócio;
- Conscientização e treinamento dos responsáveis pela execução do plano.

2.1. Análise de Risco

Conceitos Chave
- Ameaça: ocorrência que possa provocar perda
- Desastre: impacto de força externa que ocasiona perda
- Risco: medida numérica que quantifica a probabilidade de um desastre
- Aspectos da Análise de Risco
    * Caracterização dos Sistemas *(criticidade hard/soft/pessoal/info)*
    * Identificação de Ameaças *(ameaças por estatísticas/histórico)*
    * Identificação de Vulnerabilidades *(ameaças por teses de seg.)*
    * Análise dos Controles de Segurança *(atuais e planejados)*
    * Determinação da Probabilidade *(ameaça e vuln.=>probabilidade)*
    * Análise de Impacto *(conseqüências: perda I.D.C.)*
    * Determinação do Risco *(alto/médio/baixo)*
    * Recomendação dos Controles
    * Documentação de Resultados

O primeiro passo para que se possa criar um plano de contingência é a avaliação e análise de risco. Para isso é preciso entender os conceitos envolvidos:

- **Ameaça**: qualquer ocorrência que possa provocar alguma perda para a organização;
- **Desastre**: impacto de uma força externa que ocasiona perda ou prejuízo. Não precisa ser destruidor, bastando impedir a execução de alguma atividade de negócio crítica.
- **Risco**: medida numérica que quantifica a probabilidade de ocorrência de desastre.

A análise de risco se faz com base nos seguintes aspectos:

- **Caracterização dos sistemas** (limitações de hard/soft/pessoal/inf. e sua criticidade);
- **Identificação das ameaças** (ameaças - base em estatística e histórico de ataques);
- **Identificação das vulnerabilidades** (ameaças - base em teses de segurança);
- **Análise dos controles de segurança** (controles atuais e planejados);
- **Determinação da probabilidade** (ameaças e vulnerabilidades => proabilidade);
- **Analise de impacto** (Conseqüências: perda de integr./disponib./confidenc.);
- **Determinação do risco** (níveis de risco: alto, médio e baixo);
- **Recomendação dos controles**;
- **Documentação dos resultados**.

2.2. Planejamento das Contingências

Conceitos Chave
- Objetivos de cada regra de negócios relevante
- Fatores importantes
    * Regras e Responsabilidades
    * Recursos Necessários
    * Treinamentos e Testes
    * Manutenção do Plano e Freqüência de Backups

Antes mesmo de se pensar em constituir um plano de contingências, são necessárias declarações do planejamento de contingências, onde serão descritos os objetivos de cada regra de negócio que será levada em conta na criação dos planos. Os fatores mais importantes a serem considerados são:

- **Regras e responsabilidades** (quem são os responsáveis pelo plano);
- **Recursos necessários** (equipamentos, pessoal, informações);
- **Treinamentos e Testes** (freqüência, tipo, descrições);
- **Manutenção do plano e Freqüência de Backups** (freqüência, tipo).

Ainda antes de ser detalhado um plano de contingências, devem ser verificados todos os mecanismos de prevenção e estratégias de prevenção disponíveis/aplicáveis, já que estes fatores podem influenciar o conteúdo do plano de contingências.

2.3. Identificação dos Mecanismos de Prevenção

Conceitos Chave
- Reduzem riscos
- Reduzem magnitude de impactos
- Exemplos
    * No Breaks (UPS) / Geradores
    * Detectores de Incêndio
    * Ar Condicionado *(eqptos. / docs)*
    * Cofres a Prova d Fogo, Água e Fumaça
    * Armazenagem Externa
    * Backups Freqüentes
    * Criptografia

Estes mecanismos influenciam na criação do plano de contingência por reduzirem os riscos e, possivelmente, a magnitude dos impactos nos sistemas. Todos estes mecanismos

devem estar documentados no plano de contingência, incluindo os responsáveis pela manutenção e operação de cada um deles. Alguns exemplos de tais mecanismos são:

- **No Breaks (UPS) / Geradores**: fornecimento de energia elétrica por períodos de tempo adequados;
- **Detectores de incêndio**: evitar/minimizar dano por fogo;
- **Ar Condicionado**: evitar desgaste prematuro dos equipamentos por excesso de temperatura, bem como deterioração de documentos;
- **Cofres a prova de fogo, água e fumaça**: para preservação de documentos essenciais;
- **Armazenagem externa**: para armazenamento de cópias de documentos importantes e backups;
- **Backups freqüentes**: evitando perdas de informações;
- **Criptografia**: para evitar roubos e fraudes;

2.4. Estratégias de Recuperação

Conceitos Chave

- Estratégias para recuperar o sistema em caso de interrupção dos negócios
- Usadas no Plano de Contingência
    * Backups
        + Freqüência necessária (Diária/Semanal/Mensal...)
            = Incremental x Completa
            = Rotação de mídias
        + Identificação dos locais originais / processos de recuperação
        + Transporte e Armazenamento Seguros
            = Armazenamento Externo
            = Proteção contra Deterioração
    * Localidades Alternativas *(podem ser terceirizadas)*
        + Cold Site *(recursos mínimos, sem processamento)*
        + Warm Site *(alguma capacidade de processamento)*
        + Hot Site *(cópia completa do serviço, 24/7)*
    * Reposição de Equipamentos
        + Contratos com Fornecedores
        + SLA - Service Level Agreement (prazos!)
        + Inventário => Obsolência
    * Regras e Responsabilidades
        + Quem é responsável por qual estratégia e qual sua equipe
- Teste constante de planos

São as estratégias para recuperar o sistema, em caso de ocorrência de interrupções dos negócios. Os planos de contingências deverão se utilizar destas estratégias para a recuperação do sistema, coordenando as ações de maneira previamente planejada. A seleção de estratégias depende dos níveis de risco e impactos aos quais cada elemento a proteger estão sujeitos.

- **Backups**: são cópias de segurança. Devem ser realizados com a freqüência necessária (minimizando os prejuízos de perdas de informação), podendo ser desde diária até mensal, incremental ou completa. As cópias devem indicar os locais dos dados originais e processos de recuperação, deve ser realizada rotação de mídias (podendo ser fitas, CDs ou DVDs, por exemplo) e tanto o transporte quanto o armazenamento externo devem ser feitos utilizando-se de métodos seguros.

É recomendado o armazenamento em localidade externa (em outras edificações) evitando perdas em acidentes físicos de maiores proporções. A área externa deve ser razoavelmente distante, mas não tão longe que as cópias demorem a ser obtidas em caso de necessidade. Além disso, os dados de backup precisam ser adequadamente protegidos com relação a critérios de segurança física e o ambiente de armazenamento precisa ter controle ambiental para evitar deterioração das mídias. O custo de manutenção de uma localidade de armazenamento é alto, mas existem empresas especilizadas nesta atividade.

- **Localidades Alternativas**: são localidades prontas a funcionar como serviço temporário enquanto o sistema principal é recuperado, podendo ser terceirizadas. As localidades alternativas podem ser classificadas como **cold site** (recursos mínimos de infra-estrutura, sem recursos de processamento: indicados para serviços tolerantes à indisponibilidade), **warm site** (recursos médios, com alguma capacidade de processamento e comunicação, adaptada às necessidades quando necessário: indicados para serviços medianamente tolerantes à indispobibilidade) e os **hot sites** (cópia completa do serviço principal, operando 24 horas, 7 dias por semana: indicados para serviços intolerantes à indisponibilidade).

- **Reposição de Equipamentos**: são planejamentos prévios para reposição de equipamentos para situações de interrupção. É possível trabalhar com **contratos com fornecedores**, descrevendo um SLA (Service Level Agreement) em que se descreva o prazo máximo em que o fornecedor deve fornecer a reposição, indicando prioridade máxima e descrevendo todos os processos necessários para a reposição. Uma outra alternativa é o uso de **inventário ou estoque de equipamentos**, em que a empresa mantém estoque dos equipamentos críticos do sistema. Esta abordagem é mais segura, mas mais custosa, devido especialmente à obsolência dos equipamentos.

- **Regras e responsabilidades**: cada uma das estratégias de recuperação deve ter seu responsável e sua equipe, que deve ser treinada para sua execução

É importante lembrar que os planos podem apresentar falhas, então precisam ser constantemente testados. Além disso, as necessidades de proteção mudam com o tempo, exigindo que o sistema seja atualizado para que seja eficaz. Finalmente, os funcionários precisam ser bem treinados para realizar as atividades nos momentos corretos e estarem a par das últimas modificações realizadas nos planos.

2.5. Forma do Plano de Contingência

- Basicamente, 5 seções:
  - * Informação de Suporte
    - + Operação
    - + Situações de Uso
    - + Sistema envolvido
    - + Responsáveis
    - + Hierarquia de Notificação
  - * Notificação / Ativação
    - + Procedimentos Iniciais *("Primeiros Socorros")*
    - + Processo de notificação de responsáveis
    - + Avaliação de Danos (Tipo, Emergência etc.)
    - + Ativação
  - * Recuperação
    - + Seqüência de atividades de recuperação
    - + Habilitação de Sistema Alternativo
    - + Recuperação do Sistema Principal
  - * Reconstituição
    - + Desmobilização da Contingência
    - + Testes dos Sistemas Principais
    - + Integração dos Dados
  - * Anexos
    - + Responsáveis pelo plano
    - + Checklists
    - + Especificações Técnicas
    - + ...

Definidos todos os mecanismos e estratégias, os planos de contingência podem ser formalizados. Os planos devem descrever basicamente 5 seções:

- **Informação de suporte**: introduz a operação e as situações de uso, descrevendo o sistema envolvido, responsáveis pelo sistema e a árvore hierárquica de notificação.

- **Notificação / Ativação**: indica a seqüência de procedimentos iniciais, uma vez que tenha sido detectada uma interrupção. Deve descrever o processo de notificação dos responsáveis (ordem, telefone, informações a passar etc.), como avaliar os danos (causa e nível da emergência, áreas afeadas, tipos de danos encontrados etc.) e como proceder a ativação (quando os danos apurados assim o exigirem).

- **Recuperação**: deve descrever a seqüência de atividades de recuperação (habilitação de sistema alternativo, correção do sistema principal).

- **Reconstituição**: procedimento para desmobilização da contingência e volta às operações normais, procedimentos de teste dos sistemas principais e integração dos dados.

- **Anexos**: informações adicionais sobre responsáveis pelo plano, checklists de auxílio, especificações técnicas etc.

3. EXERCÍCIOS

1. É sabido que backups armazenados externamente possuem vantagens e desvantagens. Você faz parte de uma equipe de consultoria de segurança e, para compreender melhor o assunto e poder se decidir, o cliente lhe solicitou que você o ajudasse com exemplos. Sendo assim, especifique uma situação em que você usaria o backup armazenado externamente e uma situação em que você não o usaria, apresentando suas razões.

2. Sua equipe agora trabalha no Banco da Praça, uma instituição sólida com longos anos de tradição. A diretoria, entretanto, anda preocupada com problemas de segurança e é para isso que sua equipe foi criada. Em uma das análises feitas por sua equipe, foi solicitado que houvesse um sistema de backup armazenado externamente.

De posse desta solicitação e dos valores envolvidos, a diretoria do banco solicitou que este serviço fosse terceirizado. Em outras palavras, o banco deve contratar uma empresa especializada no backup armazenado externamente; neste panorama, quais as exigências que sua equipe faria com relação à empresa de backup?

3. O Banco Central das Galáxias (BCG) que administra todo o sistema bancário intergalático, incluindo os TEDs e monitoramento de transações bancárias, teve problemas de segurança e está querendo melhorar o sistema de operação em contingências com a implantação de um Cold Site. Sua equipe foi contratada para cuidar da segurança da informação, mas a diretoria, acostumada a ingerências, insiste que o sistema alternativo de funcionamento seja um simples Cold Site. Vocês acham que um Cold Site seria suficiente? Se não, o que seria? Por quê?

4. BIBLIOGRAFIA

FERREIRA, F. N. F. **Segurança da Informação**. Rio de Janeiro: Ciência Moderna, 2003.

Unidade 5: Confidencialidade da Informação

Introdução à Criptografia

Prof. Daniel Caetano

Objetivo: apresentar os conceitos de fundamentais envolvidos na confidencialidade da informação e os termos básicos da área de criptografia.

Bibliografia: TERADA, 2000. FERREIRA, 2003.

INTRODUÇÃO

Conceitos Chave:
- Problemas:
  * Como manter a confidencialidade?
- Segurança
  * Disponibilidade
  * Integridade
  * Confidencialidade
- Confidencialidade => Codificação de documentos
- Confidencialidade de Documentos Digitais
  * Um dos desafios dos tempos modernos
    + Computadores mais velozes
    + Necessidade de mecanismos mais complexos

Quando se fala em segurança da informação, sempre é importante ressaltar os quatro pontos da segurança: disponibilidade, integridade/autenticidade e confidencialidade. Muito já foi dito nas aulas anteriores sobre manutenção da disponibilidade, integridade e autenticidade, mas muito pouco foi dito sobre a manutenção da confidencialidade.

A confidencialidade é conseguida, em geral, com a **codificação** de documentos, de forma que apenas as pessoas com autorização possam lê-lo. É um conceito simples, porém a confidencialidade da informação digital tem sido um dos grandes desafios dos tempos modernos. Com a grande disponibilidade de computadores cada vez mais velozes, mecanismos de codificação outrora considerados seguros passaram a ser considerados obsoletos e inseguros.

Como conseqüência, houve um aumento da complexidade nos mecanismos de codificação, tornando sua compreensão um pouco mais distante do dia-a-dia do desenvolvedor. A partir desta aula - e até o final do curso, diversos mecanismos de proteção da confidencialidade (alguns deles usados para verificação de integridade e autenticidade também) serão analisados.

1. CONFIDENCIALIDADE DA INFORMAÇÃO

Conceitos Chave:

- Manutenção da Confidencialidade
  - * Fundamental: acesso apenas a informações permitidas
  - * Evitar acesso de pessoas não autorizadas a determinados documentos
- Métodos Simples Funcionam?
  - * Não mandar informações à pessoas erradas?
  - * Ligação Telefônica => pode ser grampeada
  - * E-Mail => pode ser interceptado
  - * Conversa pessoa => pode ser espionada
- Ideal: Controle de Acesso!
  - * Controle de acesso em mensagens?
    - + Controle de acesso NA mensagem?
- Métodos capazes de impedir leitura de pessoas não autorizadas
  - * Mais recente: verificar autenticidade e integridade
  - * Técnicas Criptográficas
- Proteger apenas dados transmitidos?
  - * E os armazenados localmente? => Laptops?
  - * Crypto FileSystems

Quando se fala em "confidencialidade da informação", o primeiro aspecto importante é o significado da expressão "manutenção da confidencialidade da informação". Considerando a política de segurança da informação de uma empresa, é fundamental que cada pessoa ou funcionário tenha acesso apenas às informações que lhe são permitidas. Pode-se dizer, então, que manter a confidencialidade da informação é evitar que pessoas não autorizadas tenham acesso às informações que não lhes competem.

Em princípio, pode parecer que métodos simples como "não mandar informações para pessoas erradas" sejam suficientes para garantir a confidencialidade da informação. Entretanto, tal colocação está distante da realidade. Uma ligação telefônica pode ser grampeada, um e-mail pode ser interceptado e mesmo uma troca de informações feita pessoalmente pode estar sujeita à espionagem.

Assim, dado que uma informação não é pública e apenas usuários seletos possuem acesso a elas (como o número de seu cartão de crédito, por exemplo), é preciso que exista uma espécie de "controle de acesso" que seja parte indissociável da mensagem, ou seja, inseparável da mesma.

Ao longo da história foram criados diversos métodos para controlar as pessoas capazes de ler determinados documentos; tais métodos permitem que a transmissão da informação por meios usuais, em um formato que apenas os receptores de uma mensagem saberão compreendê-la. Mais recentemente foram desenvolvidos métodos que possibilitam

até mesmo que este receptor seja capaz de identificar a integridade e autenticidade da informação. Estes métodos foram denominados técnicas criptográficas.

Entretanto, é importante ressaltar que não apenas as informações que são transmitidas precisam estar protegidas. A maior parte dos ataques à informação praticados na atualidade são originados em funcionários da própria empresa, que podem atuar como piratas da informação, podendo roubar bases de dados e, por exemplo, vendê-las.

Assim, as informações confidenciais armazenadas nos equipamentos locais podem também fazer uso criptografia, além do controle usual do sistema operacional sobre os acessos dos arquivos. De fato, existem sistemas de arquivos baseados em criptografia (Crypto FileSystem, por exemplo), o que significa que cada simples informação escrita no disco será, automaticamente, criptografada. Estes sistemas exigem que um disquete ou pen-drive com um arquivo especial esteja presente para que os dados possam ser lidos.

2. CRIPTOGRAFIA

Conceitos Chave:

- O que é?
  - * Kryptos => oculto
  - * Graphos => escrita
  - * Escrita Oculta?
    - + E se ninguém conseguir ler?
- Arte/Ciência/Técnica
  - * Escrever um texto que só destinatário consiga ler
  - * Nomenclatura
- Cifragem: texto puro => texto cifrado
- Decifragem: texto cifrado => texto puro
- Algoritmo de Encriptação/Decriptação
  - * Estudo dos Algoritmos: Criptologia

Mas o que é criptografia? Criptografia é uma palavra que vem do grego, *kryptos* significa oculto e *graphos* significa escrita. Assim, uma técnica criptográfica é uma técnica de "escrita oculta", no sentido de "significado oculto". Entretanto, apenas escrever de forma que ninguém possa ler é algo inútil. A criptografia é, pois, a arte/ciência/técnica de escrever um texto de forma que apenas o destinatário seja capaz de entendê-lo.

Tipo:	Texto Puro	Texto Cifrado
Nome:	Plain Text	Cypher Text
Aparência:	Legível	Aleatória

O processo de converter o texto puro em texto cifrado é chamado "cifragem" (ou encriptação) e ele é feito usando um algoritmo de criptografia. A conversão do texto cifrado

para texto puro é chamado "decifragem" (ou decriptação). A decifragem pode ser feita usando um algoritmo de decriptação, que pode ou não ser o inverso do algoritmo de encriptação. A ciência que estuda os algoritmos de criptografia e decriptação é a criptologia.

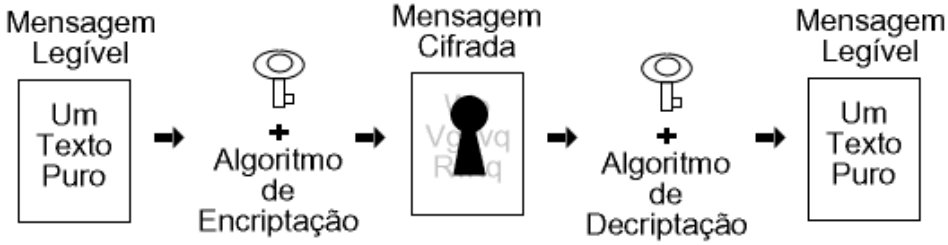
3. CHAVES DE CRIPTOGRAFIA

Conceitos Chave:

- Analogia do Cofre: basta ter uma porta?
  - * Precisa haver um segredo
- Analogia da Porta: basta haver uma porta?
  - * Precisa haver uma chave!
- Chave: é o "segredo" de uma criptografia
  - * Similar a uma "senha"
  - * Destinatário e remetente precisam ter a chave
  - * Informação compartilhada
- Exemplo de Processo de Cifragem/Decifragem
- Criptoanálise: quebra da criptografia
  - * Decodificação sem possuir a chave
  - * Encontrar a chave a partir do texto cifrado
  - * Análise de Padrões x Fatoração
- Segurança de uma criptografia
  - * Proporcional ao tempo de quebra desta criptografia
  - * Tempo deve ser longo o suficiente (informação perca a validade)
  - * Evolução dos Equipamentos
  - * Milhões de anos hoje podem ser alguns segundos no futuro

É um conceito bastante conhecido que não basta que um cofre tenha uma porta, é preciso que exista um segredo para abri-la; este mesmo conceito vale na criptografia pois não basta a existência de um algoritmo de criptografia, é preciso também uma chave.

A chave é o que garante que, mesmo se alguém conhecer o algoritmo criptográfico utilizado, esta pessoa não será capaz de decifrar o texto. Na realidade, a chave é justamente a informação que precisa ser compartilhada entre o remetente e o destinatário de uma mensagem, para que o processo de cifragem/decifragem possa ocorrer adequadamente, ou seja, para que o destinatário consiga decifrar a mensagem que foi cifrada pelo remetente.



O processo de "quebra" da criptografia, ou seja, obtenção da chave criptográfica através da própria mensagem cifrada, recebe o nome de criptoanálise. A criptoanálise pode ser feita utilizando-se diversas técnicas, como análise de padrões e fatoração, por exemplo. A segurança de um algoritmo de criptografia é diretamente relacionada ao tempo necessário para a obtenção de sucesso num processo de criptoanálise automatizado.

Assim, um algoritmo de criptografia é considerado seguro quando o tempo necessário para a realização da criptoanálise é grande o suficiente para que a informação não mais tenha valor quando for decifrada. Como o tempo de criptoanálise automatizada depende da quantidade de informação disponível e, principalmente, da velocidade dos equipamentos utilizados, é usual considerar um algoritmo seguro quando o tempo necessário para a criptoanálise é de cerca de milhões de anos com os equipamentos atuais.

Segurança dos Dados = 1 / Facilidade de Quebra da Criptografia

É claro que nenhuma informação precisa de tanto tempo para se tornar inútil ou obsoleta. Entretanto, a evolução dos computadores tem sido tal que, em alguns anos, o que levaria milhões de anos para ser processado passe a ser processado em 5.000 anos. Considerando uma evolução deste tipo, se o algoritmo fosse consumir apenas 100 anos das máquinas atuais, seriam necessários apenas 6 meses para decifrar a mensagem neste futuro projetado.

4. INÍCIO DA CRIPTOGRAFIA

Conceitos Chave:
- Ciência Antiga
  - * Egípcios => Hieróglifos?
  - * Império Romano => Alfabeto de César
- Exemplo de Texto Cifrado pelo Alfabeto de César
  - * Algoritmo de Substituição
- Sabendo língua de origem, fácil quebra
  - * Preservação de Freqüência
- Exemplo de Criptoanálise
  - * Letras que mais aparecem
- "Chave" da Cifra de César?

A criptologia é uma **ciência antiga**, mas não se sabe exatamente ao certo qual foi a primeira utilização de algoritmos de criptografia. Embora alguns considerem que os **egípcios** tenham sido os primeiros a utilizá-la, a escrita hieroglífica **não é uma escrita criptográfica**, embora se assemelhe a códigos criptográficos.

Uma das referências mais antigas ao uso real da criptografia vem da época do Império Romano, onde foi usada uma técnica simples de criptografia, chamada de "Alfabeto de César". Por essa técnica criptográfica, teríamos:

```
Texto Cifrado:    "Hvwd h xpd iudvh qrupdo qd olqjxd sruwxjxhvd."
Texto Puro:       "Esta é uma frase normal na língua portuguesa."
```

Observe que o texto cifrado parece absolutamente aleatório. Como é a técnica do "Alfabeto de César"? Por se tratar de um dos primeiros métodos, sabendo-se que ele é apenas um algoritmo de substituição e que a mensagem está na língua portuguesa, ele é um dos métodos de mais fácil quebra.

Vejamos, inicialmente, quais são as letras que mais aparecem na mensagem cifrada:

D-7 H-4 X-4 Q-3 U-3 V-3 J-2 O-2 P-2 R-2 W-2 I-1 L-1 S-1

Ora, na mensagem cifrada a **letra que mais aparece é D, seguida por H e X**. Na **língua portuguesa**, as **letras mais freqüentes são vogais**, de onde podemos supor que, provavelmente, D, H e X correspondem à vogais. **A vogal mais freqüente** é a vogal "**a**". Assim, suponhamos que "D" na mensagem cifrada seja o mesmo que "A" na mensagem original.

Considerando isso e considerando que **A é a primeira letra** do alfabeto e **D é a quarta**, temos uma distância de **4 - 1 = 3 posições**. A teoria então é que **subtraindo 3** posições **das letras cifradas são obtidas as letras não cifradas**. Verifiquemos se, com essa regra, H e X também são vogais.

H é a oitava letra do alfabeto. **8 - 3 = 5**, sendo que a **quinta letra do alfabeto é E**. **X é a vigésima quarta letra** do alfabeto. **24 - 3 = 21**, sendo que a **vigésima primeira letra do alfabeto é U**. Aparentemente a regra se aplica e, com efeito, subtraindo 3 da posição de cada letra podemos recuperar a frase original:

```
a b c d e f g h i j  k  l  m  n  o  p  q  r  s  t  u  v  w  x  y  z
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
```

Cifra	Número	Número - 3	Texto
H	8	5	E
v	22	19	s
w	23	20	t
d	4	1	a
h	8	5	e
x	24	21	u
p	16	13	m
d	4	1	a
i	9	6	f
u	21	18	r
d	4	1	a
v	22	19	s
h	8	5	e

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

Cifra	Número	Número - 3	Texto
q	17	14	n
r	18	15	o
u	21	18	r
p	16	13	m
d	4	1	a
o	15	12	l
q	17	14	n
d	4	1	a
o	15	12	l
l	12	9	i
q	17	14	n
j	10	7	g
x	24	21	u
d	4	1	a
s	19	16	p
r	18	15	o
u	21	18	r
w	23	20	t
x	24	21	u
j	10	7	g
x	24	21	u
h	8	5	e
v	22	19	s
d	4	1	a

Como o deslocamento entre as letras é 3, podemos dizer que "3" é a chave usada neste algoritmo de criptografia, que troca a letra atual pela terceira letra seguinte. Assim, a encriptação e decriptação do "Alfabeto de César" podem ser facilmente indicados com a seguinte seqüência:

Texto Puro
a b c d e f g h i j k l m n o p q r s t u v w x y z
d e f g h i j k l m n o p q r s t u v w x y z a b c
Texto Cifrado

5. EXERCÍCIOS

1. Praticamente todo algoritmo de criptografia usado na atualidade possui uma "chave criptográfica". Para que uma mensagem criptografada seja trocada entre duas pessoas, quem deve ter essa chave? Por quê?

2. A criptoanálise é o processo de "quebrar uma criptografia", isto é, identificar o conteúdo de uma mensagem criptografada sem ter a devida autorização. Como alguém pode conseguir isso?

3. O que é e para que servia o Alfabeto de César?

4. Nem sempre um algoritmo de criptografia é considerado seguro para proteger uma determinada informação. Qual é o principal critério para se considerar seguro um algoritmo de criptografia?

6. BIBLIOGRAFIA

TERADA, R. **Segurança de Dados:** Criptografia em Redes de Computador. São Paulo: Edgard Bücher, 2000.

FERREIRA, F. N. F. **Segurança da Informação**. Rio de Janeiro: Ciência Moderna, 2003.

Unidade 6: Conceitos para Criptografia

Prof. Daniel Caetano

Objetivo: apresentar os conceitos de fundamentais envolvidos na criptografia, como segurança criptográfica, criptoanálise e entropia e segurança perfeita.

Bibliografia: TERADA, 2000. FERREIRA, 2003.

INTRODUÇÃO

Conceitos Chave:
- Problema: Garantir Sigilo e Autenticidade
  * Uso da Criptografia
  * Uso da Criptoanálise
  * Técnicas Criptográficas
  * Segurança Perfeita
  * Melhoria da Criptografia
    + Fundamentos

Existem diversas situações em que se deseja manter o sigilo e garantir a autenticidade de informações. A ferramenta que será usada para este objetivo é a criptografia, cujos fundamentos foram apresentados anteriormente.

Ao mesmo tempo em que usaremos a criptografia para tentar garantir o sigilo e a autenticidade, haverá pessoas tentando "quebrar" essa criptografia, isto é, tentando violar nossas garantias de sigilo e autenticidade. Para isso, estas pessoas se usam de técnicas de criptoanálise.

Assim, é importante compreender os princípios que norteiam a segurança criptográfica, o funcionamento das técnicas criptográficas mais comuns, além do conceito teórico de "segurança perfeita", contra o qual uma técnica criptográfica possa ser comparada.

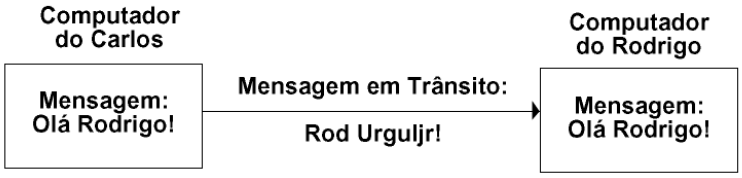
Finalmente, como na prática os algoritmos de segurança não são perfeitos, é interessante conhecer uma maneira de melhorar sua qualidade e quais são fundamentos para esta melhoria.

1. PROBLEMAS DE SIGILO E AUTENTICIDADE

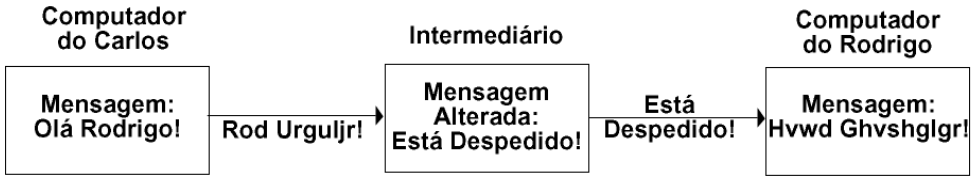
Conceitos Chave:
- Sigilo e Autenticidade
  - * Autenticação de Destino
  - * Integridade da Informação
  - * Autenticação da Origem
- Uso da Criptografia
  - * Clássica
  - * Assinaturas Digitais
- Problema: Criptoanalistas

Existem basicamente três tipos de problemas que o uso da criptografia visa eliminar, todos eles ligados às questões de sigilo e autenticidade:

1) **Autenticação de Destino**: Apenas o destinatário consegue ler a informação armazenada ou transmitida.



2) **Integridade da Informação**: Evitar que intermediário altere a informação:



3) **Autenticação da Origem**: Similar ao problema da Integridade da Informação, mas em uma situação em que Carlos nem mesmo tenha escrito uma mensagem original.

Em cada um destes, a criptografia terá um papel: no primeiro caso, usa-se a criptografia na forma clássica; no segundo e terceiro, usa-se na forma clássica e/ou na forma de assinaturas digitais. A aplicação da criptografia serão temas de aulas futuras.

Entretanto, independentemente da forma com que a criptografia seja usada, sempre haverá pessoas interessadas em quebrá-la, isto é, violar a confidencialidade e/ou integridade das informações. Para isso, estas pessoas farão uso das técnicas de *criptoanálise*.

2. CRIPTOANÁLISE

Conceitos Chave:
- Uso de Matemática e Análise de Padrões
  * Ler um texto sem ter permissão
  * Deduzir uma chave
- Ataques por Análise de Padrões
  * Só texto cifrado
  * Textos cifrados e legíveis
  * Texto Legível Escolhido
  * Texto Cifrado Escolhido
- Outros Ataques
  * Chaves Conhecidas
  * Replay
  * Personificação
  * Dicionário
- Manutenção de padrões e freqüências => algoritmos fracos

A criptoanálise é o uso de técnicas matemáticas, em geral análise de padrões, para conseguir ler um texto cifrado sem ter autorização para isso. Em geral, isso significa "deduzir a chave de criptografia" a partir das informações disponíveis.

Dependendo do tipo de informação disponível, a quebra pode ser mais complexa ou mais simples. Por esta razão, é comum analisar a segurança de algum algoritmo de criptografia considerando o tipo de informação necessário para quebrá-lo.

Os tipos de informação mais comuns utilizados pelos criptoanalistas são:

1) **Só texto cifrado**: é o tipo de ataque que se baseia apenas nas mensagens cifradas. É o tipo mais comum de ataque, já que não envolve nenhum conhecimento adicional além do conteúdo das mensagens cifradas. Para bons algoritmos, esse tipo de ataque não leva a lugar algum; em outras palavras, se este ataque for computacionalmente viável, o algoritmo é considerado inútil para uso em criptografia.

2) **Textos cifrados e legíveis**: é o tipo de ataque que se baseia em mensagens cifradas e mensagens não cifradas. Este tipo de ataque requer que o criptoanalista tenha acesso a documentos cifrados e não cifrados, não necessariamente correspondentes. A idéia é analisar vários textos legíveis, encontrar padrões de similaridades (freqüências de repetição, formatação etc.) e tentar usar estas informações para quebrar os textos cifrados. Para bons algoritmos, a chance de sucesso deste tipo de ataque é bem baixa e, por isso, se este ataque for viável, o algoritmo é considerado fraco para uso em criptografia.

3) **Texto Legível Escolhido**: é o tipo de ataque que se baseia em um (ou mais) texto legível para o qual se conhece o texto cifrado correspondente. Este tipo de ataque depende, em geral, de efetiva "ajuda interna", já que é necessário comparar um texto legível com sua versão cifrada; é preciso ter certeza de que se trata do mesmo texto. A idéia é tentar encontrar

padrões que existam em ambos os textos (legível e cifrado) para identificar a chave de criptografia. Esta técnica é, em geral, aplicada de maneira iterativa: analisa-se o legível, usa-se essa informação no cifrado, descobre-se um padrão no cifrado, verifica-se a existência do padrão no legível... e assim por diante). Algoritmos quebrados por este tipo de ataque são considerados de qualidade mediana para uso em criptografia.

4) **Texto Cifrado Escolhido**: É um processo bastante similar ao do "Texto Legível Escolhido", mas inicia-se o processo pela análise do texto cifrado, e não pelo texto legível. Esta técnica também é, em geral, aplicada de maneira iterativa.

Há também ataques específicos que não exigem o conhecimento de textos legíveis, sendo baseados na dedução direta de chaves. Estes tipos de ataques podem ser resumidos da seguinte maneira:

1) **Chaves Conhecidas**: é o tipo de ataque que, de posse de um conjunto de chaves conhecidas, o atacante as usa para criar novas chaves. É comum quando o atacante consegue diversas chaves de uma pessoa, e tenta compô-las para descobrir uma chave que ele não possui.

2) **Replay**: é o tipo de ataque que normalmente é concretizado com o uso de *trojans* ou *sniffers*, em que o atacante consiga gravar parte das informações trocadas pelos usuários legítimos (que estão trocando mensagens criptografadas) e usa tais informações em seu proveito, ou seja, para deduzir a chave de criptografia.

3) **Personificação** (Engenharia Social): é o tipo de ataque em que o atacante finge ser um dos usuários legítimos para conseguir informações de outro usuário legítimo, conseguindo informações do usuário legítimo sem que este perceba o trambique.

4) **Dicionário**: é o tipo de ataque em que várias chaves comuns são calculadas antecipadamente, com palavras comuns (ou combinando datas de aniversário com nomes, por exemplo) e estas chaves são testadas uma a uma.

Um ponto que convém ressaltar é que métodos que mantém a freqüência das letras e outros padrões são bastante fracos, pois com grande quantidade de texto e o conhecimento da língua em que o texto foi escrito, tais algoritmos se tornam fracos, isto é, de quebra relativamente fácil.

3. SEGURANÇA CRIPTOGRÁFICA

Conceitos Chave:

- Cifra de César
  - * Criptografia de Chave Secreta
- Mas, se souber o algoritmo...
  - * Facilita a identificação da chave!
  - * Criptografia Fechada
    - + A maioria dos antigos métodos criptográficos
  - * "Computacionalmente Fácil"
    - + Criptografia "Quebrável"
- Criptografia Aberta
  - * Algoritmo Público
  - * "Computacionalmente Inviável"
    - + Se não foi quebrada, é segura
- Provar Segurança?
  - * Publicidade
  - * Tempo em exposição
- Criptografia Fechada **não** significa fraca
- Mas Criptografia Aberta é que garante "força"

Anteriormente, foi apresentada a Cifra ou Alfabeto de César. O alfabeto de César consiste em substituir as letras das palavras pelas de um alfabeto "deslocado" de um certo número de posições. Na aula passada foi apresentada uma Cifra de César com um deslocamento de 3 caracteres:

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
```

Nesta cifra, tanto o autor quanto o destinatário precisam conhecer a "chave" (no caso é 3, o número de deslocamentos de cada caractere) e mais ninguém pode ter o conhecimento dela, a Cifra de César é classificada como "Criptografia de Chave Secreta". Ou seja: esta chave precisa ser mantida em sigilo para que outras pessoas não sejam capazes de ler o texto.

Entretanto, não é preciso ir muito longe para perceber que, tendo o alfabeto 26 letras, há apenas 25 chaves de criptografia diferentes. Assim, se um cracker desconfiar que foi usada uma Cifra de César (algo não tão difícil para um cracker experiente, pela aparência dos dados cifrados), ele precisa de, no máximo 25 tentativas para descobrir a chave que leva ao texto original.

Além disso, na Cifra de César, quase sempre que se usa uma chave incorreta, o texto resultante na decifragem é ilegível, um amontoado de letras sem sentido. Isso facilita ao criptoanalista a identificação da chave correta: quando um texto legível aparecer, ele encontrou a chave certa.

Em outras palavras, no caso da Cifra de César, o conhecimento do algoritmo usado (a Cifra de César) pode facilitar a identificação da chave de criptografia. A maioria dos algoritmos desenvolvidos até o século XX eram deste tipo. Por esta razão, os algoritmos eram escondidos com tanto segredo quanto as chaves, muitas vezes não havendo uma distinção clara entre algoritmo e chave.

Sempre que um algoritmo de criptografia for escondido, como nos casos citados acima, se diz que é uma "**Criptografia Fechada**", pois não se revela o processo da criptografia. Em geral, isso é feito pelo "medo" que a revelação do algoritmo facilite sua quebra, o que *não é necessariamente verdade* (embora fosse no caso de algoritmos antigos, pois o número de chaves existentes era, em geral, limitado).

Nestes casos, em que há poucas chaves e que o conhecimento do algoritmo torna *computacionalmente fácil* a descoberta da chave correta, dizemos que a criptografia é quebrável.

> De uma maneira mais precisa, diz-se que, se for computacionalmente fácil descobrir a chave a partir de um número polinomial de pares de caracteres criptografados e não criptografados, a chave ou o algoritmo são quebráveis.
>
> Por "computacionalmente fácil" entende-se que computadores atuais consigam "quebrar" a criptografia em um prazo de tempo tal que a informação ainda seja útil quando for revelada.

Com o desenvolvimento dos algoritmos matemáticos, chegou-se ao ponto em que mesmo sendo conhecido o algoritmo, é computacionalmente inviável quebrar tais chaves. Estes algoritmos, em que se revela o algoritmo publicamente, são classificados como de "Criptografia Aberta" e, hoje, só são considerados realmente seguros os algoritmos cujo algoritmo é conhecido e, ainda assim, não tenham sido quebrados.

Mas **como provar a segurança** de um algoritmo criptográfico**?** Na verdade, **não há um método matemático**. O **princípio** para comprovar a segurança de um algoritmo de criptografia é **torná-lo público** em conferências internacionais de criptografia. Se **após os testes mais exigentes dos especialistas** da área da criptoanálise ele não for quebrado, ele passa a ser **considerado seguro**.

> ATENÇÃO: O fato de alguém **não** revelar um algoritmo de criptografia não significa que este algoritmo seja obrigatoriamente "fraco". Por outro lado, para ter a certeza de que um algoritmo de criptografia é "forte", o algoritmo **deve** ser publicado e testado publicamente. Quanto maior for o tempo de publicidade de um algoritmo sem nenhuma "quebra", maior é considerada a segurança deste algoritmo.

4. TÉCNICAS CRIPTOGRÁFICAS BÁSICAS

Conceitos Chave:
- Cifra de César: Substituição Simples
- Há outras técnicas
  * Substituição Simples
    + Preserva a freqüência dos Símbolos
    + Preserva padrões
  * Cifra de Vigenère
    + Não preserva a freqüência de Símbolos
    + Adiciona um padrão: a chave
  * Cifra de Vigenère-Vernam (one-time-pad)
    + Elimina padrão da chave
    + Chave enorme!
  * Transposição
    + Preserva símbolos
    + Preserva freqüência de símbolos
    + Destrói padrões seqüenciais
    + Transposição de Bits: resolve problemas
- Composição de Técnicas => algoritmos atuais
- Aumento de Qualidade
  * Adicionar Confusão
    + Característica adicionada pela substituição, por exemplo.
    + "Ofusca" identificação do texto original
  * Adicionar Difusão
    + Característica adicionada pela transposição, por exemplo.
    + "Espalha" os dados, diminuindo padrões seqüenciais.

A maioria dos algoritmos de criptografia usa uma composição de diversas técnicas criptográficas mais simples. Algumas delas são:

1) **Substituição Simples** (ou mono alfabética): substituição se letras da mensagem original por outras letras. É um caso mais genérico que a Cifra de César, pois naquela é preservada a ordem alfabética (apesar do deslocamento). Neste caso, não. Assim, o número de possibilidades de chave cresce de 25 para 26! (4.03×10^{26}).

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
G E B K V T U Q S W O R C M I A D P L Y F J N Z X H
```

Este algoritmo ainda **preserva a freqüência das letras**, o que é uma grande vulnerabilidade.

Para reverter a operação, basta realizar a operação inversa (inverter os sinal), com a mesma chave.

2) **Cifra de Vigenère** (substituição poli alfabética): soma do valor das letras da frase com os valores das letras de uma chave de várias letras, usando o resto da divisão. Por exemplo, consideremos o valor das letras como:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Consideremos agora uma codificação da palavra COLORIDO usando a chave RUMO. Os números que representam as palavras COLORIDO e RUMO são:

C	2	R	17
O	14	U	20
L	11	M	12
O	14	O	14
R	17		
I	8		
D	3		
O	14		

A codificação é feita somando os valores das letras da palavra com os valores das letras da chave, fazendo o resto da divisão por 26 para que o número final sempre esteja entre 0 e 25:

Palavra:	C=2	O=14	L=11	O=14	R=17	I=8	D=3	O=14
Chave:	R=17	U=20	M=12	O=14	R=17	U=20	M=12	O=14
Soma:	19	34	23	28	34	28	15	28
Resto /26:	19	8	23	2	8	2	15	2
Cifra:	T	I	X	C	I	C	P	C

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Após a criptografia com a chave RUMO, a palavra COLORIDO se torna TIXCICPC. Esta criptografia tem uma grande vantagem de **não preservar a freqüência das letras**. Entretanto, a **repetição da chave para encriptação pode facilitar a quebra**. O número de possibilidades de criptografia aqui é de 26^n, onde n é o tamanho da chave.

A reversão é feita da mesma maneira, invertendo o sinal da operação e somando 26 antes do resto de divisão:

Cifra:	T=19	I=8	X=23	C=2	I=8	C=2	P=15	C=2
Chave:	R=17	U=20	M=12	O=14	R=17	U=20	M=12	O=14
Subtração:	2	-12	11	-12	-9	-18	3	-12
Soma 26:	28	14	37	14	17	8	29	14
Resto /26:	2	14	11	14	17	8	3	14
Palavra:	C	O	L	O	R	I	D	O

3) **Cifra de Vigenère-Vernam** (*one-time-pad*): É a mesma criptografia de Vigenère, mas com chaves sempre do tamanho da mensagem original. Apesar de pouco prática, ela **elimina o ponto fraco da chave de Vigenère**. Considera-se que uma chave de Vigenère-Vernam composta de uma seqüência aleatória de valores é uma criptografia forte (considerando que o cracker não tem como realizar ataques de texto-legível-conhecido).

4) **Transposição**: é a mudança da ordem das letras, com base em uma chave numérica. Por exemplo, consideremos uma chave de 4 posições: 4132, por exemplo. Podemos numerar as posições de uma palavra repetidamente, como, por exemplo, a palavra COLORIDO, e trocar a ordem das letras de acordo com a chave pré-definida (note que há dois grupos de quatro letras, um marcado pela cor azul e outro pela cor verde):

Texto Legível	Posição da Letra	Chave	Texto Cifrado
C	1	4	O
O	2	1	C
L	3	3	L
O	4	2	O
R	1	4	O
I	2	1	R
D	3	3	D
O	4	2	I

Ou seja, a palavra COLORIDO foi criptografada em OCLOORDI. Este tipo de criptografia é ruim por duas razões: **preserva a freqüência das letras** e, pior, **preserva as próprias letras**. Entretanto, é possível executar a transposição de maneira que essas características não se preservem (transposição de bits, por exemplo).

Para obter a chave de reversão da codificação da Transposição, basta aplicar a chave ao texto cifrado:

Texto Legível	Posição da Letra	Chave	Texto Cifrado
O	1	4	C
C	2	1	O
L	3	3	L
O	4	2	O
O	1	4	I
R	2	1	R
D	3	3	D
I	4	2	O

5) **Composição**: É uma combinação de várias técnicas ou funções criptográficas. Ainda que cada uma das funções simples não sejam seguras, a composição é relativamente mais segura. Por exemplo, é possível misturar a cifra de Vigenère com a transposição.

Consideremos a palavra COLORIDO com a chave RUMO. Como foi visto anteriormente, aplicando a Cifra de Vigenère, temos como resultado a palavra: TIXCICPC.

Consideremos uma chave numérica derivada de RUMO. Se as letras de RUMO fossem colocadas em ordem alfabética, o resultado seria: MORU. Considerando então que M=1, O=2, R=3 e U=4, a senha numérica relativa a RUMO seria 3412. Aplicando, temos:

Texto Vigenère	Posição da Letra	Chave	Texto Cifrado
T	1	3	X
I	2	4	C
X	3	1	T
C	4	2	I
I	1	3	P
C	2	4	C
P	3	1	I
C	4	2	C

É possível afirmar que a cifra XCTIPCIC é mais forte que a cifra TIXCICPC (obtida apenas com a técnica de Vigenère) ou mesmo que a cifra OCLOORDI (obtida apenas com uma transposição).

4.1. Confusão e Difusão

Das técnicas vistas, as duas principais são a **substituição** e a **transposição**. A primeira é aplicada substituindo caracteres do texto puro por outros caracteres e a segunda é aplicada alterando a ordem dos caracteres de forma sistemática. O fundamento de sucesso destas técnicas está intrinsecamente ligado aos conceito de confusão e difusão.

É chamado de **confusão** o efeito acrescentado pela substituição, que torna mais complexa a identificação da relação entre a chave e o texto cifrado. Por outro lado, é chamado de **difusão** o efeito acrescentado pela transposição, reduzindo a ocorrência de seqüências repetidas no texto cifrado. Em praticamente todos os **algoritmos criptográficos atuais** são aplicadas **duas técnicas** de criptografia, em composição: uma para acrescentar **confusão** e outra para acrescentar a **difusão**.

5. EXERCÍCIOS

1. Nos dias atuais, se usa a chamada criptografia aberta. Qual a garantia de segurança que se tem usando criptografia aberta?

2. Por que seria ruim a preservação da freqüência dos símbolos em uma mensagem criptográfica? Quais das técnicas criptográficas básica vistas reduz ou elimina o problema da preservação de freqüência de símbolos?

3. A palavra ABOBORA foi codificada (substituição poli-alfabética) com a seguinte chave: CABBDHK. O resultado foi a cifra: CBPCRYK. Proponha uma chave que decifre esta cifra na palavra ABACAXI.

6. SEGURANÇA PERFEITA

Conceitos Chave:

- Objetivo dos algoritmos criptográficos
  - * Mas o que é isso?
- Conjunto de textos Legíveis L e Cifrados C.
- "A probabilidade de uma chave criptografar um Li para um Cj é igual à probabilidade de qualquer outra chave tê-lo feito"
  - * Não deve haver "chaves mais prováveis"
  - * Um texto cifrado pode ter se originado em qualquer texto legível
- Condições da Segurança Perfeita
  - * Existe exatamente uma chave que leva um legível a um cifrado
  - * Todas as chaves levam um cifrado a um legível (nunca o mesmo)
- Exemplo: one-time-pad
  - * Segurança "Quase" Perfeita
- Criptosistemas Aleatórios: três propriedades
  - * Alfabeto = "a" e texto = "n" => legíveis = cifrados = a^n.
  - * H chaves igualmente prováveis. Cada chave h decripta qualquer cifrado.
  - * Cada chave h leva um cifrado a um conjunto aleatório de legíveis.
  - * Se a probabilidade de obter legível com significado é alta
    - + probabilidade de obter legível correto é baixa

Sempre que um **algoritmo criptográfico** é desenvolvido, o **objetivo** é que ele traga uma **segurança perfeita**. Mas o quando saber se um algoritmo dá uma segurança perfeita?

A idéia é relacionada ao conceito de probabilidade: a probabilidade de uma chave transformar aquele legível Li para aquele cifrado Cj é exatamente a mesma de qualquer outra chave fazê-lo.

Em outras palavras, observando um texto cifrado e seu legível correspondente, não deve existir uma chave mais provável que outra. Adicionalmente, a probabilidade de um dado texto cifrado ser relativo a um dado texto legível é a mesma de este texto cifrado ser relativo a um outro texto legível qualquer.

De maneira mais formal, diz-se que "Dado que há um conjunto de textos legíveis L e um conjunto de textos cifrados C, é considerado que existe segurança perfeita se a probabilidade de todas as chaves criptografarem um dado legível l_i para um dado cifrado c_j é a mesma de todas as chaves criptografarem um outro legível l_k para o mesmo c_j."

Disso pode-se concluir:

a) para que todo cifrado seja equiprovável de um legível, o número de cifrados deve ser o mesmo que o de legíveis.
b) o número de pares legível-cifrado é o quadrado do número de legíveis.
c) é necessário existir pelo menos uma chave para cada para cada legível (ou cifrado).
d) para que as chaves sejam equiprováveis, elas devem ser em número múltiplo do número de legíveis (ou cifrados)
e) se houver mais de uma chave por legível, aumenta-se a chance de quebra (se fossem sorteados dois números por vez na Loteria, , seria mais fácil ganhar)... logo, deve-se ter apenas uma chave por legível.
f) por conseqüência, cada chave aplicada a um mesmo legível deve levar a um dos diferentes cifrados
g) da mesma forma, qualquer chave aplicada a um dos cifrados, levará a um dos legíveis.

Pode-se concluir que as condições necessárias e suficientes para a segurança perfeita, para o caso em que temos um mesmo número de textos legíveis, cifrados e chaves:

a) Dado um par (legível, cifrado), existe exatamente uma chave que criptografa este legível para este cifrado.
b) Todas as chaves são igualmente prováveis, ou seja, todas as chaves existentes levam um cifrado em um legível qualquer (nunca o mesmo).

Em outras palavras, se existirem 3 legíveis l_1, l_2 e l_3, 3 cifrados c_1, c_2 e c_3 e 3 chaves K_1, K_2 e K_3, é possível construir a seguinte tabela de requisitos para a segurança perfeita:

Legíveis	Chave	Cifrados
l_1	K_1	c_1
l_2	K_1	c_2
l_3	K_1	c_3
l_1	K_2	c_3
l_2	K_2	c_1
l_3	K_2	c_2
l_1	K_3	c_2
l_2	K_3	c_3
l_3	K_3	c_1

É possível dar um exemplo simples de algoritmo que "quase" satisfaz a segurança perfeita. Considere o caso onde todos os textos legíveis possíveis possuem um tamanho máximo fixo "L", com um número total de chaves maior ou igual a L, sendo todas as chaves igualmente prováveis. Neste contexto, aplica-se o seguinte algoritmo:

Para cada elemento do texto legível, é adicionado o elemento da chave e é tirado o resto da divisão pelo número total de símbolos. Por exemplo, suponha L = 7, sendo os elementos iguais a 26 (letras do alfabeto).

```
Texto Legível:          ABACATE             0  1  0  2  0 19  4
Chave Usada:            USBDOPT            20 18  1  3 14 15 19
-------------------------------------------------------------
Soma:                                      20 19  1  5 14 34 23
-------------------------------------------------------------
Resto Div. por 26:                         20 19  1  5 14  8 23
Texto Cifrado:          UTBFOIX            20 19  1  5 14  8 23
```

Este algoritmo é conhecido pelo nome de "one-time-pad" (ou vigenère-vernam) e pode ser considerado de "segurança perfeita", embora as chaves não sejam todas exatamente equiprováveis.

Note que é possível construir chaves que transformam a palavra cifrada UTBFOIX em qualquer outra palavra que tenha 7 letras. Se ao invés de 26 símbolos usarmos 27, um deles indicando espaço, é possível criar chaves para transformar a palavra UTBFOIX em qualquer outra palavra, legível ou não, com 7 letras ou menos.

O fato de ser possível criar chaves que levam a palavras não legíveis é que o distancia da segurança perfeita, já que estas chaves são claramente incorretas e, portanto, destruindo a equiprobabilidade das chaves.

Um defeito que é importante ressaltar, mais uma vez, é que o One Time Pad exige uma chave tão longa (ou maior) que o próprio texto, o que é bastante inadequado. Por outro lado, desconsiderando este "detalhe", é um dos algoritmos de maior segurança conhecidos.

6.1. Criptosistemas Aleatórios

Um criptosistema aleatório tem basicamente três propriedades:

1) Se há um alfabeto com "a" componentes e os textos legíveis têm comprimento "n", o número de textos legíveis é a^n e o número de textos cifrados também é igual a a^n.
2) Todas as chaves são igualmente prováveis. Se há h chaves, cada cifrado pode ser decriptografado por cada chave h.
3) As h chaves levam cada texto cifrado a um conjunto aleatório de textos legíveis.

Uma conseqüência importante de criptosistemas aleatórios é que, se a probabilidade de obter um texto legível com significado for alta, a probabilidade de obter o texto legível correto será baixa.

7. ENTROPIA E SALTING

Conceitos Chave:

- Entropia: relativo à quantidade de informação em um conjunto
  - * Quantidade de textos possíveis
- Quanto maior o número de possibilidades...
  - * maior a dificuldade em "chutar" o conteúdo atual
  - * É mais fácil acertar na Quina ou na Sena?
- Exemplo: dois caracteres
  - * Binário (B) x Ternário (T)
  - * Perguntas: qual a mensagem, sabendo...
  - * E(B) < E(T)
- Entropia = medida da incerteza de um conjunto de resultados
- Aplicando a chaves
  - * Chaves pequenas
    - + pouca informação => menor incerteza => entropia baixa
    - + quebra fácil
  - * Chaves grandes
    - + muita informação => maior incerteza => alta entropia
    - + quebra mais difícil
- Aplicando à mensagem
  - * Mensagem pequena... x Mensagem grande
- Aumentando a Entropia das Chaves
  - * Chaves grandes e com alta variabilidade
    - + Não use frases => relação diminui variabilidade
- Aumentar entropia do texto legível?
  - * *Salting* => Acrescentar informações aleatórias na mensagem
    - + A = 65 = 01000001b
    - + 01000001 xxb
    - + 01000001 00b = A
    - + 01000001 01b = A
    - + 01000001 10b = A
    - + 01000001 11b = A
    - + Exemplo na Cifra de César

A **entropia** é uma medida que está **relacionada à quantidade de informação em um dado conjunto**. E(X), ou seja, a entropia do conjunto X, é um número que identifica a quantidade de informação do conjunto X.

É natural que, quanto maior for o número de informações que pode ser representado por um conjunto, maior é a dificuldade em identificar qual é a informação presente (ou seja: maior é a incerteza quanto ao conteúdo).

Como um exemplo, consideremos um conjunto composto por 2 caracteres:

__ __

Se for definido que estes caracteres podem ser apenas os números 0 e 1, o conjunto de dois caracteres pode representar apenas 4 informações diferentes:

00
01
10
11

Por outro lado, se for definido que estes caracteres podem ser os números 0, 1 ou 2, o número de possibilidades aumenta para 9:

00
01
02
10
11
12
20
21
22

Assim, em qual das perguntas há mais incerteza na resposta:

A) "Qual é a mensagem, sabendo que ela 2 dígitos e cada dígito pode representar 3 valores diferentes?"

ou

B) "Qual é a mensagem, sabendo que ela 2 dígitos e cada dígito pode representar 2 valores diferentes?"

Claramente a pergunta A tem uma resposta mais incerta, pois um número maior de mensagens pode ser codificado no mesmo espaço (dois caracteres). Assim, E(A) > E(B), ou seja, a entropia do conjunto de respostas A é maior que a entropia do conjunto de respostas B.

Pelo fato da incerteza da resposta crescer com a entropia, diz-se que a Entropia é, também uma **medida da incerteza** de um dado conjunto de resultados.

Existe uma fórmula para o cálculo da Entropia do conjunto X. Se um elemento do conjunto X é x_i, a componente de entropia deste elemento é:

$$E(x_i) = p(x_i) \log_2[1/p(x_i)]$$

E a entropia do conjunto é a soma das componentes de entropia de cada um de seus componentes. Note, porém, que se temos um caso extremo de um único resultado possível, com probabilidade 1, a entropia fica:

$$E(x_i) = 1 \log_2(1/1) = \log_2 1 = 0$$

O que isso significa? Significa que se só há um resultado possível, o resultado não representa qualquer informação. Um exemplo prático é um termômetro que sempre marca a mesma temperatura, independente da temperatura real. O valor indicado por ele não contém qualquer informação útil.

Considere este conceito aplicado às chaves de criptografia: se existe um número pequeno de chaves possíveis, a incerteza das chaves é pequena, ou seja, a entropia desta chave é pequena, tornando-a de "quebra" mais fácil.

Da mesma forma, se o número de textos legíveis possíveis para uma mensagem cifrada também é pequeno, este texto legível também tem uma entropia pequena. Quando a entropia é pequena, a criptografia é facilmente quebrável.

Como aumentar a entropia? Primeiramente, usando chaves grandes e com vasta variabilidade. Isso significa que não é bom usar chaves em que os elementos desta são inter-relacionados (como uma frase, por exemplo). Isso reduz a variabilidade desta chave, já que existirá uma relação entre seus elementos (as letras próximas necessariamente formam palavras existentes em alguma língua).

Uma segunda maneira é aumentando a entropia do texto legível que, em geral, é baixa devido à mensagem ser escrita em uma dada língua. A quebra da criptografia usando a freqüência de caracteres tem origem nesta baixa entropia: há caracteres com uma grande probabilidade de ocorrência, outros com baixa, reduzindo o conjunto de possibilidades.

A forma mais comum de se realizar esta tarefa é através de uma técnica chamada ***salting***. Para entender o *salting*, é preciso lembrar que, no computador, cada caractere de texto é, na realidade, um número de, por exemplo, 8 bits.

Assim, a letra "**A**", por exemplo, é representada pelo número **65**. Este número, em binário, é representado por **01000001b**. Se antes de cifrar o caractere "A" forem acrescentados 4 bits aleatórios ao fim deste número binário, o resultado é um número de **12 bits**, como por exemplo:

01000001 0011b

Como 4 bits permitem 16 combinações (0 a 15), isso significa que haverá 16 números de 12 bits que terão exatamente o mesmo significado: a letra A. Pode-se dizer, então, que o *salting* **aumenta a variabilidade** (cada letra agora poderá ser representada de 16 maneiras, aleatoriamente!), **aumentando a entropia** e tornando o algoritmo mais seguro. A partir de então, a criptografia é feita de 12 em 12 bits (e não mais de 8 em 8 bits) e, ao decriptar o texto, os 4 bits "mais baixos" (os mais da direita) devem ser eliminados para que o texto legível possa ser gerado.

Um exemplo onde isso é fácil de observar é apresentado abaixo. Imagine palavras formadas apenas pelas 4 primeiras letras do alfabeto: A, B, C e D.

Se for usado um código simples como o de César (com deslocamento de 3 letras) para codificar a frase:

CADA BABA DA CACA => FDGD EDED GD FDFD

Usando a idéia da freqüência das letras, mais uma vez é simples identificar que a letra com maior freqüência "D" (freqüências no texto: D=7, F=3, E=2, G=2) é a volgal A e com isso derivar a mensagem original.

Com a aplicação do **salting**, entretanto, a coisa muda de figura. Considere que as letras A a D sejam representadas pelos binários abaixo:

Letra	Decimal	Binário
A	0	00b
B	1	01b
C	2	10b
D	3	11b

Serão adicionados 2 bits aleatórios ao fim de cada letra:

Orig.,	Dec.	Bin.	Bin. + 2 bits	Dec. Final	Letra Final
C	2	10b	1001b	9	J
A	0	00b	0000b	0	A
D	3	11b	1100b	12	M
A	0	00b	0011b	3	D
B	1	01b	0110b	6	G
A	0	00b	0000b	0	A
B	1	01b	0111b	7	H
A	0	00b	0010b	2	C
D	3	11b	1111b	15	P
A	0	00b	0001b	1	B
C	2	10b	1001b	9	J
A	0	00b	0001b	1	B
C	2	10b	1000b	8	I
A	0	00b	0011b	3	D

Codificado pelo alfabeto de César:

```
JAMD GAHC PB JBID
IDPG JDKF SE MELG
```

Agora as letra com maiores freqüências são:

D=2, E=2, G=2, I=2, F=1, J=1, K=1, L=1, P=1, S=1

É possível observar que a freqüência foi grandemente alterada, tornando muito mais difícil qualquer suposição sobre a relação entre os elementos cifrados e os originais, legíveis.

Para reverter o processo, primeiro se decodifica com o Alfabeto de César e, posteriormente, elimina-se os bits aleatórios do *salting*.

8. EXERCÍCIOS

Código alfabético:

A- 00000b 0	H- 00111b 7	O- 01110b 14	V- 10101b 21
B- 00001b 1	I- 01000b 8	P- 01111b 15	W- 10110b 22
C- 00010b 2	J- 01001b 9	Q- 10000b 16	X- 10111b 23
D- 00011b 3	K- 01010b 10	R- 10001b 17	Y- 11000b 24
E- 00100b 4	L- 01011b 11	S- 10010b 18	Z- 11001b 25
F- 00101b 5	M- 01100b 12	T- 10011b 19	_- 11010b 26
G- 00110b 6	N- 01101b 13	U- 10100b 20	- - 11011b 27

Mensagem Cifrada (forma numérica):
12, 21, 11, 8, 112, 5, 109, 80, 85, 6, 109, 15, 62, 14, 6, 115, 16, 61, 50, 7, 110, 31, 23, 49, 7, 20, 8

1. Qual a freqüência de cada símbolo?

2. Sabendo esta mensagem foi cifrada pela Cifra de César com chave é +5, são no máximo 128 símbolos e que os 2 últimos bits da mensagem são de salting, qual a mensagem original?

3. Qual a freqüência de cada símbolo na mensagem original?

4. Neste caso, você acha que o *salting* funcionou ou não? Por quê?

9. BIBLIOGRAFIA

TERADA, R. **Segurança de Dados:** Criptografia em Redes de Computador. São Paulo: Edgard Bücher, 2000.

FERREIRA, F. N. F. **Segurança da Informação**. Rio de Janeiro: Ciência Moderna, 2003.

Unidade 7: Usos Práticos de Criptografia

Prof. Daniel Caetano

Objetivo: apresentar os conceitos para trabalhar com autenticação e identificação em programação ou usando PGP.

Bibliografia: TERADA, 2000.

INTRODUÇÃO

Conceitos Chave:
- Problemas
  * Garantir segurança mínima em aplicações multi-usuário
    + Codificação do Password
  * Criptografar mensagens e-mail e de comunicação instantânea
    + PGP

Antes de lançar mão de um maior aprofundamento com relação aos algoritmos de criptografia, serão apresentadas algumas situações em que se usam os esquemas de criptografia que serão vistos futuramente.

Tais esquemas estão presentes desde a programação ordinária para ambientes com múltiplos usuários até no simples uso de aplicativos de e-mail e comunicação instantânea. No primeiro caso, muitas vezes a tarefa é encriptar pelo menos o password. No segundo caso, muitas vezes a tarefa é relegada a softwares alternativos que atuam como um novo layer de aplicação, realizando toda a criptografia e checagem de usuários, como é o caso do PGP.

1. TRABALHANDO COM SENHAS (PASSWORDS)

Conceitos Chave:

- Sistemas com Múltiplos Usuários => Banco de Dados
  - * Informações dos usuários
  - * Login + Password
  - * Demais informações: úteis? / dispensáveis?
- Armazenamento direto
  - * Perigoso => roubo do banco de dados
  - * Login pela rede => password trafega pela rede
    - + Podem ser interceptados
  - * Problema similar: armazenamento de cartão de crédito
- Número *hash*, como uma "criptografia unidirectional"
  - * Codificar o dado para nunca mais decodificá-lo
  - * Por quê? Porque não é necessário.
  - * Exemplo com Criptografia.
- Mas o que é Hash?
  - * "Espalhamento"
  - * Função de Hash x Número de Hash
  - * Conceito de Número resumo
    - + Cada mensagem tem seu número
- O que é um bom hash?
  - * Cada mensagem tem um único número de hash
  - * Um hash diferente para cada mensagem
  - * Pode ser usado para armazenar dados que só precisam ser verificados
- Hash "Só de Ida" x Compressor (ida e volta)

Uma das atividades mais comuns quando se lida com múltiplos usuários é a necessidade de manter um banco de dados com as informações destes usuários. Tal banco de dados deve conter, minimamente:

- Nome de Login
- Senha

É claro que maiores informações (como nome do usuário, email, permissões de acesso, etc) são úteis e, em alguns casos, até mesmo indispensáveis. Entretanto, as informações básicas são, realmente, nome de login e senha.

Entretanto, o armazenamento deste tipo de informação em um banco de dados pode constituir uma brecha de segurança, se as medidas cabíveis não forem tomadas. Por exemplo: se alguém roubar os arquivos do banco de dados, automaticamente descobre a senha e o login de todos os usuários.

Mesmo que ninguém roube os arquivos de banco de dados, os dados ainda não estão seguros: em geral, o servidor de banco de dados não se localiza na mesma máquina em que é feito o login. Desta forma, para que o nome de login e a senha sejam verificados no servidor de banco de dados, é preciso que estes trafeguem pela rede. Ao trafegar pela rede, abre-se brecha para que tais dados sejam interceptados e, mais uma vez, surge uma falha de segurança.

Observe que este é um problema também quando temos o armazenamento de números de cartão de crédito no banco de dados da companhia de cartão de crédito. Imagine o que ocorreria se alguém simplesmente roubasse um banco de dados deste tipo! Mesmo que apenas um trecho do banco de dados seja roubado, os estragos poderiam ser enormes!

Como evitar (ou reduzir, ao menos) este tipo de problema? A técnica mais comum é o uso da criptografia, mas de uma maneira um pouco distinta do que é usual. É usada um algoritmo de *hash* que, em geral, pode ser considerada uma espécie de criptografia em uma única direção.

1.1. Hash como Criptografia em Apenas Uma Direção

O que seria este tal "hash" como uma criptografia em apenas uma direção? Sua função é a de um algoritmo criptográfico como outro qualquer, mas não se deseja descriptografar o dado, depois de criptografado. Nunca. Mas por que isso?

Considere o seguinte mecanismo: quando um usuário se cadastra, ele tem de digitar uma senha para o seu cadastro. Esta senha não é enviada para o banco de dados: antes ela é criptografada com algum algoritmo/chave qualquer de criptografia e o dado criptografado é enviado para armazenagem no banco de dados.

Digitado	Criptografado	Enviado na Rede	Armazenado
elefante	abfjsuee	abfjsuee	abfjsuee

Algum tempo depois, quando o usuário volta a tentar usar o sistema, ele precisa digitar seu nome e sua senha. Neste momento, logo após o momento em que o usuário digita a senha, ela é criptografada com o mesmo algoritmo/chave do momento do registro e, mais uma vez, a senha criptografada é enviada pela rede para a comparação com a existente no banco de dados.

Digitado	Criptografado	Enviado na Rede	Comparado
elefante	abfjsuee	abfjsuee	abfjsuee

Note que a senha nunca transitou pela rede ou ficou armazenada sem estar criptografada. Além disso, em momento algum essa informação é decriptada pelo sistema. Entretanto, essa informação poderia ser decriptada, usando o par algoritmo/chave corretos.

Como isso não é desejável (dado que a autenticação pode ocorrer sem decifragem alguma) usa-se um hash unidrecional como uma "criptografia unidirecional".

1.2. Uso de Hash

Primeiramente, é preciso especificar o que é um hash. Uma tradução comum para hash é "espalhamento" e, na verdade hash pode ser referir a uma "função de hash" ou a um "número de hash".

Uma função de hash é uma função que gera um número de hash. Mas o que seria um número de hash? Pois bem, um número de hash é um "número resumo" para uma dada mensagem. Isso significa dizer que este número deve ter uma tal relação com a mensagem que, se a mensagem for mudada, o "número de hash" vai mudar também.

```
hash = funcao_hash(texto);
```

Uma boa função de hash (aquelas consideradas seguras) deve gerar números de hash únicos para cada mensagem, de forma que exista um número de hash diferente para cada mensagem e cada mensagem tenha apenas um único número de hash.

Além disso, ao contrário do que ocorre na criptografia, a função de hash é, em geral, "só de ida". Isso significa que deve ser possível transformar qualquer mensagem em seu número de hash, mas, em geral, não deve ser possível transformar o número de hash de volta em sua mensagem.

> O caso de um hash em que é possível reverter o processo (ou seja, a partir do hash chegar na mensagem original) pode ser encarado como uma "compressão" de dados, como a que se obtém com o ZIP, por exemplo.

Como é possível observar, um hash que respeite:

- Mensagens diferentes não geram o mesmo número de hash;
- Não há mais de um hash para a mesma mensagem.

Pode ser usado com o mesmo propósito de armazenar senhas e números secretos que apenas precisam ser verificados (e não lidos) em bancos de dados. Ao invés de armazenar a informação original, armazena-se o hash. Sempre que for preciso checar se o valor entrado pelo usuário é correto, basta gerar o hash deste número e compará-lo com o hash do banco de dados.

Como não é possível gerar a mensagem a partir do hash (a não ser por força bruta), a informação estará armazenada com um nível de segurança aceitável. Mais detalhes sobre funções de espalhamento serão vistas oportunamente.

2. PGP - PRETTY GOOD PRIVACY

Conceitos Chave:

- "Criptografia às Massas"
  - * Identificar autores
  - * Criptografar/Decriptar mensagens
    - + Instantâneas
    - + E-Mail
- Algoritmos Usados
  - * RSA
  - * IDEA
  - * MD5
- Chave Pública e Privada
  - * Chave Pública do João: todo mundo usa para enviar mensagens ao João
  - * Chave Privada do João: só ele usa para ler as mensagens que recebe
- Conjunto de Chaves Públicas dos outros => Key Ring
  - * Para mandar mensagem
    - + Necessária chave pública do destinatário
  - * Para receber mensagem
    - + O remetente precisa de sua chave pública
  - * É preciso *publicar* a chave pública!
- Assinatura Eletrônica: verificar remetente
- PGP como Layer de Software (driver de rede)

O *Pretty Good Privacy* é um programa freeware para fornecer "criptografia às massas", por assim dizer. Ao longo dos anos o PGP evoluiu e hoje conta com ferramentas para identificar autores e encriptar/decriptar desde mensagens (de forma integrada com os mais comuns programas de e-mail) até mensagens instantâneas e pacotes de rede.

O PGP é um programa que usa os algoritmos RSA e IDEA para criptografia e MD5 para hash. Sua idéia de funcionamento é a seguinte: sempre que se cria um usuário no PGP, duas chaves são criadas: uma pública e uma privada. A chave privada, exclusiva da pessoa que criou o usuário do PGP, é armazenada de forma protegida em um diretório especial. A chave pública é guardada num diretório chamado "chaveiro" ("key ring", em inglês).

Para usá-lo, o usuário precisa adquirir chaves públicas das pessoas com quem entra em contato e distribuir sua chave pública para estas pessoas também. De uma forma geral, a chave pública serve para encriptar uma mensagem, e a privada para decriptar esta mensagem. Como todos possuem a chave pública do usuário em questão, todos podem encriptar mensagens para ele. Entretanto, só este usuário tem sua chave privada, então só ele pode decriptar as mensagens que foram encriptadas para ele.

Usando um princípio parecido, o PGP também faz uma assinatura eletrônica, usando a chave privada do usuário para criar tal assinatura na mensagem, que os outros usuários podem verificar usando a chave pública do usuário que assinou a mensagem.

Para que o usuário não precise ficar tomando contato com todos esses processos, o PGP pode ser instalado como um layer de software, atuando entre a aplicação e a camada de rede/armazenamento, de forma transparente. Assim, se um usuário instala o PGP, sempre que ele enviar um e-mail, ele irá com a assinatura eletrônica do usuário. Quando desejar enviar uma mensagem criptografada, ela o será, de acordo com a chave pública do destinatário. Quando uma mensagem encriptada chegar, ele automaticamente decripta a mensagem usando a chave privada do usuário.

3. EXERCÍCIO (LABORATÓRIO ou em casa)

Parte I - Criando e Trocando Chaves

1. Abrir o PGP Tools (Iniciar => Programas => PGP => PGP Tools)
2. Clique no primeiro botão (PGP Keys).
3. No PGP Keys, criar um par de chaves para si próprio (Keys => New Key), preenchendo Nome, E-mail e Passphrase.
4. Exportem sua chave pública para um arquivo (selecione a entrada do seu nome e siga Keys => Export).
5. Passem este arquivo para o colega do lado (por disquete, e-mail, etc)
6. Tendo recebido o arquivo do colega, importe a chave pública dele (Vá em Keys => Import e selecione o arquivo enviado por seu colega).

Parte II - Trocando Mensagens Criptografadas

1. Volte ao PGP Tools.
2. Os botões são:
1) Abre o PGP Keys.
2) Encripta uma mensagem.
3) Assina uma mensagem.
4) Encripta e assina uma mensagem.
5) Decripta/Verifica integridade de uma mensagem.
6) Apaga um arquivo de forma segura.
7) Limpa espaço livre do disco, eliminando restos de arquivos antigos.
3. Escreva uma mensagem qualquer e salve com o nome texto.txt ou texto.doc.
4. Encripte uma mensagem para você mesmo:
- clique no quarto botão (Encriptar e assinar), e selecione o arquivo que você criou.
- na próxima janela, arraste o seu nome para o bloco de baixo (receptores) e clique Ok.
- digite sua passphrase.
5. Tente descriptá-la na sua própria máquina, usando o quinto botão do PGP Tools.
6. Passe a mensagem encriptada para o colega do lado.
7. Pegue a mensagem do seu colega e tente descriptá-la, usando o 5o. botão.
8. Agora, repita o processo 4, mas dessa vez encripte para o seu colega.
9. Tente descriptá-la na sua própria máquina, usando o quinto botão do PGP Tools.
10. Passe a mensagem encriptada para o colega do lado.
11. Pegue a mensagem do seu colega e tente decriptá-la, usando o 5o. botão.

4. BIBLIOGRAFIA

TERADA, R. **Segurança de Dados:** Criptografia em Redes de Computador. São Paulo: Edgard Bücher, 2000.

Unidade 1: Verificação da Integridade Lógica

Prof. Daniel Caetano

Objetivo: Apresentar conceitos iniciais da verificação da integridade lógica e avaliar a integridade lógica através de métodos matemáticos simples.

Bibliografia: TANENBAUM, 2003; WILLIAMS, 2005; KNUTH, 1981.

INTRODUÇÃO

Conceitos Chave:

- Problemas:
    * Arquivo recebido pela rede. Está íntegro?
    * Número da conta digiado. Está correto?
- Integridade lógica => Não modificação indevida.
    * Propositais
    * Erros
- Conceito => Dígito de verificação
    * Tamanho do Documento => Tamanho do "Dígito de Verificação"
    * "Erros" propositais x Erros acidentais
- Números longos de 32 a 160 bits (4 a 20 bytes) => "resumo" ou "hash"
    * Funções de Hash
    * CheckSum
    * Cyclic Redundancy Check

Com a difusão das redes de compartilhamento de dados, um problema muito comum tem surgido: tendo recebido um arquivo, como verificar se este arquivo está completo, sem modificações que prejudiquem sua utilização? Esta análise é importante porque, em um executável, por exemplo, um único bit incorreto pode fazer com que ele não mais funcione!

Uma outra situação em que é importante saber se um dado está correto é quando é feita uma transferência bancária. Quando um cliente vai até um caixa eletrônico e digita um número de conta para realizar uma transferência de valores, um pequeno erro de digitação poderia fazer com que o valor fosse transferido para uma pessoa (ou empresa) completamente desconhecida. Isto não é interessante. Como permitir que o sistema detecte se o número de conta digitado está correto. Mas como realizar esta verificação?

Este tipo de verificação se chama "integridade lógica" e tem um papel fundamental na segurança da informação, uma vez que está relacionado à não-modificação indevida de um documento. Pouco foi visto, entretanto, sobre como verificar esta integridade lógica de uma informação ou documento.

O conceito é similar ao do dígito de verificação de uma senha, embora para documentos maiores que uma senha é comum o uso de um número de vários dígitos para verificação. A razão para isso é que com um único dígito numérico há apenas 10 variações possíveis; como há muito mais que 10 textos possíveis, até mesmo com tentativa e erro seria possível criar um texto totalmente modificado que produzisse o mesmo dígito de verificação.

Mas porque um único dígito seria suficiente para uma senha? Um único dígito é eficaz para prevenir erros acidentais em textos curtos, mas não necessariamente para prevenir erros propositais ou erros em textos muito longos.

Por esta razão, para textos mais longos, costuma-se usar um número de vários dígitos, sendo os números mais comuns de 32 a 160 bits. Estes números são chamados de "**números resumo**" ou "**números de hash**" e existe uma função que "lê" o texto original e gera este tipo de número. Tais funções são chamadas de "funções de hash".

Uma verificação de integridade muito comum é a Checagem de Redundância Cíclica (Cyclic Redundacy Check), que é um hash gerado como base no princípio de "soma de checagem" (checksum).

1. CHECKSUM

Conceitos Chave:

- CheckSum => (A + B) MOD N
  - * N => Usualmente potência de 2.
- Verificação de Checksum => Exemplo: (A + B) MOD 32
  - * Envio de mensagem => Exemplo: 10, 22, 5
- Cancelamento
  - * Dois erros podem se cancelar => pouco espalhamento
  - * Aumentar o tamanho => Não ajuda, sem ampliação do espalhamento!
    - = Aumentar tamanho => reduzir repetições de checksum
    - = Aumentar caos => reduzir chance de falhas se cancelarem
- Função de CheckSum => Procedimento de Cálculo
  - * Número => Não exclusivo, mas único para uma mensagem
    - = Não é bom para aplicações seguras!
  - * Velocidade e Rapidez; Verificação de "erros acidentais"
  - * Exemplos: Executáveis; Disquetes; FTP; ZIP => CRC

Um CheckSum (número de checagem) é algo como a soma dos valores de um texto, considerando sempre o resto de divisão por um dado número, em geral potências de 2. Seu uso é exatamente para verificar a integridade de um arquivo. Por exemplo, consideremos a **soma *mod* 32**:

Mensagem 1:	**10**	**12**	**5**		
Cálculo do checksum:	A) (10 + 12) MOD 32 =			22 MOD 32 = 22	
	B) (22 + 5) MOD 32 =			27 MOD 32 = 27	
Mensagem 1 com Checksum:	**10**	**12**	**5**	**27**	
---------- Transmissão					
Mensagem 1 depois da transm.:	**10**	**12**	**5**	**27**	
Mensagem 2:	**10**	**22**	**5**		
Cálculo do checksum:	A) (10 + 22) MOD 32 =			32 MOD 32 = 0	
	B) (0 + 5) MOD 32 =			5 MOD 32 = 5	
Mensagem 2 com Checksum:	**10**	**22**	**5**	**5**	
---------- Transmissão					
Mensagem 2 depois da transm.:	**10**	**22**	**5**	**5**	

Após o recebimento da mensagem, o checksum é recalculado *desconsiderando o checksum recebido.* Para saber se a mensagem chegou correta, compara-se, então, o checksum que acabou de ser calculado com o checksum recebido junto com a mensagem. Se estes forem iguais, a mensagem provavelmente está íntegra. Caso contrário, não está.

Como já dito anteriormente, este cálculo não é totalmente seguro: pode ocorrer uma seqüência de erros que se cancelem, ou seja, mesmo a mensagem tendo sido completamente alterada, o checksum continuará o mesmo. Uma das formas de diminuir essa probabilidade é aumentando o tamanho do checksum, usando restos de divisão maiores, como por 256, por 65535, por $4{,}29*10^9$, por exemplo (8, 16 e 32 bits, respectivamente).

Isso corrige casos como este:

A) Usando CheckSum com MOD 32

Mensagem 1:	10	12	5		
Cálculo do checksum:	A) (10 + 12) MOD 32 =			22 MOD 32 = 22	
	B) (22 + 5) MOD 32 =			27 MOD 32 = 27	
Mensagem 1 com Checksum:	10	12	5	27	
---------- Transmissão					
Mensagem 1 depois da transm.:	26	28	5	27	
Cálculo do checksum:	A) (26 + 28) MOD 32 =			54 MOD 32 = 22	
	B) (22 + 5) MOD 32 =			27 MOD 32 = 27	

Como o checksum calculado é **igual** ao recebido, a mensagem seria considerada *correta*... mas ela não está!

B) Usando CheckSum com MOD 64

Mensagem 1:	10 12 5	
Cálculo do checksum:	A) (10 + 12) MOD 64 =	22 MOD 64 = 22
	B) (22 + 5) MOD 64 =	27 MOD 64 = 27
Mensagem 1 com Checksum:	10 12 5 27	
----------		Transmissão
Mensagem 1 depois da transm.:	26 28 5 27	
Cálculo do checksum:	A) (26 + 28) MOD 64 =	54 MOD 64 = 54
	B) (54 + 5) MOD 64 =	59 MOD 64 = 59

Como o checksum calculado é **diferente** do recebido, a mensagem passou a ser considerada **incorreta**, como deveria ser.

Infelizmente, esta mudança no tamanho do checksum **pode não resolver sempre**. Em um caso como como este:

Mensagem 2:	10 22 5	
Cálculo do checksum:	A) (10 + 22) MOD 32 =	32 MOD 32 = 0
	B) (0 + 5) MOD 32 =	5 MOD 32 = 5
Mensagem 2 com Checksum:	10 22 5 5	
----------		Transmissão
Mensagem 2 depois da transm.:	10 21 6 5	
Cálculo do checksum:	A) (10 + 21) MOD 32 =	31 MOD 32 = 31
	B) (31 + 6) MOD 32 =	37 MOD 32 = 5

A mensagem claramente chegou alterada no destino, mas o checksum continua "batendo". Num caso como este, não é possível resolver simplesmente aumentando o tamanho do número do checksum, porque um erro está sendo cancelado por outro, devido à natureza simplista do cálculo realizado (soma dos elementos).

Por esta razão, é preciso de algo mais complexo, em que uma pequena variação em um byte da mensagem gere uma grande mudança no checksum, de forma a minimizar a chance de um errro cancelar outro erro. Em outras palavras, são necessárias duas coisas para quem um checksum seja bom:

a) Tamanho: para diminuir a chance de falha por repetição de checksum;
b) Caos: para diminuir a chance de cancelamento de falhas.

1.1. FUNÇÃO DE CHECKSUM

Os checksums oferecem fórmulas que, aplicadas em um texto, geram um número de tamanho arbitrário que, embora não seja exclusivo de uma única mensagem, é único para cada mensagem. Em outras palavras, um checksum pode ser igual para duas mensagens

diferentes, mas cada mensagem tem um único checksum relativo à ela. Como foi visto anteriormente, isso faz com que este **não seja um "bom hash" para aplicações seguras**.

Entretanto, ainda que o checksum possa ser igual para duas mensagens diferentes, a chance de repetição é baixa e a velocidade/facilidade com que ele é calculado pelos processadores o tornam bastante atraente. Os checksums não costumam ser usados para verificar se houve uma modificação intencional, mas sim se ocorreram modificações não-intencionais, por ruídos/erros de transmissão e/ou armazenamento.

De fato, a maioria dos sistemas operacionais usa um tipo de checksum, o CRC (Cyclic Redundancy Check) para verificar a *integridade de um arquivo executável* e, caso ocorra um Erro de CRC (o CRC fornecido pelo executável não "bate" com o CRC calculado pelo Sistema Operacional ao carregar o programa), o Sistema se recusa a executar aquele arquivo.

A CRC também é usada na leitura de disquetes (para verificar se os dados estão íntegros), podem ser usadas em transferências de arquivos e em compressores de arquivos, para verificar a integridade de um arquivo comprimido ao ser extraído.

2. IDÉIA DO CRC

Conceitos Chave:

- Cálculo de CheckSum
  - * Soma x Divisão
    - = Tamanho do Divisor ~ Tamanho do CheckSum
- Resto da Divisão (MOD)

Como já foi visto, a soma não é um método bom para o cálculo de um "checksum", uma vez que dois (ou mais) erros podem se cancelar facilmente. Entretanto, é possível observar que a divisão é um bom método, sendo uma operação de difícil cancelamento, contanto que o **divisor seja aproximadamente do tamanho do checksum em si** (ou seja: se o checksum tem 32 bits, o divisor deve ter aproximadamente 32 bits também).

A idéia por trás do CRC é tratar a mensagem como um grande número binário, e dividi-lo por um número fixo. Por exemplo, imaginemos que a mensagem seja composta por 2 bytes: 10 e 22 formam um número binário "gigante" da seguinte forma:

```
     10            22
0000.1010b    0001.0110b
--------------------------------
  0000.1010.0001.0110b
          2582
```

Consideremos agora o divisor como sendo o número 47 (um número de 8 bits qualquer) e realizamos a conta:

2582 / 47 = 54, com resto 44.

Assim, o número de checagem seria 44. O que aconteceria com o número de checagem se um BIT desta mensagem tivesse sido modificado na transmissão? Vejamos:

```
          2582
  0000.1010.0001.0110b      = Número transmitido
--------------------------- Transmissão
  0000.1110.0001.0110n      = Número recebido
          3606
```

Dividindo 3606 por 47...

3606 / 47 = 76, com resto 34.

Embora a variação de alguns bits específicos produzam pouca (ou nenhuma) mudança no valor da divisão, bits diferentes produzem variações muito diferentes no **resto** da divisão.

Assim, a idéia de tomar a mensagem toda como um grande número e achar o resto da divisão deste número por outro é muito mais interessante do que a simples soma de cada termo com um resto de divisão em cada operação.

O algoritmo CRC em si é bastante mais complexo, pois considera que os bits são coeficientes de um polinônio (**5 => 101b => $1*x^2 + 0*x^1 + 1*x^0 = x^2 + 1$**, por exemplo) e a divisão ocorre seguindo as regras de uma divisão polinomial, que é diferente das regras de divisão numérica. Entretanto, mantém-se a idéia apresentada.

3. NÚMERO DE VERIFICAÇÃO

Conceitos Chave:

- Operações com os elementos de uma mensagem
  - * Soma x Subtração x Divisão x Multiplicação...
  - * Resto da divisão pelo maior número de verificação possível + 1

O CRC é uma forma muito boa de verificar a integridade de uma mensagem, mas também uma bastante complexa. É comum, em geral, regras serem criadas para números específicos. CPF e CNPJ, por exemplo, possuem uma regra para dígito de verificação. Da

mesma forma, cada banco tem suas regras para o cálculo do dígito de verificação de números de agência e conta.

Estas regras nada mais são que cálculos sistemáticos envolvendo os elementos de uma mensagem e utilizando o truque do resto de divisão para limitar o tamanho do número de verificação.

4. ATIVIDADE

Em um sistema, é necessário checar se o número do cartão de funcionário foi digitado corretamente. Os números são no formato:

a b c d e f - g

Onde de "a" a "f" são os 6 dígitos (de 0 a 9) do número) e "g" será o dígito de verificação. Elabore uma regra de cálculo única para o dígito de verificação de forma que cada uma das seqüências abaixo tenha dígitos diferentes entre si, e o dígito varie de 0 a 9.

a) O dígito de 010101 deve ser diferente do de 101010
b) O dígito de 421111 deve ser diferente do de 241111
c) O dígito de 123456 deve ser diferente do de 123455

5. BIBLIOGRAFIA

TANENBAUM, A. S. **Redes de Computadores**. 4a. ed. Elsevier, 2003.

WILLIAMS, R. N. **A Painless Guide to CRC Error Detection**. < http://www.repairfaq.org/filipg/LINK/F_crc_v3.html > Consultado em 17/04/2005.

KNUTH, D.E. **The Art of Computer Programming**, v2. 1981.

Unidade 2: HASHs Criptográficos

Prof. Daniel Caetano

Objetivo: Apresentar conceitos da verificação da integridade lógica com hashs e hashs criptográficos.

Bibliografia: TERADA, 2000.

INTRODUÇÃO

Conceitos Chave:

- Problemas:
  - * Texto adicional já existe no banco de dados?
  - * Já existe um objeto em uma lista de objetos?
  - * Uma mensagem foi modificada?
- Comparação byte-a-byte x Número Resumo
- Checksum x Hashs Criptográficos

Quando se realiza a programação de um banco de dados de textos ou imagens, é comum a necessidade de verificar se um texto ou imagem a ser inserido no banco de dados já se encontra no mesmo.

A forma simples de fazer isso é comparar byte por byte do novo texto ou imagem... com todos os textos e/ou imagens existentes no banco de dados. Por uma série de razões, isso pode ser bastante lento e desajeitado.

Como já foi dito inúmeras vezes nos cursos básicos, os computadores só entendem números. Textos são uma seqüência enorme de números que, por muitos processos matemáticos são transformados nas imagens das letras que compõem uma frase na tela. Ocorre que para o computador comparar dois textos e verificar se são iguais, precisa verificar cada um destes números, um a um, até encontrar uma diferença. Para "provar" que ambos os textos são iguais ele precisará comparar todos os números que os compõem. Para um texto (ou imagem) de alguns megabytes (ou mesmo gigabytes) isso pode ser um desperdício de tempo absurdo.

Uma forma mais adequada de fazer isso, seria calcular uma espécie de número resumo para cada texto/imagem acrescentado no banco de dados e, quando um novo precisar ser adicionado, basta calcular seu número resumo e verificar se aquele número resumo já existe.

Assim, a condição é que cada número resumo seja único para cada arquivo; se o arquivo foi modificado, ele terá outro número resumo - sendo praticamente impossível modificar uma imagem de forma que ela volte a ter o mesmo número resumo de antes da modificação.

Assim, a utilização de *checksums* não resolvem o problema, já que eles não servem para este tipo de coisa. Apesar de serem interessantes e práticos para a verificação de erros de acidentais nas mensagens, tais como erros de transmissão e/ou erros de digitação, eles não servem para o uso acima apresentado.

Nestes casos, serão usados os chamados *hashs criptográficos*, que são números resumos bastante mais robustos (e, em geral, maiores) que os *checksums* simples.

1. "HASHs CRIPTOGRÁFICOS" x CHECKSUMs

Conceitos Chave:

- Eficácia do Checksum: Modificações Intencionais x Não-Intencionais
- Hash => Espalhamento
  - * Mudanças no texto => Mudanças no hash
- Checksum é Hash? => Hash de Baixa Segurança x Alta Segurança
- Checksum x Hash: Segurança x Custo x Detecção de Modificações
- Conceito de Colisões
  - * Ocorrência de textos diferentes com o mesmo hash
  - * Garantia de não-colisões?
    - = Hash Só-de-Ida: perda de informação: CRANIO x COATI
      - : Hashs Diferentes - Textos Diferentes
      - : Hashs Iguais - Nada se pode dizer
- Não Colisão x Reversibilidade
  - * Hash Ida-e-Volta
    - = Volta... é bom?
    - = Sem perda de informação...
      - : Comparação direta de hashs possível
    - = Tamanho?

Como dito anteriormente, a verificação de integridade oferecida pelos algoritmos de checksum não são suficientes para garantir a integridade da informação contra modificações intencionais aos documentos (e, em alguma extensão, podem falhar mesmo no caso de modificações não intencionais).

Por esta razão, foram criados algoritmos de hash (ou espalhamento) mais "poderosos", fazendo com que qualquer alteração no documento implicaria em uma mudança em seu hash.

O nome "espalhamento" vem de um aumento da componente caótica resultante do método de cálculo, fazendo com que, supostamente, uma alteração num texto nunca pudesse ser cancelada por outra alteração no mesmo texto, em termos do número de hash (ou número resumo).

Vale ressaltar que *checksums também são números de hash*, embora sejam, em geral, de "segurança mais baixa". Neste texto, serão chamados de hashs apenas aqueles com finalidade e qualidade criptográfica (a menos que o contrário esteja explícito), reservando o nome checksum a maioria dos hashs não-criptrográficos.

Apesar de seus benefícios, o cálculo de hashs criptográficos é, via de regra, mais lento e complexo que o cálculo de checksums. Por esta razão, o uso de checksums não foi substituído pelo uso de hashs criptográficos, uma vez que o checksum é bom o suficiente para lidar com modificações não-intencionais nos textos. Resumidamente:

Tipo:	**Checksum**	**Hash Criptográfico**
Segurança:	Baixa	Média/Alta
Custo Computacional:	Baixo	Médio/Alto
Modificações detectadas:	Não intencionais	Intencionais e Não-Intencionais

O hash criptográfico serve também para detectar modificações não-intencionais mas, por necessitar cálculos mais complexos, em geral acaba sendo preterido frente ao checksum, nestes casos.

1.1. Segurança em Hashs - Colisões

O conceito de "segurança" alta ou baixa de um algoritmo de hash normalmente está relacionado ao número de mensagens possíveis com um mesmo número de hash, ou seja, quantos "textos" existem (compreensíveis ou não) com o mesmo número de hash. Este número é chamado de *número de colisões*.

Em um checksum usual, o número de colisões é alto. Já em um bom hash criptográfico, espera-se que o número de colisões seja mínimo ou inexistente (ao menos para "textos" compreensíveis.

Mas porque as colisões ocorrem? Na realidade, as colisões ocorrem porque, em geral, deseja-se que o hash seja "*só-de-ida*". Isto significa que, além de todas as características já citadas, em geral deseja-se que não seja possível descobrir a mensagem original a partir do hash. Em geral, um hash é "só-de-ida" porque *há perda de informação* em seu cálculo. Assim, o conceito de um hash "só-de-ida" sem colisões talvez seja impossível na prática, já que:

a) Para não haver colisões, não pode haver perda de informação na geração do hash. Uma vez que toda a informação está no hash, teoricamente seria possível encontrar a operação inversa que forneceria o texto a partir do hash (um arquivo ZIP?).

b) Se há perda de informação na geração do hash, toda mensagem que for diferente apenas nas informações perdidas, terá o mesmo hash! Ex.: Se o cálculo do hash for "pegue apenas as letras ímpares", o texto "COATI" teria o mesmo hash que "CRANIO": "CAI".

Assim, a existência de colisões é um fato e, portanto, a existência de mensagens totalmente diferentes com um mesmo número de hash causa algumas considerações, ao comparar duas mensagens pelo seu hash:

a) Hashs diferentes => Textos diferentes
b) Hashs iguais => Textos *potencialmente* iguais.

Em outras palavras, textos com hashs diferentes sempre serão diferentes, porém textos com hashs iguais nem sempre serão hashs iguais.

1.2. Reversibilidade de Hashs

Quando se criptografia um texto puro, obtendo assim o texto cifrado, é interessante que o processo será reversível; em outras palavras, é interessante que seja possível obter o texto puro a partir do texto cifrado.

Quando existe este processo de "ida-e-volta", ou seja, sempre é possível gerar um código e um texto e depois voltar ao texto a partir do código, diz-se que há *reversibilidade*.

Apesar de ter sido apresentado que os hashs são, em geral, "só-de-ida", ou seja, sem reversibilidade, é possível construir um algoritmo de hash de forma que haja reversibilidade; entretanto, estes tipos de hash não são usuais em criptografia e segurança da informação.

Ainda assim, se adequada a a reversibilidade do hash, o algoritmo de hashs deverá ser construído de maneira que o cálculo do número seja feito sem perda de informações, e o número hash de cada mensagem será exclusivo. Desta forma, nunca duas mensagens diferentes terão o mesmo número hash.

Este tipo de hash é perfeito para o problema inicial proposto, em que se deseja verificar se um texto/imagem já existe em um banco de dados. Adicionalmente, serve para verificar se dois objetos quaisquer são iguais, sem ter que ficar comparando valor a valor do mesmo.

2. HASHS INDEXADORES

Conceitos Chave:

- Comparação de dados grandes => comparação do hash
- Hash Só-de-Ida
  - * Hashs Diferentes, textos diferentes
  - * Hashs Iguais, comparação byte-a-byte dos textos
- Hash Ida-e-Volta
  - * Comparação direta dos hashs
  - * Chave Primária em BD => É hash?
- Uso Preferencial: Hash Só-de-Ida
  - * Velocidade? => Difícil Comparar
  - * Tamanho

Desde o começo falou-se no uso dos hashs como elementos para auxiliar a comparação de imagens e textos. A este tipo de uso dá-se o nome de *indexação*. Hashs usados com esta função são chamados de *hashs indexadores*.

O mecanismo básico é sempre manter um número de hash atualizado para cada objeto comparável. Sempre que for necessário verificar se um novo objeto já existe no sistema ou no banco de dados, basta gerar seu hash e comparar com os hashs. já existentes.

Há alguma diferença no processo, entretanto, quando se usa hashs só-de-ida ou hashs de ida-e-volta.

2.1. Hashs Indexadores de Só-de-Ida

Quando se faz a indexação com o uso de hashs só-de-ida, é importante lembrar que é possível afirmar que hashs diferentes indicam mensagens diferentes... mas que hashs iguais não dizem nada sobre as mensagens.

Assim, comparando-se os hashs "só-de-ida" de duas mensagens, se estes forem diferentes é possível afirmar que as mensagens são diferentes. Por outro lado, se os hashs forem iguais, deve-se partir para a verificação byte-a-byte dos mesmos.

Isso já traz um ganho significativo, porque evita-se a comparação byte-a-byte com a maioria das mensagens.

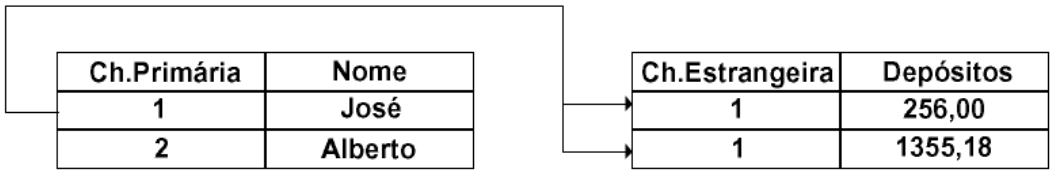
2.2. Hashs Indexadores de Ida-e-Volta

No caso do uso hashs de ida-e-volta para a indexação, como o hash é, de fato, único para cada mensagem, basta comparar diretamente. Se os hashs forem diferentes, as mensagens são diferentes; se os hashs forem iguais, as mensagens serão iguais.

Devem entrar no cálculo deste tipo de hash todos os bytes que o compõem o dado (mensagem, texto, imagem...). No caso de objetos de programação, o hash pode ser calculado com os valores dos atributos. Mais uma vez, ressalta-se a característica de *número resumo*. Sendo este número perfeito e único para cada conteúdo de um texto ou dos atributos de um objeto, ele pode ser usado para comparar estes textos ou objetos.

Ao invés de se comparar **todos** os bytes de um texto/imagem, basta comparar o número **hash** da mensagem. Da mesma forma, no caso de banco de dados, é possível comparar duas tuplas inteiras de uma dada relação apenas comparando seus "números resumo".

De uma certa forma, como alguns podem imaginar, as chaves primárias dos bancos de dados relacionais atuam como uma espécie de hash deste tipo: a chave primária resume uma tupla de uma dada relação. Se for necessário indicar uma dada tupla em uma outra relação é relativa àquela linha da primeira relação, indicamos apenas este número, quando ele passa a ser chamado de chave estrangeira.



Ch.Primária	Nome
1	José
2	Alberto

Ch.Estrangeira	Depósitos
1	256,00
1	1355,18

Note, porém, que este não é exatamente um hash, já que o número resumo *não guarda qualquer relação com a mensagem original*, e portanto é necessária uma tabela de conversão.

2.3. Hashs Indexadores de Só-de-Ida x Ida-e-Volta

Se ambos servem para a tarefa de indexação, qual deles é o mais usado?

Embora a resposta mais óbvia seja o de "Ida-e-Volta", já que este parece mais adequado à tarefa, esta não é a resposta encontrada na prática.

Apesar de todas as vantagens do hash de Ida-e-Volta, em geral é muito difícil gerar um número pequeno com estes algoritmos. Em alguns casos, como arquivos binários (imagens, por exemplo), os hashs serão sempre muito grandes, eliminando a principal vantagem da comparação por números resumo.

Assim, como nos hashs "Só-de-Ida" é possível produzir números relativamente pequenos (160 a 512 bits - 20 a 64 bytes, nos casos mais usuais), acaba sendo preferível lidar com os mesmos.

3. ALGORITMOS DE HASH COMUNS

Conceitos Chave:

- Mais usados: MD5, SHA-1, SHA-2
- Usos
  - * CRC => transferência de mídia e rede
  - * MD5 => Verificação de Download de CDs / DVDs / Índice de Música
  - * Reed-Solomon => transferência de mídia
  - * SHA-1 => Alternativo ao MD5, deixando de ser usado
  - * SHA-2 => Mais usado atualmente, sem padrão
  - * SHA-3 => Aceitando propostas. Mais provável: MD6.
- Depreciados
  - * MD4 => Muitas colisões!
  - * SHA-0 => Muitas colisões!
- Custo computacional e velocidade (aproximado)
  - CRC < Reed-Solomon < MD4 < SHA-0 < MD5 < SHA-1 < SHA-2

Existem vários algoritmos de hash, cada qual com maior ou menor segurança. Apesar disso, todos os hashs aqui citados são usado em finalidades específicas. Alguns dos hashs seguros mais comuns são o MD5, SHA-1 e SHA-2, mas existem outros, até mesmo em desenvolvimento, além de outros classificados como checksum .

- O CRC, cuja idéia já foi apresentada, é um hash que é classificado como checksum, por apresentar muita colisão. É usado primariamente para verificar dados corrompidos através da transferências de informações entre mídias e/ou por rede.

- O MD5 é usado basicamente em aplicações onde se quer uma maior qualidade quanto à verificação de integridade, mas ainda não é uma questão de segurança, pois as colisões neste algoritmo são relativamente altas (comparado a outros hashs criptográficos). Aplicações deste tipo de algoritmo são, por exemplo, para verificar a integridade de imagens de CDs ou DVDs. Um outro exemplo de MD5 é na identificação de arquivos de música, como uma espécie de número de índice.

- O Reed-Solomon, assim como o CRC, é um hash que é classificado como checksum, também por apresentar muita colisão. É usado primariamente para verificar dados corrompidos através da transferências de informações entre mídias (discos, por exemplo).

- O SHS/SHA-1, por sua vez, é um hash criptográfico, usado ainda em alguns dos sistemas de criptografia. Algumas pessoas usam o SHA-1 como uma alternativa aos usos do

MD5. O SHA-1 também não é mais recomendado, devido a um número de colisões descoberto recentemente (após cerca de 2^{63} operações, nos piores casos).

- O SHA-2 (SHA 224/256/384/512) é o algoritmo de hash criptográfico mais usado atualmente (como nos certificados de segurança digital da VeriSign), embora não tenha uma forma totalmente padronizada. Não existem falhas (colisões) detectadas até o momento.
- O SHA-3 será padronizado em breve, possivelmente baseado no código MD6.

Os algoritmos "depreciados":

- O MD4 (e anteriores), basicamente, não é mais utilizado. Ele apresenta colisões com muita freqüência (após cerca de 2^{20} operações).
- O SHA-0, também não é mais utilizado. Ele apresenta colisões com freqüência (após cerca de 2^{39} operações).

Dentre os hashs criptográficos citados, o MD4 é o mais rápido e o SHA-2 (512) é o mais lento. E é importante lembrar que, até o ano de 2007, com exceção do o SHA-2 e do SHA-3/MD6, todos eles já foram "quebrados", ou seja, foram detectadas colisões.

4. CARACTERÍSTICAS DO MD5 E DO SHA-1

Conceitos Chave:

- MD5
  - * 64 constantes (32 bits cada)
  - * Mensagem em blocos de 512 bits (16 valores de 32 bits)
  - * Saída em 128 bits (4 valores de 32 bits)
    - = Inicializados com valores específicos e alterados a cada iteração
  - * Fatores: bloco da mensagem, valor atual da saída e as constantes
- SHA-1
  - * Mensagem em blocos de 512 bits
  - * Saída em 160 bits (5 valores de 32 bits)
    - = Inicializados com valores específicos e alterados a cada iteração
  - * Fatores: bloco da mensagem, valor atual da saída

4.1. MD5:

A) O MD5 usa 64 constantes de 32 bits Tj, onde $T_j = 2^{32}*\text{sen}(j)$, com j de 1 a 64.

B) A entrada do MD5 é um bloco de 512 bits (16 valores de 32 bits) (m_0 a m_{15}).

C) A saída do MD5 é um bloco de 128 bits (4 valores de 32 bits) (d_0 a d_3), que devem ser inicalizados com valores específicos.

D) O algoritmo usa os valores m_0 a m_{15} para modificar os valores d_0 a d_3 em cada um de seus 4 passos.

Se a mensagem for menor que 512 bits, ela deve ser complementada até ter este tamanho. Se ela tiver mais que 512 bytes, deve ser quebrada em vários blocos de 512 bytes sendo que os d_0 a d_3 da saída do primeiro bloco são os d_0 a d_3 usados como entrada para o processamento do segundo bloco.

4.2. SHA-1:

A) A entrada do SHA-1 é uma mensagem que deve ter tamanho múltiplo de 512 bits.
B) A saída é um número e 160 bits (5 valores de 32 bits), que devem ser inicalizados com valores específicos.
C) Em cada passo, são processados 512 bits da mensagem original.
D) O algoritmo usa os valores dos 512 bytes para modificar os 5 valores de saída em cada um de seus passos.

5. ATIVIDADE

Uma empresa de cinema está desenvolvendo um sistema de armazenamento digital composto por um array de vários petabytes (milhares de terabytes). Este sistema será um enorme banco de dados de *versões de produção dos filmes* e a empresa já contratou a Oracle para desenvolver uma versão de seu banco de dados otimizada para suas necessidades.

Neste panorama, sua empresa foi contratada para desenvolver o sistema que manipulará este banco de dados; as necessidades são as seguintes:

1) O arquivo de um filme sempre será enviado com o mesmo nome, mesmo que sejam várias versões do mesmo;
2) Sempre que um arquivo de filme for enviado, se ele for diferente das versões já existentes no banco de dados, a nova versão deverá ser armazenada, com uma indicação da versão (por exemplo: se a última versão daquele filme armazenada era a versão número 3, a nova deverá ser a número 4).
3) Se uma versão já existente no banco de dados estiver sendo re-enviada por engano, ela deverá ser ignorada.
4) Estes filmes são armazenados apenas como arquivo, e não serão reproduzidos a partir do banco de dados.
5) Os filmes não podem perder qualidade no armazenamento.

Tendo conhecimento destas necessidades, qual o procedimento que o sistema deve realizar ao armazenar um filme? Explicite as tecnologias necessárias.

6. BIBLIOGRAFIA

TERADA, R. **Segurança de Dados:** Criptografia em Redes de Computador. São Paulo: Edgard Bücher, 2000.
Páginas 179 a 188 (Seção 7 a 7.3)

Unidade 3: Criptografia de Chave Simétrica

Prof. Daniel Caetano

Objetivo: Apresentar conceitos de criptografia simétrica e os principais algoritmos.

Bibliografia: TERADA, 2000, TANENBAUM, 2003.

INTRODUÇÃO

Conceitos Chave:

- Problema:
  - * Proteger informação armazenada
  - * Proteger informação a ser transmitida
- Como escolher criptografia?

Nos tempos atuais é cada vez mais comum a necessidade de transmissão de informações por redes. E é neste universo que um dia é solicitado que o sistema desenvolvido pela empresa passe a proteger as informações de nível de segurança mais alto, usando um tipo de criptografia.

Entretanto, existem diversos mecanismos de criptografia... qual escolher? Fica muito difícil fazer uma escolha, já que ainda não foi formalizada a classificação das mesmas e nem foram apresentados os principais algoritmos usados. Mas, antes disso, é preciso ressaltar alguns conceitos.

1. PRINCÍPIOS DE CRIPTOGRAFIA

Conceitos Chave:

- 1o. Princípio: algoritmos devem ser públicos, apenas as chaves são secretas
- 2o. Princípio: quanto maior a chave, maior a segurança (128 a 2048 bits)
- 3o. Princípio: o processo criptográfico é composto de substituição e transposição
- 4o. Princípio: cifragem perfeita é baseada no one-time-pad
  - * Lógica reversível => XOR
  - * Criptografia Quântica

O primeiro princípio importante, já apresentado de maneira informal, é o *Princípio de Kerckhoff*:

"*Todos os algoritmos devem ser públicos; apenas as chaves são secretas*"

Este princípio reforça que *não se deve partir do pressuposto de segurança pela obscuridade*.

O segundo princípio, no caso de um bom algoritmo de criptografia, é sobre o tamanho a chave:

"*Quanto maior a chave, maior a segurança*"

Bons algoritmos com chaves pequenas são facilmente quebráveis. O tamanho adequado de chave varia com o uso, com o algoritmo e com o tipo de criptografia sendo usada. Chaves de 128 bits podem ser grandes o suficiente em alguns casos, mas em outros podem ser necessárias chaves de 2048 bits, por exemplo. Nos tempos atuais, chaves de 32 e 64 bits raramente são suficientes para manter a confidencialidade da informação.

O terceiro princípio apresentado diz que os algoritmos de criptografia são, normalmente constituídos de duas técnicas criptográficas fundamentais: a **substituição** e a **transposição**, com algumas versões mais avançadas como a Cifra de Vigenère.

A substituição era a troca de caracteres por outros e, em sua variação mais simples (como na Cifra de César) preserva a freqüência dos caracteres. Numa versão mais avançada, como a Cifra de Vigenère, essa freqüência não é preservada. Os algoritmos de **transposição** são aqueles em que, ao invés de trocar os elementos da mensagem por outros diferentes, apenas **trocamos a ordem em que estes elementos** ocorrem na própria mensagem:

ABACATE => (SUBSTITUIÇÃO) => BCBDBUF
ABACATE => (TRANSPOSIÇÃO) => CATEABA

O quarto princípio é o da cifragem perfeita, em que o "*One-Time-Pad*" com uma cifra do tamanho da mensagem original, se usada uma única vez, era um exemplo. A operação seria simplesmente realizar uma **lógica reversível**.

Um exemplo de lógica reversível é a operação "ou exclusivo" (ou *eXclusive OR*, **XOR**) entre os dados e a chave. Por exemplo:

Tabela Verdade do XOR:

A	B	A *XOR* B	(A *XOR* B) *XOR* B = A
0	0	0	0
0	1	1	0
1	0	1	1
1	1	0	1

No fundo, **XOR indica 1 quando os valores são diferentes e 0 quando são iguais**. Ao considerar o resultado (A XOR B) e aplicar XOR B neste valor, volta-se ao valor original A, mostrando a reversibilidade da operação.

Em texto:	**A**	**B**	**A**	**C**	**A**	**T**	**E**
	X	X	X	X	X	X	X
	O	O	O	O	O	O	O
	R	R	R	R	R	R	R
	B	**O**	**S**	**Q**	**U**	**E**	**S**
Resultado:	Não imprimível						
Em Hexadecimal:	**41**	**42**	**41**	**43**	**41**	**54**	**45**
	X	X	X	X	X	X	X
	O	O	O	O	O	O	O
	R	R	R	R	R	R	R
	42	**4F**	**53**	**51**	**55**	**45**	**53**
Resultado (Cifrado):	**03**	**0D**	**12**	**12**	**14**	**11**	**16**
	X	X	X	X	X	X	X
	O	O	O	O	O	O	O
	R	R	R	R	R	R	R
	42	**4F**	**53**	**51**	**55**	**45**	**53**
Resultado (Decifrado):	**41**	**42**	**41**	**43**	**41**	**54**	**45**

Criptografia do tipo "one time pad" não é muito prático, pois *grandes mensagens exigem de grandes chaves*. Entretanto, pode ser que este tipo de chave venha a ter um papel importante na chamada criptografia quântica. A criptografia quântica é baseada em uma forma complexa de transmissão de dados através de fibra óptica onde é possível transmitir uma chave para uma segunda pessoa sem que uma intermediária consiga obter tal informação. Finalmente, a criptografia quântica permite até mesmo identificar que a "linha" está grampeada. Uma explicação introdutória mais detalhada sobre criptografia quântica pode ser encontrada em TANENBAUM (2003).

2. PRINCÍPIOS ADICIONAIS

Conceitos Chave:

- 5o. Princípio: Mensagens devem conter alguma redundância
  - * Redundância em excesso: facilita criptoanálise
  - * HASH / Checksum
- 6o. Princípio: É necessário algum método para evitar repetição
  - * Informação de horário

Além dos princípios já citados, há dois outros que visam garantir algumas características interessantes às mensagens, como será visto a seguir.

2.1. "As mensagens a serem criptografadas devem conter alguma redundância"

Pelo que já foi apresentado, é razoável considerar que mensagens curtas são menos "seguras", no sentido que podem ser mais facilmente decifradas em mensagens válidas (ainda que incorretas, com o uso de chaves incorretas).

Mensagem		Chave		Mensagem Cifrada
BICA	=>	ABCD	=>	CKFE
Mensagem		Chave		Mensagem Cifrada
CKFE	=>	AJNJ	=>	DUTO

Por essa razão, mensagens curtas dificultam a identificação de adulteração na mensagem; a solução para isso é, muitas vezes, acrescentar alguma redundância na mensagem, de forma a aumentá-las de tamanho. Por outro lado, é importante lembrar que *redundância em excesso facilita a criptoanálise*.

Um exemplo onde a adição de redundância facilita a identificação de adulteração na mensagem pode ser encontrado em banco de dados. Consideremos, por exemplo, um número de dois dígitos que representa a quantidade de um pedido de compra. Ora, dependendo do algoritmo, qualquer chave usada descriptografará a mensagem em um número de 2 dígitos válido. Se o mesmo foi alterado por terceiros, pedidos em quantidades incorretas serão realizados.

A solução para isso pode ser feita acrescentando alguns dígitos "zero" na frente do número, por exemplo, considerando 5 dígitos, dos quais os 3 primeiros têm de ser zero, qualquer chave que descriptografe este número para algo em que os três primeiros dígitos não são zero estará revelando que os dados foram alterados. Uma outra forma mais eficiente é enviar o hash da mensagem juntamente com a mesma (no caso de um único número pode ser um simples dígito de verificação).

2.2. "Algum método é necessário para anular ataques de repetição"

Imagine que alguém intercepte uma mensagem criptografada e, de tempos em tempos a retransmita. Dependendo da mensagem (um alerta de incêndio ou uma compra, por exemplo) isso pode causar transtornos sérios.

Uma alternativa para evitar este tipo de ataque é indicar, de alguma forma, a hora da mensagem nela mesma. Assim, uma mensagem que chegue fora de sua janela de horário (por exemplo, mais de 30s depois do envio) será simplesmente descartada como uma "mensagem repetida".

3. TIPOS DE CRIPTOGRAFIA ATUAIS

Conceitos Chave:

- Criptografia envolve dois processos: encriptação e decriptação
- Chaves iguais em ambos os processos => Chave Simétrica
- Chaves diferentes nos processos => Chave Assimétrica

Existem, basicamente, dois tipos de criptografia: a Criptografia de *Chave Simétrica* e a Criptografia de *Chave Assimétrica*. Esta classificação tem relação direta com os mecanismos de criptografia.

Como já deve ter ficado claro, a criptografia envolve sempre dois processos: a encriptação e a decriptação. Os algoritmos que usam a mesma chave para estes dois processos são considerados de "*Chave Simétrica*". Os algoritmos que usam chaves diferentes em cada um destes processos (uma para cada um deles) são considerados de "*Chave Assimétrica*".

Incialmente, será apresentada a criptografia de Chave Simétrica.

4. ALGORITMOS DE CHAVE SIMÉTRICA

Conceitos Chave:

- Algoritmos complexos: Combinação de substituição e transposição
- "Chave Simétrica", "Chave Secreta", "Chave Privada"
- Chave de N bits + Bloco de texto de N bits = Bloco Cifrado de N bits
    * Pode ser feito por software ou hardware
    * Hardware 1: Caixa P - transpõe bits
    * Hardware 2: Caixa S - substitui valores
- Composição de Caixa P e Caixa S
    * Exemplo: 5 estágios, 12 bits
    * Prática: 18 estágios, 64 a 256 bits
- Hardware x Software => Estágios x Rodadas
- Decriptação => Processo Invertido
    * Hardware Invertido x Função Inversa

Embora os algoritmos de criptografia atuais ainda se baseiem nas mesmas idéias de substituição e transposição, existe um esforço por criar algoritmos altamente complexos em que, sem a chave correta, mesmo de posse de muitos textos criptografados, um criptoanalista não consiga obter nenhuma informação dos mesmos.

A classe dos algoritmos que usa a mesma chave no processo de encriptação e decriptação é chamada de "algoritmos de chave simétrica". Outros nomes comuns associados a esta classe de algoritmos são "algoritmos de chave secreta" ou "algoritmos de chave privada", uma vez que a (única) chave usada deve ser privada (ou seja, guardada em segredo) para que as mensagens criptografadas continuem seguras.

O princípio básico é ter uma **chave de *n* bits** que é usada para criptografar um texto qualquer, que deve ser subdividido em **blocos de *n* bits**, similar ao que ocorria no cálculo dos hashs criptográficos.

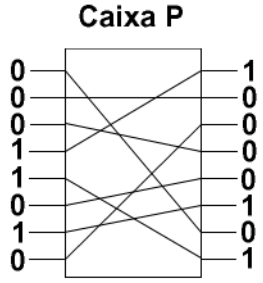
Este tipo de algoritmo pode ser facilmente implementado em hardware (para velocidade) ou em software (para flexibilidade). O mecanismo de implementação em software é análogo ao do hardware, sendo este último mais simples de compreender. Por esta razão, será apresentado primeiramente o princípio de hardware.

4.1. Caixa de Permutação (ou Transposição)

Em hardware, sabe-se que os bits são sinais elétricos que passam por fios. Então, se temos 8 bits, temos 8 fios, sendo que em cada um deles pode ou não passar corrente: o valor zero é quando não passa corrente e o valor um é quando passa corrente, por exemplo. Ou seja, um número binário como 01011000b:

Bit/Fio:	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Valor:	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**0**

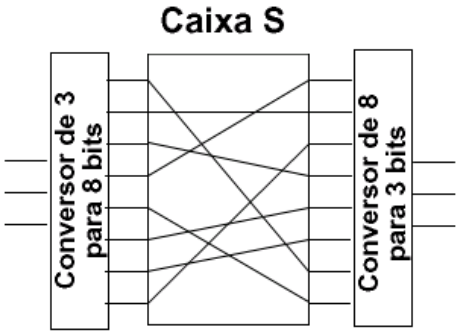
Significa que nos fios número **3**, **4** e **6** passa corrente e nos fios **0**, **1**, **2**, **5** e **7** não passa corrente. Ora, para criptografar esse número com uma transposição, por exemplo, bastaria embaralhar os fios. Então, um hardware de criptografia possível seria o representado abaixo, chamado de "Caixa P" (P de Permutação, que é o mesmo que Transposição):



É possível observar que, simplesmente *cruzando os fios*, o número 01011000b tornou-se o número 10100001b. Neste caso, a transposição do sinal é o algoritmo, e qual posição está sendo trocada com qual posição é a chave.

4.2. Caixa de Substituição

O segundo tipo de hardware para criptografia é a chamado **"Caixa S"**, de Substituição. Esta "caixa" basicamente **transforma um número em outro**, fazendo uma substituição. Normalmente usa-se um número *n* pequeno de entradas em cada caixa S, pois o número de conexões internas precisa ser 2^n, como pode ser observado na figura abaixo. É possível observar que dentro da Caixa S existe uma Caixa P.

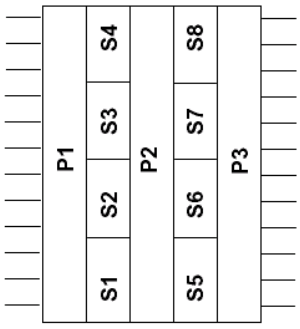


Nesta caixa, o número de 3 bits é convertido para um número de linha (3 bits viram números de 0 a 7). Esta linha será ligada (1) na entrada da Caixa P interna, que troca a posição desta linha, sendo tornando-se outra das linhas de 0 a 7. O número desta linha é então convertido em um número binário de 3 bits, novamente.

Por exemplo, o número 101b (5 em decimal), ligará a linha 5 (a sexta de baixo para cima, por começar na linha 0) da Caixa P interna. A Caixa P desvia o sinal que sai como linha 4, que é convertido em binário como 100b, que é a saída da Caixa S. Assim, o número 5 de entrada foi substituído pelo numero 4 na saída. O processo é análogo para todos os outros números. Neste caso, o processo de substituição é o algoritmo e quais conjuntos de bits são substituídos por quais é a chave.

4.3. Composição de Caixas P e S e o Software

Numa aplicação prática, são usadas várias Caixas P e S em seqüência/paralelo, tornando o texto cifrado muito mais complexo. Por exemplo, uma caixa com 12 bits:



Neste exemplos, há 12 bits de entrada/saída e 5 estágios. Na prática são usados de **64 a 256 bits de entrada/saída** e em torno de **18 estágios**. Na implementação por software é seguida a mesma estratégia de realizar permutações e substituições seqüenciais, mas ao invés de "estágios" estas etapas são chamadas de "rodadas".

A **decriptação** de textos cifrados com este tipo de dispositivo de hardware **é feita com um hardware invertido**, desfazendo as modificações do fim para o começo. No caso de software, são aplicadas de forma inversa todas as operações realizadas, também da última para a primeira.

5. ALGORITMOS COMUNS DE CHAVE SIMÉTRICA

Conceitos Chave:

- DES => Data Encryption Standard (1977)
  - Blocos de 64 bits
  - Chave de 56 bits
  - 19 estágios (transposição, 16 estágios, inversão 32x32, transposição)
- DES Tripo => Triple DES
  - Duas chaves de 56 bits (K1 e K2)
  - Criptografia DES(K1) / Descriptografia DES(K2) / Criptografia DES(K1)
- AES => Rijndael
  - Blocos de 128 bits
  - Chaves de 128 bits (ou 256 bits)
  - 10 a 14 estágios
  - 10^{10} anos para quebra por força bruta
- Modos de Cifras
  - Eliminação de padrões no texto cifrado
  - Encadeamento de blocos
  - Bloco N cifrado é aplicado em XOR (p.e.) com bloco (N+1) antes de cifrar (N+1)
- Resumo de algoritmos

Ao longo do tempo, muitos algoritmos deste tipo foram desenvolvidos. Alguns têm apenas importância histórica, outros são muito usados ainda hoje. A seguir, serão apresentados brevemente alguns destes algoritmos.

- Data Encryption Standard (DES): Desenvolvida pela IBM, foi adotado em 1977 pelo governo americano para criptografar suas informações não confidenciais. Foi amplamente utilizada em produtos de segurança, mas hoje não é útil em sua forma original. O DES é aplicado em blocos de 64 bits, com uma chave de 56 bits e é realizado em 19 estágios.

O primeiro é uma simples transposição (independente da chave), sendo que o último estágio é o inverso desta transposição. O penúltimo estágio é uma inversão dos primeiros 32 bits com os últimos 32 bits. Os outros 16 estágios intermediários são aplicações de funções parametrizadas pela chave. Os dados de 64 bits são tratados como duas partes de 32 bits: os últimos 32 bits de saída de cada estágio são sempre cópia dos primeiros 32 bits da entrada deste mesmo estágio. Os primeiros 32 bits da saída são uma operação entre a chave e os dois valores de 32 bits que compõem os 64 bits da entrada.

- DES Triplo (Triple DES): Dado que o DES passou a não ser considerado seguro (apesar de, supostamente, sua versão de 128 bits o ser), passou-se a utilizar sua variante, o DES Triplo. O princípio é ter uma chave de 112 bits, que na verdade são duas chaves de 56 bits juntas: K1 e K2. O funcionamento é feito da seguinte forma:

- Aplica-se criptografia DES com chave K1
- Aplica-se decriptografia DES com chave K2
- Aplica-se criptografia DES com chave K1 novamente

Realiza-se o processo 3 vezes, com apenas duas chaves, pois 112 bits de chave foram considerados suficientes para a segurança. O processo é realizado desta forma (EDE ao invés de EEE) para facilitar o uso de implementações DES já existentes.

- Advanced Encryption Standard (AES): Foi desenvolvido por um concurso apoiado pela NIST (National Institute of Standards and Technology). Neste concurso foram apresentados vários algoritmos, mas o vencedor (que ficou chamado de AES) foi o Rijndael, que aceita chaves e tamanhos de blocos de 128 a 256 bits, em incrementos de 32 bits. Entretanto, o AES é padronizado para blocos com 128 bits e chaves de 128 bits ou 256 bits (existe a variante de chave de 192 bits, mas praticamente não é usada). O padrão comercial deve ser o de 128/128 bits, que levaria, numa hipótese considerando os melhores hardwares atuais, cerca de 10^{10} anos para ser decifrado por força bruta.

> Segundo Tanenbaum (2003), quando um texto com essa criptografia for decifrado, já se passou tanto tempo que o Sol já explodiu e o resultado terá que ser lido à luz de velas.

Assim como o DES, o AES funciona em estágios de transposição e substituição. O número de estágios depende do tamanho de chave e bloco, mas varia de 10 a 14. A forma como foi planejado (as operações utilizadas) tornam esse algoritmo bastante rápido.

5.1. Modos de Cifras

Embora DES, Triple DES e AES sejam relativamente complexos, eles são basicamente algoritmos de transposição e substituição. Isso quer dizer que trechos de texto iguais serão sempre criptografados da mesma forma. Isso pode ser considerado uma falha se o criptoanalista tiver a possibilidade de obter muitos textos criptografados.

Alguns algoritmos mais complexos usam **encadeamentos de blocos de cifra**. Isso significa que ao codificar o bloco 10 de uma mensagem, por exemplo, antes de codificá-lo será usado o bloco 9, já cifrado, para modificá-lo (por exemplo, usando a operação XOR). O resultado desta combinação é que será cifrado como o bloco 10.

Como é possível observar, isso faz com que dificilmente blocos iguais sejam codificados da mesma forma. Para que isso ocorresse, todo o texto anterior teria que ser exatamente igual nas duas mensagens mas, neste caso, é esperado que os cifrados sejam iguais: é a mesma mensagem! A desvantagem deste sistema é que a tarefa de decifrar um texto não pode ser quebrada em processos menores, já que o processamento de um bloco depende do processamento de todos os anteriores.

Existem ainda outras formas mais complexas que não possuem essa restrição como os chamados "**modo de feedback de cifra**", "**modo de cifra de fluxo**", "**modo de contador**", dentre outros, que estão além do escopo deste curso. O importante - o fato que deve ficar registrado - é que sempre se *busca eliminar a ocorrência de padrões nos textos*, para dificultar ao máximo a criptoanálise.

5.2. Resumo dos Algoritmos Comuns de Chave Simétrica

Cifra	Comprimento de Chave	Comentário
AES (Rijndael)	128 a 256 bits	Melhor escolha
Blowfish	1 a 448 bits	Antigo e lento
DES	56 bits	Muito fraco
DES Triplo	112 bits (ou 168!)	Segunda melhor escolha
IDEA	128 bits	Bom, mas patenteado
RC4	1 a 2048 bits	Algumas chaves são fracas
RC5	128 a 256 bits	Bom, mas patenteado
Serpent	128 a 256 bits	Muito forte
Twofish	128 a 256 bits	Muito forte, bastante usado.

6. ATIVIDADE

Em um projeto de sistema foi adotado um algoritmo que usa várias permutações para codificar um determinado dado. Em uma das etapas é necessário realizar uma permutação de bits e sua equipe foi designada para desenhar a "Caixa P" de hardware que realiza tal permutação. Projete-a de forma que ela converta as seguintes informações:

De	Para	De	Para
00011100	00101100	11100000	10010010
00110000	00100010	01000001	10000001
00000110	01001000	10000001	00010001

7. BIBLIOGRAFIA

TANENBAUM, A. S. **Redes de Computadores**. Editora Campus, 2003.
Páginas 767 a 798 (Seção 8 a 8.2.5)

TERADA, R. **Segurança de Dados:** Criptografia em Redes de Computador. São Paulo: Edgard Bücher, 2000.
Páginas 41 a 94 (Seção 3 a 3.11)

Unidade 4: Criptografia de Chave Pública e Assinaturas Digitais

Prof. Daniel Caetano

Objetivo: Apresentar conceitos de criptografia de chave assimétrica (pública), os principais algoritmos e conceito das assinaturas digitais.

Bibliografia: TERADA, 2000, TANENBAUM, 2003.

INTRODUÇÃO

Conceitos Chave:

- Problema: Enviar dados criptografados;
  - * Como enviar a chave?
- Diffie & Hellman
  - * Chave única x Chave Dupla
  - * Duas chaves diferentes => chaves assimétricas

Quando se desenvolve um sistema que envolve segurança por criptografia, o problema que mais freqüentemente se discute é a questão da distribuição das chaves de forma segura: como enviar a chave da criptografia para o receptor de forma que ninguém mais tenha acesso a ela?

Desde o princípio da história da criptografia, a distribuição de chaves sempre foi um problema. Independente da complexidade do algoritmo usado, se alguma pessoa não autorizada conseguisse colocar as mãos na chave, a segurança estaria perdida.

A solução aparentemente óbvia para o problema seria guardar a chave com extrema segurança, em um cofre, longe do acesso de todos... mas infelizmente essa solução não era possível: como a chave para codificar e decodificar uma mensagem era a mesma, essa chave precisava ser distribuída. Ou seja: ela tinha de ser distribuída mas não podia ser roubada, um problema aparentemente meio complicado de ser resolvido.

Mas, em 1976, dois pesquisadores (Diffie e Hellman) deram uma solução inesperada para este problema: criaram um algoritmo de criptografia em que eram usadas duas chaves: uma para a encriptação e outra, diferente, para a decriptação.

Por usar duas chaves distintas para codificar e decodificar, este tipo de algoritmo ficou conhecido como algoritmo de chave assimétrica.

1. MECANISMO DOS ALGORITMOS DE CHAVE PÚBLICA

Conceitos Chave:

- Duas chaves:
  - * Para criptografar
  - * Para decriptar
- Criptografar: chave de criptografia do destinatário
  - * Há restrições? NÃO => Chave Pública
- Decriptar: chave de decriptografia do destinatário
  - * Há restrições? SIM => Chave privada
- Ciclo de Encriptação/Decriptação
- Relação entre as chaves: algoritmo de chave pública

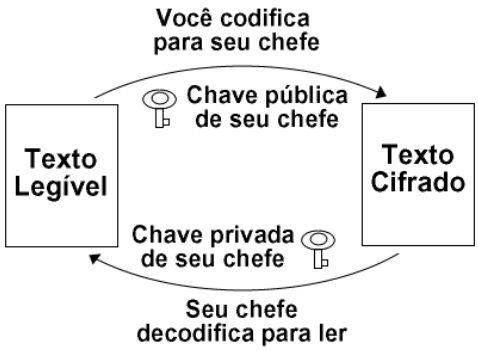
A idéia por trás de existir duas chaves é simples, mas não muito óbvia. Para compreendê-la, é mais fácil pensar em um determinado cenário como o proposto a seguir.

Imagine que você queira enviar uma mensagem para o seu chefe, sobre uma nova idéia incrível que você teve para aumentar os lucros da empresa, mas se algum funcionário interceptar esta mensagem pode, eventualmente, roubar sua idéia e ganhar a tão sonhada promoção.

A alternativa então é encriptar a mensagem para o seu chefe, de forma que só ele possa ler a tal mensagem. Entretanto, para isso será necessário obter a chave de encriptação do seu chefe, para poder gerar uma mensagem codificada que só ele possa ler. Como o seu chefe certamente não tem qualquer restrição a que alguém escreva uma mensagem codificada para ele, a chave de encriptação dele está disponível na página dele na intranet. Qualquer pessoa pode pegar esta chave e usá-la para mandar mensagens que só ele, o chefe, poderá ler. Por esta razão, a chave de encriptação que todos podem acessar é chamada de "chave pública".

Segundo o funcionamento dos algoritmos de chave assimétrica, quando esta chave pública for usada para codificar um texto, apenas a outra chave pode decodificá-lo. Por esta razão, como seu chefe não quer que outras pessoas fiquem lendo as mensagens dele, ele guarda muito bem, de forma escondida, esta outra chave, que é chamada de "chave privada" (já que só o dono dela tem acesso).

Este funcionamento é representado na figura a seguir.



Devido ao fato de a "chave pública" resolver o problema de distribuição de uma chave para que as pessoas possam encriptar uma mensagem para um determinado indivíduo, estes algoritmos ficaram mais conhecidos como "algoritmos de chave pública".

2. FUNCIONAMENTO SIMPLIFICADO DE UM ALGORITMO

Conceitos Chave:

- Exemplos:
  - * Chave Simétrica com Soma
  - * Chave Pública com Soma
- Anel de Números => Conjunto em Anel
- Ponto fraco da chave pública com soma
  - * Sempre é possível obter a chave privada da pública
  - * Dificultar!

Anteriormente vimos que, nos algoritmos de chave simétrica, a operação de decriptação era uma operação inversa da encriptação. Por exemplo, se codificarmos os números "10 20 35 15 17 28" com a chave "7 10 12 3 22 10" através de uma soma, teríamos:

Texto Legível:	**10**	**20**	**35**	**15**	**17**	**28**
Chave:	**7**	**10**	**12**	**3**	**22**	**10**
Texto Cifrado(soma):	**17**	**30**	**47**	**18**	**39**	**38**

Para a decriptação, usaríamos a mesma chave e o processo inverso da soma (subtração):

Texto Cifrado:	**17**	**30**	**47**	**18**	**39**	**38**
Chave:	**7**	**10**	**12**	**3**	**22**	**10**
Texto Legível (subtração):	**10**	**20**	**35**	**15**	**17**	**28**

O "truque" dos algoritmos de chave pública é não usar operações inversas, mas sim a mesma operação novamente, mas com uma chave diferente. Por exemplo, imagine que ao invés da soma, usamos a soma mais resto de divisão por 40, para o mesmo problema anterior: codificar os números "10 20 35 15 17 28" com a chave "7 10 12 3 22 10", "40":

Texto Legível:	**10**	**20**	**35**	**15**	**17**	**28**
Chave (Pública):	**7**	**10**	**12**	**3**	**22**	**10**
Soma:	**17**	**30**	**47**	**18**	**39**	**38**
Texto Cifrado(MOD 40)	**17**	**30**	**7**	**18**	**39**	**38**

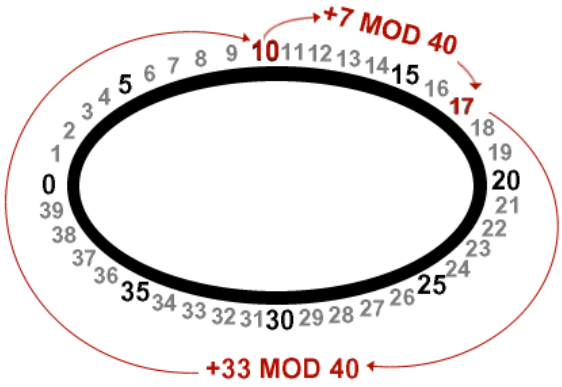
Para decodificar usando a mesma operação (soma com resto de divisão) precisamos agora de uma outra chave, que seria "33 30 28 37 18 30", "40". Vejamos abaixo:

Texto Cifrado:	**17**	**30**	**7**	**18**	**39**	**38**
Chave (Privada):	**33**	**30**	**28**	**37**	**18**	**30**
Soma:	**50**	**60**	**35**	**55**	**57**	**68**
Texto Legível (MOD40)	**10**	**20**	**35**	**15**	**17**	**28**

É possível verificar a diferença entre as chaves:

Chave Pública:	**7**	**10**	**12**	**3**	**22**	**10**
Chave Privada:	**33**	**30**	**28**	**37**	**18**	**30**

Este algoritmo é bastante simples e funciona, neste caso, por causa da aplicação do resto de divisão por 40, que torna o conjunto de números um anel de números. Observe a figura abaixo para o caso do primeiro número da mensagem (10):



Infelizmente, entretanto, o algoritmo apresentado é extremamente fraco, pois a chave privada é facilmente encontrada, sabendo-se a chave pública: a soma da chave pública com a chave privada dá sempre o número do divisor do resto da divisão:

Chave Pública:	**7**	**10**	**12**	**3**	**22**	**10**
Chave Privada:	**33**	**30**	**28**	**37**	**18**	**30**
Soma:	**40**	**40**	**40**	**40**	**40**	**40**

Os algoritmos mais seguros de chave pública fazem todo o esforço possível para dificultar a obtenção da chave privada através da chave pública, sendo este o grande ponto fraco dos algoritmos de chave pública: é sempre possível obter a chave privada através da pública. Entretanto, há meios de tornar o cálculo da chave privada tão complicado que os computadores atuais levariam milhares de anos para obtê-la a partir da chave pública, garantindo a segurança destes algoritmos mais complexos.

3. REQUISITOS PARA BONS ALGORITMOS DE CHAVE PÚBLICA

Conceitos Chave:

- Regras de Bons Algoritmos
  - a) D(E(T)) = T
  - b) Difícil deduzir D a partir de E
  - c) Forte contra ataque de texto legível escolhido
- Como atender o requisito b?
  - * **$J = I^K$ MOD P** => K=?
  - * Se P for um primo grande, demorado calcular K!

Dado um algoritmo com uma chave pública E e uma chave privada D, para ser considerado um bom algoritmo ele tem que atender a basicamente três requisitos:

1) D (E(T)) = T
2) Deve ser muito difícil deduzir D a partir de E
3) E não pode ser decifrado por ataque de texto legível escolhido.

Ou seja:

1) se aplicarmos a chave pública D a um texto legível P encriptado com a chave privada E, o resultado será o texto legível P.
2) Como todos possuem a chave pública E, se fosse fácil identificar a chave privada D a partir da pública E, a segurança não existiria.
3) Mesmo que um criptoanalista tente até o fim de seus dias, ele não deve ser capaz de obter E a partir da análise de textos legíveis contra seus textos criptografados correspondentes.

A segurança da maioria dos algoritmos de chave pública (para atender ao requisito 2) se baseia no problema de calcular o valor de K no problema a seguir:

Dados um primo P e inteiros I e J, de forma que: $I > 0$ e $J < P$, calcular um inteiro K tal que:

$$J = I^K \text{ MOD } P$$

Por exemplo: se P = 17, I = 7 e J = 10, calcular K que torne verdadeira a equação a seguir:

$$\mathbf{10 = 7^K \ MOD \ 17}$$

Neste caso, a solução é K = 9. Entretanto, quando P for um primo relativamente longo, até hoje ninguém conseguiu elaborar um algoritmo computacionalmente eficiente para resolver tal problema.

4. O ALGORITMO RSA

Conceitos Chave:

- Existem vários algoritmos: Diffie-Hellman, RSA, Rabin, ElGamal, MH etc.
  - * Quase todos são razoáveis
- RSA = Rivest, Shamir e Adleman
  - * Já resiste a ataques há 30 anos.
  - * Ponto negativo: chaves de pelo menos 1024 bits
    - => Lento
- Mecanismo
  - a) Escolher dois primos extensos P e Q (1024 bits ou mais)
  - b) Calcule N = P*Q
  - c) Calcule Z = (P-1)*(Q-1)
  - d) Escolha um número D primo com relação a Z
  - e) Encontre E de forma que (E*D) MOD Z = 1.
  - f) Divida a msg em T blocos; cada mensagem T deve ser menor que N.
  - g) Criptografe: **$C = T^E$ (MOD N)**
  - h) Decifre: **$T = C^D$ (MOD N)**
- Chave Pública: (E,N)
- Chave Privada: (D,N)
- Problema: se N fatorável?
  - * Calcula-se P e Q
  - * Com P e Q calcula-se Z
  - * Com E e Z, calcula-se D => quebrado!
- Fatoração de número de 500 dígitos: 10^{25} anos.
- Exemplo com números pequenos
- Mecanismo de Troca de Chave Pública
  - * (no texto)

Assim como no caso dos algoritmos de chave simétrica, muitos algoritmos de chave pública existem: Diffie-Hellman, RSA, Rabin, ElGamal, Curvas Elipticas, MH, etc. Quase todos eles são bons o suficiente para a maioria das aplicações (com exceção do Diffie-Hellman original, que precisou de uma modificação para se tornar seguro).

A maioria destes algoritmos é bastante complexa e, por terem características de uso similares, apenas um deles será apresentado: o RSA (uma sigla composta pelo nome de seus altores: Rivest, Shamir e Adleman).

O RSA já resistiu a todas as tentativas de rompimento nos últimos quase 30 anos e, por esta razão, é o principal algoritmo usado em segurança. O ponto negativo deste (assim como de praticamente todos os algoritmos de chave pública) é que este necessita de chaves de pelo menos 1024 bits para se obter segurança (contra cerca de 128 bits dos algoritmos de chave simétrica), o que os torna bastante lentos.

De qualquer forma, o mecanismo de funcionamento do RSA é o seguinte:

1) Escolha dois números primos extensos (1024 bits ou mais) P e Q.
2) Calcule N = P * Q e Z = (P-1) * (Q-1).
3) Escolha um número D tal que Z e D sejam primos entre si.
4) Encontre E de forma que (E * D) MOD Z = 1.

Com esses dados calculados previamente, agora divida a mensagem a criptografar T em blocos menores, de forma que cada mensagem de texto simples seja um número menor que N.

Para criptografar a mensagem T, use:

$$\mathbf{C = T^E \ (MOD\ N)}$$

Para decriptar, use:

$$\mathbf{T = C^D \ (MOD\ N)}$$

Uma vez que para a encriptação é necessário possuir E e N, a chave de encriptação (ou seja, a chave pública) é (E,N). Pela mesma razão, a chave de decriptação (chave privada) é (D,N). Se N fosse facilmente fatorável, seria possível obter P e Q e, com eles, o cálculo de Z e, sabendo Z e E, seria possível calcular D. Felizmente, os matemáticos tentam achar uma forma simples de fatorar números primos grandes há mais de 300 anos e ainda não conseguiram solucionar o problema, indicando que este é um problema bastante difícil.

Estima-se que a fatoração de um número de 500 dígitos, com o melhor algoritmo conhecido e máquina existente, leve cerca de 10^{25} anos, o que torna este algoritmo bastante seguro para chaves deste tamanho. No futuro, basta ampliar a chave para se acomodar a novas realidades de algoritmos e equipamentos.

4.1. Exemplo de RSA com primos pequenos

Consideremos P=3 e Q=11. Com isso podemos gerar N = 3*11 = 33 e Z = (3-1)*(11-1) = 2*10 = 20. Um bom valor para D seria 7, que não tem qualquer fator comum com 20. Para encontrar E, resolvemos a eqüação: (D*E) MOD Z = 1, ou seja, (7*E) MOD 20 = 1. E = 3 satisfaz ao problema, que 7*3 = 21 e 21 MOD 20 = 1. Assim, resumidamente:

P=3
Q=11
N=33
Z=20
D=7
E=3

A palavra SUZANNE será codificada (considerando A=1, B=2, ..., Z=26), com estes dados.

Texto	T	T^3	$C=T^3$ MOD 33	C^7	C^7 MOD 33	Texto
S	19	6859	28	13492928512	19	S
U	21	9261	21	1801088541	21	U
Z	26	17576	20	1280000000	26	Z
A	01	1	1	1	01	A
N	14	2744	5	78125	14	N
N	14	2744	5	78125	14	N
E	05	125	26	8031810176	05	E

4.2. Uso do RSA

É sabido que o RSA é muito lento para criptografar mensagens inteiras. Para que ele é usado então? Bem, qual era o problema original dos algoritmos de chave simétrica? Distribuir a chave? Bem, é exatamente para isso que o RSA é usado: distribuir chaves de algoritmos de chave simétrica. O procedimento é:

SE: Chave Pública do Server
SC: Chave Privada do Server
CE: Chave Pública do Cliente
CC: Chave Privada do Cliente
CS: Chave Secreta

1) Cliente pega chave pública do Server (SE).
2) Cliente criptografa sua chave pública (CE) com a chave pública do server (SE)
3) Server decifra chave pública do cliente (CE) usando sua chave privada (SD)
4) Server gera uma chave secreta (simétrica, CS).
5) Server criptografa a chave secreta (CS) com a chave pública do cliente (CE)
6) Cliente decifra a chave secreta (CS) com sua chave secreta (CD).

A partir de então, ambos passam a usar um algoritmo de chave secreta, com a chave secreta trocada entre eles.

5. ASSINATURAS DIGITAIS

Conceitos Chave:

- Requisitos da Assinatura Digital
  - * Receptor deve poder verificar a identidade alegada do transmissor
  - * O transmissor não pode repudiar o conteúdo da mensagem
  - * O receptor não tenha como forjar a mensagem
- Assinatura com Criptografia de Chave Simétrica
  - * Intermediário
  - * A para B, por C
  - * **A => K_A(B,N,T,P) => C => K_B((A,N,T,P),K_C(A,T,P)) => B**
  - * Problema: confiar no intermediário
- Assinatura com Criptografia de Chave Assimétrica
  - * Criptografia: **D (E(P)) = P**
  - * E se algoritmo atender também a**: E (D(P)) = P**
  - * Assim, codifica-se uma assinatura com a chave privada
    - = Qualquer um com a chave pública consegue verificar!
  - * Não é perfeita
    - = Roubo da chave privada
    - = Troca da chave privada
- Garantir não-repúdio de conteúdo?
  - * Assinar (criptografar) mensagem inteira?
  - * Hash + Criptografia (assinatura)
  - * Passos e Exemplo
    - = (no texto)

Para que documentos possam ser trocados pela rede substituindo os documentos físicos, é necessário um sistema que respeite os seguintes requisitos:

1) O receptor possa verificar a identidade alegada do transmissor
2) O transmissor não possa repudiar o conteúdo da mensagem
3) O receptor não tenha como forjar ele mesmo a mensagem

Há duas formas de atender a estes requisitos, o primeiro deles usando um sistema de criptografia de chave simétrica e outro usando criptografia de chave pública.

5.1. Assinatura com Criptografia de Chave Simétrica

Neste caso, além das duas pessoas trocando mensagens, é preciso ter um terceiro elementos, responsável pela garantia da identidade. O mecanismo é o seguinte:

A deseja mandar uma mensagem para B. A criptografa a mensagem com a chave K_A, simétrica, lembrando de acrescentar a identidade do receptor (B), um número aleatório (N) e a data/hora em que a mensagem foi enviada (T). Sendo a mensagem (P), isso pode ser representado como: $K_A(B,N,T,P)$.

Esta mensagem $K_A(B,N,T,P)$ é então enviada para a entidade intermediária (C), que reconhece que a mensagem veio de A, descriptografando-a com a chave de A, K_A. Em seguida, esta entidade (C) criptografa esta mensagem com a chave de B, K_B, lembrando de modificar o indicador do receptor pelo indicador do autor, o que pode ser indicado por: $K_B(A,N,T,P)$. Adicionalmente, C também envia, junto com a mensagem, um trecho usando sua própria criptografia: $K_C(A,T,P)$, para que o receptor tenha certeza de que a mensagem veio mesmo da entidade C. Assim, B poderá descriptografar a mensagem toda com sua chave K_B e ainda verificar que ela, de fato, veio da entidade confiável C.

Graficamente:

$$\mathbf{A => K_A(B,N,T,P) => C => K_B((A,N,T,P),K_C(A,T,P)) => B}$$

O grande problema desta solução é exigir uma entidade intermediária para a confirmação das identidades, o que nem sempre é desejável.

5.2. Assinatura com Criptografia de Chave Pública

Quando foi falado sobre criptografia de chave pública, foi colocada um dos requisitos básicos que era:

$$\mathbf{D (E(P)) = P}$$

Ou seja, se aplicarmos a chave pública D a um texto legível P encriptado com a chave privada E, o resultado será o texto legível P.

Considere-se agora que este algoritmo também atenda ao seguinte requisito:

$$\mathbf{E (D(P)) = P}$$

Ou seja, se uma mensagem for codificada com a chave privada, ela será decodificada pela chave pública. Neste caso, uma mensagem codificada por uma pessoa com sua chave privada poderia ser decodificada por qualquer pessoa que tenha a chave pública de quem criptografou a mensagem. Embora isso não seja muito útil do ponto de vista de confidencialidade, é muito útil no que se refere à autenticidade, já que se a chave pública de

uma pessoa decifra uma mensagem qualquer, essa mensagem só pode ser desta pessoa dona da chave pública.

O RSA, apresentado anteriormente, possui essa propriedade e é, com efeito, muito usado para assinatura digital de documentos. Recentemente foi proposto pelo NIST o DSS, Digital Signature Standard, baseado no algoritmo ElGamal. Entretanto, o DSS foi considerado pela comunidade como muito fraco (chave fixa de 512 bits), muito lento e com o agravante do algoritmo ElGamal não ter sido testado o suficiente pela comunidade.

Entretanto, a assinatura digital por chave pública não é perfeita por algumas razões: se a chave privada de uma pessoa for roubada, não haverá mais como identificar a autenticidade de mensagens com essa chave. Além disso, se uma pessoa mudar seu par de chaves depois de algum tempo, suas mensagens antigas não poderão mais ter sua autenticidade verificada.

Além disso, um dos problemas mais sérios é que ainda é lento encriptar toda uma mensagem apenas para que sua autenticidade seja verificada. Um outro fato importante é que muitas vezes é desejável possibilitar a confirmação da autenticidade sem ter que esconder a mensagem como um todo. É aqui que os números resumo (hashs criptográficos) apresentam seu grande valor, resolvendo estes dois problemas. O mecanismo é o seguinte:

1) Usuário A cria mensagem M.
2) Usuário A calcula o hash da mensagem M, H_A.
3) Usuário A criptografa o hash H_A com sua chave privada AD, gerando H_AAD.
4) Usuário A anexa HAAD (hash criptografado) à mensagem M e envia para B.
5) Usuário B lê mensagem M.
6) Usuário B gera hash H_B da mensagem M.
7) Usuário B decifra H_AAD usando a chave pública de A (AE), gerando H_A.
8) Se H_A for igual a H_B, a autenticidade e integridade está comprovada.

Por exemplo:

Ações de A
M = "Olá B!"
H_A = 19
AD = (7,20) => RSA
H_AAD = 28
Mensagem Enviada: M, 28

Ações de B
Mensagem Recebida: M, 28
M = "Olá B!"
H_B = 19
H_AAD = 28
AE = (3,20) => RSA
H_A = 19

H_B = H_A => Mensagem íntegra, autor confirmado.

Como o hash é único para a mensagem, pode-se verificar a integridade da mensagem através dele. Como ele foi criptografado com a chave privada do remetente, o destinatário pode checar sua autenticidade com a chave pública do remetente. Como o hash é um número de tamanho reduzido (perto da mensagem original), elimina-se o problema da lentidão de algoritmos com o RSA.

6. ATIVIDADE

Em um sistema sendo desenvolvido, deseja-se que as mensagens possam ser enviadas de maneira segura, isto é, criptografada. Além disso, foi solicitado que seja possível enviar também uma mensagem apenas assinada, mas que ainda assim seja possível identificar adulterações no texto. Foi determinado que o algoritmo básico seja o indicado abaixo:

(M + E) MOD N

Onde M é um trecho da mensagem, (E,N) é a chave pública. Para decifrar, usa-se a mesma regra, mas com a chave privada (D,N). O Hash que deve ser usado é dado pela fórmula:

(A+B+C+...+Z) MOD 26

Onde A, B, C...Z são os caracteres da mensagem. A mensagem teste é:

OLA BETO
ASS CARLOS

A chave pública teste, é E=(19,26); A chave privada teste é D=(7,26). Considere A=0, B=1, C=2...

a) Gere a mensagem criptografada;
b) Gere o hash da mensagem;
c) Gere a assinatura a partir do hash;
d) Critique o algoritmo.

7. BIBLIOGRAFIA

TANENBAUM, A. S. **Redes de Computadores**. Editora Campus, 2003.
Páginas 795 a 811 (Seção 8.3 a 8.4.3)

TERADA, R. **Segurança de Dados:** Criptografia em Redes de Computador, 2000.
Páginas 95 a 150 (Seção 4 a 4.11)

Unidade 5: Manipulação Segura da Informação

Prof. Daniel Caetano

Objetivo: Apresentar e discutir problemas usuais onde técnicas de segurança são necessárias e como solucioná-los.

Bibliografia: TERADA, 2000, TANENBAUM, 2003.

INTRODUÇÃO

Conceitos Chave:

- Temas já tratados
    * Política
    * Conceitos de Segurança Lógica e Física
    * Ferramentas (Criptografia, Hashs, Assinaturas Digitais)
- Configuração de Software?
- Manipulação de Banco de Dados?
- Desenvolvimento Web?

Muitas informações já foram apresentadas sobre políticas de segurança e os meios de manutenção da segurança lógica da informação, incluindo ferramentas como criptografia.

Entretanto, muitas vezes é preciso lidar com a segurança em níveis mais altos, como os de configuração de software ou na manipulação de bancos de dados, visando impedir que usuários inescrupulosos tenham a possibilidade de realizar um ataque.

Esta unidade apresenta uma breve visão sobre algumas destas questões.

1. CONTROLE DE ACESSO EM BANCO DE DADOS

Conceitos Chave:

- Controle importante: leitura e escrita no BD
- SGBDs usuais: controle de acesso integrado
  * Uso de banco de dados específico
  * Consultado sempre que acesso ao BD é feito
- Instalação inicial
  * Apenas root: senha?
- Primeiros passo:
  * Colocar senha complexa para o root
  * Configurá-lo apenas para acesos local (ou SSH)
  * Criar usuários restritos para as funções normais
- Controle de Acesso SQL-97
  * **GRANT tipo_de_privilégio [lista_de_colunas] [, tipo_de_privilégio [lista_de_colunas] ...]**
    ON { nome_da_tabela | * | *.* | nome_do_banco.* }
    TO nome_do_usuário [IDENTIFIED BY 'password']
    [, nome_do_usuário [IDENTIFIED BY 'password'] ...]
    [WITH GRANT OPTION]
  * **REVOKE tipo_de_privilégio [lista_de_colunas] [, tipo_privilégio [lista_de_colunas] ...]**
    ON { nome_da_tabela | * | *.* | nome_do_banco.* }
    FROM nome_do_usuário [, nome_do_usuário ...]

- usuário@nome.do.host x usuário@numero_do_host
- usuario@"%.usp.br" x usuário@"200.168.%"

Um aspecto muito importante da segurança da informação é relativo ao acesso das informações contidas em um banco de dados, seja este acesso para a leitura ou, mais importante, para a escrita.

Praticamente todo Sistema Gerenciador de Banco de Dados (SGBD) contém um sistema de controle de acesso integrado. As permissões de acesso são registradas em um banco de dados específico, o qual é checado sempre que uma requisição de mudança ou seleção em qualquer dos bancos de dados gerenciados é solicitada.

Na instalação padrão, muitos dos SGBDs são vulneráveis, existindo apenas uma conta de usuário (root), com todas as permissões possíveis e sem qualquer tipo e senha. A primeira providência a ser tomada é a de configurar este usuário de uma forma muito mais restritiva, colocando uma senha longa e também permitindo que o usuário "root" só possa gerenciar bancos de dados a partir da máquina local, ou seja, para que alguém possa realizar operações como usuário root, precisa estar fisicamente na máquina do servidor de banco de dados (ou conectado a ele por Telnet/SSH).

Feito isso, o próximo passo é criar "contas" específicas para cada uso e usuário do SGBD, estabelecendo restrições rígidas a partir do princípio que "tudo é proibido a menos que expressamente permitido".

1.1. Acesso em Banco de Dados SQL

Nos bancos de dados que seguem o padrão SQL-97, o controle de acessos é configurado com os comandos GRANT e REVOKE. O comando GRANT concede permissão de acesso a um usuário e o comando REVOKE revoga a permissão de acesso a um usuário.

A sintaxe do comando GRANT é:

```
GRANT tipo_de_privilégio [lista_de_colunas] [, tipo_de_privilégio [lista_de_colunas] ...]
    ON { nome_da_tabela | * | *.* | nome_do_banco.* }
    TO nome_do_usuário [IDENTIFIED BY 'password']
        [, nome_do_usuário [IDENTIFIED BY 'password'] ...]
    [WITH GRANT OPTION]
```

Ou seja:

1) Indica-se o comando GRANT.
2) Indica-se o tipo de privilégio e para quais colunas este privilégio será aplicado. Os tipos de privilégio são:

ALL PRIVILEGES	FILE	RELOAD
ALTER	INDEX	SELECT
CREATE	INSERT	SHUTDOWN
DELETE	PROCESS	UPDATE
DROP	REFERENCES	USAGE

3) Indica-se em que tabela estão aquelas colunas, com a diretiva ON.
4) Indica-se o usuário autorizado, com a diretiva TO. Não é obrigatório, mas é interessante que cada usuário seja identificado por um password, com a diretiva IDENTIFIED BY.
5) Finalmente, é indicado se aquele usuário terá permissão de autorizar outros usuários as ações às quais ele é permitido, com a diretiva WITH GRANT OPTION.

A sintaxe do comando REVOKE é mais simples:

```
REVOKE tipo_de_privilégio [lista_de_colunas] [, tipo_privilégio [lista_de_colunas] ...]
    ON { nome_da_tabela | * | *.* | nome_do_banco.* }
    FROM nome_do_usuário [, nome_do_usuário ...]
```

1) Indica-se o comando REVOKE.
2) Indica-se o tipo de privilégio que será removido e de quais colunas.
3) Indica-se em que tabela estão aquelas colunas, com a diretiva ON.
4) Indica-se o usuário de quem estão sendo revogadas as permissões, com a diretiva FROM.

Alguns SGBDs permitem que um usuário fique restrito a um único host (equipamento de acesso). Isto é alcançado com algumas modificações no nome do usuário, usando a seguinte sintaxe:

nome_do_usuário@nome_do_host

ou

nome_do_usuário@número_do_host

Se o nome do usuário ou o nome do host contiverem caracteres especiais, como por exemplo o "-" ou o "%", eles devem estar entre aspas. O caractere "%" tem o significado de "qualquer elemento", como por exemplo:

"144.155.166.%"

Se refere a qualquer host de 144.155.166.0 a 144.155.166.255. Da mesma forma:

"%.usp.br"

Se refere a qualquer host terminado por .usp.br, como por exemplo "ftp.usp.br", "www.usp.br", "sistemas.poli.usp.br" etc.

2. ACESSO REMOTO

Conceitos Chave:

- Acesso à máquina B a partir de A, através de rede
  - * A => terminal de B
- Acesso Remoto
  - * Terminais Texto (IBM 3270, Telnet, SSH => TTY, ANSI...)
  - * Terminais Gráficos (X-Server, VNC, MS Remote Access... => ?)
- Vantagem: uso remoto como se estivesse no equipamento físico
- Desvantagem: precisar de acesso físico pode ser requisito de segurança
  - * Conflito => Brecha
- Acesso Remoto com Segurança
  - * Intrínsecos: SSH e IBM DToC
  - * Extrínsecos: PGP
  - * Limitado: VNC (senha + ip)

Acesso remoto é o acesso a uma máquina B a partir de outra máquina A, pela rede, de forma que o usuário pode controlar a máquina B como se estivesse fisicamente nela. Neste

caso, diz-se que a máquina A está atuando como terminal da máquina B. Existem dois tipos comuns de Acesso Remoto:

- Terminal texto: IBM 3270, Telnet, Secure Shell (SSH)... que usam emulação de terminal de diversos tipos, como TTY, ANSI, VT-100, etc.
- Terrminal gráfico: X-Server, Virtual Network Computing (VNC), Microsoft Remote Access (usado no MSN), IBM Desktop On Call... cada um deles usando seu próprio padrão de comunicação.

A vantagem do uso destes sistemas é clara: o usuário pode operar um sistema remoto como se estivesse com acesso físico, realizando tarefas que só poderiam ser realizadas fisicamente naquele terminal.

Considerando que é comum que algumas tarefas obriguem o uso de um determinado equipamento, como uma forma de só permitir que estas tarefas sejam executadas se o usuário tiver autorização de chegar fisicamente até aquele computador (como no caso da criação e autorização global de usuários em um SGBD), os sistemas de acesso remoto podem também se tornar uma grande brecha de segurança.

Como assim? É simples: se um computador está disponível para acesso remoto, alguém pode controlá-lo de qualquer lugar no mundo como se estivesse fisicamente naquele equipamento.

A melhor solução seria, então, não disponibilizar o acesso remoto em máquinas que possuem informações importantes ou que são usadas para realizar tarefas delicadas. Entretanto, nem sempre é possível abrir mão de um sistema de acesso remoto. Para estes casos, lança-se mão de um sistema de acesso remoto com segurança.

Alguns dos sistemas citados possuem segurança intrínseca, ou seja, no próprio sistema de terminal. São exemplos o SSH e o IBM Desktop On Call. Nestes casos, nenhuma medida adicional (além da instalação e configuração do sistema) são necessários para a comunicação segura. Estes sistemas, em especial o SSH, funcionam com um esquema de criptografia de chave pública bastante seguro.

Nos casos de comunicação com sistemas de acesso remoto que não são intrinsecamente seguros (Telnet, VNC, etc) é importante trabalhar com uma conexão segura no nível do TCP/IP, o que pode ser conseguido com a instalação do driver do Pretty Good Privacy (PGP), por exemplo.

Sistemas como o VNC, em algumas plataformas, fornecem uma segurança mínima que, embora não funcione com dados criptografados, dificulta o acesso indesejado às máquinas sensíveis. Esta segurança é baseada num sistema de senha juntamente com um sistema de identificação de IP, onde é possível definir os IPs que serão permitidos acessar àquela máquina. Este sistema dificulta a vida de um invasor que, além de descobrir a senha, precisará estar em um equipamento habilitado para poder por a invasão em prática.

De qualquer maneira, a melhor alternativa para não ter estes tipos de problema de segurança é não permitir acesso remoto a uma máquina sensível.

3. PROGRAMAÇÃO WEB SEGURA

Conceitos Chave:

- Dados Transmitidos x Dados no Servidor
- Criptografia dos Dados Transmitidos
    * Nem sempre necessária
    * Secure Socket Layer (SSL)
    * Transport Layer Socket (TLS) => Novo nome do SSL
    * Implantados diretamente em servidores Web e Navegadores
    * Uso "automático" => Registro em Entidade (VeriSign).
    * Porta 80 x 443
    * Proteção Fim a Fim (não protege servidor)
- Segurança dos Dados no Servidor
    * SEMPRE necessária
    * Problemas:
        = Modificação/Destruição de bancos de dados
        = Acesso (sem autorização) a seções restritas de website
        = Slave Machine x Zombie Machine
        = Roubo de Informação
- Variáveis Globais
    * Variáveis do Servidor (estáticas)
    * Variáveis GET (pela linha de endereço, ou seja, pela URL)
    * Variáveis POST (por formulários de outra página)
    * Exemplo: http://www.caetano.eng.br/exemplo.php?pagina=37
        = PHPs Antigos: $pagina
        = PHPs recentes: $pag = $_GET['pagina']
    * Exemplo: http://www.servidor.com.br/pagina.php?AUTH=1
- Variáveis de Sessão
    * Armazenam dados da conexão atual
    * Hashs Reversíveis
        = Salting
    * Dados sensíveis => BANCO DE DADOS NO SERVIDOR!
- Acesso a Bancos de Dados
    * SQL Injection
    * SELECT * FROM usernames WHERE name LIKE "$nome";
    * $nome = AAAAA"; DROP DATABASE; SELECT "A
    * Resultado:
        SELECT * FROM usernames WHERE name LIKE "AAAAA";
        DROP DATABASE;

```
                SELECT "A";
        * Eliminar aspas / Substituir aspas / URL Encode
- Includes
        * Repetições nas páginas
                = include "nome_da_pagina.php"
                = include "pagina".$page.".php"
        * Redefinindo $page => adicionar qualquer código à pagina
                = Código externo rodando *no servidor*
        * Desabilitar inclusão de páginas de outro servidor
        * Verificar link antes de incluir seu conteúdo
- Captchas
        * Palavras alfanuméricas distorcidas
        * Evitar SPAM / Múltiplos cadastros automatizados etc
        * Uso com variável de sessão em um DB, para verificação
                = Variável de Sessão / Texto do Captcha
                = Obriga leitura da imagem
        * Evolução dos Captchas:
                = Captchas gerados por seres humanos (fotos)
                        ~ Quantos leões existem de boca aberta?
                = "Gatos"
```

Existem dois tipos de segurança quando se fala em comunicação Web: uma é com relação à criptografia dos dados transmitidos, para que ninguém consiga identificar as informações trocadas entre cliente e servidor. A outra é relativa à segurança dos dados do servidor, quanto à sua integridade e confidencialidade.

3.1. Criptografia dos Dados

Existem várias maneiras de realizar a criptografia dos dados, usando os diversos algoritmos já apresentados no curso. Entretanto, existe uma solução completa desenvolvida na década de 90 pela Netscape Communications, com o nome de Secure Socket Layer (SSL).

Esta tecnologia evoluiu um pouco nos últimos anos e teve seu nome alterado para Transport Layer Socket (TLS), para ficar com um nome mais de acordo com sua função dentro dos modelos teóricos.

É uma grande vantagem usar estes sistemas prontos pois eles estão implementados em praticamente todos os Servidores Web e Navegadores existentes, bastando para seu uso a correta configuração do Servidor Web e o cadastro em um servidor de assinaturas digitais (VeriSign, por exemplo).

A programação do site web é feita normalmente, sendo a única diferença que a porta padrão de comunicação passa a ser a port 443, ao invés da porta 80.

Vale ressaltar que a criptografia dos dados transitados não impede que um cracker atinja o site que hospeda a página: ela apenas protege as informações transitadas "fim a fim".

3.2. Protegendo o Servidor contra Ataques

A segurança das informações transitadas nem sempre é necessária. Por esta razão, apenas em casos específicos é usado o esquema TLS. Por outro lado, a segurança da integridade e confidencialidade das informações do servidor é sempre necessária.

Através de um site web mal programado um cracker pode modificar ou apagar bancos de dados inteiros, acessar seções restritas de site sem ter autorização, usar um determinado equipamento como escravo (para cometer crimes) e até mesmo para roubar informações de arquivos existentes no disco rígido do Servidor Web.

Serão apresentados, a seguir, alguns cuidados básicos que são necessários quando se trabalha com programação web.

VARIÁVEIS GLOBAIS

O primeiro cuidado (e talvez um dos mais importantes) diz respeito às variáveis globais. Para entender isso, é preciso primeiro entender como funciona a passagem de variáveis de uma página web para outra. São basicamente três tipos:

- Variáveis do Servidor (estáticas)
- Variáveis GET (pela linha de endereço, ou seja, pela URL)
- Variáveis POST (por formulários de outra página)

Em algumas linguagens (notadamente versões antigas do PHP) qualquer um destes tipos de variáveis era definido automaticamente em um programa de uma página dinâmica. Por exemplo, considere a URL abaixo:

http://www.caetano.eng.br/exemplo.php?pagina=37

Neste caso temos uma variável do tipo GET (enviada pela URL). Ao chamar uma página desta forma, automaticamente o PHP geraria uma variável chamada $pagina, inicializada com o valor 37. O mesmo ocorreria se a variável tivesse sido definida em um formulário (tipo POST) ou estivesse definida no servidor.

Esta característica, chamada de "Variáveis Globais", era bastante prática. Entretanto, isso propiciava uma baixa segurança. Mas por quê?

Simples. Imagine que, dentro de um software de uma página Web eu defina que a variável $AUTH identifica que um usuário está autenticado. Ou seja, quando o usuário entrar

com um nome e senha adequados, o programa faz com que $AUTH = 1. Se o usuário der logoff ou errar ou não digitar o nome e login, o programa faz com que $AUTH = 0.

Ora, se um cracker descobrir o nome desta variável $AUTH (há alguns nomes bastante comuns) e chamar a URL da seguinte forma:

http://www.servidor.com.br/pagina.php?AUTH=1

Se o servidor estiver com variáveis globais ligadas, ele terá acesso à região em que apenas usuários autorizados possuem acesso, mesmo sem ter realizado login algum, porque automaticamente o PHP fará com que a variável $AUTH valha 1, ou seja, indicando que o login já foi realizado com sucesso.

Por esta razão, a maioria das linguagens web permite desligar as variáveis globais e, em alguns casos (como as versões recentes do PHP) este comportamento vem desligado "de fábrica". Por esta razão, quando for necessário saber algum destes valores dentro de um código Web, é preciso indicar essa necessidade na linguagem de programação. Por exemplo, se uma página for chamada com a URL abaixo:

http://www.caetano.eng.br/exemplo.php?pagina=37

E dentro do código for necessário saber qual o número da página, é preciso que na linguagem seja solicitada a leitura do valor de "pagina" em uma variável qualquer. Como se trata de uma variável na URL, é uma variável do tipo GET. Assim, é preciso indicar que se leia o valor da variável tipo GET cuja descrição é "pagina". Em PHP, isso poderia ser feito com a seguinte instrução:

$variavel = $_GET['pagina'];

Desta forma, mesmo que o invasor descubra que a variável $AUTH é usada para definir usuários que já fizeram login, e tentar um ataque como indicado anteriormente (http://www.servidor.com.br/pagina.php?AUTH=1), como o programa não lerá essa variável da linha de comando (com um $AUTH = $_GET['AUTH'];) tais falhas de segurança estão superadas.

VARIÁVEIS DE SESSÃO

Uma forma comum de armazenar informações de login e outros via Web é através do uso da URL. Entretanto, isto torna o sistema muito vulnerável, por informações importantes sobre o funcionamento interno seriam reveladas por uma simples inspeção da URL.

Uma maneira de evitar este problema é codificar de uma forma complexa a informação ao ser colocada na URL, usando uma espécie de HASH reversível. A este tipo de variável, que também pode ser armazenada em um cookie, também é chamada de Variável de Sessão, sendo um número único que define um determinado acesso de um usuário.

Usando uma variável de Sessão, a inspeção da URL por alguém que não saiba como o HASH foi gerado não traz novas informações. Além disso, é muito difícil que alguém consiga codificar informações no formato necessário para manipular o comportamento do servidor. Por outro lado, a segurança na é perfeita e, portanto, informações muito sensíveis não devem ser codificadas em variáveis de sessão, sendo preferível deixá-las armazenadas em banco de dados no servidor.

ACESSO AO BANCO DE DADOS

Em praticamente todo web site dinâmico, existe um uso extensivo de um SGBD. Embora este uso seja muito prático e interessante, são necessários alguns cuidados para evitar problemas com possíveis ataques.

Um ataque comum a banco de dados é aquele que visa apagar todo seu conteúdo. Mais grave ainda, pode-se ter um ataque que revele todas as informações do banco de dados ao atacante. Se estas informações forem sensíveis e não criptografadas, tal revelação não é considerada aceitável.

A principal fonte de problemas ao se programar bancos de dados está no uso de variáveis dentro de queries SQL onde existem aspas (sejam elas simples ou duplas). Por exemplo:

```
SELECT * FROM usernames WHERE name LIKE "$nome";
```

Se o invasor consegue adulterar o conteúdo de $nome para algo como:

```
AAAAA"; DROP DATABASE; SELECT "A
```

Observe como será composto o query:

```
SELECT * FROM usernames WHERE name LIKE "AAAAA";
DROP DATABASE;
SELECT "A";
```

Apesar de parecer algo fora do comum, é um ataque bastante praticado e seu nome é "SQL Injection" (Injeção SQL). E, para piorar, muitos sites não estão preparados para evitar este tipo de "truque". Mas como se proteger disso?

A primeira forma é não permitir "aspas" nos valores das variáveis que são usadas internamente às queries. Para isso, basta criar uma função simples que remova as aspas de qualquer variável que seja usada dentro das queries. Algumas linguagens até mesmo fornecem este tipo de função pronta.

A segunda forma, usada quando as aspas são necessárias no texto, é substituir as aspas por algum outro caractere ao inserir o texto no banco de dados, lembrando de converter de volta, quando futuramente o banco de dados for lido. Uma forma alternativa é o uso de "URL Encode" nos textos, que converte todos os caracteres especiais (incluindo aspas) para a forma do HTML, que também não causará problemas com as queries. Muitas linguagens também vêm com funções para converter o texto de uma variável na sua forma "URL Encoded".

INCLUDES

Quando são criadas páginas dinâmicas, muitos dos trechos de código e/ou conteúdo podem ser repetidos em vários lugares. Nestes casos, estes trechos são gravados em arquivos específicos e são adicionados nas páginas necessárias usando as diretivas "include":

include "nome_da_pagina.php"

Por exemplo, se uma variável $page diz o número da página atual, é possível "incluir" na página atual o conteúdo do arquivo da página pagina1.php com a seguinte instrução (exemplo em PHP):

include "pagina".$page.".php";

Entretanto, mais uma vez, se o invasor tiver acesso à modificar o conteúdo da variável $page de alguma maneira, ele pode conseguir incluir código malicioso em sua página, que será executado como se fosse parte da página original.

Este é um tipo de ataque muito comum, onde normalmente o invasor faz com que sua página inclua código de alguma outra página (criada por ele, em outro servidor). Assim, uma forma bastante eficaz contra este tipo de ataque é através de medidas para desabilitar a inclusão de conteúdo de servidores externos ao que está executando a página.

De qualquer forma, isso nem sempre é possível, já que algumas páginas podem precisar incluir conteúdos legítimos de sites externos. Nestes casos, é preciso tomar um cuidado adicional, criando um sistema que verifica o link a ser incluído, para que possam ser excluídos links maliciosos.

CAPTCHAS

A maioria das pessoas que usam a Internet já viram um captcha, mas poucos sabem que aquilo se chama "captcha". Captchas são aquelas pequenas "palavras" alfanuméricas, apresentadas em caracteres distorcidos, que alguns serviços solicitam quando tentamos acessá-los:

No exemplo, o conteúdo é "ADCM". A finalidade dos captchas é impedir que programas automáticos sejam usados para criar contas em sites, enviar mensagens SMS e outros, impedindo o uso destes serviços para divulgação de SPAM, por exemplo.

Os captchas são gerados normalmente como uma seqüência de caracteres aleatória, escritos em um pedaço de imagem usando alguma linguagem como PHP, e posteriormente esta imagem é distorcida, com algumas linhas movimentadas, pontos coloridos aleatórios desenhados, retas traçadas por sobre o texto, filtros de embaçamento de imagem, etc.

Antes de ser enviada a imagem para o usuário, é também estabelecida uma variável de sessão e, em um banco de dados local é indicado o texto do captcha para aquela variável de sessão:

Variável de Sessão	Texto do Captcha
6a786e65fb55a4667dea234f	ADCM

Isso é feito desta forma para que, quando o usuário digitar o texto e enviar pelo formulário, seja possível verificar no banco de dados se o valor apresentado e o valor digitado batem. Esta é a única forma 100% segura para que não seja possível identificar o texto sem precisar analisar a figura.

É sabido, porém, que qualquer imagem gerada por computador pode ser identificada (lida) por um computador, ainda que isso tenha uma taxa de erros e não seja exatamente um processo rápido. Entretanto, com o avanço do poder de processamento dos computadores, talvez os captchas gerados automaticamente não sejam suficientes para conter o SPAM no futuro.

Por esta razão, já existem algumas iniciativas em bancos de "captchas gerados por seres humanos", que que são tiradas fotos de coisas da vida real (como por exemplo uma imagem de um zoológico) e uma pergunta possível seria "Quantos leões de boca aberta existem na foto?".

4. ESTEGANOGRAFIA

Conceitos Chave:

- Mensagens Criptografadas chamam atenção
  * Escondê-las!
- Esconder mensagens criptografadas em imagens
- Imagens: pontos R G B / R G B A, cada um com variação de 0 a 255
- Intensidades:
  (0,0,0) Preto
  (255,255,255) Branco
  (128,128,128) Cinza Escuro
  (255,0,0) Vermelho
- Composição de cada intensidade, variando tons de 0 a 128

```
Bit    7      6      5      4      3      2      1      0
Uso    \_____________________ Imagem _________________/   Dado adicional
```

- Imagem de 1024 x 768 pode armazenar 288K de informação
- Processo:
  1) A escreve texto.
  2) A calcula o hash do texto.
  3) A criptografa o hash do texto com sua chave privada, gerando a assinatura digital.
  4) A anexa a assinatura digital no texto.
  5) A compacta o texto e a assinatura digital com um programa como PKZip.
  6) A criptografa o ZIP com a chave pública de B.
  7) A insere o ZIP criptografado em uma imagem qualquer que ele tenha tirado.
  8) A envia a imagem para B, com algum comentário inocente como "Olhe que linda rosa!"

Um dos problemas clássicos da criptografia é que sempre que vemos uma mensagem criptografada, sabemos que existe uma mensagem e que ela está criptografada. Por esta razão, chama-se atenção ao fato de que aquela mensagem pode ser importante e sempre haverá pessoas interessadas em "quebrar" aquela criptografia apenas para descobrir o conteúdo da mensagem.

Uma forma imaginada para evitar que mensagens criptografadas chamem atenção de pessoas indesejadas é justamente o uso da esteganografia (ou, do grego, escrita oculta). A esteganografia é baseada na inserção de um texto (criptografado ou não) em um arquivo de imagem ou arquivo sonoro, de forma que as pessoas que vejam esta figura ou ouçam o arquivo sonoro não sejam capazes de dizer que há uma mensagem escondida ali.

Mas como isso funciona? Primeiramente vejamos como uma imagem é armazenada no computador, e posteriormente veremos como podemos inserir um texto em uma imagem sem que as pessoas percebam.

A imagem, nos computadores, é sempre composta por pequenos pontos, sendo que cada um destes pontos é composto por 3 pontos menores, cada um de uma cor distinta: vermelho, verde e azul (Red, Green, Blue... RGB).

Para gerar as diferentes cores, o computador atribui intensidades diferentes a cada um destes pequenos pontos e cores vermelho, verde e azul. Usualmente cada uma destas intensidades tem 8 bits, ou seja, variam de 0 a 255. É comum representar as intensidades das três cores entre parênteses, separados por vírgula, na forma (RRR, GGG, BBB), ou seja, primeiro vem o valor do vermelho, depois do verde e, por fim, o valor do azul. Desta forma, podemos identificar algumas cores:

(0,0,0)	Preto
(255,255,255)	Branco
(128,128,128)	Cinza Escuro
(255,0,0)	Vermelho
(0,255,0)	Verde
(0,0,255)	Azul
(255,255,0)	Amarelo
(255,0,255)	Magenta (um tipo de roxo roxo)

Ocorre que, apesar de definirmos 8 bits de intensidades para cada uma das cores, na prática o ser humano praticamente não consegue identificar diferenças entre dois valores consecutivos. Ou seja: (0,0,254) e (0,0,255) são, para os nossos olhos, a mesma cor. Mesmo num caso como (0,0,0) e (1,1,1), também não somos capazes de perceber diferença entre um e outro. De maneira formal, o bit menos significativo dos 8 bits é indiferente à percepção humana e, portanto, o valor original do mesmo pode ser descartado e esta posição pode ser usada para guardar qualquer tipo de conteúdo.

Assim, temos que, para cada intensidade de vermelho, verde ou azul:

Bit	7	6	5	4	3	2	1	0
Uso	_____			Imagem			_____/	Dado adicional

Como cada ponto da imagem tem 3 intensidades, temos 3 bits disponíveis para armazenar dados dentro deste ponto da imagem. Com essas contas, uma imagem de 1024 x 768, que tem então 786432 pontos, pode armazenar até 3 x 786432 = 2359296 bits... ou seja, 294912 bytes. Em outras palavras, em uma imagem de 1024 x 768 é possível colocar um texto inteiro de 288 Kbytes sem que ninguém que observe a figura se dê conta disso... sendo que isso é espaço o suficiente para colocar 3 peças inteiras de Shakespeare, se compactadas!

O processo que muitos usam para transmitir dados sensíveis de A para B é, então, o seguinte:

1) A escreve texto.
2) A calcula o hash do texto.
3) A criptografa o hash do texto com sua chave privada, gerando a assinatura digital.
4) A anexa a assinatura digital no texto.
5) A compacta o texto e a assinatura digital com um programa como PKZip.

6) A criptografa o ZIP com a chave pública de B.

7) A insere o ZIP criptografado em uma imagem qualquer que ele tenha tirado.

8) A envia a imagem para B, com algum comentário inocente como "Olhe que linda rosa!"

Para ler a mensagem, B deve fazer o processo inverso.

É claro que pessoas que precisam de um sistema deste tipo com freqüência usam programas que fazem todo este processo automaticamente, mas é importante saber o que está por trás de tal processo.

5. ATIVIDADE

Seu chefe está em uma viagem ao exterior e precisa ter acesso de leitura a alguns dados importantes do banco de dados, dados estes que só poderiam ser acessados com acesso físico, como root, ao servidor de banco de dados. Estes dados são extremamente sigilosos, de forma que não pode existir a possibilidade de vazamento da informação.

O laptop de seu chefe é equipado com o PGP, SSH, é capaz de fazer navegação via TLS, assim como todos os demais recursos de segurança que por ventura sejam necessários.

Qual seria a solução a ser apresentada ao seu chefe?

6. BIBLIOGRAFIA

TANENBAUM, A. S. **Redes de Computadores**. Editora Campus, 2003.

TERADA, R. **Segurança de Dados:** Criptografia em Redes de Computador. São Paulo: Edgard Bücher, 2000.