Sistema Operacional Linux

Rodrigo Rubira Branco
rodrigo@firewalls.com.br

O que é o ISPA?

- **Empresa Especializada em Treinamentos Tecnologicos;**
- **Diversas formaçoes profissionais divididas em categorias;**
- **Formaçao Network;**
- **www.ispa.com.br**
- **PARCERIA iSPA/SENAC**

O que é a Firewalls?

- **Empresa Especializada em Segurança;**
- **Profissionais Certificados;**
- **Atenta a Padrões Internacionais;**
- **Parceira das maiores empresas de Segurança do Mundo;**
- **Fornecedora de materiais didaticos em Segurança, Linux e Network para o ISPA**

Objetivos da Apresentaçao

- **Diferenciar segurança em ambientes opensource x closedsource**

- **Demonstrar problemas de segurança existentes em ambiente Linux e como resolve-los**

- **Apontar sistemas de segurança para ambientes Linux e as necessidades de utiliza-los**

O QUE NAO SERA VISTO

“linhas de comando e implementaçoes”

Conceitos de Segurança

CIDAL =

C onfidencialidade

I ntegridade

D isponibilidade

A utenticidade

L egalidade

LEMBRAR-SE SEMPRE DISTO NO DECORRER DA APRESENTAÇAO!

Ambiente em Questão

Trabalhamos em um mundo real de sistemas mal configurados:

*** Bugs de Software**

*** Empregados Insatisfeitos**

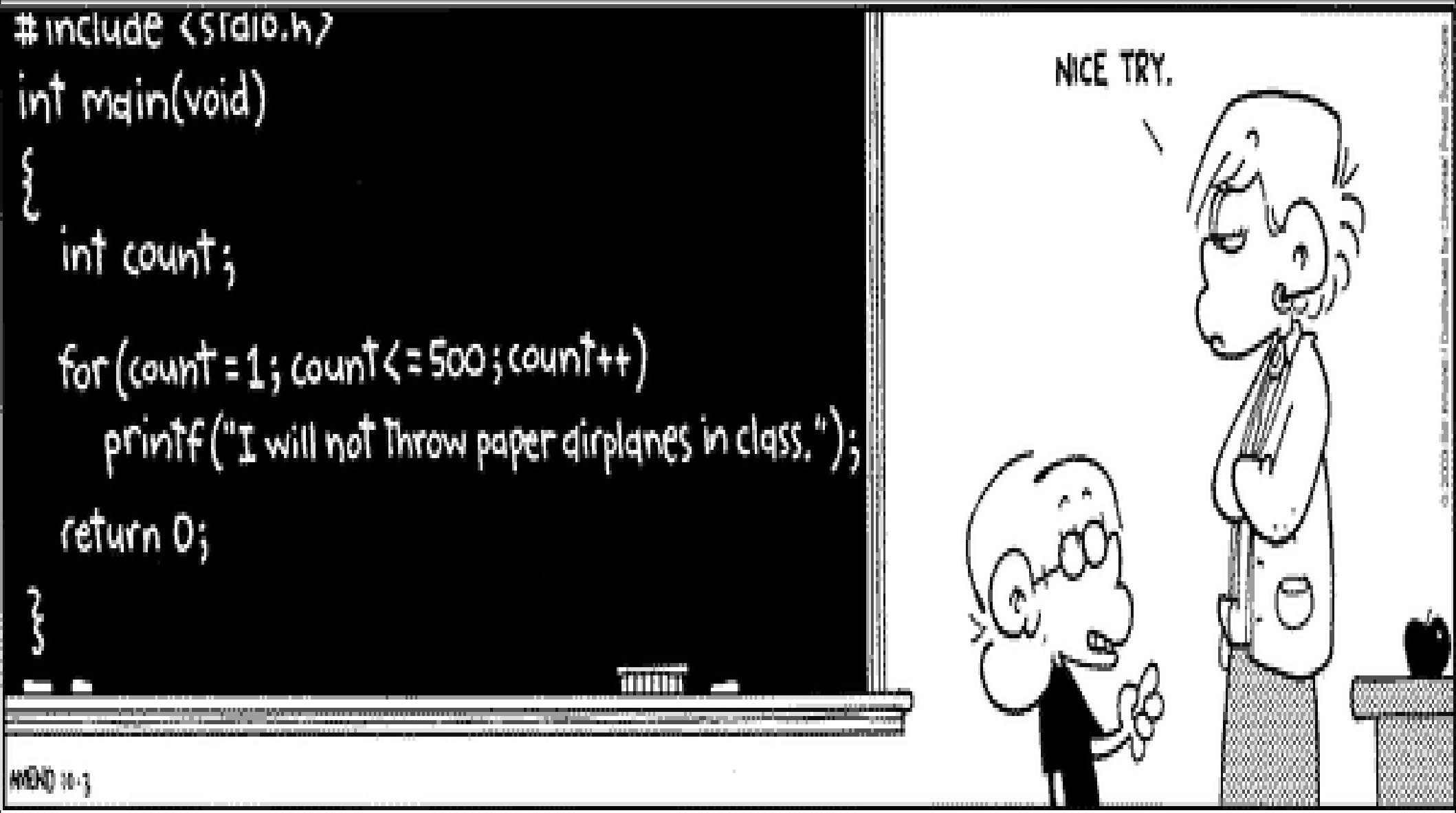
*** Administradores de Sistemas Sobrecarregados**

*** Acomodação de necessidades empresariais**

*** Falta de Educação em Segurança**

*** B2B,B2C,B2E,C2C,X2X?**

Onde os problemas começam



Defense In-Depth

- Através de diversas camadas cria-se um modelo de segurança robusto e capaz de suportar falhas.

Quando pensar em Segurança pense em uma CEBOLA.

Dificuldades: SMART

- **S**pecific = especifico
- **M**easurable = mensurável
- **A**chievable = alcancável
- **R**ealistic = realístico
- **T**ime-based = baseado em tempo

Dificuldades: FUD

- FEARS, UNCERTAINTIES, DOUBTS

(medos, incertezas, duvidas)

- Filosofia do MEDO nao existe, nao devemos temer os problemas de segurança, enfrente-os com soluçoes razoaveis e que valham o investimento!

Open x Closed

- Grande facilidade em realizar-se auditoria de codigos

- Falhas facilmente exploraveis

- Tempo de resposta e correçao independente de fornecedores

- Facilidade na aplicabilidade de correçoes menor

- Dificuldade em se auditar codigos

- Maior dificuldade em se explorar falhas existentes

- Tempo de resposta e correçao dependente de fornecedores

- Velocidade na aplicabilidade de correçoes quando disponiveis maior

Ferramentas Existentes

Ferramentas Existentes

O que veremos?

- Permissoes
- Filtros de Acesso e Conteudo
- Detecçao de Intrusos
- Segurança de Serviços
- Gerenciamento de Eventos
- Backups
- Alta disponibilidade
- Sistemas de Logon
- Virus e Programas Maliciosos
- Ferramentas de Analise
- Recursos In-depth: Seguranca em Kernel

Permissoes

- Conceitos de permissoes baseados em primitivas como Dono, Grupo Dono e Outros

- Sistema de permissoes limitado e nao condizendo com as necessidades atuais de segurança

Soluçoes: Kernel 2.6.x RBAC

Grsecurity

TrustedBSD (sistema operacional OpenSource que visa transferir para o FreeBSD quesitos da Common Criteria for Information Technology Security Evaluation).

Filtros de acesso

- Um sistema operacional Linux com Kernel 2.4.x ou superior pode atuar como um poderoso sistema de Firewall

- Iptables (netfilter) atua dentro do kernel e possui recursos de Firewall Stateful com Connection Track

- Regras como redirecionamento de portas e NAT de diversos tipos podem ser criadas

Filtros de conteudo

- Normalmente sistemas de filtragem (como netfilter) atuam ate o nivel de transporte do modelo OSI

- Filtros de conteudo atuam ate a camada de Aplicacao, reconhecendo o trafego que passa por eles

- Possibilidade de filtragem de acesso por URL, conteudo da pagina e muitas outras combinacoes (protocolo utilizado por exemplo), permitindo assim um sistema Firewall Linux bloquear todo tipo de serviço indesejado na rede (incluindo instant messengers de todo tipo e aplicativos p2p)

Deteccao de Intrusao

Linux pode atuar facilmente como um detector de intrusos, tanto para a rede (com ferramentas como o snort), como para os serviços da propria maquina (utilizando-se portsentry por exemplo).

Tal detector de intrusos pode responder automaticamente a tentativas de ataques, bloqueando as maquinas agressoras.

Serviços Seguros

Diversas opçoes existem em linux de cada um dos tipos de servidores possiveis (diferentemente dos sistemas operacionais concorrentes).

Serviços como sendmail, podem ser substituidos por equivalentes mais seguros como qmail ou postfix.

Serviços Seguros

Outros serviços, tambem antigos e portanto complexos e com possibilidades de muitas falhas possuem substitutos muito poderosos com segurança como foco principal.

Wu-ftp por exemplo pode ser substituido por proftpd ou vsftpd.

Gerenciamento de Eventos

O sistema operacional Linux normalmente lhe fala tudo o que esta acontecendo com ele.

Verificar e gerenciar logs e eventos que ocorrem e fundamental para a detecçao de problemas

A possibilidade de centralizar-se os logs (ja disponivel com syslogd) e de se executar programas quando determinado evento acontece (syslog-ng), torna o linux um poderoso sistema auto-gerenciavel quando bem implementado

Backup

Quando tratamos de segurança o backup e a contigencia sao imprescindiveis.

Sistemas como:

- **Amanda (centralizaçao de backup de servidores linux)**
- **DRDB (RAID0 via rede com linux)**
- **DD (espelhamento de disco linux)**

E outras, tornam o Linux um poderoso sistema para ser facilmente restaurado em casos de pane.

Alta Disponibilidade

Sistemas redundantes, assumindo recursos de softwares que saem do ar sao opcoes facilmente implementadas utilizando-se software open source (heartbeat);

Cluster de computadores, tornando diversos computadores PC em um unico super computador, tambem torna-se viavel com o uso de Linux (clusters mosix);

Devido ao codigo fonte aberto dos aplicativos, otimizaçoes podem facilmente ser feitas, tais como a mudança do source do apache do php para lidar com balancemanto de bancos mysql.

Gerenciamento de Logon

- Usar ou nao usar PAM? “Slackware: Because the simplicity is divine”

- Samba como servidor de logon na rede (executando scripts de logon, distribuindo atualizacoes de seguranca, etc)

- NFS e diretorios via rede

Problemas com Virus

- **Virus x Worms x Trojan x Codigos Maliciosos**

- **O que o linux enfrenta**

- **Antivirus para linux**

Analisando Tudo

- **Testando o detector de Intrusos: Scmorphism**

- **Testando os serviços instalados: Nessus**

- **Testando a Stack TCP/IP: Nmap**

Recursos In-depth

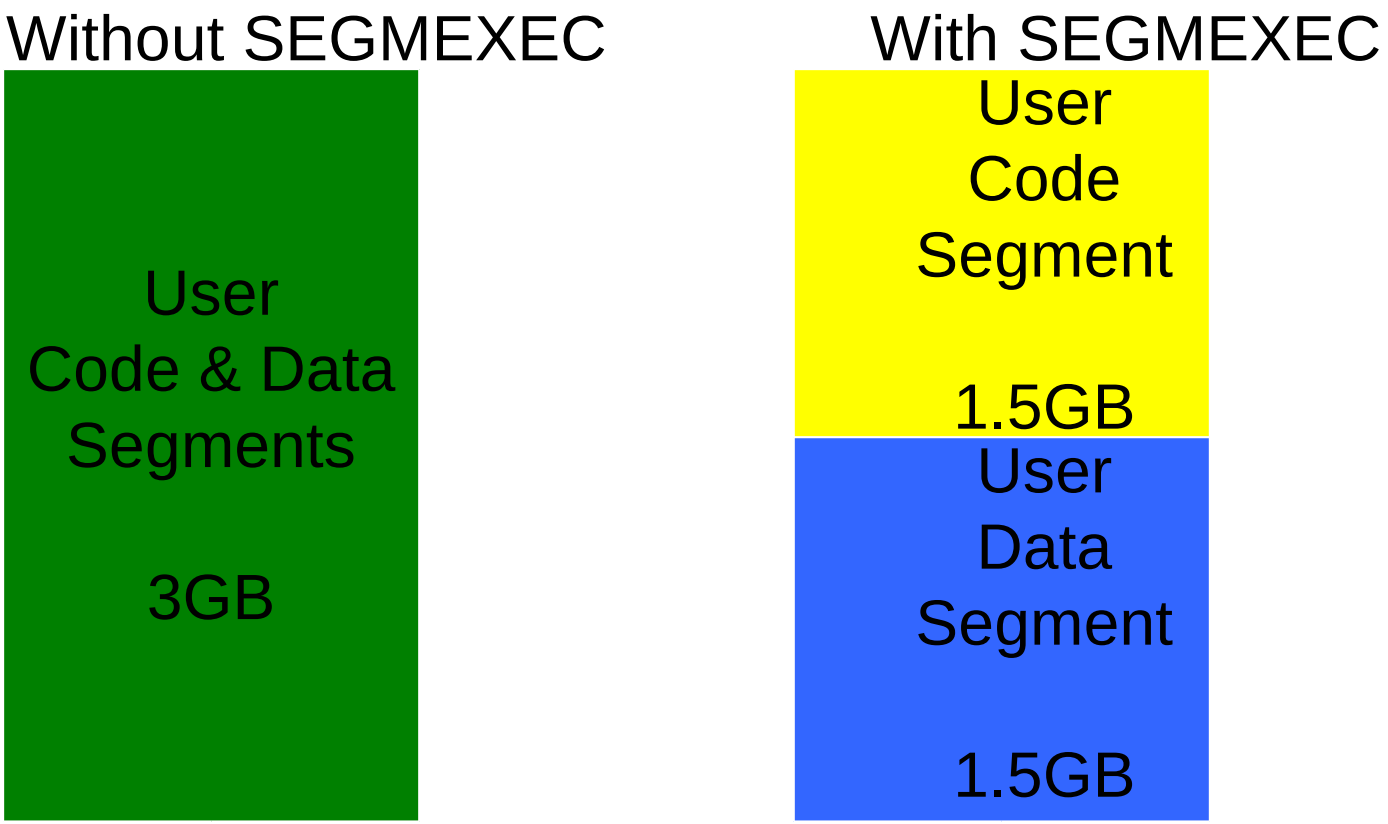
- **Stack nao executavel**

- **Endereços de bibliotecas randomizados**

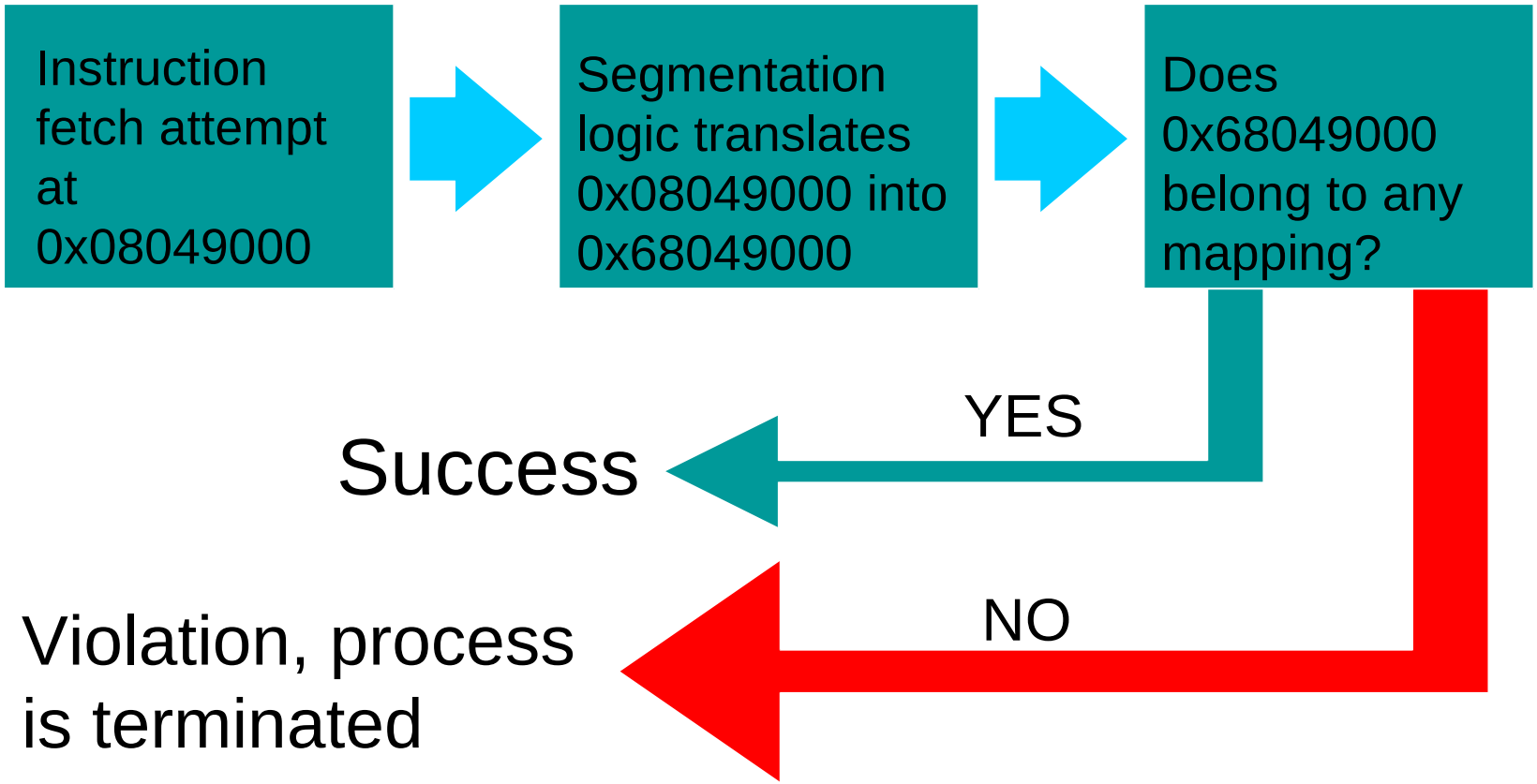
- **Inserçoes nao contiguas na Stack --> Finalizadores de bloco**

- **Heap “segura” -> Insercao de structs de controle**

PaX


Without SEGMEXEC
User
Code & Data
Segments
3GB
With SEGMEXEC
User
Code
Segment
1.5GB
User
Data
Segment
1.5GB

PaX


Instruction fetch attempt at 0x08049000
Segmentation logic translates 0x08049000 into 0x68049000
Does 0x68049000 belong to any mapping?
YES
Success
NO
Violation, process is terminated

Kernel Security

Ambiente Chroot

Modulos de Segurança (chamadas de sistema monitoradas, por exemplo ptrace) – systrace

Securelevels, proteçao contra carregamento de modulos, etc

Referencias

http://www.firewalls.com.br

http://www.bsdaemon.org

http://www.linuxsecurity.com.br

http://www.securityfocus.com

http://www.zone-h.com

Lembrete: Segurança REAL

"Segurança, segundo nossa filosofia e um processo complexo que vai desde uma boa interface com o usuario, ate uma boa administraçao e atualizaçao dos servidores, aliados a uma programaçao em constante teste e correçao.

- **INDEPENDE DO USUARIO**

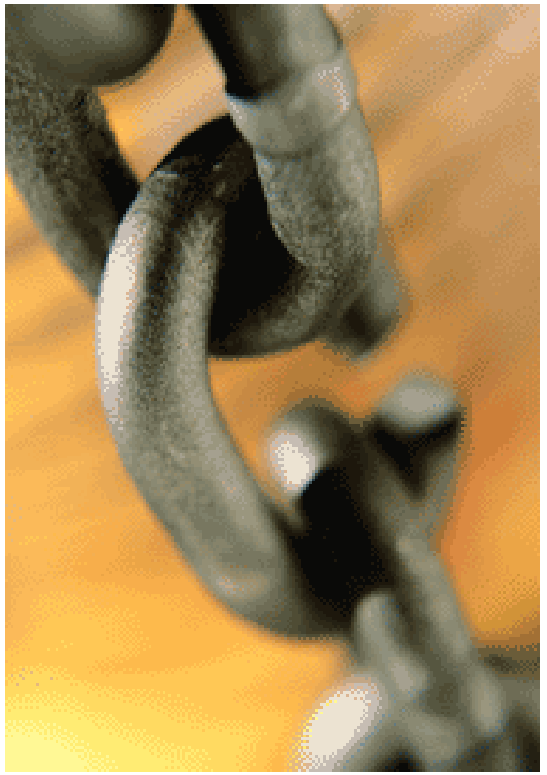
- **EMBORA SEJA REGULAMENTADA, DEVE SER IMPOSTA**

Agradecimentos

- **Eventos, Workshops, Seminarios, Palestras, RoadShows, Conversas**

- **Comunicaçao**

- **Elo mais fraco da segurança da informaçao**



FIM! Será mesmo?

DÚVIDAS?!?

Rodrigo Rubira Branco

rodrigo@firewalls.com.br

Brinde: Falhas de Software

- Stack Overflow
- Heap Overflow
- Race Condition
- Format String
- Off-by-one
- Off-by-few
- Integer Overflow
- Double-free

Stack Overflow

- Falha muito comum (provavelmente a mais comum entre todas e a mais facil de ser explorada)

- Existe grande dificuldade de detecçao por tratar-se de interaçoes e loops, uso de funçoes do proprio usuario

- Provavelmente poderia ser detectada atraves de analise de funçoes perigosas e como estao sendo usadas, dando-se grande numero de FALSOS NEGATIVOS

- Consiste da alocaçao de uma variavel com tamanho fixo e da tentativa de armazenar-se nela mais dados do que sua declaraçao continha (geralmente dado pelo usuario).

- Sobrescreve-se na pilha o RET, que quando a funçao retornar sobrescreve o registrador EIP e permite execuçao de codigo arbitrario.

Heap Overflow

-Deve-se levar em consideraçao quase todos os quesitos dos itens anteriores

- Acontece com variaveis alocadas dinamicamente e ponteiros

- Um estouro em uma variavel acarreta na sobrescrita do ponteiro de outra, permitindo execuçao de codigo arbitrario ou sobreposicao de ponteiros permitindo passar-se por autenticaçoes e escrita em arquivos aleatorios

Integer Overflow

- Enquanto os programadores e auditores de codigo cada vez mais se atentam para falhas de Stack/Heap Overflow, uma nova modalidade surge

- Integer overflow nada mais e do que a tentativa de armazenamento de valores maiores do que os possiveis em variaveis inteiras (short, long, etc)

- Devido a um integer overflow, pode-se passar por testes de condiçoes e conseguir-se acarretar situaçoes de Stack/Heap overflows mesmo onde nao existiam, dai a dificuldade em detecçao automatizada deste tipo de procedimento

Race Condition

- Uma das falhas mais dificeis de se detectar em auditorias manuais de codigo e em auditorias automaticas de fonte

- Geralmente pode ser descoberta automaticamente atraves do uso do FUZZER, por acarretar a situaçao de condiçao de corrida com o stress do software

- Consiste em se explorar condiçoes de acessos a recursos compartilhados, onde geralmente o resultado obtido consiste em execuçao de codigos com privilegios elevados (escalaçao de privilegios)

Format String

- Consiste em se explorar condiçoes em que uma funçao tem o objetivo de ser utilizada para diversas finalidades, recebendo nao apenas parametros (variaveis/ponteiros) como tambem o formato como estes devem ser impressos (se serao strings, hexa, inteiros, etc).

- Funçoes da familia printf, syslog e etc sao muito suscetiveis a este tipo de ataque

- Muitas vezes o programador esquece da string de formato, o que tambem permite o acontecimento destas falhas

Off-by-one

- Erro comum e de dificil exploraçao consiste em se declarar uma variavel (ou alocar dinamicamente memoria para ela) esquecendo-se de um byte

- Muito comum principalmente porque programadores esquecem do terminador de string “\0” que ocupa um byte em qualquer string de caracteres em C

- Atraves deste tipo de falha costuma-se passar por condiçoes de testes e acarretar-se outros tipos de falhas para execuçao de codigo arbitrario

Off-by-few

- Mesmo principio de funcionamento do off-by-one, mas neste caso seriam alguns poucos bytes

- Nao se iguala ao stack overflow por exemplo, pois esta quantidade de bytes nao e suficiente para sobrescrever-se o RET (EIP).

- A exploraçao se da igualmente o off-by-one e tambem existem tecnicas de sobreposicao de outros registradores e manipulacao de espacos de memoria para execuçao direta de codigo arbitrario

Double-free

- Apos se alocar memoria atraves do uso de funçoes da familia alloc em C, e muito comum livrar-se destes junks atraves da funçao free(ponteiro)

- Muitas vezes estas condiçoes de free estao dentro de variaveis condicionais

- Alguns casos podem ocorrer em que estas condicionais sao manipuladas (atraves de outras falhas) para serem verdadeiras e causarem o free duas vezes em um ponteiro

- Perceba-se que a falha em si consiste em se liberar memoria que ja nao mais esta alocada, pois a funçao free podera executar o codigo arbitrario armazenado naquele setor

Directory Traversal

- Acontece quando se aceita dados fornecidos pelo usuario e atraves deste tenta-se acessar/criar/gravar/manipular algum arquivo

- O usuario pode fornecer paths relativos, como ../.. ou ././ para passar por checagem de Strings ou executar/carregar arquivos indevidos

- Kernel do solaris ja apresentou este tipo de problema: Determinada system call recebia do usuario o nome do modulo a ser carregado e procurava o mesmo no diretorio /usr/modules. O usuario poderia especificar por exemplo ../../../home/meumodulo e carregar modulo proprio, ganhando privilegios de administrador