



As 10 mais críticas vulnerabilidades de segurança em **Aplicações Web**

Carlos Serrão
OWASP Portugal
ISCTE/DCTI/Adetti/NetMuST
Abril, 2009
carlos.serrao@iscte.pt
carlos.j.serrao@gmail.com

OWASP

Copyright © 2004 - The OWASP Foundation
Permission is granted to copy, distribute and/or modify this document under
the terms of the GNU Free Documentation License.

The OWASP Foundation
<http://www.owasp.org>

O que é segurança de Aplicações Web?

■ Não é Segurança de Redes

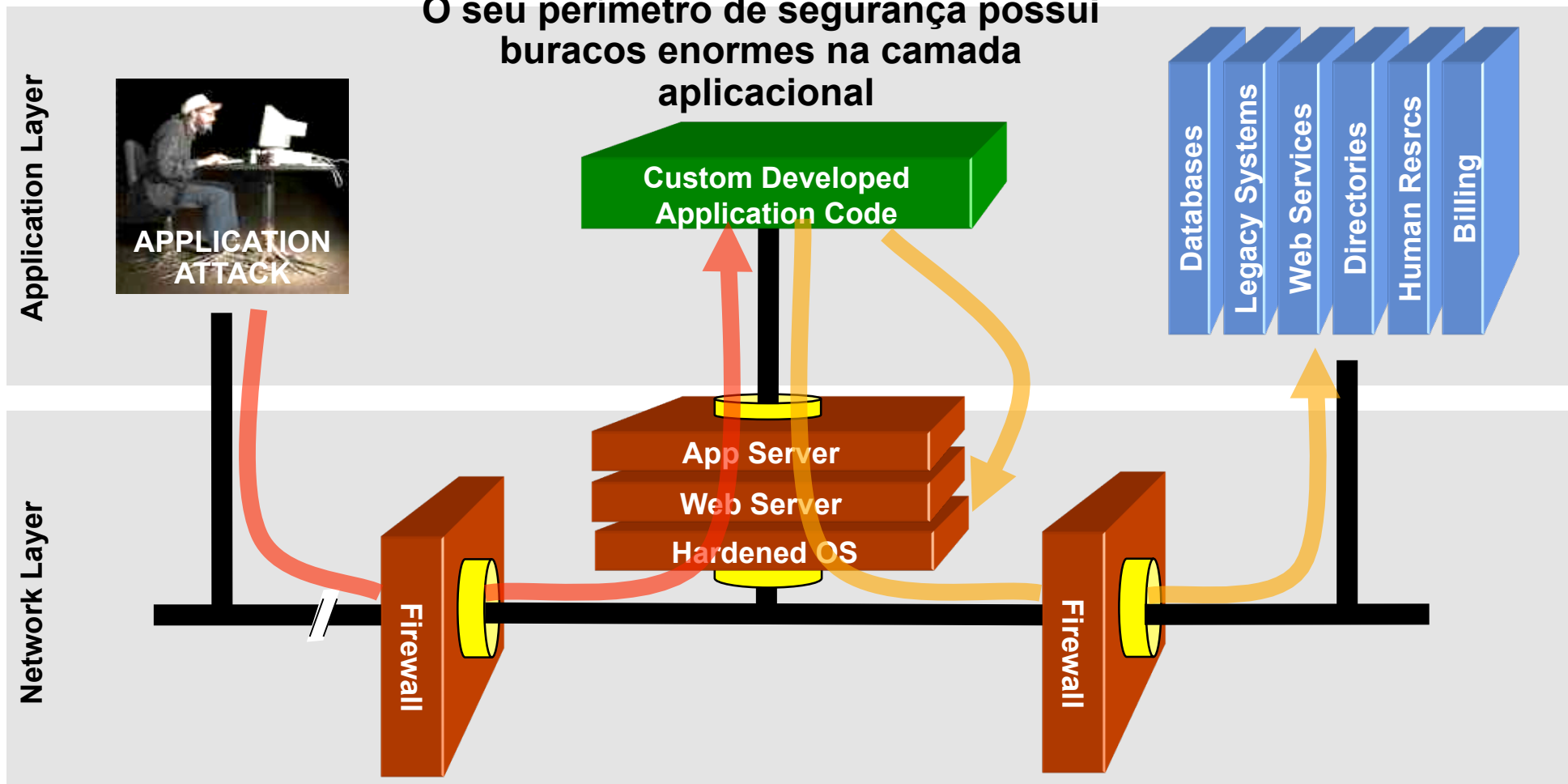
- ▶ Segurança do "código" criado para implementar a aplicação web
- ▶ Segurança de bibliotecas
- ▶ Segurança de sistemas de back-end
- ▶ Segurança de servidores web e aplicativos

■ Segurança de Redes ignora o conteúdo do tráfego de HTTP

- ▶ Firewalls, SSL, Intrusion Detection Systems, Operating System Hardening, Database Hardening

O Código faz parte do perímetro de segurança

O seu perímetro de segurança possui buracos enormes na camada aplicacional



Não é possível usar protecção ao nível da camada de rede (firewall, SSL, IDS, hardening) para parar ou detectar ataques ao nível aplicacional



Isto é preocupante?

■ Vamos lá pensar...

- ▶ Qual a probabilidade de sucesso de um ataque contra uma aplicação web?
 - Probabilidade elevada
 - Fácil de explorar sem conhecimento e ferramentas especiais
 - Quase indetectável
 - Existem milhares de programadores web, pouco preocupados com segurança
- ▶ Consequências?
 - Corrupção de dados ou destruição de BD
 - Acesso root a servidores web ou aplicativos
 - Perda de autenticação e de controlo de acesso de utilizadores
 - Descaracterização (Defacement)
 - Ataques secundários a partir da própria aplicação web

Isto é preocupante?

- A Segurança de Aplicações Web é tão importante como a Segurança de Redes
 - ▶ Porque é que grande parte do investimento em Segurança é canalizado para a segurança das redes?

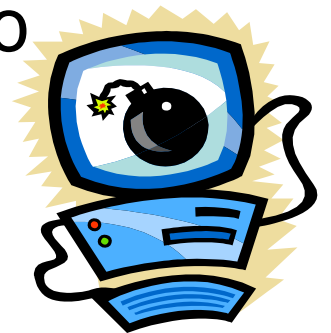
1. Parâmetros não validados

- Pedidos HTTP dos browsers para as WebApps
 - ▶ URL, Querystring, Form Fields, Hidden Fields, Cookies, Headers
 - ▶ As WebApps usam esta informação para gerar páginas

- Os atacantes podem modificar tudo num pedido

- Pontos-chave:

- ▶ Verificar tudo antes de usar o pedido HTTP
- ▶ Canonicalizar/Normalizar antes de verificar
- ▶ Validações Client-side são quase irrelevantes
- ▶ Rejeitar tudo o que não seja especificamente aceite
 - Tipo, Tamanho min/max, conj. de caracteres, regex, Valores min/max ...



2. Controlos de Acesso quebrados

- O controlo de acesso é a forma como se controla o acesso a aplicações e a informação
- O problema é que muito ambientes oferecem autenticação, mas não lidam correctamente com o controlo de acessos
 - ▶ Muitos sites tem políticas complexas de controlo de acesso
 - ▶ De muito difícil implementação
- Pontos-chave
 - ▶ Anote a política de controlo de acessos
 - ▶ Não use "IDs" que um atacante pode manipular
 - ▶ Implemente o controlo de acessos num módulo central

3. Quebra de Contas e da Gestão de Sessões

■ Gestão de Contas

- ▶ Lidar com as credenciais entre o cliente e o servidor
- ▶ Autenticação no backend necessita de credenciais também

■ Gestão de Sessões

- ▶ O HTTP é um protocolo "stateless". As WebApps necessitam de saber que pedido veio de que utilizador
- ▶ Marcar/Propagar as sessões com um ID usando cookies, campos hidden, tags no URL, etc...

■ Pontos-chave

- ▶ Manter sempre as credenciais secretas
- ▶ Usar apenas um ID de sessão aleatório fornecido pelo ambiente de aplicação

4. Falhas de Cross-Site Scripting (XSS)

- Os browsers Web executam código enviado por sites de web
 - ▶ Javascript
 - ▶ Flash, e outros...
- Mas o que acontece se um atacante conseguir que um site web origine um ataque?
 - ▶ Armazenado – a aplicação web armazena conteúdo do utilizador, e depois envia-o a outros utilizadores
 - ▶ Refletido – a aplicação web não armazena o ataque, apenas o envia a quem enviou o pedido
- Pontos-chave
 - ▶ Não vale a pena filtrar todas as combinações possíveis – usar uma especificação positiva.

5. Buffer Overflows

- As aplicações Web lêem todo o tipo de input dos utilizadores
 - ▶ Bibliotecas, DLL's, Server code, Custom code, Exec
- Código C e C++ é vulnerável a buffer overflows
 - ▶ O input ultrapassa o tamanho do buffer e escreve por cima do stack
 - ▶ Pode ser usado para executar código arbitrário
- Pontos-chave
 - ▶ Evitar usar C ou C++ (em aplicações web)
 - ▶ Ter cuidado ao usar buffers e a escrever buffers
 - ▶ Usar bibliotecas de manipulação segura de strings

6. Falhas de Injecção de comandos

- As aplicações Web envolvem múltiplos componentes/interpretadores
 - ▶ Chamadas ao SO, BD's SQL, Sistemas de Templates
- Código Malicioso
 - ▶ Enviado num pedido HTTP
 - ▶ Extraído pela aplicação Web
 - ▶ Passado ao interpretador, executado em nome da aplicação Web
- Pontos-chave
 - ▶ Usar com cuidado extremo a chamada interpretadores
 - ▶ Usar interfaces limitados sempre que possível (PreparedStatement)
 - ▶ Verificar os valores de retorno

7. Problemas com a Gestão de Erros

- Os erros ocorrem em aplicações Web em qualquer momento
 - ▶ Falta de memória, demasiados utilizadores, timeout, falhas na base de dados, ...
 - ▶ Falhas de Autenticação, Falhas de controlo de acesso, Entradas erradas
- Como responder?
 - ▶ É preciso dizer ao utilizador o que aconteceu (sem dar pistas ao atacante)
 - ▶ É preciso registar a mensagem de erro (log)
 - ▶ Logout, email, pager, apagar cartão de crédito, etc...
- Pontos-chave:
 - ▶ Ter a certeza que as mensagens de erro não mostram toda a informação
 - ▶ Desenhar a forma de lidar com os error
 - ▶ Falhar em segurança
 - ▶ Configurar o servidor

8. Uso inseguro da Criptografia

- Usar a criptografia para armazenar informação sensível
 - ▶ Os algoritmos são fáceis de usar. Integrá-los é que é complicado
- Pontos-chave
 - ▶ Nem pense em inventar um novo algoritmo
 - ▶ Seja muito cuidadoso no armazenamento de chaves, certificados e passwords
 - ▶ Re-pense se precisa mesmo de armazenar a informação
 - ▶ Não armazene passwords – use funções de resumo (hash) tais como o MD5, SHA-1 ou SHA-256
- O “master secret” pode estar dividido em duas localizações diferentes e construído sempre que precisa de ser usado
 - ▶ Ficheiros de Configuração, servidores externos, escondidos no código

9. Falhas de Administração Remota

- Muitos sites permitem administração remota
 - ▶ Muito difícil, com interfaces escondidas
 - ▶ Difíceis de proteger
- Pontos-chave
 - ▶ Eliminar toda a administração através da Internet (se possível)
 - ▶ Separar a aplicação de administração da aplicação principal
 - ▶ Limite o que se pode fazer remotamente
- Considere a utilização de autenticação forte
 - ▶ Smart-card ou token

10. Falhas na configuração de servidores Web e Aplicacionais

- Todos os servidores web e aplicativos possuem múltiplas configurações relacionadas com segurança
 - ▶ Contas por defeito e passwords
 - ▶ Mensagens de erro com demasiados detalhes
 - ▶ Má configuração de SSL, certificados usados por defeito, certificados auto-gerados e auto-assinados
 - ▶ Serviços de administração não-utilizados
- Pontos-chave:
 - ▶ Estar atento aos patches (Code Red, Slammer)
 - ▶ Usar ferramentas de scanning (Nikto, Nessus)
 - ▶ Configurar os servidores a pensar em segurança (harden!)

Um Plano de Saúde simples

■ Treino e Formação

- ▶ Ler o relatório do Top Ten!
- ▶ Formar os programadores em segurança aplicacional para web
- ▶ User a ferramenta OWASP WebGoat para aprender como as vulnerabilidades actuam

■ Políticas

- ▶ Escrever as regras de segurança da aplicação

■ Revisões

- ▶ Efectuar revisões de código e efectuar testes de penetração com frequência

Às armas... às armas ;-)

■ Clientes

- ▶ Exigir aplicações web que não padeçam destes 10 problemas

■ Programadores

- ▶ Assumir a responsabilidade de tornar o código seguro

■ Organizações de Desenvolvimento de Software

- ▶ Garantir que as aplicações web não sofrem destes 10 problemas de segurança

■ Professores

- ▶ Parem de ensinar a programar de forma "insegura"

■ Gestores de Projecto

- ▶ Separar o orçamento de segurança entre redes e aplicação
- ▶ Tornar a segurança como parte da análise de desempenho do programador