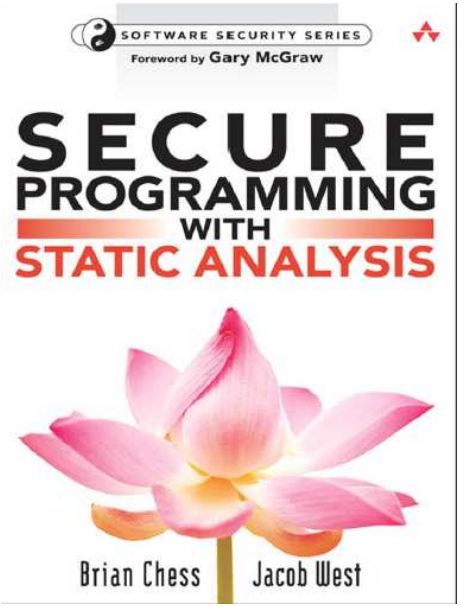
Programação Segura utilizando Análise Estática

Philippe Sevestre

Consultor Associado – LeadComm Applications & Database Security

Créditos

- **Conteúdo original: Brian Chess**
- **Versão original disponível para download:**
  - http://www.infoq.com/presentations/secure-programming-static-analysis



Agenda

- Cenário
- Problemas
- Soluções
- Análise Estática
- Adoção
- Conclusão

Cenário

- **A sociedade depende cada vez mais de sistemas**
  - Onipresentes
  - Conectados
  - Confiáveis
- **Conseqüência → Sistemas mais complexos**
- **Complexidade → Situações inesperadas**

Problemas

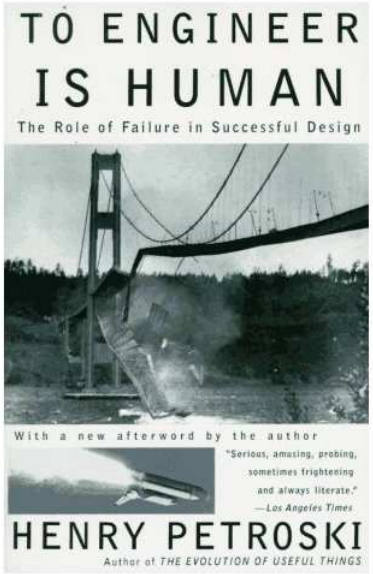
- **Não há uma fronteira claramente definida**
  - Decisões aparentemente não relacionadas à segurança possuem implicações
  - Pequenos problemas podem ter grandes conseqüências
- **Mesmo especialistas cometem erros primários**
  - Desenvolvedores tendem a repetir os mesmos erros em relação à segurança
- **É preciso fazer com que não-especialistas implementem segurança de forma correta**

Problemas

"Sucesso é antever problemas"

Henry Petroski

"Melhor prevenir que remediar"



Problemas

- **Falhas de segurança não-funcionais**
  - Erros Genéricos
    - Validação de dados de entrada
    - Buffer Overflow
    - Tratamento de erros e exceções
    - Manter a privacidade das informações
- **Variedades Comuns de Software**
  - Aplicações Web
  - Serviços acessíveis via rede/SOA
  - Aplicativos com acesso privilegiado

Problemas

- **Exemplo: Buffer Overflow**
  - Exemplo de código do MSDN para a função DirSpec (até a pouco tempo...)

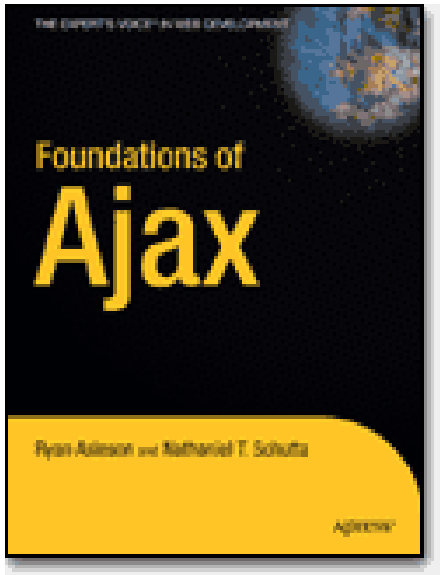
```
int main (int argc, char **argv)
{
    ....
    char DirSpec[MAX_PATH+1];
    printf("Diretório destino = %s.\n",argv[1]);
    strncpy(DirSpec,argv[1],strlen(argv[1])+1);
    ...
}
```

Problemas

- **Cross-Site Scripting**

```
<c:if test="${param.sayHello}">
  Hello ${param.name} !
</c:if>
```

- ***"Nunca foi nossa intenção que o código que está ali fosse utilizado em produção"***
  Ryan Asleson



Problemas

- **Estas vulnerabilidades são mais semelhantes do que aparentam**
  - Indicam uma falta de visão em relação ao problema de segurança
- **Explorar estas vulnerabilidades é cada vez mais fácil**

Soluções (Erradas)

Mais Esforço

- Desenvolvedores capacitados e esforçados
- Pedir que não cometam os mesmos erros

- Nem todos serão especialistas em segurança
- Implementar segurança de forma correta requer feedback

Deixar para Depois

- Codifique como sempre fez
- Deixe o problema com o firewall

- Paredes mais grossas não resolvem se vc. precisa de janelas e portas
- Time de segurança vira gargalo

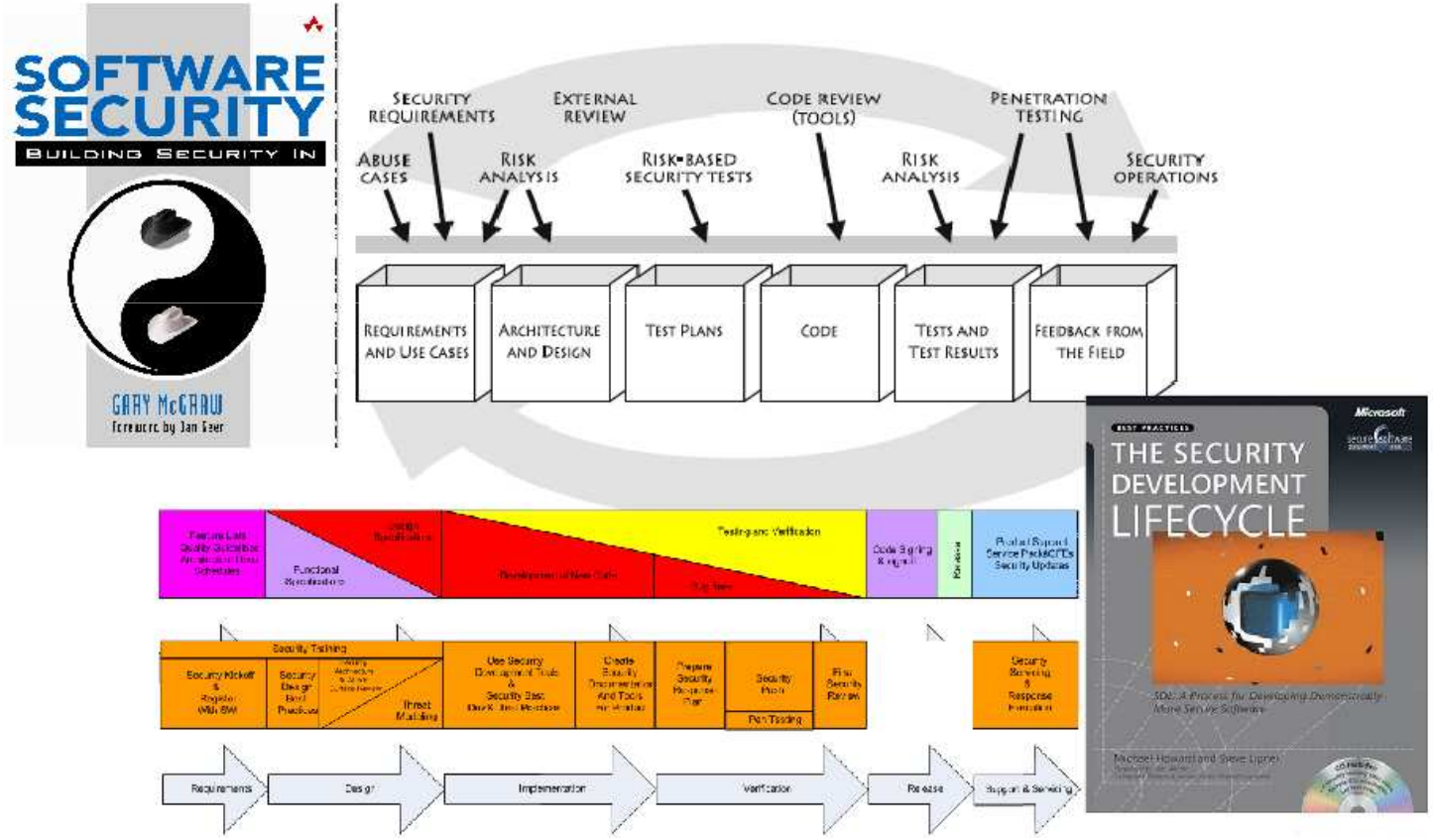
Mais testes

- PenTest na versão final
- Ataque os problemas encontrados

- PenTest é bom para evidenciar o problema
- Corrigir problemas não torna o sistema seguro

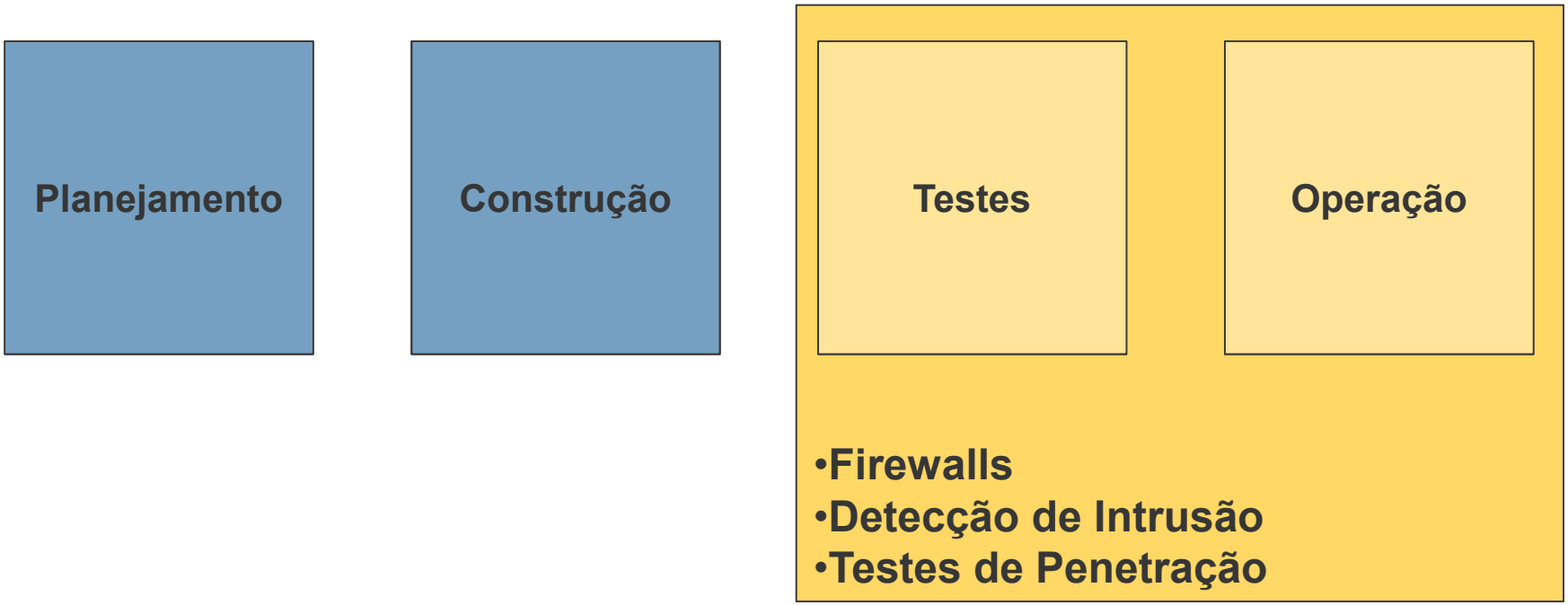
Soluções - Correta

- **Segurança no ciclo de vida**



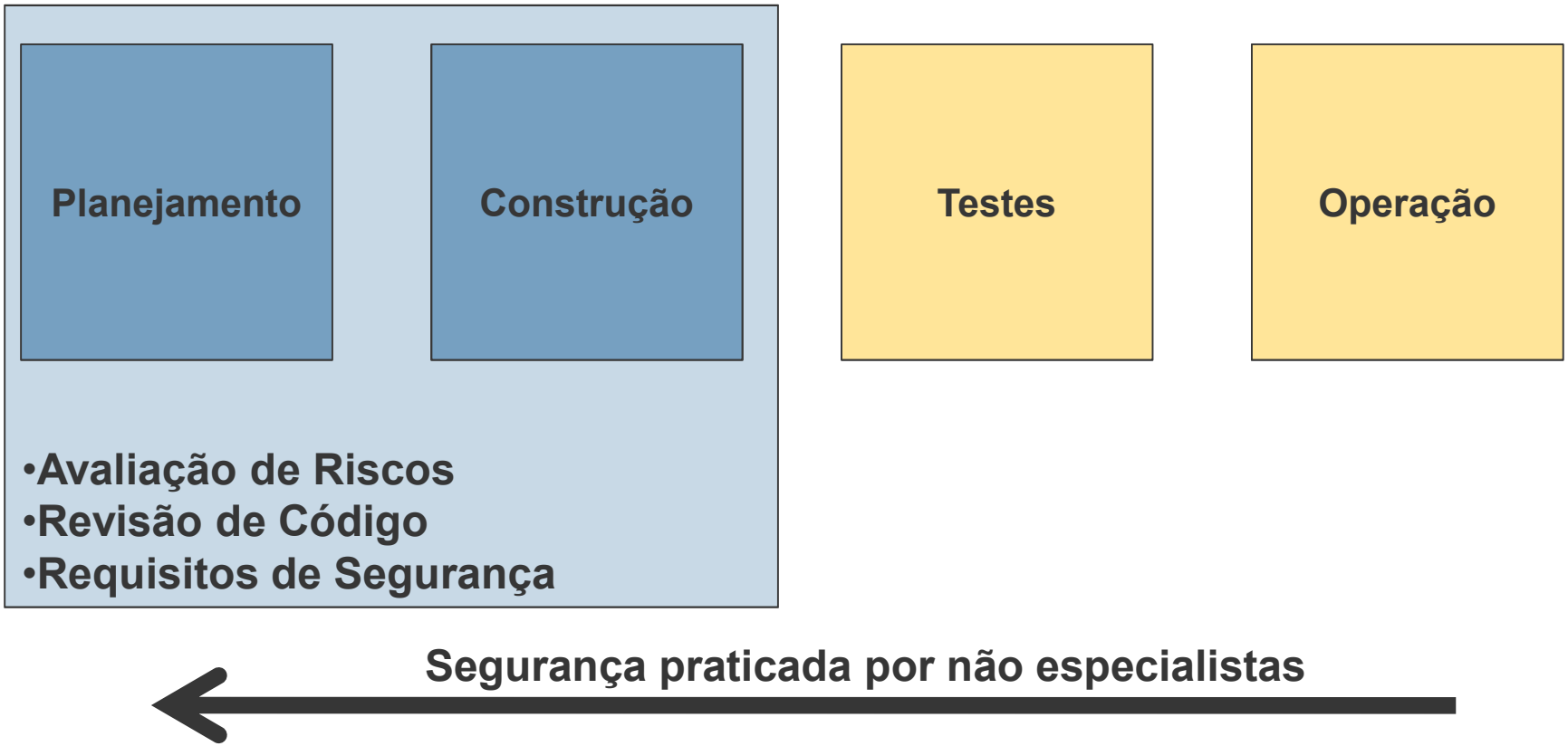
Soluções

- Segurança no ciclo de vida - Hoje



Soluções

- Segurança no ciclo de vida



Análise Estática

- Visão Geral
- Dissecando uma ferramenta de análise estática
- Análise estática na prática
- O que vem a seguir ?

Análise Estática: Definição

- Analisa o código *sem executá-lo*
- Capaz de contemplar um número bem maior de possibilidades do que um ser humano em testes convencionais
- Não possui conhecimento prévio do que o código deve(ria?) fazer
- Precisa de informações sobre o que se quer encontrar

Analise Estática

- **É uma ferramenta**
- **Não vai fazer o seviço sozinha**
  - É preciso utilizá-la de forma correta

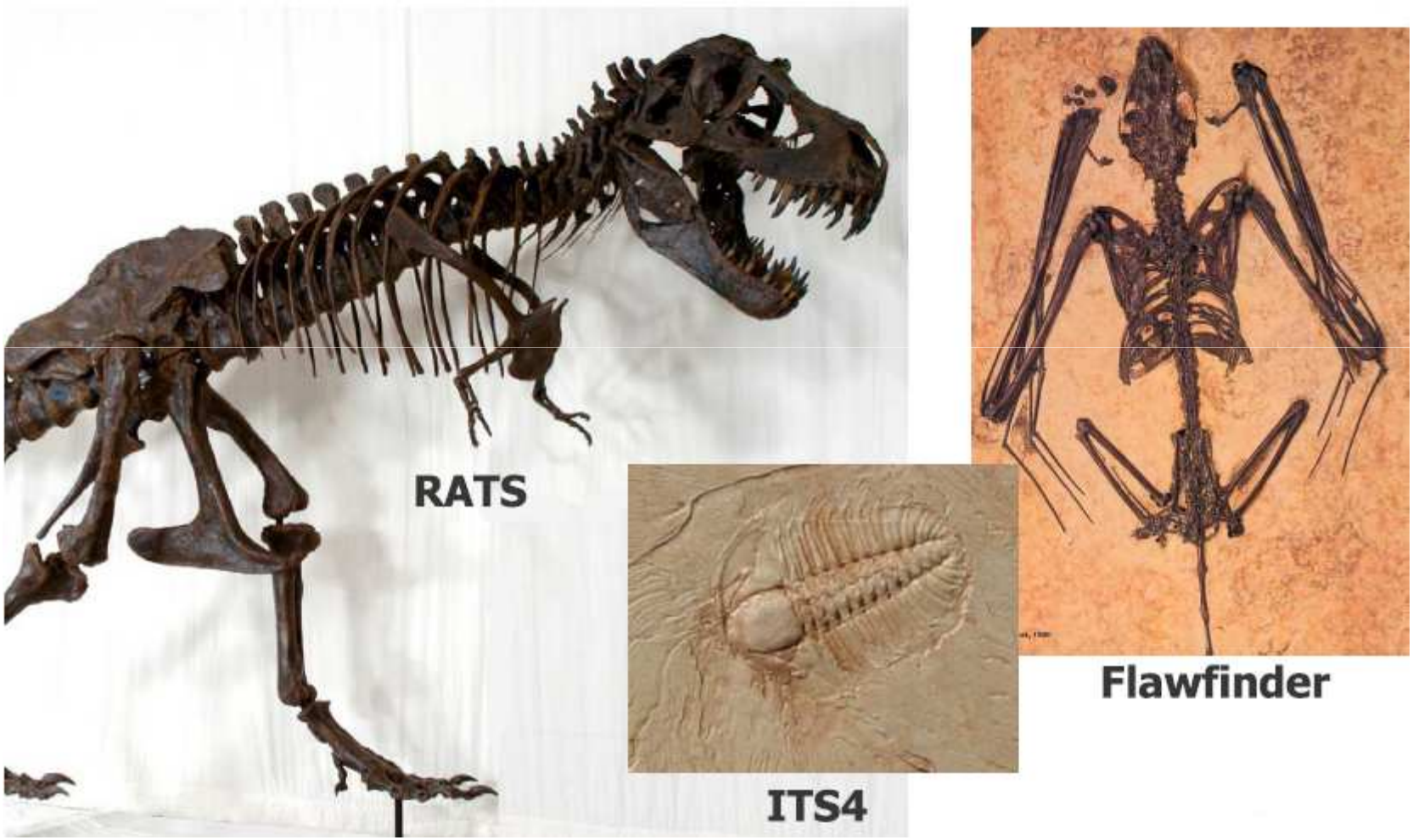
Análise Estática e suas faces

- Verificação de tipos
- Validação de regras/estilos de codificação
- Entendimento de programas
- Validação formal/Validação de propriedades
- Identificação de bugs
- Análise de segurança

Análise estática para segurança: motivadores

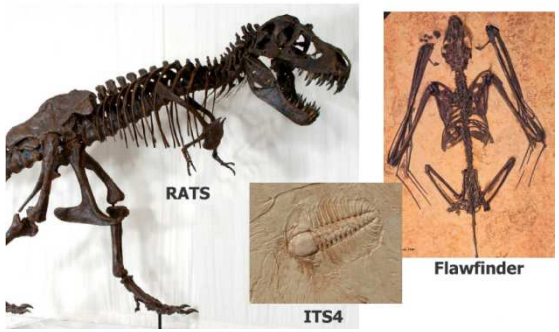
- Rápida se comparada a análise manual
- Rápida se comparada a um teste de execução
- Cobertura completa e consistente
- Aporta o conhecimento de segurança embutido na mesma para o processo
- Facilita o processo de revisão por não-especialistas em segurança

Análise Estática: Ferramentas Pré-Históricas



Análise Estática: Ferramentas Pré-Históricas

- **Versões especializas do *grep***
- **Pontos Positivos**
  - Apoio para especialistas de segurança
  - Centralização de informações sobre más práticas de codificação
- **Pontos Negativos**
  - Dificuldade de uso por não-especialistas



Análise Estática: Ferramentas avançadas

- **Ponto diferenciador importante: Priorização**

```
int main (int argc, char **argv)
{
    char buf1[1024];
    char buf2[1024];
    char *shortString = "message";

    strcpy(buf1,shortString); /* Bad Practice */
    strcpy(buf2,argv[0]); /* VULNERABILITY */
}
```

Análise Estática: O que ela NÃO encontra

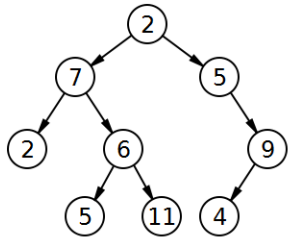
- **Erros estruturais na arquitetura**
  - Telescópio x Microscópio
- **Bugs que não forem procurados**
  - As categorias devem ser pré-definidas
- **Erros administrativos**
- **BIOS**

Análise Estática: Segurança != Qualidade

- **Ferramentas "Caça-Bug" focam em resultados em resultados de alto grau de certitude**
  - Bugs tem baixo custo unitário (são muitos !)
  - Incorporam grande base de bugs e práticas que levam a bugs
  - Falsos-positivos inviabilizam o uso da ferramenta
- **Ferramentas de segurança focam em resultados de *alto risco***
  - Requer intervenção manual
  - Falsos-negativos são o grande problema

Análise Estática: Como Funciona ?

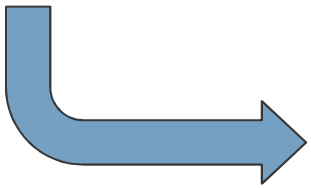
Código-fonte



Modelo

Regras

Análise



Resultados

Análise Estática: Atributos de um Analizador

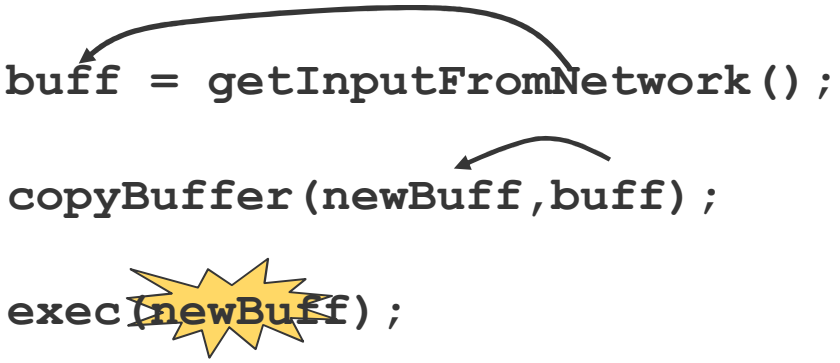
- **Algorítmos de Análise**
  - Utilização de técnicas apropriadas para identificar e priorizar apontamentos
- **Linguagens Suportadas**
  - Suporte às linguagens e dialetos relevantes
- **Capacidade**
  - Habilidade de tratar milhões de linhas de código
- **Base de Regras**
  - Modelagem de regras e propriedades de segurança
- **Gestão dos Resultados**
  - Permitir a revisão manual dos resultados
  - Priorização dos apontamentos
  - Seleção dos resultados a apresentar

Analise Estática: Construção do Modelo

- **Front-End similar ao de um compilador**
- **Suporte a uma linguagem**
  - Uma linguagem/compilador é simples
  - Multiplas combinações de linguagens/compiladores torna o problema bem mais complexo
- **Abordagem possível: análise estática do binário...**
  - O binário sempre está disponível
  - Não é necessário saber como se chegou a ele
  - Diminui o número de regras necessárias
- **... porem:**
  - Decompilação pode ser não trivial
  - A perda do contexto prejudica a análise
  - Remediação requer os fontes para apresentação dos resultados

Análise Estática: Técnicas de Análise

- **Propagação de atributos (Taint Propagation)**
  - Segue os caminhos possíveis que dados potenciamente contaminados podem tomar na execução de um aplicativo
  - Identifica pontos onde um atacante pode tirar proveito de uma função vulnerável



- **Várias abordagens disponíveis, nenhuma exclui interamente as demais**

Análise Estática: Regras

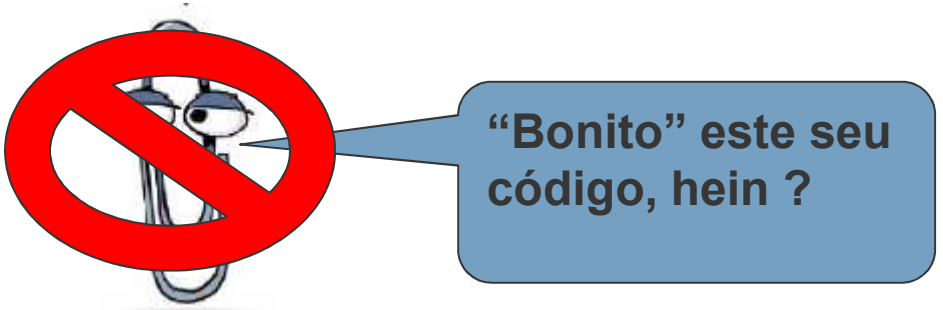
- **Especificam**
  - Propriedades de segurança
  - Comportamento de rotinas de biblioteca em relação a estas propriedades

```
buff = getInputFromNetwork();
copyBuffer(newBuff,buff);
exec(newBuff);
```

- **Regras utilizadas**
  - getInputFromNetwork() : pós-condição = resultado inseguro
  - copyBuffer(arg1,arg2): pós-condição = conteúdo de arg1 igual ao de arg2
  - exec(arg1): pré-condição = arg1 deve ser seguro

Análise Estática: Apresentação de Resultados

- **Deve convencer o desenvolvedor de que existe um problema no código**
- **Apresentação diferenciadas por audiência**
  - Auditor, caindo de paraquedas em uma base de milhões de LOCs
  - Programadores revisando o próprio código
  - Programadores revisando código de outros
- **A interface de apresentação é tão importante quanto a análise em si.**
- **Não mostrar resultados incorretos mais de uma vez**



Análise Estática: O que pode dar Errado

- **Falsos-Positivos**
  - Modelo incompleto/incorreto
  - Análise :conservadora
- **Falsos-Negativos**
  - Modelo incompleto/incorreto
  - Ausência de regras específicas
  - Análise “relaxada”

Análise Estática: Formas de Uso

- **Análise de Programas Finalizados**
  - Forma sofisticadas de PenTest
  - Qunatidade de resultados torna-os intratáveis
  - Ponto de partida para a maioria
  - Bom motivador

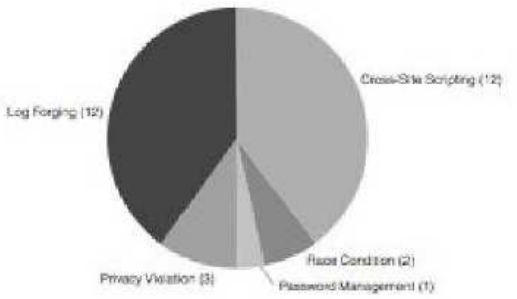
- **Análise incorporada à codificação**
  - Executada como parte do build
  - Diário/Semanal/Milestone
  - Correções feitas à medida que o código for criado

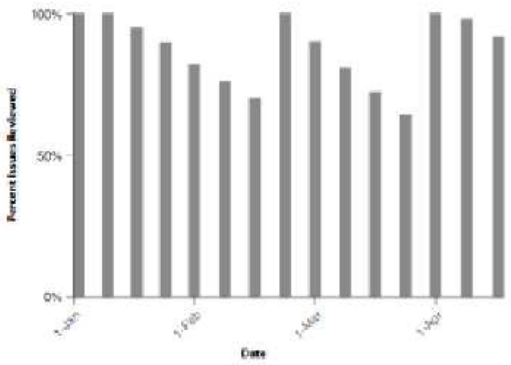
Análise Estática: Objeções

Objeção	Tradução
“Demora demais para rodar”	“Segurança não é problema meu”
“Encontra muitos falsos-positivos”	“Segurança não é problema meu”
“Não é adequado à nossa forma de trabalho”	“Segurança não é problema meu”

Análise Estática: Métricas

- **Densidade de defeitos → Densidade de Vulnerabilidades (??)**
  - Não é uma boa medida
- **Serve para responder algumas questões:**
  - Quais bugs ocorrem com mais freqüência ?
  - Qual o esforço ($$$) necessário para tornar meu aplicativo seguro





Análise Estática: Adoção

- **Requer mudança na cultura**
  - Mais do que apenas outra ferramenta
  - Porta-estandarte da segurança de software
  - Atenção: A ferramenta não irá resolver o problema por si só
- **Mantenha o foco**
  - As ferramentas são capazes de identificar diversos tipo de problema: **Iniba a maioria**
  - Foque nos problemas de entendimento mais fácil e relevantes
- **Treine logo que possível**
  - Treinamento em segurança de software é fundamental
  - Treinamento nas ferramentas ajudam a tornar o processo mais eficiente

Análise Estática: Adoção

- **Mensure os resultados**
  - Resultados encontrados pela ferramenta
  - Vulnerabilidades resolvidas
- **Deixe a ferramenta com a "sua cara"**
  - Invista em customização
  - Use a ferramenta para garantir a adoção de padrões internos de segurança
    - Conceba os padrões de codificação de forma a poderem ser verificados pela ferramenta
- **A primeira vez é a pior**
  - Custo é o dobro na primeira análise
  - Números típicos: 10% do tempo para segurança, 20% na primeira vez

Conclusão

- **Erros acontecem: esteja preparado**
- **Segurança faz parte do desenvolvimento**
- **Para auditores: A ferramenta torna viável e eficiente a análise de código**
- **Para programadores: A ferramenta aporta conhecimento em segurança**
- **Componentes críticos de uma boa solução:**
  - Algorítmos
  - Regras
  - Apresentação
  - Plano de adoção

Dúvidas ?

Obrigado !