GTER 38 | GTS 24

ACME! Cybersecurity Research

SEGURANÇA COMO PARTE INTEGRAL NO CICLO DE DESENVOLVIMENTO DE SOFTWARE

Nome: Vinícius Oliveira Ferreira

Email: viniciusoliveira@acmesecurity.org

PGP Key ID: 0xE7960563

Agenda

- O atual cenário da segurança
- Por quê ?
- Técnicas e métodos de segurança nas fases de desenvolvimento.
- Conclusões.

Agenda

- O atual cenário da segurança
- Por quê ?
- Técnicas e métodos de segurança nas fases de desenvolvimento.
- Conclusões.

Os dias são maus …

Os dias são maus …

- Inquestionavelmente, vivemos hoje uma **nova ordem cibernética mundial;**

- Entretanto, nem todos entendem as implicações práticas deste novo cenário;

Mercado de Vulnerabilidades Zero-day

- O mercado de vulnerabilidades zero-day encontra-se em franca expansão;

- Motivado principalmente pelo surgimento de um novo mercado - "The Gray Market" [1].

Mercado de Vulnerabilidades Zero-day

- White Market: Programas *bug bounty* oferecidos pelos fornecedores de *software* ou terceiros (Google, Facebook, Zero Day Initiative e etc).
  - Recompensas no geral variam de $500 a $5,000, mais reconhecimento pela comunidade.

- Black Market: Vulnerabilidades vendidas para organizações criminosas.

Mercado de Vulnerabilidades Zero-day

- **Gray Market:** Vulnerabilidades vendidas a compradores legítimos: governos, empresas de espionagem e monitoramento e etc.

- Os preços geralmente começam em $20,000, podendo chegar a $200,000.

- Empresas atuantes: VUPEN (França), ReVuln (Malta), Netragard, Endgame Systems, and Exodus Intelligence (US).

Mercado de Vulnerabilidades Zero-day

- **Dados - Gray Market [2]:**
  - Em **qualquer dia** entre 2010 e 2012, grupos privilegiados tiveram acesso a pelo menos **58 novas vulnerabilidades** envolvendo produtos da Microsoft, Apple, Oracle, ou Adobe.
  - Algumas empresas vendem planos de assinaturas de exploits para vulnerabilidades zero-day. (Endgame Systems oferece 25 exploits por ano a uma assinatura de 2.5 milhões).
  - Estima-se que todos as empresas juntas podem negociar pelo menos 100 novos exploits por ano.

Mercado de Vulnerabilidades Zero-day

- **Recomendações – NSS Labs:**
  - Profissionais de segurança devem se conscientizarem sobre os perigos trazidos por este novo cenário;
  - Como é difícil prevenir, empresas deveriam implantar ferramentas e processos para rápida detecção e mitigação de uma violação;
  - Fornecedores de *software* deveriam investir mais em programas *Bug Bounty*.

Mercado de Vu[illegible]dades Zero-day

- **Recome[illegible] NSS Labs:**
  - Profis[illegible]s de [illegible]ca devem se consc[illegible]arem so[illegible]erigos trazido[illegible] este novo [illegible]rio;
  - Como [illegible]cil prevenir, emp[illegible]ever implan[illegible]rramentas e proce[illegible]bida detecçã[illegible]igação de uma vi[illegible]
  - Fornecedores de[illegible]m investir mais em programas *Bug Bounty*.

Vazamento de armas Cibernéticas

Vazamento de armas Cibernéticas

Cybercriminals read the news as well - Roel Schouwenberg (Kaspersky).

Agenda

- O atual cenário da segurança
- Por quê ?
- Técnicas e métodos de segurança nas fases de desenvolvimento.
- Conclusões.

Segurança como parte integral

- **Truísmos em segurança:**
  - Não existe segurança absoluta;
  - Segurança é sempre uma questão econômica;
  - Deve-se manter o nível de todas as suas defesas no mesmo nível;
  - Um ataque não passa através da segurança, mas em torno dela;
  - A boa segurança é feita em camadas;
  - Nunca confiar em segurança por obscuridade;
  - **Segurança deve ser uma parte integral do desenho original do sistema;**
  - A boa segurança é simples;
  - Uma pessoa ou processo não deve ter privilégios maiores do que aqueles necessários para executar sua função;
  - Um programa ou protocolo é inseguro até que se prove o contrário;
  - Segurança é um acordo de conveniência;
  - Não subestime o valor de suas posses.

Por quê?

- Deixar a segurança para depois é como tratar qualquer problema de forma reativa.
  - E conhecemos muito bem os problemas disso;
  - Contruímos *softwares* cheios de vulnerabilidades;
    - Para então termos que apagar o incêndio depois.

O caso *ShellShock*

- Uma vulnerabilidade encontrada no *bash*, amplamente utilizado por sistemas *unix-based* (Ubuntu, Mac, e etc.).
  - Uma falha na forma como o *bash* avaliava o valor de algumas variáveis de ambiente especialmente formadas, o que permitia a execução de código arbitrário por parte de um atacante.
  - O *National Vulnerability Database* atribuiu a pontuação máxima (10/10) quanto ao risco desta vulnerabilidade [3].

O caso *ShellShock*

- O caso *ShellShock* (Linha do tempo)
  - **01/09/1989**: versão com a vulnerabilidade *shellshock* do *bash* é lançada.
  - **12/09/2014**: Stéphane Chazelas (Red-Hat) reporta a vulnerabilidade para Chet Ramey, desenvolvedor chefe do *bash*.
  - **16/09/2014**: Chet Ramey "finaliza" a criação dos *patches* de correção para todas as versões do *bash*, antes da vulnerabilidade ser tornar pública.
  - **24/09/2014**: Vulnerabilidade é divulgada ao público, juntamente com as correções de Chet Ramey (Official Patch 25 for *bash*).

O caso *ShellShock*

O caso *ShellShock* (Linha do tempo)

- **24/09/2014**: Tavis Ormandy (Google) consegue subverter o *patch* de correção, lançando uma nova versão da vulnerabilidade *Shellshock*.

- **25/09/2014**: Mídia começa divulgar os primeiros ataques.

O caso *ShellShock*

- O caso *ShellShock* (Linha do tempo)
  - **26/09/2014**: Rede-HAT, CentOS, Fedora, Debian, e Ubuntu lançam correções independentes.
  - **26/09/2014:** Chet Ramey lança nova correção oficial (Official Patch 26 for bash), mas só é suficiente para alguns casos específicos e novamente não resolveu o problema.
  - **27/09/2014:** Nova correção oficial (Official Patch 27 for bash).
  - **29/09/2014:** Apple lança correção independente para seus produtos.

O caso *ShellShock*

O caso *ShellShock* (Linha do tempo)

- **01/10/2014:** Nova correção oficial (*Official Patch 28 for bash*).
- **02/10/2014:** Nova correção oficial (*Official Patch 29 for bash*).
- **05/10/2014:** Nova correção oficial (*Official Patch 30 for bash*).

Por quê?

- Incêndios são difíceis de se apagar.

Por quê?

- Incêndios causam prejuízos.

  - David Rice, autor do livro *Geekonomics: The Real Cost of Insecure Software*, estimou que os custos anuais com falhas de segurança de *software* ficam em torno dos **180 bilhões** de dólares [4].

  - Custo médio de uma violação de dados: **3,5 milhões** de dólares [5].

  - Existem outros custos que são imensuráveis, como a perca de reputação e da confiança dos clientes.

Agenda

- O atual cenário da segurança
- Por quê ?
- Técnicas e métodos de segurança nas fases de desenvolvimento.
- Conclusões.

Segurança na Fase de Levantamento de Requisitos

- Envolva o cliente com os aspectos de segurança.
  - Isso o ajudará a compreender os possíveis riscos;
  - Permitirá a elicitação de requisitos especiais de proteção ao *software*;
  - Utilize questionários ou *checklists;*
  - Fuja dos jargões técnicos.

Segurança na Fase de Levantamento de Requisitos

- Identifique os requisitos de privacidade e cumprimento de obrigações legais.
  - Esteja atento às leis e as regulamentações locais;
  - No nosso caso: **Marco Civil da Internet**!

Segurança na Fase de Levantamento de Requisitos

- Tenha em mente os aspectos de **confidencialidade, integridade, disponibilidade e autenticidade.**
  - Faça suas perguntas considerando estes conceitos:
    - O acesso às informações deverá ser aberto ou restrito ?
    - Quais são os fatores que permitem alterações autorizadas, e quem está autorizado a fazê-los ?

Segurança na Fase de Levantamento de Requisitos

- Qual a tolerância quanto a disponibilidade do sistema ?
- Considere também a auditabilidade do *software*.
- A classificação dos dados (*Data classification*) é um grande trunfo para a definição dos requisitos corretos para proteção das informações.

Segurança na Fase de Levantamento de Requisitos

Figura extraída de [6]

Segurança na Fase de Projeto

- Modele casos de abuso (*Misuse Cases*)
  - Não é interessante que somente as funcionalidades estejam presentes nos casos de uso;
  - É necessário que os possíveis *abusos* também estejam retratados;
  - Assim pode-se analisar as ameaças que cada funcionalidade pode acrescentar ao sistema.

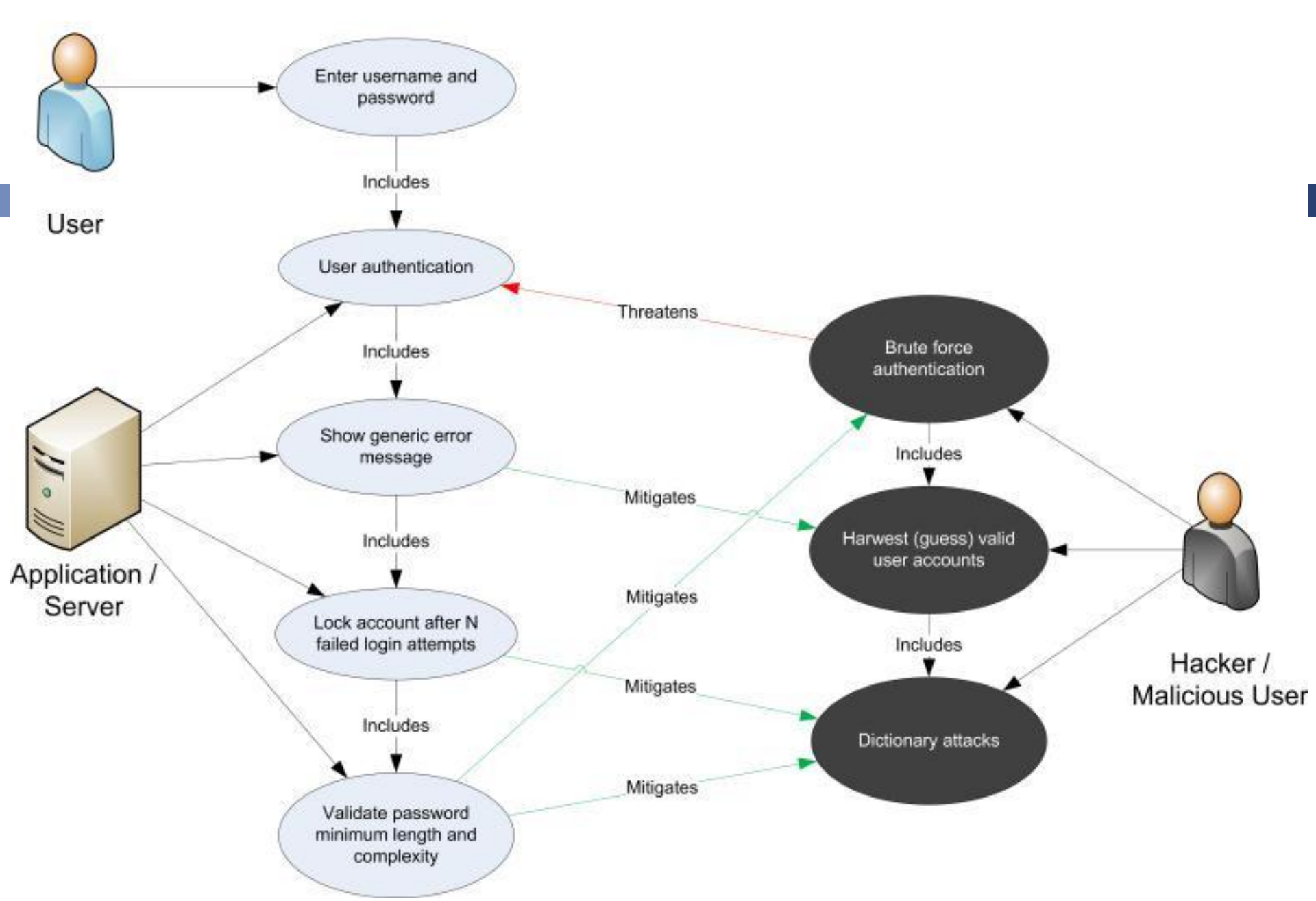


Figura extraída de [7].

Segurança na Fase de Projeto

- Desenvolva um Modelo de Ameaças (*Threat Modeling*)
  - Modelo de ameaças é uma representação estruturada das possíveis ameaças que podem envolver um sistema de *software*;
  - O objetivo é a identificação das principais ameaças que o sistema possa estar exposto;
  - Assim é possível a elaboração de um plano de contramedidas para prevenir ou mitigar os efeitos uma dada ameaça.

Segurança na Fase de Projeto

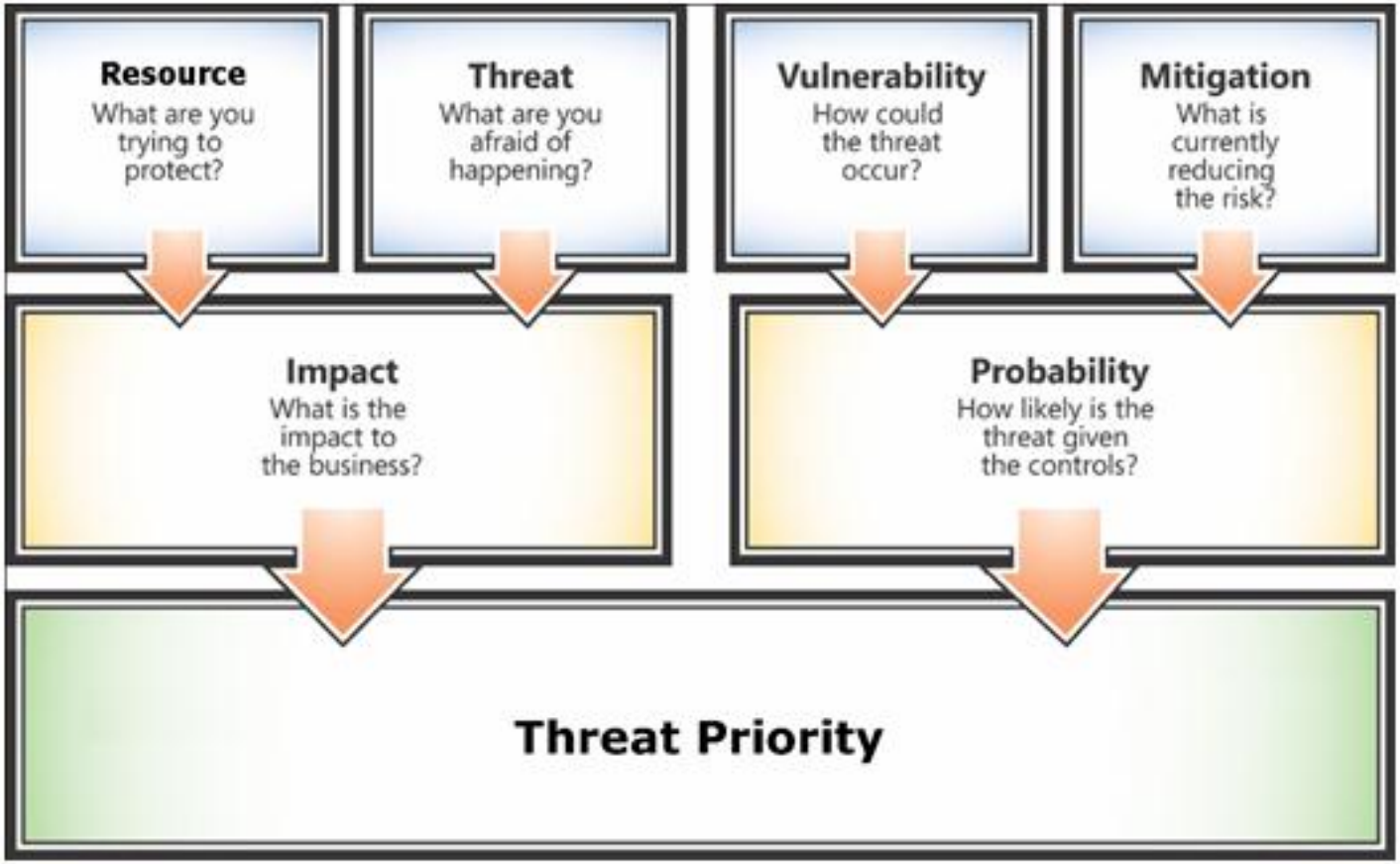
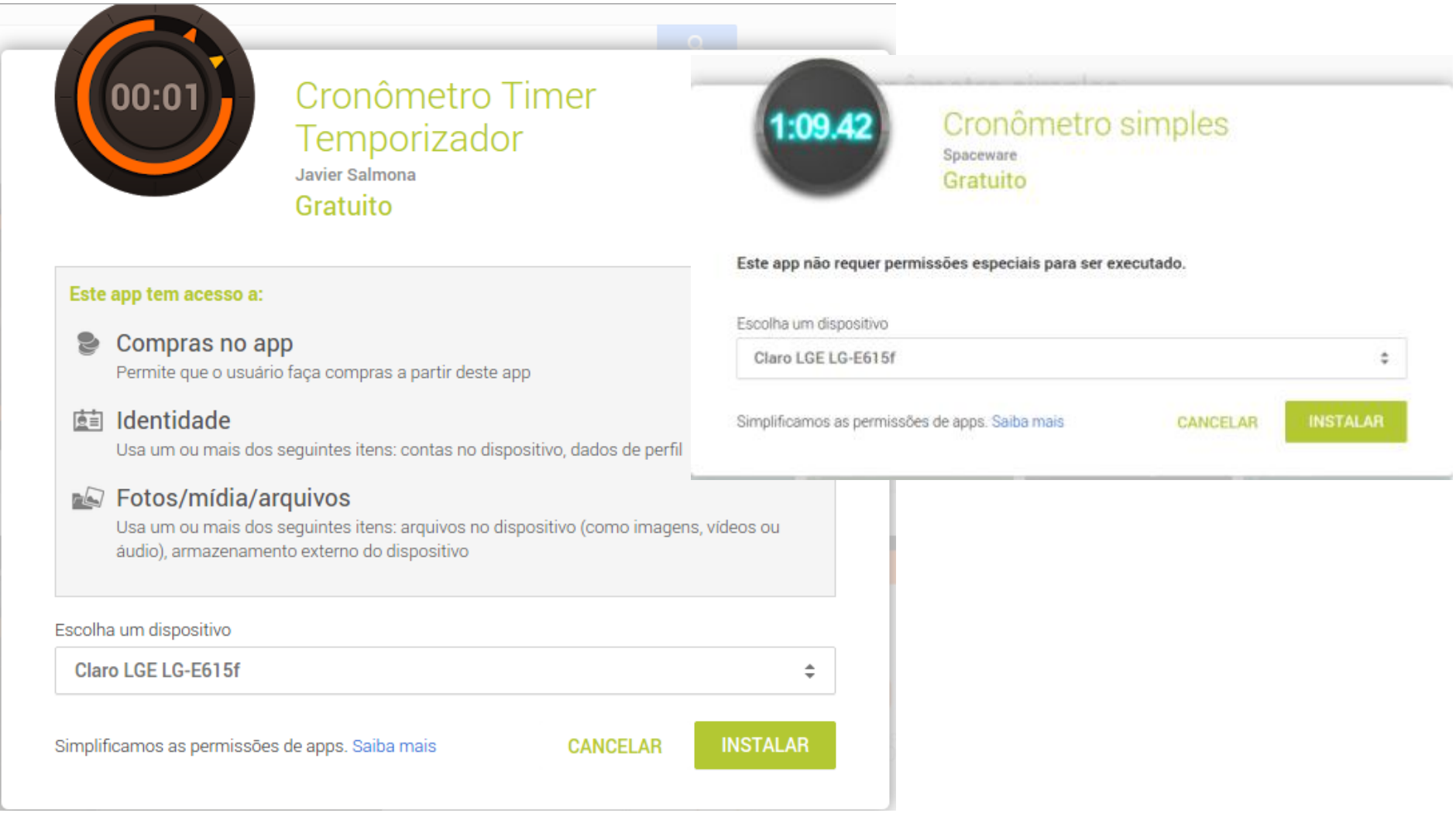


Figura extraída de [8]

Segurança na Fase de Desenvolvimento

- Desenvolva *software* seguro por padrão.
  - Os princípios de *Defesa em Profundidade* e *Privilégio Mínimo* devem nortear toda a fase de desenvolvimento do projeto.

Segurança na Fase de Desenvolvimento



Segurança na Fase de Desenvolvimento

- Escreva código seguro.
  - Um princípio básico para se ter *software* seguro.
  - Um bom código elimina as ameaças mais comumente utilizadas nos ataques: Buffer-Overflow, ataques de injeção, dentre outros
  - Não concatene mais strings em suas consultas SQL!

SELECT * FROM usuarios WHERE login = '" + user.getLogin() + "'

Segurança na Fase de Implantação

- Realize uma instalação segura.
  - Um *software* desenvolvido nos princípios do privilégio mínimo pode apresentar erros no momento da instalção.
  - Tornar menos rígida a política de segurança do ambiente de produção ou aumentar os privilégios do sistema são as soluções mais fáceis. **Porém, inaceitáveis em termos de segurança.**

Segurança na Fase de Implantação

- Realize a avaliação de vulnerabilidades e testes de penetração.
  - Ateste a segurança do *software* após sua implantação.
  - Na avaliação de vulnerabilidades busca-se identificar vulnerabilidades que possam estar contidas no *software*, para tal pode-se utilizar os testes de penetração.

Segurança na Fase de Manutenção

- Controle as mudanças.
  - Controle as alterações no *software* e também em sua configuração. Ferramentas de versionamento são um grande trunfo!
  - Acesso ao código de produção deve ser explicitamente autorizado.
  - Todas as atualizações devem ser desenvolvidas nos mesmos moldes de segurança em que o *software* foi construído.

Segurança na Fase de Manutenção

- Tenha um bom plano de resposta a incidentes.
  - Ações coordenadas são sempre mais efetivas e eficientes.
  - Um bom plano pode reduzir não somente os danos, mas também a publicidade negativa.

Segurança na Fase de Manutenção

- Um plano de resposta ao incidente pode ser dividido em quatro fases:
  - Ação imediata para interromper ou minimizar o incidente;
  - Investigação do Incidente;
  - Restauração dos recursos afetados;
  - Reportando o indicente aos canais apropriados.

Segurança na Fase de Manutenção

- Registre e gerencie *Logs* do sistema.
  - O registro de *Logs* é essencial para que se tenha um registro das mudanças no sistema;
  - O que registrar e por quanto tempo, depende da criticidade das alterações feitas;
  - Registre a data, hora, o usuário ou processo que realizou a alteração;
  - Garanta a segurança de seus *Logs*, eles podem conter informações sensíveis e confidenciais.

Segurança na Fase de Manutenção

Por fim, continue monitorando, avaliando e testando !!!

Agenda

- O atual cenário da segurança
- Por quê ?
- Técnicas e métodos de segurança nas fases de desenvolvimento.
- Conclusões.

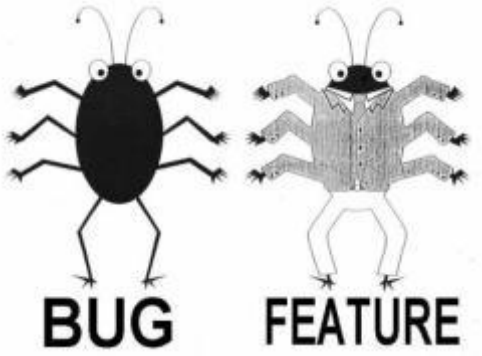
Conclusões

- A tendência é que as ameaças continuem evoluindo. Nossos *Softwares* acompanharão essa evolução?

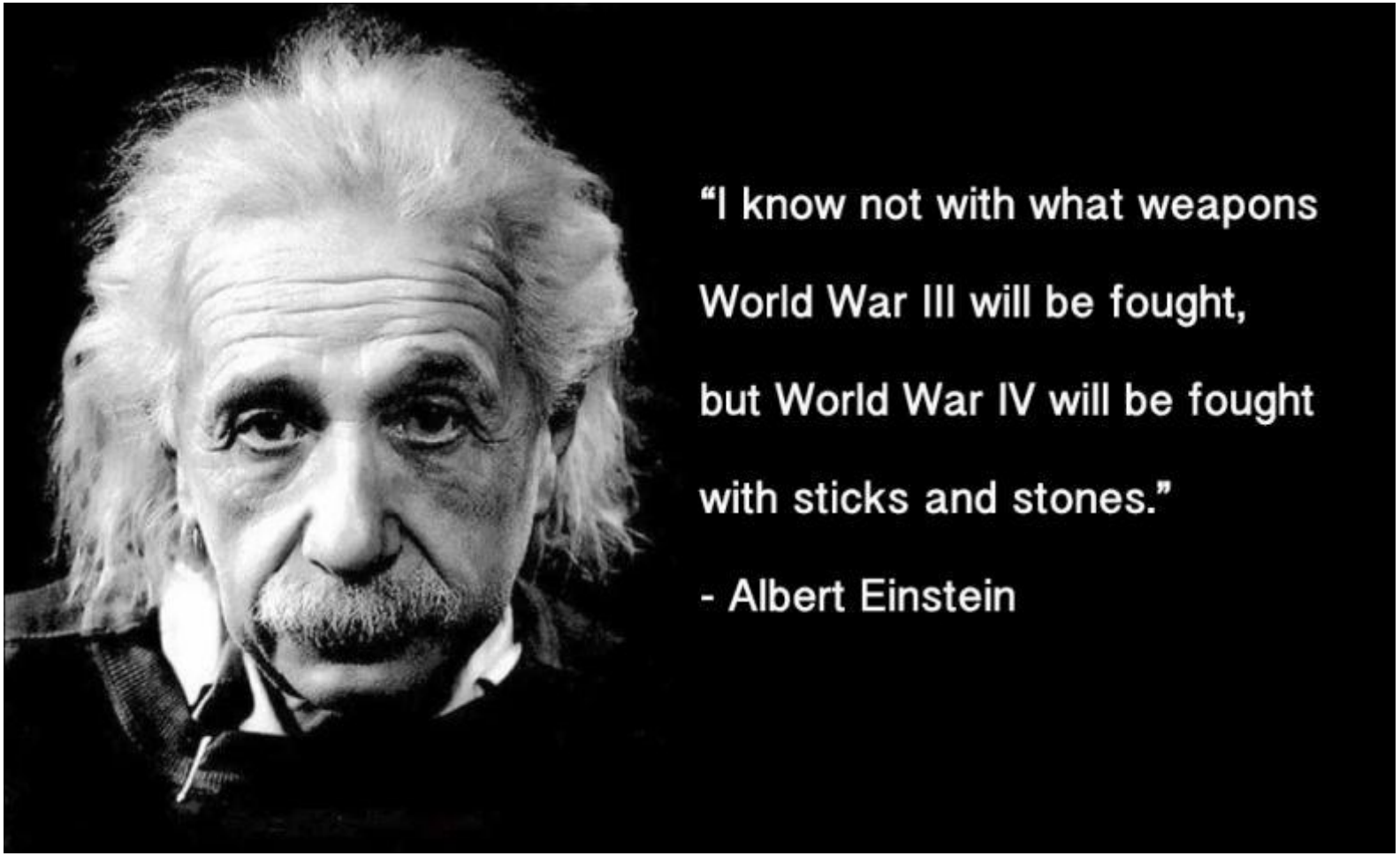
- Continuaremos a tratar a segurança de forma reativa, apagando incêndio?

Conclusões

- Dificuldades:
  - Cursos de TI não abordam segurança;
  - A engenharia de *software* não aborda segurança;
  - O usuários não se importam com segurança;
  - Funcionalidades ainda valem mais do que segurança.



Conclusões


"I know not with what weapons
World War III will be fought,
but World War IV will be fought
with sticks and stones."
- Albert Einstein

Referências Bibliográficas

[1] Lemos, R. Market For Vulnerability Information Grows. Information Security Magazine. 2012.

[2] Frei, S. The Known Unknowns - Empirical Analysis Of Publicly Unknown Security Vulnerabilities. NSS Labs. 2013.

[3] National Vulnerability Database. Vulnerability Summary for CVE-2014-7169. Disponível em: <https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-7169>. Acesso em: 23 de outubro de 2014.

[4] GREENBERG, A. A Tax On Buggy Software. Forbes. 2008. Disponível em: <http://www.forbes.com/2008/06/26/rice-cyber-security-tech-security-cx_ag_0626rice.html>. Acesso em: 23 de outubro de 2014.

Referências Bibliográficas

[5] Ponemon Institute, sponsored by IBM. 2014 Cost of Data Breach Study. 2014. Disponível em: <http://www-935.ibm.com/services/us/en/it-services/security-services/cost-of-data-breach/>. Acesso em: 23 de outubro de 2014.

[6] Berkeley Security. Data Classification. Disponível em: <https://security.berkeley.edu/tags/data-classification>. Acesso em: 23 de outubro de 2014.

[7] OWASP. Application Threat Modeling. Disponível em: <https://www.owasp.org/index.php/Application_Threat_Modeling>. Acesso em: 23 de outubro de 2014.

[8] Microsoft - SolutionAccelerators. IT Infrastructure Threat Modeling Guide. 2009.