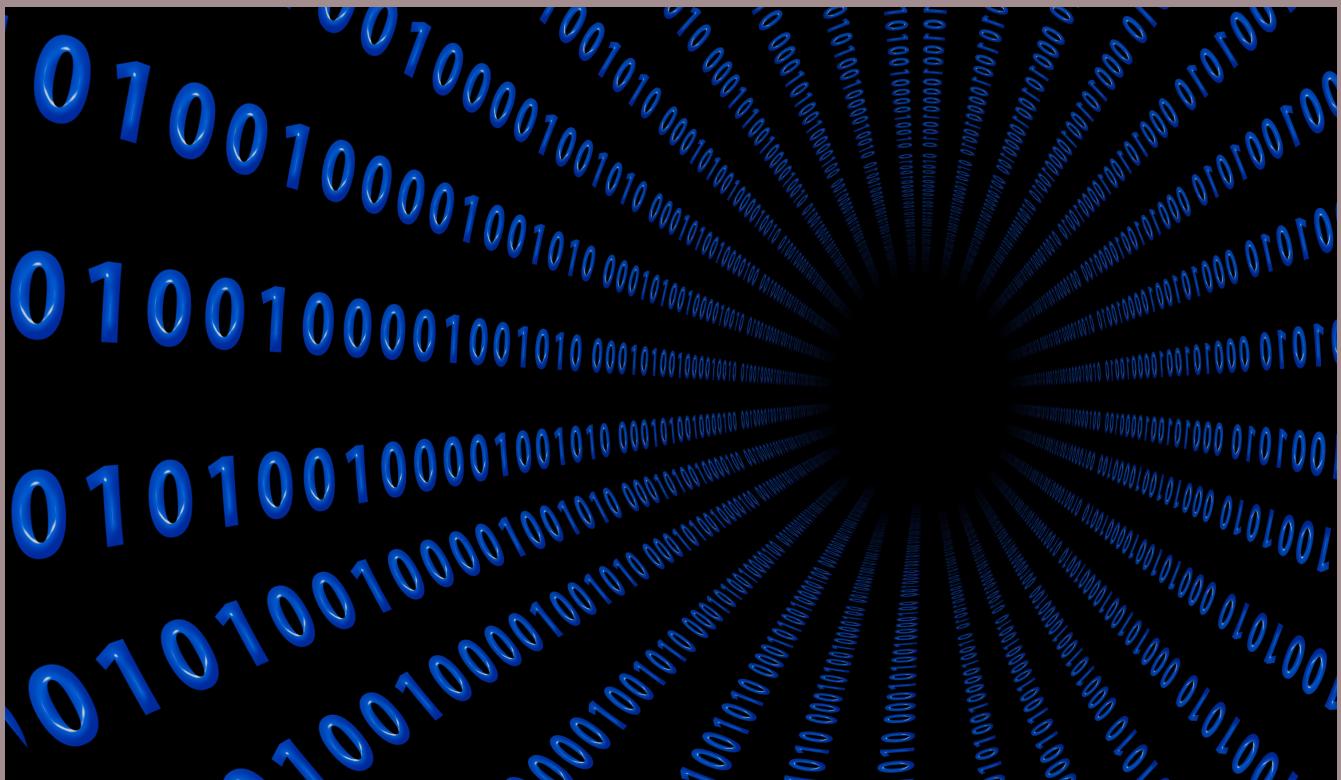




EDIÇÃO ESPECIAL

# Info Hacker

RESUMO QUINZENAL OFICIAL DA ACADEMIA HACKER

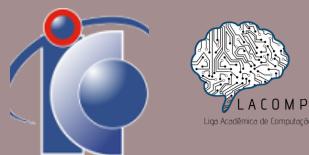


## 01: XSS - Cross Site Scripting

## 08: Três cenários de uso da Inteligência Artificial na Segurança da Informação

## 10: Smalltalk na Programação

## 13: Novos Membros da Academia Hacker



# XSS - Cross Site Scripting

por Lucas Buarque

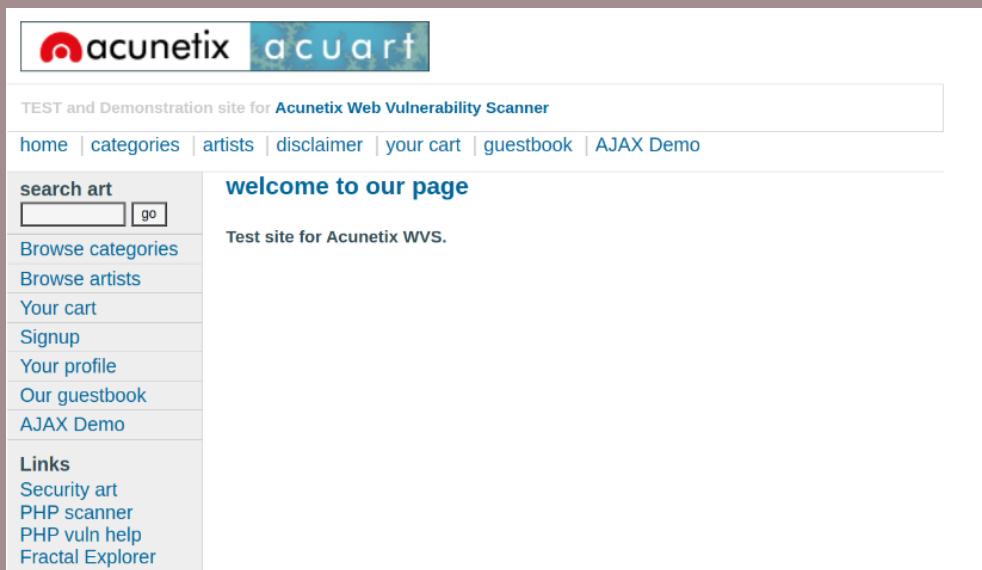
Cross Site Scripting (XSS) é um tipo de vulnerabilidade web do tipo injeção, a qual permite que trechos de código externos sejam injetados nas páginas da aplicação e executados pelo navegador do cliente. O ataque ocorre quando um indivíduo malicioso consegue manipular alguma entrada de dados do site com o objetivo de executar códigos Javascript no navegador de um usuário.

Essa falha é considerada grave e está na posição três do ranqueamento de vulnerabilidades web mais comuns de 2021 do OWASP, o qual é um projeto aberto que tem como objetivo divulgar conhecimentos sobre falhas web e conscientizar organizações a desenvolverem aplicações mais seguras. A falha de Cross Site Scripting, geralmente, é dividida em três categorias: XSS refletido (reflected), armazenado (stored) e DOM Based. Vamos focar apenas no refletido e armazenado que são os mais comuns.

## Reflected XSS

Este é o tipo mais simples de XSS. A falha ocorre quando a aplicação recebe uma entrada de dados em uma requisição HTTP e, logo em seguida, mostra esses dados na página de maneira insegura.

Podemos ver o exemplo de uma aplicação (feita para testes de segurança) vulnerável a XSS reflected na imagem abaixo:



Esta aplicação possui um campo de entrada que permite que o usuário busque por uma determinada arte disponível. Vamos digitar qualquer coisa e ver como o site responde:

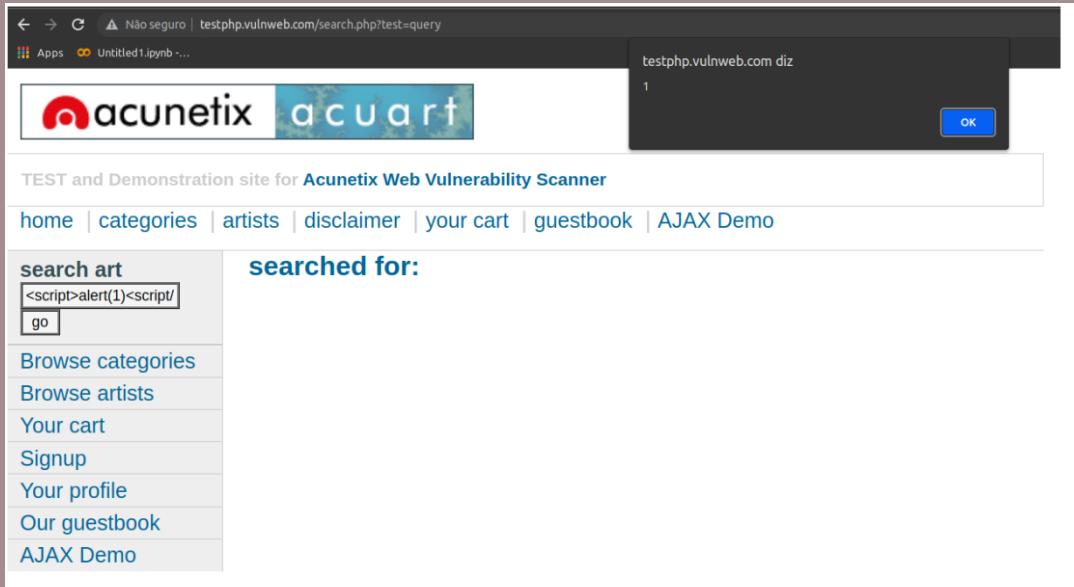
The screenshot shows a web page with the header "acunetix acuart". Below it, a banner says "TEST and Demonstration site for Acunetix Web Vulnerability Scanner". A navigation bar includes links for "home", "categories", "artists", "disclaimer", "your cart", "guestbook", and "AJAX Demo". On the left, there's a sidebar with "search art" input fields containing "buscando" and a "go" button. The main content area displays the text "searched for: buscando".

Podemos ver que digitando a palavra “buscando” e clicando em “go” o site retorna automaticamente a mensagem “searched for:” seguida do conteúdo digitado no campo de entrada. Vamos tentar manipular a entrada enviando códigos HTML e ver se o navegador processa estes códigos:

This screenshot shows the same website setup as the previous one. In the "search art" input field, the user has typed "<p>buscando</p>". When the "go" button is clicked, the search results show the manipulated input as a single paragraph: "buscando".

Dá para perceber que quando enviamos a tag html “<p>buscando</p>” que serve para adicionar um parágrafo, o site responde automaticamente forçando o texto “buscando” a aparecer em um parágrafo, ou seja, o navegador está processando o nosso código externo!

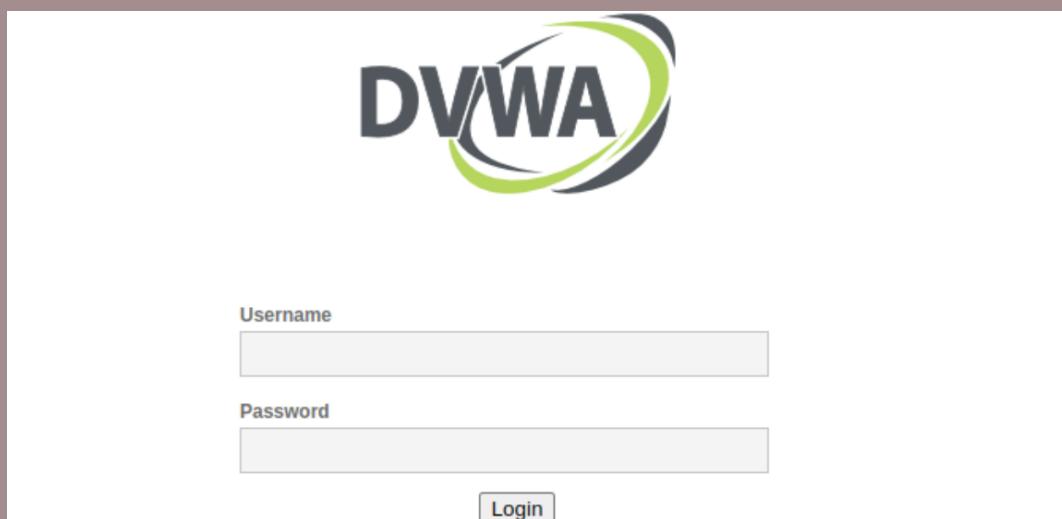
Vamos tentar elaborar uma entrada que processe um código Javascript, pois com essa linguagem podemos fazer coisas mais interessantes. Enviando a tag “`<script>`” juntamente com um “`“alert(1)”` para ver se o site exibe o número 1 em forma de alerta:



E como esperávamos, o site executa o código Javascript e mostra um alerta com o seu conteúdo. Para um indivíduo malicioso atacar um usuário comum com o XSS reflected, ele precisaria enviar o link manipulado do site vulnerável para a vítima. Após o clique no link manipulado com a entrada maliciosa, o navegador da vítima executaria o código Javascript malicioso.

## Stored XSS

Diferentemente do tipo de ataque anterior, no XSS stored o atacante utiliza campos de entrada que possuem a finalidade de armazenar aquelas informações de entrada no banco de dados da aplicação e mostrá-las em uma determinada página que não possui tratativas adequadas de segurança. Sendo assim, o atacante envia o código malicioso, a aplicação salva no banco de dados o conteúdo e todas as pessoas que acessarem a página que mostra aquele conteúdo executarão aquele código no navegador e serão vítimas do ataque. Para a demonstração deste ataque, utilizaremos a aplicação Damn Vulnerable Web Application, a qual é destinada para testes de vulnerabilidades web e está disponível para download grátis.



Ao entrar na aplicação, podemos ver um formulário de login. Vamos fazer login com o usuário padrão “admin, password”.

The screenshot shows the DVWA Stored XSS application. On the left, there's a sidebar with various security test categories: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), and XSS (Stored). The 'XSS (Stored)' option is highlighted. The main content area has a title 'Vulnerability: Stored Cross Site Scripting (XSS)'. It contains a form with fields for 'Name \*' and 'Message \*', and buttons for 'Sign Guestbook' and 'Clear Guestbook'. Below the form, a message box displays: 'Name: test' and 'Message: This is a test comment.'.

Vamos acessar a página destinada a demonstrar a falha em questão. Acessando, podemos ver que existem dois campos de entrada e um botão para salvar o conteúdo digitado. Logo abaixo, vemos que a aplicação mostra as informações que um usuário anterior digitou e enviou (“Name: Test, Message: This is a test comment”).

Primeiramente, vamos digitar qualquer coisa e ver como a aplicação responde:

This screenshot shows the DVWA Stored XSS application after inputting 'Lucas' into the Name field and 'Eu gosto de bacon.' into the Message field, then clicking 'Sign Guestbook'. The message box now shows two entries: 'Name: test' and 'Message: This is a test comment.', followed by a new entry: 'Name: Lucas' and 'Message: Eu gosto de bacon.'.

Podemos perceber que após preencher os campos necessários e clicar em “Sign Guestbook” a aplicação salva a minha mensagem no banco de dados e disponibiliza para qualquer usuário visualizá-la abaixo.

Na demonstração do ataque anterior, nós apenas fizemos o navegador executar um código Javascript simples que mostrava um alerta inofensivo na tela. Nesta demonstração iremos perceber o quanto essa falha é perigosa.

Em diversos sites, existe a opção de permanecer logado em sua conta mesmo se você fechar o navegador. Essa funcionalidade é feita através de tokens de acesso ou sessões que são armazenados no cache do navegador. Estas ferramentas de acesso identificam o usuário do sistema e dizem para a aplicação que aquele usuário já se autenticou anteriormente. Dessa forma, o usuário não precisa digitar seus dados novamente, fazendo login automaticamente no site.

A falha de XSS stored pode ser explorada para roubar essas informações armazenadas no cache do navegador, com o intuito de obter acesso à conta do usuário sem ao menos possuir os dados de login do mesmo.

Abrindo as ferramentas de desenvolvedor do Google Chrome, mais especificamente na aba “Application”, podemos ver que esta informação da sessão está armazenada em um cookie:

| Name      | Value                   | D... | P... | E... | S... |
|-----------|-------------------------|------|------|------|------|
| security  | low                     | L... | /    | S... | 11   |
| PHPSESSID | 8fc143ub2ufrha2t8bkq... | L... | /    | S... | 35   |

Para receber os dados da vítima, vamos inicializar um servidor HTTP que se comunicará com o navegador atacado. Após isso, iremos explorar a vulnerabilidade manipulando a entrada com o objetivo de executar um código Javascript no navegador da vítima, o qual enviará a informação do cookie da mesma ao servidor do atacante:

```
root@lucas:/home/lucas# python3 -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

Manipulando a entrada para armazenar o código malicioso no banco de dados do servidor:

Podemos ver que o campo “message” aparece vazio, porém o código Javascript foi armazenado no banco de dados e qualquer pessoa que acessar a página terá sua sessão roubada. Vamos analisar o lado do servidor do atacante:

```
root@lucas:/home/lucas# python3 -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
127.0.0.1 - - [09/Jun/2022 23:57:22] "GET /?PHPSESSID=8fc143ub2ufrha2t8bkqu8nqf5;%20security=low HTTP/1.1" 200 -

```

Sucesso! Um usuário acessou a página e o seu navegador enviou ao servidor do atacante as informações do cookie de sessão do usuário, ou seja, sua sessão foi roubada (lembrando que isso é uma simulação). Dessa forma, podemos acessar a página de login e substituir o cookie de sessão do meu navegador pelo da vítima (“PHPSESSID=8fc143ub2ufrha2t8bkqu8nqf5”) obtido no ataque e, logo após isso, iremos acessar o endereço base da aplicação (“/”):

The screenshot shows the DVWA login interface on the left and the Chrome DevTools Network tab on the right. The Network tab is expanded to show the 'Storage' section, specifically the 'Cookies' tab under the 'http://' domain. A red arrow points to the 'PHPSESSID' cookie entry, which has its value highlighted in red.

| Name      | Value                      |
|-----------|----------------------------|
| PHPSESSID | 0q18i13s91qv413qtqeatqchr0 |
| security  | low                        |

The screenshot shows the DVWA homepage on the left and the Chrome DevTools Network tab on the right. The Network tab is expanded to show the 'Storage' section, specifically the 'Cookies' tab under the 'http://' domain. A red arrow points to the 'PHPSESSID' cookie entry, which has its value highlighted in red.

| Name      | Value                      |
|-----------|----------------------------|
| PHPSESSID | 8fc143ub2ufrha2t8bkqu8nqf5 |
| security  | low                        |

Sucesso! Conseguimos fazer login na conta da vítima sem informarmos usuário e senha. Durante a simulação feita, foi possível perceber que a falha de Cross Site Scripting é bastante preocupante para uma aplicação Web. Os programadores devem estar preparados para tratar os vetores de entrada do site para evitar que essa falha ocorra.

Vale salientar que este exemplo mostrado na simulação foi bem simples, utilizamos apenas a tag “`<script>`” para explorar a vulnerabilidade. Em aplicações reais, na maioria dos casos, todos os payloads (carga, código malicioso) comuns são facilmente tratados. Existem diversas outras formas diferentes de explorar o XSS, como, por exemplo, utilizando a tag “`<img>`” (tag HTML utilizada para mostrar imagens no site) da seguinte forma:

```

```

Esta manipulação da tag “`<img>`” força a execução da função passada como parâmetro para a propriedade “onerror”, pois a propriedade “src”, que serve para encontrar o caminho da imagem, está com um conteúdo que não condiz com um caminho de uma imagem válida.

## Referências:

- <https://portswigger.net/web-security/cross-site-scripting>
- <https://owasp.org/www-project-top-ten/>
- <https://owasp.org/www-community/attacks/xss/>
- <https://dvwa.co.uk/>

# Três cenários de uso da Inteligência Artificial na Segurança da Informação

por Yanka Ribeiro

As ocorrências de cibercrimes estão crescendo, e conforme crescem, tornam-se mais custosas financeiramente além de demandarem tempo para serem gerenciadas. Um desafio extra, é a evolução das ameaças e técnicas para atacar sistemas e suas defesas. Isso, no contexto atual, onde nossas vidas tendem a se tornarem cada vez mais digitais, coloca em risco os dados agregados a nos – entidades na internet sabem mais sobre nós do que imaginamos!

Assim, não é surpresa que *machine learning* está sendo aplicada na segurança da informação para auxiliar na proteção de sistemas computacionais bem como nossos dados online. Neste artigo, iremos explorar 3 aplicações da inteligência artificial na cibersegurança.

## 1: Detecção de ameaças

Uma das aplicações mais significantes da IA nesse contexto é a detecção avançada de ameaças. Diferentemente de métodos tradicionais, como os antivírus baseados em assinatura, os sistemas baseados em IA agem observando comportamentos incomuns no sistema. Esse antivírus “inteligente” é treinado para pensar como um hacker e dessa forma detectar vulnerabilidades que criminosos reais poderiam explorar.

Claro, uma vez que a fraqueza é detectada, o administrador do sistema pode ser alertado antes que um ataque real aconteça, possibilitando a tomada de ações precatórias. Quando esse software inteligente é combinado com outras formas de detecção, a probabilidade de detectar com uma ótima acurácia aumenta.

## 2: Aperfeiçoamento de autenticações

Saber quando uma requisição de conexão é legítima ou não, requer monitoramento full-time por gerentes de redes, especialmente em grandes empresas que processam milhares de requisições. Agora, coloque na equação, o fato que humanos são limitados em termos de eficiência, horas de trabalho e acurácia... A autenticação pode se tornar uma ação laboriosa. Tecnologias de autenticação que possuem o apoio da IA analisam fatores individuais em tentativas de logins e devolvem um score de risco para o caso específico. Por exemplo, o sistema pode notificar sobre um usuário tentando acessar a partir de alguns IPs específicos na madrugada. Além disto, usando redes neurais – que imita o cérebro humano e se adapta através do contato com os dados – o sistema aprende a desenvolver algoritmos melhores em identificar os fatores que indicam um ataque.

Por fim, para reduzir a incidência de falsos positivos (alarmes falsos), o sistema baseado em IA pode permitir que usuários com baixo score de risco (usuários que realmente podem acessar) realizem o login de forma mais rápida, sem requerer autenticações de segurança como reconhecimento de digitais.

### 3: Detecção de engenharia social

Engenharia social envolve a manipulação do psicológico humano para revelar informações confidenciais ou praticar ações que coloquem a vítima em situação vulnerável. De acordo com o relatório da Verizon, 93% dos vazamentos de dados bem sucedidos foram causados por ataques de engenharia social.

Criminosos utilizam ferramentas de machine learning para acessar logins de funcionários e assim ganhar acesso a rede de empresas. Isso acontece porque muitos usuários não sabem diferenciar o que é real de uma voz/rosto gerada artificialmente, por exemplo.

Para elucidar, a voz de um CEO pode ser usada para promover um ataque de *phishing*, aumentando as chances de que os funcionários acreditem e caiam no golpe. Em contrapartida, ferramentas que também utilizam IA podem ser treinadas para reconhecer *deepfakes* e rotulá-los, para que então, possam mais possam ser evitados no futuro. Nessa abordagem, o modelo aprende os trejeitos corporais do CEO real e usa essa informação para comparar com o vídeo suspeito em questão sendo capaz de classificar entre verdadeiro ou falso.

#### Referências:

- <https://resources.infosecinstitute.com/topic/deepfake-phishing-can-you-trust-that-call-from-the-ceo/>
- <https://developer.ibm.com/articles/ai-and-security/>

# Smalltalk na Programação

por Maria Antônia

Dizer que Smalltalk é a linguagem de programação orientada a objetos por excelência é desmerecer-lá. Smalltalk inventou o conceito de orientação a objetos há vinte anos atrás e, como subproduto, o conceito de interação com o usuário através de janelas. Apesar de ser a linguagem mais citada como exemplo de uma linguagem de programação revolucionária, bem concebida e elegante, ela é uma das linguagens de programação da atualidade que mais carece de boa literatura.

Os programadores definem classes de objetos em suas aplicações para imitar (ou simular) o mundo real. Sendo relativamente fácil de aprender comparado a linguagens como C++ e ADA. Com o seu código-fonte fácil de ler, o que o torna a linguagem de programação ideal para iniciantes.

## Sintaxe

A sintaxe de Smalltalk é bastante diferente das linguagens tradicionais. Ao invés do que é usado na maioria das linguagens tradicionais, em Smalltalk utiliza-se sempre a ordem <objeto recebedor><mensagem>.

No exemplo abaixo, o método *publish* é formado de uma linha, onde se envia a mensagem *show* para o objeto *Transcript*, com o parâmetro *Hello, world!* (que é um objeto).

```
publish
Transcript show: 'Hello, world!'
```

Regras básicas da linguagem:

- Tudo é representado como objetos. (De longe, a regra mais importante em Smalltalk). Toda computação é disparada pelo envio de mensagens. Uma mensagem é enviada para um objeto fazer alguma coisa.
- Quase todas as expressões são da forma <recebedor><mensagem>.
- Mensagens fazem com que métodos sejam executados, sendo que o mapeamento de mensagens para métodos é determinado pelo objeto recebedor. Os métodos são as unidades de código em Smalltalk, equivalente a funções ou procedimentos em outras linguagens.
- Todo objeto é uma instância de alguma classe. 12 é uma instância da classe *SmallInteger*. 'abc' é uma instância da classe *String*. A classe determina o comportamento e os dados de suas instâncias.
- Toda classe tem uma classe mãe, exceto a classe *Object*. A classe mãe define os dados e comportamento que são herdados por suas classes filhas. A classe mãe é chamada de superclasse e suas filhas, subclasses.

Como o aprendizado do Smalltalk pode me tornar um desenvolvedor melhor? O Smalltalk tem vários recursos importantes que estavam à frente de seu tempo:

- Persistência baseada em imagem

Objetos: Tudo é um objeto, e objetos se comunicam apenas por meio de mensagens (o mais puro “OO” e um dos mais antigos)

- Programação “ao vivo”
- Técnicas avançadas de depuração, como alterações de código em tempo real
- Uma interface IDE simples e organizada
- Idiomas específicos de domínio (*DSL: Domain-Specific Language*): A única forma que o Smalltalk trabalha, portanto, os programadores só precisam se concentrar no domínio do problema usando uma linguagem e notação que é natural para esse domínio.

Em essência, a principal vantagem do Smalltalk como uma linguagem produtiva e ferramenta de aprendizado é que ela retira a maior parte, se não todo, o estresse cognitivo de linguagens tradicionais como Java.

### **Curiosidades:**

- O Smalltalk apresentou o mundo à máquina virtual para linguagens (ou VM), que permite que o software seja independente de plataforma. Essa é a mesma tecnologia que sustenta o Java (JVM) e o .NET, além do Android (Dalvik).
- A Smalltalk também foi pioneira na compilação JIT (*just-in-time*), uma técnica para melhorar drasticamente o desempenho de bytecodes como o Java.
- Do Smalltalk veio o primeiro IDE (*Integrated Development Environment*) moderno (ambiente de desenvolvimento integrado), que incluía um editor de texto, um navegador de sistema ou classes, um inspetor de objetos ou propriedades e um depurador. Isso levou a muitos IDEs que os desenvolvedores adotam hoje, como o Visual Studio, o Xcode e o IntelliJ IDEA.
- Pessoalmente, acho que nenhum desses IDEs pode se comparar com o IDE do Smalltalk em simplicidade, elegância e velocidade de desenvolvimento; o original ainda é o melhor!
- Desde o início, Smalltalk teve *closures*, que são funções de primeira classe com escopo léxico. Em essência, uma closure é uma função de retorno de chamada que pode ver variáveis não-locais no local onde elas foram definidas. Isso pode ajudar você a escrever um código muito mais compacto e legível. Os closures estão sendo adotadas em muitas das principais linguagens, como Java, C # e PHP.
- O Smalltalk foi a primeira ferramenta em linguagem para suportar programação “ao vivo” (“live” programming) e técnicas avançadas de depuração, como inspeção imediata e mudanças de código durante a execução. Hoje, a depuração ao vivo é possível em C # com o “Edit and Continue” do Visual Studio e em Java com HotSwap.

- O Smalltalk introduziu o MVC (Model-View-Controller) para o mundo. O MVC é um padrão de arquitetura de software para implementar interfaces de usuário. É popular com aplicativos GUI de desktop e aplicativos da web. Hoje em dia, é a arquitetura que a maioria dos desenvolvedores da Web aprende primeiro.
- Em grande parte, Smalltalk é responsável por nos dar desenvolvimento orientado a testes (ou TDD) e programação extrema (ou XP), ambos muito influentes nas práticas agile padrão de hoje. Smalltalk fez do “duck typing” uma coisa comum. Duck typing é quando a “verificação de tipos” é adiada até momento da execução – quando os recursos de reflexão são usados para garantir o comportamento correto. Encontramos a duck typing em muitas linguagens hoje, incluindo Java, Python, Common Lisp, Go, Groovy, Objective-C e PHP.
- O Smalltalk foi pioneiro no desenvolvimento de bancos de dados de objetos. Embora não tenham entrado no mainstream, os bancos de dados de objetos têm seus nichos de mercado.
- O melhor exemplo de um produto de banco de dados de objetos é o GemStone/S , que é bem adequado para sistemas distribuídos escalonáveis, de alto desempenho e multicamadas.
- O Smalltalk nos deu o primeiro navegador de refatoração. Naturalmente, o suporte à refatoração pode ser encontrado na maioria dos IDEs hoje.
- O Smalltalk foi fundamental no desenvolvimento da interface gráfica do usuário (ou GUI) e da interface de usuário do tipo “o que você vê é o que obtém” (WYSIWYG).

## Referências:

- <https://www.inf.ufsc.br/~aldo.vw/smalltalkbook.html>
- [https://chicoary-wordpress-com.cdn.ampproject.org/v/s/chicoary.wordpress.com/2019/03/28/como-aprender-smalltalk-pode-fazer-de-voce-um-desenvolvedor-melhor/amp/?amp\\_gsa=1&amp\\_js\\_v=a9&usqp=mq331AQKKAFQArABIIACAw%3D%3D#amp\\_ct=1656374865252&amp\\_tf=De%20%251%24s&aoh=16563748896418&referrer=https%3A%2F%2Fwww.google.com&ampshare=https%3A%2F%2Fchicoary.wordpress.com%2F2019%2F03%2F28%2Fcomo-aprender-smalltalk-pode-fazer-de-voce-um-desenvolvedor-melhor%2F](https://chicoary-wordpress-com.cdn.ampproject.org/v/s/chicoary.wordpress.com/2019/03/28/como-aprender-smalltalk-pode-fazer-de-voce-um-desenvolvedor-melhor/amp/?amp_gsa=1&amp_js_v=a9&usqp=mq331AQKKAFQArABIIACAw%3D%3D#amp_ct=1656374865252&amp_tf=De%20%251%24s&aoh=16563748896418&referrer=https%3A%2F%2Fwww.google.com&ampshare=https%3A%2F%2Fchicoary.wordpress.com%2F2019%2F03%2F28%2Fcomo-aprender-smalltalk-pode-fazer-de-voce-um-desenvolvedor-melhor%2F)
- <https://pt.m.wikipedia.org/wiki/Smalltalk>
- <https://marciobueno.com/linguagens-programacao/smalltalk-primeiras-linguagens-orientadas-objetos/>
- [https://www.researchgate.net/publication/262882317\\_Conhecendo\\_o\\_Smalltalk\\_-\\_Todos\\_os\\_Detalhes\\_da\\_Melhor\\_Linguagem\\_de\\_Programacao\\_Orientada\\_a\\_Objetos](https://www.researchgate.net/publication/262882317_Conhecendo_o_Smalltalk_-_Todos_os_Detalhes_da_Melhor_Linguagem_de_Programacao_Orientada_a_Objetos)

## Novos Membros da Academia Hacker

**Cauê Bittencourt**

**Daniel José**

**Emily Brito**

**Felipe Ferreira**

**Frederico Guilherme**

**Gabriel Nicácio**

**Guilherme Lyra**

**Jadde de Freitas**

**João Pedro**

**Jorge Lucas**

**Leila Maria**

**Lucas Barros**

**Lucas Vinicius**

**Luiz Guimarães**

**Marcos Douglas**

**Maria Antônia**

**Mateus Barros**

**Natália de Assis**

**Otávio Barbosa**

**Ruan Silva**

**Sarah de Lima**

**Yanka Ribeiro**

## **Info Hacker - Edição 010**

### **28/06/2022**



#### **Autores:**

Lucas Buarque  
Yanka Ribeiro  
Maria Antônia

#### **Revisão**

Everton Borges  
Bruno Severo



acha.ufal

## **Edições Anteriores**

<https://github.com/Academia-Hacker/Info-Hacker>