Revista Hackerspace

Vol. 1

Projeto Hackerspace
2019

Apresentação

O Projeto Hackerspace é um projeto de extensão promovido pela UFSCar Campus Sorocaba e realizado na forma de encontros que discutem a contracultura hacker, apresentando personagens, acontecimentos, aspectos socioculturais, artefatos e atividades com a finalidade de que os participantes sejam expostos à contracultura em questão e desenvolvam um projeto ou material acerca da mesma. A principal intenção do projeto é desmistificar o conceito e figura do hacker, enquanto proporcionando um espaço de exposição, produção e compartilhamento de conteúdo que se configura como dentro da contracultura de forma que participantes de diferentes níveis de conhecimento técnico possam participar.

Esta revista tem como objetivo divulgar os trabalhos desenvolvidos pelos alunos participantes do Projeto Hackerspace no ano de 2019. Esperamos que esta revista possibilite a difusão dos conhecimentos adquiridos na atividade para a comunidade em geral. Gostaríamos de agradecer todos os alunos que contribuíram com artigos ou projetos para esta edição da revista.

Organização e edição:
Gustavo Buoro Branco de Souza
Leandro Naidhig
Marcus Vinicius Natrielli Garcia
Michele Argolo Carvalho

Supervisão:
Gustavo M. D. Vieira

Conteúdo

Gabriel P. Andrade
gabrielperesdeandrade@hotmail.com
Daniel L. Júnior
daniel.dlj@hotmail.com

Parte I

Artigos

Capítulo 1

Um resumo do inicio da *hacking scene* de 3DS

Gabriel E. U. Martin
eijiuema@gmail.com

Resumo

Este artigo descreve brevemente eventos marcantes do inicio da *hacking scene* de 3DS, como o inicio da 3DBrew, as primeiras ROMs dumpadas, o *flashcart* Crown3DS, o *scam* do financiamento coletivo, o primeiro acesso à *Userland*, o *flashcart* Gateway3DS, seus clones e o escândalo do *Brickgate/Brickway*.

1.1 Introdução

Este artigo tem como objetivo resumir a história do inicio da *hacking scene* de 3DS, descrevendo brevemente eventos da cena, em ordem cronológica e trazendo algumas curiosidades.

Alguns eventos não possuem nas referências suas fontes originais, pois as principais fontes de informação do tema são postagens, mensagens em fóruns, e sites temporários que com o passar do tempo foram apagados e/ou modificados.

Para não estender, nem desviar o foco da história, muitos dos detalhes dos eventos como explicações de termos técnicos estão em notas de rodapé.

1.2 Agradecimento

Gostaria de agradecer ao usuário *zoogie* do fórum *gbatemp.net* por ter reunido os eventos importantes da cena em ordem cronológica em um único *post* [19], que possibilitou a criação deste artigo.

1.3 Lançamento do 3DS e da 3DBrew

No final de fevereiro de 2011 o console portátil da Nintendo foi lançado exclusivamente no Japão, e no final de março no ocidente [17]. Pouco antes do lançamento no ocidente foi lançada também a *3dbrew.org*, que viria a se tornar o ponto central da cena.

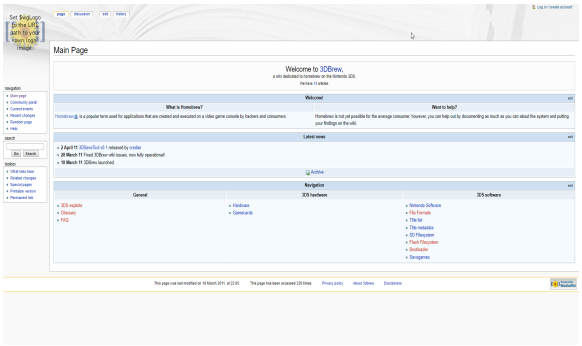


Figura 1.1: Página inicial da 3DBrew em março de 2011

1.4 Primeira ROM dumpada

Apesar do rápido lançamento da *3dbrew*, a cena demorou para ter os seus primeiros resultados. Foi apenas dois meses depois, em junho, que as primeiras *ROMs*[1] foram «dumpadas» (tiveram

[1]Cópia em arquivo de um chip de memória ROM, muitas vezes um cartucho de Video-game, o *firmware* de um sistema

seu conteúdo extraido do cartucho) pelo grupo LGC. Os cartuchos continham as versões europeias de *Super Monkey Ball 3D* e *Tom Clancy's Ghost Recon Shadow Wars*, o feito foi descoberto quando os arquivos .nfo[2] foram publicados no site LUDiBriA[3].

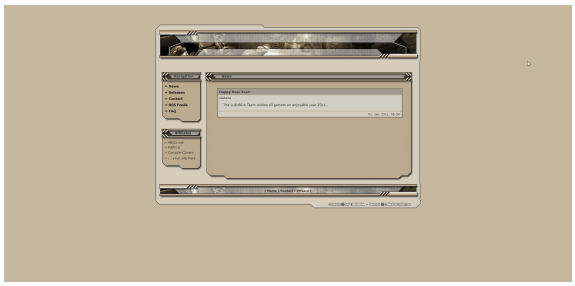


Figura 1.2: Página inicial do LuDiBriA em fevereiro de 2011

1.5 Crown3DS

Em setembro do mesmo ano um novo grupo anunciou o primeiro *flashcart* de 3DS (um *flashcart* é um cartucho que se conecta fisicamente a um dispositivo, permitindo a execução de programas). O grupo chamou o novo cartucho de *Crown3DS* e prometia que não só seria capaz de rodar aplicações *homebrew*[4]como também jogos oficiais do console.

Alguns dias depois do anúncio, o grupo lançou um site (Figura 1.3) em inglês (com diversos erros de escrita) com uma barra de progresso, supostamente para o lançamento do cartucho. O site permaneceu no ar por meses sem mudanças na barra de progresso, gerando um meme (Figura 1.4), mas à custo de um sonho.

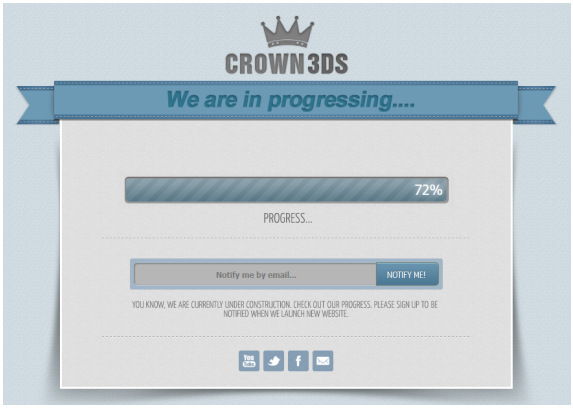


Figura 1.3: Site do Crown3DS

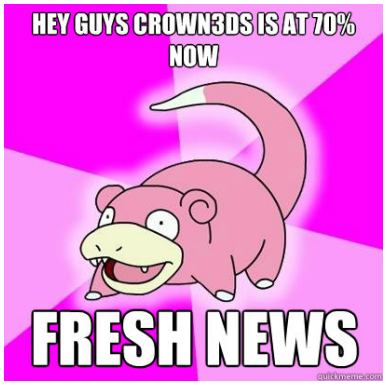


Figura 1.4: Meme sobre o site do Crown3DS

1.6 Engenharia reversa e inicio do financiamento coletivo

No dia 2 de novembro de 2012, o usuário gshock (conhecido também como JL12, um contribuidor da *3DBrew*) propôs em uma postagem [4] o financiamento coletivo do serviço de *decapping*[5]de um 3DS, feito por um profissional, com a finalidade de obter informações através de engenharia reversa. A justificativa foi de que as fotos dos circuitos internos, bem como as informações proprietárias extraídas do chip acelerariam muito o desenvolvimento da cena.

embarcado ou de uma máquina de *Arcade* [14].

[2]Extensão de nome de aquivo comumente usada para arquivos de texto que acompanham vários lançamentos da cena digital [16].

[3]Site que listava arquivos .nfo de lançamentos de jogos para consoles, o site entrou em manutenção em dezembro de 2015 (removendo todos os lançamentos) e foi posteriormente desativado.

[4]Homebrew (do inglês "feito em casa") é um termo utilizado entre programadores ou hackers, se referindo a software não-oficial produzido em casa, geralmente para consoles de videogame [13].

[5]É o processo de remover a capa protetora de um microchip, de forma que o die (pastilha onde o circuito é fabricado) seja exposto para inspeção visual do circuito impresso nele [10].

Os custos estimados do financiamento coletivo foram de U$2000 e incluíam a compra e o envio de um 3DS para *decapping*, e «talvez» a distribuição de cópias das fotos para os contribuidores. Em poucas semanas o financiamento coletivo atingiu U$1000, mas demoraria alguns meses para atingir o valor esperado.

1.7 Primeiro acesso à *Userland*

No dia 16 de dezembro de 2012 a conta de twitter associada à loja australiana de *Modchips*[6]OzModchips publicou uma imagem de um suposto 3DS hackeado (Figura 1.5). O feito foi posteriormente confirmado por diversos veteranos da cena.

O *exploit*[7] utilizado consiste em causar um *overflow*[8] durante a tela de salvamento do jogo *The Legend of Zelda: Ocarina of Time 3D* e foi o primeiro acesso à *Userland* conhecido no 3DS.

Figura 1.5: Foto do primeiro 3DS hackeado

[6]Microchip adicionado a equipamentos eletrônicos, usualmente consoles de videogames, que permite que jogos piratas e outros softwares possam ser rodados pelo equipamento [15].

[7]Pedaço de software, um pedaço de dados ou uma sequência de comandos que tomam vantagem de um defeito, falha ou vulnerabilidade a fim de causar um comportamento acidental ou imprevisto a ocorrer no software ou hardware de um computador ou em algum eletrônico [12].

[8]Em segurança computacional e programação, um transbordamento de dados ou estouro de buffer (do inglês buffer overflow ou buffer overrun) é uma anomalia onde um programa, ao escrever dados em um buffer, ultrapassa os limites do buffer e sobrescreve a memória adjacente. Esse é um caso especial de violação de segurança de memória [18].

A *Userland* ou espaço do usuário se refere à todo código que é executado fora do sistema operacional do *kernel* de um dispositivo [11], isso inclui os jogos e outros softwares do 3DS.

No entanto, pouco foi feito através do *exploit*, pois apesar de ter sido um marco na história da cena, o acesso à *Userland* permite apenas que os dados e arquivos do próprio jogo utilizado no processo fossem acessados. Além disso a falha só foi divulgada publicamente anos depois [1].

1.8 O desfecho do financiamento coletivo

No dia 31 de maio de 2013, a página do financiamento coletivo já marcava U$2300 arrecadados, a meta havia sido atingida. No entanto, gshock desapareceu do fórum e da 3DBrew. Coincidentemente, ou não, um dia antes havia sido anunciado um novo *flashcart*, o Gateway3DS.

1.9 Gateway3DS

No dia 30 de maio de 2013, foi anunciado o desenvolvimento de um *flashcart* capaz de rodar ROMs de jogos originais, o Gateway3DS [3]. Junto do anúncio foi publicado um vídeo que mostrava um cartão microSD sendo inserido em um cartucho de 3DS, que depois foi inserido em um 3DS XL branco. Ao ligar o 3DS, o jogo *Luigi's Mansion: Dark Moon* aparece no menu. O cartão microSD é removido, e posteriormente é inserido outro, trocando o jogo do menu por *Resident Evil Revelations*. O processo se repete com o jogo *Super Mario 3D Land*. Durante a inicialização de cada um dos jogos, mensagens de erro relacionadas à saves surgem na tela.

O anúncio e, principalmente, o vídeo geraram as mais diversas reações. Parte da comunidade estava muito empolgada e ansiosa para o lançamento oficial do produto. Enquanto outros eram contra o lançamento de um *flashcart* tão cedo na, até então, curta vida do 3DS. Além disso, surgiu um mar de dúvidas, principalmente a respeito da legalidade e do funcionamento do produto.

Em meio às dúvidas, os responsáveis pelo GW3DS divulgaram um *FAQ*, que sanava as principais dúvidas da comunidade:

Figura 1.6: Cartucho de 3DS e o GW3DS lado a lado

- O *flashcart* era limitado a um jogo por cartão;
- Não era possível executar aplicações *homebrew*;
- Saves ainda não estavam funcionando;
- O lançamento seria até o final de junho.

O FAQ dividiu ainda mais a comunidade. Um lado era contra o dispositivo, pois tinha como finalidade única a pirataria, uma vez que não era possível dumpar seus próprios jogos nem executar aplicações *homebrew*. O outro lado, no entanto, apoiou o dispositivo, talvez pela vontade de obter jogos de graça.

Junho chegou ao fim, e nenhum anúncio havia sido feito por parte do time do GW3DS, o que levou à suspeita de que o dispositivo era falso. Finalmente no dia 11 de julho de 2013 o time anunciou que a pré-venda havia começado, sanou mais dúvidas da comunidade e divulgou novas informações:

- O produto lançaria funcionando apenas nos *firmwares* 4.1-4.5;
- O produto contém dois cartuchos, um para executar ROMs de 3DS e outro para instalar o *exploit* e executar ROMs e aplicações *homebrew* de DS.

Enquanto as pré-compras eram enviadas, o time do GW3DS divulgou um novo vídeo mostrando as funções de *firmware spoofing* e *bypass* de trava de região. O primeiro permitiria que jogos que exigissem atualizações de *firmware* pudessem ser jogados sem a necessidade de atualizar. Já o segundo; que jogos de regiões diferentes da região de fabricação do 3DS pudessem ser jogados.

Apesar das limitações ainda existentes, como a inabilidade de jogar online, de rodar aplicações *homebrew* e a necessidade de rodar uma versão antiga do *firmware* do console, o GW3DS marca um grande avanço na *hacking scene* de 3DS.

Clones do GW3DS e o *Brickgate/Brickway*

Pouco tempo depois do lançamento do GW3DS, clones como o R4 Gold Deluxe e o Orange3DS foram lançados. Em resposta aos clones, em janeiro de 2014, o time do GW3DS inseriu uma nova função na versão «Beta 2 (3.2-3.3b)», que brickava[9] os consoles que falhassem na verificação do arquivo «launcher.dat» [2][5].

A atualização causou uma onda de bricks gigantesca entre os usuários de clones, e vitimou até mesmo unidades de usuários de *flashcarts* legítimos. O time do GW3DS posteriormente se ofereceu para consertar essas unidades.

1.10 Citra, o primeiro emulador

No dia 30 de agosto de 2013, foi feito o primeiro *commit* do emulador Citra[8], hoje o maior emulador de 3DS para PC. Apesar de ter sido iniciado em 2013, só viria a deslanchar em 2016, mostrando o quão complexo é o sistema do 3DS.

1.11 Palantine, o primeiro *custom firmware*

No dia 1 de novembro de 2014, o usuário Ryanrocks462 vazou um *custom firmware* produzido internamente por veteranos da cena [6]. O *firmware* veio a ser conhecido como Palantine, e apesar de muito limitado, era capaz de executar arquivos .cia, que hoje é a extensão mais utilizada para distribuição de software na comunidade. A importância desse vazamento é tanta, que em menos de

[9] Danificar um dispositivo eletrônico, através de dano físico, mal configuração, corrupção de *firmware*, ou problema de hardware, de forma que o dispositivo não funcione mais, tornando-o tão tecnologicamente útil quanto um tijolo [9].

uma semana o time do GW3DS soltou uma atualização que tornava possível executar arquivos .cia no dispositivo.

1.12 Ninjhax, e finalmente *homebrew*

No dia 16 de novembro de 2014, o usuário Ryukouki publicou uma prévia do que viria a ser o primeiro *exploit* capaz de executar aplicações *homebrew* [7].

O *exploit* consistia no uso de um recurso do jogo *Cubic Ninja* de escanear QRCodes. O conteúdo do QRCode era processado de forma privilegiada e sem verificações, permitindo a instalação do *exploit* que possibilitava a instalação de aplicações *homebrew*.

É importante destacar que o ninjhax só foi possível graças ao trabalho conjunto de diversos membros da cena:

- smea — 3DS research, core exploit code for all versions, ctrulib improvements, hbmenu code, testing/debugging
- yellows8 — 3DS research, ctrulib improvements, auditing, help with pretty much everything
- plutoo — 3DS research, ctrulib improvements, auditing, help with pretty much everything
- fincs — 3DSX format/code, ctrulib improvements, devkitARM integration, testing
- mtheall — ctrulib improvements, hbmenu code, testing
- GEMISIS — hbmenu code, testing
- Fluto, Arkhandar — hbmenu design
- Normmatt, ichfly — general help, testing
- case — javascript master
- lobo — webpage template

1.13 Conclusão

Desde o lançamento até o desenvolvimento de ferramentas e recursos que tornam o *hacking* do 3DS realmente útil, houve muito investimento de tempo e esforço por parte da comunidade. Os desafios, bem como o desenvolvimento do inicio da cena mostram o quão complexo, desafiador e interessante é o sistema do 3DS. Os eventos descritos neste artigo mostram apenas o inicio do desenvolvimento da cena, que hoje é muito mais desenvolvida e madura.

1.14 Bibliografia

[1] 3DBrew. 3DS Userland Flaws - 3dbrew. https://www.3dbrew.org/wiki/3DS_Userland_Flaws, 2019.

[2] crediar. Warning - gateway team bricks card on purpose!, 2014. URL: https://gbatemp.net/threads/warning-gateway-team-bricks-card-on-purpose.360568/.

[3] Devin. Gateway 3ds: Official gbatemp review, 2013. URL: https://gbatemp.net/review/gateway-3ds.79/.

[4] gshock. 3ds decapping fundraising topic, 2012. URL: https://gbatemp.net/threads/3ds-decapping-fundraising-topic.336767/.

[5] profi200. Warnung an alle flashcard nutzer!, 2014. URL: https://ngb.to/threads/161-Gateway-3DS-Erste-funktionierende-3DS-Flashcart/page14?p=187387#post187387.

[6] Ryanrocks462. [release] 3ds cfw-cia installer, 2014. URL: https://gbatemp.net/threads/release-3ds-cfw-cia-installer.373461/.

[7] Ryukouki. Introducing ninjhax - a nintendo 3ds homebrew exploit!, 2014. URL: https://gbatemp.net/threads/introducing-ninjhax-a-nintendo-3ds-homebrew-exploit.374233/.

[8] shizzy1. Initial commit, 2013. URL: https://github.com/citra-emu/citra/commit/8404376c6ba46433a3fe0ab81e029e39f85c6b65.

[9] Wikipedia. Brick (electronics) — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Brick%20(electronics)&oldid=922443887, 2019. [Online; accessed 01-December-2019].

[10] Wikipedia. Decapping — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Decapping&oldid=875163417, 2019. [Online; accessed 01-December-2019].

[11] Wikipedia. Espaço de usuário — Wikipedia, the free encyclopedia. http://pt.wikipedia.org/w/index.php?title=Espa%C3%A7o%20de%20usu%C3%A1rio&oldid=53848667, 2019.

[12] Wikipedia. Exploit (segurança de computadores) — Wikipedia, the free encyclopedia. http://pt.wikipedia.org/w/index.php?title=Exploit%20(seguran%C3%A7a%20de%20computadores)&oldid=55055446, 2019.

[13] Wikipedia. Homebrew — Wikipedia, the free encyclopedia. http://pt.wikipedia.org/w/index.php?title=Homebrew&oldid=56352762, 2019.

[14] Wikipedia. Imagem ROM — Wikipedia, the free encyclopedia. http://pt.wikipedia.org/w/index.php?title=Imagem%20ROM&oldid=56532846, 2019.

[15] Wikipedia. Modchip — Wikipedia, the free encyclopedia. http://pt.wikipedia.org/w/index.php?title=Modchip&oldid=39190435, 2019.

[16] Wikipedia. .nfo — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=.nfo&oldid=903564092, 2019.

[17] Wikipedia. Nintendo 3DS — Wikipedia, the free encyclopedia. http://pt.wikipedia.org/w/index.php?title=Nintendo%203DS&oldid=55130688, 2019.

[18] Wikipedia. Transbordamento de dados — Wikipedia, the free encyclopedia. http://pt.wikipedia.org/w/index.php?title=Transbordamento%20de%20dados&oldid=52669728, 2019.

[19] zoogie. A pretty brief history of the 3ds hacking/homebrew scene, 2014. URL: https://gbatemp.net/threads/3ds-hacking-scene-history.443396/#post-6713845.

Capítulo 2

Compreendendo o funcionamento e o impacto de um ataque de *Cross-Site Scripting* em aplicações WEB

Lucas S. A. Costa
lucasandrad1404@gmail.com
Luis F. Guimarães
luiz.guimaraes@dcomp.sor.ufscar.br

Resumo

Aplicações WEB, cada vez mais, são desenvolvidas para suportar diversos tipos de atividades que atendam às demandas de seus usuários. Entretanto, à medida que novas funcionalidades são acrescentadas, crescem também as vulnerabilidades do sistema. Ataques do tipo *Cross-Site Scripting (XSS)* foram colocados em sétimo lugar no OWASP *Top Ten Most Critical Web Application Security Risks* [1], e permanecem em segundo lugar como o tipo de ataque às quais as aplicações web são mais vulneráveis [2].

O documento a seguir tem por objetivo explicar o funcionamento de um ataque de *Cross-Site Scripting*, analisando cada um dos métodos usados para realizar os ataques e as razões que tornam possível suas ocorrências. Por fim, tendo entendido todo o processo de um ataque desse tipo, iremos mostrar as principais formas de evitá-los.

2.1 Introdução

Ataques XSS consistem de uma injeção, por parte do atacante, de código em paǵinas web que são consideradas como confiáveis por usuários em geral. Quando o browser do cliente lê e executa o código associado à página, acaba executando por consequência o código malicioso nela inserido. Como tal script está associado a uma página tida como confiável pelo browser, é possível que hackers possam acessar e manipular *cookies*, realizar roubo de credenciais ou ter acesso a informações sensíveis do usuário.

Os dois tipos mais comuns de *Cross-Site Scripting* são ataques XSS de tipo 1 e os do XSS de tipo 2. Ataques do tipo 1 baseiam-se na modificação de mecanismos de requisição do usuário (como uma URL ou formulários), de forma que ações programavelmente ativáveis da aplicação no servidor possam ser disparadas a partir do código malicioso presente em tal requisição. Já ataques do tipo 2, se caracterizam pela inserção permanente de código malicioso no servidor que hospeda a aplicação web. Dessa forma, sempre que houver a requisição de páginas que contenham tal código, ele será executado pelo lado do cliente. Há ainda um terceiro tipo de XSS, chamado *DOM Based* XSS, no qual o ataque é realizado independentemente do *payload* de terceiros na resposta do servidor [4].

2.2 XSS de Tipo 1

O *Cross-Site Scripting* de tipo 1 é também conhecido por *reflected* XSS. Tal denominação se deve ao método como as vulnerabilidades referentes ao processamento das entradas dos clientes são exploradas: o algoritmo responsável pelos danos ao cliente é executado após ter sido enviado, pelo próprio cliente, à aplicação web. O fato da aplicação retorna a string contendo esse algoritmo ao usuário, é o que faz com que esse tipo de *Cross-Site Scripting* tenha o nome de "refletido".

O ataque ocorre geralmente com o uso de engenharia social por parte do atacante. Este tem por objetivo fazer com que a vítima envie *requests* contendo código malicioso à aplicação web vulnerável. Isso pode ser feito de diversas formas, sendo uma das mais simples o clique de um usuário desprevenido em uma URL com um *script* malicioso pré-inserido. A aplicação então envia uma resposta ao cliente, e essa resposta pode, por exemplo, conter uma página HTML carregando os dados anteriormente dados como entrada por ele (página de boas vindas mostrada após um login bem sucedido, por exemplo). Quando essa resposta do servidor é processada pelo navegador do cliente, o pedaço de código inserido pelo atacante, que agora faz parte desse documento HTML, acaba por ser executado.

Como consequência do processo descrito acima, um ataque bem sucedido pode garantir que o atacante obtenha acesso aos *cookies* do usuário, e que realize basicamente qualquer ação da aplicação à qual o usuário tenha acesso.

2.3 XSS de Tipo 2

O *Cross-Site Scripting* de tipo 2, também conhecido como *Stored* XSS, é realizado através da inserção de código, que será executado pelo *browser*, em respostas HTTP enviadas pelo servidor de uma aplicação web. O primeiro passo para a ocorrência desse tipo de ataque, similarmente ao que ocorre no *Reflected* XSS, consiste no envio deste código malicioso ao servidor da aplicação web. Porém uma diferença é que, para que isso ocorra não são necessárias medidas de engenharia social por parte do *hacker*, já que ele mesmo pode enviar *scripts* que ficarão armazenados no servidor aguardando por uma requisição pelo lado do cliente. Tal permanência do código no servidor é o que faz com que o ataque receba o nome de *Stored"*.

As formas pelas quais o cliente pode ser infectado podem variar, já há algumas possibilidades diferentes para se armazenar o código invasor em servidores web. Um algoritmo inserido por *hackers* pode ser mantido em pontos do sistema como em campos para comentários, mensagens de textos enviadas ao usuário pelo sistema, nos *logs* de visitantes ou até mesmo no banco de dados da aplicação [7]. Em qualquer momento que o usuário solicite os dados que contém tal código malicioso, seu navegador o executará.

As consequências desse tipo de ataque são as mesmas do que as do ataque do tipo 1, que incluem o roubo de credenciais, acesso aos *cookies* e acesso a informações sensíveis. Uma diferença que vale ser ressaltada, entretanto, é que no *Reflected* XSS a própria vítima do ataque é levada a transmitir o *script* malicioso ao sistema, e geralmente é ela quem sofre com os efeitos do mesmo. Já no *Stored* XSS, vez que alojado no servidor da aplicação, o atacante prejudicará todos aqueles que o acessem até que seja percebida a ocorrência da invasão. Vale ressaltar a detecção da presença de códigos estranhos no sistema pode ser difícil, já que muitas vezes esse código estará presente em arquivos do tipo XML ou JSON enviados como resposta aos clientes, e não no código fonte das páginas da aplicação.

2.4 Fatores para a ocorrência de ataques XSS

Levando em conta todos os passos necessários para que se realize um ataque do tipo XSS, é válido considerar que a incorporação, nos navegadores, de mecanismos que evitem a execução de códigos maliciosos injetados nas páginas enviadas pelo lado do servidor (como em [5]), sejam eficazes em garantir a segurança dos usuários.

Porém, desenvolvedores web possuem o potencial de eliminar a grande maioria das vulnerabilidades que possibilitam ataques XSS através de boas práticas na hora de implementar a aplicação [6], sem depender de softwares de terceiros (usuários podem usar diferentes navegadores, que pos-

sivelmente nem sempre estarão atualizados). Alguns dos principais pontos de vulnerabilidades responsáveis por possibilitar a ocorrência desses ataques, estão presentes nas partes de aplicações web descritas a seguir.

- **Páginas de buscas**: esse tipo de página normalmente retorna os resultados da busca e junto deles exibem os parâmetros de pesquisa dados pelo usuário, ou os inserem na URL de alguma forma. O não tratamento dos valores desses parâmetros, gera vulnerabilidade no sistema.

- **Mensagens de erros**: é comum que aplicações considerem inválidas algumas entradas recebidas, e exibam mensagens de erro ao usuário. Tradicionalmente, tais mensagens exibem quais parâmetros passados pelo usuário causaram tal erro. Se houver esse tipo de mecanismo em sistemas, então ele deve ser cuidadosamente implementado, do contrário é um meio para que ocorram ataques.

- **Submissão de formulários**: normalmente, formulários são enviados ao servidor através de requisições HTTP e posteriormente os dados de entrada retornam de alguma forma aos clientes. Logo, a não filtragem das entradas inseridas em tais campos gera uma falha de segurança.

2.5 Protegendo aplicações contra *cross-site scripting*

Infelizmente não existe um método completamente efetivo para prevenção do *cross-site scripting*, visto que navegadores estão em constante atualização e, a cada mudança, novas vulnerabilidades podem surgir. Porém existem métodos para mitigar o impacto de uma falha de XSS em sua aplicação.

Os principais métodos de prevenção envolvem a sanitização e validação das entradas do usuário e também das respostas que serão enviadas ao mesmo (pelos motivos vistos na seção sobre *reflected* XSS). Como os navegadores implementam o HTML parser de formas distintas, torna-se muito complicado proibir todas as entradas que possivelmente possam acarretar em uma execução de código. Uma abordagem mais interessante seria proibir todos os dados não confiáveis em locais perigosos dentro de um documento HTML (como dentro de uma *tag* <*script*> por exemplo) e então construir uma *whitelist* que permite a inserção de um dado após feita sua validação.

O primeiro passo é mapear todos os locais de sua aplicação onde o usuário possa interagir inserindo dados, em seguida devem ser feitas as validações apropriadas para cada um dos campos de entrada do usuário.

Uma forma de fazer essa validação é escapar todos os caracteres especiais em HTML, codificando-os para seu valor HTML respectivo e assim alterando o seu significado original. Dessa forma o navegador não irá interpretá-los como um caractere especial de HTML mas sim como um caractere de texto comum, fazendo-o com que perca sua funcionalidade em HTML. Por exemplo, o caractere '<' em HTML representa o início de uma *tag*, após a codificação, o caractere será tratado apenas como texto comum e será exibido no navegador. Alguns dos principais caracteres especiais que devem ser sanitizados são mostrados na tabela abaixo:

Tabela 1 - Principais caracteres HTML e seus valores codificados

Caractere	Nome da entidade	Número da entidade
&	&amp;	&#38
<	&lt;	&#60
>	&gt;	&#62
"	&quot;	&#34
'	Não possui	&#39
/	Não possui	&#47

Fonte: (https://www.ascii.cl/htmlcodes.htm)

Figura 2.1: Principais caracteres HTML e seus valores codificados.

Bibliotecas destinadas a sanitização de HTML

Algumas bibliotecas foram criadas com o intuito de facilitar a vida do desenvolvedor na tarefa de validação e sanitização de códigos HTML. Algumas mais conhecidas são listadas a seguir [6]:

- HTML *sanitizer* da *Google Closure Library* (*JavaScript/Node.js, docs*).
- *DOMPurify (JavaScript, requer jsdom para-Node.js)*. Um exemplo da utilização; dessa biblioteca pode ser encontrado na Figura 2.
- PHP HTML *Purifier*.
- Python *Bleach*.

```
const createDOMPurify = require('dompurify');
const { JSDOM } = require('jsdom');

const window = new JSDOM('').window;
const DOMPurify = createDOMPurify(window);

const clean = DOMPurify.sanitize(dirty);
```

Figura 2.2: *DOMPurify* para sanitização de código.

2.6 Conclusão

No presente artigo, entendemos como funciona o processo de um ataque de *CrossSite Scripting* no geral, bem como vimos os métodos mais comuns de ataque: r *Reflected Cross-Site Scripting*, *Stored Cross-Site Scripting* e DOM *Based Sross-Site Scripting*. Por fim foram apresentadas algumas medidas básicas de prevenção contra o *cross-site scripting*.

Também pudemos ver que, apesar de amplamente conhecidos, os ataques de *cross-site scripting* representam uma boa parte dos ataques mais presentes na web. Isso se deve ao fato de que muitos desenvolvedores não dão a devida atenção à segurança de suas aplicações, por isso uma falha tão comum acaba sendo tão recorrente. Por isso é importante que todos se conscientizem da importância de utilizar boas práticas de segurança na hora do desenvolvimento de uma aplicação para que todos possamos ter uma boa convivência na web.

2.7 Referências

[1] OWASP Foundation. OWASP Top Ten Application Security Risks, 2018. Disponível em: *https://www.owasp.org/index.php/Top_10-2017_Top_10*.

[2] The Hacker-Powered security report 2018. Disponível em: *https://www.hackerone.com/resources/hackerpowered-securityreport*.

[3] KLEIN, Amit. DOM Based Cross Site Scripting or XSS of the Third Kind. Disponível em *http://www.webappsec.org/projects/articles/071105.shtml*.

[4] JavaScript HTML DOM. Disponível em: *https://www.w3schools.com/js/js_htmldom.asp*.

[5] B. Mewara, S. Bairwa and J. Gajrani, "Browser's defenses against reflected cross-site scripting attacks"2014 International Conference on Signal Propagation and Computer Technology (ICSPCT 2014), Ajmer, 2014, pp. 662-667.

[6] OWASP, "XSS (Cross Site Scripting) Prevention Cheat Sheet". Disponível em: *https://www.owasp.org/index.php/XSS*.

Capítulo 3

Computação Forense: Aspectos práticos e legislativos dos crimes virtuais

Gabriel da Silva Kyomen
kenkakyomen@hotmail.com
Mateus Grota Nishimura Ferro
mateusnishimura@hotmail.com
Matheus Vargas Volpon Berto
matheusvvb@hotmail.com

Resumo

Este documento procura expor informações e conceitos sobre a Computação Forense e suas aplicações no ambiente criminal. Para isso, relaciona-se os embasamentos dessa ciência com o processo investigativo, além de interligar sua realização com as normas legislativas regentes, destacando-se as principais dificuldades enfrentadas pelos profissionais da área.

Palavras-chave: Computação Forense; Ambiente Criminal; Processo Investigativo; Normas Legislativas.

3.1 Introdução

Desde seu surgimento, o homem criou métodos, processos e ferramentas que proporcionaram maior facilidade na realização de suas tarefas. Com o avanço tecnológico ao longo dos anos, surgiu a "Era Digital", na qual o computador e a internet também passaram a ser usados com este fim. Hodiernamente, a quantidade de informações, dados e serviços que transitam na rede é imensa.

Entretanto, este incalculável acervo de dados – alguns de fácil acesso – atraiu a atenção de criminosos, que se utilizam dos meios tecnológicos para cometer seus delitos. Para evitar tais atos, existem técnicas de proteção de informações, mas, nem sempre isso é possível, necessitando de uma investigação para averiguar o caso. Essa perícia necessita ser especializada, e é neste cenário que surge a chamada "Computação Forense", ciência responsável pela resolução de crimes cibernéticos.

Neste artigo, procura-se explanar sobre as modalidades de tais crimes, assim como seu processo, os métodos de perícia e os possíveis obstáculos que podem surgir ao longo da mesma. Além disso, relaciona-se tais informações à regulamentação legislativa sobre a área da computação forense e à realidade brasileira.

3.2 Desenvolvimento

Modalidades de crimes cibernéticos

As máquinas computacionais e dispositivos digitais atuais podem ser utilizados para beneficiar o crime, segundo duas categorias distintas. Na primeira, tais máquinas são usadas como ferramentas facilitadoras de delitos comuns, ou seja, o computador apenas dá suporte ao ato, mas este poderia ser cometido sem, necessariamente, o uso do dispositivo. Um roubo a uma joalheria, por exemplo, pode ser feito sem o uso de computa-

dores, entretanto, o computador pode facilitar a coleta de dados importantes para os criminosos, como horários da loja, câmeras e informações sigilosas.

Já na segunda categoria, os computadores exercem papéis decisivos no ato criminoso, sendo peça imprescindível para tal. Tais delitos são chamados de "crimes de informática" ou "crimes digitais", e passaram a ocupar, nos últimos anos, o topo da lista de crimes mais comuns, permanecendo inferior apenas ao narcotráfico e à falsificação de marcas e/ou patentes. Neste ramo, há diversas possibilidades de ataques, desde coleta sigilosa de dados, simples manipulações indevidas de imagens, até invasão de sites e banco de dados de grandes empresas internacionais, afetando usuários em larga escala. No Brasil, o alvo deste tipo de delito – segundo a Febraban – é o sistema bancário de grandes instituições. Cerca de 95% das perdas dos bancos são provenientes do crime cibernético, gerando um acúmulo de cerca de US$ 8 bilhões por ano.

Por ter se tornado algo tão frequente e perigoso, surgiram ao longo dos anos estudos e pesquisas que tratam sobre as aspirações e o comportamento de criminosos responsáveis pelos ataques digitais. Entender a motivação de um crime deste tipo é fundamental para, posteriormente, orientar as investigações. A invasão de sistemas computacionais pode ter como objetivos: paralisação total do sistema, disseminação de vírus e ataques distribuídos, roubo de informações pessoais – senhas, números de cartões, entre outros – ou destruição de informações.

Primeiramente, os criminosos escolhem seus possíveis alvos, reunindo as informações sobre os mesmos. Após isso, busca-se por pontos de abertura no sistema a ser invadido e obtenção de dados que permitam a entrada dos agentes criminosos. Neste caso, há mais de uma maneira de se obter tais dados, dentre elas está a técnica de "*sniffing*" – ou análise de pacotes – na qual usa-se um programa para localizar, interceptar e decodificar dados em uma rede digital.

Depois de ter conquistado o acesso ao sistema, o atacante transfere programas maliciosos, buscando fragilidades estruturais que concedam a ele a administração total e completa do sistema – possibilitada pelos comandos de *root*. Posteriormente, vestígios de sua presença são apagados e ferramentas de retorno – *back doors* – são instaladas (tais ferramentas facilitarão o retorno discreto do criminoso ao sistema no futuro).

Os passos enunciados anteriormente refletem o desenvolvimento e execução do "*modus operandi*" da ação, intrinsecamente relacionado ao nível de conhecimentos e habilidades do hacker. Tais capacidades individuais podem ser divididas em grupos, conforme a tabela a seguir:

Nível de habilidade	Habilidades	Evidências
Clueless	Nenhuma habilidade	Todas as atividades são bastante aparentes
Script Kiddie	Capaz de encontrar *exploits* prontos na Internet e executá−los seguindo instruções detalhadas. Não escrevem programas	Pode tentar cobrir rastros com o uso de *rootkits* prontos, mas com sucesso limitado. Pode ser detectado com esforço mínimo
Guru	Equivalente a um administrador experiente. Hábil em programação. Checa a existência de programas de segurança e esquemas de log seguros, evitando alvos protegidos	Cuidadosamente apaga evidências em arquivos de log. Não deixa traços óbvios de sua presença. Pode instalar *trojan horses* e *back doors* para um acesso futuro
Wizard	Possui um g rande conhecimento do funcionamento interno de um sistema. Capaz de manipular *hardware* e *software*	Praticamente não deixa evidências úteis. Pode comprometer totalmente o sistema

Tabela 3.1: Relação entre habilidades dos criminosos e o tipo ou quantidade de evidências deixadas pelos mesmos após o crime [9]

Método de análise forense

Não há maneira única e consolidada de investigar um crime cibernético. Se três diferentes investigadores recebem a mesma tarefa, é normal que eles usem diferentes abordagens. Contanto que o(s) culpado(s) sejam encontrados sem quebrar leis, não importa o processo escolhido, embora alguns sejam mais eficientes. Uma das formas de investigar é baseada nos estudos de Brian D. Carrier e Eugene H. Spafford, nos quais há um crime digital cujo ambiente foi criado por meio de software e hardware. Há três fases principais, citadas na figura 1, que não precisam acontecer em ordem específica.

Essa maneira de investigação não se restringe a sistemas que estejam em funcionamento, podendo utilizar de softwares de confiança para recuperar informações. Inclusive, a análise de dados de sistemas não funcionais, por ser feita em ambiente externo, é mais segura, já que não está sujeita a programas maliciosos que alteram e/ou

escondem informações quando desejado pelo criminoso. Apesar disso, nem sempre é possível fazer análise externamente.

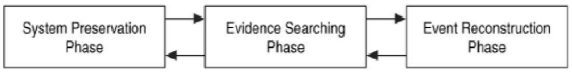


Figura 3.1: As três fases principais de uma investigação de um crime digital [1]

A primeira fase à esquerda da imagem 1 é chamada de "Fase de preservação do sistema". Esta serve para preservar a cena do crime digital e consiste em extrair informações com o mínimo de perda. A execução dessa etapa depende dos requerimentos jurídicos e de negócio da operação, pois pode limitar algumas opções, como o uso de *spywares*, por exemplo. O propósito dessa fase é reduzir a quantidade de informação que será sobrescrita, e para isso, pode ser feita mais de uma cópia dos arquivos iniciais e criadas novas cópias toda vez em que um arquivo é aberto para análise. Desse modo, essa fase perdura durante toda a investigação, já que é preciso saber todas as mudanças feitas durante a busca por evidências. É importante que os dispositivos nos quais é feita a análise não possam ser acessados por ninguém desacompanhado do investigador, principalmente por fontes externas, como uma conexão de internet. Para tanto, podem ser usados protocolos de criptografia e o dispositivo pode ser mantido fisicamente sem conexão à rede.

A "Fase de busca de evidências" normalmente começa por uma pesquisa nos lugares mais comuns de se encontrar dados para aquele tipo de incidente, baseada em documentações de eventos anteriores e parecidos, se já existiram. Por exemplo, se foi um crime on-line, são analisados primeiro os *cookies*, histórico e cache dos navegadores. Com as informações coletadas ao longo do processo, são formadas hipóteses, e então ocorrem novas buscas atrás de dados que suportem ou descartem essas proposições. É necessário que o corpo de investigadores se atente para não se prender apenas à busca de evidências que comprovem uma hipótese, pois pode ser que esta esteja errada, apesar das indicações em qualquer parte do processo.

Finalmente, a "Fase de reconstrução de eventos" é relevante para analisar os eventos que antecederam e aqueles ocorridos durante o crime. Por exemplo, não é uma boa opção analisar os arquivos como se o autor de todos eles fosse o próprio usuário já que muitos podem ter sido baixados ou passados via pendrive ou CD. Além disso, mesmo aqueles cuja propriedade é do usuário, podem ser criados automaticamente por softwares, e isso pode ser usado para defender o acusado. Para que essa fase ocorra com êxito, os peritos devem conhecer amplamente o sistema operacional do(s) dispositivo(s) utilizados e também dos programas que estão instalados nele(s).

Dificuldades e desafios

O maior problema na investigação computacional forense está nos inúmeros obstáculos conhecidos comumente pelo termo "Anti-Forense". Entre eles estão o uso de senhas, criptografia e esteganografia. Além disso, com o imparável avanço da tecnologia, o volume de dados cresce exponencialmente.

A quantidade de dados, como já citado, cresceu de maneira inimaginável nas últimas duas décadas. Sistemas que contavam com megabytes (MB) de dados, hoje apresentam até mesmo terabytes (TB), e a quantidade de arquivos pode ser exorbitante. Para viabilizar a análise de tantas informações, deve-se ter clareza e especificidade nos quesitos da elaboração do laudo, ou seja, aquele que solicita o exame deve saber bem o que está procurando, evitando perguntas vagas. Além disso, deve-se aplicar filtros e buscar palavras-chave para que mais arquivos possam ser removidos da fila para análise, acelerando o processo.

É bem comum que arquivos e programas possuam senhas, impedindo o acesso a possíveis evidências. Para alcançar essas informações, os peritos devem conhecer técnicas de quebra de senha, como ataque de força bruta, no qual um domínio de checagem é criado e este busca todas as possibilidades de senhas baseado em parâmetros, como tamanho da senha, existência de números, caracteres especiais, etc. Esse mesmo método pode ser aprimorado com o uso de um dicionário, ou seja, palavras e padrões de caracteres que talvez estejam presentes na senha. Esse dicionário pode ser construído com base em casos anteriores, senhas mais utilizadas e até mesmo informa-

ções do investigado, como datas e nomes importantes, e, durante as tentativas, termos do dicionário são combinados de formas comuns para ter uma maior chance de acerto.

O termo criptografia tem origem grega e significa "escrita escondida". A criptografia consiste em usar algoritmos matemáticos para transformar mensagens em texto ilegível para qualquer um que não seja o remetente ou o destinatário. Essa técnica é amplamente utilizada pois, mesmo com a mensagem passando por locais inseguros, é impossível ler diretamente. Esses algoritmos podem ser de sentido único, ou seja, há encriptação, mas não há como decriptar, sendo usada apenas para ocultar informação, ou de sentido duplo, no qual existem chaves, permitindo troca de mensagens/informações entre duas pontas.

Para quebrar tal barreira, o perito pode utilizar tabelas chamadas de *Rainbow Tables*, que possuem *hashes* pré-compilados. Porém, quanto maior a senha e mais diferentes tipos de caracteres, mais inviável essa opção se torna, demandando muito poder computacional e tempo. O perito também pode manter a atenção em softwares nos dispositivos investigados que possuam essa função, realizando testes até entender qual o algoritmo utilizado para tal tarefa, diminuindo o tempo para quebra do segredo.

Esteganografia também é um termo vindo do grego com sentido de "escrita encoberta" e difere da criptografia por ser um conjunto de técnicas para colocar uma mensagem no meio de outra, camuflada. O interessante desse método é que ele não chama a atenção como um arquivo criptografado ou com senha, já que a informação está disposta abertamente, porém, de modo que pareça fazer parte daquele determinado dado, levantando o mínimo possível de suspeita. Pode-se encontrar o uso de esteganografia no mundo comercial como maneira de proteger os direitos das empresas, como no caso da HP e Xerox, que adicionam pontos amarelos minúsculos nas páginas impressas por seus dispositivos em que estão descritos o número de série do equipamento e informações básicas sobre a impressão, como data e hora.

Dependendo do nível de complexidade da técnica, os criminosos podem esconder muito bem grandes evidências de suas ações, dificultando o trabalho do investigador, que, mais uma vez, pode procurar programas dentro dos dispositivos dos acusados que realizam automaticamente essa tarefa para descobrir a técnica e, assim, descobrir as informações encobertas.

Legislação

Por ser uma ciência que visa reportar suas análises e resultados a alguma instância da justiça, a Computação Forense tem relação estreita com a legislação.

Algumas leis devem ser de conhecimento do perito, como: a do exame do corpo de delito e das perícias em geral (art. 158 e 159 do Código de Processo Penal), art. 240 e art. 241 do ECA, Marco Civil da Internet e a Lei Carolina Dieckmann.

Os artigos 158 e 159, garantem o exame pericial em processos criminais e explicitam a obrigatoriedade da perícia em crimes que deixam vestígios:

Art. 158. Quando a infração deixar vestígios, será indispensável o exame de corpo de delito, direto ou indireto, não podendo supri-lo a confissão do acusado.

Art. 159. O exame de corpo de delito e outras perícias serão realizados por perito oficial, portador de diploma de curso superior.

§ 1o Na falta de perito oficial, o exame será realizado por 2 (duas) pessoas idôneas, portadoras de diploma de curso superior preferencialmente na área específica, dentre as que tiverem habilitação técnica relacionada com a natureza do exame. (Redação dada pela Lei nº 11.690, de 2008)

§ 4º O assistente técnico atuará a partir de sua admissão pelo juiz e após a conclusão dos exames e elaboração do laudo pelos peritos oficiais, sendo as partes intimidadas desta decisão.

§ 5º Durante o curso do processo judicical, é permitido às partes, quanto à perícia:

I - requerer a oitiva dos peritos para esclarecerem a prova ou para responderem a quesitos, desde que o mandato de intimação e os quesitos ou questões a serem esclarecidas sejam encaminhados com antecedência mínima de 10 (dez) dias, podendo apresentar as respostas em laudo complementar;

II - indicar assistentes técnicos que poderão apresentar pareceres em prazo a ser fixado pelo juiz ou ser inquiridos em audiência.

§ 6º Havendo requerimento das partes, o material probatório que serviu de base à perícia será disponi-

bilizado no ambiente do órgão oficial, que manterá sempre sua guarda, e na presença de perito oficial, para exame pelos assistentes, salvo se for impossível a sua conservação.

O art. 159 esclarece pontos importantes em relação ao perito como a necessidade de um curso superior e também o direito das partes de requerer o esclarecimento das provas.

É importante ressaltar que as partes podem indicar seus próprios peritos, denominados assistentes técnicos, que podem auxiliar em questões que demandam conhecimentos específicos no processo. Ademais, os assistentes técnicos podem apontar possíveis erros nos laudos realizados pelo perito indicado pelo juiz.

Outra lei relacionada com a perícia em informática, é a que trata do crime de pedofilia, descrito nos artigos 240 e 241 do ECA.

Art. 240. Produzir, reproduzir, dirigir, fotografar, filmar ou registrar, por qualquer meio, cena de sexo explícito ou pornográfica, envolvendo criança ou adolescente: Pena - reclusão, de 4 (quatro) a 8 (oito) anos, e multa.

§ 1º Incorre nas mesmas penas quem agencia, facilita, recruta, coage, ou de qualquer modo intermedeia a participação de criança ou adolescente nas cenas referidas no caput deste artigo, ou ainda quem com esses contracena.

Art. 241. Oferecer, trocar, disponibilizar, transmitir, distribuir, publicar ou divulgar por qualquer meio, inclusive por meio de sistema de informática ou telemático, fotografia, vídeo ou outro registro que contenha cena de sexo explícito ou pornográfica envolvendo criança ou adolescente: (Incluído pela Lei nº 11.829, de 2008) Pena - reclusão, de 3 (três) a 6 (seis) anos, e multa. (Incluído pela Lei nº 11.829, de 2008)

§ 1º Nas mesmas penas incorre quem: (Incluído pela Lei nº 11.829, de 2008)

I - assegura os meios ou serviços para armazenamento das fotografias, cenas ou imagens de que trata o caput deste artigo; (Incluído pela Lei nº 11.829, de 2008)

II - assegura, por qualquer meio, o acesso por rede de computadores às fotografias, cenas ou imagens de que trata o caput deste artigo. (Incluído pela Lei nº 11.829, de 2008)

§ 2º As condutas tipificadas nos incisos I e II do § 1º deste artigo são puníveis quando o responsável legal pela prestação do serviço, oficialmente notificado, deixa de desabilitar o acesso ao conteúdo ilícito de que trata o caput deste artigo. (Incluído pela Lei nº 11.829, de 2008)

O papel do especialista em computação forense nos crimes descritos é encontrar imagens que podem estar escondidas, apagadas ou criptografadas e que, caso sejam encontradas, enquadram o investigado no art. 240. O próximo passo é determinar se houve a distribuição ou compartilhamento dessas imagens com outros usuários, constituindo um crime mais grave. O compartilhamento dessas imagens é feito principalmente por aplicativos peer-to-peer.

Fora do crime da pedofilia, existe o Marco Civil da Internet, lei nº 12.965 de 23 de Abril de 2014, que rege normas, garantias, princípios, direitos e deveres para a utilização da internet no país por usuários, empresas e provedores de internet. Alguns pontos que são de interesse da computação forense são os que regular o armazenamento dos registros de acessos (logs) dos usuários.

Alguns princípios do MCI relacionados à área criminal digital são: privacidade na web e o registro de acessos. Fica assegurado, por este documento, a inviolabilidade e o sigilo da troca de informações entre os usuários, porém, pode haver a quebra do sigilo perante uma intimação judicial. Os dados dos usuários estão sob tutela dos provedores de internet e, em caso de descumprimento da lei, os mesmos estarão sujeitos a penalidades.

O armazenamento de dados e a sua proteção é responsabilidade do provedor de internet que tem como dever guardá-los por, no mínimo, um ano, podendo haver requerimento para que os registros sejam guardados por prazo superior.

Art. 10. A guarda e a disponibilização dos registros de conexão e de acesso a aplicação de internet de que trata esta Lei, bem como de dados pessoais e do conteúdo de comunicações privadas, devem atender á preservação da intimidade, da vida privada, da honra e da imagem das partes direta ou indiretamente envolvidas.

Art. 13. Na provisão de conexão à internet, cabe ao administrador de sistema autônomo respectivo o dever de manter os registros de conexão, sob sigilo, em ambiente controlado e de segurança, pelo prazo de 1 (um) ano, nos termos do regulamento.

§ 2º A autoridade policial ou administrativa ou o Ministério Público poderá requerer cautelarmente

que os registros de conexão sejam guardados por prazo superior ao previsto no caput.

Art. 22. A parte interessada poderá, com propósito de formar conjunto probatório em processo judicial cível ou penal, em caráter incidental ou autônomo, requerer ao juiz que ordene ao responsável pela guarda o fornecimento de registros de conexão ou de registros de acesso a aplicações de internet.

Além do Marco Civil, foi criado a lei nº 12.737, de 30 de Novembro de 2012, conhecida como "lei Carolina Dieckmann", que trouxe para o ordenamento jurídico o crime novo de Invasão de Sistemas, sendo papel do perito identificar o autor da invasão e os danos causados pelo invasor.

Art. 1º Esta Lei dispõe sobre a tipificação criminal de delitos informáticos e dá outras providências. "Invasão de dispositivo informático"

Art. 154-A. Invadir dispositivo informático alheio, conectado ou não à rede de computadores, mediante violação indevida de mecanismo de segurança e com o fim de obter, adulterar ou destruir dados ou informações sem autorização expressa ou tácita do titular do dispositivo ou instalar vulnerabilidades para obter vantagem ilícita: Pena - detenção, de 3 (três) meses a 1 (um) ano, e multa.

§ 1º Na mesma pena incorre quem produz, oferece, distribui, vende ou difunde dispositivo ou programa de computador com o intuito de permitir a prática da conduta definida no caput.

A computação forense no Brasil

Já no Brasil, não há uma regulamentação precisa e que suporte profissionais da Computação Forense, estes realizam apenas algumas tarefas expedidas e coordenadas pela Polícia Federal. Em muitos casos, além do não requerimento dos trabalhos destes profissionais, a resolução dos mesmos pode não ser fácil, principalmente com o uso de ferramentas que dificultam o rastreamento de evidências, como VPN's.

O método de análise forense se baseia em três princípios:

- **Réplicas**: realização de duplicação do laudo, para que haja a possibilidade de repetição de ações sobre o acervo, sem que ocorra dano real à evidência;
- **Integridade**: procedimentos que garantam a integridade dos vestígios coletados devem ser realizados previamente, por meio de criptografia;
- **Ferramentas de confiabilidade**: uso de recursos computacionais que entreguem ao profissional resultados concretos, permitindo a sequência e desenvolvimento da investigação.

A questão que diferencia um perito forense de um perito comum é a falta de um "protocolo de procedimento", em outras palavras, não há uma regularidade ou padrão em suas tarefas, pois o processo de obtenção das informações presentes nos aparelhos digitais pode nunca ser o mesmo. Portanto, qualquer atitude não premeditada pode colocar em risco os dados e evidências que se desejava obter, prejudicando imensamente a investigação.

Dessa forma, um perito computacional deve sempre estar envolvido com as novas ferramentas e tecnologias usadas pelos criminosos, para que possa agir de maneira correta para determinada situação. Entretanto, para que isso seja possível, é necessário investimento pesado e constante em estudos e pesquisas na área, fazendo com que países menos desenvolvidos, como o Brasil, tenha dificuldades em tal feito.

3.3 Considerações finais

O computador que teve seu surgimento em 1946 e foi criado com o objetivo bélico de fazer cálculos balísticos. Posteriormente, com o advento da internet, revolucionou a comunicação e a transmissão de informação.

A velocidade do processamento e transmissão de dados por toda a rede fez com que o computador se tornasse um item essencial em muitas empresas e casas de todo o mundo. Segundo o IDC Brasil, foram vendidos cerca de 1,5 milhões de máquinas apenas em 2019. Muitos serviços de empresas e do governo estão sendo disponibilizados na internet com o intuito de facilitar e desburocratizar a vida do cidadão. Porém, esse grande volume de informações pessoais atraiu o interesse de criminosos virtuais com novas práticas ilegais.

A prevenção é a melhor maneira de combater os ataques, além de dispor de equipamentos e sistemas de alta tecnologias, deve-se conscientizar as

pessoas a utilizá-los com segurança, evitando ataques de engenharia social. Entretanto, quando a prevenção é ineficaz e há uma invasão ao sistema, é necessário a investigação e apuração dos vestígios, na tentativa de elucidar o fato ocorrido e descobrir o invasor.

A Computação Forense surge como meio para a análise e coleta desses vestígios. Partindo-se dessa visão, este artigo apresentou os principais tópicos relacionados a esta área, como: modalidades dos crimes cibernéticos, método de análise, aspectos jurídicos e a dificuldade do perito forense computacional no Brasil.

Os criminosos não estão alheios à revolução computacional, havendo um número crescente de crimes que fazem o uso de computadores e celulares, podendo ser usado como um meio para a realização do delito, por exemplo: ameaças e compartilhamento de material ilegal. Ademais, eles também podem ser utilizados como meio de armazenamento de provas, como, por exemplo, planilhas financeiras, que podem servir como indícios de um crime de lavagem de dinheiro.

O aumento de crimes virtuais torna notório a necessidade de investimento e estruturação de setores investigativos especializados em combater crimes cibernéticos, além do treinamento do atual contingente policial de como obter e utilizar vestígios eletrônicos a fim de solucionar um crime. Esses rastros podem ser arquivos de *log* de rede de computadores, e-mails, arquivos de textos e imagens, sendo essenciais para a investigação.

Infere-se, portanto, que a computação é uma área do conhecimento que está em constante evolução, é imprescindível que os peritos computacionais se mantenham atualizados com o desenvolvimento técnico-científico e mantenham um constante desejo de aprendizado, além da necessidade da aprimoramento das leis e normas regulatórias brasileiras, para que possam ser capazes de solucionar todos os novos crimes da Era Moderna.

3.4 Bibliografia

[1] Brian CARRIER. File system forensics analysis, 2005.

[2] Daniel Moraes da COSTA. Boas práticas para perícia forense, 2008.

[3] Rafael Narder de Almeida. Perícia forense computacional: estudo das técnicas utilizadas para coleta e análise de vestígios digitais, 2011.

[4] Pedro M. da Silva ELEUTÉRIO and Marcio P MACHADO. Desvendando a computação forense 1. ed, 2011.

[5] REVISTA GESTÃO EM FOCO. As dificuldades do profissional forense no brasil 9. ed., 2017.

[6] JUS. Os crimes cibernéticos e a lei nº 12.737/2012, 2015. [acessado em 20 nov. 2019]. URL: https://jus.com.br/artigos/35796/os-crimes-ciberneticos-e-a-lei-n-12-737-2012-lei-car

[7] LEC. Entenda a importância do marco civil, 2019. [acessado em 20 nov. 2019]. URL: http://www.lecnews.com.br/blog/entenda-a-importancia-do-marco-civil-da-internet/.

[8] Celso Carlos Navarro MODESTO JUNIOR and MOREIRA Jander. Roteiro investigativo em perícia forense computacional de redes: estudo de caso, 2014.

[9] Paulo Lício de REIS, Marcelo Abdalla dos; GEUS. Análise forense de instrusões em sistemas computacionais: Técnicas, procedimentos e ferramentas.

[10] Wagner de Paula RODRIGUES. Análise pericial em sistema operacional, 2004.

[11] S TEELINK and R. F. ERBACHER. Improving the computer forensic analysis process through visualization, 2006.

Capítulo 4

Jenga Builder - Robô Construtor de Jenga

Ágata Rangei Drigo
Anderson Pinheiro Garrote
André Matheus Bariani Trava
Thiago Yussuki Uehara
Gabriel Eiji Uema Martin

Resumo

Este relatório descreve o planejamento da construção de um robô capaz de construir uma torre de Jenga. O robô utiliza um sistema de movimentação com três eixos lineares que movem uma garra responsável por carregar as peças da torre. As ações executadas pelo robôs são determinadas por um controlador que leva em consideração um ambiente controlado, estático e não observável.

4.1 Introdução e motivação

Esse relatório tem como objetivo apresentar e explicar sobre o projeto de robótica desenvolvido na disciplina de Introdução a Robótica. O projeto visa a elaboração de um robô construtor de torres de Jenga, um jogo de habilidade física em que jogadores devem retirar uma peça por vez de uma torre previamente construída. Espera-se, com o desenvolvimento desse projeto, estudar mais profundamente os fundamentos da robótica e as práticas da área em um projeto semelhante aos robôs utilizados na industria atual.

A proposta do projeto e desenvolver e construir um robô capaz de construir torres de Jenga. Uma torre de Jenga e composta de peças padronizadas dispostas três a três em níveis, sendo que cada nível e rotacionado 90º em relação ao anterior, conforme a figura 1.

Figura 4.1: Torre de Jenga

4.2 Funcionalidades

O robô deve ser capaz de movimentar a garra em três eixos diferentes, num envelope de trabalho de 20x20x30cm.

A garra deve ser capaz de segurar uma peça de Jenga pelos seus dois lados menores, conforme ilustrado na figura 2.

Figura 4.2: Modelo 3D da garra segurando a peça de Jenga

O robô consegue movimentar as peças de Jenga que são colocadas na posição (0,0,0) da estrutura de madeira, escolhida arbitrariamente como um dos cantos da estrutura, até o ponto central na posição (X, Y, Z): o X corresponde a altura atual da torre; o Y é o meio da estrutura em relação ao comprimento com alguma variação dependendo do posicionamento da peça; e o Z é o meio da estrutura em relação a sua largura, variando em valor dependendo da posição da peça.

4.3 Componentes e partes

Esse robô será composto de três eixos lineares que movimentam uma garra. A maioria das peças foram organizadas e elaboradas em uma ferramenta de projetos em 3D. Para a construção desses, serão necessários os seguintes componentes:

Arduino Uno

Figura 4.3: Arduino Uno

A placa de prototipagem Arduino Uno é o microntrolador do robô, sendo responsável por transformar o código do controlador em instruções para os motores.

Breadboard e Jumpers

A *Breadboard* é a estrutura onde será montado o circuito do robô, e os *Jumpers* são os cabos que serão conectados à ela e aos motores. Ao todo são usados 20 cabos macho-fêmea e 6 cabos macho-macho.

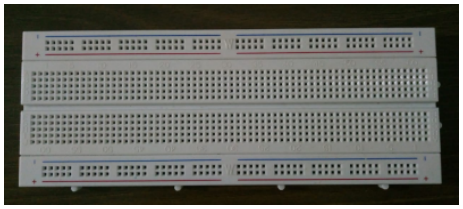


Figura 4.4: Breadboard

Figura 4.5: Jumpers

Motores

Ao todo serão usados 6 motores, sendo 4 motores de passo 28BYJ-48 (figura 6) incluindo o Driver ULN2003 (figura 7) e 2 motores servo 9g SG90 TowerPro (figura 8). Os motores de passo são responsáveis pela movimentação dos eixos enquanto os motores servo são responsáveis pelos movimentos de abertura e giro da garra.

Figura 4.6: Motor de passo 28BYJ-48

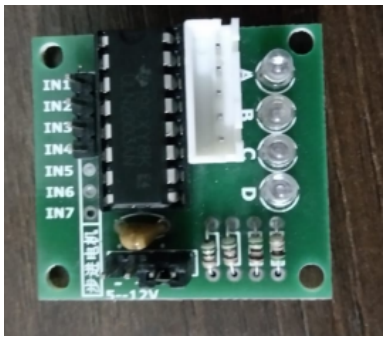


Figura 4.7: Driver ULN2003

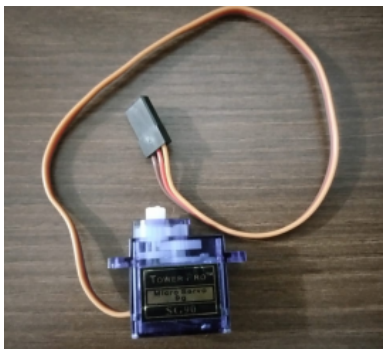


Figura 4.8: Motor servo 9g SG90 TowerPro

Garra

A garra do robô será construída baseada em um projeto público [jirobots 2019] disponível na plataforma *Thingverse*.

O projeto consiste em três modelos: a base (figura 9), o lado esquerdo (figura 10), e o lado direito (11).

Figura 4.9: Base

Figura 4.10: Lado esquerdo

Figura 4.11: Lado direito

Ambos os motores servo serão fixados à base, um deles será responsável por girar um dos lados da garra, que através dos dentes irá girar o outro lado (que estará fixado à base através de um rolamento, abrindo assim a garra). O outro motor será responsável por girar a garra.

Para fechar a garra será utilizado um elástico, assim não será necessário o uso contínuo do primeiro motor, evitando sobreaquece-lo.

Eixos

Os eixos do robô serão construídos baseados em um projeto público de uma *Plotter* [TGit-Tech 2017] disponível na plataforma *Thingverse*.

A movimentação em cada um dos eixos será realizada por um ou mais motores de passo onde estão fixadas engrenagens, que ao girarem se movem sobre uma barra dentada conforme ilustrado na figura 12.

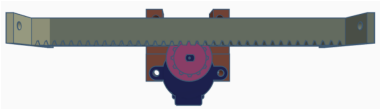


Figura 4.12: Eixo Y

Para possibilitar a movimentação em um segundo eixo, a estrutura anterior será fixada à duas estruturas semelhantes que a movimentarão conforme ilustrado na figura 13.

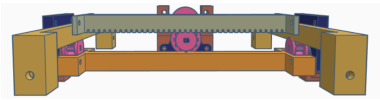


Figura 4.13: Eixo X

Para o último eixo será fixada uma nova estrutura à estrutura anterior. No entanto essa estrutura terá a engrenagem fixa, assim movimentará a barra dentada onde a garra será fixada, conforme ilustrado na figura 14.

Os eixos estarão encaixados sobre uma estrutura de madeira para terem a altura, estabilidade e suporte necessários, conforme ilustrado na figura 15.

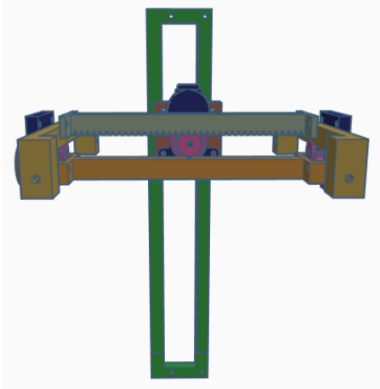


Figura 4.14: Eixo Z

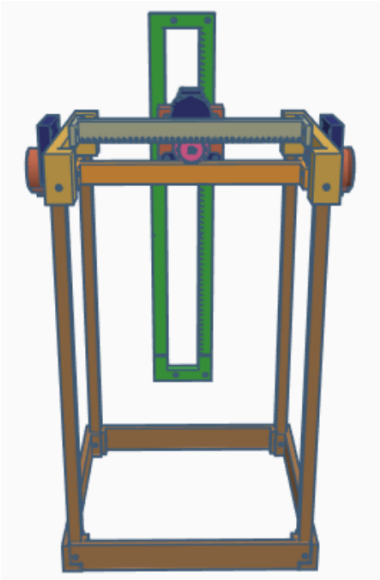


Figura 4.15: Estrutura de madeira

4.4 Arquitetura do sistema

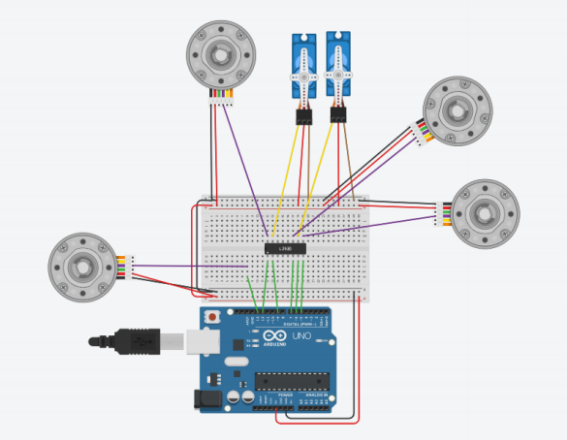


Figura 4.16: Arquitetura do sistema

As figuras redondads em cinza representam os motores de passo, as figuras longas azuis claro são os motores servos, o retângulo branco é o *breadboard* e o retângulo azul-escuro é o Arduino Uno.

4.5 Experimento e resultados esperados

O robô pegará as peças em uma posição prédefinida utilizando a garra, a levará até o local da próxima peça da torre (evitando a colisão com as peças já posicionadas) e a girará se necessário.

O robô não mantém um estado interno das posições das peças colocadas, o seu ambiente é limitado à posição inicial da peça, à posição da garra e à posição final da peça, portanto depende de ajuda externa (para sempre ter uma peça disponível na posição inicial) e não é capaz de corrigir falhas ou imprevistos.

É esperado que o robô seja capaz de montar pelo menos 3 níveis da torre precisamente. Uma quantidade maior de níveis é possível, mas provavelmente terão menor precisão, uma vez que a acurácia do posicionamento das peças dos níveis anteriores se acumula ao longo da construção.

4.6 Conclusão

Com o desenvolvimento do projeto, apesar de que, no momento de escrita, ele não esteja completo, conclui-se que o robô apresentará uma velocidade de montagem reduzida, sendo mais lento que os próprios jogadores montarem a torre.

Além disso, acredita-se ser possível a ampliação do projeto, em específico a movimentação de peças entre dois pontos diferentes, sendo necessária apenas alterações no código e alterações na estrutura do robô para que suporte peças mais pesadas e torres maiores.

E, apesar de um construtor de Jenga não ser útil no dia-a-dia, o projeto ajudou os membros a conhecerem o funcionamento de uma placa Arduino, bem como o uso de *jumper cables* e motores. Além disso possibilitou o contato dos participantes com modelagem e impressão 3D. Dessa maneira, o projeto ajudou de forma geral a aplicar o conhecimento sobre robótica adquirido durante o curso.

4.7 Bibliografia

[1] jjrobots. Robotic gripper, 2019. Acessado em 28 nov. 2019. URL: `https://www.thingiverse.com/thing:3648782`.

[2] TGit-Tech. Rpieces-tabletopxyz, 2017. Acessado em 28 nov. 2019. URL: `https://www.thingiverse.com/thing:2166533`.

Capítulo 5

Panorama da Cultura Open Source e seus Relacionamentos com a Cultura Hacker e a Sociedade

Anderson Pinheiro Garrote

Resumo

Este artigo apresenta uma visão geral sobre o termo e a cultura open source, descrevendo o que é, suas características, um breve histórico e as áreas que é empregada. A seguir são abordadas as relações entre os princípios open source e a cultura hacker e como eles também estão presentes na sociedade contemporânea. São então descritos componentes éticos e legais da cultura open source e exemplos práticos de aplicações baseadas neste modelo.

Palavras-chave: Open Source; Hacker; Sociedade.

5.1 Introdução

O termo open source está ganhando cada vez mais espaço em diversos meios, como o científico [1], na indústria e no mercado, com grandes empresas como Microsoft e IBM tomando iniciativas em direção ao desenvolvimento de aplicações com o código-fonte aberto e disponível [2][3]. Por traz do termo, existe uma cultura de desenvolvimento diferente cuja origem está ligada ao movimento hacker [4]. Este artigo busca sintetizar aspectos gerais relativos a essa cultura, ressaltar características que a associam ao movimento hacker, abordar questões éticas e legais associadas, assim como trazer exemplos práticos e atuais do uso do código aberto.

5.2 O que é Open Source

O termo open source foi criado em 1998 por Christine Peterson, em função do lançamento do código- fonte do navegador NetScape. O conceito foi uma derivação do movimento social free software, conceitualmente semelhante mas mais voltado para atuação político-filosófica [5]. Atualmente, a iniciativa Open Source Initiative (OSI) é responsável por divulgar e resguardar a cultura do código aberto, que prega a superioridade desse processo de desenvolvimento [6]. Dizer que um produto, software ou hardware é open source vai além de apenas disponibilizar o código fonte ao público, pois também significa permitir que ele seja livremente acessado, usado, modificado e compartilhado, independente do usuário [6].

5.3 Características

A OSI estabelece dez princípios para que um software seja considerado open source [9] :

1. Redistribuição Livre;
2. Código Fonte disponível;

3. Permissão para o desenvolvimento de trabalhos derivados;
4. Garantia de integridade do código fonte do autor;
5. Não discriminação de pessoas ou grupos;
6. Não discriminação contra campos de aplicação do software;
7. Distribuição da licença;
8. A licença não deve ser específica para um produto;
9. A licença não pode impedir o uso de outros softwares;
10. A licença deve ser neutra em relação à tecnologia.

Além disso, OSI também define que o termo open source ser empregado apenas sob softwares que estejam sob uma das licenças disponíveis em seu website, para garantir consistência [7]. Entretanto na prática o termo é empregado em outros contextos, como no caso do Arduíno, que é descrito como um hardware open source, apesar de estar sob uma licença Creative Commons [8].

5.4 Relação com a Cultura Hacker

O surgimento e a base filosófica da cultura do software aberto está diretamente conectada com o meio hacker, que defende que um problema não deve ser resolvido mais de uma vez e que a liberdade é primordial, características garantidas por aplicações abertas [4].

Por esse motivo, é muito comum encontrar hackers que usem, desenvolvam e apoiem aplicações open source, que permitem uma atuação de forma colaborativa e criativa, sem limites impostos por aplicações proprietárias

5.5 Relação com o Mercado de Software

Uma das principais razões de existirem licenças e organizações envolvidas na cultura open source é a padronização e a possibilidade de atingir ainda mais usuários, mesmo que não tão inseridos na cultura hacker. Mas hoje algumas empresas já adotam uma postura de que ter o código aberto não é uma coisa tão inviável comercialmente.

Recentemente, tivemos dois casos muito interessantes de empresas muito influentes do ramo da tecnologia, com cultura de desenvolvimento proprietário, buscando maneiras e ferramentas que seguem a filosofia open source.

Em 2018, por exemplo, a Microsoft comprou a plataforma GitHub, onde são hospedados repositórios de milhares de programas de código aberto, além de disponibilizar o código fonte de algumas de suas aplicações nesse mesmo local [2].

Outro acontecimento similar aconteceu em outubro de 2018, quando a IBM adquiriu a Red Hat, uma das maiores empresas de código aberto, para potencializar sua área de computação em nuvem [3]. Também é um exemplo de como o mercado caminha cada vez mais para a adoção de tecnologias

5.6 Questões Éticas e Legais

Historicamente, já houve muita discussão sobre a ética da cultura open source, principalmente se compararmos com o movimento free software, mantido pela Free Software Fundation (FSF) [11]. Ambos defendem o desenvolvimento de programas com código aberto e livre para modificações, mas divergem em alguns aspectos, pois a FSF entende que é necessária uma atuação mais incisiva e política na defesa do código livre, enquanto a OSI entende que é mais importante mostrar como o código aberto é uma alternativa melhor de desenvolvimento software.

Contudo, grande parte das aplicações e licenças são comuns às duas definições. De forma geral, as implicações legais de desenvolver, usar, modificar ou redistribuir um software open source estão descritas em suas respectivas licenças. Algumas licenças disponíveis no site da OSI são:

- Apache License 2.0 (Apache-2.0)
- 3-clause BSD license (BSD-3-Clause)
- 2-clause BSD license (BSD-2-Clause)
- GNU General Public License (GPL)

- GNU Lesser General Public License (LGPL)
- MIT license (MIT)
- Mozilla Public License 2.0 (MPL-2.0)
- Common Development and Distribution License version 1.0 (CDDL-1.0)
- Eclipse Public License version 2.0

Dentro delas são detalhados os limites de uso, exigências e condições de uso do software e sua modificação.

5.7 Exemplos de Aplicações

Seguem abaixo alguns exemplos de aplicações open source [11]:

- Apache OpenOffice;
- Code::Blocks;
- Notepad++;
- OpenCV.

5.8 Referências

[1] M.J.L. de Hoon, * , S. Imoto 1 , J. Nolan e S. Miyano.**Open source clustering software.** BIOINFORMATICS APPLICATIONS NOTE, Vol. 20 no. 9 2004, páginas 1453–1454.

[2] G1. **Microsoft compra GitHub por US$ 7,5 bilhões e anuncia mudanças.** Disponível em: *https://g1.globo.com/economia/tecnologia/noticia/microsoft-compra-github-por-us-75-bilhoes.ghtml*

[3] EXAME. **IBM compra a produtora de software Red Hat por US$ 34 bilhões.** Disponível em: *https://exame.abril.com.br/negocios/ibm-compra-aprodutora-de-software-red-hat-por-34-bilhoes-de-dolares/*

[4] RAYMOND, Eric Steven. **How To Become A Hacker.** Disponível em: *http://www.catb.org/esr/faqs/hacker-howto.html*

[5] OPEN SOURCE INITIATIVE. **History of the OSI.** Disponível em: *https://opensource.org/history*

[6] OPEN SOURCE INITIATIVE. **About the Open Source Initiative.** Disponível em: *https://opensource.org/about*

[7] OPEN SOURCE INITIATIVE. **Frequently Asked Questions.** Disponível em: *opensource.org/faq*

[8] ARDUINO. **Frequently Asked Questions.** Disponível em: *https://www.arduino.cc/en/Main/FAQ*

[9] OPEN SOURCE INITIATIVE. **The Open Source Definition**.Disponível em: *https://opensource.org/osd*

[10] FREE SOFTWARE FUNDATION. Disponível em: *https://www.fsf.org*

[11] SOURCE FORGE. **Browse Open Source Software.** Disponível em: *https://sourceforge.net/directory/*

Capítulo 6

Segurança em robôs domésticos

Gabriel P. Andrade
gabrielperesdeandrade@hotmail.com
Daniel L. Júnior
daniel.dlj@hotmail.com

Resumo

Robôs que sempre estiveram presentes em filmes e séries de ficção cientifica após ganhar seu espaço em mercados militares e industriais, também começaram a marcar presença em alguns mercados de produtos para domicilio. No entanto apesar de trazer grande ajuda, até mesmo tornando-se indispensáveis em algumas situações, essas maquinas geralmente não estão completamente seguras para os usuários principalmente se os mesmos estiverem conectados de alguma forma com a rede ou a internet. Este artigo visa explorar algumas características que estão sendo esquecidas por muitas empresas ao comercializarem esses robôs e que podem acabar causando danos.

6.1 Introdução

Os Robôs antes vistos como elementos futurísticos e fora da realidade são partícipes cada vez mais do nosso cotidiano. Antes normalmente empregados em funções militares e industriais aos poucos vão ganhando força no cenário doméstico. Em restaurantes, já encontramos garçons, cozinheiros e plataformas de pedidos automatizadas, em algumas lojas, há atendentes robôs e não seria diferente em nossas casas, nesse sentido, há robôs integrando parte dos habituais eletrodomésticos, robôs cuidadores que auxiliam crianças e idosos ou até mesmo fazendo parte do nosso corpo como braços robóticos. Todos com a proposta de facilitar nossas atividades diárias ou nos entreter, no entanto, muitas pessoas ainda questionam se esse tipo de tecnologia é segura e confiável, mas isso não parece preocupar a indústria que continua crescendo.

Em média, há 69 robôs para cada 10 mil profissionais na indústria. Na Coreia do Sul são 531 robôs a cada 10 mil trabalhadores, Singapura e Japão são 398 e 305 robôs, respectivamente, para cada 10 mil trabalhadores. O gráfico a seguir disponibilizado pela Veja[10] pode-se notar a densidade robótica em 10 países e maior ainda é a diferença quando comparada ao mesmo gráfico da Folha de São Paulo[11]. O Brasil, por sua vez, possui 10 robôs a cada 10 mil trabalhadores, muito abaixo da média mundial.

6.2 Potencial robótico

A robótica vem evoluindo com o desenvolver de novas tecnologias e com isso facilitando alguns trabalhos antes desempenhados por humanos e/ou que eram considerados perigosos, tais como trabalhos manuais em indústrias ou serviços militares. No contexto militar existem diversos modelos de robôs que são mais empregados em questões que oferecem risco a humanos como: transporte, reconhecimento, vigilância, detonação e desarmamento de minas explosivas [8]. Na indústria, por sua vez, são mais usados para aumentar a produtividade através de movimentos rápi-

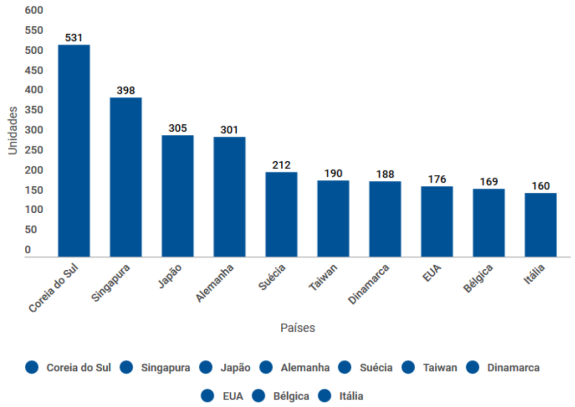


Figura 6.1: 10 países com maior densidade Robótica 2017

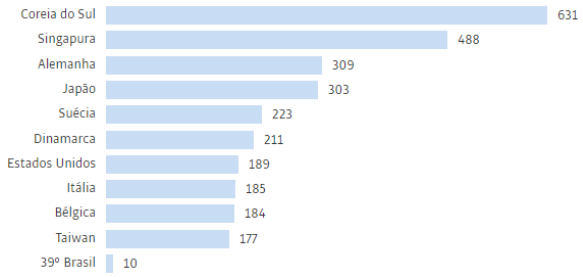


Figura 6.2: 10 países com maior densidade Robótica e Brasil 2018

dos, eficazes e precisos. Para exercer essas funções, essas máquinas precisam em sua maioria de sensores e atuadores que realizarão a percepção e interagirão com o ambiente em que estiverem. Existem diversos tipos de sensores que lhes permitem atuar com o mundo que em conjunto com algoritmos de inteligência artificial são capazes de exercer as funções de seu propósito.

6.3 Histórico de ataques

Não é de hoje que essas máquinas causam medo e preocupação em algumas pessoas, há muitos anos esse tipo de tecnologia vem causando problemas e com o seu uso crescente as ocorrências se evidenciaram. O Departamento de Trabalho dos Estados Unidos mantém em seu site[1] os registros de acidentes com robôs no trabalho[14] totalizando desde 1984 40 acidentes onde 28 desses resultaram em fatalidade. Em sua grande maioria são acidentes onde houve falha na operação da máquina ou desatenção por parte do trabalhador, mas não somente essa tecnologia está sujeita a causar danos às pessoas, notebooks, computadores e celulares são diariamente expostos e suscetíveis a vírus e ataques hackers. Devido a grande quantidade de computadores e celulares de hoje em dia e o tipo de informação que pode-se obter deles, estes dispositivos vem sendo alvo de hackers que tentam obter essas informações das mais derivadas formas a fim de inúmeros propósitos. Entre as principais motivações pode-se citar: diversão, curiosidade, dinheiro, ativismo e desafio.

Existem diversas estratégias para conseguir acesso ao sistema e adquirir as informações necessárias, no entanto, é através de brechas ou vulnerabilidades que normalmente tem-se acesso às informações. Contudo com a grande quantidade de ataques a dispositivos, respostas a essas atitudes foram tomadas e cada vez mais os softwares e hardwares tem sido incrementados a fim de impedir que suas vulnerabilidades sejam exploradas e mesmo assim continuam sendo alvos constantes. Não somente as informações inseridas nos dispositivos são almejadas pelos hackers, mas também procuram consegui-las por intermédio deste, usando o microfone e câmera, por exemplo.

Com dispositivos IoT(Internet of Things), que vem crescendo em quantidade e em ritmo acelerado, isso não é diferente. Uma estratégia para ataque em servidores chamada DDoS (Distributed Denial of Service) onde vários dispositivos fazem requisições simultâneas a um mesmo recurso web para deixá-lo fora de serviço ganhou muita força nos últimos anos. Devido a baixa segurança implementada nos dispositivos IoT como, câmera de vigilância e roteadores, eles tornam-se de fácil acesso aos hackers que os utilizam para seus ataques a servidores. O Brasil por exemplo é o quarto maior país com esse tipo de ataque[13] e a quantidade de ações originados por dispositivos IoT no Brasil quadruplicou de 2016 a 2018[6]. O que tende a piorar devido aos novos robôs que vem adotando serviços IoT em suas funcionalidades, tornando-os acessíveis pela internet.

[1] https://www.osha.gov/

6.4 Robôs domésticos

Uma pesquisa realizada [9], entre coreanos e americanos, tentou definir quais seriam as expectativas dos robôs considerando-se sua aparência, modo de iteração, papéis sociais e tarefas desejadas. A visão coreana para os robôs domésticos seria um robô mais humanoide, já para os americanos um robô mais futurístico e focado nas atividades que irá exercer. Na Coreia, os robôs de serviço dominam o mercado, incluindo-se robôs educacionais e cuidadores de idosos. Nos EUA, robôs funcionais, como aqueles que trabalham em ambientes inóspitos, robôs submarinos e exploração espacial.

Já na questão do modo de interação, os americanos e coreamos dão autonomia aos robôs quanto a sua forma de interação, em específico os coreanos, ressaltaram algumas limitações de fala. Em relação a segurança temos uma considerável diferença entre os dois povos, os americanos tem a expectativa de robôs armados para proteção e os coreanos ressaltam um design mais amigável dos robôs facilitando a aproximação e empatia com as crianças.

Tarefas desejadas são aquelas que não são dependentes das pessoas para controlar os robôs, os coreanos desejam robôs para controlar o ambiente doméstico, incluindo detecção de poeira, purificação de ar, sanitização, detecção de incêndio, controle de temperatura e umidificação, enquanto os americanos buscam robôs para controle de iluminação e aquecimento.

De forma geral, o robô domestico dependerá de qual mercado se destina pois as diferenças culturais definirão sua aceitação, mas independente da diferenciação de robô sendo humanoide ou não, traz consigo uma grande variedade de utensílios (sensores, motores, etc.) o que possibilita a realização das tarefas que lhe forem atribuídas, a Figura 3 aborda os Tipos de Tarefas.

Importante ressaltar que a Coreia do Sul e os Estados Unidos estão entre os 10 países com maior densidade de robôs. O índice da Coreia é 3 vezes maior do que o dos EUA, no entanto a população trabalhista da Coreia é quase 6 vezes menor o que demonstra que essa tenologia está mais próxima de cada cidadão no dia a dia. Ambos os países são bem estruturados tecnologicamente o que pode causar grande diferença caso a pesquisa seja replicada em outros países.

Tipo de Tarefas	Korea (n=20)	US (n=20)
Limpeza	16	12
Cozinhar	11	7
Controle do Ambiente	6	1
Funções Relacionadas a IT[1]	4	6
Segurança	2	4
Buscar e Cuidar	2	3
Funções relacionado a Saúde[2]	1	2
Jardinagem	0	4
Diversos	7	5

1. Conexão à Internet, acesso USB, etc.

2. Função de mensagem e monitoramento de saúde.

Figura 6.3: Concessões de participantes Coreanos e nos EUA que desejam robôs para executar tarefas particulares na casa (Modificado)

País	População (Milhões)	Data
Estados Unidos	155,962	Oct/18
Coreia do Sul	26,721	Sep/18
Brasil	92,08	Sep/18

Figura 6.4: População Trabalhista (Modificado) [3]

Os produtos encontrados para esse segmento são dos mais diversos. Em sua maioria são responsáveis por alguma tarefa mais simples como cortar grama e limpar o chão, mas também existem alguns mais completos e sofisticados que podem cuidar de humanos ou controlar a casa. A Vivo[17] amostrou alguns deles em seu site sendo de diversos modelos para diferentes funções e que serão citados nesse artigo.

A *Gita* da *Piaggio*[2] é um robô que tem como proposta ser uma mala que não é necessário carregar ou arrastar. Possui um par de câmeras estereoscópicas montadas na parte frontal e traseira do robô, com as quais está constantemente executando o SLAM visual (localização e mapeamento simultâneos) para identificar seu dono e o seguir ou percurso específico.

[2]https://www.piaggiofastforward.com/gita

O produto da Blue Frog Robotics[3], o Buddy é uma solução IoT que integra diversos componentes domésticos. Contando com câmera, microfone, WI-FI, bluetooth, sensor de mapeamento e localização. Entre suas funcionalidades estão a de Teleconferência, gravar vídeos, controle remoto, locomover-se e reconhecer pessoas e objetos.

Há também um segmento atuando na produção de robôs que são usados para entretenimento. Hoje já é possível encontrar muitos desses produtos que estão conectados ao WI-FI ou a internet para uma interação mais ampla com o consumidor. Como é o caso da SoftBank Robotics[4] que criou o NAO. O robô humanoide consegue locomover-se, movimentar os braços, possui sensores de toque pela sua estrutura e com seus microfones, câmera e caixas de som, consegue se comunicar.

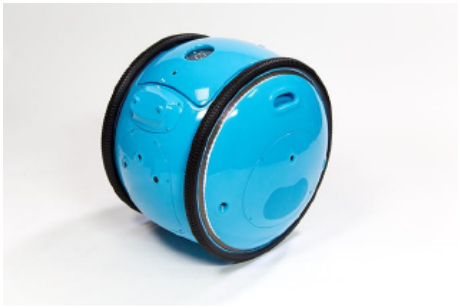


Figura 6.5: Gita

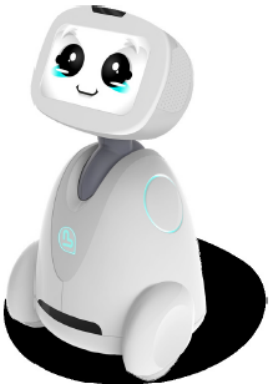


Figura 6.6: Buddy

6.5 Segurança em robôs

Em paralelo à evolução de robôs domésticos, pesquisas relacionadas à segurança também tem sido desenvolvidas, em específico para minimizar alguns problemas: canais de comunicação inseguros, texto não criptografado ou com criptografia fraca, autenticação fraca e falta de autorização suficiente para proteger funções críticas tais como instalação de software ou atualizações. Isso acarreta consequências negativas e podem permitir acesso ao robô de forma remota, acessando informações do estado interno e externo e até aos controladores dos motores.

Muitos robôs usam bibliotecas de código aberto como por exemplo o Robot Operating System (ROS)[15]. De acordo com a documentação do ROS[2]: Os tópicos são chamados de barramentos sobre os quais os nós trocam mensagens. Os tópicos têm semântica anônima de publicação / assinatura. Em geral, os nós não estão cientes de com quem estão se comunicando.", mesmo se os dados são bem criptografados na comunicação dos nós, ainda está sujeito a ataques pelo não domínio da origem do acesso, mesmo com o ROS atualizado e incrementado o nível de criptografia para a sua comunicação de dados, o sistema pode estar suscetível a ataque.

[3]https://buddytherobot.com/en/buddy-the-emotional-robot/

[4]https://www.softbankrobotics.com

Figura 6.7: NAO [1]

A empresa IOActive conduziu um estudo preliminar e superficial para identificar vulnerabilidades em sistemas de robôs domésticos, o resultado foi disponibilizado em forma de um relatório [5]. O estudo reuniu produtos de 6 empresas que produzem robôs domésticos e industriais. Foram encontradas aproximadamente 50 vulnerabilidades onde muitas delas eram comuns entre robôs das diferentes empresas.

Os problemas encontrados foram relacionados a:

- Comunicação insegura
- Problemas de autenticação
- Falta de autorização
- Criptografia fraca
- Problemas de privacidade
- Configuração padrão fraca
- Vulnerabilidades em códigos *open source*

Também foram analisadas as funcionalidades que os robôs dispunham. Por sua vez foram encontrados problemas com:

- Microfones e câmeras
- Conectividade com a rede
- Interação com serviços externos
- Aplicações de controle remoto
- Extensibilidade modular
- Funcionalidades de Segurança
- *Firmware*
- Robôs autônomos
- Conhecimento do Sistema Operacional
- Aviso de rede
- Instalação rápida
- *Backups*
- Conexão de portas

Algumas das vulnerabilidades podem ser resolvidas com pouco esforço e tornariam os sistemas dos robôs muito mais seguros, por exemplo, ao invés de se transmitir comandos sem proteção pela rede, poderia se adotar algoritmos de cifragem confiáveis e que possuem caraterísticas diferentes para serem usados em contextos diferentes. Em [12] é disponibilizado um estudo comparando tipos de algoritmos cifradores e alguns exemplos de famosos cifradores muito usados por corporações como o RSA e o 3DES. Algumas implementações dos algoritmos já existem e podem ser encontradas em diversas linguagens de programação ou em pseudo-código. O problema de usar uma implementação de terceiros sem uma revisão é a possibilidade de um ataque de implementação, que é uma atual estratégia entre os Hackers.

"A autenticação é critica, pois normalmente é a porta de entrada para a maioria dos sistemas, desta forma, um intruso, malintencionado, poderia acessar os mais variados recursos e produzir grandes prejuízos.O método tradicional para a autenticação de uma parte é através do uso de senhas"[16], no entanto de acordo com [5] nem esse simples método esta sendo utilizado pelas empresas em seus robôs. Uma implementação que não custa muito esforço computacional e que ajudaria em casos onde a comunicação é feita entre o robô e o servidor e em casos de comunicação entre o cliente e o robô em casos onde por exemplo o cliente deseja realizar a instalação de um programa na maquina. A autenticidade nesses casos é essencial para que um terceiro não instale programas maliciosos sem o consentimento de uma pessoa autorizada.

6.6 Conclusão

A tecnologia vem avançando muito e não é difícil de encontrarmos robôs resolvendo novos problemas que anteriormente eram solucionados somente por humanos. Esse mercado está crescendo rapidamente em diversos países, no entanto, o atual cenário dos robôs produzidos e comercializados demonstra a falta de preocupação por parte das empresas com a segurança de seus produtos. Sendo assim foi identificado 3 classificações para potenciais perigos que esses robôs podem desempenhar:

- Ofensivos Fisicamente
- Ofensivos a Privacidade
- Ofensivos a Terceiros

A. Ofensivos Fisicamente

Os robôs que conseguem através de seus atuadores denegrir, um humano, um animal ou objetos. O NAO pode ser programado para exercer uma atividade e com sua locomoção e braços, poderia pegar um objeto potencialmente perigoso como uma faca.

B. Ofensivos a Privacidade

Os robôs por possuírem diversos sensores, câmeras, microfones e outros tipos de dispositivos para monitorar seu estado interno e externo, podem ser facilmente invadidos caso tenha conexões externas, mesmo que o software seja de código aberto como o ROS, ainda assim está suscetível a ataques externos, liberando os seus sensores, câmeras, microfones e controle do robô para um hacker.

C. Ofensivos a Terceiros

Os robôs que possuem vulnerabilidades que permitem ser usados em ataques DDoS e sendo assim o prejudicado seria a empresa que possua um serviço web. Algumas vulnerabilidades não necessitam de muito esforço para serem corrigidas, como é o caso da transmissão de mensagens na rede com o emprego de algoritmos de cifragem e da implementação de um autenticador para algumas funcionalidades. Robôs como o Buddy, que possibilitam teleconferência e controle remoto transmitindo informações de vídeo e áudio pela internet, necessitam de forma premente, sistemas de cifragem. O NAO também faz parte desses tipos de robôs pois permite instalações de programas em seu sistema a fim de ser programado pelo usuário para exercer atividades especificas. Caso não seja realizada uma verificação do instalador, é permitido o acesso externo sem níveis de segurança.

6.7 Bibliografia

[1] Imagem nao. URL: https://goo.gl/NwjimR.

[2] Ros documentation. URL: https://goo.gl/UCXnPQ.

[3] World employed persons. URL: https://goo.gl/u5K1bA.

[4] Taylor Armerding. Robôs: muitos recursos, pouca segurança, 2017. URL: https://goo.gl/VCGfgE.

[5] Cesar Cerrudo and Lucas Apa. Hacking robots before skynet, 2017. URL: https://goo.gl/8hdGqh.

[6] Convergência Digital. Quadruplica o núumero de ataques ddos originados por dispositivos iot no brasil, 2018. URL: https://goo.gl/wCpQ6N.

[7] Redação do Site Inovação Tecnológica. Robôs domésticos põem em risco segurança e privacidade dos donos, 2009. URL: https://goo.gl/ir35v6.

[8] Bruno Lazaretti. Que robôs são utilizados por forças militares?, 2018. URL: https://goo.gl/7ycbf7.

[9] Hee Rin Lee, JaYoung Sung, Selma Šabanović, and Joenghye Han. Cultural design of domestic robots: A study of user expectations in korea and the united states, 2012. URL: https://goo.gl/5bzDN9.

[10] Carla Monteiro. Conheça os 10 países mais robotizados do mundo, 2017. URL: https://goo.gl/Uzbn6y.

[11] Filipe Oliveira. Brasil fica entre os últimos lugares em ranking de automação de empresas, 2018. URL: https://goo.gl/UiNyUE.

[12] Ronielton R. Oliveira. Criptografia simétrica e assimétrica: os principais algoritmos de cifragem, 2012. URL: https://goo.gl/BbVrnA.

[13] Rafael Romer. Brasil jé é o quarto país do mundo em ataques ddos a partir de dispositivos iot, 2016. URL: https://goo.gl/7rWBA3.

[14] UNITED STATES DEPARTMENT OF LABOR Occupational Safety and Health Administration. URL: https://goo.gl/TNf9mY.

[15] Carlos Balaguer Santiago Morante, Juan G. Victores. Cryptobotics: why robots need cyber safety, 2015. URL: https://goo.gl/bMx6D7.

[16] Flávio de Oliveira Silva. Controle de acesso a web services baseado em um protocolo de autenticação segura, 2004. URL: https://goo.gl/vgwjQS.

[17] Vivo Tech. 7 robôs domésticos que facilitam o dia a dia, 2018. URL: https://goo.gl/ewqpxc.

Parte II

Projetos

Capítulo 7

Controle de Acesso com Arduino

Amanda C. Pascon

7.1 Conceito

O projeto desenvolvido trata-se de um dispositivo para ser usado nos portões elétricos de residências, usando como chave qualquer cartão que possua Identificação por radiofrequência (*RFID*) e.g. cartão de ônibus, condomínio, empresarial, etc.

É desejado um desenvolvimento acessível à todos. Alcançando desta forma, uma funcionalidade dupla em um mesmo cartão, promovendo praticidade.

Figura 7.1: Prototipação

7.2 Recursos utilizados

Para esse projeto foi montado um protótipo de uma porta, Nela foi instalada um módulo de leitura *RFID*. Esse módulo é ligado ao Arduíno UNO. Para a conexão com a trava elétrica foi utilizada um relé para não causar danos no Arduíno.

Em relação ao código, foi utilizada a IDE do Arduíno, na linguagem C (modificada) com algumas funções para o Arduíno já presentes na *Arduino IDE*, com auxílio da biblioteca MFRC522.h, que permite suporte para o módulo, esta biblioteca pode ser encontrada na própria IDE.

Também foi seguido um tutorial, sobre o funcionamento e uso de *RFID* em Arduínos, como base [2].

7.3 Tutorial

- Baixar a IDE do Arduíno [1].
- Instalar a biblioteca MFRC522.h na IDE.
- Descobrir o ID do cartão que deseja conceder acesso (ver no tutorial disponível na seção anterior), e alterar o valor a linha 65 em `ControleDeAcesso.ino`, disponível no GitHub.
- Fazer as conexões correspondentes, entre seu módulo RFID e o Arduino UNO:

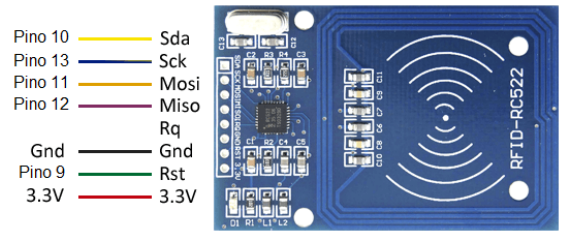


Figura 7.2: Ligação entre Arduíno (esquerda) e módulo RFID (direita)

- Ligar o pino de dados do relé no pino 4 do Arduino (o pino 4 já está configurado na linha 9 do código, livre para alteração);
- Conectar a trava elétrica, de modo a passar pelo relé, tendo o fechamento do circuito, que ocorrerá dada a identificação de um cartão cadastrado.
- Enviar o código ao Arduino.

7.4 Bugs

É necessário cuidado ao comprar seu módulo *RFID*, ele pode apresentar alguns problemas dependendo a qualidade, podendo apresentar falhas na leitura dos cartões.

7.5 Repositório

O repositório com os arquivos e demais informações sobre o projeto se encontra em:

`https://github.com/amandapascon/Projeto-Hackerspace-2019`

7.6 Conclusão

A ideia geral da construção do sistema de controle de acesso com Arduíno foi contemplada. Ademais, é possível com mais tempo e planejamento, trazer melhorias ao projeto para que seja mais dinâmico e mais acessível. Dessa maneira, defeitos decorrentes poderiam ser corrigidos e também, trazer uma melhor montagem e estrutura de todos os itens necessários para a implementação física, viabilizaria o projeto à mais pessoas.

7.7 Bibliografia

[1] Arduino ide, 2020. URL: `https://www.arduino.cc/en/main/software`.

[2] Iberê Thenório. Como usar uma identificação por rádio, 2019. URL: `https://www.youtube.com/watch?v=gcBN4NLqz_U&list=PLYjrJH3e_wDNLUTN32WittrpBxeleEqNp&t=0s`.

Capítulo 8

Facilitando Vidas por Meio de CMD

Gustavo de Jesus Rodrigues Silva

8.1 Conceito

No decorrer dos anos, a quantidade de programas de software tem aumentado gradativamente e, com eles o número de pessoas que utilizam para diversas finalidades ao longo do dia. Porém, sempre a necessidade de iniciar determinados programas e aplicações na inicialização do sistema operacional, logo ocupando tempo e eficiência de produtividade, seja no trabalho ou em casa.

O Windows possui uma ferramenta em seu sistema denominada CMD (Prompt de Comando), sendo um interpretador de linha de comando em que permite o acesso a vários recursos do sistema. Com isso é possível acessar um determinado programa, criar uma pasta, acessar um arquivo, entre muitas outras aplicações.

Logo, este projeto foi desenvolvido para ajudar a solucionar esses problemas e facilitar a vida de quem precisa abrir programas e sites repetitivos que são utilizados todos os dias, além de oferecer apoio para as pessoas que não tem conhecimento de comandos CMD do Windows. Para utilizar a ferramenta, é necessário seguir determinadas etapas de preparação.

8.2 Recursos utilizados

Na criação desse projeto foi utilizado um notebook com o sistema operacional Windows 10, em que todo o código foi criado na linguagem de programação C e em seguida compilado para ser executado no SO.

8.3 Tutorial

- Baixe todos os arquivos no projeto no repositório no GitHub.
- Compile e execute o programa do projeto no CMD ou somente baixe o executável e o execute através do CMD.
- Na execução do projeto, basta apenas digitar o “nome do programa” para que o respectivo programa seja aberto ou “abrir site” para executar determinado site através do navegador padrão.
- Para aprofundamento da funcionalidade do programa e de customização de suas configurações, o usuário tem que possuir um conhecimento básico em C.

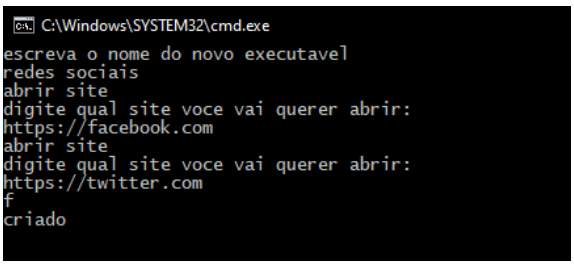
8.4 Programas Suportados

- Biblioteca
- Bloco de Notas
- Calculadora
- Paint
- Painel de Controle
- Powerpoint
- WordPad

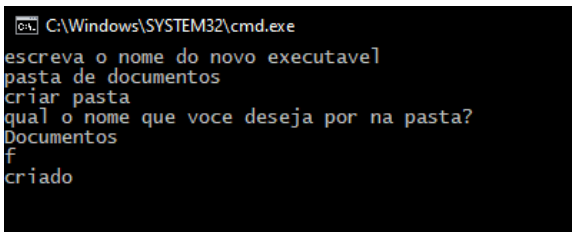
- Word
- Teclado virtual
- Lupa
- Media Player
- Explorer

8.5 Execução do Projeto

Para iniciar a execução, primeiramente inserimos a operação que será realizada, por exemplo foi definido *redes sociais*, em que o usuário poderá selecionar as rede sociais que utiliza em seu dia a dia. Para selecionar o link de uma rede social, é necessário colocar o comando *abrir site* e em seguida a sua URL, sendo que esse comando pode ser executado quantas vezes for necessário para o usuário. Depois de ter selecionado todas as redes sociais de sua preferência, para finalizar a operação, o usuário apenas digita a tecla F e será feita toda as operações que foram requisitadas.



Em seguida, temos outro exemplo de operação. no caso é para a criação de uma pasta nos documentos do usuário. No começo é definido a operação novamente, sendo ela *pasta de documentos*, que realiza a abertura dos documentos do usuário, depois a ação que será executada dentro dessa pasta que será *criar pasta*. Para criação da pasta, é necessário determinar o nome dessa pasta, em que a escolha fica a critério do usuário. Por fim, digitar a tecla F para finalizar as operações realizadas no terminal.



Várias operações pode ser realizada pelo usuário além dos exemplos exibidos anteriormente, em que muitas outras operações podem ser criadas e modificadas de acordo com as suas necessidades, porém requer o uso de conhecimento da linguagem C para incrementar no código fonte do projeto.

8.6 Conclusão

O projeto foi desenvolvido para auxiliar pessoas para iniciar programas e aplicações durante a inicialização do sistema operacional para facilitar a abertura rápida e eficiente de programas utilizados diariamente pelo usuário. É possível realizar diversas modificações no código fonte do projeto para aumentar as funcionalidades da ferramenta e de acordo com a necessidade de quem vai utiliza-la.

8.7 Repositório

O repositório com os arquivos e demais informações sobre o projeto se encontra em:

`https://github.com/Gustavo089/-2019-hakspace-design---by-little-guitar`

Capítulo 9

Ferramenta Beans

Bruno Sacconi Peres

9.1 Conceito do projeto

Este projeto se trata de um utilitário de interface gráfica para ajudar no uso dos softwares i3[3], Polybar[4] e, com sorte, muitos outros programas relacionados a *ricing*[1] que dependem de arquivos de texto para personalização e configuração.

Atualmente, Beans está sendo desenvolvido para funcionar no Arch Linux, mas ele também pode ser executado em outras distros do GNU/Linux.

9.2 Observações importantes

Este projeto ainda está em uma fase bem primitiva, e eu considero que ele é apenas uma prova de conceito, por enquanto. Atualmente, o Beans possui apenas uma interface gráfica de configuração WIP para Polybar[4].

9.3 Passo a passo

- O Beans foi criado do zero, usando um script em Perl simples para analisar os arquivos de configuração existentes. Sua interação era feita por uma interface CLI básica.
- Então, comecei a melhorar a interface, além da funcionalidade, criando menus para o usuário escolher os valores para as opções fornecidas pelo wiki de configuração do Polybar[6]. Comecei usando Xdialog[7] como interface principal, mas problemas de formatação de texto começaram a aparecer.
- Decidi migrar para o terminal Dialog[2] baseado em ncurses. Nesse ponto, tudo estava bem no que dizia respeito à interface. Mas o projeto estava ficando cada vez mais difícil de manter, pois sua base de código crescia cada vez mais rápido. Em um certo ponto, haviam mais de 1500 linhas de código em um único script Perl que nem sequer estava na metade (me refiro apenas ao parser de configuração do Polybar). Estava uma bagunça, e algo tinha que mudar.
- Dividi as funcionalidades do Beans em módulos. Ele possuía um *back-end* que servia par analisar os arquivos de configuração e gerava arquivos JSON com base neles. Logo desisti dessa ideia.
- Comecei o desenvolvilmento da interface gráfica com o GTK[5] usando C++. Como eu não tinha certeza de como integrar meu parser Perl com meu código C++, decidi portar meu parser para C++ e abandonei meus scripts Perl para sempre. Agora, tudo parece estar bem integrado e eu espero estar no caminho certo.
- Consegui tornar a configuração de cores do Polybar perfeitamente funcional!

[1] *Rice/ricing* - prática de customizar ou aprimorar o visual de uma área de trabalho, principalmente por meios não convencionais ou padrões[1]

9.4 Dependências

É difícil definir quais serão todos os pacotes e dependências necessários para o projeto, dado que ainda está em desenvolvimento. A lista definitiva ficará pronta posteriormente.

Até lá, é importante acompanhar os lançamentos das novas versões, no repositório do GitHub.

9.5 Problemas Conhecidos

Atualmente, eu ainda estou aprendendo a usar o GTK. Dessa forma o desenvolvimento da interface gráfica será lento, mas pelo menos já começou!

9.6 Imagens

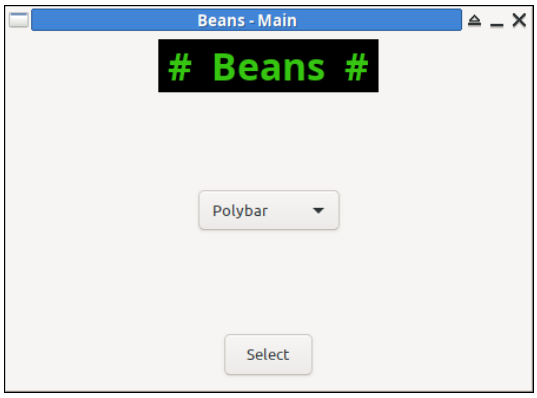


Figura 9.1: Interface de escolha de software

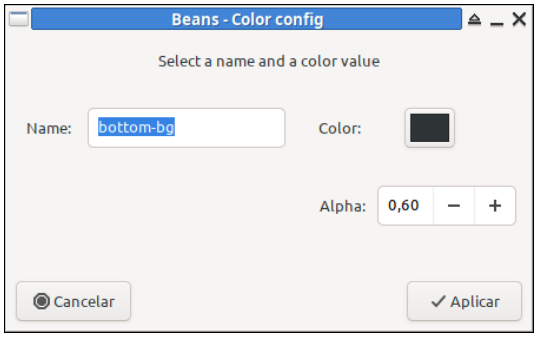


Figura 9.4: Configuração de nova cor

Figura 9.2: Interface de escolha de seção do Polybar

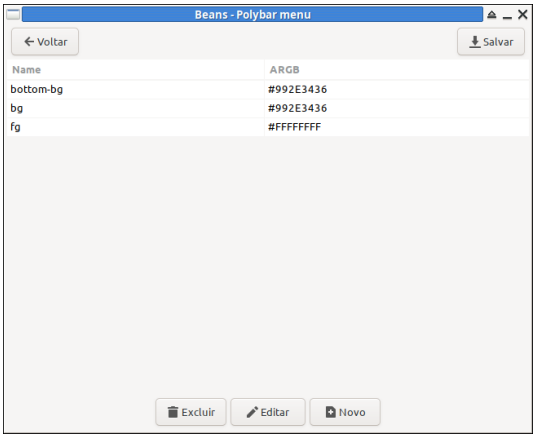


Figura 9.3: Lista de cores configuradas

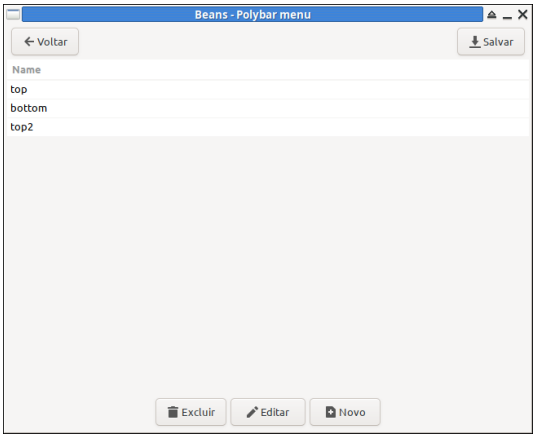


Figura 9.5: Lista de interfaces criadas

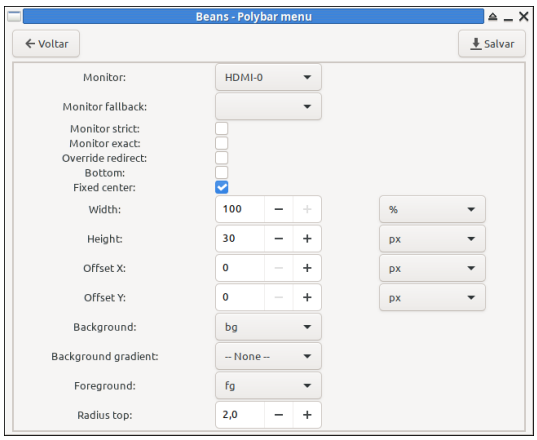


Figura 9.6: Configuração de nova interface

9.7 Repositório

O repositório com os arquivos e demais informações sobre o projeto se encontra em:

`https://github.com/mdk97/Beans/tree/dev`

É importante notar que a versão mais recente do código está na *branch* dev do repositório.

9.8 Bibliografia

[1] Definição de rice/ricing. URL: `https://www.reddit.com/r/unixporn/wiki/themeing/dictionary#wiki_rice`.

[2] Dialog. URL: `https://invisible-island.net/dialog/`.

[3] i3. URL: `https://i3wm.org/`.

[4] Polybar. URL: `https://github.com/polybar/polybar`.

[5] Site oficial do gtk. URL: `https://www.gtk.org/`.

[6] Wiki de configuração do polybar. URL: `https://github.com/polybar/polybar/wiki/Configuration`.

[7] Xdialog. URL: `http://xdialog.free.fr/`.

Capítulo 10

Jenga Builder - Robô Construtor de Jenga

Ágata Rangei Drigo
Anderson Pinheiro Garrote
André Matheus Bariani Trava
Thiago Yussuki Uehara
Gabriel Eiji Uema Martin

10.1 Conceito

O projeto visa a elaboração de um robô construtor de torres de Jenga, um jogo de habilidade física em que jogadores devem retirar uma peça por vez de uma torre previamente construída. Espera-se, com o desenvolvimento desse projeto, estudar mais profundamente os fundamentos da robótica e as práticas da área em um projeto semelhante aos robôs utilizados na industria atual.

A proposta do projeto é desenvolver e construir um robô capaz de construir torres de Jenga. Uma torre de Jenga é composta de peças padronizadas dispostas três à três em níveis, sendo que cada nível é rotacionado 90º em relação ao anterior, conforme a figura 10.1.

Figura 10.1: Torre de Jenga

10.2 Pré-requisitos e recursos utilizados

Para a impressão dos modelos 3D **foi necessária uma impressora com área de impressão de no mínimo 190 x 190mm**, os modelos podem ser encontrados na pasta *modelos-impressao*.

- 2x Motores servo 9g SG90 TowerPro
- 4x Motores de passo 8BYJ-48
- 4x Drivers ULN2003
- 1x Arduino Uno
- 1x Breadboard
- 1x Elástico
- 1x Rolamento 3 x 13 x 5mm 633zz
- 1x Parafuso M3 15mm + Arruela
- 2x Parafusos M3 6mm
- 1x Parafuso M2.5 10mm
- 1x Folha em EVA
- 1x Fonte de energia 5Vdc de 2A
- 4x Parafusos #6-32 x 2"

- 8x Parafusos #6-32 x 1-1/4"
- 4x Parafusos #6-32 x 3/4"
- 26x Porcas #6

Para controlar os motores de passo, foi necessária a biblioteca CheapStepper [1].

10.3 Passo a passo

- Modificamos o modelo do Eixo Z do projeto RPieces-TableTopXYZ [3] para adequa-lo à altura da torre de Jenga.
- Imprimimos os modelos 3D.
- Implementamos funções para abrir, fechar e girar a garra, e para mover os eixos.
- Implementamos funções para construir niveis pares, e niveis ímpares da torre, usando as funções citadas anteriormente.
- Implementamos um programa que dadas as dimensões das peças, constrói uma torre de Jenga da altura desejada (máx 15 niveis).

Obs: as funções e o programa citados acima estão disponíveis no arquivo *código/jengabuilder.ino*.

10.4 Experimento

O robô pegará as peças em uma posição prédefinida utilizando a garra, a levará até o local da próxima peça da torre (evitando a colisão com as peças já posicionadas) e a girará se necessário.

O robô não mantém um estado interno das posições das peças colocadas, o seu ambiente é limitado à posição inicial da peça, à posição da garra e à posição final da peça, portanto depende de ajuda externa (para sempre ter uma peça disponível na posição inicial) e não é capaz de corrigir falhas ou imprevistos.

É esperado que o robô seja capaz de montar pelo menos 3 níveis da torre precisamente. Uma quantidade maior de níveis é possível, mas provavelmente terão menor precisão, uma vez que a acurácia do posicionamento das peças dos níveis anteriores se acumula ao longo da construção.

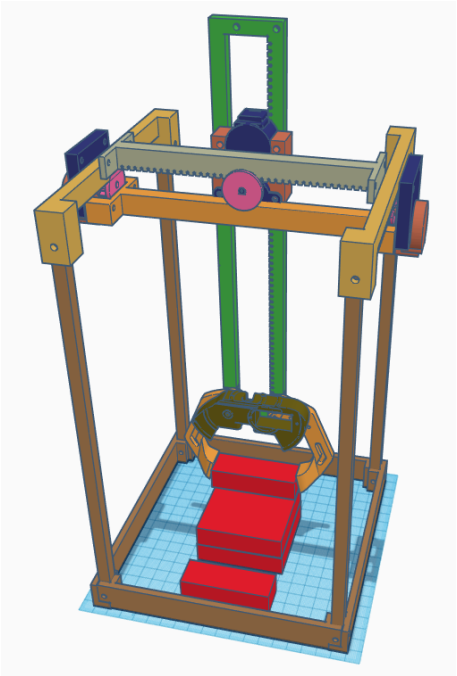


Figura 10.2: Modelo 3D do projeto

10.5 Repositório

O repositório com os arquivos e demais informações sobre o projeto se encontra em:

`https://github.com/eijiuema/hackerspace2019-jengabuilder`

10.6 Bibliografia

[1] Tyler Henry. Cheapstepper. URL: `https://github.com/tyhenry/CheapStepper`.

[2] jjrobots. Robotic gripper, 2019. Acessado em 28 nov. 2019. URL: `https://www.thingiverse.com/thing:3648782`.

[3] TGit-Tech. Rpieces-tabletopxyz, 2017. Acessado em 28 nov. 2019. URL: `https://www.thingiverse.com/thing:2166533`.

Capítulo 11

Stupefy - Um Conversor de Playlists do Spotify para o Youtube

Bruno Frítoli Carrazza
Jade Manzur de Almeida

11.1 Conceito

O projeto é um script em Python, que interliga as API's do Spotify e YouTube e permite a conversão de playlists existentes do Spotify para uma nova playlist do YouTube. Foi desenvolvido com a ideia de aproveitar o algoritmo de criação de playlists do Spotify, que usa conceitos avançados de machine learning, com a praticidade do YouTube ao reproduzir playlists.

11.2 Pré-requisitos e recursos utilizados

É necessário ter o projeto cadastrado em ambas APIs, para ter o uso autorizado das chaves de usuário. O grupo utilizou Python (v3.7) para desenvolver o projeto, além das seguintes bibliotecas e APIs:

- YouTube Data API
- Spotify Web API
- Spotipy
- Oauth2Client
- Google Auth OauthLib
- Pyfiglet
- SYS Python 3
- JSON Python 3
- Time Python 3

Além de dois módulos implementados pelo próprio grupo (*interface.py* e *settings_config.py*), onde implementamos, respectivamente, uma interface para o script e um arquivo com todas as configurações necessárias, já que o uso das API's requer chaves para o usuário.

11.3 Passo a passo

1. Registramos o projeto em ambas APIs para obter as chaves de acesso (https://developer.spotify.com e https://console.cloud.google.com).

2. Estudamos as documentações disponíveis procurando entender o que precisaríamos para implementar o código.

3. Buscamos bibliotecas que ajudassem no nosso trabalho (encontramos a biblioteca spotipy, disponível nos links anteriores)

4. Implementamos primeiro as funções de comunicação com o Spotify, por apresentar uma biblioteca que facilitasse sua interação, criamos funções de buscar a playlist requerida a partir de um link URI do Spotify e que lesse todos os itens daquela playlist

5. Implementamos uma interface gráfica com o pyfiglet para mostrar o nome do projeto ao rodar o script

6. Procuramos incessantemente por alguma biblioteca que facilitasse o acesso ao YouTube, somente para falhar e ter que ler (novamente) toda a documentação

7. Implementamos as funções de pesquisar os itens da playlist e de criar a playlist no YouTube, além das funções de autorização e login na conta da Google

8. Esbarramos com um erro super chato de limite de ações com a YouTube Data API

9. Descobrimos que só podíamos converter 50 músicas por dia

10. Ajustamos o código para funcionar de acordo com as limitações impostas pela API usada

11.4 Instalação

Inscreva seu projeto nos devidos sites das APIs para conseguir as chaves de acesso Assim que obtiver as chaves, cole-as nos lugares indicados no arquivo config.py (stupefy/settings_config/config.py). Não esqueça de adicionar à pasta stupefy/settings_config o arquivo .json com o client secret do YouTube

11.5 Execução

1. Entre na pasta do projeto pelo terminal.

2. Execute o comando python3 stupefy.py, no terminal, na pasta do projeto.

3. Abra o link para logar na sua conta do Google em que deseja criar a playlist.

4. Autorize a aplicação e cole no terminal a chave gerada para autorizar o login.

5. Cole no terminal, quando for pedido, o URI da playlist do Spotify que deseja converter.

6. Digite o nome e a descrição de sua nova playlist do YouTube.

7. Torça para não ter excedido o limite diário de músicas.

8. Espere um pouquinho.

9. Curta sua música!

Pode-se conferir os resultados da execução nas imagens abaixo:

Figura 11.1: Interface Gráfica do Terminal

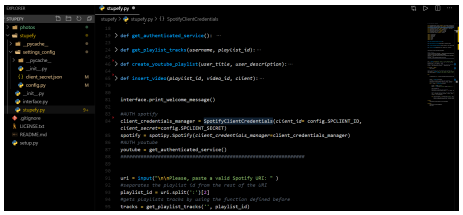


Figura 11.2: Código em Python com a Organização dos Diretórios

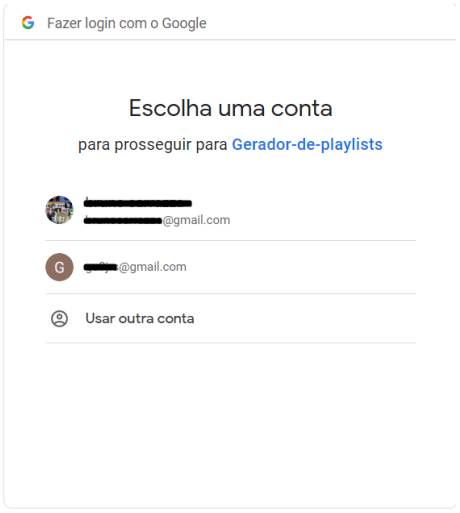


Figura 11.3: Flow de Autorização da Conta do Google (Parte 1)

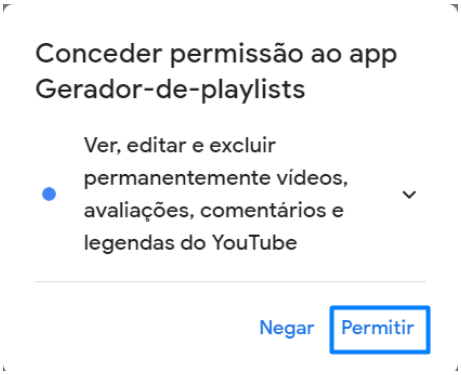


Figura 11.4: Flow de Autorização da Conta do Google (Parte 2)

Figura 11.5: Flow de Autorização da Conta do Google (Parte 3)

11.6 Bugs e Problemas Conhecidos

Devido ao uso da YouTube Data API ter um limite de ações diárias para desenvolvedores, só podemos converter 50 músicas ao dia, sem que a API lance uma exceção esquisitissíma e nem um pouco amigável ao usuário. Conhecendo esse problema, ao tentar colocar uma playlist maior no conversor, a aplicação encerra sem converter nenhuma música. Essa foi uma decisão de projeto e embora haja uma interface perguntando se o usuário deseja converter as primeiras 50 músicas da playlist, foi implementado posteriormente a interrupção do script, impedindo qualquer erro inesperado da API. Procuramos atualizar a interface em uma nova versão.

OBS: o projeto se encontra todo em inglês, devido às APIs utilizadas, que possuem interfaces padrão em inglês. Tentaremos algum jeito de traduzir as interfaces, também.

11.7 Repositório

O repositório com os arquivos e demais informações sobre o projeto se encontra em:

`https://github.com/jdmanzur/Stupefy`

Referências

[1] **Google**. YouTube Data API. Disponível em: *https://developers.google.com/youtube/v3/docs*. Acesso em:

[2] **GitHub**. Oauth2Client. Disponível em: *https://github.com/googleapis/oauth2client*. Acesso em:

[3] **Spotify**. Spotify Web API. Disponível em: *https://developer.spotify.com/documentation/web-api/*. Acesso em:

[4] **Spotipy**. Uma implementação para facilitar o uso da API Spotify Web. Disponível em: *https://spotipy.readthedocs.io/en/latest/*. Acesso em:

[5] **Google Auth OauthLib**. Google_auth_oauthlib.flow module. Disponível em: *https://google-auth-oauthlib.readthedocs.io/en/latest*. Acesso em:

[6] **Pypi**. Pyfiglet para uma interface mais amigável no terminal. Disponível em: *https://pypi.org/project/pyfiglet/0.7/*. Acesso em:

[7] **Python 3**. Time — Time Access and Conversions. Disponível em: *https://docs.python.org/3/library/time.html*. Acesso em:

[8] **Python 3**. JSON Encoder and Decode. Disponível em: *https://docs.python.org/3/library/json.html*. Acesso em:

[9] **Python 3**. SYS — System-specific parameters and functions. Disponível em: *https://docs.python.org/3/library/sys.html*. Acesso em: