O elo mais fraco da nossa segurança: S&nh5s

Entenda o que são, como criar e gerenciar senhas fortes

Caio Volpato (@caioauv) - CryptoRave2017 - 5 maio 2017

Coletivo Encripta:

Somos um coletivo de alunos da Unicamp e realizamos atividades sobre software livre, criptografia, segurança e liberdade na rede

- website: encripta.org
- Grupo no telegram: @encripta
- Página no FB: facebook.com/coletivoencripta
- Email: encriptatudo arroba riseup.net → PGP : 0x2fd9e04c6ab7bcde

Resumo:

- As senhas analisadas são bastante previsíveis (pois não temos uma boa intuição de aleatoriedade).
- Não repita a mesma senha em dois lugares.
- Sua senha não pode conter datas, nomes ou qualquer coisa pessoal.
- Use senhas verdadeiramente aleatórias, usando por exemplo o método dadoware (diceware).
- Use um Gerenciador de senhas (Password Manager).
- Habilite autenticação em duas etapas (2FA).
    - Sério!!! Passou a era da senha, não é ela sozinha que vai garantir sua segurança. Veja Kill the Password: A String of Characters Wont Protect You.
- Troque suas senhas com frequência (tipo de 3 em 3 meses).
- Assista Citizenfour.

Glossário:

Engenharia Social:

Def: Engenharia social é a manipulação psicológica de pessoas para que realizem ações ou revelem informações.

Exemplo: Um dos hackers mais famosos do mundo é o Kevin Mitnick, que escreveu um livro chamado *A Arte de Enganar*. Ele sempre pregou que a estupidez humana é a maior falha de qualquer sistema.

Cartao do Mitnick

Certa vez, uma operadora de telefonia americana tinha uma promoção na qual, na compra de um celular, você ganhava um segundo aparelho por 1 centavo. O Mitnick ligou para a loja da operadora perguntando se tinham um determinado modelo de aparelho, há quanto tempo o funcionário que atendeu trabalhava lá, etc… Então, ele ligou em outra loja da operadora se identificando como o funcionário anterior dizendo que um cliente chamado Kevin Mitnick havia comprado um aparelho, mas, como não tinham um celular naquela loja, o cliente iria buscar na outra unidade. Ele chegou lá e pegou o aparelho de graça.

Repare que o Mitnick "hackeou" a operadora sem nem usar um computador, só persuadindo as pessoas.

Motivação:

Em junho de 2016, houve um vazamento de senhas do LinkedIn. O grupo resposável pelo vazamento obteve 61 milhões de hashes (SHA1) de senhas distintas e em apenas duas horas eles conseguiram quebrar 65% das senhas :O

Um dos afetados foi o Mark Zukerberg (CEO do Facebook) que teve sua senha vazada. Sua senha era “dadada” e ele usava essa mesma senha no LinkedIn, Twitter e Pinterest.

O que aprendemos com esse episódio? Precisamos falar sobre senhas!!! Nosso dia a dia está rodeado de senhas, porém nem sempre essas elas são seguras.

Às vezes utilizamos a mesma senha em vários sites, ou temos aquela senha que um/a ex ainda tem.

Análise de senhas:

Analise de um conjunto de dados com 10 milhões de senhas link

Minha análise propõe estudar as seguintes características de interesse:

- As senhas mais comuns.
- Os tamanhos das senhas.
- Os grupos de caracteres (tipo: a senha tem apenas minúsculas e números).
- As “máscaras” das senhas.
    - (Estou chamando de máscara o “padrão” daquela senha, exemplo: a senha “abc123” tem a máscara “llldddd”, onde *l* representa as minúsculas, *d* os números, *u* as maiúsculas e *s* os caracteres especiais.)

Vamos ao que interessa: são 5,1 milhões (5.189.333) de senhas únicas, que dão no total 9.997.922 senhas.

Abaixo estão as 10 senhas mais frequentes, que juntas correspondem a 1.41% do total das senhas:

Senha	**%freq**
123456	0.56%
password	0.20%
12345678	0.14%
qwerty	0.13%
123456789	0.12%
12345	0.11%
1234	0.06%
111111	0.06%
1234567	0.05%

Abaixo os grupos:

Grupos	%
l	38.25%
l+d	29.86%
d	20.36%
u+l+d	5.71%
u+l	2.52%
u+d	1.10%
u	1.09%
l+s	0.40%

Abaixo os 11 tamanhos mais frequentes:

tam	%
8	29.81%
6	25.45%
7	16.63%
9	6.81%
5	4.95%
10	4.71%
4	3.45%
11	2.64%
12	1.91%
13	1.36%
14	0.77%

Abaixo as 20 máscaras mais frequentes, que correspondem a 65% do total:

máscara	%
llllll	11.76%
lllllllll	8.98%
lllllll	7.50%
ddddddddd	7.40%
dddddd	7.03%
lllll	3.15%
ddddddd	2.09%
lllllldd	2.00%
lllllllll	1.97%

Logo mais veremos que as senhas geradas por essas máscaras não são muito boas (baixa entropia).

Por exemplo: a máscara mais frequente llllll (11%), tem apenas 28 bits ou seja em menos de 1s todas as senhas geradas por essa máscaras são quebradas.

Lei de Zipf:

A lei de Zipf foi nomeada em homenagem ao linguista americano George Kingsley Zipf, que a popularizou.

A lei diz que se observamos o quão frequente uma palavra é em um livro (ou numa língua inteira), a *k*-esima palavra mais frequente vai aparecer proporcionalmente a *1/k*.

Ou seja, a segunda palavra mais frequente vai ocorrer aproximadamente metade das vezes do que a primeira mais frequente; já a terceira palavra mais frequente, um terço da primeira mais frequente e assim por diante…

Apesar do Zipf ter começado suas observações na língua inglesa, isso acontece em todas as línguas, inclusive em línguas extintas que ainda nem conseguimos traduzir.

É surpreendente como coisa tão humana e criativa como a linguagem pode ser expressa de uma forma tão previsível.

A lei de Zipf é um caso discreto da distribuição de Pareto, onde sua densidade vale:

$$P(x) = \frac{ab^a}{x^{a+1}}$$

onde o suporte é $x \geq b$ e $a > 1$;

Perceba que se aplicarmos log na distribuição teremos:

$$\log(P(x)) = \log\left(\frac{ab^a}{x^{a+1}}\right) = \log(a) + a\log(b) - (a+1)\log(x)$$

ou seja, o gráfico log-log da distribuição de Pareto é uma reta decrescente.

Dito isso, esse padrão não acontece apenas nas línguas, mas também descreve:

- População de cidades.
- O trafico de web-sites.
- A magnitude de terremotos.
- O número de citações em artigos acadêmicos.
- O diâmetro das crateras da lua.
- E muito mais.

Como tantos processos da vida real se comportam segundo a distribuição de Pareto, a partir da distribuição surge o Princípio de Pareto, que diz que via de regra aproximadamente 20% das causas são responsáveis por 80% dos resultados.

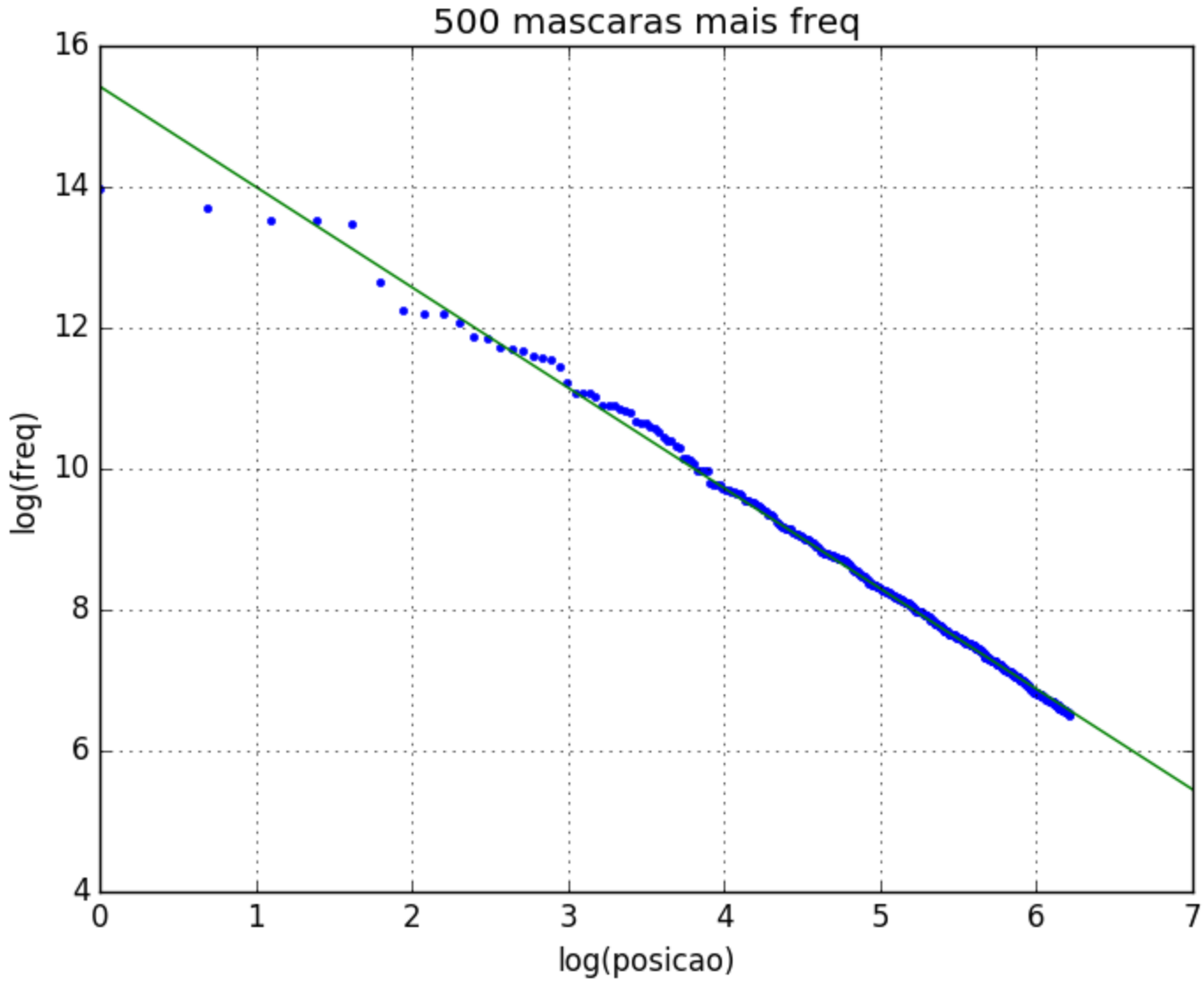
Por exemplo:

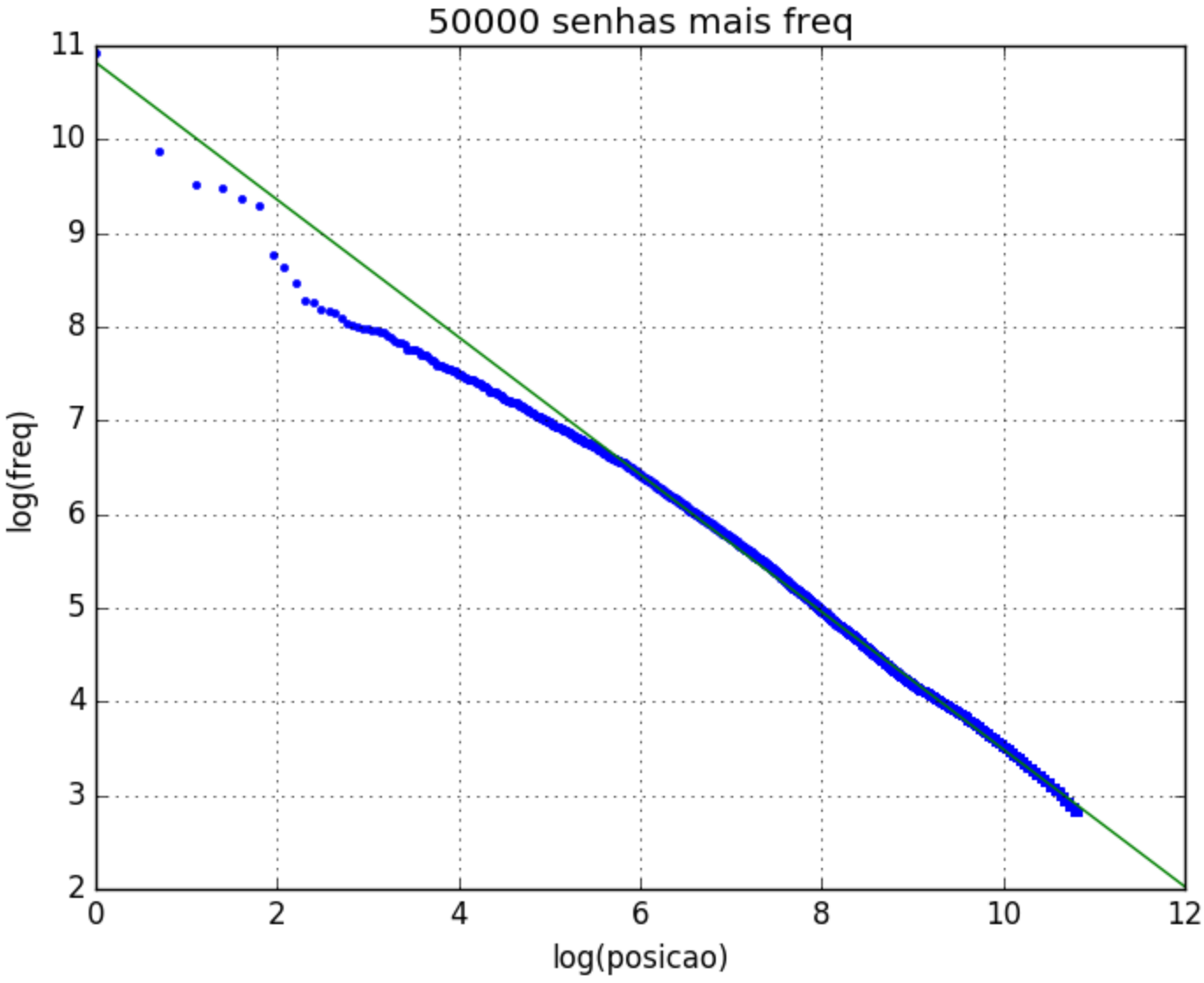
- Na língua inglesa, 18% das palavras mais frequentes correspondem a 80% das ocorrências.
- Os 20% mais ricos correspondem a 82,7% de toda riqueza mundial.
- Nos EUA, 20% dos pacientes usam 80% dos recursos de saúde.

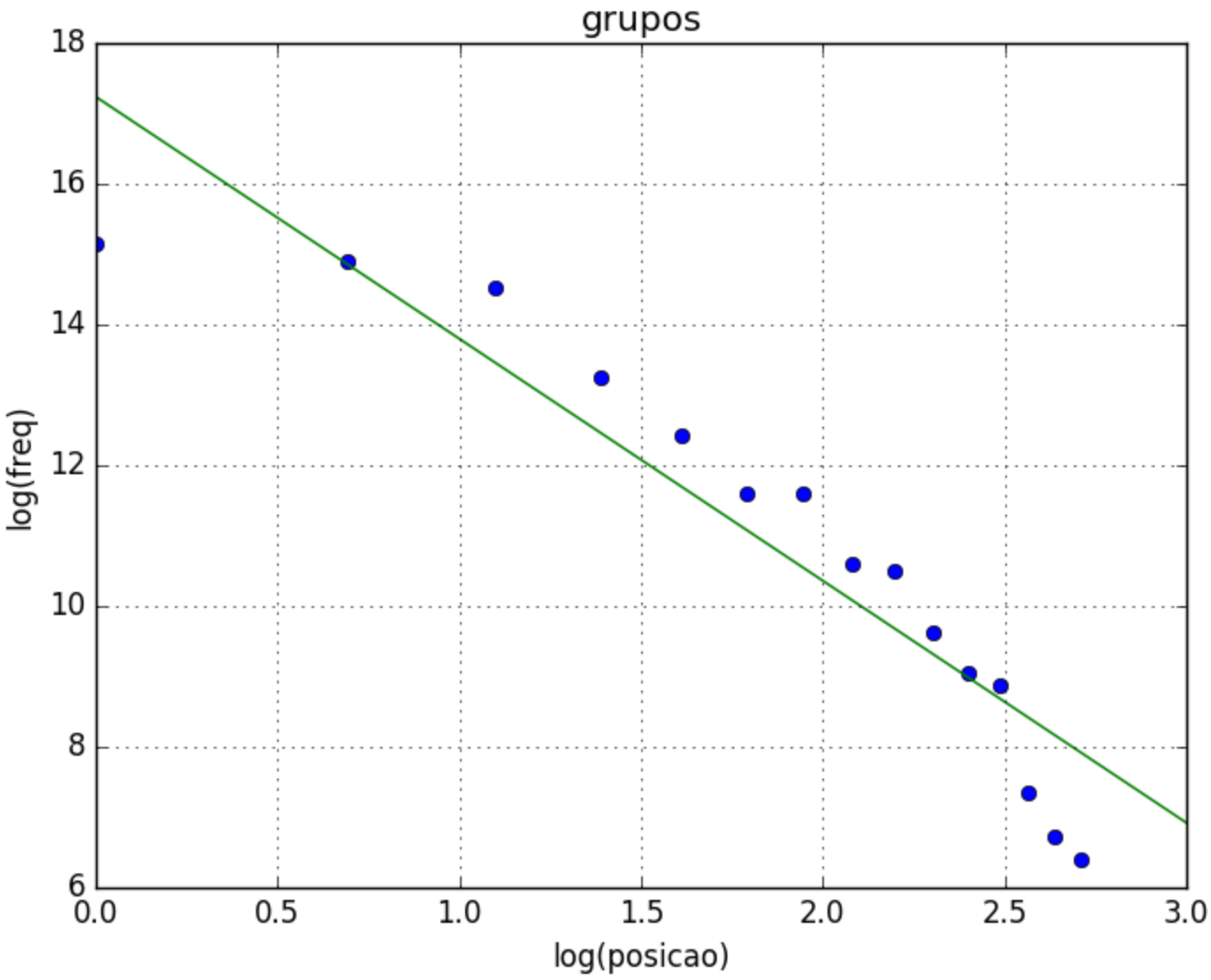
Depois, o Zipf percebeu que esse padrão 80/20 era uma consequência do princípio do menor esforço. Os falantes de uma língua naturalmente preferem usar menos palavras possível para se expressar, resultando em menos trabalho.

Voltando para as senhas:

Lembrando que o gráfico log-log da distribuição de Pareto é uma reta descreste, assim temos os gráficos:


500 mascaras mais freq
log(freq)
log(posicao)
16
14
12
10
8
6
4
0
1
2
3
4
5
6
7


50000 senhas mais freq
log(freq)
log(posicao)
11
10
9
8
7
6
5
4
3
2
0
2
4
6
8
10
12


grupos
log(freq)
log(posicao)
18
16
14
12
10
8
6
0.0
0.5
1.0
1.5
2.0
2.5
3.0

O Princípio de Pareto (aproximadamente 20% das causas são responsáveis por 80% dos resultados) também ocorre em:

- 20% dos grupos correspondem a 88% das senhas.
- 14% dos tamanhos correspondem a 88% das senhas

Feynman: o abridor de cofres

O Richard Fyanman foi um físico americano, seu trabalho era em mecânica quântica e física de partículas . Ele que inventou os diagramas de Faynman . Em 1965 ele ganhou o nobel de física.

Em sua auto biografia *O senhor está brincando, Sr. Faynman!* ele conta quando trabalhava no projeto Manhattan:

Afim de mostrar o quão insegura era a guarda de documentos secretos de fabricação de bombas atômicas ele gostava de abrir fechaduras e cofres.

Os cofres eram compostos de um disco rotatório com 100 marcações e o segredo do cofre são 3 números, ou seja, existem 1 milhão de combinações. Demora uns 5 segundos por tentativa, assim demoraria aproximadamente 60 dias para tentar todas as combinações, porém Faynman os abria em 1.5 minutos.

Como ele conseguiu esse feito?

- Ele tinha um cofre e percebeu que como os cofres são objetos mecânicos. Eles apresentavam uma tolerância mecânica de ± 2 , assim ele não precisava tentar as 100 combinações, mas apenas 20 (2 , 7 ,12 , 17 ... , 87, 92 , 97). Entao, as 1 milhão de combinações caem para apenas 8 mil demorando 10 horas.
- Muitas pessoas usavam uma data (DDMMAA) ou seja 30 , 12 e 45 combinações e considerando a tolerancia mecanica isso dá 162 combinações ou seja 18 minutos.
- Muitos usavam como senha as constantes pi e *e* ou seja 3.14159... → 31-41-59 , 2.71828... → 27-18-28.

- Muitos não mudaram a senha padrão: 25-0-25 ou 50-25-50.
- Enquanto conversa com seus colegas e seu cofre estava aberto, girando o disco ele ouvia 2 clicks, descobrindo os 2 ultimos numeros , assim ele tinha que testar 20 combinações, demorando 1.5 minutos.

Em suma, Feynman tinha a reputação de abridor de cofres, pois:

- teve um pouco de "mente aberta" e perceber a tolerancia mecanica.
- entender a natureza humana na escolha dos segredos.
- usava de eng. social para descobrir (grande parte) dos segredos.

Entropia de Shannon:

Intuitivamente pensamos que senha boa é uma senha toda complicada e difícil de lembrar.

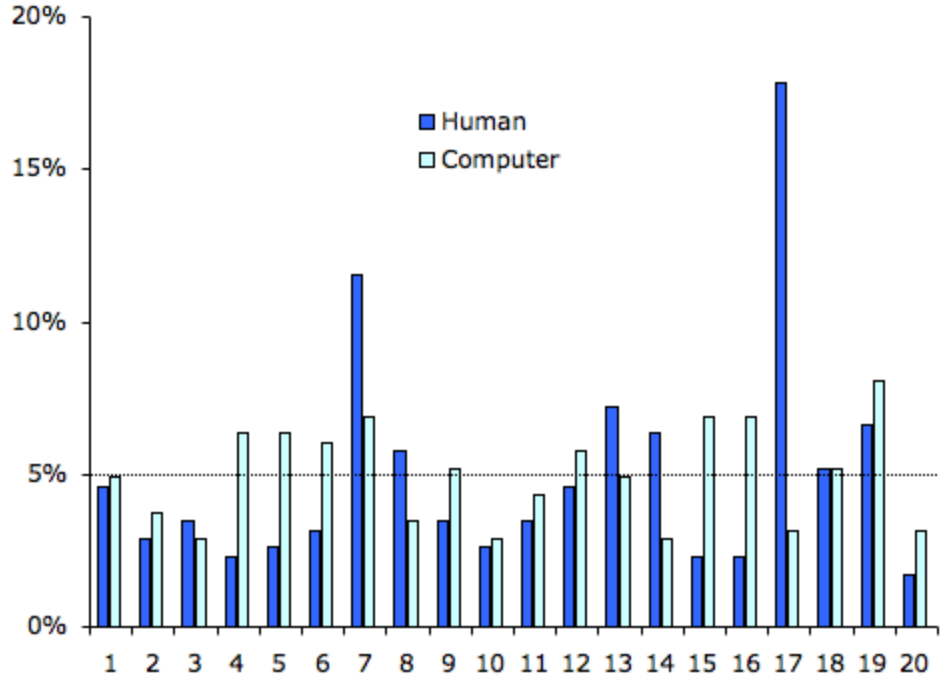
por exemplo: **S&nh5F0d@**

Ela tem:

- 9 caracteres.
- maiúsculas.
- minúsculas.
- números.
- caracteres especiais.

então ela deve ser boa. Porém, conforme veremos logo mais, **essa senha é quebrada em segundos.**

Pense em um número entre 1 e 20.



Pesquisa com n=347

Você pensou em 17, 7 ou 13? Esse post mostrou que 17 é o mais escolhido pelas pessoas, e que elas tendem a escolher números primos.

Outro exemplo é a falácia do apostador: um apostador deve apostar se uma bolinha vai cair no preto ou vermelho.

Depois dela ter caído, por exemplo, 4 vezes seguidas no vermelho, o apostador pensa que é mais “provável” que agora ela “deve” cair na preta.

Porém, a bolinha tem 50%-50% de cair no preto e vermelho, **INDEPENDENTEMENTE DO RESULTADOS ANTERIORES**.

Para fechar: escreva em um papel como você acha que será o resultado do lançamento de uma moeda 30 vezes e compare jogando de fato a moeda.

As pessoas favorecem certas sequencias, por exemplo: tendem a evitar que caia o resultado anterior repetidas vezes: dificilmente você achará que a moeda cai cara 4 vezes seguidas.

Veja esse vídeo da Khan Academy, que mostra uma visualização dessa propriedade de estabilidade de frequência.

(Quer mais um exemplo? Veja esse excelente video do Isto é Matemática, falando do problema de Monty Hall.)

Ou seja: as pessoas têm uma intuição deturpada de um resultado aleatório. Esse comportamento de manada acaba produzindo senhas mais previsíveis.

Então, como avaliar se uma senha é de fato boa? Precisamos introduzir um conceito da teoria da informação, que é o conceito de entropia de Shannon: a entropia é a “media” de informação que aquela senha contém.

A entropia *H*, medida em bits, de uma senha com comprimento *L*, utilizando *N* símbolos é definida como log na base 2 de todas as combinações que a senha pode ter, ou seja:

$H = \log_2(N^L) = L \cdot \log_2(N)$

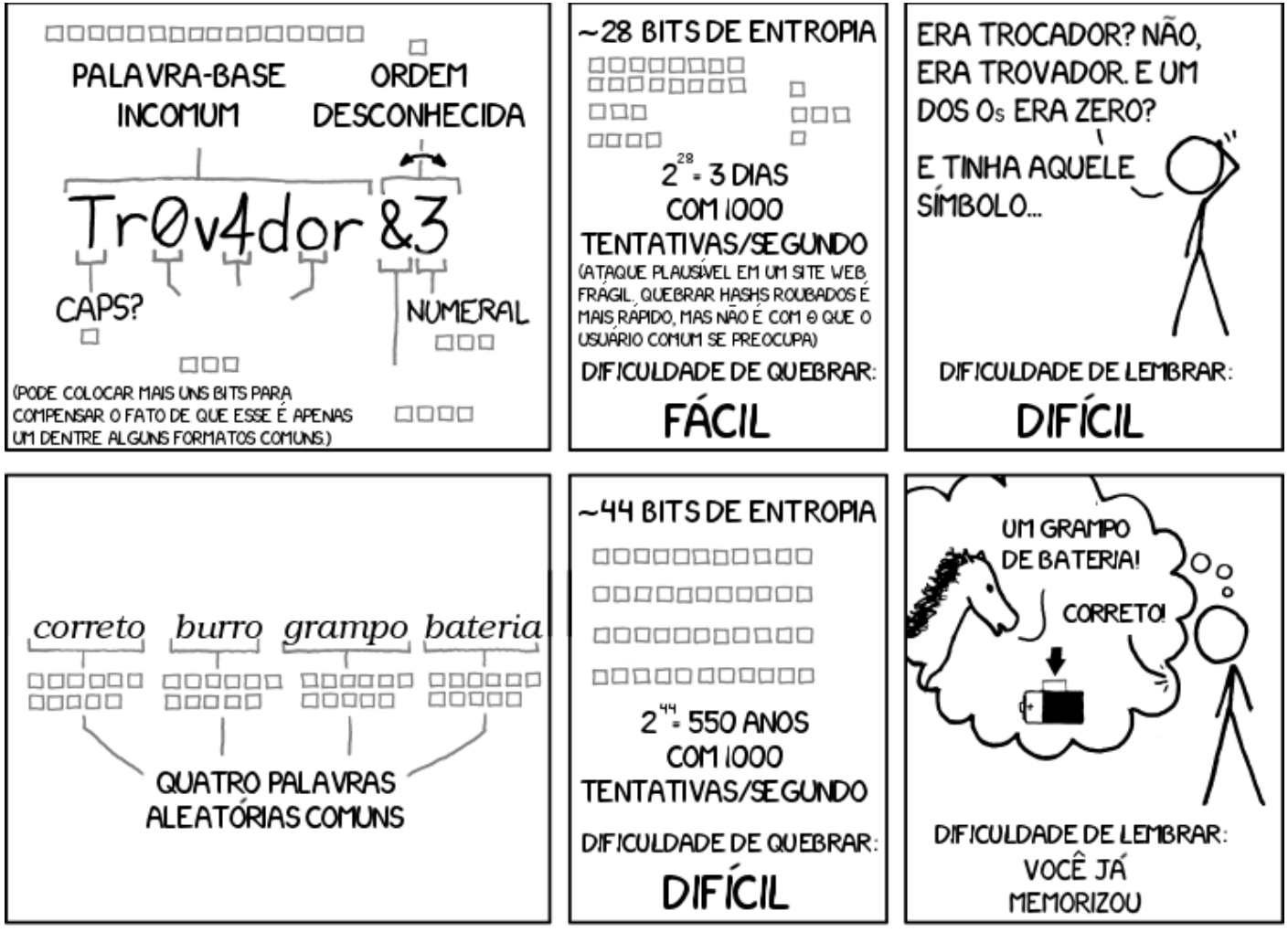
Grupo	Num símbolos (N)	entropia/simbolo (H)
números	10	3.32
letras	26	4.7
base64	64	6
“tudo”[1]	94	6.5
palavra[2]	7776	12.92

[1]: tudo = maiúsculas (26) + minúsculas (26) + números (10) + caracteres especiais (32) = 94 símbolos.

[2]: na próxima seção veremos por que uma palavra (diceware) tem essa entropia.

Ou seja, de acordo com a tabela, uma senha **verdadeiramente aleatória** com 8 caracteres usando “tudo” tem *6,5 * 8 = 52* bits de entropia.

creditos : xkcd, tradução: Oficina AntiVigilancia



Qual a lição dessa tirinha?

Essas senhas todas complicadas e super difíceis de lembrar na verdade têm uma entropia bastante baixa, porque na realidade são baseadas em apenas uma palavra comum sobre a qual, basicamente, são feitas somente algumas substituições comuns (tipo o → 0), alguns maiúsculas e um número.

Ou seja, apesar dessas senhas serem difíceis e complicadas para pessoas, para computadores elas são super fáceis de serem quebradas.

Agora fica a pergunta: quanta entropia minha senha deve ter?

No documentário Citizenfour o Snowden fala:

> Assume your adversary is capable of one trillion guesses per second.

Assim, vamos presumir 1 trilhão de tentativas por segundos (que é um valor bastante razoável).

O tempo para quebrar uma senha (*T*) é o número de combinações que aquela senha pode ter, dividido pelo número de tentativas por segundo (*P*).

ou seja: *T = 2^(entropia) / P*.

Portanto, vamos calcular a tabela abaixo (considerando *P = 1* trilhão/s) e o tamanho necessário da senha:

Entropia	Tempo para quebrar	base64	“tudo”	N. de palavras
40	1.1 segundo	7	7	3
50	19 minutos	9	8	4
60	13 dias	10	10	5
65	1.1 anos	11	10	6
70	37 anos	12	11	6
80	38 milênios	14	13	6
100	40 bilhões de anos	17	16	8

(Observação importante: esse tempo é o tempo **GARANTIDO** de quebrar a senha. Ou seja, é o tempo que demora para percorrer TODAS as possibilidades. O tempo real para quebrar a senha pode ser menor)

Vendo a tabela podemos estabelecer que **uma boa senha deve ter 80 bits** (ou seja 13 caracteres usando “tudo”), dependendo de onde você vai usar essa senha.

Quero saber se minha senha é boa: você pode olhar a primeira tabela e calcular manualmente a entropia dela, mas sugiro que use esse medidor de senha chamado Rumkin ou a lib `zxcvbn` que você pode usar aqui.

(Observação: esses dois medidores de senhas avaliam sua senha localmente no seu navegador, nunca enviando a senha para a internet. Você pode digitar sua senha neles sem medo.)

Voltando pro exemplo do começo: a senha **S&nh5F0d@** é baseada em 2 palavras comuns e somente feita algumas substituições comuns e etc…

Segundo o Rumkin, **ela tem 40,7 bits de entropia**, que segundo a nossa tabela **é quebrada em questão de segundos** (o `zxcvbn` também mostrou isso).

Método Diceware :

Preciso criar uma senha que seja fácil de lembrar e que tenha bastante entropia. Aí entra o método diceware (em português dadoware)

Diceware é um método no qual, através de um dado, uma lista de palavras, papel e lápis podemos construir senhas mais seguras.

Consulte o livreto da ThoughtWorks para pegar a lista de palavras.

O método foi inventado por Arnold G. Reinhold, ele mantém o site oficial do diceware.

O método consiste em usar o dado para sortear palavras da lista. Jogue o dado 5 vezes por palavra. Os dois primeiros números são as paginas do livreto e os três últimos, a palavra.

Por exemplo: você tirou no dado: 2-6-5-1-3 : vá na pagina 2,6 do livreto e procure a palavra 513 = egípcio.

EFF Passphrase Dice

Quantas palavras usar?

Como para cada palavra jogamos o dado 5 vezes, a lista de palavras tem *6^5 = 7776* palavras e conforme calculamos isso resulta em 12,92 bits de entropia por palavra sorteada, então **é indicado usar 6 palavras**

Aprimorando o Diceware:

Para gerar senhas ainda mais seguras, usando o diceware, podemos:

- Colocar alguns caracteres especiais e números entre as palavras (no livreto tem uma tabela, para sorteá-los usando o dado).
- Colocar aleatoriamente algumas letras em maiúscula.
- Pessoalmente, gosto de, ao invés de sortear 6 palavras, sortear 5 e, para a última, uso uma palavra que não existe na lista, nem em dicionários, etc… usando por exemplo um termo técnico, ou nome de uma marca etc…

Preciso usar um dado de verdade para gerar as senhas?

Essa é uma longa discussão, mas resumidamente: usando um dado você terá uma boa garantia que nada dará errado, porém gerar os números aleatórios usando um computador normalmente é bom o suficiente.

Dito isso, fiz esse código em Python para gerar uma senha (ele também gera números e caracteres especiais), a principal função dele é:

```
from random import SystemRandom as cryptogen
def GenDiceware(filename, n=6):
    with open(filename) as f:
        words = [line.strip() for line in f]
    return [cryptogen().choice(words) for _ in range(n)]
```

Observação 1: além da lista de palavras do livreto, você pode usar um pacote chamado wbrazilian link (270 mil palavras = 18 bits de entropia / palavra).

Observação 2: O que você **DEVE** atentar nesse código é o uso da classe SystemRandom, que utiliza um gerador de número pseudo-aleatório **CRIPTOGRAFICAMENTE SEGURO (CPRNG)**

Exemplo de senha gerada:

> evocar encher acerto 833?LANCHE# azul >Caloria<

Ela tem 47 caracteres e segundo o Rumkin ela tem 240 bits de entropia, ou seja uma senha bastante forte e fácil de lembrar \o/

Em suma, o diceware é um método de gerar senhas que é:

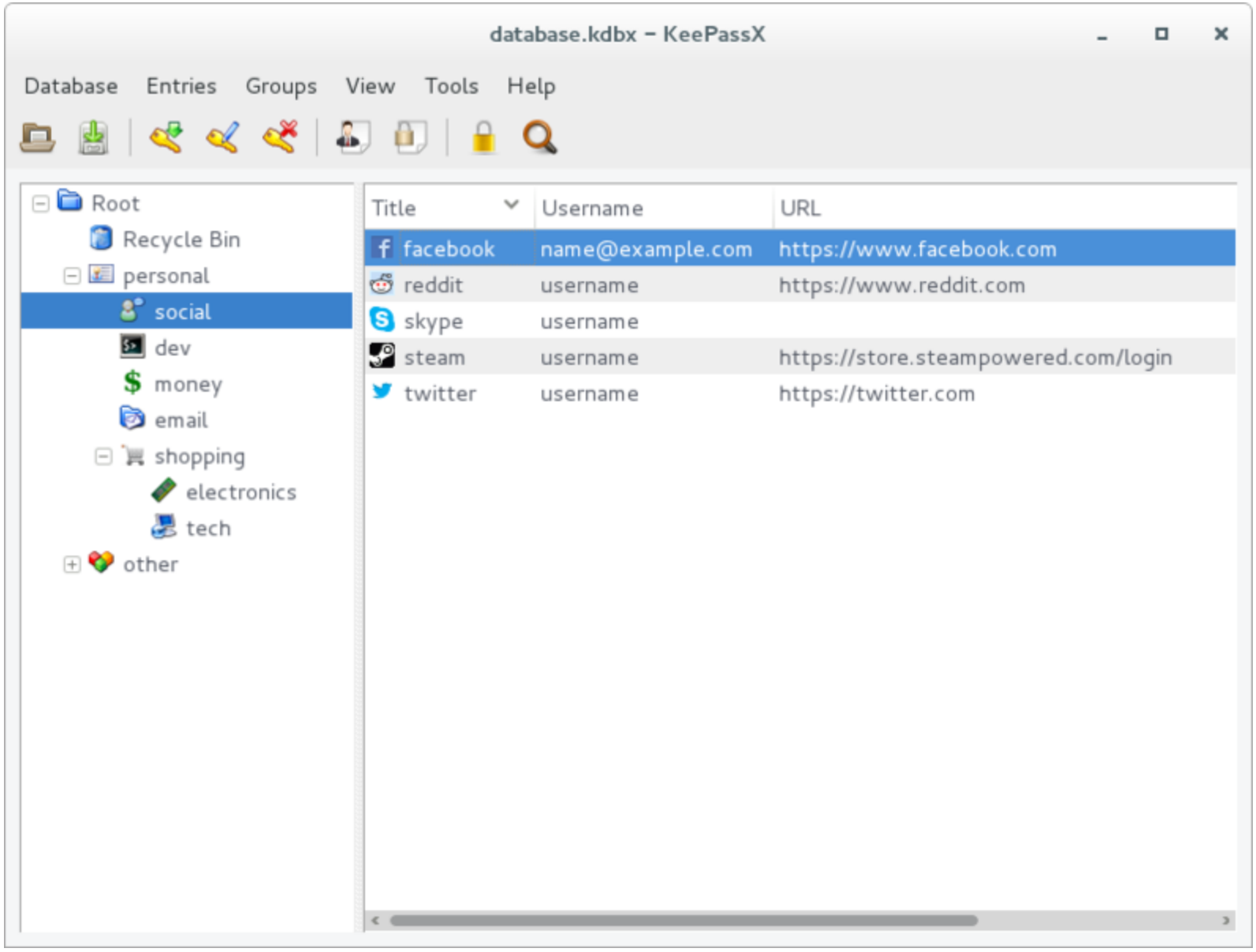
- Harder: as senhas geradas são verdadeiramente aleatórias as tornando bem mais difíceis de quebrar.
- Better: são melhores para lembrar.
- Faster: é mais rápido de pensar numa senha, basta jogar o dado (ou fazer no computador).
- Stronger: As senhas geradas são fortes, pois tem mais entropia.

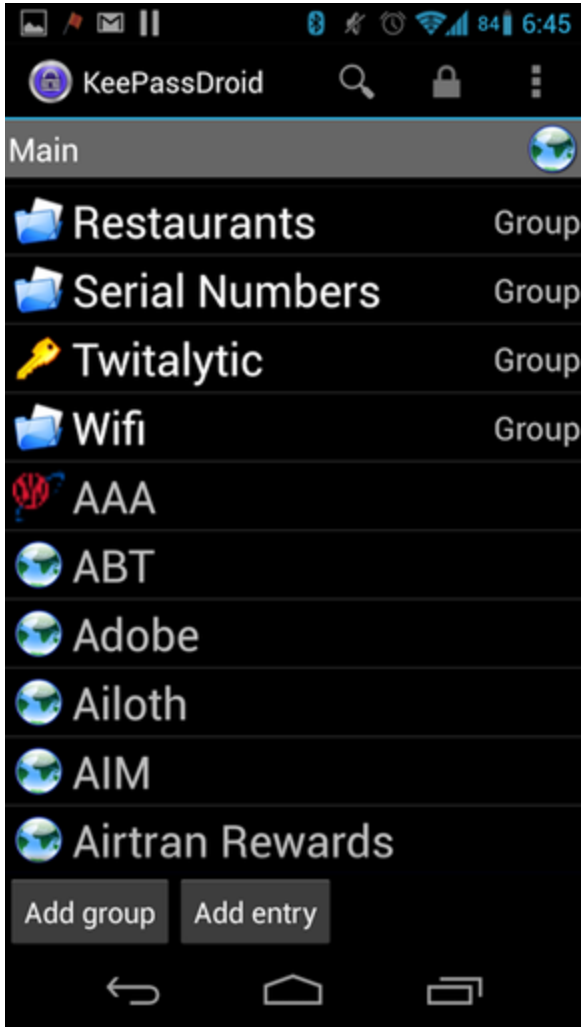
Gerenciador de senhas (Password Manager)

Agora que sabemos como avaliar se uma senha é boa e como gerar boas senhas, ainda precisamos resolver o seguinte problema: precisamos ter uma senha para cada serviço.

Uma solução para isso é um password manager (gerenciador de senhas), é um programa que cria boas senhas aleatórias automaticamente, para cada site individualmente, guardando todas essas senhas trancadas com apenas uma senha mestra.

KeePass/KeePassX é minha recomendação padrão, ele é software livre, está disponível nas principais distros GNU/Linux e existe pra Android: o KeePassDroid (na F-Droid) e tem auto-type.


database.kdbx – KeePassX
Database
Entries
Groups
View
Tools
Help
Root
Recycle Bin
personal
social
dev
money
email
shopping
electronics
tech
other
Title
Username
URL
facebook
name@example.com
https://www.facebook.com
reddit
username
https://www.reddit.com
skype
username
steam
username
https://store.steampowered.com/login
twitter
username
https://twitter.com


6:45
KeePassDroid
Main
Restaurants
Group
Serial Numbers
Group
Twitalytic
Group
Wifi
Group
AAA
ABT
Adobe
Ailoth
AIM
Airtran Rewards
Add group
Add entry

O uso dele é: você cria seu banco de senhas (.kdb ou .kdbx), cadastra as entradas de seus serviços e, então, o programa gera uma senha aleatória, como por exemplo:

>)U*%0jqvm,s*&TbZJDiFq[&$GlQ?

Quando você precisar acessar uma senha, você abre o arquivo, ele vai pedir sua senha mestra e então selecione a entrada que você precisa.

Você terá duas opções: dar um control-C (com a entrada selecionada) e control-V no campo de senha (ele limpa automaticamente a senha do seu control-c); ou você pode usar o auto-type: vá em Extras → Configurações → Avançado → e configure um atalho pro auto-type, por fim vá no campo de usuário e digite o atalho então ele vai digitar seu usuário e senha sozinho.

Dica: afim de melhorar ainda mais a segurança do KeePass, vá em configurações do database e clique no reloginho que aparece, Com isso, quando você abrir o KeePassX, ele vai demorar um pouco mais (tipo 1s no total), mas fica mais difícil de quebrar sua senha mestra pois cada tentativa demorará 1s (em computadores tão rápidos quanto o seu).

Como sincronizar minhas senhas com meu celular?

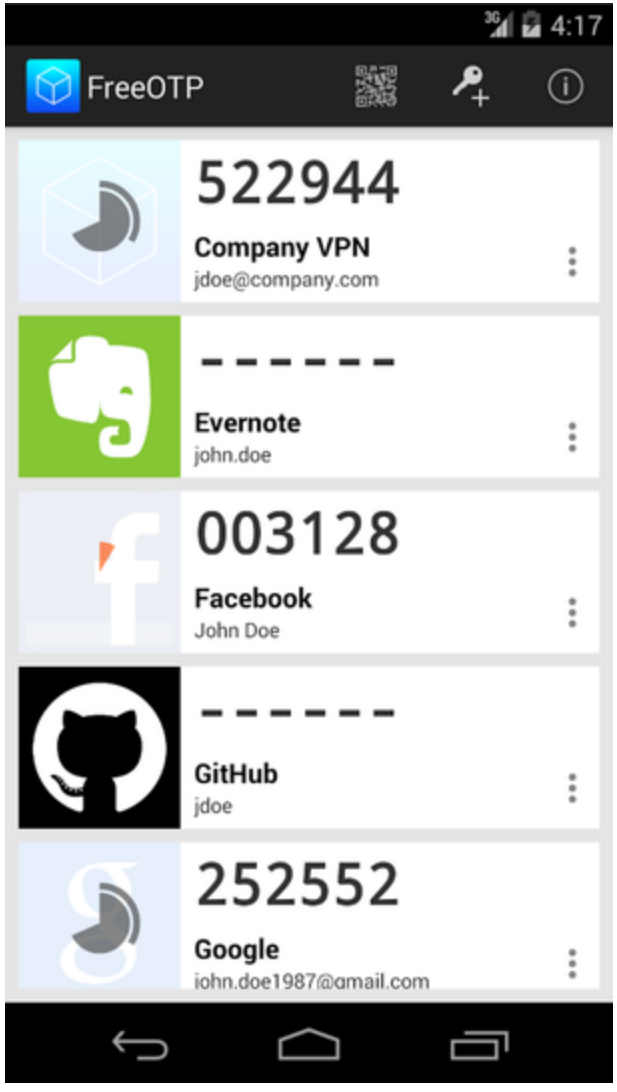
Recomendo o software livre Syncthing é um app pra android e um programa pra pc, no qual ele sincroniza pastas entre seus dispositivos, criando assim sua nuvem pessoal e descentralizada.

Habilite autenticação em duas etapas (2FA)

A ideia por trás da autenticação em duas etapas (2FA -- 2 factor authentication) é que para entrar na sua conta além da senha (que uma coisa que só você sabe), precisa de uma coisa que só você tem, no caso um app no celular.

Como usar: cada serviço é diferente, veja aqui turnon2fa.com como habilitar no serviço especifico. Normalmente, você escaneia um QR code com um app. Recomendo o freetop (é software livre, tem na F-Droid e pra iOS).

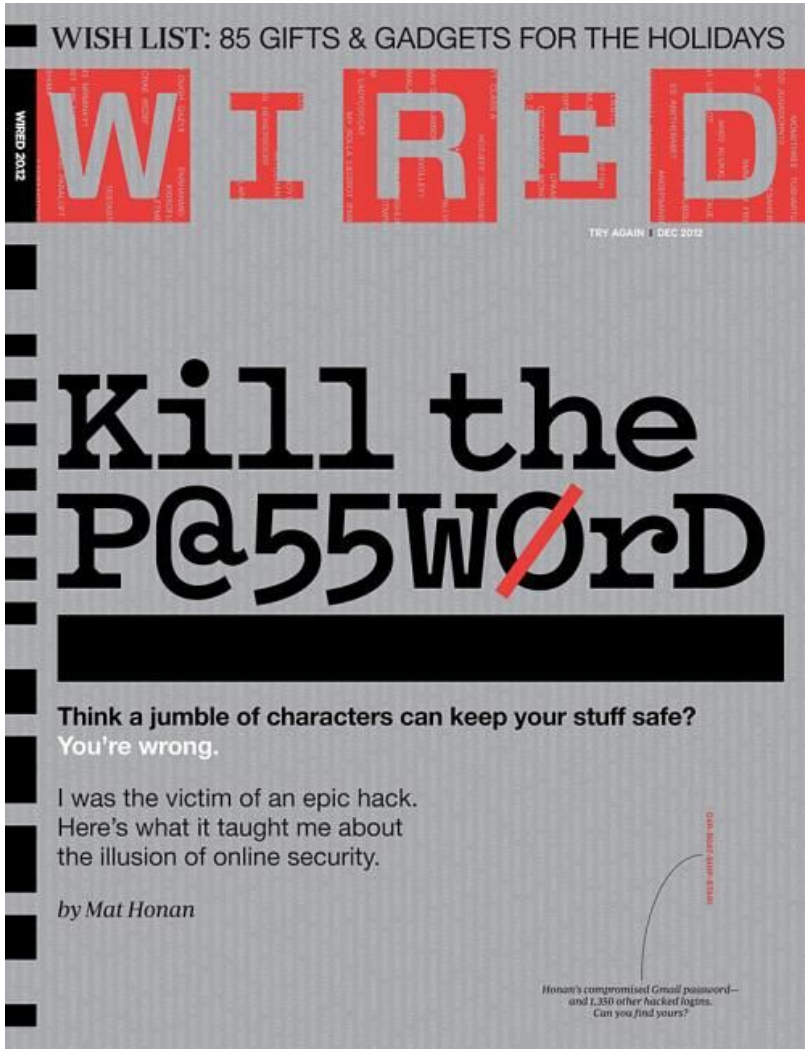
Quando você for entrar na sua conta, após colocar a senha, deverá colocar um número gerado pelo app. Cada número só dura 30s, então não tem problema se alguém ver o código num determinado momento, conforme mostrado abaixo:


4:17
FreeOTP
522944
Company VPN
jdoe@company.com

Evernote
john.doe
003128
Facebook
John Doe

GitHub
jdoe
252552
Google
john.doe1987@gmail.com

Caso você não esteja convencido que precisa disso: veja essa reportagem que foi capa da *Wired* em Dez2012: Kill the Password: A String of Characters Won't Protect You.



Resumidamente: por meio de engenharia social, descobriram algumas informações pessoais dele, ligaram no suporte da Apple, falando que ele tinha esquecido a senha dele e com as informações dele conseguiram responder as perguntas de segurança. Assim, pegaram a senha do iCloud dele, entraram no Find My mac e mandaram deletar remotamente os dispositivos dele.

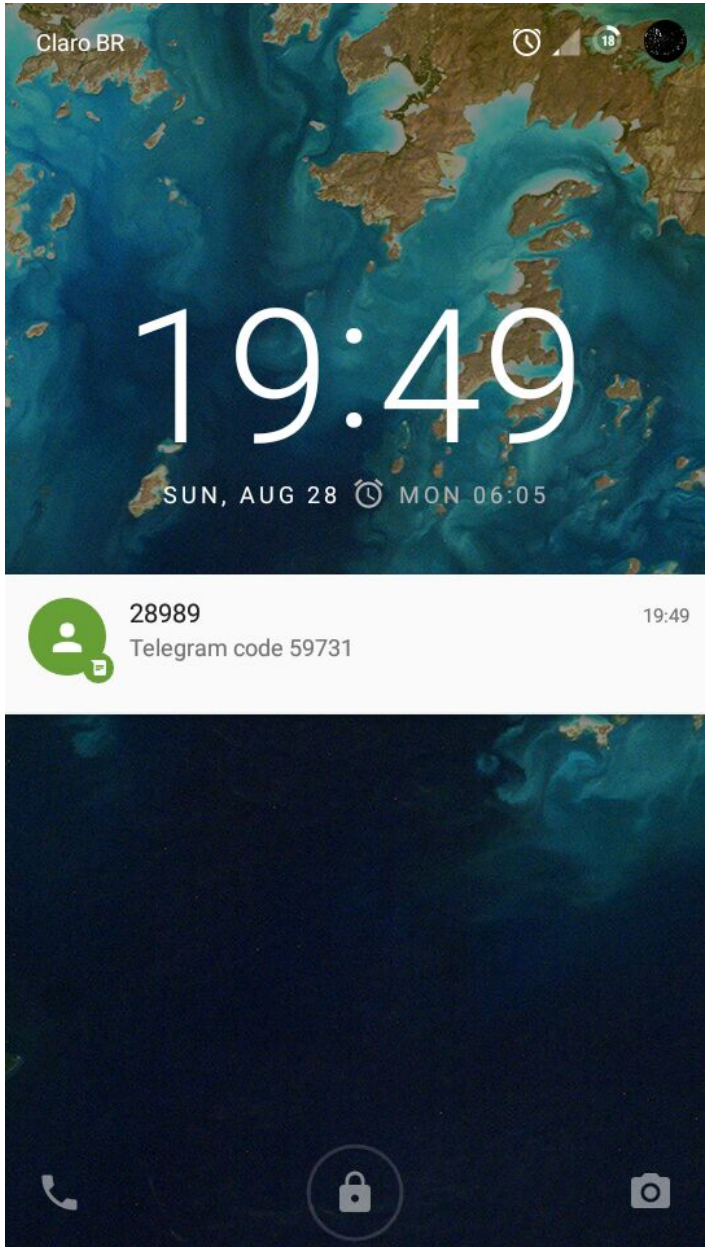
Ou seja, muito embora o Mat Honan usasse senhas robustas, conseguiram “pegar tudo” dele.

Autenticação em duas etapas com SMS:

Alguns serviços, como o Twitter, oferecem autenticação em duas etapas usando um SMS, ao invés de um app. Isso é um problema, pois SMSes e ligações telefônicas podem ser facilmente interceptadas por meio de IMSI Catchers, que são torres de celular falsas que fazem um ataque do tipo homem-no-meio.

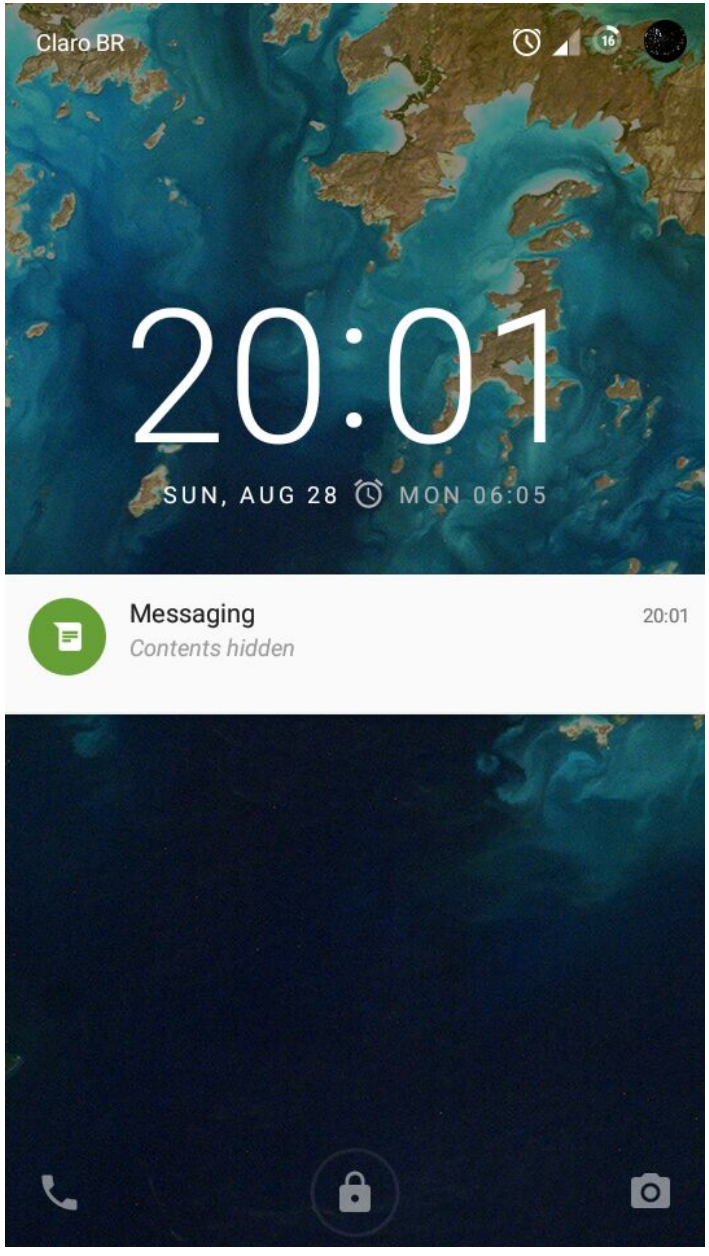
Veja um artigo da oficina anti-vigilância explicando como detectar e se prevenir contra esses ataques: Detectando antenas de celular espiãs.

Além disso, mesmo quando o celular está bloqueado, a notificação mostra a mensagem na tela de bloqueio, por exemplo:


Claro BR
19:49
SUN, AUG 28 MON 06:05
28989
19:49
Telegram code 59731

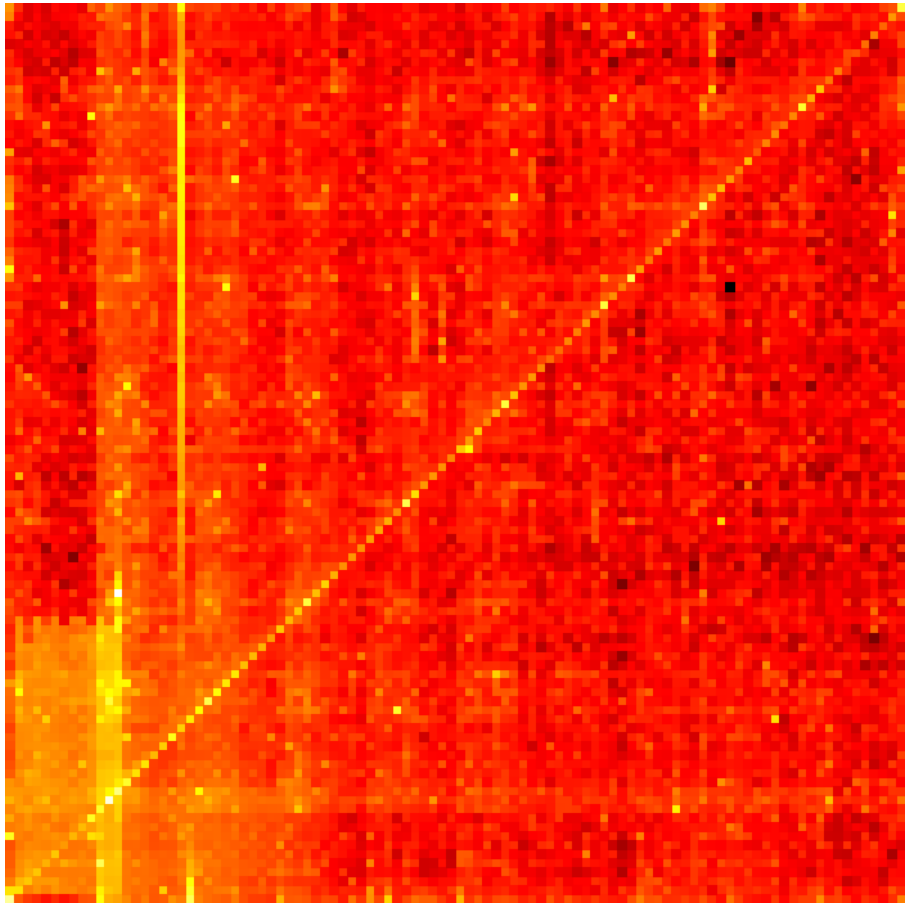
Você pode resolver isso:

- Android: vá em Configurações → Notificações, selecione o app de SMS e configure para não exibir notificações sensíveis.
- Iphone: vá em Ajustes → Notificações → selecione o app de SMS → Mostrar Pré-visualizações e configure.


Claro BR
20:01
SUN, AUG 28 MON 06:05
Messaging
20:01
Contents hidden

PinCodes:

PIN analysis



Em excelente seu post (PIN analysis), Nick Berry analisou 3.4 milhões de PinCodes de 4 digitos (10 mil combinações). E constatou:

- Os top 20 pincodes são responsaveis por 27% do total.
- Estatisticamente 1/3 de todos os PinCodes podem ser advinhados apenas testando 61 combinações.
    - 50% pode ser adivinhado testando apenas 426 combinações (muito menos que 5000).
- PinCodes contendo Anos (19XX) e datas (MMDD) são muito frequentes.

Padrão de desbloqueio do Android

Se você usa o padrão do Android (aquela gradezinha 3x3): existem 389112 possibilidades que podem ser usadas (ou seja **usar um PinCode verdadeiramente aleatorio de 6 digitos é um pouco melhor**).

Uma pesquisadora norueguesa chamada Marte Løge em sua tese de mestrado mostrou que mais de 10% dos padrões analisados, usa como padrão de desbloqueio uma "letra" que é a inicial do companheiro, filho, ou algo parecido.

veja o artigo na arstechnica falando sobre o assunto: New data uncovers the surprising predictability of Android lock patterns

Recomendação: Use uma senha forte pra criptografia de disco e um PinCode de (pelo menos) 6 digitos pra desbloquear o celular. Tem um App chamado SnooperStopper (FDroid) ele depois de 3 tentativas erradas na tela de desbloqueio o celular reinicia , pedindo a senha forte.

Recomendações Finais:

- Como Criptografar e-mails com PGP
- O Guia Motherboard para não ser hackeado basicamente:
  - mantenha seu sistema e seus programas atualizados.
  - use um password manager.
  - Habilite 2FA.
  - Faça Backups.
  - FIQUE TRANQUILÃO.
- quidsup-Do I need AntiVirus in Linux : Ele mostra alguns passos para manter seu PC com GNU/Linux seguro sem precisar de um antivírus.

Obrigado

boas senhas a tod@s