CSE508 Network Security

Stony Brook University

2021-03-18

Authentication

Michalis Polychronakis

Stony Brook University

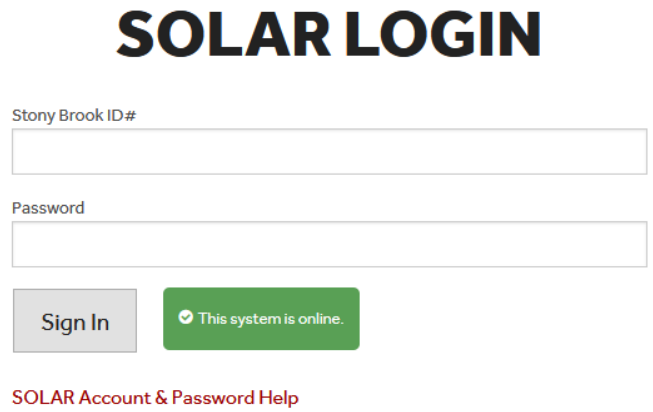
Authentication

The process of verifying someone's identity or role

- User, device, service, request, …

What is identity?

- Which characteristics uniquely identify an entity?



Authentication is a critical service

- Enables communicating parties to verify the identity of their peers
- Many other security mechanisms rely on it

Two main types

- Human to computer
- Computer to computer

Credentials

Evidence used to prove an identity

User Authentication: credentials supplied by a person

- *Something you know*
- *Something you have*
- *Something you are*

Computer authentication: crypto, location

- Computers (in contrast to humans) can "remember" large secrets (keys) and perform complex cryptographic operations
- Location: evidence that an entity is at a specific place (IP, subnet, switch port, …)

Authentication can be delegated

- The verifying entity accepts that a trusted third party has already established authentication

Something You Know: Password-based Authentication

Passwords, passphrases, pins, key-phrases, access codes, …

Good passwords are easy to remember and hard to guess

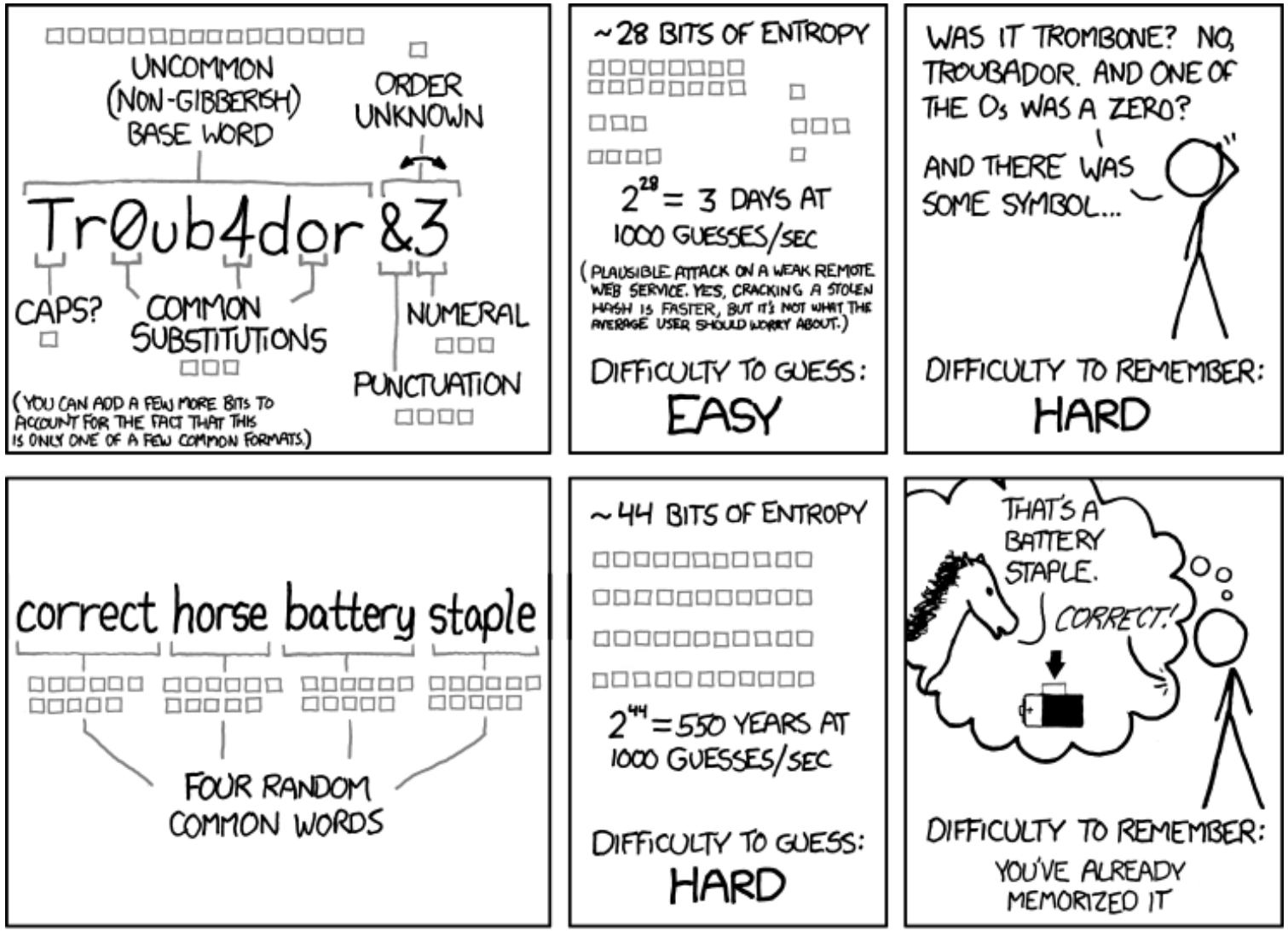
- Easy to remember ➔ easy to guess
- Hard to guess ➔ hard to remember
- Bad ideas: date of birth, SSN, zip code, favorite team name, …

Password space (bits) depends on:

- Password length
- Character set

Better way to think about strong passwords

- **Long passphrases**
- Can be combined with custom variations, symbols, numbers, capitalization, …



Password Policies (often have the opposite effect)

Password rules (often miss the point)

"At least one special character," "Minimum/Maximum length of 8/12 characters," "Must contain at least one number," "Must contain at least one capital letter"

Make passwords hard to remember! ➔ encourage password reuse

Better: encourage long passphrases, evaluate strength on-the-fly

Periodic password changing

"You haven't changed your password in the last 90 days"

Probably too late anyway if password has already been stolen

Makes remembering passwords harder ➔ more password resets

Hinders the use of password managers (!)

What users do: `password1` ➔ `password2` ➔ `password3` ➔ …

Attacking Passwords

Offline cracking

Online guessing

} *Brute force attacks*

Eavesdropping

Capturing

Password Storage

Storing passwords as plaintext is disastrous

Better way: store a cryptographic hash of the password

Even better: store the hash of a "salted" version of the password

Defend against *dictionary attacks*: prevent precomputation of hash values (wordlists of popular passwords, rainbow tables, …)

Even if two users happen to have the same password, their hash values will be different ➔ need to be cracked separately

Salting *does not* make brute-force guessing a given password harder!

Username	Salt	Password hash
Bobbie	4238	h(4238, $uperman)
Tony	2918	h(2918, 63%TaeFF)
Mitsos	6902	h(6902, zour1da)
Mark	1694	h(1694, Rockybrook#1)

Password databases are still getting leaked…

Password Cracking

Exhaustive search ➔ infeasible for large password spaces

Dictionary attacks (words, real user passwords from previous leaks, …)

Variations, common patterns, structure rules

- Prepend/append symbols/numbers/dates, weird capitalization, l33tspeak, visually similar characters, intended misspellings, …

Target-specific information

- DOB, family names, favorite team, pets, hobbies, anniversaries, language, slang, …
- Easy to acquire from social networking services and other public sites
- **Particularly effective against "security questions"**

Advanced techniques

- Probabilistic context-free grammars, Markov models, …

example_hashes [hashcat

Secure | https://hashcat.net/wiki/doku.php?id=example_hashes

hashcat
advanced password recovery

hashcat Forums Wiki Tools Events

Recent changes Log In Sitemap

Example hashes

Table of Contents

If you get a "line length exception" error in hashcat, it is often because the hash mode that you have requested does not match the hash. To verify, you can test your commands against example hashes.

Unless otherwise noted, the password for all example hashes is **hashcat**.

Generic hash types

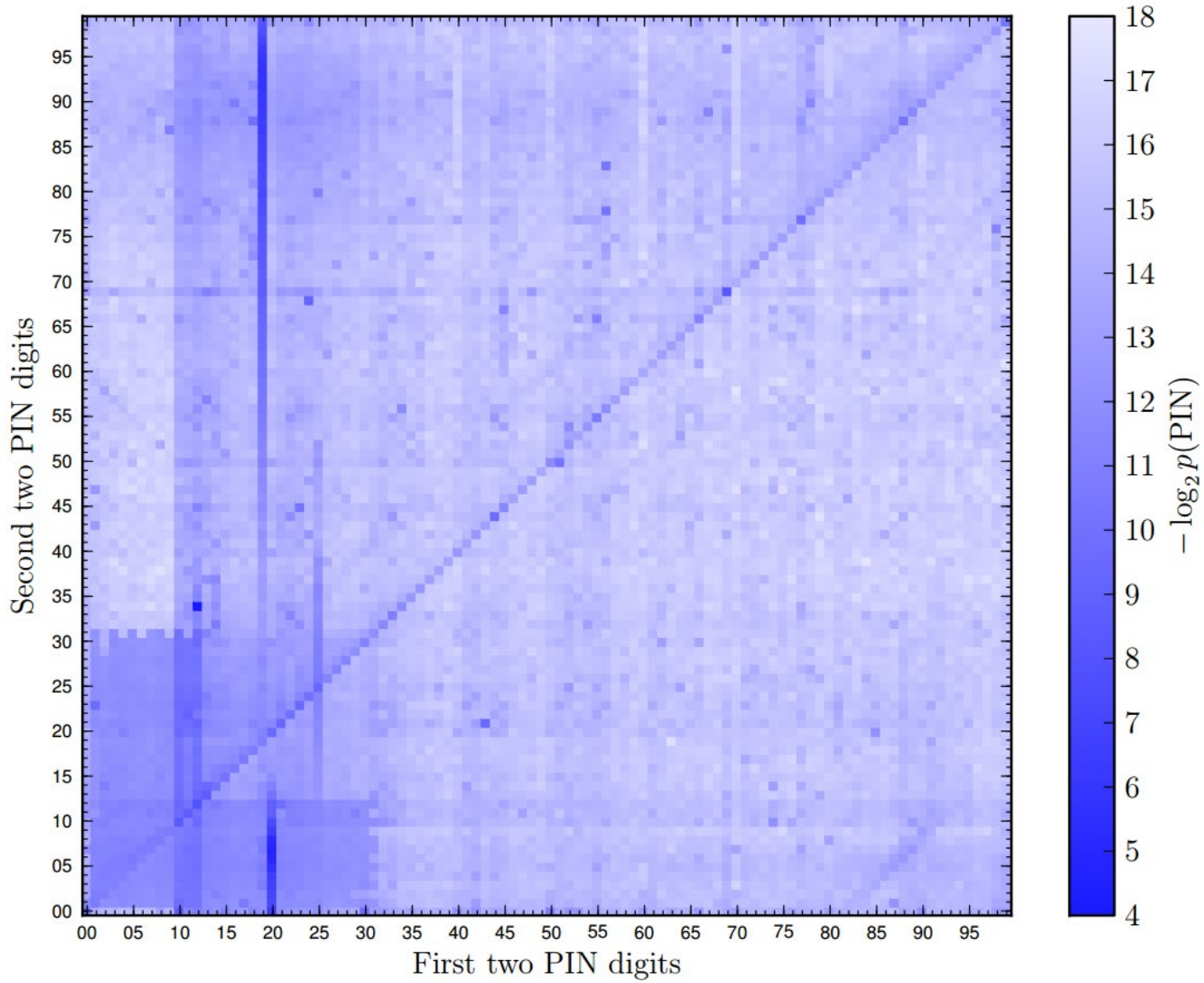
Hash-Mode	Hash-Name	Example
0	MD5	8743b52063cd84097a65d1633f5c74f5
10	md5($pass.$salt)	01dfae6e5d4d90d9892622325959afbe:7050461
20	md5($salt.$pass)	f0fda58630310a6dd91a7d8f0a4ceda2:4225637426
30	md5(utf16le($pass).$salt)	b31d032cfdcf47a399990a71e43c5d2a:144816
40	md5($salt.utf16le($pass))	d63d0e21fdc05f618d55ef306c54af82:13288442151473
50	HMAC-MD5 (key = $pass)	fc741db0a2968c39d9c2a5cc75b05370:1234
60	HMAC-MD5 (key = $salt)	bfd280436f45fa38eaacac3b00518f29:1234
100	SHA1	b89eaac7e61417341b710b727768294d0e6a277b
110	sha1($pass.$salt)	2fc5a684737ce1bf7b3b239df432416e0dd07357:2014
120	sha1($salt.$pass)	cac35ec206d868b7d7cb0b55f31d9425b075082b:5363620024
130	sha1(utf16le($pass).$salt)	c57f6ac1b71f45a07dbd91a59fa47c23abcd87c2:631225
140	sha1($salt.utf16le($pass))	5db61e4cd8776c7969cfd62456da639a4c87683a:8763434884872
150	HMAC-SHA1 (key = $pass)	c898896f3f70f61bc3fb19bef222aa860e5ea717:1234
160	HMAC-SHA1 (key = $salt)	d89c92b4400b15c39e462a8caa939ab40c3aeeea:1234
200	MySQL323	7196759210defdc0
300	MySQL4.1/MySQL5	fcf7c1b8749cf99d88e5f34271d636178fb5d130

50 Most-used (Worse) Passwords

123456	1234567	123	ashley	evite
123456789	qwerty	omgpop	987654321	123abc
picture1	abc123	123321	unknown	123qwe
password	Million2	654321	zxcvbnm	sunshine
12345678	000000	qwertyuiop	112233	121212
111111	1234	qwer123456	chatbooks	dragon
123123	iloveyou	123456a	20100728	1q2w3e4r
12345	aaron431	a123456	123123123	5201314
1234567890	password1	666666	princess	159753
senha	qqww1122	asdfghjkl	jacket025	0123456789

https://en.wikipedia.org/wiki/List_of_the_most_common_passwords

Distribution of 4-digit sequences within RockYou passwords



A birthday present every eleven wallets? The security of customer-chosen banking PINs. Joseph Bonneau, Sören Preibusch and Ross Anderson – FC '12

Wordlists

ce#ebc.dk
goddess5
20071002
271075711
zs3cu7za
scoopn
frygas1411
SL123456sl
12345687ee123
xuexi2010
daigoro
12345614
DICK4080
567891234
tilg80
6z08c861
:zark:
ravishsneha
150571611369
661189
passme
trolovinasveta
abdulkhaleque
007816
xLDSX
Florida2011
037037
WestC0untry
hitsugaiya
955998126
3n3rmax

4637324
bugger825
marmaris
jinjin111
170383gp
3484427
fl33321
zwqrfg
67070857
432106969
6856
704870704870
pv041886
20060814
512881535
milanimilani
472619
dbyxw888
85717221
cc841215
ariana19321
bbbnnn
ang34hehiu
wj112358
Brenda85
786525pb
shi461988
pingu
yeybozip
71477nak
stokurew

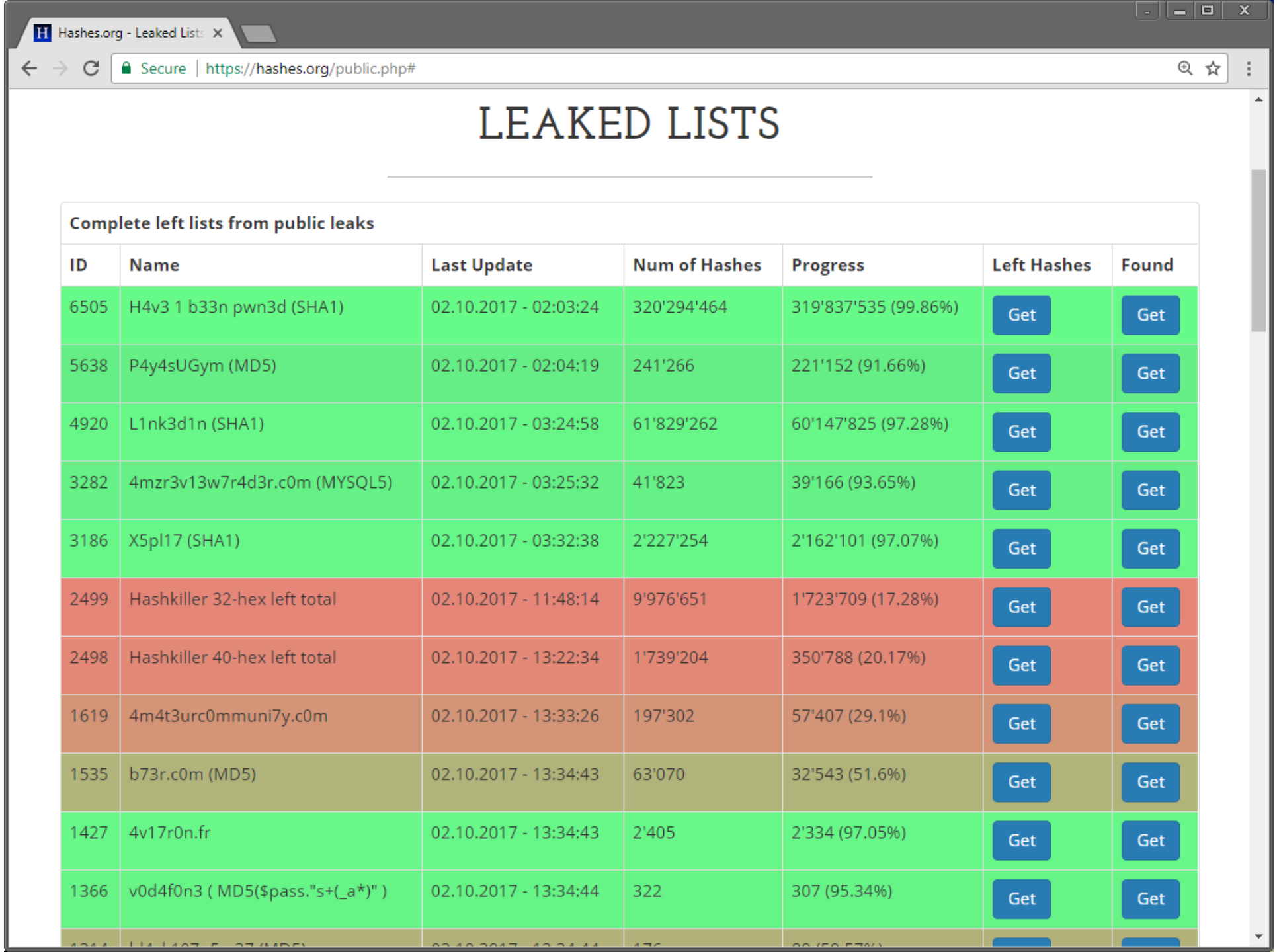
gea8mw4yz
kukumbike
260888
jordi10
lexusis
kj011a039
c84bwlrb
priyanka05
loveneverdies
u8Aqebj576
FGYfgy77
659397
327296
74748585
19720919
050769585
nicopa
2232566
bearss
n0tpublic
isitreal00
ashraf19760
48144
22471015
antyzhou115
0167005246
ec13kag
226226226226
6767537/33
mimilebrock
gueis8850

fujinshan
counter
N8mr0n
520057
adc123
bmaster
qbjh04zg
ueldaa79
EMANUELLI
yanjing
assynt
62157173
0704224950753
6903293
axaaxa
hilall
30091983
2510618981
soukuokpan
tosecondlife
p4os8m6q
015614117
acw71790
lsyljm2
2xgialdl
gaybar9
88203009
MKltyh87
quiggle
2063775206
fr3iH3it

masich
pengaiwei
coalesce
56402768
thesis
aabbcc894
marion&maxime
614850
ydz220105
584521584521
txudecp
84410545
pietro.chiara
jman1514
heryarma
39joinmam
timelapse
mwinkar
251422
willrock
YHrtfgDK
xys96exq
mercadotecnia
8s5sBEx7
0125040344
margitka
omaopa
dfTi6nh
1314520521
pixma760
pearpear

gothpunksk8er
rftaeo48
8d7R0K
5172032
aics07
34mariah
dongqinwei
samarica
cap10l4
0167387943
AE86Trueno
19700913
mcsuap
bu56mpbu
danbee
passw<>
money521
conan83
nxfjpl
rateg143
kojyihen
058336257
sarah4444
7363437
freindship
JytmvWO848
sb inbau
30907891
0515043111
1973@ati
wlxgjf

20081010
leelou44
8UfjeGbO
200358808
dellede
liang123.
captainettekt
kwiki-mart
mdovydas
tigmys2001
denial
678ad5251
woaiwuai
1591591591212
hNbDGN
cardcap
13985039393
001104
desare11
412724198
nibh1kab
asferg
hqb555
xgames7
muckerlee
choqui67
12130911
lierwei120
skytdvn
milena1995
kambala11



LEAKED LISTS

Complete left lists from public leaks

ID	Name	Last Update	Num of Hashes	Progress	Left Hashes	Found
6505	H4v3 1 b33n pwn3d (SHA1)	02.10.2017 - 02:03:24	320'294'464	319'837'535 (99.86%)	Get	Get
5638	P4y4sUGym (MD5)	02.10.2017 - 02:04:19	241'266	221'152 (91.66%)	Get	Get
4920	L1nk3d1n (SHA1)	02.10.2017 - 03:24:58	61'829'262	60'147'825 (97.28%)	Get	Get
3282	4mzr3v13w7r4d3r.c0m (MYSQL5)	02.10.2017 - 03:25:32	41'823	39'166 (93.65%)	Get	Get
3186	X5pl17 (SHA1)	02.10.2017 - 03:32:38	2'227'254	2'162'101 (97.07%)	Get	Get
2499	Hashkiller 32-hex left total	02.10.2017 - 11:48:14	9'976'651	1'723'709 (17.28%)	Get	Get
2498	Hashkiller 40-hex left total	02.10.2017 - 13:22:34	1'739'204	350'788 (20.17%)	Get	Get
1619	4m4t3urc0mmuni7y.c0m	02.10.2017 - 13:33:26	197'302	57'407 (29.1%)	Get	Get
1535	b73r.c0m (MD5)	02.10.2017 - 13:34:43	63'070	32'543 (51.6%)	Get	Get
1427	4v17r0n.fr	02.10.2017 - 13:34:43	2'405	2'334 (97.05%)	Get	Get
1366	v0d4f0n3 (MD5($pass."s+(_a*)")	02.10.2017 - 13:34:44	322	307 (95.34%)	Get	Get

Have I Been Pwned: Check if yc

https://haveibeenpwned.com

518	10,624,652,379	113,998	199,730,234
pwned websites	pwned accounts	pastes	paste accounts

Largest breaches

772,904,991	Collection #1 accounts
763,117,241	Verifications.io accounts
711,477,622	Onliner Spambot accounts
622,161,052	Data Enrichment Exposure From PDL Customer accounts
593,427,119	Exploit.In accounts
457,962,538	Anti Public Combo List accounts
393,430,309	River City Media Spam List accounts
359,420,698	MySpace accounts
268,765,495	Wattpad accounts
234,842,089	NetEase accounts

Recently added breaches

11,788	WeLeakInfo accounts
465,141	Liker accounts
637,279	Travel Oklahoma accounts
66,521	Gab accounts
1,834,006	Oxfam accounts
1,921,722	Ticketcounter accounts
20,339,937	SuperVPN & GeckoVPN accounts
645,786	Filmai.in accounts
10,585	NurseryCam accounts
358,822	People's Energy accounts

Password Hashing Functions

Hash functions are very fast to evaluate ➔ facilitate fast password cracking

Solution: slow down the guessing process (password *"stretching"*)

- Benefit: cracking becomes very inefficient (e.g., 10-100ms per check)
- Drawback: increased cost for the server if it must authenticate many users

Make heavy use of available resources

- Fast enough computation to validate honest users, but render password guessing infeasible
- Adaptable: flexible cost (time/memory complexity) parameters

Bcrypt [Provos and Mazières, 1999]

- Cost-parameterized, modified version of the Blowfish encryption algorithm
- Tunable cost parameter (exponential number of loop iterations)

Alternatives: Scrypt (memory-hard), PBKDF2 (PKCS standard)

Online Guessing

Similar strategy to offline guessing, but rate-limited

- Connect, try a few passwords, get disconnected, repeat…

Prerequisite: know a valid user name

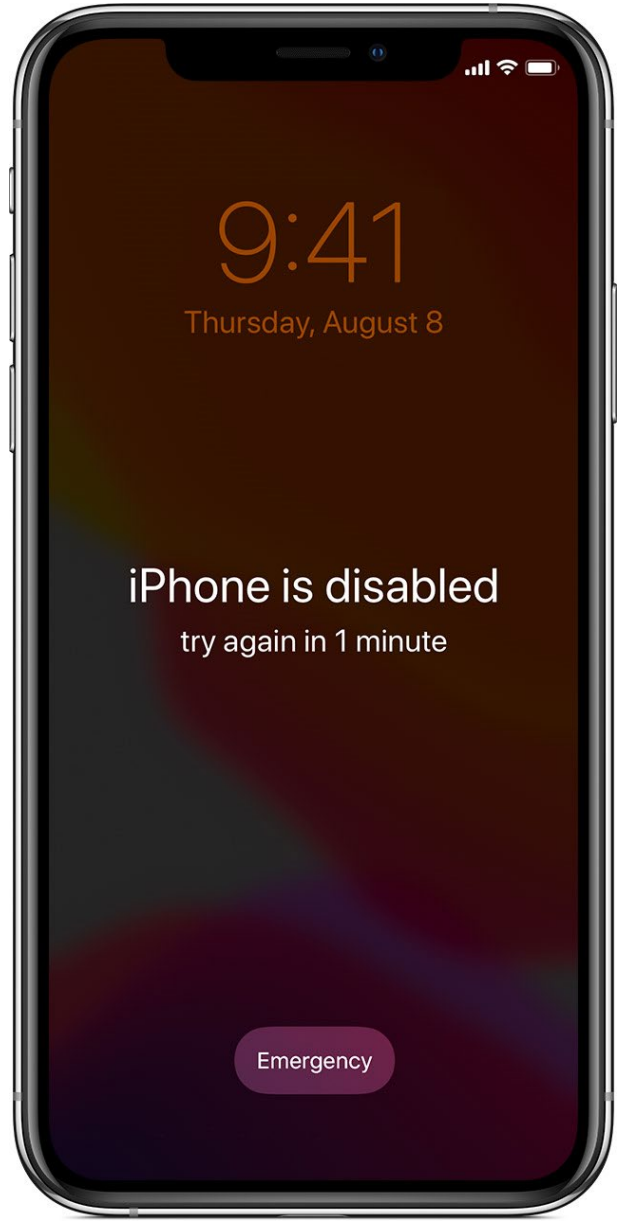
- **Credential stuffing**: try username + password combinations from previous breaches

Many failed attempts can lead to a system reaction

- Introduce delay before accepting future attempts (exponential backoff)
- Shut off completely (e.g., ATM capturing/disabling the card after 3 tries)
- Ask user to solve a CAPTCHA

Very common against publicly accessible SSH, VPN, RDP, and other servers

- Main reason people move sshd to a non-default port
- Fail2Ban: block IP after many failed attempts ➔ attackers may now be able to lock you out
- Better: disable password authentication and use a key pair ➔ cumbersome if having to log in from many/others' computers


9:41
Thursday, August 8
iPhone is disabled
try again in 1 minute
Emergency

```
LOGIN: mitch
PASSWORD: FooBar!-7
SUCCESSFUL LOGIN
```

(a)

```
LOGIN: carol
INVALID LOGIN NAME
LOGIN:
```

(b)

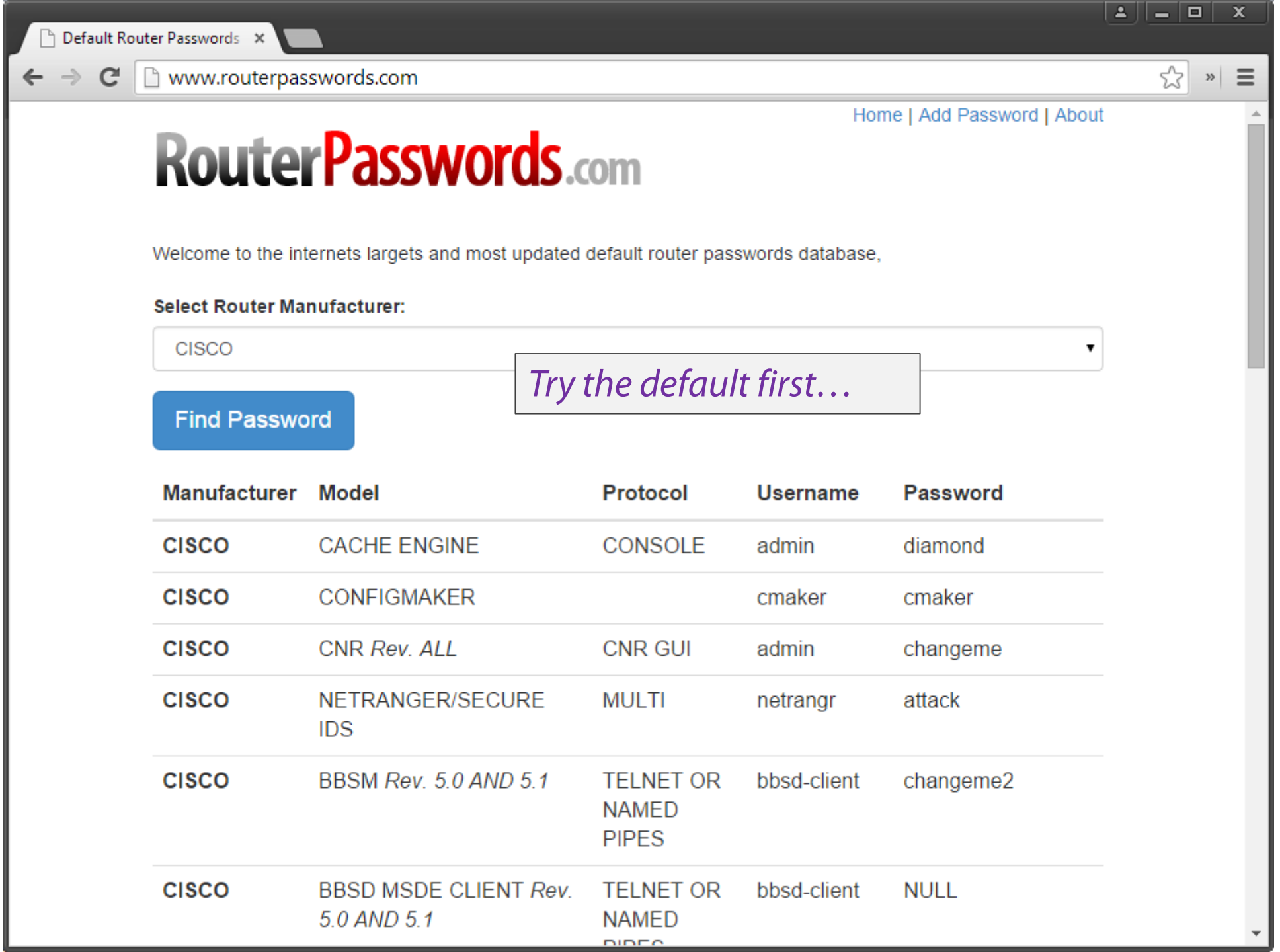
```
LOGIN: carol
PASSWORD: Idunno
INVALID LOGIN
LOGIN:
```

(c)

(a) Successful login

(b) Login rejected after name is entered

(c) Login rejected after name and password are typed ➔ less information makes guessing harder


Default Router Passwords
www.routerpasswords.com
Home | Add Password | About
RouterPasswords.com
Welcome to the internets largets and most updated default router passwords database,
Select Router Manufacturer:
CISCO
Try the default first…
Find Password
Manufacturer Model Protocol Username Password
CISCO CACHE ENGINE CONSOLE admin diamond
CISCO CONFIGMAKER cmaker cmaker
CISCO CNR Rev. ALL CNR GUI admin changeme
CISCO NETRANGER/SECURE IDS MULTI netrangr attack
CISCO BBSM Rev. 5.0 AND 5.1 TELNET OR NAMED PIPES bbsd-client changeme2
CISCO BBSD MSDE CLIENT Rev. 5.0 AND 5.1 TELNET OR NAMED bbsd-client NULL

Eavesdropping and Replay

Physical world

- Watch user type password (shoulder surfing)
- Cameras (e.g., ATM skimmers)
- Lift fingerprints (e.g., Apple Touch ID)
- Post-it notes, notebooks, …

Network makes things easier

- Sniffing (LAN, WiFi, …)
- Man-in-the-Middle attacks

Defenses

- Encryption
- One-time password schemes

Kerberos

Long-lived vs. session keys

- Use long-lived key for authentication and negotiating session keys
- Use "fresh," ephemeral session keys (prevent replay, cryptanalysis, old compromised keys) for encrypted communication, MACs, …

Kerberos: most widely used (non-web) single sign-on system

- Originally developed at MIT, now used in Unix, Windows, …

Authenticate users to services: using their password as the initial key, without having to retype it for every interaction

- A Key Distribution Center (KDC) acts as a trusted third party for key distribution
- Online authentication: variant of Needham-Schroeder protocol
- Assumes a non-trusted network: prevents eavesdropping
- Assumes that the Kerberos server and user workstations are secure…

Use cases: workstation login, remote share access, printers, …

Password Capture

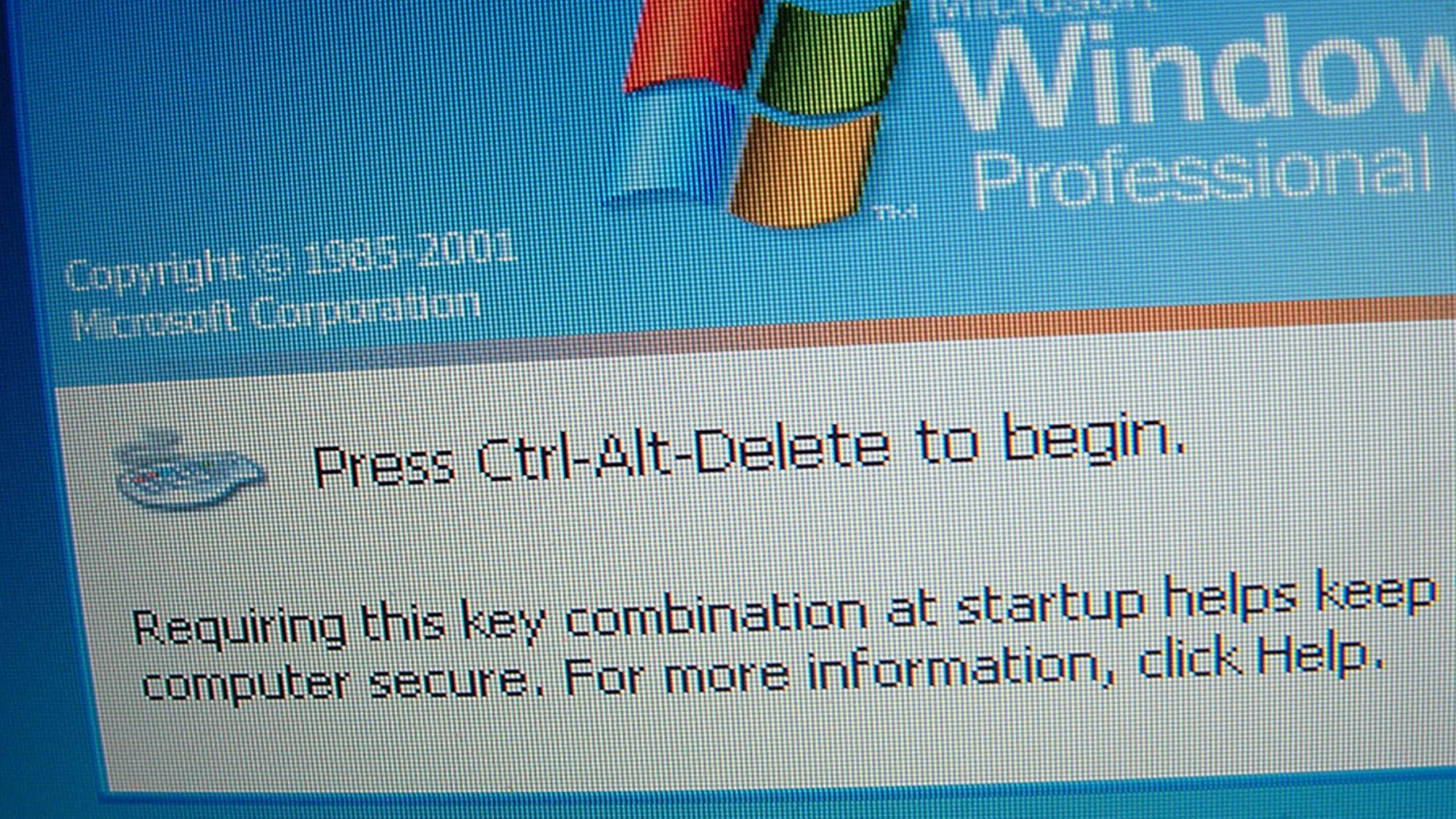
Hardware bugs/keyloggers

Software keyloggers/malware

Cameras

Phishing

Social engineering


Windows
Professional
Copyright © 1985-2001
Microsoft Corporation
Press Ctrl-Alt-Delete to begin.
Requiring this key combination at startup helps keep your computer secure. For more information, click Help.

(a) Correct login screen

(b) Phony login screen

Something You Have: Authentication Tokens

One-time passcode tokens

- Time-based or counter-based

Various other authentication tokens

- Store certificates, encryption keys, challenge–response, …

Smartcards (contact or contactless)

- Identification, authentication, data storage, limited processing
- Magnetic stripe cards, EMV (chip-n-pin credit cards), SIM cards, RFID tags, …

USB/NFC tokens, mobile phones, watches, …

- Can be used as authentication devices

Multi-factor Authentication

Must provide several separate credentials of different types

Most common: *two-factor authentication (2FA)*

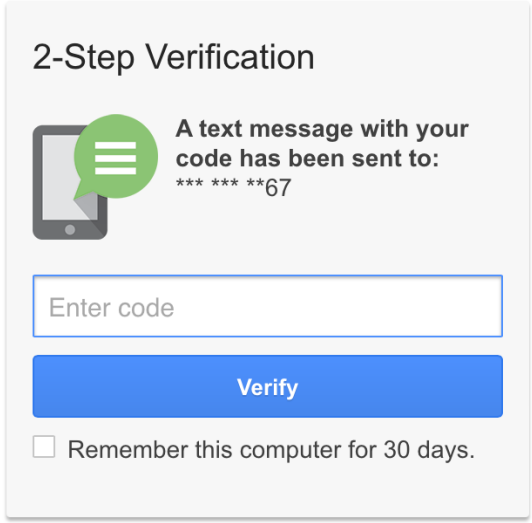
- Example: Password + hardware token/SMS message/authenticator app, …
- Example: ATM card + PIN

Motivation: a captured/cracked password is not enough to compromise a victim's account ➔ **not always true**

- *Man-in-the-Middle:* set up fake banking website, relay password to real website, let the user deal with the second factor…
- *Man-in-the-Browser:* hijack/manipulate an established session after authentication has completed (banking Trojans)
- *Dual infection:* compromise both PC and mobile device
- **More importantly: the most commonly used 2nd factor (SMS) is the least secure**

SMS Is Not a Secure 2nd Factor

(but still better than no 2nd factor)



Social engineering

- Call victim's mobile operator and hijack the phone number
- SIM swap, message/call forwarding, …

Message interception

- Rogue cell towers: IMSI catchers, StingRays,…
- Some phones even display text messages on the lock screen (!)

SS7 attacks

- The protocol used for inter-provider signaling is severely outdated and vulnerable
- Allows attackers to spoof change requests to users' phone numbers and intercept calls or text messages

'Sim swap' gives fraudste

Secure | https://www.theguardian.com/money/2015/sep/26/sim-swap-fraud-mobile-phone-vodafone-customer

sign in | become a supporter | subscribe | search | jobs | US edition

theguardian

US politics world opinion sports soccer tech arts lifestyle fashion business travel environment | all sections

home

Scams

'Sim swap' gives fraudsters access-all-areas via your mobile phone

There's a new, little-known scam designed to empty your bank account, as one Vodafone customer found to her cost

1908 15

Anna Tims

Saturday 26 September 2015 02.00 EDT

Most popular in US

Las Vegas shooting: death toll rises to 58 as police name suspect – latest updates

Confusion follows reports of Tom Petty death after heart attack

Las Vegas gunman may have used special device to fire faster, expert says

A Hacker Got All My Texts for $16

A gaping flaw in SMS lets hackers take over phone numbers in minutes by simply paying a company to reroute text messages.

By Joseph Cox

March 15, 2021, 1:10pm Share Tweet Snap

I hadn't been SIM swapped, where hackers trick or bribe telecom employees to port a target's phone number to their own SIM card. Instead, the hacker used a service by a company called Sakari, which helps businesses do SMS marketing and mass messaging, to reroute my messages to him. This overlooked attack vector shows not only how unregulated commercial SMS tools are but also how there are gaping holes in our telecommunications infrastructure, with a hacker sometimes just having to pinky swear they

Better Alternative: Authenticator App

Six/eight digit code provided after successful password validation

Time-based one-time password (TOTP)

Code computed from a shared secret key
and the current time (using HMAC)

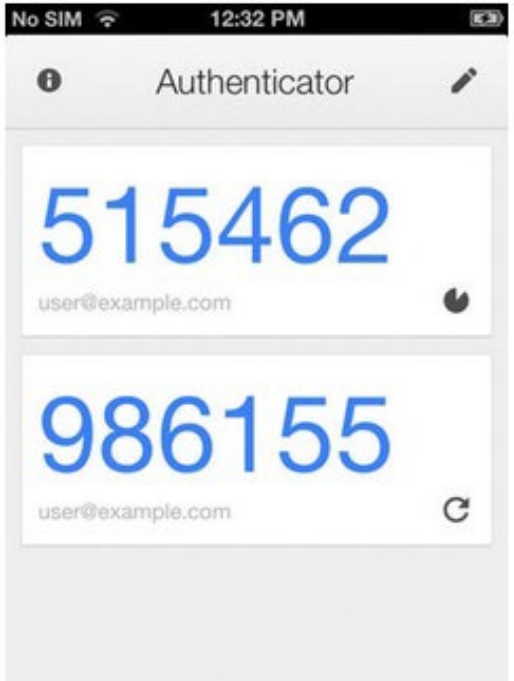
The key is negotiated during registration

Requires "rough" client–server synchronization

Code constantly changes in 30-second intervals

Phishing is still possible!

The attacker just needs to proxy the captured credentials in real time
(rather than collecting them for later use)



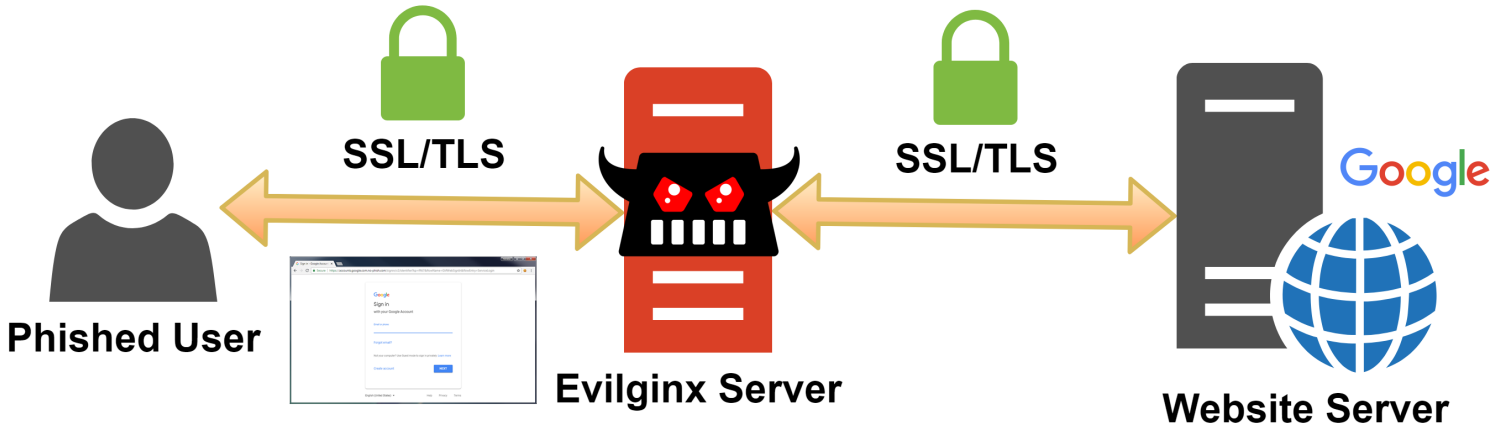
Evilginx2 https://github.com/kgretzky/evilginx2

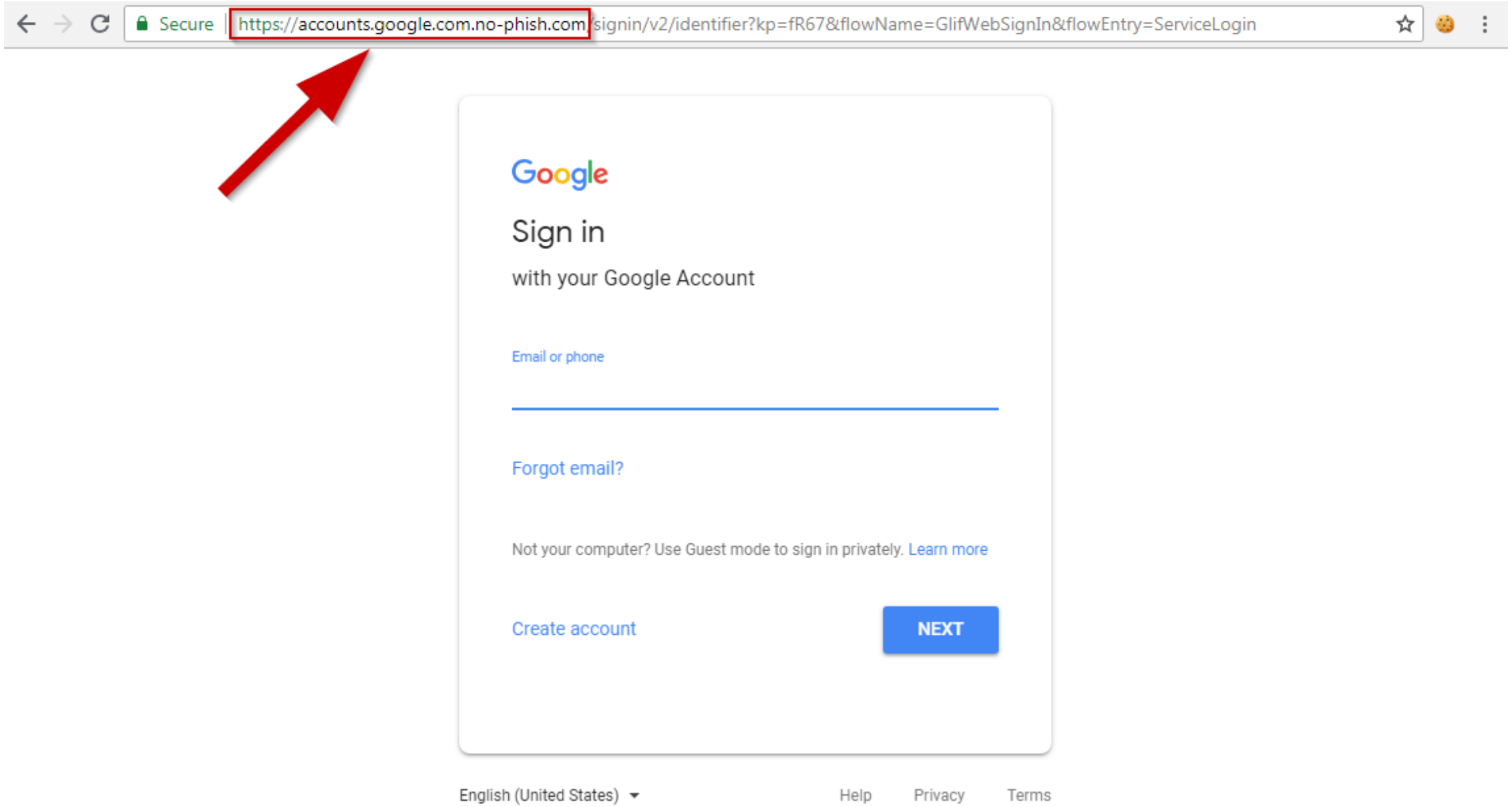
Man-in-the-middle attack framework for phishing login credentials along with session cookies

- Bypasses 2-factor authentication

No need for HTML templates: just a web proxy

- Victim's traffic is forwarded to the real website
- TLS termination at the proxy (e.g., using a LetsEncrypt certificate)

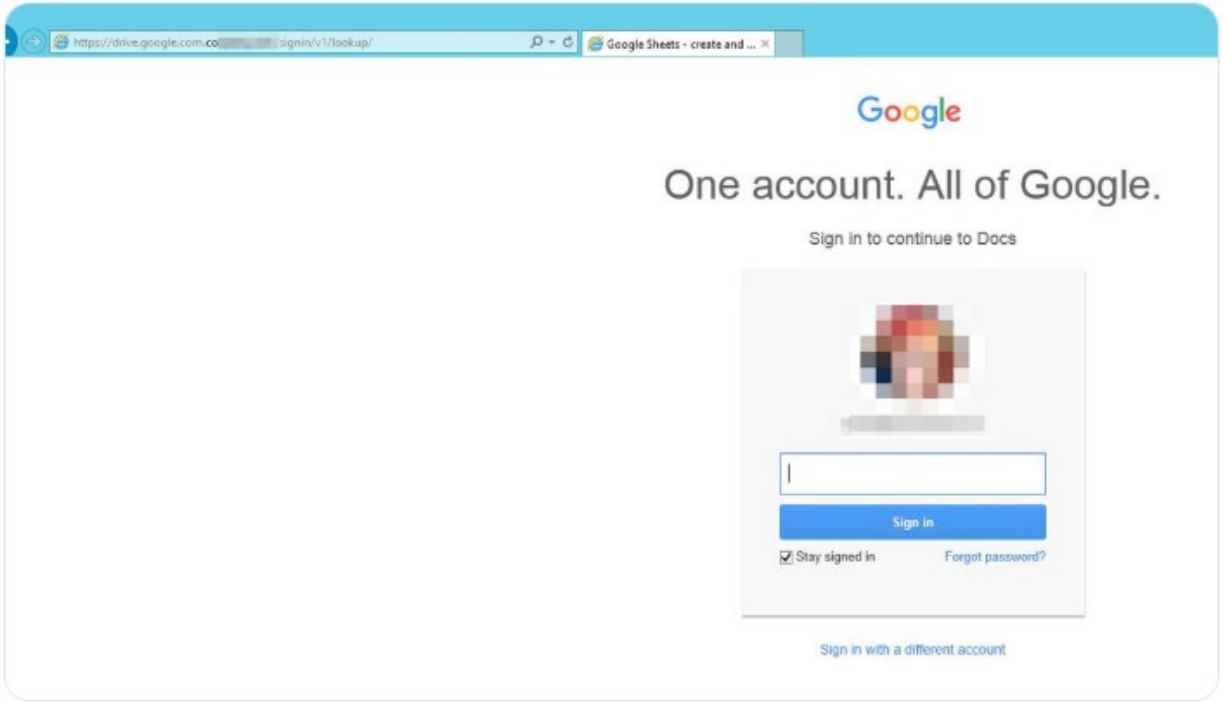




Justin Warner
@sixdub

Follow

I love digging through certificate transparency logs. Today, I saw a fake Google Drive landing page freshly registered with Let's Encrypt. It had a hardcoded picture/email of presumably the target. These can be a wealth of info that I recommend folks checking out.



5:21 PM - 22 Jul 2018

© https://twitter.com/sixdub/status/1021143236943990785

Evilginx2's Tokenized phishing URLs

Scanners look into public certificate transparency logs for newly registered domains

> *"For some phishing pages, it took usually one hour for the hostname to become banned and blacklisted by popular anti-spam filters"*

Solution: create unique phishing URLs

Response to scanner: benign page

`https://totally.not.fake.linkedin.foo.com/auth/signin`

Response to victim: malicious page

`https://totally.not.fake.linkedin.foo.com/auth/signin?tk=secret_token`

Additional countermeasure: temporarily hide the phishing page

While submitting it to bit.ly, sending it through email, appearing on CT log, …

Modlishka https://github.com/drk1wi/Modlishka

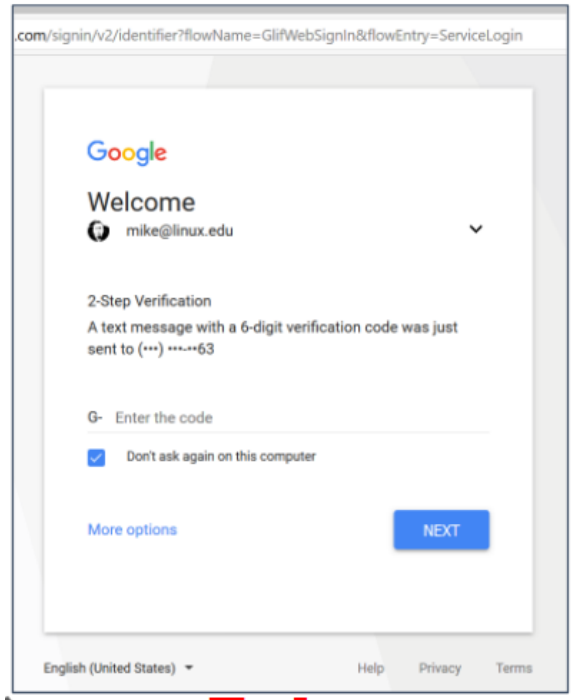
Phishing reverse proxy

- Support for the majority of 2FA authentication schemes
- No website templates
- User credential harvesting (with context based on URL parameter passed identifiers)
- Web panel with a summary of collected credentials and user session impersonation

CredSniper https://github.com/ustayready/CredSniper

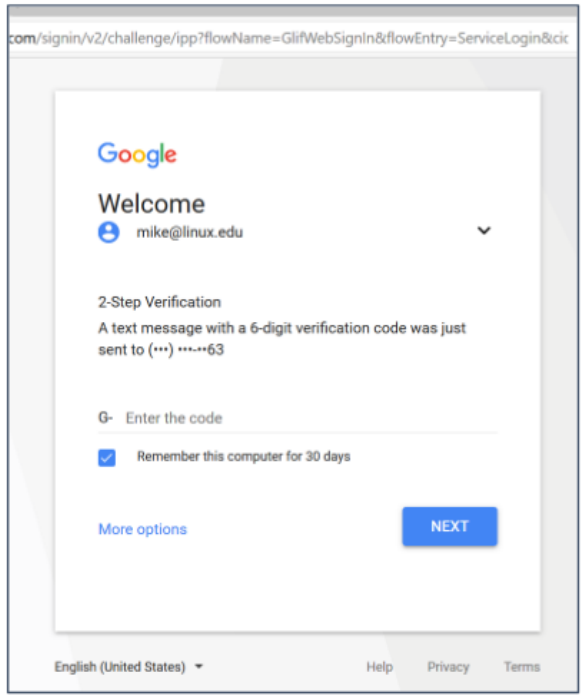
Exact login form clones for realistic phishing

Supports TLS via Let's Encrypt, and phishing 2FA tokens



Fake

Real Or Fake?



Real

Even Better Alternative: U2F Tokens

Universal Second Factor (U2F)

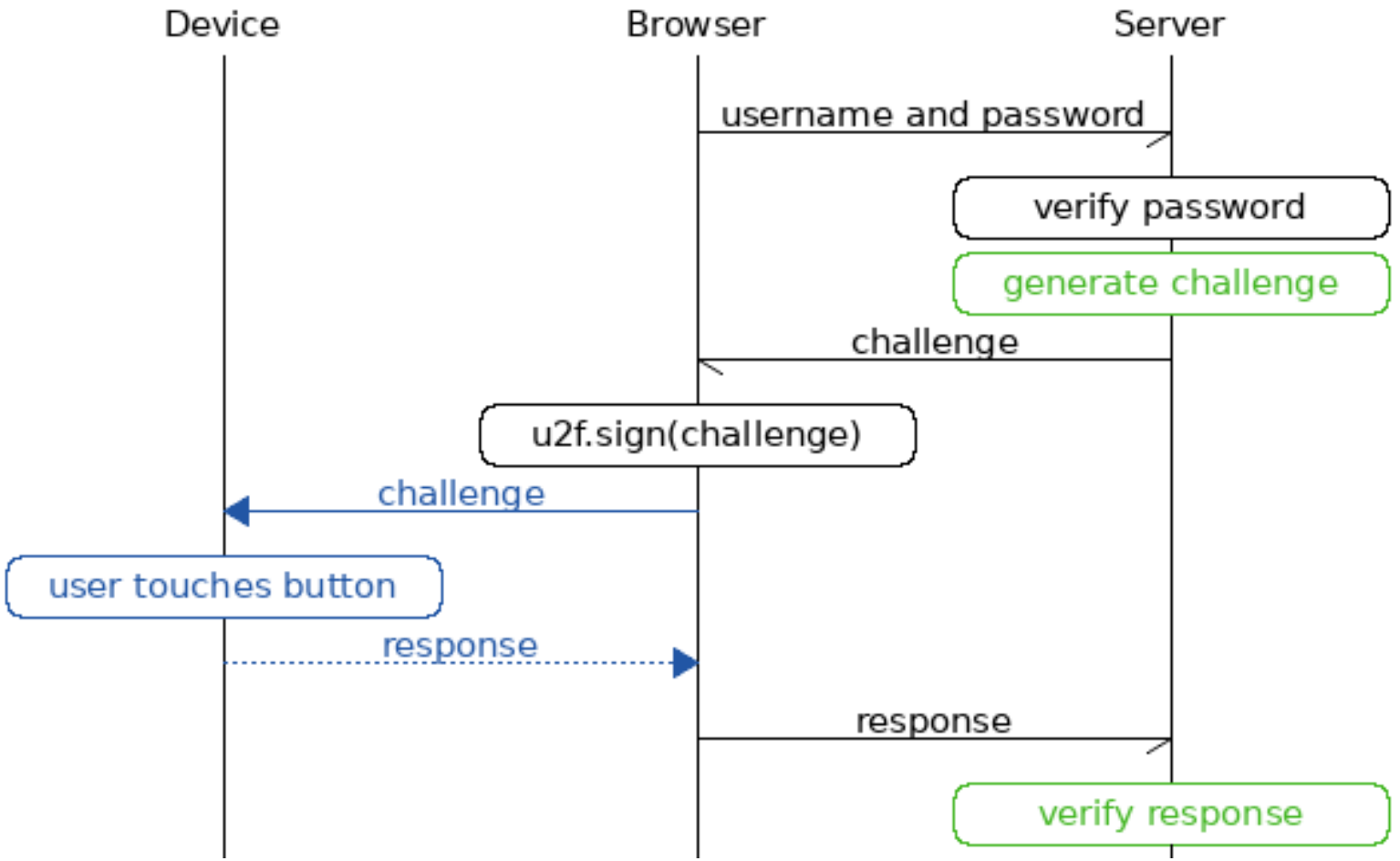
- FIDO (Fast IDentity Online) alliance: Google, Yubico, …
- Supported by all popular browsers and many online services

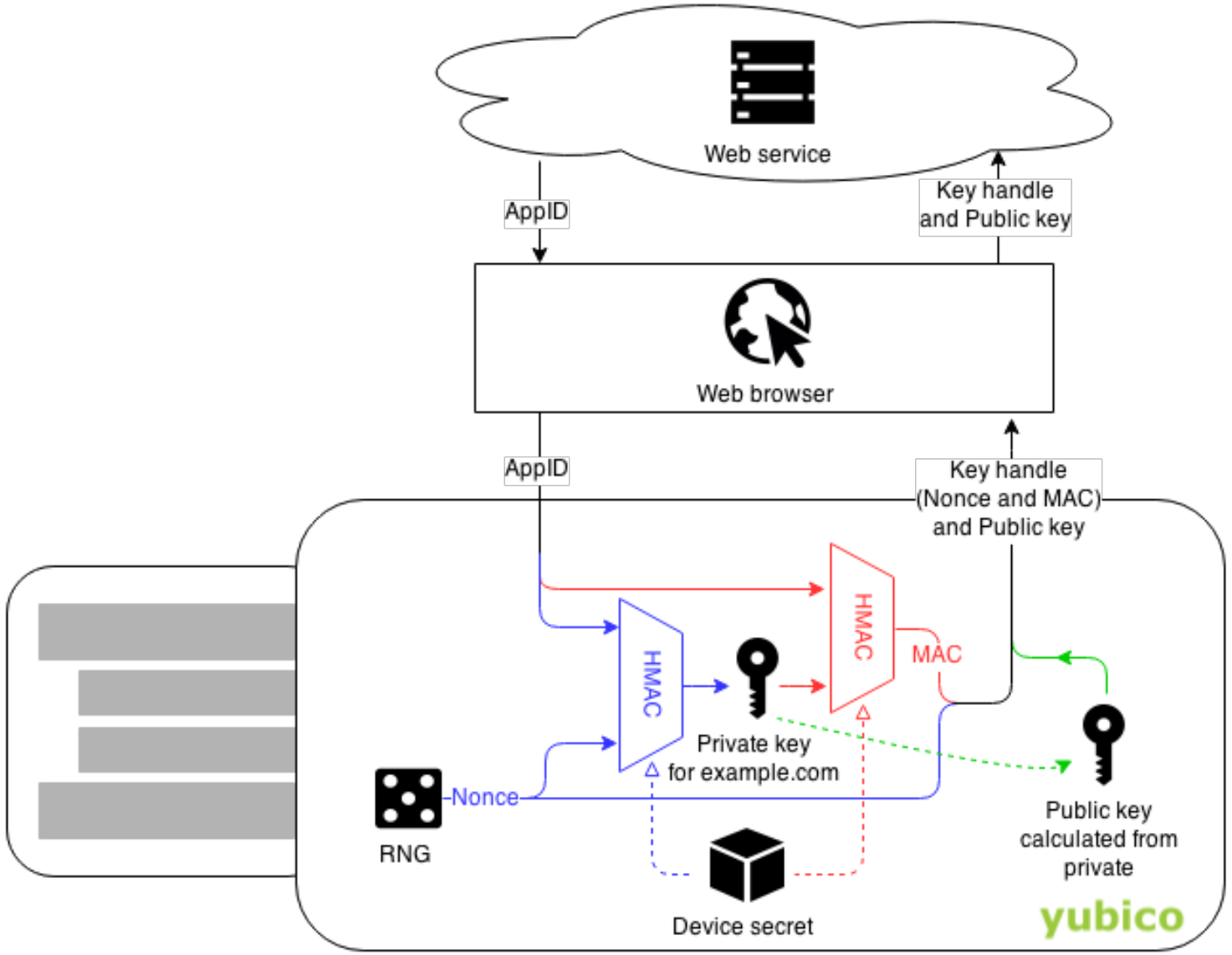
A different key pair is generated for each origin during registration

- Origin = <protocol, hostname, port>
- Private key ~~stored~~ re-generated on device
- Public key sent to server

Additions to the authentication flow:

- Origin (URI): ***prevents phishing***
- TLS Channel ID (optional): ***prevents MitM***





U2F tokens

Benefits

- Easy: just tap the button (no typing)
- Works out of the box (no drivers to install)
- USB, NFC, Bluetooth communication
- No shared secret between client and server
- **Origin checking → effective against phishing!**

Drawbacks

- Can be lost → a fallback is needed (second U2F token, Authenticator App, …)
- Cumbersome: have to pull keychain out and plug token in (or have an always pugged-in token, in which case though it can be stolen along with the device)
- Cost ($10–$70)

2FA Recap – *What threats does it prevent?*

SMS: useful against two main threats

- Credential stuffing (people tend to reuse passwords across different services)
- Leaked passwords (post-it, hardware keyloggers, cameras, shoulder surfing, …)
- Introduces new security/privacy issues: SIM swapping, SMS forwarding, SMS spam…

Authenticator Apps/Push Auth: much better alternative than SMS

- Protects against the same threats without relying on phone numbers

U2F: **additional protection against phishing**

- Modern phishing toolkits bypass SMS/Authenticator/Push 2FA through MitM
- Humans fall for typosquatting, but U2F's origin check doesn't

None of the above protect against session hijacking and Man-in-the-Browser

- Game over anyway if the host is compromised after the user has successfully logged in

Password Managers

Have become indispensable

- Encourage the use of complex/non-memorable passwords
- Obviate the need for password reuse: unique passwords per site/service

Protection against phishing: auto-fill won't work for incorrect domains

- As long as users don't copy/paste passwords out of the password manager (!)

Various options: third-party applications, OS-level, in-Browser

Password synchronization across devices

- Can the service provider access all my passwords or not?
- Preferable option: passwords should be encrypted with master password never visible to the cloud service

WebAuthn

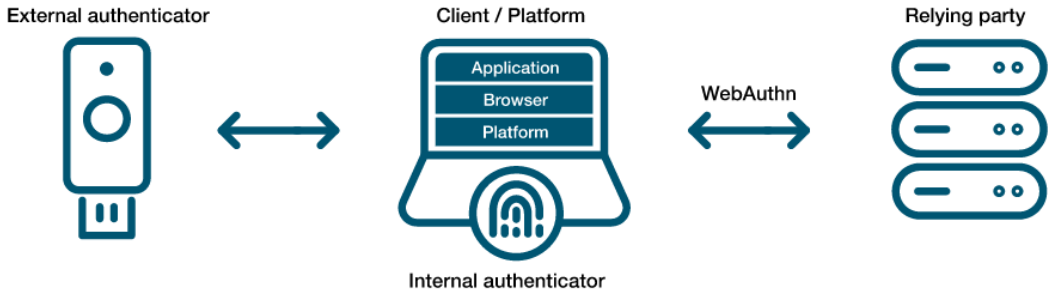
W3C Web Authentication standard (FIDO2): Successor of FIDO U2F

Use cases

- Low friction and phishing-resistant 2FA (in conjunction with a password)
- Passwordless, biometrics-based re-authorization
- 2FA *without* a password (passwordless logins)

Authenticators: devices that can generate private/public key pairs and gather consent (simple tap, fingerprint read, …)

- Built-in: fingerprint readers, cameras, …
- External: USB, BLE, NFC, …



https://www.yubico.com/authentication-standards/webauthn/

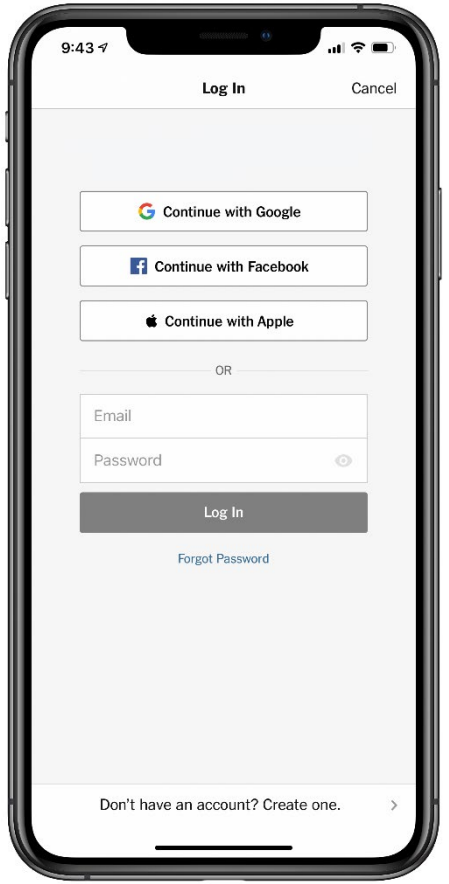
Single Sign-on/Social Login

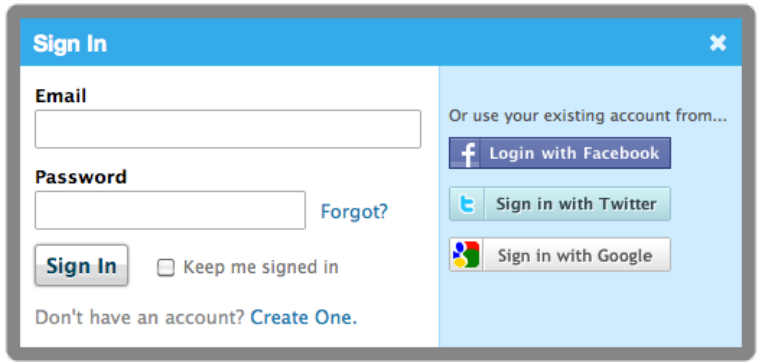
Pros

- Convenience: fewer passwords to remember
- Easier development: outsource user registration/management
- Rich experience through social features

Cons

- Same credentials for multiple sites: single point of failure
- Access to user's profile
- User tracking





Biometrics

Fingerprint reader

Face recognition

- Depth sensing, infrared cameras, …
- “liveness” detection (pulse, thermal) to foil simple picture attack

Retina/iris scanner

Voice recognition

…

Related concept: continuous authentication

- Keystroke timing, usage patterns, …

Crypto-based Authentication

Rely on a cryptographic key to prove a user's identity

User performs a requested cryptographic operation on a value (challenge) that the verifier supplies

- Usually based on knowledge of a key (secret key or private key)
- Can use symmetric (e.g., Kerberos) or public key (e.g., U2F) schemes

How can we trust a key? Why is it authentic?

- Need to establish a level of trust

Different approaches: **TOFU**, **PKI**, **Web of Trust**

- Emerging approach: PKI based on ~~blockchain~~ distributed ledger

Trust on First Use (aka Key Continuity)

Use case: SSH

- Performs *mutual authentication*

Server *always* authenticates the client

- password, key pair, …

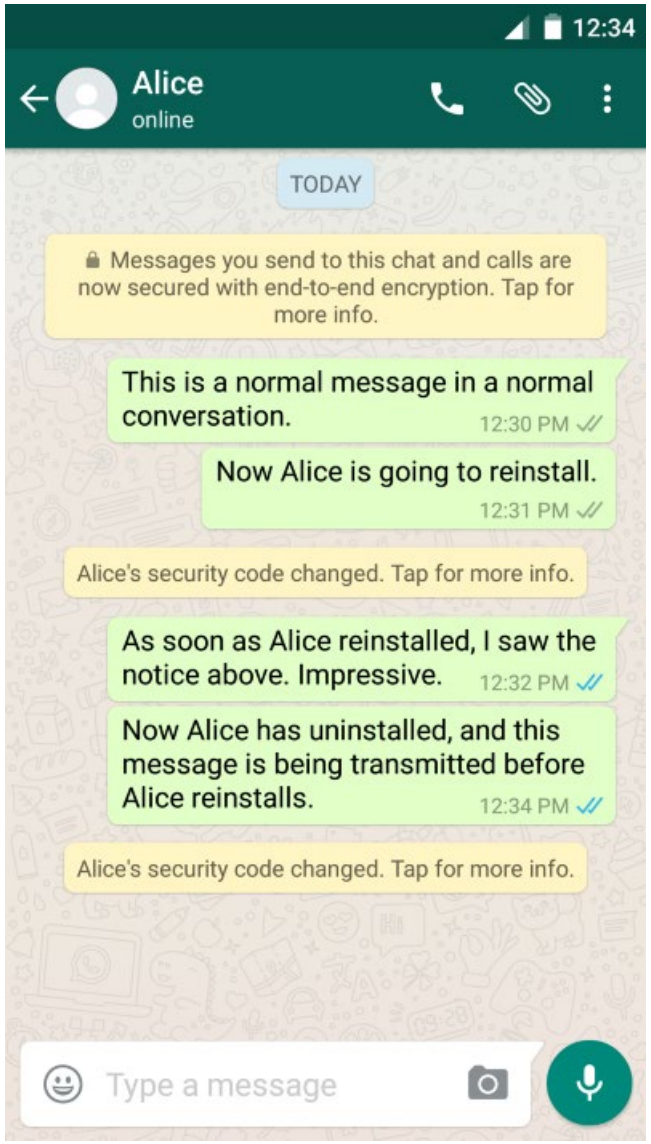
Client almost always authenticates the server – *except the first time!*

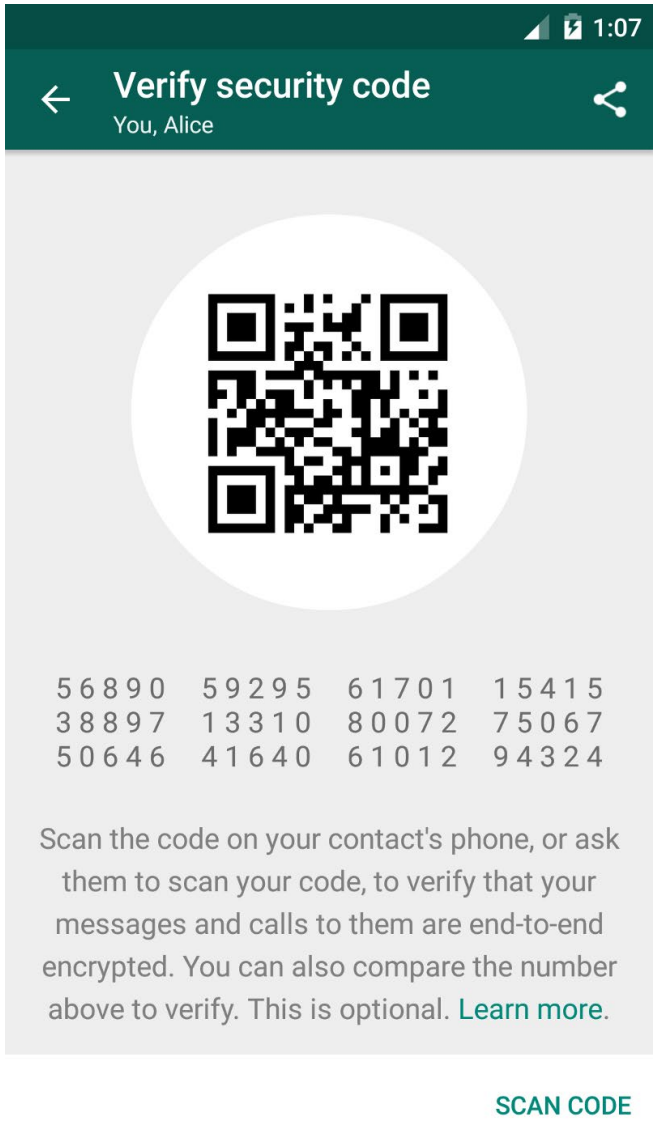
- First connection: server presents its public key
- No other option for the user but to accept it: MitM opportunity
- Subsequent connections: client remembers server's key, and triggers an alert on key mismatch

Pragmatic solution, but shifts the burden to users

- Users must determine the validity of the presented key
- Accepting a key change without verifying the new key offers no protection against MitM (unfortunately, that's what most users do)

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that the RSA host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
df:c8:52:aa:cd:e3:da:8c:ec:50:46:db:4d:21:d9:c7.
Please contact your system administrator.
Add correct host key in /root/.ssh/known_hosts to get rid of this message.
Offending key in /root/.ssh/known_hosts:1
RSA host key for 192.168.2.5 has changed and you have requested strict checking.
Host key verification failed.
```





Certificates

How can we distribute "trusted" public keys?

- Public directory ➔ risk of forgery and tampering
- More practical solution: "certified" public keys

A certificate is a digitally signed message that contains an identity and a public key

- Makes an association between a user/entity and a private key
- Valid until a certain period

Why trust a certificate?

- Because it is signed by an "authority"
- Requiring a signature by a third party prevents straightforward tampering

Public Key Infrastructures (PKI)

Facilitate the authentication and distribution of public keys with the respective identities of entities

- People, organizations, devices, applications, …
- Set of roles, policies, hardware, software, and procedures to create, mange, distribute, use, store, and revoke digital certificates and manage public key encryption

An issuer signs certificates for subjects

- Trust anchor

Methods of certification

- **Certificate authorities** (hierarchical structure – root of trust)
- **Web of trust** (decentralized, peer-to-peer structure)

Certificate Authorities

Trusted third-parties responsible for certifying public keys

- Most CAs are tree-structured
- Single point of failure: CAs can be compromised!

Why should we trust an authority?

- How do we know the public key of the Certificate Authority (CA)?

CA's public key (trust anchor) must somehow be provided out of band

- Trust has to start somewhere

Operating systems and browsers are pre-configured with tens/hundreds of trusted root certificates

- A public key for any website in the world will be accepted without warning if it has been certified by any of these CAs (more on that in the TLS lecture)

DigiNotar Files for Bankr

Secure | https://www.wired.com/2011/09/diginotar-bankruptcy/

WIRED

DigiNotar Files for Bankruptcy in Wake of Devastating Hack

SUBSCRIBE

SHARE

1

KIM ZETTER SECURITY 09.20.11 03:05 PM

DIGINOTAR FILES FOR BANKRUPTCY IN WAKE OF DEVASTATING HACK

A Dutch certificate authority that suffered a major hack attack this summer has been unable to recover from the blow and filed for bankruptcy this week.

MOST POPULAR

TRANSPORTATION
General Motors Announces an All-Electric Future
ALEX DAVIES

SECURITY
This "Ghost Gun" Machine Now Makes Untraceable Metal Handguns
ANDY GREENBERG

SECURITY
How the Las Vegas Shooter Could Have Gotten an Automatic Rifle
ANDY GREENBERG

MORE STORIES

Web of Trust (mainly used in PGP – more in the email lecture)

Entirely decentralized authentication

- No single point of failure
- No need to buy certs from CAs

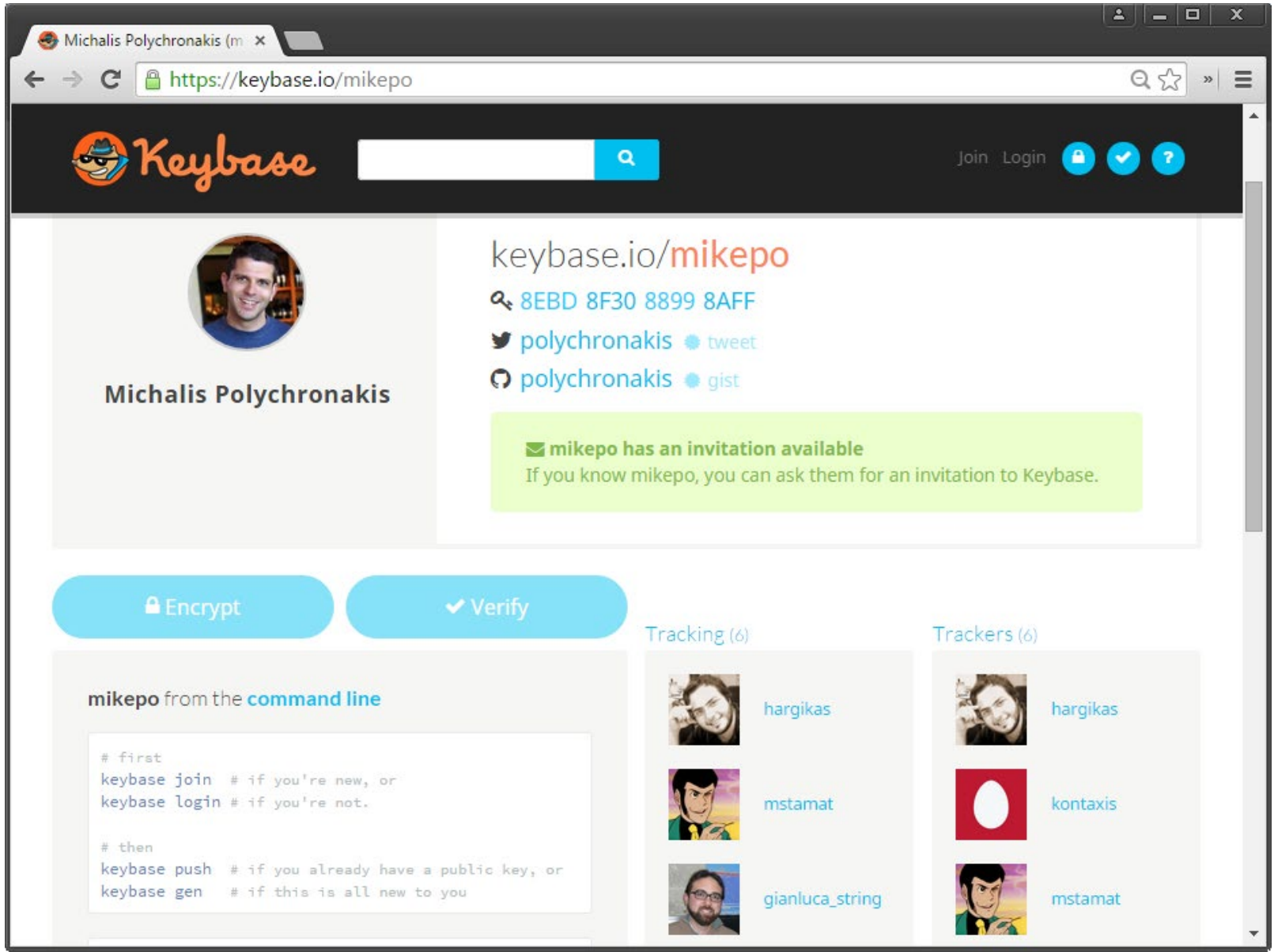
Users sign other users' keys

- Only if they deem them trustworthy
- Certificate signings can form an arbitrarily complex graph
- Users can verify path to as many trust anchors as they wish

Drawbacks

- Hard to use, requires in-person verification: key signing parties!
- Hard to know what trust level to assign transitively

WoT Alternative: Online Social "Tracking"



Keybase.io

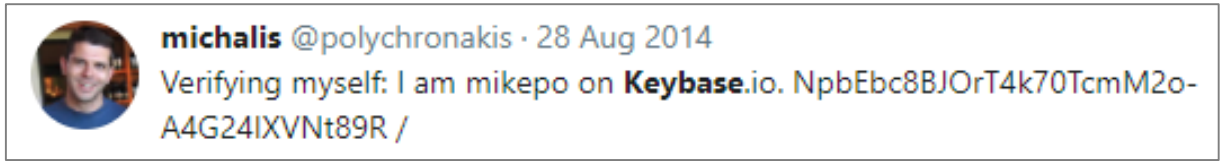
In essence, a directory associating public keys with names

Identity established through *public signatures*

- **Identity proofs**: *"I am Joe on Keybase and MrJoe on Twitter"*
- **Follower statements**: *"I am Joe on Keybase and I just looked at Chris's identity"*
- **Key ownership**: *"I am Joe on Keybase and here's my public key"*
- **Revocations**: *"I take back what I said earlier"*

Keybase identity = sum of public identities

Twitter, Facebook, Github, Reddit,
domain ownership, …



An attacker has to compromise all connected identities

The more connected identities, the harder to impersonate a user

Best Practices

Use long passphrases instead of passwords

- Never reuse the same password on different services

Use two-factor authentication when available

- Avoid SMS if possible! Use an authenticator app or U2F instead
- Remove phone number from account after authenticator/U2F setup
- Store your backup codes in a safe location

Use a password manager

- Pick non-memorable passwords and avoid copy/pasting them
- Password auto-fill helps against phishing (auto-fill won't work if the domain is wrong)
- Not only for passwords! Also for "security" questions

Use SSH keys instead of passwords