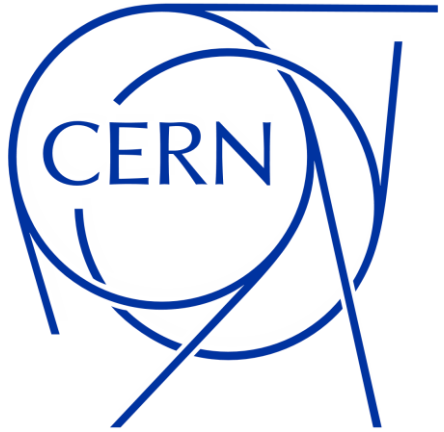


Implementing NIST password guidelines on PostgreSQL and Oracle at CERN
How-To, Challenges and Opportunities

Maurizio De Giorgi – Miroslav Potocky

26.03.2025

Maurizio De Giorgi

- Senior Database Engineer at CERN since Sep 2020
- DB on Demand: Service Manager and DevOps
- Long career in many different roles, industry, markets with a strong focus on databases and data stores
- Always looking at new technology, paradigms and trends

Maurizio De Giorgi

maurizio.degiorgi@cern.ch

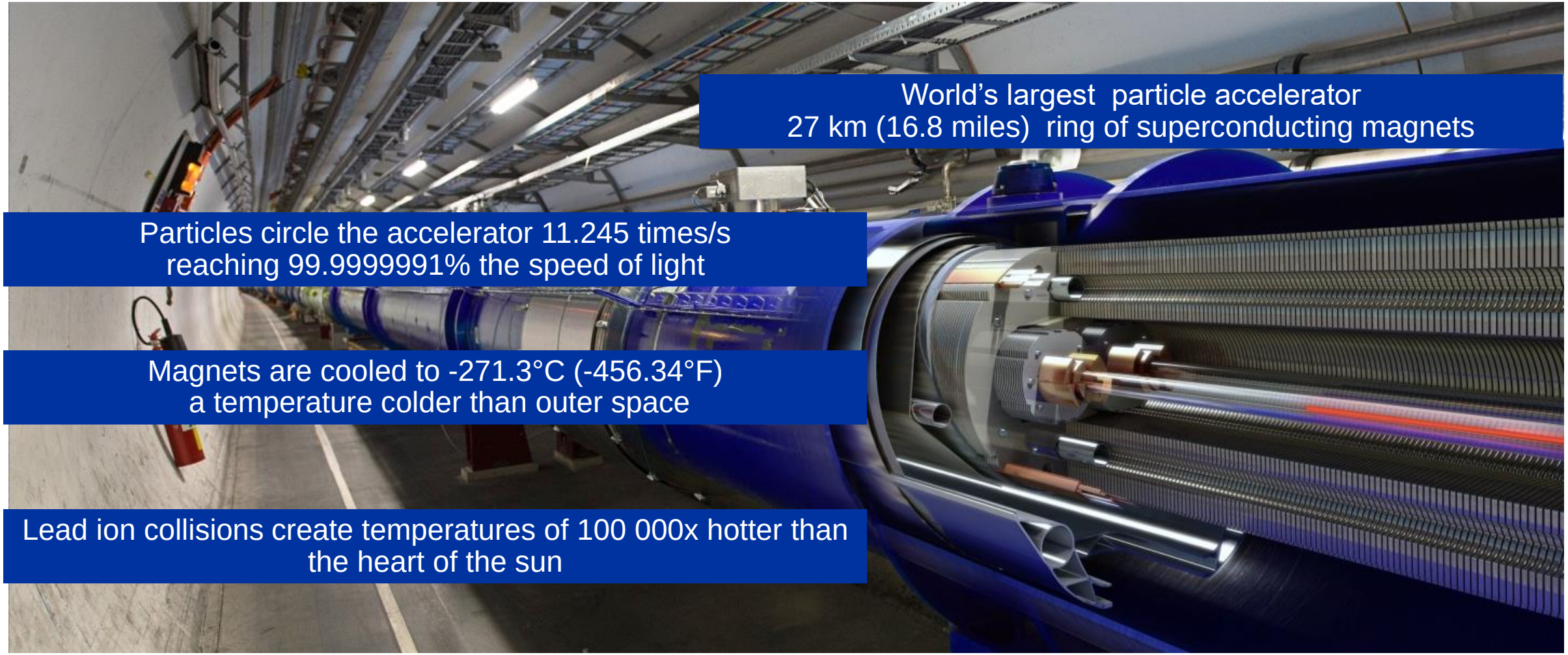
Established in 1954

23 Member states

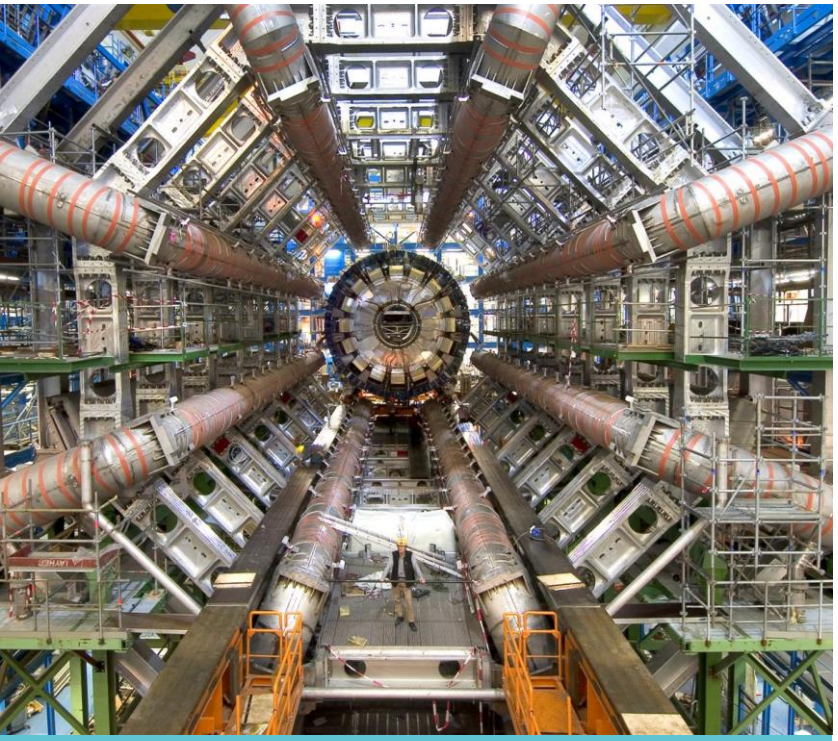
Our mission:

- Unveil how the universe works and what it is made of
- Provide a unique range of particle accelerator facilities to enable research at the forefront of the human knowledge
- Unite people from all over the world to push the frontiers of science and technology

The Large Hadron Collider

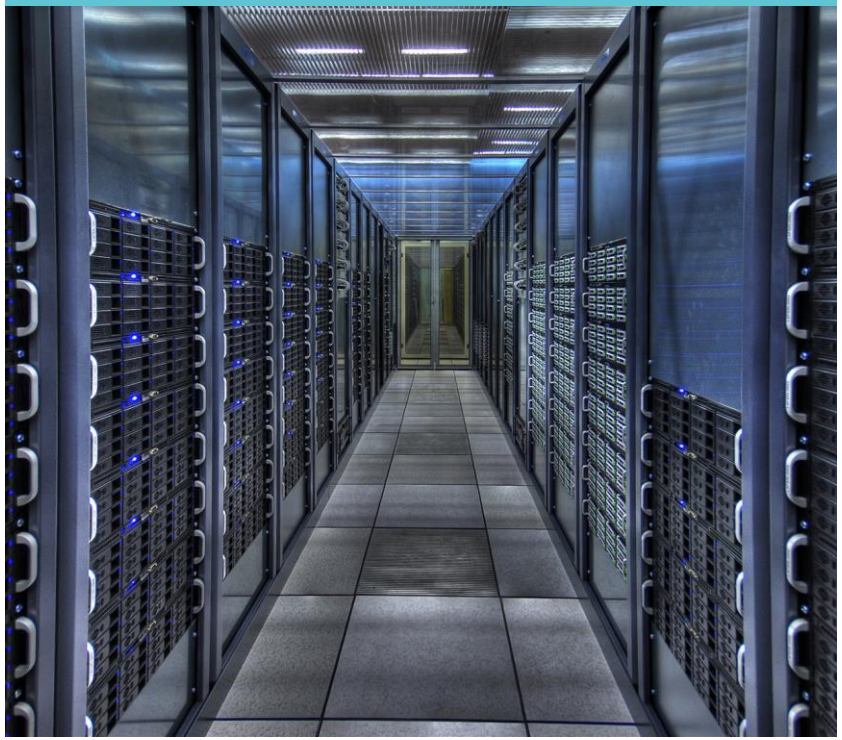


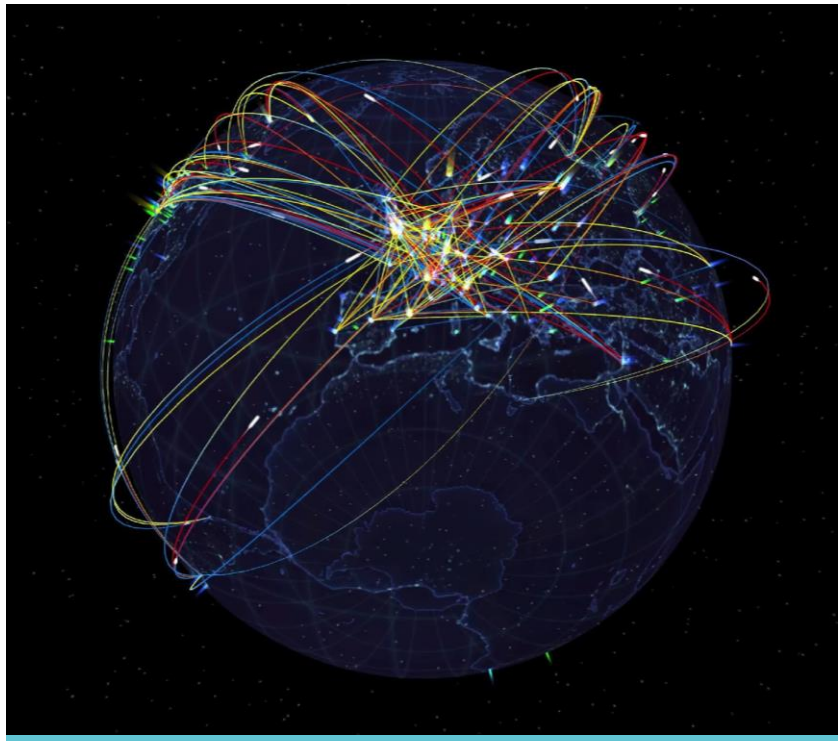
The Worldwide LHC Computing Grid (WLCG)



1 PB of data per second
Only 1% is kept (events with specific characteristics)

Tier0:
Data reconstruction + Tape archival
+ data distribution to other tiers
~ 200 PB of data per year

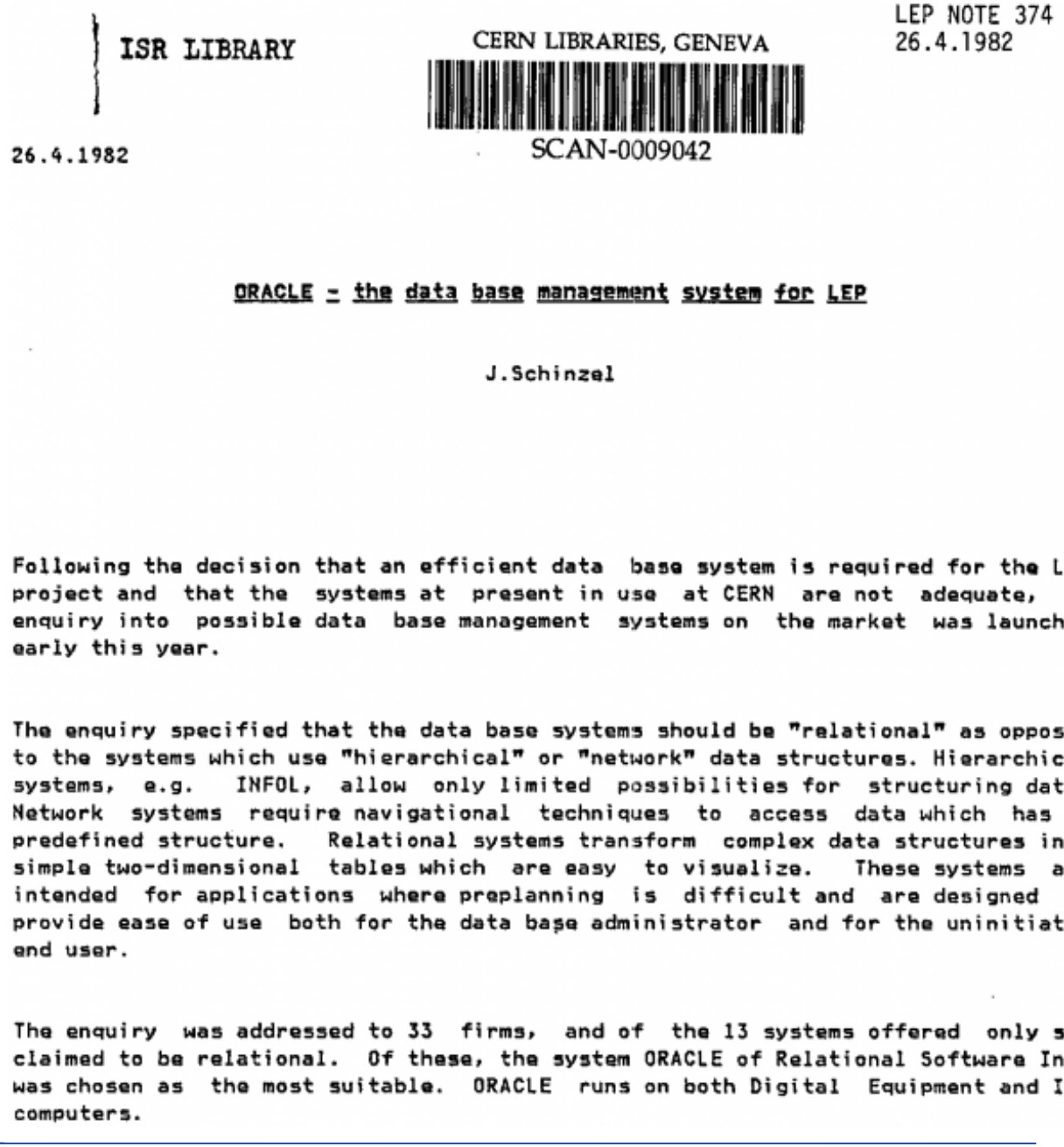




WLCG:
- 170 collaborating centers
- 36 countries
- Data analysis

Oracle at CERN

- Since 1982
- 105 Oracle databases
- More than 11.800 Oracle accounts
- RAC, Active Data Guard, OEM, RMAN…
- Complex environment
- Used by
  - Administrative Information Services
  - Engineering teams
  - Accelerator and experiments
- Full DBA support
- ≈ 5PB of data



ISR LIBRARY

CERN LIBRARIES, GENEVA

SCAN-0009042

LEP NOTE 374
26.4.1982

26.4.1982

ORACLE - the data base management system for LEP

J.Schinzel

Following the decision that an efficient data base system is required for the LEP project and that the systems at present in use at CERN are not adequate, an enquiry into possible data base management systems on the market was launched early this year.

The enquiry specified that the data base systems should be "relational" as opposed to the systems which use "hierarchical" or "network" data structures. Hierarchical systems, e.g. INFOL, allow only limited possibilities for structuring data. Network systems require navigational techniques to access data which has a predefined structure. Relational systems transform complex data structures into simple two-dimensional tables which are easy to visualize. These systems are intended for applications where preplanning is difficult and are designed to provide ease of use both for the data base administrator and for the uninitiated end user.

The enquiry was addressed to 33 firms, and of the 13 systems offered only six claimed to be relational. Of these, the system ORACLE of Relational Software Inc. was chosen as the most suitable. ORACLE runs on both Digital Equipment and IBM computers.

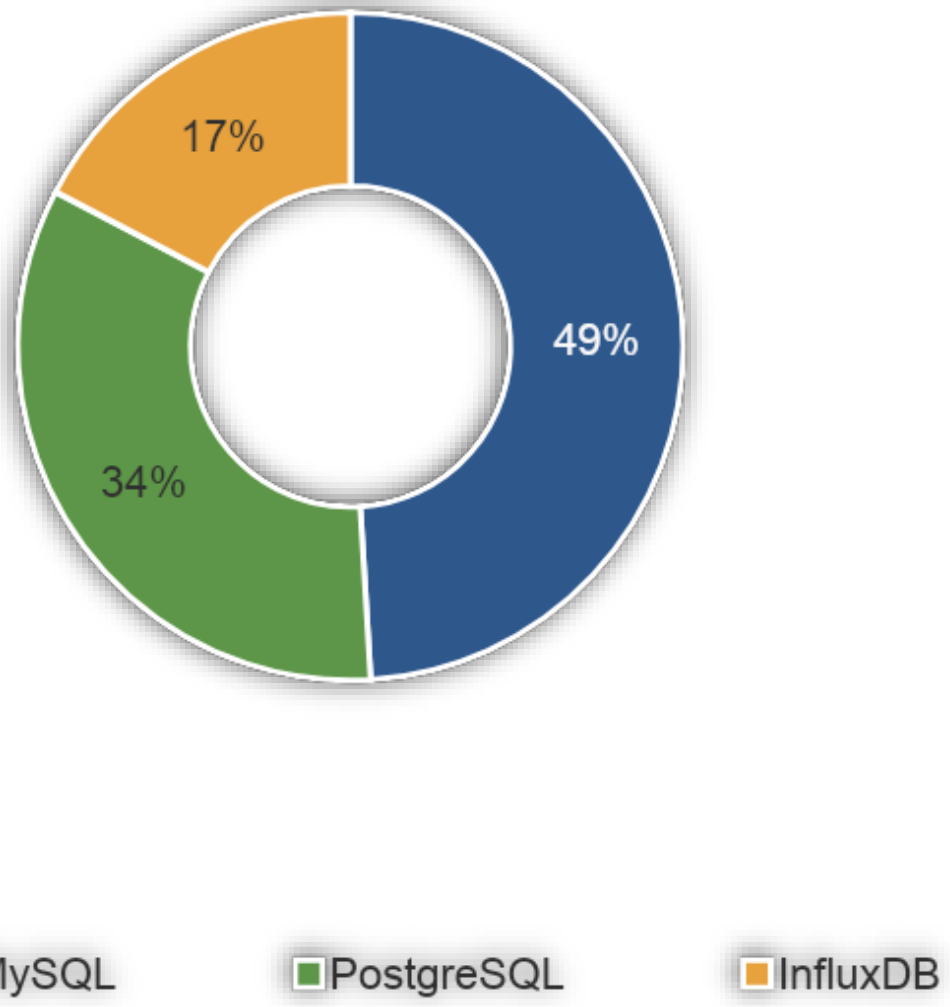
Oracle versions

- **Oracle V2 (1979):** First SQL-based RDBMS.
- **Oracle V3 (1983):** Client-server architecture, PL/SQL introduced.
- **Oracle V4 (1984):** Read consistency.
- **Oracle V5 (1985):** Distributed queries.
- **Oracle V6 (1988):** Row-level locking, PL/SQL stored procedures.
- **Oracle7 (1992):** Declarative referential integrity.
- **Oracle8 (1997):** Object-relational database.
- **Oracle8i (1999):** Native internet protocols, Java support.
- **Oracle9i (2001):** Real Application Clusters (RAC).
- **Oracle10g (2003):** Grid computing.
- **Oracle11g (2007):** Advanced compression, Automated SQL tuning.
- **Oracle12c (2013):** Multitenant architecture, in-memory database.
- **Oracle18c (2018):** Autonomous database features.
- **Oracle19c (2019):** Automatic indexing, Active Data Guard DML redirection.
- **Oracle21c (2020):** Blockchain tables, JSON enhancements.

From https://redresscompliance.com/evolution-of-oracle-database-versions-all-major-releases/

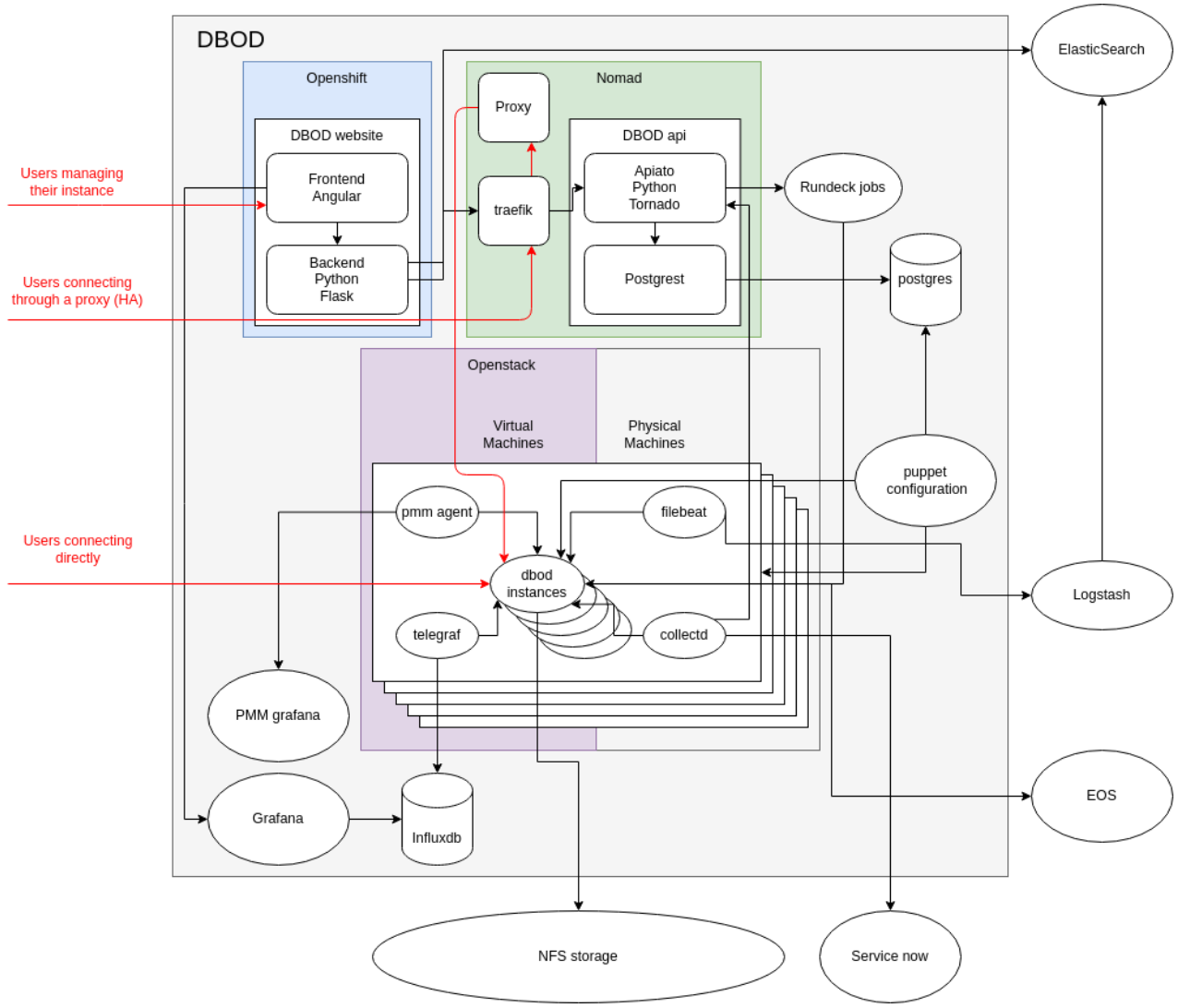
DB on Demand at CERN

- DBaaS conceived in 2011
- MySQL, PostgreSQL (Timescale), InfluxDB
- Empowers users to be their own DBA
- Flexible architecture (integration of new technology)
- More than 1200 instances
  - ≈600 MySQL, ≈400 PostgreSQL, ≈200 InfluxDB
  - ≈150 TB of data
- A number of key database applications:
  - DBOD own databases
  - Authorization and authentication (SSO)
  - Experiments (ATLAS, LHCb, etc.)
  - WLCG File Transfer Service
  - GitLab, Puppet, Foreman, Teigi (secrets)
  - Openstack (nova, ironic)
  - Security (some SOC apps)
  - Indico, Zenodo, Jira, ServiceNow



DB on Demand Architecture

- Complex DBaaS environment
- Integrated with CERN infrastructure
- Mostly open source
- Infrastructure as Code
- Deploy on VM/Bare Metal
- Systemd managed services
- NetApp Storage
  - data/wals NFS volumes
  - snapshot based backups
- EOS (EOS Open Storage)
  - snapshots copy archive
  - wals archive



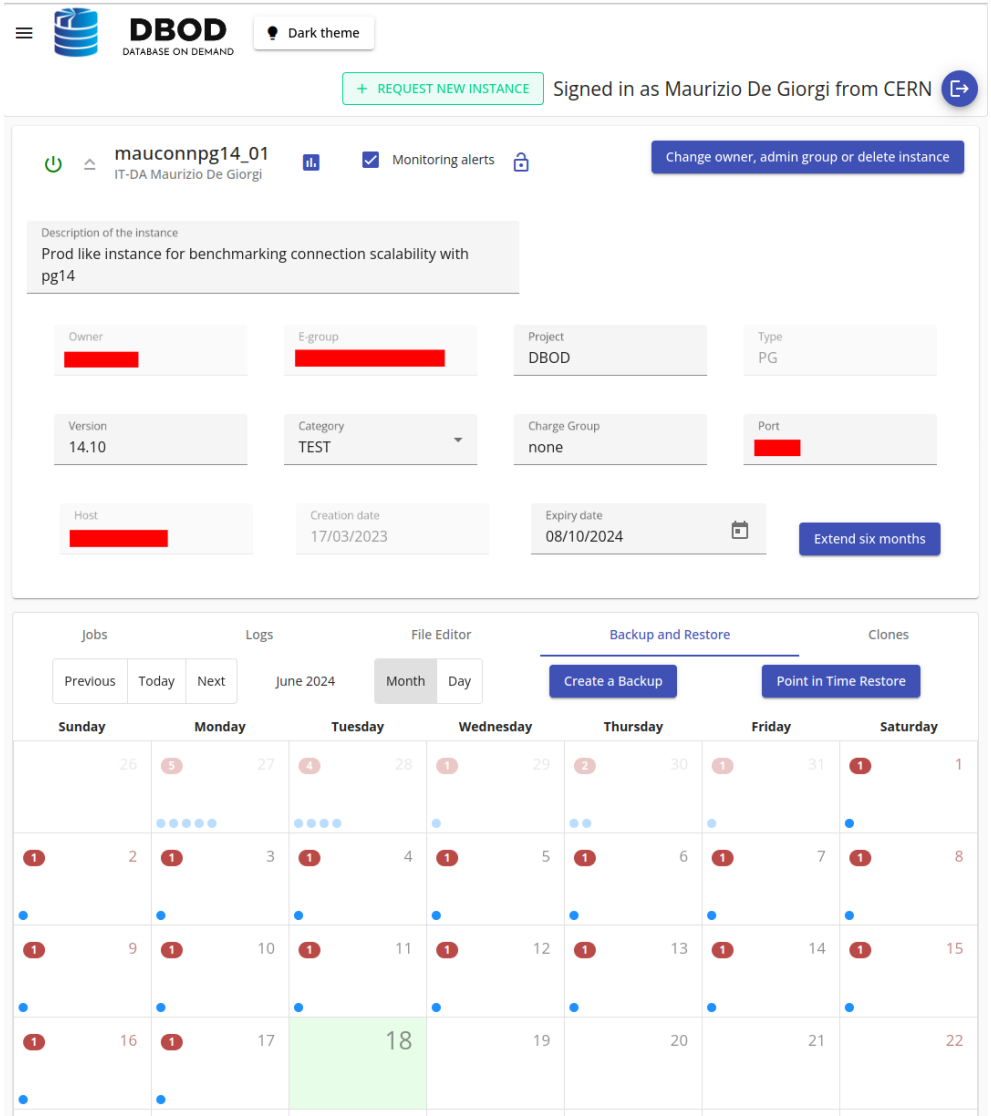
DB on Demand Automation

Web automation

- Automated backup and recovery services
- Upgrade checker to enable self-service upgrades
  - once errors and warnings in the report are fixed
- Management of configuration files
- Cloning
- Integrated monitoring
- Integrated upgrades
  - Primary-replica upgrade logic

Ops automation

- Continuous validation of backups
- Instance and storage migration
- Automated replica provisioning
- Automated replication switchover
- Detection of idle instances
- Integrated password hash cracker



Letting the people choosing their own paths

NewScientist

Sign in | Enter search keywords | Events Tours Shop Courses Jobs

News Features Newsletters Podcasts Video Comment Culture Crosswords | This week's magazine

Health Space Physics Technology Environment Mind Humans Life Mathematics Chemistry Earth Society

Subscribe now

Letter: Letting the people choose to walk their own path (1)

Published 1 April 2020

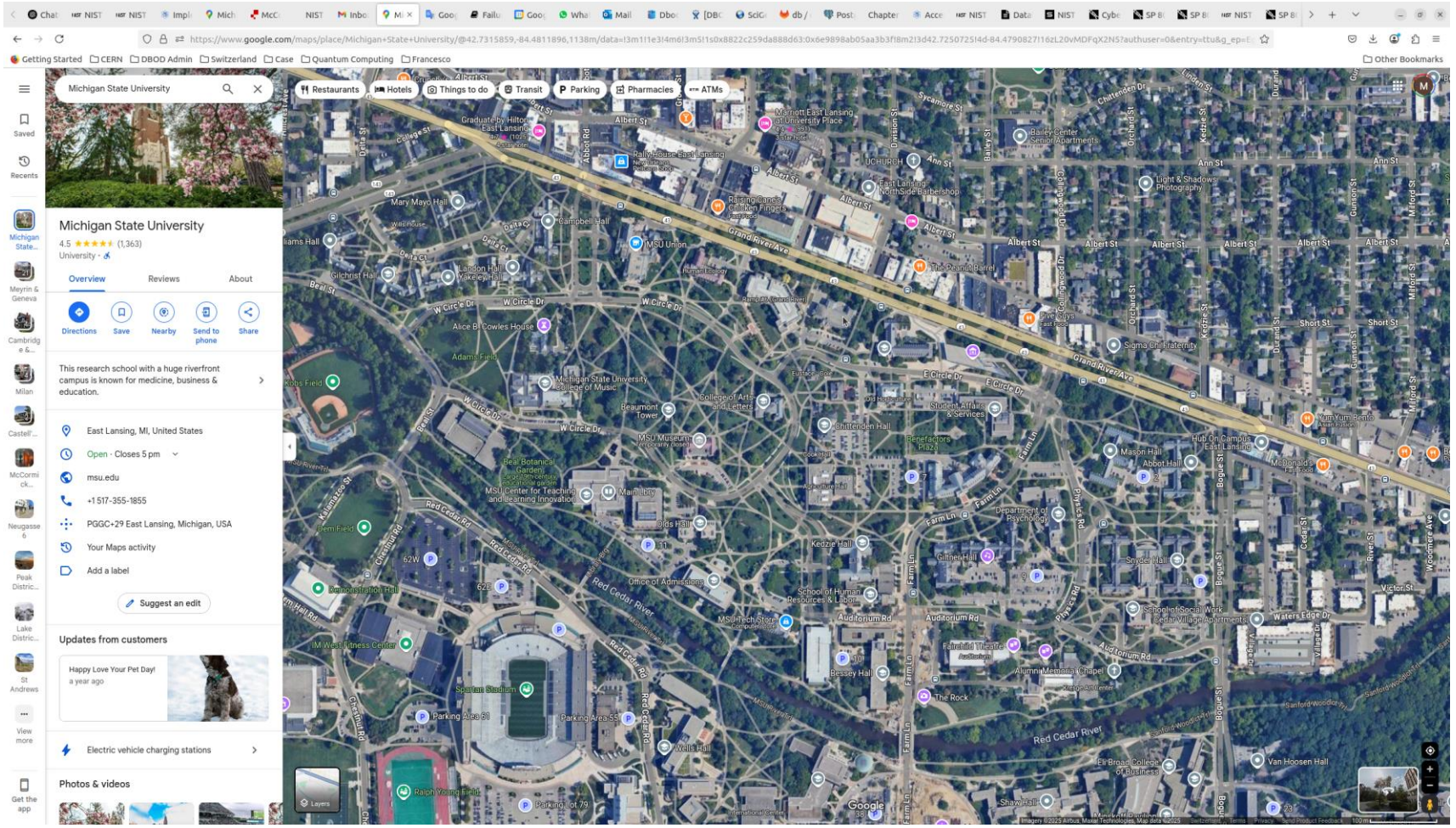
From Brian Horton, West Launceston, Tasmania, Australia

Footpaths should be added after people have used an area for a while so users decide where the paths should be, suggests Frank Bover (**Letters,** 21 March). This method has been used for decades by planners of new universities and college campuses. Michigan State University has paths designed this way: an aerial view shows paths at unusual angles that take efficient routes.

Failure to do this invariably results in "desire paths", where people take shortcuts and ignore the fixed paths, a clear indication of poor planning. Going one step further, the corridors of the McCormick Tribune Campus Center at the Illinois Institute of Technology in Chicago were designed to follow routes taken by students across the open field it was built on.

Advertisement

Letting the people choosing their own paths...



Michigan State University

People take shortcuts and ignore the fixed paths

Password reuse:

- across multiple accounts to avoid having to remember multiple complex passwords

Weak passwords:

- to meet complex password requirements

Password rotations:

- Writing down or storing passwords in an insecure location to keep track

Post-it notes:

- near computers or in plain sight to avoid forgetting complex passwords

Shared passwords:

- to simplify access to shared resources

Password managers with weak master passwords:

- defeating the purpose of password management

People take shortcuts and ignore the fixed paths

Easily guessable security questions:

- "What is your mother's maiden name?"

Same password with slight variations:

- to meet complex password requirements

Publicly available information as passwords or password hints:

- such as birthdays or pet names

Avoiding password-protected systems or applications:

- opting for less secure alternatives instead

Default or vendor-supplied passwords:

- leaving systems vulnerable to attacks

Plain text passwords:

- either digitally or physically, to avoid having to remember complex passwords

NIST Digital Identity Guidelines

NIST SP 800-63-3

Jun 2017 (incl. updates Dec 2017)

Mar 2020 (incl. updates Mar 2020)

NIST SP 800-63-4

- Jun 2020 Initial Preliminary Draft
- Dec 2022 Initial Public Draft
- Aug 2024 2nd Public Draft

1. NIST SP 800-63A-4 - Identity Proofing and Enrollment
2. NIST SP 800-63B-4 - Authentication and Authenticator Management
3. NIST SP 800-63C-4 - Federation and Assertions

NIST Digital Identity Guidelines

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
U.S. DEPARTMENT OF COMMERCE

No more periodic password changes (no password expiration)

"Verifiers SHOULD NOT require memorized secrets to be changed arbitrarily (e.g., periodically). However, verifiers SHALL require memorized secrets to be changed upon evidence of authenticity compromise."

(Section 5.1.1.2, "Memorized Secret Authenticators")

Length over complexity

"Verifiers SHOULD require subscriber-chosen memorized secrets to be at least 8 characters in length. Verifiers SHOULD permit subscriber-chosen memorized secrets at least 64 characters in length. Verifier systems SHOULD NOT impose other composition rules (e.g., requiring mix of uppercase and lowercase letters) on memorized secrets."

(Section 5.1.1.2, "Memorized Secret Authenticators")

NB: Excerpts from NIST SP 800-63-3 June 2017 version

NIST Digital Identity Guidelines

No password hints or questions

"Verifier systems SHOULD NOT allow users to provide password hints or other forms of knowledge-based authentication (e.g., security questions)."

(Section 5.1.1.2, "Memorized Secret Authenticators")

Blocklist of common passwords

"Verifier systems SHOULD maintain a list of compromised or commonly used memorized secrets (e.g., passwords) and SHOULD NOT allow these secrets to be stored or used."

(Section 5.1.1.2, "Memorized Secret Authenticators")

NB: Excerpts from NIST SP 800-63-3 June 2017 version

NIST Digital Identity Guidelines

Risk-based approach to identity proofing

"The level of assurance in an identity proofing process SHOULD be commensurate with the risk associated with the claimed identity."

(Section 4.2, "Identity Proofing")

Use multi-factor authentication (MFA)

"Multi-factor authentication (MFA) solutions SHOULD be used to provide a higher level of assurance."

(Section 5.2, "Multi-Factor Authentication")

NB: Excerpts from NIST SP 800-63-3 June 2017 version

NIST Digital Identity Guidelines

Use authenticator apps

"Verifier systems SHOULD use authenticator apps as a more secure alternative to SMS-based two-factor authentication."

(Section 5.2.2, "Out-of-Band Verification Using Authenticator Apps")

Prioritize usability

"Verifier systems SHOULD prioritize usability when implementing digital identity management systems."

(Section 6.1, "Usability and Accessibility")

NB: Excerpts from NIST SP 800-63-3 June 2017 version

NIST Digital Identity Guidelines

Address evolving threats,
improve usability,
incorporate lessons learned,
from real-world implementations
since the 2017 version,
while maintaining strong security practices
in password and secret management

NB: What's new according to NIST Special Publication NIST SP 800-63B-4 2pd - August 2024 Version

NIST Digital Identity Guidelines

Password Length:

- minimum 8 characters
- recommended 15 characters
- maximum at least 64 characters

Password Complexity:

- Complexity rules (e.g., requiring mixtures of different character types) are no longer recommended
- All printing ASCII and Unicode characters should be allowed (increased entropy, allow national characters)

Further details Appendix A, *Strength of Passwords*.

NB: What's ~~new~~ according to NIST Special Publication NIST SP 800-63B-4 2pd - August 2024 Version

NIST Digital Identity Guidelines

Password Expiration:

- Periodic password changes are no longer required
- Passwords should only be changed if there's evidence of compromise

Password Screening:

- New and changed passwords must be checked against a blocklist of common or previously compromised passwords

Password Hints:

- Password hints and authentication questions are not permitted

NB: What's new according to NIST Special Publication NIST SP 800-63B-4 2pd - August 2024 Version

NIST Digital Identity Guidelines

Rate limiting (Throttling):

- Implement throttling to limit failed authentication attempts

Password Managers:

- Allow and encourage the use of password managers
- Enable paste functionality to facilitate password manager use

Passwordless Authentication:

- The updated framework emphasizes passwordless authentication methods, particularly those offering phishing resistance

Secure Storage:

- Implement salting and hashing using memory-hard functions for secure password storage

NB: What's new according to NIST Special Publication NIST SP 800-63B-4 2pd - August 2024 Version

Passwordcheck (with cracklib enabled)

```
postgres=# create user admin with password 'changeme';
ERROR:  password must contain both letters and nonletters

  postgres=# create user admin with password 'Changem3';
ERROR:  password is easily cracked
[in the log DETAIL:  cracklib diagnostic: it is based on a dictionary word]

postgres=# create user admin with password 'changem3more!';
CREATE ROLE

postgres=# create user admin with password 'mammami4';
ERROR:  password is easily cracked
[in the log DETAIL:  cracklib diagnostic: it does not contain enough DIFFERENT characters]
```

passwordcheck

Enabling cracklib is not too difficult but it requires building postgres from source.

Beside what is possible with the simple default integration, there are other possibilities for customization, as explained at https://github.com/cracklib/cracklib/tree/main/src

Ex: https://github.com/michaelpq/pg_plugins/tree/main/passwordcheck_extra

It is also possible to integrate other word lists in addition to `/usr/share/dict/words` provided by the `words` package.

```
# contrib/passwordcheck/Makefile

MODULE_big = passwordcheck
OBJS = \
        $(WIN32RES) \
        passwordcheck.o
PGFILEDESC = "passwordcheck - strengthen user password checks"

# uncomment the following two lines to enable cracklib support
# PG_CPPFLAGS = -DUSE_CRACKLIB '-DCRACKLIB_DICTPATH="/usr/lib/cracklib_dict"'
# SHLIB_LINK = -lcrack

REGRESS = passwordcheck

ifdef USE_PGXS
PG_CONFIG = pg_config
PGXS := $(shell $(PG_CONFIG) --pgxs)
include $(PGXS)
else
subdir = contrib/passwordcheck
top_builddir = ../..
include $(top_builddir)/src/Makefile.global
include $(top_srcdir)/contrib/contrib-global.mk
endif
```

passwordcheck

postgres / contrib / passwordcheck / **passwordcheck.c** ↑ Top

Code | Blame | 164 lines (147 loc) · 4.4 KB | Raw

```
109
110         /* check if the password contains both letters and non-letters */
111         pwd_has_letter = false;
112         pwd_has_nonletter = false;
113         for (i = 0; i < pwdlen; i++)
114         {
115             /*
116              * isalpha() does not work f
117              * consider non-ASCII charac
118              */
119             if (isalpha((unsigned char)
120                 pwd_has_letter = true;
121             else
122                 pwd_has_nonletter = true;
123         }
124         if (!pwd_has_letter || !pwd_has_nonletter)
125             ereport(ERROR,
126                     (errcode(ERRCODE_INVALID_PARAMETER_VALUE),
127                      errmsg("password must contain both letters and nonletters")));
128
129 #ifdef USE_CRACKLIB
130         /* call cracklib to check password */
131         if ((reason = FascistCheck(password, CRACKLIB_DICTPATH)))
132             ereport(ERROR,
133                     (errcode(ERRCODE_INVALID_PARAMETER_VALUE),
134                      errmsg("password is easily cracked"),
135                      errdetail_log("cracklib diagnostic: %s", reason)));
136 #endif
137     }
138
139     /* all checks passed, password is ok */
140 }
```

```
STATEMENT: create user maurizio with password 'chivaconlozoppoimparaazoppicare';
ERROR: password must contain both letters and nonletters
STATEMENT: create user maurizio with password 'chivaconlozoppoimparaazoppicar3';
ERROR: password is easily cracked
DETAIL: cracklib diagnostic: error loading dictionary
/usr/lib/cracklib_dict.pwd.gz: No such file or directory
```

passwordcheck

postgres / contrib / passwordcheck / **passwordcheck.c** ↑ Top

Code | Blame | 164 lines (147 loc) · 4.4 KB | Raw

```
109
110         /* check if the password contains both letters and non-letters */
111         pwd_has_letter = false;
112         pwd_has_nonletter = false;
113         for (i = 0; i < pwdlen; i++)
114         {
115             /*
116              * isalpha() does not work f
117              * consider non-ASCII charac
118              */
119             if (isalpha((unsigned char)
120                 pwd_has_letter = true;
121             else
122                 pwd_has_nonletter = true;
123         }
124         if (!pwd_has_letter || !pwd_has_nonletter)
125             ereport(ERROR,
126                     (errcode(ERRCODE_INVALID_PARAMETER_VALUE),
127                      errmsg("password must contain both letters and nonletters")));
128
129 #ifdef USE_CRACKLIB
130         /* call cracklib to check password */
131         if ((reason = FascistCheck(password, CRACKLIB_DICTPATH)))
132             ereport(ERROR,
133                     (errcode(ERRCODE_INVALID_PARAMETER_VALUE),
134                      errmsg("password is easily cracked"),
135                      errdetail_log("cracklib diagnostic: %s", reas
136 #endif
137     }
138
139     /* all checks passed, password is ok */
140 }
```

```
STATEMENT: create user maurizio with password 'chivaconlozoppoimparaazoppicare';
ERROR: password must contain both letters and nonletters
STATEMENT: create user maurizio with password 'chivaconlozoppoimparaazoppicar3';
ERROR: password is easily cracked
DETAIL: cracklib diagnostic: error loading dictionary
/usr/lib/cracklib_dict.pwd.gz: No such file or directory
```

```
# install cracklib
yum -y install cracklib cracklib-devel cracklib-dicts words
# create dictionary
mkdict /usr/share/dict/* | packer /usr/lib/cracklib_dict
[gzip /usr/lib/cracklib_dict.pw]
```

passwordcheck

Building cracklib support into passwordcheck (excluded by default for license reasons) is not too difficult:

```
# enable CrackLib support by uncommenting the following lines in the Makefile
# PG_CPPFLAGS = -DUSE_CRACKLIB '-DCRACKLIB_DICTPATH="/usr/lib/cracklib_dict"'
# SHLIB_LINK = -lcrack
sed -i '/PG_CPPFLAGS/s/^#//g' contrib/passwordcheck/Makefile
sed -i '/SHLIB_LINK/s/^#//g' contrib/passwordcheck/Makefile
# force password to be at least 15 char
sed -i '/^(\s+)?static(\s+)?int(\s+)?min_password_length(\s+)?=(\s+)?8;(\s+)?$/s/\s+8(\s+)?$/ 15/g'
contrib/passwordcheck/passwordcheck.c
```

Install dependencies:

```
dnf -y install cracklib cracklib-devel cracklib-dicts words
```

Build the default dictionary with:

```
[root@xxx ~]# ls -al /usr/share/dict/*
-rw-r--r--. 1 root root 4953680 Aug 12  2018 /usr/share/dict/linux.words
lrwxrwxrwx. 1 root root      11 Aug 12  2018 /usr/share/dict/words -> linux.words
[root@xxx ~]# mkdict /usr/share/dict/* | packer /usr/lib/cracklib_dict
462982 46298
[root@xxx ~]# ls -al /usr/lib/cracklib_dict.*
-rw-r--r--. 1 root root    1024 Mar  5 17:18 /usr/lib/cracklib_dict.hwm
-rw-r--r--. 1 root root 2435284 Mar  5 17:18 /usr/lib/cracklib_dict.pwd
-rw-r--r--. 1 root root  115760 Mar  5 17:18 /usr/lib/cracklib_dict.pwi
```

passwordpolicy (another customization)

passwordpolicy is like the regular PostgreSQL **passwordcheck** extension, except it is **built with cracklib and has some configurations options**. Unlike the original module, this one has more strict password checks. The passwordpolicy module checks users' passwords whenever they are set with CREATE ROLE or ALTER ROLE. If a password is considered too weak, it will be rejected and the command will terminate with an error.

Website: https://access.crunchydata.com/documentation/passwordpolicy/latest/
Repo: https://github.com/eendroroy/passwordpolicy

```
  p_policy.min_password_len = 8        # Set minimum Password length
p_policy.min_special_chars = 2       # Set minimum number of special chracters
p_policy.min_numbers = 2             # Set minimum number of numeric characters
p_policy.min_uppercase_letter = 2    # Set minimum number of upper case letters
p_policy.min_lowercase_letter = 2    # Set minimum number of lower casae letters
```

passwordcheck

```
postgres=> alter user maurizio with password 'changeme';
ERROR:  password must contain both letters and nonletters
postgres=> \password maurizio
Enter new password for user "maurizio":
Enter it again:
postgres=>
```

psql — PostgreSQL interactive terminal

`\password [ username ]`

Changes the password of the specified user (by default, the current user). This command prompts for the new password, encrypts it, and sends it to the server as an `ALTER ROLE` command. This makes sure that the new password does not appear in cleartext in the command history[1], the server log, or elsewhere.

Caution

To prevent unencrypted passwords from being sent across the network, written to the server log or otherwise stolen by a database administrator, PostgreSQL allows the user to supply pre-encrypted passwords. Many client programs make use of this functionality and encrypt the password before sending it to the server.

This limits the usefulness of the `passwordcheck` module, because in that case it can only try to guess the password. For this reason, `passwordcheck` is not recommended if your security requirements are high. It is more secure to use an external authentication method such as GSSAPI (see Chapter 20) than to rely on passwords within the database.

Alternatively, you could modify `passwordcheck` to reject pre-encrypted passwords, but forcing users to set their passwords in clear text carries its own security risks.

[1] psql \s, default ~/.psql_history

passwordcheck

PostgreSQL has native support for using SSL connections to encrypt client/server communications for increased security. This requires that OpenSSL is installed on both client and server systems and that support in PostgreSQL is enabled at build time[1]

```
postgres=# select type, database, user_name, address, auth_method from pg_hba_file_rules where user_name = '{maurizio}';
  type   | database | user_name  | address |  auth_method
---------+----------+------------+---------+---------------
 hostssl | {all}    | {maurizio} | all     | scram-sha-256
(1 row)

postgres=# select name, setting from pg_settings where name in ('ssl', 'ssl_min_protocol_version', 'ssl_library');
          name           | setting
-------------------------+---------
 ssl                     | on
 ssl_library             | OpenSSL
 ssl_min_protocol_version | TLSv1.2
(3 rows)

postgres=# select name, setting from pg_settings where name = 'password_encryption';
        name         |    setting
---------------------+---------------
 password_encryption | scram-sha-256
(1 row)
```

[1] https://www.postgresql.org/docs/17/ssl-tcp.html

Tools

Computer Security Resource Center (CSRC)

- Cybersecurity and Privacy Reference Tool
  - Search for *password*
    - AC-07 UNSUCCESSFUL LOGON ATTEMPT
    - AC-12 SESSION TERMINATION
    - AU-02 EVENT LOGGING
    - IA-02 IDENTIFICATION AND AUTHENTICATION (ORGANIZATIONAL USERS)
    - ...

hashcat

hashcat

advanced password recovery

- hashcat
- Forum
- Wiki
- Tools
- Events
- Converter
- Contact

Download

Name	Version	Date	Download	Signature
hashcat binaries	v6.2.6	2022.09.02	Download	PGP
hashcat sources	v6.2.6	2022.09.02	Download	PGP

Signing key on PGP keyservers: RSA, 2048-bit. Key ID: 2048R/8A16544F. Fingerprint: A708 3322 9D04 0B41 99CC 0052 3C17 DA8B 8A16 544F

Check out our GitHub Repository for the latest development version

GPU Driver requirements:

- AMD GPUs on Linux require "AMDGPU" (21.50 or later) and "ROCm" (5.0 or later)
- AMD GPUs on Windows require "AMD Adrenalin Edition" (Adrenalin 22.5.1 exactly)
- Intel CPUs require "OpenCL Runtime for Intel Core and Intel Xeon Processors" (16.1.1 or later)
- NVIDIA GPUs require "NVIDIA Driver" (440.64 or later) and "CUDA Toolkit" (9.0 or later)

Features

- **World's fastest password cracker**
- **World's first and only in-kernel rule engine**
- Free
- Open-Source (MIT License)
- Multi-OS (Linux, Windows and macOS)
- Multi-Platform (CPU, GPU, APU, etc., everything that comes with an OpenCL runtime)
- Multi-Hash (Cracking multiple hashes at the same time)

Main page: https://hashcat.net/hashcat/
Binaries: https://hashcat.net/files/hashcat-6.2.6.7z

hashcat

Hash-Mode Hash-Name
Example

12 PostgreSQL
a6343a68d964ca596d9752250d54bb8a:postgres

11100 PostgreSQL CRAM (MD5)
postgres$postgres*f0784ea5*2091bb7d4725d1ca85e8de6ec349baf6

28600 PostgreSQL SCRAM-SHA-256
SCRAM-SHA-
256$4096:IKfxzJ8Nq4PkLJCfgKcPmA==$iRw3qwTp18uaBnsTOEExbtgWdKeBMbSSnZvqD4
sdqLQ=:hPciC1CcnBna3szR8Mf3MVc8t0W7QPbIHoMMrh4zRV0=

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

hashcat

Hash-Mode Hash-Name
Example

3100 Oracle H: Type (Oracle 7+)
7A963A529D2E3229:3682427524

112 Oracle S: Type (Oracle 11+)
ac5f1e62d21fd0529428b84d42e8955b04966703:3844574818447737813O

12300 Oracle T: Type (Oracle 12+)
78281A9C0CF626BD05EFC4F41B515B61D6C4D95A250CD4A605CA0EF97168D670EBCB5
673B6F5A2FB9CC4E0C0101E659C0C4E3B9B3BEDA846CD15508E88685A233414165504
766111066420254008225

20600 Oracle Transportation Management (SHA256)
otm_sha256:1000:1234567890:S5Q9Kc0ETY6ZPyQU+JYY60oFjaJuZZaSinggmzU8PC4=

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

hashcat

```
# plugin != caching_sha2_password

200     MySQL323
  7196759210defdc0

300     MySQL4.1/MySQL5
  fcf7c1b8749cf99d88e5f34271d636178fb5d130

11200     MySQL CRAM (SHA1)
  $mysqlna$1c24ab8d0ee94d70ab1f2e814d8f0948a14d10b9*437e93572f18ae44d9e779160c2
  505271f85821d

# plugin == caching_sha2_password

7401    MySQL $A$ (sha256crypt)
  $mysql$A$005*F9CC98CE08892924F50A213B6BC571A2C11778C5*625479393559393965 4
  14D453164774 56B484F41316E64484742577A2E316278535352 6B7554584647562F
```

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

hashcat

```
#! /usr/bin/env python3

  sqlcmd = r"""
      SELECT passwd AS hash, '@node.name@' AS nodename, usename
      FROM pg_catalog.pg_shadow
      WHERE usename like '@option.roles_like@'
      AND (
        ( '@option.roles_type@' = 'all' ) OR
        ( '@option.roles_type@' = 'unprivileged'
          AND NOT (usesuper OR userepl OR usebypassrls) ) OR
        ( '@option.roles_type@' = 'privileged'
          AND (usesuper OR userepl OR usebypassrls) )
      );
  """
  quoted_sqlcmd = shlex.quote(sqlcmd)
  SUCMD = f"sudo -u pg LD_LIBRARY_PATH={LD_LIBRARY_PATH} timeout {TIMEOUT}s "
  cmd = f"cd /tmp; {SUCMD} {BIN}/psql -h {socket} -p {port} --quiet --tuples-only
  --no-align --no-psqlrc --single-line -U postgres -c {quoted_sqlcmd}"
```

hashcat

```
SELECT authentication_string AS hash, '@node.name@' AS nodename, user, host, plugin
  FROM mysql.user
  WHERE plugin != 'caching_sha2_password'
  AND authentication_string NOT LIKE '%INVALIDSALTANDPASSWORD%'
  AND user like '@option.roles_like@'
  AND (
    ( '@option.roles_type@' = 'all' ) OR
   ( '@option.roles_type@' = 'unprivileged' AND NOT (Super_priv = 'Y' OR Repl_slave_priv = 'Y' OR Repl_client_priv
  = 'Y') ) OR
    ( '@option.roles_type@' = 'privileged' AND (Super_priv = 'Y' OR Repl_slave_priv = 'Y' OR Repl_client_priv =
  'Y') )
  )
  UNION
  SELECT  CONCAT('$mysql', SUBSTR(authentication_string,1,3),
  LPAD(CONV(SUBSTR(authentication_string,4,3),16,10),4,0),'*',INSERT(HEX(SUBSTR(authentication_string,8)),41,0,'*')
  ) AS hash, '@node.name@' nodename, user, host, plugin
  FROM mysql.user
  WHERE plugin = 'caching_sha2_password'
  AND authentication_string NOT LIKE '%INVALIDSALTANDPASSWORD%'
  AND user like '@option.roles_like@'
  AND (
    ( '@option.roles_type@' = 'all' ) OR
   ( '@option.roles_type@' = 'unprivileged' AND NOT (Super_priv = 'Y' OR Repl_slave_priv = 'Y' OR Repl_client_priv
  = 'Y') ) OR
    ( '@option.roles_type@' = 'privileged' AND (Super_priv = 'Y' OR Repl_slave_priv = 'Y' OR Repl_client_priv =
  'Y') )
  );
```

hashcat

hashcat (v6.2.6) starting in autodetect mode

You are probably missing the CUDA, HIP or OpenCL runtime installation.

* AMD GPUs on Linux require this driver:
  "AMDGPU" (21.50 or later) and "ROCm" (5.0 or later)
* Intel CPUs require this runtime:
  "OpenCL Runtime for Intel Core and Intel Xeon Processors" (16.1.1 or later)
* NVIDIA GPUs require this runtime and/or driver (both):
  "NVIDIA Driver" (440.64 or later)
  "CUDA Toolkit" (9.0 or later)

Started: Mon Mar 24 17:11:11 2025

Stopped: Mon Mar 24 17:11:11 2025

hashcat

```
[root@madegiortest02 hashcat-6.2.6]# dnf install clinfo
   ...
   [root@madegiortest02 hashcat-6.2.6]# clinfo
Number of platforms                               1
  Platform Name                                   AMD Accelerated Parallel Processing
  Platform Vendor                                 Advanced Micro Devices, Inc.
  Platform Version                                OpenCL 2.1 AMD-APP (3513.0)
  Platform Profile                                FULL_PROFILE
  Platform Extensions                             cl_khr_icd cl_amd_event_callback
  Platform Extensions function suffix             AMD
  Platform Host timer resolution                  1ns

  Platform Name                                   AMD Accelerated Parallel Processing
Number of devices                                 0

NULL platform behavior
  clGetPlatformInfo(NULL, CL_PLATFORM_NAME, ...)  AMD Accelerated Parallel Processing
...
ICD loader properties
  ICD loader Name                                 OpenCL ICD Loaderns
  ICD loader Vendor                               OCL Icd free softwarens
  ICD loader Version                              2.2.13ns
  ICD loader Profile                              OpenCL 3.0ns
```

hashcat

```
maurizio@pcitdb14:~/git_local/hashcat$ clinfo
Number of platforms                               1
  Platform Name                                   Portable Computing Language
  Platform Vendor                                 The pocl project
  Platform Version                                OpenCL 2.0 pocl 1.8  Linux, None+Asserts, RELOC, LLVM 11.1.0, SLEEF, DISTRO, POCL_DEBUG
  Platform Profile                                FULL_PROFILE
  Platform Extensions                             cl_khr_icd cl_pocl_content_size
  Platform Extensions function suffix             POCL

  Platform Name                                   Portable Computing Language
Number of devices                                 1
  Device Name                                     pthread-Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz
  Device Vendor                                   GenuineIntel
  Device Vendor ID                                0x8086
  Device Version                                  OpenCL 1.2 pocl HSTR: pthread-x86_64-pc-linux-gnu-haswell
  Driver Version                                  1.8
  Device OpenCL C Version                         OpenCL C 1.2 pocl
  Device Type                                     CPU
  Device Profile                                  FULL_PROFILE
  Device Available                                Yes
  Compiler Available                              Yes
  Linker Available                                Yes
  Max compute units                               8
  Max clock frequency                             4000MHz
  Device Partition                                (core)
    Max number of sub-devices                     8
    Supported partition types                     equally, by counts
    Supported affinity domains                    (n/a)
  Max work item dimensions                        3
  Max work item sizes                             4096x4096x4096
  Max work group size                             4096
  Preferred work group size multiple (kernel)     8
  Preferred / native vector sizes
    char                                                 16 / 16
    short                                                16 / 16
    int                                                   8 / 8
    long                                                  4 / 4
    half                                                  0 / 0        (n/a)
    float                                                 8 / 8
    double                                                4 / 4        (cl_khr_fp64)
```

"OpenCL™ (Open Computing Language) is an open, royalty-free standard for cross-platform, parallel programming of diverse accelerators found in supercomputers, cloud servers, personal computers, mobile devices and embedded platforms."
Source: https://www.khronos.org/opencl/

hashcat

```
maurizio@pcitdb14:~/git_local/hashcat/hashcat-6.2.6$ ./hashcat.bin -I
hashcat (v6.2.6) starting in backend information mode

OpenCL Info:
============

OpenCL Platform ID #1
  Vendor..: The pocl project
  Name....: Portable Computing Language
  Version.: OpenCL 2.0 pocl 1.8  Linux, None+Asserts, RELOC, LLVM 11.1.0, SLEEF, DISTRO, POCL_DEBUG

  Backend Device ID #1
    Type...........: CPU
    Vendor.ID......: 128
    Vendor.........: GenuineIntel
    Name...........: pthread-Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz
```

hashcat

```
#- [ Attack Modes ] -
#
#  # | Mode
# ===+======
#  0 | Straight
#  1 | Combination
#  3 | Brute-force
#  6 | Hybrid Wordlist + Mask
#  7 | Hybrid Mask + Wordlist
#  9 | Association
#
#- [ Workload Profiles ] -
#
#  # | Performance | Runtime | Power Consumption | Desktop Impact
# ===+=============+=========+===================+=================
#  1 | Low         |   2 ms  | Low               | Minimal
#  2 | Default     |  12 ms  | Economic          | Noticeable
#  3 | High        |  96 ms  | High              | Unresponsive
#  4 | Nightmare   | 480 ms  | Insane            | Headless
#
#- [ Outfile Formats ] -
#
#  # | Format
# ===+========
#  1 | hash[:salt]
#  2 | plain
#  3 | hex_plain
#  4 | crack_pos
#  5 | timestamp absolute
#  6 | timestamp relative
#
./hashcat-6.2.6/hashcat.bin --attack-mode 0 --workload-profile 3 \
        --outfile hashcat.out --outfile-format=1,2,3 \
        --potfile-path hashcat.potfile \
        ${HASHFILE_PATH} ${PASSWORDFILE_PATH}
```

hashcat

```
maurizio@pcitdb14:~/git_local/hashcat$ ./run_hashcat.sh ./hashfile.txt ./password-list.txt
hashcat (v6.2.6) starting in autodetect mode

OpenCL API (OpenCL 2.0 pocl 1.8  Linux, None+Asserts, RELOC, LLVM 11.1.0, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
============================================================================================================================================
* Device #1: pthread-Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz, 14944/29953 MB (4096 MB allocatable), 8MCU

Hash-mode was not specified with -m. Attempting to auto-detect hash mode.
The following mode was auto-detected as the only one matching your input hash:

28600 | PostgreSQL SCRAM-SHA-256 | Database Server

NOTE: Auto-detect is best effort. The correct hash-mode is NOT guaranteed!
Do NOT report auto-detect issues unless you are certain of the hash type.

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Hashes: 5 digests; 5 unique digests, 5 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
* Zero-Byte
* Slow-Hash-SIMD-LOOP

Watchdog: Temperature abort trigger set to 90c

Host memory required for this attack: 2 MB

Dictionary cache built:
* Filename..: ./password-list.txt
* Passwords.: 999998
* Bytes.....: 8529102
* Keyspace..: 999998
* Runtime...: 0 secs

[s]tatus [p]ause [b]ypass [c]heckpoint [f]inish [q]uit => s
```

hashcat

```
Session..........: hashcat
Status...........: Running
Hash.Mode........: 28600 (PostgreSQL SCRAM-SHA-256)
Hash.Target......: ./hashfile.txt
Time.Started.....: Mon Mar 24 18:36:58 2025 (53 secs)
Time.Estimated...: Mon Mar 24 18:41:29 2025 (3 mins, 38 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (./password-list.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:    11098 H/s (92.55ms) @ Accel:1024 Loops:512 Thr:1 Vec:8
Recovered........: 2/5 (40.00%) Digests (total), 2/5 (40.00%) Digests (new), 2/5 (40.00%) Salts
Progress.........: 966656/4999990 (19.33%)
Rejected.........: 0/966656 (0.00%)
Restore.Point....: 188416/999998 (18.84%)
Restore.Sub.#1...: Salt:3 Amplifier:0-1 Iteration:2560-3072
Candidate.Engine.: Device Generator
Candidates.#1....: 041366 -> skv20111996
Hardware.Mon.#1..: Temp: 74c Util: 98%

[s]tatus [p]ause [b]ypass [c]heckpoint [f]inish [q]uit => s

Session..........: hashcat
Status...........: Running
Hash.Mode........: 28600 (PostgreSQL SCRAM-SHA-256)
Hash.Target......: ./hashfile.txt
Time.Started.....: Mon Mar 24 18:36:58 2025 (1 min, 47 secs)
Time.Estimated...: Mon Mar 24 18:41:29 2025 (2 mins, 44 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (./password-list.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:    11106 H/s (91.18ms) @ Accel:1024 Loops:512 Thr:1 Vec:8
Recovered........: 2/5 (40.00%) Digests (total), 2/5 (40.00%) Digests (new), 2/5 (40.00%) Salts
Progress.........: 1957888/4999990 (39.16%)
Rejected.........: 0/1957888 (0.00%)
Restore.Point....: 385024/999998 (38.50%)
Restore.Sub.#1...: Salt:4 Amplifier:0-1 Iteration:3584-4095
Candidate.Engine.: Device Generator
Candidates.#1....: vn1993 -> twent3
Hardware.Mon.#1..: Temp: 81c Util: 99%

[s]tatus [p]ause [b]ypass [c]heckpoint [f]inish [q]uit => s
```

hashcat

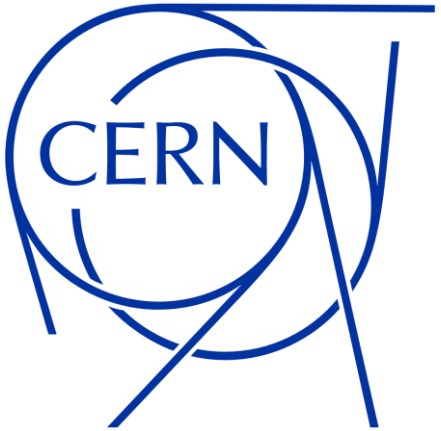
```
[s]tatus [p]ause [b]ypass [c]heckpoint [f]inish [q]uit => s

Session..........: hashcat
Status...........: Running
Hash.Mode........: 28600 (PostgreSQL SCRAM-SHA-256)
Hash.Target......: ./hashfile.txt
Time.Started.....: Mon Mar 24 18:36:58 2025 (2 mins, 58 secs)
Time.Estimated...: Mon Mar 24 18:41:31 2025 (1 min, 35 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (./password-list.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:    11054 H/s (92.29ms) @ Accel:1024 Loops:512 Thr:1 Vec:8
Recovered........: 2/5 (40.00%) Digests (total), 2/5 (40.00%) Digests (new), 2/5 (40.00%) Salts
Progress.........: 3260416/4999990 (65.21%)
Rejected.........: 0/3260416 (0.00%)
Restore.Point....: 647168/999998 (64.72%)
Restore.Sub.#1...: Salt:3 Amplifier:0-1 Iteration:2048-2560
Candidate.Engine.: Device Generator
Candidates.#1....: admiration -> 922846418
Hardware.Mon.#1..: Temp: 83c Util: 97%

[s]tatus [p]ause [b]ypass [c]heckpoint [f]inish [q]uit => s

Session..........: hashcat
Status...........: Running
Hash.Mode........: 28600 (PostgreSQL SCRAM-SHA-256)
Hash.Target......: ./hashfile.txt
Time.Started.....: Mon Mar 24 18:36:58 2025 (3 mins, 45 secs)
Time.Estimated...: Mon Mar 24 18:41:30 2025 (47 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (./password-list.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:    11076 H/s (91.27ms) @ Accel:1024 Loops:512 Thr:1 Vec:8
Recovered........: 2/5 (40.00%) Digests (total), 2/5 (40.00%) Digests (new), 2/5 (40.00%) Salts
Progress.........: 4128768/4999990 (82.58%)
Rejected.........: 0/4128768 (0.00%)
Restore.Point....: 819200/999998 (81.92%)
Restore.Sub.#1...: Salt:4 Amplifier:0-1 Iteration:2560-3072
Candidate.Engine.: Device Generator
Candidates.#1....: yummysam -> YpaxUhAJOpeNo
Hardware.Mon.#1..: Temp: 84c Util: 99%
```

hashcat

```
[s]tatus [p]ause [b]ypass [c]heckpoint [f]inish [q]uit => s

Session..........: hashcat
Status...........: Running
Hash.Mode........: 28600 (PostgreSQL SCRAM-SHA-256)
Hash.Target......: ./hashfile.txt
Time.Started.....: Mon Mar 24 18:36:58 2025 (4 mins, 19 secs)
Time.Estimated...: Mon Mar 24 18:41:29 2025 (12 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (./password-list.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:    11087 H/s (91.26ms) @ Accel:1024 Loops:512 Thr:1 Vec:8
Recovered........: 2/5 (40.00%) Digests (total), 2/5 (40.00%) Digests (new), 2/5 (40.00%) Salts
Progress.........: 4775936/4999990 (95.52%)
Rejected.........: 0/4775936 (0.00%)
Restore.Point....: 950272/999998 (95.03%)
Restore.Sub.#1...: Salt:3 Amplifier:0-1 Iteration:2048-2560
Candidate.Engine.: Device Generator
Candidates.#1....: websters1 -> was1107
Hardware.Mon.#1..: Temp: 81c Util: 98%

Approaching final keyspace - workload adjusted.


Session..........: hashcat
Status...........: Exhausted
Hash.Mode........: 28600 (PostgreSQL SCRAM-SHA-256)
Hash.Target......: ./hashfile.txt
Time.Started.....: Mon Mar 24 18:36:58 2025 (4 mins, 32 secs)
Time.Estimated...: Mon Mar 24 18:41:30 2025 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (./password-list.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:    11092 H/s (74.58ms) @ Accel:1024 Loops:512 Thr:1 Vec:8
Recovered........: 2/5 (40.00%) Digests (total), 2/5 (40.00%) Digests (new), 2/5 (40.00%) Salts
Progress.........: 4999990/4999990 (100.00%)
Rejected.........: 0/4999990 (0.00%)
Restore.Point....: 999998/999998 (100.00%)
Restore.Sub.#1...: Salt:4 Amplifier:0-1 Iteration:3584-4095
Candidate.Engine.: Device Generator
Candidates.#1....: vjq6frrfeyn -> vjht008
Hardware.Mon.#1..: Temp: 82c Util: 98%

Started: Mon Mar 24 18:36:55 2025
Stopped: Mon Mar 24 18:41:31 2025
```

hashcat

```
maurizio@pcitdb14:~/git_local/hashcat/hashcat-6.2.6$ cat hashcat.potfile
SCRAM-SHA-256$4096:R3F3eQ66B2OdR19o1/neNQ==$a8W2MiglY6SSTTiE1bbc044jQ0GCM0kvEIo865aXr20=:j+BbdesckDQLY6M+6QQlSZEIYElRYXW3FlQ9SFeai2Q=:changeme
SCRAM-SHA-256$4096:ZG9oAJDMK2sKgx9vPD2FzQ==$w5yPFEQIF2A07pw2zNkW7ANVbD+tK/HxDgxwS6gz3nk=:BBBHBeI4UlBwNi+lYc/6ufXlNrcxvFCuzqKLlCXDQlg=:changeme
maurizio@pcitdb14:~/git_local/hashcat/hashcat-6.2.6$ head hashcat.log
TOP.67e19837.000cc73a   START
TOP.67e19837.000cc73a   folder_config->cwd      /home/maurizio/git_local/hashcat
TOP.67e19837.000cc73a   folder_config->install_dir      /home/maurizio/git_local/hashcat/hashcat-6.2.6
TOP.67e19837.000cc73a   folder_config->profile_dir      /home/maurizio/git_local/hashcat/hashcat-6.2.6
TOP.67e19837.000cc73a   folder_config->session_dir      /home/maurizio/git_local/hashcat/hashcat-6.2.6
TOP.67e19837.000cc73a   folder_config->shared_dir       /home/maurizio/git_local/hashcat/hashcat-6.2.6
TOP.67e19837.000cc73a   user_options->encoding_from     utf-8
TOP.67e19837.000cc73a   user_options->encoding_to       utf-8
TOP.67e19837.000cc73a   user_options->outfile   hashcat.out
TOP.67e19837.000cc73a   user_options->rule_buf_l        :
maurizio@pcitdb14:~/git_local/hashcat/hashcat-6.2.6$ tail hashcat.log
TOP.67e19d67.00063e56   SUB.67e19d69.0002639e   START
TOP.67e19d67.00063e56   SUB.67e19d69.0002639e   straight_ctx->dict      ./password-list.txt
TOP.67e19d67.00063e56   SUB.67e19d69.0002639e   runtime_start   1742839145
TOP.67e19d67.00063e56   SUB.67e19d69.0002639e   runtime_stop    1742839429
TOP.67e19d67.00063e56   SUB.67e19d69.0002639e   hashes->digests_done_new        0
TOP.67e19d67.00063e56   SUB.67e19d69.0002639e   status-after-work       5
TOP.67e19d67.00063e56   SUB.67e19d69.0002639e   STOP
TOP.67e19d67.00063e56   proc_start      1742839143
TOP.67e19d67.00063e56   proc_stop       1742839430
TOP.67e19d67.00063e56   STOP
```



“Big red” part

Miro Potocky

Humble Introduction

- Software engineer by school
- Network engineer by studying
- Storage admin by chance
- Oracle DBA by (hopefully not just) title
- Ex HP, HPE
- Calling CERN `127.0.0.1` since 2013
- DB Services team lead
- m(dot)p(at)cern.ch

Oracle DB accounts landscape

- **11,915 Oracle accounts as of 25.March 2025**
- **Organically grown with love since the 80’s**
- **Still serving 10g (☹) clients**
- **Oldest – still used – application in PROD is from Oracle 8i times**
  - Cordoned off to separate instance with strict(-ish) firewall
  - Refused to die^H^Hecomission despite periodic verbal threats
- **VERY specific users and requirements**
  - Physicists – theoretical and real ones
  - Experiment and machine operators
  - Business computing developers
  - Data scientists
  - Various lucky (/s) people tasked with maintaining legacy software with minimum usable documentation

The “Age” problem

- **Cause**
  - Legacy, exceptions, excuses, yadda - yadda
  - User reluctance, caution or missing responsibility
- **Result**
  - Oracle schemas with 10g, 11g, 12c and above password hashes

```
SQL> select distinct password_versions from dba_users;

PASSWORD_VERSIONS
-----------------
10G (!!!!)
11G 12C
10G 11G
…
```

- **Old passwords update frequency is inversely proportional to number of locations where the password is stored and used and the proportion is compounded by age**
- **Number of locations where the password is stored is directly proportional to intensity of requests to not expire it**

The solution

- **I hope you remember what Maurizio said before (prizes!)**
- **Essentially implementation differs only slightly**
- **No need for client side password wrangling – everything server side**
  - Did you enable Native Network Encryption!?

```
SQLNET.ENCRYPTION_SERVER=REQUIRED
SQLNET.ENCRYPTION_TYPES_SERVER=(AES256)
```

  - TLS works too
  - Nuke old clients

```
SQLNET.ALLOWED_LOGON_VERSION_SERVER=12a
```

- **Password verify function**
  - See ?/rdbms/admin/utlpwdmg.sql
  - *ora12c_verify_function and ora12c_strong_verify_function*
  - *Multitenancy note: 21c -* `CREATE MANDATORY PROFILE c##omni_profile LIMIT PASSWORD_VERIFY_FUNCTION ora12c_verify_function CONTAINER = ALL;`

Verify, verify, verify

Row, row, update row, gently in the tab

Verify, verify, verify, verify hash is just a map.

- Unknown author (probably DBA)

- ***Ora12c_verify_function***
  - Password at least 8 characters, at least 1 letters, at least 1 digits, must not contain database name, must not contain user name or reverse user name, must not contain oracle, password must differ by at least 3 characters from the old password, must not be too simple like welcome1
  - Most of the rules are self-explanatory and simple to adapt
  - But "must not be too simple like welcome1" – how is that checked?
    - `IF pw_lower IN ('welcome1', 'database1', 'account1', 'user1234', 'password1', 'oracle123', 'computer1', 'abcdefg1', 'change_on_install') THEN … bad password!`
  - We can do better than that!

Cracklib’s handy wordlists

- https://github.com/cracklib/cracklib/tree/main/words/files
- **External (table) help**
  - `CREATE DIRECTORY my_wordfiles AS ….`
  - `CREATE TABLE wordfile ( word varchar2(4000) )`

```
ORGANIZATION EXTERNAL (
DEFAULT DIRECTORY my_wordfiles
ACCESS PARAMETERS (
  RECORDS DELIMITED BY NEWLINE
  FIELDS TERMINATED BY ‘,’
  LOCATION ‘wordfile.txt’ )
);
```

    - `IF pw_lower IN (SELECT * FROM wordfile) … bad password!`
  - Lots of fun can also be had with the `string_distance` function of `utlpwdmg.sql`

(Dude) where's my hash?

```
SELECT username, password FROM dba_users; --Not since 2003
```

- **Peek at sys.user$**
  - PASSWORD column with 10g hash visible
  - SPARE4 column
    - Multiple hashes visible
    - Concatenated hashes with identifiers (T – 12c, None – 10g, H: XDB, S: 11g)
    - Example:
      `'T:511A70048CFB5B531196CDD2CB51393E05E3FBFB0CB019DB39AB4AAB717BB23CA7FB2EA0AD4F60B34C38C9B84F97BA0C6A4A7530362FBF23492FB02139442AB758645C9EA1D1E33C33CB9454D0468BF9;AEB6397C8E7598A7;H:55C984560827F4CE3A0F926B2A50C7DC;S:7233E3A91B45F6B813BCFFB5D8669167CB4F498D0642558A8A3BB39948C0';`
    - Easy extraction with `SELECT REGEXP_SUBSTR(spare4, 'T:[^;]+')`

Better be quick with the change

- **10g as worst case**
  - DES-CBC with fixed key
  - 8 lowercase characters password (26^8 = 208827064576 combinations)
  - Normal commodity HW (ex. NVidia A5000 or 5090) gets over 67000MH/s for DES hash (https://gist.github.com/Chick3nman)
    - 8 Chars in several seconds
    - 10 Chars in 0.5h
    - 12 Chars in 17days
  - Nice stuff (e.g. NVidia H100) gets over 78000MH/s
    - You can get 8xH100 node for $10/h (https://www.oracle.com/cloud/price-list/)
    - 12 Chars in 42h for $420
    - 14 Chars in 24days for measly $288k (with 50 H100 nodes)
  - Rainbow tables makes things even more interesting since the salt is known

So where’s the problem?

- **It’s people.**

- **Nobody likes to change “things that work” ™**
- **Admins can’t just UPDATE sys.user$ and watch the world burn**
  - However, one can `ALTER USER name IDENTIFIED BY VALUES ‘T:ABCD123……’;`
- **Password rollover enabled for limited amount of time helps a lot**
  - `ALTER PROFILE existingProfile LIMIT PASSWORD_ROLLOVER_TIME 1; -- days`
  - `SELECT * FROM unified_audit_trail WHERE action_name='LOGON’ AND authentication_type LIKE '%VERIFIER=11G-OLD%’; -- or %VERIFIER=12C-OLD%`
  - Not available for administrative users (e.g. SYS)
  - Can’t be used for KRB, Cert, RADIUS, CMU …

End game

- **12c hash only**
  - Significantly larger hash with 160 characters compared to 16 before
  - Hash-rate with H100s is in kH/s – not (yet) viable for brute forcing (still susceptible to wordlist attacks)
  - Case sensitive and much stronger cipher
    - Watch for: `sec_case_sensitive_logon=FALSE`
- **SPARE4 column contains only 12c “T: hashes”**
- **Password verify function follows NIST rules AND checks word/password lists**
- **Educated users understanding no-expiration password is not an option**
- **Periodic change of passwords unless you can verify 2nd factor**
- **Dead tree version of account policy to wave around and hide behind**
- **Enabled password rollover to win some brownie points back after all above**

Even best passwords are leaked – therefore AUDIT!


CERN