APPLICATION SECURITY, INC.

Cryptographic flaws in Oracle Database authentication protocol

Esteban Martínez Fayó

TeamSHATTER

TeamSHATTER
The Leading Database Threat Resource

Agenda

- Overview of Oracle native logon protocol
- How did I discover the vulnerabilities?
  - OCIPasswordChange Bruteforcer
- Oracle stealth password cracking vulnerability in logon protocol
  - Workarounds
- AUTH_NEW_PASSWORD padding vulnerability
- Comparison

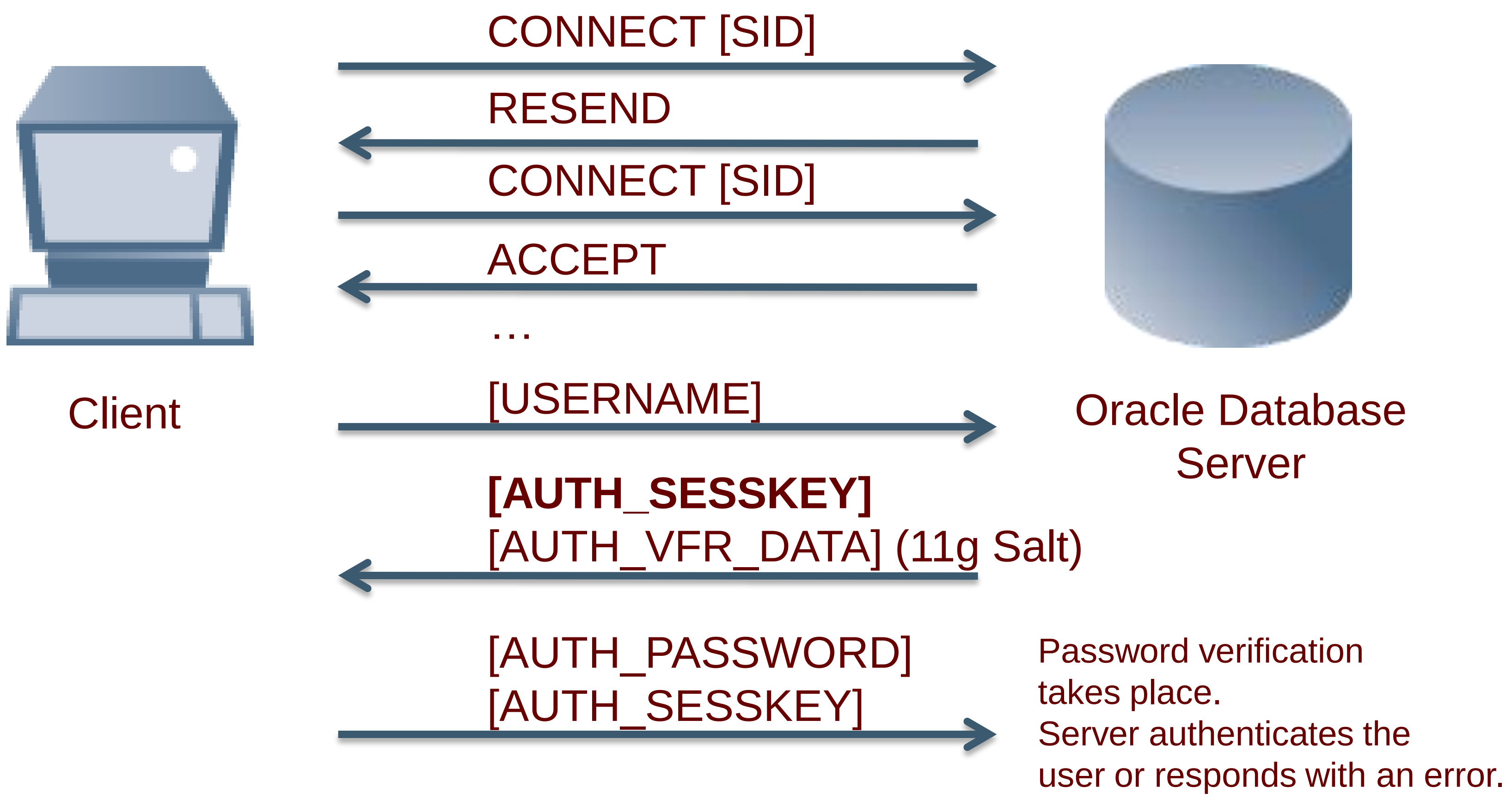
Oracle native authentication protocol

- Challenge/Response protocol to authenticate database users
- Used for users identified by passwords
  - Not used for external users.
- Different in each Database version
  - 8i/9i/10g/11g/12

Oracle native authentication protocol

- Client sends the password encrypted with random session keys
  - The Session keys are encrypted using the password hash.
- Session key (AUTH_SESSKEY)
  - Random value encrypted by password hash
  - There is one generated by Server and another generated by Client
  - A Combination of both session keys are used to encrypt the password that is sent by the client

Oracle native authentication protocol



Differences between 10g and 11g authentication

- 10g Authentication:
  - Proprietary DES based hashing algorithm (used in versions < 11)
  - Non-random salt (the username is the salt)
  - Password is case insensitive
  - 32 byte Session key (AUTH_SESSKEY).
- 11g Authentication:
  - Standard SHA-1 hashing algorithm (used in versions >= 11)
  - Random salt
  - Password is case sensitive
  - Compatible with 10g clients
  - 48 byte Session key (AUTH_SESSKEY).
- Cracking tools for DES based hashes are slower than SHA-1 hashes. See [1].

Some current known attacks

- With a sniffed authentication sequence it is possible to:
  - Do brute force attack on user password
  - Decrypt the password if the password hash is known
  - See tool [4].
- Downgrading
  - Force the authentication to be done in a lower version of the protocol (easier to crack)
  - Requires the attacker to perform some kind of man-in-the-middle attack
  - See [2] and [3].
- The vulnerabilities described in this presentation do not require packet sniffing or MiTM attacks

OCIPasswordChange Bruteforcer

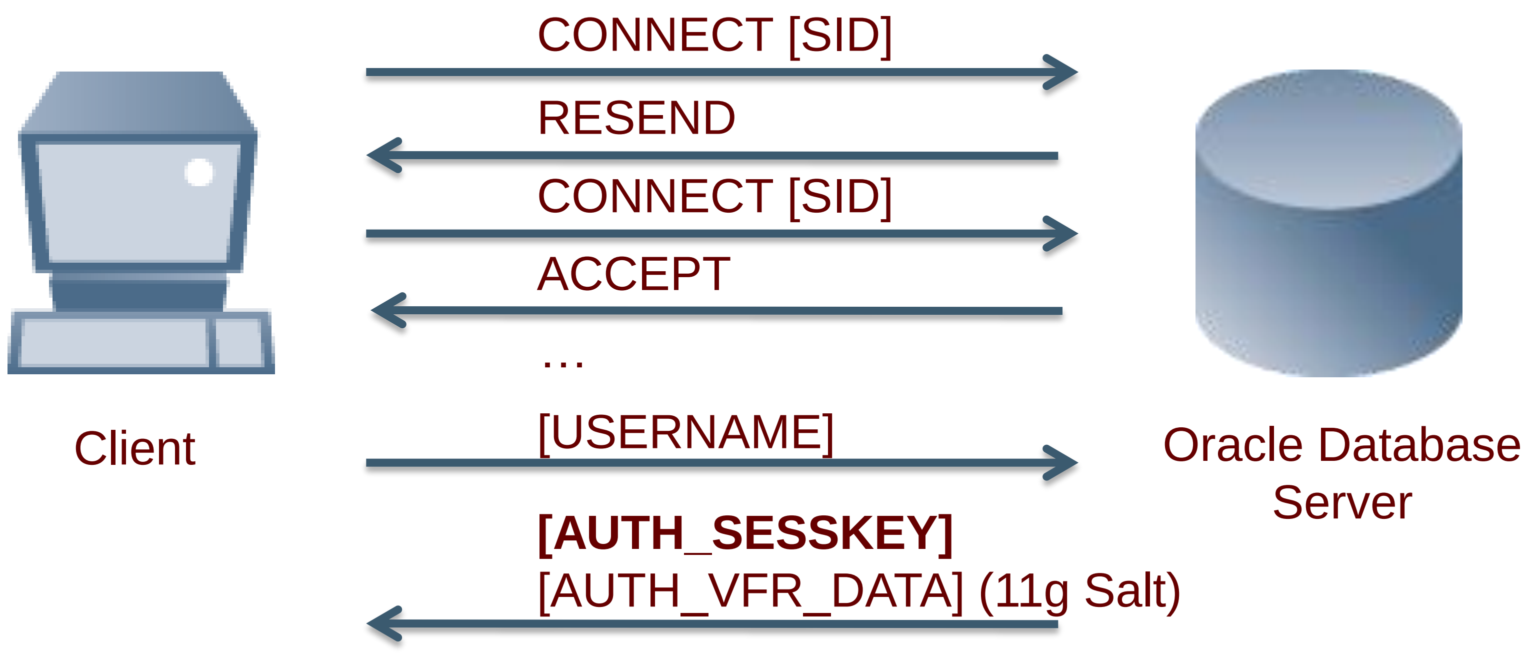
- This program uses OCIPasswordChange API to logon and change the password simultaneously
  - This is used for example for accounts with expired passwords, so that the user is able to change the password to a new one
  - The authentication process is a bit different than a normal authentication (AUTH_NEW_PASSWORD is sent).
- The program tries to logon and change the password repeatedly.

OCIPasswordChange Bruteforcer

- Strange Behavior (11g):
  - Accounts are locked after a much higher number of invalid attempts than what is configured in the user Profile
  - VERY STRANGE: Most of the times, when password is wrong, Oracle 11g client returns ORA-1017 without sending the Password.

OCIPasswordChange Bruteforcer

- Looking at the network packet capture:



Most of the times, when password is wrong, Client Stops the process here, without sending the password, knowing that the authentication is going to fail. How is it possible?

OCIPasswordChange Bruteforcer

- It means that with AUTH_SESSKEY and AUTH_VFR_DATA sent from the server it is possible to know that a password is *probably* correct or not.

OCIPasswordChange Bruteforcer

- Analyzing different cases:
  - Password is sent by the client:
    Decryption of AUTH_SESSKEY using the password hash resulted in a valid padding.
  - Password is not sent:
    Decryption of AUTH_SESSKEY resulted in an invalid padding (most of the cases when password is wrong).

Oracle 11g Session Key

- Oracle 11g Session key (AUTH_SESSKEY)
  - AUTH_SESSKEY is a 40 byte random session key encrypted using AES-192-CBC with PKCS7 padding. The key used to encrypt it is the user password hash (11g). The IV (Initialization Vector) used is always Null (0x00 bytes)
  - AES is a block cipher with 128 bit (16 byte) block size ➔ AUTH_SESSKEY is 48 byte length with 8 byte of padding
  - This means that when decrypting AUTH_SESSKEY with the correct password hash, it must end with a valid padding of 8 bytes: 8 bytes with value 0x08
  - There was no need to use PKCS7 because length is actually fixed (40 bytes).

Oracle 11g Session Key

- `CLEARTEXT_AUTH_SESSKEY`: 40 byte Random session key generated by the Server.

- `AUTH_SESSKEY = AES192_ENCRYPT_CBC (CLEARTEXT_AUTH_SESSKEY || 0x08*8, Using KEY=11gPasswordHash || 0x00*4)`

- The server gets `11gPasswordHash` from `SYS.USER$.SPARE4` column table.
- The client calculates `11gPasswordHash=SHA1(Password || SALT)`

Oracle Stealth Password Cracking Vulnerability

- To know if a given password **{pass}** is correct: Calculate
- `CLEARTEXT_AUTH_SESSKEY_WITH_PADDING = AES192_DECRYPT_CBC (AUTH_SESSKEY, Using KEY = SHA1(`**`{pass}`**`+SALT) || 0x00*4)`
- AUTH_SESSKEY and SALT (AUTH_VFR_DATA) are sent by the server.
- If the last 8 bytes of CLEARTEXT_AUTH_SESSKEY_WITH_PADDING have value 0x08, then **{pass}** is the correct password for the user.
  - In theory, this is true with probability 1-1/2^64 ➔ There may be false positives 1 every 10^19.

Oracle Stealth Password Cracking Vulnerability

- Requirements for the attack:
  - Network connection to a database instance
  - Know the **Database SID** and a valid **username** that is authenticated using a password (SHA-1 11g hash).
- What can be done:
  - Offline bruteforce of passwords for a given user
  - No audit trail left on the server for invalid login attempt.
- Affected versions: 11.1 and 11.2.
- Similar as cracking the password hash (has one additional step)
  - The cracking algorithm requires to calculate SHA-1 password hash, and then use the hash for an AES decryption (1 block).

Oracle Stealth Password Cracking Vulnerability

- POC Cracking tool (orasessbf)
  - C++, OpenSSL
  - Input data: AUTH_SESSKEY, AUTH_VFR_DATA (Salt)
  - Both values are sent by the server as the first step in the authentication process, the client just need to supply a username
  - Cracking Speed: ~1 M pwd/sec per CPU thread.
- Userenum11g.py
  - Input data: Oracle connection data (hostname/sid), username
  - Returns: AUTH_SESSKEY and AUTH_VFR_DATA for the user
  - The authentication is stopped before sending Password: No invalid login is recorded on server side.

Oracle Stealth Password Cracking Vulnerability

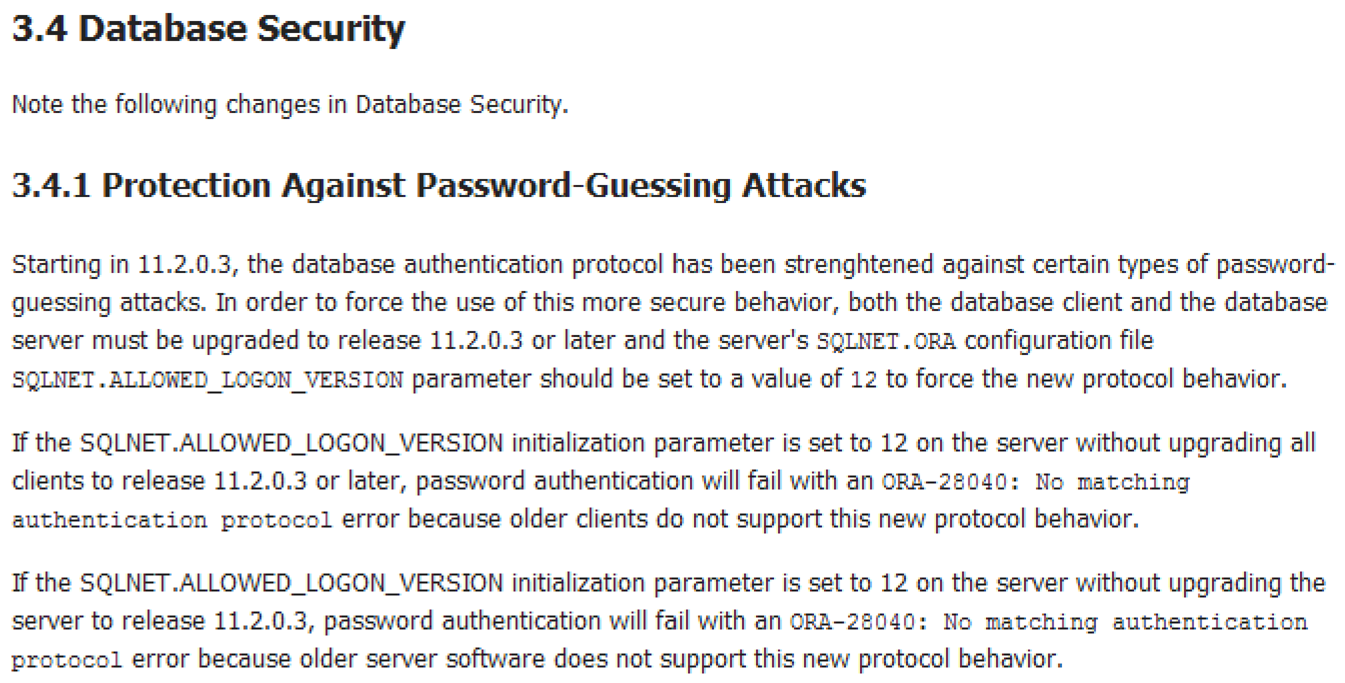
- Cracking speed can be increased with new techniques
- Using GPU
  - Can increase cracking speed by orders of magnitude.
- Dictionary and hybrid attacks
  - Much more efficient than bruteforce
  - Can try a dictionary with 10 million passwords in a few seconds.
- Rainbow tables can’t be used because of the Salt.

Oracle fix – Logon protocol version 12

- Oracle fixed this in Oracle Logon protocol version 12
  - Not part of a Critical Patch Update (CPU).
  - Changes in client and server side.
  - Available in 11.2.0.3 patchset.
  - No plans to fix it in 11.1.
- Still vulnerable by default, to stop being vulnerable:
  - Require protocol version 12 in server (SQLNET.ORA file configuration):
    SQLNET.ALLOWED_LOGON_VERSION=12
  - **Without this step the server is still vulnerable.**
  - **Oracle Clients older than 11.2.0.3 will not be able to connect.**

Oracle fix – 11.2.0.3 (Logon protocol version 12)

- From Readme file for 11.2.0.3:



3.4 Database Security

Note the following changes in Database Security.

3.4.1 Protection Against Password-Guessing Attacks

Starting in 11.2.0.3, the database authentication protocol has been strenghtened against certain types of password-guessing attacks. In order to force the use of this more secure behavior, both the database client and the database server must be upgraded to release 11.2.0.3 or later and the server's `SQLNET.ORA` configuration file `SQLNET.ALLOWED_LOGON_VERSION` parameter should be set to a value of `12` to force the new protocol behavior.

If the SQLNET.ALLOWED_LOGON_VERSION initialization parameter is set to 12 on the server without upgrading all clients to release 11.2.0.3 or later, password authentication will fail with an `ORA-28040: No matching authentication protocol` error because older clients do not support this new protocol behavior.

If the SQLNET.ALLOWED_LOGON_VERSION initialization parameter is set to 12 on the server without upgrading the server to release 11.2.0.3, password authentication will fail with an `ORA-28040: No matching authentication protocol` error because older server software does not support this new protocol behavior.

- Could someone imagine that this is to address such a critical vulnerability?

Oracle fix – Logon protocol version 12

- How did they fix it?
  - `AUTH_SESSKEY = AES192_ENCRYPT (CLEARTEXT_AUTH_SESSKEY || RANDOM_BYTE*8, Using KEY = SHA1PasswordHash[20] + 0x00*4)`
- CLEARTEXT_AUTH_SESSKEY (the Server Session Key) continues to be 40 bytes long.
- The padding was replaced by random data, instead of fixed 0x08 bytes.
- **Clients with protocol version 11 are not compatible because they are expecting AUTH_SESSKEY to be encrypted with PKCS7 padding so decryption will fail.**

Workarounds – Use external authentication

- Use external authentication (Network/SSL, Directory service, OS).
- Pros:
  - Authentication is performed by a separate component which is usually considered more secure than native database authentication
  - Allow integration with other components of the infrastructure.
- Cons:
  - May be difficult to implement. Requires architectural changes.

Workarounds – Disallow 11g authentication

- The idea is to stop using 11g authentication and use 10g
  - In this case: Not always a newer version is more secure than older version…
- Two ways:
  - `SEC_CASE_SENSITIVE_LOGON=FALSE`
  - Remove 11g hashes
- BEFORE implementing them verify that there are 10g password hashes for all users
  - `SELECT NAME FROM SYS.USER$ WHERE PASSWORD IS NULL AND SPARE4 IS NOT NULL;`

Workarounds – sec_case_sensitive_logon=FALSE

- Set SEC_CASE_SENSITIVE_LOGON initialization parameter to FALSE:
  - `ALTER SYSTEM SET SEC_CASE_SENSITIVE_LOGON = FALSE;`
- Regenerate password file (this is for SYSDBA users)
  - `orapwd file=`*`{password_file}`*` ignorecase=y`
  - *{password_file}* is {ORACLE_HOME}\database\PWD{SID}.ora (Windows) or {ORACLE_HOME}/dbs/orapw{SID} (Unix/Linux)

Workarounds – sec_case_sensitive_logon=FALSE

- Pros:
  - Good compatibility (Oracle native authentication continues to be used). Easy to implement.
- Cons:
  - Improvements in 11g authentication are lost: Case sensitive passwords, hash with random salt, stronger encryption.

Workarounds – Remove 11g hashes

- Remove 11g hashes (force Oracle to use 10g hashes, if available):
  - If 11g hashes are not available Oracle authenticates with 10g hashes using 10g authentication.
  - As SYSDBA:

    ```
    UPDATE SYS.USER$ SET SPARE4=NULL WHERE
    LENGTH(PASSWORD)=16;
    COMMIT;
    ```
  - Restart the database
  - Regenerate password file, as shown in previous slide, using `orapwd file=`*`{password_file}`*` ignorecase=y`

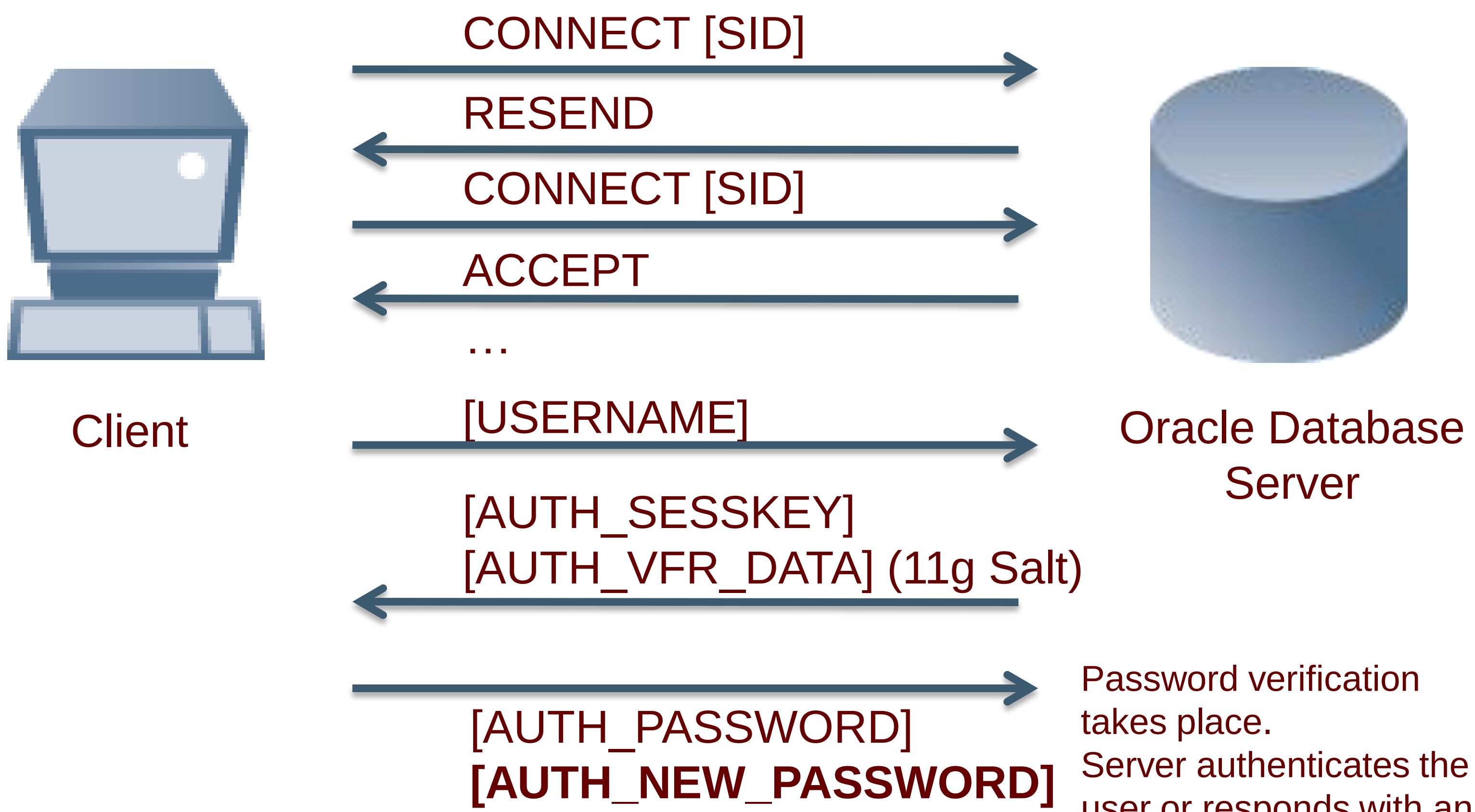
Workarounds – Remove 11g hashes

- Pros:
  - Can be implemented selectively by account.
  - Good compatibility (Oracle native authentication continues to be used). Easy to implement.
- Cons:
  - 11g hashes are regenerated when creating new users or changing passwords so the workaround needs to be reapplied.
  - Improvements in 11g authentication are lost: Case sensitive passwords, hash with random salt, stronger encryption.

OCIPasswordChange Bruteforcer (10g)

- Strange Behavior with Oracle 10g:
  - Accounts are locked after a much higher number of invalid attempts than what is configured in the user Profile
  - Once the account is locked most of the time we get ORA-1017 errors and very few (random) ORA-28000
  - Password can be changed even on locked accounts (need to know current password) (CVE-2012-0510). See advisory at [5].

Oracle native auth protocol - OCIPasswordChange



Password verification takes place.
Server authenticates the user or responds with an error.

AUTH_NEW_PASSWORD

- AUTH_NEW_PASSWORD (10g/11g) is the new password sent by the client encrypted with Server Session Key.
- Uses PKCS7 Padding.
- AUTH_NEW_PASSWORD=RANDOM_BYTES*16 || AES_ENCRYPT({NewPassword}||Padding)
- No Client side Session Key.

Change password vulnerability

- When using OCIPasswordChange to logon and change password if AUTH_NEW_PASSWORD does not have a valid padding:
  - the server does not record a failed login
  - when the account is locked, the server responds with an ORA-1017 (Invalid username/password) instead of an ORA-28000 (Account locked) error.
- With this information an attacker can mount an offline bruteforce attack on the user password. See advisory at [6].

AUTH_NEW_PASSWORD padding vuln – 10g

- Given a certain AUTH_SESSKEY and AUTH_NEW_PASSWORD that have recorded a failed login (or returned an ORA-28000 error)
- To know if {pass} is “probably correct”, calculate: PassKey1 = RIGHT (AES_128_DECRYPT (AUTH_SESSKEY, Using Key=10gHash({username}+{pass}), 16)
- NewPassword = AES_128_DECRYPT (AUTH_NEW_PASSWORD, Using Key=PassKey1)
- If NewPassword has a valid Padding then {pass} is “probably correct”.

AUTH_NEW_PASSWORD padding vuln – 11g

- Given a certain AUTH_SESSKEY and AUTH_NEW_PASSWORD that have recorded a failed login (or returned an ORA-28000 error)
- To know if {pass} is “probably correct”, calculate: PassKey1 = RIGHT (AES_192_DECRYPT (AUTH_SESSKEY, Using Key=11gHash({pass},AUTH_VFR_DATA),24)
- NewPassword = AES_192_DECRYPT (AUTH_NEW_PASSWORD, Using Key=PassKey1)
- If NewPassword has a valid Padding then {pass} is “probably correct”.

AUTH_NEW_PASSWORD padding vuln

- What means “probably correct”?
  How can you be sure (at least with a high confidence) that {pass} is the correct password?
  - Random data has a probability of ~ 1 / 256 to have a valid padding
  - The attacker needs to collect a few (6 should be enough) AUTH_SESSKEY and AUTH_NEW_PASSWORD pairs that have returned ORA-28000 from server (meaning that padding was valid)
  - If the test verifies that {pass} is "probably correct" for **all** AUTH_SESSKEY and AUTH_NEW_PASSWORD pairs, then {pass} is the correct password
  - The probability of N tests to have valid padding: ~ 1 / 256^N
    For N=6: ~ 1 / 2.8E+14

Comparison between padding vulnerabilities

- Similarities
  - Remote unauthenticated vulnerabilities
  - Both cracking algorithm requires to compute Oracle password hash and AES decryption
  - Both are related to bad use of cryptographic padding.

Comparison between padding vulnerabilities

- Differences

Oracle stealth password cracking vulnerability in logon protocol	New password padding Vulnerability (CVE-2012-0511)
No audit trail left on server. Minimum interaction with the Server required.	Audit trail for failed logon. Requires multiple logon attempts and to lock the account.
After cracking the password the attacker can log on immediately.	After cracking the password, the attacker will be able to log on once the account has been unlocked.
Affects Oracle Database Releases 11.1 and 11.2	Affects Oracle Database Releases 10.1, 10.2 and 11.1 (< CPU-APR-12)
The fix was implemented in the next version of the protocol (12)	The fix was provided in the same version of the protocol by a CPU (April 2012).

References

- [1]:http://hashcat.net/oclhashcat-plus/
- [2]: http://www.pwc.com/en_HU/hu/services/assets/oraauthdg-pub.pdf
- [3]: http://soonerorlater.hu/download/hacktivity_lt_2009_en.pdf
- [4]: http://www.soonerorlater.hu/index.khtml?article_id=513
- [5]: https://www.teamshatter.com/?p=3443
- [6]: https://www.teamshatter.com/?p=3434

- For more information about TeamSHATTER Research:
  http://www.teamshatter.com/team-shatter-exclusive/
  Follow: @TeamSHATTER

APPLICATION SECURITY, INC.

END

Questions?

Thank You.

E-mail: esteban>at<appsecinc>dot<com
Twitter: @estemf

TeamSHATTER
The Leading Database Threat Resource