[SYSS-2021-061] Oracle Database - NNE Connection Hijacking

From: Moritz Bechler <moritz.bechler () syss de>
Date: Fri, 10 Dec 2021 10:18:20 +0100

```
Advisory ID:               SYSS-2021-061
Product:                   Database
Manufacturer:              Oracle
Affected Version(s):       12.1.0.2, 12.2.0.1, 19c
Tested Version(s):         18c
Vulnerability Type:        Protection Mechanism Failure (CWE-693)
Risk Level:                High
Solution Status:           Fixed
Manufacturer Notification: 2021-03-17
Solution Date:             2021-08-07
Public Disclosure:         2021-12-10
CVE Reference:             CVE-2021-2351
Author of Advisory:        Moritz Bechler, SySS GmbH

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Overview:

Oracle Database is a general purpose relational database management
system (RDMBS).

The manufacturer describes the product as follows (see [1]):

"Oracle database products offer customers cost-optimized and high-performance
versions of Oracle Database, the world's leading converged, multi-model
database management system, as well as in-memory, NoSQL and MySQL databases.
Oracle Autonomous Database, available on premises via Oracle Cloud@Customer
or in the Oracle Cloud Infrastructure, enables customers to simplify relational
database environments and reduce management workloads."

To protect the client/server communication, a proprietary security protocol
"Native Network Encryption" (NNE) is used.
A TLS-based alternative can optionally be configured.

Due to insecure fallback behavior, a man-in-the-middle attacker can bypass
NNE's protection against man-in-the-middle attacks and hijack authenticated
connections. In some configurations, a full man-in-the-middle attack is
possible.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Vulnerability Details:

To mitigate against man-in-the-middle attacks on the initial Diffie-
Hellman key exchange, the protocol implements the mixin of an additional
shared key that is established by the authentication protocol
(typically O5Logon). This relies on the fact that both client and server
have knowledge of the user password (hash), which a potential attacker
does not have.

For more details on the protocol, refer to our paper [4].

SySS, however, found out that the JDBC Thin client implementation did
not implement that fold-in and its connections were still accepted
by database servers. The server performs a fallback to the initial
session key if the decryption/integrity check fails.

That original key is known to an attacker who has performed a classic
man-in-the-middle attack against the initial Diffie-Hellman key exchange.

For JDBC Thin client, this allows direct observation and manipulation
of the application level traffic, as both parties still use the
original keys.

Nevertheless, other clients, which implement the authentication key fold-in,
are still vulnerable. While the client expects a different session key
after authentication has completed, it can simply be dropped/ignored.
The server side of the connection at this point is already authenticated
and communication is still possible due to the key fallback. This grants
access to the database system as the original victim user.

This attack is successful in all known configurations, except if TLS
security is used.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Proof of Concept (PoC):

For protocol analysis and attacks, SySS built a proxy server implementing
the database protocol fundamentals and NNE. The proxy can perform a
man-in-the-middle attack against the Diffie-Hellman key exchange during
NNE negotiation. Then, the necessary translation and adjustment between
the client and server, which are now using different session keys, is
performed.

Launching the proxy and redirecting a client connection to it, the
man-in-the-middle attack is performed. The encrypted part of the further
protocol negotiation can be observed, including the authentication exchange.
Then, the client is dropped, and the proxy sends a predefined query to the
server.

The following log excerpt shows an OCI client (21.3) connecting as the
system user. The connection is hijacked and the system user table is
queried by the attack proxy.

  ./mitm.py --targethost 172.17.0.1 --mitmDH --hijackConnection
[...]
             |###[ Service ]###
             |  serviceId = encryption
             |  numParameters= 2
             |  unknown1  = 0
             |###[ EncryptionResp ]###
             |     version   = 12000000
             |     algo      = AES256
             |###[ Service ]###
             |  serviceId = integrity
             |  numParameters= 8
             |  unknown1  = 0
             |###[ IntegrityResp ]###
             |     version   = 12000000
             |     algo      = SHA256
             |     len1      = 0800
             |     len2      = 0800
```

```
            |     generator = [...]
            |     prime     = [...]
            |     public    = [...]
            |     rand      = 666F6F206261722062617A206261742071757578

[...]
DEBUG:root:Forward server -> client
DEBUG:root:Received encrypted payload [...]
[...]\x0cAUTH_USER_ID\x01\x00\x00\x00\x019\x00\x00\x00\x00\x0f\x00\x00\x00\x0fAUTH_SESSION_ID[...]
[authentication completed at this point]
INFO:root:Initially hijacking connection
[...]
DEBUG:root:Received encrypted payload [...]
INFO:root:###[ TTIMsg ]###
  TTCode    = 8
###[ RPA ]###
     outNbPairs= None
     \nbPairs   \
      |###[ KVPair ]###
      |  keyPtr    = None
      |  key       = b'\x00VOracle Database 18c Express Edition Release 18.0.0.0.0 - Production\nVersion 18.4.0.0'
[...]

INFO:root:b'[...]+select DISTINCT username FROM sys.all_users[...]
INFO:root:Send encrypted payload [...] len 368
[...]
DEBUG:root:Received encrypted payload [...]
INFO:root:###[ TTIMsg ]###
  TTCode    = 6
###[ Raw ]###
     load      = '[...]\x07\x06DBSNMP\x07\tAPPQOSSYS\x07\nGSMCATUSER\x07\x05GGSYS\x07\x03XDB[...]'

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Solution:

Update the Oracle Database servers and clients to the patched versions.
Enforce usage of a secured protocol version by setting the following options:

```
 SQLNET.ALLOW_WEAK_CRYPTO_CLIENTS=FALSE (server-side)
 SQLNET.ALLOW_WEAK_CRYPTO=FALSE (client-side)
```

Or use TLS-based transport security instead of Native Network Encryption.

More information:
https://www.oracle.com/security-alerts/cpujul2021.html
https://support.oracle.com/rs?type=doc&id=2791571.1 (customer account required)

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Disclosure Timeline:

2021-03-02: Vulnerability discovered
2021-03-17: Vulnerability reported to manufacturer
2021-07-20: Initial patch release by manufacturer,
2021-08-07: Final patches released by manufacturer
2021-12-10: Public disclosure of vulnerability

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

References:

[1] Product website for Oracle Database
    https://www.oracle.com/database/
[2] SySS Security Advisory SYSS-2021-061
    https://www.syss.de/fileadmin/dokumente/Publikationen/Advisories/SYSS-2021-061.txt
[3] SySS Responsible Disclosure Policy
    https://www.syss.de/en/responsible-disclosure-policy
[4] Paper "Oracle Native Network Encryption"
    https://www.syss.de/fileadmin/dokumente/Publikationen/2021/2021_Oracle_NNE.pdf

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Credits:

This security vulnerability was found by Moritz Bechler of SySS GmbH.

E-Mail: moritz.bechler () syss de
Public Key: https://www.syss.de/fileadmin/dokumente/PGPKeys/Moritz_Bechler.asc
Key ID: 0x768EFE2BB3E53DDA
Key Fingerprint: 2C8F F101 9D77 BDE6 465E CCC2 768E FE2B B3E5 3DDA

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```



```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```


Attachment: OpenPGP signature
Description: OpenPGP digital signature

By Date By Thread

Current thread:

- **[SYSS-2021-061] Oracle Database - NNE Connection Hijacking** *Moritz Bechler (Dec 10)*