pentest
security assurance

Identifying Oracle database installations during a network scan

By Mark Rowe

Introduction

One of the first phases of a network penetration test or vulnerability assessment is typically to perform a network port scan to identify all the active services. There are a large number of port scanning tools and techniques available to do this and it is not the purpose of this article to discuss them. This article focuses on some of the tools and techniques that can be used to determine whether there is an Oracle TNS listener active on any of the identified listening ports.

[The Oracle Transparent Network Substrate (TNS) protocol defines the communication protocol used between the database server and the database client applications. The TNS protocol provides an application interface to all industry-standard networking protocols e.g. TCP/IP. An example of a TNS enabled application is SQL*Net.]

The Problem

So let's assume we have used *nmap* (in my opinion one the best port scanning tools) by Fyoder to perform a scan of our example network. The results are shown below:

```
[prompt]$nmap -p 1-65535 192.168.1.1-254

Starting nmap V. 2.53 by fyodor@insecure.org ( www.insecure.org/nmap/
)
Interesting ports on  (192.168.1.2):
(The 65519 ports scanned but not shown below are in state: closed)
Port        State       Service
21/tcp      open        ftp
23/tcp      open        telnet
25/tcp      open        smtp
79/tcp      open        finger
98/tcp      open        linuxconf
111/tcp     open        sunrpc
113/tcp     open        auth
513/tcp     open        login
514/tcp     open        shell
515/tcp     open        printer
1006/tcp    open        unknown
1024/tcp    open        kdm
1029/tcp    open        unknown
1030/tcp    open        iad1
1526/tcp    open        pdap-np
2481/tcp    open        unknown

Interesting ports on  (192.168.1.100):
(The 65530 ports scanned but not shown below are in state: closed)
```

```
Port        State       Service
135/tcp     open        loc-srv
139/tcp     open        netbios-ssn
1025/tcp    open        listen
1028/tcp    open        unknown
1521/tcp    open        ncube-lm

Nmap run completed -- 254 IP addresses (2 hosts up) scanned in 79
seconds
```

As we can see we scanned the full port range of 1-65535 and two hosts have been identified with a number of open ports on each. Here we see that 192.168.1.100 has port 1521/tcp open, which the default configuration of Nmap reports as being "ncube-lm". However, we know this is also the default port of the TNS listener, which provides the TNS interface to the network. Unfortunately establishing a simple TCP connection using *netcat* written by Hobbit does not reveal anything useful about the service listening on this port.

Therefore we cannot be certain that it is a TNS listener. Of course at this point we could connect with the Oracle SQL plus client, if we have it to verify. Unfortunately, it is possible that the DBA may have changed the default port to something other than 1521. Doing this for every open port is a manual process, which would be time consuming for anything other than a small number of ports. Okay, we could automate this with a scripting language, but SQL plus is quite large and requires a number of configuration files. For example, it is not the sort of thing you want to download and install on a compromised IIS web server.

The Solution

Luckily help is at hand from several tools that are freely available on the Internet. The first we will discuss is a perl script *tnscmd.pl* by jwa@jammed.com. This tool implements several of the TNS protocol commands allowing the TNS listener to be queried. And as it is written in Perl it is fairly portable. So, if we want to query our server with port 1521 open we can do so with the following:

```
[prompt]$ tnscmd.pl -h 192.168.1.100 -p 1521
sending (CONNECT_DATA=(COMMAND=ping)) to 192.168.1.100:1521
writing 87 bytes
reading
.I......"..=(DESCRIPTION=(TMP=)(VSNNUM=135286784)(ERR=0)(ALIAS=LISTEN
ER))
```

Here we see the raw TNS reply packet from the TNS Listener ": .I......"..=(DESCRIPTION=etc.", which indicates that we are talking to a TNS Listener process.

It is worth noting that this tool also implements other TNS protocol commands that allow further information to be enumerated about the status of the listener and the databases to which it can connect. This and other interesting findings are discussed in detail in an excellent article by jwa@jammed.com, which can be found at http://www.jammed.com/~jwa/hacks/security/tnscmd/.

Another useful tool for TNS listener identification is *getsids* by Patrik Karlsson. This is a 'C' program which implements the TNS service command and is available in source and pre-compiled binary for Windows. Both of the above programs allow the IP and port to be specified. Unfortunately neither can easily be incorporated into a script that say reads from a file generated from a portscan as they both wait indefinitely for the correct response from an open port. However since both are supplied with the source this could be remedied.

A simpler technique is to use *netcat* to record the initial connection string from one SQL plus or one of the above tools to file. This file can then be piped to any open port. If we get back the correct response we know it is a TNS listener.

```
[prompt]$ nc -l -p 1521 >tns_ping.raw

[prompt]$ nc -w 3 192.168.1.100 1521 <tns_ping.raw
 I   ?    "
=(DESCRIPTION=(TMP=)(VSNNUM=135286784)(ERR=0)(ALIAS=LISTENER))
```

This can easily be scripted using your favourite scripting language and it is relatively small and portable so it can be easily transferred to compromised hosts. You can also imagine other variations on this theme to suit your requirements. Doing this against our example network reveals that 192.168.1.2 also has a TNS listener open on port 1526!

```
[prompt]$ tnsfind -f ps.out
192.168.1.2            1526   [found]
192.168.1.100          1521   [found]
```

Finally another recently released tool, which is worth a mention, is *Amap, v0.95* by DJ.Rev.Moon and vanHauser. Amap is designed to identify applications listening on network ports including applications that are non-ascii based. It does this by sending trigger packets, and looking up the responses from a supplied database of trigger and response strings. The current version does not have a trigger/response entry for the TNS protocol, however, this is easily added and we are sure future versions will have this.

Useful Tools

The homes of the tools discussed in this article are listed below:

- Nmap by Fyoder http://www.insecure.org/nmap/
- Netcat by Hobbit http://www.l0pht.com/research/tools/nc110.tgz
- Tnscmd by jwa@jammed.com http://www.jammed.com/~jwa/hacks/security/tnscmd/
- Getsids by Patrik Karlsson http://www.cqure.net/
- Amap by DJ.Rev.Moon and vanHauser http://www.thehackerschoice.com/

About Pentest