SECURITY NECROMANCY:
Further Adventures in Mainframe Hacking
• Wondered: Who’s researching this shit?
  – Windows 4,951
  – Mac OSX 2,270
  – z/OS (mainframe)
• Imagine if you will ... Your Doctor calls

IBM System z believes that the details of Security / Integrity APARs should not be made publically available.

With the critical workloads running on these systems, the impact of a vulnerability being exploited, however, could severely damage customer operations and business.

One of the benefits for not providing vulnerability details is that both external attackers and internal personnel threats can not get access to information that could put an enterprise at undue risk.

Source (ibm.com, DOC# ZSQ03054-USEN-03)
$100,000 Brick
So who are you?

• We’re just two cools dudes

• And we are gonna rock your fucking socks off
- Cut my teeth on AS400
- Love puzzles, breaking things
- Woke up in a panic – because of Mainframe!
- Realized I could get access to my own
  - Started Exploit dev research
  - Wrote first z/OS shellcode (check your CD)
- Dedicating life to helping poor, unfortunate corporations secure their shit
“in” to mainframes in the 90s courtesy of datapac (a Canadian x.25 network)

Security Consultant

Got my own mainframe
  - Realized its not as great as engineers have said (shocker!)

I’ve spoken domestically and internationally

Released multiple tools from password sniffing to user enumeration
WTF is a Mainframe?

Picture is worth a thousand words:
Reality

• Used by almost all fortune 100s
  – 90% according to IBM!
  – But seriously look at this:
Two Parts

• First Half: Networking
  – Network Job Entry
  – TN3270 protocol fun!

• Second Half: Exploit Development
  – How to write exploits
  – Program debugging
  – Shellcode development
  – First z/OS Shellcode
Butt first

You need a quick refresher on what this looks like this:
Networking
• Imagine a world where telnet still exists
• Imagine no MORE!
Userid: _ Time: 17:13:18 Date: 06/25/15
Password: Terminal: TCM0240 Model: 3278-2A
New Password: Node: UCHIMVS1 CPU: IBM 2096-R07
Transfer: System: zOS 1.13 - JES2/zOS 1.13
Account: System: NVi/TPX 5.3.0 - CA/ACF2 15.sp0

PF1=Help Enter your ACF2 Logonid and Password above, then press enter. To change your password, type it into the "New Password" field.
TN3270

- Imagine a world where telnet still exists
- Imagine no MORE!

- Basically like BBS’s back in the day
- Uses a ‘stream’
Field Attributes

• Screen is 1920 bytes long
• Each byte could be a field attribute identifying:
  ▪ Color
  ▪ Locked/Unlocked (Protected)
  ▪ Visible/Invisible (Hidden)
SAMPLE APPLICATION FORM

APPLICATION NO: ____________________________

Locked Field Length

READ DETAILED INSTRUCTIONS GIVEN SEPARATELY
BEFORE FILLING THE APPLICATION FORM.

NAME OF THE APPLICANT: ____________________________

FIRSTNAME MIDDLE LAST-NAME

DATE OF BIRTH: _____ / _____ / _____

RESIDENTIAL ADDRESS: ____________________________

________________________________________

EDUCATIONAL DETAILS

<table>
<thead>
<tr>
<th>QUALIFICATION</th>
<th>UNIVERSITY</th>
<th>YEAR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

USERPG01
Identifyin’

• No support in nmap/other tools
• Hard to identify screens
  – Without getting an emulator
• What about Hidden Fields?
• Or Protected Values?
Until NOW!
nmap -p 23 -sV 10.32.70.10

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2015-07-14 15:12 PDT
Nmap scan report for 10.32.70.10
Host is up (0.086s latency).
PORT    STATE SERVICE VERSION
23/tcp open  telnet IBM OS/390 or SNA telnetd

Service detection performed. Please report any incorrect results at https://nmap.org/submit/.
Nmap done: 1 IP address (1 host up) scanned in 1.29 seconds
nmap --script=tn3270-info 10.32.70.10 -p 23 -sV

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2015-07-14 15:24 PDT
Nmap scan report for 10.32.70.10
Host is up (0.18s latency).
PORT    STATE SERVICE VERSION
23/tcp open  tn3270 Telnet TN3270

Service detection performed. Please report any incorrect results at https://nmap.org/submit/.
Nmap done: 1 IP address (1 host up) scanned in 4.86 seconds

PORT    STATE SERVICE VERSION
23/tcp open  tn3270 Telnet TN3270
NEW!

- TN3270 Library for NMAP
- Emulates a ‘real’ 3270 screen
- Allows you to:
  - Connect
  - Show the screen
  - Send commands
  - Detect hidden fields!
<table>
<thead>
<tr>
<th>PORT</th>
<th>STATE SERVICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>3270/tcp</td>
<td>open tn3270</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>cics-enum:</th>
</tr>
</thead>
<tbody>
<tr>
<td>CICS Transaction:</td>
</tr>
<tr>
<td>CBAM: Valid - ID</td>
</tr>
<tr>
<td>CLDM: Valid - ID</td>
</tr>
<tr>
<td>CLER: Valid - ID</td>
</tr>
<tr>
<td>CIND: Valid - ID</td>
</tr>
<tr>
<td>CETR: Valid - ID</td>
</tr>
<tr>
<td>CIDP: Valid - ID</td>
</tr>
</tbody>
</table>
VTAM Application IDs

Nmap scan report for 10.32.70.10
Host is up (0.17s latency).
PORT   STATE SERVICE
23/tcp open  telnet
  | vtam-enum:
  |   VTAM Application ID:
  |       TSO: Valid - ID
  |_ Statistics: Performed 12 guesses

VTAM Macros

Nmap scan report for 10.32.70.10
Host is up (0.16s latency).
PORT   STATE SERVICE
23/tcp open  telnet
  | vtam-macro-enum:
  |   Login Macro:
  |       TSO: Valid - ID
  |       CICS: Valid - ID
  |_ Statistics: Performed 12 guesses
Hidden Fields!

Nmap scan report for (123.45.67.89)
Host is up (0.17s latency).
PORT STATE SERVICE
23/tcp open  tn3270
| tn3270-hidden:
| Hidden Field # 1: Type the number of your terminal:
| Column: 1
| Row   : 9
| Hidden Field # 2: SKRIV SYSTEMNAVN ===>
| Column: 40
| Row   : 9
But Wait!
There’s more!

I wrote one in LUA

why not Python?
tn3270lib

- Support tn3270 (not E)
- Creates a tn3270 object
- Allows for sending commands
- Blah blah blah same as nmap

BUT NOW IT MEANS I CAN INTRODUCE:
3 modes

• Proxy/Passthrough – MitM

• Mirror a targetted mainframe
  – Connects, scrapes the screen, then shares that screen on your machine
  – Takes commands you might expect your target to send and pregrabs those screens as well

• No args: TSO logon screen
[+] Starting SET'n'3270
[+] No target specified. Creating fake TSO screen on port 23
[+] Waiting for Incoming Connections on port 23
[+] Connection Received from: ('10.10.0.13', 51160)
[+] Line 1: dade
[+] Line 2: defcon23
[+] Displaying Dummy Screen
[+] Sleeping 15
SET 'n' 3270

• Supports SSL

Which is cool cause clients don’t check certs

(like, at all, no warning no nothing)
More Tools

• Big Iron Recon and Pwnage
  – By Dominic White!
  – [https://github.com/sensepost/birp](https://github.com/sensepost/birp)

• Mainframe Brute
  – Slower but prolly more reliable
  – [https://github.com/sensepost/mainframe_brute](https://github.com/sensepost/mainframe_brute)
Network
Job Entry
Jobs
(not steve)

- JCL (Job Control Language)
- Run by "JES"
- Made up of
  - STEPS
  - ProGraMs
  - etc
//COPYFILE JOB 'REPRO', NOTIFY=&SYSUID CLASS=A,
  
//**
COPY A FILE
  
//REPROE EXEC PGM=IDCAMS
  
//SYSUT1 DD DSN=ZEROKUL.JCL, ISP=SHR
  
//SYSUT2 DD DSN=THEWIZ.JCL
  
//SYSPRINT DD SYSOUT=*
  
//SYSIN DD *
  
   REPRO INFILE(SYSUT1) OUTFILE(SYSUT2)
  
/**
Network Job Entry

• Also known as NJE

• Runs on ports 175, 2252 (SSL)

• Developed in the 80s (??)
//TSoCMD JOB (TSO COMMAND),
//MSGLEVEL=(1,1)
/*XEQ ORLANDO
//STEP EXEC PGM=INJET01
//SYSTSIM DD */
/*
Initial Setup

• Systems configure JES telling them:
  – Where to connect
  – Who they will accept connections from

How?
NJEDEF

<table>
<thead>
<tr>
<th>Number of Nodes</th>
<th>Our Node Name</th>
<th>Other nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>NODE(1) NAME=NEWYORK</td>
<td>NODE(2) NAME=WASHDC</td>
<td></td>
</tr>
</tbody>
</table>

NETSRV(1) SOCKET=LOCAL
LINE(1) UNIT=TCPIP
SOCKET(WASHDC) NODE=2,

‘WASHDC’ IP Address = 10.10.0.210
Connect

Connect TCPIP

“OPEN”

“ACK”
Once established

• You can send JCL

• You can send NMR (command/control records)

• You are now a ‘trusted’ node
  – Depending on your security, of course
Interesting ‘feature’

• Users from one node don’t need to log on

• When a job is sent, the userID is sent along with the ‘NJE’ job

• So long as that account exists on the receiving side it will work.
NO Password

Note: no password or any authentication information is sent.

Nodes are TRUSTED and therefore no need to re-authenticate.
Breaking NJE

• First we need to find mainframes with NJE

• Problem: nmap
nmap --script=NJE-Test.nse 10.10.0.200 -p 175 -sV

Starting Nmap 6.47 SVN (http://nmap.org)
Nmap scan report for 10.10.0.200

175/tcp open  nje  z/OS Network Job Entry
175/tcp open  nje  z/OS Network Job Entry

Service detection performed. Please report any incorrect results.
Nmap done: 1 IP address (1 host up) scanned in 95.17 seconds
NJE Node Names

• You need this.
• No, you NEED it.
• You can’t connect otherwise

NMAP Script: NJE Node Brute
• Brute forces node names (even if the node is connected!)
Starting Nmap 6.47 SVN (http://nmap.org) at 2015-04-16 15:16 PDT

175/tcp open  nje   z/OS Network Job Entry
| nje-node-brute:
|   Node Name:
|     NEW YORK:<empty> - Valid credentials
|     Statistics: Performed 16 guesses in 9 seconds, average tps: 1

nje-node-brute:
| Node Name:
|   NEW YORK:<empty> - Valid credentials
|   Statistics: Performed 16 guesses in 9 seconds, average tps: 1
NJ E is super awesome

• Like we said before: You need three things:
  – Node Name of your target
  – Node name you want to pretend to be
  – IP Address of your target

With these you can inject JES2 commands with:
iNJEctor.py
You deserve a break today
Books from the Past!

• A lot of our research is from really old books

• Like, really old

• Older than some of you here today:
SVCs: Analysis for Integrity and Audit

Peter Goldis

Reprinted from “Technical Support”, May 1992
Fifth Edition (December 1988)
IBM
OS/VSE System Programming Library: Debugging Handbook

Volume 1


Includes Selectable Units:
- TSO/VTAM
- Scheduler/IOS Support
- Service Data Improvements
- MSS Enhancements
- 3838 Vector Processing Subsystem
- 3895 Device Support
- System Security Support
- Dumping Improvements
- Attached Processor Support
- MVS Processor Support
- Hardware Recovery Enhancements
- Interactive Problem Control System
- TSO/VTAM Level 2
- Data Management Support

Includes Program Product:
- MVS/System Extensions

IBM Corporation, Publications Development, Dept. D58, 
Bldg. 706-2, PO Box 390, Poughkeepsie, New York 12602
IBM System/360 Basic Programming Support

Basic Assembler Language

© 1964 by International Business Machines Corporation
Exploit Development
Architecture

• 23,31,64 bit modes
  – 3 sets of registers (16 ea)
  – Big Endian
  – Von Neumann Architecture
  – Stack-based (sorta)

• Virtual Address Spaces

• Program Status Word (PSW)

• Z/OS, USS, Z/Linux, Z/VM
DC in a box

AND MORE!
Where to start

• Focus on what you know
• Unix System Services
• Why? Cause C and Assembler
  – Narrowed down to:
    • Buffer Overflow POC
    • Format String Exploit POC
    • Learn testing environment
    • Shell code development and deployment
bash-4.2$ cat asm_example.s
* Buffer for SVC 23 has the following layout
* 0 - Length of reply buffer, otherwise 0
* 1 - Message len + 4 if inline; fixed if ptr
* 2 - MCS flag byte (00)
* 3 - 2nd MCS flag byte (00)
* 4-n - msg txt or ptr to data area
* see MVS Diagnosis reference.pdf for more info
PENUS START 0
ENTRY MAIN
MAIN
STM 14,12,12(13) Store all the registers R14 to
     R12 (all but r13) at R13+12
LR 12,15 R15 has base reg, store in R12
ST 13,92(,12) +92 is addr of first full word past pgm
     Store old SP there for later
LA 11,88(12) Load addr of data area past pgm
ST 11,8(13) Store this address in old SP area
LR 13,11 Put this addr in r13, new SP area
BRAS 1,*,+44 Jmp to SVC call, store addr in R1
     This effectively loads R1 with the address
     of all the args required for SVC 35
     Args for SVC35 (see above)
DC X'00'
DC X'27'
DC X'0000'
DC C'PENUS PENUS PENUS PENUS PENUS PENUS'
DC X'00'
SVC 35 Write to operator SVC call
BRC 15,*,+46 UNCOMMENT AT YOUR PERIL!!!
L 13,92(,12) Put the old SP back in R13
LM 14,12,12(13) Reload all the registers to their pre-
     Program values
LA 15,0 Set return code 0 in R15
BCR 15,14 Branch to return to calling pgm
DS 18F Stg area starts here
END MAIN End pgm
Useful example

- Execute local shell
  - Useful for Privilege Escalation

- Steps
  - Build working C or HLASM
  - Convert to machine code
  - Once working, “shellcode-ize”
    - Remove bad chars or encode
  - Test with C buffer stub program
Building Shellcode Demo
pop local shells DEMO
Pop bind shell demo
String Overflow DEMO
FMT String DEMO
What’s Next?

• MSF Integration?
• Native Exploits
• Java / Web exploits
• Privilege Escalation
• Continued Tool development / Porting
  – Generic shellcode building
  – Fuzzi
Thanks

- DEFCON for letting us talk about this
- IBM for this cool platform and online books
- Huge Mega Corps for neglecting this platform
- Dominic White for his tools
- Swedish underground community
- X3270 authors
Contact

- Phil - “Soldier of Fortran”
  @mainframed767
  mainframed767@gmail.com
  Soldieroffortran.org

- Chad – “Big Endian Smalls”
  @bigendiansmalls
  mainframe@bigendiansmalls.com
  Bigendiansmalls.com