

z/OS
3.2

Authorized Code Scanner Guide



Note

Before using this information and the product it supports, read the information in [“Notices” on page 105.](#)

This edition applies to IBM® z/OS® 3.2 (5655-ZOS) and to all subsequent releases and modifications until otherwise indicated in new editions.

Last updated: 2025-09-30

© **Copyright International Business Machines Corporation 2021, 2025.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

| | |
|---|-------------|
| Tables..... | V |
| Figures..... | vii |
| About this Document..... | xiii |
| Who should use this document..... | xiii |
| How to use this document..... | xiii |
| How to provide feedback to IBM..... | xv |
| Summary of Changes..... | xvii |
| Summary of changes for z/OS 3.2..... | xvii |
| Summary of changes for z/OS 3.1..... | xvii |
| General Content Changes..... | xvii |
| Message Changes..... | xviii |
| Chapter 1. Introduction..... | 1 |
| Chapter 2. Overview..... | 3 |
| Chapter 3. zACS configuration..... | 5 |
| Chapter 4. Running zACS..... | 25 |
| Setup..... | 25 |
| Running using the Command Line | 26 |
| Optional Parameters | 26 |
| Running using the Panel | 28 |
| Optional Parameters..... | 34 |
| Additional Panel Options..... | 40 |
| Running with Advanced Test..... | 42 |
| Filtering Run Services with Include Lists or Exclude Lists..... | 42 |
| Reading Generated Service Tables..... | 43 |
| Using the zACS z/OSMF External plug-in..... | 46 |
| Chapter 5. zACS Diagnosis..... | 63 |
| Interpreting the Potential Vulnerability Output File..... | 63 |
| Setting a SLIP Trap to Capture a Dump..... | 71 |
| Vulnerability examples and fixes..... | 71 |
| Chapter 6. zACM Configuration..... | 79 |
| Chapter 7. Running zACM..... | 81 |
| Chapter 8. zACM Diagnosis..... | 85 |
| Interpreting the zACM Potential Vulnerability Output File..... | 85 |
| zACM Vulnerability Examples and Fixes..... | 85 |

| | |
|---|------------|
| Chapter 9. System Codes..... | 87 |
| Chapter 10. Messages..... | 95 |
| Appendix A. SMF-Records..... | 101 |
| Record type 1154 Subtype 84 – zACM Compliance Evidence..... | 101 |
| Record type 1154 Subtype 84 header..... | 101 |
| Record type 1154 Subtype 84 Data section 1..... | 101 |
| Appendix B. Accessibility..... | 103 |
| Notices..... | 105 |
| Terms and conditions for product documentation..... | 106 |
| IBM Online Privacy Statement..... | 107 |
| Policy for unsupported hardware..... | 107 |
| Minimum supported hardware..... | 107 |
| Trademarks..... | 108 |
| Index..... | 109 |

Tables

1. Return and Reason codes for potential vulnerability report..... 87

2. Return and Reason codes for BPNGPCN, BPNGSVC, and BPNGMVSN..... 88

3. Return and Reason codes for BPNGUSSN and BPNKUSS.....88

4. Reason codes..... 89

5. Definition of zACS message numbers..... 95

6. Meaning of eighth character is message number.....95

7. Record type 1154 Subtype 84 Subtype Specific Data Header..... 101

8. Record type 1154 Subtype 84 Subtype Specific Data Section 1.....101

Figures

| | |
|--|----|
| 1. APF (Authorized Program Facility) load library command..... | 5 |
| 2. Define profile and grant group access to protect data sets..... | 6 |
| 3. Grant permission to LPA and the logrec data set..... | 6 |
| 4. Cylinder requirement for LOGREC data set..... | 8 |
| 5. Language Environment option needed for z/OS UNIX testing..... | 8 |
| 6. SYS1.BPN.SBPNSAMP(BPNZACS)..... | 9 |
| 7. SYS1.BPN.SBPNSAMP(BPNPCNUM)..... | 9 |
| 8. SYS1.BPN.SBPNSAMP(BPNSVCNM)..... | 10 |
| 9. SYS1.BPN.SBPNSAMP(BPNMVSNM)..... | 10 |
| 10. Optional parameter for generating the MVS table..... | 10 |
| 11. SYS1.BPN.SBPNSAMP(BPNUSSNM)..... | 11 |
| 12. IFAPRDxx parmlib member example..... | 11 |
| 13. Trial IFAPRDxx parmlib member example..... | 12 |
| 14. Configuration file attributes..... | 12 |
| 15. Custom Configuration file location..... | 12 |
| 16. userid.ZACS.FILTER.MODNAME..... | 15 |
| 17. userid.ZACS.FILTER.JOBNAME..... | 15 |
| 18. Example allocation and deallocation of data set..... | 15 |
| 19. userid.ZACS.OUTPUT..... | 15 |
| 20. Example allocation and deallocation of summary data set..... | 16 |
| 21. userid.ZACS.OUT.SUM..... | 16 |
| 22. userid.ZACS.SVCLOG..... | 16 |
| 23. userid.ZACS.PCLOG..... | 17 |

| | |
|---|----|
| 24. userid.ZACS.MVSLOG..... | 17 |
| 25. userid.ZACS.USSLOG..... | 17 |
| 26. Defining SAF profile for zACS GUI..... | 22 |
| 27. Permitting user access to ZMFAPLA..... | 22 |
| 28. ZMFAPLA refresh..... | 22 |
| 29. Opening Page | 23 |
| 30. GUI Setting Button..... | 23 |
| 31. zACS High Level Qualifier Field..... | 23 |
| 32. Define BPNZACS profile and refresh..... | 25 |
| 33. Starting tool, setting SLIP trap, and purging tool..... | 25 |
| 34. Generate PC table and SVC entries..... | 25 |
| 35. Testing with Command line Examples..... | 26 |
| 36. Example PC and SVC run..... | 27 |
| 37. zACS Panel..... | 28 |
| 38. Running all SVCs from the panel..... | 29 |
| 39. Running all PCs from the panel..... | 30 |
| 40. Running all MVS programs from the panel..... | 31 |
| 41. Running all UNIX programs from the panel..... | 32 |
| 42. Confirmation pop-up for All PCs run..... | 33 |
| 43. Confirmation pop-up for run already in progress..... | 34 |
| 44. Running a single SVC from the panel with routing number 109 and ESR 11..... | 36 |
| 45. Running PCs by module name from the panel with module name BPNTEST..... | 37 |
| 46. Opening the z/OS UNIX path popup..... | 38 |
| 47. The z/OS UNIX path popup..... | 39 |
| 48. The main panel with populated z/OS UNIX path field..... | 40 |

| | |
|---|----|
| 49. Advanced Test Option..... | 42 |
| 50. Example of excluded modules..... | 43 |
| 51. Example of included job names..... | 43 |
| 52. SVC Table Example..... | 43 |
| 53. PC Table example..... | 44 |
| 54. MVS Table Example..... | 45 |
| 55. z/OS UNIX Table Example..... | 45 |
| 56. GUI PC Table View..... | 46 |
| 57. GUI SVC Table View..... | 47 |
| 58. GUI z/OS UNIX Table View..... | 47 |
| 59. GUI MVS Table View..... | 48 |
| 60. GUI PC Empty Table View..... | 48 |
| 61. GUI PC Generation Finish View..... | 49 |
| 62. GUI Setting Icon..... | 49 |
| 63. GUI Setting Window..... | 50 |
| 64. Editing the configuration file from the GUI..... | 51 |
| 65. Closing the configuration file without saving..... | 52 |
| 66. GUI Setting Valid Data Set..... | 53 |
| 67. GUI Setting View Exclusion/Inclusion Data Set Window..... | 54 |
| 68. GUI Empty Test Results Page..... | 54 |
| 69. GUI Run Scan Options..... | 55 |
| 70. GUI Run ALL PC Scan..... | 55 |
| 71. GUI Run ALL SVC Scan..... | 56 |
| 72. GUI Run ALL z/OS UNIX Scan..... | 56 |
| 73. GUI Run ALL MVS Scan..... | 56 |

| | |
|---|----|
| 74. GUI Confirmation Prompt..... | 57 |
| 75. GUI Running a single PC module BPNTTEST..... | 57 |
| 76. GUI Running a single SVC with routing number 109 and ESR 11 | 58 |
| 77. GUI Running a z/OS UNIX file with path specified | 58 |
| 78. GUI Run Scan with Advanced Testing | 59 |
| 79. GUI Test Results Page Filled | 59 |
| 80. GUI Test Results Table Expanded rows for more details..... | 60 |
| 81. GUI Test Results filter by timestamps..... | 60 |
| 82. GUI Test Results Refresh Icon..... | 61 |
| 83. GUI View Raw Report Data Set Window..... | 61 |
| 84. GUI notification Icon..... | 62 |
| 85. GUI Notification Alert Window..... | 62 |
| 86. Potential Vulnerability Output Example 0C4..... | 63 |
| 87. Potential Vulnerability Output Example 0C1..... | 65 |
| 88. Potential Vulnerability Output Example 0E0..... | 66 |
| 89. Storage overlay output..... | 68 |
| 90. Privilege Escalation Output..... | 69 |
| 91. Basic Test Summary Box Output..... | 69 |
| 92. Advanced Test Summary Box Output..... | 70 |
| 93. PVTMOD Example..... | 71 |
| 94. LPAMOD Example..... | 71 |
| 95. Fetch Vulnerability example output..... | 72 |
| 96. Fetch Vulnerability code example..... | 72 |
| 97. Fetch Vulnerability alternative implementation..... | 73 |
| 98. Store Vulnerability example output..... | 74 |

| | |
|--|----|
| 99. Store Vulnerability code example..... | 74 |
| 100. Store Vulnerability alternative implementation..... | 75 |
| 101. Indirect Parameter and Storage Overlay Vulnerability Example Output | 76 |
| 102. Storage overlay code example..... | 76 |
| 103. Storage Overlay alternative implementation..... | 77 |
| 104. zACM Full Output Allocation..... | 79 |
| 105. zACM Summary Output Allocation..... | 79 |
| 106. zACM Filter Allocation..... | 79 |
| 107. Define BPNZACM profile and refresh..... | 81 |
| 108. Starting zACM, setting SLIP trap, and purging zACM..... | 81 |
| 109. Exclude by job name list DD..... | 81 |
| 110. Include by job name list DD | 82 |
| 111. Exclude by module list DD | 82 |
| 112. Include by module list DD | 82 |
| 113. Example job name filter list for zACM..... | 82 |
| 114. Example module filter list for zACM..... | 83 |

About this Document

Who should use this document

This document is for general users of the IBM z/OS Authorized Code Scanner (zACS)

How to use this document

To use this document:

- Read Chapter 2, “[Overview](#),” on page 3. It tells you how the IBM z/OS Authorized Code Scanner (zACS) is designed to discover vulnerabilities within Program Calls (PCs) and Supervisor Calls (SVCs) so that they can be subsequently remedied.
- Choose whether to run the Integrity Scanning Tool with either the panels or the commands:
 - If you want to use panels, read “[Running using the Panel](#)” on page 28. This topic explains how to get help while using the panels.
 - If you want to use commands, “[Running using the Command Line](#)” on page 26.
- Read [Chapter 3, “zACS configuration,”](#) on page 5 as it contains step-by-step procedures for you to follow.

How to provide feedback to IBM

We welcome any feedback that you have, including comments on the clarity, accuracy, or completeness of the information. For more information, see [How to send feedback to IBM](#).

Summary of Changes

Summary of changes for z/OS 3.2

This information contains no technical changes for this release.

Summary of changes for z/OS 3.1

The following content is new, changed, or no longer included in z/OS 3.1.

General Content Changes

New

The following content is new.

June 2024 refresh

The following sections are updated to include new information for a z/OS Authorized Code Scanner (zACS) Beta Program. (APAR OA64491 (www.ibm.com/support/pages/apar/OA64491), applies to z/OS V2R4 and newer)

- “Configuration file attributes” on page 12 has several keywords that are updated.
- “Example configuration file” on page 18 is updated.
- “Install the zACS Graphical User Interface (GUI)” on page 21 includes information for setting the zACS installation high-level qualifier.
- “Optional Parameters ” on page 26, BPNKNMVS is updated.
- “Reading Generated Service Tables” on page 43, Figure 54 on page 45 is updated.
- “Using the zACS z/OSMF External plug-in” on page 46 includes a multitude of screenshot and content updates.
- “Interpreting the Potential Vulnerability Output File” on page 63 Potential Vulnerability Output Examples 0C4 and 0C1 are updated.
- “Fetch vulnerability example” on page 72 is updated.
- “Store vulnerability example” on page 74 is updated.
- “Indirect parameter and storage overlay vulnerability example” on page 75 is updated.
- “Started task (BPNZACM)” on page 79 is updated.
- “ABEND 2600 reason code analysis” on page 88 is updated to include new reason codes 3D, 3E, 3F, 40, 41, 42, 43, 44, 45, 46, 47, 48, 1100C, and 1100D.

February 2024 refresh

- The following sections are updated to include information for a z/OS Authorized Code Scanner Self-Service Trial. (APAR OA65696: 90 day self service trial for z/OS Authorized Code Scanner (www.ibm.com/support/pages/apar/OA65696), applies to z/OS V2R4 and newer)
 - “Data set protection and permissions” on page 5 is updated with a note.
 - “Registration for self-service free trial” on page 12 is updated to include instructions for how to register for the trial.
 - “Interpreting the Potential Vulnerability Output File” on page 63 example Figure 87 on page 65 is updated.
 - “ABEND 2600 reason code analysis” on page 88 is updated to include new reason codes 06, 0A, 26, 27, 37, 38, 39, 3A, 3B, 3C, 10039, 1003A, 1003B, and 1003C.

September 2023 release

- New GUI access to z/OS Authorized Code Scanner by using z/OSMF.
 - For more information about installation, see [“Install the zACS Graphical User Interface \(GUI\)”](#) on page 21.
 - For more information about running the product, see [“Using the zACS z/OSMF External plug-in”](#) on page 46.
 - For more information about the newly added messages, see [Chapter 10, “Messages,”](#) on page 95.

Changed

The following content is changed.

July 2025 refresh

- The following sections are updated in support of APAR OA67929 (www.ibm.com/support/pages/apar/OA67929), which applies to z/OS V2R4 and later.
 - A note is added to [“Registration for self-service free trial”](#) on page 12 with guidance on avoiding ABENDs.
 - The descriptions of reason codes 3B and 1003B are updated in [“ABEND 2600 reason code analysis”](#) on page 88

May 2025 refresh

- The following sections are updated in support of APAR OA66634 (www.ibm.com/support/pages/apar/OA66634), which applies to z/OS V2R4 and later.
 - [“Setup”](#) on page 25
 - [“Reading Generated Service Tables”](#) on page 43
 - [“ABEND 2600 reason code analysis”](#) on page 88 updates reason codes 40, 46, and 47.

October 2024 refresh

- Reason codes 0A, 26, 27, 37, 38, and 3F are updated in [“ABEND 2600 reason code analysis”](#) on page 88. (APAR OA66644 (www.ibm.com/support/pages/apar/OA66644), which applies to z/OS V2R4 and newer.)

January 2024 refresh

- Assembly language instruction LA R3,outputdata is changed to LA R3,sourcedata in the [“Store vulnerability example”](#) on page 74. (APAR OA65900: ERROR IN STORE VULNERABILITY EXAMPLE (www.ibm.com/support/pages/apar/OA65900), which applies to z/OS V2R4 and newer.)

Deleted

The following content was deleted.

February 2024 refresh

- [“ABEND 2600 reason code analysis”](#) on page 88 has reason code 21 removed.

Message Changes

New messages

BPNU070E
BPNU071E
BPNU072E

Changed messages

None.

No longer issued messages

None.

Chapter 1. Introduction

System integrity is the inability of any program not authorized by a mechanism under the installation's control to circumvent or disable store or fetch protection, access a resource protected by a Security Server/Manager, or obtain control in an authorized state; that is, in supervisor state, with a protection key less than eight (8), or Authorized Program Facility (APF) authorized.

Program Call (PC) and Supervisor Call (SVC) routines, as well as AC(1), setuid, and setgid programs, provide a variety of critical services to z/OS. Some are created and maintained by IBM or IBM business partners as part of core z/OS subsystems and middleware products, for example. Others are created by vendor applications for z/OS. Some are implemented by system programmers to provide specialized in-house functionality for a client's enterprise.

PCs and SVCs have the architectural capability, depending on how they are defined, to allow an unauthorized program to invoke them, yet execute in an authorized state. The implementation of these PC and SVC routines must, therefore, handle this critical boundary carefully to ensure the system integrity of z/OS. Untrusted parameters, for example, need to be safely copied with architected instructions such as MVCSK and MVCDK so that an unauthorized program cannot fetch or update storage where it could not otherwise do so. For more information, see the z/Architecture® Principles of Operation.

AC(1) programs linked with SETCODE AC(1) in an APF authorized library allow any user to invoke them as a job step in a batch job, yet execute in an authorized state. Similarly, z/OS UNIX binaries with the APF extended attributed and linked with SETCODE AC(1) allow any user with z/OS UNIX permissions to execute them, yet they run in an authorized state. z/OS UNIX binaries with the setuid or setgid attributes can also be executed by any user with z/OS UNIX permissions to execute them, yet execute using a different UID or GID, which could be a UID or GID with elevated privileges.

If a PC or SVC routine, AC(1) program, or program with the setuid or setgid flags set, is implemented incorrectly, a security vulnerability may be introduced that compromises the system integrity of z/OS. The IBM z/OS Authorized Code Scanner (zACS) provides the ability to test these routines on a given z/OS V2R4 or above instance to address this.

Chapter 2. Overview

The IBM z/OS Authorized Code Scanner (zACS) is designed to prevent unauthorized callers from being incorrectly granted an authorized state by detecting vulnerabilities within Program Calls (PCs) and Supervisor Calls (SVCs), as well as AC(1), setuid, and setgid programs. The tool consists of a started task, a set of REXX execs, and associated data sets. The started task runs authorized while the tests generated by the REXX execs do not. By design, the zACS generates many ABENDs, and due to its volatile nature is meant to be run in a development and test environment. Both the input and output data sets contain sensitive data, as the input determines the scope of the scan and the output may describe potential vulnerabilities found on z/OS, and their respective locations. Therefore, these data sets, along with the REXX execs that help generate them, must be properly protected by an external security manager product such as RACF®.

To address the need for a non-volatile tool for use on a production system, the z/OS Authorized Code Scanner includes a z/OS Authorized Code Monitor (zACM) which passively monitors for indications of potential vulnerabilities without disrupting the system. Unlike the scanner, the monitor may be used in both development and test, as well as production environments. The monitor should be used in tandem with the scanner which actively generates test for a test environment to uncover potential vulnerabilities. Like with the scanner, the monitor provides new, foundational cybersecurity analysis via patented algorithms. Also similarly to the scanner, the monitor's output data sets contain sensitive data and must be properly protected by an external security manager.

The challenge of finding programming errors that impact system integrity is that the code base largely functions as expected during normal operation and under general forms of testing. The PC and SVC routines, as well as AC(1), setuid, and setgid programs implement a critical boundary between unauthorized users and authorized code. Any z/OS instance contains hundreds, often thousands of these routines. They are built into applications, middleware, and the operating system itself. They are created by IBM, by the z/OS vendor community, and by clients' in-house applications tailored to the needs of their respective enterprises. zACS provides the ability to scan these routines. As with any scanning tool, there may be programmer errors that cannot yet be detected and conversely the tool may produce some false positives. Fundamentally, the zACS provides new, foundational cybersecurity analysis via patented algorithms that dynamically scan authorized code on z/OS, supporting the growing movement of DevSecOps.

Chapter 3. zACS configuration

This topic assumes a high level qualifier of 'SYS1.BPN.**'. We recommend that your system be up to date on service before running the tool. Additionally, the REXX executables used in this tool assume the user is signed onto the user id 'userid'.

Run the following command in order to add the load library to the dynamic authorized program facility (APF) list:

```
SETPROG APF,ADD,DSNAME=<load-auth-dsn>,VOLUME=<volume-of-load-auth-dsn>
```

Figure 1. APF (Authorized Program Facility) load library command

Note: This command does not persist through re-IPL. It is recommended to add this to your PARMLIB. <https://www.ibm.com/docs/en/developer-for-zos/latest?topic=changes-apf-authorization-in-progxx>

Additionally, 'SYS1.BPN.SBPNLPA' is required to be in LPALIB to start the z/OS Authorized Code Scanner.

Data set protection and permissions

zACS is a tool that analyzes the SVCs/PCs of the current execution environment of a z/OS image. Since it will be reporting on security characteristics, access to the libraries for this tool is best kept to a known and trusted set of users. The executable library should not be placed in the LINKLIST or LPA.

Note: The following instructions are specific to z/OS Security Server (RACF). If another ESM is used then refer to the applicable vendor documentation for equivalent security configuration.

Example:

The following example is described with z/OS Security Server (RACF) commands where enhanced generic naming is allowed in the DATASET and the FACILITY classes, and list-of-groups access checking is active.

Step 1: Protect the data sets containing the tool

The base tool consists of the following data sets by default:

```
SYS1.BPN.SBPNCFG  
SYS1.BPN.SBPNEXEC  
SYS1.BPN.SBPNLOAD  
SYS1.BPN.SBPNLPA  
SYS1.BPN.SBPNPNL  
SYS1.BPN.SBPNSAMP
```

Note: 'SYS1.BPN.SBPNLPA' must be added to LPALIB to utilize zACS as well as zACM.

1. Define a group of users who will be given access to run zACS and to examine its results.

```
ADDGROUP BPNGRP
```

2. Define profile to protect data sets

```
ADDSD 'SYS1.BPN.**' UACC(NONE)
```

3. Grant the group access to the profile

```
PERMIT 'SYS1.BPN.**' CLASS(DATASET) ID(BPNGRP) ACCESS(READ)
```

4. Refresh the in-storage profiles so that the changes you have made to the FACILITY class take

```
SETR GENERIC(DATASET) REFRESH
```

5. Add the necessary userids to the group

```
CONNECT userid GROUP(BPNGRP)
```

Step 2: Protect the tool's generated output and configuration.

As per the instructions found in the [“Configuration file attributes” on page 12](#) section, the configuration file is placed in the following sequential data set:

```
userid.ZACS.CONFIG
```

By default, the PC/SVC/MVS/z/OS UNIX tables are generated and placed in the respective data sets:

```
userid.ZACS.PCNUM
```

```
userid.ZACS.SVCNUM
```

```
userid.ZACS.MVSNAME
```

```
userid.ZACS.USSNAME
```

The location of the PC log, SVC log, MVS log, z/OS UNIX log, output data set, Module Filter list, and Job name Filter are configurable in the configuration file and started task, however, we strongly suggest the following naming convention:

```
userid.ZACS.PCLOG  
userid.ZACS.SVCLOG  
userid.ZACS.MVSLOG  
userid.ZACS.USSLOG  
userid.ZACS.OUTPUT  
userid.ZACS.OUT.SUM  
userid.ZACS.FILTER.MODNAME  
userid.ZACS.FILTER.JOBNAME
```

Protect the tool's generated output and configuration files and permit the users to have access to them. It is assumed all output and configuration files are under the high-level qualifier 'userid.ZACS'

1. Define profile to protect the data sets

```
ADDSD 'userid.ZACS.**' UACC(NONE)
```

2. Grant the group access to the profile

```
PERMIT 'userid.ZACS.**' CLASS(DATASET) ID(BPNGRP) ACCESS(UPDATE)
```

Note: If the set of users accessing the output from the tool is different from the users who can execute it, you can define a second group containing those users and permit that group to the profile.

Figure 2. Define profile and grant group access to protect data sets.

Step 3: Grant permissions to other resources if needed for your configuration

1. Permit the group update access to LPA

```
PE CSDVYLPA.ADD.BPN.** CL(FACILITY) ID(BPNGRP) ACC(UPDATE)
```

2. If using a data set for logrec and the clearlogrec option in the configuration file is set to yes, permit the group update access to the logrec data set

```
PE 'SYS1.LOGREC' ID(BPNGRP) ACC(UPDATE)
```

Figure 3. Grant permission to LPA and the logrec data set

Service authorization

The user of zACS must have READ authority to the resource BPN.RUN in the XFACILIT class. Below is an example where resource-name would be replaced with BPN.RUN.

Note: The following instructions are specific to z/OS Security Server (RACF). If another ESM is used then refer to the applicable vendor documentation for equivalent security configuration.

The following example is described with z/OS Security Server (RACF) commands where enhanced generic naming is allowed in the DATASET and the FACILITY classes, and list-of-groups access checking is active.

An administrator with the appropriate RACF authority would:

1. Activate the XFACILIT class and RACLIST it

```
SETR CLASSACT(XFACILIT) RACLIST(XFACILIT)
```

2. Define the resource to the XFACILIT class

```
RDEFINE XFACILIT resource-name UACC(NONE) AUDIT(ALL)
```

3. Permit the user or group read access to the resource

```
PERMIT resource-name CLASS(XFACILIT) ID(user or group ID) ACCESS(READ)
```

4. Refresh the in-storage profiles so that the changes you have made to the XFACILIT class take

```
SETR RACLIST(XFACILIT) REFRESH
```

LOGREC

Note: A logstream may be used instead of a LOGREC data set. If a logstream is used this section may be skipped.

If your system uses a LOGREC data set, it is recommended to use the ENF logrec setting when running zACS to avoid filling up the data set, and then to switch back to using the LOGREC data set upon completion. The SETLOGRC ENF and SETLOGRC DATASET=dsname commands can be used to make this switch

Note: If your current setting is IGNORE you are unable to switch to ENF, you need to switch to LOGSTREAM or DATASET first and the switch to ENF. zACS does not run properly if the IGNORE setting is used.

A minimum of 10 cylinders is recommended for your LOGREC data set to ensure complete results. To verify the number of cylinders, check the size of the data set in ISPF 3.4:

```

                                Data Set
Information
Command ===>
Data Set Name . . . . : SYS1.ZATB.LOGREC
More:      +

General Data
Management class . . : **None**
Storage class . . . : **None**
Volume serial . . . : SYSA00
Device type . . . . : 3390
Data class . . . . : **None**
Organization . . . : PSU
Record format . . . : U
Record length . . . : 0
Block size . . . . : 1944
1st extent cylinders: 10
Secondary cylinders : 0
Data set name type :

Current Allocation
Allocated cylinders : 10
Allocated extents . : 1

Current Utilization
Used cylinders . . : 10
Used extents . . . : 1

Dates
Creation date . . . : 2018/06/14
Referenced date . . : 2019/08/29
Expiration date . . : ***None***
```

F1=Help F2=Split F3=Exit F7=Backward F8=Forward F9=Swap
F12=Cancel

Figure 4. Cylinder requirement for LOGREC data set.

Note: The number of test iterations, and thus logrec entries, differs depending on the PC® or SVC being tested. While 10 cylinders are sufficient for most services, in some cases, services must have a larger LOGREC data set to complete testing. Such cases result in a Return Code C, Reason Code C09, followed by an INCOMPLETE summary status.

If you are not sure the name of your LOGREC data set, you can find it with the console command:

```
D LOGREC
```

For more information about initializing and switching to a different, presumably larger, LOGREC data set, see [Initializing the logrec data set](#) in *z/OS MVS Diagnosis: Tools and Service Aids* and [SETLOGRC command](#) in *z/OS MVS System Commands*.

Note: If running on a virtual machine, the SVC76 VM option must be set to 'VM' so that the program checks created by the tool are properly sent to the LOGREC data set: [SET SVC76](#)

.

JES

To assure the submitted test jobs complete before starting the next test, the system must be set to not run jobs of the same name simultaneously. To do this, set your initiators to DUPL_JOB=DELAY if using JES2, and DUPJOBNM=NO if using JES3. These are the default settings.

z/OS UNIX

Testing for z/OS UNIX programs requires the following additional configuration:

Detecting Vulnerabilities:

In order to detect vulnerabilities in z/OS UNIX programs the following Language Environment® option must be set:

```
CEEDOPT (TRAP=((ON,NOSPIE),OVR))
```

Figure 5. Language Environment option needed for z/OS UNIX testing

You may check the current CEEDOPT option using the DISPLAY CEE,CEEDOPT command.

CEE Dump Suppression:

z/OS UNIX vulnerability scanning may generate many CEE dumps in the file system. To suppress CEE dumps during a zACS scan, it is recommended that the TERMTHDACT Language Environment option be set to QUIET. Combining this with the TRAP option above, the syntax is:

```
CEEDOPT (TRAP=((ON,NOSPIE),OVR),TERMTHDACT=((QUIET),OVR))
```

Started task (BPNZACS)

Define a profile in the STARTED class to associate the user id with the zACS address space:

1. Authorize the started task

```
RDEFINE STARTED BPNZACS.** UACC(NONE) STDATA(USER(userid) GROUP(SYS1) TRUSTED(YES))
```

2. Refresh that STARTED class

```
SETR RACLIST(STARTED) REFRESH
```

The following JCL is contained in 'SYS1.BPN.SBPNSAMP(BPNZACS)' data set. This needs to be copied into the working PROCLIB. MYOUTDD is required and must point to the potential vulnerability data set, which is used for the output of the tool, while MYSUMDD is optional and points to the potential vulnerability summary data set where only unique hits will be recorded. When this task is started, the output data set and summary data set will be cleared of any data, including any vulnerability information from a previous time the task was running.

```
//BPNZACS PROC
//*****
//*
//* THIS EXECUTES THE BPNZACS PROGRAM FOR INTEGRITY TESTING.          *
//* PUT THIS JCL IN THE PROCLIB DATA SET USED FOR STARTED TASKS      *
//* AND MAKE THE CORRESPONDING RACF UPDATES, E.G.                     *
//*   RDEFINE STARTED BPNZACS.** UACC(NONE) STDATA(USER(user)          *
//*   GROUP(SYS1) TRUSTED(YES))                                         *
//*   SETR RACLIST(STARTED) REFRESH                                     *
//*                                                                     *
//* INSTRUCTIONS:                                                       *
//*   CHANGE loadlib-dataset TO THE LOAD LIB DATASET UNDER YOUR HLQ   *
//*   CHANGE output-dataset TO THE ALLOCATED DATASET SPECIFIED FOR     *
//*   OUTPUT                                                            *
//*   CHANGE summary-dataset TO THE ALLOCATED DATASET SPECIFIED FOR    *
//*   SUMMARY                                                           *
//*****
//GOSTEP EXEC PGM=BPNGMAIN,TIME=NOLIMIT
//STEPLIB DD DSN=loadlib-dataset,DISP=SHR
//MYOUTDD DD DSN=output-dataset,DISP=SHR
//MYSUMDD DD DSN=summary-dataset,DISP=SHR
```

Figure 6. SYS1.BPN.SBPNSAMP(BPNZACS)

BNPCNUM

The following JCL is contained in 'SYS1.BPN.SBPNSAMP(BNPCNUM)' data set. This should be copied to a separate JCL data set. The output of BNPCNUM goes to 'userid.ZACS.PCNUM' containing the list of PC numbers for the system.

Note: The SYSTEM= keyword may be added to the job card to specify the current system on which zACS will be run.

```
//ZACSJP JOB NOTIFY=&SYSUID,MSGCLASS=A,REGION=5M
//*****
//* GENERATES THE PC TABLE FOR INTEGRITY TESTING.                    *
//*                                                                     *
//* INSTRUCTIONS:                                                     *
//*   CHANGE loadlib-dataset TO THE LOAD LIB DATASET UNDER YOUR HLQ *
//*****
//DELETE EXEC PGM=IEFBR14
//DELDSD DD DISP=(MOD,DELETE),DSN=&SYSUID..ZACS.PCNUM,
//        SPACE=(TRK,1),UNIT=SYSDA
//
//CREATE EXEC PGM=BPNGPCN
//STEPLIB DD DSN=loadlib-dataset,DISP=SHR
//BNPCOUT DD DISP=(NEW,CATLG),DSN=&SYSUID..ZACS.PCNUM,
//        SPACE=(TRK,(10,10)),UNIT=SYSDA,
//        DCB=(LRECL=133,BLKSIZE=0,RECFM=FBA)
//SYSOUT DD SYSOUT=*
//STDOUT DD SYSOUT=*
//STDERR DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
```

Figure 7. SYS1.BPN.SBPNSAMP(BNPCNUM)

BPNSVCNM

The following JCL is contained in 'SYS1.BPN.SBPNSAMP(BPNSVCNM)' data set. This should be copied to a separate JCL data set. The output of BPNSVCNM goes to 'userid.ZACS.SVCNUM' containing the list of SVC numbers for the system.

```
//ZACSJS JOB NOTIFY=&SYSUID,MSGCLASS=A,REGION=5M
//*****
//* GENERATES THE SVC TABLE FOR INTEGRITY TESTING. *
//* * *
//* INSTRUCTIONS: *
//* CHANGE loadlib-dataset TO THE LOAD LIB DATASET UNDER YOUR HLQ *
//*****
//DELETE EXEC PGM=IEFBR14
//DELDN DD DISP=(MOD,DELETE),DSN=&SYSUID..ZACS.SVCNUM,
// SPACE=(TRK,1),UNIT=SYSDA
//*
//CREATE EXEC PGM=BPNGSVCN
//STEPLIB DD DSN=loadlib-dataset,DISP=SHR
//BPNSVOUT DD DISP=(NEW,CATLG),DSN=&SYSUID..ZACS.SVCNUM,
// SPACE=(TRK,(10,10)),UNIT=SYSDA,
// DCB=(LRECL=133,BLKSIZE=0,RECFM=FBA)
//SYSOUT DD SYSOUT=*
//STDOUT DD SYSOUT=*
//STDERR DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
```

Figure 8. SYS1.BPN.SBPNSAMP(BPNSVCNM)

BPNMVSNM

The following JCL is contained in 'SYS1.BPN.SBPNSAMP(BPNMVSNM)' data set. This should be copied to a separate JCL data set. The output of BPNMVSNM goes to 'userid.ZACS.MVSNAME' containing the list of AC(1) authorized data sets for the system.

Note: The SYSTEM= keyword may be added to the job card to specify the current system on which zACS will be run.

```
//ZACSM JOB NOTIFY=&SYSUID,MSGCLASS=A,REGION=5M
//*****
//* * *
//* GENERATES THE MVS TABLE FOR INTEGRITY TESTING. *
//* * *
//* INSTRUCTIONS: *
//* CHANGE loadlib-dataset TO THE LOAD LIB DATASET UNDER YOUR HLQ *
//*****
//DELETE EXEC PGM=IEFBR14
//DELDN DD DISP=(MOD,DELETE),DSN=&SYSUID..ZACS.MVSNAME,
// SPACE=(TRK,1),UNIT=SYSDA
//*
//CREATE EXEC PGM=BPNGMVSN
//STEPLIB DD DSN=loadlib-dataset,DISP=SHR
//BPNMVOUT DD DISP=(NEW,CATLG),DSN=&SYSUID..ZACS.MVSNAME,
// SPACE=(TRK,(10,10)),UNIT=SYSDA,
// DCB=(LRECL=133,BLKSIZE=0,RECFM=FBA)
//SYSOUT DD SYSOUT=*
//STDOUT DD SYSOUT=*
//STDERR DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
```

Figure 9. SYS1.BPN.SBPNSAMP(BPNMVSNM)

Due to the large number of data sets on a typical system, the full generated MVS table may be quite large. An additional parameter may be added to the job if you wish to generate a table for only a specific dataset. The form of the parameter is as follows:

```
PARM=dsn,[[volume],[unittype]]
```

Figure 10. Optional parameter for generating the MVS table

This parameter should be specified on the EXEC card for execution of BPNGMVSN when the desired output is to be the list of members linked AC(1) within a specific data set.

dsn is the data set name

volume is the volume on which the data set resides. This is only required when dsn is not catalogued. If volume is not specified, the volume will be set to *SMS* in the output regardless of whether or not it is an SMS managed data set.

unittype is the unit type of the volume. If not specified it will default to 3390.

Note: If the dataset is not on the specified volume, the unit type is incorrect or allocation of the data set fails for any other reason, the member will be set to "*MISSING" in the output.

BPNUSSNM

The following JCL is contained in 'SYS1.BPN.SBPNSAMP(BPNUSSNM)' data set. This should be copied to a separate JCL data set. The output of BPNUSSNM goes to 'userid.ZACS.USSNAME' containing the list of AC(1) authorized z/OS UNIX programs for the system, as well as the z/OS UNIX programs with the setuid or setgid flags on.

Note: The SYSTEM= keyword may be added to the job card to specify the current system on which zACS will be run.

```
//ZACJSU JOB NOTIFY=&SYSUID,MSGCLASS=A,REGION=5M
//*****
//*
//* GENERATES THE USS TABLE FOR INTEGRITY TESTING.
//*
//* INSTRUCTIONS:
//*   CHANGE loadlib-dataset TO THE LOAD LIB DATASET UNDER YOUR HLQ
//*   CHANGE uss-work-dir TO THE FULL PATH OF A DIRECTORY IN THE FILE
//*   SYSTEM THAT MAY BE USED A WORK DIRECTORY. THE DIRECTORY MUST
//*   EXIST. THIS VALUE MUST MATCH THE ussWorkDir KEYWORD IN THE
//*   CONFIGURATION FILE.
//*   OPTIONALLY CHANGE include-dirs TO THE FULL PATH OF A LIST OF
//*   DIRECTORIES TO SEARCH.
//*   IF NOT SPECIFIED, THE DEFAULT SEARCH INCLUDES /bin, /sbin,
//*   and /usr/lpp.
//*   IF MORE THAN ONE DIRECTORY IS SPECIFIED, THEY MUST BE
//*   SEPARATED BY A SPACE.
//*   IF NOT USING THIS OPTION, REMOVE ,include-dirs.
//*****
//DELETE EXEC PGM=IEFBR14
//DELDN DD DISP=(MOD,DELETE),DSN=&SYSUID..ZACS.USSNAME,
//      SPACE=(TRK,1),UNIT=SYSDA
//*
//CREATE EXEC PGM=BPNGUSSN,PARMDD=PARMIN
//PARMIN DD DATA
uss-work-dir,include-dirs
//*
//STEPLIB DD DSN=loadlib-dataset,DISP=SHR
//BPNUSOUT DD DISP=(NEW,CATLG),DSN=&SYSUID..ZACS.USSNAME,
//          SPACE=(TRK,(10,10)),UNIT=SYSDA,
//          DCB=(LRECL=1030,BLKSIZE=0,RECFM=FBA)
//SYSOUT DD SYSOUT=*
//STDOUT DD SYSOUT=*
//STDERR DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//*
```

Figure 11. SYS1.BPN.SBPNSAMP(BPNUSSNM)

Registration

Update your IFAPRDxx parmlib member with the following text:

```
PRODUCT OWNER('IBM CORP')
NAME('z/OS')
ID(5655-ZOS)
VERSION(*) RELEASE(*) MOD(*)
FEATURENAME('ZACS')
STATE(ENABLED)
```

Figure 12. IFAPRDxx parmlib member example

You can update your active IFAPRDxx parmlib member with the SET PROD=xx console command.

Registration for self-service free trial

To enable the 90-day self-service trial, update your IFAPRDxx parmlib member with the following text:

```
PRODUCT OWNER('IBM CORP')
NAME('z/OS')
ID(5655-ZOS)
VERSION(*) RELEASE(*) MOD(*)
FEATURENAME('ZACS_TRIAL')
STATE(ENABLED)
```

Figure 13. Trial IFAPRDxx parmlib member example

Starting the Trial

You have 90 days from the day the trial begins to use and evaluate the function. The trial begins upon starting zACS or zACM after the ZACS_TRIAL feature is enabled. The trial extends to the entire sysplex.

Important: The directory /global/zacs must be created and read/write permissions given to both zACS and zACM prior to using the 90-day trial license. Failure to do so may result in an ABEND=S000 U2600 REASON=0000003B for the scanner or an ABEND=S000 U2600 REASON=0001003B for the monitor.

Ending the Trial

No disablement is required after the trial ends. zACS no longer runs after the trial expires, unless the full feature is enabled. The trial cannot be paused. The original start time is used to calculate the 90-day interval. If you mistakenly remove or disable the **IFAPRDxx** statement during the trial, the 90-day duration continues. Replace or enable the **IFAPRDxx** statement as soon as possible to continue the trial.

Configuration file attributes

The zACS sample configuration file can be found in 'SYS1.BPN.SBPNCFG'. Copy this file into a sequential data set named 'userid.ZACS.CONFIG' for runtime use. The suggested attributes are as follows:

```
lrec: 80
block size: 0
primary: 1 track
secondary: 0 track
recfm: FB
dsorg: PS
```

Figure 14. Configuration file attributes

Note: Files with *lrec* greater than 80 will still be constrained to 80 characters.

A custom configuration file location can be used by specifying the DD ZACSCONF on the started task. This location is only retrievable when zACS is running. If zACS is not running the default location *userid.ZACS.CONFIG* will be used by the panel and the GUI. The default location will also be used if ZACSCONF is not specified.

Specify ZACSCONF on started task as follows:

```
//ZACSCONF DD DSN=config.location,DISP=SHR
```

Figure 15. Custom Configuration file location

The configuration file contains several keywords that will need to be set before running. The symbol *&userid.* can be used and will be substituted with the current userid. System defined symbols may also be used. Single quotes within the keyword's value may only be used in the keywords *jobAccount* and *jobProgrammer* and must be escaped by doubling them. Blank spaces within a keyword's value may only

be used in the keyword *jobProgrammer*. A keyword's value may take up multiple lines, this can be done by ending the line you wish to continue with a comma. Example:

```
jobOtherKeywords = 'NOTIFY=ZACSUSER, ',  
                  'PRTY=12'
```

Will result in the value NOTIFY=ZACSUSER,PRTY=12.

The keywords are as follows:

symbolSubstitutionSupport

The rule set for system symbol substitution. 'DEFAULT' allows the use of '&userid' and '&sysname'. 'EXTENDED' allows the use of '&userid.' as well as any system symbol. When using 'EXTENDED', symbols must end in a period (.) unless they are at the end of the value or when followed immediately by another symbol. 'DEFAULT' will not treat periods as part of the symbol. If this keyword is omitted, 'DEFAULT' settings are used. This setting only applies to keywords located below it, and thus should be placed at the top of the configuration file. Any keywords placed above it will use the 'DEFAULT' rule set.

autoslip

The on/off switch on whether to automatically set the sample SLIP for a found potential vulnerability.

Note: Only one PER SLIP may be set at a time, if one is already enabled, following PER SLIPs will not be set. Additionally, if a generated PER SLIP matches on a zACS hit, zACS output may be suppressed for that occurrence.

slipIDPrefix

The one character prefix used to create the sample SLIP's ID. This prefix will be followed by a three character hexadecimal number to create the full SLIP ID for each sample SLIP trap.

clearLogrec

zACS creates many program checks that can quickly fill up the system's logrec data set. zACS will not be able to properly execute its testing if the logrec data set becomes full. When 'YES' is specified, a job step to clear the contents of the logrec data set will be run with each service tested to avoid filling the logrec data set. Note that the logrec data set will be cleared completely, including information from other programs using it. If a logstream is used instead of a logrec data set, then this keyword is ignored.

logrecDSname

The name of the logrec data set that will be cleared by the clear logrec job step. If using a logstream instead of a logrec data set, specify 'LOGSTREAM'.

testcaseJobName

The JCL job name to be used for all test jobs.

jobMsgClass

The job message class for the test JCL jobs.

jobRegion

Sets the job REGION parameter for the test JCL jobs. Valid units are 'M' and 'K'.

jobTimeout

Sets the job TIME parameter for the test JCL jobs when running a basic test. This limit is specified in minutes.

jobAdvTimeout

Sets the job TIME parameter for the test JCL jobs when running and advanced test. This limit is specified in minutes.

jobSystem

Sets the job SYSTEM parameter for the test JCL jobs.

jobSchenv

Sets the job SCHENV parameter for the test JCL jobs.

printLogOutput

The display option for the list of PC or SVC numbers submitted during an zACS run. If 'SCREEN' is specified, the display is sent to the screen. If 'LOG' is specified, this display is redirected to the

respective data sets specified by *pcLogDSname*, *svcLogDSname*, *mvsLogDSname*, and *ussLogDSname*. If 'BOTH' is specified the display will go to both the screen and the log data sets.

Note: These data sets will be overwritten the next time the tool is run.

jclDSname

The dataset containing BPNPCNUM and BPNSVCNM. Used to submit BPNPCNUM and BPNSVCNM from the panel.

pcTableDSname

The dataset containing the PC table, generated from BPNPCNUM. If omitted, the default is *<userid>.ZACS.PCNUM*.

svcTableDSname

The dataset containing the SVC table, generated from BPNSVCNM. If omitted, the default is *<userid>.ZACS.SVCNUM*.

mvsTableDSname

The dataset containing the MVS table, generated from BPNMVSNM. If omitted, the default is *<userid>.ZACS.MVSNAME*.

ussTableDSname

The dataset containing the z/OS UNIXtable, generated from BPNUSSNM. If omitted, the default is *<userid>.ZACS.USSNAME*.

ussWorkDir

The work directory in the file system to be used for z/OS UNIX testing. It is strongly recommended that this directory is only used by ZACS.

useJobAccount

The yes/no switch on whether to use the account information parameter on the JCL job card.

jobAccount

Sets the positional account information parameter for the test JCL jobs. If single quotes are used, they must be escaped by doubling them. Example: to assign the account information D548-8686,'12/8/85',PGMBIN use `jobAccount = 'D548-8686','12/8/85',PGMBIN'`

useJobProgrammer

The yes/no switch on whether to use the programmer name parameter on the JCL job card.

jobProgrammer

Sets the positional programmer name parameter for the test JCL jobs. If single quotes are used, they must be escaped by doubling them. Example: to assign the name Dave O'dell use `jobProgrammer = 'DAVE O'DELL'`

useOtherKeywords

The yes/no switch on whether to use any other keyword parameters on the JCL job card not already specified.

jobOtherKeywords

Sets the any other keyword parameters desired for the test JCL jobs. Full keyword parameter syntax must be used, with multiple keywords being separated by quotes. For example, `jobOtherKeywords = 'NOTIFY=ZACSUSER,PTY=12'`.

addCustomDD

The yes/no switch on whether to add a custom DD line in the JCL to submit testcases.

CustomDDJCL

A line to add a custom DD statement to the JCL to submit testcases.

addJobParm

The yes/no switch on whether to add a JobParm statement in the JCL to submit testcases.

JobParmJCL

A line to add a JobParm statement to the JCL to submit testcases.

includeOrExclude

The switch between an inclusion filter and an exclusion filter. If 'INCLUDE' is specified, only services found in the *jobFilterDSname* and *modFilterDSname* will be run if *filterByJobName* and

filterbyModName are specified as 'YES', respectively. If 'EXCLUDE' is specified, services found in the jobFilterDSname and modFilterDSname will be excluded if filterByJobName and filterbyModName are specified as 'YES', respectively.

filterbyModName

The yes/no switch on whether to use the include/exclude by module name list.

modFilterDSname

The sequential data set to contain a list of module names to be included or excluded from a full run. Applies to both PC and SVC runs. The suggested attributes are as follows:

```
lrec: 80
block size: 0
primary: 5 tracks
secondary: 1 track
recfm: FB
dsorg: PS
```

Figure 16. userid.ZACS.FILTER.MODNAME

filterByJobName

The yes/no switch on whether to use the include/exclude by job name list.

jobFilterDSname

The sequential data set to contain a list of job names to be included or excluded from a full run of the tool. Only applies to PC run. The suggested attributes are as follows:

```
lrec: 80
block size: 0
primary: 5 tracks
secondary: 1 track
recfm: FB
dsorg: PS
```

Figure 17. userid.ZACS.FILTER.JOBNAME

stOutDSname

The sequential data set to contain the found potential vulnerabilities. It must match the potential vulnerability data set specified by MYOUTDD in the started task. Data set must exist.

Example allocation:

```
alloc new da(<output-dataset>) dd(<myoutdd>) dsorg(ps) space(300,300) tracks lrecl(80)
blksize(0) recfm(f,b)
```

```
free dd(<myoutdd>)
```

Figure 18. Example allocation and deallocation of data set.

The suggested attributes are as follows:

```
lrec: 80
block size: 0
primary: 300 tracks
secondary: 300 tracks
recfm: FB
dsorg: PS
```

Figure 19. userid.ZACS.OUTPUT

printStats

The switch to determine if TEST IN PROGRESS and TEST DONE messages are to be printed to the potential vulnerability data set. If ALL is specified, these messages will be printed to the potential vulnerability data set, always. If ERROR is specified, TEST IN PROGRESS messages will not be displayed, and TEST DONE messages will be displayed only if a non-zero return code is detected. This option allows the user to suppress output from successful test cases. The default is ALL.

printSummary

The switch to determine if summary information is to be displayed at the end of each services tests in the potential vulnerability data set. If ALL is specified, summary information is printed after testing completes for each service. If ERROR is specified, summary information is only printed after testing completes for a service if a non-zero return code was detected. The default is ALL.

sumOutDSname

The sequential data set to contain the summary of the found potential vulnerabilities. It must match the summary data set specified by MYSUMDD in the started task. Data set must exist. Example allocation:

```
alloc new da(<summary-dataset>) dd(<mysumdd>) dsorg(ps) space(300,300) tracks lrecl(80)
blksize(0) recfm(f,b)
free dd(<mysumdd>)
```

Figure 20. Example allocation and deallocation of summary data set.

The suggested attributes are as follows:

```
lrec: 80
block size: 0
primary: 300 tracks
secondary: 300 tracks
recfm: FB
dsorg: PS
```

Figure 21. userid.ZACS.OUT.SUM

svcLogDSname

The sequential data set to contain the log of the most recent SVC run. If the data set does not exist it will be created using the attributes specified by svcLogDSlrec, svcLogDSblksize, svcLogDSPriSp, svcLogDSsecSp. The suggested attributes are as follows:

```
lrec: 80
block size: 0
primary: 5 tracks
secondary: 1 track
recfm: FB
dsorg: PS
```

Figure 22. userid.ZACS.SVCLOG

svcLogDSlrec

The record length to be used in allocating the svcLogDSname data set if it does not already exist.

svcLogDSblksize

The block size to be used in allocating the svcLogDSname data set if it does not already exist.

svcLogDSPriSp

The primary space quantity to be used in allocating the svcLogDSname data set if it does not already exist.

svcLogDSsecSp

The secondary space quantity to be used in allocating the svcLogDSname data set if it does not already exist.

pcLogDSname

The sequential data set to contain the log of the most recent PC run. If the data set does not exist it will be created using the attributes specified by pcLogDSlrec, pcLogDSblksize, pcLogDSPriSp, pcLogDSsecSp. The suggested attributes are as follows:

```
lrec: 80
block size: 0
primary: 100 tracks
secondary: 40 tracks
recfm: FB
dsorg: PS
```

Figure 23. *userid.ZACS.PCLOG*

pcLogDSLrec

The record length to be used in allocating the pcLogDSname data set if it does not already exist.

pcLogDSblksz

The block size to be used in allocating the pcLogDSname data set if it does not already exist.

pcLogDSPriSp

The primary space quantity to be used in allocating the pcLogDSname data set if it does not already exist.

pcLogDSsecSp

The secondary space quantity to be used in allocating the pcLogDSname data set if it does not already exist.

mvsLogDSname

The sequential data set to contain the log of the most recent MVS run. If the data set does not exist it will be created using the attributes specified by mvsLogDSLrec, mvsLogDSblksz, mvsLogDSPriSp, mvsLogDSsecSp. The suggested attributes are as follows:

```
lrec: 80
block size: 0
primary: 100 tracks
secondary: 40 tracks
recfm: FB
dsorg: PS
```

Figure 24. *userid.ZACS.MVSLOG*

mvsLogDSLrec

The record length to be used in allocating the mvsLogDSname data set if it does not already exist.

mvsLogDSblksz

The block size to be used in allocating the mvsLogDSname data set if it does not already exist.

mvsLogDSPriSp

The primary space quantity to be used in allocating the mvsLogDSname data set if it does not already exist.

mvsLogDSsecSp

The secondary space quantity to be used in allocating the mvsLogDSname data set if it does not already exist.

ussLogDSname

The sequential data set to contain the log of the most recent z/OS UNIX run. If the data set does not exist it will be created using the attributes specified by ussLogDSLrec, ussLogDSblksz, ussLogDSPriSp, ussLogDSsecSp. The suggested attributes are as follows:

```
lrec: 2052
block size: 0
primary: 100 tracks
secondary: 40 tracks
recfm: FB
dsorg: PS
```

Figure 25. *userid.ZACS.USSLOG*

ussLogDSLrec

The record length to be used in allocating the ussLogDSname data set if it does not already exist.

ussLogDSblksz

The block size to be used in allocating the ussLogDSname data set if it does not already exist.

ussLogDSPriSp

The primary space quantity to be used in allocating the ussLogDSname data set if it does not already exist.

ussLogDSsecSp

The secondary space quantity to be used in allocating the ussLogDSname data set if it does not already exist.

Example configuration file

The following is an example configuration file, information on changing the configuration file attributes can be found [“Configuration file attributes”](#) on page 12.

Note: Full line comments proceeded by '/' and ending with '*' will be ignored.

```

/*****
/*****
/**
/** zACS Configuration file
/**
/**
/*****
/*****

/*
/* Parsing settings
/*
/* WARNING: Any setting placed above this setting will use their
/* default settings
/*
/* Instructions:
/* Set the rules for config file parsing
/* DEFAULT allows for the user of '&userid' and '&sysname' for
/* symbol substitution. EXTENDED allows for use of all system
/* symbols, the symbols must be delimited using an ending period,
/* unless followed by a second symbol when used as the final part
/* of a string. Ex:
/* '&sysname&sysplex..tempdsn.&lpar'
/* When switching from DEFAULT to EXTENDED, existing symbols must be
/* updated to match the extended syntax. EX:
/* '&userid.ZACS.PCNUM' becomes '&userid..ZACS.PCNUM'
/* To enable the extended symbol functionality, remove the comment
/* delimiters around the symbolSubstitutionSupport setting below,
/* otherwise you will get the default processing.
/*
/*
/*****
/* symbolSubstitutionSupport = 'EXTENDED'

/*****
/*
/* Runtime settings
/*
/*
/* Instructions:
/* Configure whether you would like to automatically set SLIPs when a
/* potential vulnerability is found, default is 'OFF'
/* Set a one character prefix to use in the sample SLIP ID, default
/* is 'A'.
/* Configure whether you would like to clear the logrec database
/* while running zACS, default is 'NO'.
/* Specify your logrec data set name.
/* Required Job Card Parameters:
/* Choose the jobname you would like zACS tests to run under,
/* default is 'ZACSJ'.
/* Specify the MSGCLASS with which to submit testcase jobs, default
/* is 'A'.
/* Specify the REGION with which to submit testcase jobs, default
/* is '50' with unit 'M'.
/* Specify the TIME, in minutes, with which to submit testcase
/* jobs, default is '5'.
/* Specify the TIME, in minutes, with which to submit testcase
/* jobs for an advanced run, default is '25'.
/* Specify the SYSTEM with which to submit testcase jobs, default
/* is the special keyword 'DEFAULT' which will use the system on
/* which the job is submitted. The special keyword 'NONE' may be
/* used to omit the SYSTEM keyword
/* Specify the SCHENV with which to submit testcase jobs, default
/* is the special keyword 'NONE' which will omit the SCHENV
/* keyword. Escape single quotes by doubling them.
/* EX: 'PLEX_D01' for 'PLEX_D01'
/* Specify the full path of the work directory in the file system
/* to be used for USS testing

```

```

/* Specify whether you would like to include the mvs volume on the */
/* test mvs job. Default is YES. */
/* If mvs volume is included on the mvs job, the unit must also */
/* be specified. Default is 3390. */

/* Specify 'LOG' to have the PC/SVC log data go to the log data sets,*/
/* 'SCREEN' to have it print to the screen, or 'BOTH' to have it */
/* print to both the log data sets and the screen, default is */
/* 'SCREEN'. */
/* Note: When using the panel, this value is ignored and output */
/* will go to the log. */
/* Specify the dataset containing the JCL to generate the PC and */
/* SVC tables. */

/* Specify the dataset containing the PC table, '&userid.' can be */
/* used to substitute your userid. System symbols such as */
/* '&sysname.' are also valid */
/* Specify the dataset containing the SVC table, '&userid.' can be */
/* used to substitute your userid. System symbols such as */
/* '&sysname.' are also valid */
/* Specify the dataset containing the MVS table, '&userid.' can be */
/* used to substitute your userid. System symbols such as */
/* '&sysname.' are also valid */
/* Specify the dataset containing the USS table, '&userid.' can be */
/* used to substitute your userid. System symbols such as */
/* '&sysname.' are also valid */
/*
/*****
autoslip          = 'OFF'
slipIDPrefix      = 'A'
clearLogrec       = 'NO'
logrecDSname      = 'SYS1.LOGREC'
testcaseJobName   = 'ZAC SJ'
jobMsgClass       = 'A'
jobRegion         = '50M'
jobTimeout        = '5'
jobAdvTimeout     = '25'
jobSystem         = 'DEFAULT'
jobSchenv         = 'NONE'
ussWorkDir        = 'uss-work-dir'
useMvsVol         = 'YES'
mvsUnit           = '3390'
printLogOutput    = 'SCREEN'
jclDSname         = 'jcl-dataset'
pcTableDSname     = '&userid.ZACS.PCNUM'
svcTableDSname    = '&userid.ZACS.SVCNUM'
mvsTableDSname    = '&userid.ZACS.MVSNAME'
ussTableDSname    = '&userid.ZACS.USSNAME'

/*****

/*****
/*
/* Optional Job Card Parameters
/*
/* Instructions:
/* Configure the optional job card parameters to include on the
/* generated zACS jobs:
/* Configure whether you would like to include an account number,
/* default is 'NO'.
/* Specify your account number. Escape single quotes by doubling
/* them. EX: A,'1/2/3',B for A,'1/2/3',B
/* Configure whether you would like to include programmer name,
/* default is 'NO'.
/* Specify the programmers name. Escape single quotes by doubling
/* them. EX: T.O'NEILL for T.O'NEILL
/* Configure whether you would like to include any other keyword
/* parameters, default is 'NO'.
/* Specify any other job card parameters you would like to include,
/* if specifying other keywords, proper job card syntax must be
/* used. EX: 'NOTIFY=ZACUSER,PRTY=12'
/* Configure whether you would like to include an additional
/* customized DD, default is 'NO'.
/* Specify the DD JCL line.
/* Configure whether you would like to include an additional
/* JCL line for a jobparm, default is 'NO'.
/* Specify the jobparm JCL line
/*
/*****
useJobAccount      = 'NO'
jobAccount         = '<account-number>'
useJobProgrammer   = 'NO'
jobProgrammer      = '<programmer-name>'
useOtherKeywords   = 'NO'

```

```

jobOtherKeywords    = '<other-parameters>'
addCustomDD         = 'NO'
CustomDDJCL         = '</myDD>'
addJobParm          = 'NO'
JobParmJCL          = '<myjcl>'

/*****
/*
/* Filter settings
/*
/* Instructions:
/* Specify 'EXCLUDE' to exclude modules and/or jobnames during your
/* run or 'INCLUDE' to run a subset of modules and/or jobnames
/* during your run. This value is ignored if both filterByModName
/* and filterByJobname are 'NO'. Default is 'EXCLUDE'.
/* Change 'module-name-filter-list-dataset' to the name of the
/* data set containing the list of modules you would like to
/* include or exclude. Configure whether or not you would like to
/* include or exclude modules during your run, default is 'NO'.
/* Change 'jobname-filter-list-dataset' to the name of the data set
/* containing the list of jobnames you would like to include or
/* exclude. Configure whether or not you would like to include or
/* exclude jobnames during your run, default is 'NO'.
/*
/*
*****/
includeOrExclude    = 'EXCLUDE'
filterByModName     = 'NO'
modFilterDSname     = 'module-name-filter-list-dataset'
filterByJobname     = 'NO'
jobFilterDSname     = 'jobname-filter-list-dataset'

/*****
/*
/* Vulnerability log setting
/*
/* Instructions:
/* Change 'output-dataset' to the name of the data set to which you
/* would like to send detected potential vulnerability output.
/* This must match the data set name specified by MYOUTDD in the
/* started task.
/* Specify 'ERROR' to suppress test in progress messages printed to
/* the potential vulnerability log and to print return codes only
/* when an unsuccessful result is detected. Specify 'ALL' to print
/* test in progress messages and to print return codes to the
/* potential vulnerability log always. Default is 'ALL'
/* Specify 'ERROR' to suppress the summary information printed at
/* the end of each service in the potential vulnerability log when
/* a successful result is detected, or 'ALL' to print summary
/* information always. Default is 'ALL'.
/* Change 'summary-dataset' to the name of the data set to which you
/* would like to send detected potential vulnerability summary.
/* This must match the data set name specified by MYSUMDD in the
/* started task.
/*
/*
*****/
stOutDSname        = 'output-dataset'
printStatus        = 'ALL'
printSummary       = 'ALL'
sumOutDSname       = 'summary-dataset'

/*****
/*
/* SVC log settings
/*
/* Instructions:
/* Change 'svc-log-dataset' to the name of the data set to which you
/* would like to send run SVCs log information.
/*
/*
*****/
svcLogDSname       = 'svc-log-dataset'
svcLogDSlrec       = '80'
svcLogDSblksz      = '0'
svcLogDSPriSp      = '5'
svcLogDSSecSp      = '1'

/*****
/*
/* PC log settings
/*
/* Instructions:
/* Change 'pc-log-dataset' to the name of the data set to which you
/* would like to send run PCs log information.
/*

```



```

/*
/*****
pcLogDSname      = 'pc-log-dataset'
pcLogDSLrec      = '80'
pcLogDSblkksz    = '0'
pcLogDSPriSp     = '100'
pcLogDSSecSp     = '40'

/*****
/*
/* MVS log settings
/*
/* Instructions:
/* Change 'mvs-log-dataset' to the name of the data set to which you
/* would like to send run MVS programs log information.
/*
/*****
mvsLogDSname     = 'mvs-log-dataset'
mvsLogDSLrec     = '80'
mvsLogDSblkksz   = '0'
mvsLogDSPriSp    = '100'
mvsLogDSSecSp    = '40'

/*****
/*
/* USS log settings
/*
/* Instructions:
/* Change 'uss-log-dataset' to the name of the data set to which you
/* would like to send run USS programs log information.
/*
/*****
ussLogDSname     = 'uss-log-dataset'
ussLogDSLrec     = '2052'
ussLogDSblkksz   = '0'
ussLogDSPriSp    = '100'
ussLogDSSecSp    = '40'

/*****
/*****
/**
/** End zACS Configuration file
/**
/*****
/*****

```

Install the zACS Graphical User Interface (GUI)

This topic provides detailed instructions for installing and configuring the z/OSMF plugin User Interface with enhanced support for z/OS Authorized Code Scanner(zACS) functions. zACS must already be configured and operational before the plugin is configured and used. See [Chapter 3, “zACS configuration,”](#) on [page 5](#) for more information. Starting and stopping of zACS must be done manually on z/OS and must be done on the same system where z/OSMF runs.

It is essential to correctly configure the *stOutDSname* keyword in the configuration file. Set this parameter to the desired name of the data set to which you intend to send the zACS scan results. It is crucial that the specified data set name matches the one defined by MYOUTDD in the started task. Additionally, *printLogOutput* must be set to 'SCREEN' with *printStatus* and *printSummary* must be set to 'ALL'. By aligning these settings, the GUI will be able to accurately capture and deliver the test result data to the designated data set. Pre-existing results from scans run not using these settings may not display correctly in the GUI. Be sure *jobSystem* is set to the system that z/OSMF runs on so that the generated tables reflect the programs available on the system where the testing takes place.

Step 1: Import the Properties File in z/OSMF

Follow the steps below to import the properties file into z/OSMF:

1. Log into z/OSMF
2. Access the z/OSMF Import Manager task. If you selected the z/OSMF classic view, click the *Import Manager* task in the *z/OSMF Administration* category. Otherwise, if you selected the z/OSMF desktop view, click the *Import Manager* icon on the desktop.

3. In the *Import Manager* task, select the *Import* tab and specify the full file path and name of the properties file: `/usr/lpp/bcp/zacs/zacs.properties`.
4. Click *Import*. A message is displayed to indicate that the plug-in was added.

Note: The `zacs.properties` file contains the necessary configuration settings for zACS GUI and must be installed on the system.

Step 2: Authorize users to the zACS z/OSMF User Interface Task

A user or group must be explicitly given access to view and work with the zACS GUI in z/OSMF.

Note: The following instructions are specific to z/OS Security Server (RACF). If another ESM is used, then refer to the applicable vendor documentation for equivalent security configuration.

The RDEFINE command creates a SAF profile with the default prefix `IZUDFLT`. The SAF resource name `ZOSMF.IBM_ZACS.APP.ZACSUI` is configured in the `zacs.properties` file.

1. Define the SAF profile

```
RDEFINE ZMFAPLA IZUDFLT.ZOSMF.IBM_ZACS.APP.ZACSUI UACC(NONE)
```

Figure 26. Defining SAF profile for zACS GUI

2. Permit user access to the ZMFAPLA:

Each user must be given access to the zACS GUI. A group can also be permitted to use zACS GUI which will enable each user in the group to access zACS GUI. You can use the same group (e.g. BPNGRP), if one is already defined for zACS during configuration.

```
PERMIT IZUDFLT.ZOSMF.IBM_ZACS.APP.ZACSUI CLASS(ZMFAPLA) ID(<user ID or user group>)  
ACCESS(CONTROL)
```

Figure 27. Permitting user access to ZMFAPLA

3. Refresh the profiles to activate the changes made to the ZMFAPLA class take:

```
SETROPTS RACLIST(ZMFAPLA) REFRESH
```

Figure 28. ZMFAPLA refresh

4. Ensure the user has access to zACS data set and profiles.

It is important that the user logging in to the z/OSMF to use the zACS GUI has access to the zACS profile and data sets. Refer to section [“Data set protection and permissions” on page 5](#) for more information.

5. Ensure the user has access to the following z/OSMF APIs:

- z/OS data set and file REST interface
- z/OS jobs REST interface
- Data persistence services
- TSO/E address space services

Instructions to get access can be found in the [IBM z/OS Management Facility Configuration Guide](#).

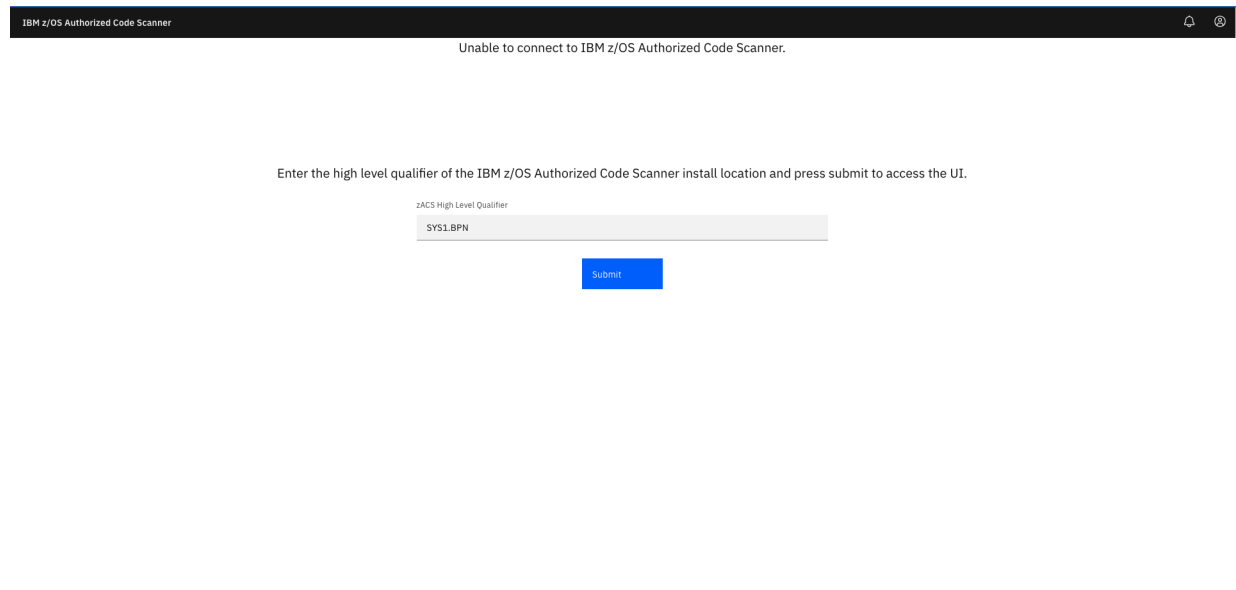
6. Launch the application.

In the z/OSMF desktop view, select the icon *IBM z/OS Authorized Code Scanner* on the desktop. If the icon is not found in the desktop, it can be viewed and pinned from the App center. In the z/OSMF classic view, expand the Software category and select *IBM z/OS Authorized Code Scanner*.

7. Ensure the zACS installation high-level qualifier is correctly set.

The zACS installation high-level qualifier must be accurately set after logging in to the GUI. The high-level qualifier consists of all but the final qualifier of the zACS installation, for example if zACS installation is located under the data sets named `SYS1.BPN.SBPN*` then the high-level qualifier is `SYS1.BPN` as shown in [Figure 31 on page 23](#). By default the high-level qualifier is `SYS1.BPN`. If

this high-level qualifier cannot be found, you are sent to an opening page to input a valid high level qualifier.



IBM z/OS Authorized Code Scanner

Unable to connect to IBM z/OS Authorized Code Scanner.

Enter the high level qualifier of the IBM z/OS Authorized Code Scanner install location and press submit to access the UI.

zACS High Level Qualifier

SYS1.BPN

Submit

Figure 29. Opening Page

If you need to change the high-level qualifier, do so through the setting page by clicking the gear icon in the upper right of the page.



IBM z/OS Authorized Code Scanner

Tables Test Results

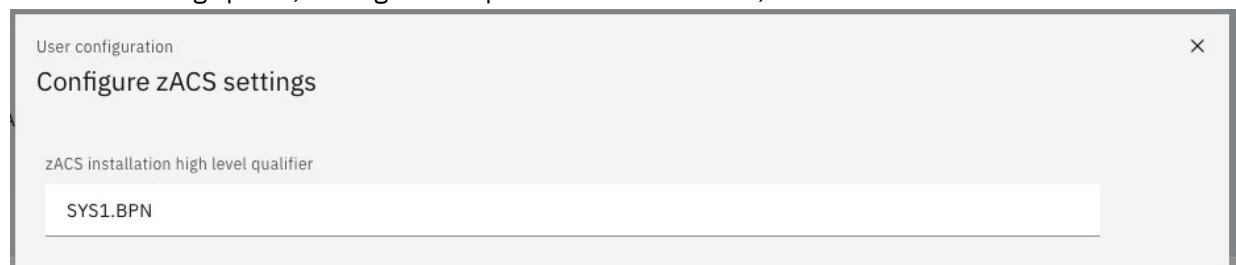
Generate and View Tables

List active PCs, SVCs, USS Programs and MVS Programs on a system. ZACS can scan these items for potential vulnerability.

PCs SVCs USS Programs MVS Programs

Figure 30. GUI Setting Button

From the settings panel, the high-level qualifier is the first field, as shown below.



User configuration

Configure zACS settings

zACS installation high level qualifier

SYS1.BPN

Figure 31. zACS High Level Qualifier Field

Chapter 4. Running zACS

This chapter assumes a high level qualifier of 'SYS1.BPN.**'.

Note: Changes made to the configuration file or filter lists while a run is in progress will not be reflected until the subsequent run.

Setup

Define a new profile for BPNZACS:

```
RDEFINE STARTED BPNZACS.** UACC(NONE) STDATA(USER(user) GROUP(SYS1) TRUSTED(YES))
SETR RACLIST(STARTED) REFRESH
```

Figure 32. Define BPNZACS profile and refresh

Start the tool with your started task by using the console:

```
S BPNZACS,REUSASID=YES
```



Warning: Starting this task clears the output data set defined in the started task, including any vulnerability data that was found.

The tool sets the following SLIPs when a zACS job is run. ZAC SJ is replaced with the name of the job, set in the configuration file.

```
SLIP SET, ID=BPN1, ERRTP=PROG, JOBNAME=ZAC SJ*, A=(RECORD, NODUMP), END
```

```
SLIP SET, ID=BPN2, COMP=U4088, JOBNAME=ZAC SJ*, A=(RECORD, NODUMP), END
```

Note: These SLIPs can override other SLIPs set on the system.

During a test run, zACS intentionally generates many ABENDs. This SLIP suppresses all memory dump generation while the zACS task is started, unless a SLIP is set afterward, overriding it. The SLIP is removed when the task is stopped with the following command:

```
P BPNZACS
```

Figure 33. Starting tool, setting SLIP trap, and purging tool.

Before testing PCs, run BPNPCNUM to generate a PC table in *userid*.ZACS.PCNUM. As PC numbers can change, run this job regularly to ensure that the table is up to date:

```
submit SYS1.BPN.SBPNSAMP(BPNPCNUM)
```

Before you test SVCs, run BPNSVCNM to generate SVC entries in *userid*.ZACS.SVCNUM. The job to generate the SVC table needs to be ran only once for every IPL, anytime the SVC table, or ESR SVCs are updated.

```
submit SYS1.BPN.SBPNSAMP(BPNSVCNM)
```

Figure 34. Generate PC table and SVC entries.

Before you test MVS programs, run BPNMVSNM to generate MVS data set entries in *userid*.ZACS.MVSNAME. Run the BPNMVSNM job each time before you test because the online status of volumes may have changed.

Note: Results are only returned from SYS1.NUCLEUS if it is in the APF list.

```
submit SYS1.BPN.SBPNSAMP(BPNMVSNM)
```

Before you test z/OS UNIX programs, run BPNUSSNM to generate z/OS UNIX data set entries in `userid.ZACS.USSNAME`. The job to generate the z/OS UNIX table should be run anytime an authorized UNIX program, or one with the `setuid` or `setgid` flag on, is added.

```
submit SYS1.BPN.SBPNSAMP(BPNUSSNM)
```

Running using the Command Line

zACS can be run via the command line by executing the BPNKNSVC, BPNKNPCX, BPNKNMVS and BPNKNUSS REXX executables to run SVCs PCs, MVS programs and z/OS UNIX programs respectively.

Note: Custom modifications to the provided REXX are not supported.

To test all SVCs in the generated SVC table, run:

```
ex 'SYS1.BPN.SBPNEEXEC(BPNKNSVC) '
```

To test all PCs in the generated PC table, run:

```
ex 'SYS1.BPN.SBPNEEXEC(BPNKNPCX) '
```

To test all MVS programs in the generated MVS table, run:

```
ex 'SYS1.BPN.SBPNEEXEC(BPNKNMVS) '
```

To test all z/OS UNIX programs in the generated z/OS UNIX table, run:

```
ex 'SYS1.BPN.SBPNEEXEC(BPNKNUSS) '
```

Figure 35. Testing with Command line Examples

Note: When kicking off a full run, potentially hundreds of JCL jobs will be submitted. Jobs from any subsequent runs will be queued up behind the initial run if it is not yet complete. It is strongly advised to wait until all jobs have completed before starting a new run. To limit the number of SVC or PCs run see [“Filtering Run Services with Include Lists or Exclude Lists”](#) on page 42. It is also strongly advised to not submit the JCL test jobs directly, bypassing the REXX execs.

Optional Parameters

To further limit the number of services tested in a run, or to enable advanced testing, optional parameters may be used when running the REXX executables.

Note: The exclude lists described in [“Filtering Run Services with Include Lists or Exclude Lists”](#) on page 42 will be ignored when optional parameters that limit tests run are specified.

- BPNKNSVC:

```
ex 'SYS1.BPN.SBPNEEXEC(BPNKNSVC) ' '<svc#>,<esr#>,<module_name>,<test_type>'
```

- Single SVC number - Used to run just a single SVC, specified as hexadecimal. If ESR Routing number is also specified, then only the hexadecimal values 6D, 74, 7A or 89 are accepted.
- ESR Routing Number - Routing number to run a single Extended SVC, the single SVC number must also be specified in hexadecimal.
- Module name - Used to run all SVCs for a single module.
- Test type - The type of test to be run. Specify 1 for advanced or 0 for basic. Advanced testing enables function code testing. Time until completion will increase when compared to a basic test. If nothing is specified for this parameter, a basic test will be run.

- BPNKNPCX:

- `ex 'SYS1.BPN.SBPNEEXEC(BPNKNPCX)' '<pc#>,<sequence#>,<module_name>,<test_type>'`
- Single PC number - Used to run just a single PC, specified in hexadecimal.
- Sequence number - Sequence number for a PC, single PC number must also be specified in hexadecimal. If a sequence number is not specified, the default is 0.
- Module name - Used to run all PCs for a single module.
- Test type - The type of test to be run. Specify 1 for advanced or 0 for basic. Advanced testing enables function code testing. Time until completion will increase when compared to a basic test. If nothing is specified for this parameter, a basic test will be run.

- BPNKNMVS

- `ex 'SYS1.BPN.SBPNEEXEC(BPNKNMVS)' '<member-name>,<test_type>'`
- Member name – Used to run mvs programs of a single member name
- Test type - The type of test to be run. Specify 1 for advanced or 0 for basic. Advanced testing enables DDname and TSO testing. If a MVS program is eligible for TSO specific testing, a second test job will be submitted. If nothing is specified for this parameter, a basic test will be run.

- BPNKNUSS

- `ex 'SYS1.BPN.SBPNEEXEC(BPNKNUSS)' '<path-and-file-name>'`
- Path and file name – full path and file name used to run a single z/OS UNIX program

Note: When running with optional parameters for PCs or SVCs, a module name may not be specified if a single PC or SVC number is already specified.

For example, run only PCs in module IEAVH607:

```
ex 'SYS1.BPN.SBPNEEXEC(BPNKNPCX)' ' ',,IEAVH607'
```

Similarly, run only ESR 26 with routing number 109:

```
ex 'SYS1.BPN.SBPNEEXEC(BPNKNSVC)' '6D,1A,'
```

Figure 36. Example PC and SVC run.

Running using the Panel

To start the panel, run the following command from the command line:

```
ex 'SYS1.BPN.SBPNEEXEC(BPNISPFR) '
```

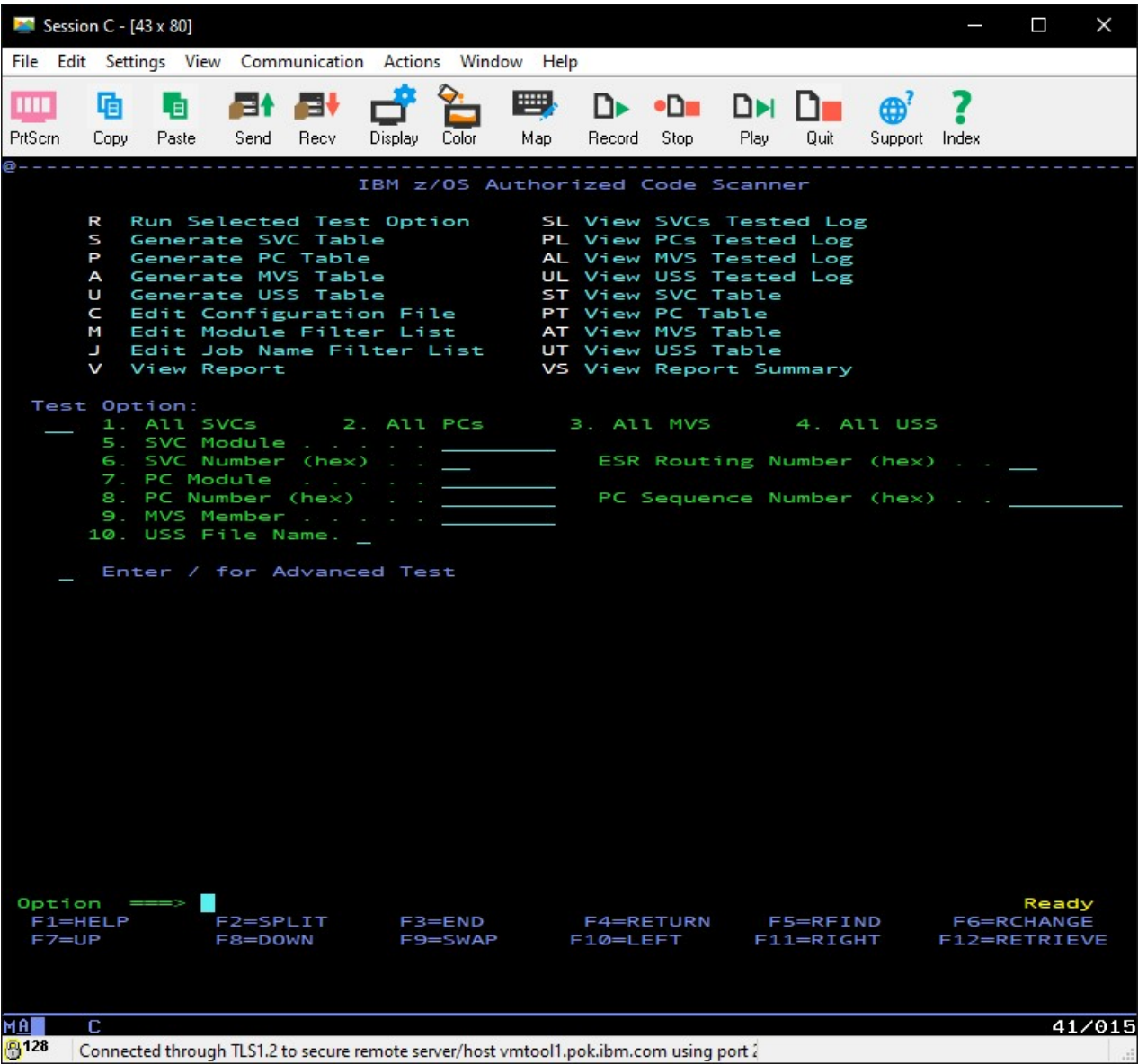


Figure 37. zACS Panel

zACS can be run via the panel by specifying the numbers 1-10 in the Test Option field, and the run option 'R' in the option field.

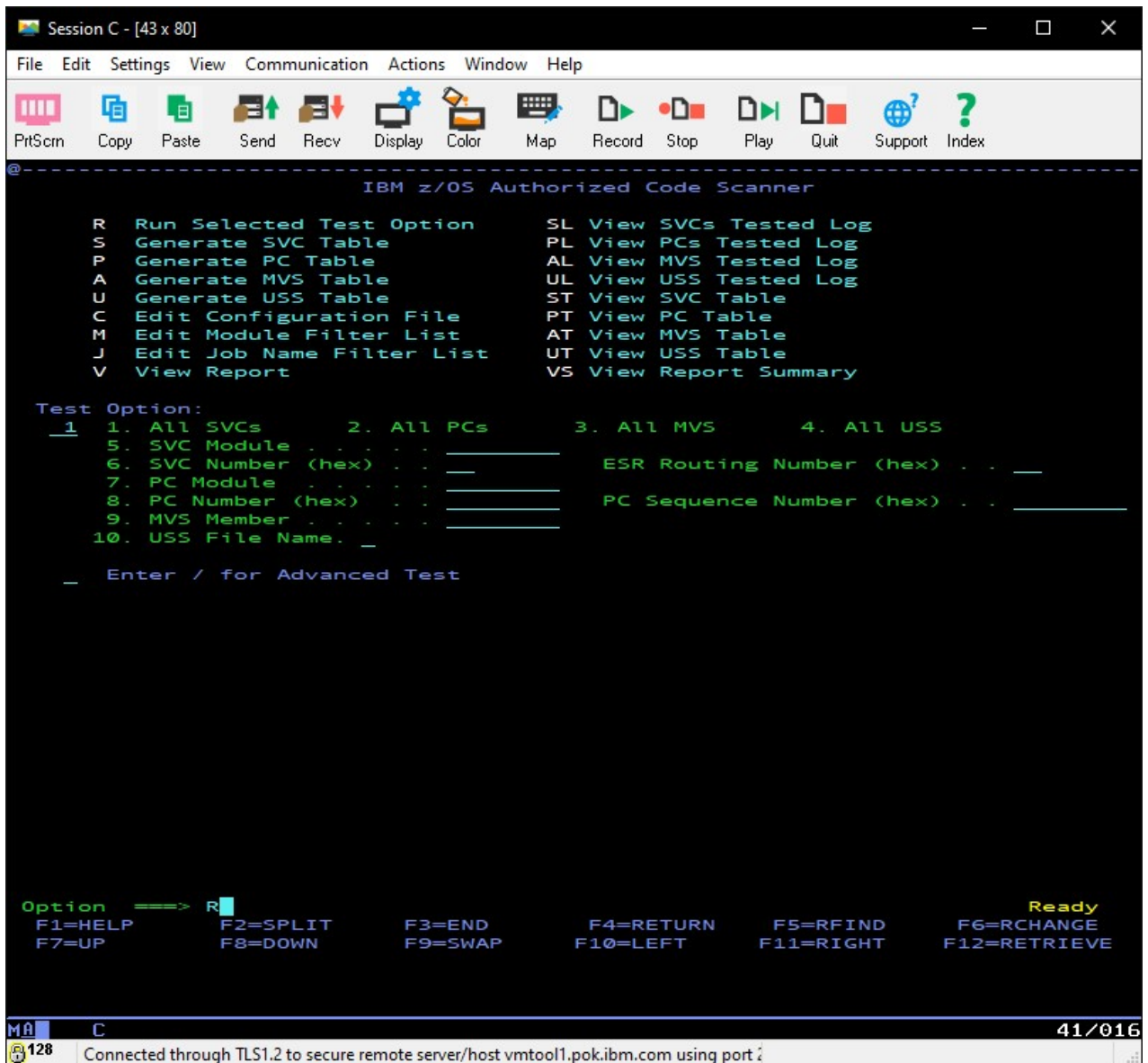


Figure 38. Running all SVCs from the panel.

To test all SVCs in the generated SVC table, specify test option 1 in the Test Option field, specify option 'R' in the Option field, and press enter.

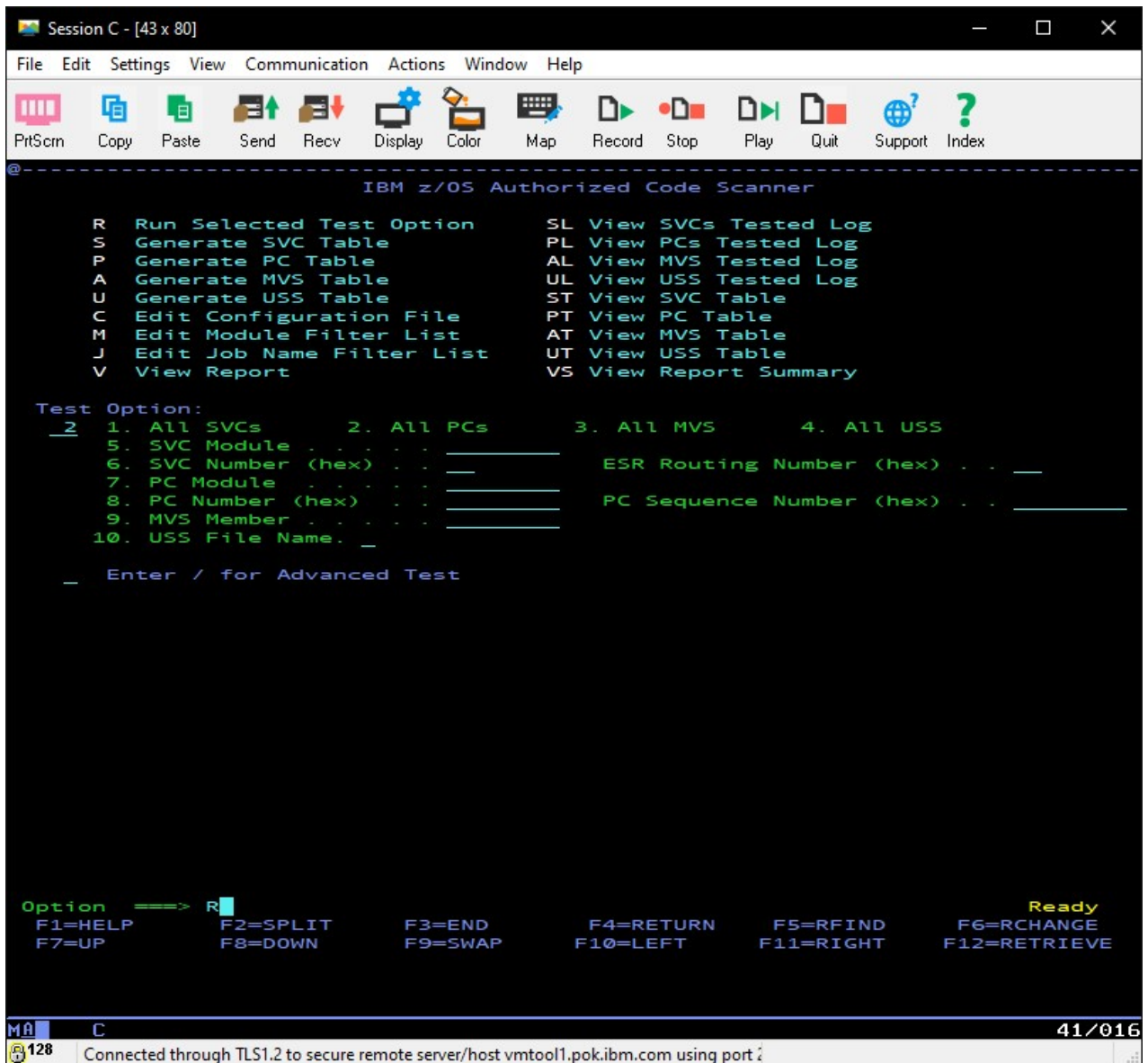


Figure 39. Running all PCs from the panel.

To test all PCs in the generated PC table, specify test option 2 in the Test Option field, specify option 'R' in the Option field, and press enter.

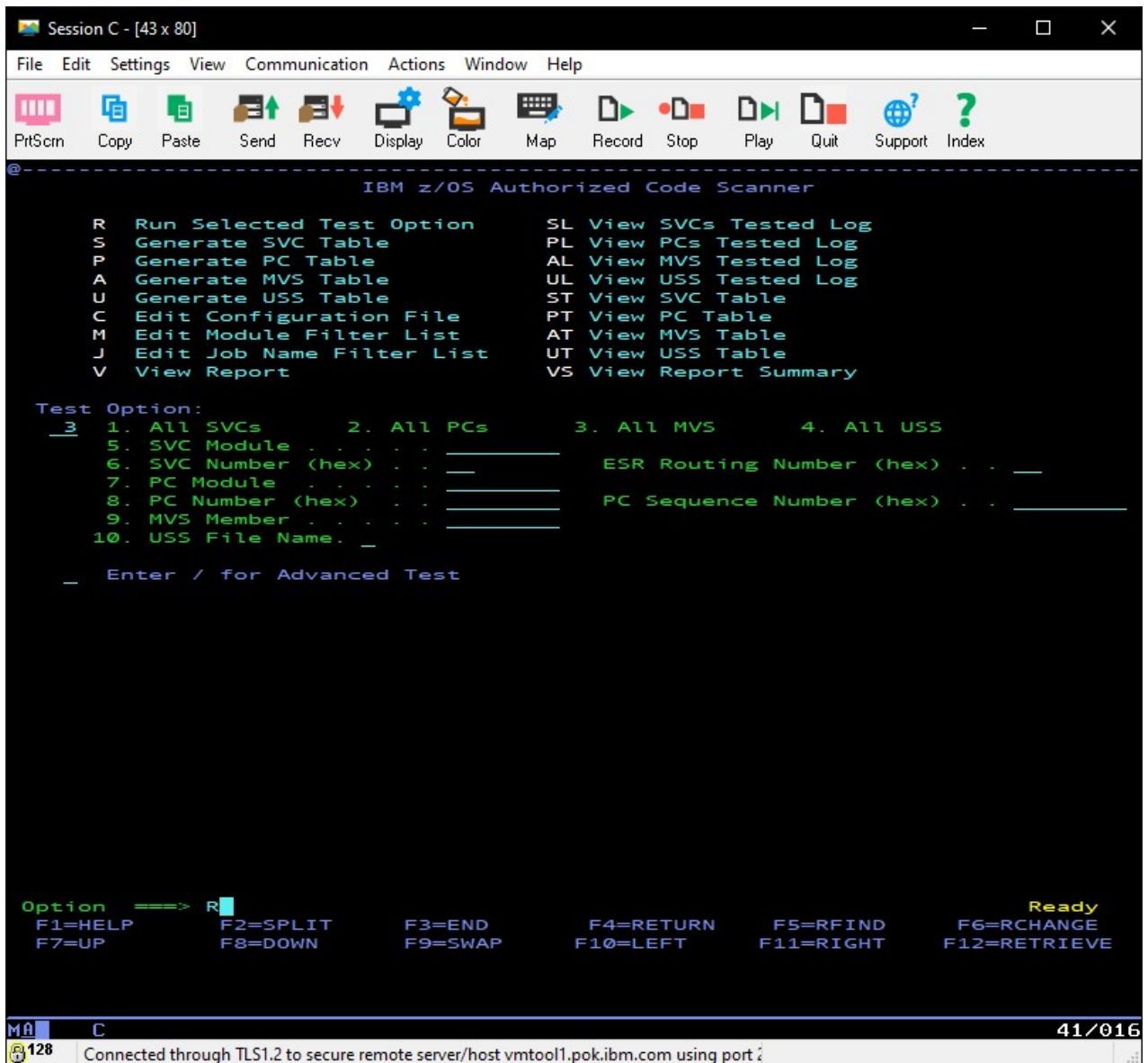


Figure 40. Running all MVS programs from the panel.

To test all MVS programs in the generated MVS table, specify test option 3 in the Test Option field, specify option 'R' in the Option field, and press enter.

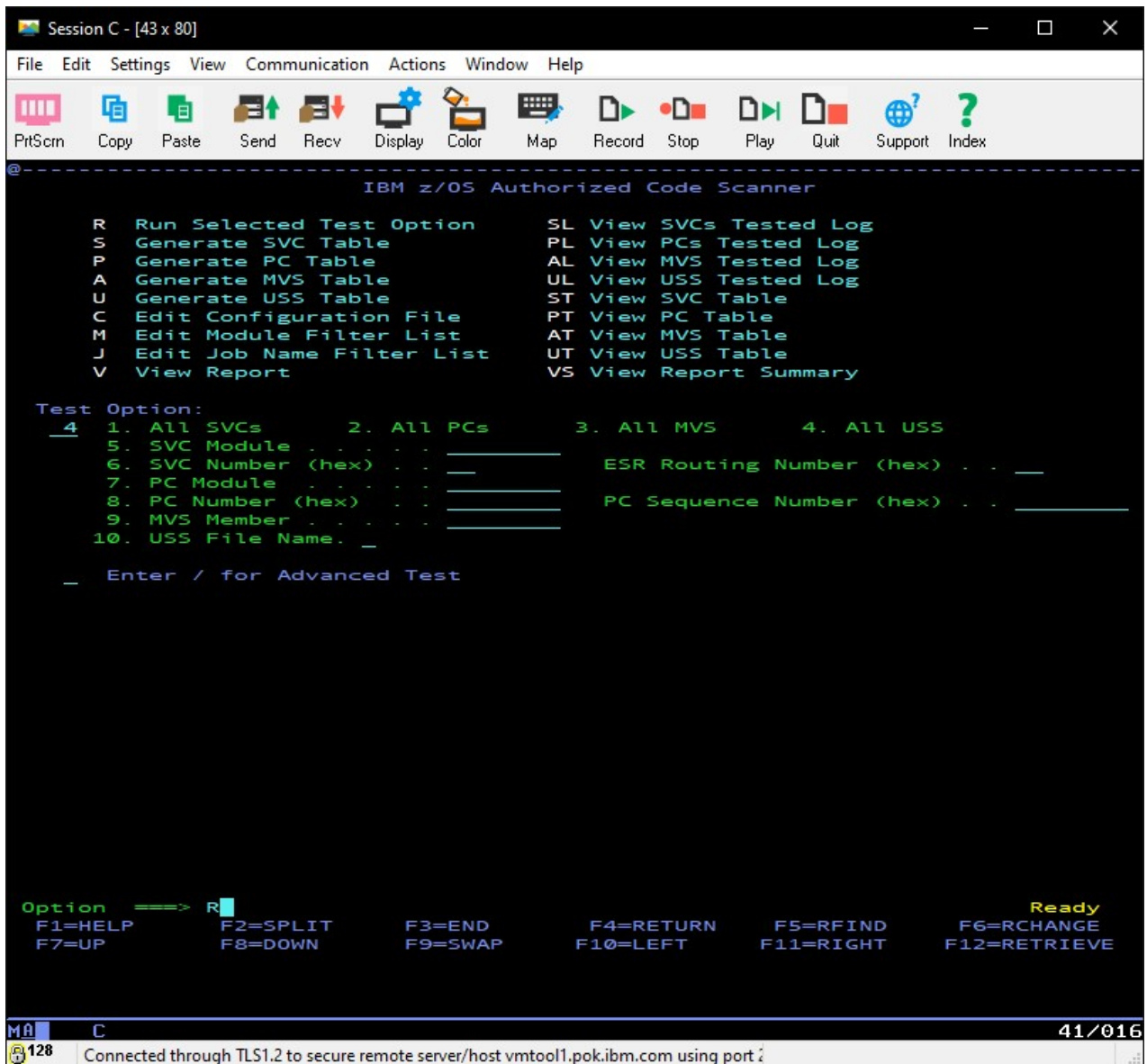


Figure 41. Running all UNIX programs from the panel.

To test all z/OS UNIX programs in the generated z/OS UNIX table, specify test option 4 in the Test Option field, specify option 'R' in the Option field, and press enter.

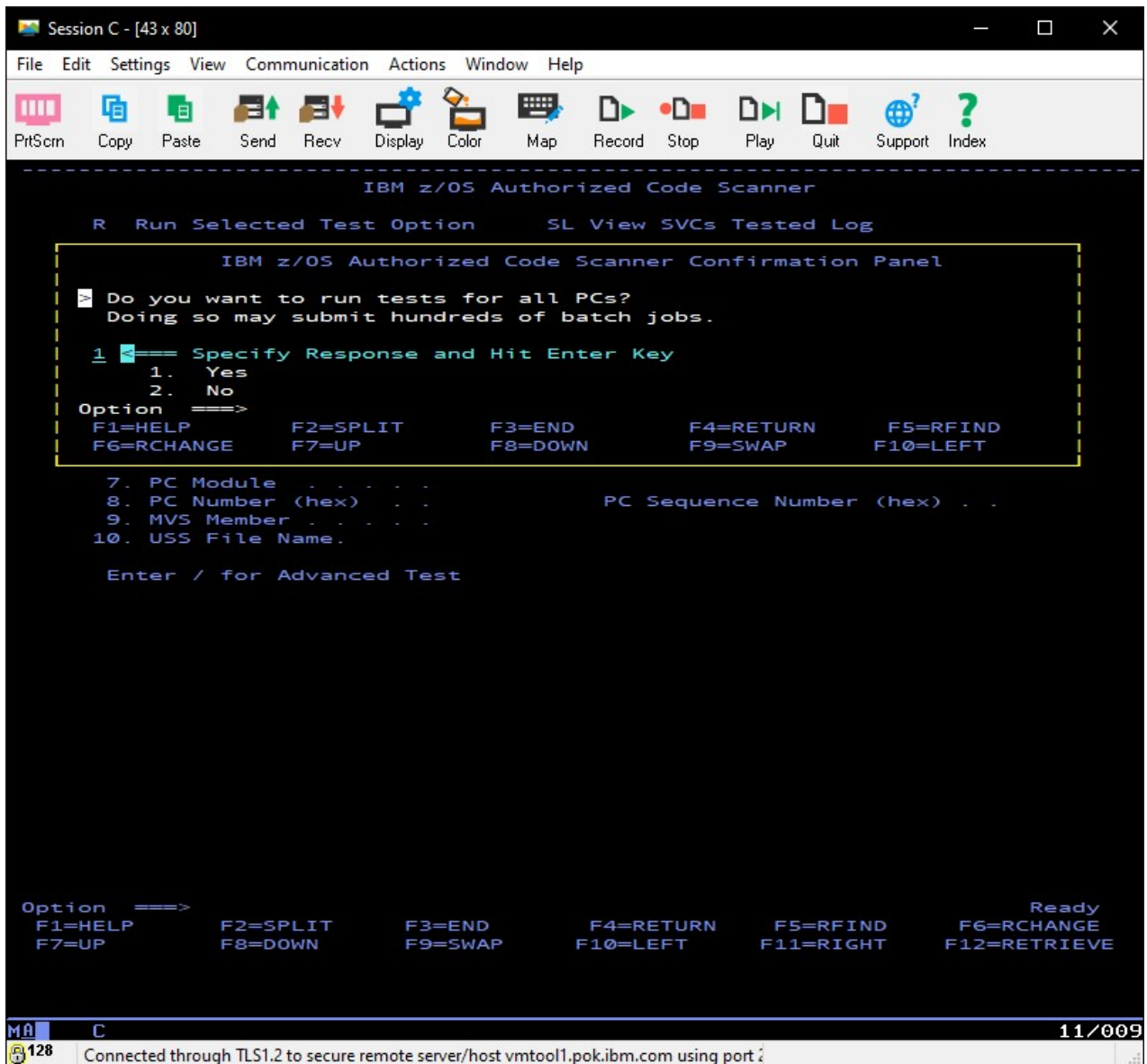


Figure 42. Confirmation pop-up for All PCs run.

When kicking off a full run, potentially hundreds of JCL jobs will be submitted. When the 'R' option is selected with test options 1-4, a confirmation window will pop up to prevent accidentally starting a large run. To continue with the run, select 1 and press enter, to prevent the run from starting select 2 and press enter.

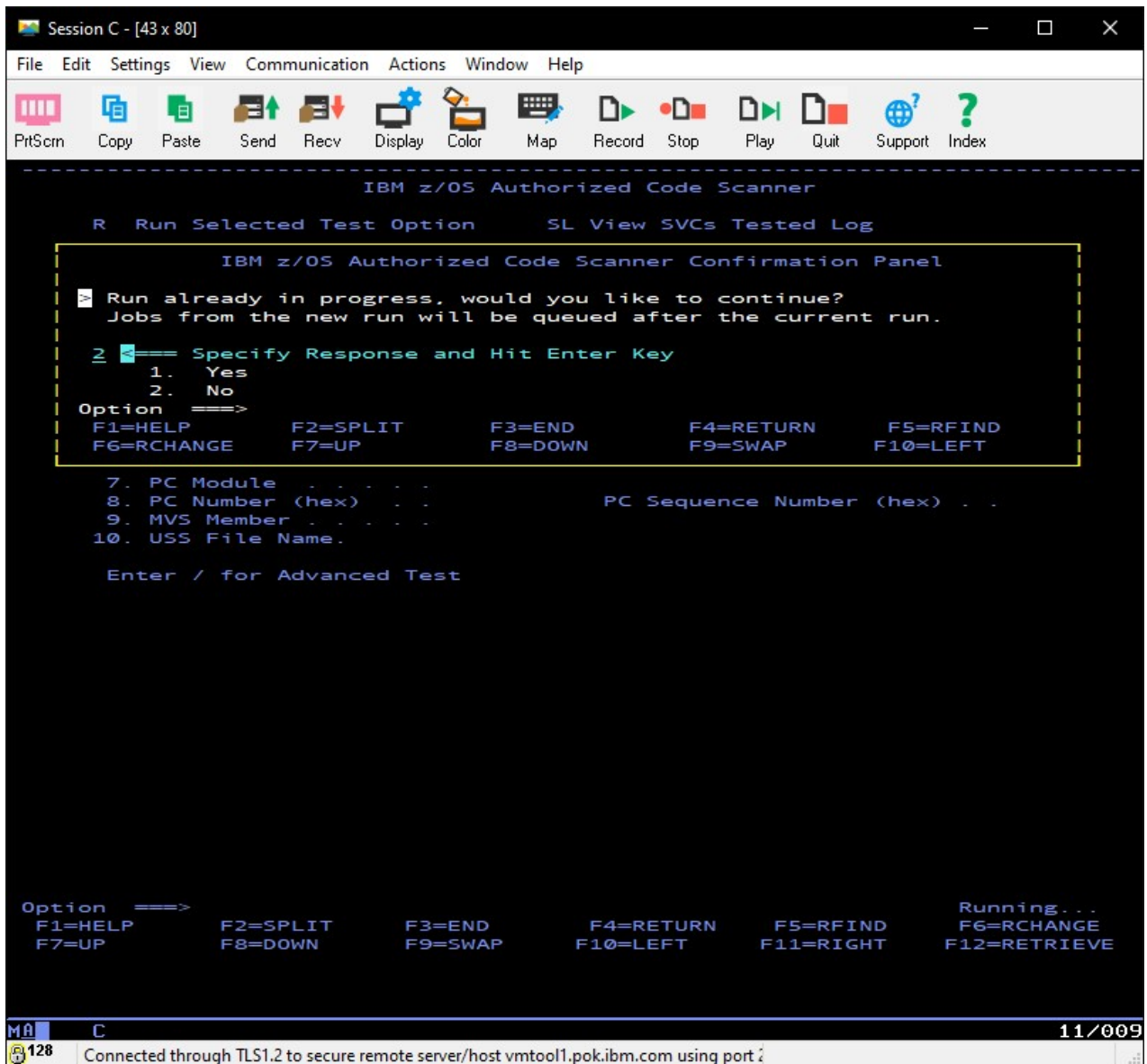


Figure 43. Confirmation pop-up for run already in progress.

Jobs from any subsequent runs will be queued up behind the initial run if it is not yet complete. It is strongly advised to wait until all jobs have completed before starting a new run. If a run is not yet complete, a confirmation window will pop up to prevent accidentally starting a new run. To continue with the run, select 1 and press enter, to prevent the run from starting select 2 and press enter.

Note: To limit the number of SVC or PCs run see [“Filtering Run Services with Include Lists or Exclude Lists”](#) on page 42.

Optional Parameters

To further limit the number of services tested in a run, optional parameters may be used when running the from the panels by specifying test options 5, 6, 7, 8, 9, or 10 in the Test Option field, filling in the necessary parameter fields, specifying option 'R' in the Option field, and pressing enter.

Note: The exclude lists described in [“Filtering Run Services with Include Lists or Exclude Lists”](#) on page 42 will be ignored when optional parameters are specified.

Test Option 5

Run all SVCs for a single module. Uses the following parameter field:

Module Name (Required)

The name of the module for which all SVCs are to be run.

Test Option 6

Run a single SVC. Uses the following parameter fields:

SVC Number (Required)

The hexadecimal SVC number to be run. If ESR Routing number is also specified, then only the hexadecimal values 6D, 74, 7A or 89 are accepted.

ESR Routing Number (Optional)

The hexadecimal routing number to run a single Extended SVC.

Test Option 7

Run all PCs for a single module. Uses the following parameter field:

Module Name (Required)

The name of the module for which all PCs are to be run.

Test Option 8

Run a single PC. Uses the following parameter fields:

PC Number (Required)

The hexadecimal PC number to be run.

PC Sequence Number (Optional)

The hexadecimal sequence number of the PC number to be run. If not specified, the default is 0.

Test Option 9

Run all MVS programs for a single member name. Uses the following parameter field:

MVS Member (Required)

The name of the member for which all MVS programs are to be run.

Test Option 10

Run a single UNIX file. Uses the following parameter field:

z/OS UNIX Path and File Name (Required)

Enter any character to open up the z/OS UNIX Path/File popup. There, place the full path, beginning with a '/', and file name to be run.

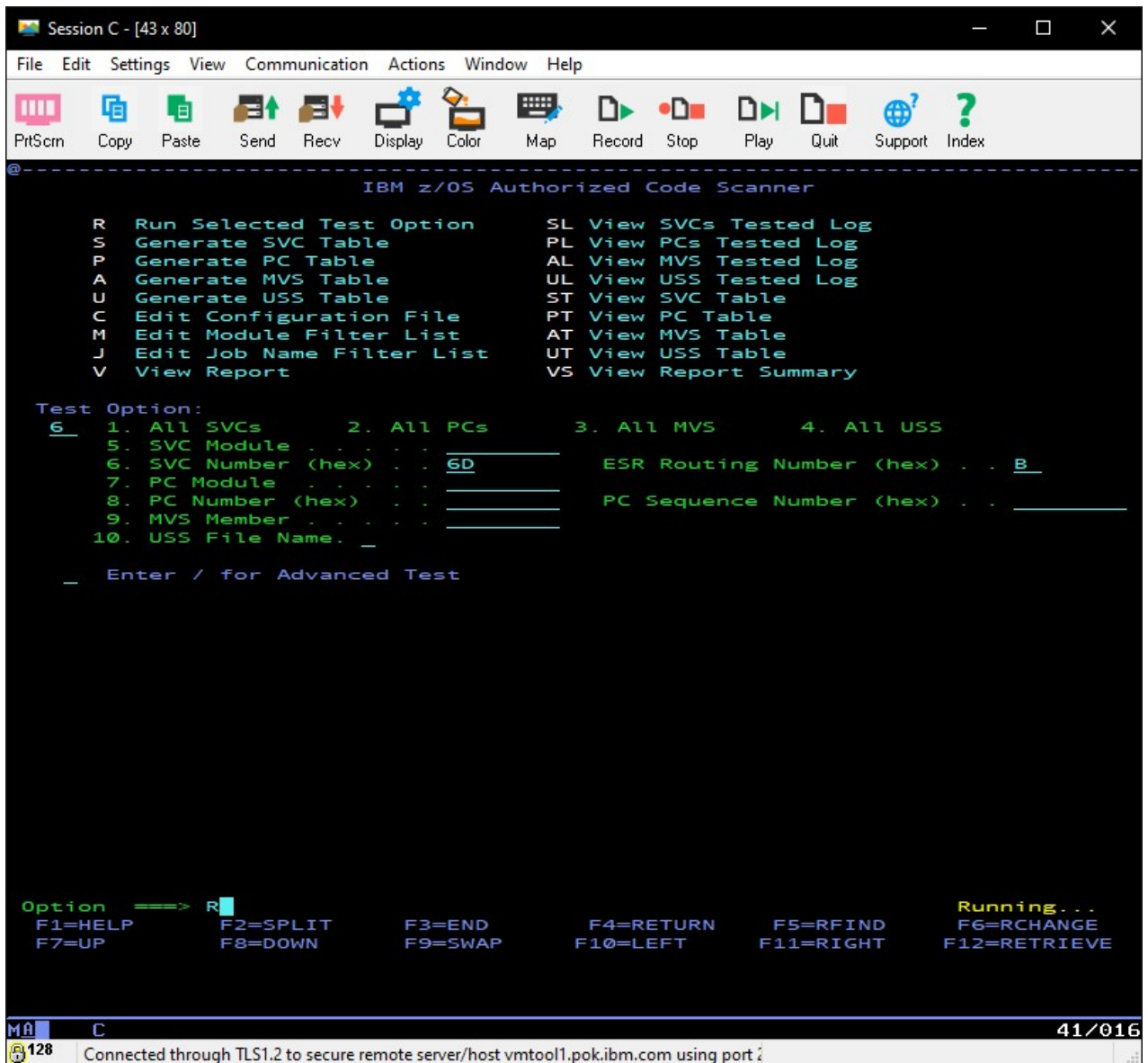


Figure 44. Running a single SVC from the panel with routing number 109 and ESR 11.

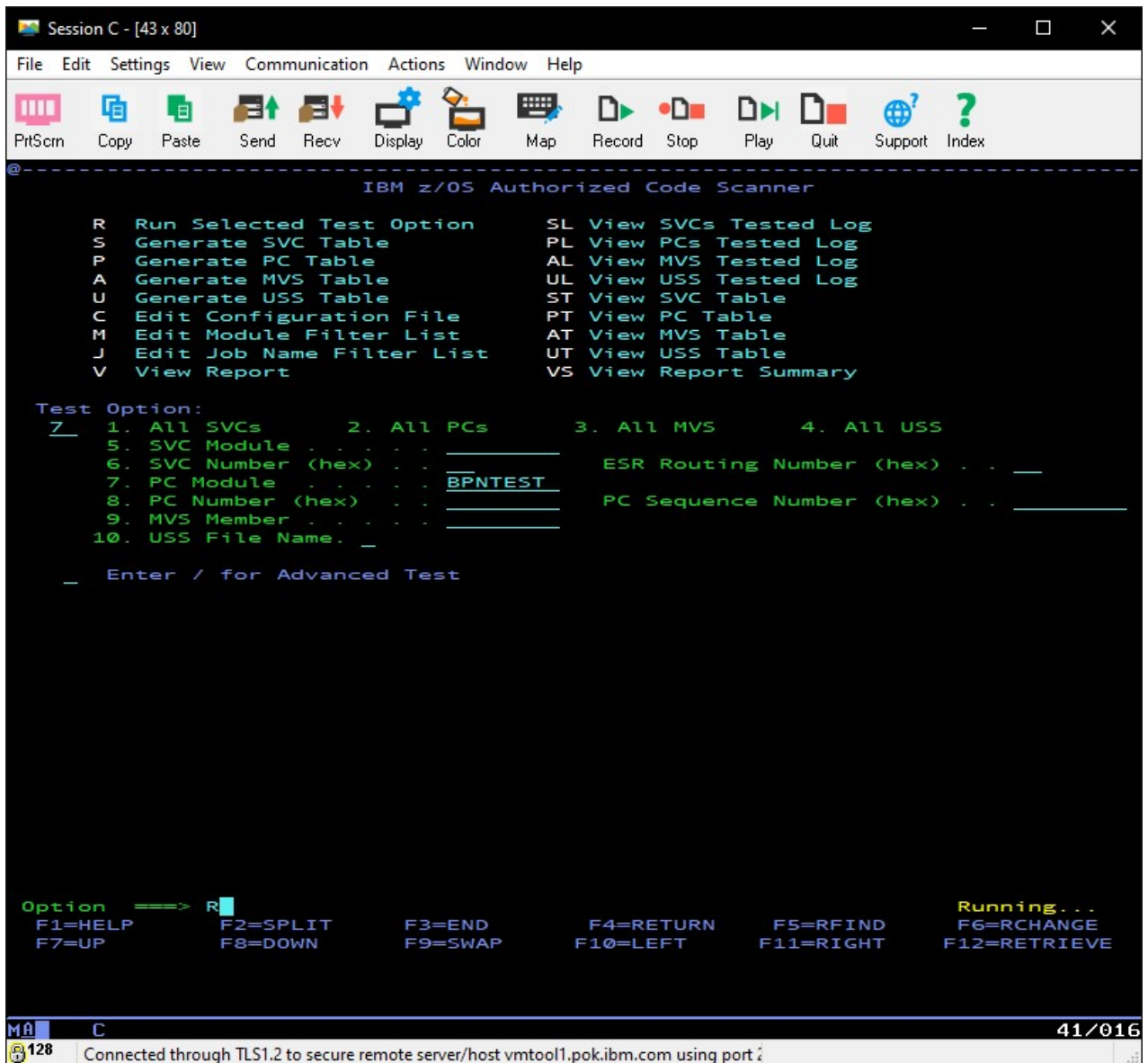


Figure 45. Running PCs by module name from the panel with module name BPNTTEST.

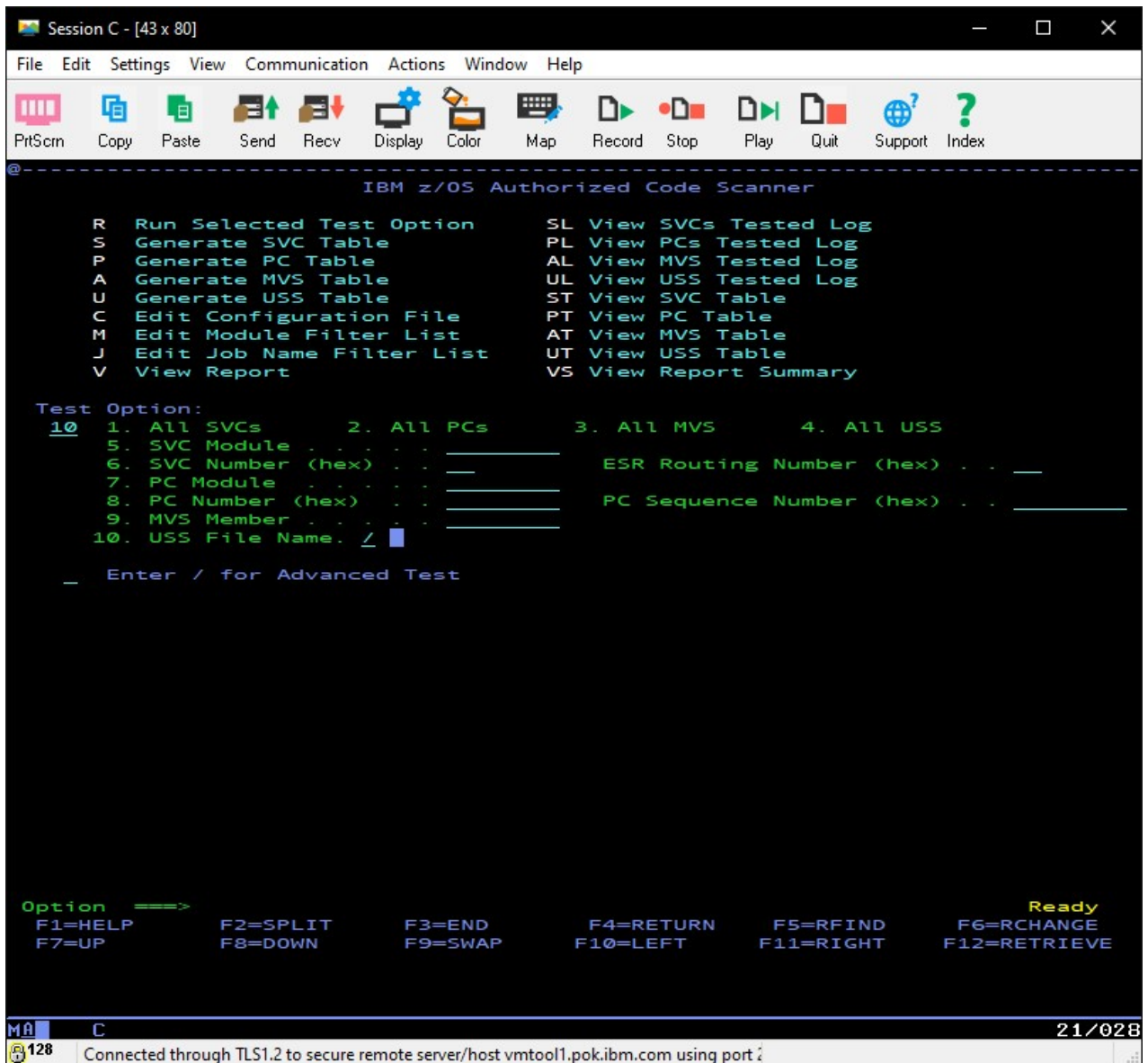


Figure 46. Opening the z/OS UNIX path popup.

Entering any character into the z/OS UNIX File Name field and then hitting 'enter' will bring up the z/OS UNIX Path/File popup.

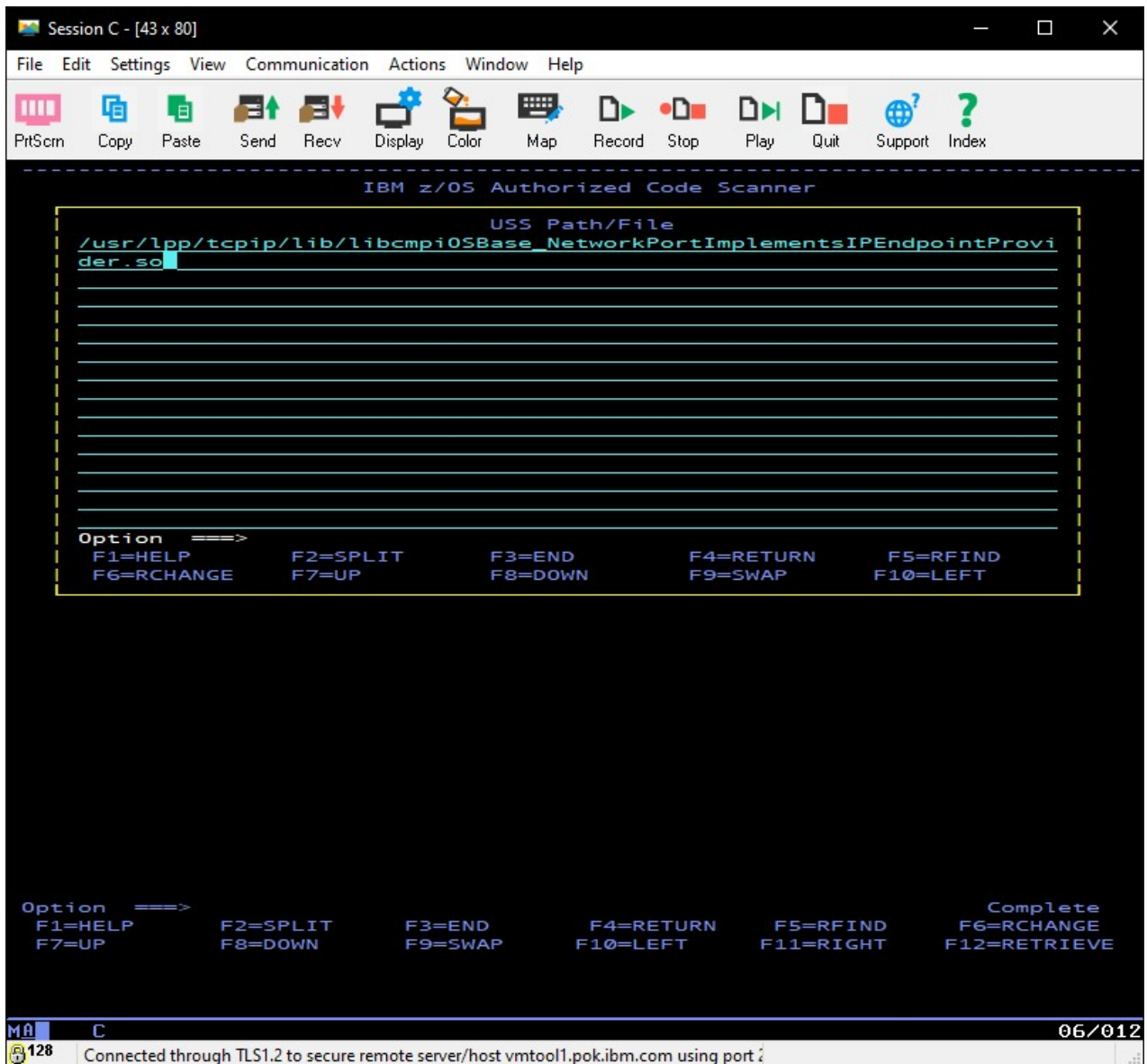


Figure 47. The z/OS UNIX path popup.

Here, a full path and file name can be entered. Paths and file name combined may be up to 1023 characters.

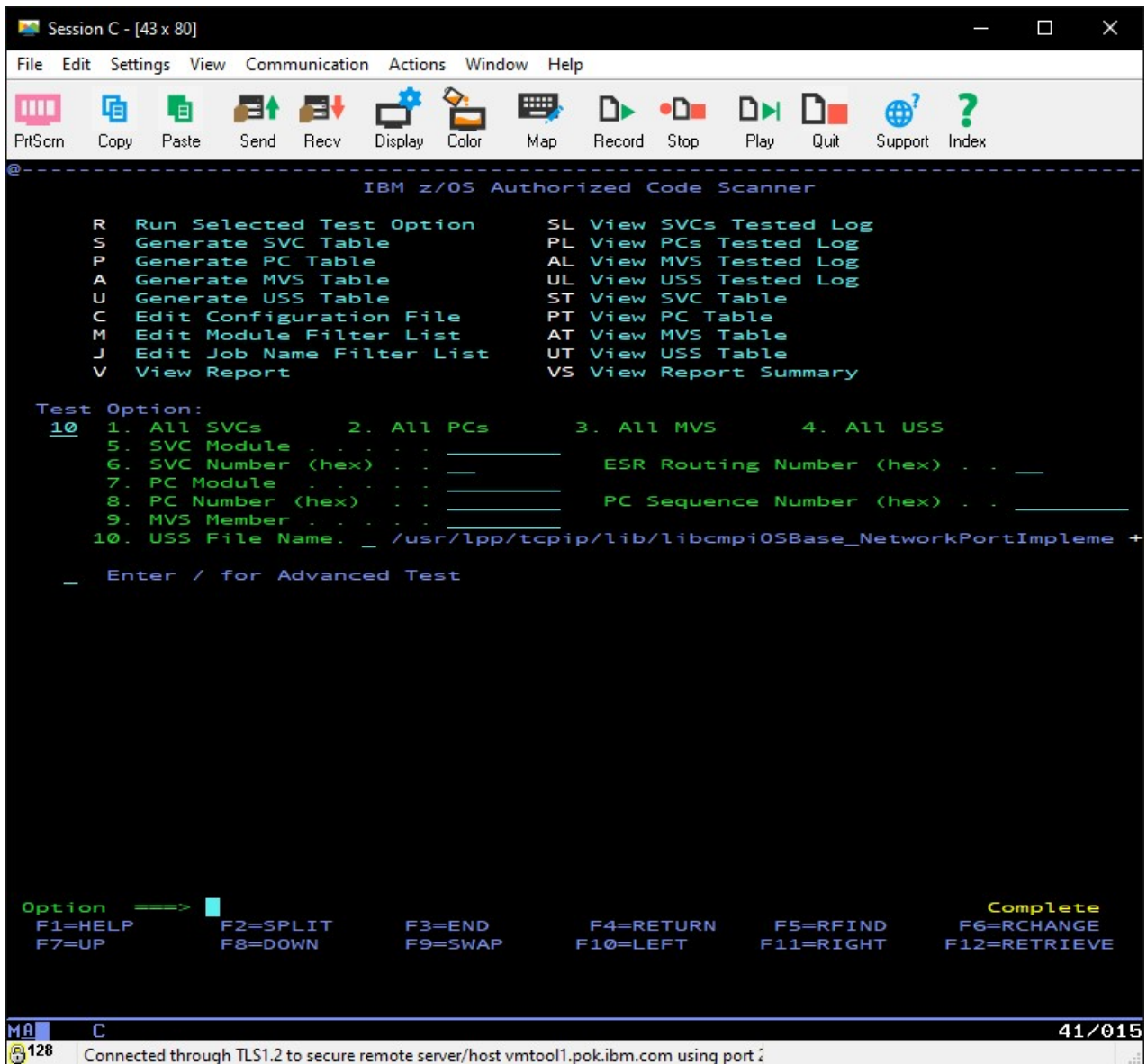


Figure 48. The main panel with populated z/OS UNIX path field.

When this field is populated the main panel will display the path name, truncated to 51 characters.

Additional Panel Options

In addition to running zACS tests, other actions may be taken from the zACS Panel by specifying its respective option in the option field and pressing enter.

Option S

Run the JCL, the containing data set of which is specified in the configuration file, to generate the SVC Table.

Option P

Run the JCL, the containing data set of which is specified in the configuration file, to generate the PC Table.

Option A

Run the JCL, the containing data set of which is specified in the configuration file, to generate the MVS Table.

Option U

Run the JCL, the containing data set of which is specified in the configuration file, to generate the z/OS UNIX Table.

Option C

Opens the configuration file, located at 'userid.ZACS.CONFIG', in edit mode, if it exists.

Option M

Opens the exclusion/inclusion by module name list, as specified by the keyword filterByModName in the configuration file, in edit mode, if it exists.

Option J

Opens the exclusion/inclusion by job name list, as specified by the keyword filterByJobName in the configuration file, in edit mode, if it exists.

Option V

Opens the vulnerabilities log, as specified by the keyword stOutDSname in the configuration file, in browse mode, if it exists.

Option SL

Opens the SVC log, as specified by the keyword svcLogDSname in the configuration file, in browse mode, if it exists.

Option PL

Opens the PC log, as specified by the keyword pcLogDSname in the configuration file, in browse mode, if it exists.

Option AL

Opens the MVS log, as specified by the keyword mvsLogDSname in the configuration file, in browse mode, if it exists.

Option UL

Opens the z/OS UNIX log, as specified by the keyword ussLogDSname in the configuration file, in browse mode, if it exists.

Option ST

Opens the SVC table, as specified by the keyword svcTableDSname in the configuration file, in browse mode, if it exists.

Option PT

Opens the PC table, as specified by the keyword pcTableDSname in the configuration file, in browse mode, if it exists.

Option AT

Opens the MVS table, as specified by the keyword mvsTableDSname in the configuration file, in browse mode, if it exists.

Option UT

Opens the z/OS UNIX table, as specified by the keyword ussTableDSname in the configuration file, in browse mode, if it exists.

Option VS

Opens the vulnerabilities summary log, as specified by the keyword sumOutDSname in the configuration file, in browse mode, if it exists.

Note: When using the panel, the option to send test log output to the screen is disabled. Log output will be sent to the log data sets specified in the configuration file even if 'SCREEN' is specified for printLogOutput.

Running with Advanced Test

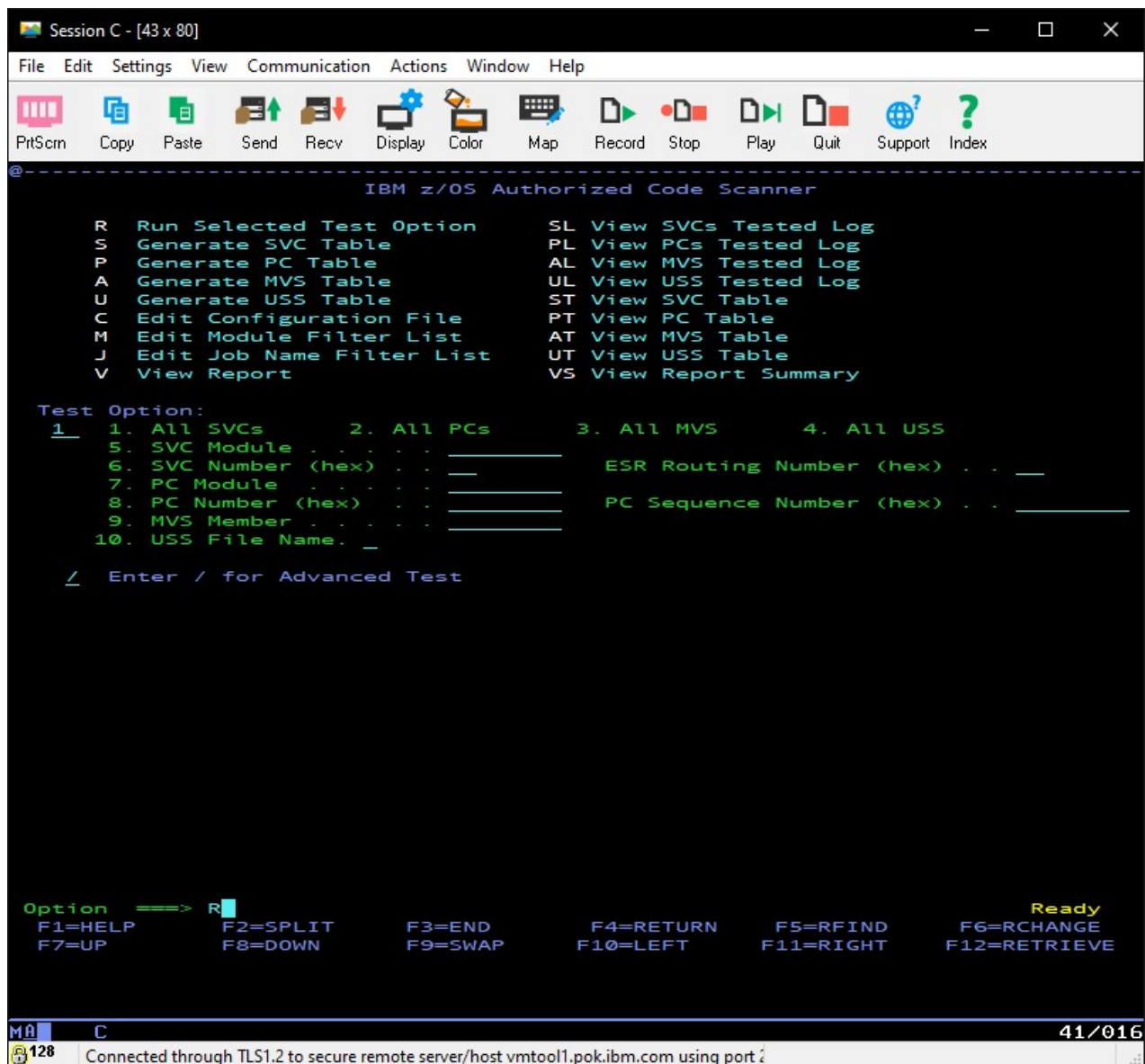


Figure 49. Advanced Test Option

Enter a '/' in the Advanced Test field to enable advanced testing, if desired. This option enables function code testing. Time until completion will increase when compared to a basic test. Advanced Test is only valid for PC and SVC tests.

Filtering Run Services with Include Lists or Exclude Lists

During a full run, services may be excluded or included based on module name or job name. In the configuration file, modFilterDSname and jobFilterDSname point to the Filter by Module Name List and Filter by Job Name List respectively. Note that only PCs can be filtered by job name. The lists are formatted as one module name or job name per line. Blank lines and line comments surrounded with '/' and '*' will be ignored.

z/OS UNIX path names and AC(1) Dataset or member names may also be used the Filter by Module Name list. DSN entries must be fully qualified and may be up to 44 characters in length and z/OS UNIX path names must be fully qualified, starting with a '/'.

Note: The exclude and include lists determine which modules are individually tested or skipped. If a module in the exclude list is called by a separate module not on the exclude list, it may still appear in the output file if a potential vulnerability within it is hit during the calling module's test.

Exclude by module example, runs all services except those with the module names listed:

```

/*****
/* Excluded Modules */
*****/
IGVSLIST
IGVLOCP
IEAVRT04
SYS1.VTAM44.VTAMLIB.TEMP
ITVDIV
ASRSERV
/bin/bpxwrtso
IEAVLSEX
IEAVESTA
IGVVST0R

```

Figure 50. Example of excluded modules.

Include by job name example, runs only services with the job names listed:

```

/*****
/* Included Job Names */
*****/
PCAUTH
GRS
IOSAS

```

Figure 51. Example of included job names.

Reading Generated Service Tables

Note: Not all PCs and SVCs have associated module names.

Note: Periods (.) may replace unprintable characters in eyecatchers.

Note: A dash may be present in place of information, such as module names, that is inapplicable or unavailable.

Note: If you are viewing the tables from the zACS GUI, some fields may not be displayed for simplicity.

Module names can be found in the 12th column of the *userid.ZACS.SVCNUM* file.

| SVC | ESR | AM | TY | A | E | P | S | R | Locks | EntryPnt | ModName | ModAddr | Offset | Eyecatchers |
|-----|-----|----|----|---|---|---|---|---|-------|----------|----------|----------|----------|---------------------------------|
| 00 | -- | 31 | 1 | N | N | N | N | N | YNNNN | 00FF17B8 | IECVEXCP | 00FF1728 | 00000090 | .IECVEXCP12/11/20HBB77D0 x!.-j. |
| 01 | -- | 31 | 1 | N | N | N | N | N | YNNNN | 00FEE74 | IEAVEWAT | 00FEEC30 | 00000244 | .^8^8'8^4m0^4v6".&0^Ux4.N{°...p |
| 02 | -- | 31 | 1 | N | N | N | N | N | YNNNN | 01222440 | IEAVEPST | 01222428 | 00000018 | .IEAVEPST04/22/20HBB77D0 {0...u |
| 03 | -- | 31 | 1 | N | N | N | N | Y | YNNNN | 0121D1A0 | IGC003 | 0121D1A0 | 00000000 | x4...IGC003 20168 HBB77D0..A. |

Figure 52. SVC Table Example

The following list includes all the information that is found in the generated SVC Table:

Column heading

Column heading definition

SVC

SVC number

ESR

ESR number

AM

Amode

TY

Type

A
APF Authorized

E
Extended

P
Preemptive

S
Assisted

R
Armode

Locks
Locks

EntryPnt
Entry pointer

ModName
Module name

ModAddr
Module address

Offset
SVC offset within module

Eyecatchers
SVC eyecatcher

| PCNum | SeqNum | AKM | ASID | S | K | EntryPnt | ModName | ModAddr | Offset | JobName | Eyecatchers |
|----------|----------|------|------|---|---|----------|----------|----------|----------|---------|----------------------|
| 00000122 | 00000000 | 0000 | 0007 | S | 0 | 0738B630 | ISGNLPA | 0738B000 | 00000630 | GRS | .ISGGELF 05/27/09 HB |
| 00000123 | 00000000 | 0000 | 0007 | S | 0 | 0933C5A8 | ISGNPVT | 09300000 | 0003C5A8 | GRS | .ISGLECA 11/30/18HBB |
| 00000200 | 00000000 | C000 | 0014 | S | 0 | 093077A8 | IEFHB410 | 093077A8 | 00000000 | ALLOCAS | .{â0}.IEFHB41004/02/ |
| 00000201 | 00000000 | C000 | 0014 | S | 0 | 09307A28 | IEFHB420 | 09307A28 | 00000000 | ALLOCAS | x4...IEFHB420 2016. |

Figure 53. PC Table example

The following list includes all the information that is found in the generated PC Table:

Column heading
Column heading definition

PCNum
PC number

SeqNum
Sequence number

AKM
AKM

ASID
ASID

S
State (Problem or Supervisor)

K
Key

EntryPnt
Entry pointer

ModName
Module name

ModAddr
Module address

Offset

PC offset within module

Eyecatchers

PC eyecatcher

| Volume | Member | PGM | CMD | TSF | PPT | LNK | Data Set |
|--------|----------|-----|-----|-----|-----|-----|------------------------|
| PPLB80 | *MISSING | - | - | - | - | - | ADLE370.V1R3M0.SCEERUN |
| ZDR25B | CSFINIT | N | N | N | 0 | N | CSF.SCSFMODE0 |
| ZDR25B | ISFHCTL | N | N | N | 4 | N | ISF.SISFLOAD |
| ABC001 | BPNTLWB2 | N | N | N | - | N | INTEG.ABC.NOTTSO.LOAD |
| ABC001 | BPNTPGM | Y | N | N | - | N | INTEG.ABC.TSO.LOAD |
| ABC001 | BPNTTSDD | Y | Y | Y | - | N | INTEG.ABC.TSO.LOAD |
| ABC001 | BPNTTSF | N | N | Y | - | N | INTEG.ABC.TSO.LOAD |
| ABC001 | BPNTTSFC | N | Y | Y | - | N | INTEG.ABC.TSO.LOAD |

Figure 54. MVS Table Example

The following list includes all the information that is found in the generated MVS Table:

Column heading**Column heading definition****Volume**

MVS Volume or *LPA* for AC(1) programs in LPA

Member

Data Set Member or *MISSING if not currently available

PGM

Whether the program is in the AUTHPGM table

CMD

Whether the program is in the AUTHCMD table

TSF

Whether the program is in the AUTHTSF table

PPT

The PPT Key to be used by the program, if one is required

LNK

The program is not explicitly APF authorized, but is in the current LNKLIST concatenation and LNKAUTH=LNKLST

Data Set

Data set name

Note: Data set names are marked as “-” for programs in LPA

Note: Only programs with a PPT key of 8, or that do not require a PPT key are testable by zACS

| A | U | G | Path and File Name |
|---|---|---|--------------------|
| - | - | - | ----- |
| Y | N | N | /bin/bpxwrtso |
| Y | N | N | /bin/IBM/BPXWRTSO |

Figure 55. z/OS UNIX Table Example

The following list includes all the information that is found in the generated z/OS UNIX Table:

Column heading**Column heading definition****A**

AC1

U

SetUIId

G

SetGId

Path and File Name

Full path and file name

Using the zACS z/OSMF External plug-in

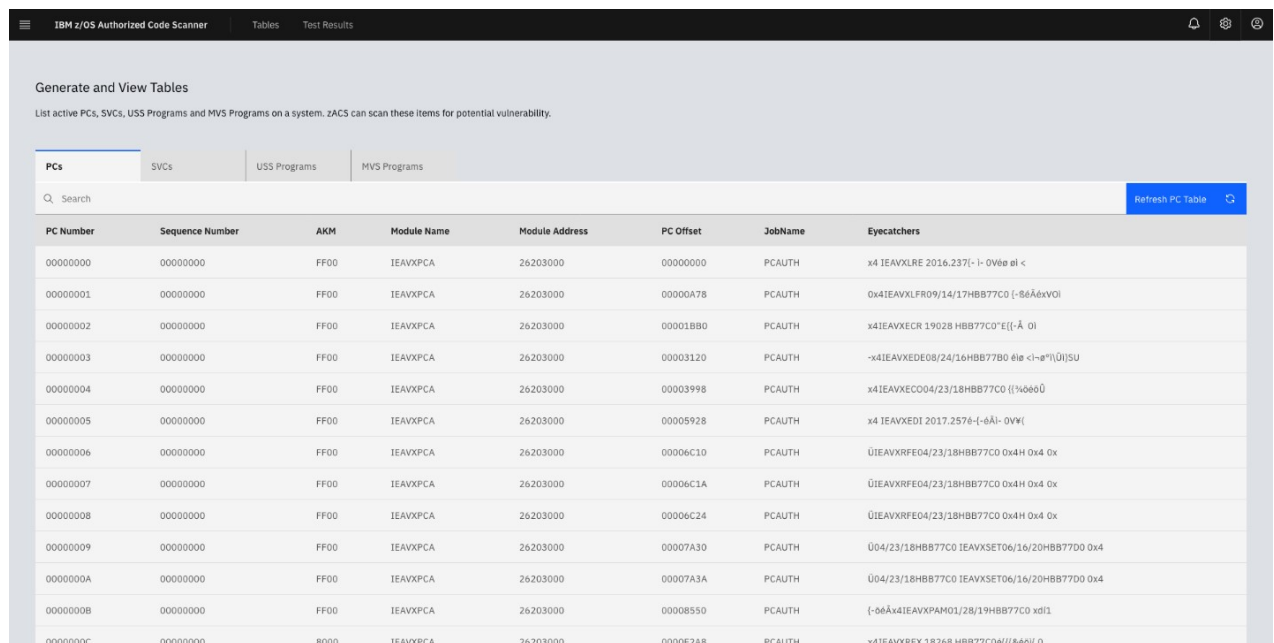
Once the zACS GUI is configured and users given the necessary access to use the zACS GUI on z/OSMF, users can log in to z/OSMF and navigate to the IBM z/OS Authorized Code Scanner. Users can then perform various functions such as:

- Generate and view PC, SVC, z/OS UNIX, and MVS tables.
- View and update the inclusion and exclusion configurations.
- Update the inclusion and exclusion data sets used for scans.
- Run potential vulnerability scans for PC, SVC, z/OS UNIX, and MVS programs.
- View potential vulnerability scan results.

Viewing the PC, SVC, z/OS UNIX, and MVS Tables

The generated tables for each type can be easily viewed on the home page of the GUI. Navigate to the home page to access the tables and explore the relevant information.

Each table has a separate tab.



| PC Number | Sequence Number | AKM | Module Name | Module Address | PC Offset | JobName | Eyecatchers |
|-----------|-----------------|------|-------------|----------------|-----------|---------|--|
| 00000000 | 00000000 | FF00 | IEAVXPCA | 26203000 | 00000000 | PCAUTH | x4 IEAVXLRE 2016.237(- i- 0V8e el < |
| 00000001 | 00000000 | FF00 | IEAVXPCA | 26203000 | 00000A78 | PCAUTH | 0x4IEAVXLFRO9/14/17HBB77C0 {-86ÂexVOI |
| 00000002 | 00000000 | FF00 | IEAVXPCA | 26203000 | 00001B80 | PCAUTH | x4IEAVXECR 19028 HBB77C0'E[{-Ã 0I |
| 00000003 | 00000000 | FF00 | IEAVXPCA | 26203000 | 00003120 | PCAUTH | -x4IEAVXEDE08/24/16HBB77B0 êle <{-a"()Ü()SU |
| 00000004 | 00000000 | FF00 | IEAVXPCA | 26203000 | 00003998 | PCAUTH | x4IEAVXECO04/23/18HBB77C0 ({%660Û |
| 00000005 | 00000000 | FF00 | IEAVXPCA | 26203000 | 00005928 | PCAUTH | x4 IEAVXEDI 2017.2576-{-6Ai- 0VY(|
| 00000006 | 00000000 | FF00 | IEAVXPCA | 26203000 | 00006C10 | PCAUTH | ÛIEAVXRFE04/23/18HBB77C0 0x4H 0x4 0x |
| 00000007 | 00000000 | FF00 | IEAVXPCA | 26203000 | 00006C1A | PCAUTH | ÛIEAVXRFE04/23/18HBB77C0 0x4H 0x4 0x |
| 00000008 | 00000000 | FF00 | IEAVXPCA | 26203000 | 00006C24 | PCAUTH | ÛIEAVXRFE04/23/18HBB77C0 0x4H 0x4 0x |
| 00000009 | 00000000 | FF00 | IEAVXPCA | 26203000 | 00007A30 | PCAUTH | Û04/23/18HBB77C0 IEAVXSET06/16/20HBB77D0 0x4 |
| 0000000A | 00000000 | FF00 | IEAVXPCA | 26203000 | 00007A3A | PCAUTH | Û04/23/18HBB77C0 IEAVXSET06/16/20HBB77D0 0x4 |
| 0000000B | 00000000 | FF00 | IEAVXPCA | 26203000 | 00008550 | PCAUTH | {-06Âx4IEAVXPAM01/28/19HBB77C0 xdlI |
| 0000000C | 00000000 | 8000 | IEAVXPCA | 26203000 | 0000E2A8 | PCAUTH | x4IEAVXREX 18268 HBB77C08{({660i 0 |

Figure 56. GUI PC Table View

IBM z/OS Authorized Code Scanner

Tables

Test Results

Generate and View Tables

List active PCs, SVCs, USS Programs and MV5 Programs on a system. zACS can scan these items for potential vulnerability.

PCs

SVCs

USS Programs

MV5 Programs

Q Search

Refresh SVC Table

| SVC Number | ESR Number | Module Name | APP | Module Address | SVC Offset | Eyecatchers |
|------------|------------|-------------|-----|----------------|------------|--|
| 00 | -- | IECVEXCP | N | 00FF4160 | 00000090 | IECVEXCP12/11/20HBB77D0 xC-j&xd *x4 jC |
| 01 | -- | IEAVEWAT | N | 00FF1930 | 00000244 | IEAVEWAT02/04/21HBB77D0 *{[{ { laC&ø |
| 02 | -- | IEAVEPST | N | 014DC8B0 | 00000018 | IEAVEPST04/22/20HBB77D0 {Øul-ø%)-~ø)-~k-~\ |
| 03 | -- | IGC003 | N | 01158B28 | 00000000 | x4 IGC003 22210 U308%00A[- µ 0[- |
| 04 | -- | IGVVM24 | N | 01558688 | 0000001A | IGVVM2402/25/21HBB77D0 µl0øl00ø*U0& {* |
| 05 | -- | IGVVM24 | N | 01558688 | 0000001A | IGVVM2402/25/21HBB77D0 µl0øl00ø*U0& {* |
| 06 | -- | CSVLINK | N | 01474568 | 00000000 | x4CSVLINK 11/18/16HBB77B0 l{Åµl-ø%)-~ø)-~\ |
| 07 | -- | CSVXCTL | N | 014769C8 | 00000000 | x4CSVXCTL 04/23/20HBB77D0 l{Åøl-ø%)-~ø)-~\ |
| 08 | -- | CSVLOAD | N | 01474840 | 00000000 | x4CSVLOAD 12/06/17HBB77C0 l{Ål-ø%)-~ø)-~\ |
| 09 | -- | CSVDELET | N | 01469C38 | 00000000 | x4CSVDELET12/06/17HBB77C0 l{Ål-ø%)-~ø)-~\ |
| 0A | -- | IGVVM24 | N | 01558688 | 0000138E | IGVVM2402/25/21HBB77D0 µl0øl00ø*U0& {* |
| 0B | -- | IGC0001A | N | 0297DC60 | 00000000 | 0&00IGC0001A 14012 HBB77A0 *~&&ø&&É&l&k& |
| 0C | -- | CSVSYNCH | N | 014752F0 | 00000000 | x4CSVSYNCH10/04/17HBB77C0 l{Ñl-ø%)-~ø)-~\ |

Figure 57. GUI SVC Table View

IBM z/OS Authorized Code Scanner

Tables

Test Results

Generate and View Tables

List active PCs, SVCs, USS Programs and MV5 Programs on a system. zACS can scan these items for potential vulnerability.

PCs

SVCs

USS Programs

MV5 Programs

Q Search

Refresh USS Table

| AC=1 | SetUId | SetGId | Path and File Name |
|------|--------|--------|--------------------|
| Y | N | N | /bin/IBM/BPXWRTSO |
| N | Y | N | /bin/IBM/FONTLINC |
| N | Y | N | /bin/IBM/FONTLOUT |
| N | Y | N | /bin/IBM/FSUMSAT |
| N | Y | N | /bin/IBM/FSUMSCRT |
| N | N | Y | /bin/IBM/FSUMSMEG |
| N | Y | N | /bin/IBM/FSUMSNWG |
| N | Y | N | /bin/IBM/FSUMLOGN |
| N | Y | N | /bin/IBM/FSUMSRML |
| N | N | Y | /bin/IBM/FSUMSTLK |
| N | Y | N | /bin/IBM/FSUMSUCP |
| N | Y | N | /bin/IBM/FSUMSUST |
| N | Y | N | /bin/IBM/FSUMSUUE |

Figure 58. GUI z/OS UNIX Table View

| Volume | Member | AUTHPGM | AUTHCMD | AUTHTSF | PPT | LNKST ONLY | Data Set |
|--------|-----------|---------|---------|---------|-----|------------|------------------------|
| PPLB80 | *MISSING | - | - | - | - | - | ADLE370.V1R3M0.SCEERUN |
| BPXLK1 | *MISSING | - | - | - | - | - | ALISONW.LINKLIB1 |
| PRODAL | *MISSING | - | - | - | - | - | AOC110.SAOFMOD1 |
| CTTPAK | \$EFACTRT | N | N | N | - | N | ARTMVS.EXIT.S.LOADLIB |
| CTTPAK | ARTIMER | N | N | N | - | N | ARTMVS.EXIT.S.LOADLIB |
| CTTPAK | ARTWTO | N | N | N | - | N | ARTMVS.EXIT.S.LOADLIB |
| CTTPAK | ISSUECMD | N | N | N | - | N | ARTMVS.EXIT.S.LOADLIB |
| CTTPAK | ISSUEOLD | N | N | N | - | N | ARTMVS.EXIT.S.LOADLIB |
| CTTPAK | MPFJES5L | N | N | N | - | N | ARTMVS.EXIT.S.LOADLIB |
| CTTPAK | MPFJES6P | N | N | N | - | N | ARTMVS.EXIT.S.LOADLIB |
| CTTPAK | MPFJES62 | N | N | N | - | N | ARTMVS.EXIT.S.LOADLIB |
| CTTPAK | OLDACTRT | N | N | N | - | N | ARTMVS.EXIT.S.LOADLIB |

Figure 59. GUI MVS Table View

Generating the PC, SVC, z/OS UNIX, and MVS Tables

To generate or refresh the tables, click the **Generate** or **Refresh** on the GUI. Selecting either option initiates the process of generating and updating the PC, SVC, z/OS UNIX, and MVS tables with the latest data. The tables must be generated before scans of a specific service type can be ran.

If the table is empty, click **Generate Table**.

| PC Number | Sequence Number | AKM | Module Name | Module Address | PC Offset | JobName | Eyecatchers |
|-------------------|-----------------|-----|-------------|----------------|-----------|---------|-------------|
| No Data Available | | | | | | | |

Figure 60. GUI PC Empty Table View

| PC Number | Sequence Number | AKM | Module Name | Module Address | PC Offset | JobName | Eyecatchers |
|-----------|-----------------|------|-------------|----------------|-----------|---------|--|
| 00000000 | 00000000 | FF00 | IEAVXPCA | 26203000 | 00000000 | PCAUTH | x4 IEAVXLR 2016.237(-l- OV6e pl < |
| 00000001 | 00000000 | FF00 | IEAVXPCA | 26203000 | 00000A78 | PCAUTH | 0x4IEAVXLFRO9/14/17HBB77C0 (-B6Å&xVOI |
| 00000002 | 00000000 | FF00 | IEAVXPCA | 26203000 | 00001BB0 | PCAUTH | x4IEAVXECR 19028 HBB77C0"e((-Å 0l |
| 00000003 | 00000000 | FF00 | IEAVXPCA | 26203000 | 00003120 | PCAUTH | -x4IEAVXEDE08/24/18HBB77B0 el8 cl-8P1)Û)5U |
| 00000004 | 00000000 | FF00 | IEAVXPCA | 26203000 | 00003998 | PCAUTH | x4IEAVXEC004/23/18HBB77C0 ({%8e0Û |
| 00000005 | 00000000 | FF00 | IEAVXPCA | 26203000 | 00005928 | PCAUTH | x4 IEAVXEDI 2017.2576(-6Å)- OVV(|
| 00000006 | 00000000 | FF00 | IEAVXPCA | 26203000 | 00006C10 | PCAUTH | ÛIEAVXRFE04/23/18HBB77C0 0x4H 0x4 0x |
| 00000007 | 00000000 | FF00 | IEAVXPCA | 26203000 | 00006C1A | PCAUTH | ÛIEAVXRFE04/23/18HBB77C0 0x4H 0x4 0x |
| 00000008 | 00000000 | FF00 | IEAVXPCA | 26203000 | 00006C24 | PCAUTH | ÛIEAVXRFE04/23/18HBB77C0 0x4H 0x4 0x |
| 00000009 | 00000000 | FF00 | IEAVXPCA | 26203000 | 00007A30 | PCAUTH | Û04/23/18HBB77C0 IEAVXSET06/16/20HBB77D0 0x4 |
| 0000000A | 00000000 | FF00 | IEAVXPCA | 26203000 | 00007A3A | PCAUTH | Û04/23/18HBB77C0 IEAVXSET06/16/20HBB77D0 0x4 |
| 0000000B | 00000000 | FF00 | IEAVXPCA | 26203000 | 00008550 | PCAUTH | (-06Åx4IEAVXPAM01/28/19HBB77C0 xdl1 |
| 0000000C | 00000000 | 8000 | IEAVXPCA | 26203000 | 0000E2A8 | PCAUTH | x4IEAVXREX 18268 HBB77C06({{840i(0 |

Figure 61. GUI PC Generation Finish View

Modifying Inclusion & Exclusion Lists

The GUI provides the flexibility to configure settings that are related to inclusion and exclusion for zACS scans. You can access the configuration options within the GUI and modify them according to your specific requirements. These settings determine the scope of the scans and which elements are included or excluded from analysis. These settings apply only to a full table scan, when not using optional parameters.

To access these settings from the GUI, click the gear icon in the upper right of the page.



Figure 62. GUI Setting Icon

You are then presented with the setting page with all the options from the `userid.ZACS.CONFIG` file.

User configuration

Configure zACS settings

zACS installation high level qualifier

SYS1.BPN

Configuration File Name

IBMUSER.ZACS.CONFIG

[Edit this configuration file](#)

Module Filter Data Set Name

IBMUSER.ZACS.EXCLUDE.MOD1NAME

The data set name is not valid or does not exist

Job Filter Data Set Name

IBMUSER.ZACS.EXCLUDE.JOB1NAME

The data set name is not valid or does not exist

Cancel

Submit

Figure 63. GUI Setting Window

Pressing **Submit** updates the high-level qualifier.

Click **Edit this configuration file** to open the configuration file for editing.

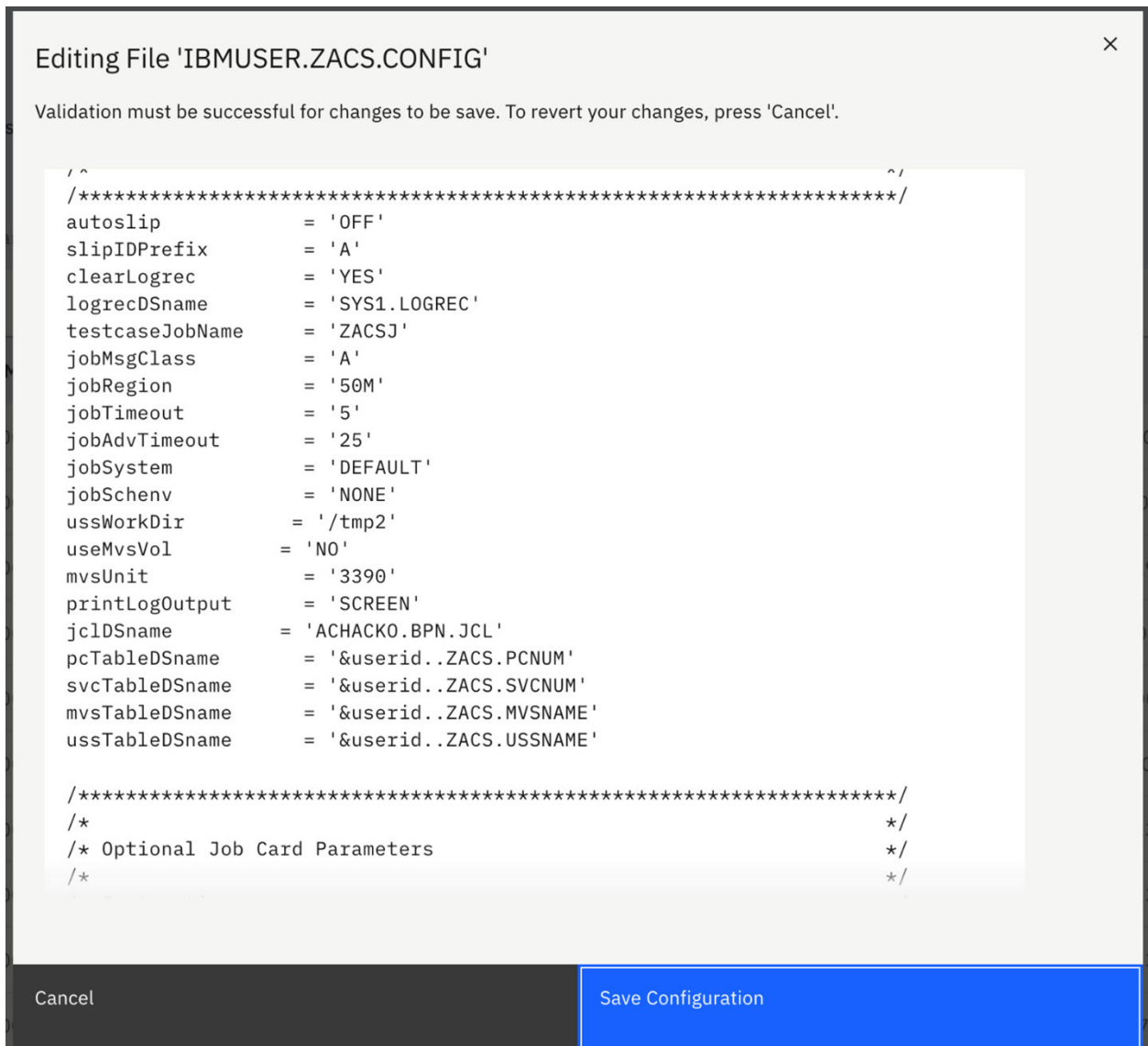


Figure 64. Editing the configuration file from the GUI

After your changes are made, click **Save Configuration** to run validation on the changes. The file is saved if validation is successful. If you want to revert the file to the previous saved state, click **Cancel** to close without saving. A confirmation window appears .

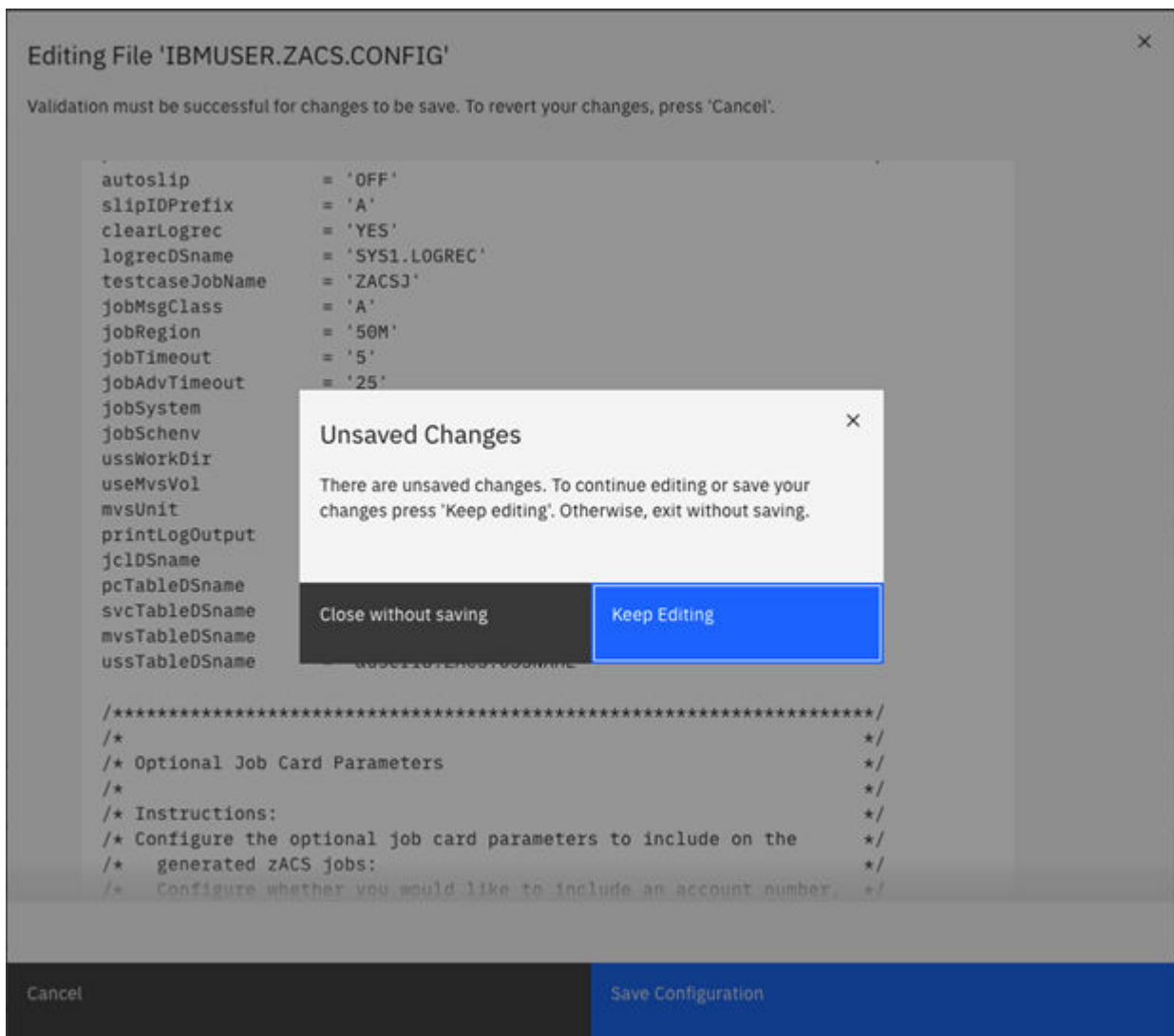


Figure 65. Closing the configuration file without saving

Filtering


Filter data sets are pulled from the configuration file and displayed on the main settings window. If the data sets exist the option to view or edit the data set appears. You must have authority to view or edit the data set to do so.

User configuration ×


Configure zACS settings


zACS installation high level qualifier

Configuration File Name


[Edit this configuration file](#) 

Module Filter Data Set Name



[Edit this Filter Data set](#) 

Job Filter Data Set Name




[Edit this Filter Data set](#) 

Figure 66. GUI Setting Valid Data Set

Clicking **View Dataset** opens a separate window where you can edit the contents of the data set. Click **Submit** to save your changes to the data set or **Cancel** to return to the previous screen without saving.

View Dataset

Module Filter Data Set Name

IBMUSER.ZACS.EXCLUDE.MODNAME

Data Set Information

```

/*****
/* Excluded Modules */
*****/
IGVSLIST
IGVLOCP
IEAVRT04
SYS1.VTAM44.VTAMLIB.TEMP
ITVDIV ASRSERV
/bin/bpxwrtso/bin/bpxwrtso/bin/bpxwrtso
IEAVLSEX
IEAVESTA
IGVVSTOR

```

Cancel

Submit

Figure 67. GUI Setting View Exclusion/Inclusion Data Set Window

Running zACS Scans

With the GUI, you can initiate zACS scans. Go to the test results page and locate **Scan Now** in the **Run Scan** section of the page.

IBM z/OS Authorized Code Scanner

Tables

Test Results

Test and View Scan Results

Run scan and view the results for PCs, SVCs, USS Programs and MVS Programs.

PC Results

7/25/2023 - 7/25/2023

0

SVC Results

7/25/2023 - 7/25/2023

0

USS Results

7/25/2023 - 7/25/2023

0

MVS Results

7/25/2023 - 7/25/2023

0

Run Scan

Test PCs, SVCs, MVS Programs and USS Programs for potential vulnerabilities.

Scan Now

View Raw Report

Test Result for PCs

Test Result for SVCs

Test Result for USS Programs

Test Result for MVS Programs

| PC Number | Timestamp | Test Type | Last Test Result | CVSS Score | Vulnerability Type |
|-------------------------------------|-----------|-----------|------------------|------------|--------------------|
| No Test Data | | | | | |
| Click Scan Now to begin testing PCs | | | | | |

Figure 68. GUI Empty Test Results Page

Clicking **Scan Now** brings up a prompt to select the type of services you want to test, PCs, SVCs, z/OS UNIX programs, or MVS programs. After selecting the type of test, click **Next**.

Scan options

Run a new scan

Please select the type of test you would like to run:

- ☒ Run PC Test
- ☐ Run SVC Test
- ☐ Run USS Program Test
- ☐ Run MVS Program Test

Cancel Next

Figure 69. GUI Run Scan Options

To run a scan against the entire table, select the first option. Programs that are excluded by an inclusion or exclusion list are skipped.

Scan options

Run a new scan

Please select the type of test you would like to run:

- ☒ All PCs
- ☐ PC Module
- ☐ PC Number
- ☐ Enable Advanced Testing ⓘ

Previous Run Test

Figure 70. GUI Run ALL PC Scan

Scan options

Run a new scan

Please select the type of test you would like to run:

☒ All SVCs

☐ SVC Module

☐ SVC Number

☐ Enable Advanced Testing ⓘ

Previous

Run Test

Figure 71. GUI Run ALL SVC Scan

Scan options

Run a new scan

Please select the type of test you would like to run:

☒ All USS Programs

☐ USS File

Previous

Run Test

Figure 72. GUI Run ALL z/OS UNIX Scan

Scan options

Run a new scan

Please select the type of test you would like to run:

☒ All MVS Programs

☐ MVS Member

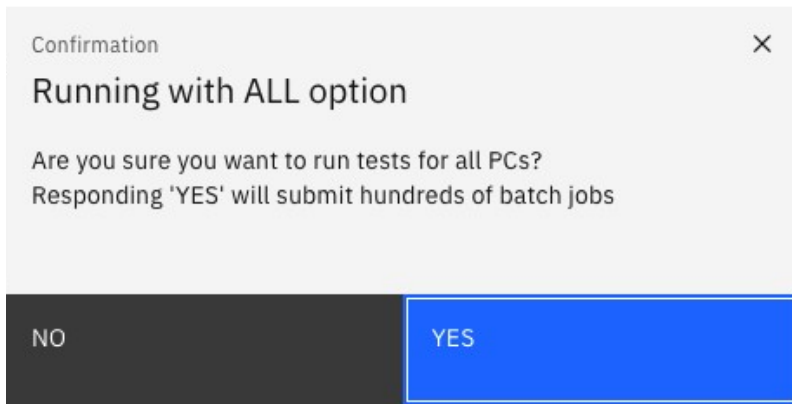
Previous

Run Test

Figure 73. GUI Run ALL MVS Scan

Click **Run Test** to trigger the execution of the zACS scans to identify potential integrity vulnerabilities in your system.

When starting a full run, potentially hundreds of JCL jobs are submitted. When the **ALL** option is selected with PC, SVC, z/OS UNIX, and MVS, a confirmation window pops up to prevent accidentally starting a large run. To continue with the run, click **YES**. To cancel click **NO**, doing so takes you back to previous screen.



Confirmation

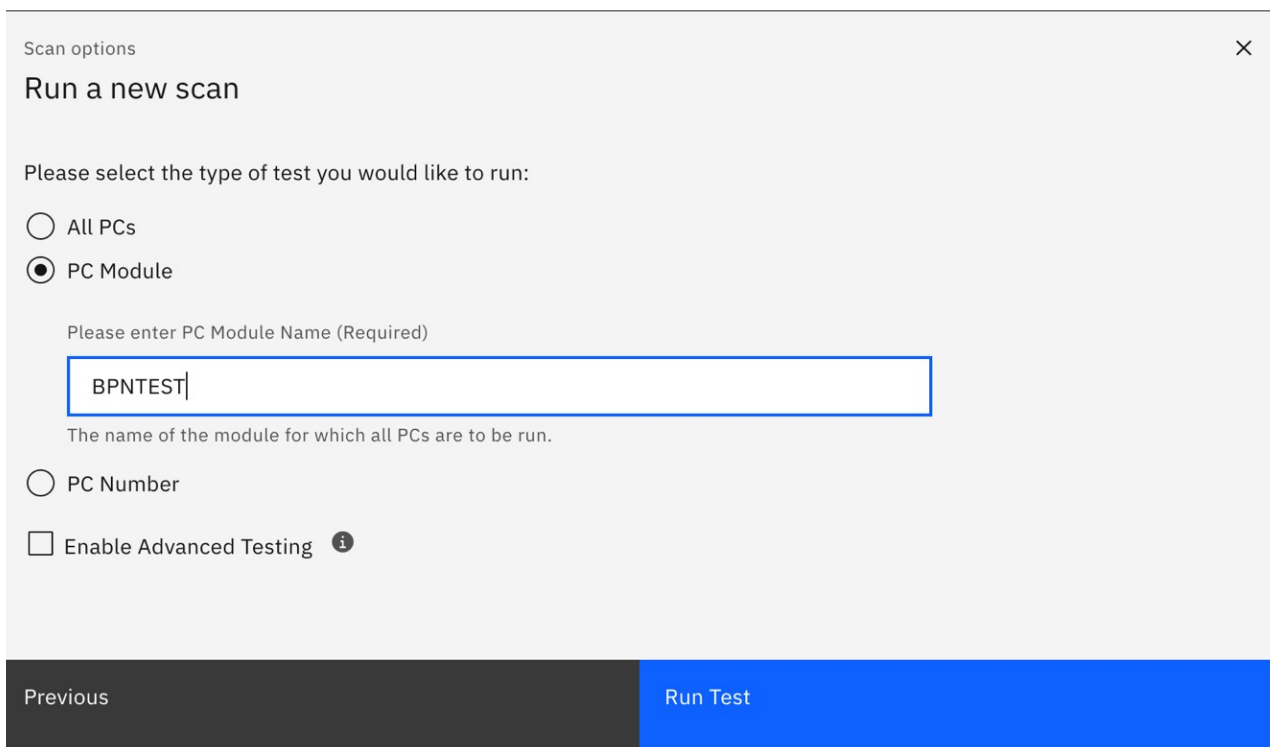
Running with ALL option

Are you sure you want to run tests for all PCs?
Responding 'YES' will submit hundreds of batch jobs

NO YES

Figure 74. GUI Confirmation Prompt

To narrow the scope of the test with optional parameters, the scan by module name and scan by number options can be used. See [“Optional Parameters”](#) on page 26 for details.



Scan options

Run a new scan

Please select the type of test you would like to run:

☐ All PCs

☒ PC Module

Please enter PC Module Name (Required)

BPNTEST

The name of the module for which all PCs are to be run.

☐ PC Number

☐ Enable Advanced Testing ⓘ

Previous Run Test

Figure 75. GUI Running a single PC module BPNTEST

Scan options
×

Run a new scan

Please select the type of test you would like to run:

☐ All SVCs
☐ SVC Module
☒ SVC Number

Please enter a SVC Number (Required)

6D

Used to run just a single SVC, specified in hexadecimal.

Please enter ESR Routing Number (Optional)

B

The hexadecimal routing number to run a single Extended SVC.

☐ Enable Advanced Testing ⓘ

Previous
Run Test

Figure 76. GUI Running a single SVC with routing number 109 and ESR 11

Scan options
×

Run a new scan

Please select the type of test you would like to run:

☐ All USS Programs
☒ USS File

Please enter USS Path and File Name (Required)

/usr/lpp/bpntest

Type the full path, beginning with a '/', and file name to be run.

Previous
Run Test

Figure 77. GUI Running a z/OS UNIX file with path specified

For PCs and SVCs, there is an option to run an advanced test. Selecting this option causes increased time until completion compared to a basic test.

Scan options

Run a new scan

Please select the type of test you would like to run:

☒ All PCs

☐ PC Module

☐ PC Number

☒ Enable Advanced Testing

Enables additional testing. Time until completion will increase.

Previous

Run Test

Figure 78. GUI Run Scan with Advanced Testing

Viewing Scan Results

The scan results are organized such that the results of each type of service are in their own tab. Circle graphs are included that depict the historical pass, fail, and incomplete results for each type of testable service, at a glance. If multiple, unique, potential vulnerabilities are detected in a single service, the count reflects the number of unique potential vulnerabilities found. These results are cumulative from the initialization of the zACS started task. Restarting the started task clears the results and counts. Also, the user can filter the data based on specific dates to analyze the results of past scans or focus on recent runs. The circle graphs update to reflect the dates selected.

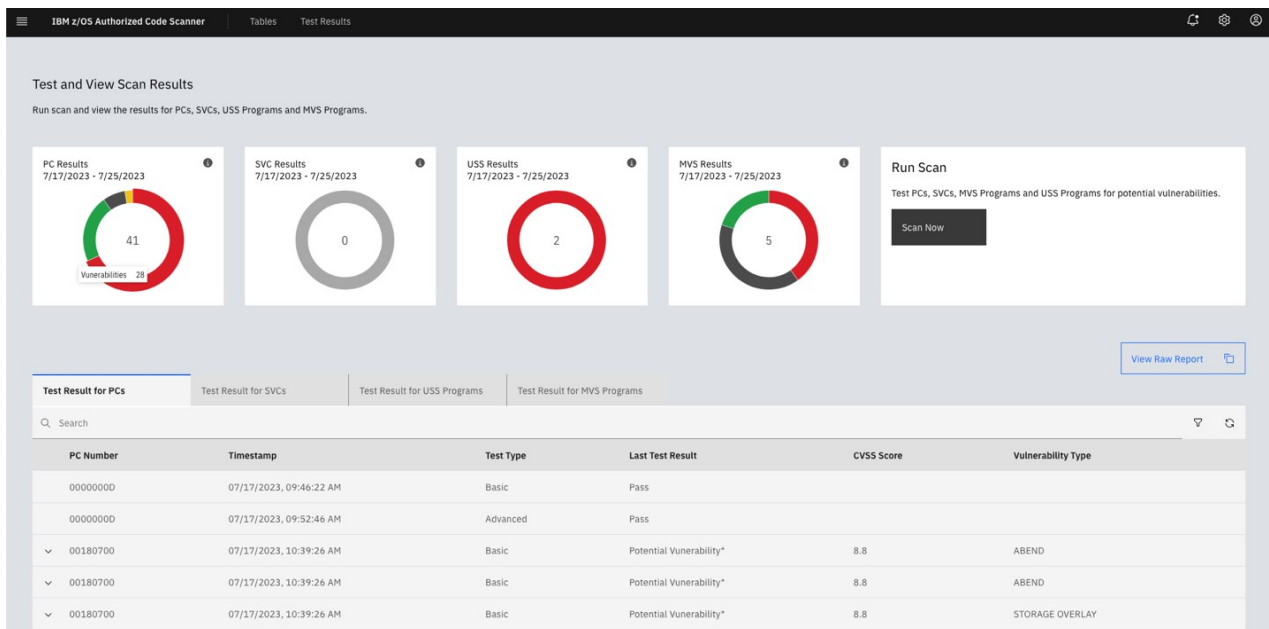


Figure 79. GUI Test Results Page Filled

The results are presented in a clear and organized manner in the table, displaying key information such as pass and fail results of each scanned service, along with the type of potential vulnerability detected and CVSS score. Rows can be expanded by using the arrow on the left side to display further details.

| IBM z/OS Authorized Code Scanner | | | | | |
|---|-------------------------|---|------------------------------|------------------------------|--------------------|
| Test Result for PCs | | Test Result for SVCs | Test Result for USS Programs | Test Result for MVS Programs | |
| PC Number | Timestamp | Test Type | Last Test Result | CVSS Score | Vulnerability Type |
| 0000000D | 07/17/2023, 09:46:22 AM | Basic | Pass | | |
| 0000000D | 07/17/2023, 09:52:46 AM | Advanced | Pass | | |
| 00180700 | 07/17/2023, 10:39:26 AM | Basic | Potential Vulnerability* | 8.8 | ABEND |
| Module Name BPNTTEST | Offset 00001A9A | ABEND Code 0C4000 | Reason Code 00000011 | | |
| 00180700 | 07/17/2023, 10:39:26 AM | Basic | Potential Vulnerability* | 8.8 | ABEND |
| 00180700 | 07/17/2023, 10:39:26 AM | Basic | Potential Vulnerability* | 8.8 | STORAGE OVERLAY |
| Key 0 Parameter data before | | Key 0 Parameter data after | | | |
| 00664200: 003FF7FF 003FF7FF 003FF7FF 003FF7FF | | 00664200: 00000020 003FF7FF 003FF7FF 003FF7FF | | | |
| 00664210: 003FF7FF 003FF7FF 003FF7FF 003FF7FF | | 00664210: 003FF7FF 003FF7FF 003FF7FF 003FF7FF | | | |
| 00664220: 003FF7FF 003FF7FF 003FF7FF 003FF7FF | | 00664220: 003FF7FF 003FF7FF 003FF7FF 003FF7FF | | | |
| 00664230: 003FF7FF 003FF7FF 003FF7FF 003FF7FF | | 00664230: 003FF7FF 003FF7FF 003FF7FF 003FF7FF | | | |
| 00664240: 003FF7FF 003FF7FF 003FF7FF 003FF7FF | | 00664240: 003FF7FF 003FF7FF 003FF7FF 003FF7FF | | | |
| 00180800 | 07/17/2023, 10:39:31 AM | Basic | Potential Vulnerability* | 8.8 | ABEND |
| 00180800 | 07/17/2023, 10:39:31 AM | Basic | Potential Vulnerability* | 8.8 | ABEND |
| 00180800 | 07/17/2023, 10:39:31 AM | Basic | Potential Vulnerability* | 8.8 | STORAGE OVERLAY |
| 00180801 | 07/17/2023, 10:39:35 AM | Basic | Potential Vulnerability* | 6.5 | ABEND |
| 00180802 | 07/17/2023, 10:39:42 AM | Basic | Pass | | |

Figure 80. GUI Test Results Table Expanded rows for more details

In each table, an option exists to filter the data based on specific dates to analyze the results of past scans or focus on recent runs. The circle graphs update to reflect the dates selected.

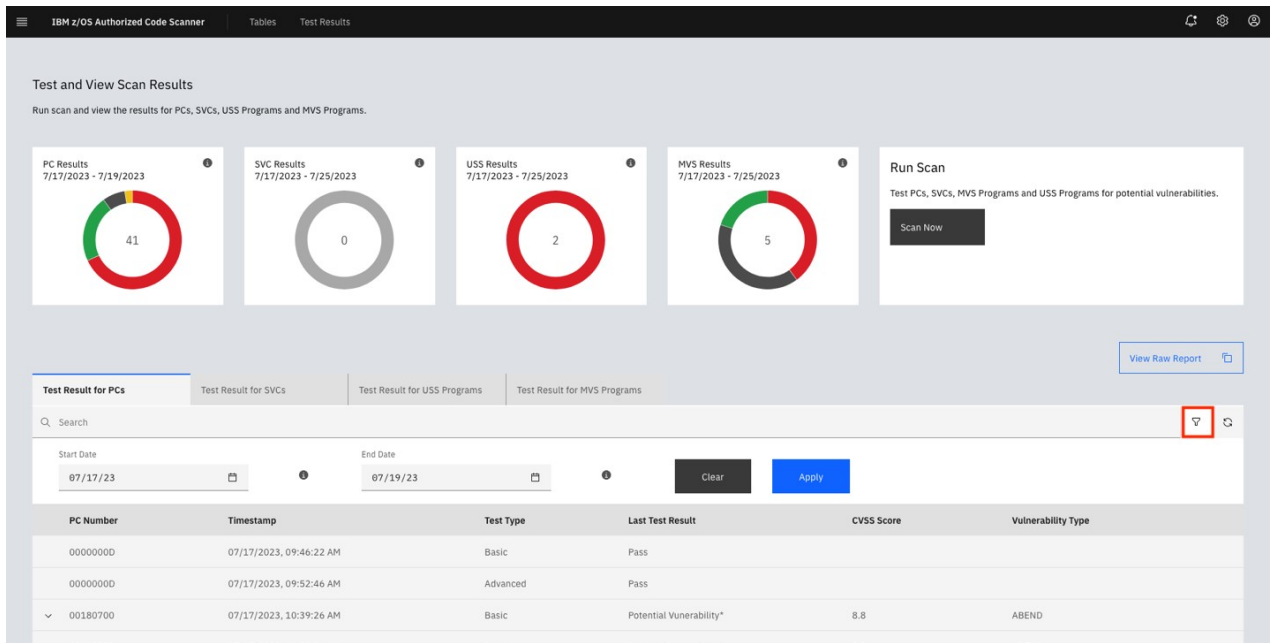


Figure 81. GUI Test Results filter by timestamps

The refresh icon refreshes all the tables and can be used to get new data.

| <div> <div>Test Result for PCs</div> <div>Test Result for SVCs</div> <div>Test Result for USS Programs</div> <div>Test Result for MVS Programs</div> </div> <div>View Raw Report</div> | | | | | |
|--|-------------------------|-----------|--------------------------|------------|--------------------|
| <div> <div>Q Search</div> <div> <div></div> <div></div> </div> </div> | | | | | |
| Member | Timestamp | Test Type | Last Test Result | CVSS Score | Vulnerability Type |
| BPNTAC10 | 07/17/2023, 09:57:41 AM | Basic | Potential Vulnerability* | 8.8 | ABEND |
| BPNTAC10 | 07/17/2023, 09:57:41 AM | Basic | Potential Vulnerability* | 8.8 | ABEND |
| BPNGMAIN | 07/17/2023, 11:07:37 AM | Basic | Pass | | |
| BPNGSVCN | 07/17/2023, 11:07:59 AM | Basic | Not Auth | | |
| BPNGSVCA | 07/17/2023, 11:09:59 AM | Basic | Not Found | | |
| <div> <div>Items per page: 100</div> <div>1-5 of 5 items</div> <div>1 of 1 page</div> </div> | | | | | |

Figure 82. GUI Test Results Refresh Icon

View Full Scan Output Report

With the GUI, you can view the full scan report that can include additional potential vulnerability details. To do so, click **View Raw Report** on the **Test Results** page to open a window that displays the unmodified results data set.

```

zACS scan output
Viewing Data Set IBMUSER.ZACS.TEST.OUTPUT

BASIC TEST IN PROGRESS ON 2024/02/08 AT 16:38:25 FOR PGM BPNTAC10
VOL: BPNO01 DSN: INTEG.BPN.DEVBB.TEST.SBPNLOAD
*** POTENTIAL VULNERABILITY FOUND IN PGM BPNTAC10 ***
VOL: BPNO01 DSN: INTEG.BPN.DEVBB.TEST.SBPNLOAD
ABEND COMPLETION CODE: 0C6000 REASON CODE: 00000006
PSW: 070C0000 803FF801
INSTR LEN: 02 FAILING INSTR: 0000 0000 0000 0000 0000 0000
BREAKING EVENT ADDR: 26400068 NO MODULE INFO WAS AVAILABLE
NO LIKELY EYECATCHER FOUND
HOME ASID: 002E PRIMARY ASID: 002E SECONDARY ASID: 002E
HOME JOB: BPNTAC10 PRIMARY JOB: BPNTAC10 SECONDARY JOB: BPNTAC10
CVSS: 8.8 (CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H)
SLIP SAMPLE FOR PGM BPNTAC10:
SLIP SET,ID=B000,COMP=0C6,JOBNAME=BPNTAC10,A=(SVCO,RECORD),END

|General Registers at time of error|
+-----+
| R0:00000000_00000048 | R1:00000000_00000030 |
| R2:FFFFFFFF_00FD6D18 | R3:FFFFFFFF_0001A000 |
| R4:FFFFFFFF_00000048 | R5:FFFFFFFF_A6400026 |
| R6:FFFFFFFF_0001A000 | R7:FFFFFFFF_80FD6D68 |
| R8:FFFFFFFF_A6400000 | R9:FFFFFFFF_26400000 |
| RA:FFFFFFFF_A6400000 | RB:FFFFFFFF_2640C8F8 |
| RC:FFFFFFFF_264000A0 | RD:00000000_2640C8F8 |
| RE:00000000_A640006A | RF:00000000_003FF7FF |
+-----+

*** END OF POTENTIAL VULNERABILITY REPORT ***
*** POTENTIAL VULNERABILITY FOUND IN PGM BPNTAC10 ***
VOL: BPNO01 DSN: INTEG.BPN.DEVBB.TEST.SBPNLOAD
ABEND COMPLETION CODE: 0C6000 REASON CODE: 00000006
PSW: 070C0000 80037801
INSTR LEN: 02 FAILING INSTR: 0000 0000 0000 0000 0000 0000
BREAKING EVENT ADDR: 26400068 NO MODULE INFO WAS AVAILABLE
NO LIKELY EYECATCHER FOUND
HOME ASID: 002E PRIMARY ASID: 002E SECONDARY ASID: 002E
HOME JOB: BPNTAC10 PRIMARY JOB: BPNTAC10 SECONDARY JOB: BPNTAC10
CVSS: 8.8 (CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H)
SLIP SAMPLE FOR PGM BPNTAC10:
SLIP SET,ID=B000,COMP=0C6,JOBNAME=BPNTAC10,A=(SVCO,RECORD),END

```

Figure 83. GUI View Raw Report Data Set Window

User Action Notifications

Every user action has a notification that is assigned to it. The messages for each notification are explained in Chapter 10, “Messages,” on page 95. To view notifications after they are dismissed from the screen, press the bell icon.

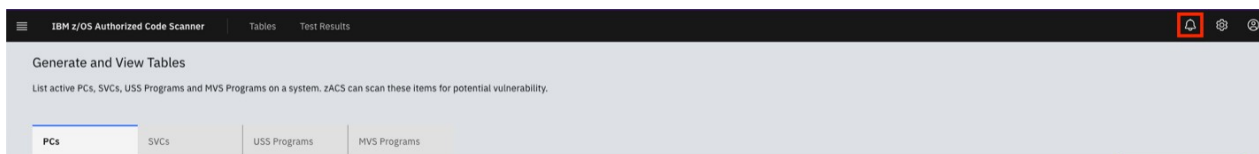


Figure 84. GUI notification Icon

Clicking the bell icon opens a window that displays the history of all notifications.

Note: A full page reload clears the historic notification list.

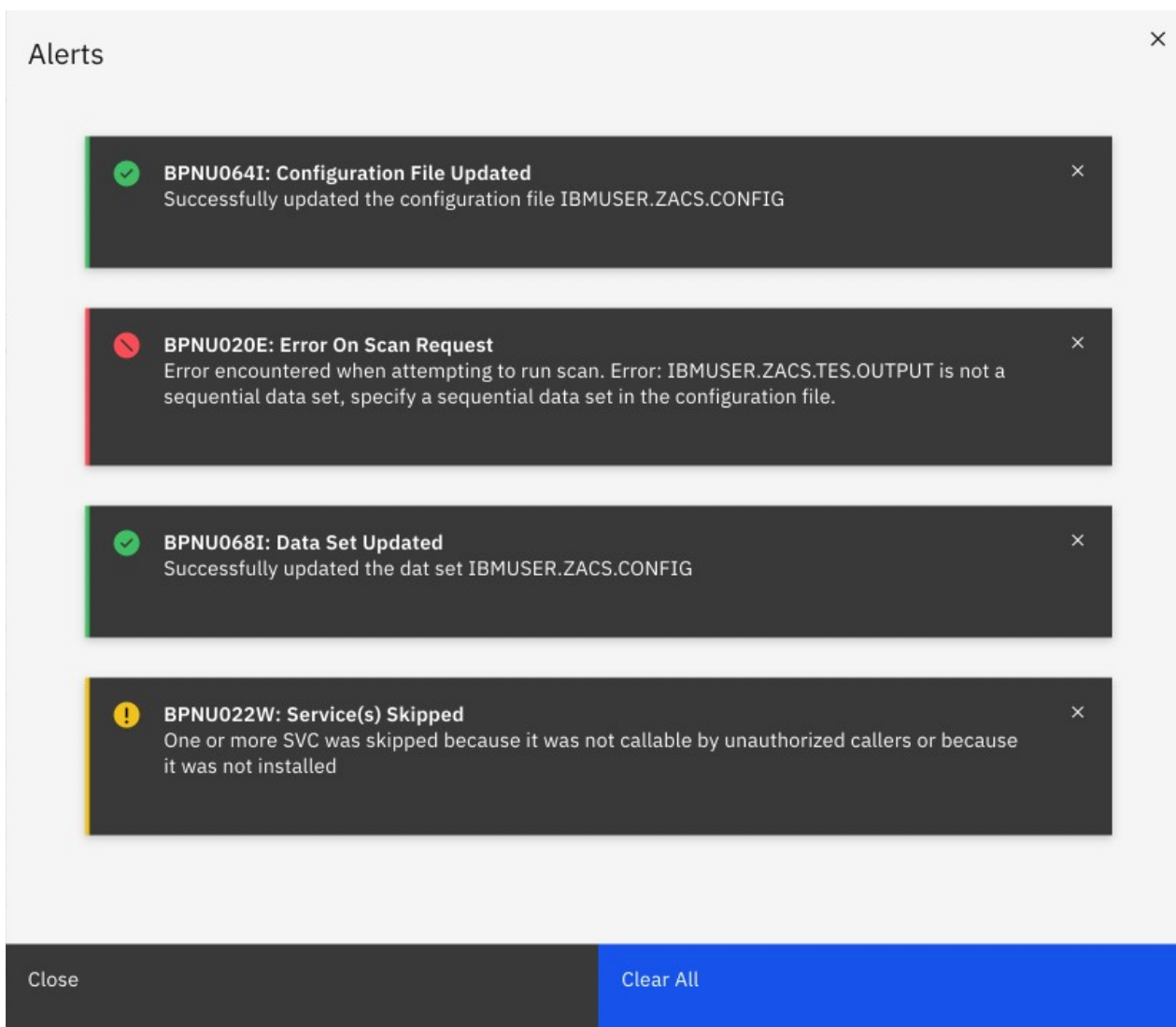


Figure 85. GUI Notification Alert Window

Chapter 5. zACS Diagnosis

After your test run is complete, you can review the results in the output file you specified for the MYOUTDD statement in the started task.

Note: Multiple tests per service are common. Some lines may be omitted if information was unobtainable.

This chapter provides examples of zACS output and explains how to interpret the results. Examples of code vulnerabilities are provided with instructions explaining how to fix them. Instructions for setting a SLIP to collect a dump for further diagnosis are also provided.

Interpreting the Potential Vulnerability Output File

Note: The following potential vulnerabilities were deliberately injected to help illustrate zACS output. The example below has a number inserted before each line for reference purposes, that do not actually appear in the generated output.

Note: Unprintable characters in eyecatchers may be replaced by periods (.)

```
01 *** POTENTIAL VULNERABILITY FOUND IN PC 00180900 ***
02 ABEND COMPLETION CODE: 0C4000 REASON CODE: 00000011
03 PSW: 070C6000 A62189DA MODULE: PVTMOD=(BPNTTEST,000019DA)
04 LIKELY EYECATCHER AT A621894C: .BPNTPC0 2021.335.é{...;ëö...i.{. 0..¥(..ëµ
05 INSTR LEN: 06 FAILING INSTR: B048 9A33 B04C D203 2000 3000
06 TRANSLATED INSTR: MVC 0(4,R2),0(R3)
07 TARGET ADDRESS CAUSING TRANSLATION EXCEPTION: 00000000_00046401
08 HOME ASID: 002C PRIMARY ASID: 0031 SECONDARY ASID: 002C
09 HOME JOB: ZACSJ PRIMARY JOB: BPNZACS SECONDARY JOB: ZACSJ
10 CVSS: 8.8 (CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H)
11 SLIP SAMPLE FOR PC 00180900:
12 SLIP SET,ID=A000,COMP=0C4,P=(BPNTTEST,000019DA),END
13 +-----+
14 |General Registers before the service |
15 +-----+
16 | R0:00000000_003FF7FF R1:00000000_003FF7FF |
17 | R2:00000000_00046000 R3:00000000_003FF7FF |
18 | R4:00000000_003FF7FF R5:00000000_003FF7FF |
19 | R6:00000000_003FF7FF R7:00000000_003FF7FF |
20 | R8:00000000_003FF7FF R9:00000000_003FF7FF |
21 | RA:00000000_003FF7FF RB:00000000_00180900 |
22 | RC:00000000_003FF7FF RD:00000000_003FF7FF |
23 | RE:00000000_003FF7FF RF:00000000_003FF7FF |
24 +-----+
25 |General Registers at time of error|
26 +-----+
27 | R0:00000000_00000000 R1:00000000_003FF7FF |
28 | R2:00000000_00046000 R3:00000000_2621D100 |
29 | R4:00000000_021690F0 R5:00000000_00000020 |
30 | R6:00000000_003FF7FF R7:00000000_003FF7FF |
31 | R8:00000000_003FF7FF R9:00000000_003FF7FF |
32 | RA:00000000_A6218980 RB:00000000_2621D0B0 |
33 | RC:00000000_26218A20 RD:00000000_2621D0B0 |
34 | RE:00000000_003FF7FF RF:00000000_00000002 |
35 +-----+
36 |Access Registers at time of error |
37 +-----+
38 |R0:00000000 R1:00000000 R2:00000002 R3:00000000|
39 |R4:FFFFFFFF R5:FFFFFFFF R6:FFFFFFFF R7:FFFFFFFF|
41 |R8:FFFFFFFF R9:FFFFFFFF RA:00000000 RB:00000000|
42 |RC:00000000 RD:00000000 RE:00000000 RF:00000000|
43 +-----+
44 *** END OF POTENTIAL VULNERABILITY REPORT ***
```

Figure 86. Potential Vulnerability Output Example 0C4

Line 01

Indicates the service number where the potential vulnerability was detected.

Line 02

Indicates the six-digit ABEND code, which allows for user ABENDs when applicable. The line also includes the ABEND's reason code.

Line 03

Indicates the failing Program Status Word (PSW) as well as the name of the module where the potential vulnerability was found, if known, and the offset where the ABEND occurred.

Line 04

Indicates a potential eyecatcher for the module

Line 05

Indicates the failing instruction and the instructions length from the System Diagnostic Work Area (SDWA). Information on the SDWA can be found in the SDWA Mapping section of SDWA information from z/OS MVS Data Areas Volume 4 (RRP - XTL). Information on the op code can be found in IBM z/Architecture Principles of Operation.

Line 06

Indicates the failing instruction translated into assembler format.

Line 07

Indicates the source or target address that caused the translation exception.

Line 08

The home, primary, and secondary address space identifiers.

Line 09

The home, primary, and secondary job names.

Line 10

Indicates the probable CVSS score and vector of the potential vulnerability. CVSS scores are calculated using CVSS version 3.1. More information can be found here <https://www.first.org/cvss/calculator/3.1>.

Line 11

Indicates the header for the sample SLIP command.

Line 12

Provides an example SLIP command when one can be automatically generated.

Lines 16-23

Indicates the contents of the general purpose registers immediately before the service was invoked.

Lines 27-34

Indicates the contents of the general purpose registers at the time of error.

Lines 38-42

Indicates the contents of the access purpose registers at the time of error. These lines only appear in AR mode.

Additional output is provided if an overlay is detected.

```

01 *** POTENTIAL VULNERABILITY FOUND IN PC 00180807 ***
02 ABEND COMPLETION CODE: 0C1000 REASON CODE: 00000001
03 PSW: 070C2000 8000008A
04 INSTR LEN: 02 FAILING INSTR: 0000 1004 0002 003C 0002 0001
05 BREAKING EVENT ADDR: 2621F5AC MODULE: PVTMOD=(BPNTTEST,000025AC)
06 LIKELY EYECATCHER AT 2621F54C: .BPNTPC7 2024.043.ïï... ï.ï. 0..ém...µx4.....
07 BREAKING EVENT INSTR: 0513
08 TRANSLATED INSTR: BALR R1,R3
09 HOME ASID: 002E PRIMARY ASID: 0026 SECONDARY ASID: 002E
10 HOME JOB: ZACSJ PRIMARY JOB: BPNZACS SECONDARY JOB: ZACSJ
11 CVSS: 8.8 (CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H)
12 SLIP SAMPLE FOR PC 00180807:
13 SLIP SET,IF,ID=A009,RANGE=(2621F5AC),JOBNAME=ZACSJ,END
14 +-----+
15 |General Registers before the service |
16 +-----+
17 | R0:00000000_003FF7FF R1:00000000_00066000 |
18 | R2:00000000_003FF7FF R3:00000000_003FF7FF |
19 | R4:00000000_003FF7FF R5:00000000_003FF7FF |
20 | R6:00000000_003FF7FF R7:00000000_003FF7FF |
21 | R8:00000000_003FF7FF R9:00000000_003FF7FF |
22 | RA:00000000_003FF7FF RB:00000000_00180807 |
23 | RC:00000000_003FF7FF RD:00000000_003FF7FF |
24 | RE:00000000_003FF7FF RF:00000000_003FF7FF |
25 +-----+
26 |General Registers at time of error |
27 +-----+
28 | R0:00000000_003FF7FF R1:00000000_A621F5AE |
29 | R2:00000000_003FF7FF R3:00000000_00000088 |
30 | R4:00000000_7FFFFFFF R5:00000000_003FF7FF |
31 | R6:00000000_003FF7FF R7:00000000_003FF7FF |
32 | R8:00000000_003FF7FF R9:00000000_003FF7FF |
33 | RA:00000000_A621F574 RB:00000000_26228110 |
34 | RC:00000000_2621F5E0 RD:00000000_26228110 |
35 | RE:00000000_003FF7FF RF:00000000_003FF7FF |
36 +-----+
37 *** END OF POTENTIAL VULNERABILITY REPORT ***

```

Figure 87. Potential Vulnerability Output Example 0C1.

Line 01

Indicates the service number where the potential vulnerability was detected.

Line 02

Indicates the six-digit ABEND code, which allows for user ABENDs when applicable. The line also includes the ABEND's reason code.

Line 03

Indicates the failing Program Status Word (PSW).

Line 04

Indicates the failing instruction and the instructions length from the System Diagnostic Work Area (SDWA). Information on the SDWA can be found in the SDWA Mapping section of z/OS MVS Data Areas Volume 4 (RRP - XTL). Information on the op code can be found in IBM z/Architecture Principles of Operation.

Line 05

Indicates the breaking event address as well as the name of the module where the potential vulnerability was found, if known, and the offset where the ABEND occurred.

Line 06

Indicates a potential eyecatcher for the module.

Line 07

Indicates instruction at the breaking event address.

Line 08

Indicates the breaking event address translated into assembler format.

Line 09

The home, primary and secondary address space identifiers.

Line 10

The home, primary, and secondary job names.

Line 11

Indicates the probable CVSS score and vector of the potential vulnerability. CVSS scores are calculated using CVSS version 3.1. More information can be found here <https://www.first.org/cvss/calculator/3.1>.

Line 12

Indicates the header for the sample SLIP command

Line 13

Provides an example SLIP command when one can be automatically generated.

Lines 17-24

Indicates the contents of the general purpose registers immediately before the service was invoked.

Lines 28-35

Indicates the contents of the general purpose registers at the time of error.

```

01 *** POTENTIAL VULNERABILITY FOUND IN PC 00180909 ***
02 ABEND COMPLETION CODE: 0E0000 REASON CODE: 00000029
03 PSW: 070C6000 A6219126 MODULE: PVTMOD=(BPNTST,00002126)
04 LIKELY EYECATCHER AT A621909C: .BPNTPC9 2021.335.é{..{!éö..i.{. 0..¥(..ép
05 INSTR LEN: 06 FAILING INSTR: B04C 9A33 B048 D203 3000 2000
06 TRANSLATED INSTR: MVC 0(4,R3),0(R2)
07 ACCESS REGISTER CAUSING TRANSLATION EXCEPTION: 02
08 HOME ASID: 002C PRIMARY ASID: 0031 SECONDARY ASID: 002C
09 HOME JOB: ZAC SJ PRIMARY JOB: BPNZACS SECONDARY JOB: ZAC SJ
10 CVSS SCORE CANNOT BE DETERMINED FROM AVAILABLE INFORMATION
11 SLIP SAMPLE FOR PC 00180909:
12 SLIP SET,ID=A003,COMP=0E0,P=(BPNTST,00002126),END
13 +-----+
14 |General Registers before the service |
15 +-----+
16 | R0:00000000_003FF7FF R1:00000000_003FF7FF |
17 | R2:00000000_003FF7FF R3:00000000_003FF7FF |
18 | R4:00000000_003FF7FF R5:00000000_003FF7FF |
19 | R6:00000000_003FF7FF R7:00000000_003FF7FF |
20 | R8:00000000_003FF7FF R9:00000000_003FF7FF |
21 | RA:00000000_003FF7FF RB:00000000_00180909 |
22 | RC:00000000_0003B000 RD:00000000_003FF7FF |
23 | RE:00000000_003FF7FF RF:00000000_003FF7FF |
24 +-----+
25 |General Registers at time of error|
26 +-----+
27 | R0:00000000_003FF7FF R1:00000000_003FF7FF |
28 | R2:00000000_003FF7FF R3:00000000_2621DB00 |
29 | R4:00000000_00000020 R5:00000000_003FF7FF |
30 | R6:00000000_003FF7FF R7:00000000_003FF7FF |
31 | R8:00000000_003FF7FF R9:00000000_003FF7FF |
32 | RA:00000000_A62190D0 RB:00000000_2621DAB0 |
33 | RC:00000000_26219168 RD:00000000_2621DAB0 |
34 | RE:00000000_003FF7FF RF:00000000_00000000 |
35 +-----+
36 |Access Registers at time of error |
37 +-----+
38 |R0:00000000 R1:00000000 R2:00000020 R3:00000000|
39 |R4:FFFFFFFF R5:FFFFFFFF R6:FFFFFFFF R7:FFFFFFFF|
40 |R8:FFFFFFFF R9:FFFFFFFF RA:00000000 RB:00000000|
41 |RC:00000000 RD:00000000 RE:00000000 RF:00000000|
42 +-----+
43 *** END OF POTENTIAL VULNERABILITY REPORT ***

```

Figure 88. Potential Vulnerability Output Example 0E0.

Line 01

Indicates the service number where the potential vulnerability was detected.

Line 02

Indicates the six-digit ABEND code, which allows for user ABENDs when applicable. The line also includes the ABEND's reason code.

Line 03

Indicates the failing Program Status Word (PSW) as well as the name of the module where the potential vulnerability was found, if known, and the offset where the ABEND occurred.

Line 04

Indicates a potential eyecatcher for the module.

Line 05

Indicates the failing instruction and the instructions length from the System Diagnostic Work Area (SDWA). Information on the SDWA can be found in the SDWA Mapping section of z/OS MVS Data Areas Volume 4 (RRP - XTL). Information on the op code can be found in IBM z/Architecture Principles of Operation.

Line 06

Indicates the failing instruction translated into assembler format.

Line 07

Indicates the access register that caused the translation exception.

Line 08

The home, primary and secondary address space identifiers.

Line 09

The home, primary, and secondary job names.

Line 10

Indicates the probable CVSS score and vector of the potential vulnerability. CVSS scores are calculated using CVSS version 3.1. More information can be found here <https://www.first.org/cvss/calculator/3.1>.

Line 11

Indicates the header for the sample SLIP command.

Line 12

Provides an example SLIP command when one can be automatically generated.

Lines 16-23

Indicates the contents of the general purpose registers immediately before the service was invoked.

Lines 27-34

Indicates the contents of the general purpose registers at the time of error.

Lines 38-41

Indicates the contents of the access registers at the time of error.

Additional output is provided if an overlay is detected.

```

01 *** STORAGE OVERLAY FOUND IN PC 00180700 00000002 ***
02 CVSS: 8.8 (CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H)
03 +-----+
04 |General Registers before the service |
05 +-----+
06 | R0:00000000_003FF7FF R1:00000000_003FF7FF |
07 | R2:00000000_00057000 R3:00000000_003FF7FF |
08 | R4:00000000_003FF7FF R5:00000000_003FF7FF |
09 | R6:00000000_003FF7FF R7:00000000_003FF7FF |
10 | R8:00000000_003FF7FF R9:00000000_003FF7FF |
11 | RA:00000000_003FF7FF RB:00000000_00180700 |
12 | RC:00000000_003FF7FF RD:00000000_003FF7FF |
13 | RE:00000000_003FF7FF RF:00000002_003FF7FF |
14 +-----+
15 |Key 0 Parameter data before the service |
16 +-----+
17 | 00057000: 003FF7FF 003FF7FF 003FF7FF 003FF7FF |
18 | 00057010: 003FF7FF 003FF7FF 003FF7FF 003FF7FF |
19 | 00057020: 003FF7FF 003FF7FF 003FF7FF 003FF7FF |
20 | 00057030: 003FF7FF 003FF7FF 003FF7FF 003FF7FF |
21 | 00057040: 003FF7FF 003FF7FF 003FF7FF 003FF7FF |
22 +-----+
23 |Key 0 Parameter data after the service |
24 +-----+
25 | 00057000: 00000020 003FF7FF 003FF7FF 003FF7FF |
26 | 00057010: 003FF7FF 003FF7FF 003FF7FF 003FF7FF |
27 | 00057020: 003FF7FF 003FF7FF 003FF7FF 003FF7FF |
28 | 00057030: 003FF7FF 003FF7FF 003FF7FF 003FF7FF |
29 | 00057040: 003FF7FF 003FF7FF 003FF7FF 003FF7FF |
30 +-----+
31 *** END OF STORAGE OVERLAY REPORT ***

```

Figure 89. Storage overlay output.

Line 01

Indicates that a storage overlay was detected in the indicated service.

Line 02

Indicates the probable CVSS score and vector of the potential vulnerability. CVSS scores are calculated using CVSS version 3.1. More information can be found here <https://www.first.org/cvss/calculator/3.1>.

Lines 06-13

Indicates the contents of the general purpose registers immediately before the service was invoked.

Lines 17-21

Indicates the contents of the storage where the overlay was detected before the service was invoked.

Lines 25-29

Indicates the contents of the storage where the overlay was detected after the service was invoked.

At the bottom of each service's output a summary is given.


```

01 *** PRIVILEGE ESCALATION FOUND IN PC 0018090E ***
02 Control Returned From Recovery
03 Problem State, Key 8, APF ON
04 CVSS: 8.8 (CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H)
05 +-----+
06 |General Registers before the service|
07 +-----+
08 | R0:00000000_003FF7FF R1:00000000_0005F000 |
09 | R2:00000000_003FF7FF R3:00000000_003FF7FF |
10 | R4:00000000_003FF7FF R5:00000000_003FF7FF |
11 | R6:00000000_003FF7FF R7:00000000_003FF7FF |
12 | R8:00000000_003FF7FF R9:00000000_003FF7FF |
13 | RA:00000000_003FF7FF RB:00000000_0018090E |
14 | RC:00000000_003FF7FF RD:00000000_003FF7FF |
15 | RE:00000000_003FF7FF RF:00000000_003FF7FF |
16 +-----+
17 |General Registers at time of error|
18 +-----+
19 | R0:00000000_003FF7FF R1:00000000_0005F000 |
20 | R2:00000000_003FF7FF R3:00000000_262D3048 |
21 | R4:00000000_009FB37C R5:00000000_003FF7FF |
22 | R6:00000000_003FF7FF R7:00000000_00000020 |
23 | R8:00000000_003FF7FF R9:00000000_003FF7FF |
24 | RA:00000000_A562A02C RB:00000000_262D3000 |
25 | RC:00000000_2562A0B0 RD:00000000_262D3000 |
26 | RE:00000000_003FF7FF RF:00000000_003FF7FF |
27 +-----+
28 *** PRIVILEGE ESCALATION END ***

```

Figure 90. Privilege Escalation Output

Line 01

Indicates that a privilege escalation was detected in the indicated service. This means the service returned to the caller with an increased level of Privilege. One or more of the state, key, or apf authorization were not returned to the level of the caller. The caller called in Problem state, Key 8, APF Off.

Line 02

Indicates whether the service being tested returned from the routine normally or from recovery

Line 03

Indicates the state, key, and whether or not APF authorization is active at the point when the service being tested returned.

Line 04

Indicates the probable CVSS score and vector of the potential vulnerability. CVSS scores are calculated using CVSS version 3.1. More information can be found here <https://www.first.org/cvss/calculator/3.1>.

Lines 08-15

Indicates the contents of the general purpose registers immediately before the service was invoked.

Lines 19-26

Indicates the contents of the general purpose registers at the time of error.

```

01 +-----+
02 | Summary for PC: 00180703 VULNERABLE |
03 +-----+
04 | Testcase Templates Run: 00000020x |
05 | Test Iterations: 00000045x |
06 +-----+
07 | Overlay Count: 00000006x |
08 | Potential Vulnerability Count: 0000001Ax |
09 +-----+

```

Figure 91. Basic Test Summary Box Output

Line 02

Indicates the service that was run and its overall status. The possibilities are as follows:

SUCCESSFUL

All tests ran with no potential vulnerabilities detected.

VULNERABLE

One or more potential vulnerabilities were detected.

INCOMPLETE

Testing was unable to complete entirely for this service. This may be caused, for example, by the logrec data set becoming full or by the test timing out if the service being tested does not return within a few seconds of being invoked.

UNDEFINED

The specified service is not defined to the system or cannot be invoked from this address space.

PROTECTED

The service cannot be invoked by unauthorized users and therefore does not apply to this tool's testing

NOTAUTH

The program to be tested is not authorized. The program is not AC(1) or the library containing it is not in the APF list.

NOTFOUND

Could not find the program to be tested.

OUTOFSCOPE

The program to be tested is out of scope of zACS testing.

Line 04

Indicates the total number of tests run on the service.

Line 05

Indicates the number of iterations run on the service. The number of iterations performed on any given service may vary.

Line 07

Indicates the number of storage overlays detected.

Line 08

Indicates the total number of potential vulnerabilities detected.

Note: The same potential vulnerability may be hit by multiple tests which can affect the counts in lines 7 and 8.

If advanced testing is used, the summary will include a Potential Function Register line, shown below.

```
+-----+
| Summary for PC: 00180914          VULNERABLE |
| Potential Function Register:      02x |
+-----+
| Testcase Templates Run:          00000448x |
| Test Iterations:                 0000066Cx |
+-----+
| Overlay Count:                   00000000x |
| Potential Vulnerability Count:    00000080x |
+-----+
```

Figure 92. Advanced Test Summary Box Output

This line indicates a register that is believed to be used to contain function codes. If none are found, the word 'none' will be in place of the register number.

Serviceability Note: There can be false positives in the resulting output of this scanner as part of its normal operation. This is not a flaw in the tool but rather an indication of possible vulnerability scenarios that may or may not apply in particular instances. System integrity within a PC or SVC routine is the responsibility of the respective product owner of that routine, whether it be IBM-, vendor-, or in-house-provided code. The three-character prefix within the routine name provides a likely indication as to the product and its owner for purposes of confirming problem diagnosis and providing a fix as needed.

Setting a SLIP Trap to Capture a Dump

When a vulnerability is found that you would like to set a SLIP trap for, the SLIP command syntax will vary depending what ABEND code <xyz> observed and if it was in PVTMOD, LPAMOD, or NUCMOD.

Note: The following example SLIP commands use ACTION=SVCD and MATCHLIM=1 by default.

PVTMOD Example:

```
SLIP SET,COMP=<xyz>,P=(<modname>,<offset>),end
```

Figure 93. PVTMOD Example.

LPAMOD Example:

```
SLIP SET,COMP=<xyz>,L=(<modname>,<offset>),end
```

Figure 94. LPAMOD Example.

NUCMOD Example:

```
SLIP SET,COMP=<xyz>,N=(<modname>,<offset>),end
```

IF Example:

```
SLIP SET,IF,RANGE=(<failing address>),end
```

After setting the SLIP, rerun the failing service number of module.

Note: PVTMOD SLIP traps will not match if the correct local lock is not held and cannot be obtained. If no module name has been provided by the zACS output, or a SLIP trap with the PVTMOD and COMP parameters will not work, set an Instruction Fetch (IF) SLIP trap. Use the JOBNAME parameter when setting the SLIP trap (the default job name for zACS jobs is 'ZAC SJ'). The DATA parameter can also be used to further qualify the event, such as by basing the result on the value in a particular register

Vulnerability examples and fixes

The following examples are samples of potential vulnerability reports from zACS. These were generated using a test program called BPNTTEST and are not real vulnerabilities on the system. Descriptions of the vulnerabilities and possible fixes for them will follow.

Fetch vulnerability example

```
*** POTENTIAL VULNERABILITY FOUND IN PC 00180901 ***
ABEND COMPLETION CODE: 0C4000 REASON CODE: 00000011
PSW: 070C6000 A6218AC2 MODULE: PVTMOD=(BPNTTEST,00001AC2)
LIKELY EYECATCHER AT A6218A3C: .BPNTPC1 2021.335.éí..íí...!éö..i.í. 0..¥(..ém
INSTR LEN: 06 FAILING INSTR: B048 9A33 B04C D203 3000 2000
TRANSLATED INSTR: MVC 0(4,R3),0(R2)
SOURCE ADDRESS CAUSING TRANSLATION EXCEPTION: 00000000_00052801
HOME ASID: 002C PRIMARY ASID: 0031 SECONDARY ASID: 002C
HOME JOB: ZACSJ PRIMARY JOB: BPNZACS SECONDARY JOB: ZACSJ
CVSS: 6.5 (CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:N/A:N)
SLIP SAMPLE FOR PC 00180901:
SLIP SET,ID=A004,COMP=0C4,P=(BPNTTEST,00001AC2),END
+-----+
|General Registers before the service|
+-----+
| R0:00000000_003FF7FF R1:00000000_003FF7FF |
| R2:00000000_00052000 R3:00000000_003FF7FF |
| R4:00000000_003FF7FF R5:00000000_003FF7FF |
| R6:00000000_003FF7FF R7:00000000_003FF7FF |
| R8:00000000_003FF7FF R9:00000000_003FF7FF |
| RA:00000000_003FF7FF RB:00000000_00180901 |
| RC:00000000_003FF7FF RD:00000000_003FF7FF |
| RE:00000000_003FF7FF RF:00000000_003FF7FF |
+-----+
|General Registers at time of error|
+-----+
| R0:00000000_00000000 R1:00000000_003FF7FF |
| R2:00000000_00052000 R3:00000000_26257350 |
| R4:00000000_02169100 R5:00000000_003FF7FF |
| R6:00000000_003FF7FF R7:00000000_003FF7FF |
| R8:00000000_003FF7FF R9:00000000_003FF7FF |
| RA:00000000_A6218A70 RB:00000000_26257300 |
| RC:00000000_26218B08 RD:00000000_26257300 |
| RE:00000000_003FF7FF RF:00000000_00000002 |
+-----+
|Access Registers at time of error|
+-----+
|R0:00000000 R1:00000000 R2:00000002 R3:00000000|
|R4:FFFFFFFF R5:FFFFFFFF R6:FFFFFFFF R7:FFFFFFFF|
|R8:FFFFFFFF R9:FFFFFFFF RA:00000000 RB:00000000|
|RC:00000000 RD:00000000 RE:00000000 RF:00000000|
+-----+
*** END OF POTENTIAL VULNERABILITY REPORT ***
```

Figure 95. Fetch Vulnerability example output.

From the report, the CVSS score indicates the probable type of error that was detected. This CVSS score of 6.5 indicates a probable fetch violation. The report indicates that the PC is suspected of fetching storage that the caller was not authorized to view. The failing instruction information as well as the general registers can be used to understand what is occurring. In this case, the op code of the failing instruction is 'D2'x which indicates an MVC instruction. The module name of BPNTTEST and offset of 'D02'x are also provided for additional diagnostics. A dump can also be captured of the module at that offset using the sample SLIP command provided in the report.

This vulnerability was caused by the following code sample:

```
LA R3,copyparms
MVC 0(4,R3),0(R2)
```

Figure 96. Fetch Vulnerability code example.

This code is vulnerable because it blindly copies information from the caller. General register 2 is being used as an input in this case, but the value provided in general register 2 is an address to which the caller does not necessarily have access. The PC routine is not accounting for the key of the caller.

The following is an alternative implementation:

```

LHI R3,1
ESTA R0,R3
SRDL R0,48
LHI R0,3
LA R3,copyparms
MVCSK 0(R3),0(R2)

```

Figure 97. Fetch Vulnerability alternative implementation.

The fix for this vulnerability is to utilize the instruction, MVCSK. The instruction will only succeed when the caller has access to the parameter provided, maintaining system integrity.

A break down of the assembler instructions in the fixed code are as follows:

LHI R3,1

LHI (Load Halfword Immediate (32)) loads general register 3 with a 1 indicating PSW for the ESTA instruction.

ESTA R0,R3

ESTA (Extract Stacked State) extracts the PSW's value and put it into general register 0.

SRDL R0,48

SRDL(Shift Right Double Logical) shifts the PSW value into the general register 1 such that the portion with the caller's keys gets set in general register 1.

LHI R0,3

LHI (Load Halfword Immediate (32)) loads the length that the copy should be into general register 0.

LA R3,copyparms

LA (Load Address) loads the address where the copy is to the program's storage (the destination or the target address).

MVCSK 0(R3),0(R2)

MVCSK (Move with Source Key) will move data from the address in general register 2 to the address in general register 3 using the key that is set in general register 1 and the length that is set in general register 0.

Store vulnerability example

```
*** POTENTIAL VULNERABILITY FOUND IN PC 00180900 ***
ABEND COMPLETION CODE: 0C4000 REASON CODE: 00000011
PSW: 070C6000 A62189DA MODULE: PVTMOD=(BPNTTEST,000019DA)
LIKELY EYECATCHER AT A621894C: .BPNTPC0 2021.335.éí..íí...;éö..i.í. 0..¥(..ém
INSTR LEN: 06 FAILING INSTR: B048 9A33 B04C D203 2000 3000
TRANSLATED INSTR: MVC 0(4,R2),0(R3)
TARGET ADDRESS CAUSING TRANSLATION EXCEPTION: 00000000_00046401
HOME ASID: 002C PRIMARY ASID: 0031 SECONDARY ASID: 002C
HOME JOB: ZAC SJ PRIMARY JOB: BPNZACS SECONDARY JOB: ZAC SJ
CVSS: 8.8 (CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H)
SLIP SAMPLE FOR PC 00180900:
SLIP SET,ID=A000,COMP=0C4,P=(BPNTTEST,000019DA),END
+-----+
|General Registers before the service|
+-----+
| R0:00000000_003FF7FF R1:00000000_003FF7FF|
| R2:00000000_00046000 R3:00000000_003FF7FF|
| R4:00000000_003FF7FF R5:00000000_003FF7FF|
| R6:00000000_003FF7FF R7:00000000_003FF7FF|
| R8:00000000_003FF7FF R9:00000000_003FF7FF|
| RA:00000000_003FF7FF RB:00000000_00180900|
| RC:00000000_003FF7FF RD:00000000_003FF7FF|
| RE:00000000_003FF7FF RF:00000000_003FF7FF|
+-----+
|General Registers at time of error|
+-----+
| R0:00000000_00000000 R1:00000000_003FF7FF|
| R2:00000000_00046000 R3:00000000_2621D100|
| R4:00000000_021690F0 R5:00000000_00000020|
| R6:00000000_003FF7FF R7:00000000_003FF7FF|
| R8:00000000_003FF7FF R9:00000000_003FF7FF|
| RA:00000000_A6218980 RB:00000000_2621D0B0|
| RC:00000000_26218A20 RD:00000000_2621D0B0|
| RE:00000000_003FF7FF RF:00000000_00000002|
+-----+
|Access Registers at time of error|
+-----+
|R0:00000000 R1:00000000 R2:00000002 R3:00000000|
|R4:FFFFFFFF R5:FFFFFFFF R6:FFFFFFFF R7:FFFFFFFF|
|R8:FFFFFFFF R9:FFFFFFFF RA:00000000 RB:00000000|
|RC:00000000 RD:00000000 RE:00000000 RF:00000000|
+-----+
*** END OF POTENTIAL VULNERABILITY REPORT ***
```

Figure 98. Store Vulnerability example output.

From the report, the CVSS score indicates the probable type of error that was detected. This CVSS score of 8.8 indicates a probable store violation. The report indicates that the PC is suspected of storing to memory into which the caller was not authorized to store. The failing instruction information as well as the general registers can be used to understand what is occurring. In this case, the op code of the failing instruction is 'D2'x which indicates an MVC instruction. The module name of BPNTTEST and offset of 'D02'x are also provided for additional diagnostics. A dump can also be captured of the module at that offset using the sample SLIP command provided in the report.

This vulnerability was caused by the following code sample:

```
LA R3,sourcedata
MVC 0(4,R2),0(R3)
```

Figure 99. Store Vulnerability code example.

This code is vulnerable because it blindly copies information to an address provided by the caller. General register 2 is being used as an output in this case, but the value provided in general register 2 is an address to which the caller does not necessarily have access. The PC routine is not accounting for the key of the caller.

The following is an alternative implementation:

```

LHI R3,1
ESTA R0,R3
SRDL R0,48
LHI R0,3
LA R3,sourcedata
MVCDK 0(R2),0(R3)

```

Figure 100. Store Vulnerability alternative implementation.

The fix for this vulnerability is to utilize the instruction, MVCDK. The instruction will only succeed when the caller has access to the parameter provided, maintaining system integrity.

A break down of the assembler instructions in the fixed code are as follows:

LHI R3,1

LHI (Load Halfword Immediate (32)) loads the general register 3 with a 1 indicating PSW for the ESTA instruction.

ESTA R0,R3

ESTA (Extract Stacked State) extracts the PSW's value and put it into general register 0.

SRDL R0,48

SRDL (Shift Right Double Logical) shifts the PSW value into general register 1 such that the portion with the caller's key gets set in general register 1.

LHI R0,3

LHI (Load Halfword Immediate (32)) loads the length that the copy should be into general register 0.

LA R3,sourcedata

LA (Load Address) loads the source address of where the data is to be copied from.

MVCDK 0(R2),0(R3)

MVCDK (Move with Destination Key) will move data at the address in general register 3 to the address in general register 2 using the key that is set in general register 1 and the length that is set in general register 0.

Indirect parameter and storage overlay vulnerability example

```

*** POTENTIAL VULNERABILITY FOUND IN PC 0018090C ***
ABEND COMPLETION CODE: 0C4000 REASON CODE: 00000004
PSW: 070C1000 A562C09E MODULE: LPAMOD=(BPNTGCC,0000009E)
LIKELY EYECATCHER AT 2562C004: .BPNTGCC 2021.335.{i...*i..}. 0..ém...µx4.....
INSTR LEN: 04 FAILING INSTR: B048 5034 0000 5800 C00C 41F0
TRANSLATED INSTR: ST R3,0(R4)
TARGET ADDRESS CAUSING TRANSLATION EXCEPTION: 00000000_00051404
HOME ASID: 002C PRIMARY ASID: 002C SECONDARY ASID: 002C
HOME JOB: ZAC SJ PRIMARY JOB: BPNZACS SECONDARY JOB: ZAC SJ
CVSS: 8.8 (CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H)
SLIP SAMPLE FOR PC 0018090C:
SLIP SET,ID=A001,COMP=0C4,L=(BPNTGCC,0000009E),END
+-----+
|General Registers before the service|
+-----+
| R0:00000000_003FF7FF R1:00000000_0004C000 |
| R2:00000000_003FF7FF R3:00000000_003FF7FF |
| R4:00000000_003FF7FF R5:00000000_003FF7FF |
| R6:00000000_003FF7FF R7:00000000_003FF7FF |
| R8:00000000_003FF7FF R9:00000000_003FF7FF |
| RA:00000000_003FF7FF RB:00000000_0018090C |
| RC:00000000_003FF7FF RD:00000000_003FF7FF |
| RE:00000000_003FF7FF RF:00000000_003FF7FF |
+-----+
|General Registers at time of error|
+-----+
| R0:00000000_00000004 R1:00000000_0000078D |
| R2:00000000_003FF7FF R3:00000000_AAAAAAAA |
| R4:00000000_00051000 R5:00000000_003FF7FF |
| R6:00000000_003FF7FF R7:00000000_003FF7FF |
| R8:00000000_003FF7FF R9:00000000_003FF7FF |
| RA:00000000_A562C02C RB:00000000_262FAA80 |
| RC:00000000_2562C0D0 RD:00000000_262FAA80 |
| RE:00000000_262FAAF4 RF:00000000_0004F000 |
+-----+
*** END OF POTENTIAL VULNERABILITY REPORT ***

```

```

*** STORAGE OVERLAY FOUND IN PC 0018090C ***
CVSS: 8.8 (CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H)
+-----+
|General Registers before the service|
+-----+
| R0:00000000_003FF7FF  R1:00000000_0004C000|
| R2:00000000_003FF7FF  R3:00000000_003FF7FF|
| R4:00000000_003FF7FF  R5:00000000_003FF7FF|
| R6:00000000_003FF7FF  R7:00000000_003FF7FF|
| R8:00000000_003FF7FF  R9:00000000_003FF7FF|
| RA:00000000_003FF7FF  RB:00000000_0018090C|
| RC:00000000_003FF7FF  RD:00000000_003FF7FF|
| RE:00000000_003FF7FF  RF:00000000_003FF7FF|
+-----+
|Key 0 Parameter data before the service|
+-----+
| 00048490: 003FF7FF 003FF7FF 003FF7FF 003FF7FF|
| 000484A0: 003FF7FF 003FF7FF 003FF7FF 003FF7FF|
| 000484B0: 003FF7FF 003FF7FF 003FF7FF 003FF7FF|
| 000484C0: 003FF7FF 003FF7FF 003FF7FF 003FF7FF|
| 000484D0: 003FF7FF 003FF7FF 003FF7FF 003FF7FF|
+-----+
|Key 0 Parameter data after the service|
+-----+
| 00048490: AAAAAAAA 003FF7FF 003FF7FF 003FF7FF|
| 000484A0: 003FF7FF 003FF7FF 003FF7FF 003FF7FF|
| 000484B0: 003FF7FF 003FF7FF 003FF7FF 003FF7FF|
| 000484C0: 003FF7FF 003FF7FF 003FF7FF 003FF7FF|
| 000484D0: 003FF7FF 003FF7FF 003FF7FF 003FF7FF|
+-----+
*** END OF STORAGE OVERLAY REPORT ***

```

Figure 101. Indirect Parameter and Storage Overlay Vulnerability Example Output

From this report, the error information points to a store protection violation. The CVSS score indicates this as well as the translated instruction at the time of error. The CVSS of 8.8 indicating a likely store violation occurred, and the translated instruction is a store instruction. As with the previous examples, the module name of BPNTPC and the offset of '9C'x are provided for diagnostic purposes such as setting a SLIP trap at the module and offset provided to get a dump of the potential vulnerability. In a separate test, zACS flagged that an overlay of storage has occurred. The storage overlay that occurred can be noted in the output report by comparing the parameter data before and after the service. The first 4 bytes of storage have been overwritten after the service was called.

This example shows a routine that does several things right, but it does one thing wrong. The zACS tool gets far enough into the routine to expose the vulnerability and demonstrate the storage overlay.

```

LA R14,myLB1
LR R15,R1
LHI R3,1
ESTA R0,R3
SRDL R0,48
LHI R0,3
MVCSK 0(R14),0(R15)
L R15,myLB2ptr
LA R14,myLB2
MVCSK 0(R14),0(R15)
LHI R0,4
L R4,myLB3ptr
L R3,myConst
ST R3,0(R4)

```

Figure 102. Storage overlay code example.

The main issue in this code is the store instruction at the end. It is storing to the provided storage without considering the key of the caller. In the above example, this routine did properly copy an input from the caller as well as another input that was pointed to by the first input using MVCSK instructions. However, the input that was copied is then used as a target for a store instruction using key 0. This is an example of an indirect parameter causing a vulnerability after the initial parameters have been copied safely by the routine.

The following is an alternative implementation:


```

LA R14,myLB1
LR R15,R1
LHI R3,1
ESTA R0,R3
SRDL R0,48
LHI R0,3
MVCSK 0(R14),0(R15)
L R15,myLB2ptr
LA R14,myLB2
MVCSK 0(R14),0(R15)
LHI R0,4
L R4,myLB3ptr
LA R3,myConst
MVCDK 0(R4),0(R3)

```

Figure 103. Storage Overlay alternative implementation.

This fixes the vulnerability from the store because it considers the key of the caller when attempting to write to the storage provided by the caller. If the caller does not have access to the provided storage, the attempt to write to storage will fail with an ABEND 0C4. This will prevent the vulnerability caused by the original store instruction.

A break down of the assembler instructions in the fixed code are as follows:

LA R14,myLB1

LA (Load Address) loads the address of the local storage for the caller's parameter to be copied into.

LR R15,R1

LR (Load Register) loads the value from register 1 which contains a parameter from the caller into register 15.

LHI R3,1

LHI (Load Halfword Immediate (32)) loads general register 3 with a 1 indicating PSW for the ESTA instruction.

ESTA R0,R3

ESTA (Extract Stacked State) extracts the PSW's value and put it into general register 0.

SRDL R0,48

SRDL(Shift Right Double Logical) shifts the PSW value into the general register 1 such that the portion with the caller's keys gets set in general register 1.

LHI R0,3

LHI (Load Halfword Immediate (32)) loads the length that the copy should be into general register 0.

MVCSK 0(R14),0(R15)

MVCSK (Move with Source Key) will move data from the address in general register 15 to the address in general register 14 using the key that is set in general register 1 and the length that is set in general register 0. This is copying the first parameter from the caller into local storage.

L R15,myLB2ptr

L(Load) load the value of the pointer passed as part of the caller's first parameter.

LA R14,myLB2

LA (Load Address) loads the address of the local storage for the caller's second parameter to be copied into.

MVCSK 0(R14),0(R15)

MVCSK (Move with Source Key) will move data from the address in general register 15 to the address in general register 14 using the key that is set in general register 1 and the length that is set in general register 0. This is copying the first parameter from the caller into local storage.

LHI R0,3

LHI (Load Halfword Immediate (32)) loads the length that the copy should be into general register 0.

L R4,myLB3ptr

L(Load) load the value of the pointer passed as part of the caller's second parameter. This is where the output is copied into.

LA R3,myConst

LA (Load Address) loads the address of the local storage for the caller's second parameter which to be copied into as part of the output.

MVCDK 0(R4),0(R3)

MVCDK (Move with Destination Key) will move data at the address in general register 3 to the address in general register 4 using the key that is set in general register 1 and the length that is set in general register 0. Safely copying out to the caller provided storage.

Chapter 6. zACM Configuration

The z/OS Authorized Code Monitor resides in 'SYS1.BPN.SBPNLPA'. This dataset must be added to LPALIB to utilize zACM as well as zACS.

Output and filter data set allocation

Both a full output and summary output data set must be allocated and specified in the started task in order to run zACM. The suggested allocations are as follows:

```
alloc new da(<output-dataset>) dd(<myoutdd>) dsorg(ps) space(300,300) tracks lrecl(80)
blksize(0) recfm(f,b)

free dd(<myoutdd>)
```

Figure 104. zACM Full Output Allocation

```
alloc new da(<summary-dataset>) dd(<mysumdd>) dsorg(ps) space(300,300) tracks lrecl(80)
blksize(0) recfm(f,b)

free dd(<mysumdd>)
```

Figure 105. zACM Summary Output Allocation

If you are using one or more filter list the recommended allocation for each list is:

```
alloc new da(<filter-dataset>) dd(<myfltd>) dsorg(ps) space(10,10) tracks lrecl(80) blksize(0)
recfm(f,b)

free dd(<myfltd>)
```

Figure 106. zACM Filter Allocation

Data set protection and permissions

Note: If zACS was previously configured on this system, the dataset protection and permissions apply to zACM as well.

The same dataset protections apply to zACM and zACS.

The following are suggested datasets names for zACM use.

```
userid.ZACM.OUTPUT
userid.ZACM.OUT.SUM
userid.ZACM.INCLUDE.MODNAME
userid.ZACM.EXCLUDE.MODNAME
userid.ZACM.INCLUDE.JOBNAME
userid.ZACM.EXCLUDE.JOBNAME
```

For more information, see [“Data set protection and permissions”](#) on page 5.

Started task (BPNZACM)

Define a profile in the STARTED class to associate the *userid* with the zACM address space:

1. Authorize the started task.

```
RDEFINE STARTED BPNZACM.** UACC(NONE) STDATA(USER(userid) GROUP(SYS1) TRUSTED(YES))
```

2. Refresh that STARTED class.

```
SETR RACLIST(STARTED) REFRESH
```

The following JCL is contained in the SYS1.SAMPLIB(BPNZACM) data set. This JCL needs to be copied into the working PROCLIB. MYOUTDD is required and must point to the potential vulnerability data set, which is used for the output of the tool. MYSUMDD is also required and must point to the potential vulnerability summary dataset where only unique hits are recorded. MYJOBEX, MYJOBIN, MYMODEX, and MYMODIN are optional and must point to a dataset that contains the job exclusions, job inclusions, module exclusions, and module inclusions respectively if they are being used. When this task is started, the output data set and summary data set are cleared of any data, including any vulnerability information from a previous time the task was running.

An optional parameter can be added to the started task to enable First Failure Data Capture. This feature sets a SLIP ID=BPNZ that captures a memory dump when a zACM potential vulnerability is detected. The syntax to enable First Failure Data Capture is `PARM=(##,description)`. ## is a decimal number up to 99 that is used as the SLIP's **matchlim** and *description* is an optional description to be used in the SLIP. The total length of the **PARM** must not exceed 56 characters.

Also, optional Automatic SLIP Setting functions can be enabled by specifying an alphabet letter as the first character on the **PARM**.

```
PARM=(A)
```

A is any single capital letter.

Enabling this option causes all unique example SLIP traps that are generated while zACM is running to be set so that a memory dump can be captured on subsequent instances of the same error. The letter used to enable Automatic SLIP Setting is used as the prefix for the SLIP ID in the generated SLIPs. The set SLIPs are deleted when zACM is stopped.

Automatic SLIP Setting and First Failure Data Capture can be enabled together. Enabling both allows for unrepeatable potential vulnerabilities to capture memory dumps, but also ensures that repeatable potential vulnerabilities can still capture memory dumps if the First Failure Data Capture **matchlim** runs out. To enable both First Failure Data Capture and Automatic SLIP Setting, begin the **PARM** with the Automatic SLIP Setting SLIP ID prefix, followed by the First Failure Data Capture parameters, resulting in `PARM=(A,##,description)`.

Note: Only one **SLIP PER** trap can be set at a time, if one is already enabled, following **SLIP PER** traps are not set. Also, if a generated **SLIP PER** trap matches on a zACM hit, zACM output can be suppressed for that occurrence.

Registration

zACM and zACS share a [“Registration”](#) on page 11.

Chapter 7. Running zACM

Monitor setup

Define a new profile for BPNZACM:

```
RDEFINE STARTED BPNZACM.** UACC(NONE) STDATA(USER(user) GROUP(SYS1) TRUSTED(YES))  
SETR RACLIST(STARTED) REFRESH
```

Figure 107. Define BPNZACM profile and refresh

Running the monitor

Start the tool with your started task using the console:

```
S BPNZACM,REUSASID=YES
```

Warning: Starting this task will clear the output data set defined in the started task, including any vulnerability data that was found.

The tool sets the following SLIP when the task is started.

```
SLIP SET, ID=BPN0,ERRTYP=INTEGMON, A=RECORD, END
```

```
SLIP SET, ID=BPNZ,ERRTYP=INTEGMON, A=(SVCD, RECORD), MATCHLIM=##, DESC=YYYY, END
```

Note: BPNZ is only set if PARM is specified on the started task. ## refers to the number set for the match limit in the PARM. YYYY refers to any description text set in the PARM.

Note: The SLIP may appear differently depending on filtering options.

Note: This SLIP may override other SLIPs set on the system.

These SLIPs will be removed when the task is terminated with the following command:

```
P BPNZACM
```

Figure 108. Starting zACM, setting SLIP trap, and purging zACM.

Note: zACM will monitor the system in the background, writing entries to the potential vulnerability log and summary log as needed.

Filtering monitored services with include or exclude lists

Two types of filtering that can be applied to zACM, filtering on job name and filtering by module. Both types of filtering support exclude and include lists. Both a job name and module filtering list can be specified at the same time, but each may only have either their include or exclude list specified at once. Module and job name filters are limited to 100 entries each. Lines that are comments do not count as entries.

Note: To update the lists, you must stop and restart zACM.

To add a filter list to zACM add one or more of the following DDs to the started task:

To include these filters, add any of the following optional DD statements to the started task for zACM.

```
//MYJOBEX DD DSN=<job-exclude-dataset>, DISP=SHR
```

Figure 109. Exclude by job name list DD

Note: Note: The addition of a job exclusion list will change the BPN0 and BPNZ slips in the following manner:

```
SLIP SET, ID=BPNO, ERRTP=INTEGMON, A=RECORD, JF=IMCBE, END
SLIP SET, ID=BPNZ, ERRTP=INTEGMON, A=(SVCD, RECORD), MATCHLIM=##, DESC=YYYY, JF=IMCBE, END
```

```
//MYJOBIN DD DSN=<job-include-dataset>, DISP=SHR
```

Figure 110. Include by job name list DD

Note: The addition of a job inclusion list will change the BPN0 and BPNZ slips in the following manner:

```
SLIP SET, ID=BPNO, ERRTP=INTEGMON, A=RECORD, JF=IMCBI, END
SLIP SET, ID=BPNZ, ERRTP=INTEGMON, A=(SVCD, RECORD), MATCHLIM=##, DESC=YYYY, JF=IMCBI, END
```

```
//MYMODEX DD DSN=<mod-exclude-dataset>, DISP=SHR
```

Figure 111. Exclude by module list DD

Note: The addition of a module exclusion list will change the BPN0 and BPNZ slips in the following manner:

```
SLIP SET, ID=BPNO, ERRTP=INTEGMON, A=RECORD, MF=IMCBE, END
SLIP SET, ID=BPNZ, ERRTP=INTEGMON, A=(SVCD, RECORD), MATCHLIM=##, DESC=YYYY, MF=IMCBE, END
```

```
//MYMODIN DD DSN=<mod-include-dataset>, DISP=SHR
```

Figure 112. Include by module list DD

Note: The addition of a module inclusion list will change the BPN0 and BPNZ slips in the following manner:

```
SLIP SET, ID=BPNO, ERRTP=INTEGMON, A=RECORD, MF=IMCBI, END
SLIP SET, ID=BPNZ, ERRTP=INTEGMON, A=(SVCD, RECORD), MATCHLIM=##, DESC=YYYY, MF=IMCBI, END
```

Note: Empty data sets may be specified but will not affect filtering results.

Job name filtering

The job filtering data sets require that each entry be on an individual line that specifies the job name. The job name can be up to 8 characters. Job names shorter than 8 characters must be padded with blanks. A comment can be indicated by placing a # in column 1. That line will be ignored. An example job name filter data set can be seen below.

```
TESTJOB
JOBNAME
FULLNAME
#SAMPLE COMMENT
```

Figure 113. Example job name filter list for zACM

Note: For the first two entries, an additional blank character is added to the end of the entry to pad the length to 8 characters.

Module filtering

Module filtering can occur through two different methods, by module name or by module address. The module name filtering data sets may include a mix of filtering both by module name and by module address. A comment can be indicated by placing a # in column 1. That line will be ignored. An example module filter data set can be seen below.

```

N MODNAME 002C TESTJOB
N MODNAME1 FFFF
N MOD      FFFF JOBNAME
N M1       0123
A 0AB013F8 0AB014E8 002C TESTJOB
A 0AB013F8 0AB014E8 0000
A 0AB013F8 0AB014E8 0000 TEST
A 0AB013F8 0AB014E8 0123
#SAMPLE COMMENT

```

Figure 114. Example module filter list for zACM

The first column of an entry must be N or A. N specifies that it is an entry to filter by module name, and A specifies that it is an entry to be filtered by module address. It can also be a # in the case of a comment.

The second column of an entry must be a space.

If you are filtering by name:

- Columns 3-10 must contain the module name padded with spaces to 8 characters.
- Column 11 must be a space.
- Columns 12-15 must contain a 4 character ASID must then be specified in columns 12-15. The ASID should be the primary ASID of the address space that the module name will be found in. A value of FFFF can be utilized to indicate to utilize the primary address space when not in home asc mode, and to utilize the home address space otherwise.
- Column 16 must be a space.
- Columns 17-24 must contain the job name, padded to 8 characters. The job name can be left as 8 blank characters to indicate to not additionally filter on the job name with the module name entry.

Wildcard support exists for both the module name and job name field in the entry. Wildcard is indicated by utilizing a ? character, and it will match any character in that spot in the entry.

If you are filtering by address:

- Columns 3-10 must contain the 8 character, hexadecimal start address of the module.
- Column 11 must be a space.
- Columns 12-19 must contain the 8 character, hexadecimal end address of the module. The end address should be 1 byte past the end of the module that is being filtered.
- Column 20 must contain a space.
- Columns 21-24 must contain a 4 character ASID. The ASID represents the primary ASID. If the address is in common storage, a value of 0000 may be specified for the ASID.
- Column 25 must contain a space.
- Columns 26-33 must contain the job name, padded to 8 characters. The job name can be left as 8 blanks if job name filtering should not be used in conjunction with the module name address filtering.

Wildcard support exists for the job name field in the entry. Wildcard is indicated by utilizing a ? character, and it will match any character in that spot in the entry.

Chapter 8. zACM Diagnosis

Example zACM output is provided and is explained on how to interpret the results. Examples of code vulnerabilities are provided with instructions that explain how to fix them.

The zACM output file that you specified for the MYOUTDD statement in the started task must be reviewed periodically for new potential vulnerability hits. These hits are recorded in MYSUMDD.

Interpreting the zACM Potential Vulnerability Output File

zACM finds similar potential vulnerabilities to zACS, and has similar looking output.

- [Figure 86 on page 63](#)
- [Figure 87 on page 65](#)
- [Figure 88 on page 66](#)

zACM Vulnerability Examples and Fixes

Potential vulnerabilities that zACM finds must be fixed in a similar fashion to other potential vulnerabilities found by zACS.

- [“Fetch vulnerability example” on page 72](#)
- [“Store vulnerability example” on page 74](#)

Chapter 9. System Codes

Return and reason codes

This topic covers the return and reason codes of the potential vulnerability report for zACS, as well as the return and reason codes for the table generation and z/OS UNIX test programs.

| Table 1. Return and Reason codes for potential vulnerability report. | |
|--|---|
| Return code | Description |
| RC: 0 | Success. |
| RC: 4 | <p>Warning.</p> <ul style="list-style-type: none">• RSN: 405 - potential vulnerability• RSN: 407 - PC or SVC cannot be run by unauthorized users• RSN: 408 - PC or SVC undefined• RSN: 409 - service timed out. A test did not complete in the allotted time. This may occur if the service typically requires an extended time to complete or the service is delayed. It could also be an indication of an enabled wait or a loop in the service.• RSN: 40A - The program being tested is not authorized. Most likely this is due to the program being in a data set not in the APF list or the program is linked AC(0)• RSN: 40B - The program being tested was not found.• RSN: 40C - The program cannot be tested because it requires a system key. |
| RC: 8 | Internal Error. |
| RC: C | <p>Environmental Error.</p> <ul style="list-style-type: none">• RSN: C09 - logrec error<ul style="list-style-type: none">– In most cases this is an indication that the logrec data set is full. Clear or replace your logrec data set or use the SETLOGRC ENF option to bypass the logrec data set. If you use the SETLOGRC ENF option, be sure to set it back after completing your scan so to not miss logging logrec events.– If you are running on a virtual machine, ensure the SVC76 option is set to 'VM'• RSN: C0B - There was an error when the program being tested was loaded.• RSN: C0C - The testing program, BPNKMVS, was not executing in an authorized state. This can be because the data set it is contained within is not in the APF list or a data set in the STEPLIB concatenation for the job this is running under is not in the APF list.• RSN: C0D - The testing program, BPNKUSS, encountered an error while setting up the test. Possible reasons include:<ul style="list-style-type: none">– The z/OS UNIX work directory is not accessible– The user does not have an OMVS segment |

Table 2. Return and Reason codes for BPNGPCN, BPNGSVC, and BPNGMVSN.

| Return code | Description |
|--------------------|--------------------------|
| RC: 0 | Success. |
| RC: 4 | Syntax error on MVS PARM |

Table 3. Return and Reason codes for BPNGUSSN and BPNKUSS

| Return code | Description |
|--------------------|--|
| RC: 0 | Success. |
| RC: 4 | Warning. <ul style="list-style-type: none"> • RSN: 25 – Attempt to wait for program completion failed • RSN: 31 – Unable to close loadmod of program to be tested |
| RC: 8 | Application Error. <ul style="list-style-type: none"> • RSN: 1 – Error creating output file of find command • RSN: 2 – Error creating error file for find command • RSN: 7 – Input is too large • RSN: 9 – Too many parameters specified • RSN: 10 – The built find command has too many arguments • RSN: 11 – Invalid work directory specified • RSN: 17 – Error creating stdin file for program to be tested • RSN: 18 – Error creating stdout file for program to be tested • RSN: 19 – Invalid input parameters • RSN: 27 – Invalid program name |
| RC: 12 | Environmental Error. <ul style="list-style-type: none"> • RSN: 3 – Unknown error executing find command • RSN: 4 – Error resetting offset of output file of find command • RSN: 12 – OMVS environment not available • RSN: 15 – Name of program to be tested is too long • RSN: 22 – Error writing to stdin file of program being tested • RSN: 24 – Error invoking program to be tested • RSN: 26 – Error resetting offset of stdin file of program being tested • RSN: 28 – Error opening loadmod of program to be tested • RSN: 29 – Error reading loadmod of program to be tested • RSN: 30 – Error extracting loadmod of program to be tested • RSN: 32 – Stat of program failed |

ABEND 2600 reason code analysis

This topic covers the reason codes that occur from the 2600 ABEND code for zACS and zACM.

Table 4. Reason codes

| Reason code | Description | Explanation |
|-------------|----------------------------------|---|
| 01 | zACS start failed | Check your zACS set up and configuration for any errors. |
| 06 | zACS is not started | An attempt to use zACS was made but zACS is not started. Start zACS. |
| 0A | BPNGENFL not in LPA | BPNGENFL was not found in LPA while starting zACS |
| 0D | zACS already started | An attempt to start the tool was made when the tool is already started. |
| 0E | zACS registration failed | The tool is not registered. |
| 12 | zACS not started task | An attempt was made to run the tool from a non started task. zACS is required to run as a started task. Run zACS as a started task. |
| 13 | zACS incompatible release | An attempt was made to start the tool on an incompatible version of z/OS. z/OS 2.4 and above are supported. |
| 14 | zACS task busy | An attempt was made to run a test while testing is already in progress. |
| 15 | zACS task busy | An attempt was made to run a PC test while testing is already in progress. |
| 16 | zACS task busy | An attempt was made to run an SVC test while testing is already in progress. |
| 17 | zACS task busy | An attempt was made to run an ESR test while testing is already in progress |
| 19 | zACS emulator disallowed | zACS is not permitted to run on an emulator. |
| 1A | zACS security error | The user running a zACS test case must have read access to the BPN.RUN resource in the XFACILIT class. See message ICH408I for more details |
| 1B | zACS PC number max exceeded | More than 256 PCs supplied for a single LX. |
| 1C | zACS testing storage unavailable | Unable to obtain storage needed for testing. Region size on the zACS job may need to be increased. |
| 1D | zACS append failed for log | Failed to open the Potential Vulnerability Log for append |
| 1E | zACS open failed for new log | Failed to open the Potential Vulnerability Log |
| 1F | zACS append failed for summary | Failed to open the Potential Vulnerability Summary Log for append |
| 22 | PCX zACS not Started | PC testing could not be completed because zACS is not started |
| 23 | SVC zACS not Started | SVC testing could not be completed because zACS is not started |
| 24 | EXT zACS not Started | Extended SVC testing could not be completed because zACS is not started |
| 26 | BPNGPC1 not in LPA | BPNGPC1 was not found in LPA while starting zACS |
| 27 | BPNGPC2 not in LPA | BPNGPC2 was not found in LPA while starting zACS |

Table 4. Reason codes (continued)

| Reason code | Description | Explanation |
|-------------|---|--|
| 2F | zACS task busy | An attempt was made to run a MVS test while testing is already in progress. |
| 30 | MVS zACS not Started | MVS testing could not be completed because zACS is not started |
| 33 | zACS task busy | An attempt was made to run a z/OS UNIX test while testing is already in progress. |
| 36 | z/OS UNIX zACS not Started | z/OS UNIX testing could not be completed because zACS is not started |
| 37 | BP NHLKUP not in LPA during table generation | BP NHLKUP was not found in LPA while attempting to generate the PC Table |
| 38 | BP NHLKUP not in LPA during start-up | BP NHLKUP was not found in LPA while starting zACS |
| 39 | zACS Problem with trial date | There was a problem comparing the trial start date to the current date |
| 3A | zACS Trial Expired | Trial is expired |
| 3B | zACS Trial creation failed | Unable to start trial. Check that the /global/zacs directory exists and that zACS has read/write permissions to it. |
| 3C | zACS Trial get status failed | Failed to get the status of the trial |
| 3D | MDB Exit zACS is not started | The MDB Exit could not run properly because zACS is not started |
| 3E | Could not add MDB Exit | The MDB Exit could not be added |
| 3F | BPNGMDBX was not found in LPA while starting zACS | BPNGMDBX was not found in LPA while starting zACS |
| 40 | Could not allocate data set to supplied DD | Could not allocate temporary data set to DD name extracted from IEC130I message during MVS testing. The top half of R0 contains the value from S99INFO and the bottom half contains the value from S99ERROR. |
| 41 | Could not open supplied DD | Could not open temporary data set allocated to DD name extracted from IEC130I message during MVS testing |
| 42 | Steplib too big | More than 2 entries to testcase job steplib |
| 43 | TSO zACS task busy | TSO testing could not be completed because zACS is not started |
| 44 | TSO zACS not started | TSO testing could not be completed because zACS is not started |
| 45 | TSO data set information request failed | Could not get information about the data set containing the module being tested in the TSO environment |
| 46 | Could not open supplied DD | Could not open temporary data set allocated to DD name extracted from IEC130I message during TSO testing. |

Table 4. Reason codes (continued)

| Reason code | Description | Explanation |
|-------------|--|--|
| 47 | Could not allocate data set to supplied DD | Could not allocate temporary data set to DD name extracted from IEC130I message during TSO testing. The top half of R0 contains the value from S99INFO and the bottom half contains the value from S99ERROR. |
| 48 | TSO Environment not available | Could not get establish TSO Environment during MVS table generation |
| 10001 | zACM start failed | Check your zACM set up and configuration for any errors. |
| 1000D | zACM already started | An attempt to start the tool was made when the tool is already started. |
| 1000E | zACM registration failed | The tool is not registered. |
| 10012 | zACM not started task | An attempt was made to run the tool from a non started task. zACM is required to run as a started task. Run zACM as a started task. |
| 10013 | zACM incompatible release | An attempt was made to start the tool on an incompatible version of z/OS. z/OS 2.4 and above are supported. |
| 10019 | zACM emulator disallowed | zACM is not permitted to run on an emulator. |
| 1001D | zACM append failed for log | Failed to open the Potential Vulnerability Log for append |
| 1001E | zACM open failed for new log | Failed to open the Potential Vulnerability Log |
| 1001F | zACM append failed for summary | Failed to open the Potential Vulnerability Summary Log for append |
| 10020 | zACM open failed for new summary | Failed to open the Potential Vulnerability Summary Log |
| 10039 | zACM Problem with trial date | There was a problem comparing the trial start date to the current date |
| 1003A | zACM Trial Expired | Trial is expired |
| 1003B | zACM Trial creation failed | Unable to start trial. Check that the /global/zacs directory exists and that zACM has read/write permissions to it. |
| 1003C | zACM Trial get status failed | Failed to get the status of the trial |
| 11001 | zACM latch obtain failed | ENF Listener failed to obtain latch |
| 11002 | zACM latch release failed | ENF Listener failed to release latch |
| 11003 | zACM latch obtain failed | Started task failed to obtain latch |
| 11004 | zACM latch create failed | Started task failed to create latch |
| 11005 | zACM latch release failed | Started task failed to release latch |
| 11006 | zACM Multiple Job Filters Specified | Both <i>include by job name</i> and <i>exclude by job name</i> filters were specified on the started task. Only one job name filter may be specified. |

Table 4. Reason codes (continued)

| Reason code | Description | Explanation |
|-------------|--|--|
| 11007 | zACM Multiple Module Filters Specified | Both <i>include by module name</i> and <i>exclude by module name</i> filters were specified on the started task. Only one module name filter may be specified. |
| 11009 | zACM PARM on started task too long | See documentation on started task configuration for parm values and lengths. |
| 1100A | zACM PARM on started task has a comma not in the 1 st or 2 nd column | See documentation on started task configuration to understand PARM= values and locations. |
| 1100B | zACM PARM contained a quote within it | This is an invalid character as per the documentation. Remove the quote from the parameter |
| 1100C | Bad number of started task PARM | A number between 1-99 was expected to enable First Failure data capture but was not provided. This number will be used as the matchlim for the BPNZ SLIP. The number should be the first one or two characters in the PARM to enable First Failure data capture without enabling automatic generated SLIP setting. To enable both First Failure data capture and automatic generated SLIP setting, the PARM must begin with a Letter, followed by a comma, followed by the one or two digit matchlim number. |
| 1100D | Description missing from started task PARM | On zACM started task PARM, BPNZ SLIP Description was expected because comma was supplied after matchlim number, but one was not provided |
| 21000 | zACM failed to open MYJOBEX | Failed to open job exclusion data set |
| 22000 | zACM failed to open MYJOBIN | Failed to open job inclusion data set |
| 23000 | zACM failed to open MYMODEX | Failed to open module exclusion data set |
| 24000 | zACM failed to open MYMODIN | Failed to open module inclusion data set |
| 21XXX | zACM failed while processing MYJOBEX | Failed while reading from the job exclusion list. XXX refers to the line that failed. See Filtering section for details on the format of entries in the job exclusion list. |
| 22XXX | zACM failed while processing MYJOBIN | Failed while reading from the job inclusion list. XXX refers to the line that failed. See Filtering section for details on the format of entries in the job inclusion list |
| 23XXX | zACM failed while processing MYMODEX | Failed while reading from the module exclusion list. XXX refers to the line that failed. See Filtering section for details on the format of entries in the module exclusion list. |
| 24XXX | zACM failed while processing MYMODIN | Failed while reading from the module inclusion list. XXX refers to the line that failed. See Filtering section for details on the format of entries in the module inclusion list. |

Table 4. Reason codes (continued)

| Reason code | Description | Explanation |
|--------------------|---|--|
| 25XXX | zACM detected more than 100 job name filter entries | Job name filter entries are limited to 100 entries. Lines that start with a # are comments and do not count towards the number of entries. XXX refers to the line number on which the 101 st job name entry was found. Remove any job name entries past the 100 th to fix the problem. |
| 26XXX | zACM detected more than 100 module filter entries | Module filter entries are limited to 100 entries. Lines that start with a # are comments and do not count towards the number of entries. XXX refers to the line number on which the 101 st module entry was found. Remove any module entries past the 100 th to fix the problem. |

All other Reason Codes are unexpected, Contact z/OS Support for diagnosis.

Chapter 10. Messages

zACS message numbers begin with the three-character component prefix (BPN), followed by a fourth character that identifies the subcomponent. Characters five through seven are numeric. The eighth character is the message type. The following table shows how the zACS messages are defined.

| Table 5. Definition of zACS message numbers | | |
|---|---------|---------------------------|
| Character | Meaning | Where the messages appear |
| U | UI | z/OSMF Logs |

Characters five through seven are numeric. The eighth character is the message type:

| Table 6. Meaning of eighth character is message number | | |
|--|--------------------------------|---|
| Character | Meaning | Action Required |
| I | Informational (status message) | No action required. |
| E | Error | Possible problem that might require action |
| W | Warning | Possible unexpected results that might require action |

BPNU001E **Error Loading *type* Table: An error occurred reading data set *dsname*.**

Explanation

In the message text:

type

The service table type that could not be read. It can be one of PC, SVC, MVS or z/OS UNIX.

dsname

The name of the data set that could not be read

The table data could not be read correctly. The table might show empty data message when this occurs.

User response

Check that data set is not open or in use via the ISPF interface and resubmit the request to generate the table.

BPNU002I **Table Retrieved: Successfully retrieved the table data from *dsname*.**

Explanation

In the message text:

type

The service type of the table retrieved. One of PC, SVC, MVS or z/OS UNIX.

dsname

The name of the data set retrieved.

User response:

None

BPNU003E **Error Generating *type* Table: An error occurred when attempting to generate or refresh the *type* table.**

Explanation

In the message text:

type

The service type of the table that could not be generated. One of PC, SVC, MVS or z/OS UNIX.

The refresh of the table data received an error.

User response:

The data set containing the job to generate the tables is read from the configuration file. Check that the configuration file is not open in edit mode and that correct permission are set to run the job. If the target data set, to where the table is to be generated, already exist, ensure it is not open in edit mode.

BPNU004W ***type* Table Generation Job Already Running: Table refresh job might take longer than normal to complete.**

Explanation

In the message text:

type

The service type of the table to be generated. One of PC, SVC, MVS, or z/OS UNIX.

Table generation is already in progress. All subsequent refresh attempts will be queued behind the currently running job. As a result, the table may take longer than expected to display.

User response:

Wait for the table generation job to complete.

| | |
|-----------------|--|
| BPNU020E | Error On Scan Request: Error encountered when attempting to run scan. <i>response</i> |
|-----------------|--|

Explanation

In the message text:

response

The response received from either the REST API or zACS.

The request to submit a scan failed with the parameters provided. The scan is not started.

User response:

Check that the parameters of the scan request are selected correctly and the high level qualifier is set in the setting panel.

| | |
|-----------------|---|
| BPNU021I | Scan Initiated: Scan job(s) submitted successfully |
|-----------------|---|

Explanation:

Scan request was accepted, and scan has been queued to start.

User response:

None

| | |
|-----------------|--|
| BPNU022W | Service(s) Skipped: One or more <i>type</i> was skipped because it was not callable by unauthorized callers or because it was not installed |
|-----------------|--|

Explanation

type

The service type of the table to be generated. One of PC or SVC.

One or more PC or SVC was not callable by unauthorized caller, or it was not installed.

User response:

None

| | |
|-----------------|--|
| BPNU023E | Configuration For Scan Not Valid: The configuration in <i>dsname</i> is |
|-----------------|--|

not valid for use with zACS GUI.

keyword* must be set to *value

Explanation

In the message text:

dsname

The name of the data set containing the configuration file.

keyword

The name of the configuration keyword that is incorrect

value

The value the keyword must be set to

User response:

Ensure the configuration printLogOutput, printStatus and printSummary are configured correctly.

| | |
|-----------------|---|
| BPNU040E | Could Not Load Results: An error occurred trying to get the scan results from <i>dsname</i>. |
|-----------------|---|

Explanation

In the message text:

dsname

The name of the data set containing the scan results

Attempt to load the results from the scan output data set was not successful.

User response:

Check that the scan output data set is correctly set in the configuration file and user has permission to read the data set

| | |
|-----------------|--|
| BPNU041I | Retrieved Scan Results: Successfully retrieved the scan result data from <i>dsname</i>. |
|-----------------|--|

Explanation

In the message text:

dsname

The name of the data set containing the scan results

Successfully read the scan output data set.

User response:

None

| | |
|-----------------|---|
| BPNU060E | Error Reading Configuration File: Could not access configuration file in <i>dsname</i> |
|-----------------|---|

Explanation

In the message text:

dsname

The name of the data set containing the configuration file

Could not read the zACS configuration file.

User response:

Check that the configuration file is not open in edit mode and the current user has permission to view the configuration file

BPNU061E **Configuration For Scan Not Valid: The configuration in *dsname* is not valid for use with zACS GUI. *keyword* must be set to *value***

Explanation

In the message text:

keyword

The configuration keyword where the incorrect value was detected

value

The value detected at the *keyword*

keyword in the configuration file does not have a valid value.

User response:

Check that *value* is set to a valid, existing data set name

BPNU062E **Error Updating zACS High Level Qualifier: An error occurred when updating zACS installation high level qualifier. *reason***

Explanation

In the message text:

reason

The reason from the REST API

The zACS installation high level qualifier could not be saved to the z/OSMF persistence store.

User response:

Ensure the SAF permission are set correctly to access the z/OSMF persistence store.

BPNU063E **Error Updating Configuration File: Could not update configuration file. *reason***

Explanation

In the message text:

dsname

The data set name containing the configuration file

reason

The reason from the REST API

Failed to update the configuration file. No values are changed.

User response:

Check that the configuration file is not open in edit mode and the current user has permission to edit to the configuration file.

BPNU064I **Error Updating Configuration File: Could not update configuration file. *dsname***

Explanation

In the message text:

dsname

The data set name containing the configuration file

Updated the configuration file with the values from the form.

User response:

None

BPNU065W **Unable To Update Configuration File: Line data will exceed record length. Changes will be discarded**

Explanation:

In the message text: Updates to the configuration data would cause line data to exceed record length. The line to be written exceeds the maximum lrecl of the data set. Any new changes will be reverted.

User response:

Check the configuration file for comments or spacing that may cause the new value(s) to exceed the data set's lrecl. In case of a data set name, ensure a data set name of up to 44 characters will fit on the line.

BPNU066E **Multi Line Entry Detected In Configuration File: Unable to write entry across multiple lines. Starting in line *line number*.**

Explanation**line number**

The first line number of the configuration that contains the multi-line delimiter

The keyword/value combination(s) to be updated is split across multiple lines and cannot be updated from the GUI. No changes have been made.

User response:

Edit the configuration file through ISPF to put the keyword and full value on a single line.

BPNU067E **Error Updating Data Set: Update to data set *dsname* failed. *reason***

Explanation

dsname

The name of the data set that could not be updated

reason

The reason from the REST API

Failed to update the inclusion or exclusion data set.

User response:

Check that *dsname* is not currently open for editing and that the current user has permission to write to the data set.

| | |
|-----------------|---|
| BPNU068I | Data Set Updated: Successfully updated the data set. <i>dsname</i> |
|-----------------|---|

Explanation

dsname

The name of the data set that was updated

Updated the data set successfully.

User response:

None

| | |
|-----------------|---|
| BPNU069E | Error Reading Data Set: Failed to read data set. <i>dsname</i> |
|-----------------|---|

Explanation

In the message text:

dsname

The name of the data set that could not be read

Failed to access the data set.

User response:

Check that *dsname* is not currently open for editing and that the current user has permission to read the data set.

| | |
|-----------------|--|
| BPNU070E | Error Accessing High-Level Qualifier: Did not find the IBM z/OS Authorized Code Scanner that is installed at high-level qualifier <i>HLQ</i>. |
|-----------------|--|

Explanation

In the message text:

HLQ

The high-level qualifier provided by the user

The high-level qualifier that is provided by the user might not be found, or zACS is not installed at that location.

User response:

Update the high level qualifier to the installation location of the IBM z/OS Authorized Code Scanner. The high-level qualifier consists of all but the final

qualifier of the zACS installation, for example if zACS installation is located under the data sets named SYS1.BPN.SBPN* then the high-level qualifier is SYS1.BPN. Do not include the final period.

| | |
|-----------------|---|
| BPNU071E | Error validating configuration file: Configuration file <i>dsname</i> validation was unsuccessful. <i>reason</i> |
|-----------------|---|

Explanation

In the message text:

dsname

The data set name containing the configuration file.

reason

The reason for the failure.

Validation failed for the configuration file. The reason of the first failure is provided.

User response:

Fix the error and save the configuration file.

| | |
|-----------------|--|
| BPNU072E | Error saving configuration file: Temporary file <i>dsname</i> could not be created, configuration file updates could not be made. <i>reason</i> |
|-----------------|--|

Explanation

dsname

The data set name containing the temporary configuration file.

reason

The reason for the failure.

A temporary file is used when validating configuration file changes. The temporary file could not be created for the provided reason.

User response:

Check that the temporary data set does not already exist and that the current user has permission to create the file.

| | |
|-----------------|--|
| BPNU073E | Error in configuration file: The member <i>member</i> does not exist in the data set <i>dsname</i>. |
|-----------------|--|

Explanation

member

The member that was expected but not found.

dsname

The data set in which the missing member was searched.

Expected to find *member* in *dsname* but the member could not be found.

User response:

Check the data set for misspellings of member names.
If the member does not exist, a copy can be obtained
from **SYS1.BPN.SBPNSAMP**.

Appendix A. SMF-Records

Record type 1154 Subtype 84 – zACM Compliance Evidence

Record type 1154 Subtype 84 is written to record zACM compliance evidence when an ENF86 signal is received. Record type 1154 Subtype 84 is mapped by the BPNN1154 macro.

Record type 1154 Subtype 84 header

| Table 7. Record type 1154 Subtype 84 Subtype Specific Data Header | | | | |
|---|----------------------|--------|----------|---------------------------------------|
| Offsets | Name | Length | Format | Description |
| 0 0 | SMF1154_84_TRN | 2 | Binary | Number of triplets: 1 |
| 2 2 | * | 2 | Reserved | Reserved |
| 4 4 | SMF1154_84_S1_Offset | 4 | Binary | Offset to first data section |
| 8 8 | SMF1154_84_S1_Length | 2 | Binary | Length of data section 1 |
| 10 A | SMF1154_84_S1_Number | 2 | Binary | Number of repeated data section 1s: 1 |
| | SMF1154_84_Hdr_End | | | |

Record type 1154 Subtype 84 Data section 1

Data section 1 contains identifying information for zACM including the record version, eye catcher, fmids, and a bit to state zACM is on.

| Table 8. Record type 1154 Subtype 84 Subtype Specific Data Section 1 | | | | |
|--|----------------------------|--------|----------|----------------------------------|
| Offsets | Name | Length | Format | Description |
| 0 0 | SMF1154_84_S1_ZACM_Version | 2 | Binary | Version of this section: 1 |
| 2 2 | SMF1154_84_S1_ZACM_EYEC | 4 | EBCDIC | Eye catcher for the record: ZACM |
| 6 6 | SMF1154_84_S1_ZACM_FMID | 8 | EBCDIC | FMID for ZACM: HAL47C0 |
| 14 E | * | 16 | Reserved | Reserved |
| 30 1E | SMF1154_84_S1_ZACM_ACTIVE | 1 | Binary | 1 means ZACM is active |
| 31 1F | * | 3 | Reserved | Reserved |

Appendix B. Accessibility

Accessible publications for this product are offered through [IBM Documentation for z/OS \(www.ibm.com/docs/en/zos\)](http://www.ibm.com/docs/en/zos).

If you experience difficulty with the accessibility of any z/OS documentation see [How to Send Feedback to IBM](#) to leave documentation feedback.

Notices

This information was developed for products and services that are offered in the USA or elsewhere.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America*

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

*Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan*

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

This information could include missing, incorrect, or broken hyperlinks. Hyperlinks are maintained in only the HTML plug-in output for IBM Documentation. Use of hyperlinks in other output formats of this information is at your own risk.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

*IBM Corporation
Site Counsel
2455 South Road*

Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or

reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use session cookies that collect each user's name, email address, phone number, or other personally identifiable information for purposes of enhanced user usability and single sign-on configuration. These cookies can be disabled, but disabling them will also eliminate the functionality they enable.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at ibm.com/privacy and IBM's Online Privacy Statement at ibm.com/privacy/details in the section entitled "Cookies, Web Beacons and Other Technologies," and the "IBM Software Products and Software-as-a-Service Privacy Statement" at ibm.com/software/info/product-privacy.

Policy for unsupported hardware

Various z/OS elements, such as DFSMSdfp, JES2, and MVS, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

Minimum supported hardware

The minimum supported hardware for z/OS releases identified in z/OS announcements can subsequently change when service for particular servers or devices is withdrawn. Likewise, the levels of other software products supported on a particular release of z/OS are subject to the service support lifecycle of those

products. Therefore, z/OS and its product publications (for example, panels, samples, messages, and product documentation) can include references to hardware and software that is no longer supported.

- For information about software support lifecycle, see: [IBM Lifecycle Support for z/OS \(www.ibm.com/software/support/systemsz/lifecycle\)](http://www.ibm.com/software/support/systemsz/lifecycle)
- For information about currently-supported IBM hardware, contact your IBM representative.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at [Copyright and Trademark information \(www.ibm.com/legal/copytrade.shtml\)](http://www.ibm.com/legal/copytrade.shtml).

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Index

A

abend reason code analysis [87](#)
about [xiii](#)
accessibility
 contact IBM [103](#)
additional panel options [40](#)
assistive technologies [103](#)

B

BPNMVSNM [5](#)
BPNPCNUM [5](#)
BPNSVCNM [5](#)
BPNUSSNM [5](#)
BPNZACS [5](#)

C

capture a dump [71](#)
CEE Dump Suppression [5](#)
command line [26](#)
configuration [5](#)
configuration file [5](#)
contact
 z/OS [103](#)

D

data set protection [5](#)
Data Set Protections and Permissions zACM [79](#)
diagnosis [63](#)

E

example configuration file [5](#)
exclude lists [42](#)

F

fetch vulnerability example [71](#)
Filtering Monitored Services with Include or Exclude Lists
zACM [81](#)
filtering run services [42](#)

H

how to use this document [xiii](#)

I

include lists [42](#)
install the zACS GUI [5](#)
interperting output file [63](#)
interpreting the potential vulnerability output file [63](#)

introduction [1](#)

J

JES [5](#)
Job name filtering zACM [81](#)
job names [43](#)

K

keyboard
 navigation [103](#)
 PF keys [103](#)
 shortcut keys [103](#)

L

locating module names and job names [43](#)
LOGREC [5](#)

M

messages [87](#)
Module Filtering zACM [81](#)
module names [43](#)
Monitor Setup [81](#)

N

navigation
 keyboard [103](#)

O

optional parameters [26](#), [34](#)
Output and Filter Data Set Allocation [79](#)
overview [3](#)

P

panel [28](#), [34](#)
potential vulnerability [63](#)

R

registration [5](#)
return and reason codes [87](#)
running [26](#), [28](#)
Running the Monitor [81](#)
running using the command line [26](#)
running using the panel [28](#)
running with advanced test [42](#)

S

- service authorization [5](#)
- setting a SLIP [71](#)
- setup [25](#)
- shortcut keys [103](#)
- started task [5](#)
- Started Task BPNZACM [79](#)
- storage overlay example [71](#)
- store vulnerability example [71](#)
- summary of changes [xvii](#)

T

- trademarks [108](#)

U

- user interface
 - ISPF [103](#)
 - TSO/E [103](#)

W

- who [xiii](#)

Z

- z/OS UNIX configuration [5](#)
- z/OS UNIX detecting vulnerabilities [5](#)
- zACM Configuration [79](#)
- zACM Diagnosis [85](#)
- zACM Registration [79](#)
- zACM Running [81](#)



Product Number: 5655-ZOS

SC283123-40

