z/Auditing Essentials

VOLUME 4

Are Your PORTs Safe?

Julie-Ann Williams

Barry Deller

Craig Warren

David Hayes

For a technology
to be useful
it must be understood.

It is this understanding of z/OS
that is the mission of
The z Exchange.

Foreword

This little book asks two important questions that must not be allowed to remain unanswered in any organization utilizing mainframes: (1) Are the endpoints of the communications to and from your systems properly controlled? and (2) How do you know? Various aspects of auditing are discussed; including how you obtain assurance about the control of IP communications touching your systems. One assurance you can have right now, irrespective of any audits, is that the endpoints of the communications to and from your mainframes (the Ports) are vital to controlling access to your systems and the data they contain.

We hope this book becomes essential reading for those who are actively working with the configurations and controls relevant to Ports; those who are just learning about this topic and those who are trying to decide if they should care about the control of Ports on the mainframe. In many organizations, the configuration and control of IP networks involving mainframes has been the parlance of those outside of the people who actually maintain and operate the mainframes. As this book illustrates, a comprehensive and capable control environment for IP exists on the mainframe, but its value is only realized when implemented.

Most of us learn something new every time we evaluate IP controls involving mainframes. Please take the time to look through this book and find out if you and your organization might benefit from exploring the configurations and controls discussed. Don't forget the second question as you work through the Port controls: How do you know your system is secure?

David Hayes

1 Table of Content

2 About Audit Essentials

2.1 Overview

In the previous Audit Essentials volumes we addressed various z/OS component Configuration Options. But not every process starts and ends on your System z environment(s). This book deals with internetworking protocols and their related mainframe resources.

It is brought to you by the same team that produced the earlier publications. The aim here is to make the whole process of auditing a System z environment slightly less intimidating and more accessible to people who have already been around the audit industry for a while (whatever the Enterprise Security Manager software) but without requiring you to have had 30 years working with mainframes and how they connect to internet based tasks!

It is **NOT** supposed to teach you everything that you need to effectively audit (or secure) your entire internetworking environment. You will need to work with the specialist teams for various protocols to understand your local environment.

First, a few words about the Team that put this book together…

Julie-Ann Williams has been messing around with IBM mainframe computers for most of the last 40 years. She has been helping Customers to get ready for external audits since 1987.

She has an unusual blend of skills encompassing detailed, classic mainframe knowledge (don't call it legacy!) as well as "newer" technologies like WebSphere, TCP/IP and UNIX combined with communications abilities and Mentoring. She's pretty sure that the last two used to be called having a chat with your colleagues in the good old days.

Julie-Ann took the lead in writing this book ably assisted by a number of Industry Experts including very significant contributions from:

Barry Deller has been helping mainframe customers secure their environments for over 30 years! He is unusual in that he speaks all 3 variants of mainframe security (ACF2, RACF and TSS).

He thrives in situations that exploit his eye for detail (his wife wishes he wouldn't be quite so nerdy at work!).

Craig Warren has been in the z Industry for over a quarter of a century! He laughs at those who say the mainframe is dead and his work on bleeding edge z projects gives him good reason. He's also our resident STIG wrangler.

David Hayes has worked around information systems, specifically in the area of internal control, for several decades. He recently retired from the US Government Accountability Office where he worked primarily with controls related to mainframe environments. He says that his favorite comment from people he's audited is: "How on earth did you find THAT?!"

And about the book...

As usual with books from this team, there's nothing to memorize. There will be no tests.

What you will find are de-jargon-ified explanations of concepts and specific

parameters. It is a distillation of a number of people's personal experiences in the field written in "Clear English".

Previous z/Auditing Essentials volumes have looked at such diverse things as: CICS security implementation; fundamentals of System z configuration as it impacts security – "Front Doors to System z"; external security manager (ACF2, RACF and TSS) configuration and use – "Back Doors to System z"

This volume looks at the z/OS internetworking protocols which can be used to gain access to mainframe resources.

It is the end result of a conversation that Paul Robichaux ("The Boss" at NewEra Software Inc.) had with one of the best known figures working in mainframe security today, Stu Henderson (www.stuhenderson.com). It was one of those conversations that go back and forth on a topic. This time the topic was the pain involved in identifying all of the disparate elements that go into securing TCP/IP ports in the mainframe environment.

2.1.1 Icons Used in this Book

Icons are used as shorthand ways of saying the same important things.

The following icons indicate expert knowledge that you will need in order to understand how to audit System z:

DISA STIG

The United States Department of Defense (specifically the Defense Information Systems Agency (DISA)) creates and maintains an extensive collection of Security Technical Implementation Guides (STIGs). These extensive guides document the approved configuration for controls for DISA's great many IT systems. The STIGs contain a wealth of information not readily available elsewhere and have become adopted broadly across the information systems industry as control baselines for platforms as diverse as your cell phone and your mainframe!

Organizations that do business with the US Government are likely to be required to maintain their systems in compliance with the appropriate DISA STIGs.

The STIGs contain thoughts about the setup of z/OS internetworking and we agree with the vast majority of those recommendations. But the z/OS DISA STIGs do not provide control configuration advice for ALL of the relevant parameters. Where there is a missing STIG element, we will try to provide advice from a non z DISA STIG that falls within a similar category.

They even have a twitter feed https://twitter.com/USDISA

The Author

Any advice listed alongside the "White Hat" icon (aka "Good Hacker" or "Ethical Hacker" for those readers outside of North America) will be from: either me (Julie-Ann Williams) based on my 30+ years of experience working with Customers and their mainframe security needs; or my team of Co-Authors and the Peer Review Team - without whom this book would have been impossible!

Between us we've seen pretty much everything that mainframe security can throw at us! And we'd like to share that knowledge with you ☺ One last thing… this logo is only shown so you can see when we/I disagree with the STIG and/or other industry "advice" or at least offer an alternative.

2.2 What is an Audit?

This is a tough one to answer quickly. A financial audit of an organization tests that organization's defined policies against reality, and tends to start (when you have a good Auditor) with a request being made for an end user to perform a transaction e.g. getting a bank teller to make a financial action. This activity is then followed through all of its interaction points with systems.

So, for example, the end user might logon to a web site but the actual data may be being served from DB2 on the mainframe. Each interaction with a system is tested to see why the user was able to perform the operation and who else might be able to do so.

So the use of the term audit is troublesome as it means so many different things in practice.

Since audits, in any form, focus on controls like those covered by this book, it is important for you, gentle reader, to have a clear understanding of the audit exercise you are, or will be, involved with. Here are a few things to consider before you rush off to get a specific set of reports generated when the request to participate in an audit comes in to you…

The definition of the word audit comes from auditory. In essence, it is listening and, by inference, observing. So, audit at the most basic level involves the observation of something for some purpose.

Operationally, audit is often the term used in reference to the monitoring of some aspect of information systems. As a business process, audit means the evaluation of some aspect of an organization's internal control in relation to its control objectives.

While there is considerable focus on compliance in terms of audit, it is important to recognize that an organization which has achieved compliance is at the **MINIMUM** level of controls permissible. You still have more to do!

In a competitive environment filled with SO MANY requirements from SO MANY sources, you not only have to achieve or exceed compliance for your organization, but you need to do so in a manner that gives your employer an operational and/or cost advantage over their peers. The organizational approach to accomplish this involves the development and adoption of effective control objectives (or goals) and a system of internal control (also known as management control) to implement and measure your performance against your objectives.

Understanding this, the term audit can be workably defined, in the context of this book, as any process that allows for the measurement of the effectiveness of specific controls against some defined control objective.

Many readers of this book are likely to have a focus on information security. While effective information security is essential for any organization and dominates many compliance requirements, you must understand that information security is a component of internal control which encompasses all of the decisions an organization makes (good or bad) about how it conducts itself. As you read this book and work with the security controls in systems, remember that security controls without clear and effective organizational support for the implementation, operation and maintenance of those controls (internal control) will probably not create optimal/sought-for results.

In today's audit environment, a financial audit is actually TWO audits: one is a test of the accuracy of the financial information that has been reported to stockholders and regulatory bodies, while the other audit is a test of the organization's internal control

over financial reporting (how likely is it that the organization's process of tabulating the numbers gets it right?). Since virtually every financial transaction involves computers, a major part of the audit of internal control over financial reporting is an audit of the behavior of the computers. In an information system environment including z/OS, there are many techniques and approaches to achieve a specific level of control.

As you use this book and work with systems, keep focused on the level of control desired or mandated, not just on the particular settings of individual controls. Remember, we are working in systems of controls that must function together to achieve the desired results.

What we IT folks typically become involved with is considered a security audit. More often than not, these will include an examination of the integrity of your operating system. Someone will look at the contents of your RACF (or ACF2 or TSS) database to see how the operating system elements are secured. It relies on checking a known set of parameters against a list telling the auditor what they should be set to. This type of audit rarely, if ever, touches on the data that has actual value to your organization.

We would categorize the series of z/Audit Essentials books as of a list of appropriate parameters for ensuring the integrity of your operating systems.

2.3 Internetworking 101

The drive to enable the mainframe to be at the heart of business leaves us "old guard" System z folks with a dilemma. We must allow our business processes to connect to the mainframe because that's where the data lives, but creating a balance whereby only authorized processes can do so presents a challenge.

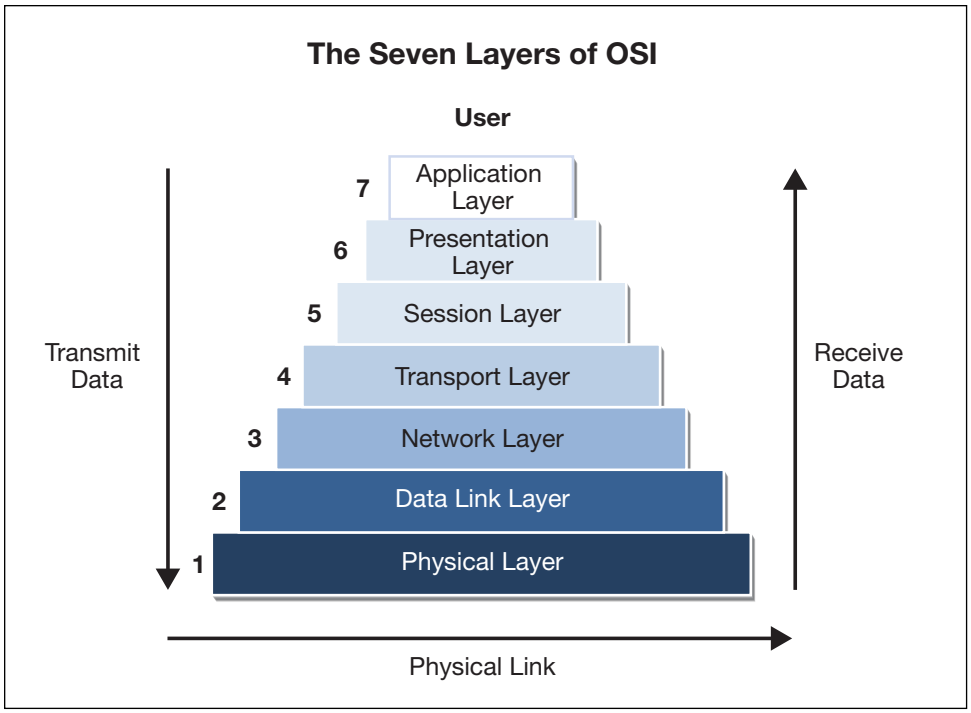
Our mainframe security specialists are great at their jobs but this is an area that few understand well. So I thought we'd start off with a brief, hopefully not too technical outline of what you need to know to follow the content of this volume (whilst acknowledging that I still need to impart a certain amount of specific language). It will **NOT** leave you feeling like an internetworking guru!

Firstly it's important to understand that any and all movement of data across the interwebs is governed by "protocols". Think of protocols as the rules that define how a particular type of communication is managed. It's like the basic grammar of a language with special emphasis on punctuation use.

Protocols are deployed at specific levels of software/hardware and are adhered to by ALL manufacturers of equipment which can be attached to the internet. And just before we launch into geek-speak… the protocols define how the software works and exactly where on the hardware that's implemented. To map protocols to hardware we use the "7 Layers of OSI" model.

Tech speak alert!

- In the late 1970s/early 1980s the International Standards Organization (ISO) developed the Open Systems Interconnection (OSI) model.
- It divides network communication into seven layers.
- Layers 1-4 are considered the lower layers, and are mostly concerned with moving data around.
- Layers 5-7, the upper layers mostly concern application-level data.
- The whole structure is usually represented as a layered pyramid:



We need to read the diagram from the bottom up, so the physical layer (1) represents the hardware communication protocols. Layer 2 is mostly concerned with moving data around the network with the next layer (3) representing the networking communications protocols like IP, and layer 4 being the last of the "invisible" layers which concerns itself with the correct delivery of packets and so forth including TCP and UDP. The session layer (5) is where we start to get more interested. It's where things like Telnet, SMTP and FTP are represented. Layer 6 is where the correct handling of data is represented. It gets involved with things like compression and/or decompression of data. And the top layer (7) represents the interface between the computer and a carbon-based life-form aka human.

Although… the reality of layer 7 in modern architectures is the backend system handing off data used (eventually) by an HTTP server in one direction to paint a user's screen in one direction only. The other way, some application server is handing off data to the backend to process into the system of record. Not so much a direct human interface in terms of mainframes these days!

Most people are aware of TCP/IP and think of it as the network protocol that runs the internet. This is one of those cases where 'most people' are misinformed. TCP/IP is in fact a suite of protocols that control certain forms of communication across a network like File Transfer, Mail etc. The internet is just a collection of these networks (or sub-nets) that are connected, Even the term internet is just a contraction of the term INTERconnected NETwork.

Now you have an outline of some of the language used in this unique part of the IT world and you genuinely know everything that you need to about the Seven Layer OSI model! Let's talk in more detail about TCP/IP Ports.

2.3.1 TCP/IP PORTs

Wikipedia has the following to say:

"In computer networking, a port is an endpoint of communication."

That is, it represents a specific point of entry to a computer system running a TCP/IP stack. The port is defined at the host to give detailed instructions on how it can be used and what process it will link to once a connection has been successfully established (a process known as binding).

A TCP/IP port represents a "Front Door" to your system.

You should always know which ports are likely to be found in use on your systems.

And if you find ones that aren't allowed on your system that are in use (maybe because some external force has convinced your mainframe that it's allowed to start a service?) automate the response to immediately shut them down!

The ports don't always show up just because they are allowed to be used. And an unscrupulous hacker might take advantage of being able to start communication using a surprise port (google the Pirate Bay hack of various Scandinavian mainframes).

If you're not monitoring all activity, you'll never know!

You can use port access control to protect against unauthorized use of ports. Specifically, you can control an application's ability to explicitly bind to, or listen on, specific TCP and UDP ports or port ranges by either reserving particular ports or by controlling access to unreserved ports enhancing the security of communication between endpoints. Ensuring that users use a specific entry point (SOCKET call which includes both the port and the IP address) into your system means that you stand an outside chance of being able to understand what has happened when things go awry.

But none of that happens by default!

And if it were easy, there would be instruction lists out there to follow to implement safe and secure ports. Rest assured that any "quick fix" solution that you or I could think of isn't the magic bullet you think it is. I have a number of young hacker friends who can find their way past that particular "road block" with ease. Oh, and be prepared to have heated conversations about security vs performance!

2.3.2 Just another platform?

A good friend of mine (Dr. Herbert Daly) is an Academic at the University of Wolverhamptons' School of Mathematics and Computer Science. He's always had an interest in mainframes and often gets guest lecturers from the industry in to talk to his students as well as participating in the annual "Master the Mainframe" competition run by IBM.

Herb advises his students to follow their dreams and not get bogged down in unnecessary detail. So whilst many of these students have peers at other universities who scoff at the idea of working on "such an outdated platform", they themselves understand the merits of working on, what Herb calls "Just Another Platform".

So whilst there are relatively few Best Practice recommendations in the arena of mainframe internetworking, there are over-arching bits of advice at a hardware level or even places where the advice for another platform makes sense to apply to the mainframe. This book will look at any and all advice that's available in the public arena.

There are some pretty solid "Rules of Thumb" that can be applied to securing and/or auditing any environment…

- Always educate your Users on what security means to you
- Always have a way to find out who owns what
- Always question requests for access
- Never grant more access than is actually needed
- Never grant access for longer than it is needed
- Never stop questioning requests for access!
- Strictly control **ALL** changes to the system's configuration

And of course, don't forget to:

ALWAYS MONITOR EVERYTHING THAT HAPPENS ON YOUR SYSTEM!!!

If you're not monitoring it (by which I mean recording specific audit data i.e. SMF and syslogd records in the mainframe environment) then you can't prove what has happened if/when the bad guys get into your system.

In days gone by, all we were really auditing for was failing access requests. Today we need to take a much more robust approach in order to be able to confirm: who did what to which resources and when.

2.4 What's the problem with Ports?

The reason for the creation of this book can be summed up by asking 2 simple questions:

1. Are your PORTs Safe?
2. How do you know?

If you've ever been asked to either secure or audit a z/OS installation that's making use of off-host activity then it's likely that you've answered "I don't know!" to either or both of these questions. And there isn't a whole heap of stuff out there to help you get to the right answers.

This book looks at the configuration control elements in and around the following TCP/IP and UDP elements:

- PROFILE
- DATA
- RESOLVER
- TELNET
- FTP
- SMTP
- PAGENT

This in no way represents the entirety of your z/OS internetworking protocols! What it does is give tight focus on a few of the critical paths into your systems in a not terribly well controlled world (internetworking NOT z/OS!!).

2.5 The Configuration of Control Elements

2.5.1 Overview

We define “Control Elements” as those parameters which affect the shape of your mainframe internetworking implementation – how you see it AND how external users will see it.

With the complexity available for internetworking connections, you can expect some of the discussions to get a bit technical. We’ve tried to keep it as simple as possible but where the subject is so broad that we can’t distill all of the information you need here then we urge you to do a bit of independent research. Google® can be your friend.

2.5.2 Control Element Descriptions

Throughout the rest of this book, the sub-headings in the format x.x.1 (or x.x.x.1 etc.) will provide a Description of the parameter being explained. A Description is a textual explanation of the Control Element and how it “fits” in the overall system. It will also include the format of the specific parameters involved and where you might find it.

2.5.3 Control Element Recommendations

Sub-headings in the format x.x.2 (or x.x.x.2 etc.) will provide a discussion of the reasons behind making a decision on values for parameters. A Recommendation is a textual discussion of the reason for a Control Element Rule.

2.5.4 Control Element Rules

Our recommended Parameter/Options settings can be found under sub-headings in the format x.x.3 (or x.x.x.3 etc.). This will include the actual recommended value(s) and where necessary a brief explanation of what can go wrong if you don’t choose this setting. A Control Element setting is a binary (ON|OFF) or quantitative Value (1, 2, 3 etc.).

2.6 What z/OS components are included in the book?

z/OS Communications Server exploits z/OS UNIX services even for the more traditional z/OS environments and applications. So it’s vital that a full-function z/OS UNIX environment (including a z/OS UNIX file system and a security product) is defined and active before z/OS Communications Server can be started successfully. This doesn’t necessarily include any kind of network security definitions!

z/OS Communications Server provides the networking protocols (both SNA and TCP/IP) for z/OS. And it supports two TCP/IP environments:

- A native z/OS environment in which users can exploit the popular TCP/IP protocols in z/OS application environments such as batch jobs, started tasks, CICS applications, etc.
- A z/OS UNIX System Services environment that lets you create and use applications that conform to the POSIX or XPG4 standard (an elderly UNIX specification still relevant today).

This complex implementation makes it somewhat harder to gather all of the information that you need in order to understand exactly what’s defined in your system because it can be held in a multitude of places. And that’s without thinking about the way that a single named parameter can have multiple uses based on location or that an embedded DELETE can completely change the picture!

To make life even more complicated, PAGENT (Policy Agent – pronounced "P Agent" despite what it looks like ☺ It's the software that "manages" all of your z/OS "policies". More on this later…) is often only installed and running on z/OS because some other piece of software has required its presence. If the install has been automatically triggered by SMP/e then you may not even be aware that it was about to be deployed!

Who gets responsibility for looking after the product under those circumstances? Well it's typically assigned to the team that first installed the software. Even worse, who is responsible for ensuring that PAGENT doesn't open up any exposures on your system? Again, typically it's a non-owned environment and that makes for tricky controls.

At the very least you need some confidence that the parameters used to control the shape of these elements aren't changed without you knowing about it!

In an environment that's seriously lacking in documentation, it came as a surprise (to me!) to find that the RACF STIG (v6r38) even contains a "Rule" directly relating to the multiple potential locations for UNIX service configuration files on z/OS. It's a bit wordy (and will be the only STIG included in full) but if you haven't used the STIGs before this should show you that you don't need special training to be able to use these resources and improve the overall security profile of the organization for who you work:

Group ID (Vulid): V-3215
Group Title: ITCP0010
Rule ID: SV-3215r2_rule
Severity: CAT II
Rule Version (STIG-ID): ITCP0010

Rule Title: Configuration files for the TCP/IP stack are not properly specified.

Vulnerability Discussion: The TCP/IP stack reads two configuration files to determine values for critical operational parameters. These file names are specified in multiple locations and, depending on the process, are referenced differently. Because system security is impacted by some of the parameter settings, specifying the file names explicitly in each location reduces ambiguity and ensures proper operations. Inappropriate values could result in undesirable operations and degraded security. This exposure may result in unauthorized access impacting data integrity or the availability of some system services.

IAControls: DCCS-1, DCCS-2

Check Content:

a) Display the active started tasks executing on the domain using SDSF, or equivalent JES display product, and locate the TCPIP started task.

If TCPIP is inactive, review the procedure libraries defined to JES2 and locate the TCPIP JCL member.

Automated Analysis

Refer to the following report produced by the IBM Communications Server Data Collection:

- PDI(ITCP0010)

b) Ensure the following items are in effect for the TCPIP started task JCL:

1) The PROFILE and SYSTCPD DD statements specify the TCP/IP Profile and Data configuration files respectively.

2) The RESOLVER_CONFIG variable on the EXEC statement is set to the same file name specified on the SYSTCPD DD statement.

c) If both of the above are true, there is NO FINDING.

d) If either of the above is untrue, this is a FINDING.

Fix Text: Review the TCP/IP started task JCL to ensure the configuration file names are specified on the appropriate DD statements and parameter option.

During initialization the TCP/IP stack uses fixed search sequences to locate the PROFILE.TCPIP and TCPIP.DATA files. However, uncertainty is reduced and security auditing is enhanced by explicitly specifying the locations of the files. In the TCP/IP started task's JCL, Data Definition (DD) statements can be used to specify the locations of the files. The PROFILE DD statement identifies the PROFILE.TCPIP file and the SYSTCPD DD statement identifies the TCPIP.DATA file.

The location of the TCPIP.DATA file can also be specified by coding the RESOLVER_CONFIG environment variable as a parameter of the ENVAR option in the TCP/IP started task's JCL. In fact, the value of this variable is checked before the SYSTCPD DD statement by some processes. However, not all processes (e.g., TN3270 Telnet Server) will access the variable to get the file location. Therefore specifying the file location explicitly, both on a DD statement and through the RESOLVER_CONFIG environment variable, reduces ambiguity.

The systems programmer responsible for supporting ICS will ensure that the TCP/IP started task's JCL specifies the PROFILE and SYSTCPD DD statements for the PROFILE.TCPIP and TCPIP.DATA configuration files and TCP/IP started task's JCL includes the RESOLVER_CONFIG variable, set to the name of the file specified on the SYSTCPD DD statement.

CCI: CCI-000366

As I said, wordy but understandable and pretty explicit in what you should be doing if you do not satisfy the conditions. If you do direct business with the US Government then these shouldn't be a surprise to you (you're already declaring that you do comply!). If you don't do business with the US Government then the DISA STIGs represent an accessible way of keeping up to date with the ever changing security threat landscape.

This book only looks at the elements required to be able to understand how your ports and PAGENT handle security and will be limited to the following topics:

- General Ports (UDP/TCP)
- PORTRANGE
- UNRSV
- RESTRICTLOWPORTS
- NETACCESS
- GLOBALCONFIG EXPLICITBINDPORTRANGE
- UDPCONFIG EPHEMERALPORTS
- TCPCONFIG EPHEMERALPORTS
- DATA
- RESOLVER
- PAGENT
- Intrusion Detection
- Policy Based Routing

- Specific access ports (FTP/TN3270)
- TELNET
- FTP
- External Security Manager

Sometimes the defaults are OK. Sometimes they aren't!

Sometimes parameters look like a perfect performance option so you implement them only to find that it opens your systems up by degrading security! Be very careful when implementing changes ☺

There's a whole industry growing out there around the management of ports, protocols and services. It's known as PPSM (Ports, Protocols and Services Management). Don't underestimate the work involved in managing this stuff let alone auditing it all!

2.6.1 The TCP/IP Started Task

Just before we move into the deeply technical world of TCP/IP protocol parameters…

There are a number of things we need to talk about which specifically deal with how you start TCP/IP on a mainframe i.e. the Started Task(s). Any task that needs to use z/OS resources is started using something called Job Control Language or JCL. The JCL that is used to start your TCP/IP protocols is a very sensitive piece of data.

STIG ITCP0060 states that started tasks for the Base TCP/IP component must be defined in accordance with security requirements. Their justification is a little specific but it is worth taking extra care because failure to get it right could prevent TCP/IP from initializing as expected:

"The TCP/IP started tasks require special privileges and access to sensitive resources to provide its system services. Failure to properly define and control these TCP/IP started tasks could lead to unauthorized access. This exposure may result in the compromise of the integrity and availability of the operating system environment, and customer data."

The guidelines provided (i.e. if these are true then there is no audit finding) are summarized here:

- The userid deployed to start any instance of TCP/IP must have the following properties:
  - RACF NOPASS attribute to prevent signon with the userid
  - z/OS UNIX attributes: UID(0), HOME directory '/', shell program /bin/sh
  - A matching entry in the STARTED resource class exists enabling the use of the userid(s) to start the TCPIP stack.

Whilst we're discussing the base TCP/IP Started Task, we need to also mention the Unix parameters associated with it. These are typically found in the Unix Systems Services file /etc/inetd.conf. STIG ZUSS0014 states that there can be issues if you get it wrong. Specifically, it says:

"Undesirable values can allow users to gain inappropriate privileges that could impact data integrity or the availability of some system services."

Of course the TCP/IP started task isn't the only element that needs to be controlled. You'll see both references to specific STIGs and the DISA logo where these values are relevant to the other elements I'm going to discuss in the detail part of the book. Let's get started…

3 General Ports (UDP/TCP)

3.1 A GENERAL OVERVIEW

At its simplest, ports are the mechanism by which systems/users are able to connect to other systems using the TCP/IP and/or UDP protocols.

You can control access to particular ports by port number, by reserving the port using the PORT or PORTRANGE profile statements in your TCPIP.PROFILE data (after quite a lot of other set up work!). And you can use the PORT and/or PORTRANGE statements to reserve well-known or configured ports for the applications that need to bind to them. But this isn't the default. By now you should be using the optional SAF parameter to provide additional access control through your ESM.

Port numbers 1-1024 are generally considered to be 'well-known' ports in the world of internetworking. They are the obvious choice for a hacker trying to get your system to talk to them and need special attention.

In this chapter we look at the parameters which control general UDP and TCP ports working our way through each relevant parameter (this book should in no way be viewed as a way to understand ALL of the work involved in defining z/OS internetworking).

3.2 WHAT CAN GO WRONG?

The parameters in this section have a wide cross section of usage, so getting it wrong can mean anything from: someone not being able to sign on to the system they normally use; all the way through to a hacker deliberately using vulnerabilities to gain access to your systems and/or data!

3.3 THE BEST OF PRACTICES

As with other computer communications, the best of practices is represented by tying the system down as tightly as possible and making sure anyone with elevated access to anything important is closely monitored.

In general though, the best practice for TCP/IP usage is to enable all of the appropriate options using at least the minimum levels. This fact is reflected in the National Institute of Standards and Technology (NIST) and related Defense Information Systems Agency (DISA) STIG standards.

The emphasis here is to **NOT** leave anything undefined. This goes back to the ownership concept I introduced earlier. Establishing "ownership" of both resources and applications is vital in establishing a stronger link between the geek stuff and management concepts needed during an effective audit.

3.4 PROFILE

The TCP/IP PROFILE (or PROFILE.TCPIP) contains the set of parameters that is used to initiate the TCPIP daemon started task. And that's as accurate as I can be!

The search order z/OS uses to identify the location of these PROFILE.TCPIP parameters is:

1. //PROFILE DD statement

   This will be a member in a JES2 PROCLIB concatenation which is a list that will have been set up by your z/OS Systems Programmers or Operations Analysts.
2. *jobname.nodename.*TCPIP
3. TCPIP.*nodename*.TCPIP
4. *jobname*.PROFILE.TCPIP
5. TCPIP.PROFILE.TCPIP

The "words" in italics are system variables and will be filled in depending on other settings within the z/OS system on which TCP/IP is being started. And the first file found in the search order is used. So it's vital to keep control over who can make changes to data (or create an element higher up the search order than you use in your systems) in all of these source locations. So vital that there's a STIG specifically about it (ITCP0010) which suggests that you always specify the //PROFILE DD card in the TCP/IP started task JCL and another that discusses correctly securing the datasets (ITCP0070).

For such an important and impactful set of parameters, its location really is unpredictable without deep examination of your definitions. With luck there will be documentation that tells you where to find your parameter library. If not, the process is explained in z/Auditing Essentials volume 1 ☺

Here's a handy summary of the PROFILE.TCPIP parameters with a description of what can go wrong if they change without your knowledge:

Parameter	Used for:	What can go wrong:
DEVICE	TCP/IP supports more than a dozen different types of device attachments to the network. There are two statements used to configure any adapter for TCP/IP on a z/OS host: • DEVICE statement • LINK statement	If I can change your DEVICE statements, I am in control of the shape of your network.
LINK	Parameter triplet to DEVICE and HOME which controls the devices which are attached/contactable in your network.	If I can change your LINK statements, I am in control of the shape of your network.

HOME	Many mainframes have a second network card defined and its home address would be included in the HOME statement.	The HOME statement's parameters consist of IP address and link name pairs. The HOME statement must include an IP address and link name pair for every hard-coded link active in the stack. If I change these relationships then I change the whole way your TCP/IP network functions (up to and including being able to route traffic between LPARs you thought were isolated!).
IPCONFIG	IPCONFIG controls deal with a number of things including datagram forwarding. Forwarding is the act of moving a datagram between two different networks or subnetworks. TCP/IP on z/OS is usually a multi-homed host. In other words, it has more than one IP address (link) associated with it. TCP/IP can be configured to not allow any datagrams to be forwarded. This prevents hackers from using your mainframe as a router. The option controlling this is the **DATAGRAMFWD** option.	Quite often, the mainframe has at least one network card connecting it to the network, with one or more specialized links connecting it to other hosts. If the links are redundant (that is, the routes to each link have equal costs), the IP layer can be configured to utilize all routes of equal cost. The option controlling this is the MULTIPATH option. If I change this parameter then I can use your system to run a “man in the middle” attack. Any hacking I do will look like you did it!
TCPCONFIG	The most significant parameters within this statement block are the parameters controlling the size of send and receive buffers. These parameters can have a significant impact on network performance, particularly when doing bulk data transfer.	If I change these values I can cause you unexpected performance problems which could lead to a Denial Of Service.

UDPCONFIG	Similar to TCPCONFIG but specifies your UDPCONFIG controls. The significant parameters control ephemeral UDP port usage, restriction of low ports and various checksum functions.	If I change these values I can cause similar problems to those affected by TCPCONFIG but for different applications. IF I can change both TCPCONFIG and UDPCONFIG you can consider yourself pwned. Yes pwned it's a hacker insult – look it up while I nose around your system!
BEGINROUTES	The big advantage of static routing is the simplicity. A static route identifies a destination and the appropriate link to take to reach that destination. Static routing usually takes advantage of default routes: when the destination is not explicitly coded, send the packet to the default router and let that router figure out how to get the packet to its destination.	So static routing is easy but it's not very resilient. Most organizations don't use it. Instead, a dynamic routing protocol such as OSPF is used. Dynamic routing takes more effort to plan and set up, but once the planning stage is completed, the network effectively takes care of itself. It can also make more efficient use of network topology: instead of dumping everything to a default router, OSPF can take advantage of more intelligent network design. As a hacker, I can exploit static routing to divert traffic to somewhere away from your secured environment to deal with it.

AUTOLOG	All IP applications require the TCP/IP stack to be running to work correctly. It is usual to start the associated IP servers at the same time as TCP/IP. Starting the applications can be accomplished by the AUTOLOG statement block (with assistance from a PORT statement block). The AUTOLOG statement contains a list of started task names that should be started and remain functional while TCP/IP itself is running. By default, TCP/IP checks every 5 minutes for an active service.	If I remove a service from AUTOLOG, it won't be started automatically AND the TCP/IP task won't monitor to make sure it's working even if I do start it manually. Also, autologging has some limitations. For example, in mainframe land, it's quite usual to be running more than one TCP/IP stack at the same time. An application like FTPD makes its services available to all active TCP/IP stacks in the LPAR. In this case, FTPD is referred to as a generic server. If FTPD is running with multiple TCP/IP stacks, then autologging could result in some confusion, as each TCP/IP stack attempts to stop and restart the FTPD server. The way that this works gives a hacker scope to cause a Denial Of Service across your entire mainframe TCP/IP network. You can override this behavior and have a generic server associated with a specific TCP/IP stack only. Servers which can be controlled this way: • DCAS • FTPD • RPCBIND • SNTPD • syslogd • TIMED • TN3270E TELNET • z/OS UNIX POPPER • z/OS UNIX Portmap • z/OS UNIX REXECD • z/OS UNIX RSHD • z/OS UNIX TELNETD
SMFCONFIG	Configures SMF logging using standard records (118 and 119).	If I can switch this off I can browse around your system without there being any record of it! - STIGs AAMV0380
SMFPARMS	Configures SMF logging using non-standard records (you decide the record numbers).	If I can switch this off I can browse around your system without there being any record of it! - STIGs AAMV0380

The remaining PROFILE parameters require a slightly more in-depth explanation. I'll try to keep it in plain English but bear with me when it gets a bit technical…

3.4.1 PORT

3.4.1.1 Description

IBM says of z/OS TCP/IP ports: Use the PORT statement to reserve a port for one or more specified job names or to control application access to unreserved ports.

I say: PORTs are one of the most likely entry point a hacker is going to use to get into your system. There are lists of "well known ports" at verified IP addresses available for free on the internet (there are even lists of confirmed internet facing mainframes!). If you don't keep absolute control over who can use them, you're asking for trouble! This is not an area you can avoid just because your hardware happens to be a mainframe.

The PORT statement has a number of parameters and options. What those parameters and options are set to can affect how and by who a port can be used. But, by default, all ports are available to all users from all locations. It's up to your organization to define its security requirements for these "front doors" to your systems but if it were up to me… all z/OS installations should be using the external security manager to both control the who/what/when but also to provide a complete audit trail of any activity associated with the ports.

I'll describe some of the PORT statement parameters in the following table but please don't assume that this is everything! It's just the ones that we're focusing on in this book. They are key word parameters so if you don't specify them, they don't exist:

Parameter	Used for:	What can go wrong:
number	Specifies the PORT being defined. The same port number can appear in more than one PORT statement with different users or more than once in the same PORT statement.	This port cannot appear in a range specified by the PORTRANGE statement. If a PORTRANGE statement including this port number is specified prior to this statement, this port is ignored. If the PORTRANGE statement follows this statement, the PORTRANGE statement is ignored. These are critical parameters and you must ensure appropriate controls over who can access the data.
UNRSV	Indicates any unreserved port (any port number in the range 1 – 65535) that has not been reserved by a PORT or PORTRANGE statement. For UDP, access control is applied when an application issues a bind to a particular port number to establish a local port. For TCP, access control is applied depending on the value of the WHENBIND or WHENLISTEN parameter	This tells the system which applications or users are permitted to access application-specified unreserved ports so changing it can either give access to unexpected people or even stop your services working causing a "Denial Of Service". PORT parameters, changing this value can cause unexpected results.

UDP or TCP	Indicates whether this is a UDP or TCP PORT.	Like so many of the PORT parameters, changing this value can cause unexpected results.
RESERVED or *jobname*	This optional parameter Indicates either that the port is not available for use by any user or that it is reserved for a specific JES2 jobname. You use RESERVED to lock certain ports. It is valid for either TCP or UDP protocols and has a number of sub-parameters: NOAUTOLOG DELAYACKS | NODELAYACKS SHAREPORT SHAREPORTWLM BIND ipaddr	Like so many of the PORT parameters, changing this value can cause unexpected results.
SAF resname	This parameter allows you to associate the PORT with a RACF profile (defined with UACC(NONE)). And that lets you specifically grant READ access only to those users who need to be able to. The format is: EZB.PORTACCESS.*sysname.tcpname.resname* where: • EZB.PORTACCESS is constant • *sysname* is the value of the z/OS &SYSNAME. system symbol • *tcpname* is the name of the procedure used to start the TCP stack • *resname* is the 1-8 character value following the SAF keyword	If I can change the profile to one I DO have access to then I have unrestricted listening capability to that PORT. Or, if I change the profile to one that doesn't have anyone in the access list I have provoked a "Denial Of Service" attack. For example by preventing your TCP/IP stack from being able to "listen" on its expected ports. You should be treating your PROFILE.TCPIP data with great care!
DENY	Indicates that port access should be denied and can only be specified for unreserved ports (on the PORT UNRSV statement) and only when the specified jobname is an asterisk (*).	Like so many of the PORT parameters, changing this value can cause unexpected results.

WHENLISTEN	WHENLISTEN indicates that port access control is targeted to TCP applications that are acting as servers (i.e., applications able to accept incoming client TCP connections) that issue an explicit bind to a user-specified unreserved port. Permission to use the unreserved port is determined when a TCP listen is issued. If a listen is not issued, no access control check is made. The WHENLISTEN parameter is not available for the UDP protocol, and it is the default for the TCP protocol.	Like so many of the PORT parameters, changing this value can cause unexpected results.
WHENBIND	Indicates that permission to use an unreserved port is determined when an explicit bind to a specific local port is issued. This is the default, and only option, for the UDP protocol, and it can affect UDP applications that bind to a specific local port. If the WHENBIND parameter is specified for the TCP protocol, it can affect TCP client applications that bind to a specific local port for outbound connections.	Like so many of the PORT parameters, changing this value can cause unexpected results.

You can and should be reserving ports (either individually or in ranges) for use by a specific (or wildcarded) job name even if you aren't using SAF. If you do this then the ports are only available for use by authorized processes/users. If you don't, ANYONE CAN GET IN!

Any port that you don't want to be used can be locked out using the RESERVED parameter.

If you'd rather keep control of who is coming into your system and what they can do once they get there (verifying that the user ID associated with an application at the time of a bind to the port is authorized to access the port) then you should be using the SAF key word. This will also enable collection of much more detailed audit records using SMF.

3.4.1.2 Recommendation

The default values for the PORT statement leave all of the doors wide open and this is a bad thing.

Enable SERVAUTH protection in RACF (or ACF2 or TSS) and make use of it.

3.4.1.3 Applicable Rule

SERVAUTH must be enabled – STIG ITCPR052.

The SAF keyword should be used on all individual port definitions.

As an absolute minimum you should have a EZB.PORTACCESS.* profile defined in the SERVAUTH class. This allows the use of SMF to record audit data for listening

activity of all protected ports as well as enabling the controlled use of them – STIG AAMV0380.

The RACF profiles should all have UACC(NONE) – STIG ITCP0050.

3.4.2 PORTRANGE

3.4.2.1 Description

We already mentioned PORTRANGE in the previous topic (3.4.1) so this is a continuation…

PORTRANGE is used to define a group of ports that are to be treated with similar characteristics. It can help reduce the complexity of your PROFILE.TCPIP data. However, you should be aware of a couple of things:

1. There is no mechanism provided to ensure that you don't have overlapping port ranges defined. If a PORTRANGE overlaps with an individually defined PORT then it takes priority taking you into the realms of a possible Denial Of Service caused by the port suddenly not allowing multiple users because…
2. Ports defined by the PORTRANGE statement can NOT be shared by multiple users.

The values which can be specified (for both start/end port number and start/quantity of ports) range from 1 to 65535. Any definitions over 65535 will be ignored and the port(s) will not be controlled.

As with the PORT statement, I'd recommend using SAF and the format is the same if you're specifying it on either PORT or PORTRANGE (3.4.1.1).

One of the first tasks of any audit of the internetworking environment at your organization should be to establish how all of your ports are defined and secured.

3.4.2.2 Recommendation

The default values for this statement also leave wide open doors to your mainframe!

Enable SERVAUTH protection in RACF (or ACF2 or TSS) and make use of it.

3.4.2.3 Applicable Rule

SERVAUTH must be enabled – STIG ITCPR052.

The SAF keyword should be used on all port definitions.

As an absolute minimum you should have a EZB.PORTACCESS.* profile defined in the SERVAUTH class. This allows the use of SMF to record audit data for listening activity of all protected port ranges as well as enabling the controlled use of them – STIG AAMV0380.

The profiles should all have UACC(NONE) – STIG ITCP0050.

3.4.3 UNRSV

3.4.3.1 Description

You can also control which applications are allowed to access ports that have not been reserved by a PORT or PORTRANGE statement i.e. they are unreserved. You do this by specifying the PORT statement with the UNRSV keyword.

PORT UNRSV statements control access to any unreserved ports specified on explicit binds. Meaning that if the job name (when the work originates on z/OS – or the ACEE name that is created when you start work – often your RACF Userid) trying to make the connection doesn't match the one defined in the UNRSV

parameter, the service WILL be allowed to access the application's specified unreserved port. This represents best practice behavior!

Again, the SERVAUTH class EZB.PORTACCESS.sysname.tcpname.resname profiles are your friends in making sure you keep control and that there's less unmonitored activity in your system.

3.4.3.2 Recommendation

Starting to use PORT UNRSV access control can have unexpected consequences particularly on **unexpected** external users of your systems. You may be surprised at just how many unexpected users are making their way onto your system without being stopped.

Enable SERVAUTH protection in RACF (or ACF2 or TSS) and make use of it.

3.4.3.3 Applicable Rule

SERVAUTH must be enabled – STIG ITCPR052.

The SAF keyword should be used on all individual port definitions.

As an absolute minimum you should have a EZB.PORTACCESS.* profile defined in the SERVAUTH class. This allows the use of SMF to record audit data for listening activity of all protected ports as well as enabling the controlled use of them – STIG AAMV0380.

The profiles should all have UACC(NONE) – STIG ITCP0050.

3.4.4 RESTRICTLOWPORTS

3.4.4.1 Description

The RESTRICTLOWPORTS parameter (on both the UDPCONFIG and TCPCONFIG statements) allows you to control who can access ports which haven't been specifically reserved. It differs from UNRSV in that it ONLY deals with ports 1 through to 1023. Use UNRSV to secure any ports above 1023.

These are the "well known" ports that hackers will typically try first.

Access will be granted to users of those ports who satisfy either of the following conditions:

- the application is APF-authorized
- the application has OMVS superuser [UID(0)] authority

3.4.4.2 Recommendation

The very nature of TCP/IP services means that they are "platform agnostic". In other words, if you learn how to use a TCP/IP service (e.g. FTP) on a single platform (like Windows) then you can use the service anywhere that you find it with little or no alteration to parameters.

It is vital that you restrict who can enter your systems and RESTRICTLOWPORTS represents a good, quick way to do so.

3.4.4.3 Applicable Rule

RESTRICTLOWPORTS must be enabled – STIG ITCP0030.

3.4.5 NETACCESS

3.4.5.1 Description

NETACCESS allows for the "one-to-one mapping between a network, subnetwork or host and a Security Access Facility (SAF) resource name".

Huh?

What they mean is that because of the rather complicated way that the control of network resources by RACF et al has been (somewhat clumsily) implemented, it's necessary to have TCPIP hold a map of what the actual resource names are rather than what's specified on the parameter controls.

Those of you who have been around for a while will remember the old "discrete profiles" we used to use for datasets in RACF (or ACF2 or TSS) when we knew where they were going to be from day to day. This implementation reminds me a little of that, particularly the part where if TCPIP doesn't know about a RACF profile for a particular resource then it doesn't even bother issuing the SAF (System Authorization Facility) check!

As with all of the SAF parameters, the profiles to control use of the map are stored in the SERVAUTH class. This time the format is: EZB.NETACCESS.sysname.tcpname.saf_resname

3.4.5.2 Recommendation

There is no default value for this parameter. Even specifying it at all is optional!

It is vital that you restrict who can enter your systems and NETACCESS represents a good, quick way to do so.

3.4.5.3 Applicable Rule

SERVAUTH must be enabled – STIG ITCPR052.

The SAF keyword should be used on all individual port definitions.

As an absolute minimum you should have a EZB.NETACCESS.* profile defined in the SERVAUTH class. This allows the use of SMF to record audit data for the use of all defined unreserved ports as well as enabling the controlled use of them.

The profiles should all have UACC(NONE) – STIG ITCP0050.

3.4.6 GLOBALCONFIG EXPLICITBINDPORTRANGE

3.4.6.1 Description

All of the GLOBALCONFIG parameters set various "default" processing values for a TCP/IP region. EXPLICITBINDPORTRANGE specifically deals with how multiple TCP/IP stacks work together in a port pooling environment like VIPA.

As is so often the case, there are a few rules governing this:

- All TCP/IP stacks in the sysplex that engage in EXPLICITBINDPORTRANGE processing must have exactly the same port range specified.
  - o IBM's recommendation to ensure this happens is for you to specify the GLOBALCONFIG EXPLICITBINDPORTRANGE statement in a file that is specified in an INCLUDE statement in the TCP profiles data set of all the participating stacks.
- The port range defined on the EXPLICITBINDPORTRANGE parameter must not overlap any existing port reservations of any TCP/IP stacks in the sysplex. Any reserved ports that are within the EXPLICITBINDPORTRANGE range are

not included in the EXPLICITBINDPORTRANGE port pool. This can sometimes leave you with a surprisingly smaller port pool, potentially leading to a Denial Of Service.

- The EXPLICITBINDPORTRANGE port range must be large enough to allow for all applications in the entire sysplex that might issue explicit bind() calls for the IPv4 INADDR_ANY address, or for the IPv6 unspecified address (in6addr_any), and port 0.
  - o If new TCP/IP stacks or systems are introduced into the sysplex, the extent of the port range defined by EXPLICITBINDPORTRANGE should be re-evaluated.
- Changing the range specified for EXPLICITBINDPORTRANGE affects every TCP/IP stack in the sysplex that takes part in the pooling.
- Ports in the EXPLICITBINDPORTRANGE range are usually assigned to a stack in blocks of 64 ports.
  - o When increasing the number of available ports you should use multiples of 64 (multiplied by the number of stacks in the port pooling group).
- If the port range defined by the EXPLICITBINDPORTRANGE parameter is too large, there are fewer ports available for local ephemeral port allocation. Be very careful when making changes.

3.4.6.2 Recommendation

There is no existing standard for Best Practice on what the range for explicit binds should be. The decisions are made entirely by your organization. Whatever your standards are, that's what you need to audit.

However, there is one factor which will affect ALL users of ALL ports: make sure that the number you have defined in your TCPIP.PROFILE is the same as the number that are available for use. There is nothing to prevent you defining overlapping port ranges or from defining so many explicit bind ports that there's nothing left for ephemeral binds.

3.4.6.3 Applicable Rule

Refer to your own, internal standards to establish what the value SHOULD be and confirm that's what you're using.

You can establish if the values are set appropriately in terms of number but not in terms of security by issuing the UNIX command: netstat -s

3.4.7 UDPCONFIG EPHEMERALPORTS

3.4.7.1 Description

Ephemeral ports are port numbers that TCP/IP temporarily assigns to a socket. One reason that you might want to exploit them would be to facilitate port controls on firewalls.

Geek-speak warning!

They are deployed when:

- An application explicitly binds to port 0.
- An implicit bind() call is processed as the application sends data, and the port number is not known.

The default ephemeral port numbers for both TCP and UDP are in the range 1024 to 65535. However, ephemeral ports are about the bottom of the priority list. So if

anything else declares that it's using a port then the ephemeral port range loses out.

The allocation of ephemeral ports is managed by the INADDRANYPORT and INADDRANYCOUNT parameters in the active BPXPRMxx member used by your organization to control the shape of z/OS UNIX Systems Services.

By comparing the values for "Configured Ephemeral Ports" with that of "Ephemeral Ports In Use" and for "Ephemeral Ports Max Usage" with that of "Ephemeral Ports Exhausted" you can check that there's no accidental ranging issues that take ports from the pool available to the ephemeral ports.

3.4.7.2 Recommendation

Just like the TCP ephemeral ports, there are no existing standards for Best Practice on what the range for ephemeral UDP binds should be. The decisions are made entirely by your organization. Whatever your standards are, that's what you need to audit.

And don't forget the ever present danger of incorrect definitions leading to less available ports than you expected.

3.4.7.3 Applicable Rule

Refer to your own, internal standards to establish what the value SHOULD be and confirm that's what you're using.

You can establish if the values are set appropriately in terms of number but not in terms of security by issuing the UNIX command: netstat -s

3.4.8 TCPCONFIG EPHEMERALPORTS

3.4.8.1 Description

The description of TCP ephemeral ports is pretty much identical to the UDP ones described above (3.4.7) except they're used by different processes/services. Both types are assigned from the port range 1024 to 65535 and you typically pick a subset of that range which provides sufficient capability for your organization's needs. Again, use the netstat –s command to establish those numbers (3.4.7.1).

3.4.8.2 Recommendation

I run the risk of sounding quite repetitive here but… Just like the UDP ephemeral ports, there are no existing standards for Best Practice on what the range for ephemeral TCP binds should be. The decisions are made entirely by your organization. Whatever your standards are, that's what you need to audit.

And don't forget the ever present danger of incorrect definitions leading to less available ports than you expected.

3.4.8.3 Applicable Rule

Refer to your own, internal standards to establish what the value SHOULD be and confirm that's what you're using.

You can establish if the values are set appropriately in terms of number but not in terms of security by issuing the UNIX command: netstat -s.

3.4.9 DELETE

3.4.9.1 Description

The inclusion of the DELETE statement is called for if you're changing the way you handle ports, port ranges and a number of other elements in the PROFILE.TCPIP data. And I'm deliberately saying "data" rather than dataset or member. That's because the comment also refers to data that's incorporated into the PROFILE.TCPIP data by way of INCLUDE statement(s) within the active profile.

The DELETE statement works on a number of elements in the PROFILE.TCPIP data. At time of publishing that list was:

- ATMARPSV (deprecated at V2R2)
- ATMLIS (deprecated at V2R2)
- ATMPVC (deprecated at V2R2)
- DEVice
- LINK
- **PORT**

The last one (PORT) is the reason for the inclusion of the DELETE statement in this book. If you find it anywhere in a production TCP/IP stack it's an audit finding, at least as far as the US Government is concerned. And I agree! In fact, I'd go a step further than this… unless you can categorically demonstrate that you have full separation of your environments at the hardware level AND that you don't run any production work on non-production systems, you should NEVER have DELETE specified in any PROFILE.TCPIP data.

I have never worked on a single mainframe that had a total separation of test/development and production environments. Of course you might be one of the sites I haven't visited but it is unlikely that things are as clearly defined as you think.

3.4.9.2 Recommendation

There is a DISA STIG (ITCP0030) which specifically refers to not allowing DELETE statements (either active OR commented out!) within PROFILE.TCPIP in any production environment.

I say: You need extreme justification to leave any DELETE statements in any PROFILE.TCPIP data given the way most of us run our mainframe systems. If you can see production data from your test or development systems then you can't differentiate those systems as non-production.

Sorry about this gentle reader but it's not an easy thing to establish if you do have complete separation of your test and production environments. You need to start from the IODF definitions used to run your mainframe(s) and work outwards from there. Fortunately, I have already described the process in z/Auditing Essentials Volume 1 ☺

3.4.9.3 Applicable Rule

DELETE should only be used during managed change control events specifically related to altering the use of ports in TCP/IP. Outside of that specific use, it should NEVER be used in a live TCP/IP stack.

3.5 DATA

Also known as TCPIP.DATA this is where the TCP/IP stack determines what host name it should be using (the value that is returned on gethostname socket function calls processed by this stack) amongst other things. Most of the parameters simply control the shape of your TCP/IP system. They wouldn't immediately leap out at you as security related options. However, changing the values can change the shape of the environment so radically that any security you DO have defined will not be used!

The rules governing where to find the data are just as complicated as those for TCPIP.PROFILE (3.4). In fact, the only significant difference is that this time the potential JCL statement will be the //DATA DD card.

The general rule for all of the external holding areas for TCPIP definitions data is to use the top level holding point. That is, specify all TCP/IP parameters in z/OS sequential datasets (more specifically, parameter library members) so that the values can't be overridden by a higher level AND that you can adequately control access to the data using your ESM (ACF2, RACF or TSS).

3.5.1 Description

Here's a handy summary of the parameters with a description of what can go wrong if they change without your knowledge:

Parameter	Used for:	What can go wrong:
ALWAYSWTO	Issues WTO messages for servers.	Your TCP/IP environment will effectively become invisible to z/OS!
DATASETPREFIX	Sets high-level qualifier for dynamic allocation of data sets.	Data will be accessed from somewhere other than your carefully defined security environment
DOMAINORIGIN or DOMAIN	Specifies the domain origin that is appended to the host name to form the fully qualified domain name of a host.	You will look like a completely different host which may not even be known to your firewalls!
HOSTNAME	Specifies the TCP host name of the z/OS communication server.	Just like the parameter above, you will not look how expected, either to external systems OR your own ESM definitions!
LOADDBCSTABLES	Indicates to FTP which DBCS translation tables can be loaded.	Users expecting to communicate with your systems in, for example, Japanese may be presented with apparently random symbols!
LOOKUP	Specifies the order in which the DNS and the local host file are to be used for name resolution.	A hacker could use this to reroute your traffic to somewhere more convenient for them to deal with rather than trying to operate inside your secured environment.

MESSAGECASE	Specifies case translation for the FTP server and osnmpd.	Any automation that's looking for messages in a specific case may not see the activity and so won't act on the correct triggers.
NOCACHE	Specifies that the resolver should not use the system-wide cache for any queries associated with applications that use this TCPIP.DATA file. The cache is bypassed for any queries from a DNS server and results obtained from a DNS server are not updated in the cache.	This is a classic way for a hacker to hide their tracks. However, it is also a performance enhancing option.
NOCACHEREORDER	Specifies that the resolver should not reorder the list of IP addresses that are provided in response to a resolution request for the associated host name.	Not reordering the cache can be seen as a performance option but like NOCACHE, it's also a classic hacker tool used to hide activity.
NSINTERADDR or NAMESERVER	Defines the IP address of a name server. The IP address can be either IPv4 or IPv6.	Another of those pesky hacker tools that also represent a performance option. The parameter allows applications to go straight to a specific name server to resolve the binds. That's good for speed. It's also a good point to override the way the system acts and could be used to instigate a man in the middle attack with your system hosting the attack! - STIG ITCP0025
NSPORTADDR	Specifies the name server port number.	And yet another performance option that could be hijacked to route traffic somewhere other than the expected destination.
OPTIONS	Specifies if resolver debug messages should be issued and the number of periods (.) that need to be contained in a domain name for it to be considered a fully qualified domain name.	Debug messages can't be dangerous right? Wrong! Debug messages are a great way to find out what the system you find yourself in the middle of expects you to do. Also another way to change the shape of the messages that you'll see AND fool the ESM into thinking it doesn't need to care about you.

RESOLVERTIMEOUT	Specifies how long the resolver waits for a response while trying to communicate with the name server.	Reduce this value to provoke a Denial Of Service!
RESOLVERUDPRETRIES	Specifies how many times the resolver tries to connect to the name server when using UDP datagrams.	Increase this value to provoke a Denial Of Service!
RESOLVEVIA	Specifies the protocol used by the resolver to communicate with the name server. If you use the autonomic quiescing of unresponsive name servers function, you should use UDP as your protocol to communicate with the name server.	If you're in a system that uses autonomic quiescing or unresponsive name servers and you change the value to TCP you have yourself an unexpected situation. The request would be failed but because UDP is the default and will be switched back to automatically, all you should see is a brief interruption to service.
SEARCH	Specifies the list of domain names that are appended, in the order listed, to the host name to form the fully qualified domain name of a host.	Change this value and the shape of your environment has changed – potentially radically".
SOCKDEBUG	Turns on tracing of TCP/IP socket library calls.	See my previous comment about debug information not being dangerous! This would let me see what libraries are checked for socket calls and thus map the programming environment in which I find myself.
SOCKNOTESTSTOR	Stops checking of TCP/IP socket calls for storage access errors on the parameters to the call. This statement: • improves response time. • is in effect unless SOCKTESTOR is specified. • works for all TCP/IP C sockets across the system the way sock_do_test_stor() works for a specific socket application.	Another performance option that can be used to subvert the way your system works.

SOCKTESTSTOR	Checks TCP/IP socket calls for storage access errors by enabling additional parameters to be specified on the call. This statement works for all TCP/IP C sockets across the system the way sock_ do_test_stor() works for a specific socket application.	Another performance option that can be used to subvert the way your system works.
SORTLIST	Specifies the ordered list of network numbers (subnets or networks) for the resolver to prefer if it receives multiple addresses as the result of a name query.	Change the value the system prefers to use and change where that traffic will end up. A very useful tool for a hacker.
TCPIPJOBNAME or TCPIPUSERID	Specifies the member name of the cataloged procedure used to start the TCPIP address space.	If I can pick up my own cataloged procedure to start a TCP/IP instance, I can control more or less everything about that stack.
TRACE RESOLVER	Traces all queries to and responses from the name server.	Trace data helps me to map out an unfamiliar system.
TRACE SOCKET	Traces TCP/IP C socket library calls.	Again, trace data helps me to map out an unfamiliar system.
; or #	Uses either character to indicate a comment.	Changing the value to make the TCP/IP startup interpret lines intended as comments to be read as actual TCPIP.DATA content. For example, commented out DELETE statements could change the shape of your TCP/IP environment. That's why you aren't even allowed commented out DELETE statements as far as DISA is concerned.

There really aren't any defined Best Practices for TCPIP.DATA but do try to be careful when implementing performance options. They often don't just affect performance!

3.5.2 Recommendation

It's critical that you control who can change these values and monitor constantly for any changes. DISA STIGs ITCP0020 and ITCP0040 discusses the potential to downgrade security in your TCP/IP stack should these statements be incorrectly specified.

3.5.3 Applicable Rule

Refer to your own, internal standards to establish what the parameter values SHOULD be and confirm that's what you're using.

3.6 RESOLVER

Back in the early days of the Internet there were so few sites available that we used to be able to remember the IP address numbers of that site and go straight there. Things are a little more complex these days! We carbon-based life-forms stand no chance of remembering everything we need to work in a seemingly endlessly connected world. We need some help and the RESOLVER provides that service.

The RESOLVER does a number of things for people or processes trying to use TCP/IP:

- Access name servers to provide name-to-address or address-to-name resolution
- Allocate and read the TCPIP.DATA file
- Establish TCP/IP stack affinity for certain socket APIs
- Provide protocol and services information

It's a critical part of the system and without which the vast majority of TCP/IP services on z/OS would be unusable. As such you can start it using just the defaults. As always, sometimes this is good, sometimes not so good. An untailored system only allows for:

- System-wide caching, which uses the default maximum cache size, the default maximum time-to-live (TTL) value, and the default setting for the cache reordering function.
- Network operator notification of unresponsive Domain Name System (DNS) servers, using the default threshold setting.

Both functions are too important to leave to mere mortals to remember!

3.6.1 Description

The RESOLVER is integrally linked to z/OS UNIX Systems Services. It's started automatically when OMVS initializes and (unless you customize it) that's the only option to automate. You customize resolver functions by using resolver configuration statements in an optional resolver setup file (found by following the trail from the RESOLVER started task JCL to see what's in the //SETUP DD statement.

Parameters that you should pay attention to (i.e. make sure they aren't changing unexpectedly):

Parameter	Used for:	What can go wrong:
DEFAULTIPNODES	Identifies the local host file.	If I change this value then I control what your TCP/IP stack does.
DEFAULTTCPIPDATA	Identifies the TCPIP.DATA to use. If you don't specify the DEFAULTTCPIPDATA statement, the default file is TCPIP.TCPIP.DATA	Allowing defaults to define your system means that I know what they are and can exploit that. Tie down where TCPIP.DATA lives and who can change it and I will find it more difficult to hack your systems ☺

GLOBALIPNODES	Identifies a local host file that contains hard-coded IP addresses and host names that can be used globally.	Changes to this value will significantly alter the way your system behaves.

GLOBALTCPIPDATA in your TCPIP.PROFILE data identifies the file that is first searched by the resolver for resolver configuration information (before it moves on to other locations). Parameters that you specify in the file that is identified by the GLOBALTCPIPDATA statement become the global settings for the entire z/OS image and for all TCP/IP stacks. A single change could alter the way that your entire TCP/IP environment works.

Whatever your settings are the important thing is to ensure that you know about any changes. By altering the content of any of these parameters a hacker could fundamentally change the way your TCP/IP environment behaves. Even an accidental change by someone with authority to do so can cause a Denial Of Service if it goes badly.

3.6.2 Recommendation

Don't use defaults to provide the environment that you require. Always fully specify the values that your organization has decided on even if that means supplying the default value for a parameter.

It's vital that you control who can change these values and monitor constantly for any changes.

3.6.3 Applicable Rule

Refer to your own, internal standards to establish what the parameter values SHOULD be and confirm that's what you're using.

3.7 PAGENT

There are some very clever things that can be done on a mainframe with regards to encryption. Not to say that clever things can't be done on other platforms but that's not what we're here to talk about.

One very clever thing that we can do is called Application Transparent - Transport Layer Security or AT-TLS (as defined by RFC 2246). This allows you to take an application (any application which requires mainframe resources) and encrypt all of the traffic back and forward. You can do this without having to make a single change to the application in question.

AT-TLS answers many of the concerns we have about intercepted traffic. The question of who can see what on your network becomes less urgent when all traffic that anyone sees will be "scrambled".

Why am I talking about AT-TLS in a section about PAGENT?

Most organizations install Policy Agent (PAGENT) only when they are doing something that needs it, like deploying AT-TLS (although from z/OS 1.10 you can deploy AT-TLS purely within RACF, etc. but if you do, you can't use all of the advanced functions of PAGENT). There's often no planned implementation as the service will run with zero input from your local Systems Programmers.

IBM's z/OS Policy Agent is software which enables not just AT-TLS but all of the following types of internetworking protocol policy management within your environment:

Policy Type	Text file format	LDAP format	Comments
QoS	Yes	Yes	Quality of Service policy
IDS	Yes	Yes	Intrusion Detection Service policy
AT-TLS	Yes	No	Application Transparent – Transport Layer Security policy
IPSec	Yes	No	IPSEC policy
Policy-based routing	Yes	No	Routing policy

Local policies are defined in Policy Agent configuration files, in the LDAP server, or both (if supported).

Remote policies are defined in Policy Agent configuration files on the policy server.

Policies from configuration files and the LDAP server are combined into a single list. This requires unique policy object names for each type (QoS, IDS, IPSec, Routing, and AT-TLS).

On a policy client, policies for a given type are retrieved either locally or remotely, but not both.

PAGENT requires access to appropriate CERTAUTH profiles in your RACF (or ACF2 or TSS) database to provide it with the certificates it needs to work correctly rather than just start up. Another thing it needs is a TTLS policy. This describes the AT-TLS rules you want active and can be found in the Policy Agent configuration file (default location: /etc/pagent.ttls.conf). One more bit to look out for… AT-TLS will always write messages to the z/OS UNIX syslog not the z/OS environment (or even SMF). So you must have syslogd set up appropriately for your environment. See STIGs ISLG0010, ISLG0020 and ISLG0030 for further information.

This “Brave New World” that we find ourselves operating in means we now have to take the contents of syslogd just as seriously as we do with SMF data. To quote one of my co-authors on this book who was summarizing the contents of the syslogd related STIGs:

“Record it in detail, store it securely for as long as SMF, make sure you don’t lose any records during peaks and don’t let anyone fiddle with the controls!”

3.7.1 Is it a Server or a Client?

Both!

PAGENT is a client/server protocol suite which manages policies for various other internetworking protocols. I’ve used the AT-TLS example throughout so far so I’ll stick with it for this super simplified version of events…

You ask for a connection to a mainframe application and the request is routed through your local AT-TLS policy service. That’s the PAGENT Client. It searches through its definitions to figure out what to do with the request. When it identifies that you’re asking for a mainframe session, the policy routes your request to the AT-TLS policy service on the mainframe. That’s the PAGENT Server. The 2 policies then negotiate the connection rules that will be used for your mainframe application session.

In the world of internetworking protocols nothing is ever simple!

So there are multiple types of configuration files. The role of the PAGENT instance determines which files are used and for what. PAGENT acting as a policy server provides centralized policy services for a set of policy clients.

It can also act as the policy client retrieving remote policies from the policy server. The policy client installs these remote policies in the corresponding TCP/IP image. In either case (policy client or policy server), local policies can also be stored in local configuration files.

You should monitor closely for any changes to all of these elements. Any alteration can stop AT-TLS working as expected potentially "unscrambling" all of your network traffic (making it "easy" for a hacker to pick out user/password combinations etc.).

AT-TLS support is enabled by specifying the TTLS parameter on the TCPCONFIG statement in your PROFILE.TCPIP data. AT-TLS functionality is managed by the Policy Agent, which must be active to be able to enforce the AT-TLS policy.

Policy Agent must wait for TCP/IP to be active, so the AUTOSTART statement in PROFILE.TCPIP might be a good place to trigger startup of this service if you don't have it handled by other automation. Just to show you where to start from, this is the sort of thing that you're looking for in PROFILE.TCPIP:

```
TCPCONFIG TTLS  ; Required for AT-TLS
AUTOLOG
    PAGENT      ; POLICY AGENT, required for AT-TLS
ENDAUTOLOG
```

If you see something like that your organization is using PAGENT and the dire warnings about monitoring everything become relevant to you!

There are more similarities than differences in the way you define PAGENT whether as Server or Client.

3.7.1.1 Description

The first step in getting PAGENT up and running on z/OS (and therefore the place to start looking for what to examine during an audit) is to define a RACF (and/or ACF2 and/or TSS) Keyring. This will contain the elements needed to assign digital certificates to the sessions managed by PAGENT.

Next you must define the protocol to TCPIP.DATA e.g. by specifying TCPCONFIG TTLS.

You should also include PAGENT in the AUTOLOG definitions for the automated start of Unix System Services elements. This is normally found in SYS1.PARMLIB (or equivalent).

Then there are some additional RACF (et al.) requirements:

Profile in SERVAUTH	Used for:
EZB.INITSTACK.sysname.tcpprocname	Controls which users have access to the TCP/IP stack before PAGENT is active. Give READ access to all users who do not require PAGENT policies to access the TCP/IP stack; for example, PAGENT, NETVIEW, DB2 etc.
EZB.PAGENT.sysname.tcpprocname.*	Controls which users can start, stop, and refresh PAGENT. Give READ access to users who are allowed to run the TSO/Unix commands Pagent and/or pasearch

BPX.DAEMON	The Userid associated with the startup of the PAGENT Started Task must have READ access to this resource.

PAGENT is initiated by a z/OS Started Task. The JCL is typically held in a member called PAGENT in one of the JES2 procedure libraries (often but not always SYS1.PROCLIB).

The location and name of the main configuration file is set by the environment variable PAGENT_CONFIG_FILE. It contains the following information relevant to PAGENT:

Parameter	**Used for:**
AutoMonitorApps	Allows you to configure the Policy Agent to monitor and automatically start or restart a set of related applications. The following set of applications can be monitored: • Defense Manager daemon (DMD) • IKE daemon (IKED) • Network Security Server daemon (NSSD) • Syslog daemon (SYSLOGD) • Traffic Regulation Manager daemon (TRMD)
AutoMonitorParms	Allows you to configure global parameters that control how the Policy Agent monitors and starts or restarts the applications defined in AutoMonitorApps.
ClientConnection	The Policy Agent acting as a policy server uses this statement to specify the listening port. The Policy Agent acting as a policy client uses this connection to retrieve remote policies.
Codepage	Specifies the EBCDIC code page to use for reading all configuration files and policy definition files. The default is IBM®-1047. All statements read from the files are converted to the IBM-1047 code page from the specified code page. This can cause translation problems if the data was not created using IBM-1047.
CommonIPSecConfig	Specifies the path of a local IPSec policy file that contains common IPSec policy statements. To define a common set of policies for multiple stacks, the IpSecConfig statement can specify the same file as the CommonIpSecConfig statement.
CommonTTLSConfig	Specifies the path of a local AT-TLS policy file that contains common AT-TLS policy statements. To define a common set of policies for multiple stacks, the TTLSConfig statement can specify the same file as the CommonTTLSConfig statement.

DynamicConfigPolicyLoad	The Policy Agent acting as a policy server uses the DynamicConfigPolicyLoad statement to obtain the file names of the configuration files to be retrieved by policy clients. To retrieve remote policies with the policy client, you must define EZB.PAGENT.sysname.image.ptype in the SERVAUTH class and grant READ access to the policy client's user ID; the user ID is defined on the Userid parameter on the PolicyServer statement. The ClientName parameter on the PolicyServer statement is used as the image name.
LogLevel	Specifies the level of tracing for the Policy Agent. The trace records can help debug errors in policy definition. The only parameter to LogLevel is an integer that specifies the level of logging and tracing. The following levels are supported: 1 - SYSERR - System error messages 2 - OBJERR - Object error messages 4 - PROTERR - Protocol error messages 8 - WARNING - Warning messages 16 - EVENT - Event messages 32 - ACTION - Action messages 64 - INFO - Informational messages 128 - ACNTING - Accounting messages 256 - TRACE - Trace messages
ServerConnection	The Policy Agent acting as a policy client uses this statement to connect to the Policy Agent acting as a policy server. This statement includes security information and the location of the policy server and the policy client uses this connection to retrieve remote policies.
ServicesConnection	The Policy Agent uses this statement to specify the listening port, listening TCP/IP image, and security level for connections to the Policy Agent. Applications that use this connection are known as services requestors. One such services requestor is the Configuration Assistant for z/OS Communications Server, which is uses this connection to retrieve import policies or TCP/IP profile information.

TcpImage and PEPInstance	Specifies a TCP/IP image and its associated image configuration file to be installed to that image. These statements are synonyms and you can use either of them. If no TcpImage statement is specified, any policy definitions are installed to the default TCP/IP image (resolver supplied TCPIPuserid statement or TCPIPjobname statement). The parameters FLUSH or NOFLUSH can be used to force deletion of some policy types from the stack at startup and certain other events. The parameters PURGE or NOPURGE can be used to delete some policy types from the stack during normal shutdown (for example, KILL or STOP).

The environment variables for the TCP/IP stack are usually specified in a member (for example, ENVVARS) of the TCP/IP parameters library (for example, TCPIP.PARMLIB). And the PAGENT JCL has ddname STDENV that points to the member with the environment variables definitions.

And finally, PAGENT sends all of its communicative messages to syslogd. So that has to be set up first otherwise you'll have nothing!

3.7.1.2 Recommendation

TCPCONFIG TTLS should be set in your TCPIP.PROFILE data if you are using AT-TLS.

syslogd must be active.

PAGENT must be included in AUTOLOG processing.

And you must control (and monitor) who can make changes to the contents of any and all of the z/OS and USS files mentioned in the previous section.

Note: You can't use dynamic monitoring for file updates by specifying the -i startup option. It is not supported for files read on behalf of policy clients.

3.7.1.3 Applicable Rule

Refer to your own, internal standards to establish what the parameter values SHOULD be and confirm that's what you're using.

3.7.2 IPCONFIG IPSECURITY

3.7.2.1 Description

AT-TLS without IPSec provides security between endpoints (in "Transport Mode").

IPSec adds security between points in "Tunnel Mode". This provides the added benefit of extra security to the final destination (in a subnet when the packet has be unencrypted by some intermediate process).

Contained within the PROFILE.TCPIP data, this parameter works against the IPv4 functions within TCP/IP on z/OS. It specifically enables IPv4 IP filtering and IPv4 IPSec tunnel support. And it's normally selected when you are deploying AT-TLS as it enables communication with PAGENT.

Tech speak alert!

The location of the IPSec policy is managed by PAGENT and can be found in the

CommonIPSecConfig parameter value.

To define a common set of policies for multiple stacks, the IpSecConfig statement can specify the same file as the CommonIpSecConfig statement. Stack-specific IPSec policies are defined in an IPSec stack-specific policy file identified by an IpSecConfig statement.

The file (z/OS sequential dataset or USS file) pointed to contains some of the following parameters used to affect IPSec protocol requests:

Parameter	Used for:
IPSecConfig	Defines the path to a local IPSec policy file that contains stack-specific IPSec policy statements. To define a common set of policies for multiple stacks specify the IPSecConfig statement without a path name in each image configuration file.
LocalSecurityEndpoint	Encapsulates a local security endpoint's IP address or host name and identity information. You **MUST NOT** use the same local identity on more than one TCP/IP stack or z/OS system image. IKED assumes that an identity is not used by any other stack, and this assumption can lead to a disruption of IPsec service for other stacks if identities are shared between stacks. And that's a bad thing.
RemoteSecurityEndpoint	Encapsulates remote security endpoint IP addresses or hostnames and identity information. Also defines the identity requirements for remote security endpoints with which negotiations for dynamic VPN tunnels are allowed. The statement can also list one or more Certificate Authorities to be used with the allowed security endpoints.

IPSECURITY can only be enabled or disabled during TCP/IP start up. So you don't need to worry about dynamic changes but you should confirm that nothing has changed since the last time the instance of TCP/IP was started.

3.7.2.2 Recommendation

If you are running something that requires IPv4 filtering or IPSec tunneling then IPSECURITY should be enabled.

And this isn't a one sided issue, you must also have considered the implications of where your mainframe is tunnelling to. Just because the pipe is secure doesn't mean it's going to the right location or carrying the appropriate payload. There are a number of IPSEC/VPN STIGs about restricting the reach of tunnels within the network and at the borders.

It's vital that you control who can change these values and monitor constantly for any changes.

3.7.2.3 Applicable Rule

If you have deployed or are planning to deploy AT-TLS (or any other exploiting protocols) then you should have IPSECURITY specified on the IPCONFIG statement.

3.7.3 TCPCONFIG TTLS

3.7.3.1 Description

Again, this is a parameter that you are only likely to take much notice of if you've deployed AT-TLS.

The default for this parameter is NOTTLS meaning that AT-TLS is not enabled. Even if you've set all the other parameters that are needed for AT-TLS, if a single one of them is wrong, the whole service will be unavailable.

Setting the value to TTLS adds a few more requirements to the mix. You MUST have SERVAUTH class active in RACF (or ACF2 or TSS) AND you must both define and grant access to the appropriate EZB.INITSTACK resource profile. If the user initiating the connection does not have authority to the profile, AT-TLS use will not be allowed for that user.

3.7.3.2 Recommendation

There are very few reasons not to run AT-TLS. Even the old performance arguments don't really hold true anymore due to the time and effort expended by IBM to give us super-fast crypto capability.

3.7.3.3 Applicable Rule

If you have deployed or are planning to deploy AT-TLS (or any other exploiting software) then you should have TTLS specified on the TCPCONFIG statement in PROFILE.TCPIP.

3.8 Intrusion Detection

In an increasingly interconnected world it's important not just to know when your system is breached but also that you look for patterns of behavior. This can lead to the earlier identification of unusual behavior which requires further examination.

Intrusion Detection Services sit at the entry points to your system and looks at 2 specific things: intrusion and extrusion.

Intrusion detection is about looking for, and stopping, players getting into your systems in order to steal your data, disrupt your business or maybe cause a "Denial Of Service".

Extrusion detection is more about spotting and stopping attempts to use your system to launch an attack from.

3.8.1 Description

You use IDS policies to specify event conditions and the actions to take for those events. All IDS policies support logging events to a specified message priority level in syslogd, on the z/OS system console, or both.

Most IDS policies support the following functions:

- Discarding packets when a specified limit is reached
- Writing statistical records to the INFO message level of syslogd at a specified time interval, and the option to write these records only when exceptional events have occurred
- Tracing all or part of the triggering packet to an IDS-specific CTRACE facility, SYSTCPIS

IDS will assign a correlator value to each event and all messages written to the z/OS system console and syslogd use this correlator. Records written to the IDS trace

use this correlator too. A single detected event can involve multiple packets; the correlator value identifies which messages and packets are related to each other.

I like to use these facilities to make sure that internetworking messages are also written to the z/OS log. That way I can have my automation software keep an eye on what's happening too.

Configuring IDS to run the way that you want referencing the policies that you have defined is a complex process. IBM provides a wizard type function (IBM Configuration Assistant for z/OS Communications Server) to simplify the process but you can still "kick it old skool" and hand code every parameter using the ISPF Editor if you want.

The Configuration Assistant is a z/OS Management Facility (z/OSMF) task. So you need to be running z/OS 1.11 or higher and have z/OSMF configured/available. Using the Configuration Assistant, there are two types of reusable objects:

- Traffic descriptors which define the IP traffic type (e.g. TCP or UDP).
- Requirement maps containing attack protection, scan protection, and traffic regulation.
    - o A single requirement map should contain a full set of IDS requirements that will govern the level of IDS for a TCP/IP stack.

And finally, not all IDS policy options are available in an LDAP configuration file.

3.8.2 Recommendation

There is an entire SRG covering IDPS; Intrusion Detection and Prevention Systems (IDPS) Security Requirements Guide with 60 STIGs in it. Additional STIGs regarding IDPS are located in the network infrastructure policy manual and overview docs. Which ones are appropriate will depend on your network implementation etc. It's less about mainframe security and more general network security concepts and practical implementations and outside the scope of this guide.

Basically IDS is there, use it! ☺

Tech speak alert!

The location of the IDS policy is managed by PAGENT and can be found in the CommonIDSConfig parameter value. The file (z/OS sequential dataset or USS file) pointed to contains some of the following parameters used to affect IPSec protocol requests:

Parameter	Used for:
IDSConfig	Provides an override to CommonIDSConfig. Changing the value specified in this parameter would result in a different files being used to start IDS. Results could be unpredictable!
IDSAction	Defines the action taken by the IDS rule. This statement is associated with an IDS rule with the same ActionType value.
IDSAttackCondition	Defines your requirements for attack detection, reporting, and prevention. There are several attack types. For each attack type, the single highest priority rule is used.

IDSExclusion	Let's you specify IP addresses, and optionally ports, that are to be excluded when monitoring for certain attacks. E.g. you can use an IDSExclusion statement to exclude a printer connection from being reset by TCP_QUEUE_ SIZE attack detection if the printer remains in a persist state for a period of time while out of paper.
IDSReportSet	Let's you specify what you want to see under specific attack conditions. A report set can include type of action, statistics interval, logging level, trace data, and trace record size. If a packet meets a policy rule's condition during its validity period, the reports specified in the policy rule's action, such as logging the packet, are produced.
IDSRule	Let's you enable intrusion detection services based on the ConditionType parameter for an IDSRule statement.
IDSScanEventCondition	Let's you define the conditions that scan processing monitors. These policies are searched by this ScanEvent condition type and a protocol condition of ICMP, ICMPv6, TCP, or UDP. For protocols TCP and UDP, the policy search also includes local destination port and bound IP address.
IDSScanExclusion	Let's you specify IP addresses and optionally, ports, that are to be excluded when monitoring for scans. E.g. responses from name servers might appear to be scans, unless the name servers are excluded using this statement.
IDSScanGlobalCondition	Let's you define your global scan detection and reporting requirements. The action defines the reporting and tracing actions to take when a scan event is detected.
IDSTRCondition	Let's you define your requirements for traffic regulation for TCP connections and UDP receive queues. TCP rules are mapped when a local application does a listen on a socket or when an inbound connection handshake completes. UDP rules are mapped when an inbound packet arrives at a local bound socket. UDP TR policy supersedes the TCPIP PROFILE setting of UDPQUEUELIMIT for covered ports.

There are no Best Practices for the implementation of Intrusion Detection Services due to the highly complex nature of internetworking. It is vital that you define your own requirements and audit to that standard.

3.8.3 Applicable Rule

Refer to your own, internal standards to establish what the parameter values SHOULD be and confirm that's what you're using.

3.9 Policy Based Routing

Policy Based Routing allows the TCP/IP stack to make routing decisions that take more than just the destination IP address into account. These additional elements can include job name, source port, destination port, protocol type (TCP or UDP), source IP address, NetAccess security zone, and multilevel secure environment security label.

E.g. you might want to prioritize high-bandwidth links for batch traffic, but for interactive traffic you don't mind using low-latency links. In this case you could define policies such that Telnet traffic is routed over the low-latency links, and FTP

traffic is routed over the high-bandwidth links.

You also might want to define a policy to ensure that traffic tagged with a security label and zone is routed to a secured network over an appropriate outbound interface. Or you might want to control the links that are used by Enterprise Extender traffic to keep that traffic from being impacted by other IP traffic loads.

Of course it wouldn't be internetworking if there weren't some provisos…

- Policy-based routing applies only to TCP and UDP traffic that originates at the TCP/IP stack.
  - o These types of traffic are always routed by the main route table even when policy-based routing is in use:
    - Traffic that uses protocols other than TCP and UDP
    - Traffic that the TCP/IP stack uses
    - Traffic for the Ping and Traceroute commands
- If your organization makes use of Common INET (CINET) to run multiple z/OS Communications Server TCP/IP stacks concurrently, CINET has no knowledge of the policy-based route tables that are used by those TCP/IP stacks. CINET only knows of the routes in the main route table of each TCP/IP stack.

The combinations and possibilities are virtually endless. And that's just one reason why there are no Best Practice standards for the implementation of Policy Based Routing. You need to understand what your site intends to use and audit to that standard.

3.9.1 Description

Policy Based Routing does not represent cutting edge internetworking. Most perceived advantages of using it can be dismissed in the vast majority of mainframe installations. That's because it doesn't recognize that there is more than one TCP/IP stack available.

That said, I have worked with organizations that use a single mainframe LPAR to control all of their network traffic. Under this circumstance it can be helpful to separate out different types of traffic onto differently performing routes. I'm equally sure that there are other good reasons to use Policy Based Routing it's just that I've not seen them.

Again, we're in a realm where Best Practices are few and far between because of the virtually infinite different ways of doing things.

3.9.2 Recommendation

Avoid using policy-based routing in a CINET environment, unless at least one of the following conditions is true:

- All applications establish affinity with a particular TCP/IP stack.
- The route destinations in each TCP/IP stack route table are mutually exclusive with the route destinations on the other TCP/IP stacks, including the default route.

3.9.3 Applicable Rule

Refer to your own, internal standards to establish what the policy based routing values SHOULD be and confirm that's what you're using.

4 Specific access ports (TELNET/TN3270/FTP)

4.1 A GENERAL OVERVIEW

We've separated out the specific access ports in this document as they represent a clear and present danger to your mainframe environment. By default, any user ID that is valid on z/OS can log into FTP or use Telnet. Both open up the possibility of exfiltration of data from your mainframe.

Now, don't get me wrong. Sometimes it's really useful to be able to get data off of the mainframe. But there are better ways to do it today than using the default options which don't really care about security or even auditability.

4.2 WHAT CAN GO WRONG?

As with other elements, getting the definitions wrong can have unpredictable results (anything from ports suddenly not requiring the use of SAF to get to mainframe resources, right on through to a complete "Denial Of Service" attack). These specific access ports have a large number of the mainframe TCIP/IP DISA STIGs against them and that's because they represent the primary paths into your system for hackers.

4.3 THE BEST OF PRACTICES

Secure everything! Monitor everything!

Your security product (ACF2, RACF and/or TSS) should be used to ensure that only specifically authorized users can get onto your mainframe using the specific access ports. This automatically enables the collection of SMF data for the activity. All you have to do is make sure you are a) recording the SMF data and b) that you create the additional reports needed to confirm appropriate use of the resources – STIG IFTP0060.

4.4 TELNET

Telnet is the utility that we use to sign on to UNIX systems everywhere (RFC 854). It's fantastic that the mainframe has a fully functional UNIX environment that we can exploit to help us mainframers keep relevance in an increasingly interconnected world. We do, after all, host over 80% of the world's data. It's not surprising that other platforms want access to our stuff!

Telnet (sometimes called otelnet) on z/OS does not use its own listener but engages the services of INETD to act as its listener. It defaults to using port 23 but organizations often change this to (slightly) obscure the route through to their machines. And you logon to the environment by using either RLOGIN or OTELNET.

One last quirk… You use an ASCII terminal or terminal emulator (like the VT100 emulator) to work in this UNIX environment coming in this way. If you use OMVS from ISPF then you're using EBCDIC. And the code pages aren't consistent either. Beware of unusual things happening if data is edited from the "wrong" place!

There are 9 DISA STIGs covering just the implementation of Telnet on a mainframe - ITNT0010, ITNT0030, ITNT0050, ITNT0060, IUTN0010, IUTN0020, IUTN0030, IUTN0040 and AAMV0380!

The STIGs touch on subjects as diverse as: making sure that your access ports are correctly defined; insisting that SMF is recording all related activity on the

mainframe; ensuring that you correctly identify the service as “restricted” to anyone who happens to end up on your Telnet services; recommending the correct approach to use when defining which UNIX resources will be used; and finally insisting that the z/OS files (both datasets and HFS or zFS) are correctly secured.

More STIGs seem to be added quite frequently at the moment so check regularly.

4.4.1 PORT

4.4.1.1 Description

The PORT parameter in the TELNETPARMS statement block specifies the port that Telnet listens on for non-secure (basic) connection requests.

The default value for Telnet port is 23.

4.4.1.2 Recommendation

There is little value to changing the port number from 23. Leaving it where it is means that you’re being a good “netizen” by ensuring that all machines participate equally. And to be honest, if you’re still relying on “security through obscurity” you shouldn’t be working in a mainstream IT environment!

Telnet is one of those “well known ports” and it just IS port 23.

4.4.1.3 Applicable Rule

The PORT parameter in the TELNETPARMS statement block should be set to 23 unless you have solid justification for behaving otherwise. I’d advise that you think long and hard about whether you want to still be allowing the use of “raw” Telnet when AT-TLS secured Telnet is an available option.

4.4.2 SECUREPORT

4.4.2.1 Description

The SECUREPORT parameter in the TELNETPARMS statement block specifies which port Telnet listens on for secure connection requests from a client using the SSL protocol. If the SECUREPORT parameter statement is not coded, Telnet does not support secure access from a client using SSL.

4.4.2.2 Recommendation

The decision on whether to set this parameter is entirely down to whether you allow SSL to be used for authentication in your organization.

That decision might be swayed by the deprecation of SSL. Direct quote from Wikipedia:

“Both SSL 2.0 and 3.0 have been deprecated by the IETF (in 2011 and 2015, respectively). Over the years vulnerabilities have been and continue to be discovered in the deprecated SSL protocols (e.g. POODLE, DROWN).”

4.4.2.3 Applicable Rule

You should not specify SECUREPORT in the TELNETPARMS statement block unless your organization specifically allows pure SSL to be used for connections.

4.4.3 TTLSPORT

4.4.3.1 Description

Tech speak alert!

The TTLSPORT parameter in the TELNETPARMS statement block specifies which port Telnet listens on for secure connection requests from a client that uses the TCP/IP AT-TLS interface. If you use the TTLSPORT statement, then you can define security parameters in AT-TLS policy rather than the Telnet profile. The location of the AT-TLS policy is managed by PAGENT and can be found in the CommonTTLSConfig parameter value. Without this configuration file specified, AT-TLS simply doesn't run. The file (z/OS sequential dataset or USS file) pointed to contains some of the following parameters used to affect AT-TLS protocol requests:

Parameter	**Used for:**
TTLSCipherParms	Defines the cipher specifications for your AT-TLS environment or an AT-TLS connection. TTLSCipherParms can be specified inline in a TTLSEnvironmentAction or TTLSConnectionAction statement or referenced by a TTLSEnvironmentAction or TTLSConnectionAction statement.
TTLSConnectionAction	Defines attributes for a subset of connections that need attributes different from those specified on the TTLSEnvironmentAction or TTLSGroupAction statement that is referenced by the same TTLSRule statement.
TTLSConnectionAdvancedParms	Defines attributes for a subset of connections that need attributes different from those specified on the TTLSEnvironmentAdvancedParms statement that is referenced by the same TTLSRule statement.
TTLSEnvironmentAction	Defines the attributes for an AT-TLS environment. A TTLSEnvironmentAction statement is required if the TTLSGroupAction statement, referenced on the same TTLSRule statement, specifies TTLSEnabled as On.
TTLSEnvironmentAdvancedParms	Defines the attributes for an AT-TLS environment. A TTLSEnvironmentAction statement is required if the TTLSGroupAction statement, referenced on the same TTLSRule statement, specifies TTLSEnabled as On.
TTLSGroupAction	Defines the parameters for a Language Environment process required to support secure connections. The TTLSGroupAction statement indicates whether a selected connection should use AT-TLS security. It can also specify the environment variables the Language Environment process should be initiated with.
TTLSGroupAdvancedParms	Defines additional, advanced attributes for an AT-TLS group.
TTLSGskAdvancedParms	Defines advanced attributes for an AT-TLS environment that are specific to System SSL.

TTLSGskLdapParms	Defines a set of LDAP parameters to be used for Certificate Revocation List (CRL) checking for an AT-TLS environment action. A TTLSGskLdapParms statement can be specified inline in a TTLSEnvironmentAction statement or referenced by a TTLSEnvironmentAction statement.
TTLSKeyringParms	Defines a set of key ring parameters for an AT-TLS environment action. A TTLSKeyringParms statement can be specified inline in a TTLSEnvironmentAction statement or referenced by a TTLSEnvironmentAction statement. This includes the location of the key ring which could be a USS file or a SAF KEYRING.
TTLSRule	Defines an actual AT-TLS rule and is by far the most complex of this set of parameters. The FLUSH/NOFLUSH and PURGE/NOPURGE parameters are used to specify whether or not AT-TLS policies are deleted at startup (and when a MODIFY REFRESH command is entered) and shutdown, respectively. The information provided on the TTLSRule statement defines an AT-TLS rule. The AT-TLS rule must have at least one local IP address, remote IP address, local port, remote port, job name, or user ID specification. The AT-TLS rule must have a direction specification and a TTLSGroupActionRef parameter. The AT-TLS rule can contain a priority, a TTLSConnectionActionRef parameter, a TTLSEnvironmentActionRef parameter and an IpTimeCondition specification. An IpTimeCondition specification identifies a time period when the AT-TLS rule is in effect.
TTLSignatureParms	Defines the PAGENT client elliptic curve preferences and the signature algorithm pair specifications for your AT-TLS environment or an AT-TLS connection. A TTLSSignatureParms statement can be specified inline in a TTLSEnvironmentAction or TTLSConnectionAction statement or referenced by a TTLSEnvironmentAction or TTLSConnectionAction statement.

The point of including these here is not to make you a Crypto Specialist but to show that there are an awful lot of parameters that refer backwards and forwards within a single statement set (my spellchecker absolutely HATES all of the parameter names!). You need to keep an eye on them all.

And one last thought on this… The policy data can be updated using the ISPF editor OR zOSMF. ISPF leaves an audit trail of sorts but zOSMF is a little trickier in its behavior. And any changes will be implemented virtually immediately. If anything goes wrong you need to be sure you have a way of getting back to the previous policy.

4.4.3.2 Recommendation

There is little justification for not using AT-TLS. However, I would urge you to think about the implementation in YOUR organization rather than rely on defaults to get you going.

4.4.3.3 Applicable Rule

Refer to your own, internal standards to establish what the values SHOULD be and confirm that's what you're using.

4.4.4 CONNTYPE

4.4.4.1 Description

The CONNTYPE parameter refers to the connection type and can be coded in TELNETPARMS, PARMSGROUP or both. It's only valid when either SECUREPORT or TTLSPORT are also coded.

Tech speak alert!

There are a number of values which can be specified in the CONNTYPE parameter:

Value	Used for:
SECURE	Indicates that the traditional SSL handshake is used to start the SSL connection. If the client does not start the handshake within the time specified by SSLTIMEOUT, an attempt is made to do a negotiated SSL handshake. If the client rejects the negotiated attempt, the connection is closed. Telnet is initialized for secure ports SECUREPORT or TTLSPORT with CONNTYPE SECURE and for basic ports with CONNTYPE BASIC.
NEGTSECURE	Indicates that a TN3270 negotiation with the client determines if the client is willing to enter into a secure connection. If the client agrees, SSL protocols are used for all subsequent communication. If the client does not agree, the connection is closed.
BASIC	Indicates that a basic (non-SSL) connection is used.
ANY	Indicates that the client can connect as secure or basic. Telnet first tries a standard SSL handshake. If the handshake times out, negotiated SSL (see CONNTYPE NEGTSECURE) is attempted. • If the client is willing to enter into a secure connection, SSL protocols are used for all subsequent communication. • If the client is not willing to enter into a secure connection, a basic connection is used.
NONE	Indicates that any client connection request is rejected.

Changing the value will change the behavior of your Telnet service. It could be that your secure system gets opened up to attack/monitoring OR the opposite is used to cause you a Denial Of Service.

4.4.4.2 Recommendation

Pick a value that is appropriate for the type of service that you will be operating and monitor the statements for changes. You should be aware that both BASIC and ANY will allow unencrypted traffic over the Telnet protocols.

4.4.4.3 Applicable Rule

Refer to your own, internal standards to establish what the value SHOULD be and confirm that's what you're using.

4.5 TN3270

In the old days before the internet people used lease lines (some still for various reasons often related to high levels of security need) but it's expensive. This (and other communications protocols including SNA) was managed in mainframe land by an IBM product called VTAM (now part of z/OS Communications Server).

TN3270 is a specific variant of Telnet that we have in mainframe land. Think of it as the place where TCP/IP meets SNA.

Telnet or a functional equivalent is used as the primary method of connection between client workstations and the SNA mainframe environment. To make this form of remote connection as seamless as possible to "old skool" mainframe users, the TN3270E protocol simulates actual SNA terminals (controlled by VTAM) as closely as possible.

Telnet assigns LUs (Logical Units) based on the LU mapping statements supplied in your organization's VTAM definitions. And a single instance of the TN3270E server can support up to 128,000 emulated 3270 display terminals.

4.5.1 Description

TN3270E can be viewed as the standard IP-based method of communicating with a mainframe. The term TN3270 can be used synonymously with TN3270E. And TN3270E is currently defined in RFC 2355.

- The TN3270E server can listen on multiple ports.
- More than one instance of the TN3270E server can run concurrently.
- Listening can be controlled so that it is only active on one specific IP address whether you run a single or multiple instances of TN3270E.

TN3270 can be implemented in z/OS either as part of the TCP/IP address space or (from z/OS V1R6 and now the usual method) in its own address space.

The definitions of the TN3270E server are identical in both methods. Except, if you run the TN3270E server as a separate task, its definitions cannot be placed in the TCP/IP PROFILE.TCPIP data.

The startup JCL has a PROFILE DD statement that points to a profile data set containing parameters to control the TN3270E server. It is absolutely vital that you use the external security manager to control who has access to this file.

Within the PROFILE.TCPIP are two fundamental statement blocks used to define TN3270E server behavior:

- TELNETPARMS
- BEGINVTAM

TELNETPARMS parameters:

Parameter	Used for:	What can go wrong:
DEBUG EXCEPTION CONSOLE	Activates TN3270E session tracing but only for time-outs and explicit errors.	If I can change this value, I can flood your system with unexpected activity.
TCPIPJOBNAME	The TCPIP jobname associated with the stack that you want this traffic to go to.	If I can change this then I can point your TCP/IP traffic to my own stack giving me complete control of your environment!
PORT	The PORT to use for both binding and listening.	If I change this I can stop you from being able to connect!
INACTIVE	Allow TN3270E to close/ tidy up inactive sessions after a set period of inactivity.	If I change this then I effectively halt your system by filling it up with half abandoned sessions meaning new ones can't start.
PRTINACTIVE	Allow TN3270E to close/ tidy up inactive sessions after a set period of inactivity.	Same problem as INACTIVE.
TIMEMARK	Works in tandem with SCANINTERVAL to identify inactive sessions.	Same problem as INACTIVE.
SCANINTERVAL	Works in tandem with TIMEMARK to identify inactive sessions.	Same problem as INACTIVE.
LUSESSIONPEND	Enables specific behavior such that when a client user enters the LOGOFF command to end a session with a VTAM application, the connection isn't immediately dropped. Instead, the client is returned to the screen from which the logon originally was invoked.	If I can change this value then I can change your environmental behavior so that it will drop me out into a much more useful place than just a logoff screen.
MSG07	Enables the sending of a VTAM USS message 7 to the client in the event of a logon failure. This statement should normally be coded. Message 7 is for logon failures.	If I change this value your users won't know that a logon attempt has failed. That might give me the time I need to complete the hack I am performing with their userid.

TELNETDEVICE	Controls the SNA session characteristics you want for both the TN3270E portion and SNA portion of the connection. The second column indicates the device name. The last two columns indicate the logon mode entry (session characteristics) to be used for a TN3270 and a TN3270E connection, respectively.	I can either change these values to make it easier for me to get on to your systems OR to make it harder for your real users to work!

BEGINVTAM parameters:

Parameter	**Used for:**	**What can go wrong:**
PORT	Used to connect this BEGINVTAM block with a TELNETPARMS statement for the same port number. Huh? This instance of TN3270E makes connections to port 23 using these BEGINVTAM statements as well as the TELNETPARMS statements for the same port number.	If I can change this value then I can stop your Telnet activity dead in the water. A mismatch in the parameter sets will result in failures.
DEFAULTLUS	When a TN3270E client connects to the TN3270E server, it needs to be mapped to an LU that the TN3270E server can use to represent this client on the SNA session. If the client does not specify a specific LU, and if no other mapping statement directs a different LU to be used for this client, then an LU from DEFAULTLUS mapping is assigned to the connection.	If I can change this then I could, for example, reduce your DEFAULTLU pool to zero preventing use of Telnet.
LUGROUP	Doesn't do any mapping. Instead, it defines a group of LUs that can be used for TN3270E terminal sessions.	Changing these values will prevent Telnet from being able to negotiate with VTAM. Result? Your users can no longer use Telnet.
PRTGROUP	Doesn't do any mapping either. It defines a group of LUs that can be used for TN3270E printer sessions.	If I change these values then I can control where your print output goes to. Printing of an open, inter-office memo – boring. Printing off customer details – Bingo!

IPGROUP	Another statement that doesn't do mapping. It defines a group of IP addresses that identify TN3270E clients (an IPGROUP is referred to as a client identifier, and there are many other client identifiers available).	Changing these values will prevent Telnet from being able to negotiate with TCP/IP. Result? Your users can no longer use Telnet.
LUMAP	Maps a group of LUs to a group of clients.	If I can change this value then I can stop your Telnet activity dead in the water. A mismatch in the parameter sets will result in failures.
PRTMAP	Maps a PRTGROUP to a group of client, so that a TN3270E connection from a client can associate a printer from this group. **Note:** Together, these two LUMAP and PRTMAP statements form the one-to-one mapping that is necessary for a TN3270E printer connection to utilize the ASSOCIATE command.	See my earlier comment about controlling your printers!
USSTCP	Specifies that the VTAM USS message 10 panel (a logon panel similar to native SNA terminals) be presented for the initial connection. If LUSESSIONPEND is coded, a client is returned to this screen after logging off from an application. This is another way to implement the STIG recommendation (ITNT0020) about showing a welcome screen.	If I can change this value I can drop myself into a more useful network connection than a logoff screen.
DEFAULTAPPL	An older methodology for getting a logon/logoff screen. In the unlikely scenario that you don't want a USSMSG10 panel, this statement could be used to direct a client to a specific application (TSO for example) at connect time. This can be used to prevent hackers from browsing around your environment so easily (ITNT0020).	If you're still using this and I can change this value I can drop myself into a more useful network connection than a logoff screen.

LINEMODEAPPL	Allows for a client to negotiate line mode when connecting to the TN3270E server. In the example shown above, the setup connects the client to the TSO application.	If I can change this value then I can control where in your network I get dropped into if I'm connecting with a LINEMODE terminal emulation.
ALLOWAPPL	Limits the selection of application for a TN3270E client. This is a security statement to control what applications can be selected from the USSMSG10 panel.	If I can change this value then I can control where in your network I get dropped into if I'm connecting with a LINEMODE terminal emulation.

VTAM must also be set up to allow TN3270E to allocate LUs. This book does not discuss the required VTAM elements.

4.5.2 Recommendation

The TELNETPARMS statement block (ended with ENDTELNETPARMS) contains the TN3270E protocol and other non-VTAM attributes and is found in your PROFILE.TCPIP data.

You must create a unique TELNETPARMS block for each port and for each qualified port and you can only code the PORT, SECUREPORT, or TTLSPORT statements (4.4.1) in the TELNETPARMS statement block.

The BEGINVTAM statement block (ended with ENDVTAM) contains parameters used to define characteristics that are related to the mapping of the VTAM configuration.

Whilst there are some ground rules for Best Practice, VTAM (and any networking) is a complex matter and you should ensure that you are auditing for what you expect to be defined.

The BEGINVTAM statements are not the same as the VTAM definitions for LUs. These will be found within your VTAMLST concatenation. You can establish the VTAM search order by looking at the output of the started task associated with it.

And finally, this section does not cover all of the TELNETPARMS statements, just those that are vital to TN3270E. If you need to know more: z/OS Communications Server: IP Configuration Guide.

4.5.3 Applicable Rule

Refer to your own, internal standards to establish what the TN3270E values SHOULD be and confirm that's what you're using.

But there are a few "rules of thumb":

SERVAUTH must be enabled – STIG ITCPR052.

As an absolute minimum you should have a EZB.TN3270.* profile defined in the
As an absolute minimum you should have a EZB.TN3270.* profile defined in the
SERVAUTH class with UACC(NONE) – STIG ITCP0050. This allows the use of SMF to record audit data for the use of TN3270 **(When specified with the SAFCERT sub-parameter, CLIENTAUTH)** as well as enabling the controlled use of them – STIG AAMV0380.

4.6 FTP

All computer systems use data. That data is normally held within containers that we call files. And it is only natural that we will need to move those files around. The File Transfer Protocol or FTP is a standard method for achieving that.

It's a service that is implemented on the majority of systems and the parameter sequences are usually the same across most instances. In fact, that's one of the things that enables hackers to poke around unfamiliar systems, the tools are "platform agnostic" i.e. they're the same everywhere they're implemented.

4.6.1 Description

The z/OS FTP service is a UNIX Systems Services (USS) application. You can start an FTP service in the z/OS environment but it immediate forks itself off into the USS space and tells the parent task to kill itself. This can make it trickier than other z/OS tasks to keep an eye on.

The FTPD task can be executed using /usr/sbin/ftpd and it's possible that some folks do just that. But starting the service as a z/OS started task means that the FTP server can be autologged by the TCP/IP started task. So, my advice is to start the FTP daemon using JCL.

An example of the started task JCL:

```
//FTPD      PROC MODULE='FTPD',PARMS=''
//FTPD      EXEC PGM=&MODULE,REGION=4096K,TIME=NOLIMIT,
//           PARM='/&PARMS'
//SYSFTPD   DD DISP=SHR,DSN=SYS1.TCPPARMS(FTPSDATA)
```

The SYSFTPD DD points to the FTPD configuration file, usually referred to as the FTP.DATA data set. Just like the TCP/IP started task, the FTP server searches other locations to locate the FTP.DATA configuration data set if SYSFTPD is not defined. So define it and make sure that the security definitions are tied down tight as with other examples of DD cards in previous sections – STIG IFTP0080. There is a related STIG for the USS files too – STIG IFTP0070.

Some FTP.DATA parameters that you should pay attention to (i.e. make sure they aren't changing unexpectedly):

Parameter	Used for:	What can go wrong:
ACCESSERRORMSGS	Sends detailed login failure messages to the requestor.	These messages usually tell me everything I need to be able to understand why my first attempt failed. Great if I'm a genuine user who struggles to remember their password. Not so great if I'm a hacker who wasn't sure what the application name I should be using was but now have as detailed a response as I could ever hope for!

ANONYMOUS	Allow someone to use FTP without needing to identify who they are.	I'm not sure that I really need to explain why this would be a bad thing! There's a bunch more parameters that enable anonymous processing of various kinds. They should all **NOT** be used: ANONYMOUSFILEACCESS ANONYMOUSFILETYPEJES ANONYMOUSFILETYPESEQ ANONYMOUSFILETYPESQL ANONYMOUSFTPLOGGING ANONYMOUSHFSDIRMODE ANONYMOUSHFSFILEMODE ANONYMOUSHFSINFO ANONYMOUSLEVEL ANONYMOUSLOGINMSG ANONYMOUSMVSINFO STIG - IFTP0020
APPLNAME	Specifies the FTP server application name.	IF you don't specify this value the job name of the JCL that starts the FTP server will be used. That's fine as long as it was you starting the server. What if I did so without your knowledge? Changing the APPLNAME could affect which security profiles would be used when deciding who should have access.
AUTOMOUNT	Enables USS to mount any volume it needs to be able to satisfy the FTP request.	So I don't need to be too accurate about whether I'm looking for current or old data? Cool! Ditto with AUTORECALL and AUTOTAPEMOUNT.
BANNER	Specifies that a "welcome banner" should be displayed to any new connection.	If you don't show me one of these that says I'm entering a secured system then I'm not doing anything wrong/illegal by poking around. STIG - IFTP0060
FTPLOGGING	Specifies that the FTP server logs FTP session activity for unknown users (that is, users that are not anonymous users).	Without these logs you'll be quite unlikely to be able to track what's gone missing from your system.

SMF	Specifies which SMF records will be used to keep audit activity data.	This kind of audit data isn't available on other platforms. It's a positive bonus to have the data when you're trying to understand what a hacker has done on your system. STIG - IFTP0060
STARTDIRECTORY	Specifies where you will find yourself when an FTP connection is established.	It's tempting to make this value as generic as possible but if you put me into your system in root then I'm going to find getting around much easier.
VERIFYUSER	Specifies whether the FTP server should verify whether a user attempting to log into FTP has been granted access to the server's port profile in the SERVAUTH class.	Changing this value could alter the way users are given permission to use FTP. For example switching it off disables any RACF profile checks to EZB. FTP.** resources.

The basic z/OS UNIX Systems Services FTP is quite "open" in security terms. As supplied, it is open to everyone with an id. However, this is addressable by use of 1 (or more) of 3 options at your disposal to restrict the availability of this service:

- write (in Assembler) and install the FTCHKPWD exit (but be aware of STIG - IFTP0040 which discourages this as an option), or
- configure TLS authentication which was the old way to add a SAF check (RACF, ACF2 or TSS)
  - In this case (as long as the SECURE_LOGIN option is set to VERIFY_USER in FTP.DATA) the profiles that are checked for READ access will be in the SERVAUTH class and look like this: EZB. FTP.<*systemname*>.<*ftpdaemonname*>.PORTxxxx
  - Client Authentication requires that the user present a valid x.509 client certificate
- Just use RACF (or ACF2 or TSS from z/OS v1r10) to define a resource profile with UACC(NONE) in SERVAUTH for: EZB. FTP.<*systemname*>.<*ftpdaemonname*>.PORT*xxxx*.
  - Again, this requires that SECURE_LOGIN VERIFY_USER be set in FTP.DATA
  - Grant READ access to the resource to anyone who needs to use FTP

It's an acknowledged strength to have all of your mainframe security handled by your external security manager (RACF, ACF2 or TSS). Be strong ☺

One more thing… STIG IFTP0010 discusses, in some detail, the Userid requirements for the FTP server to ensure that you don't accidentally allow more access than you intended using FTP. The Userid requirements include: the PROTECTED attribute (in RACF this is the NOPASSWORD parameter on the AU command, etc.), OMVS segment attributes: UID(0), HOME('/'), PROGRAM('/bin/sh').

And finally, there is a UDP version of FTP. It's called TFTP (Trivial File Transfer Protocol) and it's available on z/OS as with the other, platform agnostic protocols. The protocol does not require ANY security checking and so should never be allowed to start! You can prevent its use by locking down the use of port 69 so that no one is authorized to use it – STIG IFTP0090.

4.6.2 Recommendation

Never allow anonymous use of FTP – STIG IFTP0030.

Never allow any use of TFTP – STIG IFTP0090.

Consider using a "welcome banner" to deter hackers/unauthorized users from poking around what they now know to be a secured and monitored system –STIG IFTP0050.

All FTP activity should be encrypted – STIG IFTP0100.

4.6.3 Applicable Rule

SERVAUTH must be enabled – STIG ITCPR052.

As an absolute minimum you should have a EZB.FTP.*.*.PORT* profile defined in the SERVAUTH class with UACC(NONE) - STIG ITCP0050. This allows the use of SMF to record audit data for all use of FTP **(as long as the SECURE_LOGIN option is set to VERIFY_USER in FTP.DATA)** as well as enabling the controlled use of the service – STIG AAMV0380.

4.7 A brief word about SMTP

Simple Mail Transfer Protocol has been around for a while. It's the base protocol we use to get email from one place to another. Both IBM and DISA have things to say about it. Let's start with IBM's statement of withdrawal of z/OS SMTPD:

"If you are currently using Communications Server SMTPD on z/OS®, you should migrate to CSSMTP. IBM® has issued a statement of direction indicating that SMTPD will be withdrawn in a future release. CSSMTP has been designed to support z/OS users of SMTPD that create mail on the JES spool data set using batch jobs or that use SMTPNOTE for delivery to the Internet. The mail messages you send today using SMTPD can probably be sent with CSSMTP."

There are some differences between SMTPD and CSSMTP and they are mostly concerned with security and/or the ability to use the mainframe email service for exfiltration of data. Some of the primary differences that require migration consideration are as follows:

- CSSMTP does not provide SMTP listener support; it cannot be used to receive mail to z/OS for delivery to TSO users. You must continue to use SMTPD for this kind of support or consider using sendmail for delivering mail to mailboxes defined in the z/OS UNIX file system. Either of these options leaves an insecure door to your mainframe.
- CSSMTP only delivers mail to systems that have TCP/IP host names. If you are using a local TSO user ID on the POSTMASTER statement for your SMTPD server, then the local TSO user ID must be changed to an email address with the format userid@host.domain on the MailAdministrator statement for CSSMTP.
- CSSMTP does not perform DNS lookups to determine a target server for each recipient. CSSMTP requires a connection to an SMTP or ESMTP server that receives the mail and handles next-hop delivery. Typically, the target server is located at a remote destination, but could also be a local instance of sendmail.
- If mail that CSSMTP sends to the target server becomes undeliverable, CSSMTP is not able to receive the undeliverable mail notification because CSSMTP is not a server.
- CSSMTP processes a mail message directly from the JES spool data set and sends the mail message to the target server. If CSSMTP cannot deliver the mail

message immediately, the message is held for a short period of time, defined by the RetryLimit statement. After this retry period expires, CSSMTP saves the mail message in the z/OS UNIX file system for the period of time defined by the ExtendedRetry statement. If CSSMTP cannot deliver the mail within any configured retry time limit, the mail becomes undeliverable. CSSMTP stores the mail in a z/OS UNIX directory between retry intervals.

- CSSMTP does not support source routing, such as the <@host1,@host2:userid@host3> or <NJEuserid%NJEhost> formats.

  For example, in the address string @host1,@host2,...,@hostn:user_id@host_name, the @host1,@host2,...,@hostn portion of the address is ignored and user_id@host_name is used as the recipient value. If you are using source routing, you might want to change it to be an email address with the format userid@domain (mailbox). Otherwise, the mail might not be delivered.

And there's a STIG specifically about it too (ITCP0040) which talks about making sure you have the resources correctly secured if you're using CSSMTP.

And another STIG (ITPC0050) suggests that you must have the following SERVAUTH class profile defined if you run CSSMTP: EZB.CSSMTP.*sysname.writername.JESnode*

5 RACF (and other z/OS External Security Managers) Round-Up

Take a deep breath if you're under 40 and work on a mainframe…

RACF (or ACF2 or TSS) doesn't always know about everything that goes on in the mainframe!

If you're over 40 you'll remember the days before RACF was an "always call" product. It only got involved if you specifically told it that you were interested in securing a resource. I like to think that it was a blip in historical mind sets that led us to believe that we'd only ever need to secure up to 5% of our mainframe data. And IBM was just getting over having confidently declared to the world that one day there might be a need for up to 10 computers worldwide! Mind you, that was before we started connecting our "Big Iron" to the Internet and the term "Big Data" was invented!

Auditing (and securing) our z/OS internetworking protocols can **NOT** be achieved using RACF (et al) alone. Even the bits that CAN be secured using just the external security manager aren't that way by default. You must set up everything correctly to even raise an eyebrow with RACF (or the others).

That said, I'm providing you with a list of ALL of the SERVAUTH profiles that I've talked about so far and maybe a few related ones. You should not consider this to be an exhaustive list – you might be doing something I've never thought of!

SERVAUTH Profile	**Used for:**
BPX.DEMON	The Userid associated with the startup of the PAGENT Started Task must have READ access to this resource.
EZ or EZA prefix	Older TCP/IP 3.2 and 3.4 profiles. You should not have any of these defined.
EZB.*	"Backstop" profile for all supported z/OS internetworking protocols. You should have this defined with UACC(NONE) and no one in the access list to prevent protocols being used if they have not been correctly defined in RACF (etc.). New functions and features are introduced quite regularly in z/OS world. This is particularly important in an environment where. If you haven't told your system that no one should be able to use the protocol, then they will be allowed. RACF only gets involved if you ask it to.
EZB.CSSMTP.*sysname.writername.JESnode*	Controls who can use CSSMTP services.

EZB.FTP.*	"Backstop" profile for all FTP activity (as long as the SECURE_LOGIN option is set to VERIFY_ USER in FTP.DATA). You should have this defined with UACC(NONE) and no one in the access list to prevent uncontrolled use of FTP. Access to the required elements of FTP should be controlled using the following 4 profile types:
EZB.FTP.*sysname.* *ftpdaemonname.*ACCESS.HFS	READ access to this resource restricts a user to FTP only for data that is stored within USS HFS files. FTP can access data in the z/OS space too. If you don't want general users to be able to access any other data that they have READ authority to over FTP, then you must use this profile. This (or rather the lack of implementation of this) is one of the "features" that the Pirate Bay hackers used to offload HUGE amounts of mainframe data.
EZB.FTP.*sysname.* *ftpdaemonname*.PORT*xxxx*	Limits which port(s) can be used to login to FTP. Implementing these profiles to grant access to FTP means that you can keep a tighter control over who can use which ports for what.
EZB.FTP.*sysname.ftpdname.* SITE.DEBUG	Limits who can issue site wide debug commands. Implementing these profiles means that you can prevent people from poking around your environment by having FTP provide them with a huge amount of site specific information. From time to time your Systems Programming and/or Network Control teams may require temporary access to this profile. But this should only ever be a temporary granting of access.
EZB.INITSTACK.*sysname.* *tcpprocname*	Controls which users have access to the TCP/IP stack before PAGENT is active. Give READ access to all users who do not require PAGENT policies to access the TCP/IP stack; for example, PAGENT, NETVIEW, DB2 etc.
EZB.IPSECCMD.*sysname.* *tcpname.command_type* or EZB.IPSECCMD.*sysname.DMD_* *GLOBAL.command_type*	Limits who can issue which IPSEC commands either by TCP/IP instance or across all stacks. You should have this defined with UACC(NONE) but with anyone who needs to issue IPSEC commands being granted READ access.
EZB.NETACCESS.*sysname.* *tcpname.security_zonename*	Controls local user inbound and outbound access to network resources, and local user access to local IP address when explicitly binding to local interface (or using job-specific or destination-specific source IP addresses). Over simplifying a little – who can access which controlled networks.

EZB.NETMGMT.*sysname.tcpname*.IPSEC.CONTROL	Controls whether a user can issue NMI control requests to the local IKE daemon to manage IP filtering and IPSec function. E.g. activate and deactivate requests to a TCP/IP stack. We do not discuss NETMGMT further in this book. It is another quite complex subject.
EZB.PAGENT.*sysname.tcpprocname*.*	Controls which users can start, stop, and refresh PAGENT.
EZB.PORTACCESS.*sysname.tcpname.resname*	Controls who can listen on which non-ephemeral TCP or UDP port(s).
EZB.STACKACCESS.*sysname.tcpname*	Controls user ability to open a socket and get host name or host ID.
EZB.TN3270.sysname.*tn3270name*.PORT*xxxx*	Controls which users can use TN3270 services. You should have this defined with UACC(NONE) but with anyone who needs to use TN3270 being granted READ access (When specified with the SAFCERT sub-parameter, CLIENTAUTH).
IST.* or IST.NETMGMT.*	These profiles indicate the presence of a network management product on your system. E.g. IST.NETMGMT.*systmname*.SNAMGMT appears to be used by parts of the Mainview, Netmaster, Sysview and Omegamon products. Firstly, if you're seeing profiles that start IST at least your people have thought to secure the network management product using RACF. Yay! Secondly, all network management products can be used to subvert your systems. That's common knowledge and means that it's one of the first things a hacker will look for in your z/OS environment. We do not discuss network management protocols further in this book. It's an endlessly complex subject!

Don't forget that even if you have all of these profiles implemented, you are not necessarily preventing unauthorized access to your mainframes via internetworking protocols. That's one of the reasons that the PAGENT Intrusion Detection Services has been made available to us. There are nearly always multiple definition steps involved in getting any z/OS internetworking protocol to "play nice".

6 And Finally . . .

It's been a long, complicated ride so far (and thanks for sticking with me to the end ☺) but I'm not done yet. There are a couple of additional points I need to deal with that don't fit in any of the other sections:

I have always tried to ensure that I execute audits in as cooperative a way as possible with the people responsible for the day-to-day operations of their environments. I find that if the folks on the "front line" have concerns, then those are good places to look for problems. I achieve this by reminding people that the content of my report can list things that need work without assigning blame for why these changes are needed (amongst other things).

I'm actually trying to work with you to get you to a safer place in this increasingly unsafe IT world and my emphasis in a report can help to get you the budget that you need to undertake the work. It's always worked well for me but I often encounter teams who bring suspicion to the task. I've also encountered a lot of organizations where audit reports are just used to punish people and those teams are quite naturally suspicious of the whole process.

I hope that I've managed to convince quite a few teams around the world that audit can be a valuable tool to them. And hopefully the z/Essentials series has been helping too ☺

I also hope that the security teams I've worked with have felt empowered to take more control of their environments as a result of working with them. And so to one of my personal ambitions…

I long for a world where Applications Developers work alongside Security Teams to make sure that they are implementing code that "does no harm" in their environments. I've seen it a few times in strongly controlled environments but I have to say that it's much more common for Security Teams to have to work late into the night to implement emergency security support in a application which is going live the next day!

In my splendidly crafted, fully cooperative world we security geeks will understand the language of developers and vice versa. Until then there's a few more things you need to know:

> Developers don't care about TCP/IP protocols or ports.
>
> They talk about (amongst many other things) SOCKETs when dealing with TCP/IP applications.

Wikipedia has the following to say about network sockets:

> "A network socket is an internal endpoint for sending or receiving data within a node on a computer network. Concretely, it is a representation of this endpoint in networking software (protocol stack), such as an entry in a table (listing communication protocol, destination, status, etc.), and is a form of system resource."

I asked a friend who develops software to explain TCP/IP sockets to me and he said:

> "Two endpoints want to communicate, in a secured manner over an unsecured public network (the Internet). You want to provide the best possible security environment for protecting both the host at each end and the packets of information that will be exchanged between the parties. There are many

possible Network Security (IPSec) control points you should consider in order to assure the desired state of security. The PORT and IPAddress, together a SOCKET, are the fundamental building blocks that make endpoint communication possible. They represent opportunities for you to either enhance security or expose the host and increase endpoint vulnerability."

So if you ask a developer how they communicate with z/OS over TCP/IP, they will likely talk about sockets. And what they mean is the TCP/IP address of the mainframe partition along with the port number that they use to make that connection. It's a bit more complicated than that because a socket behaves like a "stack" (think of an organized pile) that developers write to or read from. And there are always local complexities of language to deal with but that's part of the fun of trying to understand what's going on!

And finally, we've had the luxury of being quite isolated as a platform in mainframe land. Let's face it, there are few people who really understand the platform and fewer who can explain it to someone who doesn't! But we're talking about participating in a wider IT world now where we are "Just Another Platform" connected to the internet.

If you want to run a secure 21st century mainframe then you need to get used to talking to people with little to no mainframe experience and treating them as the experts in their fields that they really are whether you are auditing OR securing System z!

Once again, thanks for sticking with me. It means a lot to a grumpy old mainframer that somebody is interested in what I have to say ☺