Ray Overby

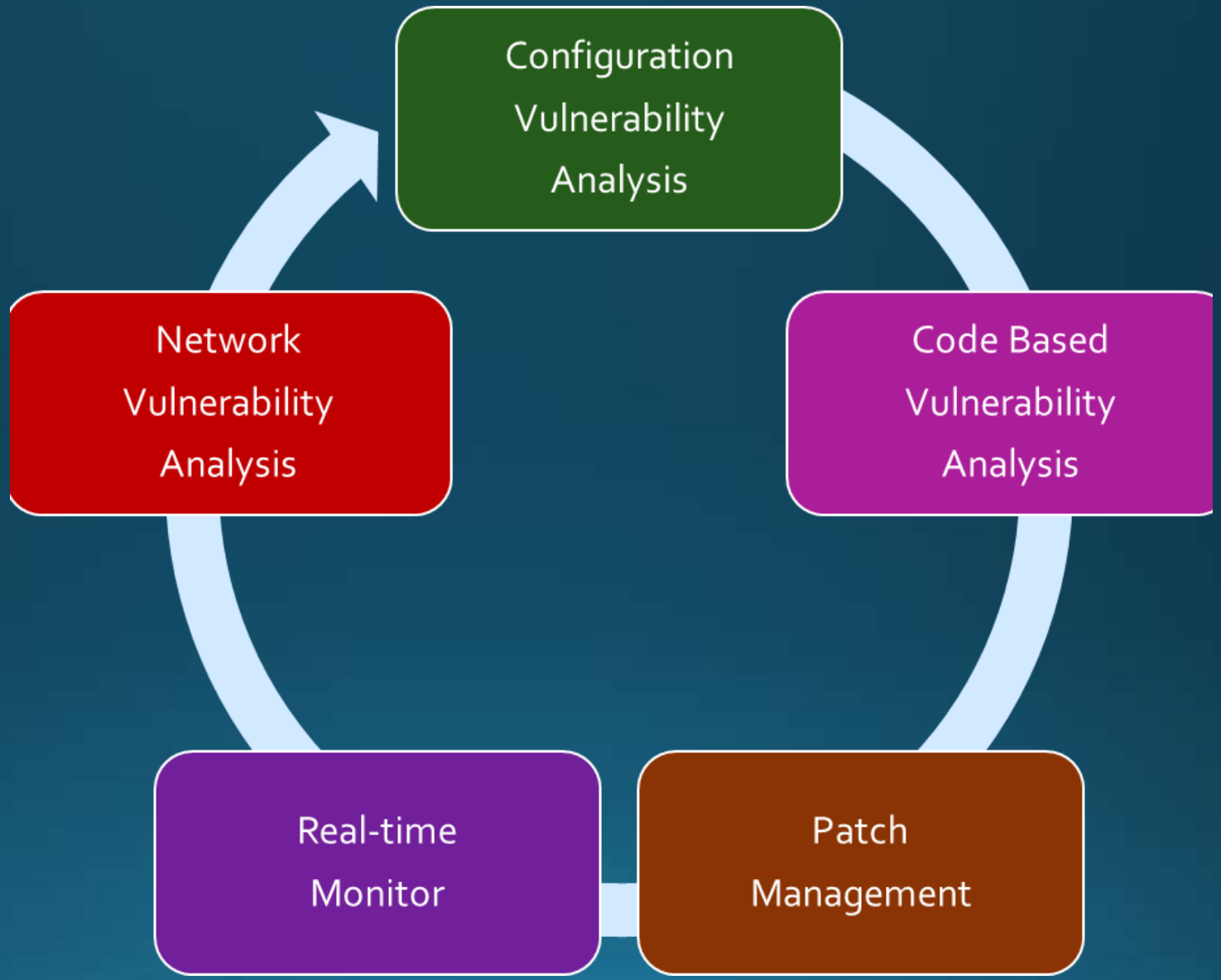
President, Key Resources, Inc.

Guidelines for Managing z/OS Vulnerabilities

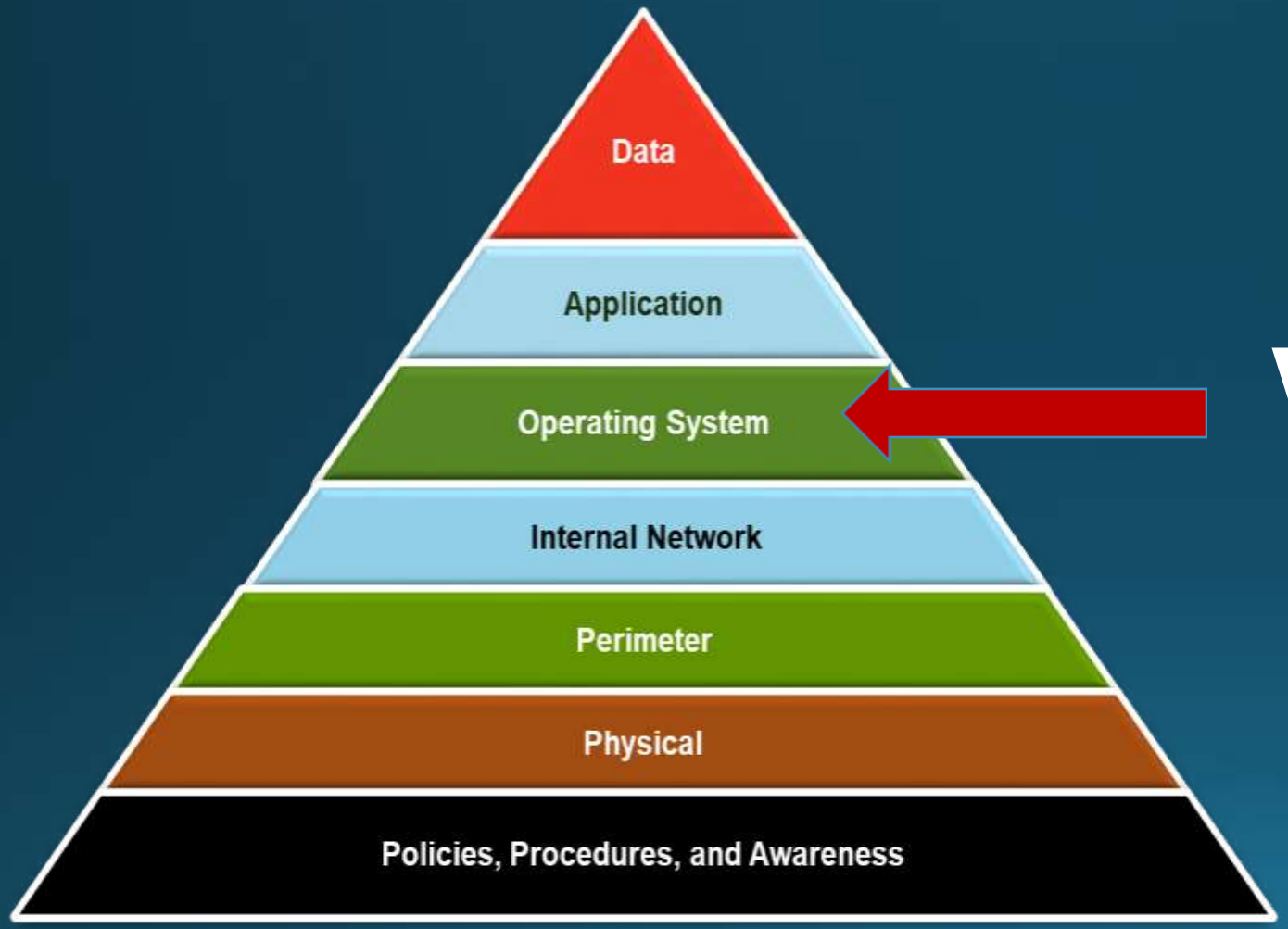
Agenda

- Level Set: What Comprises Mainframe Vulnerability Mgmt.
  - Components
  - Layers
- So, Isn't the Mainframe Operating System Layer Secure?
- Why Is it Vulnerable?
- Demo: What We've Found; What is an OS Layer Vulnerability?
- What Should We be Doing?

Mainframe Vulnerability Analysis



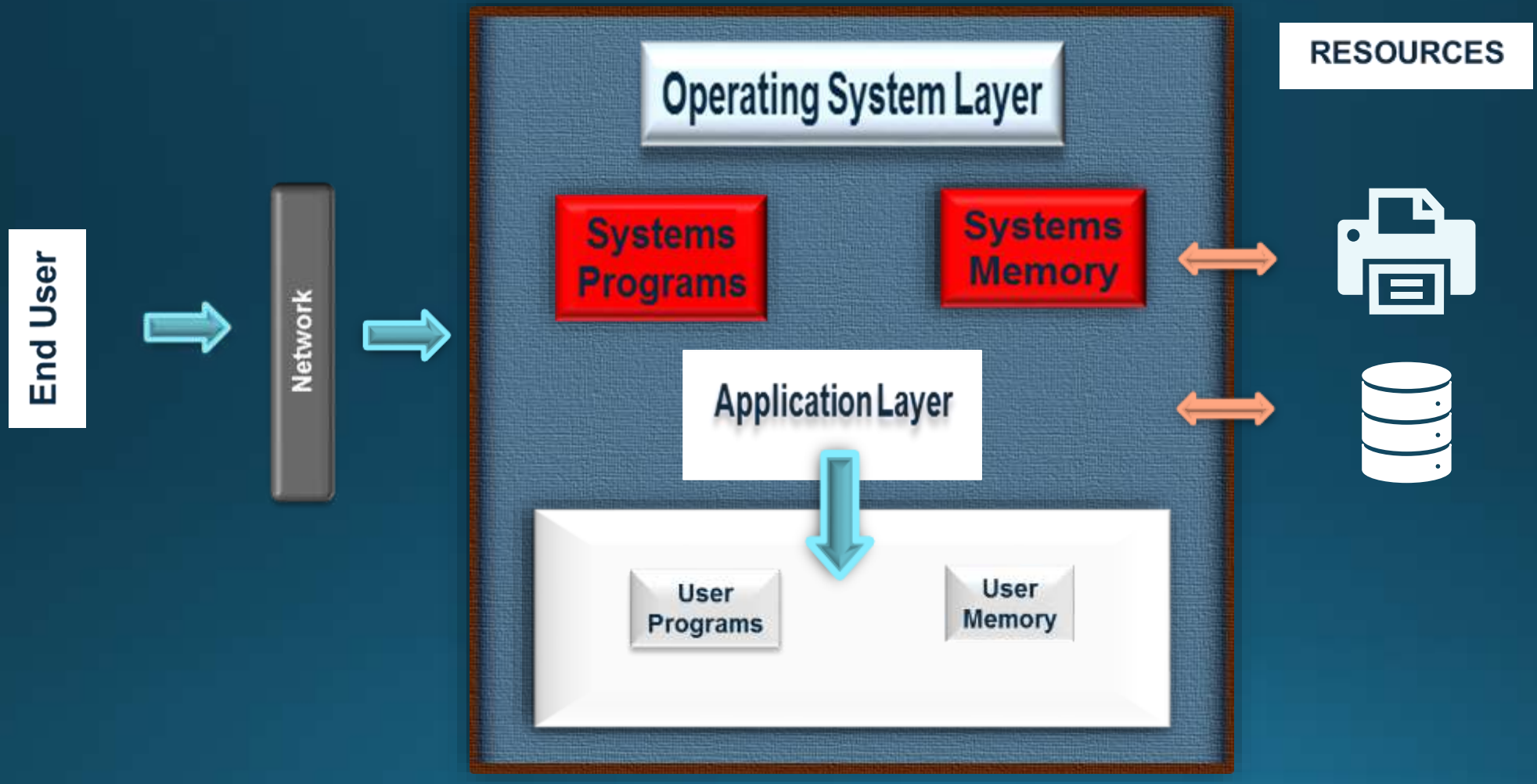
Defense in Layers: Mainframe Vulnerability Mgmt.



Why the Gap?

z/OS Architecture - What does Integrity Really Mean

What is a Privileged Instruction?



Our Premise: The Mainframe Operating System Layer is Vulnerable

It's All About Integrity

Systems Integrity is the reason mainframes are the most secure computer platforms

You can't trust Vendors to provide secure software and you can't rely on the software community to actively or organically surface vulnerabilities

What does that mean?

z/OS is Vulnerable

- Vendor software builds and releases are hurried with less testing; developers do not always have the skillset necessary to write integrity based software. Let's face it. Software has holes. And hackers love to exploit them. Mainframes are the big fish!
- Without Integrity you cannot have security; vulnerabilities in the OS layer can breach integrity without warning.
- ESM's: RACF, CA ACF2, and CA Top Secret are essential for establishing permissions and access control, but they were not architected to protect against OS level vulnerabilities
- IBM's System Integrity architecture is the reason mainframes are highly secure computer platforms, but vulnerabilities in OS level code will allow breaches (without the Enterprise Security Manager (ESM) issuing any type of warning).
- Mainframe Data Security should be built around a strong inclusive Vulnerability Management Program; NOT just around ESM's, Network Penetration testing, Configuration Management tools, and Interactive Application Security testing.

Why Is it Vulnerable?

- The attack surface is the boundary where attacks should be prevented.
- With respect to z/OS Integrity, the attack surface is between user or non-authorized user programs and authorized system services:
  - Program Calls (PCs)
  - Supervisor Calls (SVCs)
  - Authorized Programs (APF)
- These three interfaces are the methods used for requesting authorized system programs to provide services to a user program.
- Supervisor state can modify any area of memory (assuming credentials of other users including administrators or system personnel).

Why Is it Vulnerable?

- So how does a user program break through the attack surface and gain supervisor state?
- Typically, this occurs when one of the PCs, SVCs, or APF programs is either designed incorrectly or contains coding errors that allow a user program to bypass the integrity controls and obtain supervisory state.
- If a user program can bypass the controls in any of these methods of obtaining supervisor state it has broken through the attack surface and circumvents the IBM Statement of Integrity
- How bad can it get: A rogue user program can deny availability by overwriting critical system areas causing the system to crash.

DEMO
TRAP DOOR EXPLOIT

Trap Door Exploit Demo

You just saw a demonstration of an exploit of a Trap Door vulnerability. The exploit does not require APF. The exploit could be a CLIST or a REXX exec.

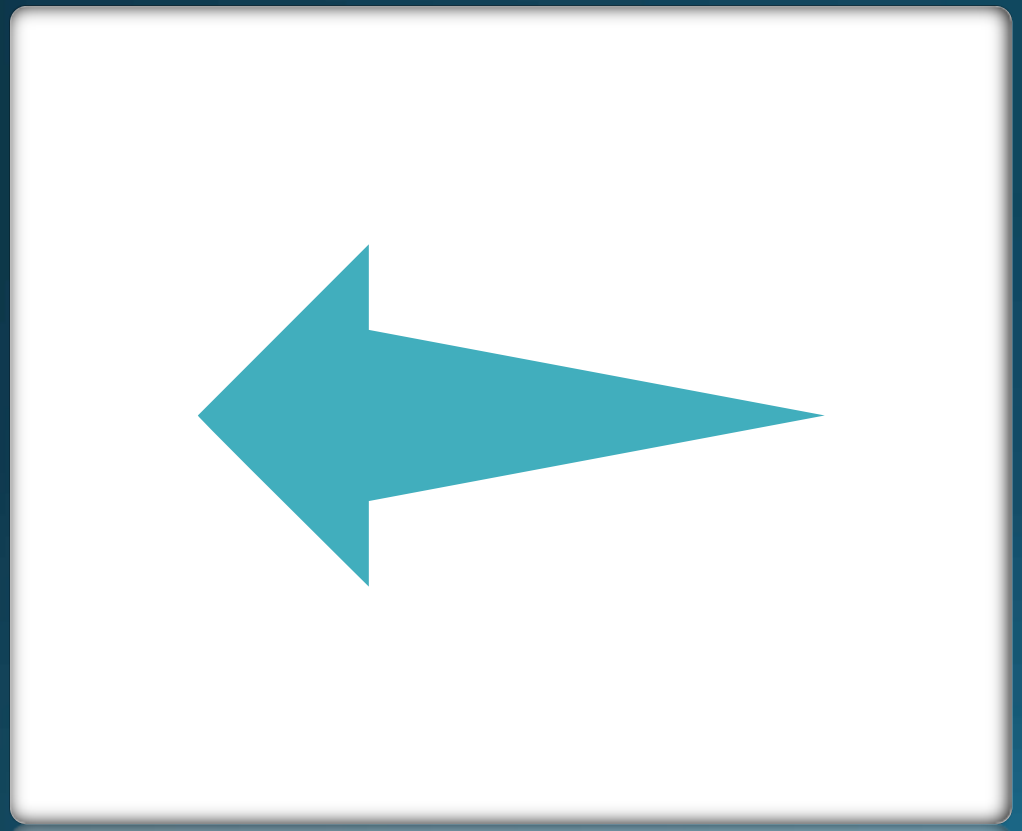
The DEMO:

- ✓ *Demonstrated that the user does not have access to the dataset before the exploit program was executed.*
- ✓ *Demonstrated that the user now has access to the dataset, and no security logging occurred after the exploit program was executed.*

Trap Door Exploit Demo

- The user is logged on to TSO on a z/OS 2.2 system with a RACF Userid that has no extra-ordinary security authorities.
- The exploit dynamically elevated the RACF credentials of this Userid.
- It gives the exploiter the Privileged attribute for bypassing RACF authorization checking.
- There is no logging of this activity.
- This is a typical Trap Door exploit of an OS Layer Vulnerability found in z/OS operating system code that has been found on production mainframes.

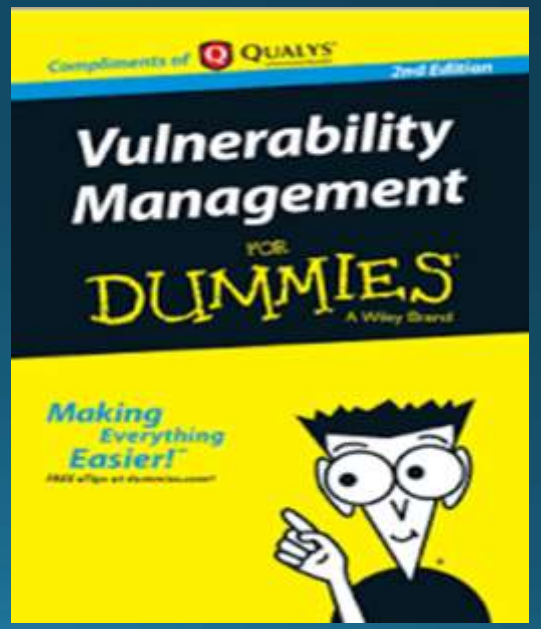
Trap Door Exploit Demo



- The user is logged on to TSO on a z/OS 1.13 system with a RACF Userid that has no extra-ordinary security authorities.
- The exploit dynamically elevated the RACF credentials of this Userid.
- It gives the exploiter the Privileged attribute for bypassing RACF authorization checking.
- There is no logging of this activity.
- This is a typical Trap Door exploit of an OS Layer Vulnerability found in z/OS operating system code that has been found on production mainframes.

What is the Mainframe Vulnerability Lifecycle?

You've All Seen Multitudes of These Diagrams and Books for Applications & Networks
NOT ONE INCLUDES z/OS



What is the Mainframe Vulnerability Lifecycle?

Guess what?

It's the Same for z/OS as it is for Every Other Layer!!!!!

What Should We Be Doing?

A Mainframe-based Vulnerability Management Program should include **ALL** layers.

Mainframe Data Security should be built around a strong Vulnerability Management Program; NOT just access and privilege management.

ESM's are essential for establishing permissions and access control, but this is **NOT** a complete security solution.

Secure your environment at every level. Make mainframe OS-level integrity a part of your overall security strategy.

Vulnerability Management across **all** platforms and operating systems is now the standard for many international compliance programs.

What Should We Be Doing?

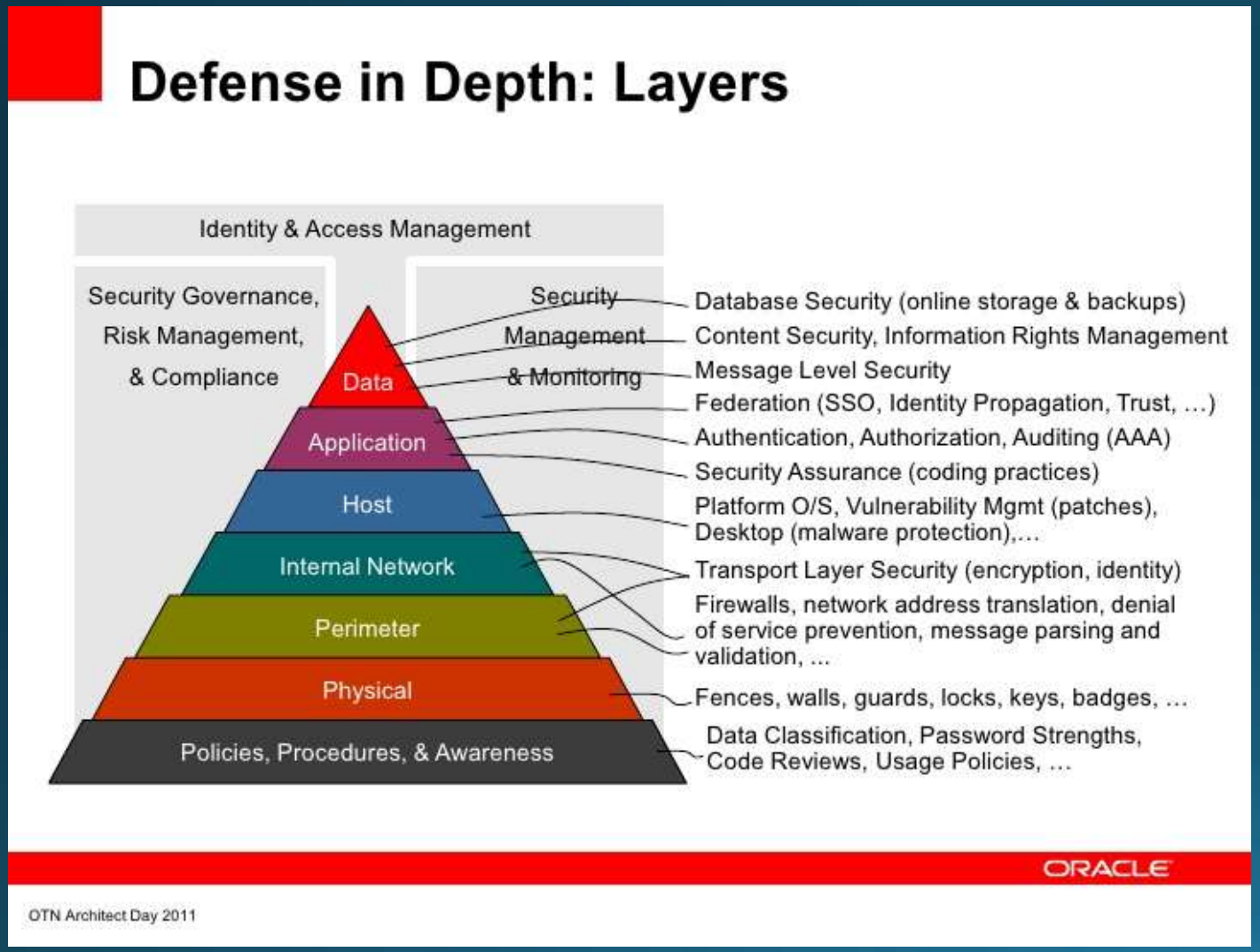
It should be noted that the IBM z/OS Statement of Integrity only applies to IBM code. It does not apply to any ISV code or installation written code. You, the z/OS system owner, are responsible for verifying the integrity of any code you add to z/OS.

https://www.ibm.com/it-infrastructure/z/capabilities/system-integrity.

Update you contracts to make sure vendors are responsible for fixing Integrity Vulnerabilities in a reasonable amount of time. Vendors should not be able to fall back on "product will need to be re-architected".

SEIM architectures will not address Integrity vulnerabilities in the OS layer. No data will be sent to the SEIM because forensic evidence is suppressed.

Remember: The Layered Approach



Protect Information rather than Systems – use an Interactive Application Security Testing (IAST) approach to accurately find vulnerabilities in the OS layer that, when exploited, allow unauthorized and undocumented access to sensitive information.

Thank You !

Contact Information

Ray Overby

Ray Overby
Key Resources, Inc.
ray.overby@krisecurity.com
Mobile: 1 847 406 0566
www.krisecurity.com

Mobile: 1 847 406 0566
www.krisecurity.com