Vulnerability Patterns on z/OS: Lessons on System Integrity

Laurie Ward – LWard@us.ibm.com
IBM Z Center for Secure Engineering

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

BigInsights
BlueMix
CICS*
COGNOS*
DB2*
DFSMSdfp
DFSMSdss
DFSMShsm
DFSORT
DS6000*
DS8000*
FICON*
GDPS*
HyperSwap
IBM*
IBM (logo)*
IMS
Language Environment*
MQSeries*
Parallel Sysplex*
PartnerWorld*
RACF*
Rational*
Redbooks*
REXX
SmartCloud*
System z10*
Tivoli*
UrbanCode
WebSphere*
z13
zEnterprise*
z/OS*
zSecure
z Systems
z/VM*

* Registered trademarks of IBM Corporation

The following are trademarks or registered trademarks of other companies.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.
Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.
Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.
ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.
Java and all Java based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.
Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and
Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
OpenStack is a trademark of OpenStack LLC. The OpenStack trademark policy is available on the OpenStack website.
TEALEAF is a registered trademark of Tealeaf, an IBM Company.
Windows Server and the Windows logo are trademarks of the Microsoft group of countries.
Worklight is a trademark or registered trademark of Worklight, an IBM Company.
UNIX is a registered trademark of The Open Group in the United States and other countries.
* Other product and service names might be trademarks of IBM or other companies.

Notes:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

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

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

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

This information provides only general descriptions of the types and portions of workloads that are eligible for execution on Specialty Engines (e.g, zIIPs, zAAPs, and IFLs) ("SEs"). IBM authorizes customers to use IBM SE only to execute the processing of Eligible Workloads of specific Programs expressly authorized by IBM as specified in the “Authorized Use Table for IBM Machines” provided at www.ibm.com/systems/support/machine_warranties/machine_code/aut.html (“AUT”). No other workload processing is authorized for execution on an SE. IBM offers SE at a lower price than General Processors/Central Processors because customers are authorized to use SEs only to process certain types and/or amounts of workloads as specified by IBM in the AUT.

IBM Z Center for Secure Engineering

What we do:

- Work across IBM to ensure that IBM Z product development follows best secure engineering practices
- Perform security reviews of IBM Z based products
- Perform security testing and build IBM Z unique testing tools

The Tall Question & Short Answer

Q: How many lines of code does it take to compromise all security and integrity on the z/OS solution stack?

The Tall Question & Short Answer

Q: How many lines of code does it take to compromise all security and integrity on the z/OS solution stack?

A: One.

Background: What is System Integrity?

- **Property of a system that prevents users from circumventing security mechanisms**
- **In z/OS, there is no way for an unauthorized problem program to:**
  - Bypass store or fetch protection
  - Bypass password/RACF protection
  - Obtain control in an authorized state

Background: What is “Authorized” on z/OS?

- Supervisor State (vs. Problem State)
- PSW Key 0-7 (vs. User Key 8-15)
  - also known as “System Key”
- PKM 0-7 (Program Key Mask)
  - Allows program to change to run in Key 0-7
- APF Authorization
  - A program loaded from an APF–authorized library and was link–edited with authorization code AC=1.

Background: Boundaries from user programs to authorized or privileged programs

- SVC routines
- PC routines
- APF authorized programs
- Program Properties Table programs
- UNIX set-user-id and set-group-id programs

Focus on the Boundary
& Specially Architected Instructions

The focus of this discussion is on the boundary between...

Unauthorized Requester

and its use of an

Authorized Service (PC or SVC)

Focus on the Boundary
& Specially Architected Instructions

The focus of this discussion is on the boundary between...

Unauthorized Requester

and its use of an

Authorized Service (PC or SVC)

The Requester's Parameters are NOT to be trusted. They must be *referenced in the caller's key*

MVCK – Move With Key
MVCSK – Move With Source Key
MVCDK – Move With Destination Key
MVCOS – Move With Optional Specifications

Agenda:
Vulnerability Patterns for z/OS

1. The Unintentionally Authorized PC
2. Untrusted Parms, Untrusted Regs
3. Untrusted, Indirectly Anchored Parms
4. Control Block Masquerade
5. Buffer Overflow
6. User Key Common Storage
7. Weak Security Configuration

Vulnerability Pattern #1:

The Unintentionally authorized PC

Vulnerability Pattern #1: The Unintentionally Authorized PC

Critical keyword on the ETDEF service defining a PC:

AKM

(The Authorization Key Mask)

Vulnerability Pattern #1:
The Unintentionally Authorized PC

Critical keyword on the ETDEF service defining a PC:

AKM

(The Authorization Key Mask)

AKM(0) restricts the PC usage to callers running in key 0

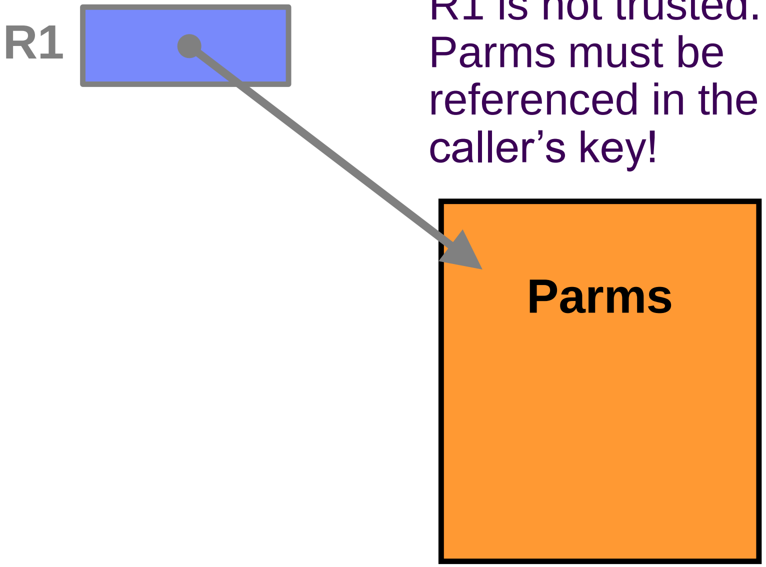
AKM(0:15) allows the PC to be used by any caller

If a PC target routine is *intended for authorized callers* but ***inadvertently allows unauthorized ones***, it’s highly likely to have an exposure!

Vulnerability Pattern #2:

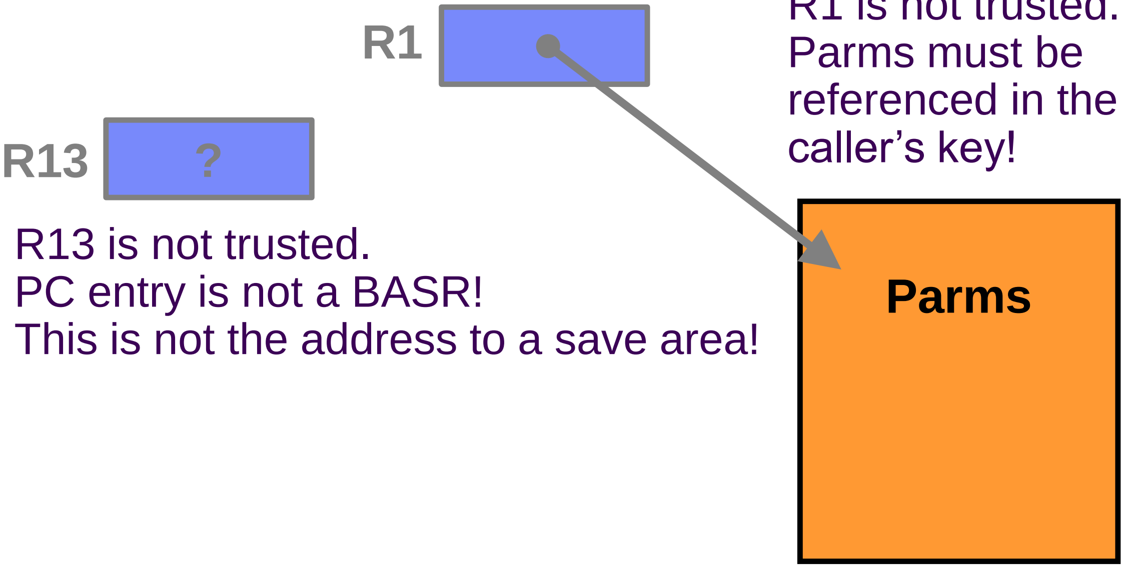
Untrusted Parms, Untrusted Regs

Vulnerability Pattern #2: Untrusted Parms, Untrusted Regs



R1 is not trusted. Parms must be referenced in the caller’s key!

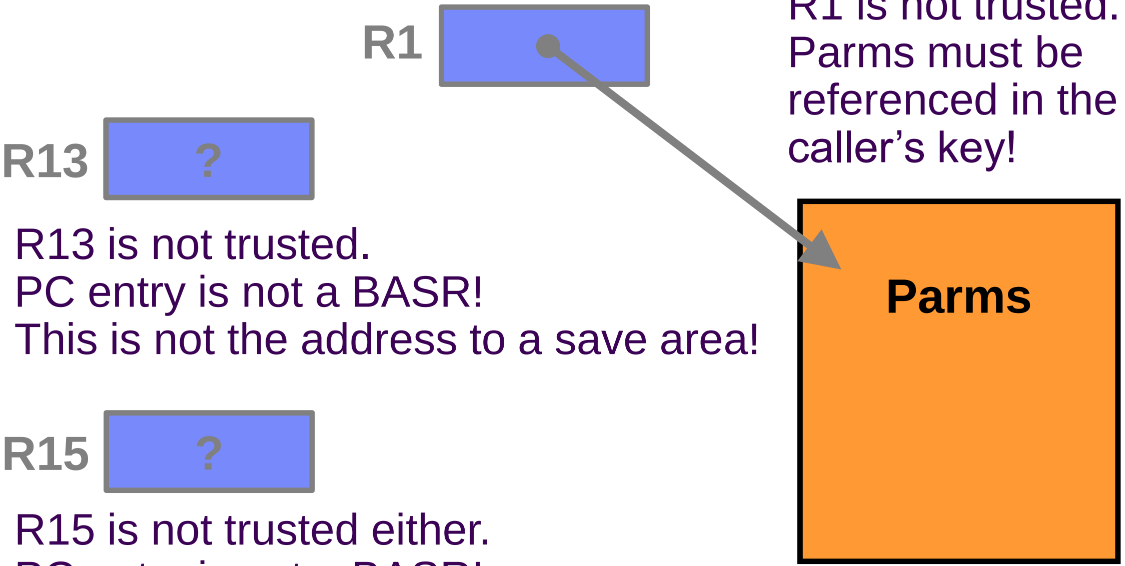
Vulnerability Pattern #2: Untrusted Parms, Untrusted Regs



R1 is not trusted.
Parms must be referenced in the caller’s key!

R13 is not trusted.
PC entry is not a BASR!
This is not the address to a save area!

Vulnerability Pattern #2: Untrusted Parms, Untrusted Regs

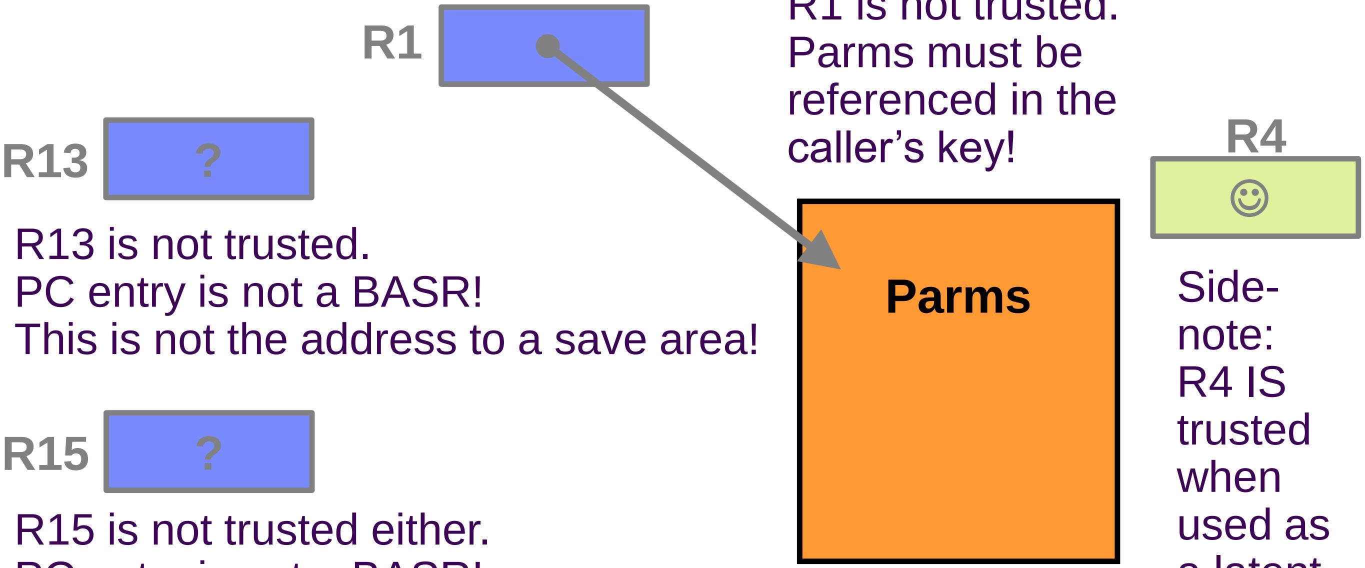


R1 is not trusted.
Parms must be referenced in the caller’s key!

R13 is not trusted.
PC entry is not a BASR!
This is not the address to a save area!

R15 is not trusted either.
PC entry is not a BASR!
This is not the entry point address!

Vulnerability Pattern #2: Untrusted Parms, Untrusted Regs



R1 is not trusted.
Parms must be referenced in the caller’s key!

R13 is not trusted.
PC entry is not a BASR!
This is not the address to a save area!

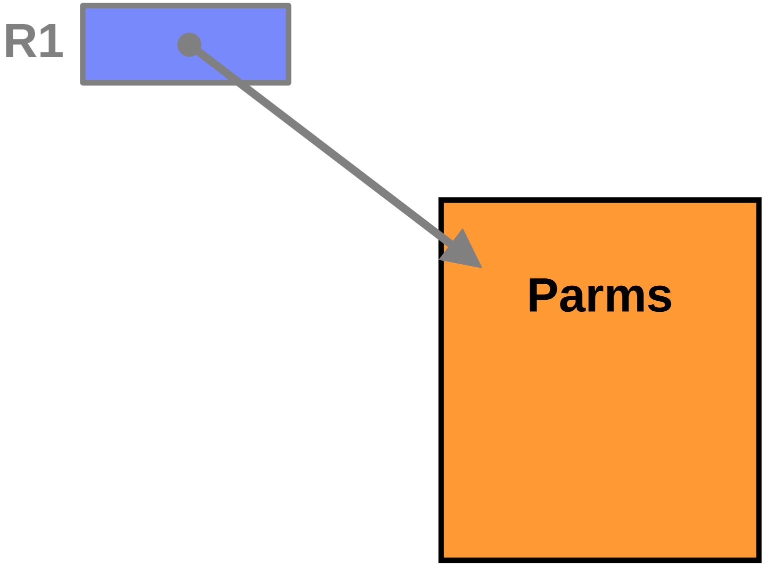
R15 is not trusted either.
PC entry is not a BASR!
This is not the entry point address!

Side-note: R4 IS trusted when used as a latent parm on ETDEF

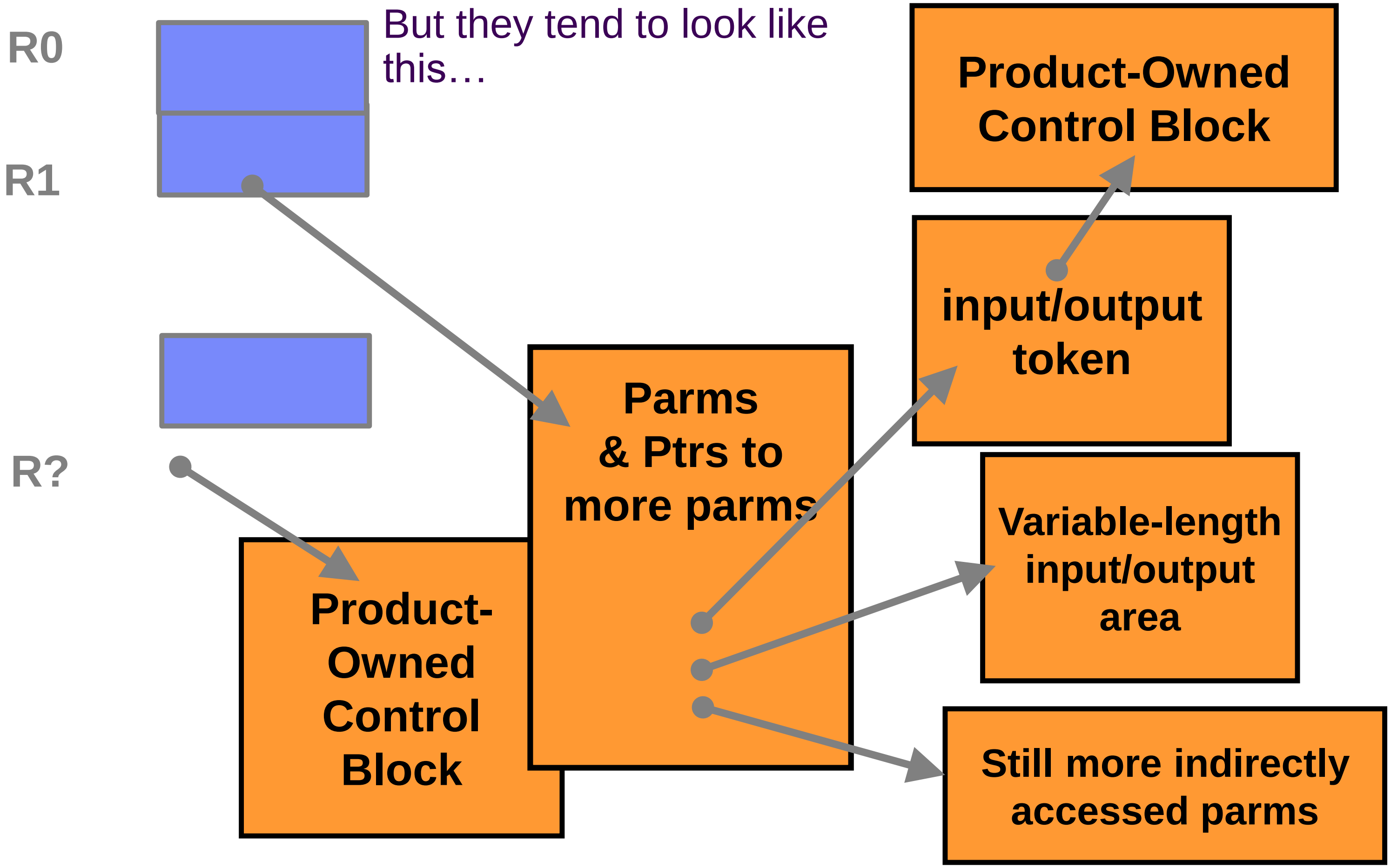
Vulnerability Pattern #3: Untrusted, Indirectly Anchored Parms

Vulnerability Pattern #3:
Untrusted, Indirectly Anchored Parms

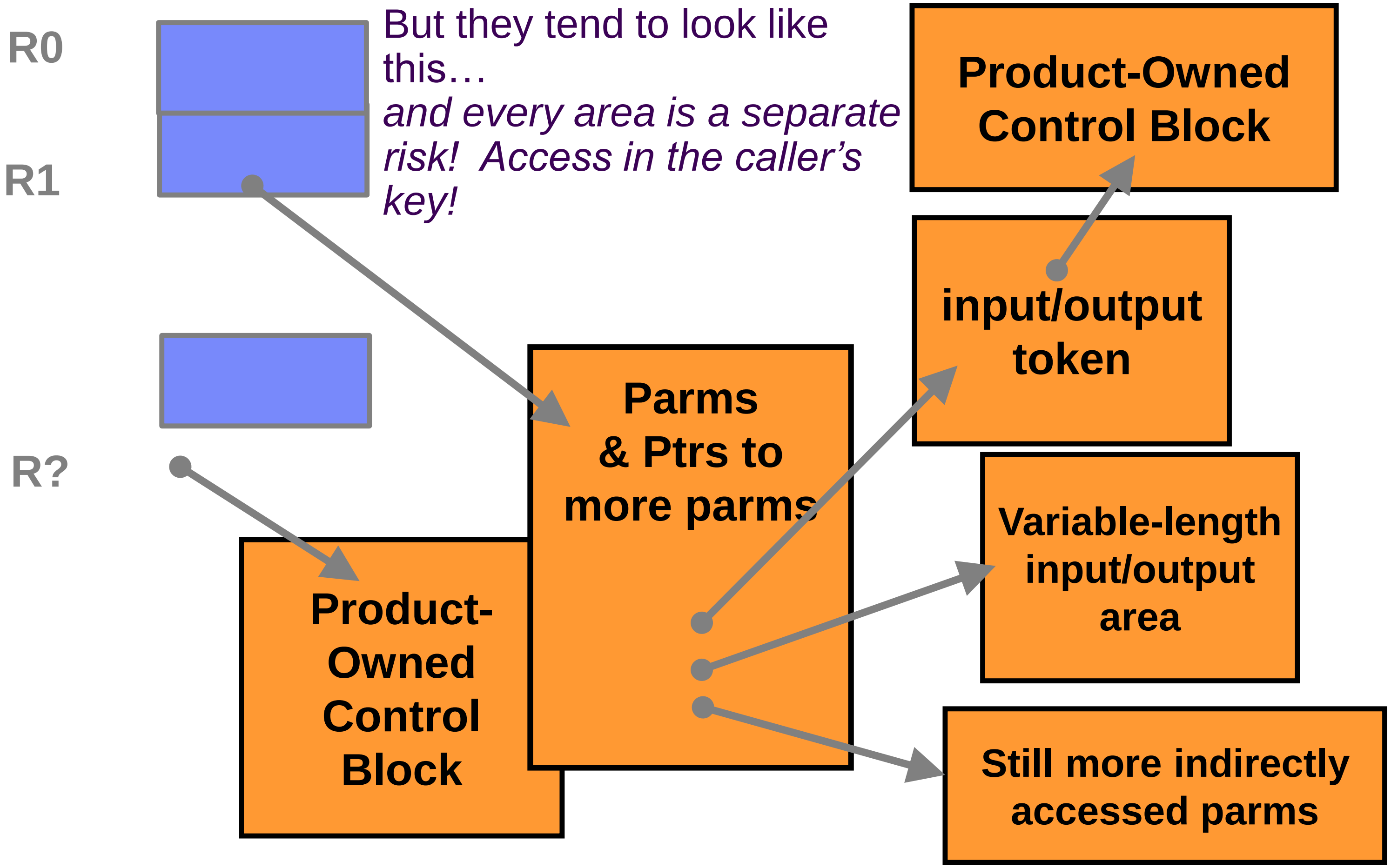
We just described a parameter list format like this…



Vulnerability Pattern #3:
Untrusted, Indirectly Anchored Parms


R0
R1
R?
But they tend to look like this…
Product-Owned Control Block
input/output token
Parms & Ptrs to more parms
Variable-length input/output area
Product-Owned Control Block
Still more indirectly accessed parms

Vulnerability Pattern #3: Untrusted, Indirectly Anchored Parms



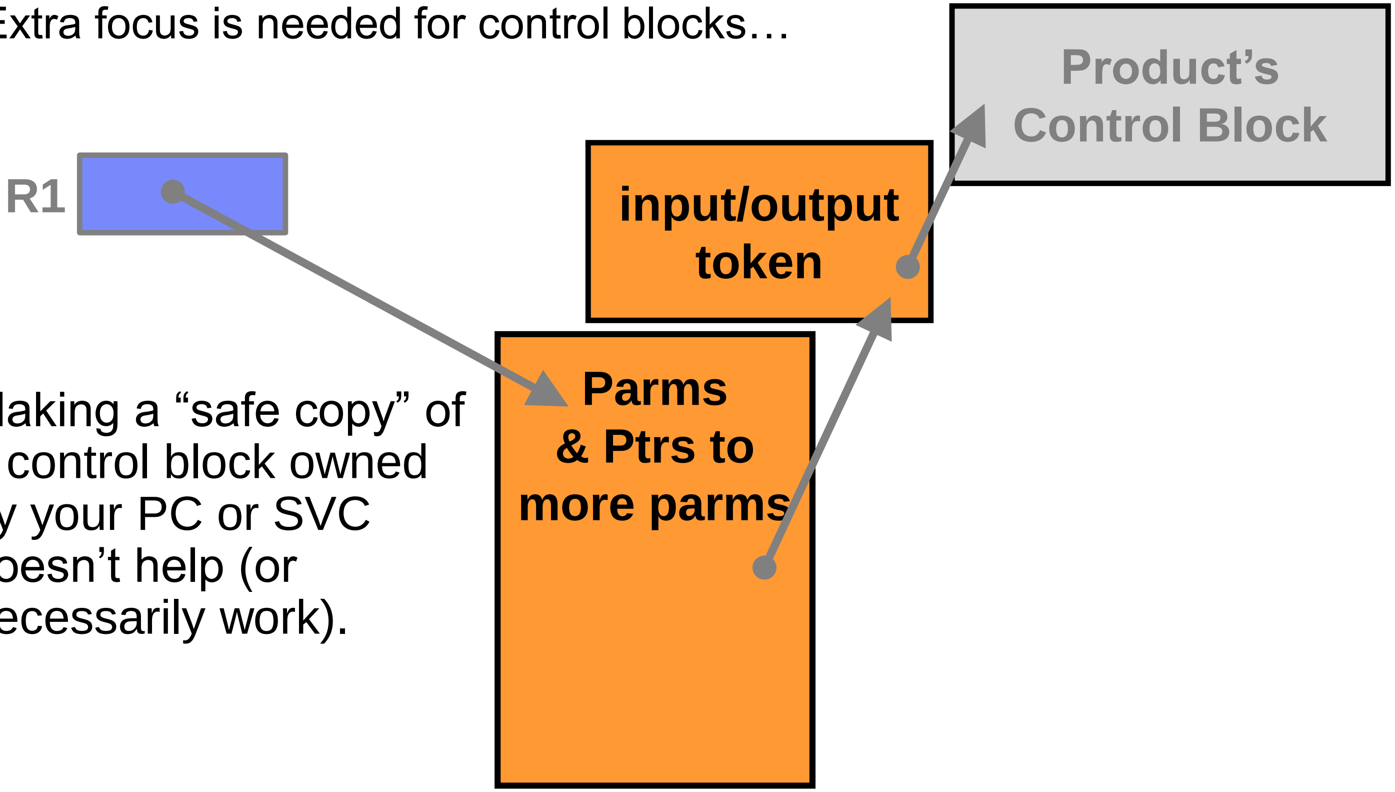
But they tend to look like this…
and every area is a separate risk! Access in the caller's key!

R0
R1
R?
Product-Owned Control Block
input/output token
Parms & Ptrs to more parms
Variable-length input/output area
Product-Owned Control Block
Still more indirectly accessed parms

Vulnerability Pattern #4: Control Block Masquerade

Vulnerability Pattern #4: Control Block Masquerade

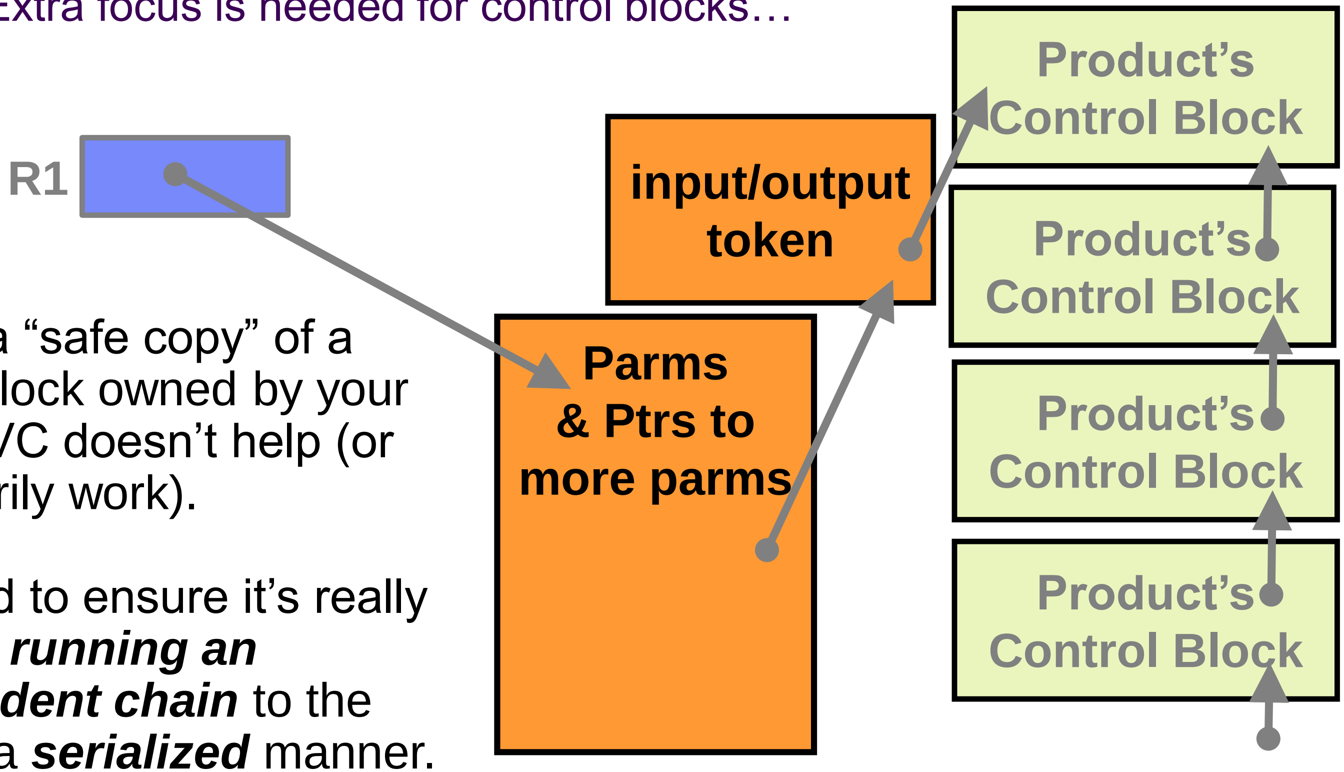
Extra focus is needed for control blocks…



Making a "safe copy" of a control block owned by your PC or SVC doesn't help (or necessarily work).

Vulnerability Pattern #4: Control Block Masquerade

Extra focus is needed for control blocks…



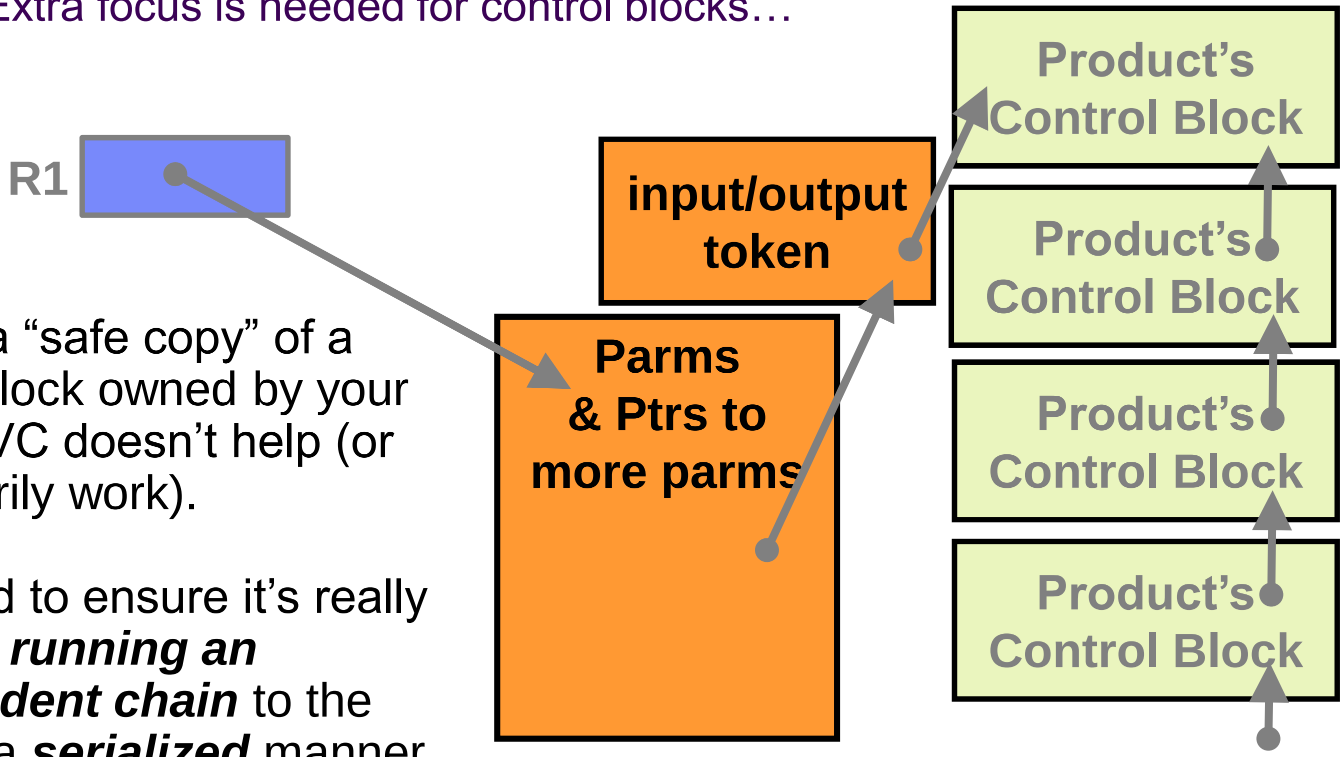
Making a "safe copy" of a control block owned by your PC or SVC doesn't help (or necessarily work).

You need to ensure it's really yours by ***running an independent chain*** to the block in a ***serialized*** manner.

Vulnerability Pattern #4: Control Block Masquerade

Extra focus is needed for control blocks…

Note: Checking the “eye-catcher” and the caller’s key is not an integrity test!



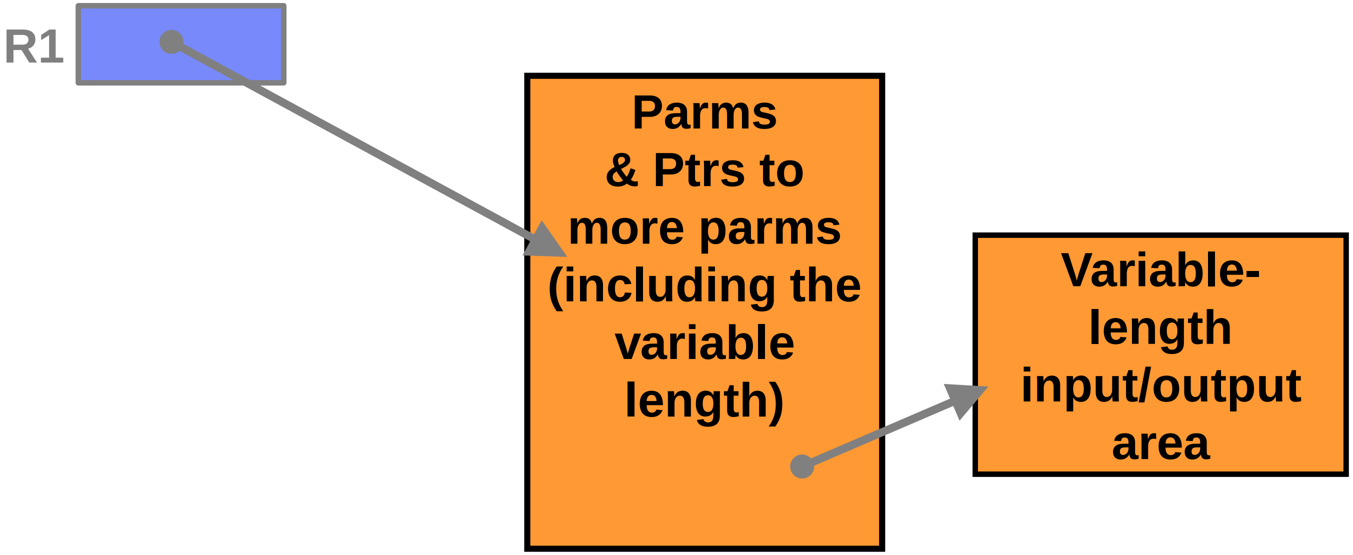
Making a “safe copy” of a control block owned by your PC or SVC doesn’t help (or necessarily work).

You need to ensure it’s really yours by ***running an independent chain*** to the block in a ***serialized*** manner.

Vulnerability Pattern #5:
Buffer Overflow

Vulnerability Pattern #5: Buffer Overflow

Extra focus is also needed for variable length areas

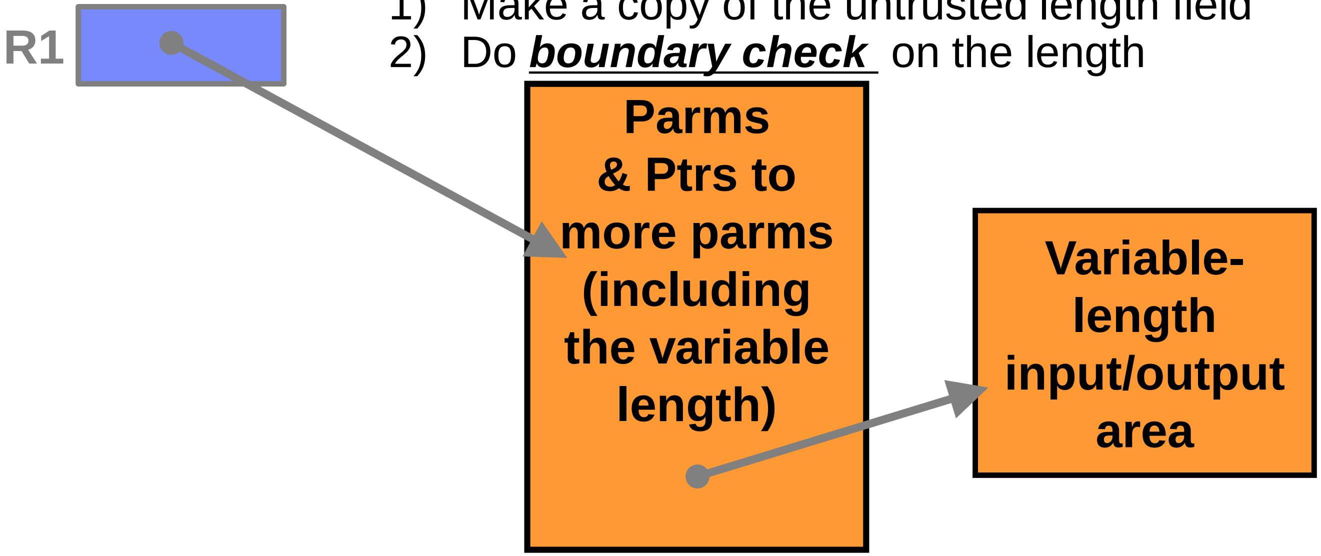


R1
Parms & Ptrs to more parms (including the variable length)
Variable-length input/output area

Vulnerability Pattern #5:
Buffer Overflow

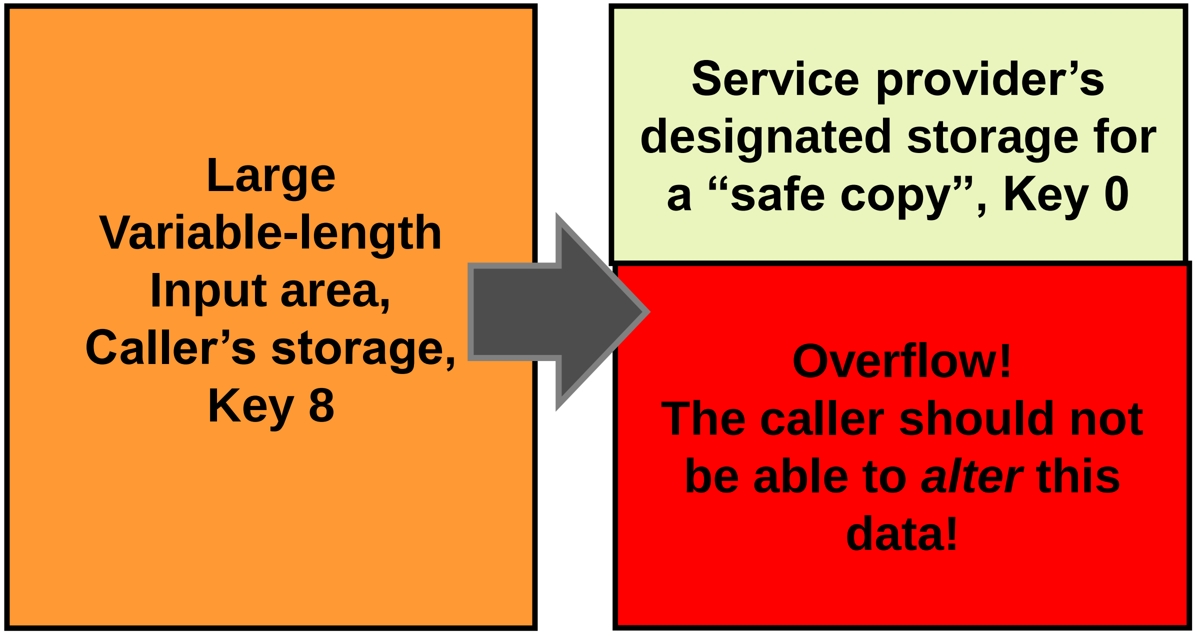
Extra focus is also needed for variable length areas

1) Make a copy of the untrusted length field
2) Do ***boundary check*** on the length


R1
Parms & Ptrs to more parms (including the variable length)
Variable-length input/output area

Vulnerability Pattern #5: Buffer Overflow

Clarifying the overflow from the caller’s input area



Large Variable-length Input area, Caller’s storage, Key 8

Service provider’s designated storage for a “safe copy”, Key 0

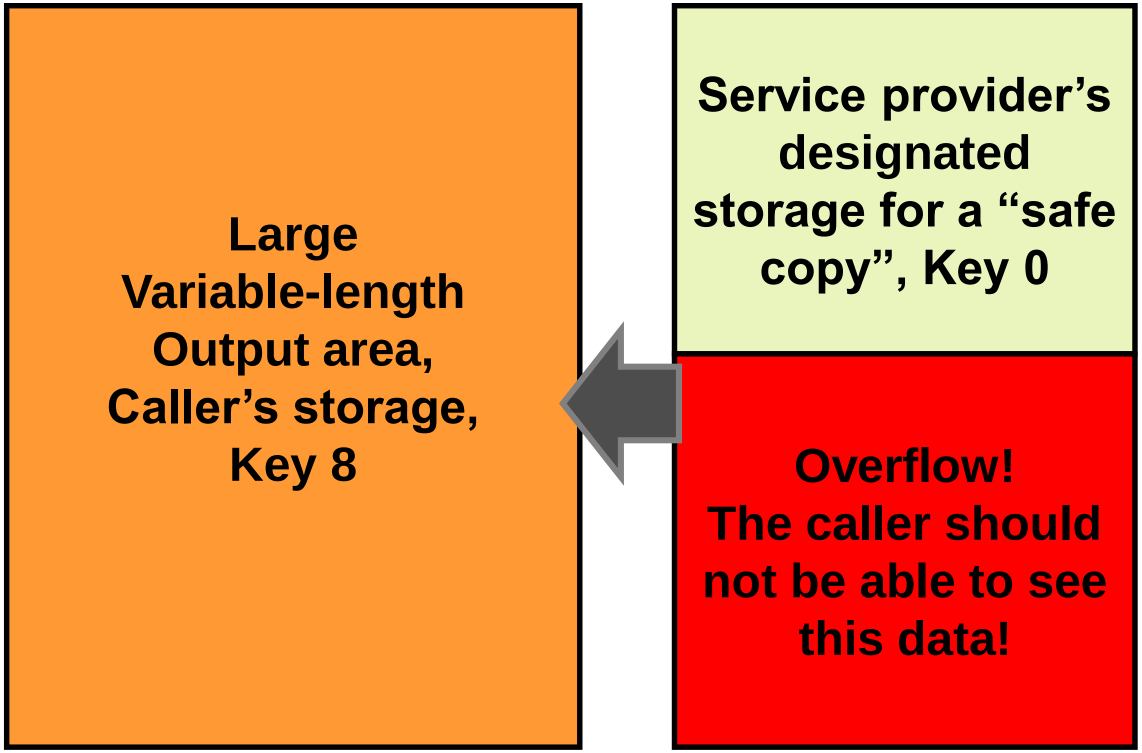
Overflow! The caller should not be able to *alter* this data!

Copying with the caller’s key does not protect against overflow!

Length boundary must be checked!

Vulnerability Pattern #5: Buffer Overflow

Clarifying the overflow **into** the caller’s output area



Large Variable-length Output area, Caller’s storage, Key 8

Service provider’s designated storage for a “safe copy”, Key 0

Overflow! The caller should not be able to see this data!

Copying with the caller’s key does not protect against overflow!

Length boundary must be checked!

Vulnerability Pattern #6:
User Key Common Storage Areas

Vulnerability Pattern #6:
User Key Common Storage

- Common Storage areas are accessible to all address spaces in a z/OS system
  - CSA/ECSA
  - SQA/ESQA
  - SCOPE=COMMON Data Spaces
- Creating and using ***user key common storage*** allows tampering by any unauthorized user program!

Vulnerability Pattern #6: User Key Common Storage

- Common Storage areas are accessible to all address spaces in a z/OS system
  - CSA/ECSA
  - SQA/ESQA
  - SCOPE=COMMON Data Spaces
- Creating and using ***user key common storage*** allows tampering by any unauthorized user program!

- Can lead to the **complete compromise** of an application, its data and of z/OS via ability to view/modify storage!
- z/OS 2.3 is last release to allow use of this

Vulnerability Pattern #6: 1st Alternative to User Key CSA

Fetch Protected System Key (0-7) Common Storage:

- Obtain/Create *fetch protected system key (0-7)* common storage area(s)
- PC or SVC routine required to read/update the system key common storage for unauthorized user key callers
- *Validation* required to ensure users *only have access* to appropriate section of common storage
- *RACF resources* can be used to provide more granular protection

Vulnerability Pattern #6: 2nd Alternative to User Key CSA

- **Space Switching PC Routine:**
  - Implement a space switching PC routine to a system or subsystem address space
  - *PC routine gathers data* from and returns data to unauthorized user key callers
  - Data is maintained in *system/subsystem address space* storage
  - *Validation* required to ensure callers only have access to read/write appropriate segments of data

Vulnerability Pattern #7:
WEAK SECURITY CONFIGURATION

Vulnerability Pattern #7: Weak Security Configuration

- Some common security configuration weaknesses include:
  - **Unpatched code – Not applying all security/integrity PTFs**
    - IBM System Z Security Portal

https://www-03.ibm.com/systems/z/solutions/security_subintegrity.html

– If you are an IBM z Systems customer (or their authorized representative), follow the steps described on this page to obtain access to the z Systems Security Portal for z Systems Security/Integrity APAR information (currently z/OS and z/VM).

– The z Systems Security Portal is intended to help you stay current with security and system integrity fixes by providing current patch data and associated Common Vulnerability Scoring System (CVSS) ratings for new APARs. Security Notices are also provided to address highly publicized security concerns.

Vulnerability Pattern #7: Weak Security Configuration

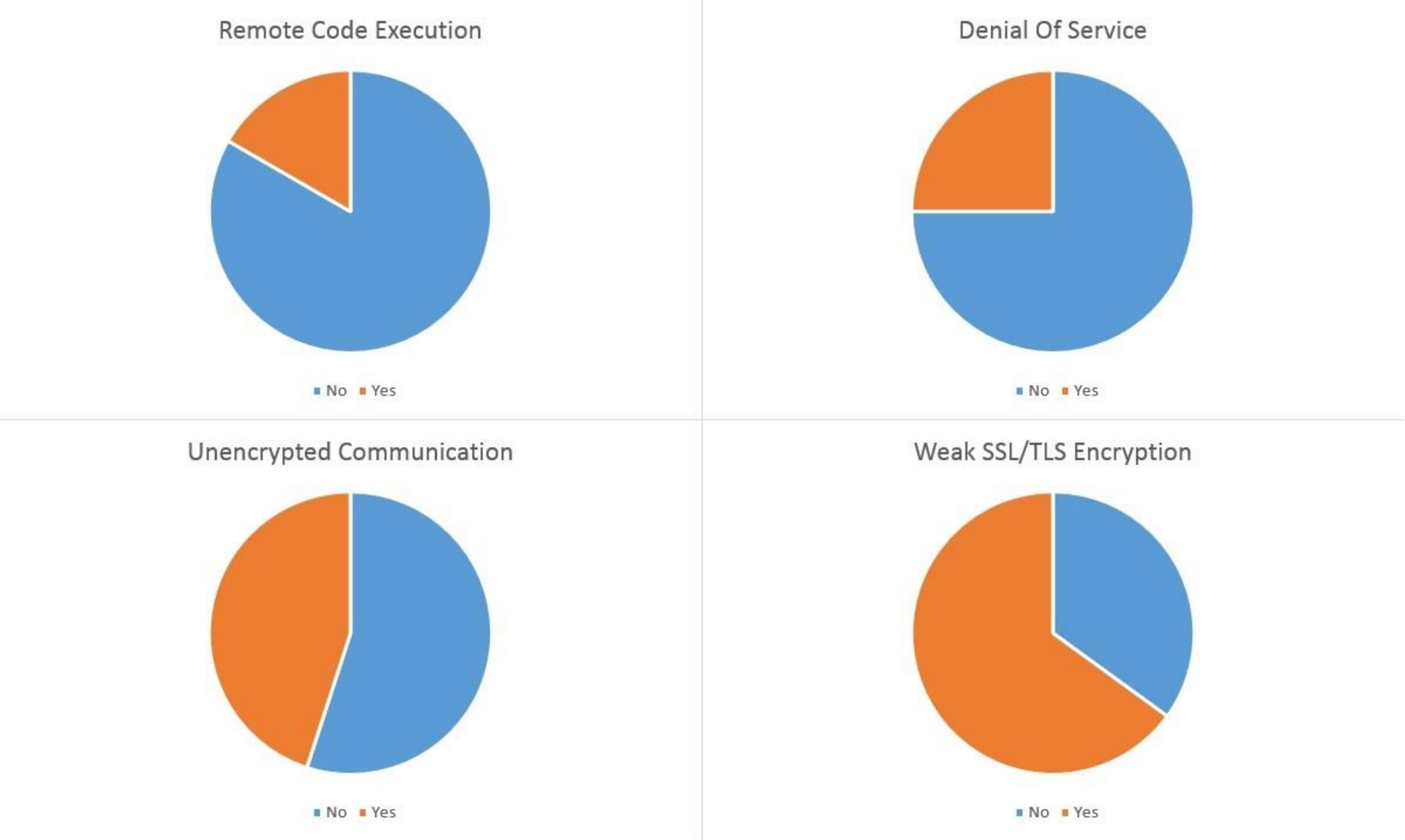
- Some common security configuration weaknesses include:
  - **Privilege escalation – Write access to APF program libraries**
    - Programs in the APF list with UACC(ALTER) or equivalent
  - **Weak SSL/TLS encryption – Enabling weak cipher suites**
    - Forgetting to install needed system security level 3 FMIDs
    - Purposely enabling weak ciphers for compatibility reasons
  - **Information disclosure – Insecure or unencrypted services**
    - Unencrypted services like telnet, rlogin, FTP, HTTP, RSH
    - AT-TLS with desktop clients that do not enforce encryption
    - Services that reveal too much information about the system
      - Consider IKJTSOxx LOGON PASSWORDPREPROMPT(ON)

Vulnerability Pattern #7:
Weak Security Configuration

- Some common security configuration weaknesses include:
  - **Denial of service – Programs that could crash your systems**
    - Do not rely on your network firewall as your only protection
    - Open SMTP mail relays or echo servers could be exploited
  - **Trusting authorized tools – Unknown authorized programs**
    - Do you have a magic SVC or PC for getting authorization?
    - Do you have programs from the CBT tape in APF libraries?
  - **Remote code execution – Running unknown programs**
    - Did you know FTP, NJE, and CICS can all be configured so they allow unknown users to submit jobs via TCP/IP?
    - Do you know if your own system is configured for this?

Vulnerability Pattern #7:
Weak Security Configuration

Results from an unnamed z/OS network



Remote Code Execution

Denial Of Service

Unencrypted Communication

Weak SSL/TLS Encryption

Vulnerability Pattern #7:
z/OS Health Checks Were Created To Help

- CICS_JOBSUB_TDQINTRDR
- CICS_JOBSUB_SPOOL
- CICS_CEDA_ACCESS
- CSAPP_FTPD_ANONYMOUS_JES
- CSAPP_MVRSHD_RHOSTS_DATA
- CSAPP_SNMPAGENT_PUBLIC_COMMUNITY
- CSV_APF_EXISTS
- ICSF_KEY_EXPIRATION
- JES_NJE_SECURITY
- RACF_AIM_STAGE
- RACF_AUDIT_CONTROLS
- RACF_BATCHALLRACF
- RACF_CERTIFICATE_EXPIRATION
- RACF_CSFKEYS_ACTIVE
- RACF_CSFSERV_ACTIVE
- RACF_ENCRYPTION_ALGORITHM
- RACF_FACILITY_ACTIVE
- RACF_GRS_RNL
- RACF_IBMUSER_REVOKED
- RACF_ICHAUTAB_NONLPA
- RACF_JESJOBS_ACTIVE
- RACF_JESSPOOL_ACTIVE
- RACF_OPERCMDS_ACTIVE
- RACF_PASSWORD_CONTROLS
- RACF_RRSF_RESOURCES
- RACF_SENSITIVE_RESOURCES
- RACF_TAPEVOL_ACTIVE
- RACF_TEMPDSN_ACTIVE
- RACF_TSOAUTH_ACTIVE
- RACF_UNIX_ID
- RACF_UNIXPRIV_ACTIVE
- SDSF_CLASS_SDSF_ACTIVE
- USS_INETD_UNSECURE_SERVICES
- USS_SUPERUSER
- VSM_ALLOWUSERKEYCSA

Vulnerability Pattern #7: 1st z/OS Health Check Example

- RACF_SENSITIVE_RESOURCES – EXCEPTION-HIGH

- IRRH204E The RACF_SENSITIVE_RESOURCES check has found one or more potential errors in the security controls on this system.

- Report this problem to the system security administrator and the system auditor.

- Examine the report that was produced by the RACF check. Any resource that has an "E" in the "S" (Status) column has excessive authority allowed to the resource. This authority might come from a universal access (UACC) or ID(*) access list entry that is too permissive, or the profile is in WARNING mode. If there is no profile, PROTECTALL(FAIL) is not in effect.

Vulnerability Pattern #7: 2nd z/OS Health Check Example

- JES_NJE_SECURITY – EXCEPTION-MEDIUM
- IAZH122E *nodecount* nodes that can be or are currently connected have no password and have specified SECSIGNON=NO (or don't support secure signon)
- All nodes that can connect to your system should have their identity verified before they join your network. Even if the node is not locally a trusted node, it could be trusted by other nodes or be used to submit jobs claiming to be from trusted nodes. Passwords (either defined in JES or preferably using secure signon if an NJE over TCP/IP connection) are the standard method to perform this authentication.

Vulnerability Pattern #7: 3rd z/OS Health Check Example

– CSAPP_FTPD_ANONYMOUS_JES – EXCEPTION-MEDIUM

– ISTH021E One or more FTP servers allow anonymous users to submit jobs.

– Check CSAPP_FTPD_ANONYMOUS_JES determined that one or more FTP servers allow anonymous users to submit jobs.

– IBM suggests that ANONYMOUSLEVEL be set to 3 and ANONYMOUSFILETYPEJES be set to FALSE when ANONYMOUS is configured on the FTP server. Specifying ANONYMOUSLEVEL less than 3 or ANONYMOUSFILETYPEJES TRUE allows anonymous users to submit jobs.

Recap:
Vulnerability Patterns for z/OS

1. The Unintentionally Authorized PC
2. Untrusted Parms, Untrusted Regs
3. Untrusted, Indirectly Anchored Parms
4. Control Block Masquerade
5. Buffer Overflow
6. User Key Common Storage
7. Weak Security Configuration

Backup
(3rd Alternative to User Key CSA)

- **User Key Shared Memory Area:**
  - Create user key (>=8) shared memory area(s) via *IARVSERV (31-bit) or IARV64 (64-bit)*
    - Source for 31-bit in system address space or data space
  - *PC or SVC routine required to give unauthorized user key callers access* to shared memory area via IARV64 or IARVSERV calls
  - *Validation required to ensure callers are trusted or authorized* to use shared memory area
    - Each validated caller will be able to read/write an entire shared memory area
  - *RACF resource can be used* to provide protection

Backup
(4th Alternative to User Key CSA)

- **User Key z/OS UNIX Shared Memory Segment:**
  - User key (8 or 9) z/OS UNIX shared memory segment(s) created by a UNIX privileged address space
  - *Each user address space attaches* to a shared memory segment
  - Only users that are *permitted access via z/OS UNIX* permissions can attach to a shared memory segment
  - Permitted users must be considered *trusted for entire segment* access
  - Each permitted user is able to read and/or write to an entire shared memory segment

Backup
(5th Alternative to User Key CSA)

- **User Key SCOPE=ALL Data Space:**
  - Alternative to a SCOPE=COMMON data space
  - Create a user key SCOPE=ALL data space associated with a system/subsystem address space
  - *PC or SVC routine required to ALESERV* data space to give access to unauthorized user key callers
  - Validation required to *ensure callers are trusted or authorized* to use data space
    - Each validated caller will be able to read/write entire data space
  - *RACF resource can be used* to provide protection

Backup
4th z/OS Health Check Example

- CICS_CEDA_ACCESS – EXCEPTION-MEDIUM
- DFHH0001E The CEDA transaction is accessible to unauthenticated users.
- The IBM supplied transaction CEDA is accessible to the default user or CICS security is turned off.
- This means anyone who can connect to the IP address and port number of one of the CICS regions listed below can change the configuration of CICS.
- The regions listed below have a RC/RSN with more specific information about why the region failed the check:

Backup
5th z/OS Health Check Example

- CICS_JOBSUB_SPOOL – EXCEPTION-MEDIUM

- DFHH0002E The spool is accessible to unauthenticated users.

- The SPOOL=YES is defined and the IBM supplied transaction CECI is accessible to the default user or CICS security is turned off.

- This means anyone who can connect to the IP address and port number of one of the CICS regions listed below can submit jobs to run on the z/OS system remotely without authentication.

- The regions listed below have a RC/RSN with more specific information about why the region failed the check:

Backup
6th z/OS Health Check Example

- CICS_JOBSUB_TDQINTRDR – EXCEPTION-MEDIUM

- DFHH0003E A TDQ defined to the internal reader *program* is accessible to unauthenticated users.

- At least one TD QUEUE defined to the internal reader and the IBM supplied transaction CECI are accessible to the default user or CICS security is turned off.

- This means anyone who can connect to the IP address and port number of one of the CICS regions listed below can submit jobs to run on the z/OS system remotely without authentication.

- The regions listed below have a RC/RSN with more specific information about why the region failed the check: