Coding for System Integrity in z/OS for ISVs

Karl Schmitz
z/OS System Integrity Competency Center
kdsch@us.ibm.com

Trademarks

Coding for System Integrity in z/OS

- ***Definition***
- ***Guidelines***
- ***Examples***

System Integrity Definition

- Announced with MVS in 1973
- 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

Types of Authorization

- ***PSW Key 0-7***
- ***PKM 0-7***
- ***Supervisor State***
- ***APF Authorization***

Ways of becoming authorized

- ***SVC routines***
- ***PC routines***
- ***APF authorized programs***
- ***Program Properties Table***

z/OS System Integrity Guidelines

- *Creating predictable interfaces*
- *Dealing with user supplied storage*
- *Dealing with user supplied control blocks*
- *Dealing with user supplied values*
- *Protecting data*
- *Authorization requirements*
- *Serializing Resources*

Predictable Interfaces

- ***Interfaces between unauthorized and authorized programs must behave predictably***
  - *Applies to intended and unintended interfaces*
  - *Security checking must be performed in authorized code*
  - *Only load modules intended to run as authorized jobsteps or commands should be linked AC(1)*

User Supplied Storage

- ***Access caller supplied storage in the key of the caller***
- ***For example, use MVCSK or MVCDK***

User Supplied Control Blocks

- ***Verify system control blocks through trusted pointers in system key storage***
- ***Serialize as appropriate***

User Supplied Values

- ***Verify that values are legitimate***
- ***Beware of values that might change after verification***

Protecting Data

- ***Authorized programs must protect data from unauthorized tampering***
- ***Do not use key 8 common storage***

Authorization Requirements

- Services that bypass security checks must be restricted to authorized callers
- Callers allowed to bypass security checks must provide equivalent controls
- Do not provide services that make unauthorized callers authorized

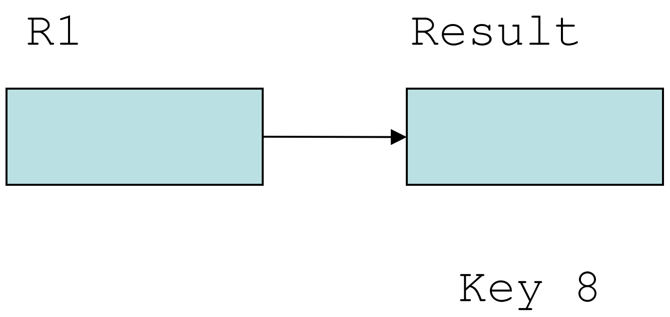
Serializing Resources

- ***Serialize to control multiple access to resources***
- ***Serialization technique must be one restricted to authorized programs***

System Integrity Exposures Examples

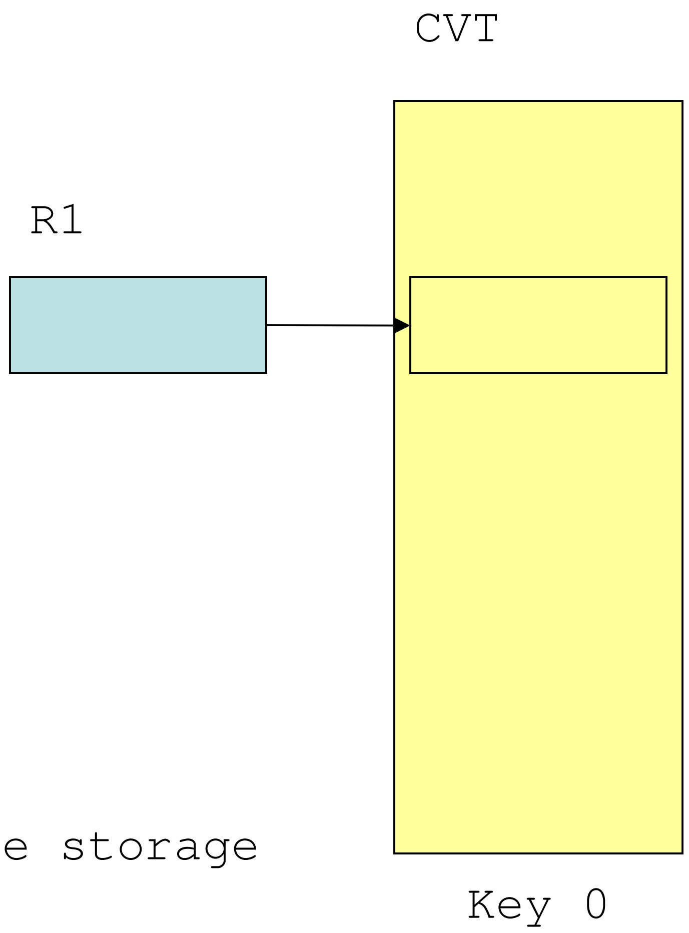
Example 1: SVC (Key 0, Supervisor State)

```
.
.
MODESET EXTKEY=TCB . .
OC        0(4,R1),0(R1)
MODESET EXTKEY=ZERO . .
.
.
ST        R5,0(R1)
.
.
```



Example 1

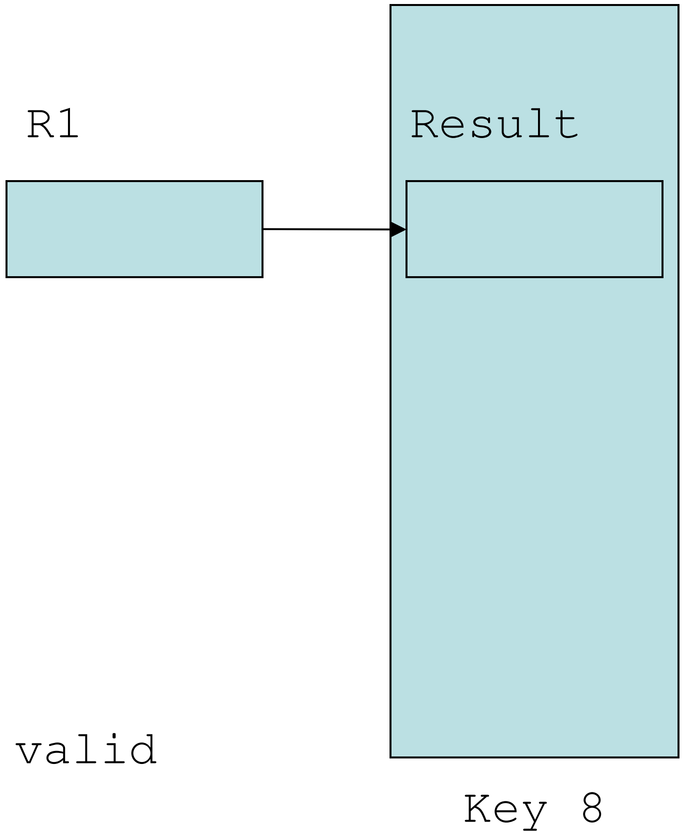
```
.
.
MODESET EXTKEY=TCB . .
OC          0(4,R1),0(R1)
MODESET EXTKEY=ZERO . .
.
.
ST          R5,0(R1)
.
.
```



Attempts to validate storage

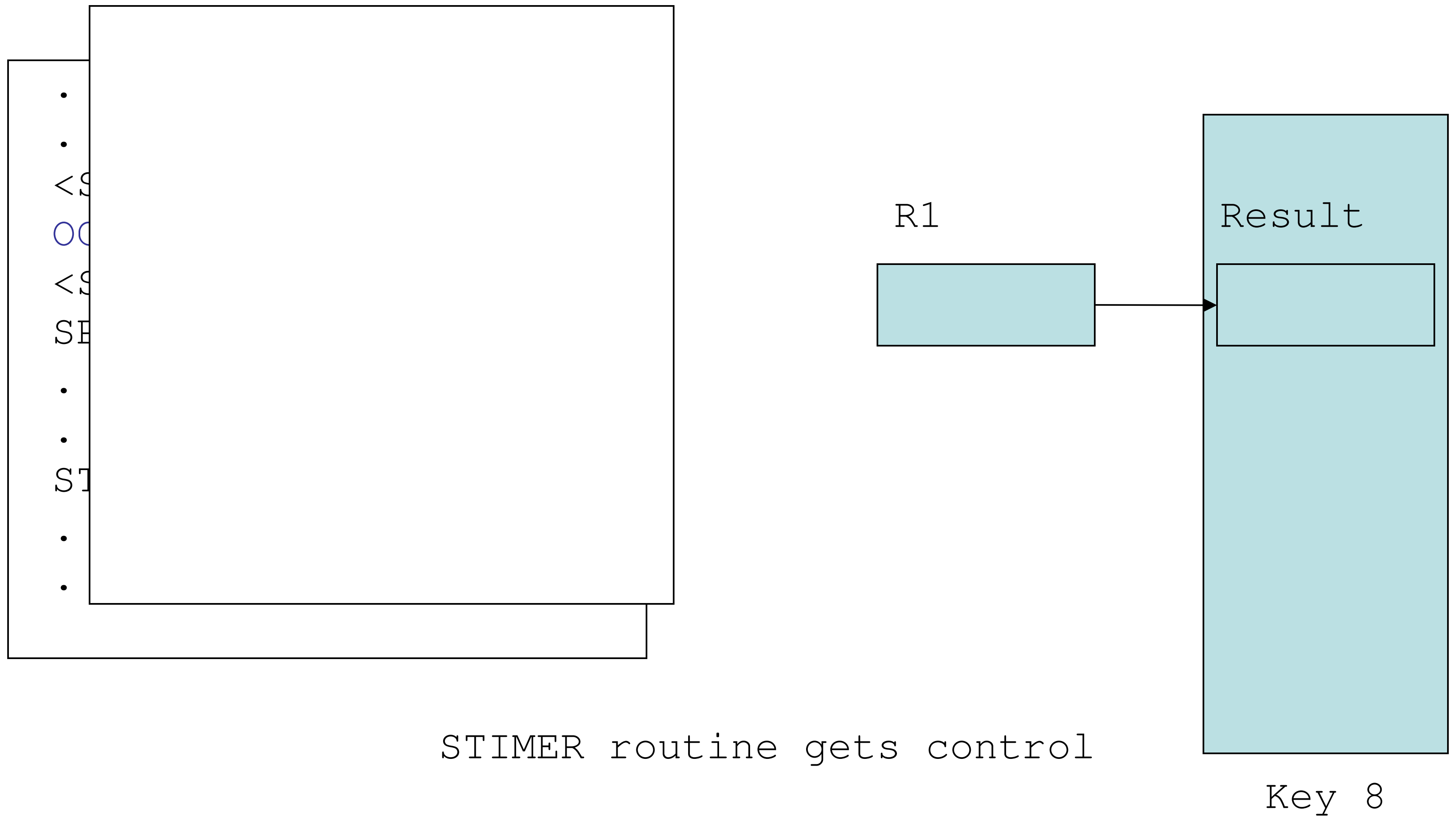
Example 1

```
.
.
MODESET EXTKEY=TCB . .
OC          0(4,R1),0(R1)
MODESET EXTKEY=ZERO . .
.
.
ST          R5,0(R1)
.
.
```



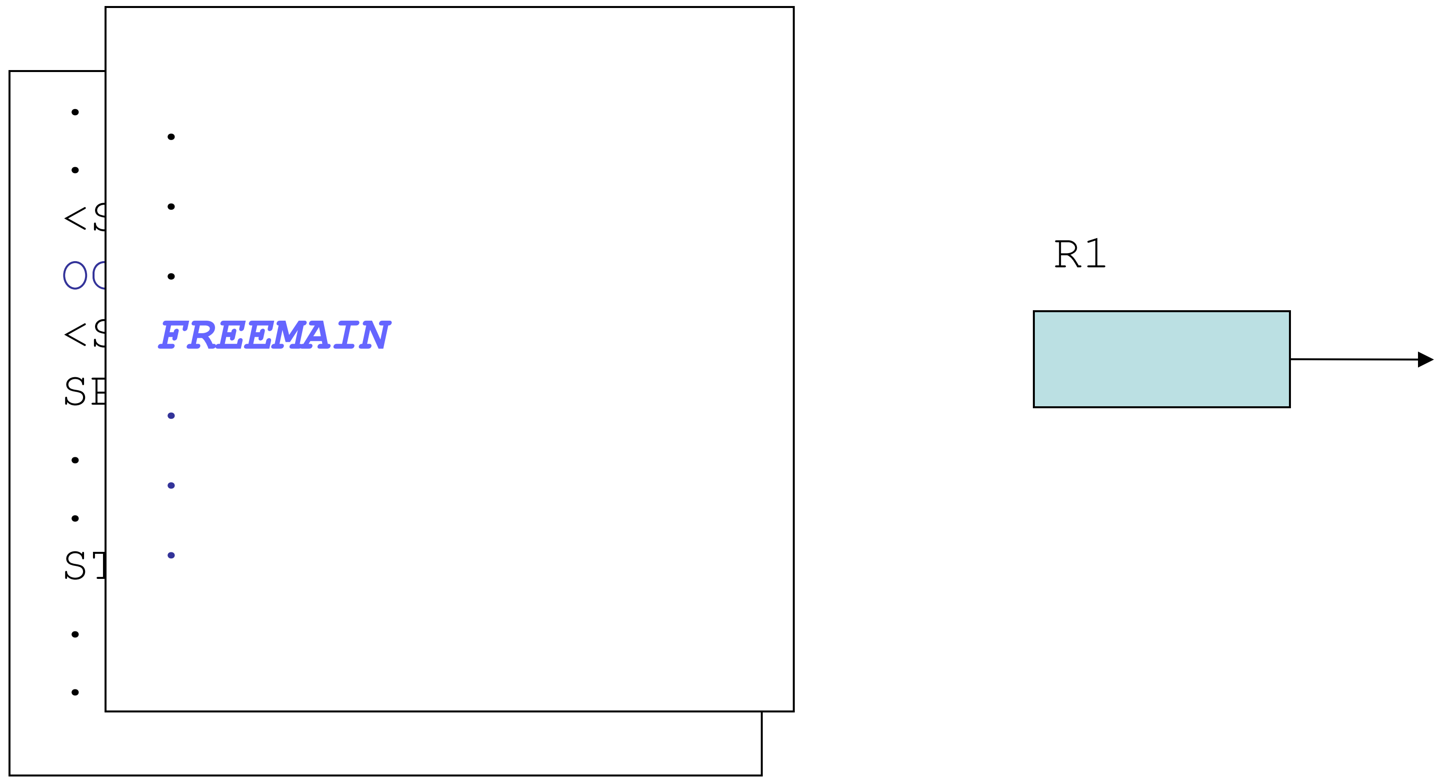
Storage appears valid

Example 1



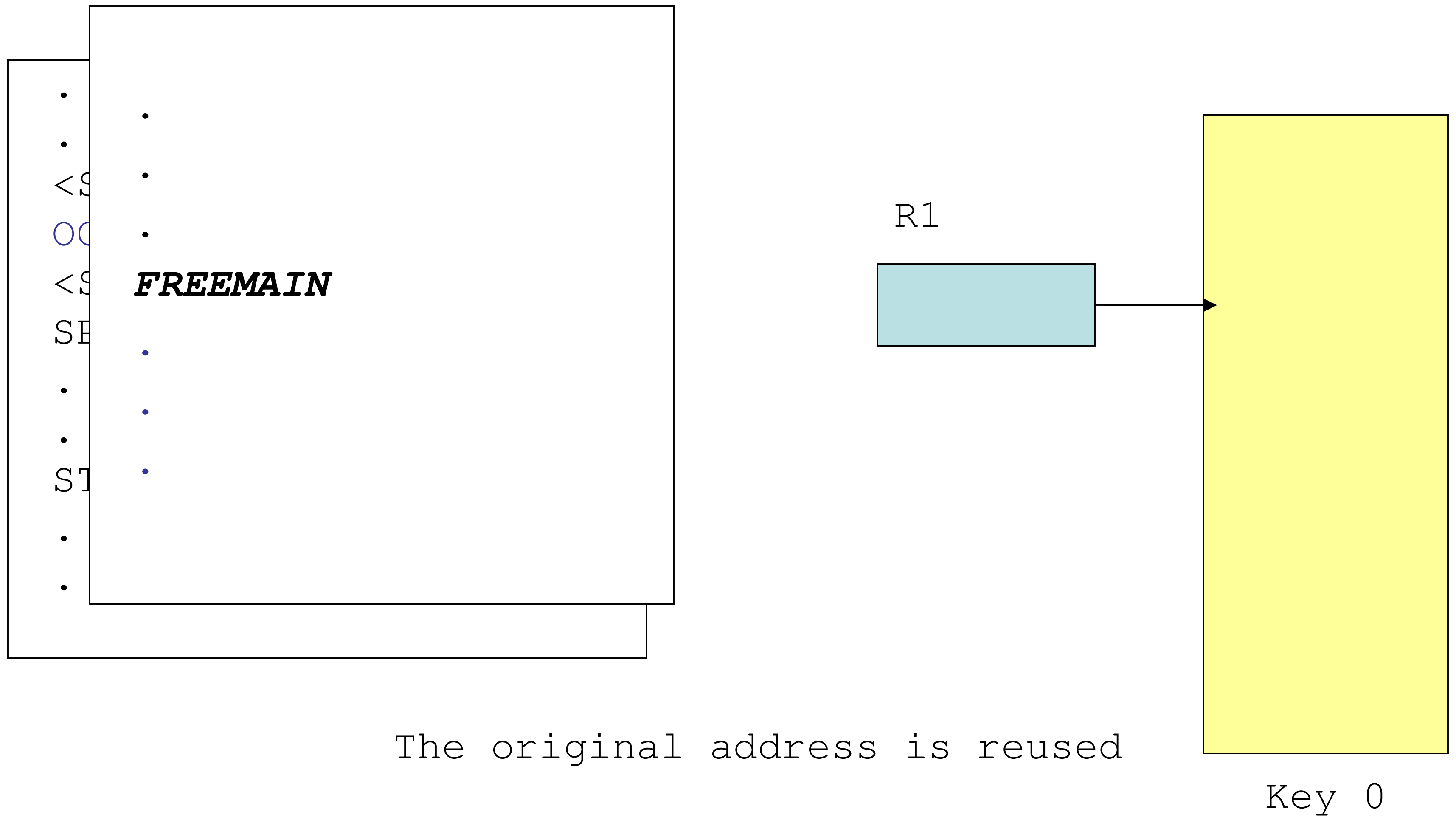
STIMER routine gets control

Example 1



The key 8 storage is freed

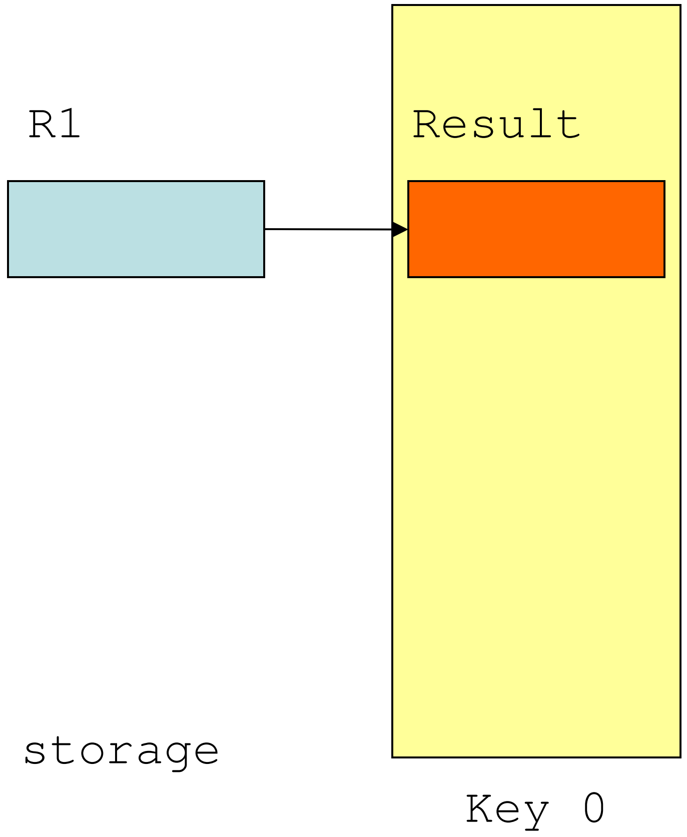
Example 1



The original address is reused

Example 1

```
.
.
MODESET EXTKEY=TCB . .
OC          0(4,R1),0(R1)
MODESET EXTKEY=ZERO . .
.
.
ST          R5,0(R1)
.
.
```



Overwrites key 0 storage

Example 1

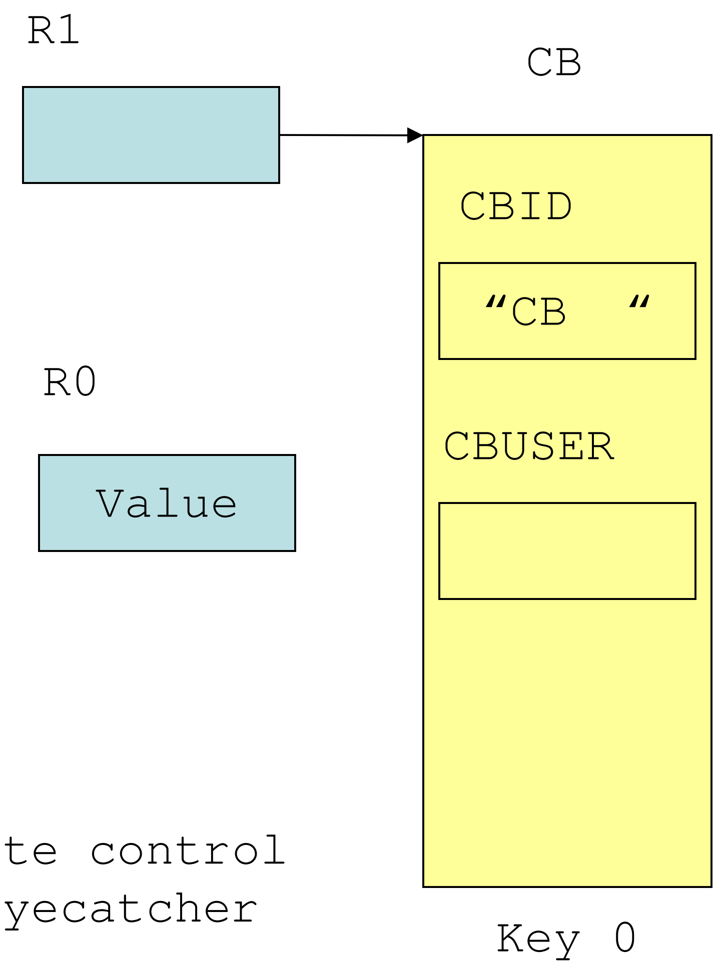
- *Violates guidelines for dealing with user supplied storage:*
  - *Access caller supplied storage in the key of the caller*
  - *For example, use MVCSK or MVCDK*

Example 2: SVC (Key 0, Supervisor State)

- *Imaginary set of services*
  - *CBINIT – creates control block and returns address*
  - *CBSET – accepts control block address*
  - *CBTERM – accepts control block address and frees control block*

Example 2

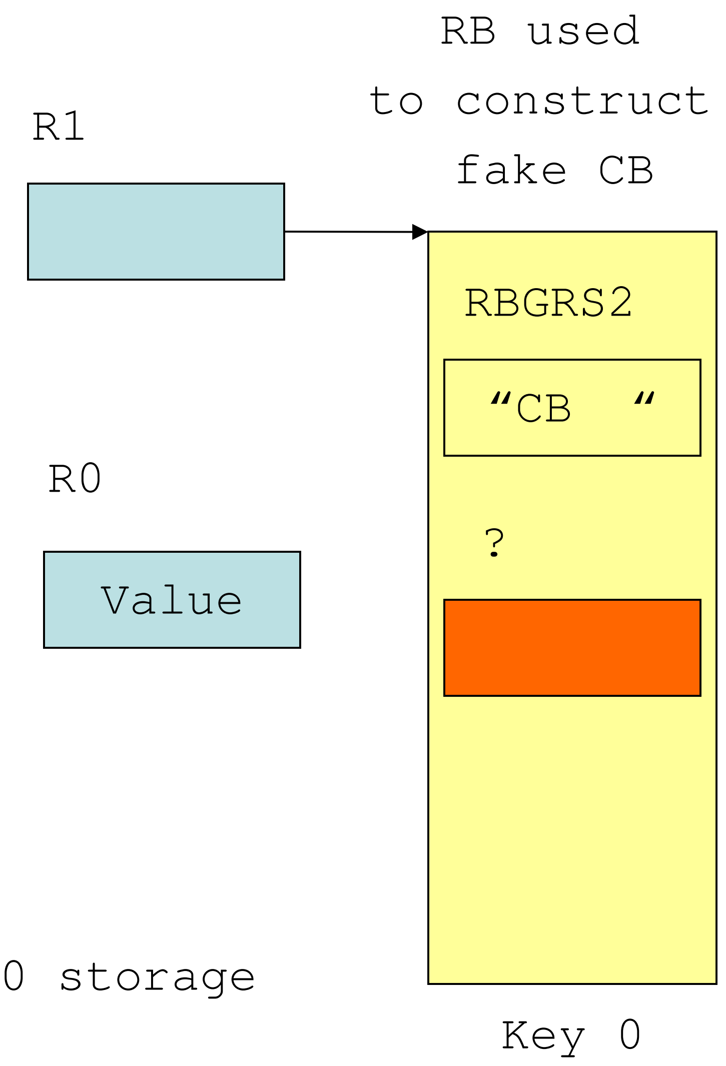
```
.
USING   CB,R1
SR      R5,R5
IVSK    R5,R1
LTR     R5,R5
BNZ     ERROR
CLC     =C'CB  ',CBID
BNE     ERROR
ST      R0,CBUSER
.
```



Attempts to validate control blocks's key and eyecatcher

Example 2

```
.
USING   CB,R1
SR      R5,R5
IVSK    R5,R1
LTR     R5,R5
BNZ     ERROR
CLC     =C'CB   ',CBID
BNE     ERROR
ST      R0,CBUSER
.
```



Overwrites key 0 storage

Example 2

- ***Violates guideline for dealing with user supplied control blocks:***
  - ***Verify system control blocks through trusted pointers in system key storage***
  - ***Treat unverified control blocks as user supplied storage***

Example 3: SVC (Key 0, Supervisor State)

```
IGC00ATH     CSECT                         Bad Auth SVC
             BALR        12,0
             USING       *,12
             L        2,28(5)              Caller's RB
* Resume address < Beginning of PLPA
             CLC         21(3,2),361(3)  Is caller in LPA?
             BL       RETURN
             L        2,180(4)             JSCB
* R0 != 1 request auth off
             BCT      0,AUTHOFF
AUTHON       OI    236(2),X'01'    Set JSCBAUTH
             B           RETURN
AUTHOFF      NI          236(2),X'FE'   Clear AUTH
RETURN       BR          14
             END         IGC00ATH
```

Example 3

- ***Many ways to misuse this SVC***
- ***Violates guidelines for dealing with authorization requirements:***
  - ***Services that bypass security checks must be restricted to authorized callers***
  - ***Callers allowed to bypass security checks must provide equivalent controls***
  - ***Do not provide services that make unauthorized callers authorized***

Example 4: APF authorized Unix program (Key 8, Problem State, APF authorized)

```
main(int argc, char * argv[])

char    valueBuffer[100];
strcpy(valueBuffer, argv[1]);
```

Example 4

- ***Classic buffer overflow vulnerability since length of input can be greater than 100***
- ***Violates guidelines for dealing with user supplied values:***
  - ***Verify that values are legitimate (in this case, length of argument string.***

Conclusion

- ***The security of z/OS requires attention to detail***
- ***Developers of authorized programs should follow the guidelines described in this presentation***