RSAConference2022
San Francisco & Digital | June 6 – 9

TRANSFORM

SESSION ID: **SAT-R02**

Uncovering “BadAlloc” Memory Vulnerabilities in Millions of IoT Devices

Omri Ben-Bassat
Security Researcher
Section 52, Microsoft Defender for IoT, Microsoft.

Tamir Ariel
Security Researcher
Section 52, Microsoft Defender for IoT, Microsoft.

Disclaimer

Presentations are intended for educational purposes only and do not replace independent professional judgment. Statements of fact and opinions expressed are those of the presenters individually and, unless expressly stated to the contrary, are not the opinion or position of RSA Conference LLC or any other co-sponsors. RSA Conference does not endorse or approve, and assumes no responsibility for, the content, accuracy or completeness of the information presented.

Attendees should note that sessions may be audio- or video-recorded and may be published in various media, including print, audio and video formats without further notice. The presentation template and any media capture are subject to copyright protection.

Index

- Intro
- Quick Reminder – Integer Overflows
- Memory Allocator 101
- Affected Products
- Notable Examples
- Technical Analysis Texas Instruments “SimpleLink” SDK
- Exploitation SimpleLink POC
- Demo
- Mitigation techniques
- Q&A

RSAConference2022

Quick Reminder

Integer Overflows

Summation

$$8 + 8 = ??$$

Summation

$$8 + 8 = 88$$

Summation

8 + 8 = 88

Summation

8 + 8 = 16

Summation

$$8 + 8 = 16$$

$$4{,}294{,}967{,}295(2\text{^}32-1) + 8 = 7$$

(on 32-bit systems)

Multiplication

$$2 * 2 = 4$$

$$2{,}147{,}483{,}649(2^{31}+1) * 2 = 2$$

(on 32-bit systems)

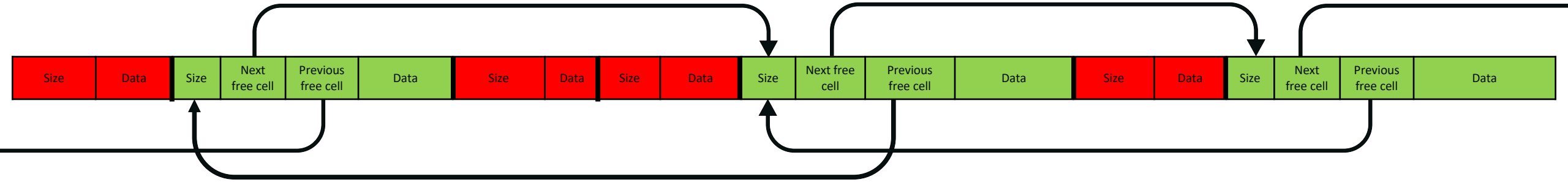
RSAConference2022

Memory Allocator

101

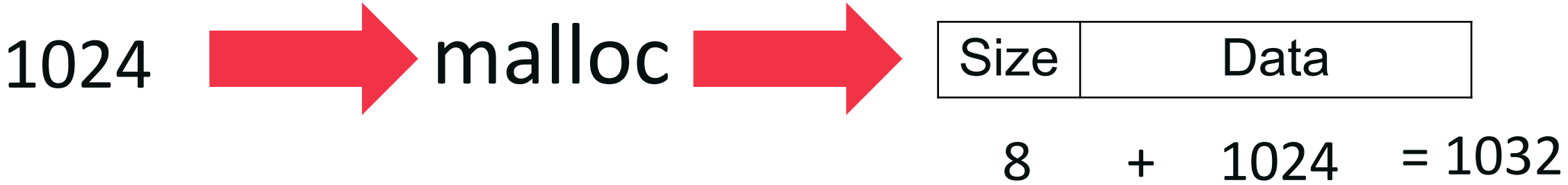
Heap Layout

- Free memory is managed by the allocator using a single/double linked list of free blocks.



Allocated Free

Calculating total block size



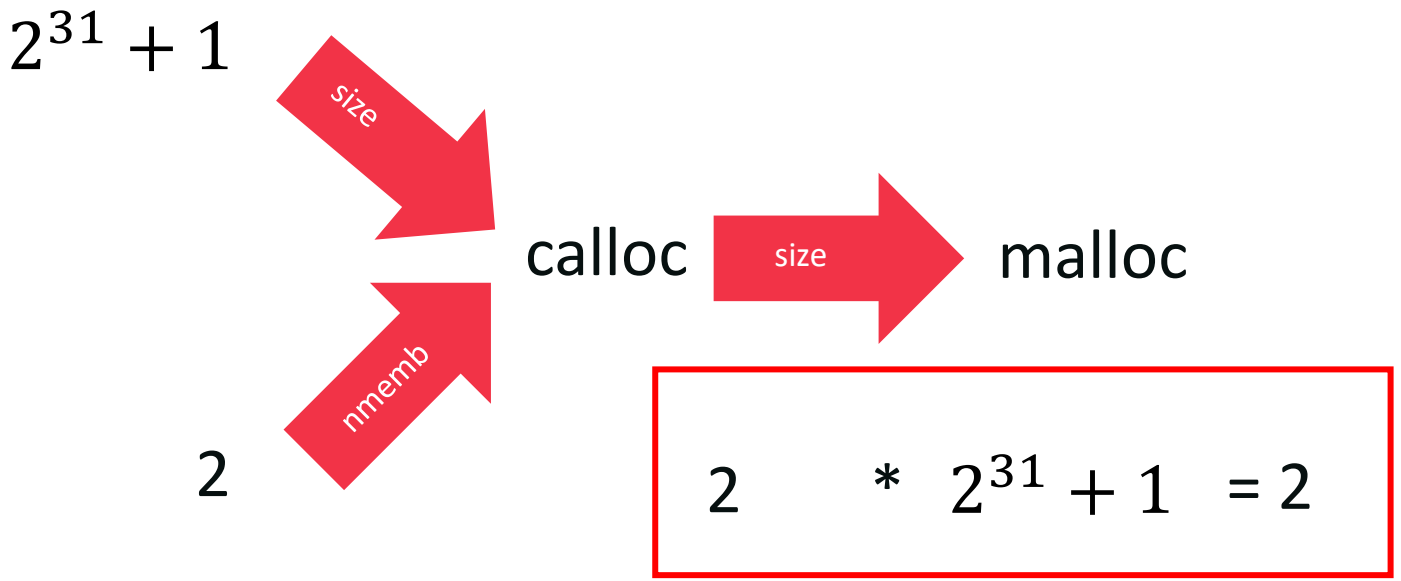
What will happen if I ask for a very large chuck of memory?

$4{,}294{,}967{,}295$
$(2^{32} - 1)$

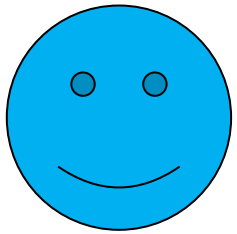
Size	Data

$8 + 2^{32} - 1 = 7$

calloc



Good Alloc

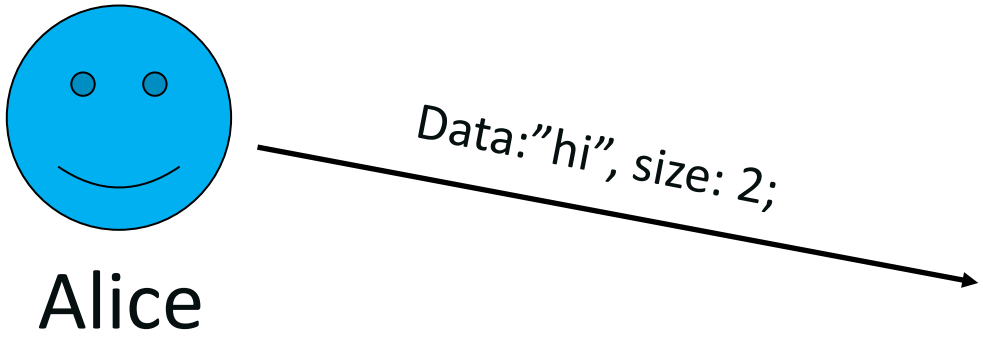


Alice

Server

```
Read_user_data(user_data, size)
        …
            Buf = malloc(size)
            If (buf != NULL)
                        memcpy(buf, user_data, size)
                        return “ok, thank you!”
            else
                        return “sorry too much data”
        …
```

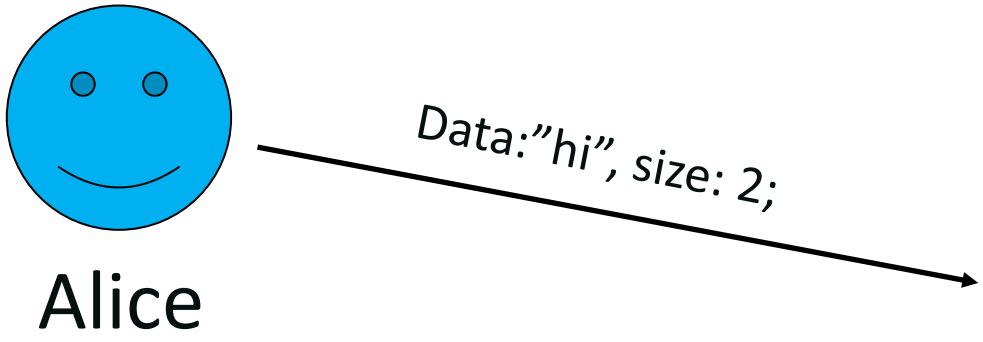
Good Alloc



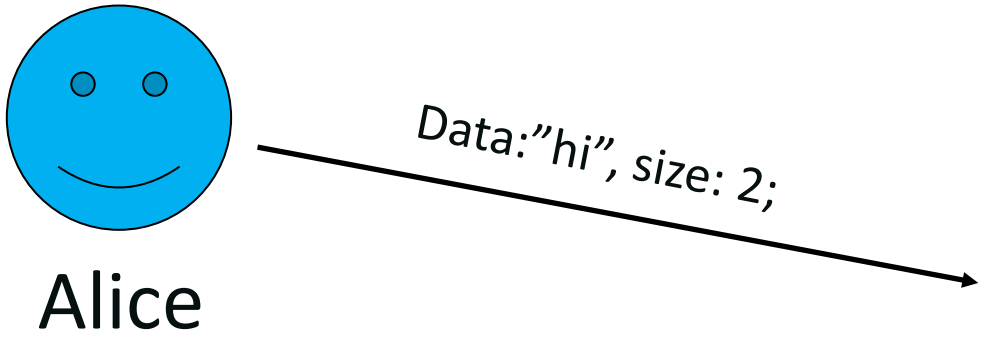
Server

```
Read_user_data(user_data, size)
        …
            Buf = malloc(size)
            If (buf != NULL)
                        memcpy(buf, user_data, size)
                        return "ok, thank you!"
            else
                        return "sorry too much data"
        …
```

Good Alloc

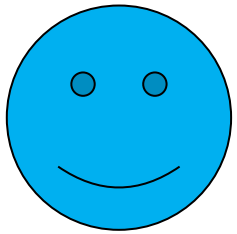

Data:"hi", size: 2;
Alice
Server
Read_user_data(user_data, size)
…
Buf = malloc(**size**)
If (buf != NULL)
memcpy(buf, user_data, **size**)
return "ok, thank you!"
else
return "sorry too much data"
…

Good Alloc




```
Read_user_data(user_data, size)
        …
            Buf = malloc(size)
            If (buf != NULL)
                        memcpy(buf, user_data, size)
                        return "ok, thank you!"
            else
                        return "sorry too much data"
        …
```

Good Alloc



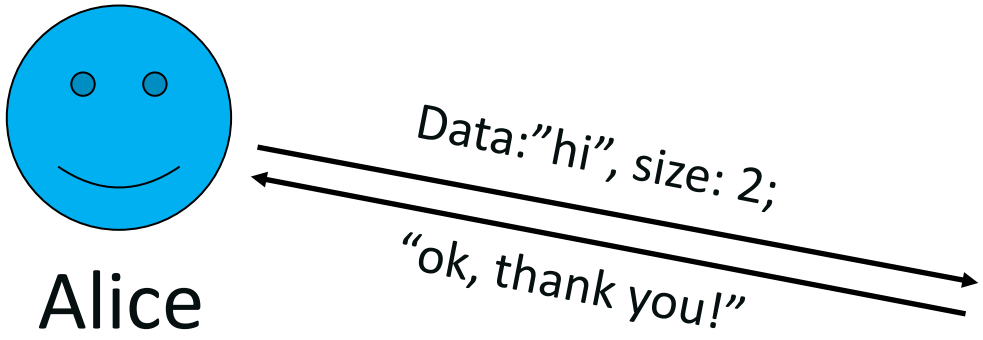
Alice

Data:"hi", size: 2;

Server

```
Read_user_data(user_data, size)
        …
            Buf = malloc(size)
            If (buf != NULL)
                        memcpy(buf, user_data, size)
                        return "ok, thank you!"
            else
                        return "sorry too much data"
        …
```

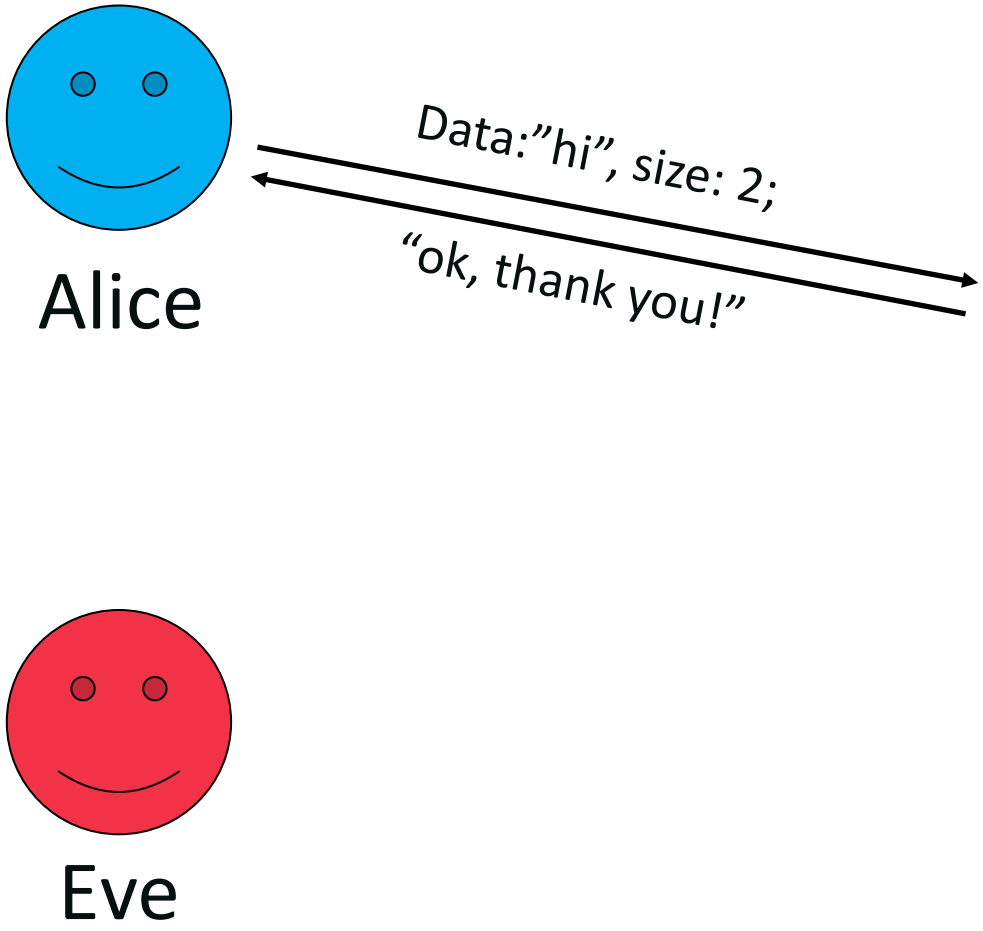
Good Alloc



Server

```
Read_user_data(user_data, size)
        …
            Buf = malloc(size)
            If (buf != NULL)
                        memcpy(buf, user_data, size)
                        return "ok, thank you!"
            else
                        return "sorry too much data"
        …
```

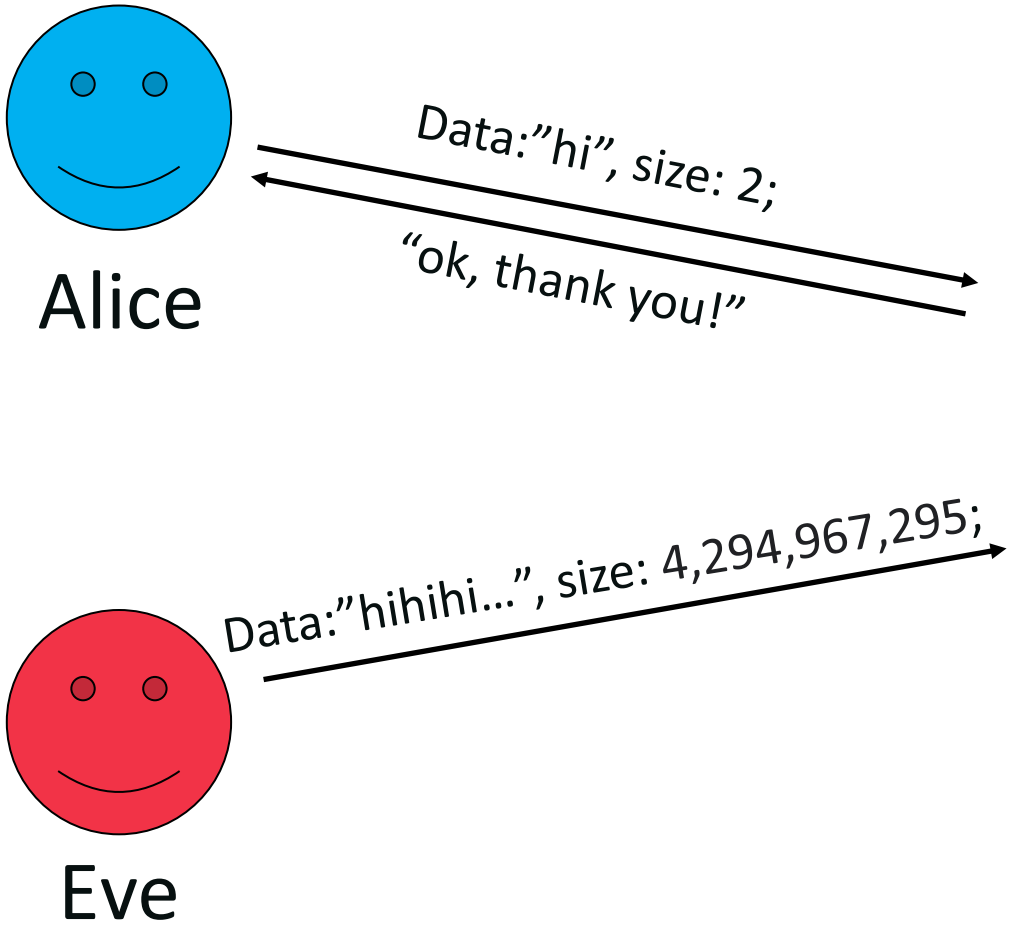
Good Alloc



Server

```
Read_user_data(user_data, size)
        …
            Buf = malloc(size)
            If (buf != NULL)
                        memcpy(buf, user_data, size)
                        return "ok, thank you!"
            else
                        return "sorry too much data"
        …
```

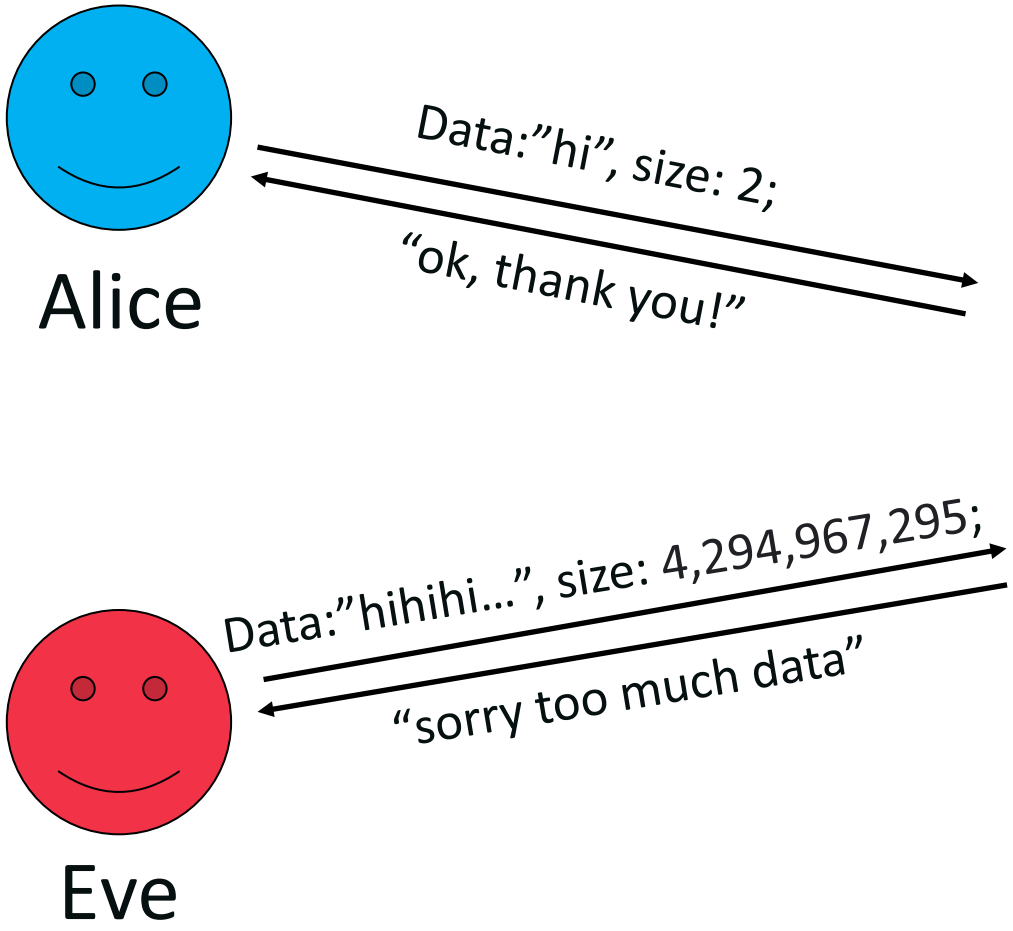
Good Alloc



Server

```
Read_user_data(user_data, size)
        …
            Buf = malloc(size)
            If (buf != NULL)
                        memcpy(buf, user_data, size)
                        return "ok, thank you!"
            else
                        return "sorry too much data"
        …
```

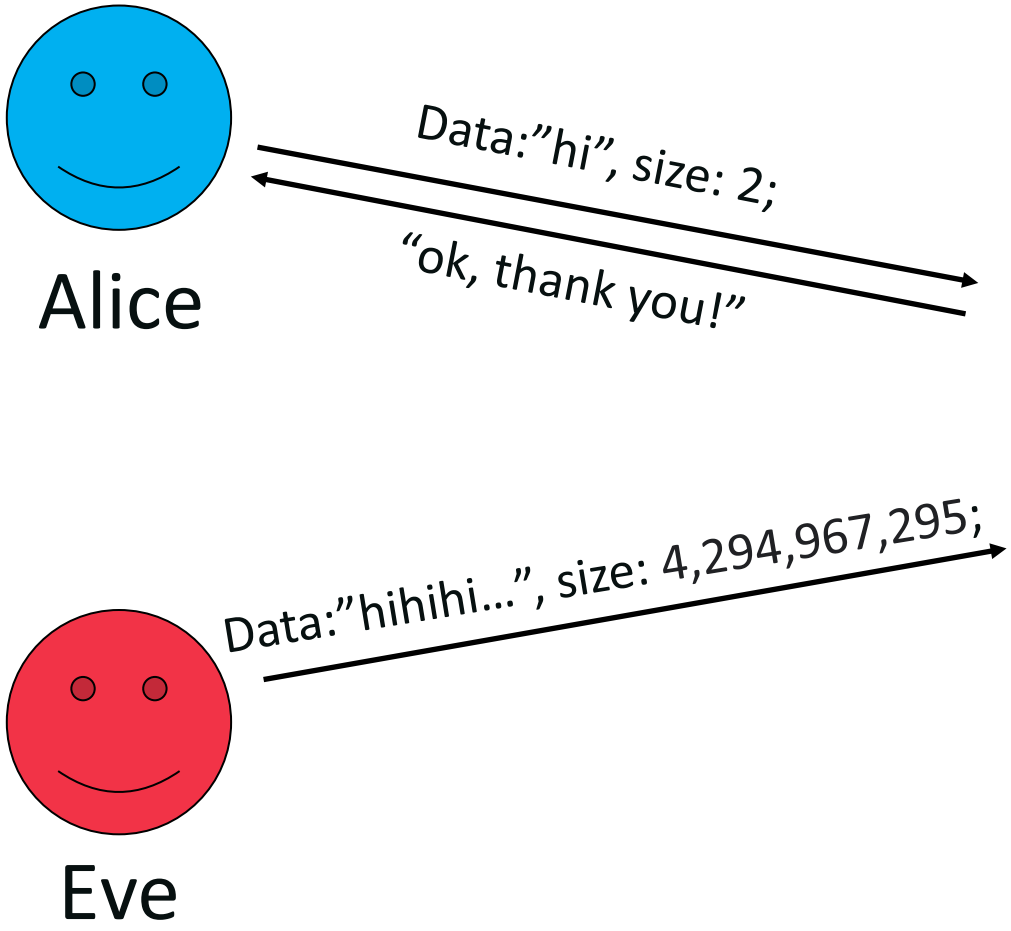
Good Alloc



Server

```
Read_user_data(user_data, size)
        …
            Buf = malloc(size)
            If (buf != NULL)
                        memcpy(buf, user_data, size)
                        return "ok, thank you!"
            else
                        return "sorry too much data"
        …
```

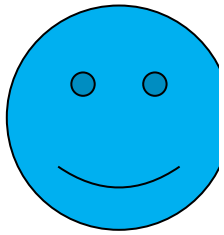
Bad Alloc



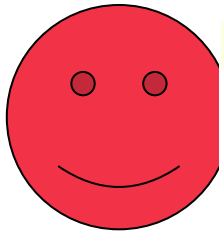
Server

```
Read_user_data(user_data, size)
        …
            Buf = bad_malloc(size)
            If (buf != NULL)
                        memcpy(buf, user_data, size)
                        return "ok, thank you!"
            else
                        return "sorry too much data"
        …
```

Bad Alloc



Alice



Eve

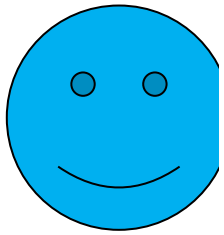
```
void * pvPortMalloc( size_t xWantedSize )
{
    BlockLink_t * pxBlock, * pxPreviousBlock, * pxNewBlockLink;
    static BaseType_t xHeapHasBeenInitialised = pdFALSE;
    void * pvReturn = NULL;

    vTaskSuspendAll();
    {
        /* If this is the first call to malloc then the heap will require
         * initialisation to setup the list of free blocks. */
        if( xHeapHasBeenInitialised == pdFALSE )
        {
            prvHeapInit();
            xHeapHasBeenInitialised = pdTRUE;
        }

        /* The wanted size is increased so it can contain a BlockLink_t
         * structure in addition to the requested amount of bytes. */
        if( xWantedSize > 0 )
        {
            xWantedSize += heapSTRUCT_SIZE;

            /* Ensure that blocks are always aligned to the required number of bytes. */
            if( ( xWantedSize & portBYTE_ALIGNMENT_MASK ) != 0 )
            {
                /* Byte alignment required. */
                xWantedSize += ( portBYTE_ALIGNMENT - ( xWantedSize & portBYTE_ALIGNMENT_MASK ) );
            }
        }
```

ta, **size**)

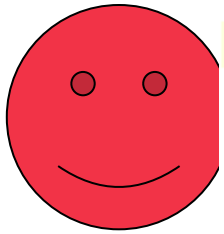
”

h data”

Bad Alloc



Alice



Eve

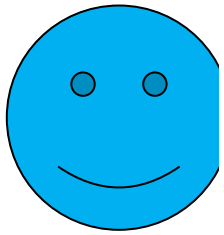
```
void * pvPortMalloc( size_t xWantedSize )
{
    BlockLink_t * pxBlock, * pxPreviousBlock, * pxNewBlockLink;
    static BaseType_t xHeapHasBeenInitialised = pdFALSE;
    void * pvReturn = NULL;

    vTaskSuspendAll();
    {
        /* If this is the first call to malloc then the heap will require
         * initialisation to setup the list of free blocks. */
        if( xHeapHasBeenInitialised == pdFALSE )
        {
            prvHeapInit();
            xHeapHasBeenInitialised = pdTRUE;
        }

        /* The wanted size is increased so it can contain a BlockLink_t
         * structure in addition to the requested amount of bytes. */
        if( xWantedSize > 0 )
        {
            xWantedSize += heapSTRUCT_SIZE;

            /* Ensure that blocks are always aligned to the required number of bytes. */
            if( ( xWantedSize & portBYTE_ALIGNMENT_MASK ) != 0 )
            {
                /* Byte alignment required. */
                xWantedSize += ( portBYTE_ALIGNMENT - ( xWantedSize & portBYTE_ALIGNMENT_MASK ) );
            }
        }
```

ta, **size**)

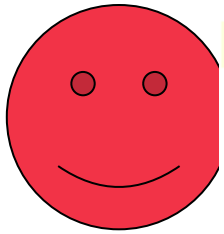
”

h data”

Bad Alloc



Alice



Eve

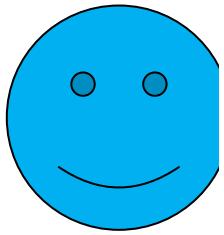
```
void * pvPortMalloc( size_t xWantedSize )
{
    BlockLink_t * pxBlock, * pxPreviousBlock, * pxNewBlockLink;
    static BaseType_t xHeapHasBeenInitialised = pdFALSE;
    void * pvReturn = NULL;

    vTaskSuspendAll();
    {
        /* If this is the first call to malloc then the heap will require
         * initialisation to setup the list of free blocks. */
        if( xHeapHasBeenInitialised == pdFALSE )
        {
            prvHeapInit();
            xHeapHasBeenInitialised = pdTRUE;
        }

        /* The wanted size is increased so it can contain a BlockLink_t
         * structure in addition to the requested amount of bytes. */
        if( xWantedSize > 0 )
        {
            xWantedSize += heapSTRUCT_SIZE;

            /* Ensure that blocks are always aligned to the required number of bytes. */
            if( ( xWantedSize & portBYTE_ALIGNMENT_MASK ) != 0 )
            {
                /* Byte alignment required. */
                xWantedSize += ( portBYTE_ALIGNMENT - ( xWantedSize & portBYTE_ALIGNMENT_MASK ) );
            }
        }
```

ta, **size**)

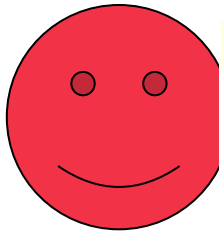
”

h data”

Bad Alloc



Alice



Eve

```
void * pvPortMalloc( size_t xWantedSize )
{
    BlockLink_t * pxBlock, * pxPreviousBlock, * pxNewBlockLink;
    static BaseType_t xHeapHasBeenInitialised = pdFALSE;
    void * pvReturn = NULL;

    vTaskSuspendAll();
    {
        /* If this is the first call to malloc then the heap will require
         * initialisation to setup the list of free blocks. */
        if( xHeapHasBeenInitialised == pdFALSE )
        {
            prvHeapInit();
            xHeapHasBeenInitialised = pdTRUE;
        }

        /* The wanted size is increased so it can contain a BlockLink_t
         * structure in addition to the requested amount of bytes. */
        if( xWantedSize > 0 )
        {
            xWantedSize += heapSTRUCT_SIZE;

            /* Ensure that blocks are always aligned to the required number of bytes. */
            if( ( xWantedSize & portBYTE_ALIGNMENT_MASK ) != 0 )
            {
                /* Byte alignment required. */
                xWantedSize += ( portBYTE_ALIGNMENT - ( xWantedSize & portBYTE_ALIGNMENT_MASK ) );
            }
        }
```

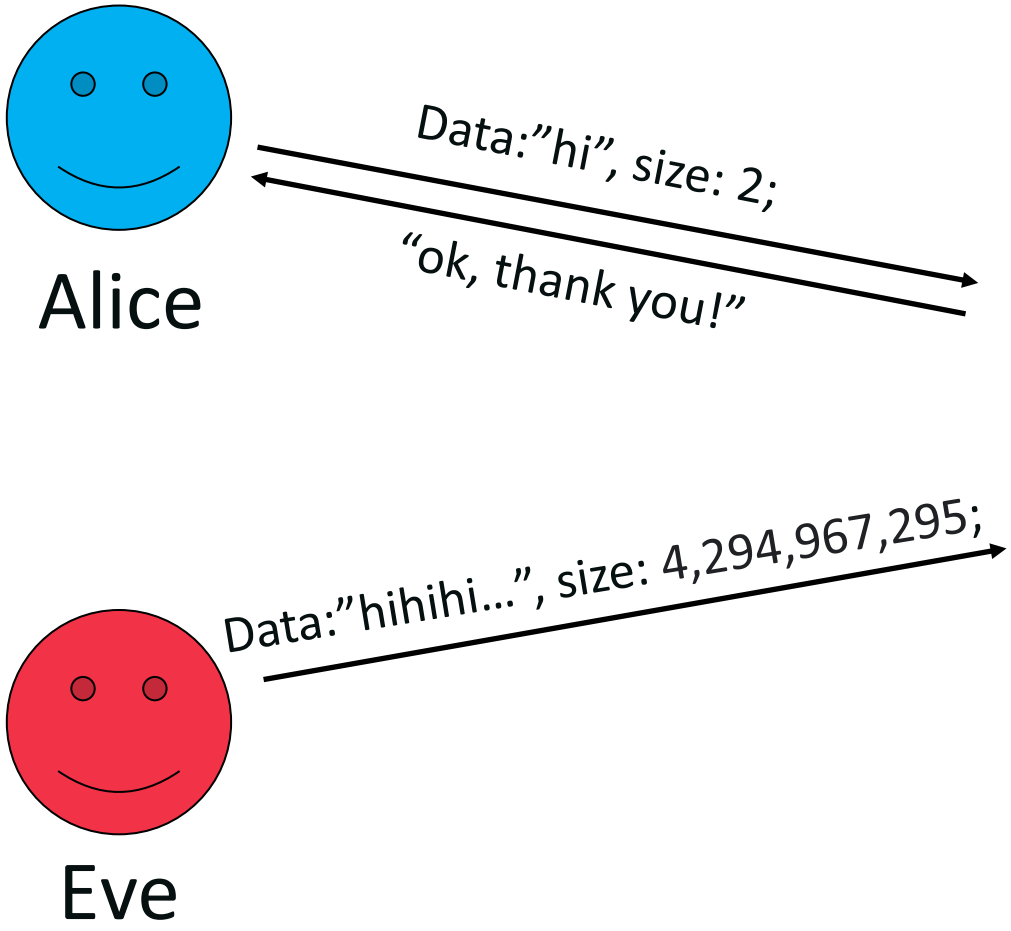
(2^32-1) + 8 = 7

ta, **size**)

”

h data”

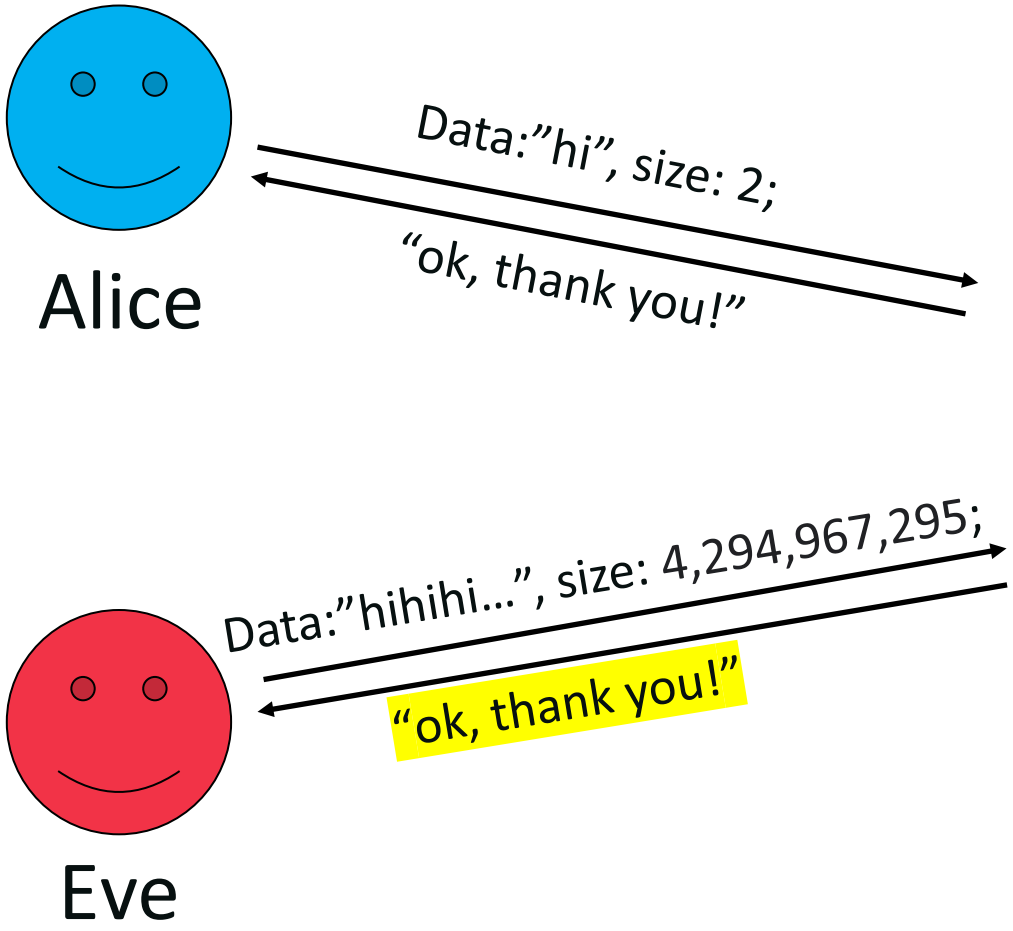
Bad Alloc



Server

```
Read_user_data(user_data, size)
        …
            Buf = bad_malloc(size)
            If (buf != NULL)
                        memcpy(buf, user_data, size)
                        return "ok, thank you!"
            else
                        return "sorry too much data"
        …
```

Bad Alloc



Server

```
Read_user_data(user_data, size)
        …
            Buf = bad_malloc(size)
            If (buf != NULL)
                        memcpy(buf, user_data, size)
                        return "ok, thank you!"
            else
                        return "sorry too much data"
        …
```

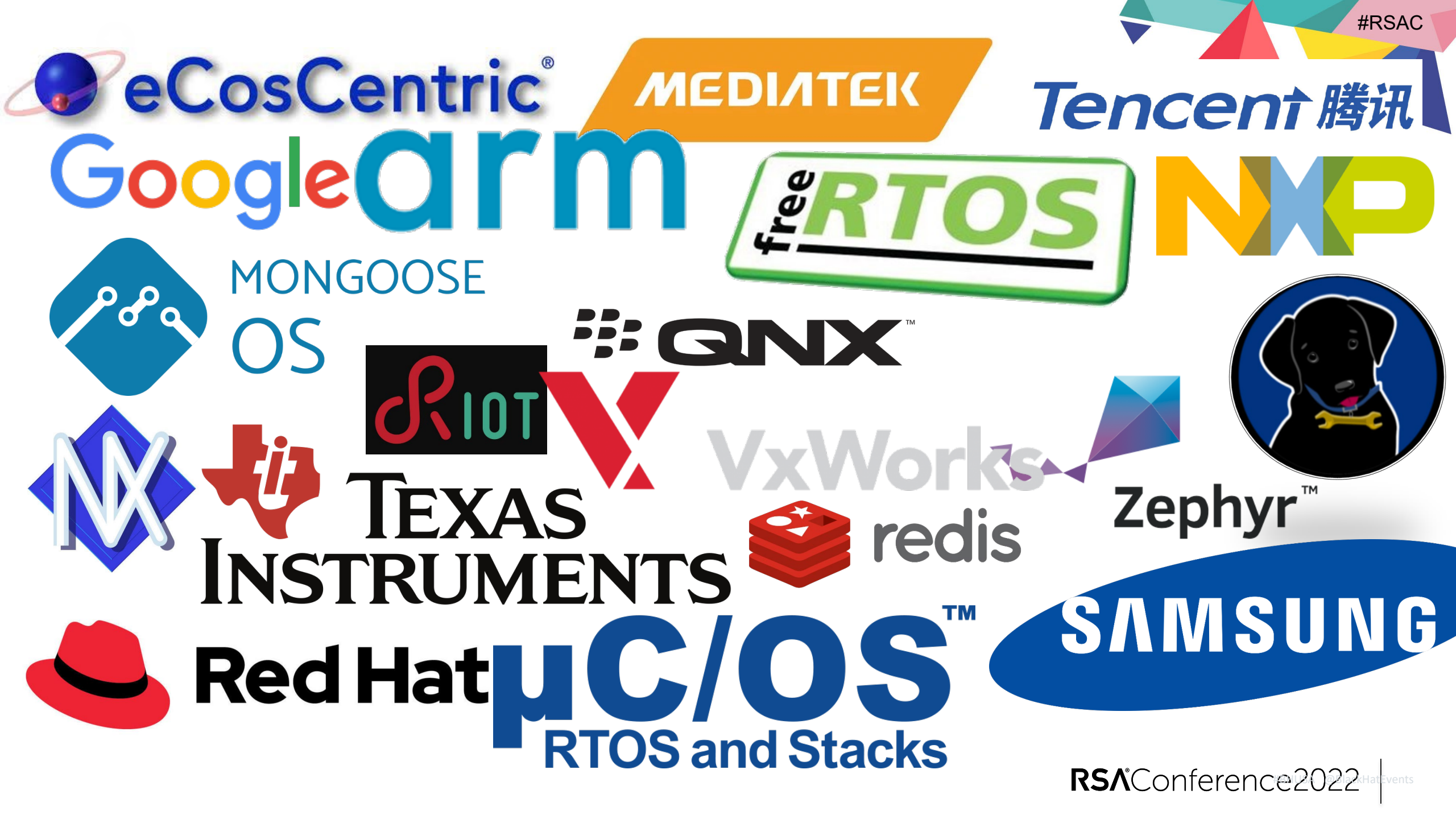
Bad Alloc

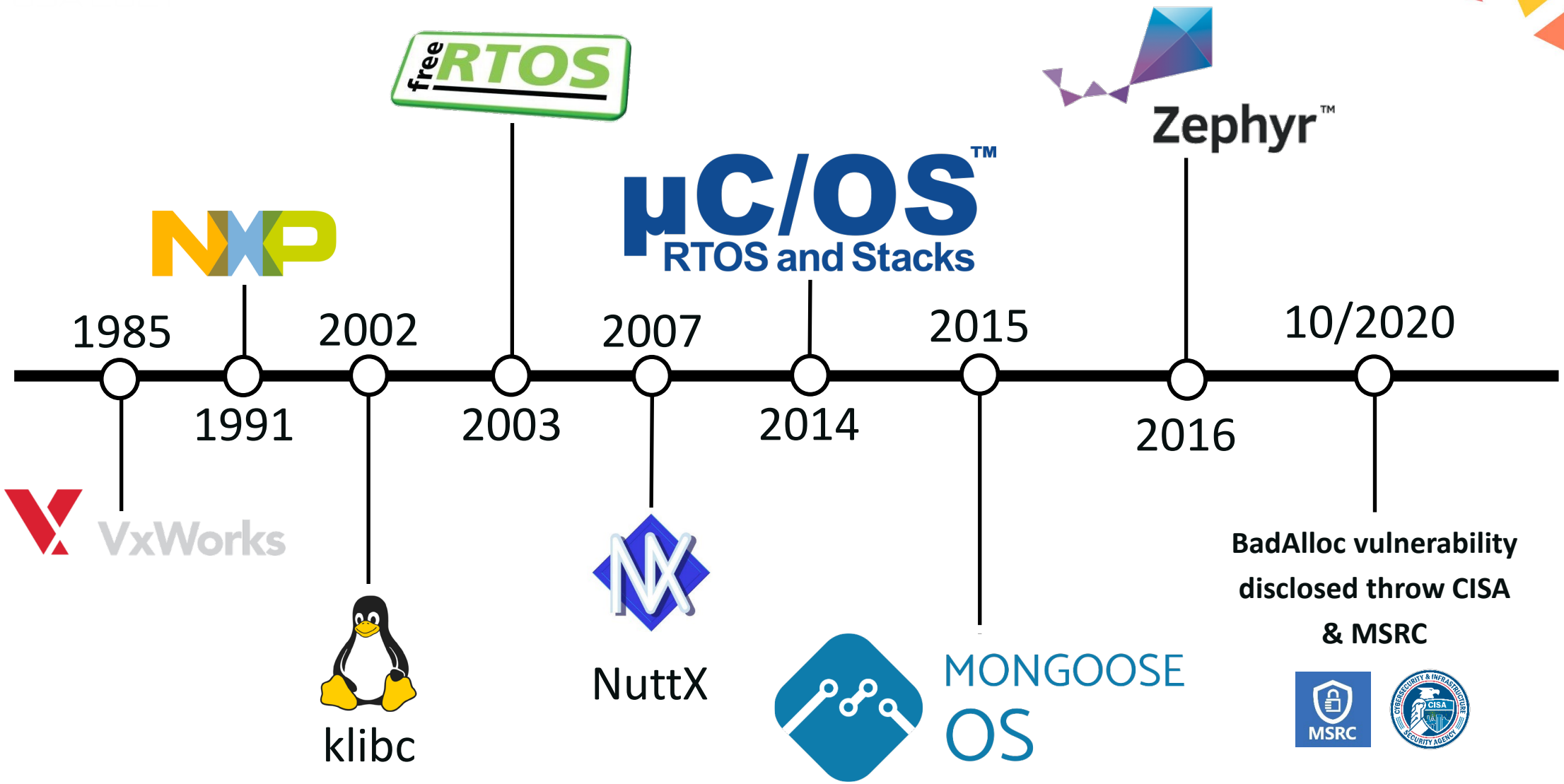
hihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihi
hihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihi
hihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihi
hihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihi
hihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihi
A hihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihi
hihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihi
hihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihi
hihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihi data, **size**)
hihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihi u!”
hihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihi uch data”
hihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihi
hihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihihi

Eve

RSAConference2022

Affected Products


eCosCentric
MEDIATEK
Tencent 腾讯
Google
arm
freeRTOS
NXP
MONGOOSE OS
QNX
RIOT
VxWorks
Zephyr™
redis
TEXAS INSTRUMENTS
Red Hat
µC/OS™
RTOS and Stacks
SAMSUNG


freeRTOS
Zephyr™
µC/OS™
RTOS and Stacks
NXP
1985
1991
2002
2003
2007
2014
2015
2016
10/2020
VxWorks
klibc
NuttX
MONGOOSE OS
BadAlloc vulnerability disclosed throw CISA & MSRC
MSRC
CISA

RSAConference2022

Notable Examples

VxWorks 5.1 - 1993

```
void * calloc(size_t __nmemb,size_t __size)

{
  void *__s;

  __s = (void *)memPartAlloc(memSysPartId,__nmemb * __size);
  if (__s != (void *)0x0) {
    bzero(__s,__nmemb * __size);
  }
  return __s;
}
```

```
843. void *calloc
844.     (
845.     size_t elemNum, /* number of elements */
846.     size_t elemSize /* size of elements */
847.     )
848.     {
849.     FAST void *pMem;
850.     FAST size_t nBytes = elemNum * elemSize;
851.
852.     if ((pMem = memPartAlloc (memSysPartId, (unsigned) nBytes)) != NULL)
853. bzero ((char *) pMem, (int) nBytes);
854.
855.     return (pMem);
856.     }
```

VxWorks 5.1 - 1993

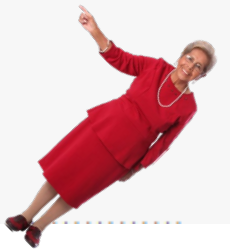
```
void * calloc(size_t __nmemb,size_t __size)

{
  void *__s;

  __s = (void *)memPartAlloc(memSysPartId,__nmemb * __size);
  if (__s != (void *)0x0) {
    bzero(__s,__nmemb * __size);
  }
  return __s;
}
```

```
843. void *calloc
844.    (
845.    size_t elemNum, /* number of elements */
846.    size_t elemSize /* size of elements */
847.    )
848.    {
849.    FAST void *pMem;
850.    FAST size_t nBytes = elemNum * elemSize;
851.
852.    if ((pMem = memPartAlloc (memSysPartId, (unsigned) nBytes)) != NULL)
853. bzero ((char *) pMem, (int) nBytes);
854.
855.    return (pMem);
856.    }
```

VxWorks 5.1 - 1993

```
void * calloc(size_t __nmemb,size_t __size)

{
  void *__s;

  __s = (void *)memPartAlloc(memSysPartId,__nmemb * __size);
  if (__s != (void *)0x0) {
    bzero(__s,__nmemb * __size);
  }
  return __s;
}
```

```
843. void *calloc
844.     (
845.     size_t elemNum, /* number of elements */
846.     size_t elemSize /* size of elements */
847.     )
848.     {
849.     FAST void *pMem;
850.     FAST size_t nBytes = elemNum * elemSize;
851.
852.     if ((pMem = memPartAlloc (memSysPartId, (unsigned) nBytes)) != NULL)
853. bzero ((char *) pMem, (int) nBytes);
854.
855.     return (pMem);
856.     }
```



Klibc - 2002

index : klibc/klibc.git

klibc main development tree

about summary refs log tree commit diff stats

path: root/klibc/calloc.c

author H. Peter Anvin <hpa@zytor.com> 2002-08-06 00:25:09 +0000
committer H. Peter Anvin <hpa@zytor.com> 2002-08-06 00:25:09 +0000
commit 74b67d34871be80a0ed5ef636f5d3ec9d97c0b99 (patch)
tree 91d66a855bca52ee84b4cc860d88061104767347 /klibc/calloc.c
parent 1b20b39d14c1bf37f011453a23a8d8306036b096 (diff)
download klibc-74b67d34871be80a0ed5ef636f5d3ec9d97c0b99.tar.gz

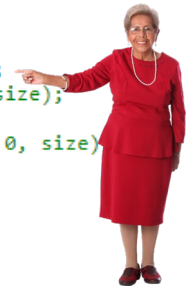
Add calloc() and realloc()

Diffstat (limited to 'klibc/calloc.c')

-rw-r--r-- klibc/calloc.c 20

1 files changed, 20 insertions, 0 deletions

```
diff --git a/klibc/calloc.c b/klibc/calloc.c
new file mode 100644
index 0000000000000..490e3002fe7da
--- /dev/null
+++ b/klibc/calloc.c
@@ -0,0 +1,20 @@
+/*
+ * calloc.c
+ */
+
+#include <stdlib.h>
+
+/* FIXME: This should look for multiplication overflow */
+
+void *calloc(size_t nmemb, size_t size)
+{
+  void *ptr;
+
+  size *= nmemb;
+  ptr = malloc(size);
+  if ( ptr )
+    memset(ptr, 0, size);
+
+  return ptr;
+}
+
```



Klibc - 2002

index : klibc/klibc.git

klibc main development tree

about summary refs log tree commit diff stats

path: root/klibc/calloc.c

author H. Peter Anvin <hpa@zytor.com> 2002-08-06 00:25:0
committer H. Peter Anvin <hpa@zytor.com> 2002-08-06 00:25:09 +
commit 74b67d34871be80a0ed5ef636f5d3ec9d97c0b99 (patch)
tree 91d66a855bca52ee84b4cc860d88061104767347 /klibc/cal
parent 1b20b39d14c1bf37f011453a23a8d8306036b096 (diff)
download klibc-74b67d34871be80a0ed5ef636f5d3ec9d97c0b99.tar.g

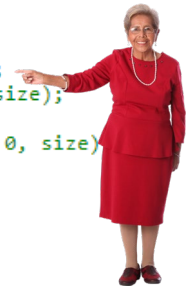
Add calloc() and realloc()

Diffstat (limited to 'klibc/calloc.c')

-rw-r--r-- klibc/calloc.c 20

1 files changed, 20 insertions, 0 deletions

```
diff --git a/klibc/calloc.c b/klibc/calloc.c
new file mode 100644
index 0000000000000..490e3002fe7da
--- /dev/null
+++ b/klibc/calloc.c
@@ -0,0 +1,20 @@
+/*
+ * calloc.c
+ */
+
+#include <stdlib.h>
+
+/* FIXME: This should look for multiplication overflow */
+
+void *calloc(size_t nmemb, size_t size)
+{
+  void *ptr;
+
+  size *= nmemb;
+  ptr = malloc(size);
+  if ( ptr )
+    memset(ptr, 0, size);
+
+  return ptr;
+}
+
```



Klibc - 2002

index : klibc/klibc.git

klibc main development tree

about summary refs log tree commit diff stats

path: root/klibc/calloc.c

author	H. Peter Anvin <hpa@zytor.com> 2002-08-06
committer	H. Peter Anvin <hpa@zytor.com> 2002-08-06 00:25:09
commit	74b67d34871be80a0ed5ef636f5d3ec9d97c0b99 (patch)
tree	91d66a855bca52ee84b4cc860d88061104767347 /klibc/cal
parent	1b20b39d14c1bf37f011453a23a8d8306036b096 (diff)
download	klibc-74b67d34871be80a0ed5ef636f5d3ec9d97c0b99.tar.

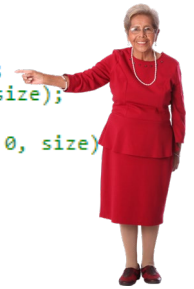
Add calloc() and realloc()

Diffstat (limited to 'klibc/calloc.c')

-rw-r--r-- klibc/calloc.c 20

1 files changed, 20 insertions, 0 deletions

```
diff --git a/klibc/calloc.c b/klibc/calloc.c
new file mode 100644
index 0000000000000..490e3002fe7da
--- /dev/null
+++ b/klibc/calloc.c
@@ -0,0 +1,20 @@
+/*
+ * calloc.c
+ */
+
+#include <stdlib.h>
+
+/* FIXME: This should look for multiplication overflow */
+
+void *calloc(size_t nmemb, size_t size)
+{
+  void *ptr;
+
+  size *= nmemb;
+  ptr = malloc(size);
+  if ( ptr )
+    memset(ptr, 0, size);
+
+  return ptr;
+}
+
```



Klibc - 2002

index : klibc/klibc.git

klibc main development tree

about summary refs log tree commit diff stats

path: root/klibc/calloc.c

author H. Peter Anvin <hpa@zytor.com> 2002-08-06 00:25:0

committer H. Peter Anvin <hpa@zytor.com> 2002-08-06 00:25:09 +

commit 74b67d34871be80a0ed5ef636f5d3ec9d97c0b99 (patch)

tree 91d66a855bca52ee84b4cc860d88061104767347 /klibc/cal

parent 1b20b39d14c1bf37f011453a23a8d8306036b096 (diff)

download klibc-74b67d34871be80a0ed5ef636f5d3ec9d97c0b99.tar.g

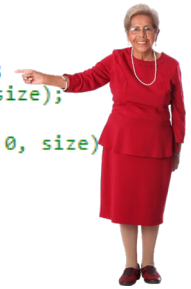
Add calloc() and realloc()

Diffstat (limited to 'klibc/calloc.c')

-rw-r--r-- klibc/calloc.c 20

1 files changed, 20 insertions, 0 deletions

```
diff --git a/klibc/calloc.c b/klibc/calloc.c
new file mode 100644
index 0000000000000..490e3002fe7da
--- /dev/null
+++ b/klibc/calloc.c
@@ -0,0 +1,20 @@
+/*
+ * calloc.c
+ */
+
+#include <stdlib.h>
+
+/* FIXME: This should look for multiplication overflow */
+
+void *calloc(size_t nmemb, size_t size)
+{
+  void *ptr;
+
+  size *= nmemb;
+  ptr = malloc(size);
+  if ( ptr )
+    memset(ptr, 0, size);
+
+  return ptr;
+}
+
```



klibc - 2002

index : klibc/klibc.git

klibc main development tree

about summary refs log tree commit diff stats

path: root/klibc/calloc.c

author H. Peter Anvin <hpa@zytor.com> 2002-08-06 00:25:0
committer H. Peter Anvin <hpa@zytor.com> 2002-08-06 00:25:09 +
commit 74b67d34871be80a0ed5ef636f5d3ec9d97c0b99 (patch)
tree 91d66a855bca52ee84b4cc860d88061104767347 /klibc/cal
parent 1b20b39d14c1bf37f011453a23a8d8306036b096 (diff)
download klibc-74b67d34871be80a0ed5ef636f5d3ec9d97c0b99.tar.g

Add calloc() and realloc()

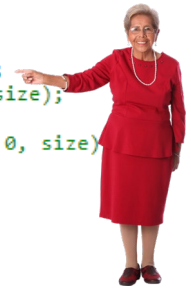
Diffstat (limited to 'klibc/calloc.c')

-rw-r--r-- klibc/calloc.c 20

1 files changed, 20 insertions, 0 deletions

```
+/* FIXME: This should look for multiplication overflow */
```

```
+++ b/klibc/calloc.c
@@ -0,0 +1,20 @@
+/*
+ * calloc.c
+ */
+
+#include <stdlib.h>
+
+/* FIXME: This should look for multiplication overflow */
+
+void *calloc(size_t nmemb, size_t size)
+{
+  void *ptr;
+
+  size *= nmemb;
+  ptr = malloc(size);
+  if ( ptr )
+    memset(ptr, 0, size);
+
+  return ptr;
+}
+
```

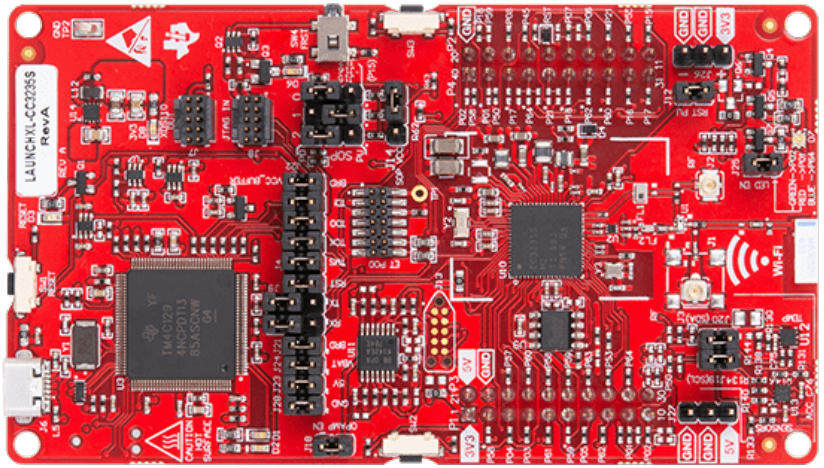


RSAConference2022

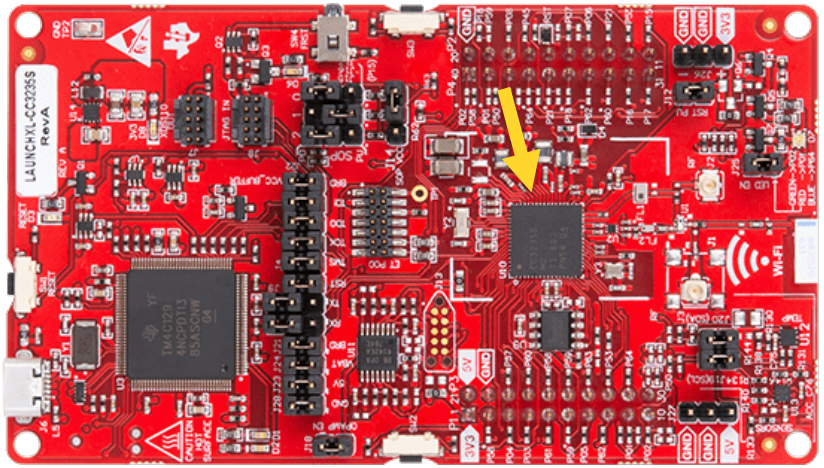
Technical Analysis

Texas Instruments “SimpleLink” SDK

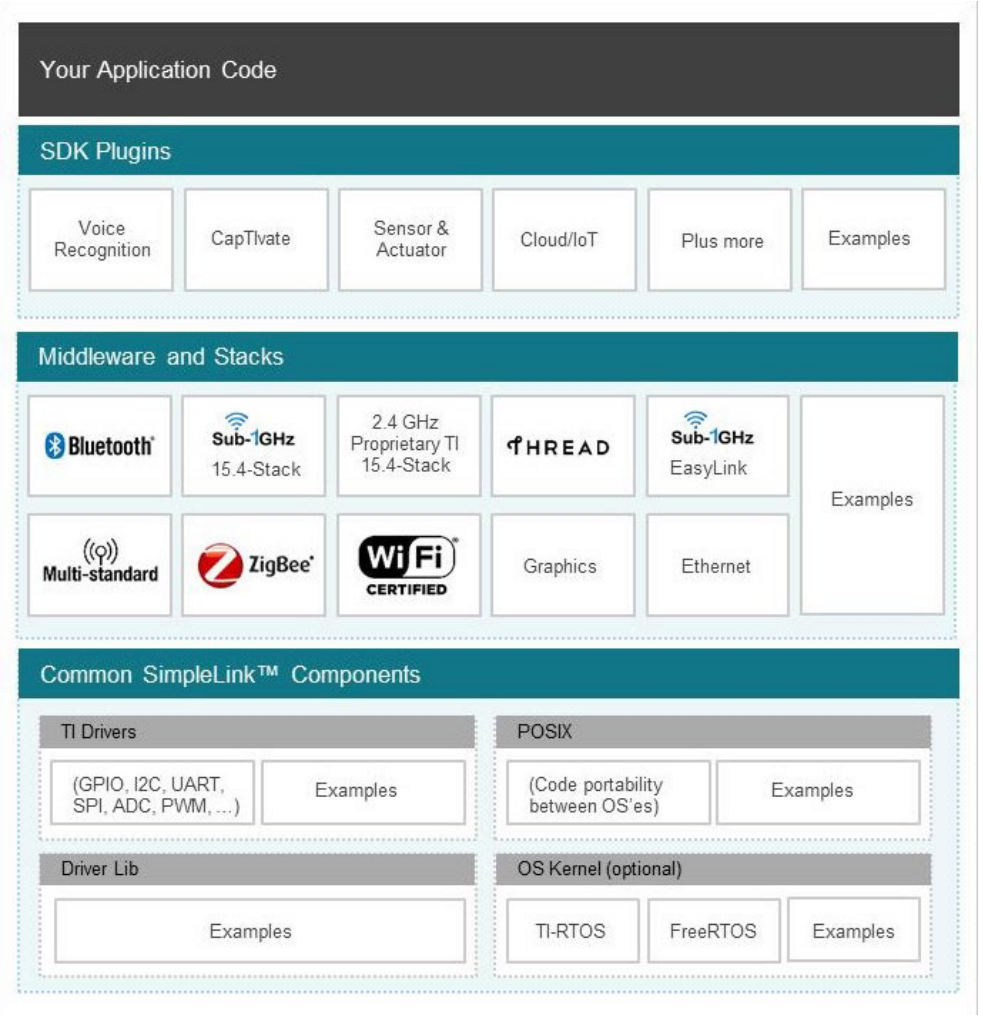
Texas Instruments “SimpleLink” SDK



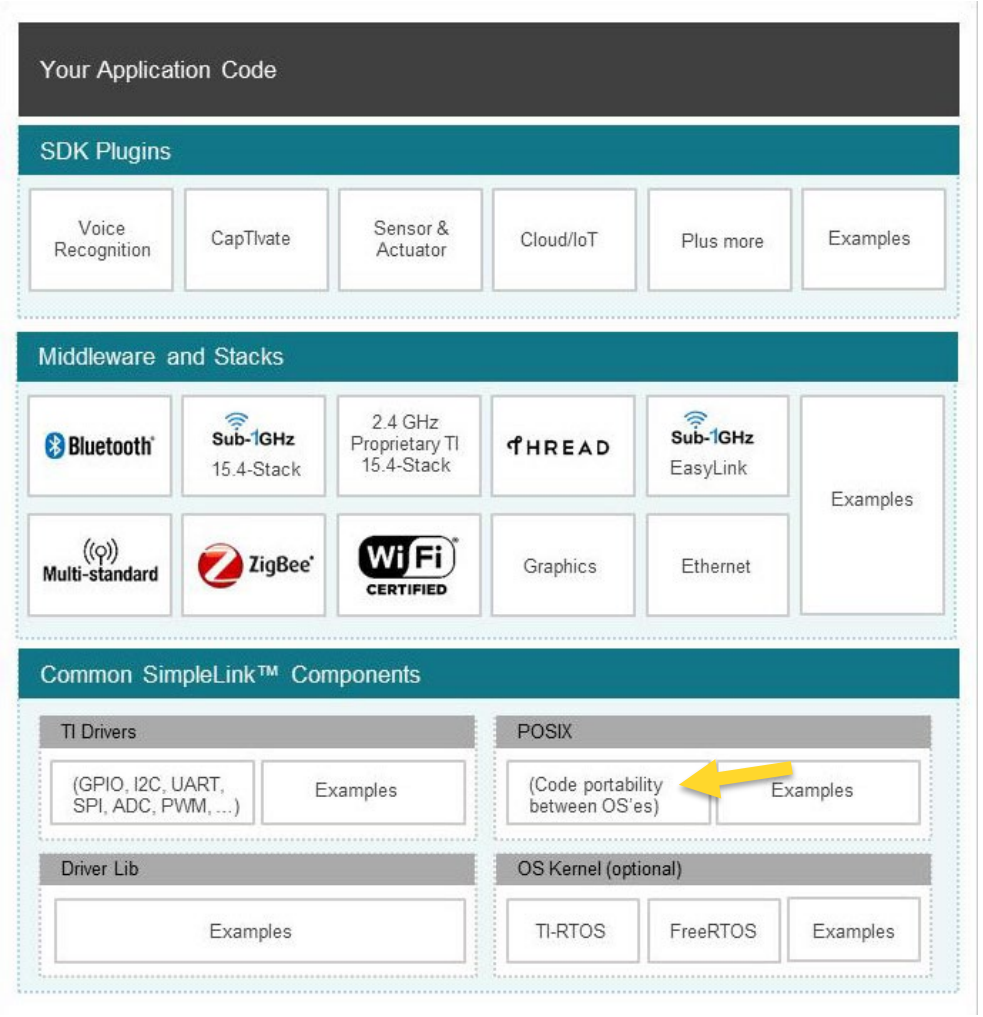
Texas Instruments “SimpleLink” SDK



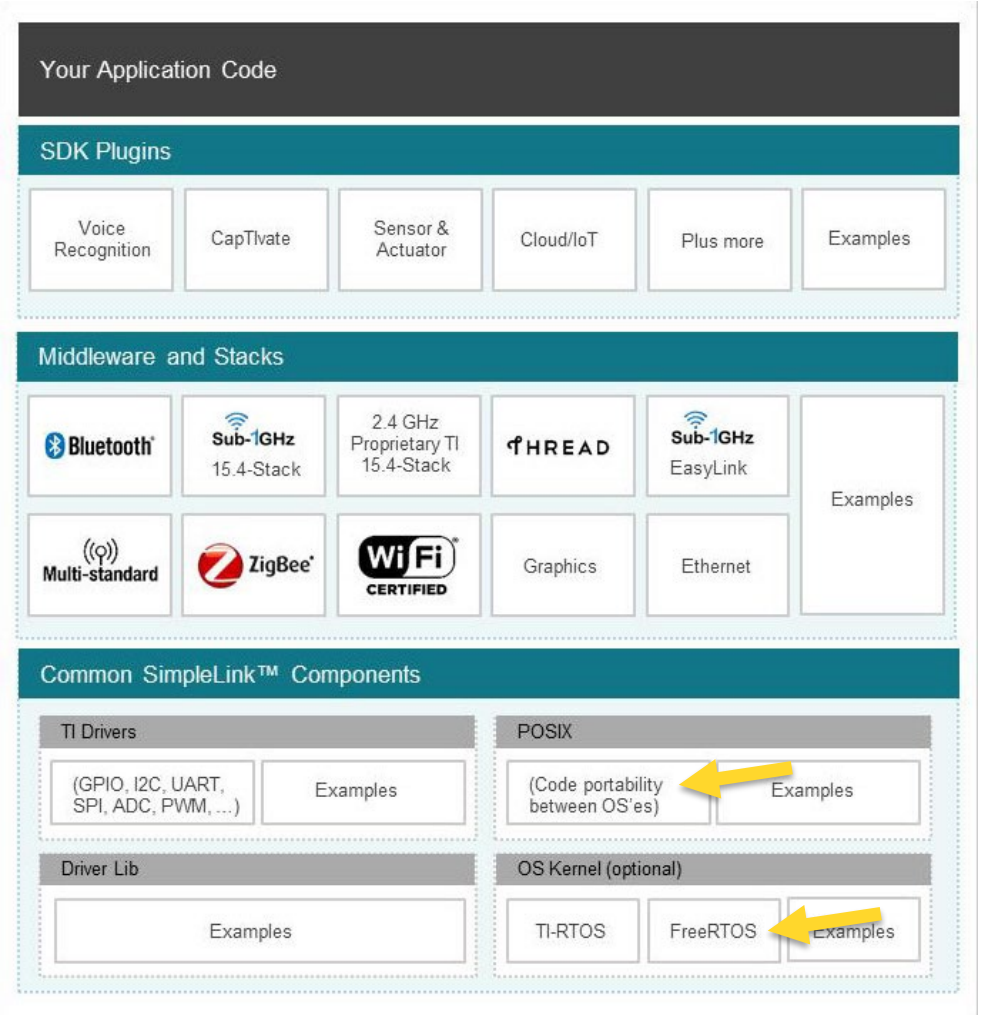
Texas Instruments “SimpleLink” SDK



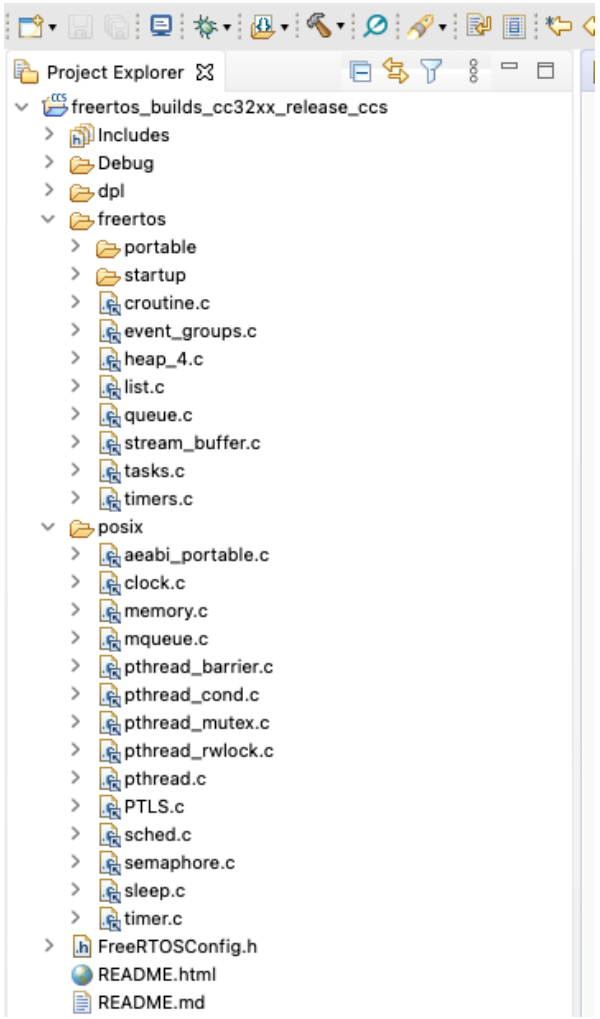
Texas Instruments “SimpleLink” SDK



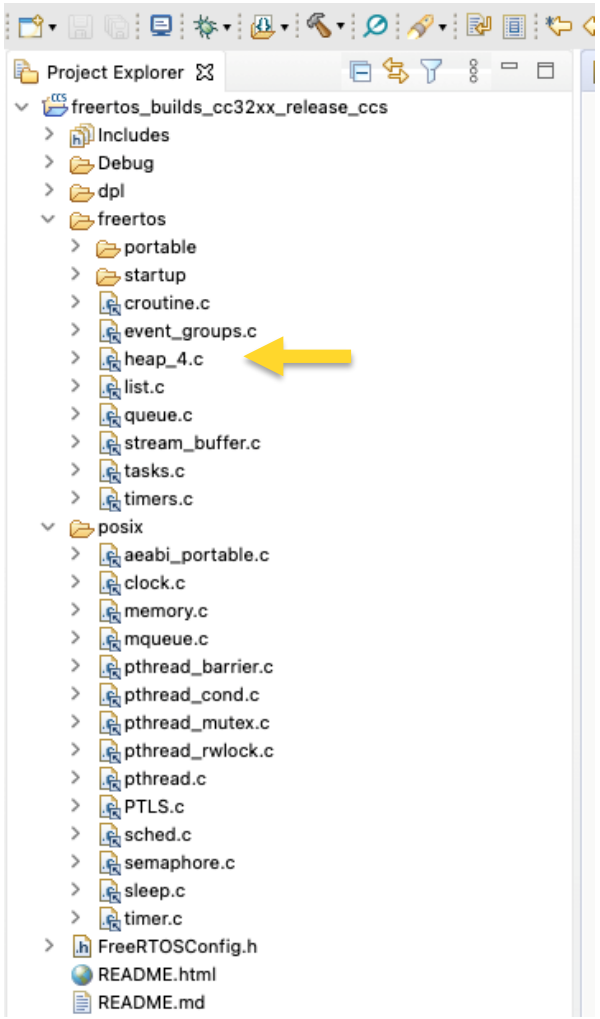
Texas Instruments “SimpleLink” SDK



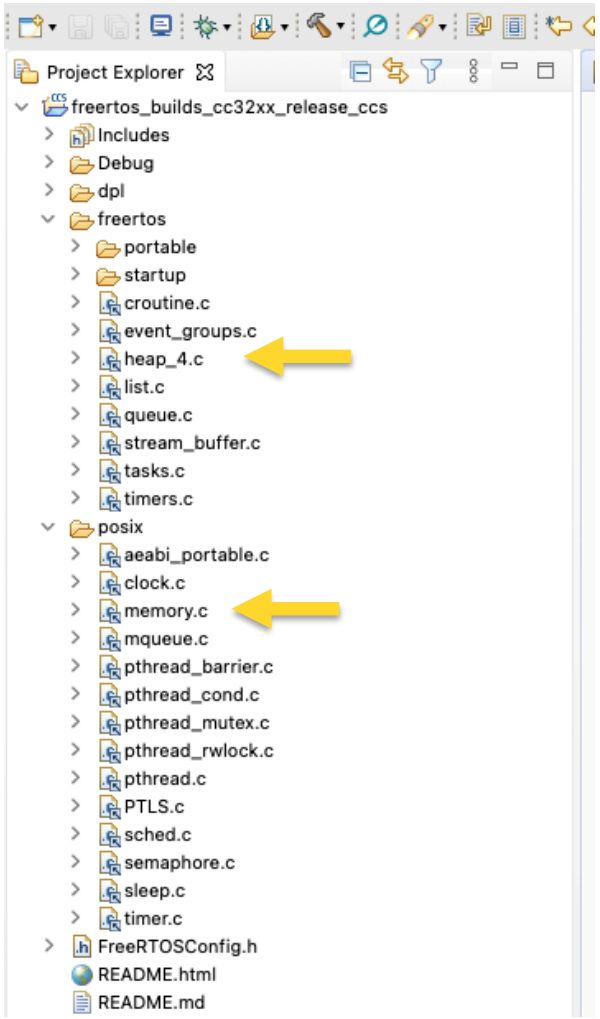
Texas Instruments “SimpleLink” SDK



Texas Instruments “SimpleLink” SDK



Texas Instruments “SimpleLink” SDK



calloc is safe

```
/*
 *  ======== calloc ========
 */
void ATTRIBUTE *calloc(size_t nmemb, size_t size)
{
    size_t nbytes;
    void *retval;

    /* guard against divide by zero exception below */
    if (nmemb == 0) {
        errno = EINVAL;
        return (NULL);
    }

    nbytes = nmemb * size;

    /* return NULL if there's an overflow */
    if (nmemb && size != (nbytes / nmemb)) {
        errno = EOVERFLOW;
        return (NULL);
    }

    retval = malloc(nbytes);
    if (retval != NULL) {
        (void)memset(retval, (int)'\0', nbytes);
    }

    return (retval);
}
```

calloc is safe

```
/*
 *  ======== calloc ========
 */
void ATTRIBUTE *calloc(size_t nmemb, size_t size)
{
    size_t nbytes;
    void *retval;

    /* guard against divide by zero exception below */
    if (nmemb == 0) {
        errno = EINVAL;
        return (NULL);
    }

    nbytes = nmemb * size;

    /* return NULL if there's an overflow */
    if (nmemb && size != (nbytes / nmemb)) {
        errno = EOVERFLOW;
        return (NULL);
    }

    retval = malloc(nbytes);
    if (retval != NULL) {
        (void)memset(retval, (int)'\0', nbytes);
    }

    return (retval);
}
```

calloc is safe

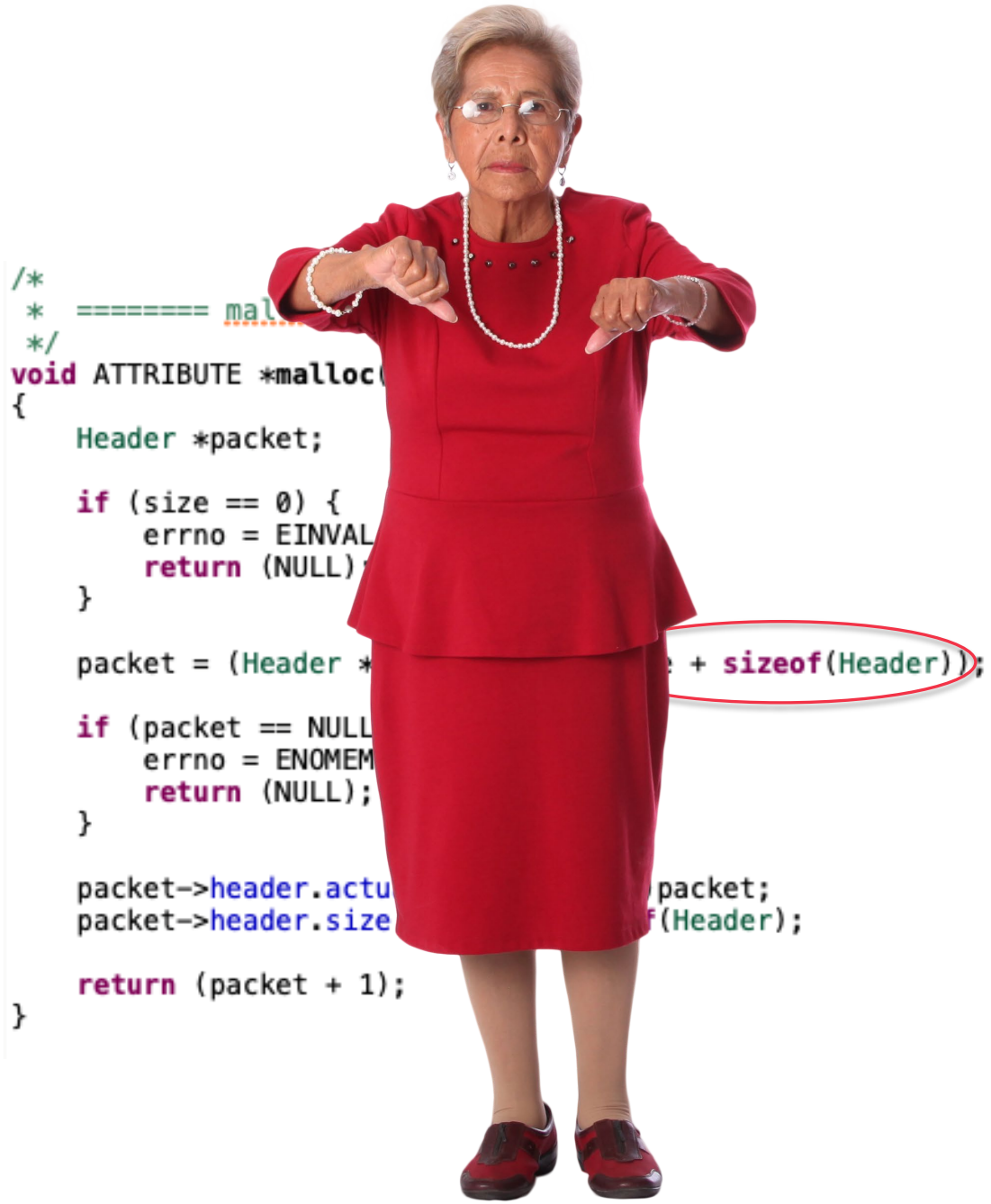
malloc isn't

```
/*
 *  ======== malloc ========
 */
void ATTRIBUTE *malloc(size_t size)
{
    Header *packet;

    if (size == 0) {
        errno = EINVAL;
        return (NULL);
    }

    packet = (Header *)pvPortMalloc(size + sizeof(Header));

    if (packet == NULL) {
        errno = ENOMEM;
        return (NULL);
    }

    packet->header.actualBuf = (void *)packet;
    packet->header.size = size + sizeof(Header);

    return (packet + 1);
}
```

malloc isn't

```
/*
 *  ======== malloc ========
 */
void ATTRIBUTE *malloc(size_t size)
{
    Header *packet;

    if (size == 0) {
        errno = EINVAL;
        return (NULL);
    }

    packet = (Header *)pvPortMalloc(size + sizeof(Header));

    if (packet == NULL) {
        errno = ENOMEM;
        return (NULL);
    }

    packet->header.actualBuf = (void *)packet;
    packet->header.size = size + sizeof(Header);

    return (packet + 1);
}
```

malloc isn't


/*
* ======== mal
*/
void ATTRIBUTE *malloc(
{
Header *packet;
if (size == 0) {
errno = EINVAL
return (NULL);
}
packet = (Header *
+ sizeof(Header));
if (packet == NULL
errno = ENOMEM
return (NULL);
}
packet->header.actu
packet;
packet->header.size
(Header);
return (packet + 1);
}

RSAConference2022

Exploitation

SimpleLink POC

OTA(Over the Air) Updates

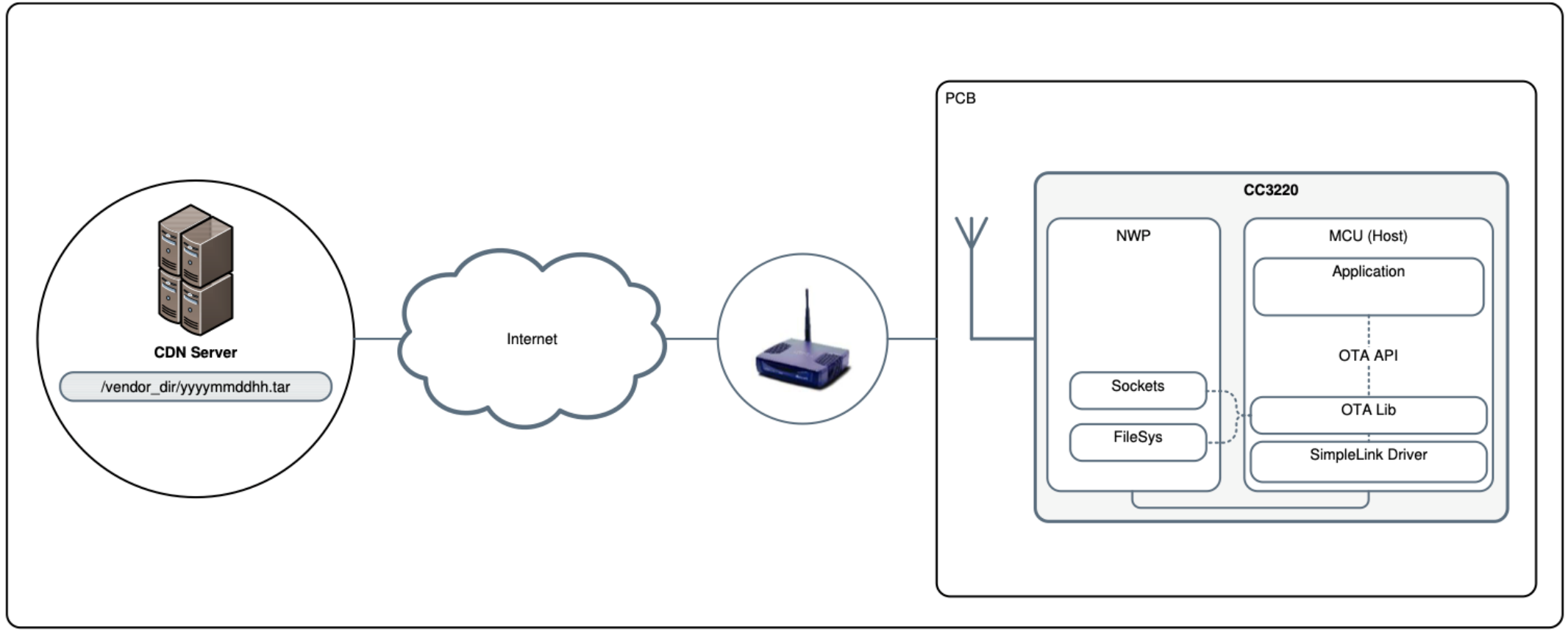
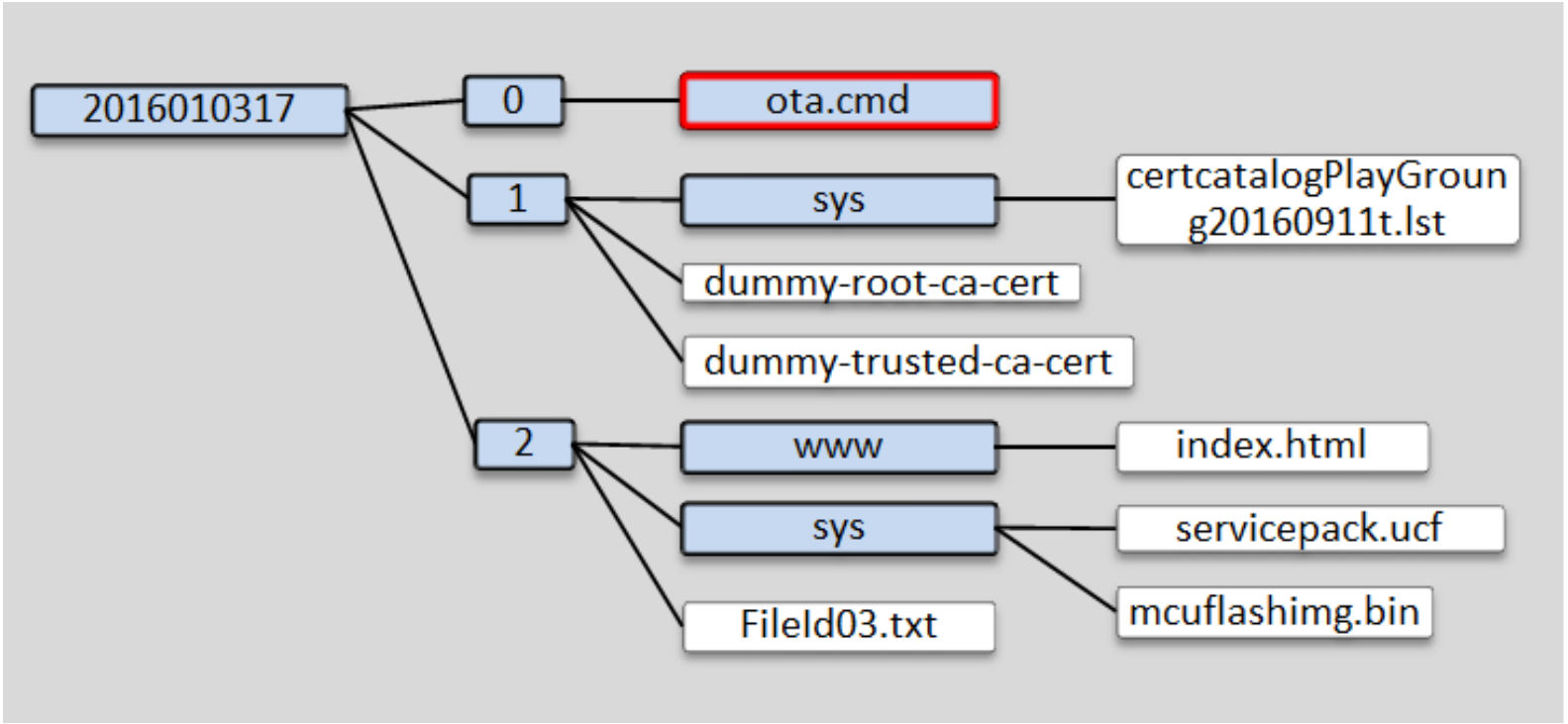


Figure 1-1. OTA System Diagram

Bundle command signature file



Bundle command signature file

```
[
    {
        "filename": "/local/FileId03.txt",
        "signature_base64": "kc8XfFOfMfr4HBJiPxTRHyb99d2uOoICme0AYU94+...",
        "certificate": "dummy-trusted-ca-certcert",
        "secured": 1,
        "bundle": 0
    },
    {
        "filename": "/sys/servicepack.ucf"
        "signature_base64": "EEC6GZG1Oq6Agigmb2f9ny9rNK2Mg9hFClpgMhd4jCW/...",
        "certificate":"",
        "secured": 1,
        "bundle": 1
    },
    {
        "filename": "/sys/mcuflashimg.bin",
        "signature_base64": "dRTARlzLFKAog34ZUareCmo9j2lrHnvc+v3qqW9C/...",
        "certificate": "dummy-root-ca-certcert",
        "secured": 1,
        "bundle": 1
    }
]
```

Bundle file parsing

```
667
668 int16_t _BundleCmdSignatureFile_Parse(
669     OtaArchive_BundleCmdTable_t *pBundleCmdTable,
670     uint8_t *pRecvBuf,
671     int16_t RecvBufLen,
672     int16_t *ProcessedSize,
673     uint32_t SigFileSize,
674     uint8_t *pDigest)
675 {
676     int16_t retVal = 0;
677     char *  pSig = NULL;
678
679     /* Get the entire signature file */
680     retVal = GetEntireFile(pRecvBuf, RecvBufLen, ProcessedSize, SigFileSize,
681                            &pSig);
682     if(retVal < 0)
683     {
684         return(retVal);
685     }
686     if(retVal == GET_ENTIRE_FILE_CONTINUE)
687     {
688         return(ARCHIVE_STATUS_BUNDLE_CMD_SIGNATURE_CONTINUE);
689     }
690
691     /* Verify the signature using ECDSA */
692     retVal = verifySignature(pSig, SigFileSize, pDigest);
693     if(retVal < 0)
694     {
695         _SlOtaLibTrace((
696                            "[_BundleCmdSignatureFile_Parse] "
697                            "signature verification failed!\r\n"));
698         return(retVal);
699     }
700
701     pBundleCmdTable->VerifiedSignature = 1;
702
703     return(ARCHIVE_STATUS_BUNDLE_CMD_SIGNATURE_DOWNLOAD_DONE);
704 }
705
706 OtaArchive_BundleFileInfo_t * _BundleCmdFile_GetInfoByFileName(
```

GetEntireFile

```
154 ****************************************************************************/
155
156 int16_t GetEntireFile(uint8_t *pRecvBuf,
157                       int16_t RecvBufLen,
158                       int16_t *ProcessedSize,
159                       uint32_t FileSize,
160                       char **pFile)
161 {
162     int16_t copyLen = 0;
163     static bool firstRun = TRUE;
164     static int16_t TotalRecvBufLen = 0;
165
166     if(firstRun)
167     {
168         TotalRecvBufLen = RecvBufLen;
169         firstRun = FALSE;
170         if(TotalRecvBufLen < FileSize)
171         {
172             /* Didn't receive the entire file in the first run. */
173             /* Allocate a buffer in the size of the entire file and fill
174                 it in each round. */
175             pTempBuf = (char*)malloc(FileSize + 1);
176             if(pTempBuf == NULL)
177             {
178                 /* Allocation failed, return error. */
179                 return(-1);
180             }
181             memcpy(pTempBuf, (char *)pRecvBuf, RecvBufLen);
182             *ProcessedSize = RecvBufLen;
183
184             /* didn't receive the entire file, try in the next packet */
185             return(GET_ENTIRE_FILE_CONTINUE);
186         }
187         else
188         {
189             /* Received the entire file in the first run. */
190             /* No additional memory allocation is needed. */
191             *ProcessedSize = FileSize;
192             *pFile = (char *)pRecvBuf;
193         }
194     }
195     else
196     {
197         /* Avoid exceeding buffer size (FileSize + 1) */
198         if(RecvBufLen > ((FileSize + 1) - TotalRecvBufLen))
199         {
200             copyLen = ((FileSize + 1) - TotalRecvBufLen);
```

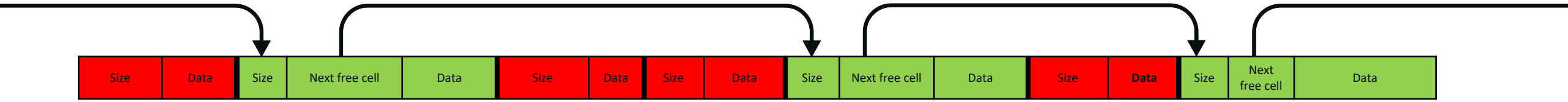
GetEntireFile

```
154 ********************************************************************************/
155
156 int16_t GetEntireFile(uint8_t *pRecvBuf,
157                       int16_t RecvBufLen,
158                       int16_t *ProcessedSize,
159                       uint32_t FileSize,
160                       char **pFile)
161 {
162     int16_t copyLen = 0;
163     static bool firstRun = TRUE;
164     static int16_t TotalRecvBufLen = 0;
165
166     if(firstRun)
167     {
168         TotalRecvBufLen = RecvBufLen;
169         firstRun = FALSE;
170         if(TotalRecvBufLen < FileSize)
171         {
172             /* Didn't receive the entire file in the first run. */
173             /* Allocate a buffer in the size of the entire file and fill
174                it in each round. */
175             pTempBuf = (char*)malloc(FileSize + 1);
176             if(pTempBuf == NULL)
177             {
178                 /* Allocation failed, return error. */
179                 return(-1);
180             }
181             memcpy(pTempBuf, (char *)pRecvBuf, RecvBufLen);
182             *ProcessedSize = RecvBufLen;
183
184             /* didn't receive the entire file, try in the next packet */
185             return(GET_ENTIRE_FILE_CONTINUE);
186         }
187         else
188         {
189             /* Received the entire file in the first run. */
190             /* No additional memory allocation is needed. */
191             *ProcessedSize = FileSize;
192             *pFile = (char *)pRecvBuf;
193         }
194     }
195     else
196     {
197         /* Avoid exceeding buffer size (FileSize + 1) */
198         if(RecvBufLen > ((FileSize + 1) - TotalRecvBufLen))
199         {
200             copyLen = ((FileSize + 1) - TotalRecvBufLen);
```

heap corruption => code execution

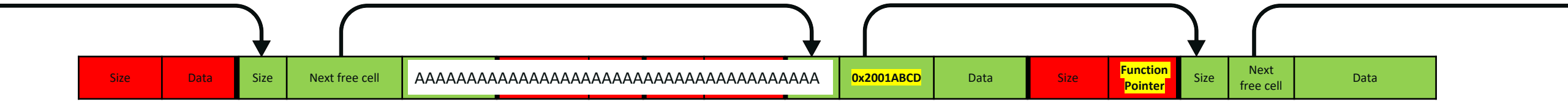
- **Heap overflow**
- Find function pointer which we can override in memory.
- Override “next free” pointer of next block to desired address.
- Force another allocation with user-controlled data.
- Force call to overridden function pointer.

heap corruption => code execution



Allocated Free

heap corruption => code execution



Allocated Free

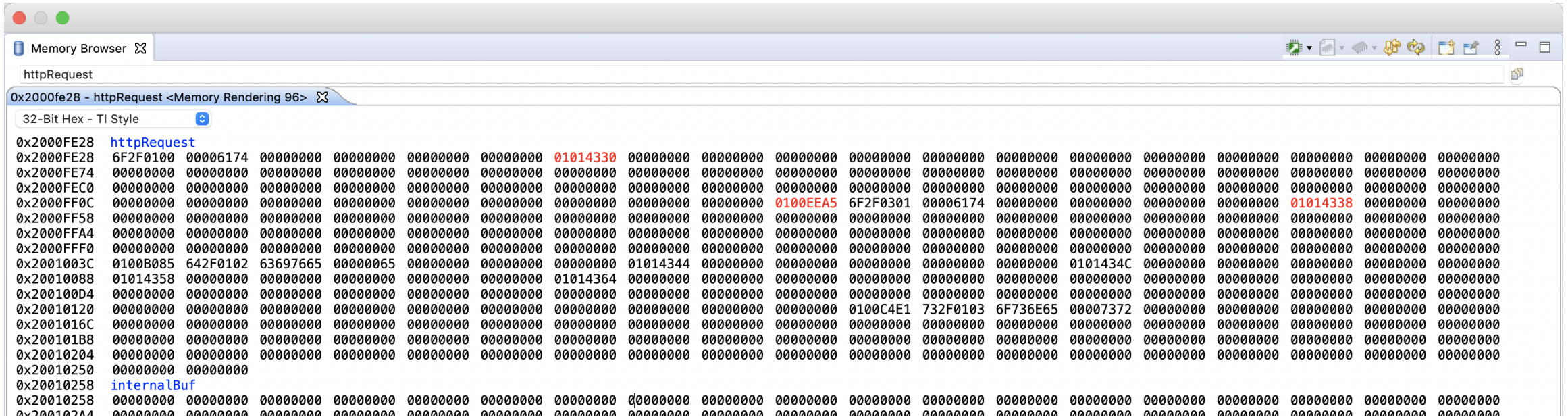
httpRequest struct

```
2134 //*****************************************************************************
2135 void httpGetHandler(SlNetAppRequest_t *netAppRequest)
2136 {
2137     uint16_t metadataLen;
2138     int32_t status;
2139     uint8_t requestIdx;
2140
2141     uint8_t argcCallback;
2142     uint8_t     *argvArray;
2143     uint8_t     **argvCallback = &argvArray;
2144
2145     argvArray = gHttpGetBuffer;
2146
2147     status = httpCheckContentInDB(  netAppRequest,
2148                                     &requestIdx,
2149                                     &argcCallback,
2150                                     argvCallback);
2151
2152     if(status < 0)
2153     {
2154         metadataLen =
2155             prepareGetMetadata(status, strlen (
2156                                     (const char *)pageNotFound),
2157                                HttpContentTypeList_TextHtml);
2158
2159         sl_NetAppSend (netAppRequest->Handle, metadataLen, gMetadataBuffer,
2160                        (SL_NETAPP_REQUEST_RESPONSE_FLAGS_CONTINUATION |
2161                         SL_NETAPP_REQUEST_RESPONSE_FLAGS_METADATA));
2162         INFO_PRINT("[Link local task] Metadata Sent, len = %d \n\r",
2163                    metadataLen);
2164
2165         sl_NetAppSend (netAppRequest->Handle,
2166                        strlen(
2167                           (const char *)pageNotFound), (uint8_t *)pageNotFound,
2168                        0); /* mark as last segment */
2169         INFO_PRINT("[Link local task] Data Sent, len = %d\n\r",
2170                    strlen ((const char *)pageNotFound));
2171     }
2172     else
2173     {
2174         httpRequest[requestIdx].serviceCallback(requestIdx, &argcCallback,
2175                                                 argvCallback,
2176                                                 netAppRequest);
2177     }
2178 }
2179
2180 //*****************************************************************************
```

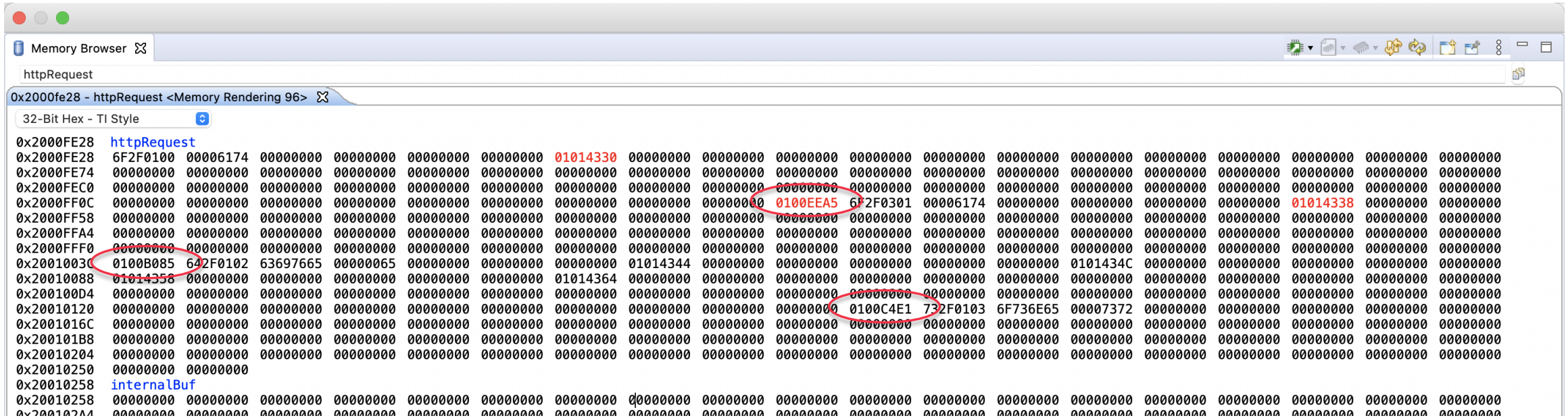
httpRequest struct

```
2134 //*****************************************************************************
2135 void httpGetHandler(SlNetAppRequest_t *netAppRequest)
2136 {
2137     uint16_t metadataLen;
2138     int32_t status;
2139     uint8_t requestIdx;
2140
2141     uint8_t argcCallback;
2142     uint8_t     *argvArray;
2143     uint8_t     **argvCallback = &argvArray;
2144
2145     argvArray = gHttpGetBuffer;
2146
2147     status = httpCheckContentInDB(  netAppRequest,
2148                                     &requestIdx,
2149                                     &argcCallback,
2150                                     argvCallback);
2151
2152     if(status < 0)
2153     {
2154         metadataLen =
2155             prepareGetMetadata(status, strlen (
2156                                     (const char *)pageNotFound),
2157                                HttpContentTypeList_TextHtml);
2158
2159         sl_NetAppSend (netAppRequest->Handle, metadataLen, gMetadataBuffer,
2160                        (SL_NETAPP_REQUEST_RESPONSE_FLAGS_CONTINUATION |
2161                         SL_NETAPP_REQUEST_RESPONSE_FLAGS_METADATA));
2162         INFO_PRINT("[Link local task] Metadata Sent, len = %d \n\r",
2163                    metadataLen);
2164
2165         sl_NetAppSend (netAppRequest->Handle,
2166                        strlen(
2167                           (const char *)pageNotFound), (uint8_t *)pageNotFound,
2168                        0); /* mark as last segment */
2169         INFO_PRINT("[Link local task] Data Sent, len = %d\n\r",
2170                    strlen ((const char *)pageNotFound));
2171     }
2172     else
2173     {
2174         httpRequest[requestIdx].serviceCallback(requestIdx, &argcCallback,
2175                                                 argvCallback,
2176                                                 netAppRequest);
2177     }
2178 }
2179
2180 //*****************************************************************************
```

httpRequest struct



httpRequest struct



RSAConference2022

Demo

Mitigation techniques

- How should you do it?
  - Start by checking-out the advisory.
  - Reverse Engineer the binaries(always the best approach, in life).
  - Source code review if it's public.
  - "Unit tests" - compile a small application the verify the environment.

- It's also recommended to check the macros that are being used in these functions.

Mitigation techniques

- Recommended function to check –
  - malloc
  - calloc
  - realloc
  - memalign
  - valloc
  - pvalloc
  - aligned_alloc

Mitigation techniques

glibc

```
3348  void *
3349  __libc_calloc (size_t n, size_t elem_size)
3350  {
3351    mstate av;
3352    mchunkptr oldtop, p;
3353    INTERNAL_SIZE_T bytes, sz, csz, oldtopsize;
3354    void *mem;
3355    unsigned long clearsize;
3356    unsigned long nclears;
3357    INTERNAL_SIZE_T *d;
3358
3359    /* size_t is unsigned so the behavior on overflow is defined.  */
3360    bytes = n * elem_size;
3361  #define HALF_INTERNAL_SIZE_T \
3362    (((INTERNAL_SIZE_T) 1) << (8 * sizeof (INTERNAL_SIZE_T) / 2))
3363    if (__builtin_expect ((n | elem_size) >= HALF_INTERNAL_SIZE_T, 0))
3364      {
3365        if (elem_size != 0 && bytes / elem_size != n)
3366          {
3367            __set_errno (ENOMEM);
3368            return 0;
3369          }
3370      }
3371
```

Mitigation techniques

glibc

```
3348  void *
3349  __libc_calloc (size_t n, size_t elem_size)
3350  {
3351    mstate av;
3352    mchunkptr oldtop, p;
3353    INTERNAL_SIZE_T bytes, sz, csz, oldtopsize;
3354    void *mem;
3355    unsigned long clearsize;
3356    unsigned long nclears;
3357    INTERNAL_SIZE_T *d;
3358
3359    /* size_t is unsigned so the behavior on overflow is defined.  */
3360    bytes = n * elem_size;
3361  #define HALF_INTERNAL_SIZE_T \
3362    (((INTERNAL_SIZE_T) 1) << (8 * sizeof (INTERNAL_SIZE_T) / 2))
3363    if (__builtin_expect ((n | elem_size) >= HALF_INTERNAL_SIZE_T, 0))
3364      {
3365        if (elem_size != 0 && bytes / elem_size != n)
3366          {
3367            __set_errno (ENOMEM);
3368            return 0;
3369          }
3370      }
3371
```

Mitigation techniques

Embedded Artistry libc

```
/*
 * This is sqrt(SIZE_MAX+1), as s1*s2 <= SIZE_MAX
 * if both s1 < MUL_NO_OVERFLOW and s2 < MUL_NO_OVERFLOW
 */
#define MUL_NO_OVERFLOW (1UL << (sizeof(size_t) * 4))

void* calloc(size_t num, size_t size)
{
        /* num * size unsigned integer wrapping check */
        if((num >= MUL_NO_OVERFLOW || size >= MUL_NO_OVERFLOW) && num > 0 && SIZE_MAX / num < size)
        {
                return NULL;
        }
```

Mitigation techniques

Embedded Artistry libc

```
/*
 * This is sqrt(SIZE_MAX+1), as s1*s2 <= SIZE_MAX
 * if both s1 < MUL_NO_OVERFLOW and s2 < MUL_NO_OVERFLOW
 */
#define MUL_NO_OVERFLOW (1UL << (sizeof(size_t) * 4))

void* calloc(size_t num, size_t size)
{
        /* num * size unsigned integer wrapping check */
        if((num >= MUL_NO_OVERFLOW || size >= MUL_NO_OVERFLOW) && num > 0 && SIZE_MAX / num < size)
        {
                return NULL;
        }
```

1<<16 = 65536

SIZE_MAX = 0xffffffff

Mitigation techniques

musl libc

```
32
33  void *calloc(size_t m, size_t n)
34  {
35          if (n && m > (size_t)-1/n) {
36                  errno = ENOMEM;
37                  return 0;
38          }
```

Mitigation techniques

musl libc

```
32
33  void *calloc(size_t m, size_t n)
34  {
35          if (n && m > (size_t)-1/n) {
36                  errno = ENOMEM;
37                  return 0;
38          }
```

Apply

- RTOS vendors:
  - Map allocation functions such as malloc or calloc.
  - Option 1(preferred) - Add a check for integer overflow on requested allocation.
  - Option 2 – limit max allocation size to prevent possible overflows.
- RTOS users/application developers:
  - Check the advisory ICSA-21-119-04 to see if your system is effected.
  - Option 1(preferred) - Update your system and apply patches from vendor.
  - Option 2 – Update your system and apply patches from vendor.

Advisory

- CVSS 9.8
- 28 CVE

CYBERSECURITY & INFRASTRUCTURE SECURITY AGENCY

Search

CISA.gov | Services | Report

Alerts and Tips | Resources | Industrial Control Systems

Industrial Control Systems > ICS-CERT Advisories > Multiple RTOS (Update D)

ICS Advisory (ICSA-21-119-04)

More ICS-CERT Advisories

Multiple RTOS (Update D)

Original release date: November 30, 2021 | Last revised: January 31, 2022

Print | Tweet | Send | Share

Legal Notice

All information products included in https://us-cert.cisa.gov/ics are provided "as is" for informational purposes only. The Department of Homeland Security (DHS) does not provide any warranties of any kind regarding any information contained within. DHS does not endorse any commercial product or service, referenced in this product or otherwise. Further dissemination of this product is governed by the Traffic Light Protocol (TLP) marking in the header. For more information about TLP, see https://us-cert.cisa.gov/tlp/.

1. EXECUTIVE SUMMARY

- **CVSS v3 9.8**
- **ATTENTION:** Exploitable remotely/low attack complexity
- **Vendors:** Multiple
- **Equipment:** Multiple
- **Vulnerabilities:** Integer Overflow or Wraparound

CISA is aware of a public report, known as "BadAlloc" that details vulnerabilities found in multiple real-time operating systems (RTOS) and supporting libraries. CISA is issuing this advisory to provide early notice of the reported vulnerabilities and identify baseline mitigations for reducing risks to these and other cybersecurity attacks.

The various open-source products may be implemented in forked repositories.

2. UPDATE INFORMATION

This updated advisory is a follow-up to the advisory update titled ICSA-21-119-04 Multiple RTOS (Update C) that was published August 17, 2021, to the ICS webpage on us-cert.cisa.gov.

3. RISK EVALUATION

Successful exploitation of these vulnerabilities could result in unexpected behavior such as a crash or a remote code injection/execution.

4. TECHNICAL DETAILS

4.1 AFFECTED PRODUCTS

- Amazon FreeRTOS, Version 10.4.1
- Apache Nuttx OS, Version 9.1.0
- ARM CMSIS-RTOS2, versions prior to 2.1.3
- ARM Mbed OS, Version 6.3.0
- ARM mbed-ualloc, Version 1.3.0
- BlackBerry QNX SDP Versions 6.5.0 SP1 and earlier
- BlackBerry QNX OS for Safety Versions 1.0.1 and earlier safety products compliant with IEC 61508 and/or ISO 26262
- BlackBerry QNX OS for Medical Versions 1.1 and earlier safety products compliant with IEC 62304
- A full list of affected QNX products and versions is available here

Microsoft

Q&A

https://msrc-blog.microsoft.com/2021/04/29/badalloc-memory-allocation-vulnerabilities-could-affect-wide-range-of-iot-and-ot-devices-in-industrial-medical-and-enterprise-networks/

Bing for ”ICSA-21-119-04”