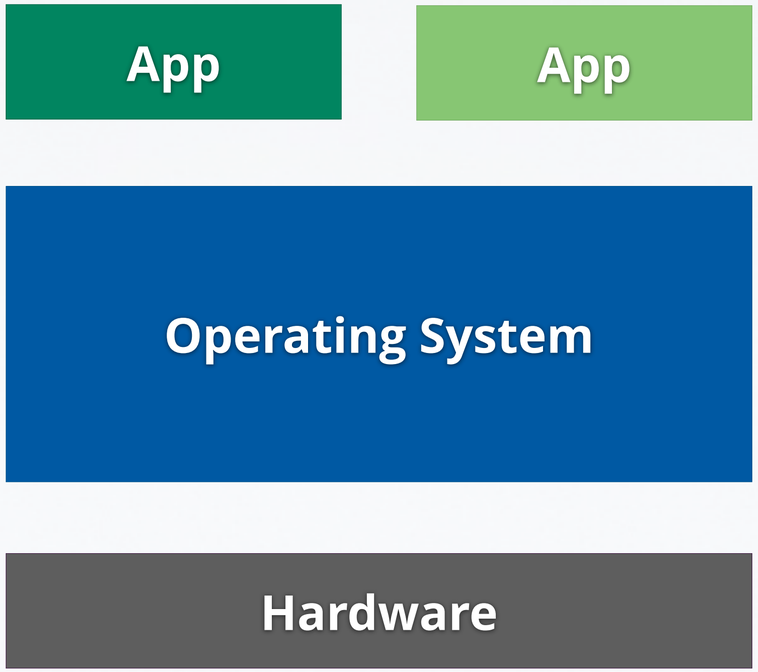
TECHNISCHE UNIVERSITÄT DRESDEN

Faculty of Computer Science Institute of Systems Architecture, Operating Systems Group

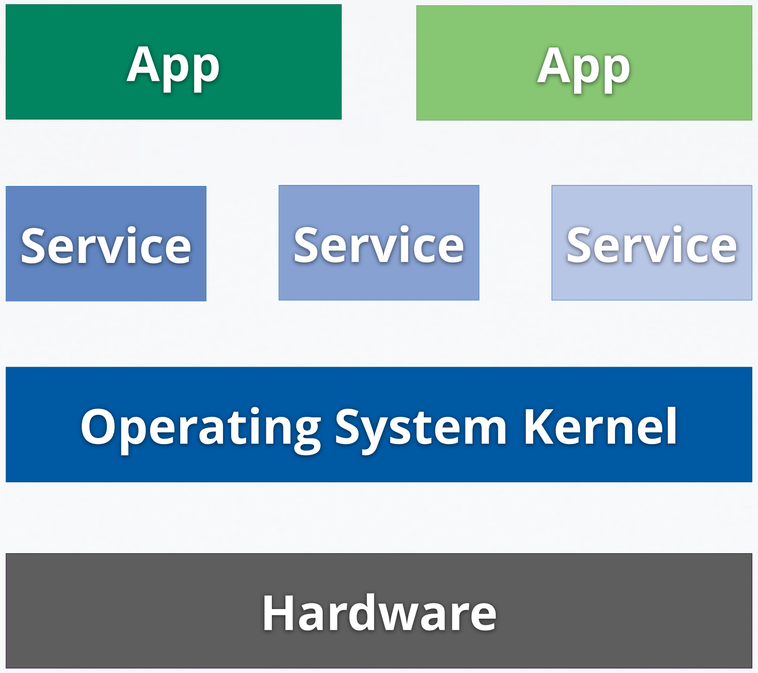
ISOLATION, INTERFACES, AND SANDBOXING

CARSTEN WEINHOLD

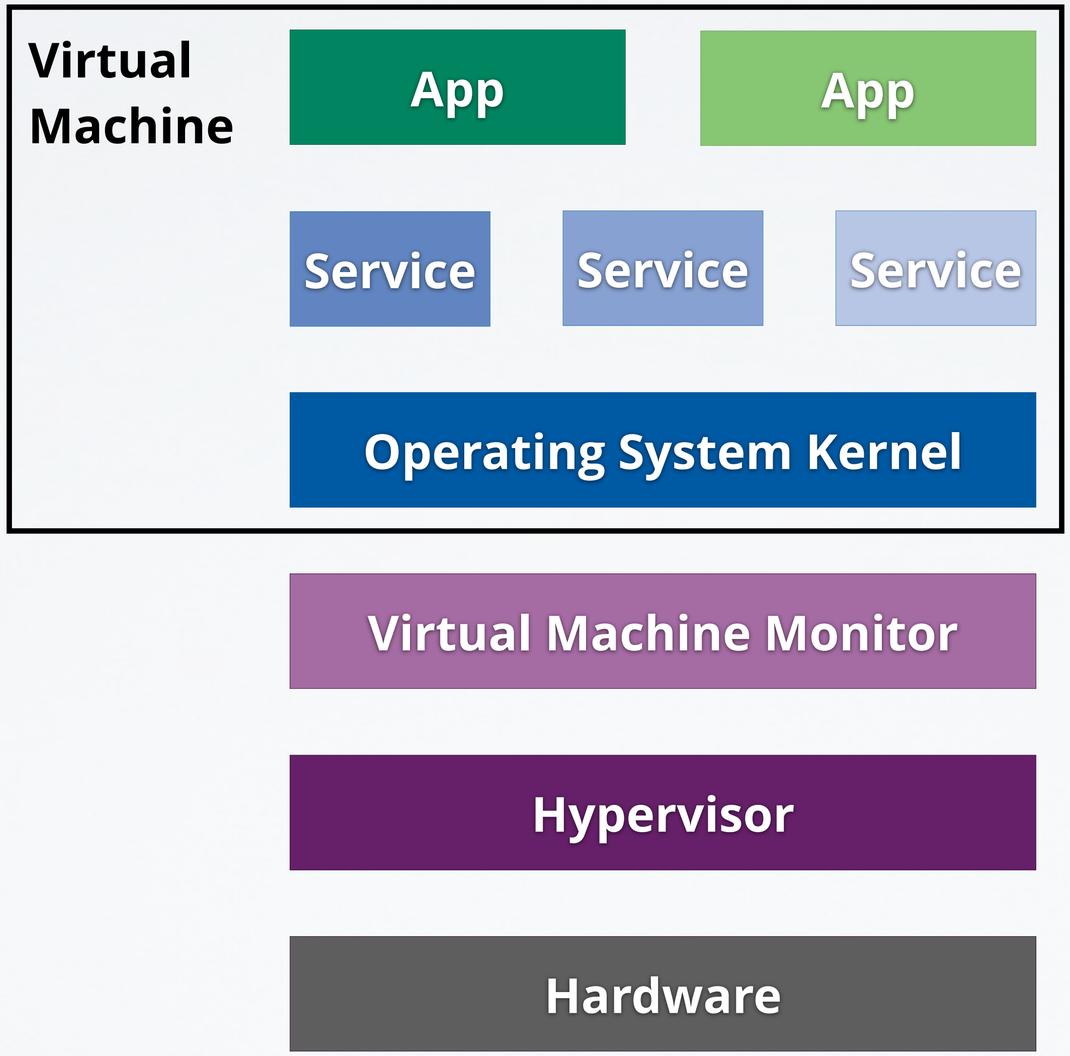
APP ISOLATION



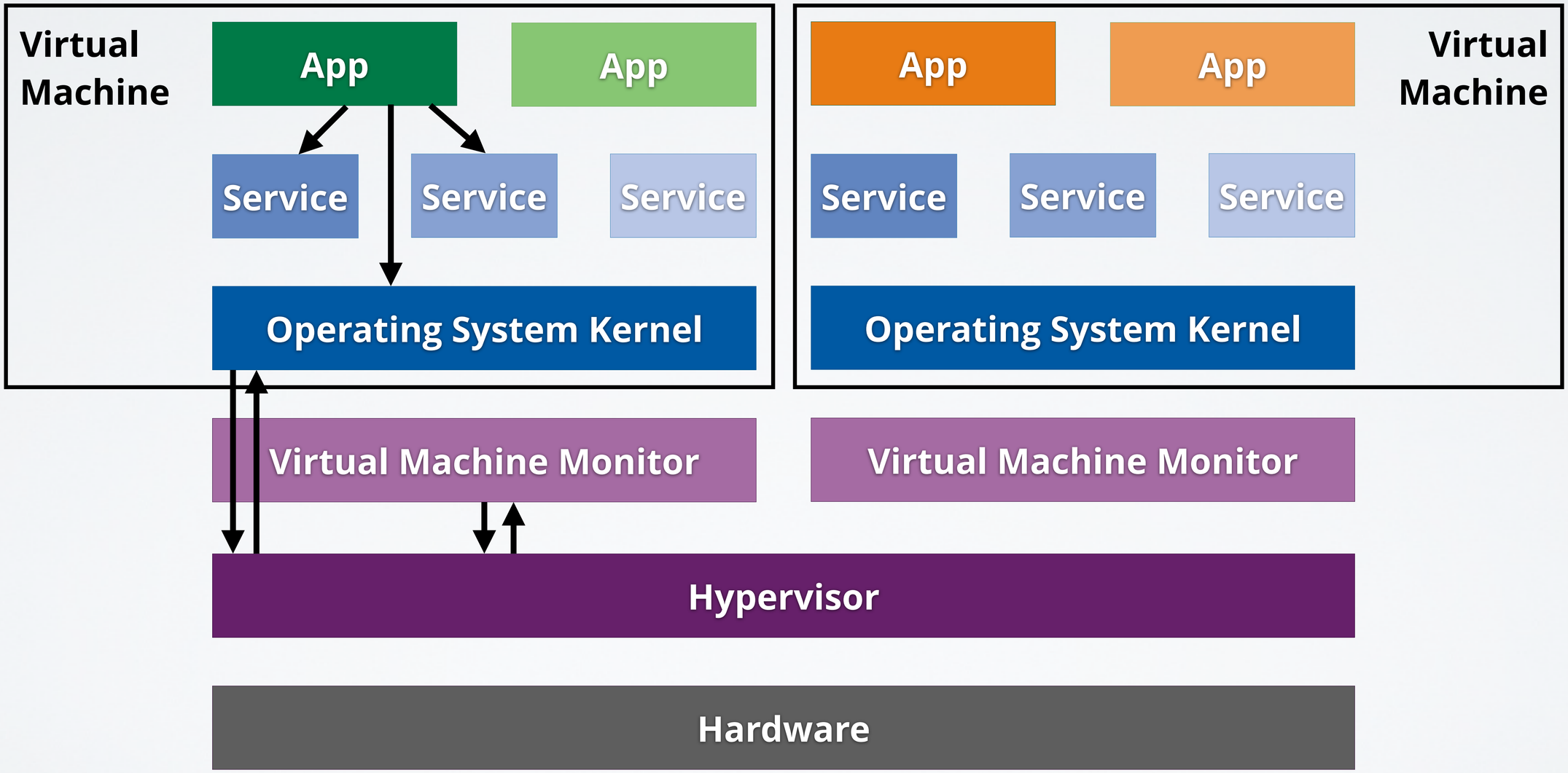
OS/SERVICE ISOLATION



VM-BASED ISOLATION

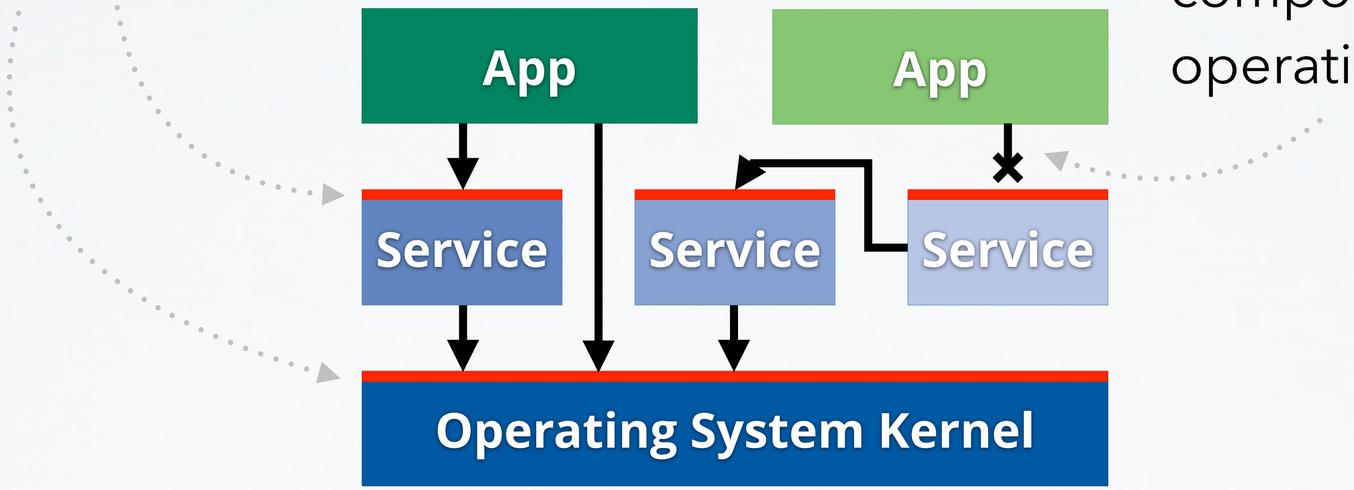


VM-BASED ISOLATION



INTERFACES

Isolated components interact with each other through interfaces.

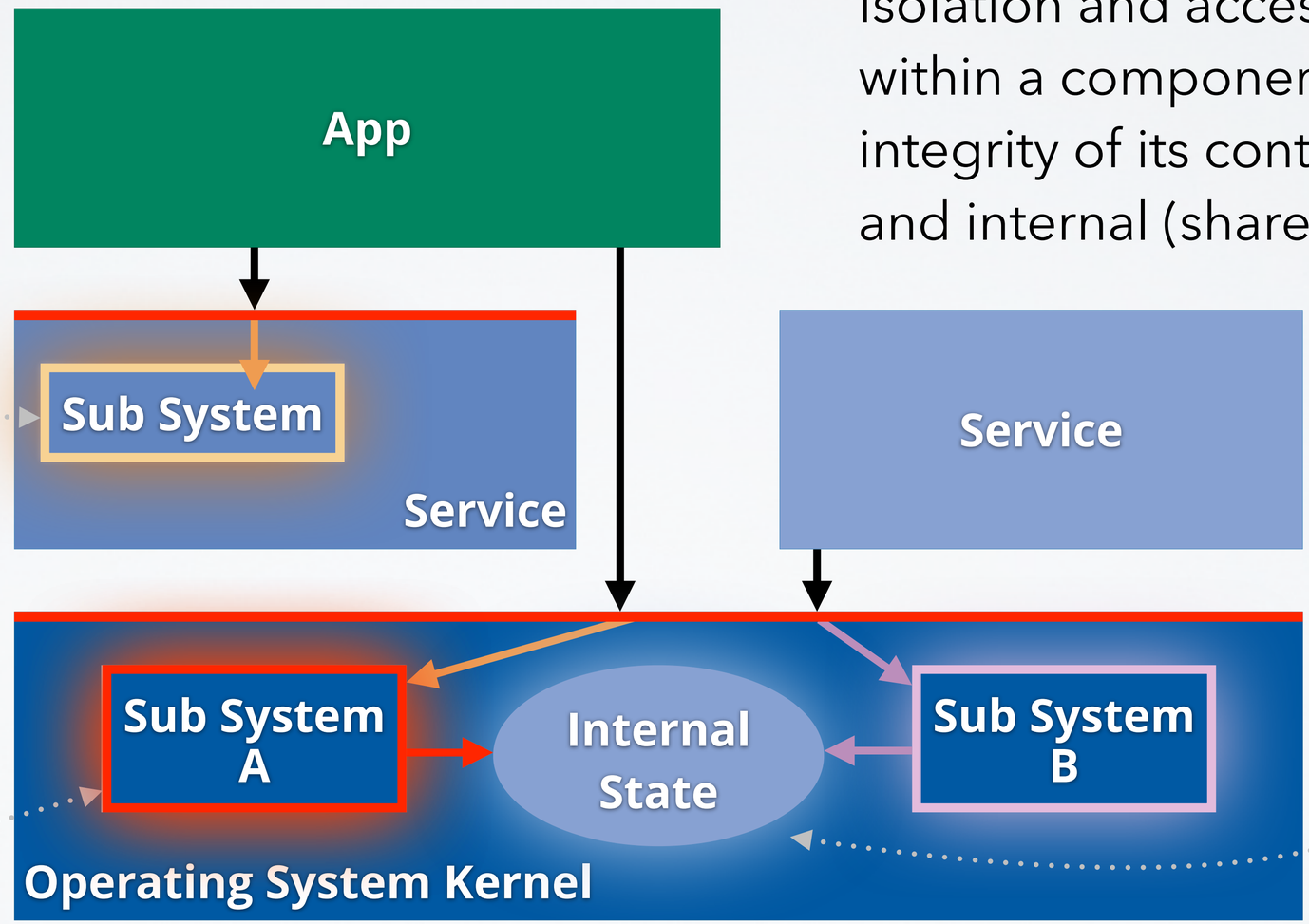


Some access control can be enforced at interface (via hardware or by a more privileged component like the operating system kernel).

ATTACK SURFACE

Interfaces shall limit access to internal sub systems.

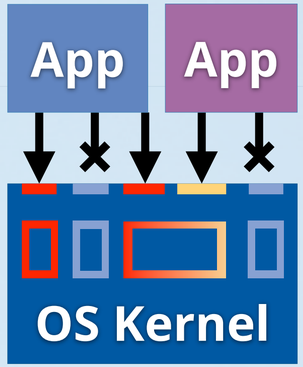
Vulnerabilities in sub systems may expose internal state.

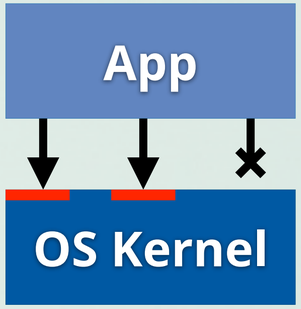


Isolation and access control within a component rely on integrity of its control flow and internal (shared) state.

SANDBOXING

- Sandboxes restrict programs such they can only access a (minimal) subset of interfaces or system-level objects
  - **Namespaces:** BSD jails, Linux containers, …
  - **System-call filters:** SELinux, Seatbelt, …
  - **Voluntarily:** drop root rights, Linux seccomp, OpenBSD pledge, FreeBSD capsicicum, …
- Can be combined with program splitting (e.g., render processes in web browsers)





TECHNISCHE UNIVERSITÄT DRESDEN

Faculty of Computer Science Institute of Systems Architecture, Operating Systems Group

IN-THE-WILD IOS EXPLOIT CHAIN

Discussion of Google Project Zero Blog Post

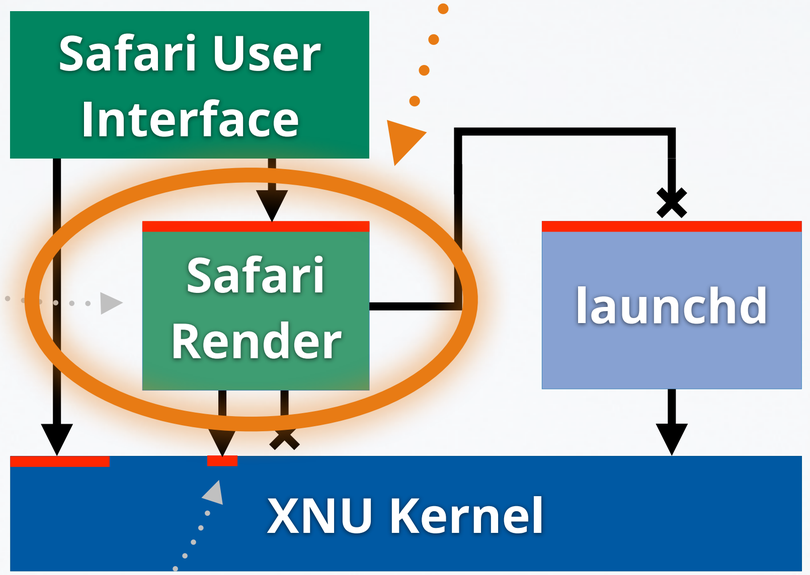
CARSTEN WEINHOLD

IOS BROWSER SANDBOX

Safari web browser is split into multiple processes, with "render processes" being sandboxed.

Nothing can go wrong here ... ?

Other processes run with higher privileges than the web browser, but they are isolated.



The XNU kernel implements part of the graphics driver that the render process is allowed to use.

The XNU kernel enforces sandbox restrictions and all other isolation.

DISCUSSION OF BLOG POST

LESSONS LEARNED

- Often more than one component has to be attacked
- Multiple bugs may may be necessary to gain full access:
  1. Find bug in web browser (or another app) and exploit it
  2. Interact with kernel (of another privileged component) and exploit bug in it to escape from sandbox
- Other exploit chains could require jumping from one sandboxed process to another, before exploiting a privilege escalation bug
- **Sandboxing makes attacks harder, but not impossible**
- If one exploit in the chain does not work (or a component has no bug), it will break the exploit chain

REFERENCES

Original source:

https://googleprojectzero.blogspot.com/2019/08/in-wild-ios-exploit-chain-1.html

Annotated version for this lecture:

Part 1, Part 2