

---

# Preventing Kernel Hacks with HAKC

Derrick McKee, Yianni Giannaris, Carolina Ortega, Howard Shrobe,  
Mathias Payer, Hamed Okhravi, and Nathan Burow

NDSS 2022



**DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited.** This material is based upon work supported by the Under Secretary of Defense for Research and Engineering under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Under Secretary of Defense for Research and Engineering. © 2022 Massachusetts Institute of Technology. Delivered to the U.S. Government with Unlimited Rights, as defined in DFARS Part 252.227-7013 or 7014 (Feb 2014). Notwithstanding any copyright notice, U.S. Government rights in this work are defined by DFARS 252.227-7013 or DFARS 252.227-7014 as detailed above. Use of this work other than as specifically authorized by the U.S. Government may violate any copyrights that exist in this work.

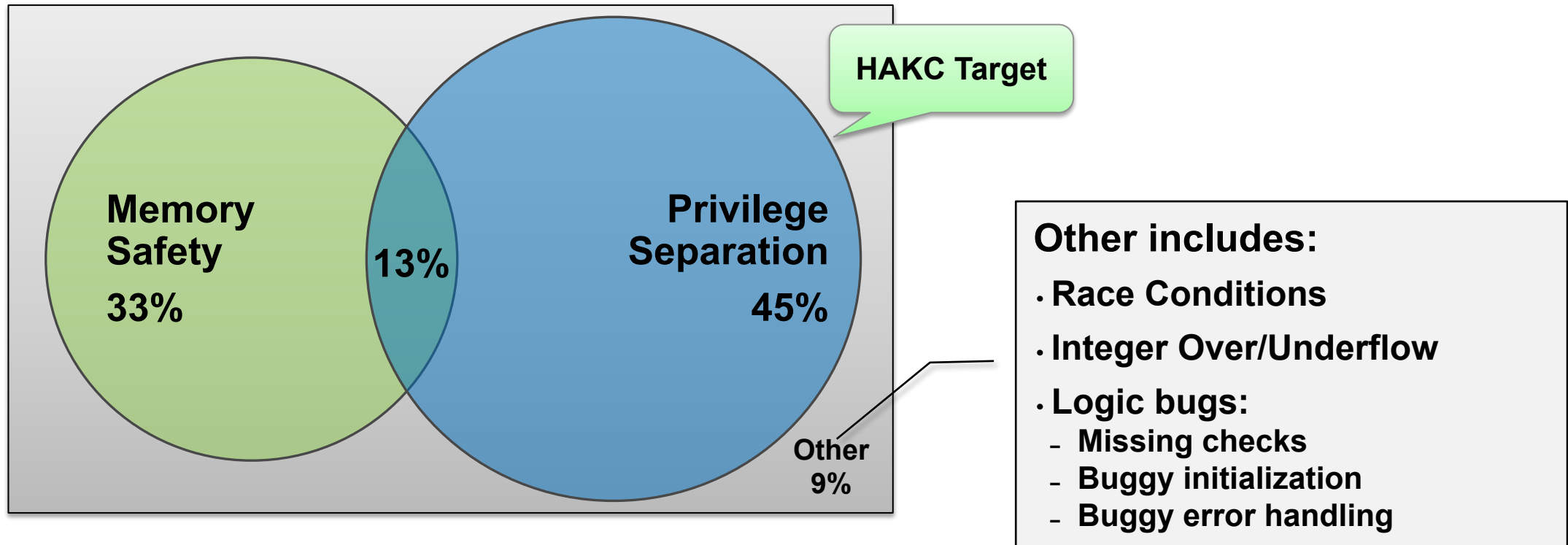
---



# What is the Largest Class of OS Vulnerabilities?



We analyzed the past 5 years of vulnerabilities in Linux: **508** with *critical* or *high* severity



**Privilege Separation mitigates the highest percentage of Linux CVEs.**



# What is HAKC?

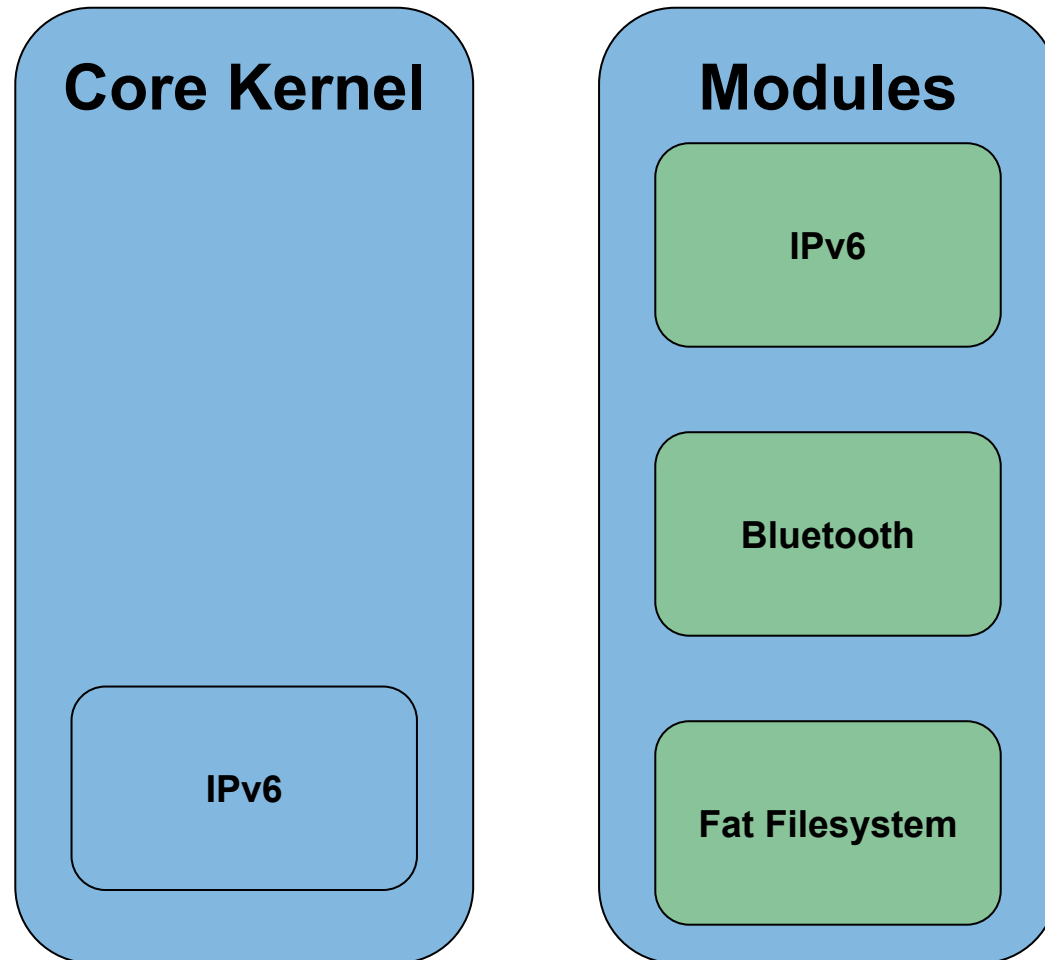


- **Hardware Assisted Kernel Compartmentalization (HAKC)**
- **Enforcement mechanism for compartmentalization policies**
- **Runs on bare metal without virtualization by using new hardware extensions**
  - PAC computes a hash of a pointer and user-specified context
  - MTE colors an address range one of 16 colors
- **HAKC restricts bugs to their compartments, which limits their reach**

HAKC enforces compartmentalization to prevent the most common class of bugs.



# Linux Today





# Kernel CVE Example



```
void release(struct entry *e) {  
    if(e->offset < OFFSET_MAX) {  
        int *i = e->buffer + e->offset;  
        e->cb(i);  
        *i -= 1;  
    }  
}
```

Attacker controlled

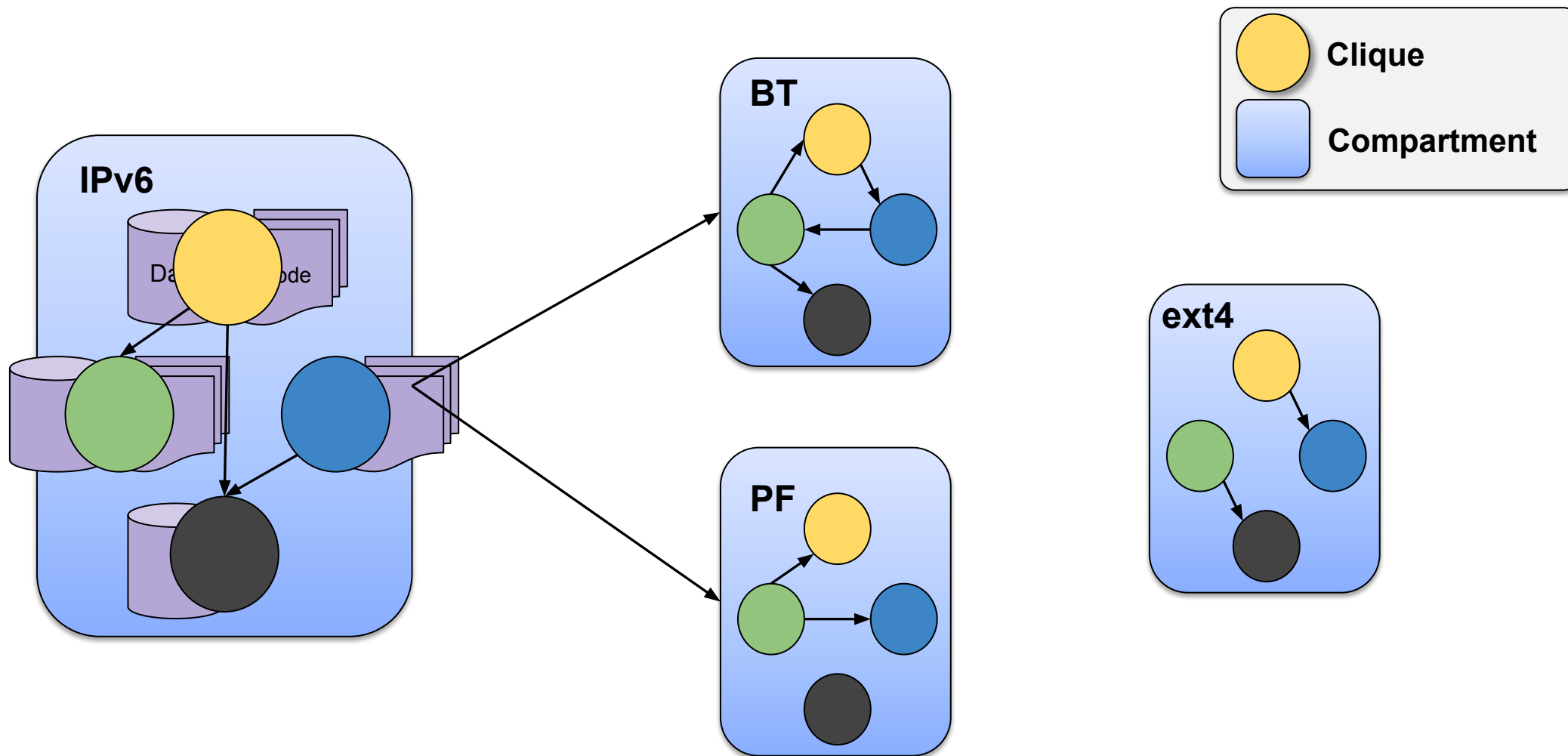
Missing lower bound check!

i is any arbitrary kernel data

- Adapted from CVE-2016-4997

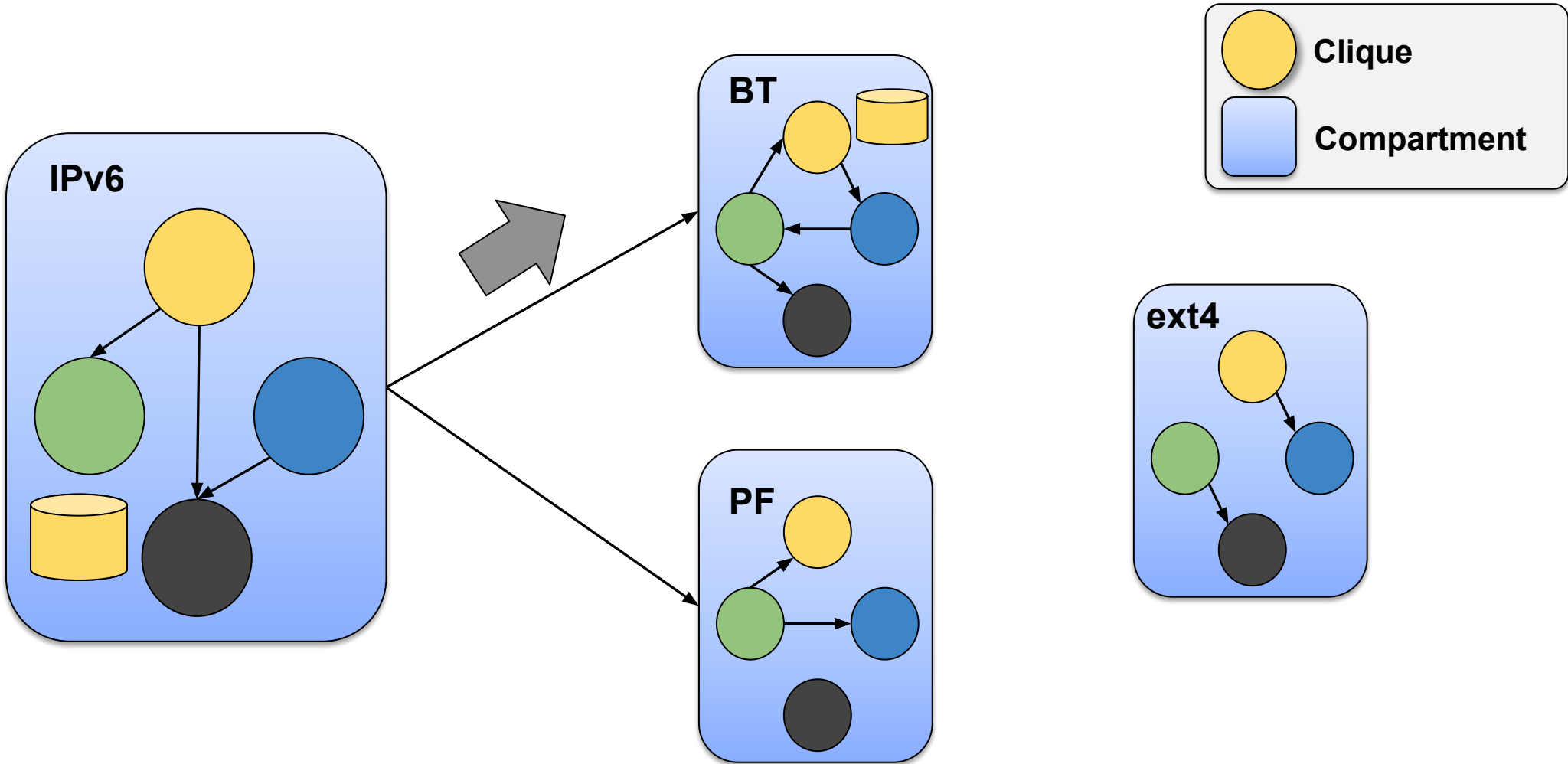


# How Does HAKC Work?





# How Does HAKC Work?





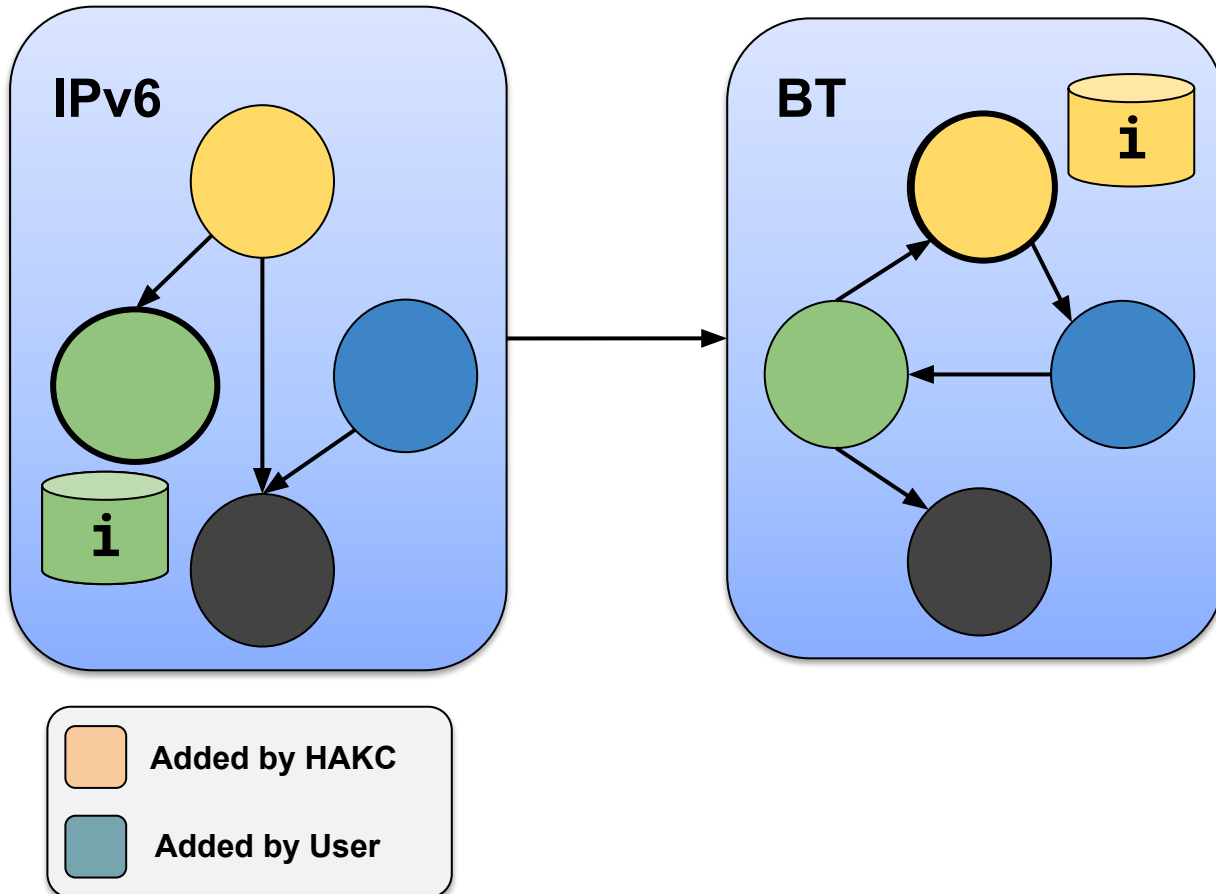
# How Does HAKC Work?



```

void release(struct entry *e) {
    if(e->offset < OFFSET_MAX) {
        int *i = e->buffer + e->offset;
        HAKC Data Check for i
        HAKC Compartment Transition Check
        HAKC Data Ownership Transfer to e->cb
        e->cb(i);
        HAKC Data Ownership Reversal
        *i -= 1;
    }
}

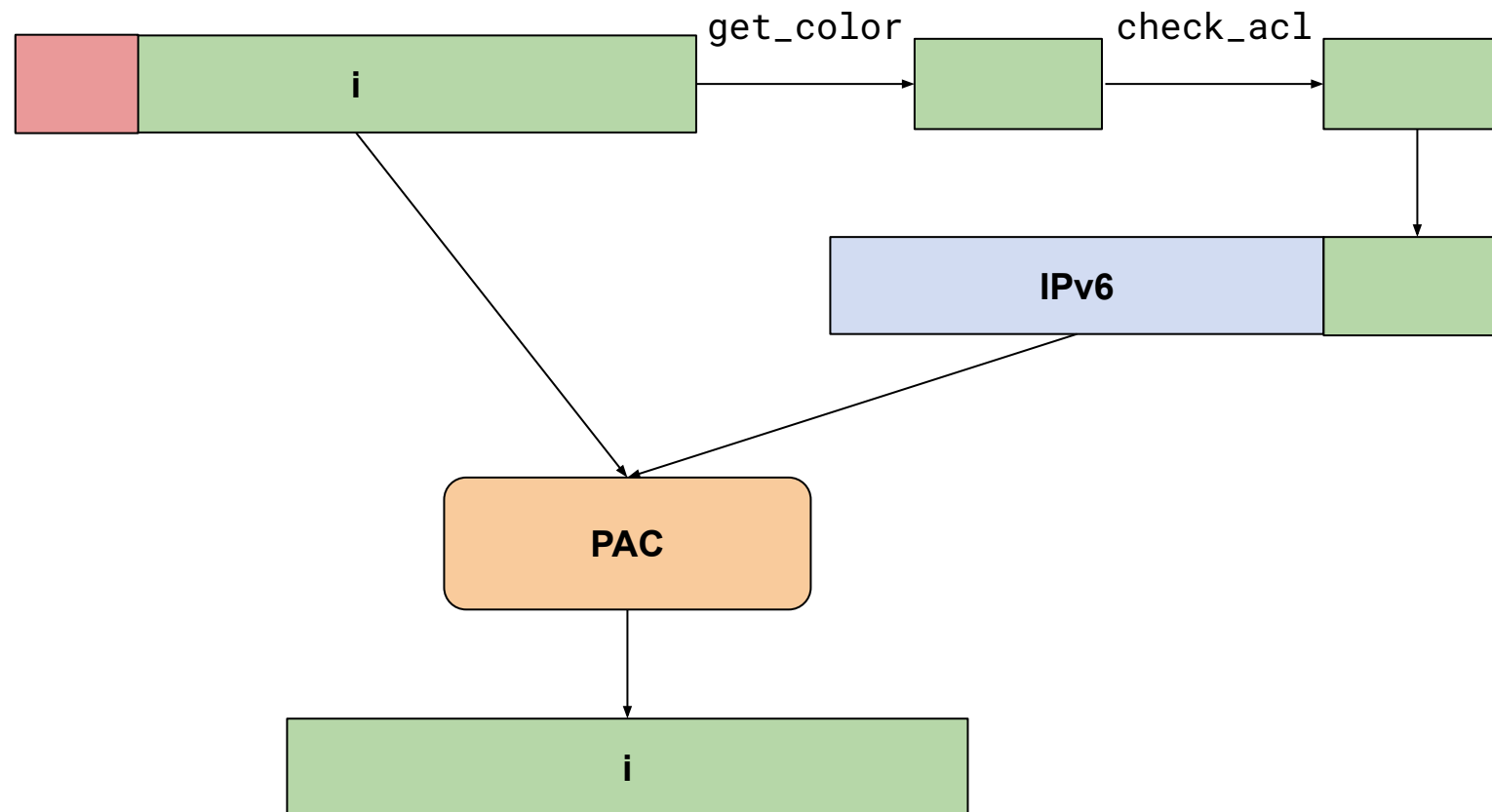
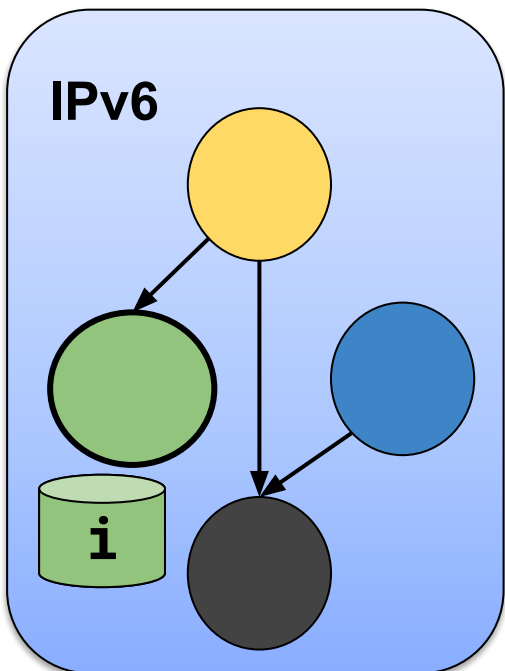
```







# HAKC Checks

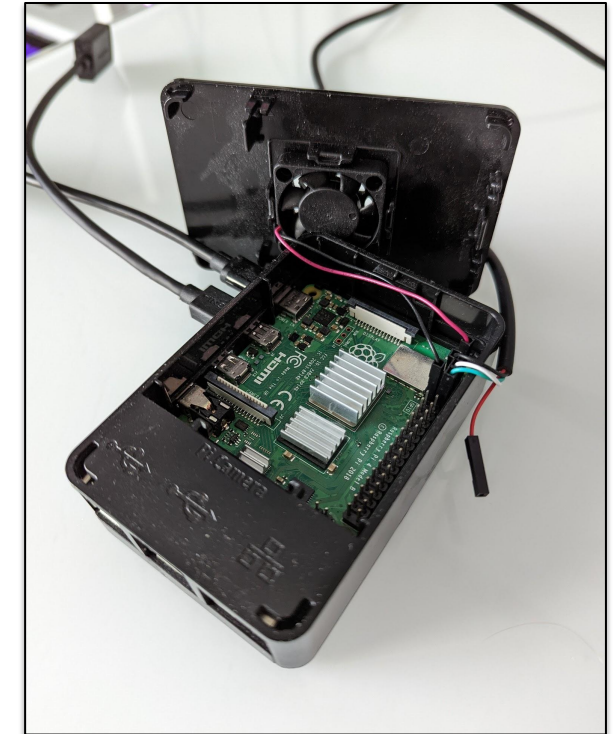




# HAKC Experimental Results

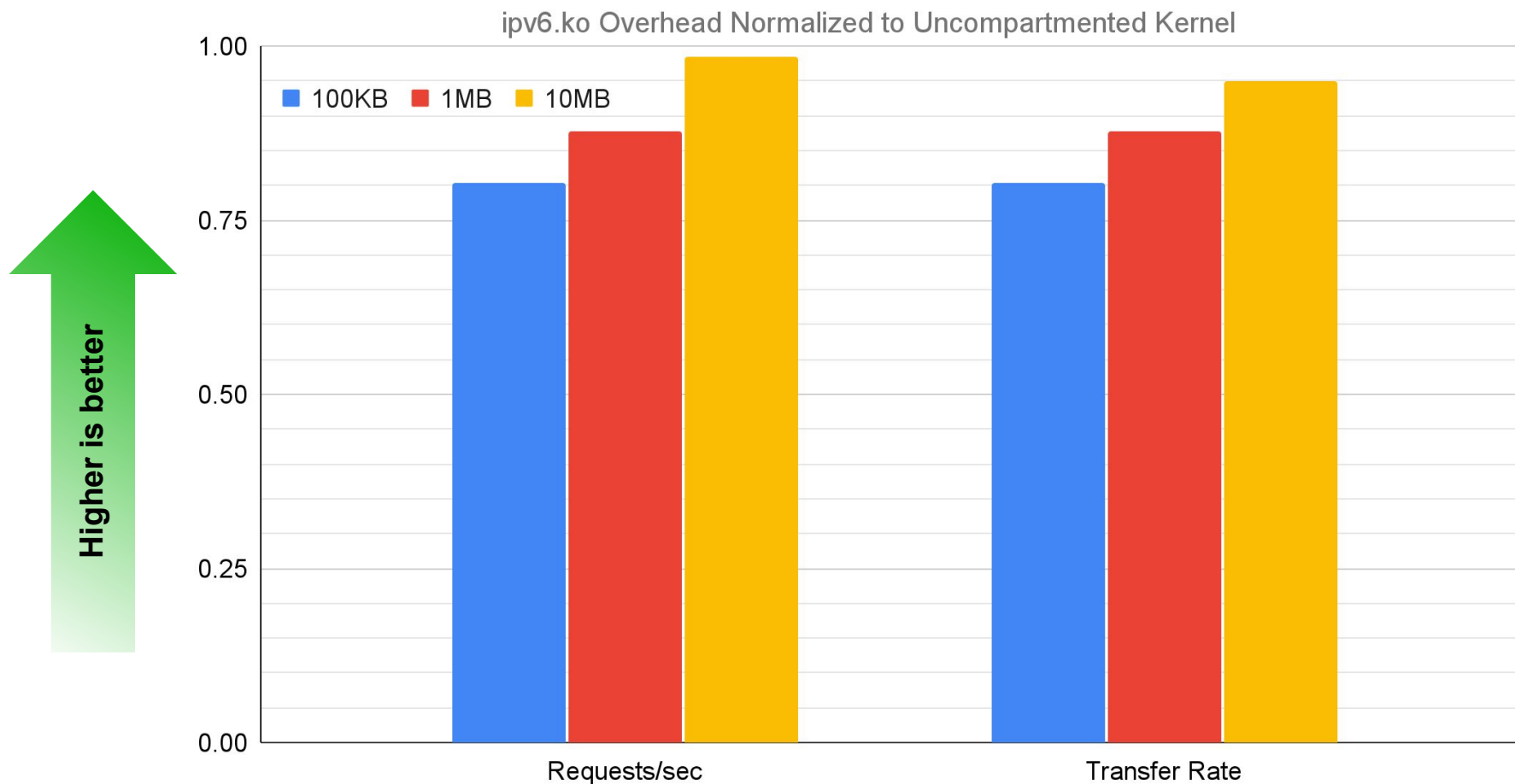


- **Compartmentalized the IPv6 and one of the packet filtering modules**
- **Measured the performance using Apachebench**
- **User simulation study browsing popular websites**





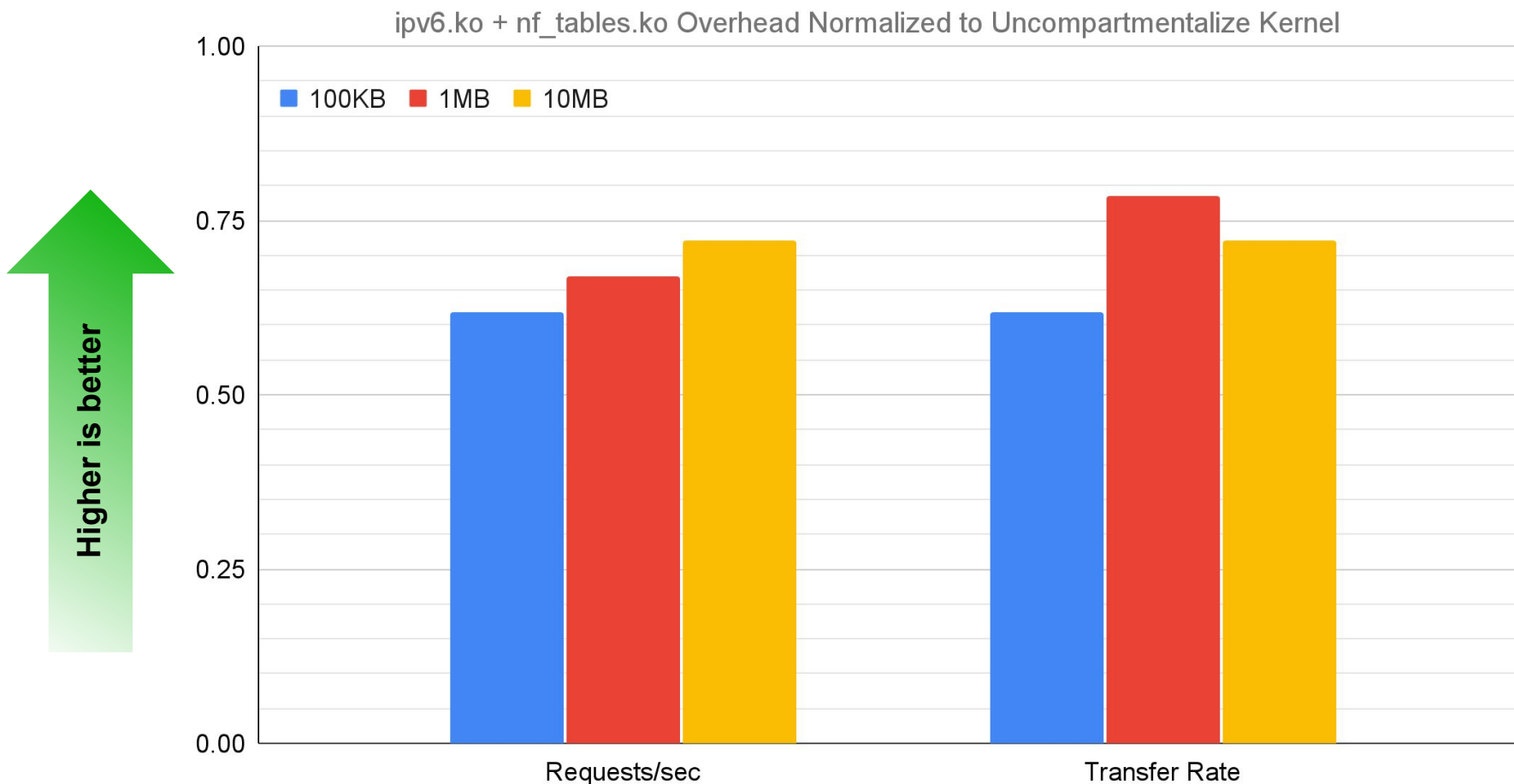
# Single Module Apachebench Overhead



**HAKC imposes a low performance overhead of  $\leq 20\%$ .**



# Multiple Module Apachebench Overhead



**Overhead increases linearly with Compartment count in the worst case.**



# User Browsing Overhead



Website	Load Time Delta (s)	Stdev (s)
linkedin.com	-0.47	0.065
hdfcbank.com	-0.12	0.085
google.cn	-0.068	0.086
bing.com	-0.087	0.13
investing.com	38	62
okezone.com	-11	20
cnn.com	-9.8	15
yahoo.com	-4.9	15

HAKC Load time is  
**1.19 ± 4.34s slower**

**Users browsing the internet with a HAKC protected IPv6 module will notice no difference.**



# HAKC Summary



- **Novel two-level enforcement mechanism for arbitrary compartmentalization policies**
- **By utilizing new hardware extensions, HAKC does not require any virtualization layer or trusted monitor**
- **HAKC enforces compartments that provide strong protection for data and control-flow at low overhead**
- **Available at <https://github.com/mit-ll/HAKC>**
- **Contact**
  - **Email: [derrick.mckee@gmail.com](mailto:derrick.mckee@gmail.com)**
  - **Twitter: [@unbound\\_brewer](https://twitter.com/unbound_brewer)**