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.

- Memory Safety: 33%
- Privilege Separation: 45%
- Other: 9%

Other includes:
- Race Conditions
- Integer Over/Underflow
- Logic bugs:
  - Missing checks
  - Buggy initialization
  - Buggy error handling

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.
Kernel CVE Example

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

- Adapted from CVE-2016-4997

**i** is any arbitrary kernel data

Missing lower bound check!
How Does HAKC Work?

IPv6

BT

PF

Clique

Compartment

ext4
How Does HAKC Work?

IPv6

BT

PF

ext4

Clique

Compartment

IPv6

BT

PF

ext4
void release(struct entry *e) {
    if(e->offset < OFFSET_MAX) {
        int *i = e->buffer + e->offset;
        *i -= 1;
    }
    e->cb(i);
}
HAKC Checks

IPv6

get_color

check_acl

IPv6

PAC

i
HAKC Experimental Results

- Compartmentalized the IPv6 and one of the packet filtering modules
- Measured the performance using Apachebench
- User simulation study browsing popular websites
HAKC imposes a low performance overhead of ≤ 20%.
Overhead increases linearly with Compartment count in the worst case.
## User Browsing Overhead

<table>
<thead>
<tr>
<th>Website</th>
<th>Load Time Delta (s)</th>
<th>Stdev (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>linkedin.com</td>
<td>-0.47</td>
<td>0.065</td>
</tr>
<tr>
<td>hdfcbank.com</td>
<td>-0.12</td>
<td>0.085</td>
</tr>
<tr>
<td>google.cn</td>
<td>-0.068</td>
<td>0.086</td>
</tr>
<tr>
<td>bing.com</td>
<td>-0.087</td>
<td>0.13</td>
</tr>
<tr>
<td>investing.com</td>
<td>38</td>
<td>62</td>
</tr>
<tr>
<td>okezone.com</td>
<td>-11</td>
<td>20</td>
</tr>
<tr>
<td>cnn.com</td>
<td>-9.8</td>
<td>15</td>
</tr>
<tr>
<td>yahoo.com</td>
<td>-4.9</td>
<td>15</td>
</tr>
</tbody>
</table>

HAKC Load time is $1.19 \pm 4.34$s 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
  – Twitter: @unbound_brewer