



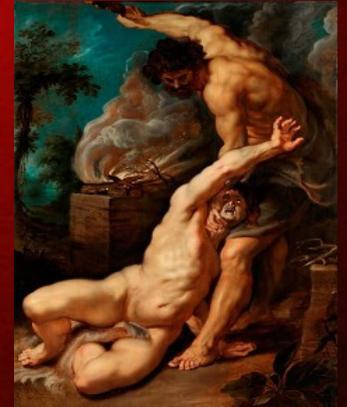
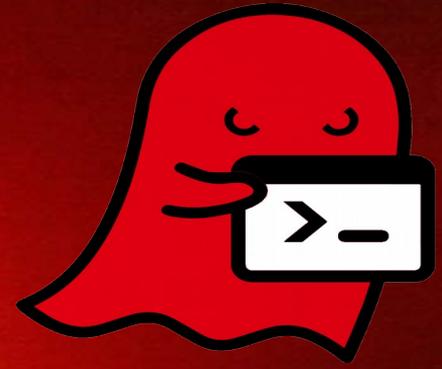
hexhive

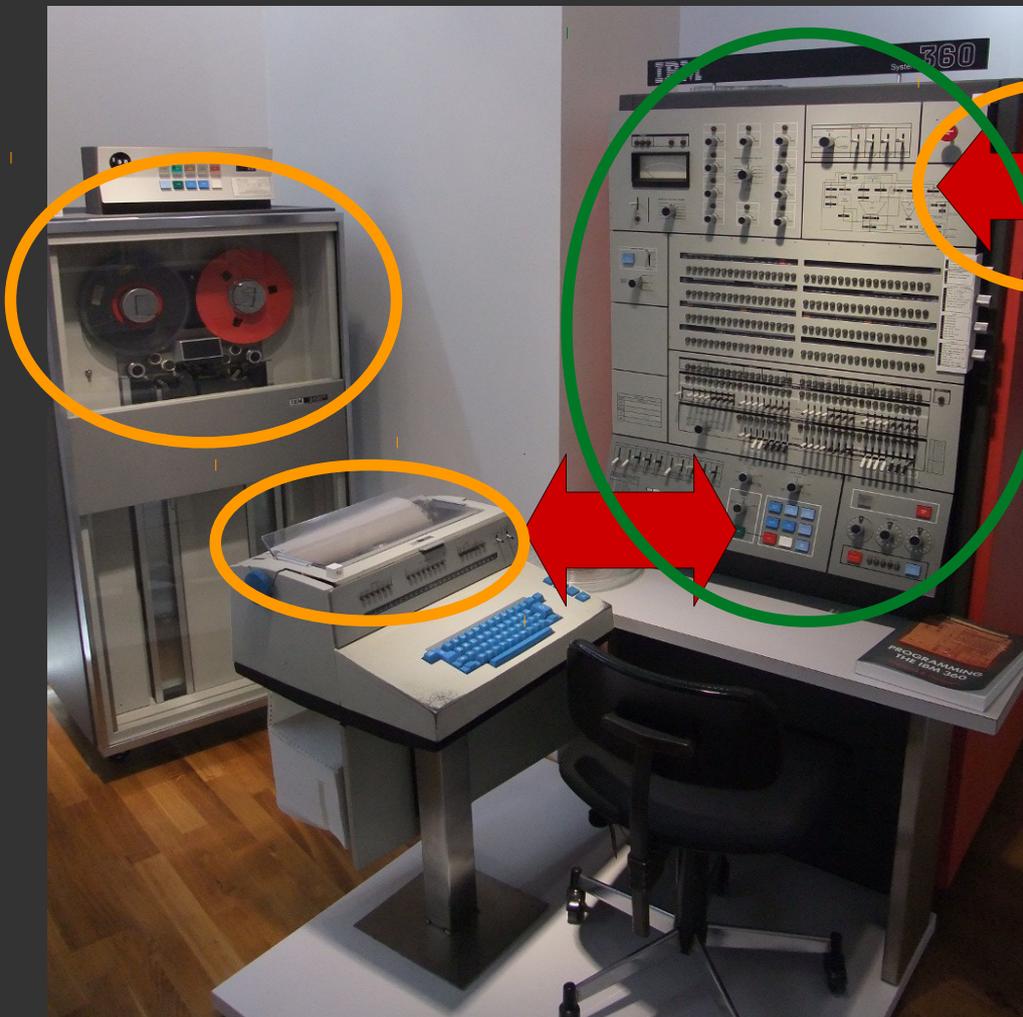
HexPADS: a platform to detect “stealth” attacks

Mathias Payer (@gannimo), Purdue University
<http://hexhive.github.io>



Deployed defenses
focus on memory
corruption



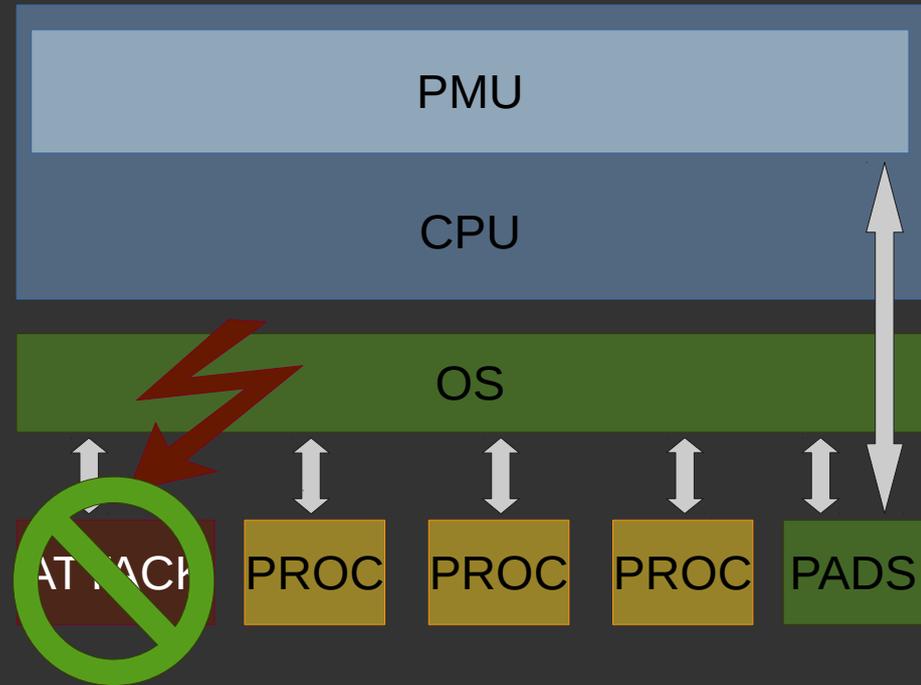


Consider
program state
and behavior

HexPADS Design

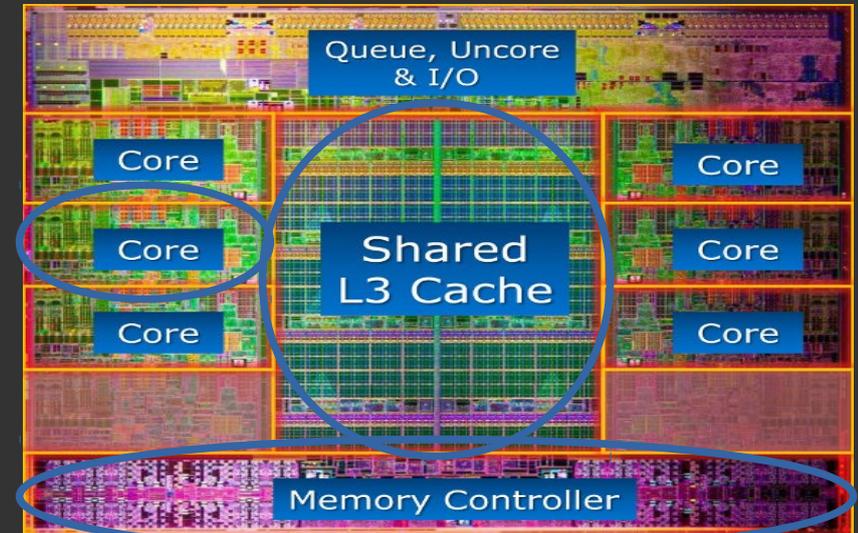
HexPADS Design

- Host-based Intrusion/Attack Detection System
- Measure fine-grained process-level *runtime* behavior
 - Operating system provides basic runtime characteristics
 - Performance Monitoring Unit (PMU) allows counting/sampling of detailed and fine-grained events
- Detect attacks based on signatures/anomalies
- Take evasive action/counter measure



Default Metrics (always collected)

- Number of executed instructions
- Number of last level cache accesses
- Number of last level cache misses
- Minor/major page faults
- Execution time



(c) Intel

Additional Metrics

- Anything in /proc
 - Opened files, network ports, and IPC
 - Loaded libraries
 - Memory maps
- Any measurable PMU event
 - Memory/cache hierarchy events
 - Instruction mix and behavior
 - Execution profile and branch records
- System calls

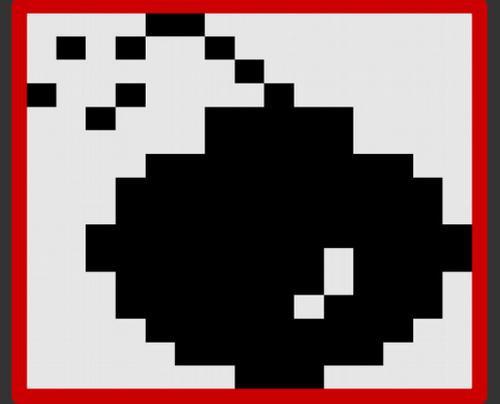
```
1 [ 0.7% ] 5 [ 0.7% ]
2 [ 0.7% ] 6 [ 0.7% ]
3 [ 0.0% ] 7 [ 1.3% ]
4 [ 1.3% ] 8 [ 0.0% ]
Mem[|||||] 1105/15935MB Tasks: 121, 231 thr: 1 running
Swp[ ] 0/16267MB Load average: 0.34 0.26 0.20
Uptime: 02:23:30
```

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
5035	gannimo	20	0	30252	2344	1448	R	1.3	0.0	0:00.59	htop
2409	gannimo	20	0	1342M	37052	26940	S	1.3	0.2	0:00.10	nautilus -n
1433	root	20	0	13180	980	536	S	0.7	0.0	1:20.43	/usr/sbin/ninja /etc/ninja/ninja.conf
2321	gannimo	20	0	1524M	92708	34236	S	0.7	0.6	2:19.22	complz
1634	root	20	0	663M	117M	98256	S	0.7	0.7	2:40.31	/usr/bin/X -core :0 -seat seat0 -auth /var/run/lightdm/r
2484	gannimo	20	0	880M	42888	26500	S	0.7	0.3	0:17.87	/usr/bin/python /usr/bin/terminator
1932	gannimo	20	0	22304	640	420	S	0.7	0.0	0:03.85	upstart-dbus-bridge --daemon --session --bus-name
2359	gannimo	20	0	403M	13672	8200	S	0.0	0.1	0:33.10	indicator-multiloop
2022	gannimo	20	0	636M	34688	13048	S	0.0	0.2	0:11.76	/usr/lib/unity/unity-panel-service
1978	gannimo	20	0	636M	34688	13048	S	0.0	0.2	0:37.56	/usr/lib/unity/unity-panel-service
2143	gannimo	20	0	280M	5124	3956	S	0.0	0.0	0:10.91	/usr/lib/x86_64-linux-gnu/indicator-application/indicato
2197	gannimo	20	0	280M	5124	3956	S	0.0	0.0	0:05.91	/usr/lib/x86_64-linux-gnu/indicator-application/indicato
1953	gannimo	20	0	286M	10628	2892	S	0.0	0.1	0:42.11	/usr/bin/ibus-daemon --daemonize --xim
1984	gannimo	20	0	286M	10628	2892	S	0.0	0.1	0:24.76	/usr/bin/ibus-daemon --daemonize --xim
4163	gannimo	20	0	1704M	154M	84476	S	0.0	1.0	0:18.03	/usr/lib/libreoffice/program/soffice.bin hexpads.ods --s
2594	gannimo	20	0	568M	9112	5308	S	0.0	0.1	0:00.11	zeitgeist-datahub
4488	gannimo	20	0	1167M	281M	56508	S	0.0	1.8	0:31.85	/usr/lib/thunderbird/thunderbird
1896	gannimo	20	0	42732	4836	928	S	0.0	0.0	0:13.33	dbus-daemon --fork --session --address=unix:abstract=/tm

F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice F8Nice F9Kill F10Quit

Implementation

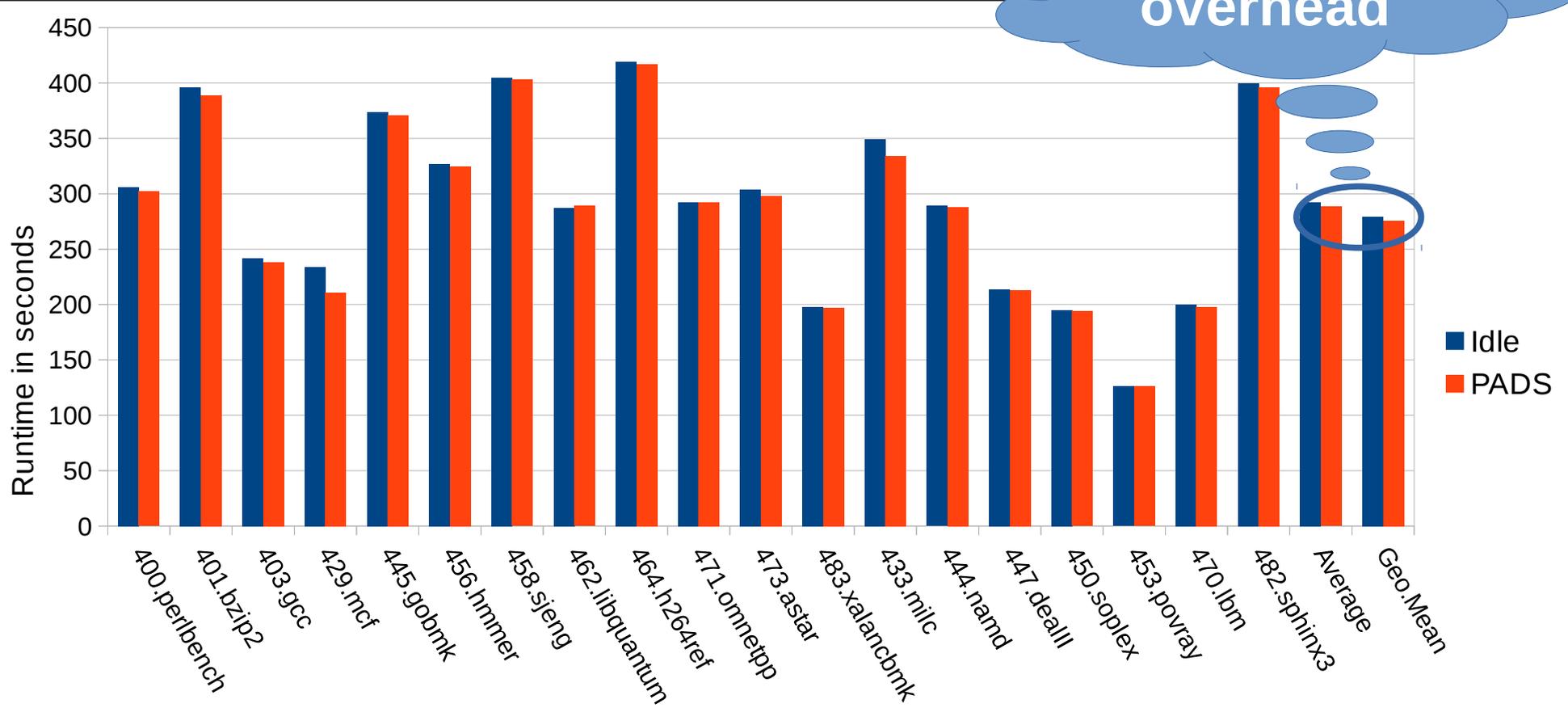
- Modular implementation
- Collect metrics for all processes
- Keep configurable history
- Run detection modules every iteration



<http://github.com/HexHive/HexPads>

Evaluation

SPEC CPU2006



Rowhammer

- Cause DRAM bit flips by accessing adjacent cells
 - High amount of cache misses: $> 500,000/s$
 - High cache miss rate: $> 70\%$
 - Low page fault rate: $< 1\%$
- Possible extension: use sampling
 - Detect and correlate actual accesses
 - Detect “nearby” accesses



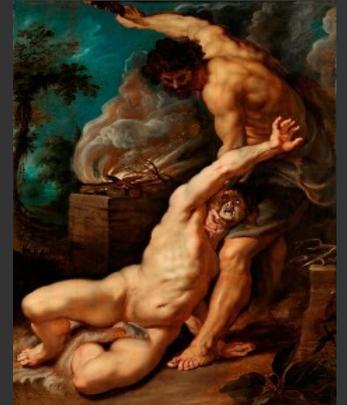
Cache-based side/covert channels

- Communicate through access timing
 - Same pattern as rowhammer
 - Additional challenge: which process is bad?
- Possible extension: longer history
 - Consider development over time



Cross-VM ASL INtrospection (CAIN)*

- CAIN attacks leak ASLR base addresses in co-located VMs
 - High amount of page faults/allocated pages/cache misses/per instr.
 - Followed by inactivity
- Possible extension: study access patterns
 - Push detection to VMM level
 - Check page similarity
 - Evaluate page access patterns

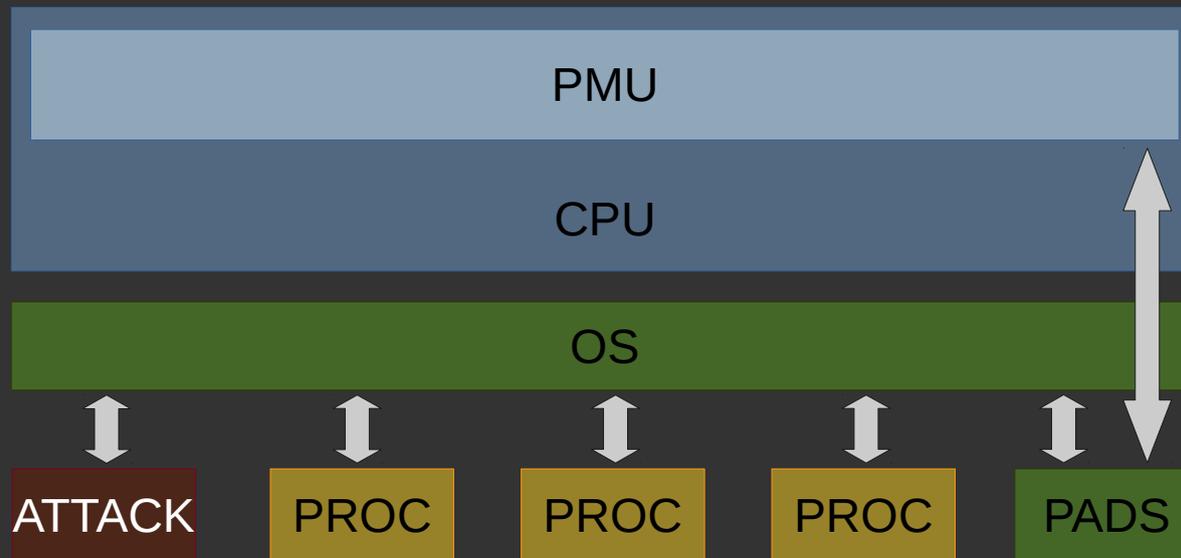


CAIN: Silently Breaking ASLR in the Cloud.

Antonio Barresi, Kaveh Razavi, Mathias Payer, and Thomas R. Gross. In WOOT '15

Upcoming Challenges

- Move collection to VMM to allow per-machine correlation
- Extend and develop new detection modules
- Synthesize detection modules by applying machine learning



Conclusion

Conclusion

- HexPADS is a modular IDS/ADS framework
- Process-based collection of runtime/performance information
- High precision and negligible overhead through PMU
- Ongoing work:
 - More detection modules
 - Machine learning
 - Push framework to VMM level
- Go clone the project at <https://github.com/HexHive/HexPADS>



hexhive

Thank you!
Questions?

Mathias Payer (@gannimo), Purdue University
<http://hexhive.github.io>