IncognitOS

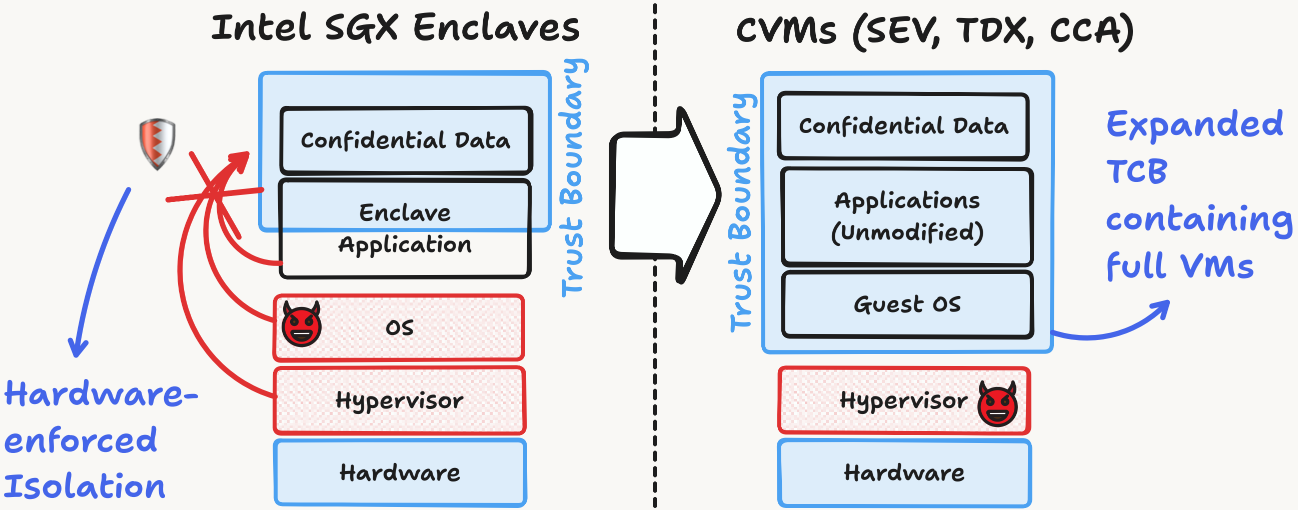
A Practical Unikernel Design for Full-System Obfuscation in Confidential Virtual Machines

Kha Dinh Duy,
Jaeyoon Kim, Hajeong Lim and Hojoon Lee
{khadinh,jena9925,hajeong.lim,hojoon.lee}@skku.edu

System Security Lab
Department of Computer Science and Engineering
Sungkyunkwan University, South Korea

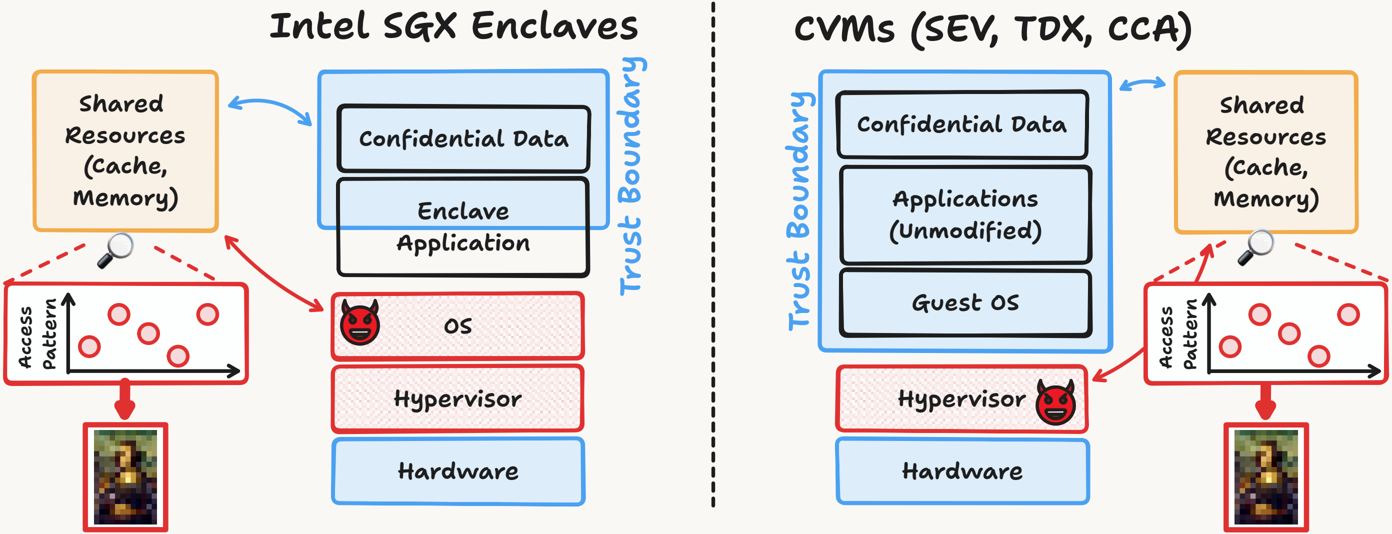
Scan for paper!

Today's Confidential Computing Landscape



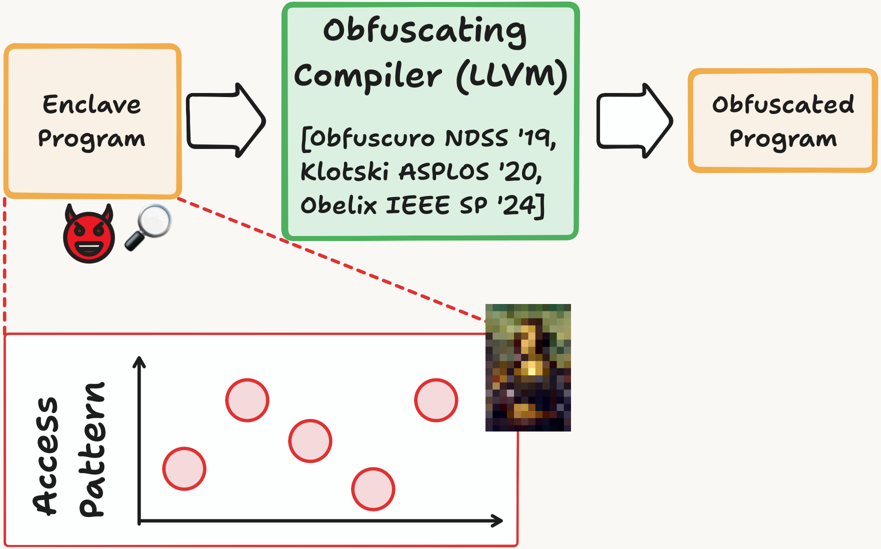
Shift from userspace enclaves to confidential virtual machines (CVMs)

Side-channel Attacks Against Trusted Execution



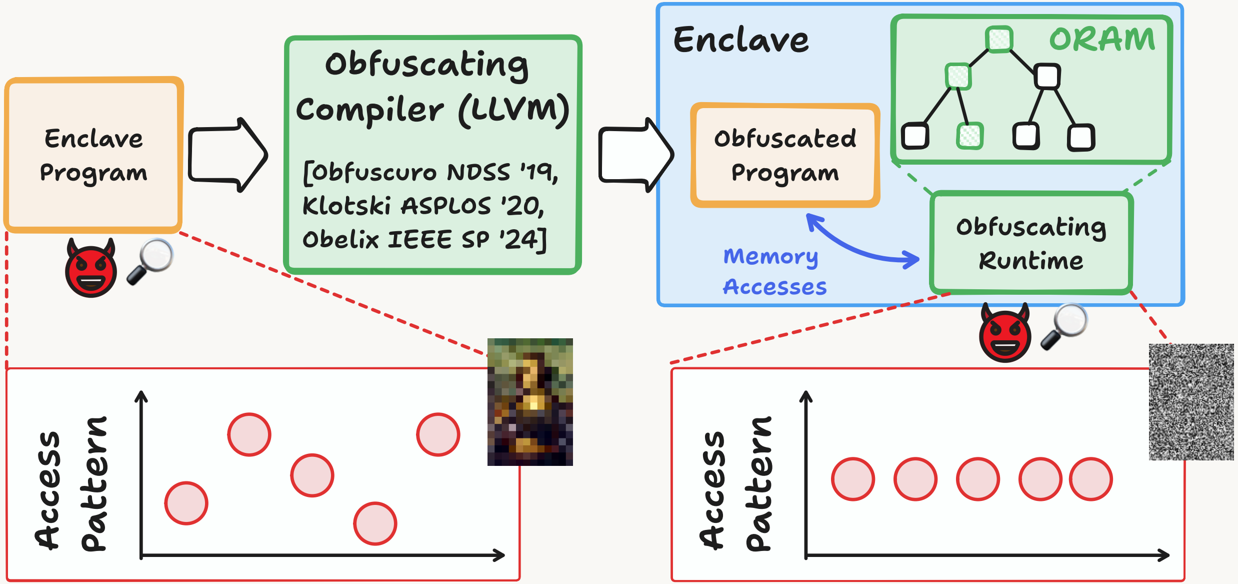
Side-channel attacks threaten enclaves and CVMs alike

Existing Defense: ***Obfuscation Engines***



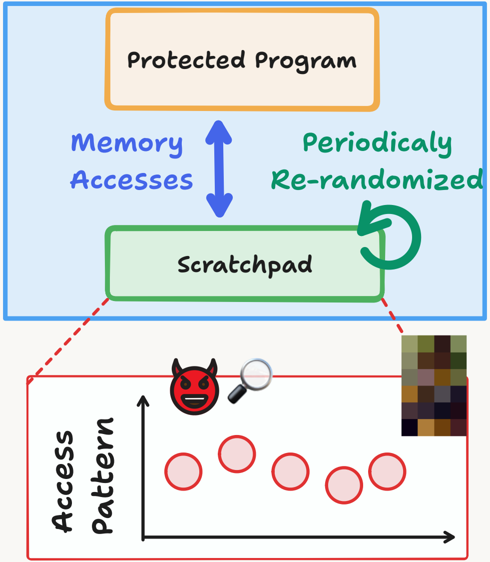
Workflows of obfuscation engines [1, 6, 7]

Existing Defense: ***Obfuscation Engines***



Workflows of obfuscation engines [1, 6, 7]

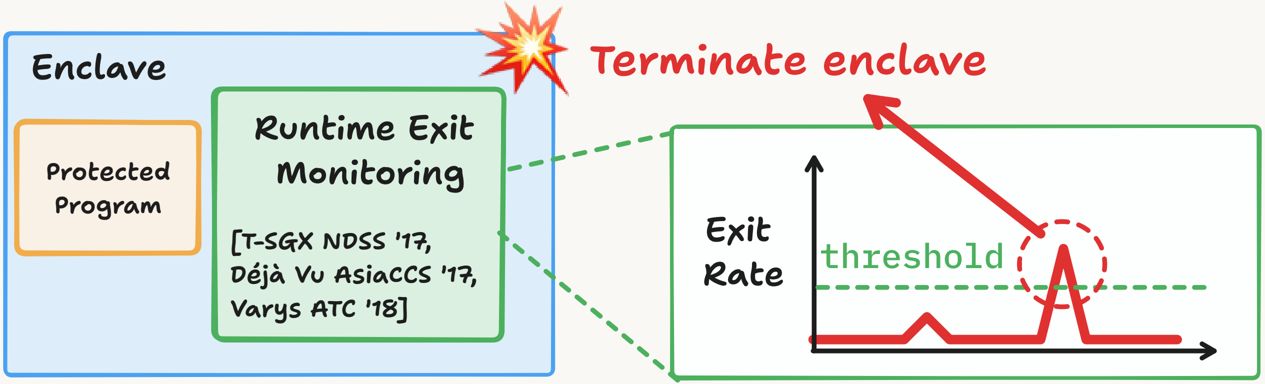
Existing Defense: ***Periodic Re-randomization***



- Strict obfuscation policies introduces prohibitively *high overheads* to protected programs
- *Periodic re-randomization* policies [2, 7] render obfuscation practical

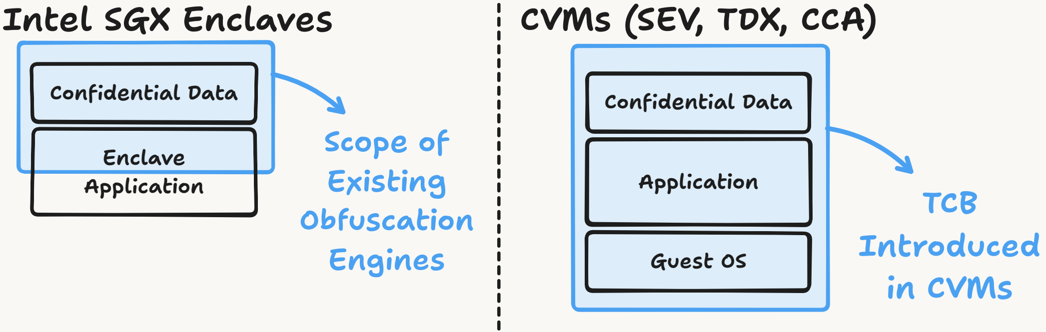
Existing Defense: ***Threshold-based Termination***

Observation: Most SGX attacks rely on *frequent enclave exits (AEXs)* (e.g., timer interrupts, page faults)



Threshold-based enclave termination[3–5]

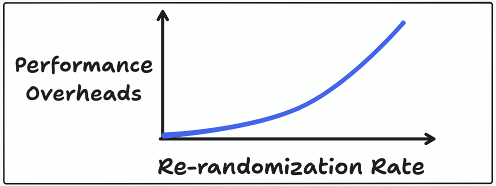
Challenges of CVMs Obfuscation: **Expanded TCB**



- Support for *full-VM protection* (guest OS + application) remains unsolved
- Potential *high overheads* with user + kernel obfuscation

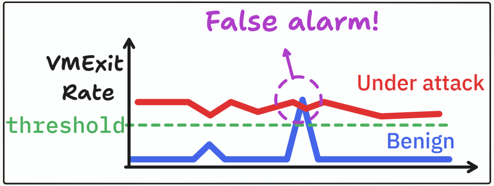
Policy-Wise Limitations

Fixed-rate Re-randomization



- Inherent *trade-offs* between *security* and *performance*

Threshold-based Termination



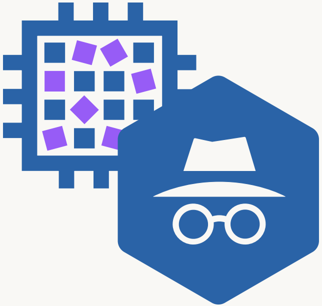
- *False-alarms* under normal CVM workloads

SSLab

SUNG KYUN KWAN UNIVERSITY(SKKU)

Introducing IncognitOS

IncognitOS is a design for practical full-VM obfuscation of CVM workloads



Full-VM Obfuscation

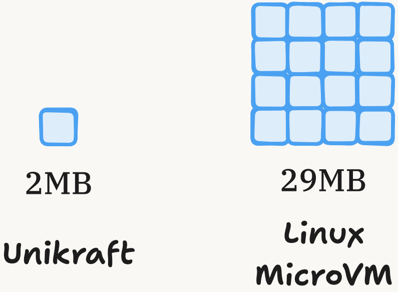
- Embrace *unikernels* as a foundation for OS-level design

Policy Enhancement

- Advance existing policies with an *adaptive obfuscation* strategy

Why Unikernel?

- **Reduced memory footprint**
  → feasible full-system memory randomization
- **No user-kernel separation**
  → efficient accesses to hardware (MMU) and kernel subsystems



Memory footprint of Unikraft unikernel vs. Linux MicroVMs[1]

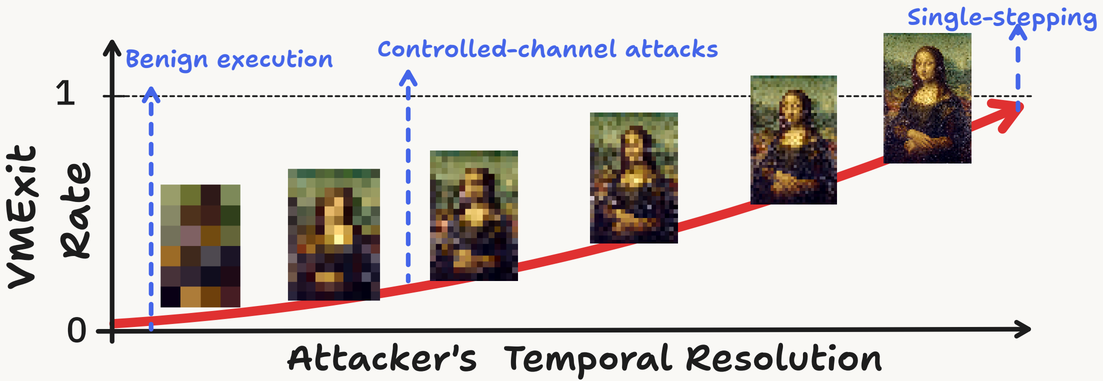
1: https://unikraft.org/docs/concepts/performance

SSLab

SUNG KYUN KWAN UNIVERSITY(SKKU)

Adaptive Obfuscation

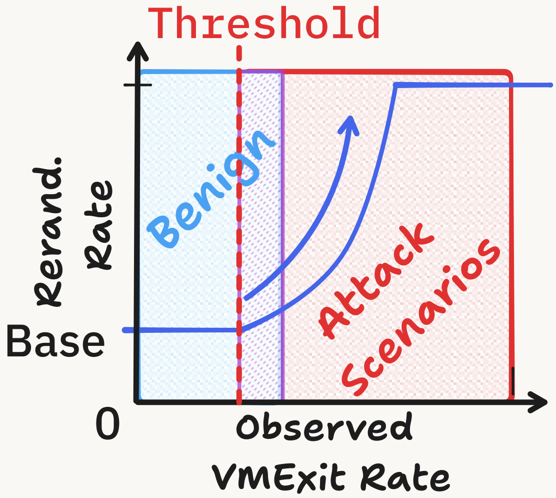
***Key idea:** Adapt memory randomization rate based on VMExit rate measurements.*



Adaptive Obfuscation

An *immune system* against side-channel attacks

- 🙂 **Normal**: Uses sane randomization rate & proactively looks for threats
- 😡 **Threat detected**: Schedules defense (re-randomizations) based on threat level



SSLab

SUNG KYUN KWAN UNIVERSITY(SKKU)

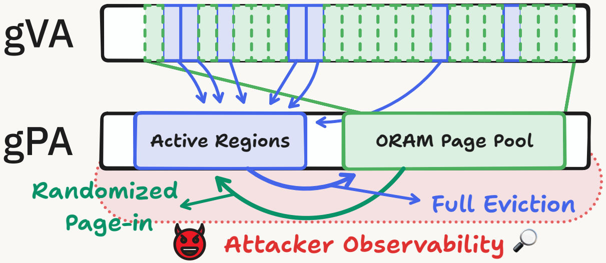
Kernel Subsystems Enforcing Adaptive Defense

Scheduling

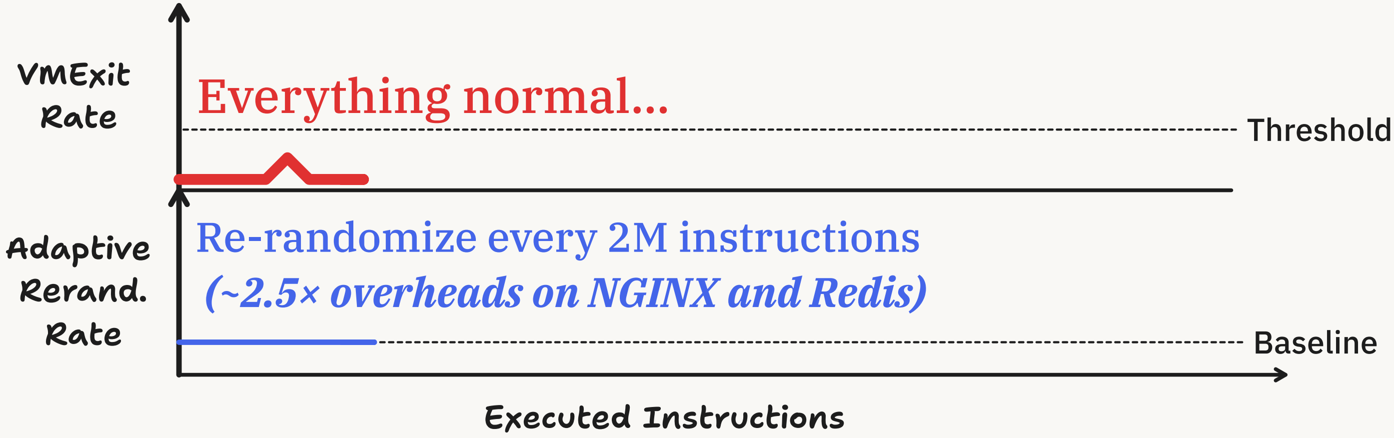
- Samples current VMExit rate *independently of untrusted interrupts* through static binary instrumentation

Paging

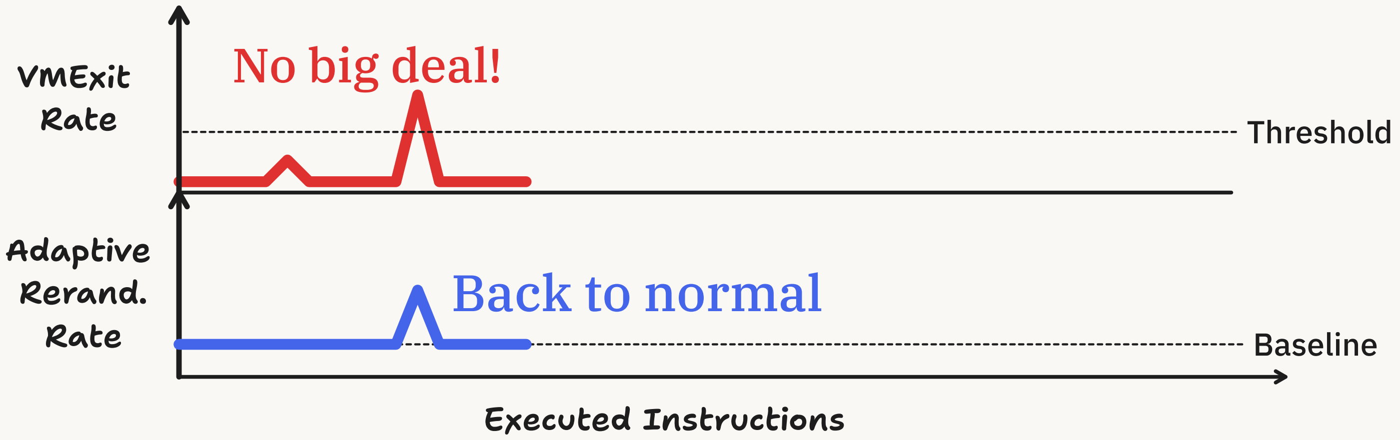
- Transparently randomizes physical memory using OS-level access to *hardware MMU*



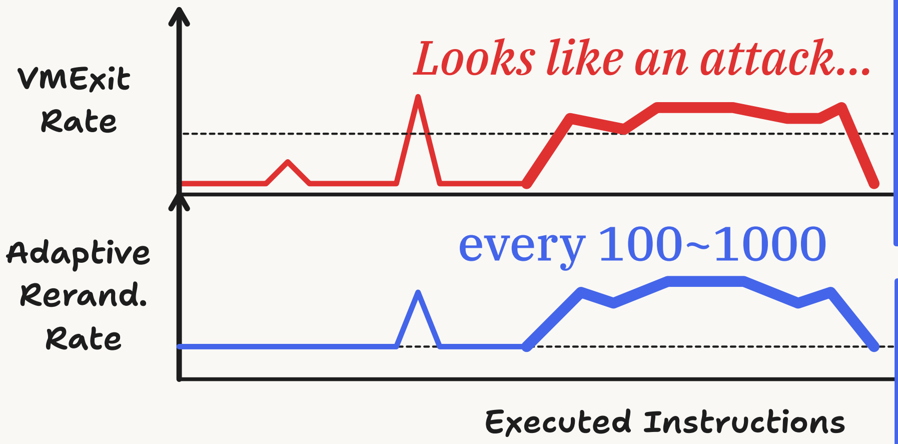
Visualization of Adaptive Obfuscation in Action

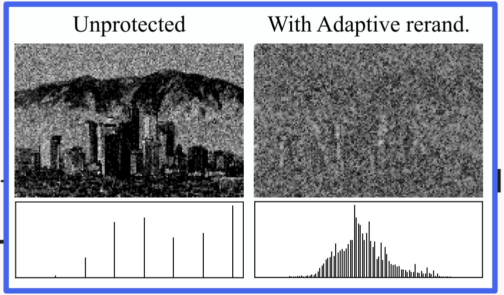


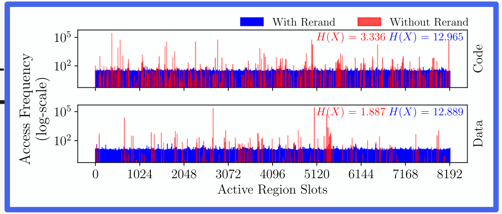
Visualization of Adaptive Obfuscation in Action



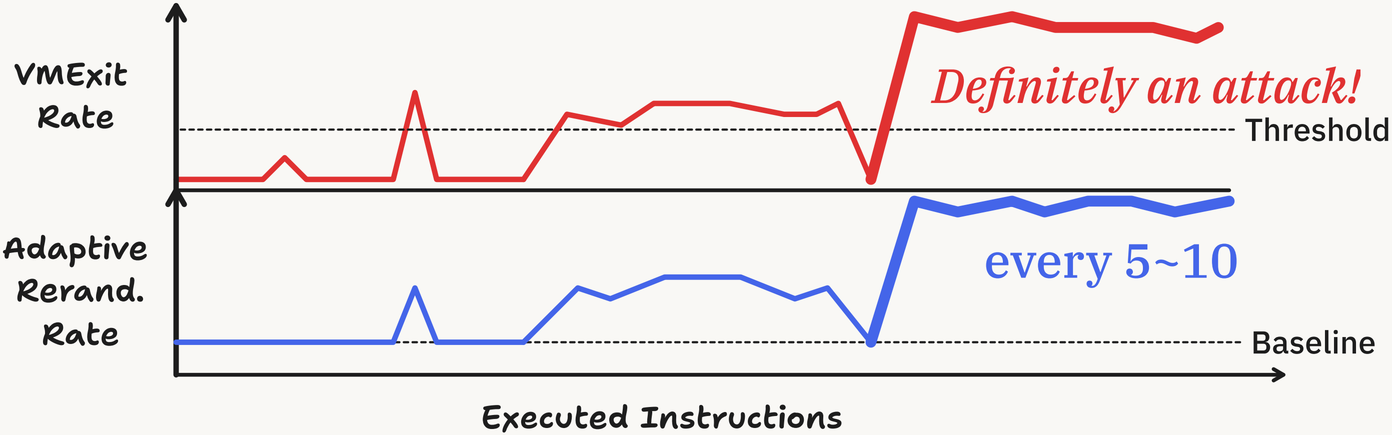
Visualization of Adaptive Obfuscation in Action







Visualization of Adaptive Obfuscation in Action



SSLab

SUNG KYUN KWAN UNIVERSITY(SKKU)

The End

- Check out the paper for more details, e.g.,
  - Kernel subsystems inner working
  - Implementation based on AMD SEV-SNP CVMs and Unikraft unikernel
  - Detailed evaluations and security analysis
- ❓ Got questions? Meet me at the poster session.

Scan for paper!

Job Hunt

- I am expected to finish my Ph.D. later this year
- Looking for a *postdoc or industry* position in *system security*
- Had experiences in *operating systems, TEEs, software compartmentalization* and *software security*

Scan for website

References

[1] Ahmad, A. et al. 2019. OBFUSCURO: A Commodity Obfuscation Engine on Intel SGX. *Proceedings 2019 Network and Distributed System Security Symposium* (San Diego, CA, 2019).
[2] Brasser, F. et al. 2019. DR.SGX: Automated and adjustable side-channel protection for SGX using data location randomization. *Proceedings of the 35th Annual Computer Security Applications Conference* (San Juan Puerto Rico USA, Dec. 2019), 788–800.
[3] Chen, S. et al. 2017. Detecting Privileged Side-Channel Attacks in Shielded Execution with Déjà Vu. *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security* (New York, NY, USA, Apr. 2017), 7–18.
[4] Oleksenko, O. et al. 2018. Varys: Protecting {}SGX{} Enclaves from Practical {}Side-Channel{} Attacks. *2018 USENIX Annual Technical Conference (USENIX ATC 18)* (2018), 227–240.
[5] Shih, M.-W. et al. 2017. T-SGX: Eradicating Controlled-Channel Attacks Against Enclave Programs. *Proceedings 2017 Network and Distributed System Security Symposium* (San Diego, CA, 2017).
[6] Wichelmann, J. et al. 2024. Obelix: Mitigating Side-Channels through Dynamic Obfuscation. *2024 IEEE Symposium on Security and Privacy (SP)* (Los Alamitos, CA, USA, May 2024), 189–189.
[7] Zhang, P. et al. 2020. Klotski: Efficient Obfuscated Execution against Controlled-Channel Attacks. *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems* (New York, NY, USA, Mar. 2020), 1263–1276.