Bunker: A Privacy-Oriented Platform for Network Tracing

Andrew Miklas, Stefan Saroiu, Alec Wolman, Angela D. Brown
University of Toronto and Microsoft Research

Network Tracing Raises Privacy Concerns

- Network tracing is an indispensable tool
  - Traffic engineering, fault diagnosis and recovery
  - Research studies
- Customers' privacy is vital concern to ISPs
  - ISPs view possessing raw traces as a liability

Threat Model for Raw Traces

- ISPs view raw data traces as a liability:
  - Accidental disclosure
  - Operational and remote attacks
  - Subpoenas

- **Implications**:
  1. Nobody can have access to raw data
  2. Trace anonymization can help mitigate privacy concerns

Two Approaches for Anon.

1. Offline anonymization
   - Trace anonymized after raw data is collected
   - Problem: high privacy risks

2. Online anonymization
   - Trace anonymized simultaneously with collection
   - Problem: high engineering costs

Both approaches have serious shortcomings

Simple Tasks can be Very Slow

- Regular expression for phishing:

  " ((password) | (<form) | (<input) | (PIN) | (username) | (<script) | (user id) | (sign in) | (log in) | (login) | (signin) | (log on) | (sign on) | (signon) | (passcode) | (logon) | (account) | (activate) | (verify) | (payment) | (personal) | (address) | (card) | (credit) | (error) | (terminated) | (suspend))[^A-Za-z]"

- libpcre: 5.5 s for 30 M = 44 Mbps max

Our Solution: Bunker

- Combines the best of both worlds
  - Avoids privacy issues of offline anon.
  - Avoids SW engineering challenges of online

- Idea:
  - We can use buffer-on-disk (like in offline anon.) if we can lock-down the trace data + software; only information exposed is anonymized trace

Outline

- Motivation
- Design of Bunker
- Security attacks
- System evaluation
- Conclusions

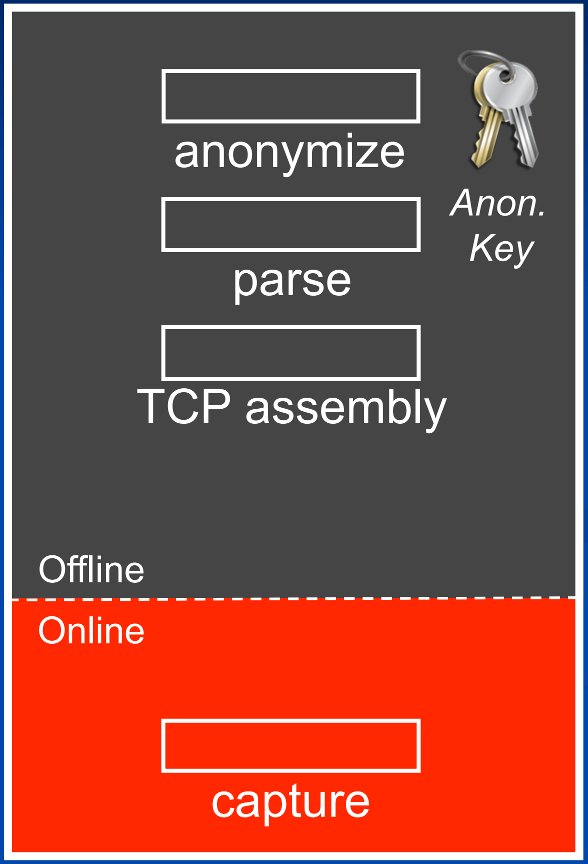
Main Idea: Lock-down Raw Data in Bunker

- "Closed-box" protects sensitive data
  - Contains all raw trace data & processing code
  - Restricted access to closed-box (e.g., no console)

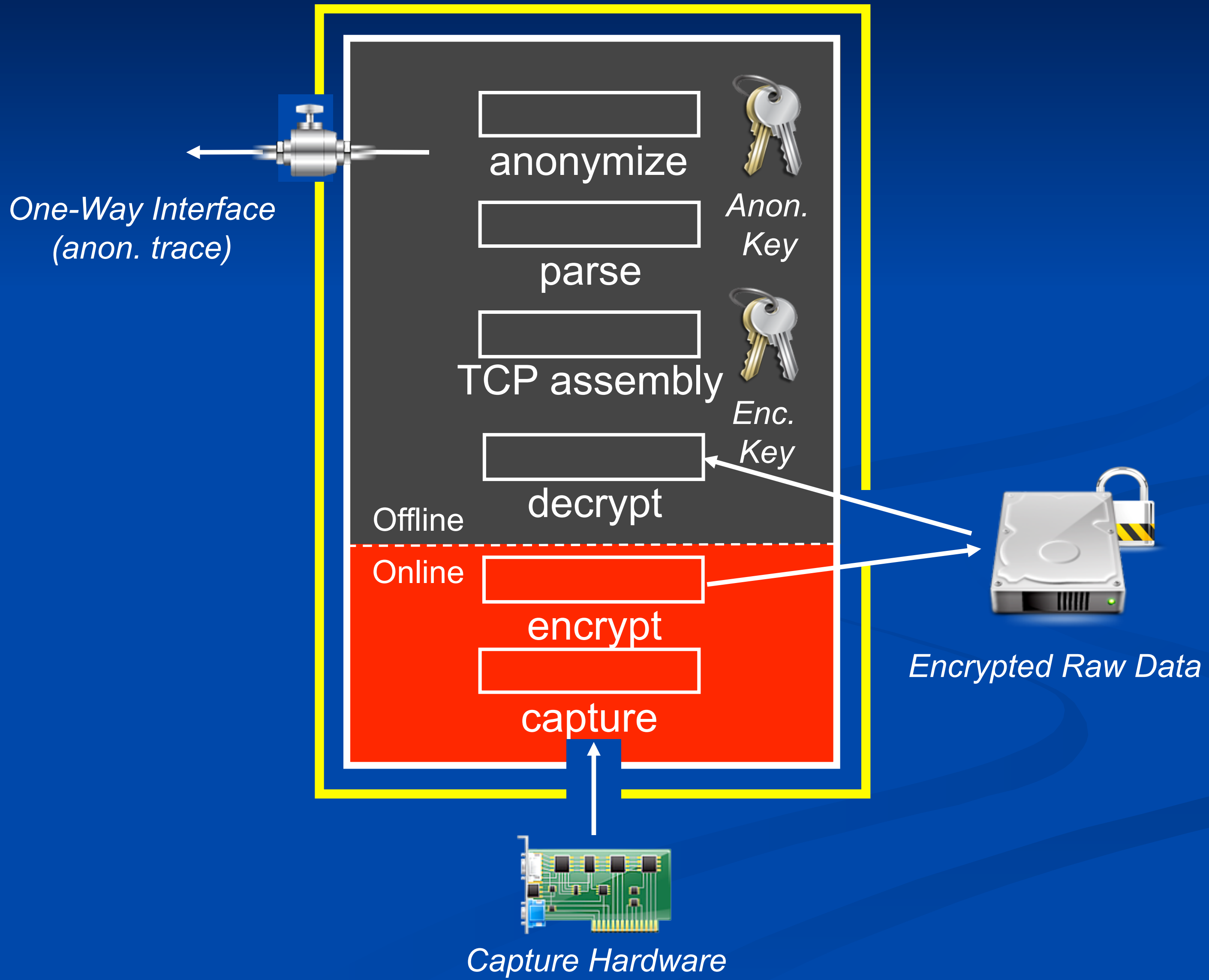
Main Idea: Lock-down Raw Data in Bunker

- "Closed-box" protects sensitive data
  - Contains all raw trace data & processing code
  - Restricted access to closed-box (e.g., no console)
- "Safe-on-reboot": erases data from closed-box
  - ECC RAM is cleared by BIOS upon reboot
  - Encryption protects on disk data
    - Randomly generated key held in RAM inside closed-box
  - Data on disk cannot be decrypted after reboot

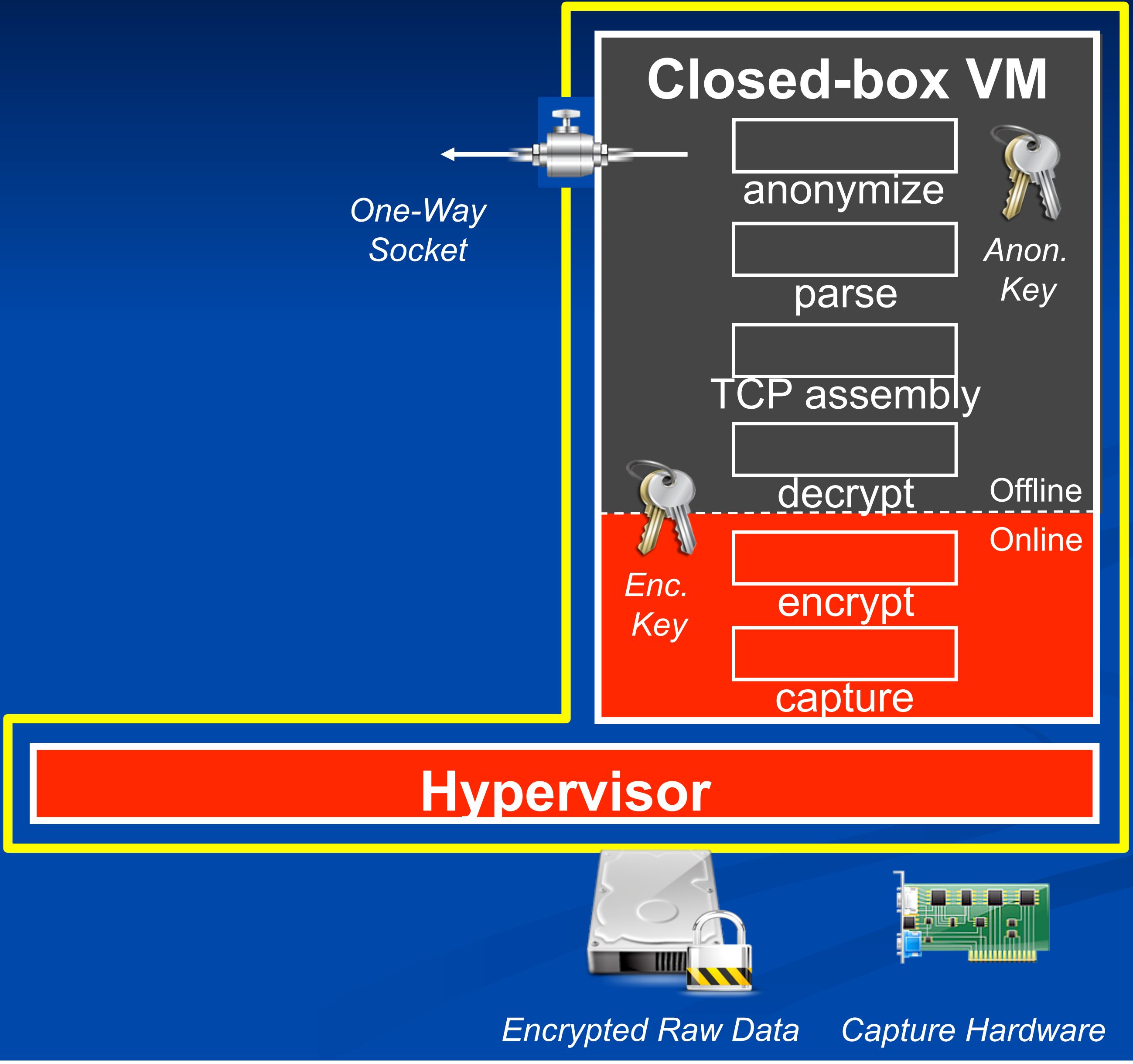
Generic Tracing System



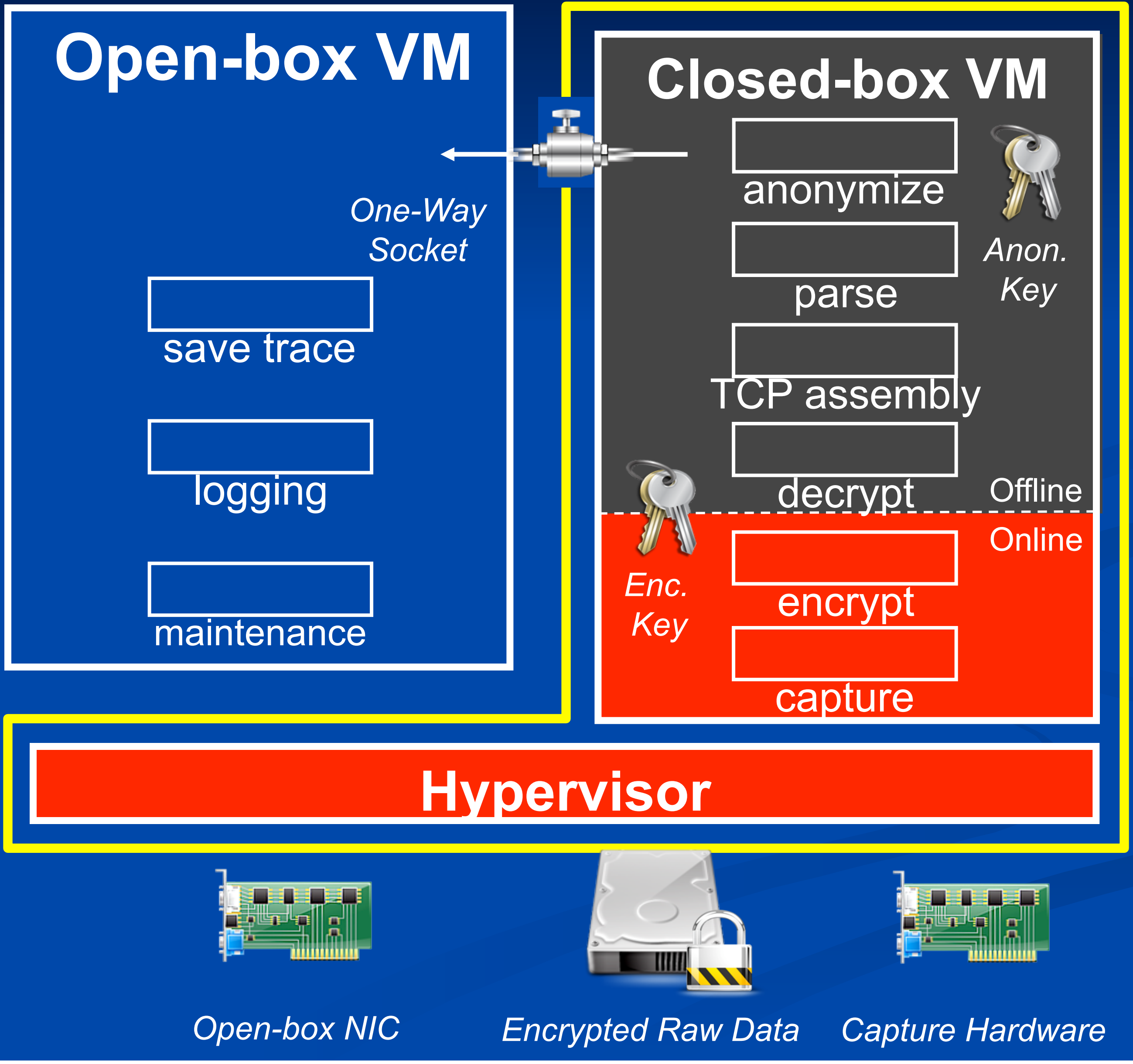
Bunker's Logical Design



VM-based Implementation



VM-based Implementation



How We Implemented Closed-box?

- Eliminate all I/O + drivers from kernel except the ones needed
    - custom-made menuconfig

- Use firewalls to restrict network communication
  - e.g., standard iptables configuration

How to Use Bunker?

- Upon bootup Bunker offers two configurations
  1. Debugging: all drivers enabled
  2. Tracing: most I/O + drivers disabled

- Upon choosing tracing configuration
  - Display and keyboard freeze (no drivers)
  - Kernel's init runs a script to start trace
  - Operator can log in open-box VM via its NIC

Benefits

- Strong privacy properties
  - Raw trace and other sensitive data cannot be leaked
- Trace processing done offline
  - Can use your favorite language! (e.g., Python)
  - Parsing can be done with off-the-shelf components

Outline

- Motivation
- Design of Bunker
- Security attacks
- System evaluation
- Conclusions

Why is Bunker secure?

- Bunker has large TCB but narrow interfaces
  - Bunker remains secure as long as vulnerability cannot be exploited through the narrow interfaces

- Three classes of attacks:
  - Attacking the closed-box's interfaces
  - Hardware attacks
  - Trace injection attacks

Three Classes of Attacks

1. Attacking the closed-box's interfaces
2. Hardware attacks
3. Trace injection attacks

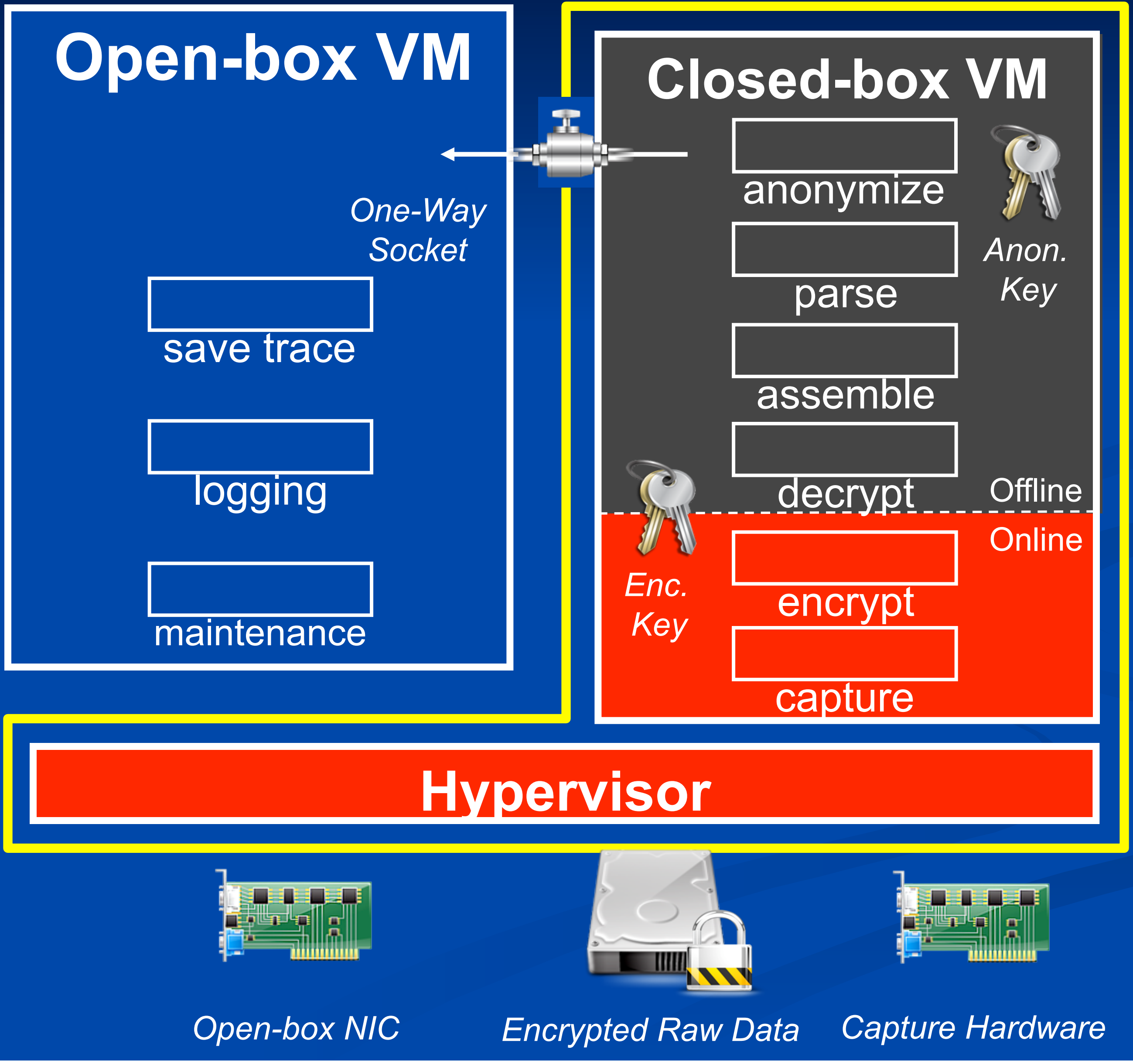
Three Classes of Attacks

1. Attacking the closed-box's interfaces
2. Hardware attacks
3. Trace injection attacks

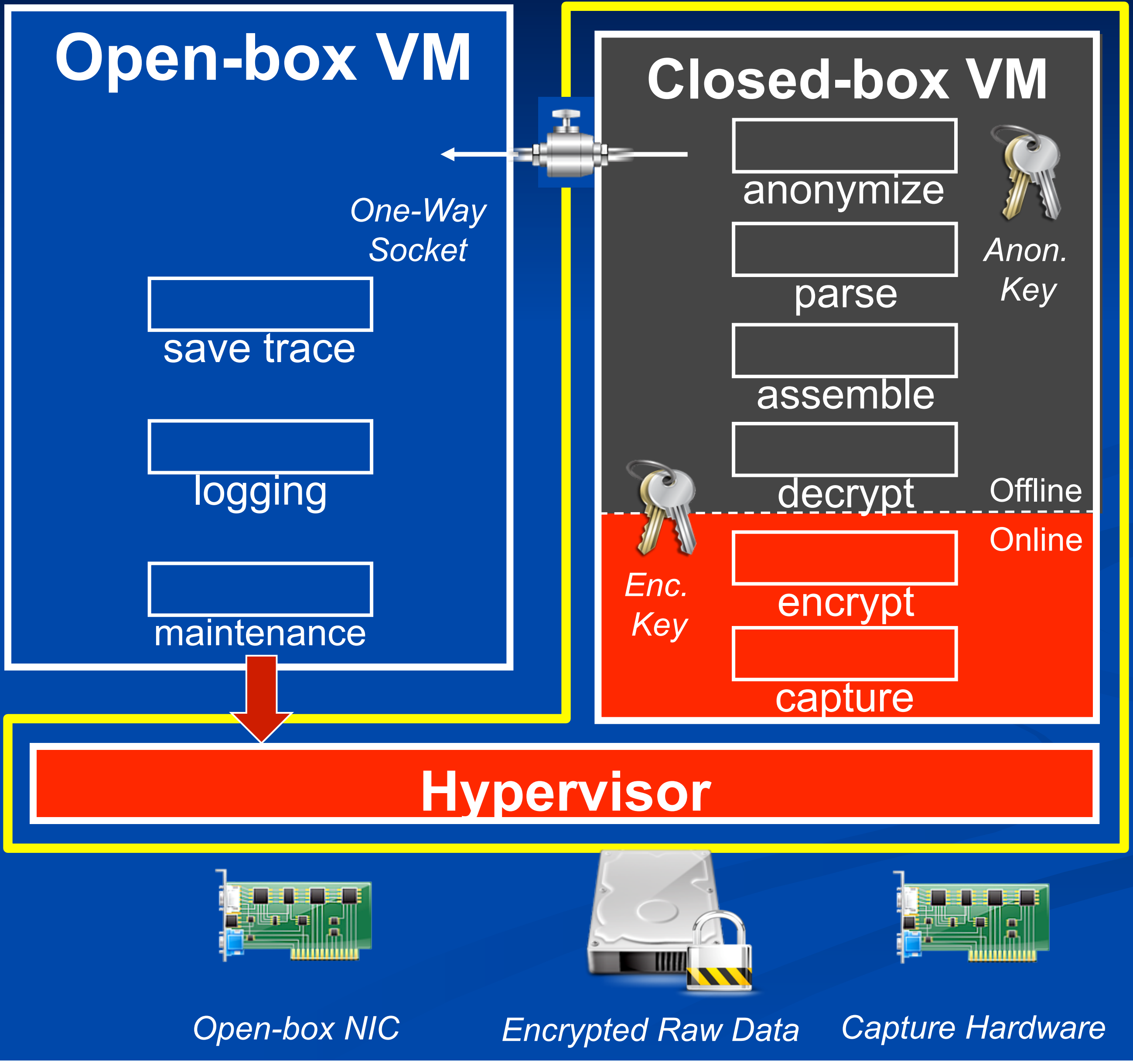
Three Classes of Attacks

1. Attacking the closed-box's interfaces
2. Hardware attacks
3. Trace injection attacks

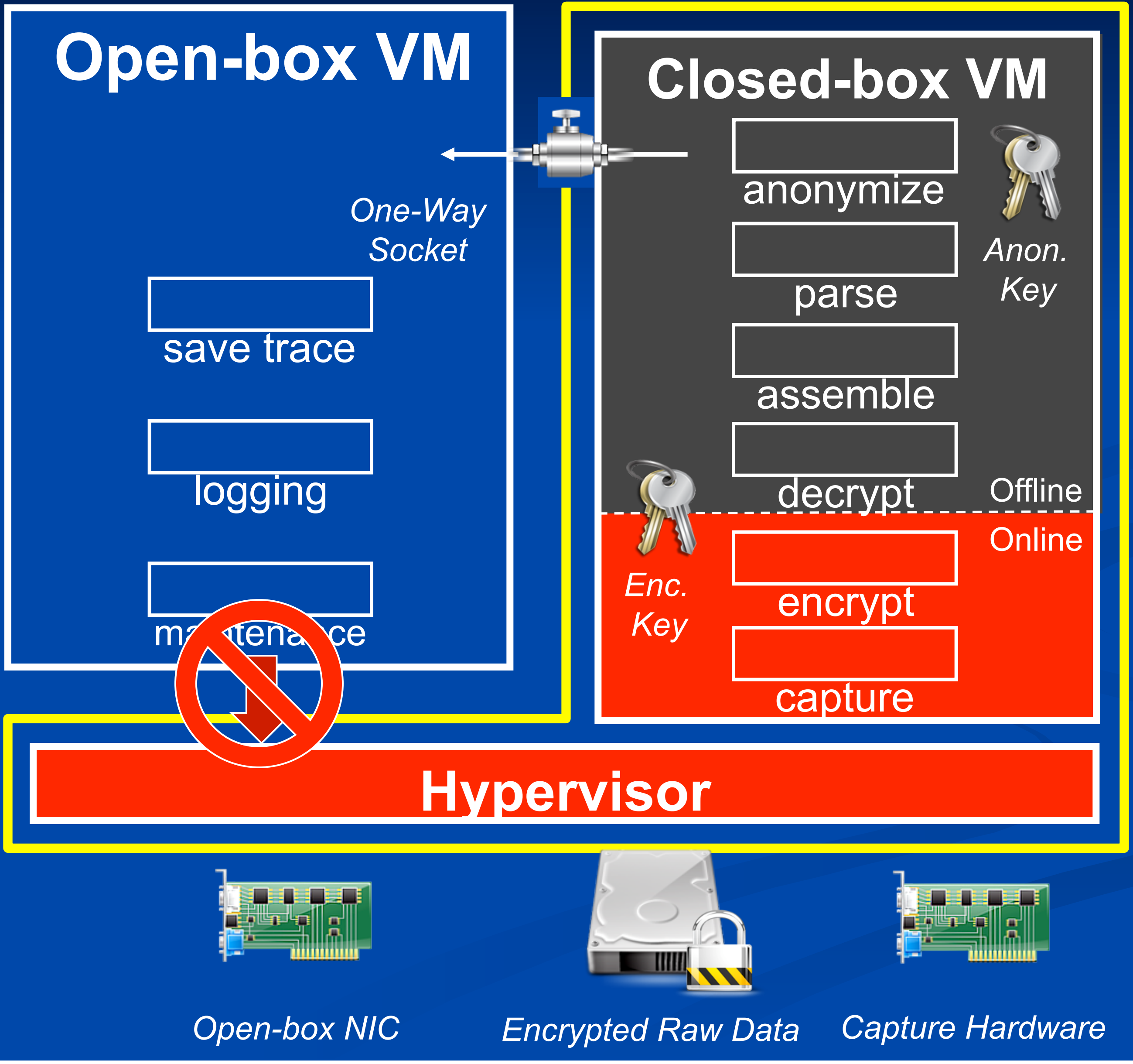
Attacking the Interfaces



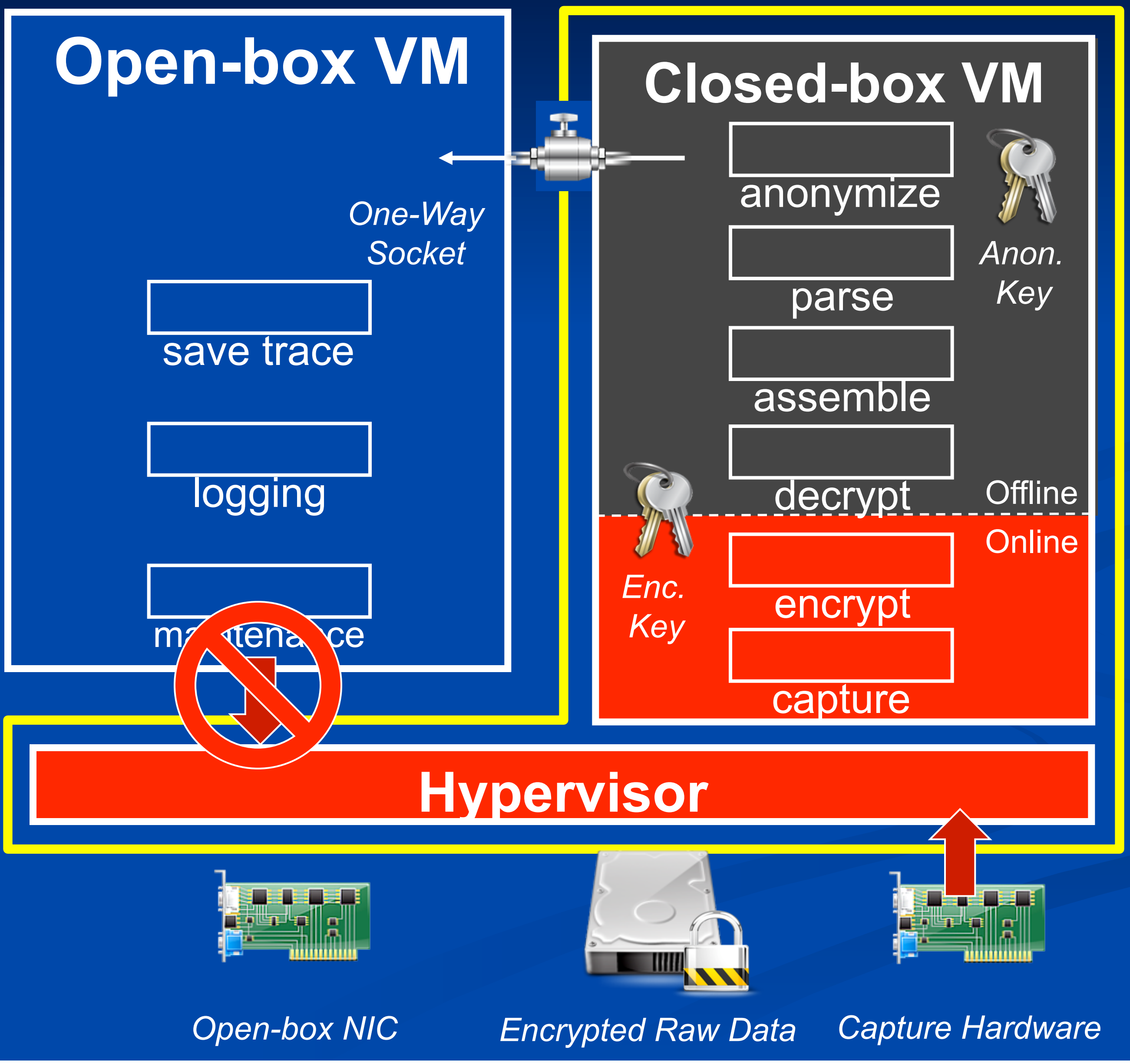
Attacking the Interfaces



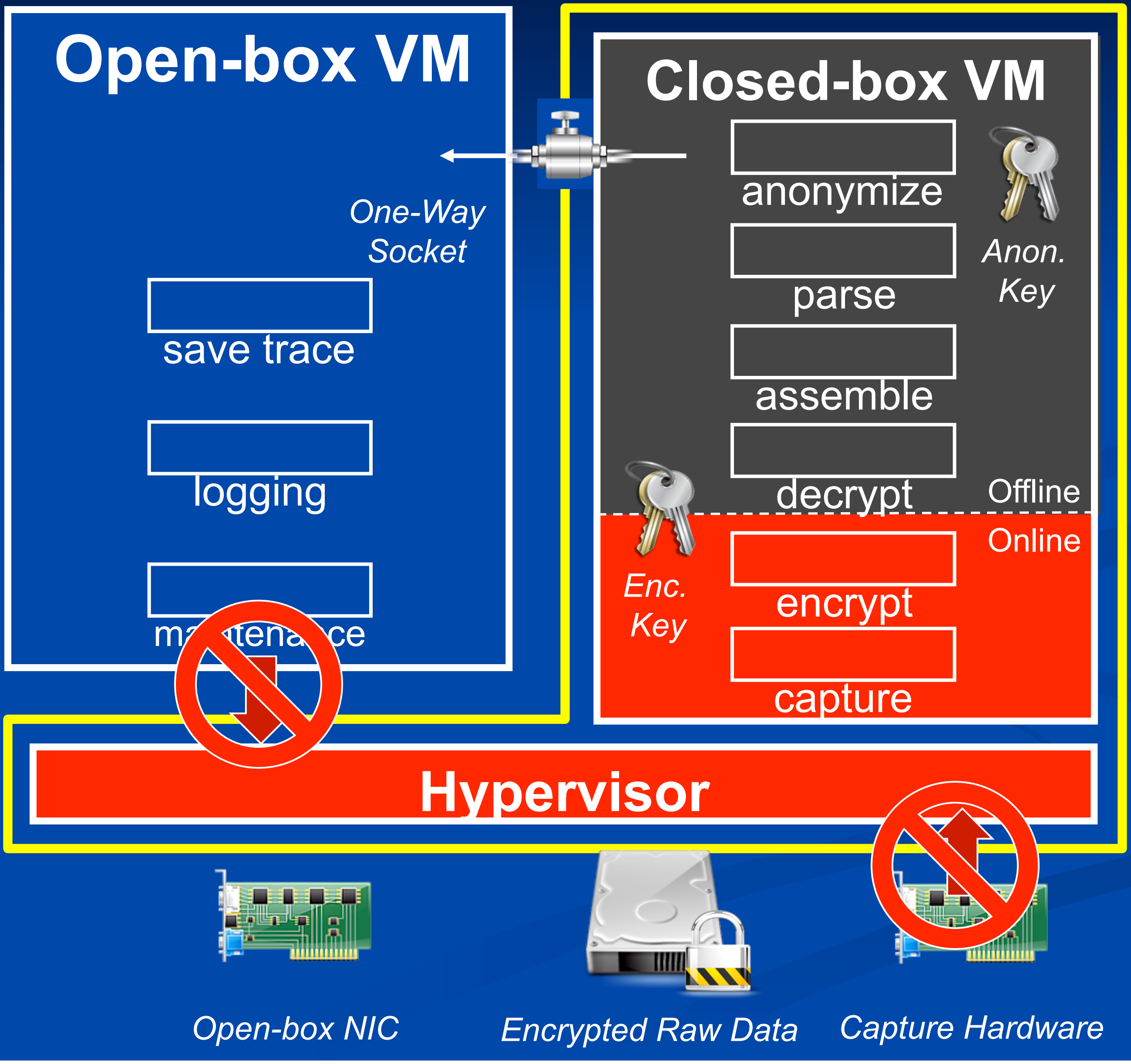
Attacking the Interfaces



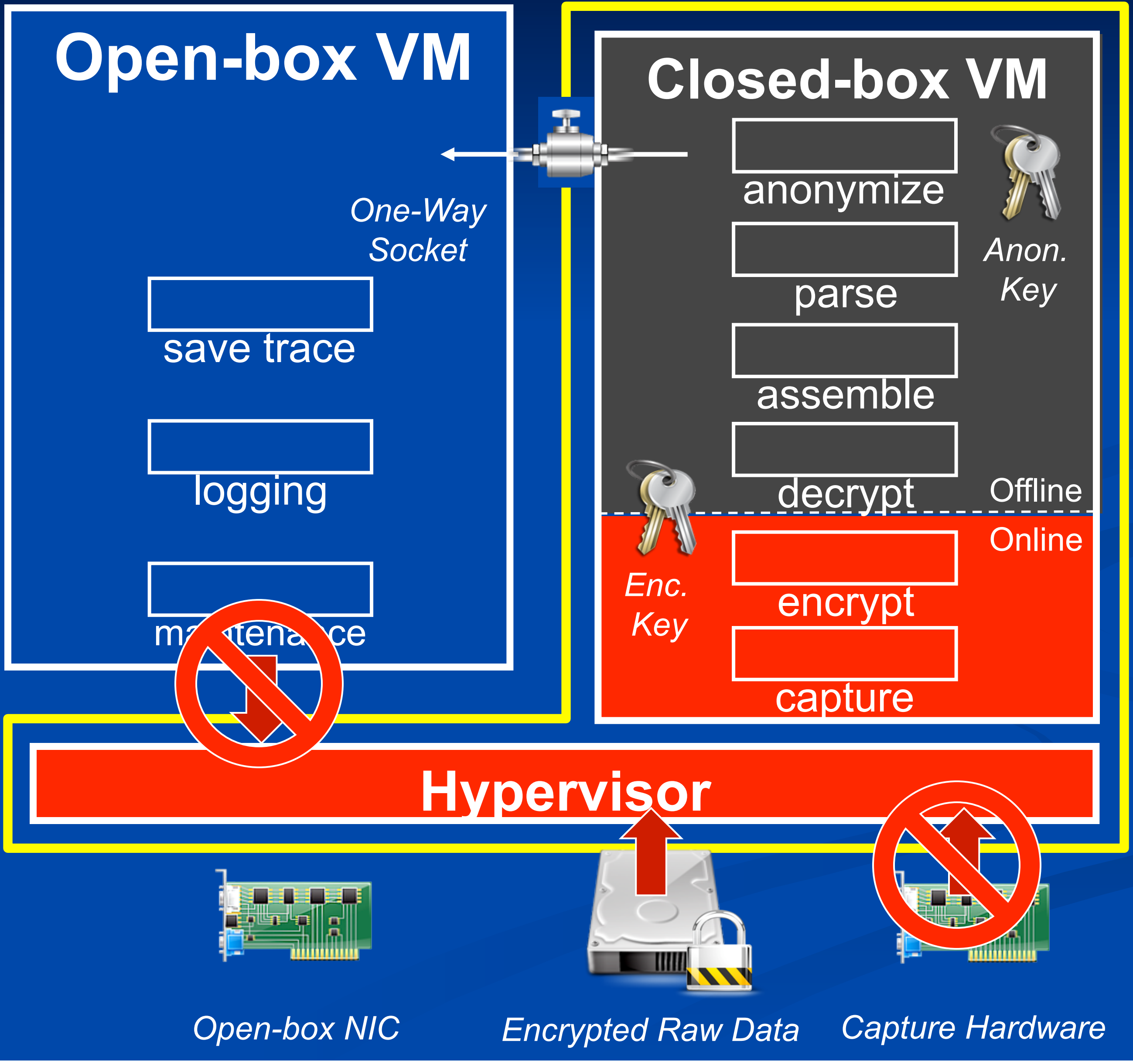
Attacking the Interfaces



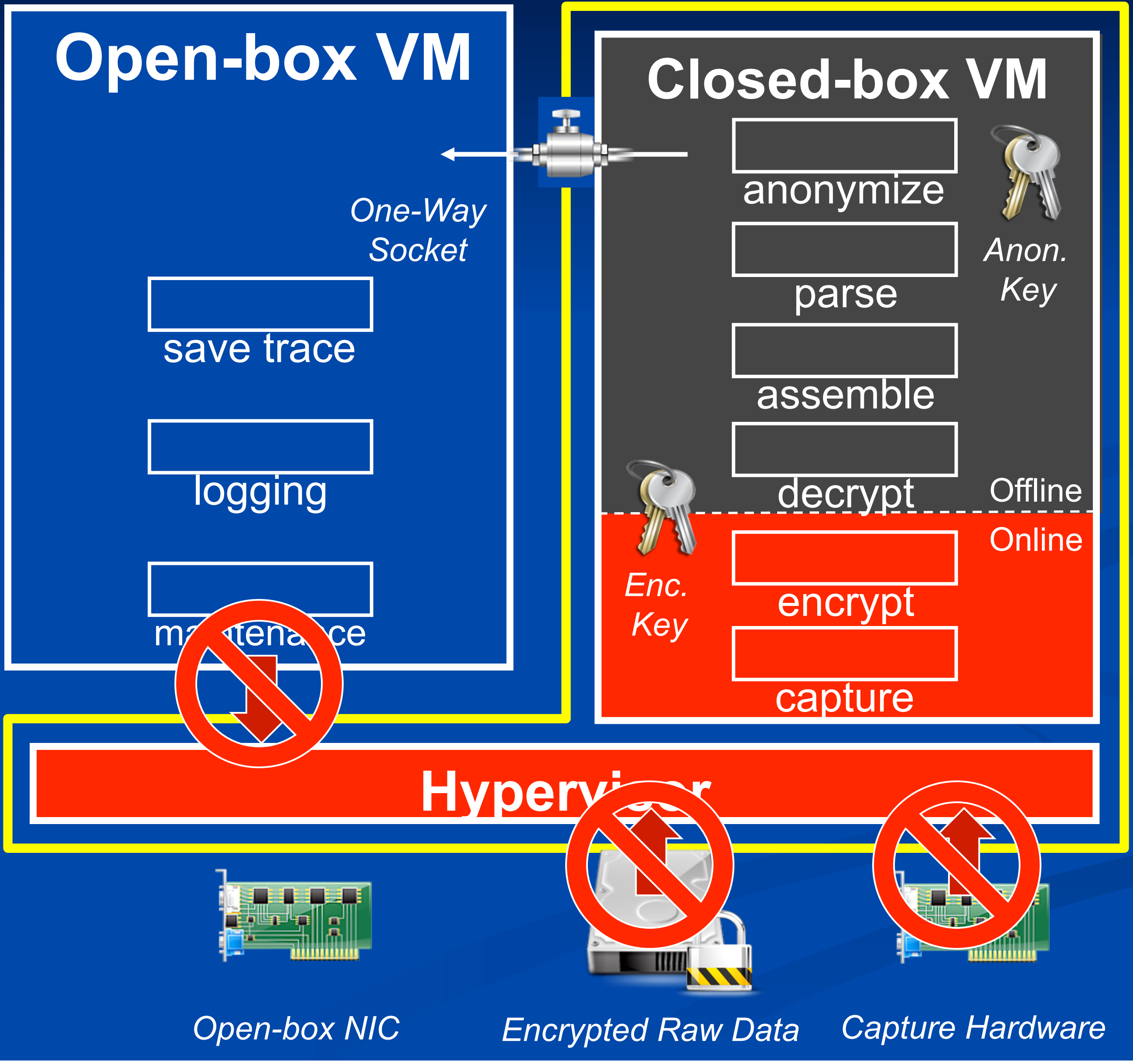
Attacking the Interfaces



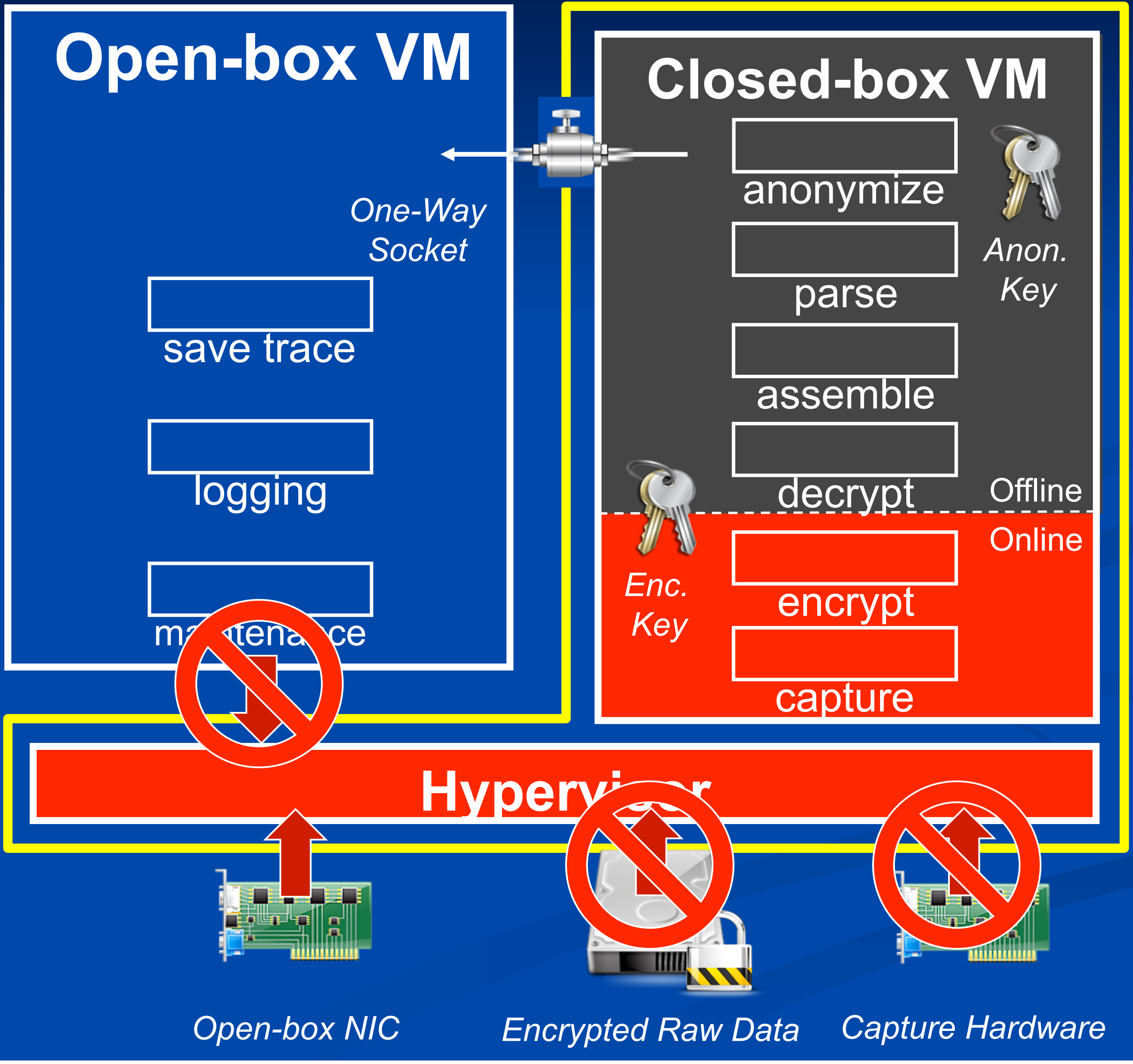
Attacking the Interfaces



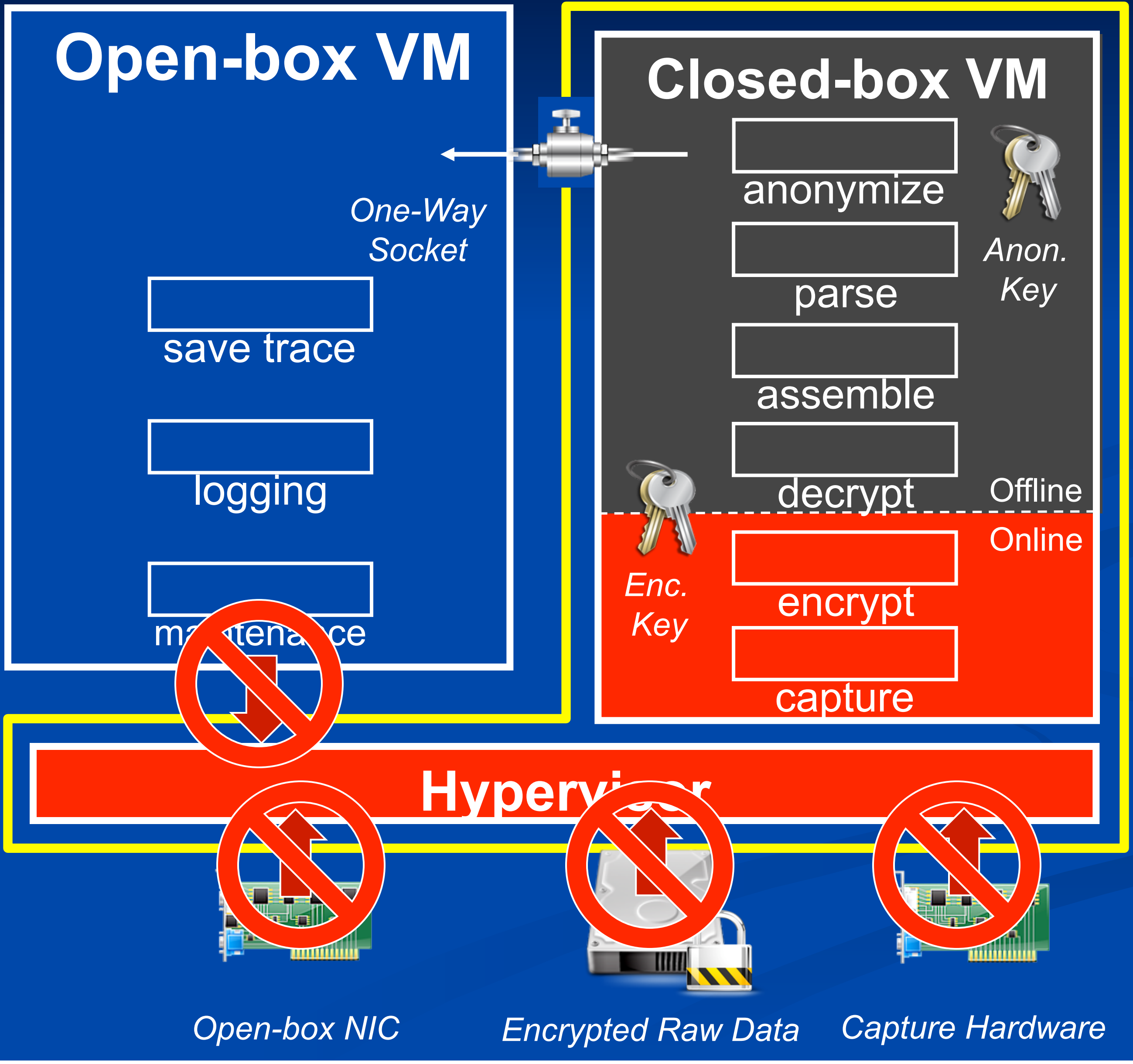
Attacking the Interfaces



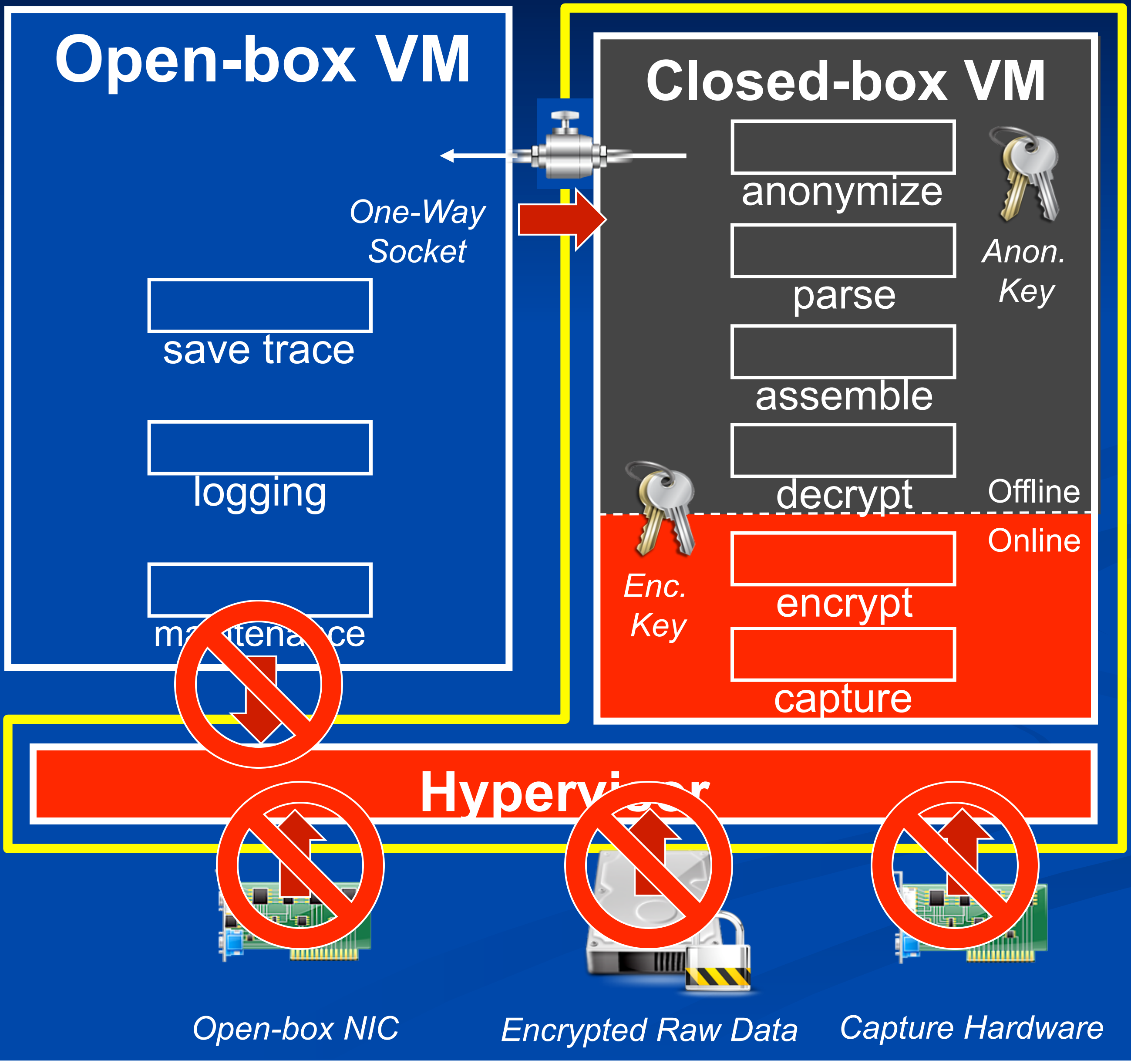
Attacking the Interfaces



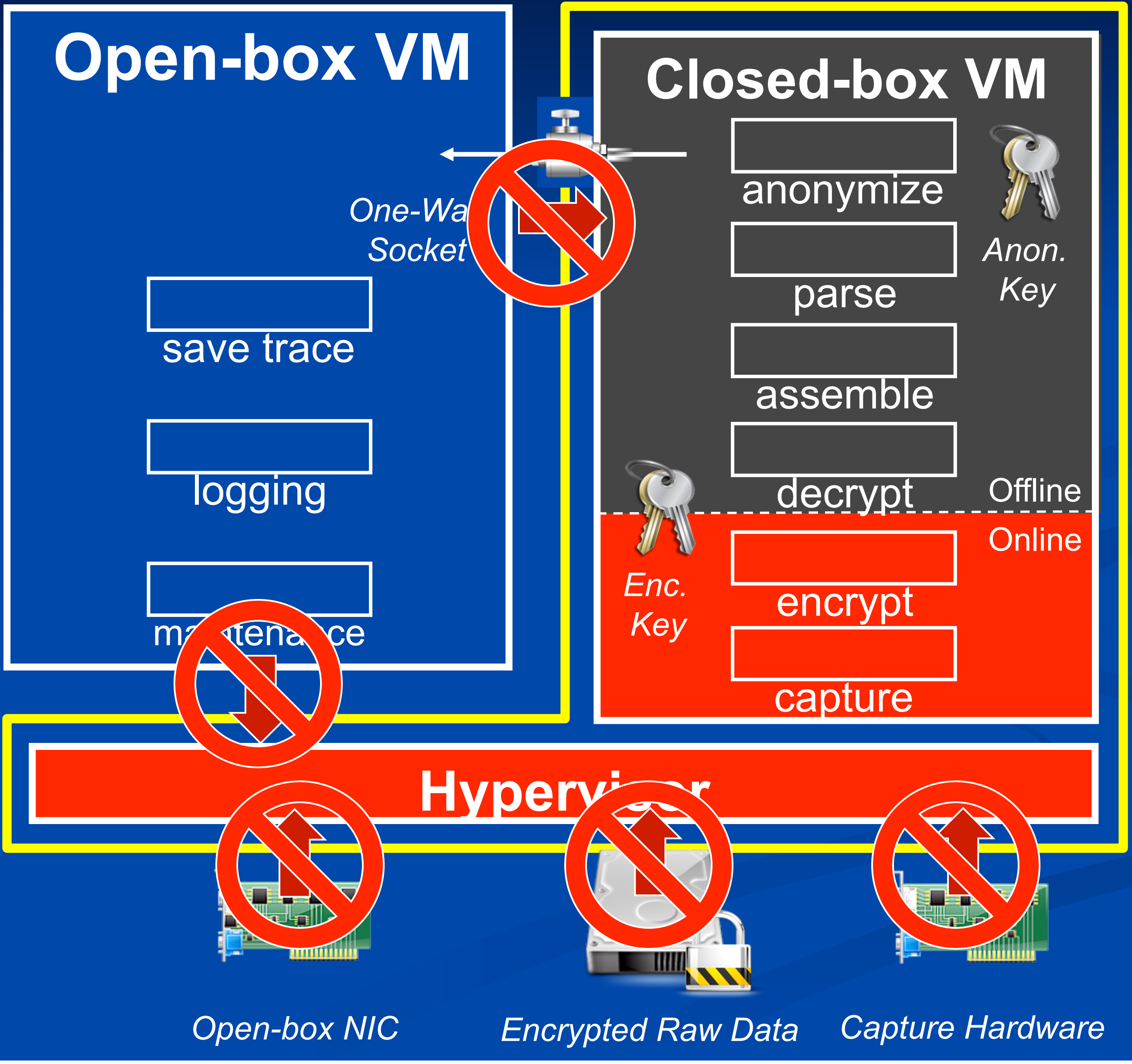
Attacking the Interfaces



Attacking the Interfaces



Attacking the Interfaces



Three Classes of Attacks

1. Attacking the closed-box's interfaces
2. Hardware attacks
3. Trace injection attacks

Attacker Tampers with Hardware

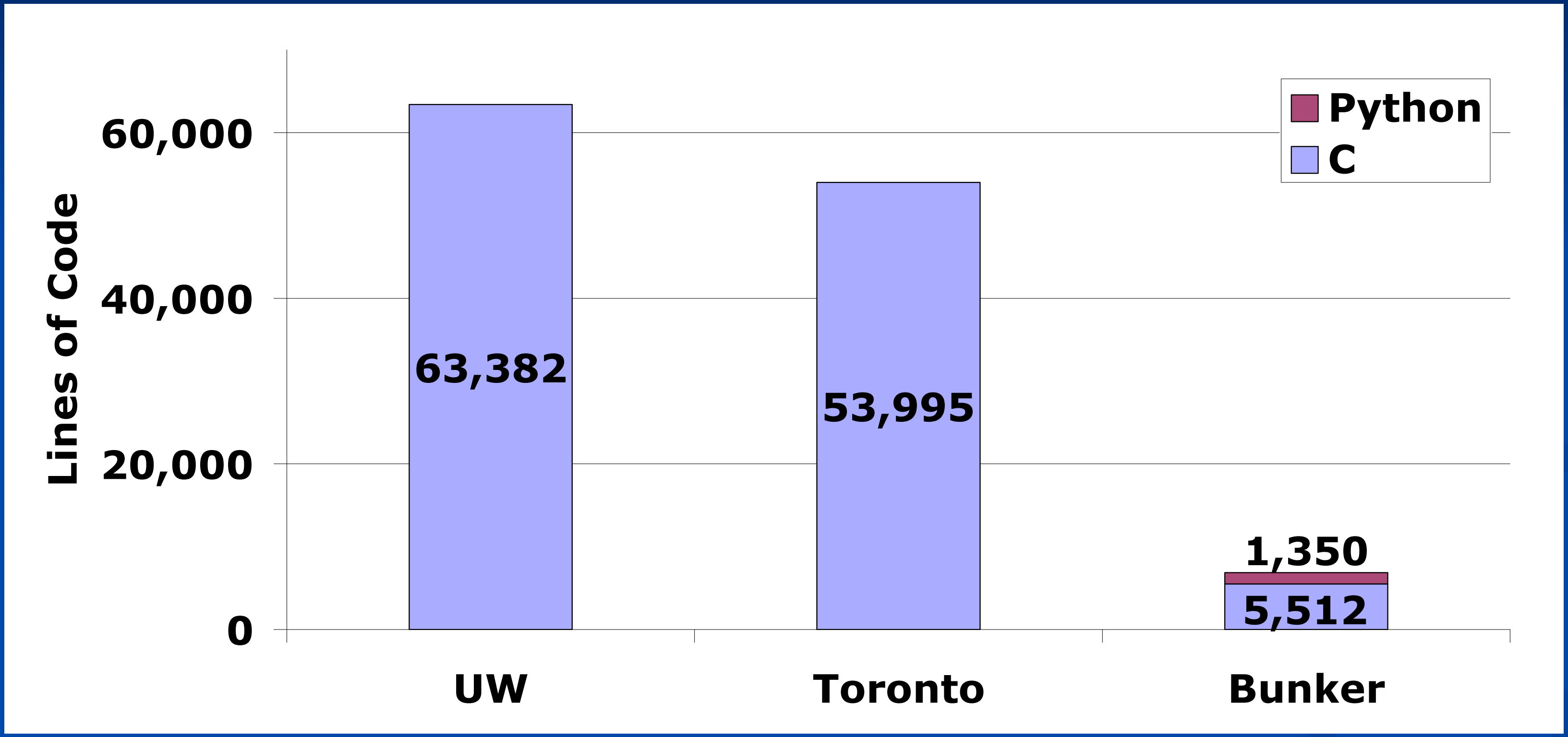
- Safe-on-reboot eliminates most H/W attacks
- Attack left: extracting keys from RAM while system is running
  - Cold-boot attacks
  - Attaching bus monitor
  - Specialized device to dump RAM without OS support

- Need hardware support
  - Secure co-processors could thwart such attacks
  - TPMs are not useful!

Outline

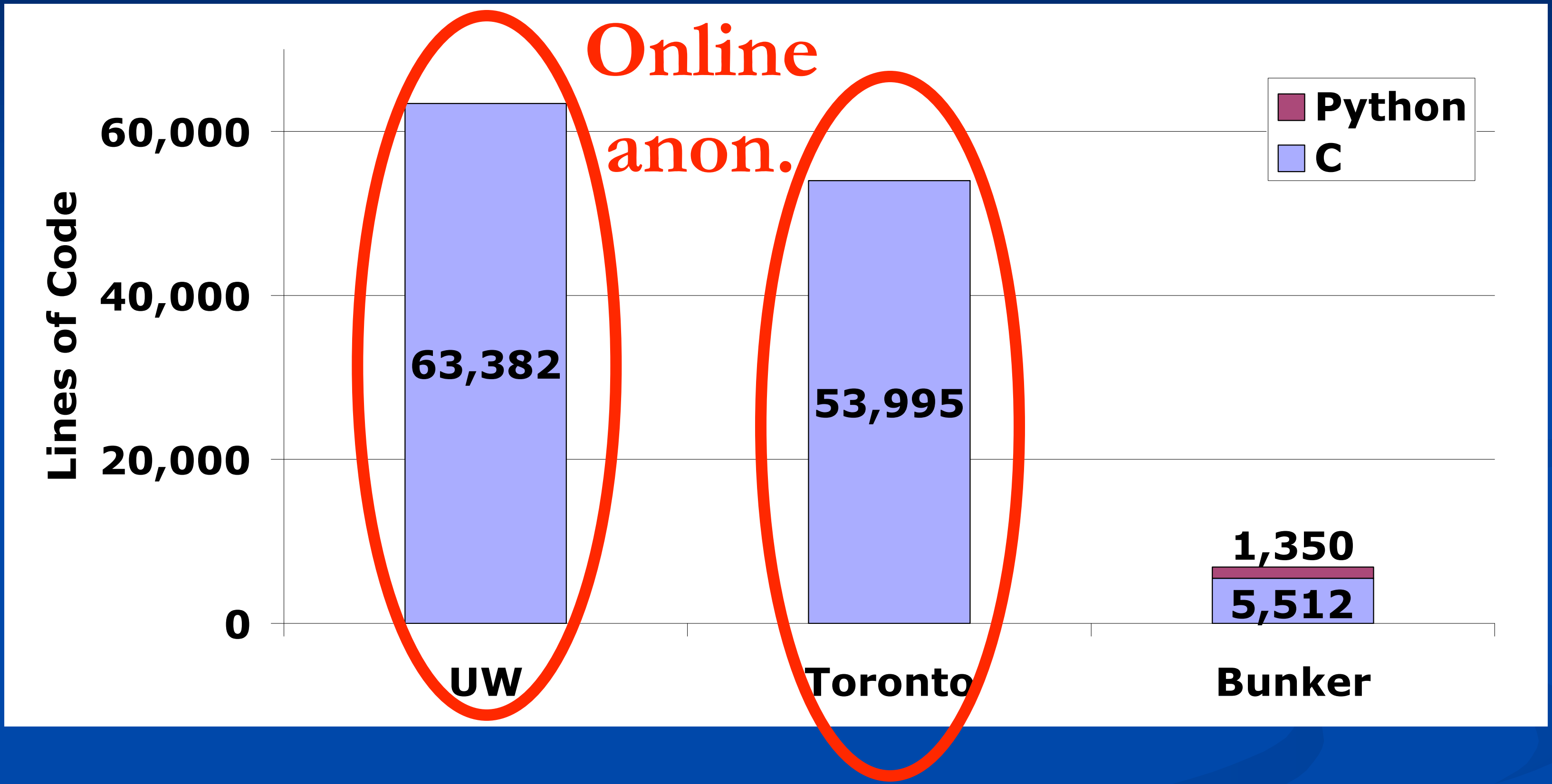
- Motivation
- Design of Bunker
- Security attacks
- System evaluation
- Conclusions

Software Engineering Benefits



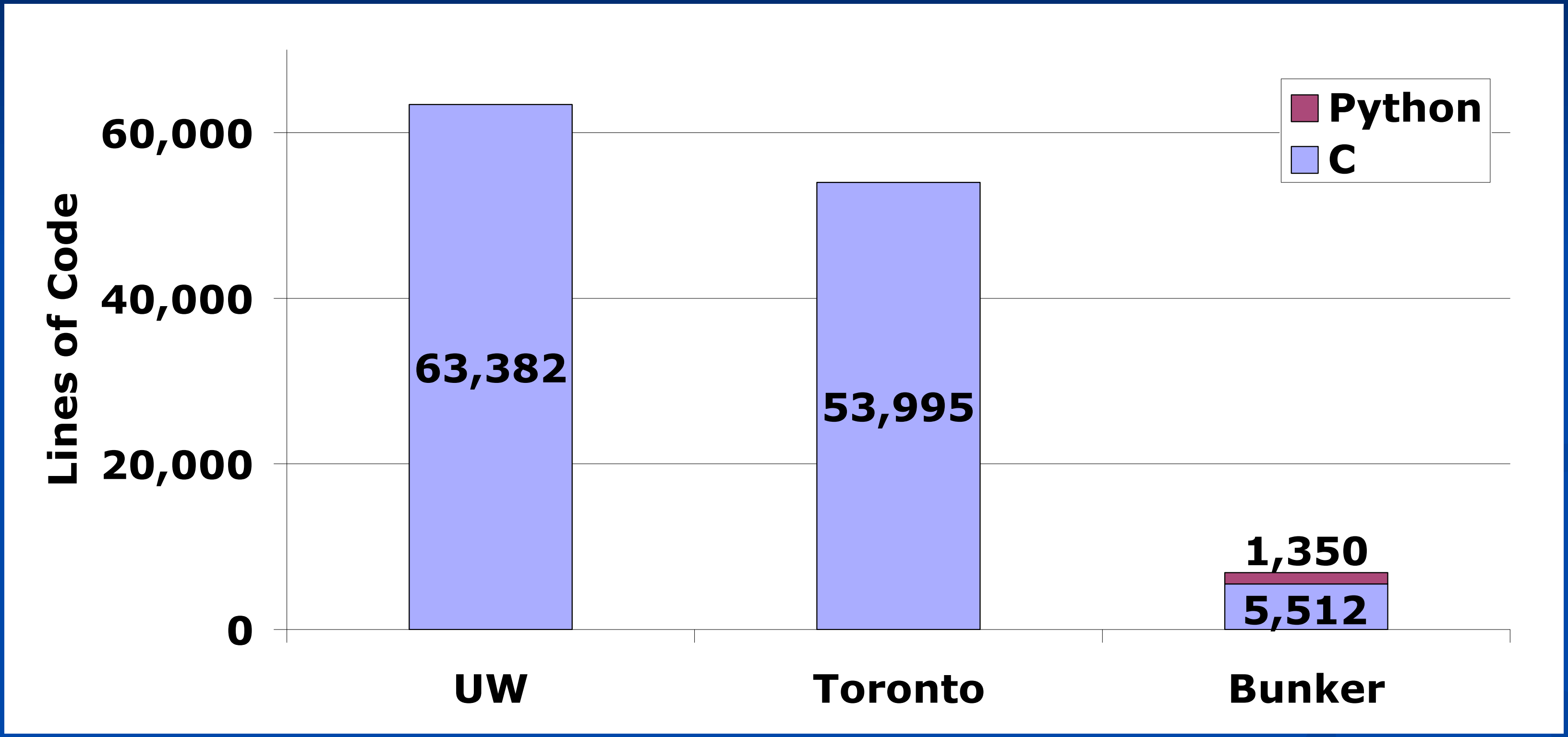
Develop. time: 2 months (Bunker) vs. years (UW/Toronto)

Software Engineering Benefits



Develop. time: 2 months (Bunker) vs. years (UW/Toronto)

Software Engineering Benefits



Develop. time: 2 months (Bunker) vs. years (UW/Toronto)

Conclusions

- Today's network tracing requires privacy properties
  - Operators + researchers look "deep" into the packets
- Offline anon. does not offer privacy properties
- Online anon. requires serious engineering
- Bunker provides
  - the privacy of online anonymization
  - the simplicity of offline anonymization

Questions?

Code available at:

http://www.cs.toronto.edu/~stefan/bunker