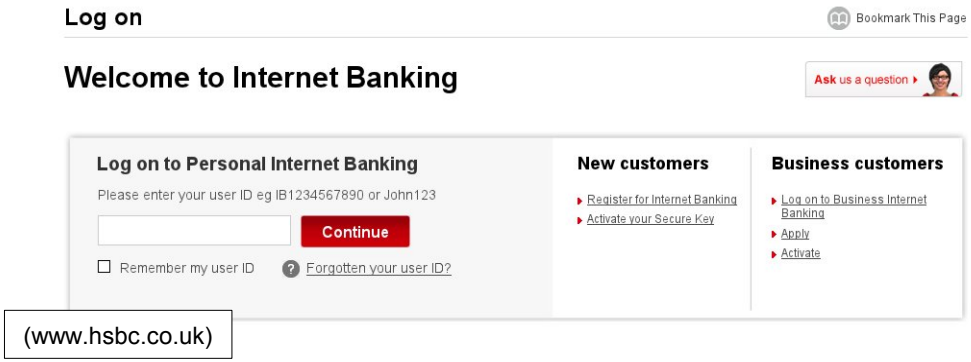
Understanding DMA Malware

DIMVA2012 – 9th Conference on Detection of Intrusions and Malware & Vulnerability Assessment

Patrick Stewin and Iurii Bystrov, July 26th, 2012, Heraklion, Greece
patrickx@sec.t-labs.tu-berlin.de

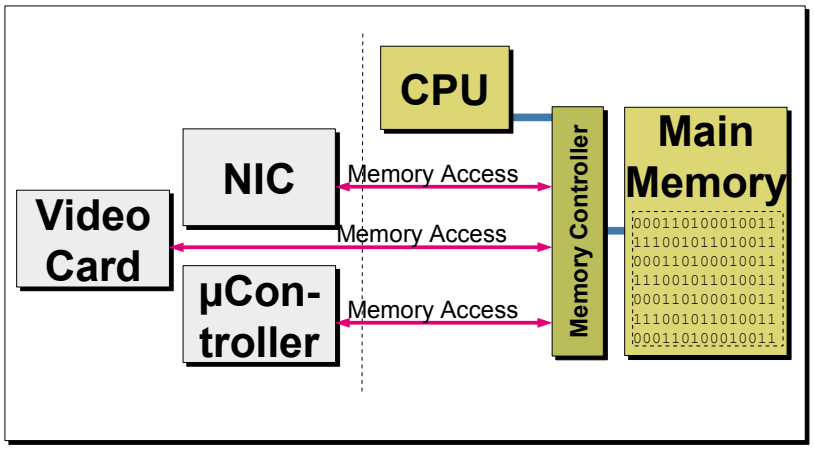
Malicious Software Arms Race



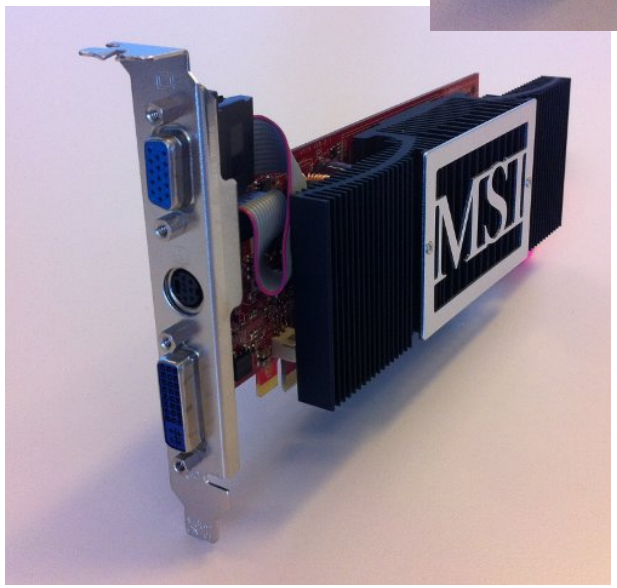
(www.hsbc.co.uk)

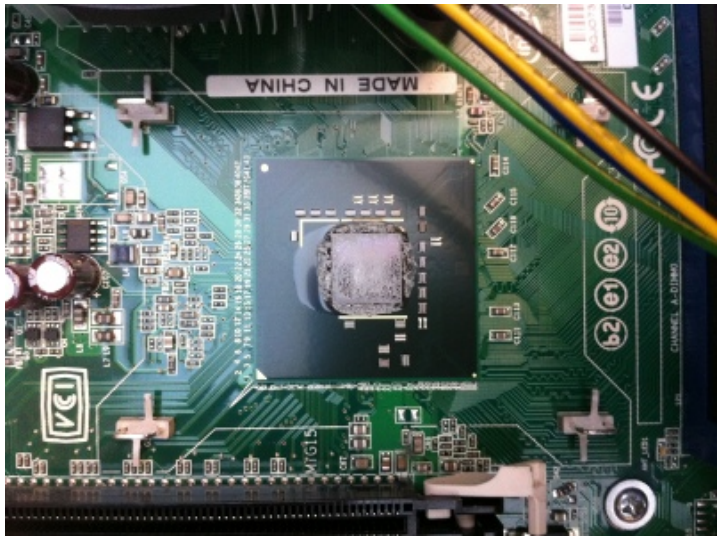
- Countermeasures
  Anti-virus, firewalls, etc.
- Stealth attacks,
  see *rootkit evolution*

Dedicated Hardware

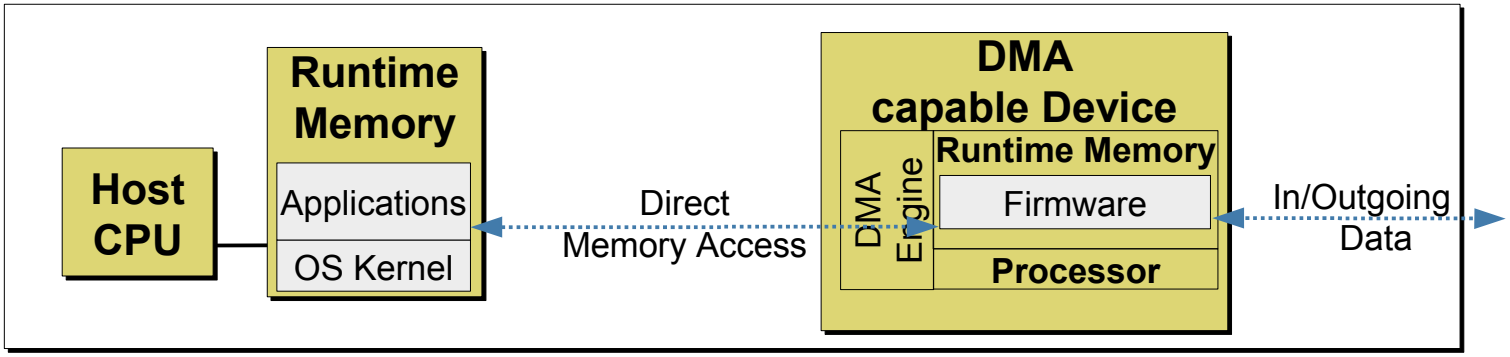


Separated Execution Environments



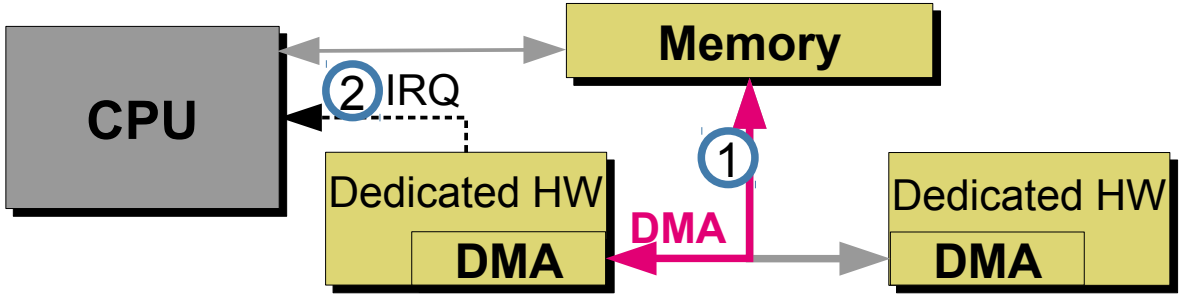


Common Hardware (HW) Features



Direct Memory Access (DMA) capable Computer Platform

- Precondition for stealth malware → 1st party DMA:

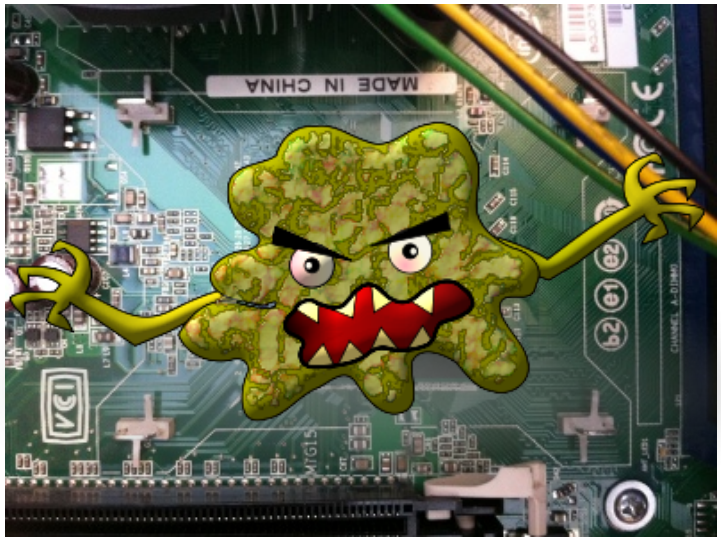


DMA Malware Definition

- **More than controlling a DMA engine**

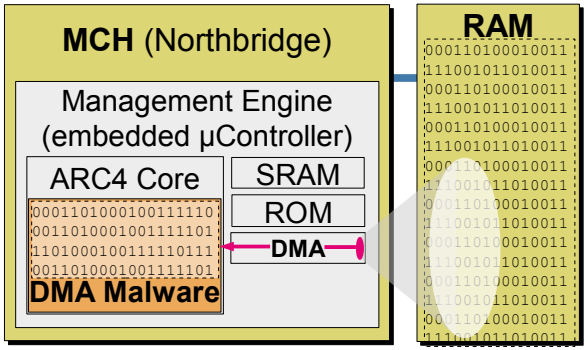
- Malware functionality executed on dedicated HW
- No physical access
- Rootkit/stealth capabilities

- Optional:
  Survival of power off mode



DMA Malware Properties

- **Three phases**
  - Search
  - Process data
  - Exfiltration/infiltration



DMA Capable Device Integrated in Intel Chipsets

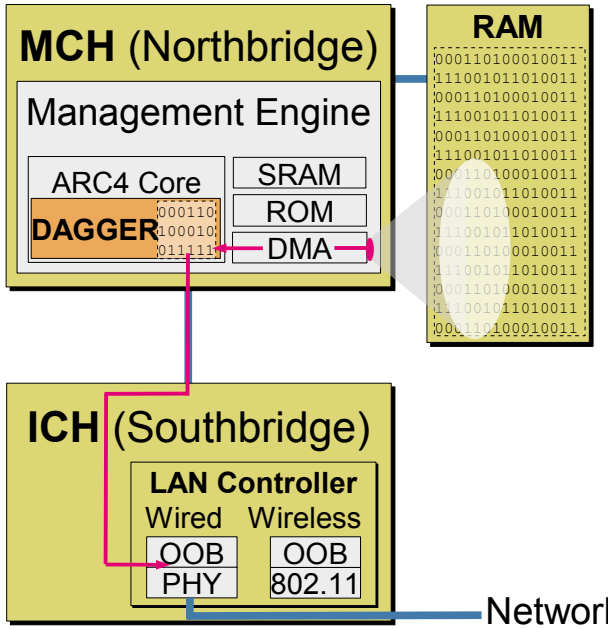
- **Core functionality**
  - Virtual/physical memory address mapping
  - Overcoming address randomization
  - Search space restriction

Comparison of DMA Attacks

	Malware Functionality	Without Physical Access	Rootkit/Stealth Capabilities	Survives Reboot/ Standby/ Power off
USB [Maynor '05]	no	no	no	yes
Firewire [Dornseif et al.'04/'05] & [Boileau'06]	yes	no	yes	yes
PCMCIA [Aumaitre et al.'10]	yes	no	yes	yes
ME [Tereshkin'09]	no	yes	no	(yes)
NIC [Duflot et al.'10]	yes	yes	yes	yes
NIC [Delugre'10]	yes	yes	yes	yes
Video+NIC [Triulzi'08/'10]	yes	yes	yes	yes
this work	yes	yes	yes	(yes)

DAGGER – Our DMA Malware Example

- *DmA based keyloGGER*
- Implements all three phases
  - Search keyboard buffer
  - Monitor keyboard buffer
  - Exfiltrate keystroke codes
- Evaluation of core functionalities
- Proof of concept for stealing short-living runtime data stealthily!
- Infiltration via security vulnerability



DAGGER Monitoring the Host's Keyboard Buffer

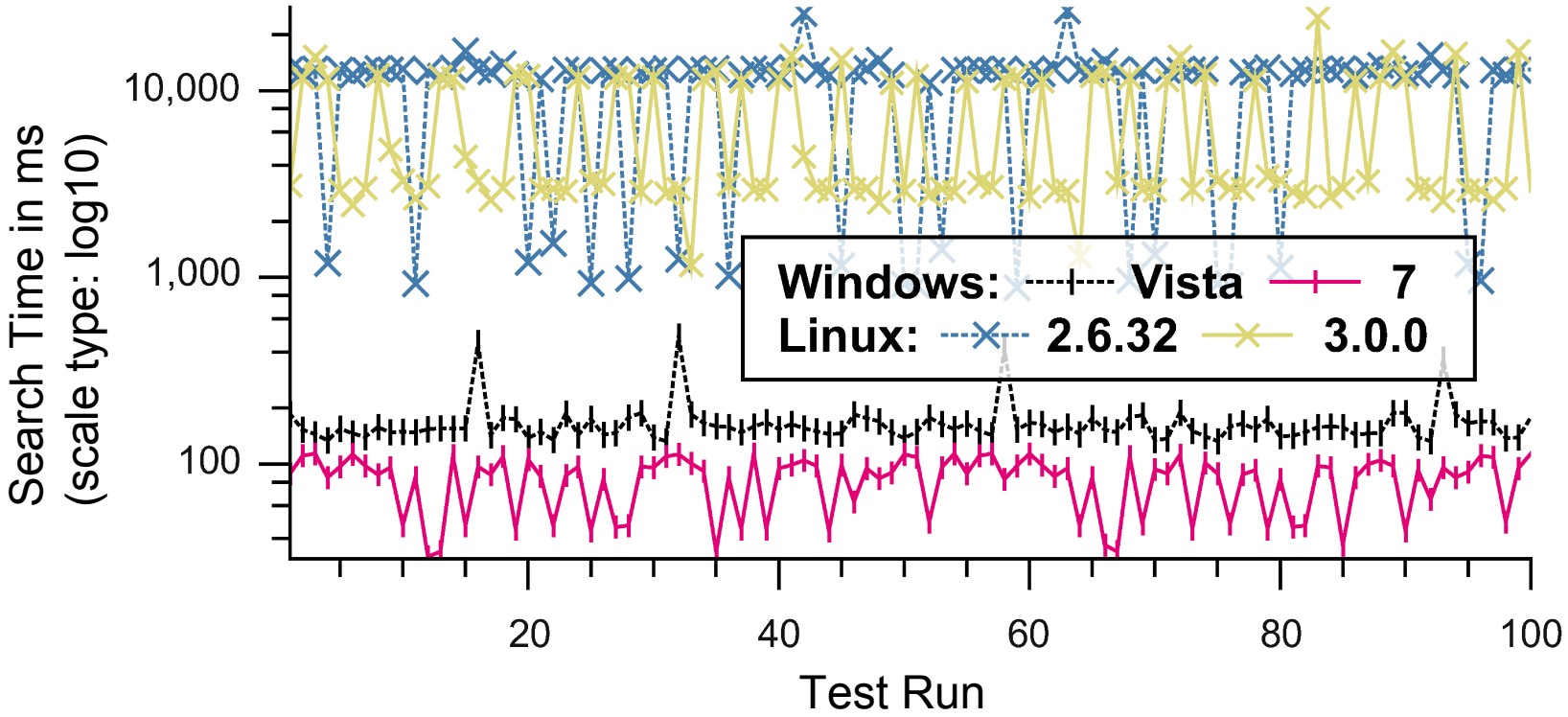
DAGGER Implementation

- Different search strategies

	Windows	Linux
Virtual/physical memory address mapping	page tables	offset
Address randomization	randomization mechanism in place	no randomization
Search space restrictions	*Object Manager Namespace Directory*	address ranges

- Platform: Intel Q35 chipset, 2GB RAM, 4-core 3GHz CPU

Evaluation – Several Operating System Kernels

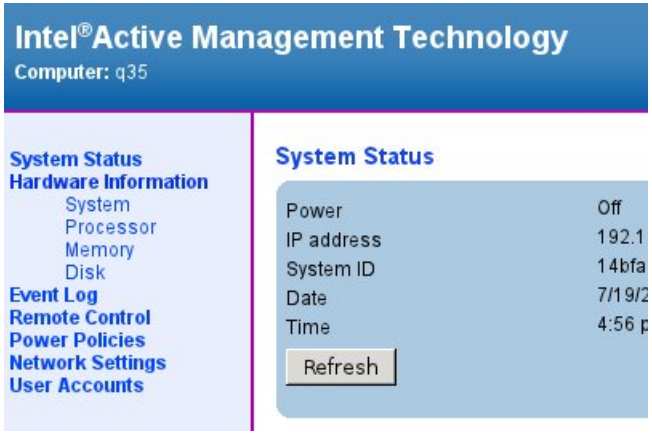


Evaluation – Attacking Linux Harddisk Encryption

- *Aggressive search mode*
- Linux Unified Key Setup (LUKS)/ Device Mapper's crypt (dm-crypt) setup
- DAGGER can catch pre boot authentication passphrase

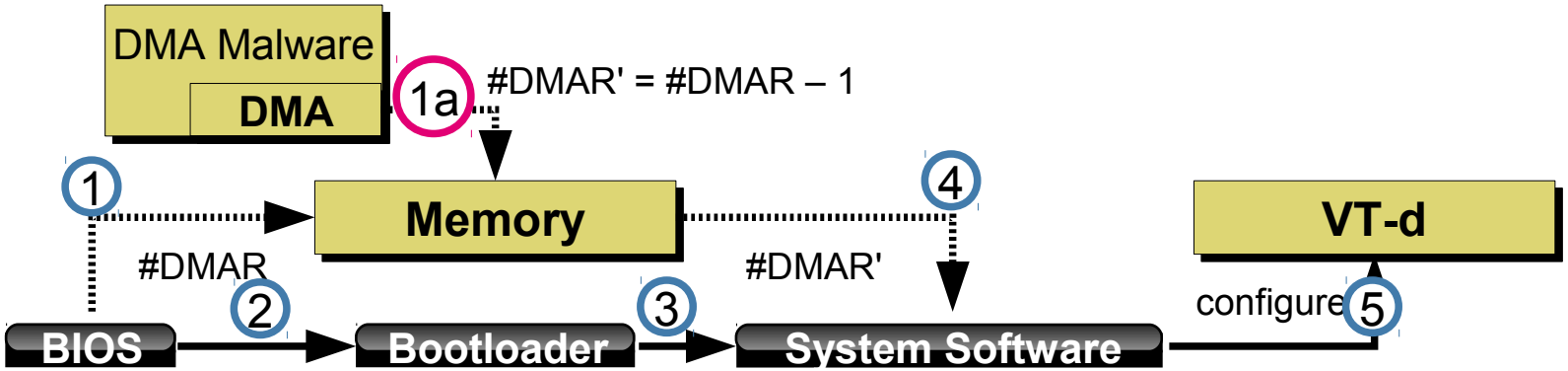
Evaluation

- Anti-virus software, firewalls, Wireshark, Mamutu, etc.
- Several USB keyboards
- Windows swap behavior
- Performance overhead for host system
- Manageability Engine firmware condition
  - Status tools
  - Active Management Technology webserver



Countermeasures

- *Input/Output Memory Management Unit (I/OMMU)*
- Intel: *Virtualization Technology for Directed I/O (VT-d)*
- Issues
  - Missing (Windows) or experimental (Linux) drivers
  - *CoPilot* [Petroni et al.'04]/*DeepWatch* [Bulygin'08] or DAGGER? → policy conflict
  - Attack with DAGGER's execution environment



DMA Malware Modifying the Number of DMA Remapping Engines (DMAR, part of VT-d)

Conclusion

- DMA Malware definition
- Focus on stealth attacks
- Evaluation of DMA Malware core functionalities
- DMA Malware is
  - Effective
  - Efficient enough for real attacks
- Specialized countermeasures must be developed

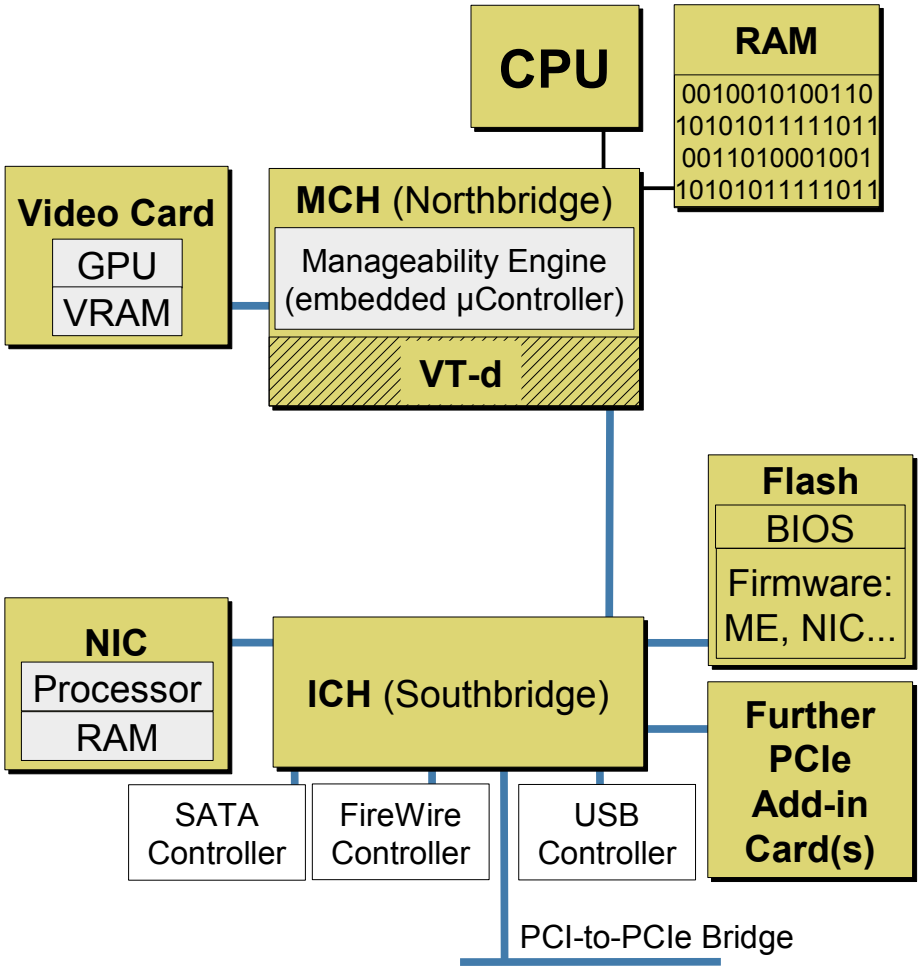
Understanding DMA Malware

DIMVA2012 – 9th Conference on Detection of Intrusions and Malware & Vulnerability Assessment

Patrick Stewin and Iurii Bystrov, July 26th, 2012, Heraklion, Greece
patrickx@sec.t-labs.tu-berlin.de

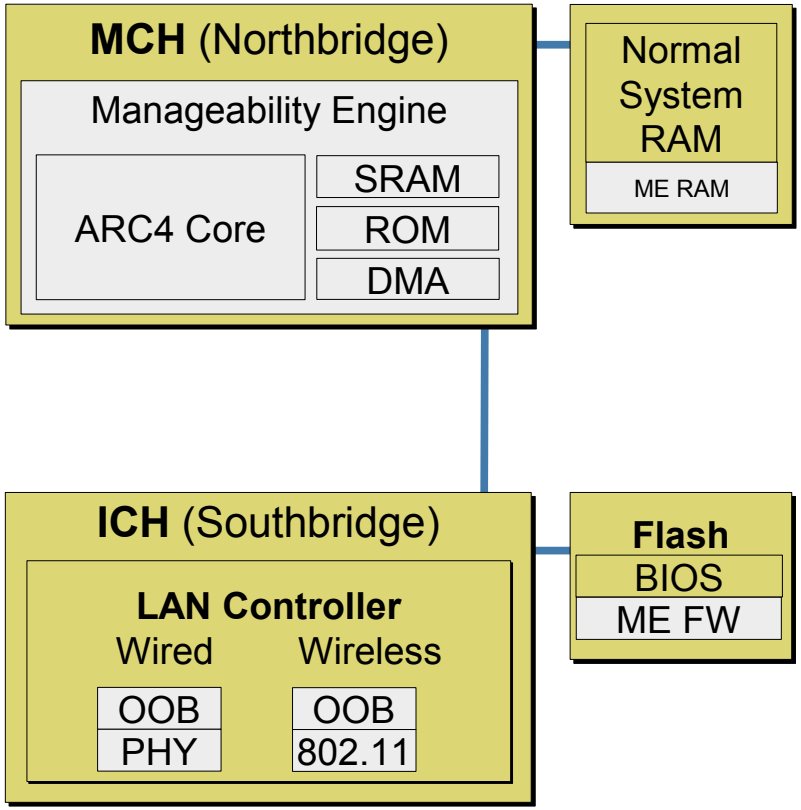
Backup

Background – x86 Platform

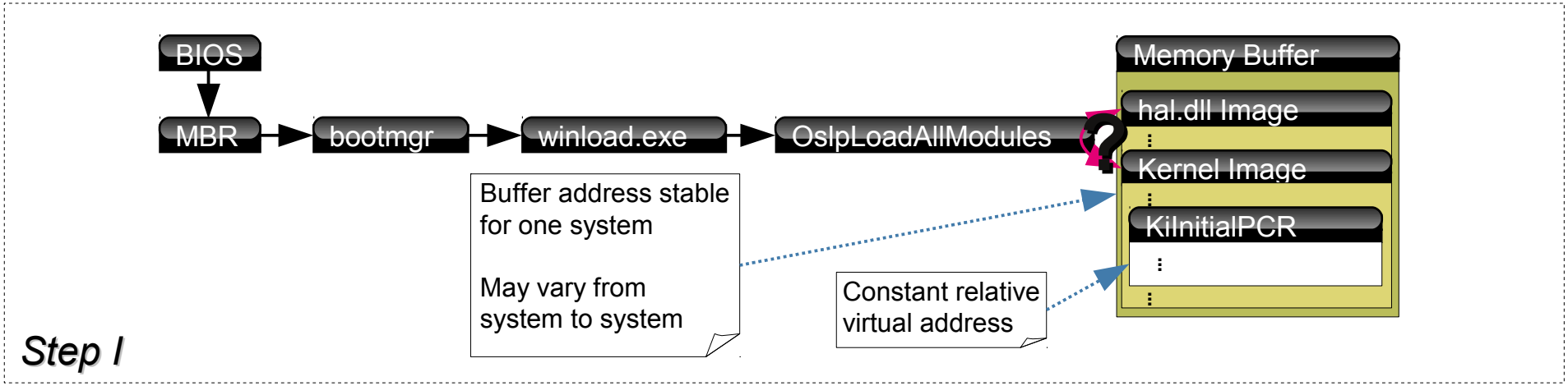


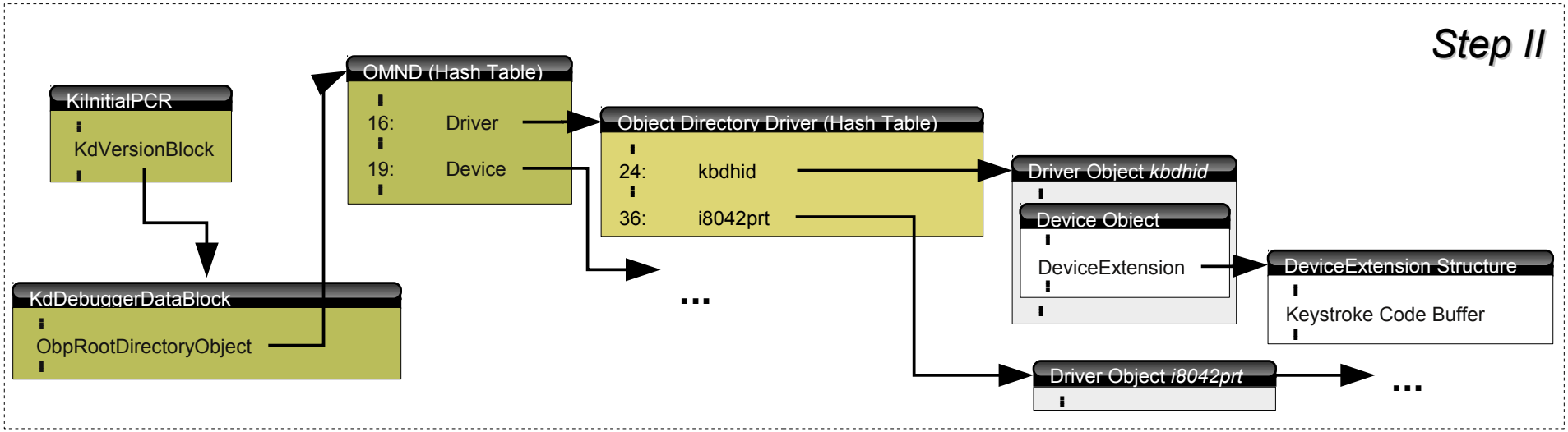
DAGGER – Example DmA based KeyloGGER Malware

- Implementation based on Intel Manageability Engine (ME)
  - Executes firmware such as *Active Management Technology*, *Identity Protection Technology*, *Integrated Trusted Platform Module*, etc.
- Objectives
  - Find keyboard buffer
  - Permanently monitor keyboard buffer
  - Exfiltrate keystroke codes

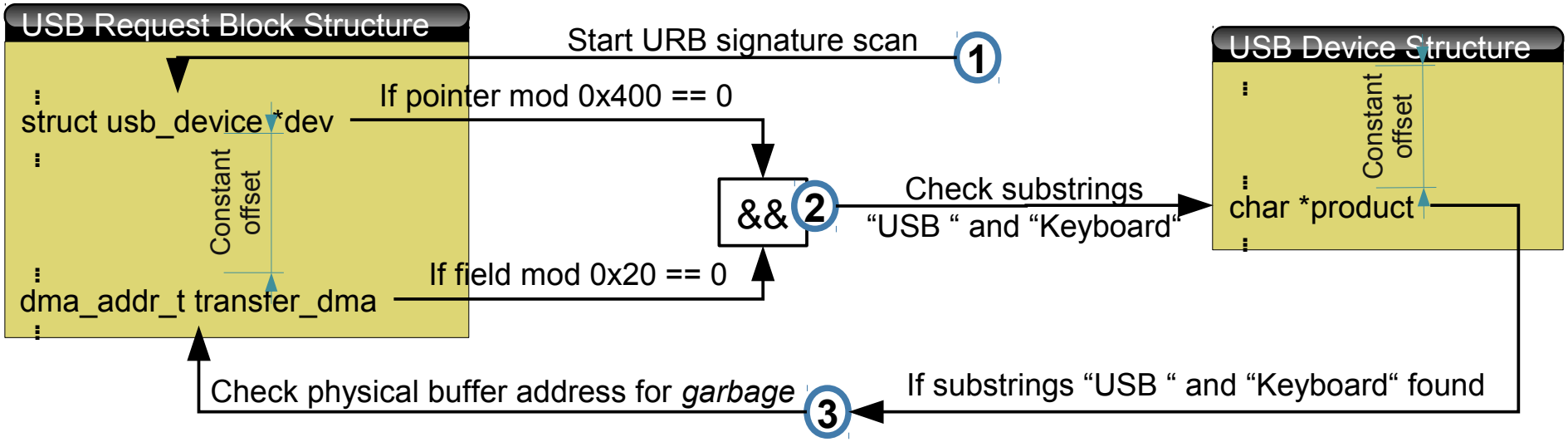


Windows Attack Details

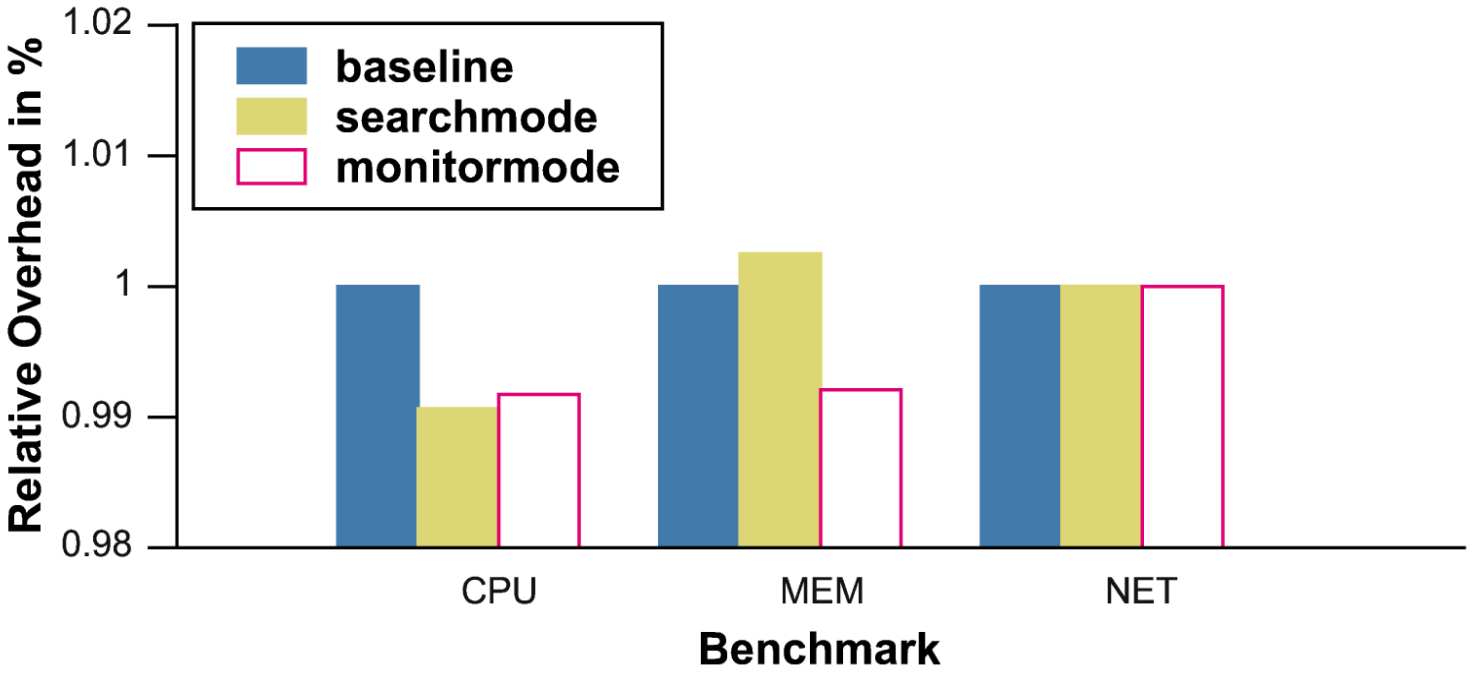




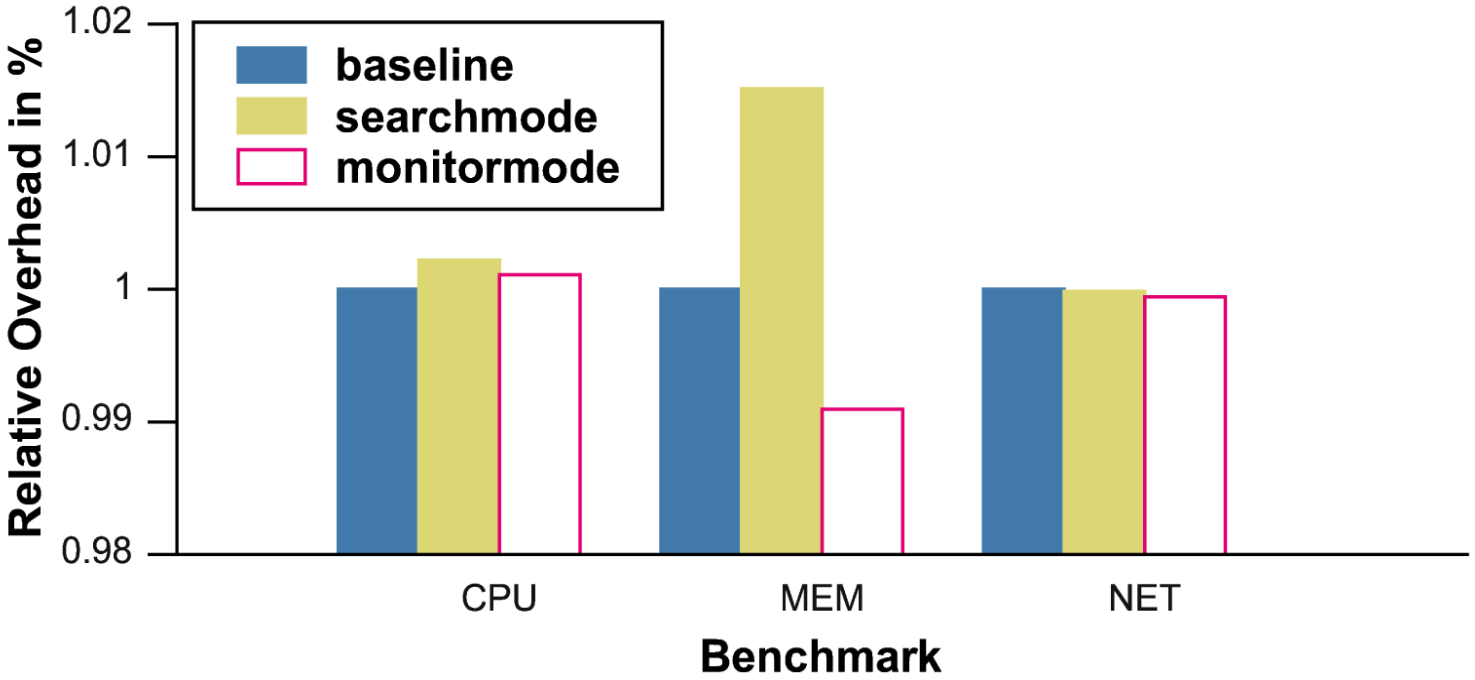
Linux Attack Details



DAGGER Evaluation – Performance Overhead, Windows Host



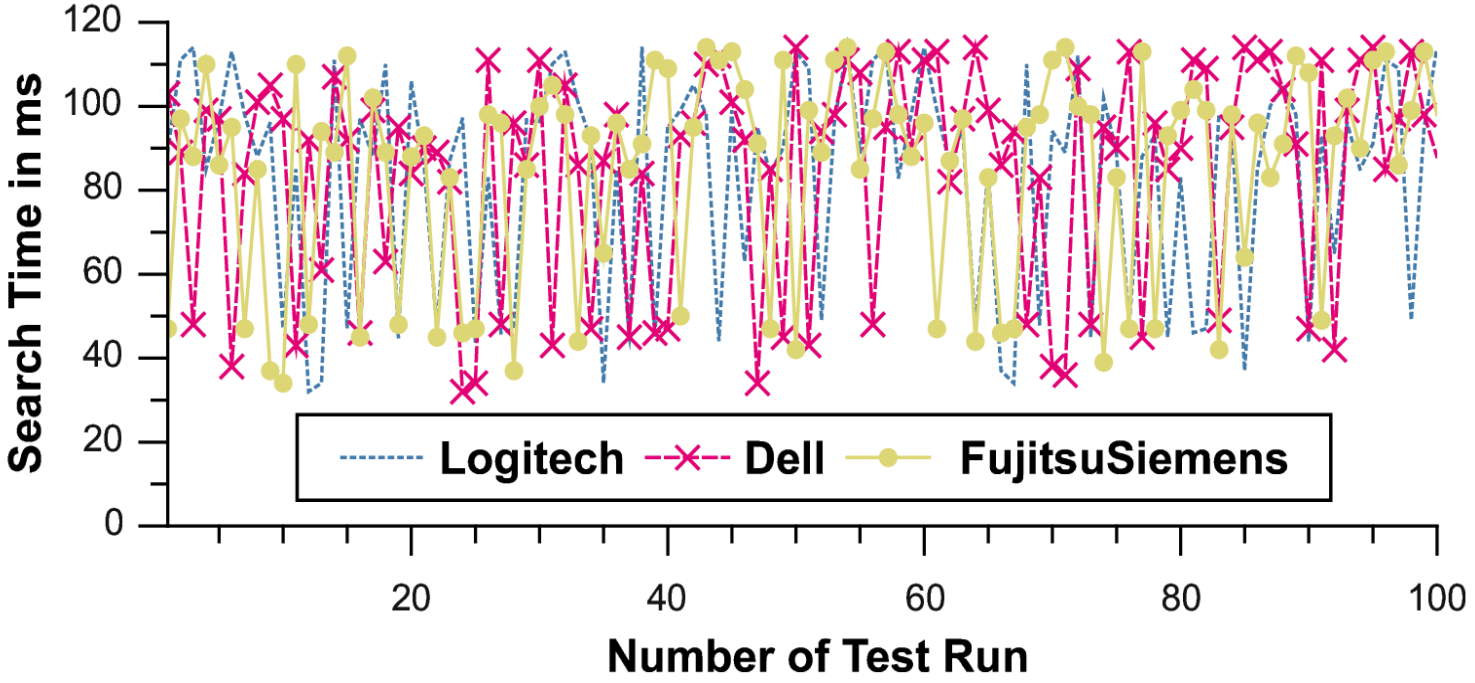
Evaluation – Performance Overhead, Linux Host



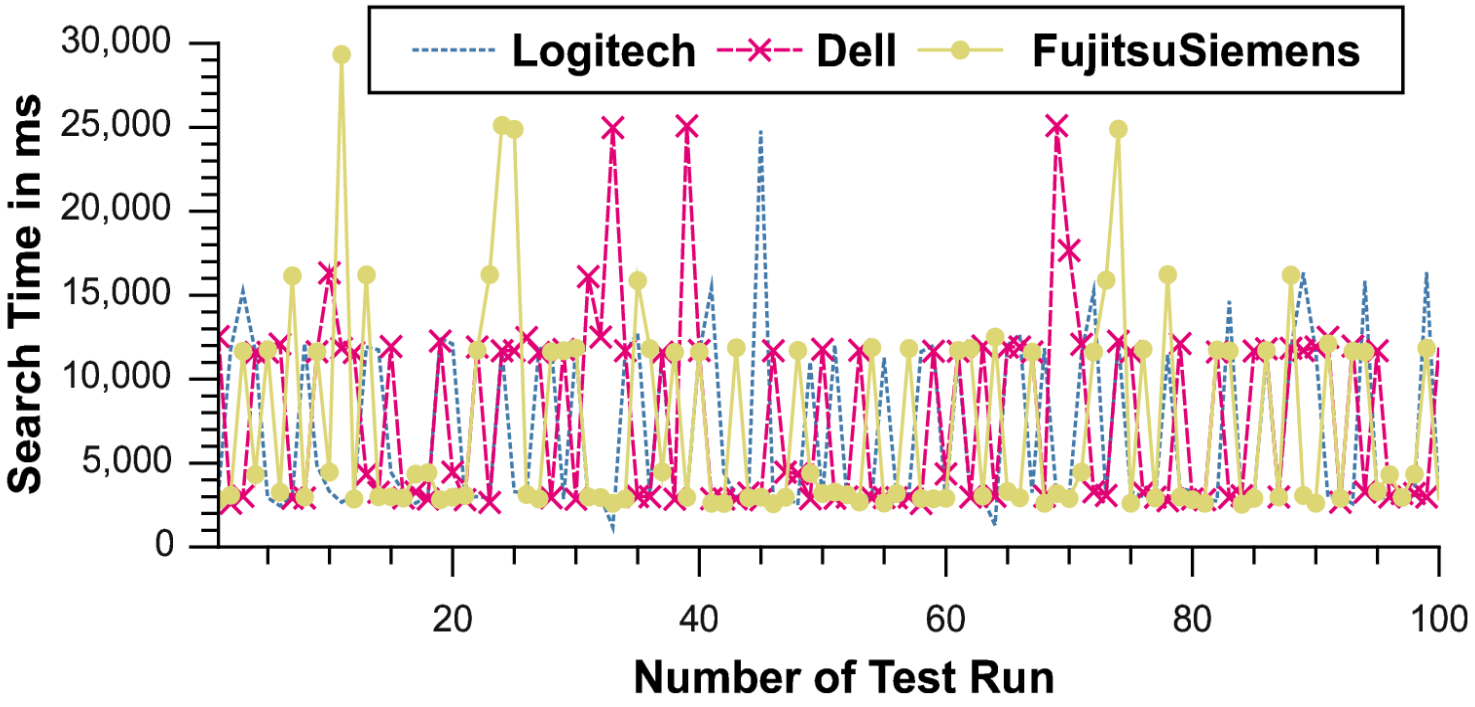
Evaluation – Effectiveness and Efficiency

- Several Operating System Kernels
  - Windows 7
  - Windows Vista
  - Linux 3.0.0
  - Linux 2.6.32
- Several Keyboards
  - Logitech
  - Dell
  - FujitsuSiemens
- Swap file behavior
  - Windows 7

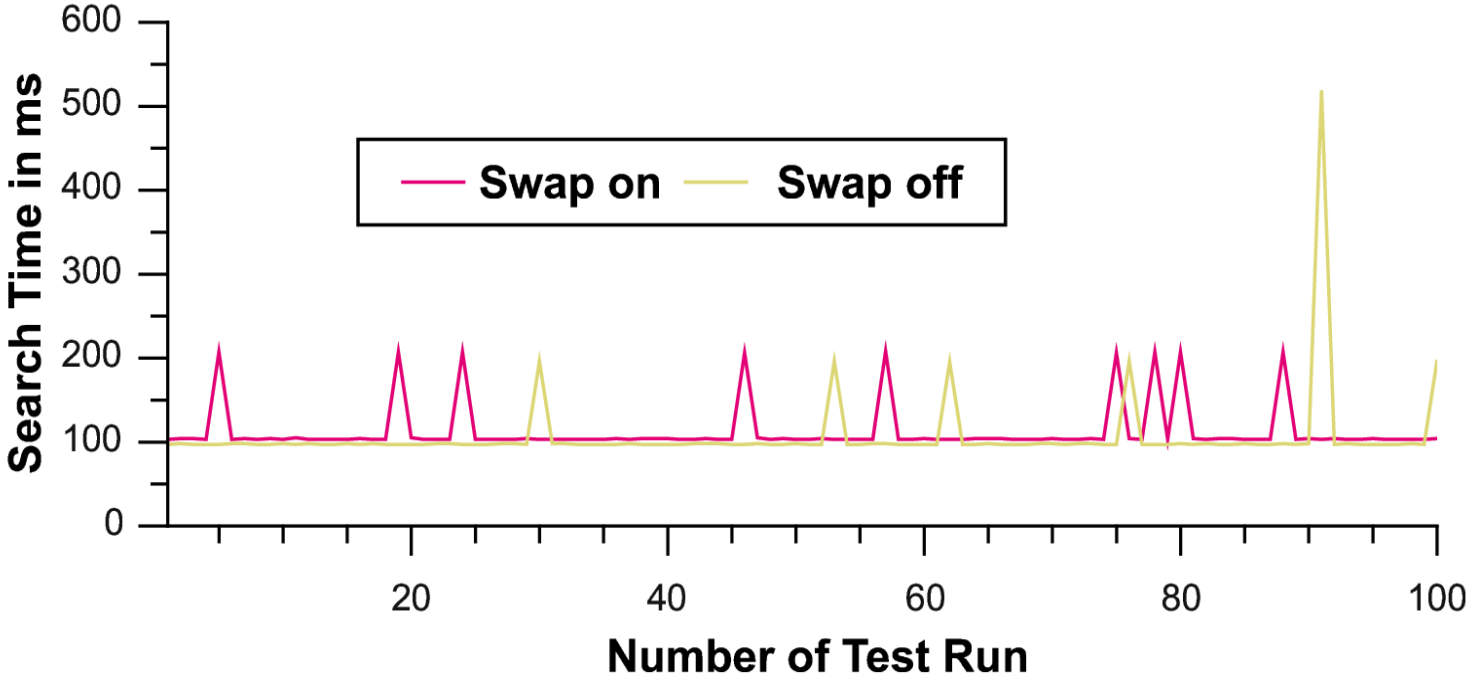
Evaluation – Several Keyboards, Windows Host



Evaluation – Several Keyboards, Linux Host



Evaluation – Swap File Behavior, Windows Host



Evaluation – ME Firmware Condition

- Different hooking strategies for Windows and Linux attacks
- Windows
  - *Local Manageability Service* driver
    - *AMT Status Tool*
  - *Manageability Developer Toolkit*
    - *Manageability Connector Tool*
- Linux
  - *Intel AMT Open-source Tools and Drivers*
    - *ME Status*
    - *ZTCLocalAgent*
- AMT webserver