THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

MAYALOK: A Cyber-Deception Hardware Using Runtime Instruction Infusion

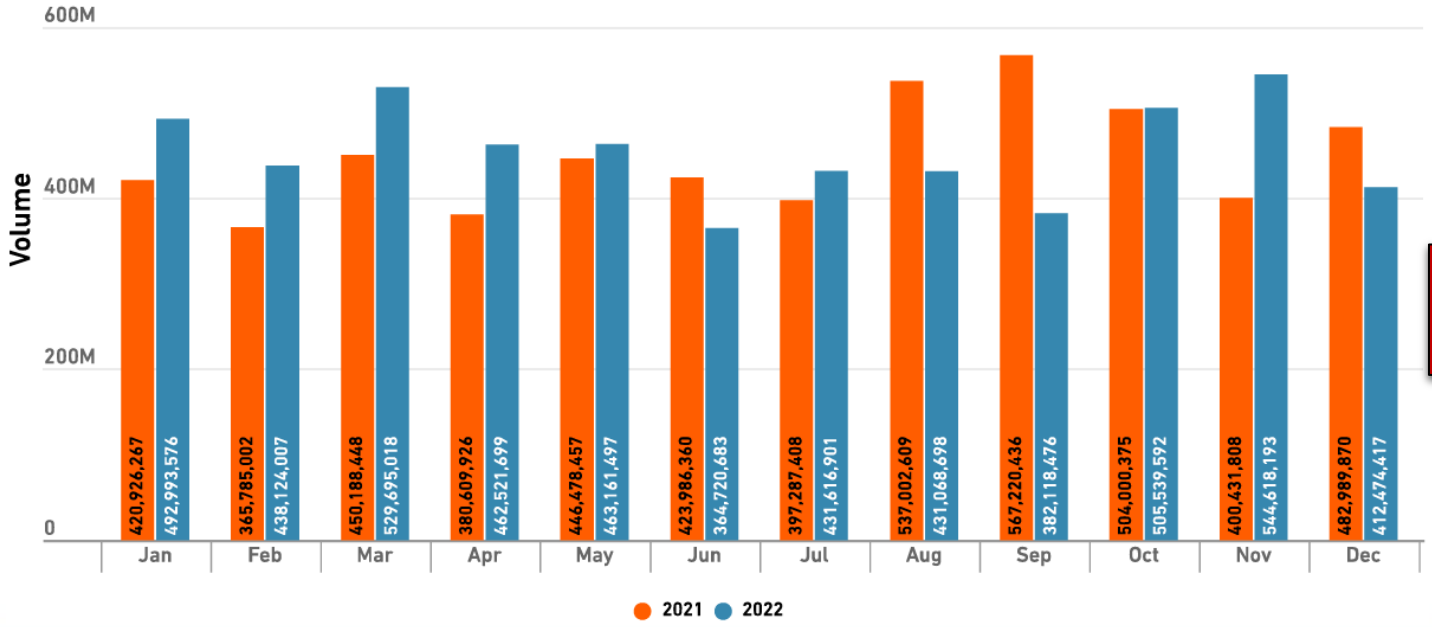
Preet Derasari, Kailash Gogineni, and Guru Venkataramani
Department of Electrical and Computer Engineering
The George Washington University
Washington, D.C., USA

GW Department of Electrical & Computer Engineering
School of Engineering & Applied Science

Evolving Digital Threats

Malware facts of 2022[1]:

- A total of **5.5 billion** attacks attempted
- **465,501** were never-before-seen malware variants
- Ransomware held majority at **493.3 million**
- Other prominent threats: cryptojacking, IoT malware, malicious PDFs, encrypted attacks



Total volume of attacks in 2021 and 2022 is **10.9 billion**!

[1] "2023 SonicWall Cyber Threat Report", SionicWall Whitepaper, 2023.

Reactive Defenses Against Malware

Detect-and-Stop

- *Signature-based* → binary analysis, hash-check, fingerprinting
- *Trace-based* → runtime logs of syscalls, instructions, network packets
- × Fails against unknown malware due to static nature
- × Unfeasible runtime cost

System Hardening

- *Software* → compiler optimizations, feature reduction, digital signatures
- *Hardware* → isolated enclaves, memory protections, cryptographic modules
- × Patching after deployment can be impossible
- × Risks exposing defense mechanism to attackers

Cyber-Deception

Proactive in protecting sensitive assets

Interact with adversaries using honey-resources

Profile attackers to build robust defenses

Why Cyber-Deception?

Prevention

Protect sensitive system assets

- **Slow down** attackers with bloated system assets
- **Redirect** malicious actions to honey-resources
- *e.g., honey-pot servers, moving target defenses*

Engage

Sup

- **Confuse** attackers with mani
- **Influence** the attack path tow
- *e.g., instrumenting malware binaries, software honey-patches*

Current techniques are underwhelming in their ability to provide **efficient** and **effective** cyber deception!

Research

Preserve attack forensics

- **Study** the collected data to create robust system designs
- **Reinforce** security measures with high-fidelity attacker details
- *e.g., honey-VMs used as sandboxes*

Related Efforts

- Qassrawi and Hongli, “Deception methodology in virtual honeypots,” IEEE NSWCTC, 2010.
  - Honeypot VMs for attacker interaction
  - Vulnerable to artifact discovery techniques
- Sajid et al., “Soda: A system for cyber deception orchestration and automation,” ACM ACSAC, 2021.
  - API hooking-based malware instrumentation
  - High execution time overheads and limited to API calls
- Gallagher et al., “Morpheus: A vulnerability-tolerant secure architecture based on ensembles of moving target defenses with churn,” ACM ASPLOS, 2019.
  - Hardware-based obfuscation using MTD ensembles
  - High impact on performance of all applications

Supporting Cyber-Deception

Dynamic

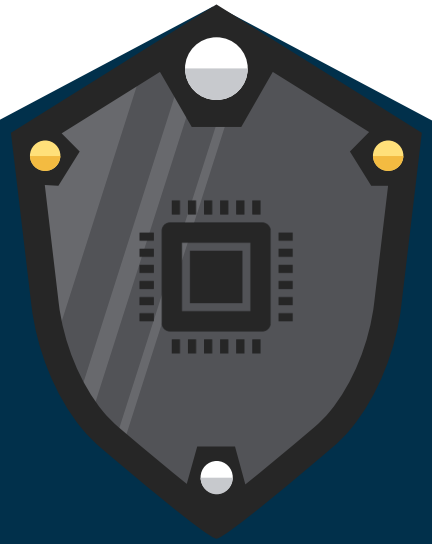
Implement deception tactics during malware execution

Coverage

Target multiple applications with the same deception platform

Independent

Modify malware execution path with minimal external dependencies



Hardware provides these features!

Low-overhead

Sustain a nominal runtime cost for implementing deception tactics

Transparent

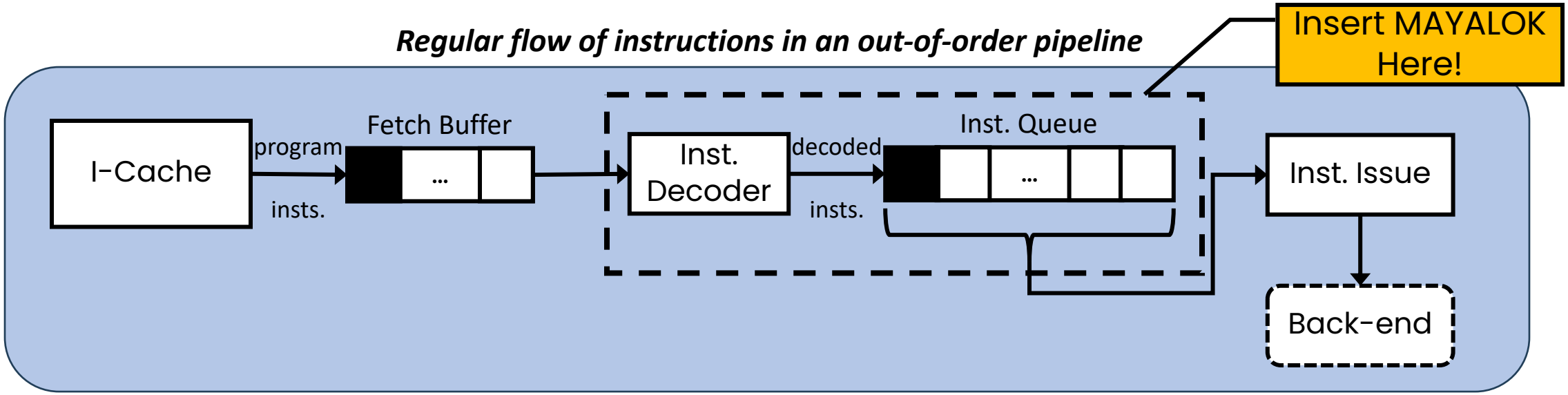
Deception activity should be invisible to suspecting adversaries

Customizable

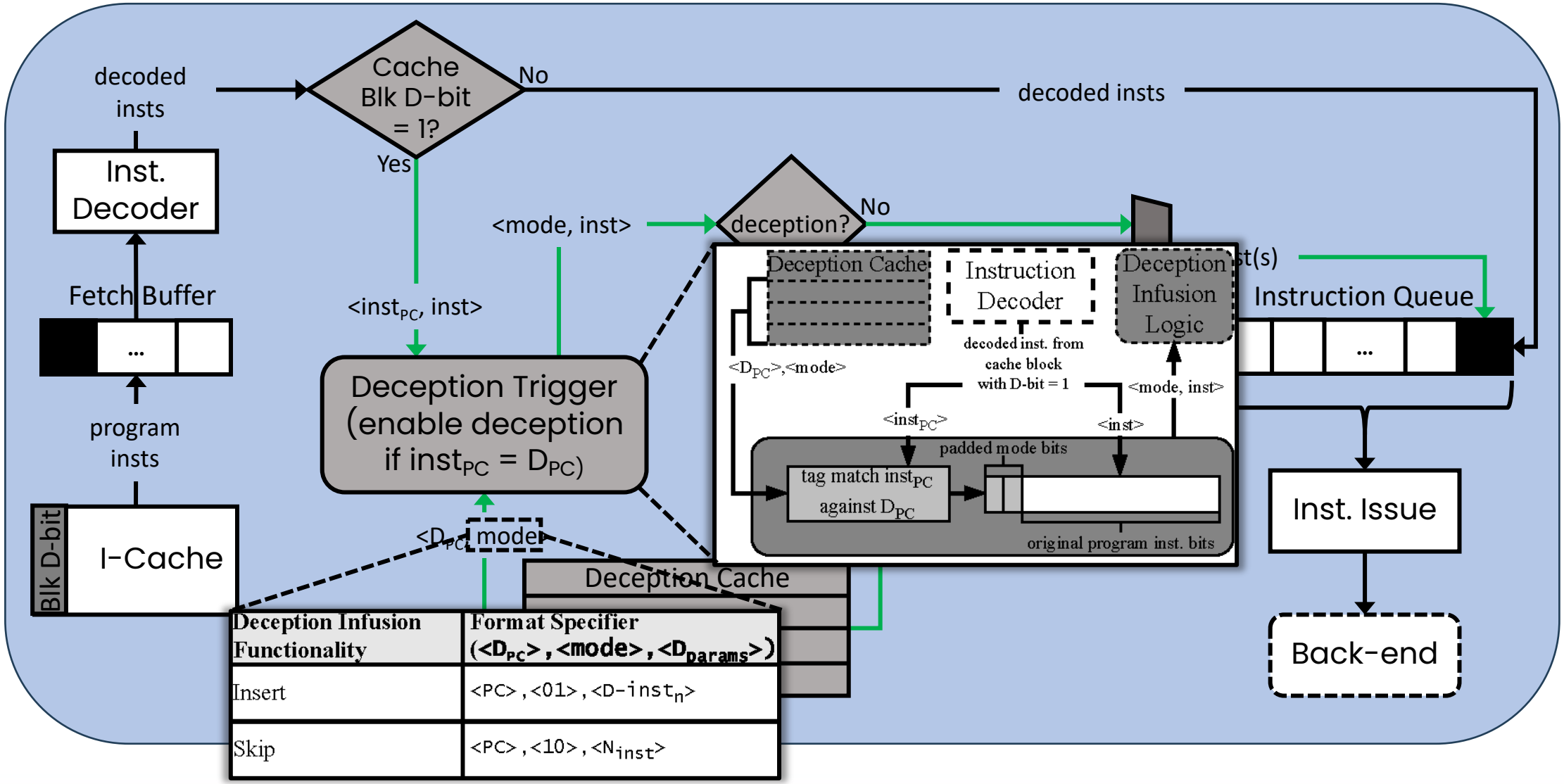
Provide the ability to alter deception tactics depending on malware type

The MAYALOK* Cyber-Deception Hardware

- A **front-end** implementation in the processor pipeline:
  - ✓ Each application executing **malicious instructions** is tagged
  - ✓ Instructions are **dynamically** inserted/removed
  - ✓ At the **fetch/decode** stage
  - ✓ Preserves the **efficiency** of processor pipeline for non-deception instructions



MAYALOK in Sanskrit means an imaginary realm


decoded insts
Cache Blk D-bit = 1?
No
decoded insts
Yes
Inst. Decoder
Fetch Buffer
...
program insts
Blk D-bit
I-Cache
<mode, inst>
deception?
No
<inst_PC, inst>
Deception Trigger (enable deception if inst_PC = D_PC)
<D_PC, mode>
Deception Cache
Deception Cache
Instruction Decoder
Deception Infusion Logic
decoded inst. from cache block with D-bit = 1
<D_PC>,<mode>
<inst_PC>
<inst>
<mode, inst>
padded mode bits
tag match inst_PC against D_PC
original program inst. bits
st(s)
Instruction Queue
...
Inst. Issue
Back-end
Deception Infusion Functionality
Format Specifier (<D_PC>,<mode>,<D_params>)
Insert
<PC>,<01>,<D-inst_n>
Skip
<PC>,<10>,<N_inst>

MAYALOK Hardware Overhead

- MAYALOK hardware structure with the highest overhead is the Deception Cache:
  - Requires a **2KB** cache to store relevant **D_{PC} and D_{params}**
  - Supports **16 entries** of **128 bytes** each for a malware process
  - Sufficient to track most malicious **$inst_{PC}$** based on our analysis of popular malware samples
- Instructions targeted for infusion are tagged pre-runtime
  - Using binary analysis tools like Angr
  - Optimizes the deception trigger mechanism

(All experiments were conducted on an x86 build of the Gem5 simulator clocked at 2GHz)

Security Analysis on Malware

Malware Sample (Type)	Security Outcome	Infusion Mode	Attacker Penalty
Return-to-libc (Buffer Overflow)	Pointer Protected	Insert	Failed overflow
RSA Timing Attack (Side Channel)	Keys Protected	Insert	No Side Channel
Petya (Ransomware)	Files Protected	Insert & Skip	∞ time
Credential Stealer (InfoStealer)	Credentials Protected	Insert	Fake data supply

Conclusion

- ❑ Showed the promise of a new security paradigm i.e., **cyber-deception**
- ❑ Presented an **efficient** hardware support for runtime **instruction infusion**
- ❑ Demonstrated the **effectiveness** of MAYALOK against multiple malware samples

Future of hardware in cyber-deception:

- ✓ **Additional primitives** for improved cyber-deception support
- ✓ **Automating** cyber-deception to adapt itself with new attack vectors

"MAYAVI: A Cyber-Deception Hardware for Memory Load-Stores", ACM GLSVLSI. Knoxville, Tennessee. June 5 – 7, 2023.

Thank You

Preet Derasari, Kailash Gogineni, and Guru Venkataramani

Reach out for questions: **preet_derasari@gwu.edu, guruv@gwu.edu**

Department of Electrical & Computer Engineering
School of Engineering & Applied Science

This research was supported by the U.S. Office of Naval Research