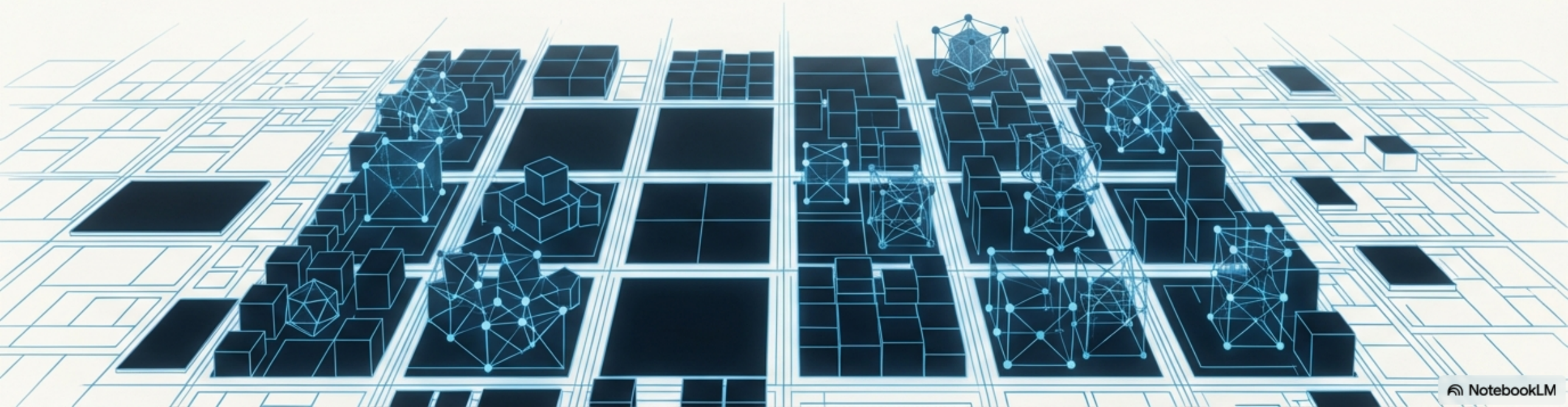
Memory Doesn’t Lie

Hunting Modern Malware Beyond the File: A Comparative Study of Amadey Detection and the Amadey Pipeline Toolkit (APT)

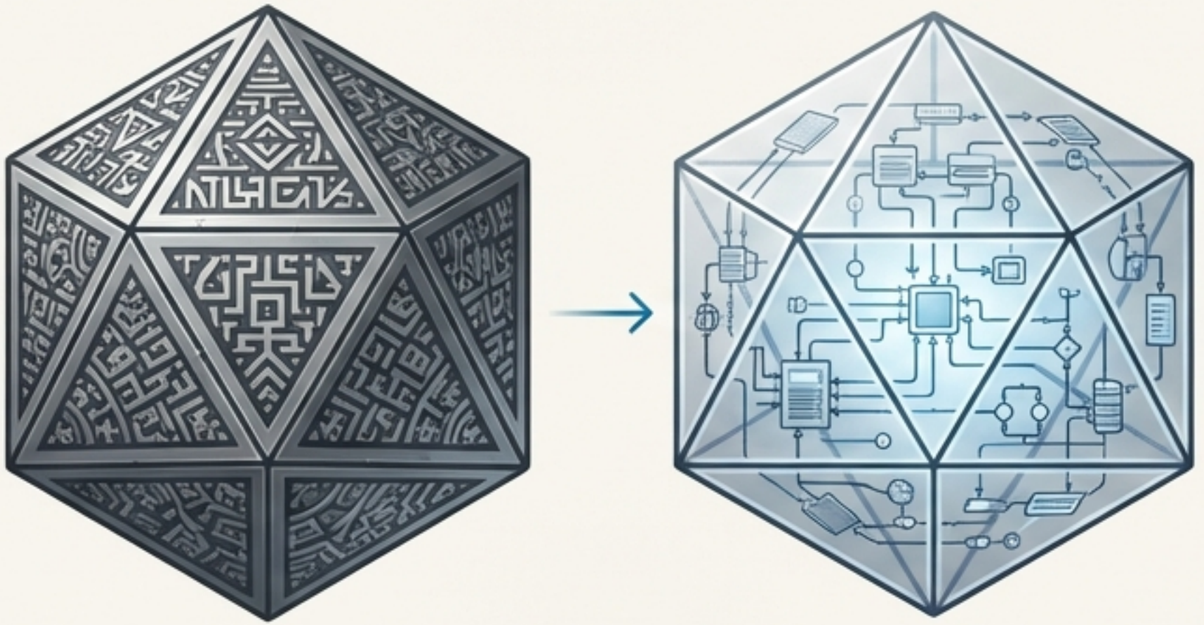


Modern Malware Hides in Plain Sight

Static analysis, which inspects files at rest, is easily defeated by common obfuscation techniques, creating a critical blind spot for defenders.

Static Detection's Blind Spot

- Analyzes files on disk, looking for known byte patterns (signatures).
- Fundamentally vulnerable to evasion via **Packers**, **Encryption**, and **Polymorphism**.



Packed/Encrypted File on Disk

Unpacked Logic in Memory

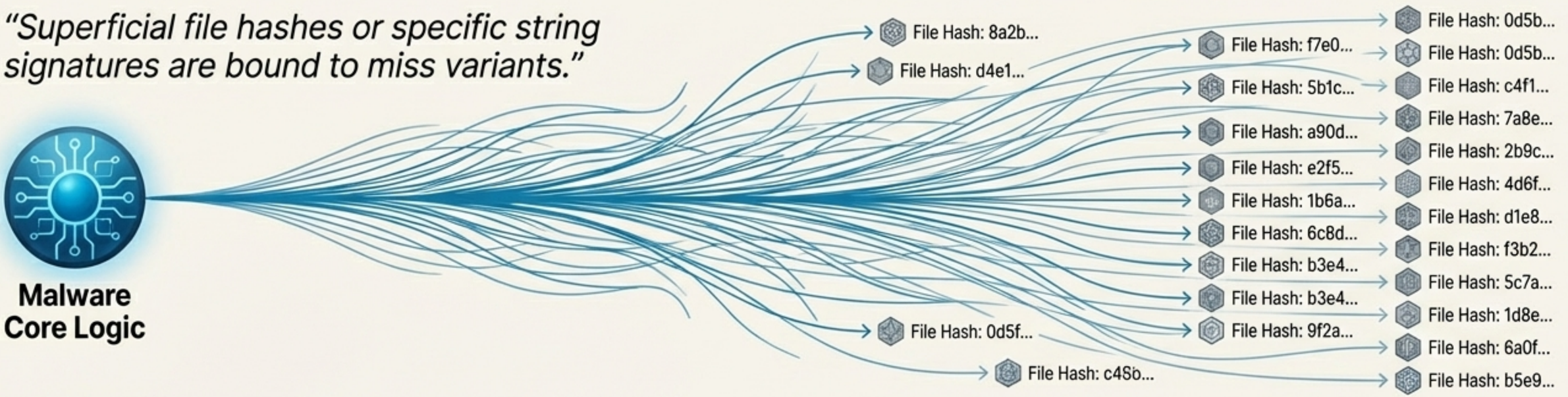
Case Study: Amadey's Deception

- Amadey employs a **double-encoding scheme (custom + Base64)** to hide all critical strings and its malicious payload.
- This renders traditional, **signature-based static analysis ineffective.**

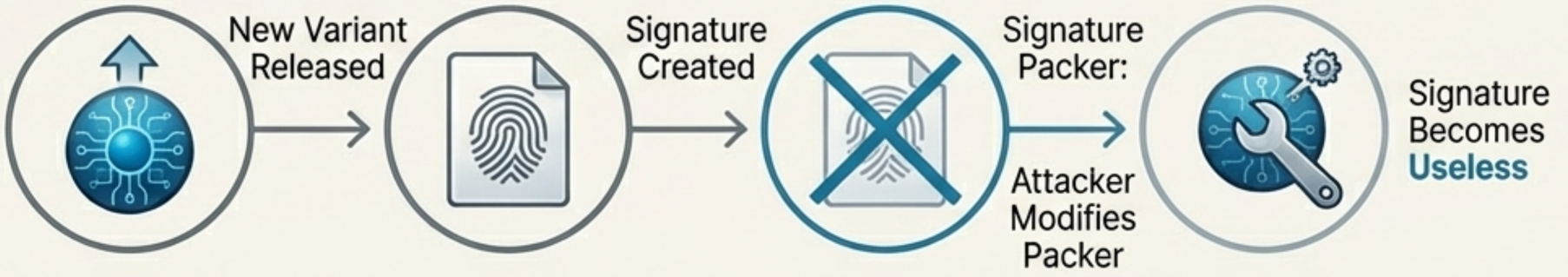
Why Signature-Based Defenses Are a Losing Battle

Attackers can generate endless new file variants with minimal effort, making hash-based blocklists and static signatures a strategy of diminishing returns.

"Superficial file hashes or specific string signatures are bound to miss variants."



The Vicious Cycle (Illustrated)



The Hunt Moves to Memory

To uncover a malware's true intent, we must analyze it during execution. Memory forensics bypasses all layers of on-disk obfuscation.

A Fundamental Shift in Perspective

- **Static Analysis Asks:**
  "What does this file *look like*?"
- **Memory Analysis Asks:**
  "What is this process *actually doing*?"

On Disk

In Memory

Core Principle

Malware must eventually decrypt and expose its true payload in memory to function. Memory analysis captures this unpacked, operational state.

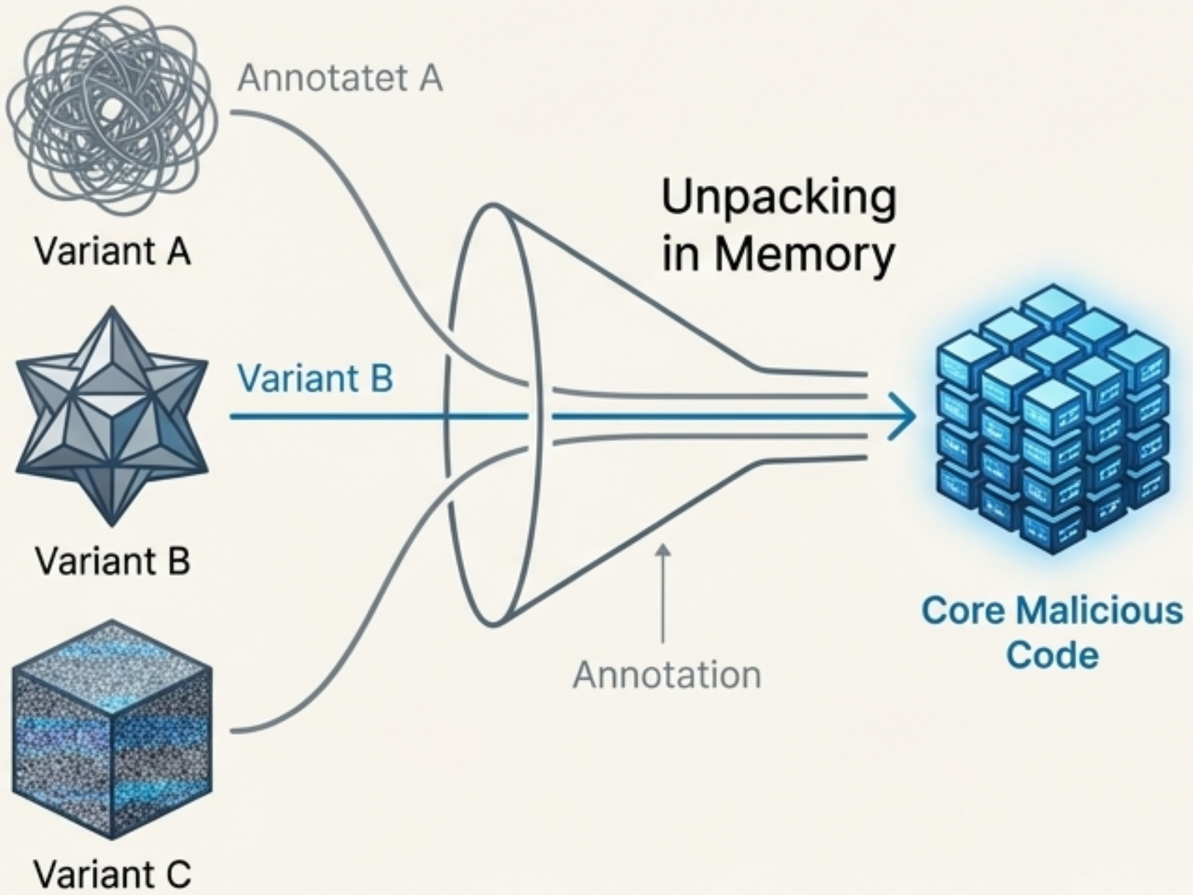
Discovering ‘Execution Invariants’

Regardless of on-disk obfuscation, a malware’s core functions create unavoidable and repeatable patterns—its true fingerprints—in memory.

Defining Invariants: The Attacker’s Achilles’ Heel

Definition: Core code or data patterns that are essential for the malware’s function and remain consistent across variants.

Why They Exist: Attackers must reuse core logic for efficiency, stability, and profitability (especially in Malware-as-a-Service models).



Common Examples

Hardcoded decryption routines.

Target data strings (e.g., `cookies.sqlite`).

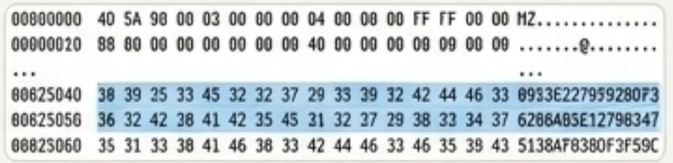
Unique sequences of API calls.

Amadey's Invariants: Exposing Its DNA in Memory

Despite its double-encoding, Amadey leaves behind distinct, repeatable traces that are perfect for creating robust detection rules.

Decryption Keys

Multiple hardcoded 32-character hexadecimal strings used as decryption keys become visible in memory after unpacking.



Developer Artifacts

The debug path string `\Amadey\Release\Amadey.pdb` often persists in memory across repacked variants, acting as a stable signature.

`\Amadey\Release\Amadey.pdb`

Immutable Logic (Code Snippet)

This unique x86 instruction sequence represents a core function and appears consistently. It can be captured as a byte pattern.

```
; Amadey Memory Snippet
mov     [ebp-0xC], eax
cmp     [ebp-0xC], 8
je      ...
lea     eax, [ebp-0x218]
```

Memory Convergence: A Cross-Family Phenomenon

The principle of seeking execution invariants applies broadly. Different malware families reveal their own characteristic behaviors and artifacts in memory.

Amadey (Loader)

Consistent C2 communication structures and unique string decoding logic.

RedLine (Stealer)

Plaintext strings revealing target browser paths (`cookies.sqlite`) and cryptocurrency wallet keywords.

FormBook (Loader)

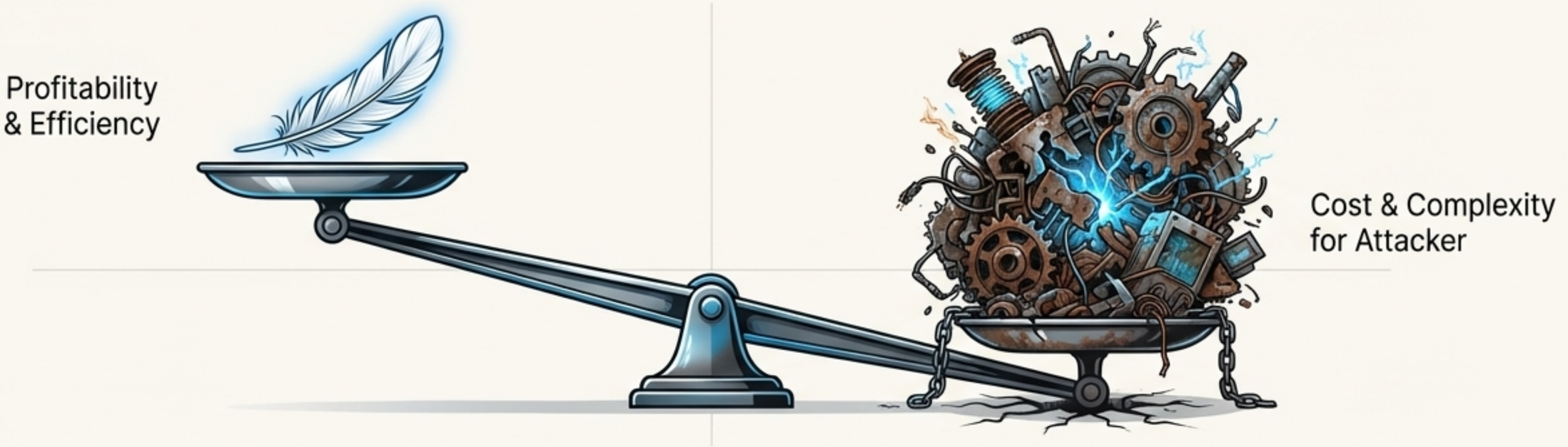
Distinctive API call patterns (e.g., `VirtualAllocEx`, `WriteProcessMemory`) related to process injection from its 'Babushka Crypter'.

SmokeLoader (Loader)

A common trait of injecting its main payload module into the `explorer.exe` process.

The Attacker's Dilemma: The High Cost of True Evasion

While erasing memory footprints is theoretically possible, it imposes a crippling cost on the attacker in terms of development complexity, performance, and stability.



Evasion Tactic: Metamorphic Engines

Cost: Extreme development complexity, high risk of bugs, and difficult to maintain.

Evasion Tactic: Constant In-Memory Re-encryption

Cost: Severe performance overhead and potential for system instability.

Evasion Tactic: Custom Virtual Machines

Cost: Creates a new, large, and stable signature—the VM interpreter itself.

For most Malware-as-a-Service (MaaS) operations, the ROI for these techniques is too low. Reusing stable, core logic is the profitable and therefore common choice.

Codifying Invariants with YARA for Automated Detection

YARA is the ideal tool for translating our knowledge of execution invariants into powerful, flexible, and scalable detection rules for process memory.

Memory Dump **YARA Scan** **Match on Invariant**

How it Works

- YARA's flexible rule language is designed to find generalized text and binary patterns, making it perfect for catching polymorphic variants based on their core invariants.

Versatile Scanning

- It can scan both live process memory and offline memory dump files, fitting multiple analysis workflows.

The Data Speaks for Itself: Static vs. Memory Detection

A **practical test** on 100 Amadey samples and their corresponding memory dumps shows the **dramatic superiority of memory-based detection**.

STATIC FILE SCAN (100 Samples)

71%

(71 of 100 Samples Detected)

MEMORY DUMP SCAN (63 Dumps)

95%

(60 of 63 Dumps Detected)

The Winning Rules: The most effective memory-based YARA rules were not family-specific signatures, but generic, behavior-based signals like `DetectEncryptedVariants` and `meth_get_eip`, proving the power of hunting for invariants.

Introducing APT: The Amadey Pipeline Toolkit

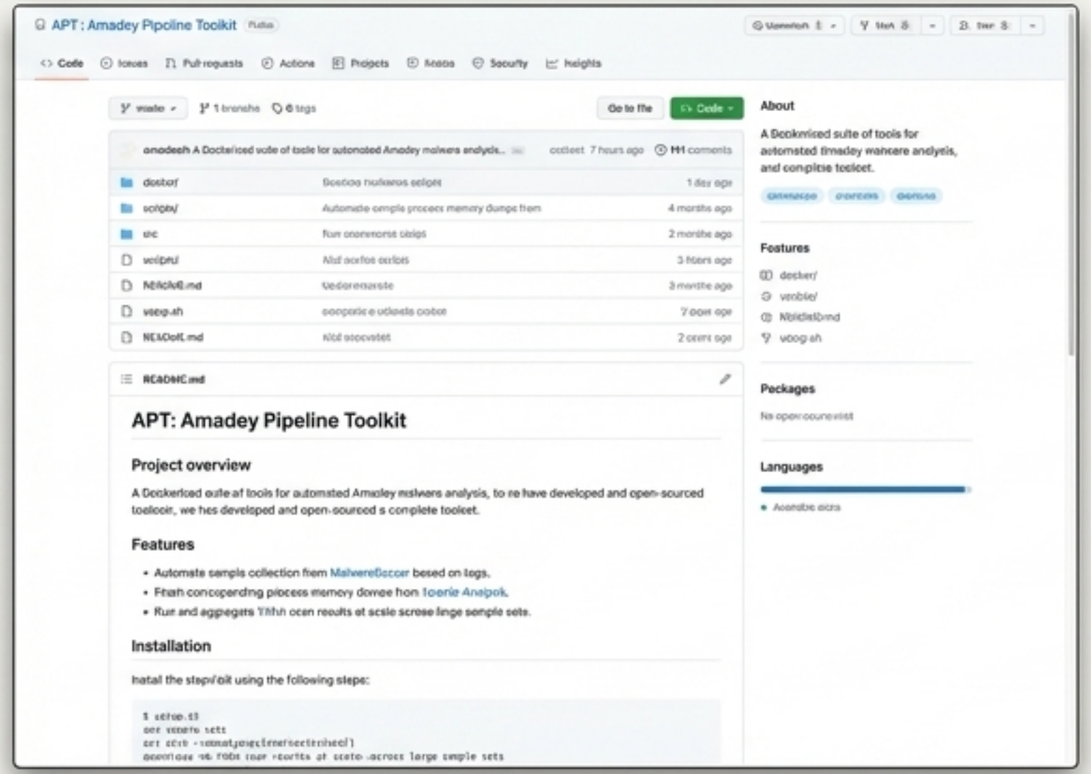
To make this advanced memory analysis workflow repeatable and accessible, we have developed and open-sourced a complete toolset.

What is APT?

- A Dockerized suite of Python and shell scripts designed for safe, isolated malware analysis.

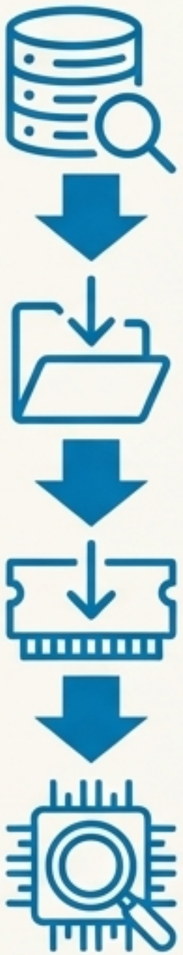
Core Capabilities:

- Automate sample collection from MalwareBazaar based on tags.
- Fetch corresponding process memory dumps from Hybrid-Analysis.
- Run and aggregate YARA scan results at scale across large sample sets.



The Automated Hunt: From Hash to Verdict in Four Steps

APT streamlines the entire investigative process—from sample acquisition to memory scanning—into a simple, command-line driven workflow.



1 Collect Hashes

```
python3 malwarebazaar_hunt.py --tag amadey
```

2 Download Samples

```
python3 malwarebazaar_download.py --file ...
```

3 Fetch Memory Dumps

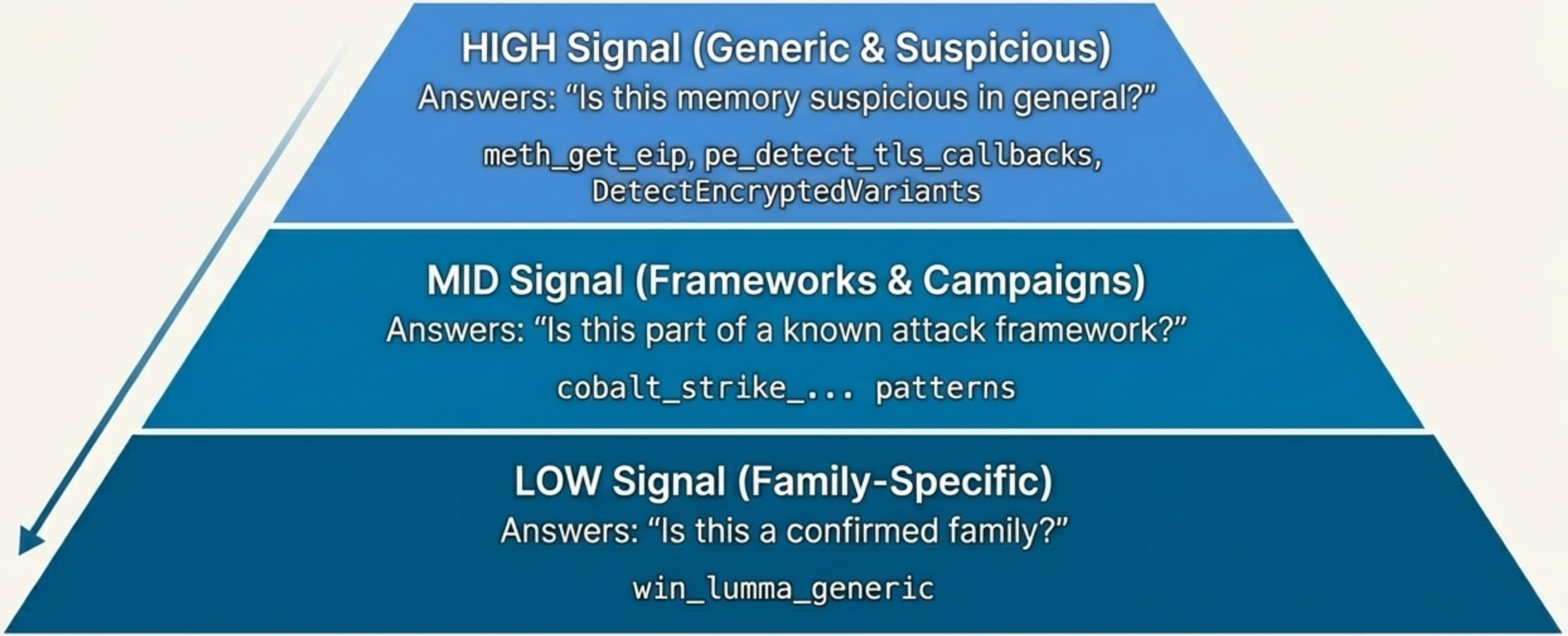
```
bash fetch_memory_dump.sh --sha256-list ...
```

4 Scan & Analyze

```
python3 scripts/yara_eval.py --rules ... --target ...
```

Beyond Detection: Profiling Malware with Layered Signals

Using a broad ruleset like YaraHub, APT enables a multi-layered analysis that describes *how* a sample is suspicious, not just *that* it is.



Memory Doesn’t Lie

The ground truth of malware behavior resides in memory, making it the most resilient and strategic battleground for defenders.

- ✓ Static analysis is fundamentally outmatched by modern, evasive threats like Amadey.
- ✓ Memory analysis, focused on “Execution Invariants,” bypasses obfuscation to reveal the malware's true nature.
- ✓ This strategy is robust, as evading it imposes prohibitive costs on attackers.
- ✓ The APT toolkit provides the practical, open-source means to operationalize this superior approach today.

Get the Tools. Join the Hunt.

github.com/windshock/apt

Clone the repo. Run the pipeline. Hunt smarter.

Questions?