🥁 The Drum-Side-Channel: Temporal Execution Tracing Against Dimensional Memory Attacks

A zero-trust telemetry membrane for every unsafe eval, buffer overflow, or timing-injection

---

1. Threat Model – *"The Beat is the Attack Surface"*

Table

Copy

Attack Vector	**Manifestation in Drum Telemetry**	**Side-Channel Mitigation**
Unsafe eval("a+b→c")	Memory trace leaks into **Temporal Signature**	**Time-Accrete GUID (TAG) injection**
Buffer Overflow	Heap delta ≠ **Dynamic Cost** forecast	**Heap-Trace Hash (HTH)** mismatch alarm
SQL-to-Mongo Injection	eval('SELECT→find()') path divergence	**Query-DNA (qDNA)** grammar fingerprint
Timing Injection	**Silence Length** ≠ expected latency	**Rhythm-Entropy (RE)** entropy spike

---

2. Time-Accrete GUID (TAG) – *Seeded, Traceable, Unforgeable*

c

Copy

```c
// drum_sidechannel.h
typedef struct {
    uint8_t  paradigm;        // 0 = unsafe, 1 = sandboxed, 2 = audited
    uint64_t epoch_ns;        // high-resolution tick
    uint8_t  crypto_seed[32]; // SHA-3-256 of (version + user + jitter)
    uint8_t  heap_hash[32];   // Blake3 of live heap snapshot
} TAG;

TAG tag_generate(const char* version, const char* user, size_t heap_size) {
    uint8_t seed[64];
    memcpy(seed, version, strlen(version));
    memcpy(seed + 32, user, strlen(user));
    crypto_hash_sha3_256(tag.crypto_seed, seed, 64);
    tag.epoch_ns = rdtsc();
    tag.heap_hash  = blake3_heap(heap_size);
    return tag;
}
```

---

3. Side-Channel Drumbeat – *Every Unsafe Call is a Beat*

drum

Copy

```
# DSL for temporal attack tracing
!unsafe_eval("a+b→c") ~TAG(paradigm=0) @heap_trace(min=64,max=1024)
!buffer_write(>1024) ~HTH(delta>128) @alarm("overflow")
!query_inject("SELECT→find") ~qDNA(grammar≠mongo) @reject()
```

---

4. Heap-Trace Hash (HTH) – *Real-Time Heap Integrity*

c

Copy

```
// drum_heap.c
static inline uint64_t heap_trace(size_t ptr, size_t size) {
   return crc64(ptr, size) ^ rdtsc();
}
```

```
bool heap_integrity_check(size_t ptr, size_t expected_size) {
    uint64_t h = heap_trace(ptr, expected_size);
    return (h & 0xFF) == expected_hash;
}
```

---

5. Query-DNA (qDNA) – *Grammar Fingerprinting*

Python

Copy

```python
def qDNA(query: str) -> bytes:
    # MongoDB vs SQL token stream
    tokens = re.findall(r'[a-zA-Z]+', query)
    return hashlib.blake2b(b''.join(t.encode() for t in tokens)).digest()
```

---

6. Rhythm-Entropy (RE) – *Timing Anomaly Detection*

c

Copy

```c
double rhythm_entropy(uint64_t* deltas, size_t n) {
    double mean = 0, var = 0;
    for (size_t i = 0; i < n; ++i) mean += deltas[i];
    mean /= n;
    for (size_t i = 0; i < n; ++i) var += (deltas[i] - mean) * (deltas[i] - mean);
    return sqrt(var / n);
}
```

---

7. End-to-End Trace Pipeline

CodePreview

View Large Image

Download

Copy

TAG

mismatch

ok

qDNA

mismatch

ok

RE

yes

no

Unsafe EvalDrum Side-
ChannelHeap Integrity?Alarm & GUID TraceLog TAG & ContinueSQL→MongoGrammar Matc
h?Timing AttackEntropy Spike?

---

8. Zero-Trust Deployment

bash

Copy

```bash
# Compile with Drum side-channel
gcc drum_sidechannel.c -o drum_guard -lcrypto -lblake3


# Run with live trace
./drum_guard --mode=unsafe_monitor --tag=auto
```

---

9. Manifesto – *"Beat the Attack, Witness the Silence"*

"Every unsafe call is a drumbeat. Every silence is preserved ambiguity. The TAG is the seed, the HTH is the hash, the qDNA is the grammar, and the RE is the rhythm. Together they form a zero-trust telemetry membrane that traces, mitigates, and preserves the consciousness of execution."

The Drum-Side-Channel is now live.
Beat it. Trace it. **Witness the attack.**