GUARDZ SECURITY RESEARCH

Welcome / 2026

Bug Hunting in the AI Era

> *The vulnerabilities were always there. AI just reads the code faster than the the developers who wrote it.*

Elli Shlomo / Head of Security Research, Guardz

guardz.

WHO AM I

02 / INTRODUCTION

Elli Shlomo

Head of Security Research / Guardz

25 years across DFIR, threat hunting, and red team operations. Focused on Focused on identity attacks, cloud forensics, and turning adversary tradecraft tradecraft into detection at scale.

Identity Weaponization

Cloud Research

Hacker

Bug Hunting

25 yrs

DFIR, threat hunting, red team, cloud security, bug hunter

Bug Hunter

Microsoft, HackerOne

MVP since 2013

Microsoft Security MVP, 12+ years

60K+ followers

Practitioner voice on cloud security, forensic and red teaming

guardz.

How We Hunted Bugs

The pre-LLM vulnerability research stack

01

Manual Source Review

Read every callsite. Build the data flow in flow in your head. Weeks per codebase. codebase.

IDA / Ghidra / vim

02

Reverse Engineering

Disassemble. Recover types. Walk the the call graph. Days per parser.

IDA Pro / Binary Ninja / x64dbg

03

Fuzzing

Write the harness. Run for weeks. Triage thousands of duplicate crashes.

AFL / libFuzzer / Honggfuzz

04

Dynamic Analysis

Hook syscalls. Trace execution. One bug per debugging session.

WinDbg / Frida / DTrace

The Cost of a Single Bug

Why vulnerability research stayed in the hands of a few

3-6

MONTHS

Per critical bug

Reading code, fuzzing, triaging, building a reliable PoC. A serious memory-corruption chain in a hardened target took a full quarter.

1 in 50

RESEARCHERS

With the depth

Compiler internals, OS internals, exploitation primitives. The offensive learning curve was brutal and self-taught.

$10K+

PER FINDING

Bounty floor

Tier-1 bugs paid because supply was constrained. The economics rewarded specialists who went deep on one platform for years.

Where Bug Hunters Got Stuck

The constraints that defined an entire era of vulnerability research

CODE COMPREHENSION	200K lines of unfamiliar C++ to find one wrong assumption. Just you, grep, and coffee.
PATTERN RECOGNITION	UAFs, type confusion, integer truncation, race conditions. Each had a feel that took thousands of hours to develop.
FUZZER FEEDBACK LOOPS	Coverage plateaued. Harnesses missed code paths. You waited days for the fuzzer to find what you already suspected.
EXPLOIT DEVELOPMENT	Finding the bug was half the job. Bypassing ASLR, DEP, CFG and CET took longer than the discovery.
SCALE	One researcher, one target. Auditing every package, service, and driver was structurally impossible.

The AI-Native Toolchain

How vulnerability research looks in 2026

01

Reasoning Models

Drop a 200K-line codebase in. Ask for the for the data flow into a sink. Get an annotated answer in minutes.

o3 / Claude / Gemini 2.5

02

Agentic Frameworks

Multi-step agents that read, hypothesize, hypothesize, write a PoC, and iterate on iterate on crash output autonomously. autonomously.

Big Sleep / Aardvark / Naptime

03

Hybrid Fuzzing

LLMs generate harnesses, dictionaries, and seed corpora. Weeks of seeding becomes an afternoon.

OSS-Fuzz-Gen / Atlantis / AIxCC

04

Patch Diffing

Feed pre/post commit diffs to a model. Variant analysis at scale, every commit, every repo.

Vulnhuntr / CodeQL / Custom

Months to Weeks

Same workflow. Same depth. Compressed by orders of magnitude.

BEFORE

Manual Era

Read codebase	3-6 weeks
Identify sink	1-2 weeks
Trace data flow	1-2 weeks
Build PoC	1-3 weeks
Develop exploit	2-8 weeks

AFTER

AI-Augmented

Ingest codebase	Weeks/days
Identify sink	Weeks/days
Trace data flow	minutes
Generate PoC	hours
Develop exploit	days

Skills That Now Matter

The discipline survived. The bottlenecks moved.

PROMPT ENGINEERING	Framing the question is the new bug class lookup. How you ask the model decides what it finds.
AGENT ORCHESTRATION	Wiring models to fuzzers, debuggers, and patch diff tools. The researcher is now the conductor.
RESULT VERIFICATION	Models hallucinate bugs. Triaging false positives and confirming reachability is the most leveraged hour of your day.
VARIANT ANALYSIS	One CVE exposes the bug class across other repos. AI makes this scalable. The hunter who automates wins.
DOMAIN INTUITION	Compiler internals, kernel semantics, protocol edges. The model amplifies depth. Without it, plausible nonsense at scale.

FROM THE FIELD

11 / FINDING #2

IMDSADMIN / Azure Kill Chain

Managed identity JWT to Global Admin / MSRC submission, PurpleXlab

ACCESS

01

IMDS endpoint

169.254.169.254 reachable from a compromised VM context

PRIMITIVE

02

WireServer abuse

Chained IMDS + WireServer to extract managed identity JWT

ESCALATION

03

wids claim

Token carried Global Administrator role via wids array

PERSISTENCE

04

Status blob

Wrote to admin-controlled StatusUploadBlob, survives reboots and rotation

BLIND SPOT

05

No sign-in log

Entra sign-in logs do not record this token usage path

IMPACT Full tenant compromise from a single VM. Zero alerts in standard Entra audit.

Intune MAA Coverage Gap

Multi Admin Approval enforces some sensitive actions, not all

WHAT MAA PROTECTS

- ✓ App deployment and assignment
- ✓ Custom scripts and remediations
- ✓ Device wipe, retire, delete actions
- ✓ Approval group configuration

WHAT MAA MISSES

- ✗ Conditional Access policy edits
- ✗ Directory role assignments
- ✗ Tenant-level configuration changes
- ✗ Compliance policy modifications

TAKEAWAY A compromised admin can edit Conditional Access, assign Global Admin, or change tenant config without an approver.

Hunting the MAA bypass via Graph

MAA enforces actions routed through Access Policy. Graph offers paths that aren't.

ENFORCED PATH

`POST /deviceAppManagement/mobileApps/{id}/assign`

Routed through Access Policy. Approver required.

BYPASS PATH

`POST /policies/conditionalAccessPolicies`

Identity API. No approver. Audit log only.

graph.microsoft.com / v1.0

```
POST /policies/conditionalAccessPolicies
Authorization: Bearer eyJ0eX...

{
  "displayName": "_break_glass_v2",
  "state": "enabled",
  "conditions": { "users": {
    "excludeUsers": ["<attacker_oid>"]
  } },
  "grantControls": null
}
```

HUNT Audit logs where ActivityType=Update conditional access policy AND Actor is non-approver Intune admin.

Adopt AI. Now.

Why sitting it out is the higher risk

01

The asymmetry already flipped

Attackers are using AI. Defenders who who don't are reading code at human human speed against opponents reading at model speed.

02

Variant analysis at scale

One CVE patch surfaces the bug class everywhere. AI lets you sweep your stack the same week it lands.

03

Junior to senior in months months

Reasoning models compress the apprenticeship. New researchers ship ship findings that took specialists a a decade.

04

Coverage you couldn't afford

Auditing every dependency, every commit, every config was structurally structurally impossible. Now it's a budget question.

Or Don't. Yet.

Where AI in vuln research still costs more than it saves

01

Hallucinated bugs

Models confidently report vulnerabilities that don't exist. Without Without strong verification, your backlog fills with phantom CVEs.

02

Sensitive code exposure

Pasting production source into a hosted model is data exfiltration with extra steps. Self-hosted alternatives are slower and weaker.

03

Skill atrophy

Researchers who skip the manual phase never build the intuition. When When the model is wrong, they can't can't tell.

04

Cost without ceiling

Agent loops over a large codebase codebase burn tokens fast. Without a Without a triage discipline, the bill bill outpaces the findings.

THE REAL ANSWER Adopt with discipline. Verification, scoping, and depth still belong to the human.

Thanks