

Lighting the Way in the Dark

Binary-Only Race Detection in Go with Zorya-Volos

ZORYA

KMSEC
WORLD CLASS CYBER SECURITY
kmsecurity.co.za

JOBURG BSIDES

Agenda

1. Whoami
2. The hard problems
3. How Zorya stands out
4. How Zorya Works
5. CVEs picked up Zorya MCP
6. Finding Races with Volos



BSIDES JOBURG

KMSEC
WORLD CLASS CYBER SECURITY
kmsecurity.co.za

Whoami

- **MSc Computer Science**
  - "Automated Detection of Race Condition Vulnerabilities in Binary Programs using Symbolic Execution"
- **Offensive Security (~15 years)**
- **Research (~20 years)**
  - Chrome XSS Auditor bypass
  - Chrome pre-link dns leakage bug
  - NoScript bypass
- **Bootloader Security + Binary Analysis**
  - **Zorya , XEGMAP, (more to come!)**
  - 1 bug fixed in coreboot woohoo!
  - 100s hours of consulting on bootloaders sec and embedded stuff
- **Author**
  - **Android Security Cookbook**
  - Soon to be published in "**Journal of Computer Virology and Hacking Techniques**"
- **White Collar Boxer**



Keith Makan

Founder + Principal Researcher (KMSEC)

KMSEC
WORLD CLASS CYBER SECURITY
kmsecurity.co.za

Why Binary Analysis?

01	Compiler Derived Vulns	• Do not manifest at source code level • Can cause code exec vulns • Have affected widely used software
02	Hardware derived bugs	• Difficult to patch around • Affects cloud environments / mobile apps • Often require microcode / firmware updates which are risky!
03	Lack of Samples	• Binaries are rarely distributed with input samples • Inputs maybe complex i.e. mp3s, images etc
04	Obfuscation / Stripping	• Binaries are often distributed without symbols / debug information • Companies may apply IP protective obfuscation • Saftey analysis of the final product becomes harder

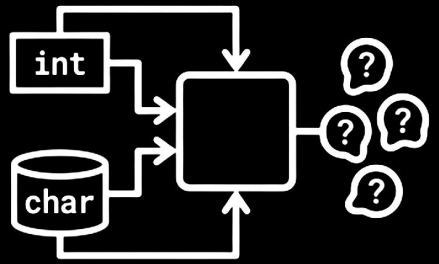
KMSEC
WORLD CLASS CYBER SECURITY
kmsecurity.co.za

Why is binary analysis hard?



Variable Liveness / Data Flow

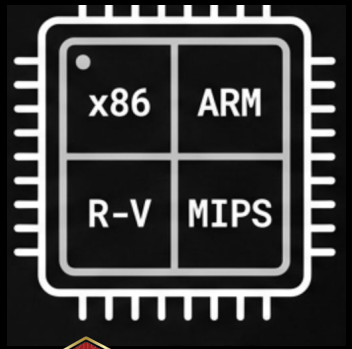
Loss of Typing Information





Loss of Semantic / Symbol Information

Multiple ISAs / Hardware Complexities



BSIDES JOBURG

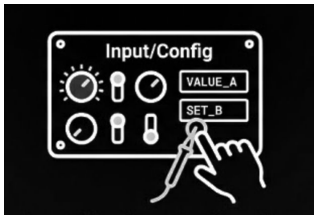
KMSEC
WORLD CLASS CYBER SECURITY
kmsecurity.co.za

Why are detecting race conditions hard?



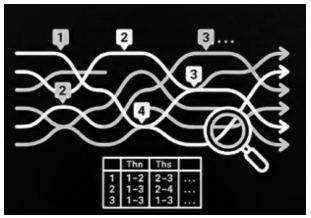
Shared Resource Analysis

Channels, sockets files, shared memory



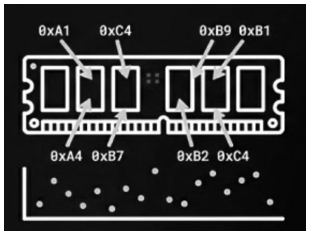
Input / Config Sampling

Finding samples that trigger thread activity



Thread Interleaving Analysis

Threads spawn in different schedulings, reorder, die and respawn



Memory Interaction Sampling

Sample ahead of time?
during execution?
analyze after execution?

BSIDES JOBURG
KMSEC
WORLD CLASS CYBER SECURITY
kmsecurity.co.za

Detecting race conditions is hard : example

```c
void* thread1_func(void* param) {
    X = 1;
    // The hardware/compiler may move this load above the store
    pthread_mutex_lock(&lock);
    read_Y_val = Y;
    pthread_mutex_unlock(&lock);
    return NULL;
}

void* thread2_func(void* param) {
    Y = 1;
    // The hardware/compiler may move this load above the store
    pthread_mutex_lock(&lock);
    read_X_val = X;
    pthread_mutex_unlock(&lock);
    return NULL;
}
```

```c
int main() {
    long iterations = 0;
    long total_anomalies = 0;
    const long max_iterations = 1000000; // Total run length
    const int sample_rate = 1000;       // Output every 1000 iterations

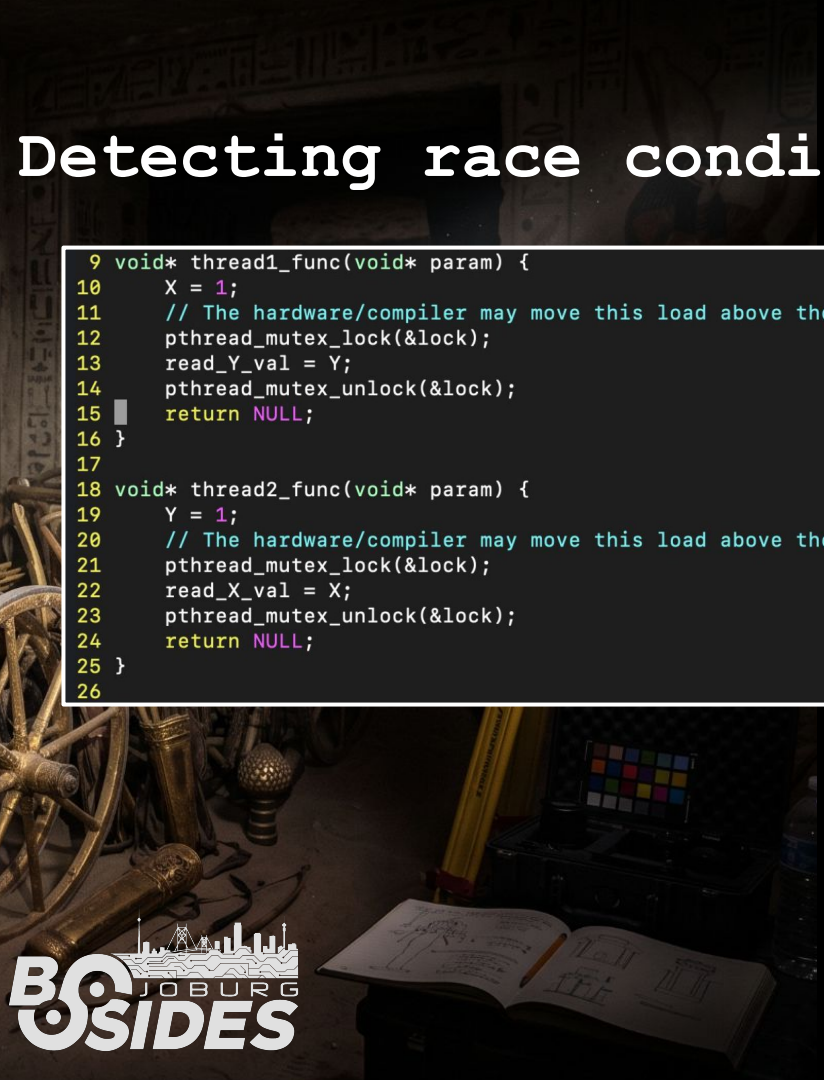
    // Header for the data file
    printf("#Iteration\tAnomalies\n");

    while (iterations < max_iterations) {
        iterations++;
        X = 0; Y = 0;

        pthread_t t1, t2;
        pthread_create(&t1, NULL, thread1_func, NULL);
        pthread_create(&t2, NULL, thread2_func, NULL);

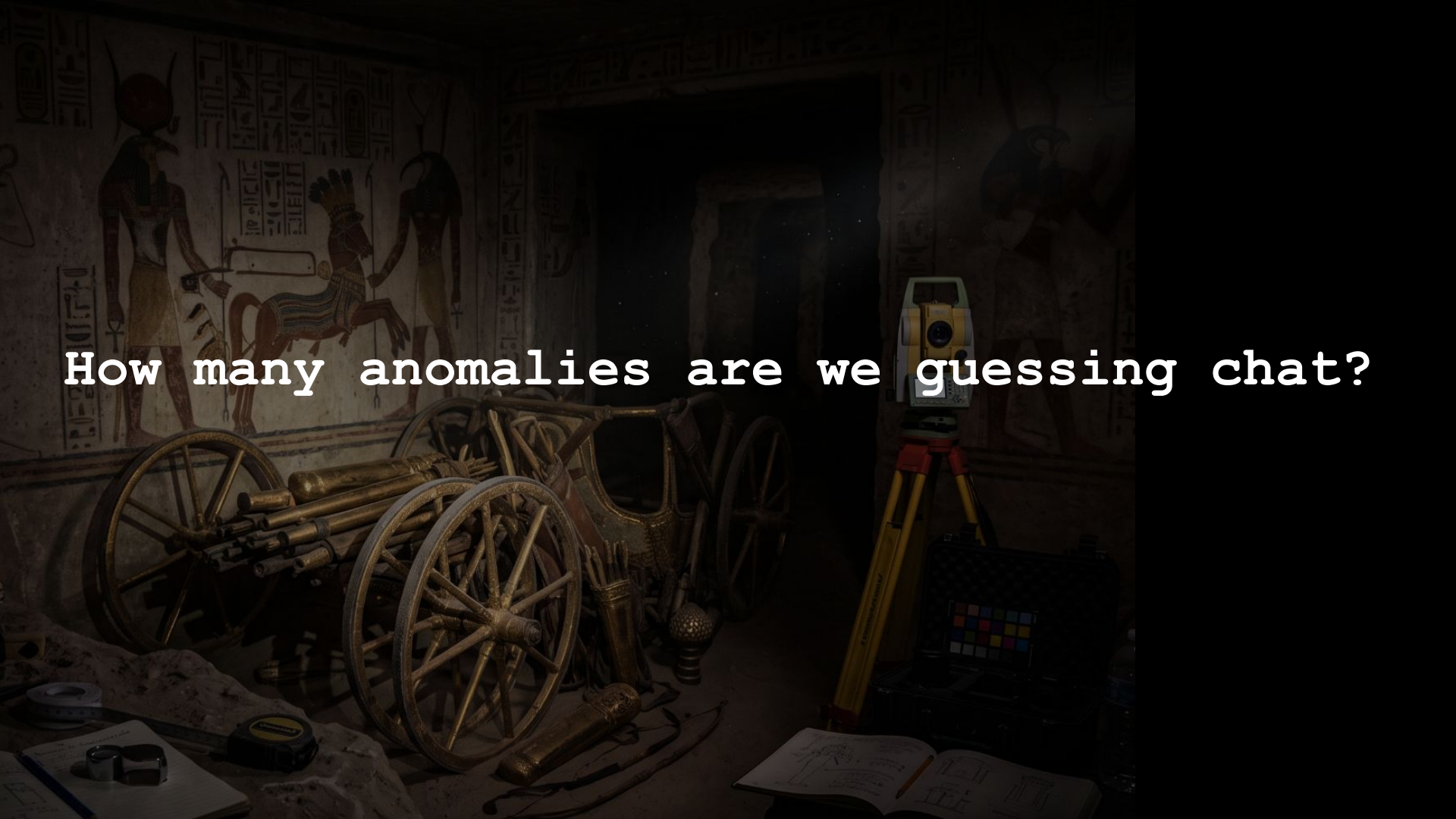
        pthread_join(t1, NULL);
        pthread_join(t2, NULL);

        // Check for the weak memory signature
        if (read_X_val == 0 && read_Y_val == 0) {
            total_anomalies++;
        }
```

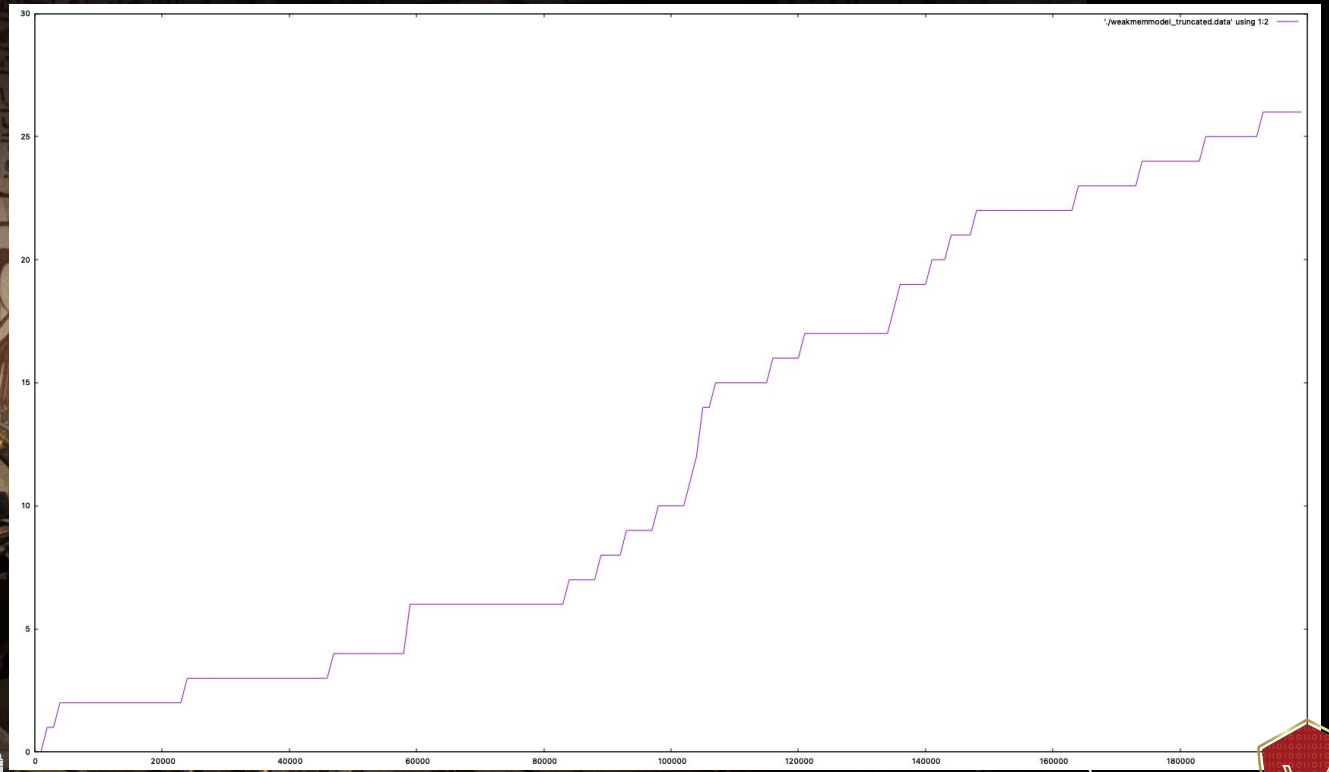


BSIDES JOBURG

KMSEC
WORLD CLASS CYBER SECURITY
kmsecurity.co.za


How many anomalies are we guessing chat?

How many Anomalies?



25

Anomalies

Iterations

180 000

BSIDES JOBURG

KMSEC
WORLD CLASS CYBER SECURITY
kmsecurity.co.za

Symbolic Execution (quick intro)

1. Convert conditionals to constraints
   a. (Optional) perform constraint ops
   b. (Optional) perform solver load balancing
2. Map to symbolic states
   a. (Optional) state merging
   b. (Optional) state pruning
3. Interrogate / Explore
   a. Find states of worth, prove we cant find it
4. Easy right?

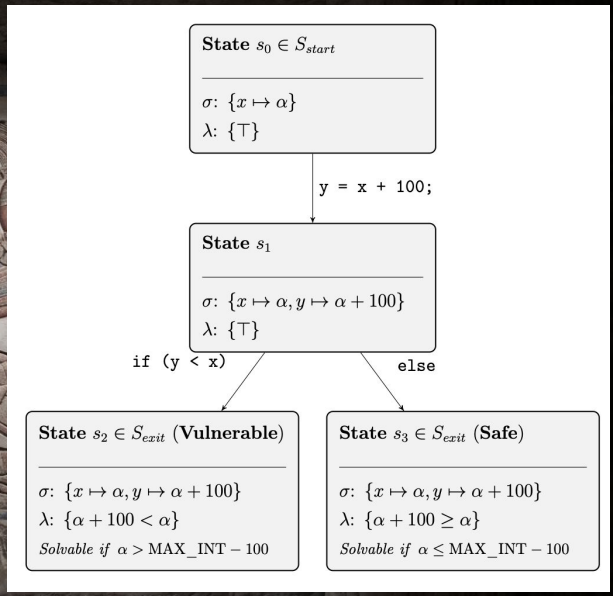
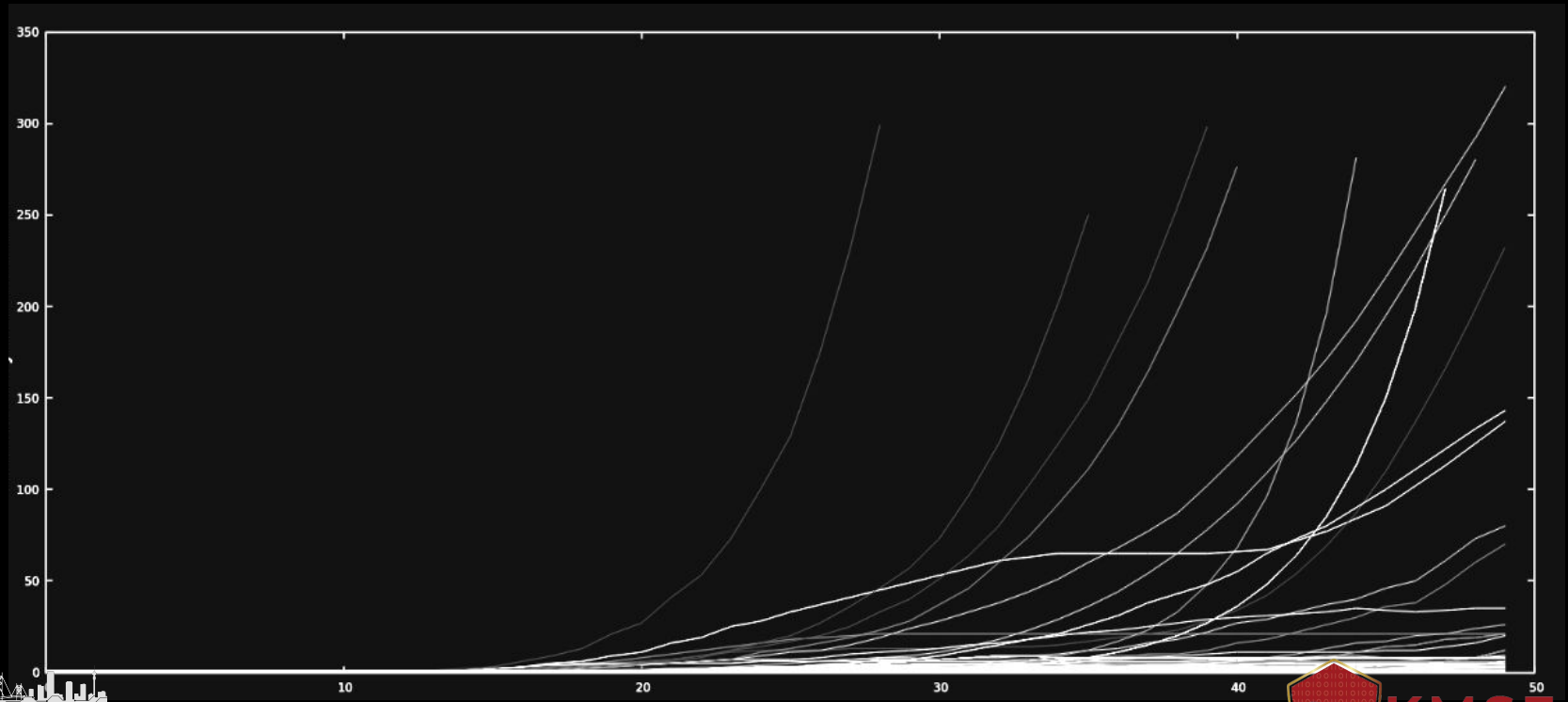


Fig: symbolic execution map

BSides Joburg
KMSEC
WORLD CLASS CYBER SECURITY
kmsecurity.co.za

Why is symbolic execution hard to scale?



BSIDES JOBURG
KMSEC
WORLD CLASS CYBER SECURITY
kmsecurity.co.za

Introducing Zorya



Go Binary

Models the unique idiosyncrasies of Go's runtime model into analysis

Concolic + Literal

Blends both literal and concolic execution

Finds Real Bugs

Demonstrated to easily uncover many real CVE ID'd bugs, which affect widely used software critical to secops

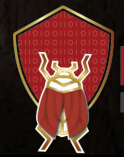
Rust Developed

Developed in Rust, to ensure we weed out any compiler or common bugs

Plugin Arch

Allows extension via a plugin architecture





KMSEC
WORLD CLASS CYBER SECURITY
kmsecurity.co.za

How Zorya stands out

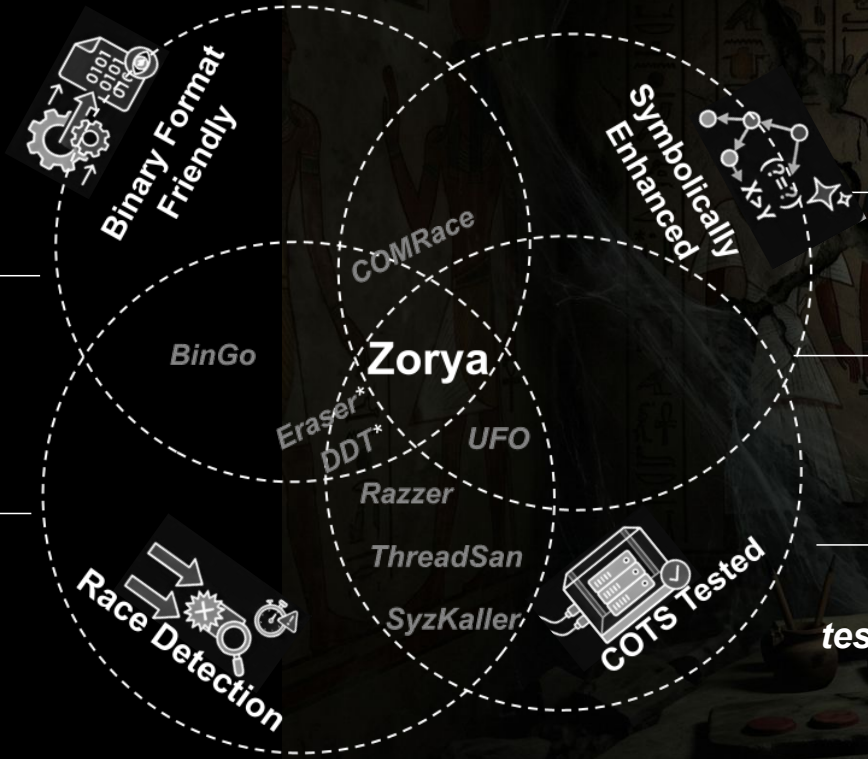
analysis without source code required!

automated test case generation

reproduces input samples

find real reproducible bugs

tested on real world binaries



JOBURG BSIDES

KMSEC
WORLD CLASS CYBER SECURITY
kmsecurity.co.za

How Zorya stands out

Study	Target	Year	Analysis	RCDS	BFF	LSIP	COTS	Symbolic
Zorya	**Go**	**2026**	**LSA**/HBA	OTF	**Yes**	**Yes**	**Yes**	**Yes**
BinTSAN	**C/C++**	2024	LSA/HBA	OTF	**Yes**	Yes	**Yes**	No
TaskGrind	**C/C++**	2024	HBA	PM	**Yes**	No	**Yes**	No
HardRace	**C/C++**	2024	LSA/HBA	PM	**Yes***	**Yes**	**Yes**	**Yes**
COMRace	**C/C++**	2022	LSA	CT	**Yes**	No	**Yes**	**Yes**
BiRD	**C/C++**	2022	LSA/HBA	PM	**Yes**	No	No	No
BinGo	**Go**	2022	LSA	PM	**Yes**	No	**Yes**	No
ConVul	**C/C++**	2019	HBA	OTF	**Yes**	**Yes**	**Yes**	**Yes**

Zorya + Volos Process

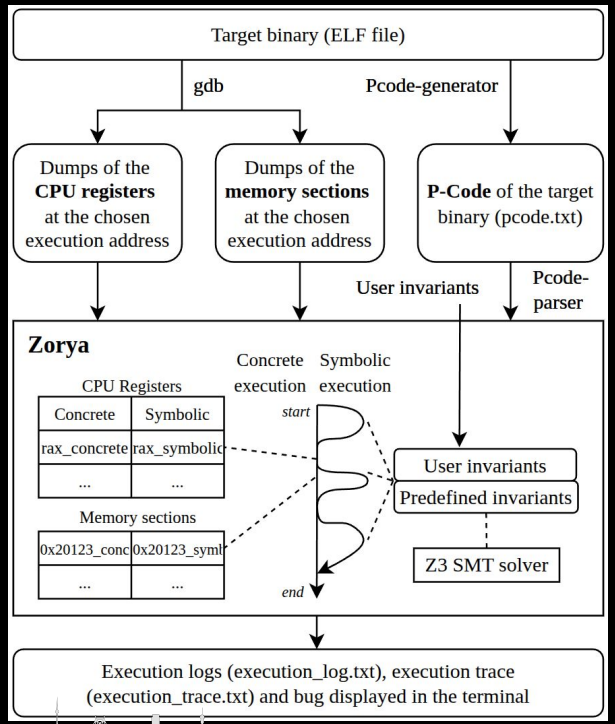


Fig: Zorya Process

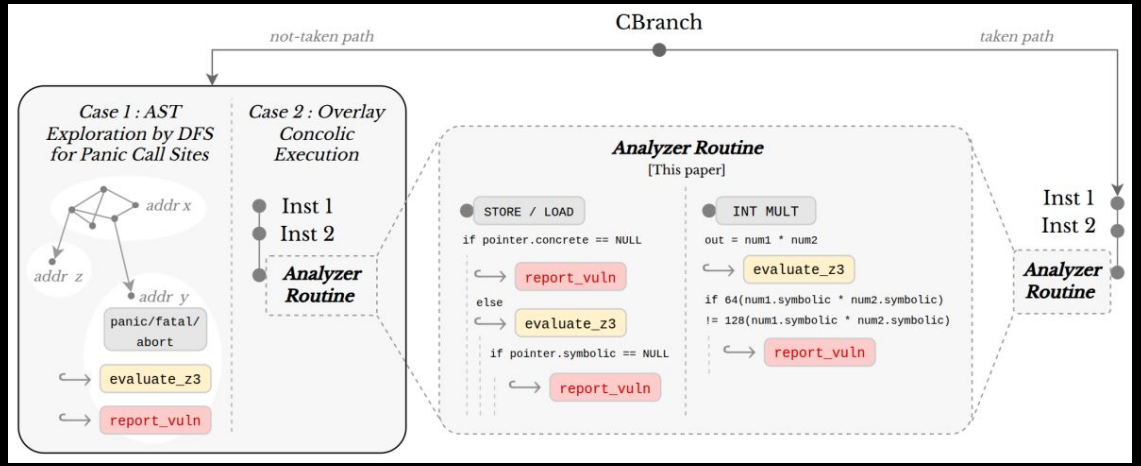


Fig: overlay mode

BSIDES JOBURG
KMSEC WORLD CLASS CYBER SECURITY
kmsecurity.co.za

Zorya + Volos Process



Real bugs found with Zorya

Project	Bug type	Status	Report
Nil-pointer dereference			
runatlantis/atlantis	Nil-ptr deref	Fixed	atlantis #6492
crossplane/provider-http	Nil-ptr deref	Open	provider-http #178
kedacore/keda	Nil-ptr deref	Fixed	keda #7798
Integer overflow / underflow			
erpc/erpc	Integer overflow	Open	erpc #869
trufnetwork/kwil-db	Integer overflow	Open	kwil-db #1701
cometbft/cometbft	Integer overflow	Fixed	cometbft #5846
gnolang/gno	Integer overflow	Open	gno #5639
XinFinOrg/XDPoSChain	Integer underflow	Open	XDPoSChain #2362
kedacore/keda	Float-to-int overflow	Ongoing	keda #7796
multiversx/mx-chain-go	Float-to-int64 overflow	Reported	private report
Index out-of-bounds / slice bounds			
mandiant/gopacket	OOB slice	Fixed	gopacket #23
mandiant/gopacket	OOB slice	Fixed	gopacket #25
0xPolygon/bor	OOB slice	Fixed	bor #2221
zeromicro/go-zero	OOB slice	Fixed	go-zero #5607
zeromicro/go-zero	Slice bounds	Fixed	go-zero #5614
seaweedfs/seaweedfs	Slice bounds	Fixed	seaweedfs #9712
seaweedfs/seaweedfs	Index out of range	Fixed	seaweedfs #9713
multiformats/go-multiaddr	Slice bounds	Open	go-multiaddr #288
multiformats/go-multiaddr	Index out of range	Ongoing	go-multiaddr #289
nats-io/nats.go	Slice bounds	Ongoing	nats.go #2100
Security advisory			
pion/dtls	ECDHE_PSK panic	Published	CVE-2026-54908
pion/stun	XOR-MAPPED-ADDR panic	Published	CVE-2026-54909
LeJamon/go-xrpl	MPT int64 wrap	Published	GHSA-j5cw-qr8c-mmmv7
LeJamon/go-xrpl	MPT scaling overflow	Ongoing	GHSA pending
gochain/gochain	Disclosed advisory	Published	GHSA-rmvq-87f6-4pfc

Table: 28 findings across 148 targets (binary + function entry point), May–June 2026 with the Zorya MCP Server. Average cost: 3 € per target identification.

CVE-2026-54908

CVE-2026-54909

KMSEC
WORLD CLASS CYBER SECURITY
kmsecurity.co.za

CVE-2026-54908

dtls / Security / Advisories / GHSA-wg4g-wm44-ch5j

Denial of service via panic while parsing a crafted ECDHE_PSK ServerKeyExchange message

Moderate JoTurk published GHSA-wg4g-wm44-ch5j on Jun 10

Package	Affected versions	Patched versions
github.com/pion/dtls/v3 (Go)	<=v3.1.2	>=v3.1.4

Description

Impact

Remote denial of service via panic while parsing a crafted ECDHE_PSK ServerKeyExchange message.

Patches

Upgrade to v3.1.2 or later. This version includes this patch #839 which fixes the issue.

Workarounds

No workaround. Please upgrade to v3.1.4 or a newer version.

Severity

Moderate

CVE ID

CVE-2026-54908

Weaknesses

No CWEs

Credits

Sean-Der — Remediation developer

kajaaz — Finder

BSIDES

KMSEC
WORLD CLASS CYBER SECURITY
kmsecurity.co.za

CVE-2026-54908 How does the null pointer deref work?

pkg/protocol/handshake/message_server_key_exchange.go

```
@@ -96,6 +96,10 @@ func (m *MessageServerKeyExchange) Unmarshal(data []byte
96  96          return errLengthMismatch
97  97      }
98  98
    99  +   if len(data) == 0 {
   100  +       return errBufferTooSmall
   101  +   }
   102  +
99  103 +   if _, ok := elliptic.CurveTypes()[elliptic.CurveType(data[0])]; ok {
100 104         m.EllipticCurveType = elliptic.CurveType(data[0])
    105     } else {
```

Patch adds a defensive check against the length of read in data

BSIDES JOBURG
KMSEC WORLD CLASS CYBER SECURITY
kmsecurity.co.za

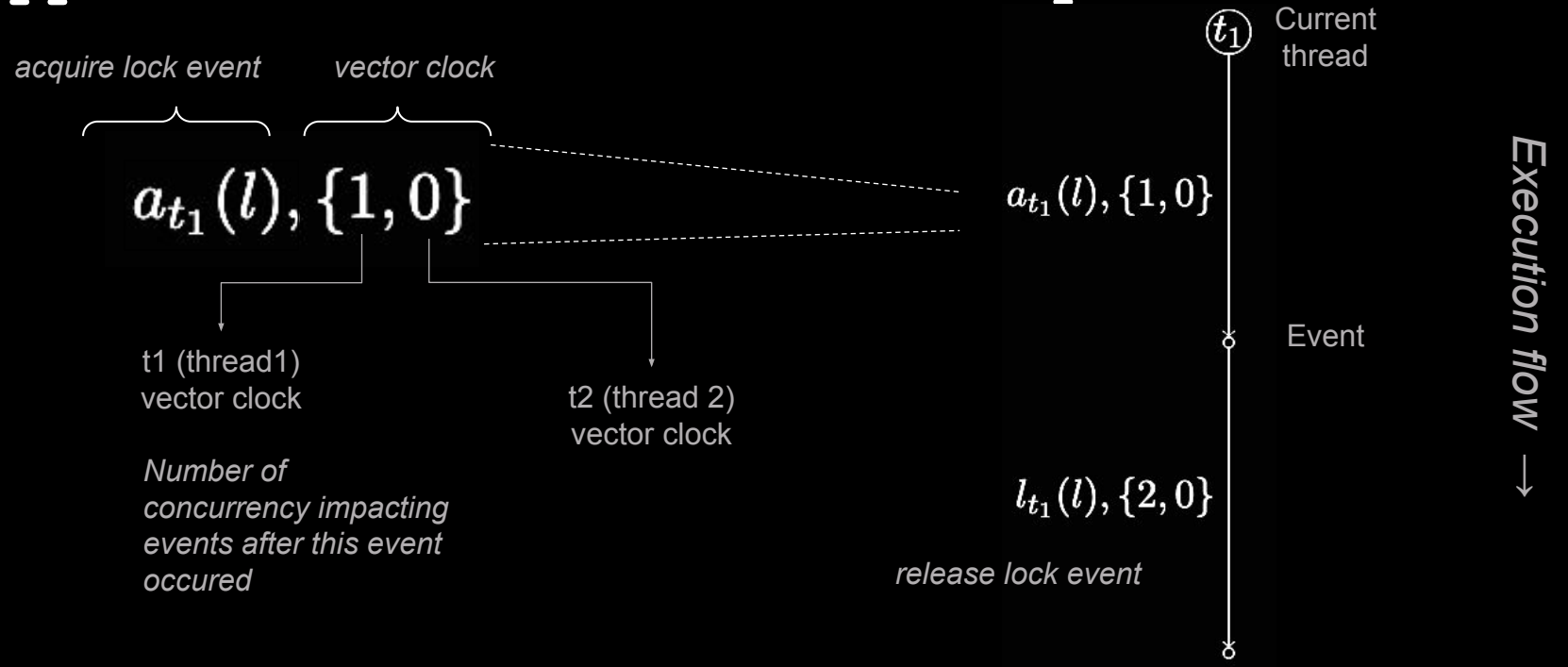
How does Zorya-Volos detect Race conditions?



BSIDES JOBURG

KMSEC
WORLD CLASS CYBER SECURITY
kmsecurity.co.za

Happens-Before : Release / Acquire



BSIDES JOBURG
KMSEC
WORLD CLASS CYBER SECURITY
kmsecurity.co.za

Happens-Before : Release / Acquire

Code

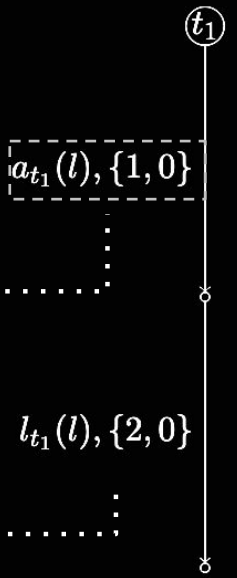
```
go func() {
  mutex.Lock()
  ...

  mutex.Unlock()
 }()
```

Disassembly

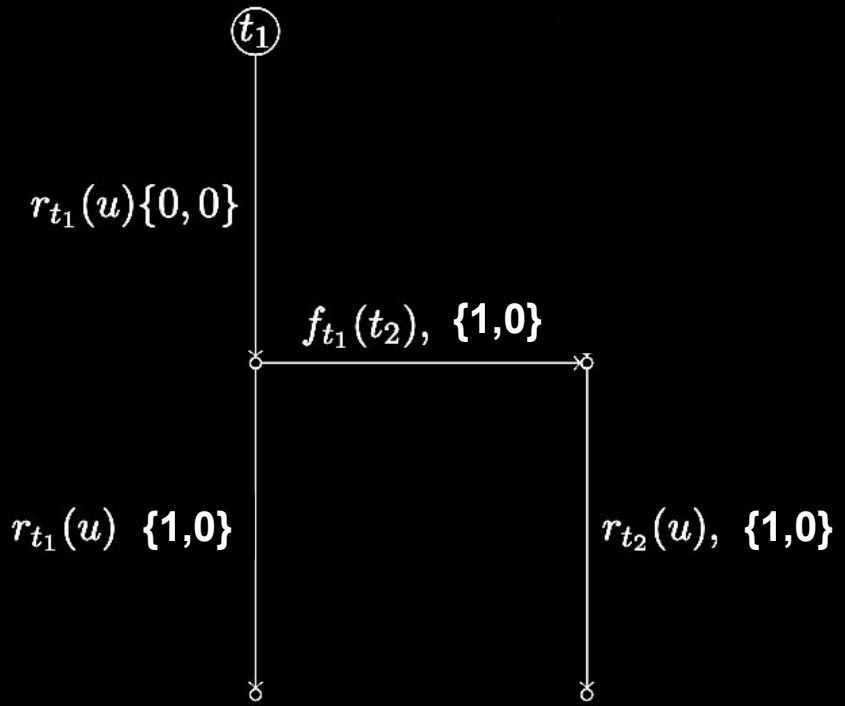
```
0108d5b1  lea     rax, [rel _main.mutex]
0108d5b8  call    _sync.(*Mutex).lockSlow
0108d5bd  lea     rcx, [rel _main.mutex]
```

```
0108d5d4  lea     rax, [rel _main.mutex]
0108d5db  nop     dword [rax+rax], eax
0108d5e0  call    _sync.(*Mutex).unlockSlow
```



BSIDES JOBURG
KMSEC
WORLD CLASS CYBER SECURITY
kmsecurity.co.za

Happens-Before Fork New Routine



BSIDES JOBURG
KMSEC
WORLD CLASS CYBER SECURITY
kmsecurity.co.za

Binary Lock-Set Analysis

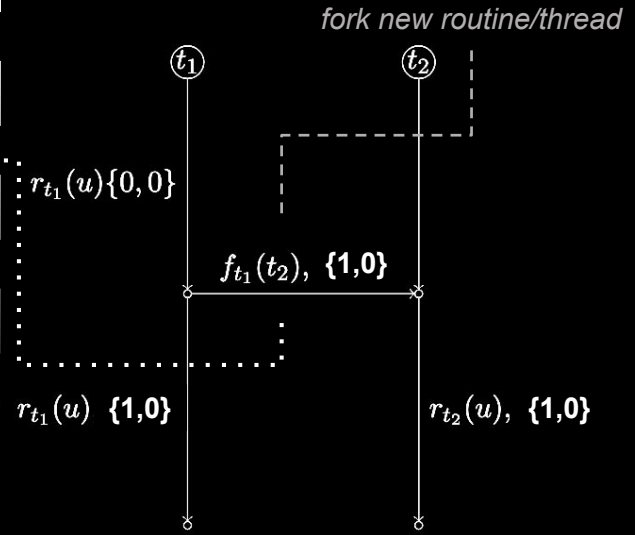
Code

```
func main() {
    // Main Thread
...
    // Child Routine 1
    go func() {
...
    }()
...
    // Child Routine 2
    go func() {
```

Disassembly

```
0108d5e5  lea     rax, [rel data_10ade10]
0108d5ec  call    _runtime.newproc
0108d5f1  lea     rax, [rel data_10ade18]
0108d5f8  call    _runtime.newproc
```

Fork/Clone event



BSIDES JOBURG
KMSEC WORLD CLASS CYBER SECURITY
kmsecurity.co.za

Running Volos

```
zorya <path> --lang <go|c|c++> [--compiler <tinygo|gc>] \
  --mode <start|main|function|advanced> <addr> \
  [--thread-scheduling <all-threads|main-only>] \
  [--arg "<arg1> <arg2>"] \
  [--negate-path-exploration|--no-negate-path-exploration] \
  [--force-pty] \
  [--symbolic-registers "REG1 REG2|all"] \
  [--symbolic-memory "0xADDR:SIZE ..."] \
   [--no-symbolic-registers] [--no-symbolic-memory]
```

BSIDES JOBURG

KMSEC
WORLD CLASS CYBER SECURITY
kmsecurity.co.za

Running Volos

zorya <path> `--lang <go|c|c++> [--compiler <tinygo|gc>]` **--mode <start|main|function|advanced> <addr>** `[--thread-scheduling <all-threads|main-only>] ...`

FLAG:

- --lang Specifies the language used in the source code (go/c/c++)
- --compiler When Go was chosen as 'lang', specifies the used compiler (tinygo or gc)
- --mode Specifies the strategy mode to determine the starting address for binary analysis.

  Options include:
  - start → Use the binary's entry point
  - main → Analyze the main function (main.main preferred in Go binaries)
  - function → Specify a function address manually
  - advanced → Define an arbitrary execution address with fine-grained symbolic variable selection (registers, memory)

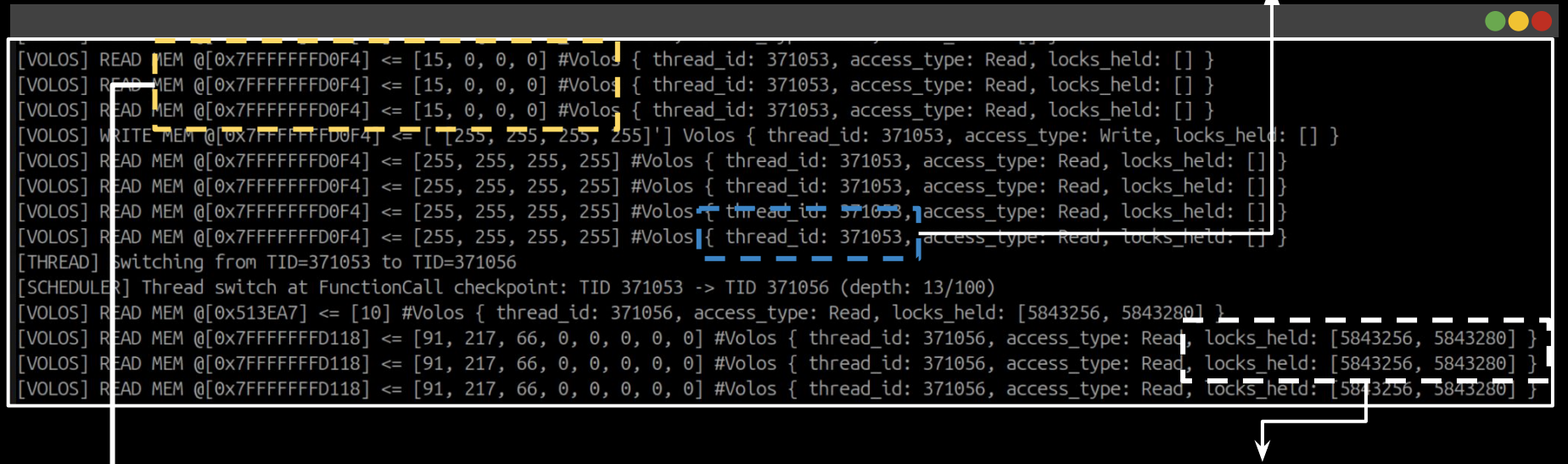
--thread-scheduling Thread scheduling strategy for Go GC binaries:

- all-threads → Load + schedule all dumped OS threads
- main-only → Execute only the main thread (simpler/more deterministic)

BSIDES JOBURG
KMSEC WORLD CLASS CYBER SECURITY
kmsecurity.co.za

Finding a race with Volos

thread id contextualization



view of memory read/writes addresses + values

locks/futexes held

BOSIDES JOBURG

KMSEC
WORLD CLASS CYBER SECURITY
kmsecurity.co.za

Finding a race with Volos

```
[VOLOS DETECTOR] DATA RACE FOUND AT 0x1f948
------------------------------------------------------------------
Access 1:
  Goroutine ID: 371053
  Op Type:      Write
  Locks Held:   [5843280]
--- VS ---
Access 2:
  Goroutine ID: 371056
  Op Type:      Write
  Locks Held:   NONE (UNPROTECTED)
------------------------------------------------------------------
REASON: One or more threads accessed this memory without a shared lock.
==================================================================

[VOLOS] READ MEM @[0x572E50] <= [32, 59, 87, 0, 0, 0, 0, 0] #Volos { thread_id: 371056, access_type: Read, locks_held: [] }
[VOLOS] READ MEM @[0x573C2F] <= [2] #Volos { thread_id: 371056, access_type: Read, locks_held: [] }
[VOLOS] WRITE MEM @[0x573C2F] <= ['[1]'] Volos { thread_id: 371056, access_type: Write, locks_held: [] }
[VOLOS] READ MEM @[0x573C2F] <= [1] #Volos { thread_id: 371056, access_type: Read, locks_held: [] }
```

BSIDES JOBURG

KMSEC
WORLD CLASS CYBER SECURITY
kmsecurity.co.za



Thank You

KMSEC
WORLD CLASS CYBER SECURITY

kmsecurity.co.za

ZORYA

zorya.karolinagorna.net

JOBURG BSIDES