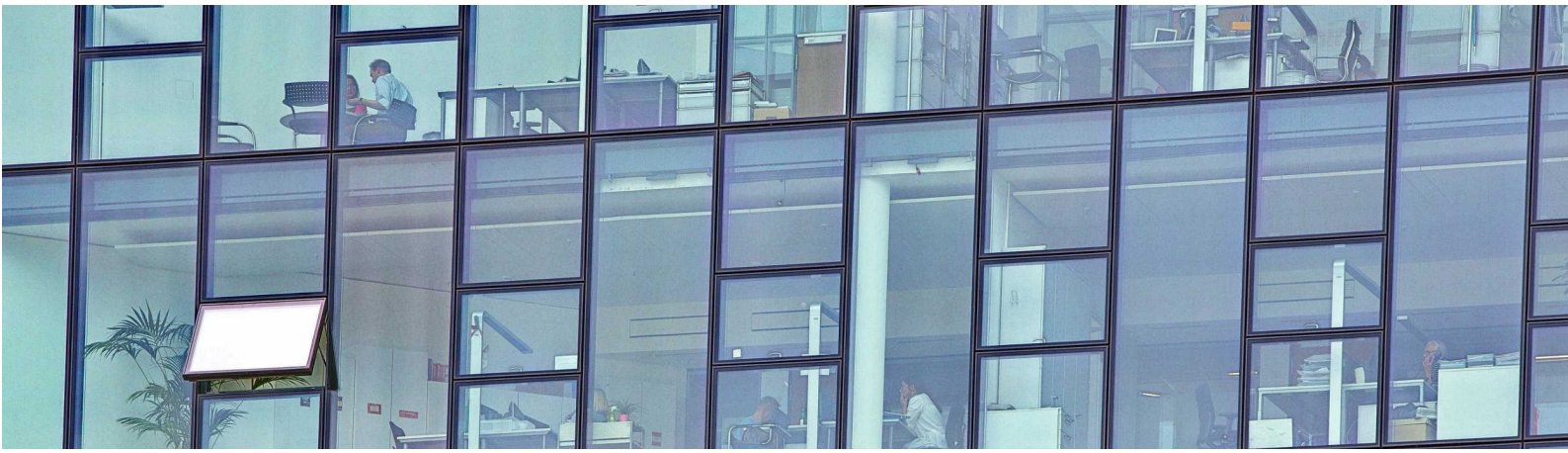


# Run SLIMalloc Everywhere

Since November 2022, the NSA recommends to use "*memory-safe*" programming languages because:

- 60-70% of all Apple vulnerabilities
- 70% of all Microsoft vulnerabilities
- 90% of all Google Android vulnerabilities

} are memory safety issues.

SLIMalloc makes C programs "*memory safe*". It detects, blocks and reports the root causes of 70-90% of all past and future vulnerabilities (in operating systems, third-party libraries, and applications).

---

## Detect, Block, Report

...while accelerating your machines.

Acting within every single process lets SLIMalloc detect memory violations – and block, correct, and /or redirect code execution when needed.

In contrast, security monitoring solutions watch programs from the outside and investigate "*post-mortem*" events, always too late.

SLIMalloc is proactive because it enforces the intended memory access and blocks the unintended ones.

Our technical report published on ResearchGate presents the background technology, examples and comparative performance tests.

Defense that blocks and documents attacks, and lets you take action – in real-time.

Attacks and errors are blocked and reported while programs run unharmed.

Know who does what, which program is vulnerable – exactly when, why and where.

Programs keep running instead of crashing for unknown (often long-delayed) reasons.

Focus on your activity instead of being forced to continuously watch and repair.

Streamline code validation, developments, deployments, and production follow-up.

Eliminating uncertainty lets your teams plan for the future, and build on a solid basis.

## “Memory safety”, why and how?

### Understanding the History of Computer Science

Linking the dots requires to having been there programming long enough (since 1979) to have seen the dots coming to life

Unexpected memory modifications can either bring garbage output and/or program crashes or, when carefully planned, undetected control flow hijacking (machines are remotely hacked by injecting and running arbitrary code).

This has greatly been facilitated by C++, the first language to introduce *by-design* (that is, imposed by the C++ language) function pointers in C structures - breaking a major computer-science taboo: the interdiction to mix code and data.

After C++, every single *“modern”* language happily jumped on the opportunity... including the “memory-safe” ones that are vulnerable to memory corruption: most of their capacity relies on libraries and OS functions written in C/C++ (code that is not checked by their memory-safe runtimes so their memory can be modified by external code without detection).

The C language allows you to do anything – even the utterly dangerous C++ class construction – but it does not impose it to you, hence, probably, today's urgent need felt by some industry players to replace C by programming languages that do not leave this choice to the programmer.

## Making C programs “memory safe”

### Take back control of your machines

Hire TWD to deploy game-changer capacity in your distributed infrastructure

Whatever will be decided by those in control of the industry, there is an obvious fix: making C “memory safe”.

Then, C and C++ programs, libraries and operating systems are no longer at risk.

As a welcome bonus, the “memory safe” languages will finally have a chance to become completely memory safe.

# TWD Industries AG

SLIMalloc

## From data center to cloud to edge

Memory issues bypass encryption layers such as the HTTPS “secure” protocol, firewalls, and intrusion detection systems. There's nowhere to hide. The only effective defense is to deploy a memory allocator which, like SLIMalloc, blocks the memory violations – even in closed-source software.

The system and applications are vulnerable by just reading network packets, data, or parsing a font or an image (a task that Windows implements in the kernel, leading to recurring exploits) when an end-user opens a web page or an email.

Even worse, successful memory exploits are undetected.

## TWD Support Options for SLIMalloc

### Consulting services

Those bringing real-time security without a single false positive or false negative probably know a thing or two about what they do

Running SLIMalloc lets you block the threat and see the scale of your exposure, but it also document the location of all the vulnerabilities in the system, libraries, and applications. This allows you to fix the software your activity relies on, whether you develop or deploy open or closed-source software.

TWD has developed interfaces to extra services that let SLIMalloc end-users feed their own repository with the vulnerabilities discovered in real-time on their infrastructure. This lets you develop pro-active and re-active measures such a structured and automated incident response and even real-time retaliation.

# Secure Allocators vs Security Layers

"LLVM ASan" aims for correctness while "GWP-Asan" is merely for probabilistic verifications.

| Technology (4) | LLVM ASan (shadow-ram)<br><br>CTI (compile-time instrumentation)<br><br>"systematic" (many cases bypassed) | Valgrind (VM)<br><br>DBI (dynamic binary instrumentation) | Dr. Memory (Google, MIT)<br><br>DBI (dynamic binary instrumentation) | TCmalloc (allocator)<br><br>"probabilistic"<br><br>GPerftools heap-checker library | SLIMalloc (allocator)<br><br>"systematic"<br><br>detect and block (or recover) errors |
|---|---|---|---|---|---|
| Slowdown vs Acceleration (2) | 2x slowdown | 20x slowdown | 10x slowdown | 25%-50% slowdown | 30% to 5x acceleration! |
| **Detects and Blocks** | **LLVM ASan (shadow-ram)** | **Valgrind (VM)** | **Dr. Memory (Google, MIT)** | **TCmalloc (allocator)** | **SLIMalloc (allocator)** |
| Heap OOB<br>out-of-bounds | most / no | yes / no | yes / no | some / no | most / yes |
| WWW<br>write-what-where | some / no | no / no | no / no | no / no | some / yes |
| UAF<br>use-after-free (dangling ptr) | yes / no | yes / no | yes / no | some / no | yes(3) / yes |
| Leaks | yes / no | yes / no | some(1) / no | yes / no | yes / yes |

(1) for Dr. Memory, non-freed memory is not a leak; only blocks missing a pointer are leaks.
(2) slowdown or acceleration, as compared to the imposed default system memory allocator.
(3) all UAFs are caught if application/library is recompiled with SLIMalloc (no APIs involved).
(4) "systematic" doesn't mean exhaustive, it means that checks are not merely "probabilistic".

Blocking memory errors on-the-fly lets programs avoid corruption, continue unharmed instead of aborting (when violations are detected) or crashing, often delayed (with ignored violations). It lets SLIMalloc document bugs and attacks (no mere core-dumps) and even retaliate in real-time.

This is invaluable in production, when input-related errors are triggered and nobody is watching. It also helps if you don't want to crash/fix/recompile a program 10,000 times to find all its bugs.

All memory-safety layers have blind areas and most cause a "*considerable performance overhead*" (NSA, November 2022) including the Intel MPX technology that the GCC compiler documented and made available for a while as an alternative to the ASan libraries.

SLIMalloc is without equivalent on the market.

# TWD Industries AG

SLIMalloc

What the NSA says...

*"The overarching software community across the private sector, academia, and the U.S. Government have begun initiatives to drive the culture of software development towards utilizing memory safe languages."*
National Security Agency (November 2022)

What the industry says...

## Microsoft

*"70% of the vulnerabilities addressed through a security update each year continue to be memory safety issues."*

## Google

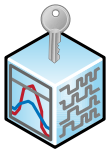
*"Google estimated that 90% of Android vulnerabilities are memory safety issues."*

## Apple

*"Apple's T2 Security Chip has an unfixable flaw allowing hackers to disable macOS security features and install malware."*

## TWD products since 1998

After the dot.com crisis, and 20+ million RA licenses deployed, TWD declined $20m from Summit Partners, the largest and oldest venture capitalist. In 2009, using organic growth, more than 280 million RA licenses have been deployed in 136 countries. Long-term projects often demand long term investments.



### SLIMalloc (2020+)

As a memory allocator (either used by one application or system-wide) SLIMalloc first offered a faster and flawless design – safe against memory allocation errors and against system errors (both were handled without corrupting memory so programs could keep going instead of crashing). In 2023, SLIMalloc increased its memory-safety coverage to the system, third-party libraries, and C programs to deliver "memory safety" to the C programming language - the root cause of 70-90% of all vulnerabilities for decades, according to the NSA.

### Global-WAN (2010+)

Global-WAN was (and still is) the only Level-2 distributed VPN relying on "post-quantum" (a decade before NIST PQE standards) and "unconditional" security (safe forever by preserving the whole key-space and therefore the algebraic plausibility of any potential key and plaintext).

### G-WAN Application Server (2009+)

G-WAN was 4x faster than Microsoft IIS 7.5 and offered servlets (in 18 scripted languages: C, C++, Java, C#, etc). Like RA (1998), it was immediately deleted as a "virus" by Microsoft partners called in 2004 the VIA (Virus Information Alliance). Porting G-WAN to Linux made it faster and more scalable. Despite its many features, G-WAN uses less CPU/RAM than NGINX - and had no vulnerabilities.

### Directory Server (2003-2009)

The patented DS (Directory Server) allowed small businesses and large accounts deploy RA without configuring routers and firewalls, manage access rights, deploy and inventory assets - without any setup. It also allowed people to "recycle" RA licenses so that any newly deployed slot would replace the oldest slot (among the slots marked as available for recycling – others being "fixed" slots).

### Remote-Anything (1998-2009)

As a Desktop-Sharing and file-transfer application (competing with HP Carbon-Copy, Symantec PCAnywhere, and Traveling Software Laplink), RA was much smaller (one single 50 KB executable file), much faster (even on slow links), and much safer (no vulnerability in its lifetime).

For more information, contact TWD at:

+41 554-142-093
(Worldwide)

TWD Industries AG
Paradiesli 17
CH-8842 Unteriberg

twd.ag