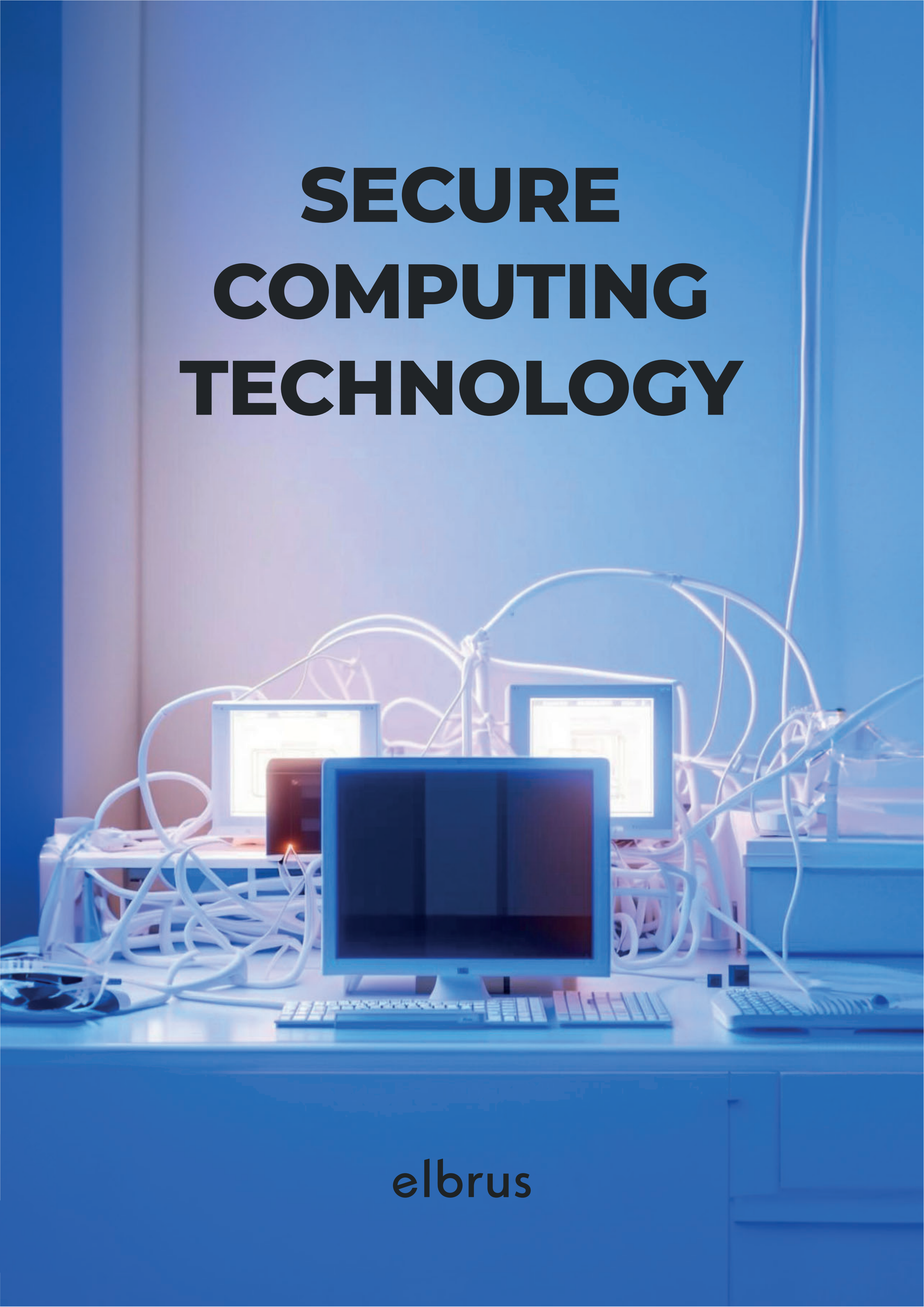
SECURE COMPUTING TECHNOLOGY



elbrus

SECURE COMPUTING TECHNOLOGY

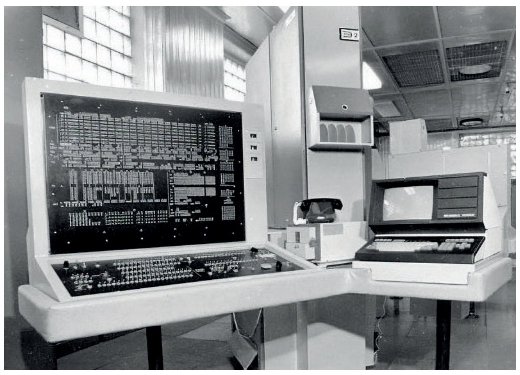
Secure Computing Technology (SCT) is one of fundamental features of Elbrus architecture, conceived according to “Secure by Design” principle. It is a special mode of operation with fine-grained memory access control — at object-level rather than page-level — implemented in Elbrus processor hardware. This technology not only prevents most common types of security vulnerability exploits, including the notorious “buffer overflow” attack, but also helps with debugging software during its development — thanks to processor hardware detecting violations and stopping the program the very same moment it attempts to perform an illegal memory operation (rather than allowing it to run further until more memory is corrupted to the point of utter malfunction with no trace back to the origin of primal error).

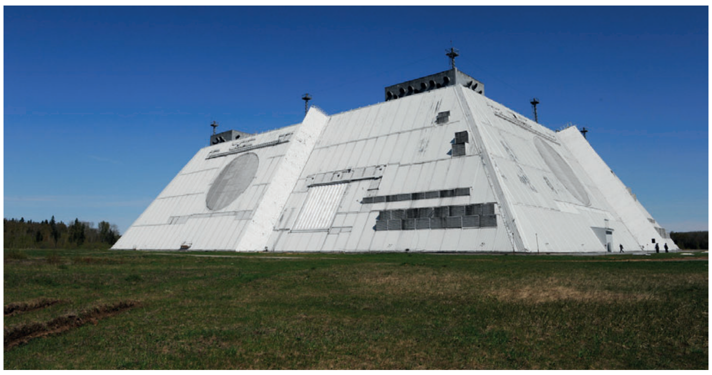
The demand for hardware-enforced control over software functioning has seen sharp increase during the last decade. One of the most renowned projects in this area is CHERI (Capability Hardware Enhanced RISC Instructions), carried out by University of Cambridge, UK, with implementations for ARM and RISC-V architectures. Cybersecurity and Infrastructure Security Agency of the USA promotes such technologies as well. Elbrus platform is leading this trend with its production-grade Secure Computing implementation.

SECURE BY DESIGN

HISTORY OF SECURE COMPUTING

Soviet-built Elbrus-1, Elbrus-2 and Elbrus-3 computer systems had a complete implementation of SCT (called “Protected Execution Mode” at the time), with majority of system-level and application-level software running this way, as well as software development and debugging — making it fast and user-friendly for the developers. That led to the A-135 air defense system, covering the Moscow region and based on Elbrus-2, to be put into service in a record-breaking timeline. Developed by several independent teams spread throughout the vast country, the system contained over 1 million lines of source code, and it was SCT that made such a large-scale project a success.


Elbrus-2 computer

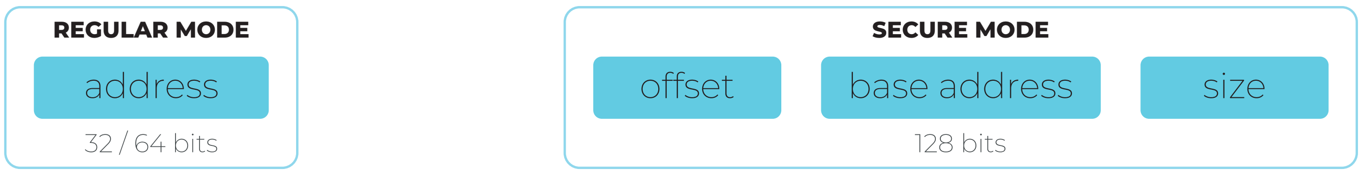

A-135's Don-2N radar station

CORE ELEMENTS OF SECURE COMPUTING

DESCRIPTORS INSTEAD OF POINTERS

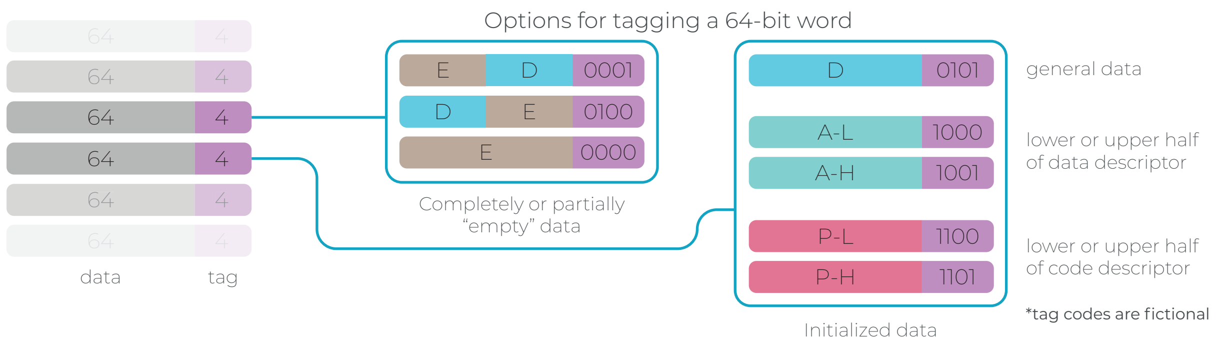
In its regular mode of operation, just like other processors (x86, ARM, etc.), Elbrus accesses memory using ordinary addresses — plain numbers represented by 32- or 64-bit values. From the processor’s point of view, those numbers are no different from other data stored in registers or in memory, thus may easily be manipulated by regular arithmetic instructions, or transferred to/from arbitrary locations.

When in secure mode, Elbrus uses 128-bit structures, called descriptors, which combine object’s base address, current offset and total size. From the programmer’s point of view, the use of descriptors is totally transparent in high-level languages, provided that language standards are adhered, including the use of established abstractions for data size (such as `sizeof` operator) instead of hardcoded values — because each pointer will consume 128 bits rather than 32 or 64.



HARDWARE-ENFORCED MEMORY TAGGING

For each 64-bit word, Elbrus adds a 4-bit tag — a shadow value that describes the kind of data stored in that 64-bit word: an empty (uninitialized) value, general data, lower or upper half of a descriptor. More precisely, there are distinct kinds of descriptors — one for data and one for code; also, an empty value may be tagged in either lower or upper 32-bit half of 64-bit word. Competing processors without hardware-enabled memory tagging may use software emulation, though that incurs very high performance overhead and additional memory consumption. Elbrus hardware provides near-native performance and stores the tags in extra ECC memory cells (still leaving enough space for error correction codes), making the Secure Computing affordable even for production use.


Options for tagging a 64-bit word
64
4
data
tag
E
D
0001
D
E
0100
E
0000
Completely or partially "empty" data
D
0101
general data
A-L
1000
A-H
1001
lower or upper half of data descriptor
P-L
1100
P-H
1101
lower or upper half of code descriptor
*tag codes are fictional
Initialized data

HARDWARE-ENFORCED DATA INITIALIZATION CONTROL

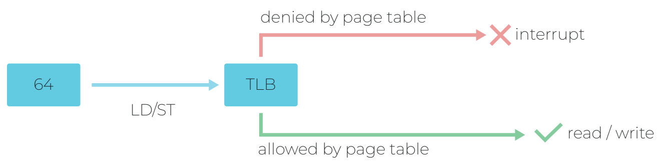
As long as a word, or its half, is tagged as "empty value" (by default), the processor may load it into register but will refuse to store the result of any attempted operation. Once a meaningful content is written, the tag is adjusted accordingly.

HARDWARE-ENFORCED MEMORY ACCESS CONTROL VIA DESCRIPTORS

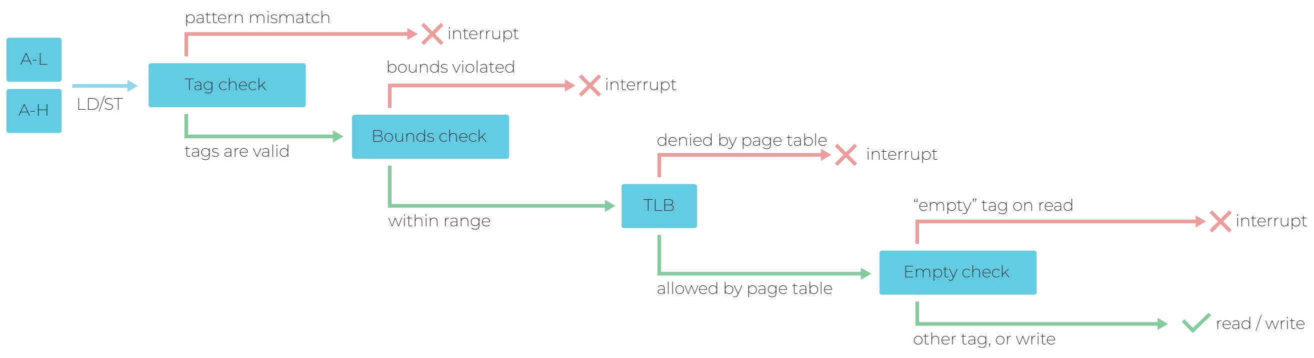
In its regular mode of operation, just like other processors (x86, ARM, etc.), Elbrus authorizes memory access based on page table solely — providing control granularity at page scale on per-process basis. When in secure mode, Elbrus also checks that the requested memory range is within the bounds of a specific object (an array, a structure, or a basic field — depending on what the descriptor holds), providing control granularity on a byte-wise level within a process. If a descriptor is supposed to represent code rather than data, Elbrus checks that it really targets an executable segment, not data.

Upon detecting a violation, Elbrus processor hardware signals an interrupt to stop the program immediately at the point of erroneous request. The checks are performed in constant time before cache memory is queried, therefore preventing Spectre-class timing attacks.
Tagging ensures that descriptors can only be created in a proper fashion and not constructed manually, nor tampered by modifying an existing descriptor. Hence the integrity of bounds control is assured.

REGULAR MODE



SECURE MODE



COMPILER, KERNEL AND SYSTEM LIBRARIES SUPPORT

As of now, SCT is available for Linux user-space applications: both the kernel and system libraries support the use of 128-bit descriptors as an alternative to regular 32- and 64-bit mode addressing (more precisely, it is a multi-lib solution with distinct binaries for each variant). There is also `syscall` translation layer provided which performs additional validation of input parameters passed to kernel.

SECURE COMPUTING OBJECTIVES

- Detection of most common programming errors:
  - buffer overflow (ex.: CWE-119 ... CWE-122),
  - use of uninitialized data (ex.: CWE-457),
  - use after free (ex.: CWE-416).
- Prevention of side-channel information leaks (ex.: Spectre).
- Isolation of address space at object level that is more efficient than process- and page-level isolation.

ADOPTION CHALLENGES

As of now, only a limited (although sufficient for many applications) set of system libraries and utility programs has been ported to SCT. The major challenges for porting are:

- hardware-dependent programming techniques (explicit and implicit assumptions about pointer size, bit-wise logical operations on pointers as if they were ordinary numbers, type-casting of pointers to numbers and back to pointers) that need to be recognized and rewritten in hardware-agnostic way;
- programming errors (such as the use of uninitialized data) that need to be identified and fixed.

MCST leads its ongoing initiative to make more software available for Secure Computing, and opens a collaborative platform for this task.
Worth to note is that the advantages of SCT come at a price — as more memory and registers are spent for storing-128 bit descriptors, and time is spent for tagging memory as "empty" by default. The performance hit is 10–20 % on average, which may usually be considered insignificant, though.

SIMILAR TECHNOLOGIES

Secure Computing Technology was implemented in every generation of Elbrus computers, starting with Elbrus-1 in 1973, and was unique in its class for quite a long time. The closest counterpart nowadays is CHERI project run by a research team in Cambridge since 2010 and supported by major chipmakers.

VALUE FOR IT INFRASTRUCTURE SAFETY AND ROBUSTNESS

Information system security depends on the quality of its program code. According to a study conducted by ARM, around 70 % of discovered vulnerabilities are memory-related. Programming languages with automatic memory management, like Java and .NET, are safe from this but not suitable for many use cases, so unsafe C and C++ forming the foundation of Linux ecosystem will still remain paramount in foreseeable future.
Static and dynamic code analysis and certification software tools have their limitations. Therefore, the only reliable remedy is hardware-enforced security. That fact is pointed out in a recent research report published by US Cybersecurity and Infrastructure Security Agency, which recommends using "Secure by Design" approach in general and CHERI as a particular example.

Secure Computing Technology implemented in Elbrus processors is a well-established combination of hardware capabilities and corresponding software stack, already used in production. Its broader adoption could lower the security risks significantly, as well as make software development more efficient. While alternative solutions may require considerable efforts and expenses, Secure Computing Technology is *the* answer to vital and pressing issues of IT sustainability.

RUSSIAN VERSION
OF THIS DOCUMENT