BASIC SOFTWARE PROTECTION TOOLS AGAINST HACKING

Organizational security features:

Software Access Policy.

2. Technical customer support.

3. Personalization of licenses.

Legal remedies:

1. Patenting of technical solutions.

2. Contracts and license agreements.

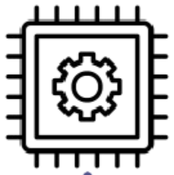
Fighting websites, distributing hacked software

Software protection tools:

1. Encryption of program code and files.

2. License activation and management.

3. Countering software reverse engineering.



Software and hardware security features:

1. Generating the HWID.

2. Using USB keys.

2. Physical separation of software software and equipment.

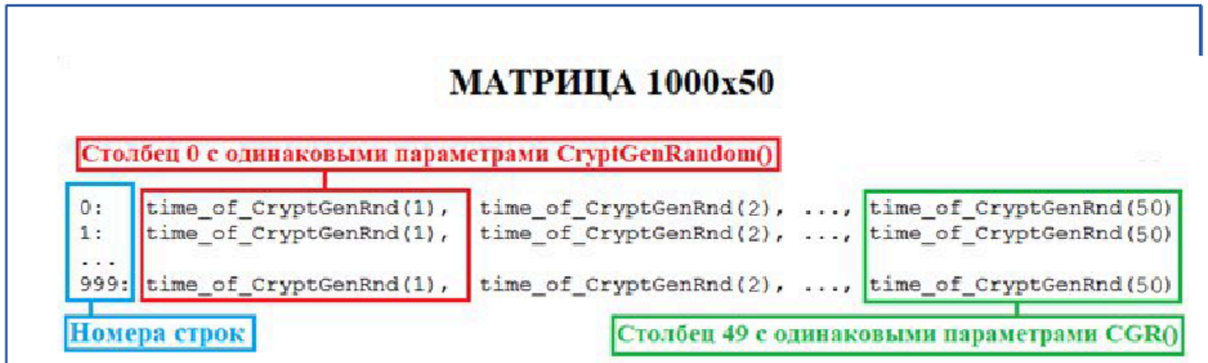
STATIC AND DYNAMIC IDENTIFICATION

Static identification is defined as obtaining the PC's HWID based on various methods. hardware or software identifiers. This approach is also referred to as an attribute approach (getting various attributes /properties of the installed device hardware or software).

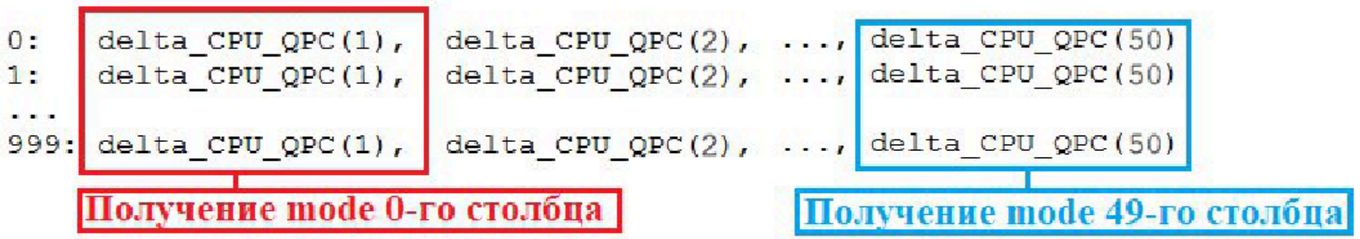
Dynamic identification of PC hardware - measurement of several unique features parameters, the combination of which can uniquely identify the set A PC with the same hardware and software. Such HWID generation is harder to detect and circumvent, because there is no explicit use of any static characteristic.

DYNAMIC IDENTIFICATION VERSION # 1

An attempt was made to implement dynamic identification of the PC hardware **based on the code execution speed matrix.**



Forming the execution speed matrix CryptGenRandom()



Расчёт mode матрицы

Basic implementation steps:

1. Select the function to measure
2. Select hardware timers
3. Generate matrices, calculate the mode of each matrix.
4. Compare the mode of two matrices
5. Matrices are equal if there are more than 50 matches%

DYNAMICUSER IDENTIFICATION CLARIFYING DETAILS

After the first version failed, an attempt was made to contact the authors of the scientific papers. articles in this area to find out a number of questions. The following responses were received:

1. Use standard Windows APIs.
2. Don't work with the GPU.
3. Use two timers with different frequencies.
4. The more accurate both timers are, the better.
5. Data about the characteristics of the PC with tests is lost.
6. Matrix for each timer, and then subtract modulo.

The author of the article does not remember the key details that are missing from the article.

B OC Windows there are several different timers available to use:

1. _RTC with a frequency of 32.768 kHz.
2. _PIT with a frequency of 1.193180 MHz.
3. _ACPI with a frequency of 3.58 MHz.
4. _NRET with a frequency from 10 to 24 MHz.
5. _LAPIC with no fixed frequency.

DYNAMIC IDENTIFICATION VERSION #2

Based on the responses of the authors of scientific articles, it was decided to study and test how to measure the speed of code execution based on processor ticks as the most accurate measurement option – for the first matrix. WinAPI QueryPerfomanceCounter for the second matrix.1fence

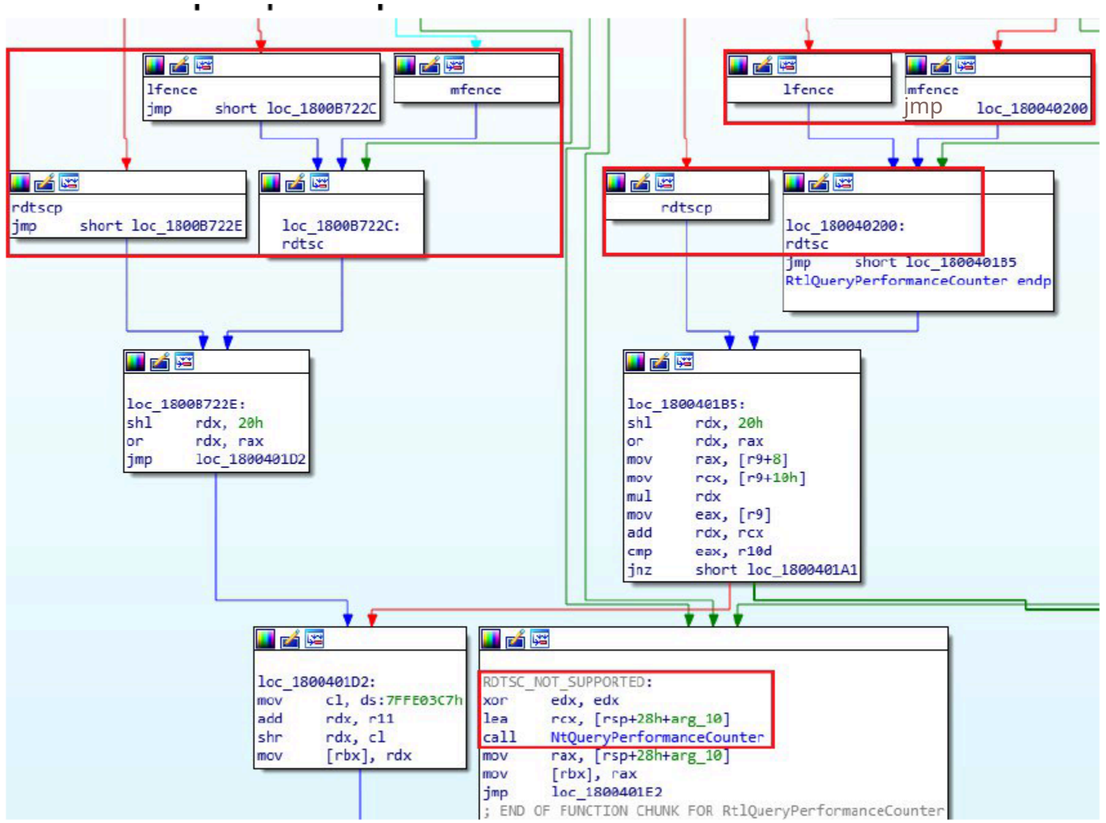
```
LFENCE ()
        StartTicks = RDTSC()

        ... measured code...

LFENCE ()
        EndTicks = RDTSC()

        DiffTicks = EndTicks - StartTicks
```

Measuring processor ticks



Asm function listing ntdll. RtlQueryPerformanceCounter

DYNAMIC IDENTIFICATION VERSION #2. results

Tests on one PC showed 100% matching of delta matrices in 10 cases out of 10 without load on the CPU. However, matrices generated without CPU load and with a load are considered to be two matrices that are not very similar (the percentage of identity varies about 70%).

Software comparisonreceived and expected
stability of delta matrices

Recommendations for further research:

1. Tests on multiple PCs with the same settings.
hardware components.
2. Work with NRET and ACPI timers from the kernel.
3. Stabilize the RAM frequency.
4. Apply STM.
5. Tests on different Intel & AMD CPU models.

results

1. The native version of HWID generation based on speed matrix measurements could not be implemented code execution in 2025 on both old and new hardware.

2. There is no ready-made implementation on the Internet (there are 2 projects-attempts for 100 lines of code in C and JS, the authors of which are not we tried to dig deeper, but just took the first time measurement function and tried to solve it head-on issue)

3. The author of the article (Iskander Sanchez-Rola) says that "it was so long ago, I already forgot everything. No source code" 4. AI also solves the problem head-on (incorrectly), does not answer anything worthwhile.

In the end, is it real? I think so, but the article clearly hides key details for implementation. Perhaps this was done intentionally and right now a working implementation is spinning on someone's prod for detecting bots, counting uniques, and so on.

An even deeper reserch is needed (it took me several months to complete this research in the WRC format jobs after the main job).