

Malicious code detection tool

# Static and dynamic analysis tool

Our software can do both static and dynamic detection analysis

1. static - using regex for pattern matching with locally saved malicious database without executing the code
2. Dynamic - by using docker creating an isolated environment run user's code for security and also to see how the code will react to sensitive data.

# Tools to use

**Qt:** Cross-platform framework for developing graphical user interfaces (GUIs).

**Regex Tool:** A tool used for pattern matching or detecting specific text patterns.

**Docker Environment:** A platform to create isolated and portable containerized environments.

**OpenSSL:** A toolkit for implementing secure communication using encryption protocols.

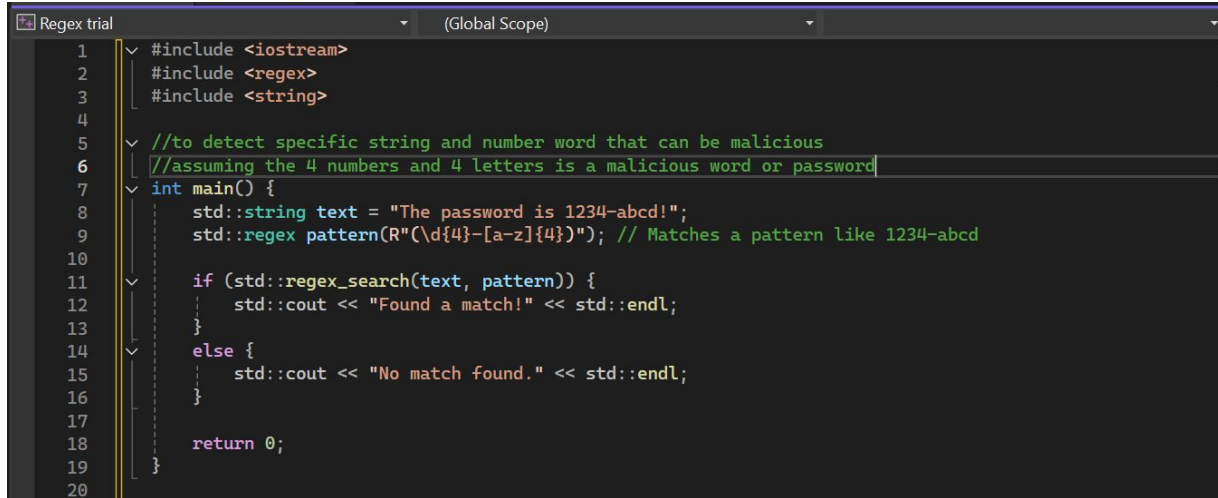
**Inno Setup:** A free installer tool for packaging software into a single executable file

# Regex rules

```
//regex pattern

/*Character Classes:
:
[a-z]: Matches any lowercase letter.
[A-Z]: Matches any uppercase letter.
[0-9]: Matches any digit.
Quantifiers:
:
*: Matches zero or more occurrences (e.g., a* matches a, aa, or nothing).
+: Matches one or more occurrences (e.g., a+ matches a or aa).
{n}: Matches exactly n occurrences (e.g., \d{3} matches exactly three digits).
Anchors:
:
^: Matches the start of a string.
$: Matches the end of a string.
Groups and Alternation:
:
(...): Groups patterns (e.g., (abc|def) matches either abc or def).
Special Characters:
:
\d: Matches any digit (equivalent to [0-9]).
\w: Matches any word character (alphanumeric + underscore).
\s: Matches any whitespace character (space, tab, newline).*/
/*
```

# Regex Pattern matching sample code



The screenshot shows a code editor window titled "Regex trial" with a "(Global Scope)" dropdown. The code is written in C++ and demonstrates how to use the `std::regex` library to search for a specific pattern in a string. The pattern is `R"(\\d{4}-[a-z]{4})"`, which matches a sequence of four digits followed by a hyphen and four lowercase letters. The code defines a `main` function that sets a text string to "The password is 1234-abcd!" and uses `std::regex_search` to check if the pattern is present. If a match is found, it prints "Found a match!"; otherwise, it prints "No match found.".

```
1  #include <iostream>
2  #include <regex>
3  #include <string>
4
5  //to detect specific string and number word that can be malicious
6  //assuming the 4 numbers and 4 letters is a malicious word or password
7  int main() {
8      std::string text = "The password is 1234-abcd!";
9      std::regex pattern(R"(\\d{4}-[a-z]{4})"); // Matches a pattern like 1234-abcd
10
11     if (std::regex_search(text, pattern)) {
12         std::cout << "Found a match!" << std::endl;
13     }
14     else {
15         std::cout << "No match found." << std::endl;
16     }
17
18     return 0;
19 }
20
```

# Regex sample on how to detect sql

```
21
22 //to identify SQL injections
23
24 #include <iostream>
25 #include <regex>
26 #include <string>
27
28 int main() {
29     std::string userInput = "SELECT * FROM users WHERE username = 'admin' --";
30     std::regex pattern(R"(--\s*)"); // Matches SQL comments
31
32     if (std::regex_search(userInput, pattern)) {
33         std::cout << "Potential SQL injection detected!" << std::endl;
34     }
35     else {
36         std::cout << "No security issues found." << std::endl;
37     }
38
39     return 0;
40 }
```