ISSN: 2277-9655
Impact Factor: 1.852

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

Quines: A Survey

Ankur Singh Bist
Govind Ballabh Pant University of Agriculture and Technology, India
ankur1990bist@gmail.com

Abstract

In this paper a survey study is made on various issues of Quines and their impact on malicious activities. Different scenarios are discussed to make analysis of concerned problem. Various approaches are also discussed to focus on problems that have been found in this domain.

Keywords: Computer viruses, Quines.

Introduction

Now days the use of internet has become a common activity. It caused the increment in the activities produced by computer viruses. The damaging activities caused by malicious codes are increasing day by day causing harm to society. There are various schemes that are designed to defend against computer viruses.

A quine is a computer program which takes no input and produces a copy of its own source code as its only output. The standard terms for these programs in the computability theory and computer science literature are self-replicating programs, self-reproducing programs, and self-copying programs [1].

Quines

A quine is a fixed point of an execution environment, when the execution environment is viewed as a function [1]. Quines are possible in any programming language that has the ability to output any computable string, as a direct consequence of Kleene's recursion theorem. For amusement, programmers sometimes attempt to develop the shortest possible quine in any given programming language [1].

The following Java code demonstrates the basic structure of Quine [1].

```
public class Quine
{
 public static void main( String[] args )
 {
  char q = 34;     // Quotation mark character
  String[] l = {   // Array of source code
  "public class Quine",
  "{",
  "  public static void main( String[] args )",
  "  {",
  "    char q = 34;      // Quotation mark character",
  "    String[] l = {    // Array of source code",
  "    ",
  "    };",
  "    for( int i = 0; i < 6; i++ )            // Print opening code",
  "      System.out.println( l[i] );",
  "    for( int i = 0; i < l.length; i++ )     // Print string array",
  "      System.out.println( l[6] + q + l[i] + q + ',' );",
  "    for( int i = 7; i < l.length; i++ )    // Print this code",
  "      System.out.println( l[i] );",
  "  }",
  "}",
  };
  for( int i = 0; i < 6; i++ )          // Print opening code
    System.out.println( l[i] );
  for( int i = 0; i < l.length; i++ )   // Print string array
    System.out.println( l[6] + q + l[i] + q + ',' );
  for( int i = 7; i < l.length; i++ )   // Print this code
    System.out.println( l[i] );
 }
```

The source code contains a string array of itself, which is output twice, once inside quotation marks.

Self-organization and replication

While features of self-organization and self-replication are often assumed the hallmark of living systems, there are many instances of abiotic molecules exhibiting similar characteristics under proper conditions. For example Martin and Russel show that physical compartmentation by cell membranes from the environment and self-organization of self-contained reduction oxidation so called reduction reactions are the most conserved attributes of living things, and they argue therefore that inorganic matter with such attributes would be life's most likely last common ancestor [2] .

ISSN: 2277-9655
Impact Factor: 1.852

Virus self-assembly within host cells has implications for the study of the origin of life, as it lends further credence to the hypothesis that life might have started as self-assembling organic molecules [2].

Multi-quines ---Introns

We initialize by saying what a bi-quine (or more generally a multi-quine) is. To start, here is what it is *not*: a bi-quine is *not* a program which prints a second program, which in response prints the first again (actually, it is that, but things are a bit more subtle) [2]. This is too simple to do (we have proved the existence of such using the fixed-point theorem --- one program is almost a quine, and the other is only a sequence of calls to print the code of the other one [2,3] .

A multi-quine is also *not* a polyglot quine (a quine that can be read, and is a quine, in several different languages) [2,3] . True, polyglot quines in real are multi-quines if you imagine well about it (the converse is not true), but polyglot quines don't exist for every proper combination of programming languages (although it is true that some people have been incredibly smart at constructing them) whereas multi-quines do — polyglot quines are a hack whereas multi-quines are a general phenomenon [3].

A *bi-quine* is a very interesting kind of program: when run normally, it is a quine. But if it called with a particular command line argument, it will print a program that will look different, its “brother”. Its brother is *also* a quine, but in a different programming language, so its brother prints its self listing when run normally. But when run with a specific command line argument, the brother prints the listing of the original program [3] [4].

```
UPDATE ContainerContents SET OldContents='%contents%' WHERE TagID='%id%'
```

Query 1 - Updating known contents[4]

```
%content%' WHERE TagId='%id%';

SET @a='UPDATE ContainerContents SET NewContents=concat(\'%content%\\\' WHERE TagId=\\\'%id%\\\'; SET @a=\', QUOTE(@a), \'; \', @a); %payload%; --';

UPDATE ContainerContents SET NewContents=concat('%content%\' WHERE TagId=\'%id%\'; SET @a=', QUOTE(@a), '; ', @a); %payload%; --
```

Exploit 1 - SQL virus using quines for MySQL. Whitespace is for readability only [4].

The architecture and organization of quines or multi-quines give rise to malicious activities. It is required to develop methods to search our pattern for these entities to avoid them in malicious activities. Pattern matching approaches and code emulation can be used to detect these types of problems.

Conclusion

In this paper we reviewed various approaches and methods to explain various issues of Quines. The purpose of this analysis is to evolve the solution set for the particular problem of viruses caused by quines and try to evolve some more efficient approaches for concerned problem. We organised the information that will make vision clear to all those working in this area.

References

[1] www.wikipedia.com.
[2] http://en.wikipedia.org/wiki/Abiogenesis
[3] http://www.madore.org/~david/computers/quin.html
[4] http://www.rfidvirus.org/exploits/sql_quine/index.html