Identifying Algorithmic Complexity Vulnerabilities Caused by Input-Dependent Nested Loops

Mujahid Masood,Rafique Nazir
mujahid.masood@stud.tu-darmstadt.de,rafique.nazir@stud.tu-darmstadt.de

Abstract

The single-threaded event model of JavaScript makes it vulnerable to a specific class of denial of service attack called algorithmic complexity attacks. These attacks consist of exploiting the worst case performance of algorithms to trigger slow computations that block the event loop for a large period of time. In this project we study the prevalence of a particular class of algorithmic complexity vulnerabilities called input-dependent nested loops.

1. Introduction

1.1 Motivation

Loops are the very important programming construct.If a function uses a loop,depending upon the number of iterations of the loop;a significant amount of execution time of a function, is used by a loop.For example in listing 1 ,total execution time of 10 iterations of a nested loop is approximately 35 milliseconds.In simpleLoop function,number of iteration is controlled by the variable length,which is initially set to 10.If variable length's value is changed from 10 to 100,loop execution time changes significantly from 35 milliseconds to 3.5 seconds,which is 100 times of the original execution time.

Listing 1. Simple nested loop

```
function simpleLoop() {

        var length=10;

    for (var i = 0; i <length;i++) {

      for (var j=0;j <length;j++) {
        console.log(i,j);
      }
    }

}
```

In some cases,variables controlling the loop iteration, are also passed to a function as a parameter as shown in listing 2.We pass a parameter named array to function selectionSort,which is used by two nested loops inside the function.These kind of input dependent loops are prone to attacks.For example,If an attacker controls the input of the function selectionSort in listing 2 he or she can increase the overall execution time of the function.In our example,if we pass an array of size 200 to selectionSort its execution time is approximately 5 seconds.

Listing 2. Quadratic complexity algorithm for computing a repetitive sum

```
function selectionSort(array) {
 for (var i = -1; ++i < array.length;)
     {
     for (var m = j = i; ++j < array.length;)
     {
       if (array[m] > array[j])
          m = j;
       }

        var temp = array[m];
        array[m] = array[i];
        array[i] = temp;
    }
    return array;
}
```

2. Problems Of Using Function Input Dependent Loops

Most of the npm modules use function input dependent loops and they are deployed in the production environments. For example useragent,dash and react-metric-graphics are the few npm modules which use function input parameters in loops.An attacker can easily exploit function input dependent loop vulnerabilities for ReDos attacks.

In our project we perform static analysis on the npm modules using the *Google Closure Compiler* to identify the function input dependent loop vulnerabilities in the npm modules.

3. Our Approach

4. Future Work

How to automatically verify the correctness of the identified vulnerabilities

4.1 Citations

Use citations to refer to other papers (Herlihy and Moss 1993; Fraser et al. 1992) and books (Jr. and White 2000; Aho et al. 1986).

4.2 Tables

Table 1 shows how a table looks like.

4.3 Figures

Figure 2 shows a simple figure with a single picture and Figure 3 shows a more complex figure containing subfigures.

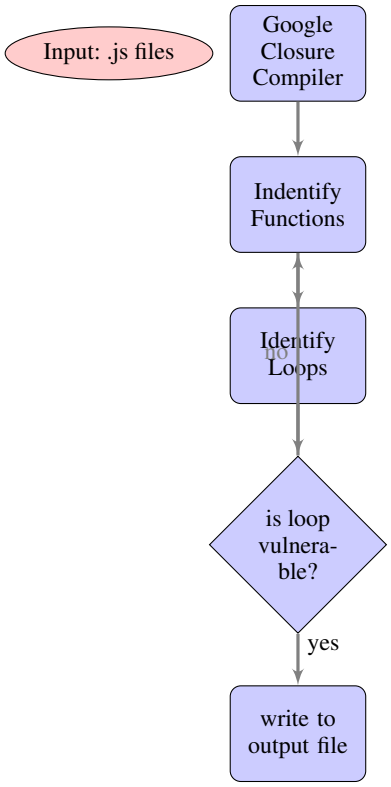

Figure 1. Workflow for indentifying function input dependent loops in npm modules

English	German
cell phone	Handy
Diet Coke	Coca Cola light

Table 1. Translations.

Figure 2. SOLA logo.

(a) TUDaLogo (b) SOLALogo

Figure 3. Two pictures as part of a single figure through the magic of the subfigure package.

4.4 Source code

The listings package provides tools to typeset source code listings. It supports many programming languages and provides a lot of formatting options.

Listing 3. Example usage of the listing package

```
class S {
    int f1 = 42;
    public S(int x) {
            f1 = x;
    }
}
```

Listing 3 shows an example listing. Code snippets can also be inserted in normal text: `\lstinline|int f1 = 42;|` gives `int f1 = 42;`

4.5 Miscellany

Capitalization. When referring to a named table (such as in the previous section), the word *table* is capitalized. The same is true for figures, chapters and sections.

Bibliography. Use `bibtex` to make your life easier and to produce consistently formatted entries.

Contractions. Avoid contractions. For instance, use “do not” rather than “don’t.”

Style guide. A classic reference book on writing style is Strunk’s *The Elements of Style* (Jr. and White 2000).

5. Limitations

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

6. Results

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

7. Conclusion

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

References

A. V. Aho, R. Sethi, and J. D. Ullman. *Compilers: principles, techniques, and tools*. Addison-Wesley Longman Publishing Co., Inc., 1986. ISBN 0-201-10088-6.

C. Fraser, D. Hanson, and T. Proebsting. Engineering a simple, efficient code generator generator. *ACM Letters on Programming Languages and Systems*, 1(3):213–226, Sept. 1992.

M. Herlihy and J. E. B. Moss. Transactional memory: Architectural support for lock-free data structures. In *Proc. of the 20th Intl. Symp. on Computer Architecture (ISCA'93)*, pages 289–300, 1993.

W. S. Jr. and E. B. White. *The Elements of Style*. 2000.