Secure Coding in Nanorobotics Applications

Whitney Nelson
Department of Computer Science
Hampton University
whitney.nelson@my.hamptonu.edu

Chutima Boonthum-Denecke
Department of Computer Science
Hampton University
chutima.boonthum@hamptonu.edu

ABSTRACT

Nanorobotics is an upcoming technology used across a vast spectrum of fields, such as healthcare, surgery, security, and military. It has become an important aspect in technology because of its ability to solve enormous problems with device designed in nanometers. Nanorobots will be able to reach places, fix the tiniest problems, and affect different attributes that were once unimaginable just ten years ago. Nanorobotics is a growing phenomenon and with such great capabilities, security will be an essential aspect of this technology. Every day there are accounts, private information, and government files that are attacked and threatened by outside resources. Just as any newly formed piece of technology, nanorobots have the potential of being attacked by hackers and terrorists. This research project will include an in-depth review on security practices related to nanorobotics applications and a comparative analysis of programming in Java and C++, which are two programming languages implemented for a nanorobotics application. The simulation programs resembles (a) a task similar to what a nanorobot would have and (b) an adversary program that will find security concerns or vulnerabilities that appear in both languages. Finally, this research will provide a set of the best security practices in coding in nanorobotics application.

CCS CONCEPTS

• **Security and privacy → Software and application security →** Software security engineering; Usability in security and privacy.

KEYWORDS

Secure Software Engineering, Secure Coding, Nanorobot

ACM Reference format:
W. Nelson, C. Boonthum-Denecke. 2017. Secure Coding in Nanorobotics Applications. In *Proceedings of 2017 ADMI Symposium*, Virginia Beach, Virginia USA, March 23-26, 2017.

1 INTRODUCTION

Nanorobots are devices ranging in size from 0.1-10 nanometers and constructed of nanoscale or molecular components. These nanorobots have the ability to affect a vast amount of areas such as medicine, surveillance, agriculture, military etc. Artificial non-biological nanorobots have not so far been created; they remain a hypothetical concept at this time. However, there are biological nanorobots that are currently in existence which proves the ability of creating artificial nanorobots. As nanorobots becoming a developing product, security techniques must be taken into consideration. As mentioned before, nanorobots will be affecting many important areas and without the correct security techniques threats could become an issue.

In the last 20 years the amount of cyber threats that have been imposed on products continues to increase each year. As technology becomes more available to everyone, more hackers have tried to take advantage by stealing, cheating, or even harming others. There are four main types of threats or security concerns that arise in nanorobots: malicious insiders, exploited insiders, careless insiders, and self-replication. Malicious insider is the typical hackers that are seen on the news. These threats usually happen when people purposely going into an unauthorized technology and is then attacked to hurt, harm, or steal from the owners of that particular piece of technology. Exploited insiders use hacking for selfish intent. These types of insiders usually attack software with a selfish motivation. Careless insiders are people who use technology carelessly. These are people who do not take the necessary precautions to use software and can cause security problems with such recklessness. Nanorobots may have the ability to self-replicate. This capability can cause many dangerous issues within the body. There could be a possible overload of nanorobots doing the same functions overall leading to a negative outcome.

2 SECURE CODING

Securing coding is the practice of developing software to prevent potential vulnerabilities. Vulnerabilities include bugs, defects, and logic flaws that can be exploited throughout software. Many developers make the mistake of not having security as a top concern during all stages of development and maintenance. This practice forces developers to go back into software to implement security significantly increasing the cost of production of software. For this reason, it is important for developers to produce secure code from the initial stages of development. The Software Engineering Intitule at Carnegie Mellon University defined the top ten security protocols that be used during development [4]:

1. Validate input.
2. Heed compiler warnings
3. Architect and design for security policies.
4. Keep is simple
5. Default deny
6. Adhere to the principle of least privilege
7. Sanitize data sent to other systems
8. Practice defense in depth
9. Use effective quality assurance techniques
10. Adopt secure coding standard.

Applying these rules become the responsibility of the developer and should be requirements for implementation. Adhering to a guideline of practices always helps to reduce vulnerabilities although the total elimination of flaws may not be possible. Despite its imperfection, striving for the reduction of most vulnerabilities is essential in application design.

2.1 Secure Coding in Java (Overview)

There are aspects of Java that were created with security in mind. For example, pointer manipulation is not easily accessible to the programmer as well exceptions are thrown when a string or an array is out of bounds. James A Goslin in Java Coding Guidelines outlines the important attributes on secure coding when developing in Java. Limiting the lifetime of sensitive data on the memory, providing sensitive mutable classes with unmodifiable wrappers, storing passwords in hash functions, and minimizing the privilege code are just a few examples of essential guidelines to developing secure code in java. Java has vulnerabilities that show up in deadlocks and race conditions as well because of its capability to multi-thread.

2.2 Secure Coding in C++ (Overview)

C++ is an objected-oriented language whose versatility in functionality help gain its popularity. However, this versatility has causes vulnerabilities when coding in this language. C++ has careless access to memory that can cause buffer overflows and format string attacks. It also does not check bounds which lead to errors or modification in code. Despite this major vulnerability issue in C++, there are also some key security features. Safe-Secure C/C++ (SSCC) is a set of methods to eliminate vulnerabilities resulting from buffer overflows and other programming errors C++ using a mixture of compile-time, link-time, and runtime tests, plus some design-time restrictions. C++ also enforces access-control restrictions within thread safety attributes to limit race condition vulnerabilities [3].

2.3 Security Vulnerabilities Related to Nanorobot

As discussed before, nanorobots will be changing the face of important industries like Medicine and Military. The proposed responsibilities suggest that nanorobots will be holding sensitive information like military data, medical information, and performing medical procedures. With this in mind, it is essential that the design of nanorobots takes security as its top priority. Capacity is not a concern on the nanoscale which is an advantage towards security. However, subjects such as privileges, validation, and defects are vulnerabilities that should always be avoided when developing nanorobotic applications. We are looking for areas of vulnerabilities in nanorobotic applications as well as ways to improve those areas.

3 METHOD AND DESIGN

In this research project, we looked at two major object oriented languages to focus in on: Java and C++. These two languages were chosen not only because their popularity in overall usage in programming but also because they are more versatile and more likely to be used in robotic applications. Java and C++ also have security features that can be implanted to produce a safe application.

We chose to look at how a spy robot would need to have secure features. In theoretical sense, a spy nanorobot would access its way into a room, capture some bit of information and return that information safely to a database. More than likely because the nanorobot is a "spy," it would capture sensitive data that should be protected against hacker's access. Thus, a method must be provided to ensure the safety of this cyber traveling of information.

Encryption and decryption are two methods used to hide or keep particular information confidential. We used these two methods in the initial design of the two programs we will be running to protect against malicious software, virus, worms, hackers, or other threats. The most important aspect of this design is security implementation in every function in the initial phases.

Despite the implantation of the programs being embedded in security, no code is perfect. This is why we will design a malicious piece of software whose goal will be to decrypt the protected information. This software will work as a test, looking for different flaws and vulnerabilities that may reside in each language.

3.1 Sample Code in Java

```
class Password {
    public static void main (String args[])
                throws IOException {

        Console c = System.console();
        if (c == null) {
            System.err.println("No console.");
            System.exit(1);
        }

        String username =
             c.readLine("Enter your user name: ");

        String password =
             c.readLine("Enter your password: ");

        if (!verify(username, password)) {
            throw new SecurityException(
                       "Invalid Credentials");
    }
}
```

Figure 1. Vulnerable Java Code

In the sample above the console reads in a String object for the user name and password that remains exposed until the garbage collector reclaims the memory. This exposure makes sensitive data more vulnerable to leakage. This can be limited by using the **`console.readPassword()`** method allowing the password to be returned in a characters versus a String object(shown below). Thus, the password is cleared form the array right after its usage.

```
class Password {
    public static void main (String args[])
                throws IOException {

    Console c = System.console();

    if (c == null) {
        System.err.println("No console.");
        System.exit(1);
    }

    String username =
          c.readLine("Enter your user name: ");
    char[] password =
          c.readPassword("Enter your password: ");

    if (!verify(username, password)) {
        throw new SecurityException
                ("Invalid Credentials");
    }
    // Clear the password
    Arrays.fill(password, ' ');
}
```

Figure 2. Safe Java Code

3.2 Sample Code in C++

The code below shows a vulnerability where an array is out of bounds

```
01  bool IsValid(void) {
02    char Password[12];
03
04    gets(Password);
05   return 0 == strcmp(Password, "correct");
06  }
07
08  int main(void) {
09    bool passStatus;
10
11    puts("Enter password:");
12    passStatus = IsValid();
13    if (PwStatus == false) {
14      puts("Access denied");
15      exit(-1);
16    }
17  }
```

Figure 3. Vulnerable C++ Code

This code results in a vulnerability when the user enters an amount of characters above 12. The **gets()** function copies characters from standard input into Password until end-of-file is encountered or a newline character is read. The Password array contains only enough space for an 11-character password and a trailing null character. Since the array only has space for 11 character, it results in writing beyond the bounds of the array. This vulnerability can be avoided by using **`strcpy()`** which copies only the source string, not allowing the null character create the buffer overflow.

```
01  char buffer[128];
02
03  _Bool IsValid(void) {
04    char Password[12];
05
06    fgets(buffer, sizeof buffer, stdin);
07    if (buffer[ strlen(buffer) - 1] == '\n')
08      buffer[ strlen(buffer) - 1] = 0;
09    strcpy(Password, buffer);
10    return 0 == strcmp(Password, "goodpass");
11  }
12
13  int main(void) {
14    _Bool passStatus;
15
16    puts("Enter password:");
17    passStatus = IsValid();
18    if (!passStatus) {
19      puts("Access denied");
20      exit(-1);
21    }
22    else
23      puts("Access granted");
24    return 0;
25  }
```

3.3 Crypt Tool

Creating secure code Soap UI by Smartbear is a popular open source testing software for API's. Some its key features test for functionality and security on software. The key functions we will be utilizing is creating security test, generating scans, analyzing the results, and creating report from our results. This software can be used by both C++ and Java applications and is has the ability to adapt to specific software.

4 CONCLUSION AND FUTURE WORK

Currently we are in the process of building the two small applications with the enforcement of the security guidelines for each language. After the implementation of the two applications, they will be tested for vulnerabilities. Then, the given vulnerabilities will be logged to evaluate which areas will be concerns in nanorobotic application. We will look for different ways to limit the vulnerabilities and suggestions of language choice for nanorobotic applications.

ACKNOWLEDGMENTS

This work is partly supported by the National Science Foundation HBCU-UP ACE Implementation Award Number: HRD-1238838 NanoHU: Nanoscience Transforming STEM Education at Hampton University.

REFERENCES

[1] Dimov, Daniel (2014) Information Security of Nanorobots. General Security.

[2] Izzet Pembeci, Gregory Hager (2001) A Comparative Review of Robot Programming Languages. John Hopkins. 1-29.

[3] Long, Fred; Mohindra, Dhruv; Seacord, Robert; Sutherland, Dean; Svoboda, David (2014) Java Coding Guidelines.

[4] Piessens, Frank (2011). C and C++ vulnerability exploits and
Countermeasures.

[5] Software Engineering Institute, Carnegie Mellon University (2014). Confluence. Top 10 Secure Coding Practices.

[6] Stephen Turner (2014) Security vulnerabilities of the top ten programming languages: C, Java, C++, Objective-C, C#, PHP, Visual Basic, Python, Perl, and Ruby. Journal of Technology Research. 116, 1-16..