International Journal of Innovative Research in Computer Science & Technology (IJIRCST)
ISSN: 2347-5552, Volume-9, Issue-6, November 2021
https://doi.org/10.55524/ijircst.2021.9.6.60
Article ID IRPS120371, Pages 271-275
www.ijircst.org

Pseudo-Code Attack (PCA) in Software Engineering

Sushil Bhardwaj

RIMT University, Mandi Gobindgarh, Punjab, India
Correspondence should be addressed to Sushil Bhardwaj; sushilbhardwaj@rimt.ac.in

ABSTRACT- Software development has been more important in recent technological advancements in both hardware and software. The creation of scripting languages is critical to the development of software. The development of programming languages is applicable to software metric calculations such as line of code, code minimization, re-usability, and so on. Various attacks may be carried out during the metric computation phases in order to decrease the model parameters and delay the software package offering. Many attacks in networks have been discovered and avoided; this Pseudo-code assault is a new technique for boosting metric estimation in software development. This article suggested a method for introducing the Pseudo-code threat and detecting the presence of the Pseudo-code assault in the code base. Estimated size and duration are key quality assurance metrics in software development process. Software quality is measured by the average time between failures and the number of defects per line of code.

KEYWORDS- Engineering, Pseudo-Code Attack, Software, Software Development.

I. INTRODUCTION

Most software development organizations have a purpose or ambition to create safe software that ensures the reliability and availability of their products [1]. During software development, defects or faults are created either via the program's design or through its implementation. Failures, especially vulnerabilities, increase the cost to developers, forcing them to devote more effort to software maintenance rather than new features. The majority of software developers rely on testing to reduce their maintenance costs and provide software that is highly available and resilient [2-7]. Unfortunately, testing focuses mostly on verifying planned functionality rather than finding vulnerabilities.

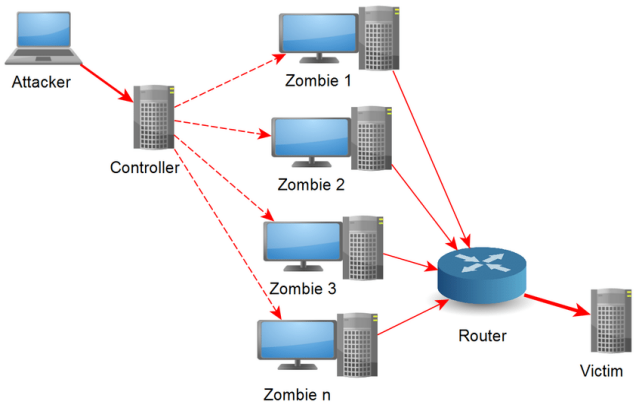

Figure 1: Illustrates the components of Distributed Denial of Service (DDoS) attack [8]. The attacker, via a controller, sends various zombies to victim's computer

In the network field, there are many Distributed Denial of Service (DDoS) assaults as shown in Fig. 1 above [8]. The new kind of DDoS assault is the Pseudo-code Attack (PCA). In the software development process, a Pseudo-code attack is conceivable. The Pseudo-code assault in DDOS is discussed in this article. These assaults aim to decrease system performance in order to produce a deterioration of system characteristics that are otherwise normal. DDOS hinders software development from being used as planned. Network-based DOS assaults are widespread, and they aim to deplete the system's resources. Source code that does not release a system resource on a regular basis may be investigated in a similar way, resulting in resource consumption. The software development cycle as shown in Fig. 2, is primarily concerned with the creation of fresh and novel software for the benefit of society. When the software team creates a new version, the revision is added to the programs based analysis. The number of read and write costs in the new version are calculated using this functional point analysis. It will also be compared against the previous version in order to determine the new version's quality. If the estimated cost is less than or equivalent to the previous version, the program will be released [9-13].

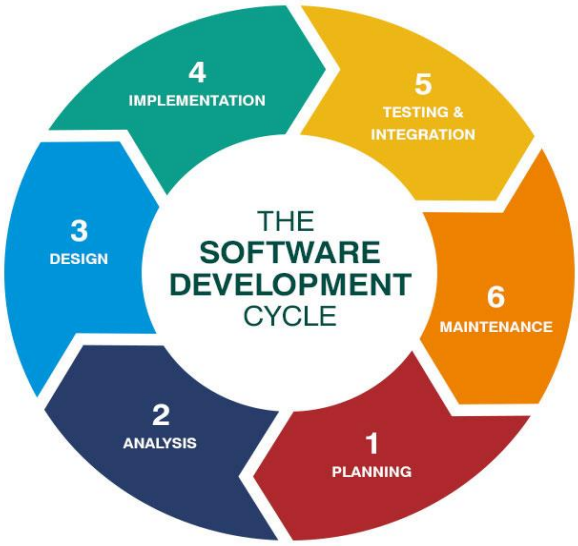

Figure 2: Illustrates the different phases in development lifecycle of software [14]

As a result, the Pseudo-code exploit has emerged as a new kind of DDoS assault. This attack is placed in the main memory automatically. The newly produced segment script will be inserted and relocated to the old code whenever the programs based analysis calls. As a consequence, the amount of read & write actions on memory increases. This procedure will result in the Full Function Points (FFP) being shown at a higher cost. As a result, the freshly created section is placed on hold until it is ready for release, and the development state evolves once again. As a result, software releases are delayed. This program code is not a malware or any other kind of virus. As a result, no antivirus detection program can identify it.

II. DISCUSSION

A. Consequences of Pseudo-Code Attack (PCA)

The assaults are stored in the memory space as Pseudo-code. It has a variety of effects on the system's performance. The system operations vary from their regular functioning process when a Pseudo-code assault occurs. The following are the consequences of a Pseudo-code assault. Because the PCA is placed in the system memory, it lowers server performance. As a result, the cost of full operational Points rises. The Pseudo-code consumes extra memory on the system. As a result, the disk storage capacity is enhanced. As a result, this assault slows down the system's disk drive and creates reading delays [15-19].

B. Issues and Challenges in Research

Designing software safety standards is complex and fraught with problems, which adds to the complexity of doing so [20]. The following are some of the current problems and challenges in software development:

1) Security is Always Evolving

The method for creating requirements should be accurate and straightforward. As a result, future security needs should include study into a new kind of software danger.

2) In A Positive Tone, the Security Needs should be State

Security mechanisms are often stated in a negative tone, making accuracy and reliability more challenging. A negative criterion, for example, might be "The software must not enable remote exploitation", which is difficult to verify. Validating and confirming that requirement may need testing what the program cannot do rather than what the system should accomplish.

3) Security is Continually Changing

The method for creating requirements should be accurate and transparent. As a result, future security needs should be researched to get expertise into a new kind of software danger.

4) The Safety Criteria for Creating Software Ought to be Agnostic of Operating System and Programming Language

When a method is linked with a certain language or platform, it becomes less viable. Fortunately, software specifications may be stated in a consistent tone to cover all potential scenarios that may arise during development. Certain of the needs will be less visible in some languages; for example, any program that hides system storage administration will not have criteria for failed memory allocation attempts, and so on.

5) For the Design Process to Function, the Security Criteria must be Verifiable and Tested

The basic concept behind prerequisites driven software development is to create requirements that may be created at each step of the design process. It may be verified or tested throughout the application design testing process. After the requirement analysis, assessment is done at each step to ensure that the criteria of each need are met to that stage in the history. Need testing occurs at the conclusion of the development cycle, mostly during test phase, and ensures that the requirement has been included. For it to be possible to monitor the progress of a security need throughout the design phase, and test to ensure the criteria was integrated into the software design, it must be simultaneously verifiable and testable.

6) Some Security Software Criteria, but Not All, May be Required for a Development

Anything from memory space to cryptography will be covered under the criteria [21]. However, certain services may just need a portion of those security criteria. There should be a process in place so that the privacy needs for a development may be chosen depending on the non-security requirements. Many problems arise during software development as a result of these types of assaults. A suitable model has to be created in order to design safe and scalable software. The study aims on creating a model depending on the Cosmic Full Function Points (CFFP) and energy point analysis to address these problems. It reduces the number of unnecessary read and write activities on the system. Using these methods, a susceptible DDoS assault may be avoided while the program is being developed.

C. Micro-Motivation

In the software design lifecycle, software sizing as well as energy points are used and the research problems along with obstacles associated with software scaling is also addressed.

1) Software Sizing

In order to provide a reliable software cost estimate, it is necessary to determine the size of the program [22]. As a result, choosing the best technique for estimating size is crucial. In most instances, the estimation risk is determined by exact size estimations rather than any other cost-based criterion. Given the inherent uncertainties in size estimate, it is critical that software sizing be done as precisely and consistently as feasible. However, software scaling is difficult for a variety of reasons. It is carried out in a variety of situations, some with extensive understanding of the system and others with little or no expertise. There are many options for the architecture and language that will be used to express the concept and specifications.

In most cases, software initiatives consist of a mix of repurposed, new, and enhanced techniques. Even when the change and reuse actually occur in the specifications and development rather than simply in the program code, a sizing approach must be able to incorporate all three stages. Measures for software sizing to create appropriate comparisons inside or between systems, software sizing metrics are used to standardize the other measurements. Without a size metric, productivity figures cannot be computed. Any software measuring application must have a size measure. The frequent usage of size measure during cost and schedule estimate is likely; nevertheless, there are numerous other potentially valuable uses, such as achieved value, vulnerability analysis, change management, and progress monitoring.

2) Energy Points

The majority of system programs and equipment developers are focused on addressing issues with the least amount of storage space and the fastest possible performance [23]. However, the usage of energy for processing is becoming more of a problem. The following are some of the measures that may be taken to reduce energy points in application development:

- Use shared servers to run numerous applications.
- Fewer logs.
- Compile interpreted languages.
- Delete historical data.
- Reduce the amount of data that is translated between components.

D. Challenges and Issues in Research

For a variety of reasons, software sizing and the software energy point process encounter many difficulties in obtaining an exact and appropriate estimate [24]. The estimate process in nature is not easy, especially with insubstantial goods, since software is insubstantial. One of the most challenging aspects of software sizing is obtaining the data necessary to validate the utility of any proposed models, metrics, or functional sizing methods. The majority of software scaling methods relied on tiny amounts of data. Some of the methods, for example, use 30 UML files to calculate the size and cost. As a consequence, the resulting method in this circumstance is unreliable and cannot be generalized. In the estimate process, there is a problem with maintaining the appropriate and correct dataset to test any sizing methods, measurements, or cost models.

As a consequence, there is a genuine issue with improving the software estimating process. The structure of the development cycle, in which all software needs are understood, as well as the problem of requirement creep, the connection between cost variables, and how each element may influence the software sizing outcome. Another issue is that there are no clear guidelines or standards for the entire application design process. It's now in the ad-hoc phase and is therefore not bound by any standards. Presently, the operational size estimation is done by hand, which takes a long time. Attempts to automate functional size measuring methods have been made in order to boost productivity and minimize the risk of human mistake. To complete the research gap analysis, it is essential to identify tool suppliers. As a result, the planned study cantered on creating a method for safe application development that is dependent on CFFP energy points.

E. Suggested Technique for Energy Point Calculation

There are two types of measures: direct and indirect. Price, line of coding, performance, memory capacity, and frequency of mistakes are all direct measurements. Function, reliability, sophistication, effectiveness, and dependability are examples of indirect measurements. These metrics are also impacted in the software building process by DoS attacks via micro-motivation. The energy point may be used to detect this kind of assault at the software project level. The functional point is used to determine the energy point. The functional point is critical in determining the software's quality. The developed program is divided into parts, with each module being used to calculate energy.

This study is primarily concerned with the transition from energy to a functioning state. In terms of functionality, designed subsystems are the primary suppliers. The entrance, exit, read, and write operations of the modules are identified using functional point assessment. The outcome is a numerical number that is used to calculate the energy point. The read and write energy in designed subsystems is calculated using energy points. These read and write energies are utilized to determine whether or not a pseudo-code DDoS assault is present in the programs. The energy point computation is analyzed using MCRose CFFP analysis. The modules are sent to MCRose in real-time, and the energy computations are based on the entrance, exit, and amount of read or write functionality.

1) Joulemeter

The Joulemeter calculates the energy consumption of a virtual machine, computer, or piece of application by monitoring the system resources like memory usage, CPU usage, disk usage, and so on, and translating the resource use to true power consumption using intelligently trained authentic energy models. In datacentres, client computing, and application development, a Joulemeter

may be utilized to get insight into energy consumption and to make various power allocation and provisioning choices.

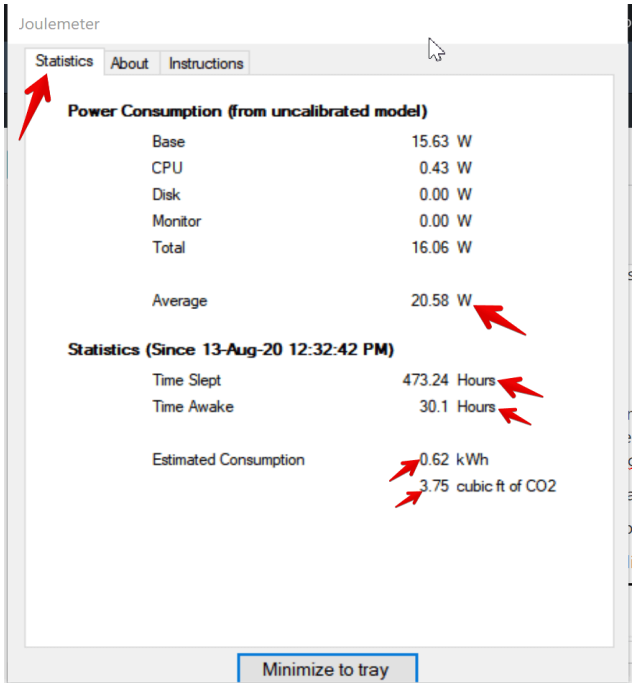

Figure 3: Illustrates the Joulemetre showing power consumption statistics [25]

IT managers who control power management settings, PC users who want fine-grained insight into their computing energy consumption, and hobbyist developers who want to utilize power measurer to optimize their application and hosted application architecture for energy use will all benefit from the technique. In a variety of situations, the Joulemeter modelling tool may be employed to reduce power use. The ability to monitor VM power enables power planning methods for virtual datacenters to be developed. Monitoring and analyzing PC sleep, coupled with remote awakening, enables corporate buildings to optimize desktop power usage. It is possible to separate the effect of physical components on battery life. Users may trade off power optimization parameters to extend battery life, and programmers can make suitable design trade for their program. The picture of the Joulemeter is shown in Fig. 3 above. The Joulemeter's read/write energy calculations may be adopted to determine whether or not there are any pseudo-coding assaults in the system.

III. CONCLUSION

Amongst the numerous risks to networking infrastructures, DDoS assaults have grown to represent a significant danger to the accessibility, availability, and functioning of several internet dependent services. According to recent studies, hundreds of similar assaults are attempted every week, inflicting significant harm to both government and commercial websites. A novel method for detecting and preventing DDoS is presented in this paper. Pseudo-code assaults, which may be detected with a Joulemeter, are also important in software development process to combat DDoS attacks.

REFERENCES

[1] Hussein M, Nouacer R, Radermacher A, Puccetti A, Gaston C, Rapin N. An end-to-end framework for safe software development. Microprocess Microsyst. 2018;
[2] Sawant AA, Bari PH, Chawan P. Software Testing Techniques and Strategies. J Eng Res Appl. 2012;
[3] Singh AP, Chandak S, Agarwal A, Malhotra A, Jain A, Khan AA. Utility of High-Resolution Sonography for Evaluation of Knee Joint Pathologies as a Screening Tool. J Diagnostic Med Sonogr. 2021;
[4] Mahat RK, Panda S, Rathore V, Swain S, Yadav L, Sah SP. The dynamics of inflammatory markers in coronavirus disease-2019 (COVID-19) patients: A systematic review and meta-analysis. Clinical Epidemiology and Global Health. 2021.
[5] Maini E, Venkateswarlu B, Maini B, Marwaha D. Machine learning–based heart disease prediction system for Indian population: An exploratory study done in South India. Med J Armed Forces India. 2021;
[6] Aliya S, Kaur H, Garg N, Rishika, Yeluri R. Clinical Measurement of Maximum Mouth Opening in Children Aged 6-12. J Clin Pediatr Dent. 2021;
[7] Matreja PS, Kaur J, Yadav L. Acceptability of the use of crossword puzzles as an assessment method in pharmacology. J Adv Med Educ Prof. 2021;
[8] Igbe O, Ajayi O, Saadawi T. Denial of service attack detection using dendritic cell algorithm. In: 2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference, UEMCON 2017. 2017.
[9] Iyer M, Tiwari S, Renu K, Pasha MY, Pandit S, Singh B, et al. Environmental survival of SARS-CoV-2 – A solid waste perspective. Environ Res. 2021;
[10] Gupta S, Mishra T, Varshney S, Kushawaha V, Khandelwal N, Rai P, et al. Coelogin ameliorates metabolic dyshomeostasis by regulating adipogenesis and enhancing energy expenditure in adipose tissue. Pharmacol Res. 2021;
[11] Prakash P, Radha, Kumar M, Pundir A, Puri S, Prakash S, et al. Documentation of commonly used ethnoveterinary medicines from wild plants of the high mountains in shimla district, himachal pradesh, india. Horticulturae. 2021;
[12] Catlos EJ, Perez TJ, Lovera OM, Dubey CS, Schmitt AK, Etzel TM. High-Resolution P-T-Time Paths Across Himalayan Faults Exposed Along the Bhagirathi Transect NW India: Implications for the Construction of the Himalayan Orogen and Ongoing Deformation. Geochemistry, Geophys Geosystems. 2020;
[13] Agarwal A, Agarwal S. Morbid Adherent Placenta Score: A Simple and Practical Approach on Application of Placenta Accreta Index. Journal of Ultrasound in Medicine. 2021.
[14] Malik S, Nigam C. A Comparative study of Different types of Models in Software Development Life Cycle. Int Res J Eng Technol. 2017;
[15] Agarwal A, Raj Singh M, Joon P. Sonourethrography With Pharmaco-Penile Doppler in Penile Fractures: A Complete and Productive Imaging Combination. J Diagnostic Med Sonogr. 2021;
[16] Agarwal S, Agarwal A, Chandak S. Role of placenta accreta index in prediction of morbidly adherent placenta: A reliability study. Ultrasound. 2021;
[17] Yadav A, Maini B, Gaur BK, Singh RR. Risk Factors for Serum Bilirubin Rebound After Stopping Phototherapy in Neonatal Hyperbilirubinemia. J Neonatol. 2021;
[18] Bishnoi S, Huda N, Islam SMU, Pant A, Agarwal S, Dholariya R. Association between psychological status and functional outcome in surgically managed fractures around hip in geriatric patients-a prospective study. Malaysian Orthop J. 2021;

[19] Wani AM, Rastogi R, Pratap V, Ashraf O, Neha. Comparative role of ultrasonography and magnetic resonance imaging in evaluation of biliary tract anomalies and pericholecystic adhesions in patients with gall bladder stone disease. J Int Med Sci Acad. 2021;
[20] Wong WE, Gidvani T, Lopez A, Gao R, Horn M. Evaluating software safety standards: A systematic review and comparison. In: Proceedings - 8th International Conference on Software Security and Reliability - Companion, SERE-C 2014. 2014.
[21] Vidas T, Larsen P, Okhravi H, Sadeghi AR. Changing the Game of Software Security. IEEE Security and Privacy. 2018.
[22] Wilkie FG, McChesney IR, Morrow P, Tuxworth C, Lester NG. The value of software sizing. Inf Softw Technol. 2011;
[23] Saini M, Kaur K. A review of open source software development life cycle models. Int J Softw Eng its Appl. 2014;
[24] Vatsa A, Kumar S. Software Production Issues and Mitigation Techniques: A review. Int J Recent Res Asp. 2016;
[25] Find PC Power Consumption.