CGC International Journal of Contemporary Technology and Research
ISSN: 2582-0486 (online) Vol.-3, Issue-2 DOI: 10.46860/cgcijctr.2021.06.31.205

Software Security: Role in SDLC

Shanky Goyal*, Navleen Kaur, SachinMajithia
Department of Information Technology,
Chandigarh Engineering College, Landran
Email Id: *shanky.it@cgc.edu.in

Abstract: **This research emphasizes mainly on the need for software security. Softwares are developing at a faster pace so it is required to impose security on them in order to secure them from cybercrimes. Softwares have been facing problems by the attackers who are constantly kept on breaching the Data. Therefore this survey comprises the phases that are an integral part of the SDLC from the security point of views such as Design and testing phase. Moreover, it quests upon the data related to threats and attacks. Not only this, but it also involves the prerequisites that have to be determined before developing the software like, what are the approaches that should be followed and what are the best suitable designs to secure the software?**
.

Indexed Terms-Software, Security, threats. (Keywords)

I. INTRODUCTION

Instructions that advise the computer how to function. Likewise, software is a bunch of data, instructions, or programs used to make a computer work and execute explicit tasks. Software security can represent the moment of truth for the whole organizations nowadays. So how can we better secure our system?

The answer to this inquiry is on higher priority than at any other time. When an organization disregards security issues[1], they expose themselves to risk. This problem takes place because of the vulnerabilities that are present in our software. However, these vulnerabilities are generated during the development of software. An adversary exploits these vulnerabilities through several attacks. This outcome in the breaking of delicate information that is put away in business software. These days, Governments are enacting and authorizing data protection measures. For instance, the European Union's GDPR expects organizations to incorporate information security shields at the earliest phases of development.

According to Statistics which reports on the number of Data Breaches, approximately 3,809,448 records are stolen from every day. On November 2, 1998, the Morris Worm was presented as the first computer worm that circulates by means of the internet, and was the first to acquire huge media attention. It additionally brought about the main lawful offense conviction in the US under the 1986 Computer Fraud.

Another worm named as Code Red was discovered on the web on 15 July, 2001. The computers running Microsoft's IIS internet server were attacked. It absolutely was the primary massive scale, mixed threat attack to target enterprise networks successfully.

There are thousands and millions of softwares[2] that are working on a daily basis for providing services to customers. For example Medical Softwares that maintain the "Electronic health records" of patients are so important to be secured to prevent cybercriminals from stealing their records. The data of these medical softwares are also provided to the Insurance companies in order to facilitate the patients financially. But the personal information of patients is also in danger over this network as cybercriminals can use this data for their monetary gain. Because we know, online credentials are so valuable these days.

You may think the preferred customer credits you have acquired are not important to cybercriminals. Reconsider it again. After around 10,000 American Airlines and United accounts were hacked, cybercriminals booked free flights and upgrades utilizing these taken credentials. Despite the fact that the preferred customer credits were given back to the clients by the airlines, this exhibits the worth of login credentials.

Therefore, Building Secure Software's [3] is as paramount as writing efficient code along with quality algorithms. For this to be done there is a satisfactory solution that gives an organized way to deal with software security-the secure development lifecycle (SDL).

We aimed (1) to analyse the security vulnerabilities; (2) to survey the various attacks; (3) to assess how these factors could be managed to protect the software.

II. DESIGN PHASE

There are various phases of SDLC –Requirement gathering, Design, coding, testing and maintains as shown in fig 1.

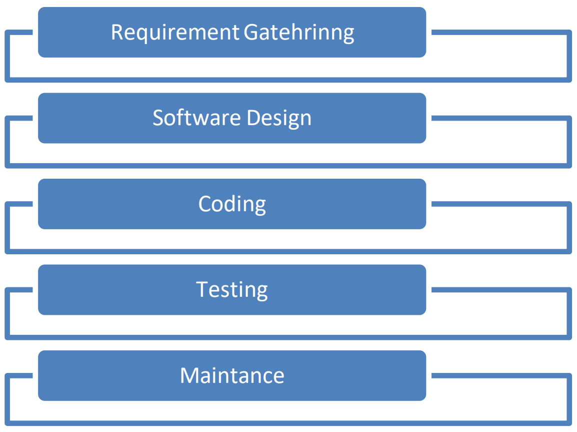

Fig 1: SDLC Life Cycle

The purpose of design stage is to design software that meets the security prerequisites in order to prevent our system from Forgery. Along with the requirements certain types of vulnerabilities need to be kept in mind that arises from coding problems.The coding problems featured in this phase are: buffer overflow [4] (Stack and heap overflow) and format String vulnerabilities. However, a secure design also plays a vital role to control attributes such as Confidentiality, Integrity, and availability. To accomplish this task, later stage security engineering efforts are made that introduce us with UML (Unified Modelling Language).

This stage design focuses mainly on the development of the physical solution itself.

2.1 Security Prerequisites

To meet the objective of security, following are the security preconditions that should be achieved:

2.1.1. Identification requirements:

Identification requirement is a security necessity that indicates the extent to identify its externals before interacting with them. These requirements includes

i. Who you say you are(name, user identifier)
ii. What you have(digital certificates, an employee card, a hardware key)

2.1.2. Authentication requirements:

Authentication follows identification, if identity is adequately significant to indicate, then so is authentication.

The typical objective of the authentication is to identify who you are. The user has to enter his credentials like username and password to allow the server to set up the session to provide the information. After ensuring the identity server authenticates the client for further process.

Due to the close connection betwixt identification and authentication, they are once in a while grouped together in security requirements.

2.1.3. Authorization requirements:

Authorization depends on both identification and authentication. It is a process done by a server to determine whether an authenticated user has a permission to elicit a file or not, if not, the user will get a pop-up showing a message "YOU ARE NOT AUTHORIZED TO ACCESS THIS PAGE".

Thereby prevents unauthorized users from obtaining access to confidential data or call for the permission to access restricted services.

These security specifications depend on distinguishing an individual, confirming that the individual is who or what it professes to be, and approving the access level and set of activities related with the username and IP addresses.

Moreover, there are various security policies implemented to preserve the CIA triads.One of the most common security policies is IPSEC. IPSEC is a standard set-up of conventions between two correspondence closes across the IP network that gives information validation, integrity and confidentiality.

Simultaneously, the IKE (Internet Key Exchange) protocol is utilized to deal with the cryptographic keys utilized by hosts for IPSEC. Hence, these are the essential Security Specifications that needs to be taken care of before designing any software.

2.2 Security Vulnerabilities

Therefore Security requirements depend not just on correctness since implementation defects are likewise there that produce security vulnerabilities [5].

These security vulnerabilities arise from not using the type-safe languages. Hence it is required, not to utilize that type-unsafe language, for example, C that outcomes in boundary defects that can be exploited to run malicious scripts. Following are the problems that are created due to not using the secure coding standards:

2.2.1. Buffer overflow:

Buffer Overflow is a portmanteau of Buffer that is continuous memory associated with a variable and field and overflow is to put more into the buffer than it can hold. In programming, Buffer overflow is actually an oddity in a program where a program, while writing data into a buffer overruns the buffer's boundary and overwrites neighbouring memory areas. Also buffer overflow occurs when data written to a buffer [8] also corrupts data values in memory addresses adjacent to the destination buffer due to insufficient bounds checking. This can happen when replicating data starting with one buffer then onto the next without first watching that the data fits within the destination buffer.

Buffer overflows are used to:

i. Overwrite local variables
ii. Crash the program
iii. Inject malicious code into program

For example: Char buffer can store only upto four variables, but user has entered text containing 7 characters including Null terminator. So in that case strcpy command overwrites the frame pointer causing the segmentation fault.

Therefore generally in these cases it is recommended to use security relevant features in the program such as introduce another local variable to avoid the problem of overwriting the frame pointer. However still the problem is not solved because every time that local variable is going to return out the address of Null Terminator.

To avert the buffer overflow from occuring, the call to strcpy could be supplanted with strlcpy, which takes the maximum capacity of A (including a null-termination character) as an extra parameter. When accessible, the strlcpy library function is preferred over strncpy which does not null-terminate the destination buffer if the source string's length is more noteworthy than or equivalent to the size of the buffer (the third argument passed to the function), now just look at the idea of how an attacker injects his malicious code.

Following are the two steps:
1.Load malicious code into the program.
2.Somehow get %eip (Instruction pointer) to point to the infected code.

The main goal of the adversary is to use a general-purpose shell. It is a command line prompt that gives an attacker general admittance to the system. Moreover, an attacker takes advantage of the return address by simply pushing the address of his own malicious code.

The most genuine approach to stay away from or forestall buffer overflow is to utilize automatic protection at the language level. However some techniques also exist for C to avoid buffer overflow.

Below given are few points to solve this problem:

i. Use those languages that are specifically typed and don't permit direct memory access, like COBOL, Java, Python.
ii. Many programming languages other than C/C++ give runtime checking and at times even arrange compile-time checking which may send an admonition or raise an exemption when C or C++ would overwrite data.

Every interpreted language will protect against buffer overflows, signalling a well-defined error condition. As a result, Buffer overflows must thus be avoided by maintaining a high degree of correctness in code which performs buffer management.Just like buffer overflow, another vulnerability "FORMAT STRING" has also been found in the type-unsafe languages.

2.2.2. FORMAT STRING VULNERABILITY:

Format strings are utilized in many programming languages to embed values into a text string. At times, this mechanism can be mishandled to perform buffer overflow attacks, extricate information or execute arbitrary code.

Consider, C's printf family supports Formatted I/O[6].

```
Void print(int age, char* name)
{
printf("Name: %s\t age: %d\n", name,age);
}
```

Format strings are typically the first or one of the first arguments to the Printf style function. Moreover Format Strings use the format specifiers that how the data should look like.

Format specifiers depict:

- Position in string indicates stack arguments to print.
- Kinds of specifiers indicate types of arguments.

For example: Consider two functions

```
Void safe()
{
printf(buf);
}

Void vulnerable()
{
printf("%s", buf);
}
```

Here the problem arises when buf contains format specifiers, they will be interpreted by printf but when buf is used as the format string it could be potentially exploited. We have seen this vulnerability in light of the indiscreet utilization of the core format string functions that are utilized in C[7]. These format string functions are open to the various attacks. Hence, the better method to banish this issue is to appropriately approve user input or to all the more likely stay away from the passing user controlled inputs to function at every possible opportunity.

So, it is recommended not to use printf without the format parameters.To resolve these kinds of complications throughout the software design, several Coding standards were adopted that consists of rules and recommendations to provide normative requirements for code[8], while on the other hand recommendations are meant to provide guidance that, when followed, should improve the safety, reliability, and security of software systems. Softwares that follow standard's guidelines will lead to higher-quality-softwares and robustness that are more resistant to attacks.

2.3 Later Stage Security Engineering-

UMLsec- It is one of the foremost model-driven security engineering approaches. UMLsec broadens UML specification with an UML profile that gives conventional images to be utilized in explaining system design elements.

Also UML provides basic security factors such as Confidentiality and Integrity that together constitute CIA triad.

It likewise permits considering diverse threat situations relying upon adversary strengths.UML also provides symmetric encryption.

Secure UML gives UML-based language for modelling role- based access control(Access control is a security strategy that manages who or what can view or utilize resources in a computing environment) policies and authorization constraints of the model-driven engineering approaches. This methodology is still firmly combined with system design models. Secure UML characterizes a bunch of vocabulary that addresses RBAC concepts like roles, role permissions, and user-assigned roles.

This model driven methodology performs identification, authentication and authorization of clients and substances by assessing required login qualifications that can incorporate passwords, individual ID numbers. It is a significant part of layered defence to ensure access control system. Every one of these things can be portrayed with the assistance of UML conventional images.That is the reason applications are created from high level UML models.

UML gives norms for software development Likewise UML has huge visual components to build and they are not difficult to follow.

III. TESTING PHASE

This phase is categorized into two parts, Security Threats and Security tools".So far,we have been discussing the architecture of software. Now it's time to discuss one of the most important phases of SDL (Software development lifecycle) and that is software security[9] testing. The sole purpose of testing is to check behaviour and performance of software.However, based on this scheme, testing is diversely classified into various categories such as Manual and Automated testing. Not only this, itis also classified as Functional and Non-Functional testing.

Given below are the levels of testing:

i. Unit Testing
ii. Integration Testing
iii. System Testing
iv. Acceptance Testing

But all of these testings are based on the working of software and here we are looking for the security of software. Therefore to check all the vital attributes of security are:

i. Confidentiality
ii. Integrity
iii. Availability
iv. Non-repudiation

We really need to dive deeper searching for the tests and tools that are important to maintain these attributes.Likewise the foremost intention of security testing is to discover how vulnerable a system might be and to decide if its data and resources are shielded from possible intruders.

3.1 Security Threats-

Here are the various sorts of threats which can be utilized to take advantage of software vulnerability:

3.1.1 SQL injection-

It is the most well-known kind of attack that is executed by attackers to infuse malicious sql queries into the entrance field for execution. It is a sort of attack which exploits escape clauses present in the execution of web applications that permits a hacker to hack the system.This results in fetching the personal information of the users from the Server database and hence violates the Confidentiality. To prevent web applications from SQL injections10], Data inputs must be sanitized.Fig1 is showing the number of vulnerabilities meeting specified limitations year wise.

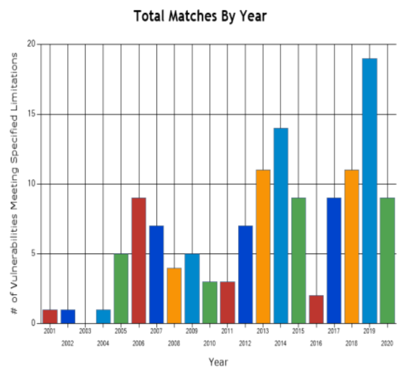

Fig 2. Number of vulnerabilities meeting specified limitations year wise

3.1.2 Denial of service-

Denial-of-service (DOS) is an attack strives by hackers to make all the services unavailable for the legitimate users.

DOS attacks normally fall into two classifications:

i. Buffer Overflow attacks [11]
ii. Flood attacks

Many major companies have been the focus of DOS attacks. This threat simply violates the availability attribute of the CIA triad. Various Vulnerability IDs have been registered in National vulnerability Database One of which is (NVD - CVE-2020-3257)

Multiple vulnerabilities in the Cisco IOx application environment of Cisco 809 and 829 Industrial Integrated Services Routers (Industrial ISRs) and Cisco 1000 Series Connected Grid Routers (CGR1000) that are running Cisco IOS Software could permit an attacker to cause a denial of service (DOS) condition or execute arbitrary code with raised advantages on an affected device.

3.1.3 Identity spoofing

With regards to information security, Identity spoofing is an attack wherein an attacker utilizes the authenticated credentials of a real user to break into the system. A considerable lot of the protocols in the TCP/IP suite don't provide mechanisms for authenticating the source or destination of a message, and are in this manner helpless against spoofing attacks when additional safety measures are not taken by applications to confirm the identity of the sending or accepting host.One of the data Depicts by (NVD - CVE-2002-1183)Windows 98 and Windows NT 4.0 don't as expected check the Basic Constraints of digital certificates, permitting far off attackers to execute code, otherwise known as "New Variant of Certificate Validation Flaw Could Enable Identity spoofing" (CAN-2002-0862).

3.1.4 Password cracking

Password cracking is the main part while doing software security testing.In request to get to the private spaces of an application, hackers can utilize a password cracking tool or can figure a typical username/password. Different password cracking attacks are also there. For example Brute force attack, Dictionary attack. All these attacks are possible because of the simple usernames and passwords. Non-Complex passwords are easily available online because cracking of passwords with the help of password cracking tools becomes easy for attackers.

An issue was found on V-Zug Combi-Steam MSLQ devices before Ethernet R07 and before WLAN R05. Password authentication utilizes MD5 to hash passwords (NVD - CVE-2019-17216). In this way, Cracking is conceivable with minimal effort.

IV. MALWARE ATTACK

Malware is an attack in which hackers launch malicious code into the software [12].Malicious code is a code that cyber attackers create to obtain entrance or cause harm to a computer or network, normally without the victim's knowledge. This results in giving them unauthorized access into the system. Malware is frequently hard to recognize and devices are commonly contaminated without the user in any event, taking note.Malware attacks can happen on a wide range of devices and operating systems, including Microsoft Windows, macOS, Android, and iOS. It has been seen that TCP function included in the firmware of Mitsubishi Electric MELQIC IU1 series IU1-1M20-D firmware version 1.0.7 and prior permits remote attackers to sidestep access limitation and to stop the network functions or execute malware by means of a uniquely created packet. (NVD - CVE-2020-5545).

There are many types of attacks present in cyberspace. Some attacks are used for data leak, some are used to track user systems and others are just for monetary gain like in the past, one malware was detected named as ransomware. It attacks on a data storage of company and encrypts all the company's data and demands a huge amount of money. Following are the different types of malwares:

4.1. Virus

A virus is a malicious executable code that joins itself to a reasonable program to contaminate it. The virus works when the genuine program runs and can perform any function, for example, erasing a document. The premier activity acted in virus is that when a infected program is executed, it will initially execute the virus followed by the execution of native program code. Subsequent to corrupting all the files from the current user's computer, the virus engenders and sends its code through the network to the users whose email address is put away in the current user's computer. Most viruses require end-user initiation and can actuate at a particular time or date. Viruses can be innocuous and essentially show an image or they can be damaging, for example, those that alter the data. Viruses can be spread through email and instant message connections, Web document downloads, and web-based media scam links. Symptoms of having virus in a system are:

i. Frequent pop-up windows.
ii. Changes to our homepage.
iii. Frequent crashes.
iv. Unusually sluggish computer performance.
v. Unusual activities like password changes.

The purpose of the virus is to modify the information. In 1949, viruses were firstly recognized in the publication of "Theory of self-reproducing automata" by the great professor John von Neumann. Various well-known viruses are Boot sector virus, Direct action virus, Browser hijacker and many more.

4.2. Worms

A worm is a malignant code that can recreate itself and send duplicates from one computer to another like a virus, however it is distinctive in execution. It doesn't change a program rather it is enacted upon arrival to replicate and propagate once more. The unreasonable replication brings about halting the system; it gobbles system resources to bring it down. A worm vivaciously looks for additional machines to ruin, and the defiled machine acts as a worm delivering machine for different machines associated with it.Other than the underlying infection, they presently don't need user support. Worms share similar patterns.

They all have an empowering weakness, an approach to engender themselves, and they all contain a payload.Worms usually slow down networks. As previously mentioned in the introduction, the first worm was released by the Computer Science student named Robert Morris onto the Internet from the Massachusetts Institute of Technology. Various kinds of worms are Internet worms, Email worms, File-Sharing worms and IRC worms. Symptoms shown by the presence of computer worms on a system are:

i. Programs opening and running automatically.
ii. Irregular Web Browser performance.
iii. Firewall warnings.
iv. Modified files.

4.3. Spyware

"Spyware runs quietly in the background, collecting information". As the phrase itself shows the definition of spyware. This malware is intended to track and keep an eye on the user.Spyware is unwanted software that spies on a computer system to steal the personal information of a user. It accumulates all the personal information and transfers it to publicists, data firms or outside clients for his financial increase. While trying to defeat safety efforts, spyware frequently changes security settings. Spyware is utilized for some reasons. It incorporates activity trackers, keystroke collection, and data capture.Cybercrime statistics depicts:

i. A complete of 978 million individuals in 20 nations were influenced by cybercrime in 2017, as indicated by Norton Cyber Security Insights Report Global Results.
ii. Victims of cybercrime universally lost $172 billion.

Spyware frequently packages itself with genuine software or with Trojan horses. Spyware is the simplest danger that is possible to taint a device and it tends to be difficult to recognize.

4.4. Trojan Horses

"The term is derived from the Ancient Greek story of the beguiling deception that prompted the fall of the city of Troy". A Trojan horse is a malignant code that looks real yet can grab hold of your computer. This malignant code exploits the advantages of the user that runs it. A Trojan is intended to harm or to do any criminal behaviour on a network. A Trojan can go about as an ensured application to trap a user and afterward tries to bamboozle a user into stacking and executing the malware on a device. Trojans are found in image files, audio files or games. A Trojan horse differs from a virus since it ties itself to non-executable records. User might think that he has received an email from his friend or a colleague and click on what looks like a legitimate file. However, this isn't the situation, this email is shipped off him by an attacker and the file that he tapped on-and downloaded and opened has gone onto introduce malware on your device. At the point when a user executes the program, the malware can spread to different records and harms the system. This is how cybercriminals infiltrate the systems and attack on them. There are different sorts of Trojan malwares like Backdoor Trojan, Downloader Trojan, Fake AV Trojan, Info stealerTrojan and so on. This is a never-ending list as common types of Trojans can be listed out from alphabet A to Z.

4.5. Ransomware

Ransomware is a type of attack that is usually done with the purpose of monetary gainIn other words, it is defined as a kind of malware that keeps users from getting to their system and requests deliver instalment to recapture access.One of the three businesses in the UK faces this kind of malware where they have to pay the demanded amount to the team of hackers.The most common method to inject malware into the system is through malicious spam. These malicious spams use social engineering to trick the people in order to gain access to their systems. There are three main types of ransomware:

i. Scareware
ii. Screen lockers
iii. Encrypting ransomware

Based on the above threats and attacks, security testing cannot be done manually, Therefore we have to identify the automated tools to execute all security test cases.

V. SECURITY TOOLS

There are several methods that are especially designed to help in securing the software and many a times, they help to track the data loss and the attacker's information[13]. In today's time, software security is a major concern for companies. Therefore following are the various software security tools:

5.1. Penetration Testing Tool

Penetration testing tools are utilized as a component of penetration test (Pen Test) to computerize certain undertakings, improve testing proficiency and find gives that may be hard to track down utilizing manual analysis strategies alone.Penetration testing is a good technique of accessing security by actively looking for the exploitable vulnerabilities. It is usually done by Red Teams, Tiger Teams as pen testing is a black hat activity. Penetration testing can be applied at various levels of granularity such as

i. A single process
ii. The complete web application
iii. Network of many applications

Pen testers utilize creativity and mechanized tools to quickly investigate a system attack surface. As indicated by the 1967 Ware report, a Team of specialists headed by Willis Product of RAND corp.formally assessed the security problem for time-sharing computer systems. They used the term "Penetration". Two frequent penetration testing tools are static analysis tools and dynamic analysis tools.

5.2. Unit Testing Tool

Unit testing tools are the tools that analyse the source code whether they are free from flaws or not[14]. The main purpose of doing unit testing is to determine whether the services at the micro level are free from defects. By scanning smaller bits of code earlier in the process, unit testing tools enable developers to find flaws faster, fix them more easily. Hence unit testing can streamline application security.

5.3. Vulnerability Scanner

A vulnerability scanner is a computer program intended to evaluate computers, networks or applications for known shortcomings. Vulnerability checking offers an approach to discover application indirect accesses, malicious code and different dangers that may exist in software brought or applications developed internally. Nowadays vulnerability scanners allow for both authenticated as well as non-authenticated scans. The advanced vulnerability scanner frequently customize vulnerability reports just as the installed software, open ports, certificates and other host information that can be questioned as part of work process.

5.4. Fuzz Testing

Fuzz testing is a software security technique that involves the random data as inputs to the program. It is a kind of random testing. The goal of Fuzz Testing is to make sure that wrong things don’t happen. This test is usually carried out to avoid the crashing of software. Moreover, thrown exceptions, non-termination are the things that can be the foundation of security vulnerabilities. Fuzz testing complements the functional testing in various ways as it tests the features directly. Following are the kinds of Fuzz Testing:

i. Black-box(This tool knows nothing about the program or its input)
ii. Grammar-based(This tool generates input informed by a grammar)
iii. White-box (This tool produces new contributions at least partially informed by the code of the program being fuzzed)

VI. CONCLUSION

This paper is written with a purpose of knowing about the Importance of Software Security. It involves the Overview of Software security and what are the phases in the SDL that affects the security during the Development of software. Moreover this research includes the findings of various threats that are increasingly growing day by day and are responsible for harming the security of software. Along with this ,It tells us how to conquer these attacks by using the security tools.

REFERENCES

[1]. N. R. Mead and G. McGraw, "A portal for software security," in IEEE Security & Privacy, vol. 3, no. 4, pp. 75-79, July-Aug. 2005, doi: 10.1109/MSP.2005.88.

[2]. G. McGraw, "Managing software security risks," in Computer, vol. 35, no. 4, pp. 99-101, March 2002, doi: 10.1109/MC.2002.993782

[3]. L. Williams, A. Meneely and G. Shipley, "Protection Poker: The New Software Security "Game";," in IEEE Security & Privacy, vol. 8, no. 3, pp. 14-20, May-June 2010, doi: 10.1109/MSP.2010.58.

[4]. G. McGraw, "Software security," in IEEE Security & Privacy, vol. 2, no. 2, pp. 80-83, March-April 2004, doi: 10.1109/MSECP.2004.1281254.

[5]. J. Wilander and M. Kamkar. A comparison of publicly available tools for dynamic buffer overflow prevention. In Proceedings of the Network and Distributed System Security Symposium, pages 149--162, February 2003.

[6]. L. O. Andersen. Program analysis and specialization for the C programming language. PhD thesis, University of Copenhagen, 1994

[7]. R. Jones and P. Kelly. Backwards-compatible bounds checking for arrays and pointers in C programs. In Proceedings of the International Workshop on Automatic Debugging, pages 13--26, May 1997

[8]. Xiangyu Zhang and R. Gupta, "Hiding program slices for software security," International Symposium on Code Generation and Optimization, 2003. CGO 2003., San Francisco, CA, USA, 2003, pp. 325-336, doi: 10.1109/CGO.2003.1191556

[9]. B. Potter and G. McGraw, "Software security testing," in IEEE Security & Privacy, vol. 2, no. 5, pp. 81-85, Sept.-Oct. 2004, doi: 10.1109/MSP.2004.84.

[10]. J. Epstein, S. Matsumoto and G. McGraw, "Software security and SOA: danger, Will Robinson!," in IEEE Security & Privacy, vol. 4, no. 1, pp. 80-83, Jan.-Feb. 2006, doi: 10.1109/MSP.2006.23.

[11]. S. Chen, J. Xu, E. C. Sezer, P. Gauriar, and R. K. Iyer. Non-Control-Data Attacks Are Realistic Threats. In the Proceedings of the 14th USENIX Security Symposium, Baltimore, MD, Aug. 2005.

[12]. T. Pietraszek and C. V. Berghe. Defending against Injection Attacks through Context-Sensitive String Evaluation. In the Proceedings of the Recent Advances in Intrusion Detection Symposium, Seattle, WA, Sept. 2005.

[13]. S. Barnum and G. McGraw, "Knowledge for software security," in IEEE Security & Privacy, vol. 3, no. 2, pp. 74-78, March-April 2005, doi: 10.1109/MSP.2005.45.

[14]. Erlingsson Ú. (2007) Low-Level Software Security: Attacks and Defenses. In: Aldini A., Gorrieri R. (eds) Foundations of Security Analysis and Design IV. FOSAD 2007, FOSAD 2006. Lecture Notes in Computer Science, vol 4677. Springer, Berlin, Heidelberg.