Computer Science and Information Technologies
Vol. 1, No. 3, November 2020, pp. 98~105
ISSN: 2722-3221, DOI: 10.11591/csit.v1i3.p98-105

Modification data attack inside computer systems: A critical review

Vahid Kaviani J, Parvin Ahmadi Doval Amiri, Farsad Zamani Brujeni, Nima Akhlaghi
Department of Engineering, Islamic Azad University, Isfahan (Khorasgan) Branch, Isfahan, Iran

Article Info

Article history:

Received Jan 23, 2020
Revised May 28, 2020
Accepted Jun 16, 2020

Keywords:

Computer security
Control data attack
Memory security
Modification
Non-control data attack

ABSTRACT

This paper is a review of types of modification data attack based on computer systems and it explores the vulnerabilities and mitigations. Altering information is a kind of cyber-attack during which intruders interfere, catch, alter, take, or erase critical data on the personal computers (PCs) and applications through using network exploit or by running malicious executable codes on victim's system. One of the most difficult and trendy areas in information security is to protect the sensitive information and secure devices from any kind of threats. Latest advancements in information technology in the field of information security reveal huge amount of budget funded for and spent on developing and addressing security threats to mitigate them. This helps in a variety of settings such as military, business, science, and entertainment. Considering all concerns, the security issues almost always come at first as the most critical concerns in the modern time. As a matter of fact, there is no ultimate security solution; although recent developments in security analysis are finding daily vulnerabilities, there are many motivations to spend billions of dollars to ensure there are vulnerabilities waiting for any kind of breach or exploit to penetrate into the systems and networks and achieve particular interests. In terms of modifying data and information, from old-fashioned attacks to recent cyber ones, all of the attacks are using the same signature: either controlling data streams to easily breach system protections or using non-control-data attack approaches. Both methods can damage applications which work on decision-making data, user input data, configuration data, or user identity data to a large extent. In this review paper, we have tried to express trends of vulnerabilities in the network protocols' applications.

Corresponding Author:

Vahid Kaviani J,
Department of Engineering,
Islamic Azad University, Isfahan (Khorasgan) Branch, Isfahan, Iran.
Email: kaviani@khuisf.ac.ir

1. INTRODUCTION

By the rise in the number of computers and networks, and upon seeking more security and assurance simultaneously, the area of security has become both more interesting and challenging. In fact, aggressors attempt to access delicate basic resources to exploit them. As with numerous inspirations, there are a lot of news broadcasts concerning abuse of data and attacks on systems all over the globe. Although a lot of researches and studies have been launched to secure networks and successfully prevent a large number of attacks, there are many varieties of attacks, most of which are still new and open to further studies. In this survey paper, an attempt has been made to elaborate more on the methods and tools attackers use to modify information and data.

2. DEFINITION OF ATTACK

In terms of definition, an attacker is the person who can maliciously intercept, interrupt, sniff, alter, steal, or remove important data inside a computer or application by breaching the network or through breaching to the system directly, like running an executable code on the target computer. Data modification has different types. Intruders misuse different resources of victims to reach their goals. Specifically, they exploit software vulnerabilities or hardware weaknesses to penetrate the system [1]. In this survey, we have tried to show the trends of modification data attack. In the following review, the manner in which these kinds of attacks will take place and their countermeasures are explained.

2.1. Definition of problem (data modification attack)

Generally, most of the intruders know that there is a breach, or better to say, insecure application on some personal computers (PCs). They can misuse insecure software and systems to do the modification. This approach is called control-based attack, in which an intruder misuses a memory flaw, such as a buffer overflow or use-after-free, to overwrite control-data such as a return address or function pointer and thereby modifies the control-flow of the program. In order to get the control of an application, which is referred to as hijacking [2], primarily it is necessary to inject specific data which can be run to get the control of system. This method is known, in the cyber world, as control-data attack. Another type of modification data attack results in the run of computation which is known as non-control-data attack. This type of attack, which injects wrong data in system call, needs to inject invalid code by corrupting the data or using a valid code with invalid data entry or through entering an invalid path by corruption. In order to detect those attacks, anomaly-based analysis is used in intrusion detection systems (IDS) to look for any flaw from a pattern which shows irregularity from normal behavior of the programs [3].

Most of the security systems launch this kind of detection in local systems and build their patterns using sequences of system calls. This approach can detect various control-data attacks; however, most of the non-control-data ones evade that procedure [4]. Data modification is the way known by control-data-attackers which alters the flow of programs. That means it modifies user characteristics, configuration, and user input data to achieve attacker's goals [5]. Stated that non-control data attacks are of a serious nature. They can be a threat against many real applications. Moreover, server side applications widely include the infrastructure needed for this type of attack. Since the recent countermeasures, which have been developed against control-based attacks, have succeeded, the expansion rate of non-control data attacks has been increased dramatically [6].

In recent years, a lot of experts and scientists have been addressed to find out the way to mitigate control-based-attacks. One of the most significant researches in this scope, which is a part of governmental afford to develop a formal model of for assessing integrity of control-flow, used this model to find the correctness of defenses against an attacker [7]. Chen *et al*. [5] indicated that those countermeasures which address control-based-attacks will not prevent non-control-data attacks. Those intruders who are using non-control approach can evade the defense system. For example, those who use mimicry attacks can evade the security mechanism and such kind of attack can potentially bypass security. Several improvements of this approach have been offered, notably by adding information available at the system level, such as the parameters of the system calls or their execution context. The detection of control-data attacks is enhanced in not only accuracy but also completeness, however, non-control-data attacks remain mostly undetected [8].

Most of modifications in memory data flow attacks [9, 10] which are emerged by penetrations and worms, are known as the control-data attack. With recent advancement in security areas, exploiting network and computer systems is becoming more tough; hence, it is essential for intruders to get enough information form victims' devices and applications, and they must be equipped with powerful tools and enough expertise to run successful cyber-attacks against sensitive data and network infrastructure. Intruders mostly exploit the computer systems by breaching the security vulnerabilities such as integer overflow, format string vulnerability, and buffer overflow and low-level memory corruption flaws. These vulnerabilities are not the only approach. There are other types of vulnerability such as malwares, zombie bots and rootkits which can cause denial of service (DDoS) attack [11].

There is much more detail about these trends of attacks which is included in these papers [12, 13] and can broadly be found on the Internet. Another sort of modification attack, which causes breach by memory corruption, uses a similar pattern that is known as control-data-attack. It alters the stream of instruction of a program by means of changing registry counter while facing program error. This causes a change in the addressing of flow program inside the processor. All these could happen through launching malicious program which contains harmful codes. The attacks mostly occur in system calls or etc. A quick study of the United States Computer Emergency Response Team (US-CERT), shown in Figure 1, suggests security policies in [14, 15] and the Microsoft Assurance Report in [16] shows that control-data attack is an important vulnerability and should be classified as critical danger.

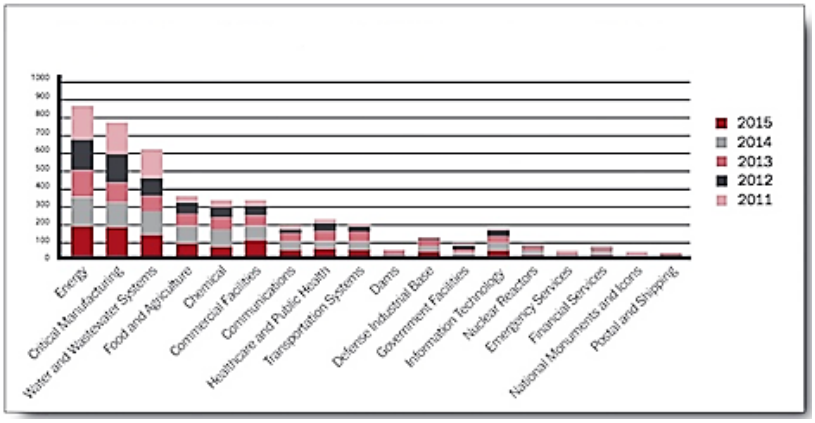

Figure 1. Number of ICS-CERT reported vulnerabilities by sector

3. RELATED WORKS

Schlesinger *et al.* used special hardware architecture known as the Minos. It can prevent those attacks that are caused by memory corruption and hijack in systems like control data attacks and some of non-control data ones [17]. Moreover, in order to mitigate control data attacks, which can inject malicious codes to change the processor counter, it protects memory from any kind of violation of safety. The Minos checks the integrity of all data in memory. Another related work in [18] paper which is related to the prevention of both non-control data and control data attacks is known as YARRA. YARRA is another extension of C++ compiler which protects important unsafe components, sensitive data type and other vulnerability in interpretation between higher level languages to lower level. It is also proved that this project can successfully assess and validate the results of implementations in real environment systems which were vulnerable to a variety of memory corruption attacks.

4. CURRENT TRENDS IN NON-DATA AND CONTROL DATA ATTACKERS

Recent achievements in mitigation of control-data attacks have offered many unsure approaches. It is more logical to investigate the impact and domain of control-data attacks. Most of hackers, due to complicated penetration processes and lack of knowledge, prefer to run non-control-data attacks against their victims. Although skillful intruders have enough knowledge and experience to penetrate by control data attack approaches, the enhancements in intrusion detection systems have pushed many attackers to focus and penetrate taking advantage of non-control-data vulnerability [19]. If it is considered as fact, and if the breaches of control flow become impossible to penetrate, intruders will be motivated to gain access of the victims' systems.

Some research and studies have figured out that the countermeasures for memory corruptions are not well addressed and it is still a problem at hand. The need for more research projects in this area seems inevitable. Types of attacks using specific memory addressing vulnerabilities such as: stack guard [20], library safe [21], and format guard [22].

5. SOME SOPHISTICATED ATTACKS AND EXPLOITS

As a result of reviewing related papers in this area, it is concluded that in terms of duration and impact of attacks, a large number of attacks misused the application vulnerabilities which allowed intruders to overwrite randomly in memory and the address of a potential application. Some remarkable samples are known as memory flaw injection, integer overflow, format string and stack overflow.

5.1. Low-level software

Most of the penetrations take place by outside intruders and most of the computer victims were breached through controlling software behavior. Normally, this kind of attack can pass through common network communications and inject malicious code into the memory by low-level language vulnerabilities. Through abusing this weakness, these low-level breaches can threaten the execution of the program and hijack control over its behavior. Low level vulnerabilities in languages such as C++ and C must be addressed to tackle these security issues. Whenever high level languages are interpreted to low-level ones, it could be a potential

risk for injection. The only benefit of unprotected interpretation is saving more memory and deliberations. Software developers show their aim utilizing the reflections of more elevated languages. If these reflections are not saved in low-level executions, then this error can lead the product to act in surprising ways. Frequently, an aggressor will have the capacity to exploit this error and redirect the low-level execution of the product to perform arbitrary functionalities of the attacker's choice [17].

5.2. Data properties

File settings are used extensively by many web applications. For instance, in order to manage and configure several configurations of Apache web server, it can be configured by the framework manager using httpd.conf. Admin can manage and specify the address of data and executable files, security policies and access control parameters. The same configuration and files are used in file transfer protocol (FTP), secure shell (SSH) [23], and other network server applications. Most of server apps process configuration files at very early steps of program execution in order to launch internal data structures. During the process, above sources are utilized to manage and control the behavior of server applications and rarely the service chain of server could break or change once it enters the process. Attackers take control of victims' behavior by modifying configuration data structures. The server application at runtime process can find file path of data and executable files. As a matter of fact, a web server can prevent any malicious activity by a common gateway interface (CGI) program, which is a path directive. Configuration files are included in a list of trusted and secured programs which specify the location of executable apps. If the configuration file is manipulated through memory, it can potentially lead attackers to bypass access control [24].

5.3. User data property

During execution of authentication process for initializing remote user, security protocols validate the user data identifications such as ID and group ID access privileges which are cached through memory for granting access. Remote access decision utilizes cached information for data identification. The unauthorized access could happen by changing the identity of the user and modifying cache information previously stored in the memory [25]. The impact of this process usually takes place under Boolean variable on a single register of memory**.**

5.4. Decision-making data

Decision making for authentication consists of multiple steps. Authentication of users is a critical decision which can result in granting access and authorizing either the right users or intruders. However, manipulating unprotected memory with binary codes is not an easy approach for attackers [17].

5.5. User input string

Another approach for intruders to lunch a successful attack is to use non-control-data attack by changing user entry. In order to prevent this kind of attack, validating of inputs is vital and plays a key role in reinforcing security policies. Intruder would gain control of system when modifications on user input takes place after the validation process. The following step will be pursued in an attack: First, legitimate data entry is used to proceed through the validation process of an app. Injection and modification of cached input data is performed. Finally, modified data is run and used to be considered as legitimate user data. This type of attack is known as TOCTOU, which is time of check to time of use [18].

5.6. Vulnerabilities of server app services

There was a quick investigation done by CERT in 2015 which stated 87 memory vulnerabilities. The Most well-known of these weaknesses are integer overflow, format string and buffer overflow, and etc. In the study of [18] it was revealed that from 87 weaknesses, 73 of them are capable of exploits which are mentioned in the following:

- 18% vulnerabilities of hypertext transfer protocol (HTTP) service,
- 10% vulnerabilities of database service,
- 8% vulnerabilities in remote login,
- 5% vulnerabilities in mail service, and
- 4% vulnerabilities in FTP service.

There are many types of memory flaws and corruptions. Although a significant portion of these vulnerabilities are illustrated in the Figure 2, which shows trends of incidents, there are many uncovered weaknesses in non-control-data attacks that need to be investigated further.

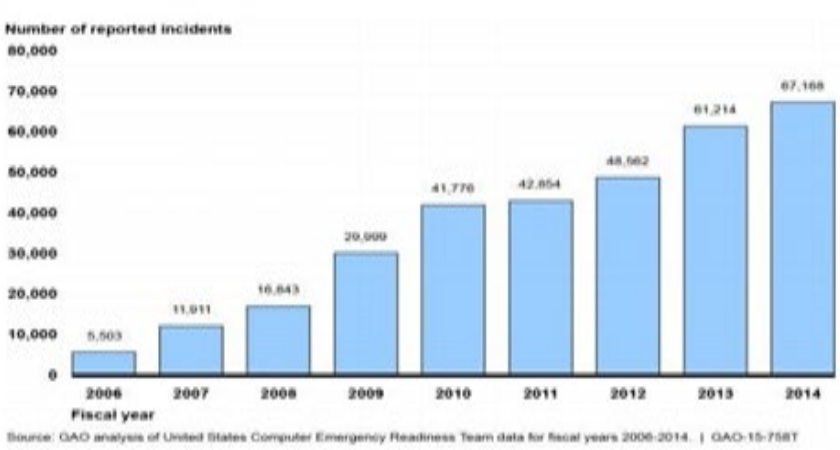

Figure 2. Incident reported to the U.S. Computer Emergency Readiness Team by Federal Agencies, fiscal years 2006 through 2014

5.7. Heap corruption attacks

Memory corruption vulnerabilities could happen by telecommunication network (TELNET) and HTTP daemon. If the daemon is run as root, HTTP or Telnet daemon let manipulation on configuration data user take place and let the intruder get root shells. The point is that some HTTP daemons can be launched as an unprivileged user, i.e., a user with special permissions such as the administrator. In this case, the root privilege compromises the whole system, whether the attack is a control-data attack or a non-control data attack [25].

5.8. Stack buffer overflow attack against user input data

Another sample of stack buffer overflow weaknesses could potentially be GHTTP and HTTP, which have buffer overflow vulnerability in their login processes. In comparison with other types of memory corruption, the stack overflow does not inject the code in random memory locations. The unsecure buffer on the stack is the most popular spot to exploit. This vulnerability, which is known as stack smashing method, can inject malicious codes to be written in stack in a return address and cause root compromise [26].

5.9. Attacking null httpd

Another type of memory corruption is Null HTTPD which is a web server in Linux [1]. There is an available vulnerability which can be exploited by overwriting on heap memory while the entry function is using the buffer. When the corrupted heap buffer is injected, the program control jumps to malicious codes and the root is compromised.

6. PURPOSING COUNTERMEASURE TECHNIQUES

Reviewing all studies and research projects shows that several server applications must be protected due to their vulnerabilities. There are several protective methods which are proposed in the following. As in a conducted study [26], the countermeasures are categorized in two main methods: 1) security software, which can secure and cover memory-safety from corruption, and 2) preventing memory overflow. In the following plenty available security measures to avoid most of the novel attacks are presented:

a. Some security measures are based on practical protection in below arrangements, such as:
- pointer restriction [27] and
- random space addressing [28]

b. Other types of security measures are based on integrity exploits in control flow attacks, such as:
- system call based on intrusion detection approaches [29-31],
- control data protection approaches [32-34], and
- non-executable-memory-based protections [35]

6.1. Security-critical non-control data

As presented in a study done by Chen *et al.* [5], there are some security-critical data applied by non-control-data attacks, which are illustrated as follows.

- Configuration data. Configuration files are used by a number of applications to identify and categorize access control policies, as well as file path directives, so that the location of the secure executables is determined. In order to initiate unintended applications (such as root shells), or even bypass the web server access controls, an attacker may overwrite such configuration data.
- User input data. A very common approach in software engineering is to consider user input data as distrusted and apply those upon validation. In case an attacker alters the input data after it has gone through validation process, the program could be executed with malicious input.
- User identity data. UIDs and GIDs are kept in memory as the common authentication procedures are being executed. Upon modifications in the IDs, impersonating a user with administrative privileges could help an attacker gain control over the program.
- Decision making data. One type of values used to make decisions in applications are Boolean values (whether authenticated or not), which could be modified by an attacker to redirect the flow of the application and make it run through unintended procedures.

In addition, Hu *et al.* [36] applied the following items to improve the previous types of security-critical data:

- Passwords and private keys. Full privileges to a system might be given to an attacker upon disclosure of passwords and private keys.
- Randomized values. A number of security related mechanisms (including CFI, ASLR, SSP) apply CFI enforcement tags, random canary words, and randomized addresses. In case an attacker manages to get his hands on the random canaries in the stack, he could perform stack-smashing attacks without the urge to modify Stack Smashing Protector (SSP).
- System call parameters. Privilege escalation, or unintended execution of programs could be a result of alterations in security-critical system calls' parameters (such as execve, setuid).

6.2. Intrusion detection approach

One of the most essential approaches to secure the communications of network and stream of data is IDS. Intrusion detection systems can be monitored based on the behavior of data packets and application processes. Typically, the host base IDS inspects the memory and the system call. If any kind of deviation from the normal model occurs at runtime, it will be considered as malicious activity and will be immediately blocked at the very beginning of the process [37].

6.3. Control data protection techniques

Recent papers have investigated the methods to mitigate corruption of control data attacks. There are two techniques, the compiler technique, and the microprocessor architecture technique. In compiler technique, DIRA is an automated compiler which uses integrity checker to make sure of the data flow. This technique is the most popular security measure because control-data attacks are currently considered as the most dominant attacks [38].

6.4. Memory safety enforcement

Another security software which can verify the safety of memory is CCured. This program tries to estimate the portion of vulnerability of codes which, in turn, means it is based on analysis and static information to avoid flaws such as null value in pointer or out of range addressing. This technique, which is known as type-safety, is an inference algorithm to enforce the security of memory [39].

6.5. A challenge to overcome memory corruption attacks

The review of studies on this area shows that although, proposing a common and practical approach to secure memory vulnerabilities and to stop memory corruption attacks is still an open and critical problem, the special approaches can only overcome attacks by addressing different types of countermeasures depending upon the exploit's characteristics and the features of memory vulnerabilities. Most of the common defensive techniques just provide control flow integrity. Hence, the security coverage is insufficient. Further studies in the field to address constraints of deployments are inevitable and highly recommended [40].

7. CONCLUSION

The review of all these papers in the area of non-control and control data attack shows that there is no ultimate security and reliable application to prevent all verities of modification data attacks. Recent researches and studies in this scope.

REFERENCES

[1] A. Lazzez and T. Slimani, "Forensics investigation of web application security attacks," *International Journal of Computer Network and Information Security*, vol. 7, no. 3, pp. 10-17. 2015.

[2] K. K. Vasan and P. A. R. Kumar, "Taxonomy of SSL/TLS Attacks," *International Journal of Computer Network and Information Security*, vol. 8, no. 2, pp. 15-24, 2016.

[3] N. C. S. Iyengar, A. Banerjee, and G. Ganapathy, "A fuzzy logic based defense mechanism against distributed denial of service attack in cloud computing environment," *International Journal of Communication Networks and Information Security*, vol. 6, no. 3, pp. 233-245, 2014.

[4] J. C. Demay, E. Totel, and F. Tronel, "Automatic Software Instrumentation for the Detection of Non-control-data Attacks," in *Intrusion Detection: 12th International Symposium*, *Proceedings, RAID 2009*, Saint-Malo, France, pp. 348-349, September 23-25, 2009.

[5] S. Chen, J. Xu, E. C. Sezer, P. Gauriar, and R. K. Iyer, "Non-control-data Attacks are Realistic Threats," *14th USENIX Security Symposium*, pp. 177-191, 2005.

[6] A. Sotirov, "Bypassing Memory Protections: The Future of Exploitation," 2009, [Online]. Available: http://www.usenix.org/events/sec09/tech/slides/sotirov.pdf.

[7] M. Abadi, M. Budiu, U. Erlingsson, and J. Ligatti, "Control-flow integrity: Principles, implementations, and applications," *ACM Trans. Inf. Syst. Secur*. vol. 13, no. 1, 2009.

[8] J. Demay, E. Totel and F. Tronel, "SIDAN: A tool dedicated to software instrumentation for detecting attacks on non-control-data," *2009 Fourth International Conference on Risks and Security of Internet and Systems (CRiSIS 2009)*, pp. 51-58, 2009.

[9] W. Z. Khan, X., Yang, M. Y. Aalsalem, and Q. Arshad, "Comprehensive study of selective forwarding attack in wireless sensor networks," *International Journal of Computer Network and Information Security*, vol. 3, no. 1, pp. 1-10, 2011.

[10] Z. Hu, V. Mukhin, H. Loutskii, and Y. Kornaga, "Stochastic RA-Network for the Nodes Functioning Analysis in the Distributed Computer Systems," *International Journal of Computer Network and Information Security*, vol. 8, pp. 1-8, 2016.

[11] B. B. Kodada, G. Prasad, and A. R. Pais, "Protection against DDoS and Data Modification Attack in Computational Grid Cluster Environment," *International Journal of Computer Network and Information Security*, vol. 4, no. 7, pp. 12-18, 2012.

[12] A. One, "Smashing the stack for fun and profit," *Phrack Magazine*, vol. 7, no. 49, 1996.

[13] Tim Newsham, Format String Attacks, [Online]. Available: https://seclists.org/bugtraq/2000/Sep/214

[14] CERT Security Advisories 1988-2004. *Software Engineering Institute*, [Online]. Available: http://www.cert.org/advisories/

[15] United States Computer Emergency Readiness Team. Technical Cyber Security Alerts, [Online]. Available: http://www.us-cert.gov/cas/techalerts/

[16] Microsoft Security Bulletin, [Online]. Available: http://www.microsoft.com/technet/security/

[17] C. Schlesinger, K. Pattabiraman, N. Swamy, D. Walker, and B. Zorn, "Modular Protections against Non-control Data Attacks," in *2011 IEEE 24th Computer Security Foundations Symposium*, Cernay-la-Ville, pp. 131-145, 2011.

[18] J. R. Crandall and F. T. Chong, "Minos: Control Data Attack Prevention Orthogonal to Memory Model," *37th International Symposium on Microarchitecture (MICRO-37'04)*, Portland, OR, USA, pp. 221-232, 2004.

[19] M. Dhakar and A. K. Tiwari, "A New Model for Intrusion Detection based on Reduced Error Pruning Technique," *International Journal of Computer Network and Information Security*, vol. 5, pp. 51-57, 2013.

[20] B. A. Kuperman, C. E. Brodley, H. Ozdoganoglu, T. N. Vijaykumar, A. Jalote, "Detection and prevention of stack buffer overflow attacks," *Communications of the ACM*, vol. 48, no. 11, pp. 50-56, 2005.

[21] T. T. Baratloo and N. Singh, "Transparent run-rime defense against stack smashing attacks," In *Proceedings of USENIX Annual Technical Conference*, June 2000.

[22] M. B. Cowan, S. Beattie, and G. Kroah-Hartman, "FormatGuard: Automatic protection from printf format string vulnerabilities," In *Proceedings of the 10th USENIX Security Symposium*, Washington, DC, August 2001.

[23] S. Z. Melese and P. S. Avadhani, "Honeypot System for Attacks on SSH Protocol," *International Journal of Computer Network and Information Security (IJCNIS)*, vol. 8, no. 9, pp. 19-26, 2016.

[24] The Apache Software Foundation, [Online]. Available: http://www.apache.org/

[25] Null HTTPd Remote Heap Overflow Vulnerability, [Online]. Available: http://www.securityfocus.com/bid/5774 and http://www.securityfocus.com/bid/6255

[26] Ghttpd Log() Function Buffer Overflow Vulnerability, [Online]. Available: http://www.securityfocus.com/bid/5960

[27] C. Cowan, S. Beattie, J. Johansen, and P. Wagle, "PointGuard: Protecting pointers from buffer overflow vulnerabilities," In *Proceedings of the 12th USENIX Security Symposium.* Washington, DC, August 2003.

[28] PaX Address Space Layout Randomization (ASLR), [Online]. Available: http://pax.grsecurity.net/docs/aslr.txt

[29] H. Feng, J. Giffin, Y. Huang, S. Jha, W. Lee, and B. Miller, "Formalizing sensitivity in static analysis for intrusion detection," In *Proceedings of the 2004 IEEE Symposium on Security and Privacy*, May 2004.

[30] J. Wilander and M. Kamkar, "A Comparison of Publicly Available Tools for Dynamic Buffer Overflow Prevention," In *Network and Distributed System Security Symposium (NDSS)*, vol. 3, pp. 149-162, 2003.

[31] H. Feng, O. Kolesnikov, P. Fogla, W. Lee, and W. Gong, "Anomaly detection using call stack information," In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, May 2003.

[32] J. R. Crandall and F. T. Chong, "Minos: Control data attack prevention orthogonal to memory model," To appear in *Proceedings of the 37th International Symposium on Microarchitecture*. Portland, OR. December 2004.

[33] Smirnov and T. Chiueh, "DIRA: Automatic detection, identification and repair of control-data attacks," In *Proceedings of the 12th Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, February 3-4, 2005.
[34] G. Suh, J. Lee, and S, "Devadas. Secure program execution via dynamic information flow tracking," In *Proceedings of the 11th International Conference on Architectural Support for Programming Languages and Operating Systems*. Boston, MA. October 2004.
[35] S. Andersen and V. Abella, "Data Execution Prevention. Changes to Functionality in Microsoft Windows XP Service Pack 2, Part 3: Memory Protection Technologies," [Online]. Available: http://www.microsoft.com/technet/prodtechnol/winxppro/maintain/sp2mempr.mspx
[36] H. Hu, Z. L. Chua, S. Adrian, P. Saxena, and Z. Liang, "Automatic generation of Data-Oriented Exploits," in: *Proceedings of the USENIX Security Symposium*, 2015.
[37] P. Barford, J. Kline, D. Plonka, A. Ron, "A signal analysis of network traffic anomalies," In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurement*, *ACM*, pp. 71-82, 2002.
[38] O. Ruwase and M. S. Lam, "A practical dynamic buffer overflow detector," In *Proceedings of the 11th Annual Network and Distributed System Security Symposium*, pp. 159–169, February 2004.
[39] T. Jim, G. Morrisett, D. Grossman, M. Hicks, J. Cheney, and Y. Wang, "Cyclone: A safe dialect of C," in *Proceedings of USENIX Annual Technical Conference. Monterey*, CA, June 2002.
[40] D. Jang, Z. Tatlock, and S. Lerner, "SafeDispatch: Securing C++ Virtual Calls from Memory Corruption Attacks," in *Network and Distributed System Security Symposium (NDSS)*, 2014.

BIBLIOGRAPHIES OF AUTHORS

Vahid Kaviani is an Invited Lecturer in Computer Engineering Department at the IAUN University and Security Trainer in Metaco Security Academy Institute. He received his Master degree in Information Security from public university (UTM) University Technology of Malaysia. His recent publications include Efficient High-Rate Key Management Technique for Wireless Body Area Networks (IEEE explorer, 2016); Efficient Algorithm for Feature Intruder Detection System (IRJET, 2016); Gravitational Search Algorithm for Feature Selection in Intrusion Detection System (Master Thesis, 2013). His research interests include SCADA & IDS vulnerabilities. He is currently teaching computer courses in different universities and applying for PhD position in Information Security.

Parvin Ahmadi Doval Amiri received the B.Sc. degree in in Software Engineering, Islamic Azad University Kashan Branch, Kashan, Iran in 2009; the M.Sc. degree in Software Engineering, Islamic Azad University Babol Branch, Mazandaran, Iran, in 2013, respectively. From September 2009 to now, she is full-time lecturer in Islamic Azad University Isfahan (Khorasgan) Branch September 2009 to March 2017. March 2015 to now she is expert and responsible for computer Laboratory at Faculty of Engineering. Her research areas include Distributed Systems Specifically Cloud Computing, Cloud Federation, Virtualization Technology, and Big Data management and processing approaches and Security related issue in IoT and cloud computing and Distributed Systems.

Farsad Zamani Boroujeni received his PhD in Computer Science from Faculty of Computer Science and Information Technology, University Putra Malaysia. Currently, he is working as a faculty member at Azad University, Isfahan (Khorasgan) branch.

Nima Akhlaghi was born in Esfahan province in 1993. He is an IT Engineer with many expertise knowledge and work experience in information security, system analysis and Network administration. He graduated in B.C. from IAUN University. His last position was the director of student science association in Computer Engineering faculty at IAUN University. Also, he was the founder of IAUN LUG when he was student at IAUN University. He worked in many startups and Hi-tech projects and enterprises. He was a member of the Esfahan's young elite's association. His Recent works and interests are in security application and security solutions. He has developed many security countermeasures for Servers, IoTs, Apps and other type of systems.