International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 2 Issue: 11 3587 – 3590

A Survey on Response Computation Authentication Techniques

Usha Sunil Gound
Student
Dept. of Computer Engg.
DGOI, COE, Daund, Pune
Email:ashwet.kdm@gmail.com

Prof. Tanaji A.Dhaigude
Assistant Professor
Dept. of Computer Engg.
DGOI, COE, Daund, Pune
Email:tanajidhaigude@gmail.com

Shweta Satish Kadam
Student
Dept. of Computer Engg.
DGOI, COE, Daund, Pune
Email: *goundusha9@gmail.com*

Avinash Shivaji Gaikwad
Student
Dept. of E & TC Engg.
SCOE, Vadgaon (Bk), Pune
Email:avinash.gaikwad12@gmail.com

***Abstract*—** as we know the problems regarding data and system security are challenging and taking attraction of researchers. Although there are many techniques available which offers protection to systems there is no single Method which can provide full protection. As we know to provide security to system authentication in login system is main issue for developers. Response Computable Authentication is two way methods which are used by number of authentication system where an authentication system independently calculates the expected user response and authenticates a user if the actual user response matches the expected value. But such authentication system have been scare by malicious developer who can bypass normal authentication by covering logic in source code or using weak cryptography. This paper mainly focuses on RCA system to make sure that authentication system will not be influenced by backdoors. In this paper our main goal is to take review of different methods, approaches and techniques used for Response Computation Authentication.

Key words: Security System, Response Computable Authentication, password, hacking.

*****

I. INTRODUCTION

User authentication is baseline for access control, for granting privileges to user at very first user should gone through authentication process ,if user is authenticated and if it is authorized user then and then only user can access system. Through login process authentication system checks a user's identity to grant privileges to system. Login module is became an attraction for attacker to hack system. Common attack on login module is through backdoors. Backdoors means malicious developers purposely cover code in login module to bypass normal authentication, to gain access of system. Basically authentication system is fall into two categories, first category is after a user responds to authentication challenge, authentication system calculates expected user response using credential on server and matches it with users response if it matches then user is authenticated which is based on Response Computable Authentication[1]. Second category is user's response is used as input to authentication computation which is based on techniques such as public-key cryptography and zero knowledge proof. In this paper we mainly focus on backdoors in response computable authentication system.

Backdoors entries may be purposely inserted or carelessly left in software. Backdoors are a method of bypassing authentication or other security controls in order to access a computer system or the data contained on that system. Backdoors entries are a risk that should be detected before system being deployed. Backdoors can present at a number of different levels, backdoors present at system level, in a cryptographic algorithm or within application code. Application backdoors are often inserted in code by genuine user to access code in critical situation.

Application backdoors are detected by observing the source code or observing binary statically. It is difficult to detect backdoors dynamically because backdoor may use secret data or functionality that cannot be tested for using only dynamic methods.

There are several categories of application backdoors which can be detected using automated static analysis:

• Special credentials
• Unintended network activity
• Deliberate information leakage

Modern static analysis methods can detect many classes of common vulnerabilities. Static analyzers do this by building a semantic model of the software which typically includes control flow and data flow graphs. This model is then scanned for patterns that typically lead to vulnerabilities such as buffer overflows. Static analysis methods can also be targeted at detecting code that offers backdoor functionality to an attacker Who knows of its existence Binary static analysis has the powerful capability of being able to use static analysis techniques when source code is not available, which is the typical case when a consumer is concerned about detecting a backdoor in a product they have purchased.

Types of Backdoors:

a) Bypassing Response Comparison Backdoor [1].

- U triggered Backdoor [1]- In this type of backdoor special user inputs can be used to trigger covered logic in source code or vulnerabilities.

- G triggered Backdoor [1] -in this type at global state like specific time or MAC address Response Comparison function L () returns true value and user can enter into system.
- I triggered Backdoor [1]: in this type Response Comparison Function L () function returns true when login failure frequency falls into specified range.

b) Response Computation Collision based Backdoor [1].

In this category, when given specific password pw *and* challenge *cha*, the response-computation function *f*() generates the same response*'*. *Response'* is determined by the password *pw*, which is set by the normal user and is unknown to the attacker. Therefore, the attacker cannot predicate the exact response generated by *f*().

If output of response computation function F () is different for different passwords then attacker can guess passwords. Then attacker can easily guess exact response generated by f().

II. WORKING OF RESPONSE COMPUTABLE AUTHENTICATION:

Basically to eliminate backdoors following three steps are used

- Explicit Response Comparison:

  This method is performed to make sure that response comparison function value L () is derived from expected response and actual user. Actually the verification process L () is divided into two steps: response *computation* and *response comparison* to increase the execution transparency. Response-computation function f *usually* contains a lot of cryptographic computations and is backdoor-prone. Response computation is put it in NaPu to prevent control flow hijacking (e.g. through exploiting vulnerabilities in *f*). This step can guarantee that the comparison statement cannot be bypassed, and eliminates Response Comparison bypass backdoor.

- Function Purification:

  When response computation function value is calculated then it is put into NaPu such that it can take only explicit input like passwords and further for each and every time for calculation of response computation function f Proposed system resets the memory used by function and for each run allocate memory in fixed manner in this way function purification method removes backdoors entries from internal and global states.

- Backdoors usability testing:

  This method performs collision testing to verify whether the response computing function F () is not suffering from high collision probability.

III. REVIEW ON AUTHENTICATION TECHNIQUE.

1. In [1] Author designed a model that decomposes the authentication module into components. Components with simple logic are verified by code analysis for correctness; components with cryptographic obfuscated logic are sandboxed and verified through testing. The key component of proposed module NaPu, a native sandbox to ensure pure functions, which protects the complex and backdoor-prone part of a login module.
2. In [2] Author proposed a method to disrupt a static disassembly, it makes attacker more tough to disassemble the executable code. By using reverse engineering process attacker can reconstruct executable code and find vulnerabilities in system to gain access of system. In reverse engineering executable code is disassemble which translate machine code to assembly code. That machine code is used to get source code.
3. In [3] Author proposed Defensive strategy Blue-chip that has both a design-time component and a runtime component. Blue-chip uses unused circuit identification (UCI) during the design verification phase, to identify suspicious circuitry—those circuits not used or otherwise activated by any of the design verification tests. When Blue Chip identifies suspicious circuitry it removes it and replaces it with hardware which generates exception. Author proposes a technique to prevent all hardware attacks.
4. In [4] Author proposed a system that is able to examine multiple execution paths and identify virulent actions that are executed only when certain conditions are met. This enables us to automatically determine the complete view of the program under analysis and identify under which circumstances suspicious actions are carried out. This technique also helps to observe behavior of malware that means many malware samples show different behavior depending on input read from the environment.
5. In [5] Author proposed a model to encrypt the program which is conditionality dependant on input value and remove the key from program. Author implemented tool which is compiler level based that takes a malware source program and automatically generates an obfuscated binary.
6. In [6] Author had concentrated on Observing for indicators that the software is trying to hide its behavior from detecting dynamically.
7. In [7] Author worked on to develop a architecture that build a voting machine which is finite –state transducer that implements the uncovered essentials required for an election. And also stressed on fact that machine will behave correctly on Election Day.
8. In [8] Author constructed circuits that have malicious behavior, but that would search for detection by the UCI algorithm and also passes design-time test cases. To search for such circuits, Author implemented one class of malicious circuits and performs a bounded exhaustive enumeration of all circuits in that class.

9. In [9] Author proposed a techniques that allows to build trustworthy hardware systems from components designed by untrusted designers or gained from untrusted third-party IP providers.
10. In [10] Author designed methods to strengthen the fundamental assumption about trust in microprocessors, by employing practical, lightweight attack detectors within a microprocessor.
11. In [11] Author worked on implementation and evaluation of a kernel root kit identification system for the Windows platform called Limbo, which prevents kernel root kits from entering the kernel by checking the legitimacy of every kernel driver before it is loaded into the operating system. Limbo determines whether a kernel driver is a kernel root kit based on its binary contents and run-time behavior.
12. In [12] Author introduced a technique for implementation and evaluation of Native Client, a sandbox for untrusted x86 native code. Native Client aims to give browser-based applications the computational performance of native applications without compromising safety. Native Client uses software fault isolation and a secure runtime to direct system interaction and side effects through interfaces managed by Native Client. Native Client provides operating system portability for binary code while supporting performance-oriented features generally absent from web application programming environments.
13. In [13] Author presented two simple password-based encrypted key exchange protocols based on that of Bellovin and Merritt. While one protocol is more suitable to scenarios in which the password is shared across several servers, the other enjoys better security properties.
14. In [14] Author Tested the thing that it is possible to extract private keys from an Open SSL-based web server running on a machine in the local network. Authors result demonstrated that timing attacks against network servers are practical and therefore security systems should defend against them.
15. In [15] Author proposed a novel approach for automatically detecting deviations in the way different implementations of the same specification check and process their input. This approach automatically builds a symbolic formula from implementation and also reduces the number of lines of inputs needed to find deviations. and works on binaries directly, without access to the source code..
16. In [16] Author worked on to enhancing network intrusion detection with integrated sampling and filtering stream of packets.
17. In [17] Author provided the techniques to help vendors, independent testing agencies, and others to verify critical security properties in direct recording electronic (DRE) voting machines.

IV. CONCLUSION:

The privacy of host based system and network based system is major research challenge in the field of Intrusion detection systems under real time environment. Attackers exploits vulnerabilities of the system and networks to bypass authentication and gain access of system. Hence one must have security strategy which can able to secure system and network, there are many traditional methods were presented to solve this problem. They are not much effective. Thus to solve this problem recently some more techniques were presented. In this paper we have discussed different methods of Authentication System.

V. REFERENCES

[1] Shuaifu Dai1, Tao Wei1, 2, Chao Zhang1, Tielei Wang3, Yu Ding1, Zhenkai Liang4, Wei Zou1 "A Framework to Eliminate Backdoors from Response-Computable Authentication". 2012 IEEE Symposium on Security and Privacy.

[2] C. Linn and S. Debray. Obfuscation of executable code to improve resistance to static disassembly. In CCS '03: Proceedings of the 10th ACM conference on Computer and communications security, 2003.

[3] M. Hicks, M. Finnicum, S. T. King, M. M. K. Martin, and J. M. Smith. Overcoming an Untrusted Computing Base: Detecting and Removing Malicious Hardware Automatically. In *2010 IEEE Symposium on Security and Privacy (SP)*, pages 159–172, 2010...

[4] A. Moser, C. Kruegel, and E. Kirda. Exploring Multiple Execution Paths for Malware Analysis. In *SP '07. IEEE Symposium on Security and Privacy*, pages 231–245, 2007.

[5] M. Sharif, A. Lanzi, J. Giffin, and W. Lee. Impeding malware analysis using conditional code obfuscation. In *Proceedings of the 15th Annual Network and Distributed System Security Symposium*, San Diego, CA, February 2008.

[6] Tyler Shields, Veracode Chris Wysopal, Veracode "Detecting "Certified Pre-owned" Software"March 25, 2009.

[7] C. Sturton, S. Jha, S. A. Seshia, and D. Wagner. On voting machine design for verification and testability. In *Proceedings of the 16th ACM conference on Computer and communications security*, CCS '09, pages 463–476, New York, NY, USA, 2009. ACM.

[8] C. Sturton, D. Wagner, and S. T. King. Defeating UCI: Building Stealthy and Malicious Hardware. In *32th IEEE Symposium on Security and Privacy*, 2011.

[9] A. Waksman. Silencing Hardware Backdoors. In *32nd IEEE Symposium on Security and Privacy*, 2011.

[10] A. Waksman and S. Sethumadhavan. Tamper Evident Microprocessors. In *IEEE Symposium on Security and Privacy*, pages 173–188, 2010.

[11] J. Wilhelm and T.cker Chiueh. A forced sampled execution approach to kernel rootkit identification. In *10th International*

Symposium on Recent Advances in Intrusion Detection (RAID'07), pages 219–235, 2007.

[12] B. Yee, D. Sehr, G. Dardyk, J. B. Chen, R. Muth, T. Ormandy, S. Okasaka, N. Narula, and N. Fullagar. Native Client: A Sandbox for Portable, Untrusted x86 Native Code. In *30th IEEE Symposium on Security and Privacy*, pages 79–93, 2009.

[13] Michel Abdalla David Pointcheval," Simple Password-Based Encrypted Key Exchange Protocols" Feb. 14{18, 2005. Springer-Verlag, Berlin, Germany.

[14] D. Brumley and D. Boneh. Remote timing attacks are practical. In *Proceedings of the 12th conference on USENIX Security Symposium - Volume 12*, pages 1–1, Berkeley, CA, USA, 2003. USENIX Association.

[15] D. Brumley, J. Caballero, Z. Liang, J. Newsome, and D. Song. Towards automatic discovery of deviations in binary implementations with applications to error detection and fingerprint generation. In *Proceedings of USENIX Security Symposium*, Aug. 2007.

[16] J. Gonzalez and V. Paxson. Enhancing Network Intrusion Detection with Integrated Sampling and Filtering. In *Recent Advances in Intrusion Detection (RAID)*, volume 4219 of *Lecture Notes in Computer Science*, pages 272– 289. Springer Berlin / Heidelberg, 2006.

[17] N. Sastry, T. Kohno, and D. Wagner. Designing voting machines for verification. In *Proceedings of the 15th conference on USENIX Security Symposium – Volume 15*, Berkeley, CA, USA, 2006. USENIX Association.