ISSN 2278-3091
Volume 9, No.3, May - June 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse176932020.pdf
https://doi.org/10.30534/ijatcse/2020/176932020

Logic Bomb: An Insider Attack

Palash Sandip Dusane[1], **Yallamandhala Pavithra**[2]
[1]Department of Information Technology, SRM Institute of Science and Technology, Kattankulathur, India, palashdusane@gmail.com
[2]Department of Information Technology, SRM Institute of Science and Technology, Kattankulathur, India, pavithrays1997@gmail.com

ABSTRACT

Cyberattacks are one of the biggest threats to the Computer World. Security researchers and professionals are continuously making various efforts to prevent such attacks. Most of the times these attacks come from outside, but sometimes it can be an insider attack too. Insider attack can be more lethal, as the malicious person will have authorized system access. Logic Bomb is one of such examples of an insider attack. A successfully triggered logic bomb can cause system failure, auto-deletion of hard drives, manipulation of data, etc. Logic Bombs are generally hidden or embedded in genuine code where they stay dormant until their conditions are not satisfied, this makes them very hard to detect. This malware is normally programmed by a developer of the software. The attackers usually exploit software development lifecycle to insert a logic bomb. Such type of hidden malicious code in the system software can be a serious threat to their IT infrastructure. As the world is now moving toward smart and digital cities, all the IoT and Automated systems should be protected from such attack. This paper involves the study of an insider logic bomb attack, its preventive measures, proposed code-level detection system and detailed steps for recovery.

Key words: Code-Level Detection System, Insider Attack, Logic Bomb, Malware.

1. INTRODUCTION

Cyberattacks are performed by both outsider and insider entities. In a comparison of both; insider attack can be more dangerous because of the malicious person having authorized access to system and network. One such effective insider attack is a Logic bomb attack.

A logic bomb is a piece of malicious code that lies dormant and hidden within a legitimate software until a condition is satisfied to trigger its payload. This malware is normally embedded by developers into genuine software [1]. A logic bomb has a flaw that it only works for a software for which it has been designed, it doesn't replicate on other applications [2]. Presence of Logic bomb in system poses great risks to its security and integrity. Working of logic bomb code in a genuine code is shown in figure 1.

Richard Clarke (Former White House counterterrorism expert) shares his concerns about the cyber war in his book named "Cyber War: The Next Threat to National Security and What to do about it." In this book, he mentioned that the U.S. is very vulnerable to this type of attack because its infrastructure is much dependent on computer networks in comparison with other countries. He showed concern that the malicious code developers could trigger logic bombs, shutting down the banking and various other systems [3].

One such cyberattack occurred in South Korea on 2013-3-20 14:00:00 local time, a logic bomb in the code erased the hard drives of banks and media companies. It triggered on specific time-date and started wiping all data from machines. At least two broadcasting companies and three banks were attacked on that day. This attack prevented South Koreans from making ATM transactions as the attack put some ATMs out of operation. According to security researchers, the malicious code also included a module to connect remote Linux machines for deleting their master boot record.

Time Bomb is a type of logic bomb. It is programmed to get executed at a specific time and date. It will stay inactive until its specific condition is met. Execution of a time bomb will result in destructive effects on system and network. Time bomb like "Friday the 13th" activates on a specified day and deletes all the files from infected computers [4].

It is very hard to detect a logic bomb in the testing phase. Functional testing cannot detect such malicious code, as the tester is unlikely to supply values which will trigger the bomb. Static analysis tools are unable to detect these bombs, as they are aimed at finding programming mistakes like failure to validate a user input or failing to protect against a buffer overflow. Requirements-based testing also fails to detect the bomb, as a tester is unlikely to enter values required to trigger them. Logic bombs cannot be detected by dynamic tools either, because their execution may not result in runtime errors, which is usually detected by dynamic analysis [5].

Antivirus based on anomaly detection may not detect logic bomb as it will not cause events which are externally observable such as unusual system call. Signature detection is also not a full-proof solution, as the malicious code is custom embedded to its host software, and may not have a unique signature [6].

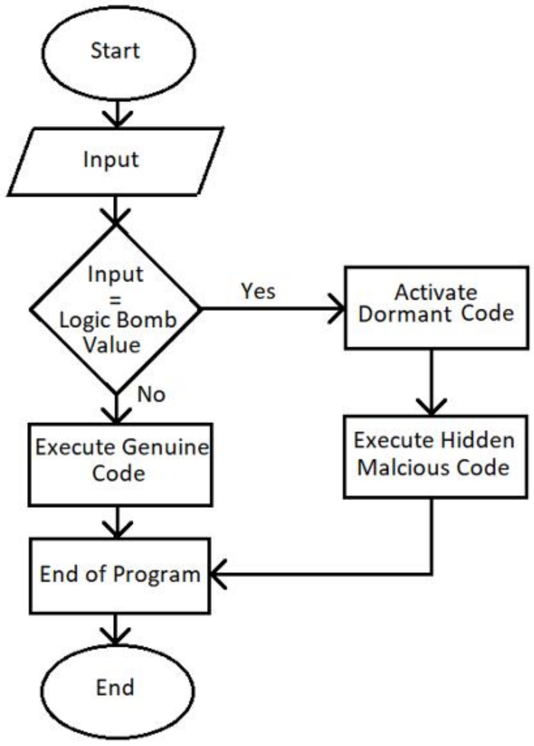


Figure 1: Working of Logic Bomb.

2. RECENT CASE STUDY

In July 2019, a former contractor of Siemens pleaded guilty for planting malicious code in spreadsheets that crashes the software every few years. He created those custom spreadsheets in 2002 for various projects of Siemens in related to the power generation industry.

For so many years when the logic bomb went off, he was called and paid to fix the issue, where he used to just fix the clock for the next attack. However, in May 2016 when he went on vacation, he had to give his administrative password to Siemens employee to solve the issue. The Siemens employee found that logic bomb while checking the spreadsheets. The former contractor is now facing up to 10 years of jail time along with a fine of up to $ 250,000 [7].

3. PREVENTION AGAINST LOGIC BOMBS

As logic bombs are mostly planted by software developers of the same company or externally hired contractors, the best practices that involve putting procedural and technical controls should be in place. Following are some necessary steps to prevent logic bomb attack:

- Conduct a complete background check of developers at the time of joining.
- Provide only the necessary access to developers to avoid any misuse of resources.
- Keep the system updated by patching it regularly. This will make it more difficult for a malicious developer to attack the system.
- Use updated anti-malware and IDS systems [8].
- Secure account management and password controls. Ensure different password for different host systems [9].
- Separate the development and testing process.
- Implement the principle of least privilege to ensure less possible targets for an attacker [10].
- Prepare baseline for the known process and regularly compare it to the current view. This will be useful to find rogue process [11].
- Perform regular backups, use endpoint protection and use business email screening functionality for mails [12].
- Log all activities of system users and monitor them for unusual patterns [13].
- Keep an eye on employees having signs of dissatisfaction, anger, revenge, etc. [14].
- Keep backups to restore any damage caused by the logic bomb [15].
- Monitor auto-updating software. Use Integrity checker to validate if any software has been modified [16].
- Perform IT Security Risk Assessment [17].
- Manual inspection of critical program code.

4. PROPOSED DETECTION SYSTEM

As we have seen so far, the detection of a logic bomb is very difficult, but what if we try to find its existence in the development phase itself. One of the prevention steps mentioned is to monitor the critical code, but the major problem faced by the code reviewer is to scan the whole project at once. It's an intensive, time-consuming and erroneous process. The proposed system will simplify the code review process by detecting probable logic bomb on code-level. It will scan the program and find keywords that may lead to a logic bomb attack. Thousands of lines of code will be filtered out into few lines of code categorized with risk factor High, Medium and Low. According to the final report generated by the system, the code reviewer will be able to focus on high-risk code. Block diagram of the proposed detection system is shown below in figure 2.

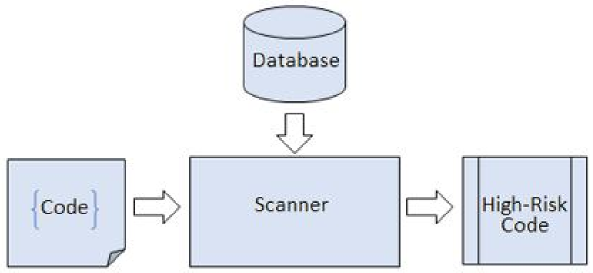
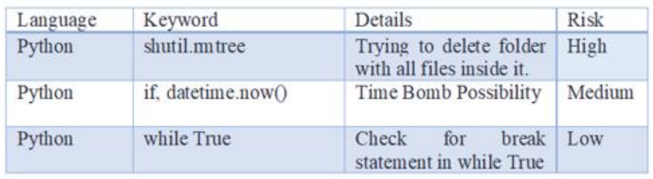


Figure 2: Block Diagram of Detection System.

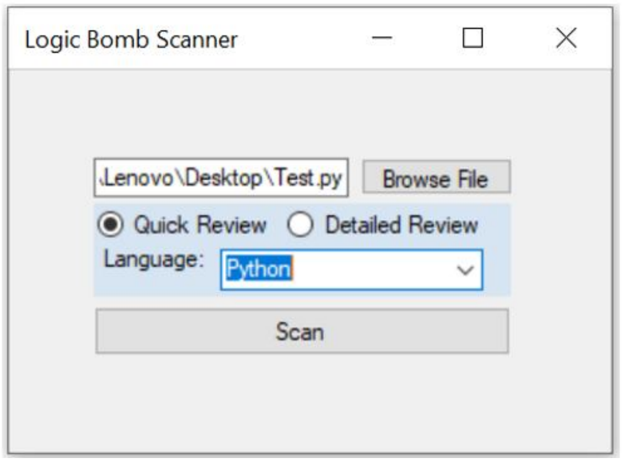
The system will identify high-risk code based on keywords present in the database. Authorized code reviewer will able to edit database by adding or removing vulnerable keywords. User will provide programming file as an input to the system on which system will return a detailed report of keywords found in the program with their description and risk category, as shown in figure 3.



Language	Keyword	Details	Risk
Python	shutil.rmtree	Trying to delete folder with all files inside it.	High
Python	if, datetime.now()	Time Bomb Possibility	Medium
Python	while True	Check for break statement in while True	Low

Figure 3: Example of Records in Database.

The system will allow a user to perform two types of scan, Quick review scan and Detailed review scan. In Quick review scan, the system will scan the program for keywords of a particular programming language whereas Detailed review scan will be for keywords of all programming languages. The scanning process will check the code line by line for keywords. If any code is matched with a keyword then it will be notified to the user in the report. An example GUI of system is shown below in figure 4.



Logic Bomb Scanner

\Lenovo\Desktop\Test.py [Browse File]

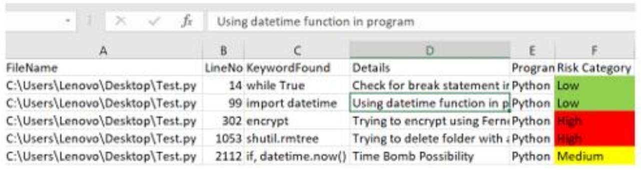
[✓ Quick Review] Detailed Review

Language: Python

[Scan]

Figure 4: Graphical User Interface (GUI) Example.

The proposed system will provide output in the form of a table and pie chart, as shown in figure 5 and 6. It will also have a feature to export the report in excel and pdf file.



Using datetime function in program

	A	B	C	D	E	F
	FileName	LineNo	KeywordFound	Details	Progran	Risk Category
	C:\Users\Lenovo\Desktop\Test.py	14	while True	Check for break statement ir	Python	Low
	C:\Users\Lenovo\Desktop\Test.py	99	import datetime	Using datetime function in p	Python	Low
	C:\Users\Lenovo\Desktop\Test.py	302	encrypt	Trying to encrypt using Ferne	Python	High
	C:\Users\Lenovo\Desktop\Test.py	1053	shutil.rmtree	Trying to delete folder with a	Python	High
	C:\Users\Lenovo\Desktop\Test.py	2112	if, datetime.now()	Time Bomb Possibility	Python	Medium

Figure 5: Sample Excel Report.

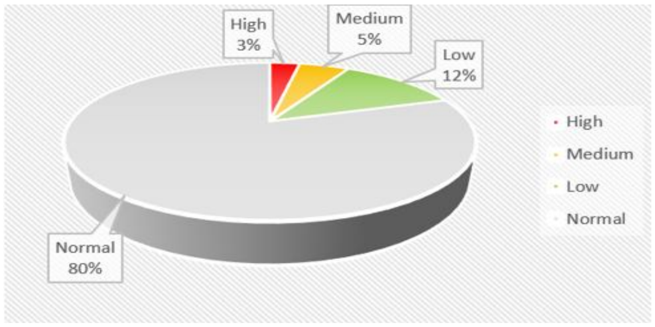


Figure 6: Sample Pie Chart.

In this way, the code reviewer has to check only the lines of code that the system predicted to be of high-risk. This will save a lot of time and work used for intensive code monitoring. Performance of this system will depend on the updated database. Organizations will be able to customize the system's database based on the technology they are using for their software development.

5. STEPS TO RECOVER FROM LOGIC BOMB ATTACK

Following steps can be used to recover from a logic bomb attack on a system or a network:

1) Quarantine (Isolate) the infected system from the network to prevent more damage. If the network has been attacked then disconnect the whole network or at least part of the network where critical data resides.
2) Look for symptoms and review all relevant log files to find the start of an attack. In this way, you will able to locate logic bomb in the infected host system.
3) Collect the evidence for Investigation of the Incident. This will help to find the attacker behind the logic bomb attack.
4) Remove the logic bomb from the system in a safe mode. Perform various checks to make sure malware has been successfully removed from the system.
5) Restore the system with your last backup (if available). Make sure that your last backup doesn't have the same logic bomb.
6) Plug-in the system to the network.
7) Monitor the system/software for a few months to make sure that everything is working properly.

6. CONCLUSION AND FUTURE SCOPE

The system can be attacked by a Logic bomb from both sides i.e. inside and outside. Outsider attack can be handled by avoiding intrusion with tools like firewalls. Insider attack can be more dangerous as it is performed by disgruntled or greedy developers who will have direct access to the company's resources. This paper discusses everything about the insider logic bomb attack starting from its prevention methods, detection system and finally, steps to recover from it.

The proposed detection system identifies malicious instructions on a code-level. It simplifies the code monitoring process by pointing out the probable logic bomb code. Accuracy and efficiency of this system depend on the keywords present in the database. The system may have high false positives, but these false positives will eventually point out bad programming practices. If the database is properly updated then false negatives will be very less. To find accurate data we have to implement this proposed system. Even with all the false positives, this system will be able to provide security in the development phase by providing a faster and automated code review. This paper will be helpful for various organizations to deal with insider attack; moreover, it will also be helpful for new learners to understand logic bomb malware.

REFERENCES

1. Muni Prashneel Gounder, Mohammed Farik. **New Ways To Fight Malware**, International Journal of Scientific & Technology Research, ISSN 2277-8616, Volume 6, Issue 06, June 2017.
2. Manju Khari, Chetna Bajaj. **Detecting Computer Viruses**, International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), ISSN: 2278 – 1323, Volume 3 Issue 7, July 2014.
3. Ankur Singh Bist. **Detection of Logic Bombs**, International Journal of Engineering Sciences & Research Technology, ISSN: 2277-9655, 3(2): February 2014.
4. Trupti Shah. **Network Security- Virus Attacks and Defence using Antivirus Software**, International Journal of Advanced Research in Computer Science & Technology (IJARCST 2014), ISSN: 2347 – 8446, Vol. 2, Issue 4 (Oct. - Dec. 2014).
5. Babak Bashari Rad, Mohammad Kazem Hassan Nejad. **Feature Selection for Malware Classification and Detection: A Literature Review**, International Journal of Advanced Trends in Computer Science and Engineering (IJATCSE), ISSN 2278-3091, Volume 9, No.1.1, 2020.
6. Agrawal, Hira & Alberi, J. & Bahler, L. & Micallef, J. & Virodov, A. & Magenheimer, M. & Snyder, S. & Debroy, V. & Wong, E.. (2012). **Detecting hidden logic bombs in critical infrastructure software**, 7th International Conference on Information Warfare and Security, ICIW 2012. 1-11.
7. **Siemens Contractor Pleads Guilty to Planting 'Logic Bomb' in Spreadsheets**. Available: https://thehackernews.com/2019/07/siemens-logic-bomb.html.
8. Imtithal A. Saeed, Ali Selamat, Ali M. A. Abuagoub. **A Survey on Malware and Malware Detection Systems**, International Journal of Computer Applications (0975 – 8887) Volume 67– No.16, April 2013. https://doi.org/10.5120/11480-7108
9. Agrawal, Hira & Bowen, Thomas & Narain, Sanjai. (2013). **Defending Software Systems against Cyber Attacks throughout Their Lifecycle**, 74-89. 10.5422/fordham/9780823244560.003.0003.
10. Raj, Gaurav & Singh, Dheerendra & Bansal, Abhay. (2014). **Analysis for security implementation in SDLC**, Proceedings of the 5th International Conference on Confluence 2014: The Next Generation Information Technology Summit. 221-226. 10.1109/CONFLUENCE.2014.6949376.
11. Sharma, Bobby. (2017). **A Pragmatic Way of Logic Bomb Attack Detection Methodology**, Indian Journal of Science and Technology. 10. 1-5. 10.17485/ijst/2017/v10i20/110608.
12. A.Anupriya, V.Nithya, S.Ponmalar. **A Survey: Analysis of Virus and Malware Detection**, International Journal of Innovative Research in Computer and Communication Engineering, ISSN 2320-9801, Vol. 6, Issue 3, March 2018.
13. Liu, Liu & Vel, Olivier & Han, Qing-Long & Zhang, Jun & Xiang, Yang. (2018). **Detecting and Preventing Cyber Insider Threats: A Survey**, IEEE Communications Surveys & Tutorials. PP. 1-1. 10.1109/COMST.2018.2800740.
14. Muhammad Saleem , Mirza Naveed Jahangeer Baig , Mufeeza Manzoor , M.Tahir Usman , Saba Akram , Saira Khursheed. **A Brief Overview of Network Attacks & Overcome Techniques**, International Journal of Scientific & Engineering Research, ISSN 2229-5518, Volume 11, Issue 1, January-2020.
15. Suhail Qadir Mir, Mehraj-ud-din Dar, S M K Quadri, Bilal Maqbool Beig, **Information Availability: Components, Threats and Protection Mechanisms**, Journal of Global Research in Computer Science, Volume 2, No. 3, March 2011.
16. Nicolas Robillard. **Diffusing a Logic Bomb**, Global Information Assurance Certification Paper, GIAC Security Essentials Certification (GSEC) Version 1.4b (Option 1), SANS Institute 2004.
17. Sami Haji, Qing Tan and Rebeca Soler Costa. **A Hybrid Model for Information Security Risk Assessment,** International Journal of Advanced Trends in Computer Science and Engineering (IJATCSE), ISSN 2278-3091, Volume 8, No.1.1, 2019.