iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 7 **Issue:** III **Month of publication:** March 2019

DOI: http://doi.org/10.22214/ijraset.2019.3157

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887
Volume 7 Issue III, Mar 2019- Available at www.ijraset.com

Malware Analysis of API Calls using FPGA Hardware Level Security

Dr. Alex Roney Mathew
*Department of Computer Science (**Cyber Security**), Bethany College, West Virginia, USA*

***Abstract:** Malware, also referred to as the distribution of a malicious code present relentless opposition to cybersecurity. In this generation, identifying malware in systems is becoming a war seemingly with no end since the malware is ever evolving and the data security needs to be the first priority. The software is embedded in the perfect way to a user who is unknown by what we call a Malware author. The security-critical resources of the OS come in contact with the malware on the network or the system hosting for the purpose of eliminating their data or to get information which may be private like credit card numbers and passwords. To reduce this crime, Malware authors apply the Application Program Interface (API).*
***Keywords:** Malware, Cyber Security, API, FPGA*

I. INTRODUCTION

An Application Program interface consists of a set of tools and commands for constructing software applications. Primarily, an Application Program Interface states the way components of the software should coordinate. They are also used when elements of the graphical user interface are programmed. An API with good qualities ease the development of a program by resourcing required building blocks. The blocks are then put together by a programmer.

Field programmable gate arrays are critical parts of systems that are intertwined. These systems have become essential parts of our daily lives. Their everywhere presence has made privacy and security mechanisms of preservation essential aspects. FPGA s are traditionally preserved for intricate work. The collection of gates combining to form FPGA may be set to run a specified algorithm with a specific set of instructions.

Taking a look at the API calls, or the cheat codes which tell the system to perform an action, API calls are protective packers which makes it difficult for static analysis to be carried out (Tamada et al., 20). Requests which are performed by the API finds out what a file in the system is set to do rather than reverse engineering a file that is packed protectively. The API calls a file makes the basis of knowing a malicious file some of which resemble malware.

Malware is analysed and executed after being traced. The approaches that are used in the analysis are two; API call analysis and control flow analysis. These two approaches sense malware presence on the basis of the behavior of the old and the new ones. Despite this, authors of malware design codes to get through to the destination or even rearrange the program sequence (Andraka, 98). The current API analysis style does not detect malware which has codes designed in them to get to the required destination. Some known techniques have emphasized bringing out APIs seen in malware. They spy the API called and then make calculations on the occurrence and a cumulative number of happenings called by an API function.

The FPGA works co-ordinately with the Malware-Aware Processor. This is a detector that is based on hardware which classifies malware and deals with it besides monitoring the functions the malware performs in the system. Using hardware-level security, hardware implementation is done using classifiers that are simple. The application of hardware is with its advantages over the detection of software in this case. Hardware implementations are far better off than software implementations which need resources that are to be induced, incur costs that are huge and are performance limited. This argument highlights the importance of simple classifiers which are functional in the detection of malware.

Based on API calls, detectors of malware use information of high level from these calls which are used later on to acquire the common characteristics of malware. In the system design, platforms that are reconfigurable have security which is challenging. Systems that are based on the FPGA could need to keep their operations and communications private. Regarding data compromise and integrity together with confidentiality of the data, the system could be under the threat of alliterations that are malicious FPGA rely on the API calls. They also have technology which accepts the customization and control of switches inside the FPGA that are programmable and results in the programming of functionality. Modern devices containing the FPGA are fused with flash and static memory. The FPGA is a dominant family which is discovered using memories that are based on SRAM. The latter in this case is static memory. When the system powers up configurations of the FPGA takes place before the uploading of the configured settings

to the SRAM. The content that is set is lost in the SRAM as soon as the FPGA is switched off. So to maintain the configurations that facilitate functionality then the FPGAs need to be on since at the loss of power the configurations are lost.

The FPGA hardware-level security has the information on the malware as the API call contains information on the behavior of the malware. The information is obtained by an approach which is dynamic. An additional feature in the hardware level security is implemented to improve the functionality of the API call sequence whose main work was to detect malware after deep scanning. These worked well in conjunction with the API call sequence (Sami et al., 75). If an API sequence exists, then attacks that are anonymous will be checked by confirming the existence of the sequence of the API call. The FPGA had a new detection system of malware known as the API-based malware detection system which was primarily on the analysis of API call sequence.

Based on hardware level security, malware tends to fuse with this system for obstruction as a technique. This makes the detecting of a malware challenging to an individual. Malware that is recent can make their activities unseen through the use of a logic bomb and confirming the presence of the defender. I just remembered that the symbol execution method could be applied. The FPGA hardware design is best suited for hardware that is cryptographic. The main advantages being low cost and faster market time. The system is versatile as the system functionality can be adjusted to provide reactions to malicious attacks. FPGAs fall under several hardware attacks. The main attack being the side channel attack.

The side channel attack is responded to by the API call. This refers to the situation where relevant information is leaked without intentions from a system where programs run as commanded. The leakage may be in many different forms, one being a dissipation of power. The FPGA hardware security level has power dissipation as the most common techniques when perfuming an attack that is side-channeled. Two principal methods are used; Simple Power Analysis (SPA), Differential Power Analysis (DPA). The weight of the data has direct relations to the consumption of power. The higher the hamming weight, the higher the operation power with data. SPA calculates the weight by finding measurements on the height of the pulse of the signal of power consumption at the required cycle of the command that had access to the significant byte.

DPA proves to be stronger than SPA since the attacker is at no need of knowing the elements of the algorithm that was applied. This tactic gets stronger by the use of analysis that was statistical to aid in getting back the information that was side channeled. These tactics can also be used in the invasion of algorithms that are cryptographic. The FPGA seem vulnerable as it has wires that are loaded heavily and consist of lines that are segmented by transistors. These long worse make it useful on measuring if the consumption of power (Pellerin & Scott, 30). For it to keep up to tabs with the value of FPGAs which are volatile, an encryption key which is secret is kept in the device that is programmed and later on a battery that is external keeps on toes with the value of the FPGAs (Cummings & Haruyama, 110). This makes the device to accept bitstream that is encrypted and uses its unique engine decryption to get data which is un-ciphered. Without the secret key, no attacks could be launched to the bitstream. So to access the secret key, the attackers need to acquire the key by all means, and this involves attacks that are intruding to regain data on bitstream.

The FPGA has a vital feature on malware analysis which allows us to conduct remote configuration. This feature gives way for upgrading the system, getting rid of the suspicious element that would create a security breach or improving the algorithms. This characteristic needs proper secure since it gives way for attackers to carry on their malicious activities. Reconfiguration can be done by the attacker without requiring access or permission from the user. An attacker gets between the requests of the user and comes up with his/her configurations which are fake and sends back the feedback to the user. For this to be avoided, then a secure connection is required, and an engine checking integrity should be installed (Alazab et al., 80).

When an API calls the FPGA responds by providing protection like for instance automatic security against hardware of the Trojan type since the designing software gives the FPGA configurations in a precise manner where the information here cannot be visible or be present for the incoming attackers. Additionally, the FPGA structure provides a hard time to make the coordination of hardware Trojans a bit hard. This is however limited as an attacker may decide to change crucial components which are not functional like the power supply network. The detection of some hardware may be difficult since changes applied which are malicious are carried out in a way that makes the analysis challenging. Despite these facts, it is easy to assume that a lot of hardware Trojans which are structural can be analyzed, detected and later on treated with immediate effect (Peiravian & Xingquan, 65).

An API call is detected by a method that is dependent on architecture, but we learn on further that the request is also made by a system trap (Moore, 40). The syscall sequence recognizer is developed to be up to date with the pattern of calls made by the system when the pattern does not match a pattern that is needed. All this is done by a machine that corresponds with all legal patterns and sounds an alarm when a pattern that is illegal is discovered. This machine is developed by carrying out an analysis which is static belonging to the source code of the required program to come up with a call model known as the deterministic finite automation (DFA) which only deals with sequences that are legal. The algorithm is made up of an analysis that flows controllably to create a DFA which cooperates with the call sequences in the system which is activated by going with any path in the flow of control.

As time flew by the FPGAs grew in size, the value of the applications increased thus bringing up the issue of security which was to be strengthened. Vendors in the FPGA department have come up with new techniques like the use of IP cores though the application of FPGA is written in a file that looks like a design, elements of security on information and encryption in conjunction with authentication were given as an application to FPGA bit streams. This was considered not enough given the fact that the FPGA was launched into an environment that is not convincing though the measurements on the improvement of the fields were taken. This included the creation of methodologies that were tolerant to faults for applications. Currently, FPGA security has the required strength to be sent in areas that are sensitive both nationally and commercially. A developer of applications comes up with an application of FPGA which is secured and having the same processes and tools that are used for other applications.

At the final stages of the process of design, the bitstream is developed, and later on, encrypted by the Xilinx software which can also give supplies on the random key and the vector required for initialization. The software used here gives two essential things the first being bitstream that is encrypted and lastly a file that needs key insertion (Guajardo et al., 55).

The FPGA code has no relation and no information concerning the API. In this situation, FPGA functions like other connected devices to the system. In this case, the meaning of this statement is that the API opens the FPGA and handles the machine, open the FPGA program and load it, distribute memory, execute the FPGA function, send and receive data from FPGA and stop the execution of operations and stop the functioning of the FPGA. All the API calls are conducted in a program that runs in the processor of the host. The API gives calls to receive and send data from the FPGA hardware-level security, pack and unpack the FPGA, close and open the machine.

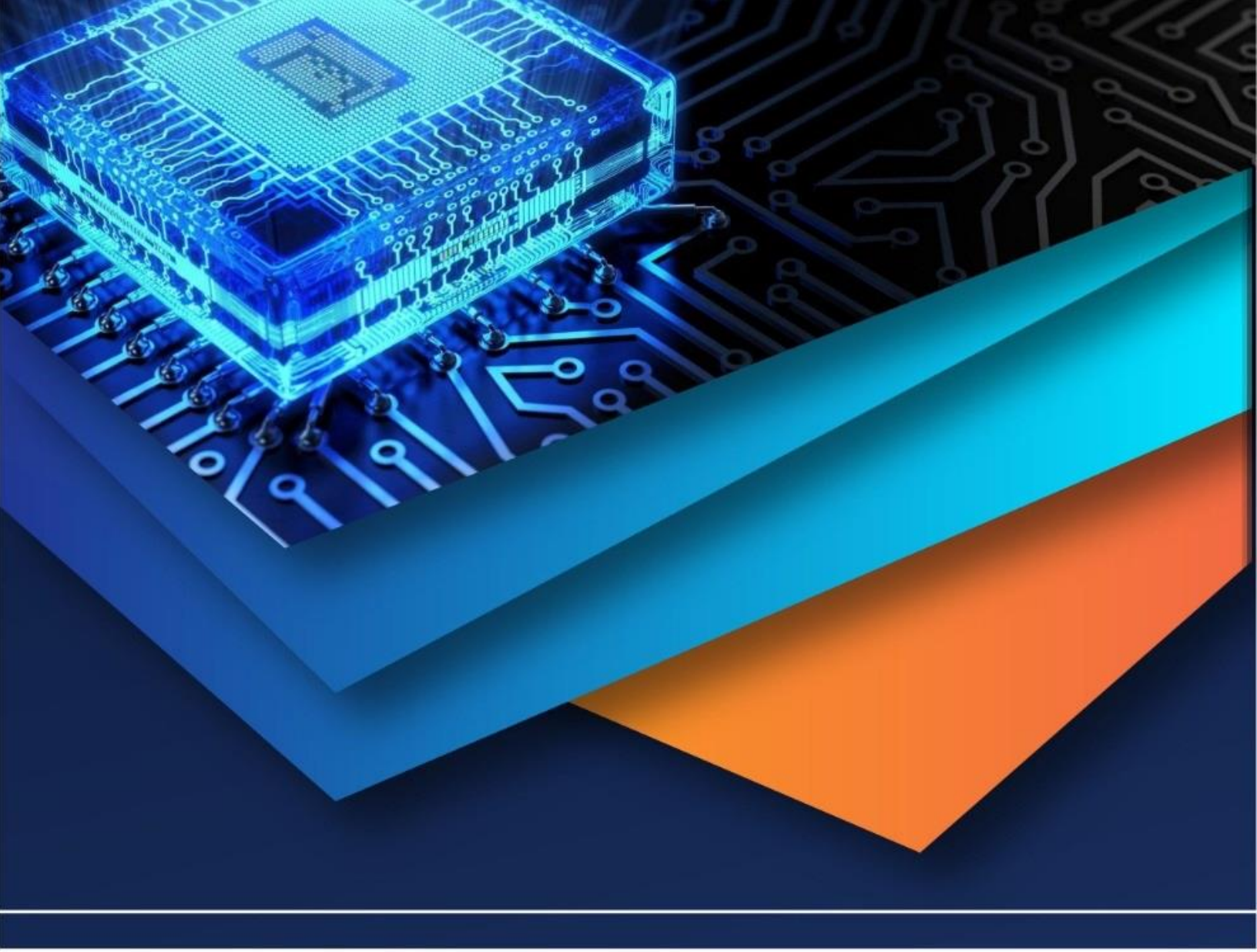
FPGAs offer the chance to see the effect of the combination of hardware and software due to a performance that is increased and function well. Malware analysis is well dealt with by the FPGA hardware security level since it may function as an acceleration device (Kumar et al., 55). It is sad to say that FPGA is not enabled automatically. An experienced programmer has his job set on reconfiguring the program by rewriting and later on designing it better. Unfortunately, there is no assurance that there will be an increase in performance. In this case, an individual who develops software is obligated to adhere to a logarithmic consideration, analysis on timing, and experimenting with the benchmark technique to find out whether or not FPGA acceleration implementation is the best choice.

II. CONCLUSIONS

Malware analysis of API calls using FPGA Hardware-level security is the best analysis. APIs may be the best option to take care of the advancing complexion of FPGAs since they have the same look like systems. From this discussion, Xilinx defines APIs a transition level between software of a higher level and configurable hardware together with several processors on the FPGAs. FPGA relies on API calls which are detected by an architecture technique to detect malware and take immediate action since it has information obtained by the API call which is conducted in the host's processor. The FPGA fall under several attacks which are responded to by the API calls since it has a vital feature which allows remote configuration to prevent attackers from sending fake configurations. The FPGA has its configurations set once the system is on and at the point where it goes off then the configurations are lost and this makes it have no relations with the API but function like other system parts. The combination of hardware and software through FPGA sheds light to us that there is an increase in functionality and performance and FPGA has been proven to have the required strength to be sent in places that are sensitive for better functionality.

REFERENCES

[1] Sami, Ashkan, et al. "Malware detection based on mining API calls." *Proceedings of the 2010 ACM symposium on applied computing*. ACM, 2010.
[2] Peiravian, Naser, and Xingquan Zhu. "Machine learning for android malware detection using permission and api calls." *2013 IEEE 25th international conference on tools with artificial intelligence*. IEEE, 2013.
[3] Moore, Richard John. "Intercepting system API calls." U.S. Patent No. 6,959,441. 25 Oct. 2005.
[4] Alazab, Manoun, et al. "Malware detection based on structural and behavioural features of api calls." (2010).
[5] Tamada, Haruaki, et al. "Design and evaluation of dynamic software birthmarks based on api calls." *Info. Science Technical Report NAIST-IS-TR2007011, ISSN* (2007): 0919-9527.
[6] Andraka, Ray. "A survey of CORDIC algorithms for FPGA based computers." Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays. ACM, 1998.
[7] Pellerin, David, and Scott Thibault. *Practical fpga programming in c*. Prentice Hall Press, 2005.
[8] Guajardo, Jorge, et al. "FPGA intrinsic PUFs and their use for IP protection." *International workshop on cryptographic hardware and embedded systems*. Springer, Berlin, Heidelberg, 2007.
[9] Kumar, Sandeep S., et al. "The butterfly PUF protecting IP on every FPGA." *2008 IEEE International Workshop on Hardware-Oriented Security and Trust*. IEEE, 2008.
[10] Cummings, Mark, and Shinichiro Haruyama. "FPGA in the software radio." *IEEE communications Magazine* 37.2 (1999): 108-112.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY