International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Analyzing Zero Day Attack, its Identification and Prevention

Dr. Aruna Pavate[1], Prasanna Salunkhe[2]

[1,2]School of Computer Science and Information Technology, Symbiosis Skills and Professional University Pune, India

Abstract

All systems with software code are susceptible to an attack called "zero day", which exploits vulnerabilities that are not yet disclosed or known to the public or vendor. Organizations around the world are often left unprotected against these attacks. Cyber criminals follow zero day vulnerabilities closely to commit malicious actions. The goal of this paper is to design a framework utilizing the most efficient methods to detect and contain zero day exploits, propose improvements, and compare current anti-malware tools (AMTs). Analyzing the ability of multiple AMTs to detect zero day malware will assist cyber security professionals in selecting the most compatible for their respective environments and defend against these attacks.

Keywords: Zero day, malware, cyber security, exploit

1. INTRODUCTION

According to a recent report in 2017 from Watchguard, thirty percent of malware was "zero day"[1], a previously unknown vulnerability which has been exploited, and were not detected by anti-malware tools (AMT's). This is because AMT's mostly rely on malware signatures, scans against a known malware database, and miss one third of all malware without a refined threat prevention solution. Even with heuristic scans, a detection method of combining rules and/or algorithms, zero day malware is very difficult to detect. Heuristic based scans are able to detect malware without a signature and improves the level of security, however this method does not provide measures necessary for future and evolving malware because the scan relies on searching for specific pieces of code that may indicate malicious intent. If the search query does not contain specific instructions, or the file is encrypted, then the malware will evade detection. Some sophisticated malware are able to lay dormant and for a specified period of time. This enables it to evade detection for that period, then embed its malicious code into a file or application.

Today, most antivirus programs use both signature and heuristic-based methods in combination, in order to catch any malware that may try to evade detection. However, how can we prevent evolving malware on operating system platforms if we do not know of its existence? Our proposed method is to have the AMT implement a sandboxed environment, a virtual platform designed to capture and examine malware and prohibit *malicious code from leaving a contained area*, that runs at the operating system level. The enhanced sandboxed environment will contain mock system files designed to act as a honeypot for malware to interact with. Malicious behavior will be analyzed utilizing this technique by the AMT. The goal of this analysis is to determine whether the file is malicious and its behavioral attributes. Some

behavioral attributes of malware is where it may traverse, the types of files malware tend to attach itself, as well as the amount of time the malware stays dormant without performing any action. If any malware is zero day, the sandboxed environment will detect and contain it at a higher rate than a conventional AMT because it has gained knowledge from analyzing these test files. These actions will be recorded in a database and compared with future occurrences. With a much wider and stronger database, it will give security expert's a much better opportunity to detect the zero day malware that may infect their systems. Furthermore, this paper will survey a comparison of detection rates for malware between a multitude of currently available anti-malware tools on virustotal.com.

The remainder of this paper is structured as follows: Section II describes detection and containment methods for malware by anti-malware tools. Section III addresses ethical concerns with the creation and use of malicious files. Related research and their outcomes are documented in Section IV. Lastly, Section V includes our methodology, results, and conclusion of our research.

2. Identification and Prevention

A. Signature Based Detection

Signature based detection methods rely on specific criteria to create an evolved fingerprint of known malware. A timeline is utilized so that it can be saved in a central signature database and used by antivirus programs. Signatures are compiled from a number of bytes or hash values of the file that are analyzed. Signature based detection is currently used by most commercial antivirus programs today. The difficulty of creating these signatures is minimal as long as certain behaviors are understood about the specified malware. Signature based detection methods work well against attacks that are behavioral, once it is observed how the malware acts, the signature is recorded and stored in a database. When the behavior of the malware is known, a signature coincides with it. Code reordering is another technique used to evade signature based detection methods. If an attacker alters the malware code, the signature based system will not be able to detect it because it uses a specific signature to be able to detect the malware.

Blacklisting used to be a popular way for signature based systems to detect malicious or suspicious activity that may incur on a system. This creates a database of known signatures and any known exploits or malware in the cyber world. Blacklisting relies heavily on a continuously updated database to be an widely effective evasion technique. The main problem with this technique is if the blacklist is not updated on a regular basis then malware can inject itself rather easily without anyone knowing. Whitelisting has been more used currently because it allows the user of the computer to have a list of permissible applications. For example, if software that is not on the list is installed, the computer will be locked down preventing usage. This can be considered an extreme method of prevention but it can be very useful and helpful to people who do not have the proper detection software on their computer enough to eradicate sophisticated malware. However, whitelisting restrictions hinders the efficiency particularly if a user needs access to an application with a certain time frame. Whitelisting is also extremely vulnerable because if you decide to whitelist an application such as a browser, malware that utilizes browsing scripts can run freely and operate within[2].

The reason signature based detection does not detect zero day malware efficiently is it only detects known attacks based on past signatures. A signature needs to be created for every malware attack. Zero day malware is malicious code that exploits a vulnerability that is not known to anyone; hence the reason signature based detection can't detect it. There isn't any known signature for zero day

malware. This makes it extremely difficult and almost impossible for traditional virus scanning software to detect a zero day attack.

Virustotal is an online system commonly used in which files are uploaded to identify if it is malicious. VirusTotal will scan, and detect, if appropriate, any type of binary content, such as a Windows executable, Android APKs, PDFs, images, javascript code, etc. Most antivirus companies involved in Virustotal will have solutions for multiple platforms, and produce detection signatures for any kind of malicious content.[3]. Virustotal is an efficient program to detect malware for the everyday user without an abundance of vital information scanned, however, it is not usable for detecting zero day malware.

Moreover, in some cases users may lessen their security to keep the system's performance high [13, 14, 15, 26, 27] or use lightweight cryptography [28, 16, 17, 18]. It is also worth mentioning that some systems are vulnerable to attacks that cannot be prevented using cryptographic protocols, such as: jamming [19], MAC misbehaving [20],

packet dropping [21], wormholes [22, 23] and localization

[24, 25].

B. Sandboxing

Sandboxing is one of the techniques suggested in our research. It is a virtual cage which will not allow malware to infect the primary operating system set by the sandbox's restrictions. In order to improve the detection of new malware by anti-malware tools, we recommended recording detections in a sandboxed environment using a central database. The purpose of this database is to collect all malware variants and make them available to a front- end application that will be used to try to detect more evolved forms of malware. This would generate new malware variants and detection signatures based on existing variants that have already been recorded in the database. In concept, the application would take a malware variant, create a duplicate version of it, and include unique code from another malware variant in order to create a new malware variant of the original copy. In other words, this application would be designed to automatically create new malware variants based on combining previous malware variants, providing millions of possible future variants. This would be a key action in order to try and see if it can detect any zero day malware because like discussed previously, zero day malware is not known to anyone. The more variants and possibilities that are created within the sandboxed environment, increases the likelihood that a variant of zero day malware can be detected.

Once new malware variants are automatically generated using the front-end application and the backend malware database, security vendors and researchers will be able to audit their anti-malware tools using the new variants. This improved way of AMT's will test its ability to detect possible future variants as well as provide cyber security professionals with new detection signatures to use should one of the malware variants appear in the future. While it is not sure what the future holds in terms of new malware, this unique technique of relying on old malware variants to predict possible future variants could significantly improve the way malware tools are audited.

While sandboxing has proven to be an effective way to detect zero day malware, it is not foolproof. Zero day malware is created with a purpose -- to evade detection. It would not be as devastating if traditional software were able to detect it with ease. Sandboxing should be used along with a number of other different techniques in order to try to prevent as much zero day malware as possible. There are many evasion methods such as "stalling code". Stalling code "delays the execution of malicious so that a sandbox times out". This "evasion method" works extremely well because malicious activity remains dormant, therefore no anomalies are seen. This can be routinely called a "blind spot"[4]

according to Lastline, a security start up company. All an attacker has to do is program their particular malware to inject itself after a particular period of time. The stalling code takes into account the timeframe of sandboxes then runs after the timeout period. Another evasion method is "blind spot in the sandbox implementation". When sandboxing monitors and detects malware, it introduces a detection method called "hooking". "Hooking" gets notifications that a function or a library call has been implemented. This method seems great, however, the main problem is that modification needs to be done to the program in order for it to work effectively.

Malware writers study "mouse and button replication implementation"[5] as an evasion technique of sandbox environments. This malware is able to detect user patterns of when a human clicks the mouse or scroll for within the environment. By gathering the intelligence from numerous sandbox sequences, malware writers can develop malware that can evade sandbox detection. An example would be if a person uses a pdf for a certain amount of time and there is a usual number of mouse clicks and scrolls during this usage, malware writers would be able to study this behavior and use this as an evasion technique. While sandboxes can be designed to recognize behavior of malware, it also has flaws. Since some malware can detect mouse clicks and scrolling speed inside a sandboxed environment, it can put itself into "a dormant state" or it simply "terminates itself". This then gives the cyber criminal a lot of information so that it can create the perfect piece of zero day malware with the right amount of dormant time in it, or a specific injection speed.

C. Dynamic Heuristic Detection

Dynamic heuristic detection "uses an emulation based technology which loads the target program into a software based CPU emulator". The CPU has the ability to morph into its own virus based computer system that would allow the malware program to execute independently within this system. While the malware program is being emulated within this virus environment, the virus behaviors are logged by the emulator. From this logging, the antivirus program is able to determine if there is a virus in the system. Dynamic heuristic detection also works with any type of encrypted viruses, which allows for greater detection than the signature-based scanning method. Encrypted viruses are starting to be more widely used because most of the anti- virus programs are signature based. Since these programs cannot detect encrypted viruses, attackers are creating more of them.

One of the major advantages that dynamic heuristic detection has is that it performs very well with "file open operation". While anything is being emulated within the emulation program, dynamic heuristic scanning checks the registers for any values that it may contain. These values "specify the task that the target program wants the operating system to perform on its behalf". Heuristic scanning is recommended over signature-based scanning because when it is complete, the registers need to include "certain values" to ensure it correlates with the current sequence that is done at that moment. This is one reason dynamic heuristic scanning is essential in these types of situations, as it "only cares about the values of the registers at the time of the interrupt calls".

As with many detection methods, there are flaws that make it not desirable to use. One limitation of heuristic scanning is the amount of wait time from the CPU emulator. In a constant fast paced environment such as zero day malware detection, this time can be critical to whether a system is infected or it is not. Some viruses are coded with instructions that will enable them to operate at a specified date or time. This causes a problem for dynamic heuristic scanning because it will not detect viruses if no condition is met by the heuristic scanning. Depending on the type of heuristic detection method, the scan may not be able to detect the virus because it is looking for too many variants within

a vast amount of space. In order for dynamic heuristic scanning to become a more effective form of malware detection, the scanning methods need to be researched further based on past zero day malware injections.

D. Dynamic Heuristic Scanning vs. Sandboxing

While it may be easy to determine one method is more efficient than another, the best approach is to compare techniques used and match them together to maximize effectiveness. One does not want to limit themselves to one technique because this can be detrimental to their efforts in finding the most effective way to detect and contain zero day malware. Both sandboxing and dynamic heuristic scanning have their benefits as discussed and "a multi scanning approach" should be used in order to utilize the different scanning engines that each method provides. A piece of malware can just be sandboxed and contained within an environment, but no resolution may come about from it. The malware can possibly evade detection by the sandbox and its tools. If sandboxing is combined with a sophisticated dynamic heuristic scan tool as well, then there stands to be an even greater chance to malware becomes eradicated and the system will be protected. If the malware is detected by the combination use of these methods, and would not have been by just one of them, then the malware that was detected can be analyzed and its behavior can be used to detect future malware variants.

3. TESTING CONCERNS

To determine if certain malware can be detected, different antivirus scanning engines and methods need to be conducted. This raises a concern of ethics in the cyber community because in order to create different variants and unseen types of malware, malware needs to be created by the hundreds or even thousands. The reason this is done is to try to get as many different kinds of variants of malware possible so that they can be screened and put into the databases for anti-virus engines to conduct their research. This might sound promising and may seem like it can lead to great discoveries, but it also raises a red flag to the entire community of how easy it can be to create a catastrophic cyber event.

Ethics also play a major role in testing, especially for malware because testing and creation of malware can be done by inexperienced testers or people who do not know how to fully handle the malware they are creating. While they may think they are doing a service to virus testers and cyber experts, in actuality what they are doing is creating an even bigger problem. This realization is unethical and harmful. Malware can then get into the hands of malicious intent and wreak havoc upon systems without the knowledge or intent of the creators. There are a lot of strict standards in the cyber community about who is reliable with creating malware and who can be trusted with the safety of it. If allowed to just do anything they want, researchers who are not trained and trusted in the community can run "unreliable virus generators, irrelevant virus simulators and random making of virus code and string".[6] This would be detrimental to the cyber community and the ethical concerns from the ramifications that would happen would be extraordinary.

Existing Anti-Malware Tools

FireEye, a cybersecurity firm, conducted research for recent exploits for vulnerabilities found at that time. This work is related to our current research and illustrates the effectiveness of the AMTs listed on Virustotal to detect new malware variants. The research conducted used "High-profile" Flash exploits such as (CVE 2014-0497, 2014-0502, 2014-0515, 2014-9163, 2015-0310, 2015-0311, and 2015-0313) and IE exploits with SWF components (CVE-2014-0322 and CVE-2014-1776), with the earliest known sample exploit of the given vulnerability, then plotted the detection rates

recorded by Virustotal (VT) with respect to time in Figure 1. curves that start with a high detection rate climb steeply from day zero. While there are mixed results, there are some patterns. The first days of a new attack's lifetime were plotted in the first 20 days of each CVE in Figure B. The Flash component of the CVE-2014-0322 didn't have any detections during its first 20 days on VT, hence, does not appear in Figure 2.

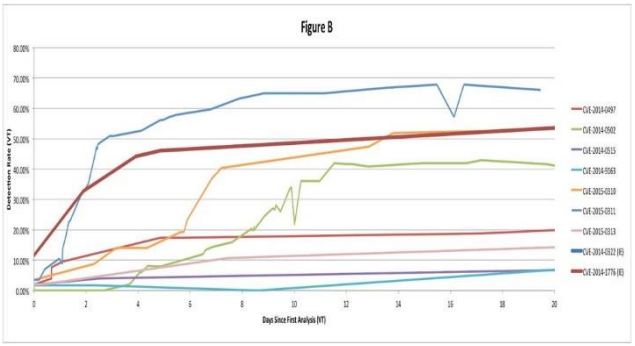

Figure 2: Detection rate of new attacks in first 20 days on VirusTotal [8]

Day zero on VT saw 1-2 out of 47-57 detections across every single high-profile Flash exploit in FireEye's sample set. The results of this particular research suggests detection ratios on VT are "far from representative of detection ratios in actual attacks". It is documented that flashvars, filenames, and URLs are not present. The conclusion of this study suggested behavioral analysis will not be effective if the exploit aborts early, when environment and parameters don't look right, or when second stage payloads have been taken down. It was determined since these detection ratios are so low across the board it indicates that the industry has trouble detecting previously unknown Flash exploits.[9] The same detection rate for unknown exploits, not just Flash, (ie: Java, Internet Explorer) can be generalized.

4. ANALYSIS

A. Methodology

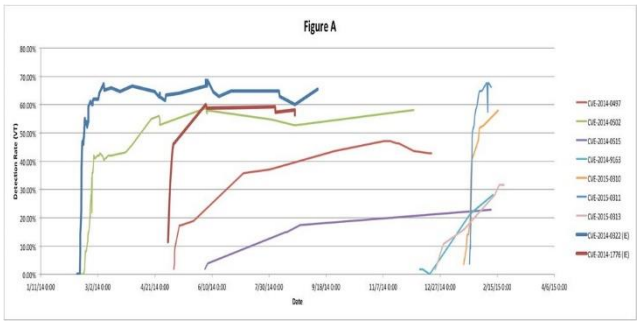

Figure 1: Earliest known exploit detection rates on VirusTotal [7]

According to this related study, the detection rate on the Y- axis is the percentage of detections in individual analyses on VT. ie: 25 detections from 50 products would result in 50% on the Y-axis. The X-axis is the date of each analysis run on VT. The previous work in this graph showed The AMT detection rates tested in our research are audited based on malware samples that we created with metasploit.[10] We have configured a malicious executable (exe) file that is encoded with Shikata Ga Nai. This encoder implements a polymorphic XOR additive feedback encoder. The decoder stub is generated based on dynamic instruction substitution and dynamic block ordering. Registers are also

selected dynamically.[11] Another malware sample used in our testing is a pdf embedded with a metasploit payload. This module embeds a metasploit payload into an existing PDF file. The resulting PDF can be sent to a target as part of a social engineering attack [12].

To simulate zero day malware, we used the tool msfvenom within metasploit to create a python script which we decoded with base64decode [13]. The purpose of the script was open a reverse shell to a remote machine. Once we decoded the source code of the python script, we were able to modify it. Using base64decode, we implemented messages using the "#" character within the code which could be used for instruction for other zero day malware. After adding a few messages in the source code, it also caused segmentation to further evade AMT detection, such as an attacker sends fragmented packets to evade an intrusion detection system. We then encoded the script after modification, converted it into an exe file, and placed it into our sandboxed environment for testing. All variants were created with evasion techniques that are designed to explicitly avoid AMT detection.

B. Results

Table 1 represents ten anti-malware tools listed on virustotal which scanned our malware embedded in PDF and EXE files. The "●" symbol signifies if the malware was detected by an AMT. The results are depicted below:

TABLE I
ANTI-MALWARE TOOL RESULTS

AMT	Malicious PDF	Malicious EXE
Ad-Aware	•	•
AVG	•	•
BitDefender	•	•
ClamAV	•	•
Comodo	•	•
Kaspersky	•	•
Malwarebytes		•
McAfee	•	•
Sophos	•	•
Symantec	•	•

These results show that almost every malware tool was able to detect the malicious pdf and malicious exe files. Malwarebytes was the only program not to detect even one and this was the malicious pdf file. Our future analysis will look more in depth to malwarebytes and the way the algorithms are configured and to see why it was not able to detect the malicious pdf.

Table 2 represents ten anti-malware tools listed on virustotal which scanned our simulated "zero day" malware where we created and modified the source code of a malicious python script. The "●" symbol signifies if the malware was detected by an AMT. As predicted, the zero day malware was able to evade detection on every single anti-malware tool that we tested. This illustrates the nature of zero day malware and how more research is necessary to detect, record, and contain these types of malicious software. Even zero days may perform a lot of the same actions non zero day malware, it contains signatures that have never been analyzed previously, therefore it is highly unlikely traditional methods such as signature and heuristic based will detect its presence.

Table 3 is a combination of Tables 1 & 2 and depicts the detection rate of our encoded EXE file, PDF file injected with malicious code, and our modified python source code which simulated the zero day malware The "●" symbol signifies if the malware was detected by an AMT. The results are depicted below:

Table 3: Overall Detection Rate

AMT	Malicious PDF	Malicious EXE	Zero-Day	Detection Rate
Ad-Aware	•	•		2/3
AVG	•	•		2/3
BitDefender	•	•		2/3
ClamAV	•	•		2/3
Comodo	•	•		2/3
Kaspersky	•	•		2/3
Malwarebytes		•		1/3
McAfee	•	•		2/3
Sophos	•	•		2/3
Symantec	•	•		2/3

This table is a collaboration of all of our research of the malware tools combined and gives a much easier view of how effective each one of the tools are. As stated previously the tools performed very well in detecting the regular malware but not the zero day malware.

Upon concluding this research with the results that were gathered we want to change the results of the zero day malware detection in some way so that possibly we can create a streamlined detection system for creating malware techniques that would be able to combine various signatures from different malware and provide more detection for the zero day malware that is out there.

5. Conclusions

With the amount of malware steadily increasing and risk it poses to the public, we suggest creating an automated system which would create variants of existing malware as well as variants of the previously created variants. This would potentially create millions of malware strains that would then be entered into a database and enable security firms to audit their AMTs. In addition, we suggest designing an AMT which would be able to emulate a real environment using dynamic heuristic and sandboxing methods which aim to detect evasive malware techniques. After scanning our created malware, we found that most AMTs were able to detect the malicious PDF and EXE files. Extra measures were implemented in the EXE

file to hide malicious intent with the Shikata Ga Nai encoder. The malicious PDF file was also injected with code where once opened by the user would open a backdoor for an intruder to utilize. We expected the PDF file to remain dormant unless it was activated and that it would evade detection from most of the AMTs. The detection rate is as follows: EXE file: 10/10. PDF file: 9/10. Malwarebytes was the only AMT that did not detect the malicious PDF file.

It is concluded the ten AMT's listed in our study had a detection rate of 95 percent (19/20) when scanning our EXE and PDF malware variants. When scanning our zero day exe file, the detection rate was 0 percent (0/10) It is also imperative to note, beyond the scope listed in this particular study, the overall detection rate of VT for the EXE file was 41/61, the PDF file was 33/66, and zero day malware was 0/56.

The results of our research showed that while most Anti-malware Tools listed on Virustotal.com were able to detect malware created with evasive techniques, it was not successful in detecting zero day malware. This further confirms our proposal to have a unique anti-malware detection tool which combines sandboxing, a proactive database, and dynamic heuristic techniques.

REFERENCES

1. "Internet Security Report - Q4 2016", WatchGuard Technologies. [Online]. Available: https://www.watchguard.com/wgrd-resource- center/security-report
2. Finjan Cybersecurity "Blacklisting vs. Whitelisting", 2017 [Online] Available: https://blog.finjan.com/blacklisting-vs-whitelisting- understanding-the-security-

benefits-of-each/
3. "VirusTotal" Frequently Asked Questions [Online] Available: https://www.virustotal.com/en/faq/
4. D. Keragala. "Detecting Malware and Sandbox Evasion Techniques" 2016. [Online] Available: https://www.sans.org/reading- room/whitepapers/forensics/detecting-malware-sandbox-evasion- techniques-36667
5. E. Messmer "Malware-detecting 'sandboxing' technology no silver bullet" 2013. [Online] Available: http://www.networkworld.com/article/2164758/network-security/malware- detecting--sandboxing--technology-no-silver-bullet.html
6. D. Harley, A. Lee. "Heuristic Analysis - Detecting Unknown Viruses" 2009. [Online] Available: https://www.welivesecurity.com/media_files/white-papers/Heuristic_Analysis.pdf
7. [7]D. Caselden, C. Souffrant, G. Jiang. "Flash in 2015" Figure A [Online] Available: https://www.fireeye.com/blog/threat- research/2015/03/flash_in_2015.html
8. [8]D. Caselden, C. Souffrant, G. Jiang. "Flash in 2015" Figure B [Online] Available: https://www.fireeye.com/blog/threat- research/2015/03/flash_in_2015.html
9. D. Caselden, C. Souffrant, G. Jiang. "Flash in 2015" [Online] Available: https://www.fireeye.com/blog/threat- research/2015/03/flash_in_2015.html
10. Metasploit Penetration Testing Tool [Online] Available: https://www.metasploit.com/
11. "Vulnerability and Exploit Database" Shikata Ga Nai Encoder[Online] Available: https://www.rapid7.com/db/modules/encoder/x86/shikata_ga_nai
12. "Vulnerability and Exploit Database" Adobe PDF Embedded EXE [Online] Available:https://www.rapid7.com/db/modules/exploit/windows/fileformat/adobe_pdf_embedded_exe
13. Base 64 Decode and Encode [Online] Available: https://www.base64decode.org/
14. T. Hayajneh, S. Ullah, B. Mohd and K. Balagani, "An Enhanced WLAN Security System with FPGA Implementation for Multimedia Applications," IEEE Systems Journal, 2015.
15. T. Hayajneh, B. Mohd, A. Itradat and A. Quttoum, "Performance and Information Security Evaluation with Firewalls," International Journal of Security and Its Applications, SERSC, vol. 7, no. 6,
16. pp. 355-372, 2013.
17. T. Hayajneh, S. Khasawneh, B. Mohd and A. Itradat, "Analyzing the Impact of Security Protocols on Wireless LAN with Multimedia Applications," in Proc. of The Sixth International Conference on Emerging Security Information, Systems and Technologies (SECURWARE), 2012.
18. Bassam Jamil Mohd, Thaier Hayajneh, Khalil M. Ahmad Yousef, Zaid Abu Khalaf, Md Zakirul Alam Bhuiyan, Hardware design and modeling of lightweight block ciphers for secure communications, Future Generation Computer Systems, 2017, ISSN 0167-739X, http://dx.doi.org/10.1016/j.future.2017.03.025.
19. B. J. Mohd, T. Hayajneh, Z. AbuKhalaf, K. Yousef "Modeling and Optimization of the Lightweight HIGHT Block Cipher Design with FPGA Implementation," Security and Communication Networks, John Wiley, Vol. 9, No. 13, pp 2200-2216, 2016. (DOI: 10.1002/sec.1479)

20. BJ Mohd, T. Hayajneh, M. Z. Shakir, K. A. Qaraqe, AV Vasilakos "Energy Model for Light-Weight Block Ciphers for WBAN Applications," In Proc. of IEEE 4th International Conference on Wireless Mobile Communication and Healthcare (IEEE MobiHealth'14), Athens, Greece, 2014.
21. K. Panyim, T. Hayajneh, P. Krishnamurthy and D. D. Tipper, "On limited-range strategic/random jamming attacks in wireless ad hoc networks," in Proc. of IEEE Conference on Local Computer Networks (IEEE LCN), 2009.
22. Hayajneh, Thaier, Ghada Almashaqbeh, and Sana Ullah. "A Green Approach for Selfish Misbehavior Detection in 802.11-Based Wireless Networks." Mobile Networks and Applications 20.5 (2015): 623-635.
23. Hayajneh, T.; Krishnamurthy, P.; Tipper, D.; Kim, T. Detecting malicious packet dropping in the presence of collisions and channel errors in wireless ad hoc networks. In Proceedings of the IEEE International Conference on Communications, Dresden, Germany, 14–18 June 2009; pp. 1–6.
24. Hayajneh, T.; Krishnamurthy, P.; Tipper, D.; Le, A. Secure neighborhood creation in wireless ad hoc networks using hop count discrepancies. Mobile Netw. Appl. 2012, 17, 415–430.
25. Hayajneh, T.; Krishnamurthy, P.; Tipper, D. Deworm: A simple protocol to detect wormhole attacks in wireless ad hoc networks. In Proceedings of the IEEE 3rd International Conference on Network and System Security, Gold Coast, Australia, 19–21 October 2009; pp. 73–80.
26. Hayajneh, T.; Doomun, R.; Krishnamurthy, P.; Tipper, D. Source—Destination obfuscation in wireless ad hoc networks. Secur. Commun. Netw. 2011, 4, 888–901.
27. Doomun, R.; Hayajneh, T.; Krishnamurthy, P.; Tipper, D. Secloud: Source and destination seclusion using clouds for wireless ad hoc networks. In Proceedings of the IEEE Symposium on Computers and Communications, Sousse, Tunisia, 5–8 July 2009.
28. Bhuiyan, Md Zakirul Alam, et al. "Event Detection through Differential Pattern Mining in Cyber-Physical Systems." IEEE Transactions on Big Data (2017). (DOI: 10.1109/TBDATA.2017.2731838)
29. T. Hayajneh, R. Doomun, G. Al-Mashaqbeh, BJ Mohd "An energy-efficient and security aware route selection protocol for wireless sensor networks," Security and Communication Networks, John Wiley, Vol. 7, No. 11, pp 2015-2038, 2014. (DOI: 10.1002/sec.915)
30. B. J. Mohd, T. Hayajneh, and A. V. Vasilakos, "A survey on lightweight block ciphers for low-resource devices: Comparative study and open issues," Journal of Network and Computer Applications, vol. 58, pp. 73–93, 2015.