Multi Platform Honeypot for Generation of Cyber Threat Intelligence

Sanjeev Kumar
Department of Computer Applications
National Institute of Technology
Tiruchirapali, India
405918054@nitt.edu

B. Janet
Department of Computer Applications
National Institute of Technology
Tiruchirapali, India
janet@nitt.edu

R. Eswari
Department of Computer Applications
National Institute of Technology
Tiruchirapali, India
eswari@nitt.edu

***Abstract*—The threat landscape is exponentially increasing which become more worsen when a greater number of emerging devices are connected to internet such as Internet of Things (IoTs), Embedded Systems, Cyber Physical devices etc. To control the damage of cyber threats, there is a need to monitor the cyber criminals continuously to understand the tools and technique used by the attackers in order to develop cyber defense mechanism to protect cyber Eco-systems.**

In this research, a multi-Honeypot platform as a tool is presented for cyber threat intel generation to implement the multiple classes of Honeypots such as Windows, IoTs, Embedded etc. Honeypot is widely used by the security researchers, security companies to understand the tools and tactics about the attackers but these are quite complex to deploy and maintain especially due to diverse set of IT systems and intensive resource requirements to deploy High Interaction Honeypots. This complexity is reduced in this research by implementation of Para-Virtualization based approach to enable multiple classes of Honeypot sensors of different platforms on a light weight hardware.

It is addressed that time window to collect the data and to conclude it as a cyber threat intel with support of evidences should be probabilistically determined. After applying the analysis such as behavior analysis and deep learning methods to determine about the unknown threat patterns, the attack data sets are correlated into different cyber threat events and converted into an actionable cyber threat intelligence to disseminate the information in an automated manner. In the end, threat intelligence is generated and experiments are documented. The deep learning-based analysis inspired by neural networks is integrated in the design to determine the unknown classified threat events.

Keywords— Cyber Threat Intelligence, Cyber Security, Honeypots, Malware, Attack Patterns

I. Introduction

When Lance Spitzner introduced the concept of Honeypot as so called Gen I(Generation 1) Honeynet [1], the idea was to catch the script kiddies and malicious attackers who were trying to perform the attacks on a network. This idea was introduced in the year of 1999, till it was converted into more advanced GenII(Generation II) Honeynet concept with real resources in place in line with the production resources. Then in the year 2005, the GenIII Honeynet Technology were introduced with inclusion of Virtual Honeynet , it means more number of Honeypots can be implemented using Virtualization. The data captured through different classes of Honeypot sensors deployed at various Geo-locations are collected at central locations for further data analysis. Also, data analysis on the Honeynet client node is developed as Roo-Honeywall walleye console [2]. The walleye is the GUI interface of live connections captured on Honeypots. Figure 1 depicts the few open source honeypot tools developed over the period of time by various researchers. Started from low interaction honeypot which emulate the protocols and services to be attacked by the attackers to the high interaction which provide the real resources. Though low interaction Honeypots are less resource intensive but only capable to capture known attacks whereas high interaction class of Honeypots are capable to detect zero-day attacks. The example shows the few low interaction honeypots- KFSensor, Dionaea, Honeyd, Kippo and high interaction Honeypots as Modern Honeynet Network(MHN) and Honeynet.

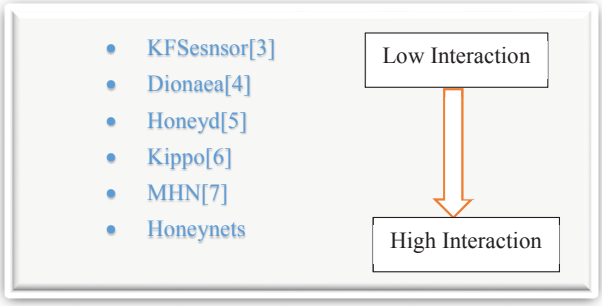

Fig. 1. Honeypot Tools

II. Background

A. Honeypot as Threat Indicator Detection

Another scope to enhance the detection approach of SIEM or other related security defensive solutions is use of Honeypots. Honeypots is an environment where the vulnerabilities are intentionally introduced in order to catch the latest and complex attacks who are trying to attack the asset of the organizations. Moreover honeypot had shown the capabilities of zero-day attack detection [8]. The evidences captured on the Honeypot is treated as attacks infection with low false positive and also these evidences can be used as contexts of the attacks. These information can lead to generation of threat intelligence as detection indicator to SIEM [9-11]. The research published recently in [12] discuss the Symantec DeepSight Threat Management System which is a kind of early warning system consists of HoneyPot Networks.

B. Cyber Threat Intelligence-Issue and Challenges

Though Cyber Threat Intelligence is a research Topic now a days, there are lots of challenges to be faced by this growing area of research[13]. In present scenario, there are multiple heterogeneous sources of threat data, their data quality and integration with threat intelligence sharing platform is the biggest issue. The data collected from various sources has a

978-1-7281-4392-7/19/$31.00 ©2019 IEEE

Authorized licensed use limited to: Universitaetsbibliothek Innsbruck. Downloaded on March 23,2021 at 09:09:31 UTC from IEEE Xplore. Restrictions apply.

different attribute that need to be processed and converted into a unified format so that this would be utilized to protect the cyber infrastructures. The raw data such as apache web server logs has a different representation than the email server or intrusion detection logs. There is a need to represent and integrated these various forms of data sets to generate a unified and directly digestible cyber threat intelligence [14-16].

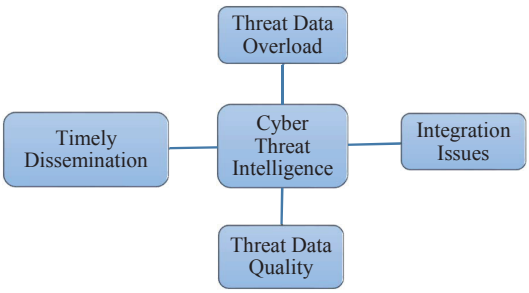

Fig. 2. Issue and Challenges of Cyber Threat Intelligence

III. Multi-Honeypot Architecture for Cyber Threat Intelligence Generation

Here in this section, the multi-platform Honeypot architecture is presented for generation of cyber threat intelligence. The similar concept is presented in T-Pot as a Multi-Honeypot Architecture [17] but their objective is for the purpose of integration and easy-to-use of Honeypot installations and maintenance. The T-Pot architecture is mainly for capturing and collection of cyber-attacks and using the ELK stack for visualization, their motivation is not to generate the cyber threat intelligence in standard actionable formats. In the proposed system, the cyber threat intelligence is generated in actionable formats after performing the data analysis techniques such as i) Sandbox Analysis ii) Report Parser to extract the IoCs iii) Attack Pattern Analysis.

The data gathered through multiple Honeypots sensors are stored, normalized and aggregated into a common archival storage. Then depending upon the data set, captured data is feed to corresponding analysis engine to further analyse the data. For network logs, the network analysis techniques mainly including the signature-based detection approaches are used. For malware samples, normally researchers are relying on anti-virus-based detection approaches which are either purely signature based or some heuristic based. Using these approaches, the window executable files are classified as benign, malicious and unknown files.

Over the period of various incidents reported in Cyber security, it is well understood that by applying a single techniques, one cannot defend against the current complex security landscape, thereby it is becoming essential to apply multiple techniques and algorithms.

In this context, the honeypots plays a vital roles because on honeypot the captured security events are treated as malicious pattern which can be taken as a feed to strengthen the detection capabilities of SIEM[18] or any defensive security solutions, In an example, if an honeypot places in an organizational network in promiscuous mode and it is scanned by the remote IP address, then that alerts can be an indicator for other devices to cross-check the infection/scan generated by that particular remote IP address. In an another example, if a malware is downloaded into a honeypot system, then completed infection life cycle is recorded including the payload contents of the malwares. The payload will act a source of evidence to support the cyber threat intelligence. But question is whether the defensive security device will take that directly, answer is no. Because the semantics of language understand by each devices is different.

A. The Design

Figure 3 depict the proposed system and its implemented design architecture of multi-platform honeypot.

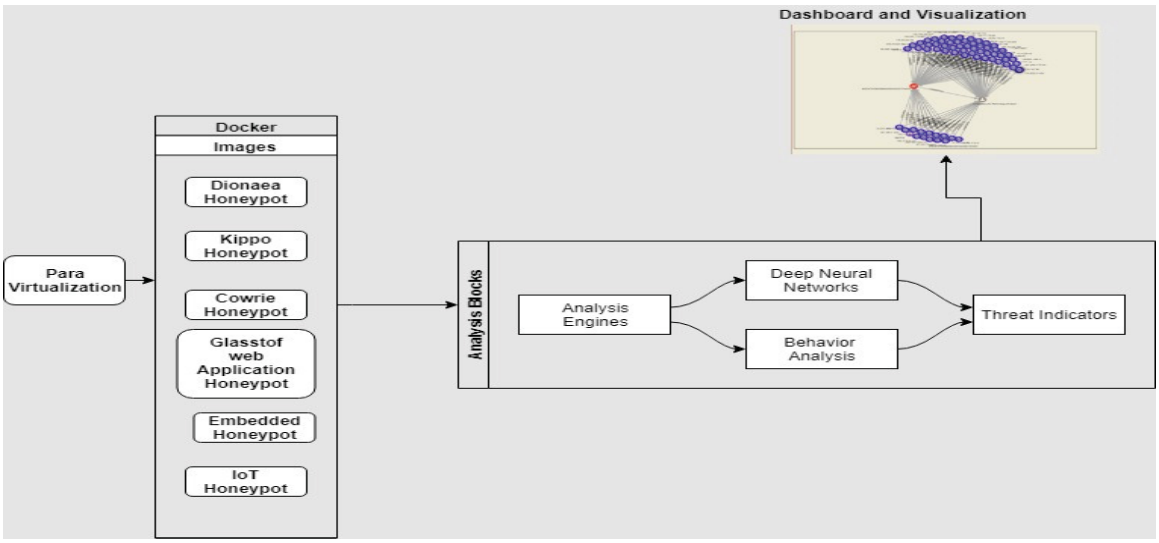

Fig. 3. System Design Architecture

Authorized licensed use limited to: Universitaetsbibliothek Innsbruck. Downloaded on March 23,2021 at 09:09:31 UTC from IEEE Xplore. Restrictions apply.

The major building blocks of the proposed system design include:

1. **Para-virtualization** Hypervisor: The software of enabling the para-virtualization containers are running on the base operating system by using the hardware resources of the base machine. The containers are created and make them live as instances of honeypot images. The honeypot as threat capturing sensors are compiled and enabled as an instance of the containers.
2. **Honeypot images**: Honeypot images are designed in such a way that that attacker will not able to fingerprint them which make them looks like a real production resource. This is inspired by the game theory approach in the field of Artificial Intelligence, like mimic the environment as per the attacker perspective. The baseline of Honeypot Containers is performed to make the balance between normal machine and hardened machine so that attacker will not be able to guess about the target is a Honeypot.
3. **Analysis Engines Blocks**: In this process of system design, the analysis engines are running integrated with the system. The various analysis techniques for known attack data detection is implemented such as Signature based, Pattern Knowledge base are running as an instance to process and alert about the attack data. And for detection and labelling of unknown attack patterns, the deep learning-based methods are applied such as CNN[19] for malware classifications.
4. **Threat Visualization and Cyber Threat Intelligence Generation:** Once the data is collected from Honeypot sensors, it is analyzed through analysis block and then there is a need to convert this raw information into actional-able formats to be directly digested by a machine in the form of cyber threat intelligence. Once such cyber threat data sharing format is STIX 2.0 as Structural Threat Information Expression formats [20]. Further to help the cyber analyst so that no need to go deep into data sets, graph-based visualization is performed using STIX visualizer.

The honeypot tools are used for attack data collector from where the data is pushed and collected to a central location. Then the collected data sets are processed and converted into set of feature matrix used by deep learning models. Apart from deep learning models, there is need to extract the features of the malware binaries either through static code analysis or dynamic analysis. The process of static code analysis is time consuming and also requires a lot of skills to perform it. Further in recent days, the static code of the malware is easily obfuscated by the malware writer. Hence dynamic behavior analysis approach is used to extract the malware patterns from behavior report which can be used for applying computational deep learning algorithms such as Convolutional Neural Network (CNN) Model.

B. Process of Honeypot Creations

Figure 4 depict the baselining process of honeypot as tool. The honeypots are created in such a way that it is a balance between the normal genuine machine and hardened machined so that attacker will not be able to fingerprint that there is a honeypot is running on the target machine.

The step by step process is defined as:

Start

{

i. Select the Honeypot
   a. Hardened the Honeypot container
   b. Select the container
   c. Perform scanning and information gathering

ii. Attack testing of Honeypot
   a. Create an isolated network
   b. Exploit Launching and logging
   c. Test and Validate the Honeypot capturing

iii. Deploy Honeypot in Production environment
   a. Make live the Honeypot containers in a production environment.
   b. Observe the attack evidences.

}

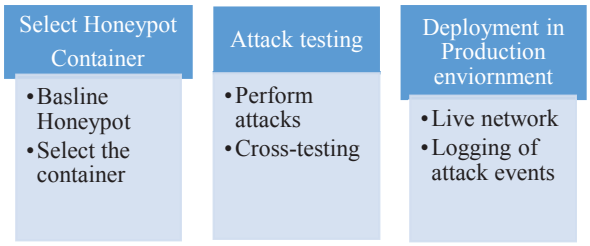

Fig. 4. Baselining of Honeypot Images

Table 1 indicate the testing and baselining results of the Cowrie Honeypot and depict the logging directory where the attacks events are logged. Cowrie Honeypot is mainly designed to catch the SSH brute force attempts performed by the attackers by providing them fake environment and directory structure.

The results indicated in table 1 depict the i) ssh login attempt and successfully logged in using root and default password, ii) username/password are logged into a directory iii) remote command executed by the attackers iv) file downloaded into a Honeypot system by attacker vi) all the commands executed by the remote attacker on the honeypots. All these attack patterns are named as attack events and the finally cyber-attack is observed as the summation of observed attack events on a Honeypots.

Authorized licensed use limited to: Universitaetsbibliothek Innsbruck. Downloaded on March 23,2021 at 09:09:31 UTC from IEEE Xplore. Restrictions apply.

The equation (1) depicts that the attack event can be any sort of patterns observed on a Honeypot.

$$A = \in E1, E2, E3 \ldots . En \quad (1)$$

The attack events can be defined as:
E1: Connection on a Honeypot from remote IP
E2: Brute Force attack observed on a Honeypot from Remote IP
E3: Command execution on a Honeypot from remote IP
……
En: Any other traced recorded on a Honeypot

TABLE 1: COWRIE HONEYPOT TESTING AND BASELINING RESULTS

Honeypot Type: Cowrie			
Sr. No	**Test Data**	**Success (yes/No)**	**Descriptions**
1	SSH Login with root	Yes	Successfully logged in with root/password after brute force attempts Not able to login with other combination of username/passwords
2	User/password logging (var/log/cowrie/ cowrie.log)	Yes	Username/passwords attempts are logging in a directory: /home/cowrie/cowrie/var/log/ cowrie/cowries.log
3	Remote commands loggings	Yes	/home/cowrie/cowrie/var/log/ cowrie/cowries.log /home/cowrie/cowrie/var/log/ cowrie/cowrie.json
4	File download	Yes	/home/cowrie/cowrie/downlo ads
5	Attacker commands	Yes	/home/cowrie/cowrie/var/log/ cowrie/cowries.log /home/cowrie/cowrie/var/log/ cowrie/cowrie.json

Table 2 below indicate the skeleton of baselining of Honeypot by providing a balance between fake and real environment so that attacker will not able to guess about he is being monitored by the honeypot. The default configurations and modified configurations are depicted in the table. We have tried to emulate the real Ubuntu like environment by including the real file system into cowrie honeypot.

TABLE 2: COWRIE HONEYPOT FINGERPRINTING & BASELINE

Honeypot Type	**Configuration file**	**Default Settings**	**Modified Settings**
Cowrie	/etc/cowrie.cfg	/etc/cowrie.cfg Hostname= svr04 openSSH_6.0p1 #listen_port = 2222 telnet=false listen_endpoints =	Hostname= Mailserver openSSH_5.5p1 #listen_port = 22 telnet=true listen_endpoints =
		tcp:22,23	tcp:2222
	/etc/userdb.txt (Configuring Users)	Default user/password list	Added some fake users
	/honeyfs/etc	Motd,issue, host.conf, hosts Hostname etc	Modified like a Real ubuntu distro
	/honeyfs/proc	cpuinfo meminfo modules mounts net version	Modified like a Real ubuntu distro
	honeyfs/proc/net	arp	Modified like a Real ubuntu distro

IV. EXPERIMENTAL RESULTS

It is very important to convert the raw attack events into standard data sharing formats that is directly digested and integrated into most of the security solution. In present scenario, large scale of data captured from various sources, of-course Honeypot is one them, the representation of those information in actionable formats is a cumbersome process. In this research, we have adopted the STIX[18] representation of captured attack events on Honeypot sensors. One example of mapping into STIX Programming language is depicted in below table 3, It is just an example to represent that attack event into standard sharing language, this may not be accurate syntax of STIX. The attack raw data is processed and mapped into STIX Objects Language which are directly digestible by a machine. Table 3 indicate the feature attributes for generation of cyber threat intelligence. First column depicts the feature name such as type the indicator- Malware/Report, then unique ID is maintained for each indicator. Created date and timestamp, name and label of the cyber-attack indicator. The name may be defined as Indicator of Compromise (IoC) The label may be treated as threat report, malware class and attack indicators.

TABLE 3: FEATURE ATTRIBUTE OF THREAT INTELLIGENCE

Feature Name	**Description**
Type	Malware/Report
ID	Unique ID
Created	Date and Timestamp
Name	IoC Scanner
labels	Threat Report/Malware/Indicator
Kill Chain Phase	Reconnaissance

One successful brute force attempt is described in table 4 which is typical brute force attack attempts before actual gaining the access of the system.

TABLE 4: BRUTE FORCE SUCCESSFUL EVENT

Feature Name	**Description**
Type	Document
Threat Score	**3.2171805**
Event ID	cowrie.login.success
Message	**LOGIN ATTEMPT [ROOT/ADMIN] SUCCEEDED**
Username	root
Source IP	**X.X.X.X**

Authorized licensed use limited to: Universitaetsbibliothek Innsbruck. Downloaded on March 23,2021 at 09:09:31 UTC from IEEE Xplore. Restrictions apply.

Password	admin
Honeypot Type	COWRIE

Table 5 depict the malware hashes captured on Honeypot sensors. The captured window executable files are maintained by assigning them the unique numeric number as depicted in column 1 in table 5. Each file is having unique hash as identification of the samples. Third column depict the date and time of the binaries captured on Honeypots. It is noted that all the depicted samples are captured on same date and year but in different timestamps. Then these samples are searched on www.virustotal.com [21] to know their labelling. It is observed here that the malware samples captured on 24-04-2018, 17:42 are not labelled by any popular anti-virus engine as per the virus total labelling, whereas the samples collected on same date, but after some later time are labelled by popular anti-virus engines. It may be note that this shall be considered as cyber-attack campaign in which firstly attacker downloaded an encrypted malware samples which may be unpacked on the Honeypot system, then further downloader is being downloaded into the system.

TABLE 5: SKELTON OF MALWARE HASHES CAPTURED W.R.T ANTI-VIRUSES LABELLING

BIN_ID	MD5	BIN_CAPTURE_TIME	AV_CLASSIFICATION
1	A48AEA09D191B2CA4E4852BC96C6365A	24-04-2018 17:42	-
2	D41D8CD98F00B204E9800998ECF8427E	24-04-2018 17:42	-
3	70EBAB2F417D274751CD1891AC16912F	24-04-2018 20:47	DOWNLOADER
4	09BD8F7A7C67DE84D043184C34389209	24-04-2018 20:48	UKPA
5	7F7CCAA16FB15EB1C7399D422F8363E8	24-04-2018 23:26	WANNACRY
7	6E9589B0446C51F742B82448816517A7	25-04-2018 06:33	WANNACRY

ACKNOWLEDGMENT

We would like to thank to Centre for Development of Advanced Computing(C-DAC), Mohali for considering National Institute of Technology(NIT),Tiruchirappalli as a model deployment location under threat intelligence generation project.

CONCLUSION AND FUTURE WORK

In this research, a multi-Honeypot Framework is presented. The design architecture helps us to understand the tools and techniques used by cyber criminals and their way of exploitation the system. The main challenge is that emulating the complete IT infrastructure is quite impossible to lure the attackers, thus there is a need to address such issues. The para-virtualisation based design architecture is designed and presented by which it is possible to emulate the multiple clusters of Honeypot sensors such as Low Interaction-Dionaea, Cowrie, IoT Honeypot, Embedded Honeypots as well as High Interaction Honeypots such as emulating the Window and Linux Operating Systems. These Honeypot sensors are implemented on a low-cost hardware by implementing the instances of the Honeypots in a para-Virtualization.

Next challenge is to analyse and represent the attack evidences into a cyber threat intelligence. The data collected through various clusters of Honeypots are analysed through data analysis techniques such attack pattern detection, behaviour analysis of malware samples, static YARA IDS signature-based detection. However, all these approaches are resource-intensive and time consuming but can be used a layer's data analysis techniques along with the presently popular Deep Learning Methods. In the end, the incubation of more automated deep learning algorithms is proposed towards the completeness of this research. More analytical capabilities need to be added in the framework to generate actionable cyber threat intelligence.

REFERENCES

[1] https://www.honeynet.org/HPW2015/interview-Lance-Spitzner-founder-Honeynet-Project

[2] https://www2.honeynet.org/projects/old/honeywall-cdrom/

[3] http://www.keyfocus.net/kfsensor/

[4] https://www.div0.sg/single-post/dionaea-malware-honeypot

[5] http://www.honeyd.org/

[6] https://github.com/desaster/kippo

[7] https://github.com/threatstream/mhn

[8] Patel, R., & Thaker, C.S. (2012). Zero-Day Attack Signatures Detection Using Honeypot.

[9] Kumar, Sanjeev & Singh, Paramdeep & Sehgal, Rakesh & Bhatia, J.. (2012). Distributed Honeynet System Using Gen III Virtual Honeynet. International Journal of Computer Theory and Engineering. 537-541. 10.7763/IJCTE.2012.V4.527.

[10] Bhatia J.S., Sehgal R.K., Kumar S. (2011) Honeynet Based Botnet Detection Using Command Signatures. In: Al-Majeed S.S., Hu CL., Nagamalai D. (eds) Advances in Wireless, Mobile Networks and Applications. ICCSEA 2011, WiMoA 2011. Communications in Computer and Information Science, vol 154. Springer, Berlin, Heidelberg.

[11] Thomas Antoniou Ailianos et.al, SIEM Optimization Using Honeypots, 2014.

[12] ZHUMANGALIYEVA, NAZYM KENZHEGALIYEVNA et al. Analysis of intrusion detection systems. Journal of Mathematics, Mechanics and Computer Science, [S.l.], v. 103, n. 3, p. 55-74, oct. 2019. ISSN 2617-4871.

[13] Conti, Mauro & Dargahi, Tooska & Dehghantanha, Ali. (2018). Cyber Threat Intelligence: Challenges and Opportunities. 10.1007/978-3-319-73951-9_1.

[14] Saxe, Joshua & Berlin, Konstantin. (2015). Deep neural network based malware detection using two dimensional binary program features. 11-20. 10.1109/MALWARE.2015.7413680.

[15] Sean Barnum, Standardizing Cyber Threat Intelligence Information with the Structured Threat Information eXpression (STIX™), 2014

[16] Xiaojing Liao, Kan Yuan, XiaoFeng Wang, Zhou Li, Luyi Xing, and Raheem Beyah. 2016. Acing the IOC Game: Toward Automatic Discovery and Analysis of Open-Source Cyber Threat Intelligence. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16). ACM, New York, NY, USA, 755-766. DOI: https://doi.org/10.1145/2976749.2978315.

[17] https://dtag-dev-sec.github.io/mediator/feature/2015/03/17/concept.html

[18] https://en.wikipedia.org/wiki/Security_information_and_event_management.

[19] https://www.tensorflow.org/tutorials/images/cnn

[20] https://oasis-open.github.io/cti-documentation/stix/intro.html

[21] https://www.virustotal.com/gui/

Authorized licensed use limited to: Universitaetsbibliothek Innsbruck. Downloaded on March 23,2021 at 09:09:31 UTC from IEEE Xplore. Restrictions apply.