Tracing Intruders Using Honeypot With Metasploit Contents

Alin Boby
Institute of Information and Communication Technology
Bangladesh University of Engineering and Technology
Dhaka, Bangladesh
alinboby@gmail.com

Hossen Mustafa
Institute of Information and Communication Technology
Bangladesh University of Engineering and Technology
Dhaka, Bangladesh
hossen_mustafa@iict.buet.ac.bd

***Abstract* — In recent years, it is impossible to say that a system is fully secure with no vulnerability. After exploiting a system, hacking professionals use techniques to hide their real identities and sweep out log records before leaving in such a way that security experts cannot trace them. Researchers and network administrators have applied several approaches to monitor and analyze malicious traffic for malicious content by monitoring network components, aggregating IDS alerts, and using different types of honeypots. However, there is limited effort to trace an attacker. In this paper, we propose a web application honeypot that contains undetectable encoded metasploit contents and integrated into real web application from different location. When an intruder accesses these contents, exploited code in the contents will run on intruder's system and our system will get a hidden backdoor through Metasploit console immediately. We collect and store all activities and resources information of the intruder system into database. We implemented the system as a case study and by tracing the attackers, our proposed system was able to successfully detect 103 attackers and collect information of 67 attackers within a short period of time.**

Keywords- Computer Security; Honeypot; Attacker Detection

I. INTRODUCTION

In the era of information, black or gray hat hackers often target web applications that are vulnerable to attacks. Many advanced attacks can be launched by the attacker using simple tools. According to the "Web Application Vulnerability Report 2015" of Acunetix [1], major percentage of websites are vulnerable to Cross Site Scripting (XSS), Denial of Service (DoS), Secure Sockets Layer (SSL) related vulnerabilities, SQL Injection, etc., as shown in Fig. 1. Some of these attacks can be prevented by strengthening the security of the system. Intrusion Prevention System (IPS) with penetration testing [2], firewall, vulnerability scanner, etc., are being used to identify major security lacking in a system [3]. However, these were not designed to look at the behavior of millions of concurrent sessions as a whole, but only to examine individual suspected sessions. This eliminates the ability to identify an attack composed of millions of valid requests. Anti-Virus (AV) software are also used which can detect and prevent known attacks but these signature based detection mechanisms have limitation to capture new hacking techniques that use modification of the code [4] or zero-day exploits [5]. However, people are still using this type of software daily as there is no other cheaper and/or better alternatives. Signature-based detection technique is used in almost all commercial and non-commercial AV software, but these cannot be completely effective against zero-day malware [6]; many evaluations conducted by renowned security firms confirmed this [7]. These evaluations often employ sophisticated malware, involve elaborated schemes, and require more resources than generally available to an average person to replicate. Some research papers investigate the creation of simple zero-day malware that can comprehensively exploit hosts and evade the installed AV software. Also, researchers and network administrators have applied several approaches to monitor and analyze malicious traffic for malicious content by monitoring network components, aggregating IDS alerts, and using different types of honeypots. While there is significant effort in attack detection, there is limited effort to trace an attacker.

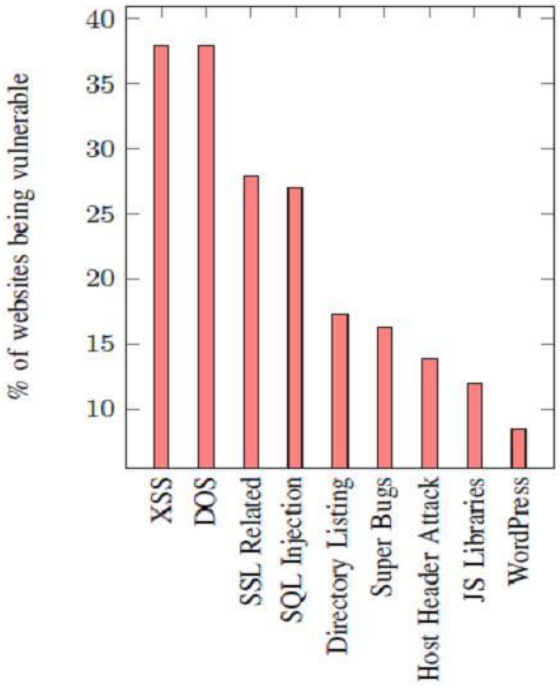

Fig. 1: Top Web Vulnerabilities

In recent years, researchers have been working on designing different types of honeypots to trace attackers [8][9]. But many attacks by undetectable exploits and proxy IP are not detectable through these proposed systems. However, it is possible to detect such advanced attacks when the honeypot can establish a direct access to the attacker's system. In this paper, we propose a web application honeypot that contains encoded metasploit contents integrated into a web application. These contents can be exploited by an attacker using brute-force or SQL injection attacking method. Our proposed system diverts the attackers to the honeypot with metasploit contents; when the attackers copy any of these contents to their system and try to access any of the contents, exploited code in the content will give us backdoor to control and trace the resources, and activities of the attacker. We implemented our proposed system as a case study and deployed in the Internet like a regular web application. By tracing the attackers, our proposed system was able to detect 103 attackers and collect information of 67 attackers in a short period of time. Our proposed system can collect and store all activities and resources information of the intruder system into database. Analysis of the stored information can give insights into attacking methodologies, techniques, and levels; such insights can help security researchers to design more secured systems.

The rest of the paper is organized as follows: in Section II, we discuss backgrounds and related works of honeypots; we present the proposed model in Section III and discuss implementation details and results in Section IV; finally, we conclude the paper in Section V.

II. BACKGROUND AND RELATED WORKS

A honeypot is a server that is configured by mirroring a real production system to lure and detect potential attackers who seek to gain unauthorized access to information systems. It is used for trapping intruders by detecting, deflecting, or reducing risky behavior in the information system. It consists of a computer, a network site, and data which appears to be a part of a network, but it is actually an isolated and monitored system. A honeypot can record all actions and interactions with users. Since honeypots do not provide any legitimate services, all activities are considered unauthorized and malicious. It is used to study activity traces left by attackers and subsequently rectify the system security to prevent future attacks. Generally, a honeypot consists of a computer, applications, and data that simulates the behavior of a live system but acts as a decoy [10]. There are two broad categories of honeypots available today based on their level of interaction: high-interaction honeypot and low-interaction honeypot. Some authors classify a third category, medium-interaction honeypots [11], that has higher interaction from low-interaction honeypots but lower than high-interaction honeypots. Based on planned use, honeypots can be divided into production honeypots and research honeypots [12]. Christian Seifert proposed HoneyC [8], a low interaction client honeypot; it uses emulated clients that are able to solicit as much of a response from a server that is necessary for analysis of malicious contents. HoneyC consists of three components: Visitor, Queuer, and Analysis Engine. Dionaea [9] is a low-interaction server-side honeypot which emulates vulnerabilities in Windows services targeted by malware. Glastopf [13] is a Python web application honeypot that emulates web vulnerabilities and handles unknown attacks of the same category. Kippo [14] is a medium-interaction honeypot that is built using SSH and python; it can record brute force attacks and replay attacker's interactions in emulated shell on the fake SSH server during attacker's attempt to guess login credentials of an SSH server. Thug [15] is a Python client-side low-interaction honeypot that emulates a web browser; it can interact with the malicious website to explore its exploits and malicious artifacts.

Last few years, many researchers worked widely with honeypot. Several models and designs using honeypot have been proposed for security against various attacks. Richardson et al. [16] proposed honeypots to protect back-end servers from attacks. Back-end servers handle more complex request and manage valuable information. They propose a network model that grants isolation to a back-end server from unauthorized traffic, blacklisting of misbehaving clients. Thus, it can limit back-end DoS attacks. The back-end server is isolated from the network by a separate connection to a masquerading router that changes all IP and MAC entries on packets exiting the router to the current values for the router itself. This layer of indirection prevents the discovery of the actual MAC address of the back-end server's network card. This indirection also facilitates the masquerading router to allow legitimate traffic to pass to the back-end server or to the attached honeypot. The DoS attacks on back-end servers can be reduced by limiting further packets from any traffic arriving at the honeypot. This can be supplemented by blacklisting clients that exceed their permissions. Khattab et al. [17] use roaming honeypot that allows the locations of honeypots to be unpredictable to mitigate service-level DoS attacks. The servers in the honeypot are changed frequently so that hackers cannot identify and shut down the honeypots. They propose their roaming honeypots scheme to mitigate the effects of service-level DoS attacks, in which many attacking machines acquire service from a victim server at a high rate. The locations of honeypots are continuously and unpredictably changing within a pool of back-end servers. Each server alternates between providing the service and acting as a honeypot in a manner unpredictable to attackers. The roaming honeypots scheme detects and filters attack traffic from outside a firewall, and mitigates attacks from behind a firewall. Against service-level attacks, the honeypot provides filtering effect which secures the service against attacks launched from outside a firewall (external attacks) and connection-dropping effect which mitigates attacks launched from behind the firewall (internal attacks).

Khattab et al. [18] extended the work done in [17] to propose a scheme of honeypot back-propagation to backtrack and find the source of the DoS attack. They offer honeypot back-propagation, a hierarchical trace back scheme which can traces back and suspend sources of attacks without major effect on the performance of legitimate traffic streams. The core idea of the proposed scheme is that when a roaming honeypot accepts packets, it starts a trace back process by notifying autonomous systems across the path(s) towards attack sources. Within each autonomous system, attack hosts are recognized, and filtering rules are set up to block their network access. Honeypot back-propagation provides a high payoff in this

regard. First, it uses accurate attack signatures, and thus, reduces collateral damage. Second, it helps ISPs to accurately locate compromised hosts on their networks. Third, incremental benefits are possible with partial deployment of honeypot back-propagation because network messages involved in the scheme can be piggybacked on Border Gateway Protocol (BGP) messages to traverse legacy networks. They address low-rate attacks by a progressive honeypot back-propagation scheme. They evaluated their schemes analytically and using NS-2 simulations. The results show that attacks can be stopped within seconds under many scenarios.

Anirudh et al. [10] deployed a honeypot for an Internet of Things (IoT) system to block DoS attacks from malicious attackers. Their proposed system also collects information on the attacker. Generally, attacks are concentrated towards the main server rather than the individual devices connected in the system. In their model, all requests from clients are passed to the IDS. Legitimate requests pass through the IDS onto the server. If the IDS detect any anomalies in the requests, the requests are passed onto the honeypot and the information related to the attacker are stored as logs in a database. It blocks the client completely off the server if verification fails. Otherwise, if the client passes the verification, the data is passed onto the server.

Moore et al. [19] used honeypot technique to detect ransomware. Ransomware attack often would progress alphabetically through mapped drives; so, they map an early letter of the alphabet to the honeypot area. They used two approaches to detecting ransomware: initially, a honeypot folder monitored with a File Server Resource Manager (FSRM), followed by observing changes to the Windows Event Logs. EventSentry is configured following the instructions to set up file auditing to event 4663: an attempt is made to access an object. Actions are setup to follow the three tiers: email, stop server service and finally shutdown the service. These would be linked to filters, with the required thresholds to trigger the action. Determining this threshold needs some consideration but for the experiment, a 10 second period is considered. In the experimental setup, normal activity is monitored and averaged over a day.

Prevention of zero-day attack using methods for isolating the malicious traffic by using a honeypot system was deployed by Musca et al. [6]. They build the honeypot to collect information. Instead of building firewalls and writing intrusion detection and prevention systems, they lured in attackers and study penetration methods. They used an isolated environment to deploy the honeypot system that only tracks malicious activity because it is not used as a production system. Using a protected machine, they capture the collected data through an encrypted tunnel. The attack analysis framework automatically detects unknown attacks and generates signatures for the Snort intrusion detection or prevention system.

Danchhenko et al. [20] proposed honeypot system to detect suspicious activity on Remote Protocol. They have examined two remote access protocols: Remote Desktop Protocol (RDP) and Virtual Network Computing (VNC) with Remote Frame Buffer (RFB) protocol. These protocols operate on a client-server scheme. Using this method, they propose to obtain data about attacks on servers held by malefactors, for research and further development of the security architecture.

Low-interaction aggressive web application honeypot uses JavaScript into the browser's response to trace attacker's information [21] based on their IP addresses when XSS or SQL injection attack happens. Some client-side attacks can be predicted by behavior analysis using previously recorded client honeypot data [22].

Recently, Naik et al. [23] presented a honeypot that can discover and predict an attempted fingerprinting attack by using a Principal components analysis and Fuzzy inference system. Their proposed system is successfully tested against the five popular fingerprinting tools: Nmap, Xprobe2, NetScanTools Pro, SinFP3 and Nessus.

Ahmed et al. [31] proposed to use honeypot in cyber deception system. They propose a proactive and reactive phase deceptive honeypot allocation policy to design a cyber deception mechanism.

As we can find from the above discussion, there are several works to identify different types of attacks or trace back attackers using honeypot to prevent attack. However, in this research, we aim at hacking back the attacker system using honeypot to get more insight of the attackers.

III. PROPOSED HONEYPOT MODEL

In this paper, we design a web-based system using honeypot to identify attackers and trace their resources, and activities by getting access through reverse exploits. The proposed honeypot system consists of four (4) components as discussed in the following.

i. Attack Detection Module (ADM) detects attacks and generates log records in honeypot database.

ii. Web Application Honeypot (WAH), placed in a different location from real server, contains metasploit contents.

iii. Metasploit Content Generator (MCG) automatically generates a number of metasploit contents for web application honeypot.

iv. Data Capture and Analysis Module (DCAM) extracts attacker system information and stores into database for further analysis.

A. Workflow of The Proposed System

The workflow of the proposed system is shown in Fig. 2; the workflow has 6 steps.

1) MCG generates given number of encoded and undetectable metasploit files and transfers to admin panel directory in WAH.
2) Directories, pages, and links of WAH are merged into RWA to make it seem real to the attacker.

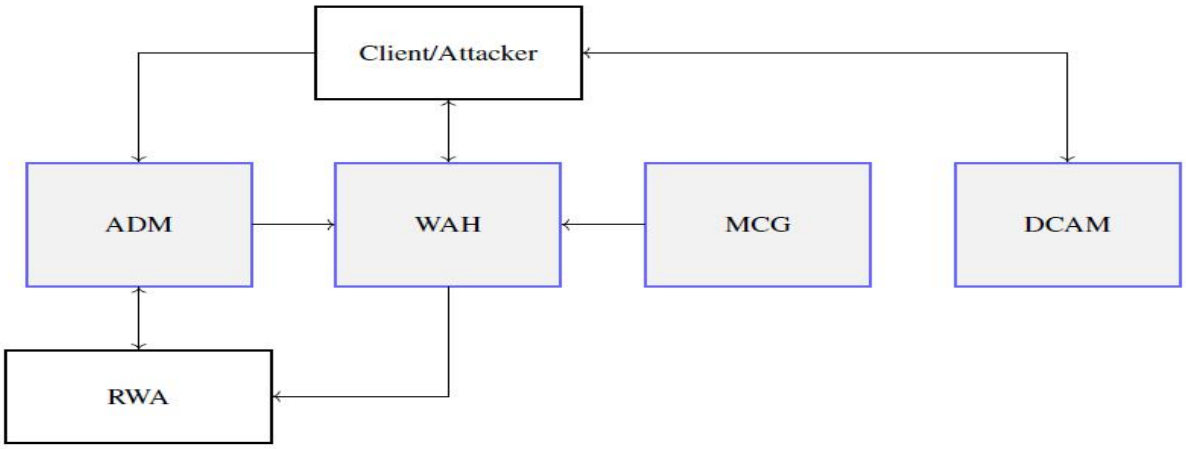

Fig. 2: Workflow of the proposed system

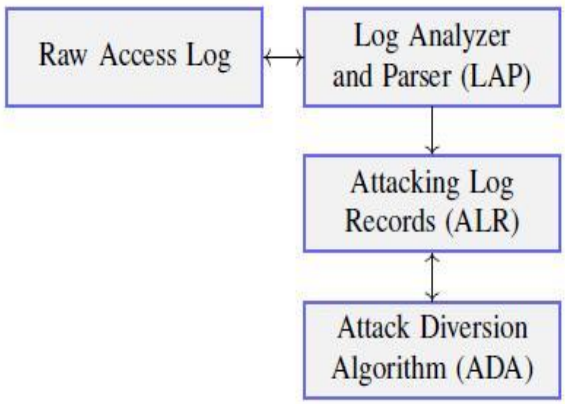

Fig. 3: Attack Detection Module

3) ADM stores attacking IP address into ALR from WAH web server access log. ADM detects attack while client is trying to access the Real Web Application (RWA).
4) ADM diverts either attacker into WAH or real client to RWA login panel. It also diverts a client from RWA for maximum number of login failure.
5) Attackers can login to WAH and may copy or open metasploit contents available in WAH admin panel.
6) DCAM extracts attacker resource information as well as activities and stores data into database while attacker is opening any of these metasploit contents.

We discuss the details of each module in the following.

B. Attack Detection Module

Attack Detection Module (ADM) contains Log Analyzer and Parser (LAP), Attacking Log Records (ALR) and Attack Diversion Algorithm (ADA) as shown in Fig. 3. LAP extracts log records except login panel access from raw access log file in Web Application Honeypot (WAH). Real Web Application (RWA) also contains raw access log file where attack detection process may cause more false positive alerts; so, we only consider raw access log from WAH to update ALR. Login page link in WAH contains ADA that can check IP address from ALR and divert attackers to fake login page in WAH. ADA would pass legitimate users to the RWA.

1) Log Analyzer and Parser: ADM focuses on SQL injection attack because it is the most common and popular vulnerability into web application [24]. An SQL injection attack comprises of injecting a deformed SQL query into a web application via client-side input. Several tools are used to create SQL injection attack; attackers use web analysis tools to check feasibility of SQLi attack. Web analysis and vulnerability scanner tools are also used to scan open port and directory listing. LAP gets access logs for the attack attempts. ALR generated by LAP contains exceptional log that includes port scanning, dictionary attack, or SQLi attack. LAP extracts data from raw access log, web server log file, and blacklisted IP list and stores unique IP, attacking type and access details into ALR to mark attacking IP addresses.

2) Attacking Log Records: While analyzing raw access log and web server log file, we consider the following to update attacking log records:

- Log contains ports that are not permitted but tried to be accessed.
- Log contains IP address that are already blacklisted in attacking log table.
- Log URL contains SQLi attack, directory listing and dictionary attack.

3) Attack Diversion Algorithm: While a user accesses the login page, ADA detects attack based on marked IP addresses by the ALR. Initially, it diverts attacker to login panel in honeypot to get user credentials and updates the user table in honeypot database. After checking login information, ADA

gives the attacker permission to access into honeypot admin panel that contains metasploit contents.

If the IP address from user session is not found in ALR, it treats the user as a real client and redirects the client to login page in RWA to get user credentials. The last portion of algorithm checks the user login information and lets client access the admin panel in RWA if credentials are matched to information in RWA database. If login credentials are wrong, ADA also counts login attempts and checks number of attempts to maximum attempt limit. After maximum tryouts, ADA treats user as an attacker for brute-forcing and adds attacker IP address into ALR. Then, ADA diverts attacker directly to admin panel in honeypot mimicking a successful login. In such case, ADA updates the ALR with attack information. ADA process is shown in Algorithm 1.

C. Web Application Honeypot

The proposed honeypot is a web application honeypot as shown in Fig. 4 and it is integrated with ADA in the login link script. WAH has a web server where login page is connected to WAH database which is like RWA. Admin panel of WAH contains different metasploit contents in PDF and JPEG format. WAH web server has some directories and fake page files that look like real links. The fake contents are generated by metasploit content generator. The idea is to lure the attacker to download and open the metasploit contents.

Algorithm 1: Attack Detection Algorithm

Input: IP_u, U_{name} and P_{user}
Get IP_u from LLS
Search IP_u in ALR
if IP_u found in ALR then
    Divert to LP_{wah}
    Get U_{name} and P_{user}
    if P_{user} matches to P_{wah} from DB_{wah} then
        Divert U_{name} into AP_{wah}
    else
        Show invalid login message
        go to 5
    end if
else
    Divert to LP_{rwa}
    Get U_{name} and P_{user}
    if P_{user} matches to P_{rwa} from DB_{rwa} then
        Divert U_{name} into AP_{rwa}
    else
        Show invalid login message
        Count A_n
        if $A_n >$ then
            Add IP_u into ALR
            go to 7
        else
            go to 14
        end if
    end if
end if

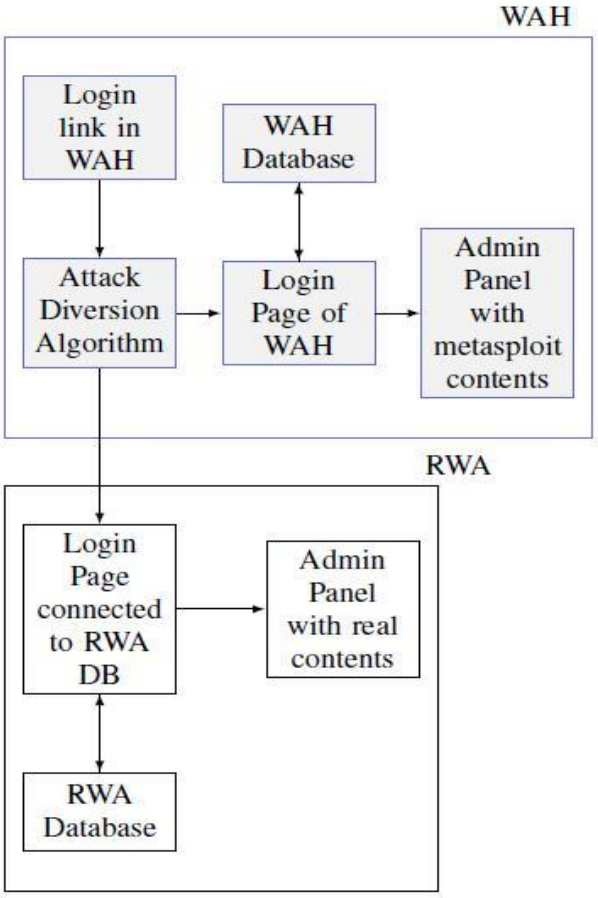

Fig. 4: Proposed model of honeypot

D. Metasploit Content Generator

We designed Metasploit Content Generator (MCG) to automatically generate metasploit contents for WAH. It uses Metasploit Framework (MSF) to generate single exploit. By this generator, we can generate different types of metasploit files which will be used in different operating system of attacker. To create undetectable metasploit content, we update the signature of every exploited file. To change signature, we decode a file, add some random comments in it and finally, encode it again. This allows us to bypass the AV scanners when attackers download the content.

E. Data Capture and Analysis Module

Data Capture and Analysis Module (DCAM) is designed to extract resources and activities information from the attacker system. Every code for specific exploit type contains auto-script to run meterpreter automatically with specific port, host, and another auto-script. We designed two auto-script: 1st one in meterpreter shell script is configured by specific MSF command list and 2nd one contains meterpreter command list that extracts information when the attacker opens metasploit content. DCAM also stores extracted information into database dynamically for further analysis about attacker motive and skill. DCAM architecture is shown in Fig. 5.

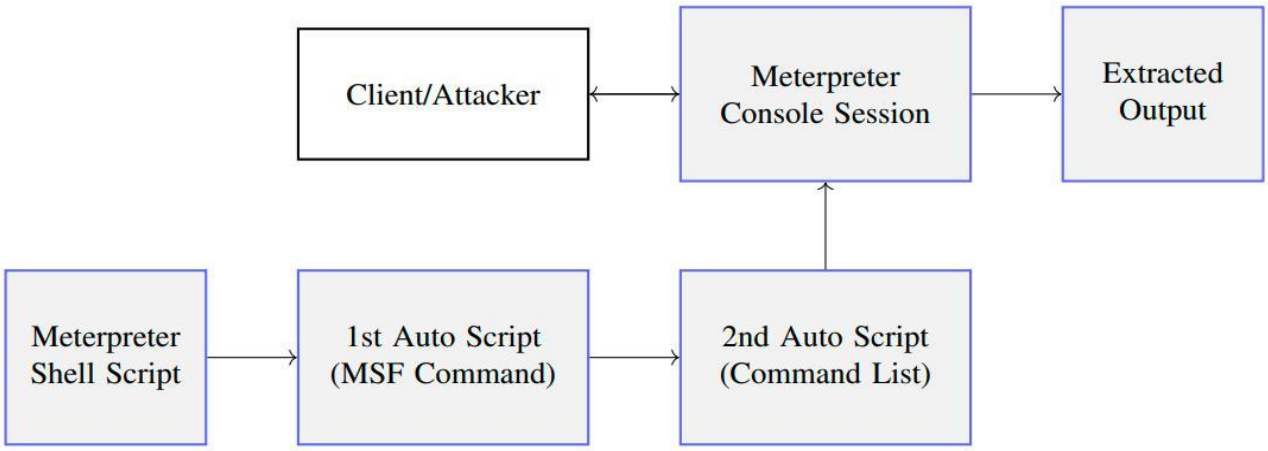

Fig. 5: Data Capture and Analysis Module

IV. SYSTEM IMPLEMENTATION AND RESULTS

We implemented the proposed Web Application Honeypot (WAH) on a server and routed the DNS entry of the real application to the honeypot server. WAH is the most important part of the system with other three components: ADM, MCG and DCAM. We discuss implementation strategies for these components in the following.

A. Implementation of ADM

Standard web servers like Apache [25] and IIS [26] generate log in Common Log Format (CLF). The CLF log file contains a separate line for each HTTP request, can be readily analyzed programmatically. A line in a file stored in the CLF is composed of several tokens separated by spaces as shown below:

host identifier auth-user date-time request status bytes

Several logs are maintained on a web server which includes access log, error log, php error log, SSL request log, etc. While simulating SQLi attack on WAH web server, we observed that access log file contains 'UNION' in case of attack. Some captured log records are shown in Fig. 6.

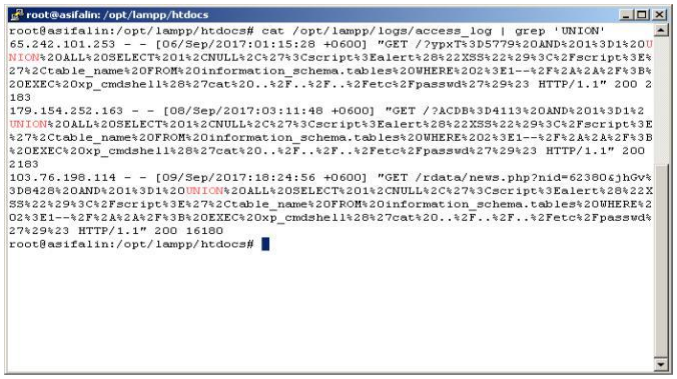

Fig. 6: Analyzing access log to detect SQLi attack

In this figure, the entry with IP address 103.76.198.114 was from our simulation but the other 2 IP addresses, 65.242.101.253 and 179.154.252.163, were for actual SQL Injection attacks. For log analyzer and parser (LAP) in attack detection module (ADM), we developed a PHP script that can analyze log files to detect web attacks. LAP script extracts attacking logs from access log, extracts data and stores marked IP address, and details into ALR in WAH.

B. Implementation of WAH

We deployed our proposed WAH in a server that contains metasploit contents and Content Management System (CMS). In this system, we used Apache as Web server, MySQL as Database server and Wordpress as CMS as it is very popular. MCG and DCAM are also deployed in this server for generating the metasploit files and creating a terminal to check if any attacker opens metasploit files.

We create different types of metasploit contents using MSF framework. Kali [27] recommends that we use a robust, secure terminal emulator when operating the command-line interfaces. It may be konsole [28], gnome-terminal, and recent versions of PuTTY. We used several tools and web applications for testing our proposed system and compared it with other existing system. We implemented WAH in a Linux server with public IP address. To integrate WAH into RWA, we configured DNS record of RWA that assigns a subdomain to WAH where the WAH contains all metasploit contents. To pose the system as vulnerable, parameter passing is opened in the RWA; such vulnerability will lure the attacker to the honeypot.

Attackers generally use tools to analyze host and can easily find out if any framework is used to develop the web application in host [29]. Also, determining the operating system of a host is important to every attacker for listing possible security vulnerabilities, defining the available system

calls to set the specific exploit payloads, and for many other OS-dependent tasks. We added meta contents with framework information and OS information in WAH so that an attacker easily finds out and launches attack to break into the system. The first target of an attacker is to get user login information from web application. A fake table containing fake user and password information in WAH database is created. To make it seem valid, fake user login table has the real table structure but with fake information.

We implemented Attack Diversion Algorithm (ADA) using PHP in the login page of WAH. DNS entry of the RWA is configured to WAH and the ADA redirects user to either RWA login page or WAH login page. ADA is also implemented in RWA login page to detect continuous login attempt. When attack is detected, ADA inserts client IP address into ALR for dictionary attack and diverts user directly to WAH admin panel.

C. Implementation of MCG

To generate metasploit content automatically, we used metasploit framework, NXCrypt [30] and PHP. Here, the metasploit framework is a Ruby-based, modular penetration testing platform; it provides libraries, tools as well as complete environment to generate metasploit contents that can be used to evade detection. Using our MCG, we generate metasploit contents and transfer these contents into WAH admin panel. We generate fully undetectable contents with defined exploit type, exploit name, number of exploits, IP address and port. We used NXCrypt and PHP to automate the MCG process.

Antivirus software companies usually search for signature of malware for identification. When they find a malware, they add its signature to their virus/malware database along with the corresponding disinfection methods and when it next encounters that malware, the software alerts the computer owner. Obviously, zero-day exploits, or malware that is new and never been seen by the Antivirus software, will not be detected by such a detection scheme. Another method of getting past the Antivirus software is to just change the signature of the malware. In other words, if we can change the encoding of the malware without changing its functionality, it should sail right past the Antivirus software without detection.

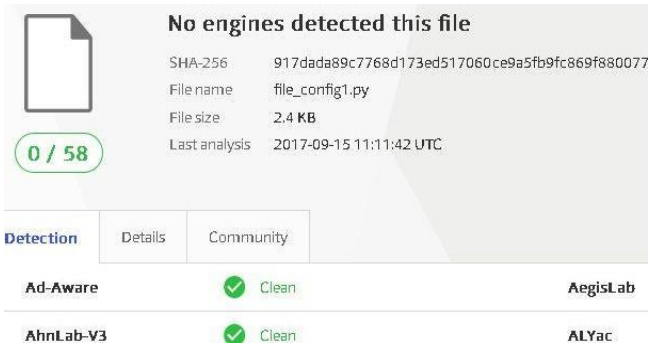

Fig. 7:MCG generate undetectable exploit by AV

We can re-code any malware and get this desired result. To make metasploit files as undetectable, we generate encoded exploit by our own bash script that uses MSFvenom framework, NXcrypt and PHP script. Metasploit contents generated by our MCG are tested by using https://www.virustotal.com for 58 antivirus systems and the Clean result is shown in Fig. 7.

D. Implementation of DCAM

We used meterpreter module of the metasploit framework for implementing DCAM. We developed a program using shell script that runs commands in meterpreter console. To configure meterpreter console for opening session, first auto-script containing MSF command is executed by shell script. Another auto-script in MSF command containing meterpreter auto command list with spooling facility is automatically executed to extract attacker resources information when an attacker accesses metasploit contents. Once the data is collected, it is automatically stored in the DCAM database. We used MySQL for our implementation.

E. Experimental Analysis

We deployed the implemented system in Internet similar to a real web application. We discuss our findings in the following for a deployment period of 2 months.

1) Detection of Attacks: We found that user log data in WAH login panel contains total 422 log records where distinct IP address count is 103. That means 24.41% different IP addresses came to WAH panel several times. The chart shows the comparison in Fig. 8.

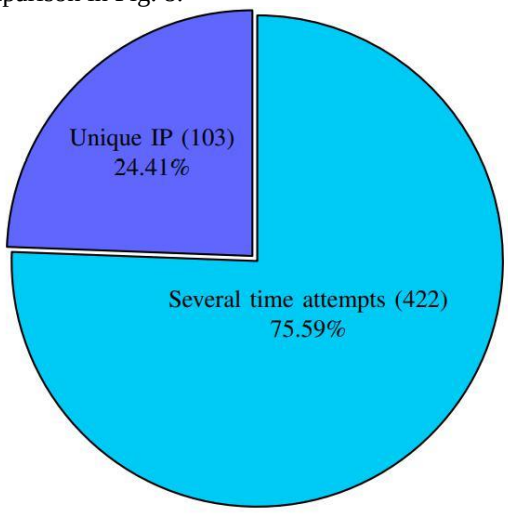

Fig. 8: Unique IP addresses found in WAH login panel

Out of the 103 unique attackers, LAP detected 3 attacks as SQLi attack from server access log and stored these 3 records into ALR as SQLi attack. We have manually analyzed server access log and found out 3 records only. That means that our LAP is 100% successful in detecting any kind of SQLi attacks. Since 3 attacks are SQLi attacks among 103 records,

remaining 100 attacks are from Dictionary and brute forcing attacks. DCAM shows that 67 attackers out of 103 attackers are caught by metasploit contents in WAH; these attackers transferred metasploit contents and tried to open the contents. Remaining resistant attackers did not either transfer or open metasploit contents. Fig. 9 shows the successful ratio for tracing the attacker out of all attacks.

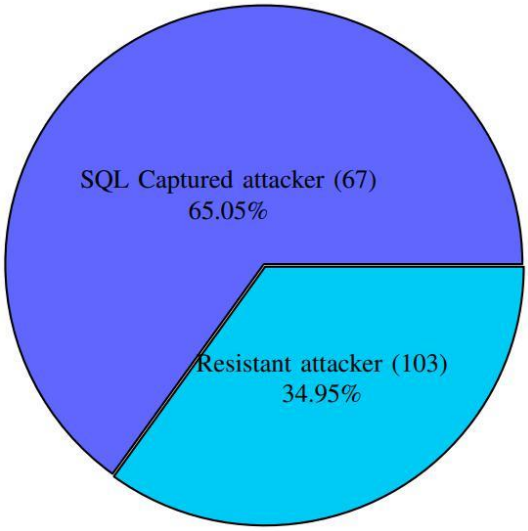

Fig. 9: Successful ratio for capturing the attacker

2) Attacker Resources Information in DCAM: The capturing rate of different types of extracted information from the 67 traced attacker is shown in Fig. 10. As we can see, our proposed system was able to get sysinfo (system statistics), ps (selection of the active processes), checkcm (if system is virtual), dumplinks (get recent document), etc., information from the attacker machine. It automatically stored extracted information into DCAM database for further analysis. The figure shows that we were able to get system info successfully for 67% attackers. It also shows that metasploit contents in WAH are successfully generated by MCG and DCAM can extract information if attacker opens metasploit contents generated by MCG. The extracted information can be very essential for researching attacker motivation, activities, and skills.

V. CONCLUSION

Several types of honeypots were proposed by researchers to detect suspicious activities. Few models were suggested to prevent the major attacks like DDoS or SQL injection attack. Unlike existing works, we propose the concept of reverse hacking by web application honeypot to trace attacker and deeply examine attackers' system resources, and their motivation. Our proposed system, containing four components, has been implemented successfully to detect SQLi, brute-forcing and dictionary attack, and then, trace resource information and activities log from attacker system.

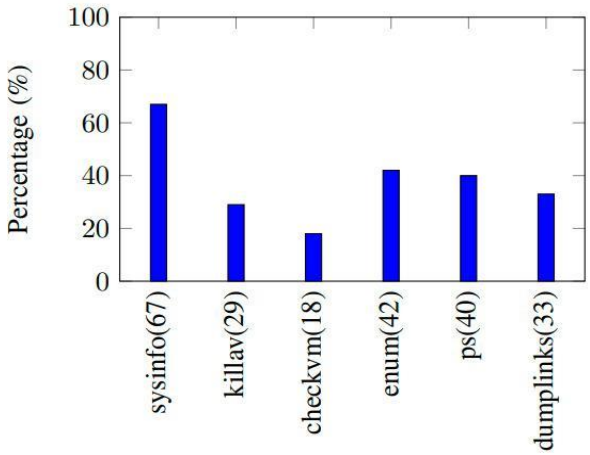

Fig. 10: Different Types of Extracted Information

Experimental results show that our proposed system can successfully divert an attacker to the honeypot and can trace attacker resources using metasploit contents.

In future, we would like to analyze log data to detect more attacks, generate various types of metasploit contents for different platform which must look like unique and adding more vulnerabilities in WAH in order to attract more attackers of various expert levels. Additionally, more information can be captured from the attacker's system.

REFERENCES

[1] "Acunetix Web Application Vulnerability Report 2015," http://www.acunetix.com/acunetix-web-application-vulnerability-report-2015/.

[2] F. Holik, J. Horalek, O. Marik, S. Neradova, and S. Zitta, "Effective penetration testing with metasploit framework and methodologies," in Computational Intelligence and Informatics (CINTI), 2014 IEEE 15th International Symposium on. IEEE, 2014, pp. 237–242.

[3] Y. Stefinko, A. Piskozub, and R. Banakh, "Manual and automated penetration testing. benefits and drawbacks. modern tendency," in Modern Problems of Radio Engineering. Telecommunications and Computer Science (TCSET), 2016 13th International Conference on. IEEE, 2016, pp. 488–491.

[4] N. Thamsirarak, T. Seethongchuen, and P. Ratanaworabhan, "A case for malware that make antivirus irrelevant," in Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2015 12th International Conference on. IEEE, 2015, pp. 1–6.

[5] H. Gupta and R. Kumar, "Protection against penetration attacks using metasploit," in Reliability, Infocom Technologies and Optimization (ICRITO)(Trends and Future Directions), 2015 4th International Conference on. IEEE, 2015, pp. 1–4.

[6] C. Musca, E. Mirica, and R. Deaconescu, "Detecting and analyzing zero-day attacks using honeypots," in Control Systems and Computer Science (CSCS), 2013 19th International Conference on. IEEE, 2013, pp. 543–548.

[7] "Acunetix Web Application Vulnerability Report 2016," http://www.acunetix.com/acunetix-web-application-vulnerability-report-2016/.

[8] C. Seifert, I. Welch, P. Komisarczuk et al., "Honeyc-the low-interaction client honeypot," Proceedings of the 2007 NZCSRCS, Waikato University, Hamilton, New Zealand, vol. 6, 2007.

[9] T. K. Lengyel, J. Neumann, S. Maresca, B. D. Payne, and A. Kiayias, "Virtual machine introspection in a hybrid honeypot architecture." in CSET, 2012.

[10] M. Anirudh, S. A. Thileeban, and D. J. Nallathambi, "Use of honeypots for mitigating dos attacks targeted on iot networks," in Computer, Communication and Signal Processing (ICCCSP), 2017 International Conference on. IEEE, 2017, pp. 1–4.

[11] O. Ayeni, B. Alese, and L. Omotosho, "Design and implementation of a medium interaction honeypot," International Journal of Computer Applications, vol. 70, no. 22, 2013.

[12] J. Riden and C. Seifert, "Different Kinds of Honeypots," https://www.symantec.com/connect/articles/guide-different-kinds-honeypots/, 2008.

[13] L. Rist, S. Vetsch, M. Kossin, and M. Mauer, "Know your tools: Glastopf-a dynamic, low-interaction web application honeypot," The Honeynet Project, vol. 4, 2010.

[14] C. Valli, P. Rabadia, and A. Woodward, "Patterns and patter-an investigation into ssh activity using kippo honeypots," 2013.

[15] A. Dell'Aera, "Thug: a new low-interaction honeyclient," https://github. com/buffer/thug, 2012.

[16] T. Richardson, "Preventing attacks on back-end servers using masquerading/honeypots," in Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2006. SNPD 2006. Seventh ACIS International Conference on. IEEE, 2006, pp. 381–388.

[17] S. M. Khattab, C. Sangpachatanaruk, D. Mosse,´ R. Melhem, and Znati, "Roaming honeypots for mitigating service-level denial-of-service attacks," in Distributed Computing Systems, 2004. Proceedings. 24th International Conference on. IEEE, 2004, pp. 328–337.

[18] S. Khattab, R. Melhem, D. Mosse,´ and T. Znati, "Honeypot back propagation for mitigating spoofing distributed denial-of-service attacks," Journal of Parallel and Distributed Computing, vol. 66, no. 9, pp. 1152–1164, 2006.

[19] C. Moore, "Detecting ransomware with honeypot techniques," in Cybersecurity and Cyberforensics Conference (CCC), 2016. IEEE, 2016, pp. 77–81.

[20] N. M. Danchenko, A. O. Prokofiev, and D. S. Silnov, "Detecting suspicious activity on remote desktop protocols using honeypot system," in Young Researchers in Electrical and Electronic Engineering (EIConRus), 2017 IEEE Conference of Russian. IEEE, 2017, pp. 127–128.

[21] S. Djanali, F. Arunanto, B. A. Pratomo, A. Baihaqi, H. Studiawan, and M. Shiddiqi, "Aggressive web application honeypot for exposing attacker's identity," in Information Technology, Computer and Electrical Engineering (ICITACEE), 2014 1st International Conference on. IEEE, 2014, pp. 212–216.

[22] Y. Alosefer and O. F. Rana, "Predicting client-side attacks via behaviour analysis using honeypot data," in Next Generation Web Services Prac-tices (NWeSP), 2011 7th International Conference on. IEEE, 2011, pp. 31–36.

[23] N. Naik, P. Jenkins, N. Savage, and L. Yang, "A computational intelligence enabled honeypot for chasing ghosts in the wires," Complex & Intelligent Systems, vol. 7, no. 1, pp. 477–494, 2021.

[24] N. Antunes and M. Vieira, "Detecting sql injection vulnerabilities in web services," in Dependable Computing, 2009. LADC'09. Fourth Latin-American Symposium on. IEEE, 2009, pp. 17–24.

[25] "The Apache HTTP Server Project," https://httpd.apache.org/.

[26] "Internet Information Services (IIS)," https://docs.microsoft.com/en-us/iis/get-started/introduction-to-iis/iis-web-server-overview.

[27] R. Hertzog, J. O'Gorman, and M. Aharoni, "Kali linux revealed," Mastering the Penetration Testing Distribution, 2017.

[28] "Konsole - KDE's Terminal Emulator," https://konsole.kde.org/.

[29] A. V. Arzhakov and I. F. Babalova, "Analysis of current internet wide scan effectiveness," in Young Researchers in Electrical and Electronic Engineering (EIConRus), 2017 IEEE Conference of Russian. IEEE, 2017, pp. 96–99.

[30] "NXcrypt; a polymorphic 'python backdoors' crypter," https://github.com/Hadi999/NXcrypt.

[31] Anwar, Ahmed H., Charles A. Kamhoua, Nandi O. Leslie, and Christopher Kiekintveld. "Honeypot Allocation for Cyber Deception Under Uncertainty." IEEE Transactions on Network and Service Management (2022).