Mapana – Journal of Sciences
2022, Vol. 21, No. 4, 99-118
ISSN 0975-3303/https://doi.org/10.12723/mjs.63.6

A Comprehensive Research Study on Low-Interaction Secure Shell Honeypot

Sudesh Pahal* & PreityPriya†

Abstract

This paper details information acquired from a secure shell honeypot, including plaintext login credentials and comprehensive attack data. As the number of data breaches and password leaks rises year after year, more dictionaries of reverse-engineered hashed passwords develop. Besides contributing to educational password dictionaries, this article also attempts to provide information about the geographical makeup of hackers encountered, as well as favored protocols. Its goal is to encourage developers to produce practical honeypot solutions for organizations with limited resources for their cyber-protection, as well as to encourage organizations to implement such measures and study their data. The low-interaction, user-friendly honeypot created is capable of running without manual intervention, and without interfering with parallelly running processes. Besides collecting login credentials used with SSH, in plaintext, its capabilities include recording, analyzing, and sending notifications about suspicious network traffic.

* Department of Electronics & Communication Engineering, Maharaja Surajmal Institute of Technology (MSIT), Guru Gobind Singh Indraprastha University, New Delhi, Delhi, India; pahal.sudesh@gmail.com
† Department of Electronics & Communication Engineering, Maharaja Surajmal Institute of Technology (MSIT), Guru Gobind Singh Indraprastha University, New Delhi, Delhi, India; w.preitypriya@gmail.com

Keywords: Honeypot, SSH logging, Network security, Deception technology

1. Introduction

1.1. Background

A network is a set of devices that use communication protocols to share resources. It establishes an architecture that allows a variety of equipment types to organize, unify and control hardware and software components of the network.

While networks have brought humanity closer than ever, their improper implementation or inadequate security can have very serious real-world consequences [1, 2], such as the remote deployment of computer viruses and worms, or the launch of Denial of Service (DoS) attacks.

Network security refers to the protection of data from unauthorized access, damage and development, and the implementation of policies and procedures for recovery from breaches and data losses. It can be implemented via an offensive approach, a defensive approach, or a hybrid approach. While offensive security is realised by deploying a proactive approach to security through the use of ethical hacking, defensive security uses a reactive approach to security that focuses on prevention, detection, and response to attacks.

Honeypots are emulated deceptive systems that can be used to assess where hackers infiltrating a network are coming from, the level of threat, their modus operandi, data of interest and the effectiveness of the hosting party's security stack. They are designed to trick the attacker into thinking a genuine system has been pawned, by purposely engaging them and identifying malicious activities performed by them over the internet. Honeypots are deliberately configured with known vulnerabilities in place, to make attractive targets for attackers. Since no interaction with a honeypot is authorized, all traffic is suspicious. Honeypots can thus automatically and accurately detect, analyze, and defend against zero-day and advanced attacks - providing insight into malicious activity within networks using a preventive, deceptive approach to security. The usage of tactics that rely on a thorough understanding of the system environment and its analysis to detect potential flaws influences the development and deployment of preventive and protective measures that discourage or eliminate cyberattacks to a large extent. Due to this reason, honeypots are now being used in both, governmental and non-governmental organisations such as banks, industrial control systems, educational institutions, etc.

1.2. Related Work

As defined by Joshi and Sardana [3], a honeypot is "A program that takes the appearance of an attractive service, set of services, an entire operating system or even an entire network, but is in reality, a tightly sealed compartment built to lure and contain an attacker". Covered by Tsikerdekis et al [4], most of the work available today concentrates on the development of unique honeypots that frequently target a specific feature, without offering a comprehensive understanding of how they might be built to prevent detection by attackers.

As summarised by Campbell et al [5], honeypots can be classified as (i) low-interaction, medium-interaction or high-interaction, on the basis of their functionality and supported services, (ii) deception, intimidation or reconnaissance on the basis of their mode of deployment, or (iii) production and research, on the basis of their deployment category. By conducting a comprehensive analysis of existing honeypot literature, they concluded that by the early 21st century, developed countries such as the United States of America and South Africa had provided far more insights into the usage and significance of honeypots than other countries, possibly due to their higher level of dependence on computing networks for daily functioning in those times.

Their insights made it evident that most of the research in this field took place when (i) internet usage started to grow in the absence of security standards (2002-2003), and (ii) internet-supported devices became commonplace, which led to its utilization for a diverse range of activities such as business, banking, social networking and the like (2006-2012). Themes such as new types of honeypots, improving the accuracy in threat detection, lowering false positives and avoiding detection appeared to be preferred over studies on the ethics of honeypots, mainly by researchers motivated by academic incentives that come with journal publication.

Further explained by Tsikerdekis et al [4] and summarized in Table 1, honeypots that follow the Secure Shell (SSH) protocol without allowing much shell functionality and allow interactions for limited periods of time can be classified as low-interaction honeypots, usually placed in networks not being monitored by Intrusion Detection Systems (IDS). They are prone to detection and are configured as such. High-interaction honeypots, however, are configured to avoid detection to discover zero-day attacks and the modus operandi of hackers. For this reason, they emulate legitimate systems very thoroughly. This functionality is determined by the deployment category, i.e., research or production. While the former is placed within the network's Demilitarized Zone [6] to gather a wide range of threat intelligence, the latter maintains proximity to real assets for very specific intelligence from both, internal and external threats.

Anti-detection mechanism	Characteristic			
	Type (Research/ Production)	Interaction (Low/ High)	Scalability (Low cost/ High cost)	Implementation (Software/Hard ware)
Automatic honeypot redeployment	Either	Low	Low cost	Software
Honeypot delay reduction	Either	Either	Low cost	Software
Honeypot process transparency	Research	High	Low cost	Software
Dedicated hardware	Production	Either	High cost	Hardware

Dynamic intelligence on honeypots	Research	High	High cost	Software

Table 1. Analysis of related works

Depending on the type of implementation, i.e. (i) hardware - regular computers or specialized Supervisory Control and Data Acquisition (SCADA) systems, (ii) software-simulated hardware using virtualization, or (iii) hybrid, the scalability of honeypots becomes a notable factor, especially in the case of botnets, and/or state-sponsored attacks.

Exploring the theme of avoiding honeypot detection, this study laid out possible approaches that can be studied and implemented for more realistic emulations. Proposing (i) automatic honeypot redeployment - redeployment of the honeypot with an altered configuration upon detection by an attacker, (ii) honeypot delay reduction - minimization of delays caused by event logging - prone to detection unless the latency of the virtual honeypot network is lowered to match a physical network's link latency, (iii) honeypot process transparency - hiding unrealistic modified sequences of events such as the forwarding of connections between a honeypot's frontend and backend, by emulating a three-way Transmission Control Protocol (TCP) handshake while hiding the same, (iv) dedicated hardware - using specific hardware components to reduce software delays, increase system security, and enabling the system to support honeynets; and (v) dynamic intelligence on honeypots -the usage of machine learning and artificial intelligence to disable unexpected programs, dynamically change directory structures to increase attractiveness, and encourage attackers to reveal their geo-cultural identities on the basis of their interactions; the authors concluded that while a honeypot environment's alignment with an attacker's expectation of legitimate systems determines the chances of detection, constraints such as available hardware, development and maintenance costs, and legal restraints don't enable developers to build extremely efficient honeypots.

While these studies explore past literature and future implementation strategies in detail, the challenge of minimizing detection also depends on a thorough understanding of the challenges that require these solutions in the first place. Prior to the study by Tsikerdekis et al [4], Du [7] conducted research on the same, determining that honeypots mainly face issues in (i) hiding capture tools while collecting as much data as possible, (ii) capturing session data encrypted on the hacker's side, and (iii) collecting and transmitting data through secret channels. To combat the same, they proposed the following solutions: (i) Capture Tool Hiding via a) Module Hiding - deleting the pointer of the capture module of any data capturing tool loaded to the Linux kernel upon system initialization, and b) Process Hiding - changing the system call used to query process information in a system using the "ps" command, in order to stop programs using the system call from accessing the file, thus hiding the process. This can be effective as the program(s) within the honeypot would be executing multiple system processes; (ii) Session Encryption Data Capture - while the execution of Trojan shells upon logging in can be exposed easily, changing the index of pointers of system calls such as read() and write() can enable the implementation of the capture module's own functions, which would result in direct access to the data that is part of such system calls, and (iii) Establishment of Hidden Data Transmission Channel - hiding the transfer of logs to centralized honeypot servers by configuring the capture module to transfer data via User Datagram Protocol (UDP) streams after altering the kernel on each endpoint such that data packets cannot be accessed. This would require the capture module to match the preset destination UDP port and magic number (a constant numerical value used to identify different protocols) on the endpoints within the Local Area Network (LAN) in order to make network sniffers on the endpoints ignore the packets.

Although this study was highly specific and dealt with issues directly at the kernel level, the highlighted approaches have certain drawbacks: (i) the capture module cannot be unloaded once it has been loaded, and the root user cannot locate it, and (ii) if the capture module contains a bug, the kernel may become unstable and the system may crash. These issues may have an impact on the normal operation of the honeypot, as well as the overall performance of the honeynet. The lack of implementation of these suggestions provides no insight into the feasibility of these methods, especially in the long term.

Finally, recent comprehensive surveys [8,9, 10, 11] of the research on honeypots and honeynets for Internet of Things (IoT), Industrial Internet of Things (IIoT), and Cyber-Physical Systems (CPS) over the period 2002-2020 dealt with the taxonomy and analysis, key design factors, and open issues for future honeypots and honeynets for IoT, IIoT, and CPS environments revealed that the key to the design and implementation of competent honeypots lies in a good understanding of its target application area, purpose, cost, deployment location, intended level of interaction with the attacker, resource level, services, simulation or emulation, realistic service to the attacker, tools that will be used, the possibility of fingerprinting and indexing, and the liability issues that may come up.

To conclude, attackers have been able to detect honeypots and identify ways to exploit them because of

- the lack of research and expertise in emerging domains such as machine learning, unexplored protocols, anti-detection mechanisms, optimized deployment location, and the constant threat of insider attacks, and hardware vulnerabilities
- to date, much of the research has been focused on the creation of unique honeypots that typically focus on a single component without offering a comprehensive knowledge of how they could be structured to prevent detection by attackers
- the data been collected with certain restrictions, such as short time ranges, cultural biases, a narrow range of tools/technologies tested, etc.
- the large majority of these honeypots are built on outdated systems, with poor maintenance and irregular development cycles. Accessible to both, security professionals and attackers, they are predictable due to their limited adaptability and poor deception [4].

The integration and expansion of these categories could provide a clearer understanding of current issues, and the methods of eradicating them.

Proposed solutions are either valid under very strict conditions - on the basis of necessary hardware and software - or aren't comprehensive of the above-mentioned factors. Additionally, for a honeypot to be feasible and effective, a certain degree of deception is absolutely necessary, which isn't provided by the default configurations of most non-commercial honeypots.

2. Problem Statement

As mentioned earlier, the primary limitation of currently available honeypots lies in their deception capabilities, and the level of technical knowledge required for their efficient usage. In today's highly connected and extremely vulnerable digital space, honeypots are a necessary defence mechanism not only for niche research institutions and/or large organisations with a considerable security-focused workforce but also for smaller organisations dependent on the internet for any degree of daily functioning - regardless of their technical expertise [9]. Thus, arises the problem statement, and the proposed solution:

"The availability of open-source honeypots makes defensive network security easier for organisations across industries. However, the level of technical expertise required to customise their configuration and improve their deception abilities is not available to small organisations. This gap in requirement vs availability means that the advancement in honeypot research has not yet resulted in enough real-world implementation of proposed deception solutions to make this technology feasible for the global community. To minimise the need for small organisations to have extreme familiarity with honeypots before using them, more open-source honeypots should be built and deployed with advanced deception capabilities in their base configuration. This way, a wider range of individuals and organisations would be able to protect their networks, or study new attack methods being leveraged by hackers across the globe - without getting detected themselves."

In order to study this solution's feasibility, the creation of a low-interaction honeypot has been carried out for network monitoring.

3. Materials & Method

3.1. Architecture

A basic low-interaction honeypot has been created, with support for Hypertext Transfer Protocol (HTTP), Hypertext Transfer Protocol Secure (HTTPS), Secure Shell (SSH) and File Transfer Protocol (FTP) protocols. It is capable of logging all network traffic on its interfaces, parsing them, and sending summarised notifications on Slack Messenger - a messaging application built for and used extensively by businesses. The honeypot is capable of responding to attacker vulnerability probes and appears open to SSH connections, enabling the collection of login credentials being used from the attacker's side, for further analysis. As explained in Fig. (1), Python has been used as the programming language to deploy this honeypot on a virtual machine configured as a CentOS 8 x64 server, for minimal manual intervention over a period of multiple weeks of log collection. The honeypot system makes use of network monitoring tools on the server for the collection of the above-mentioned logs.

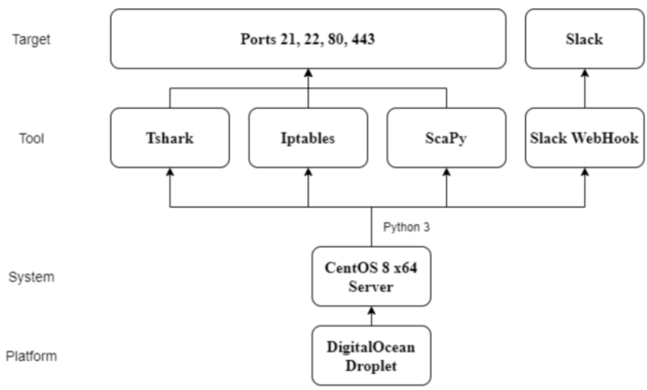


Fig. (1). Summary of Targets, Tools, the System and the Platform in use

3.2. Methodology

The research technique used for this study involved carrying out a comprehensive review of literature on honeypots. This required gathering qualitative and quantitative data from a variety of sources - including books, journal papers, conference proceedings, and the Internet. Keywords such as "honeypot", "SSH logging", "network security", and "deception technology" were used for the same.

Parameters such as honeypot detectability, type (research/production), interaction (low/high), scalability (low cost/high cost), and implementation (software/hardware) were evaluated. After gathering this information, the sources were examined to see if they were pertinent, and duplicate information was eliminated. It was found that several sources featured more than one theme while the data was being gathered. In these situations, the prevailing subject matter was regarded as the principal theme of that source.

Finally, the advantages and disadvantages of each existing/proposed honeypot model were compared and combined to create a user-friendly, low-interaction honeypot that addresses

- support for detection of multiple communication protocols
- support for logging SSH credentials used via communicating with the system
- support for providing notifications of event summary via business channels

as discussed in this paper.

3.2.1. Protocol Support Module

In order to capture all TCP network traffic at the default interface, Tshark - a network protocol analyzer - has been employed for FTP, SSH, HTTP and HTTPS logging on ports 21, 22, 80 and 443. Scapy - a packet manipulation program - has been used to check for FTP, SSH, HTTP and HTTPS SYN (synchronize) requests from any source and log each request with the source IP address, source port and destination port. Additionally, it replies with custom SYN-ACK (synchronize-acknowledge) packets to these requests - thus appearing vulnerable to insecure connections from attackers.

```
-P INPUT ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p tcp -m multiport --dports 21,22,80,443 -m conntrack --ctstate NEW,ES
TABLISHED -j ACCEPT
-A INPUT -p tcp -m multiport --dports 21,22,80,443 -m conntrack --ctstate ESTABL
ISHED -j ACCEPT
```

Fig. (2). Iptables INPUT chain rules instructing the system to enable communication on ports 21, 22, 80 and 443

```
-P OUTPUT ACCEPT
-A OUTPUT -o lo -j ACCEPT
-A OUTPUT -m conntrack --ctstate ESTABLISHED -j ACCEPT
-A OUTPUT -p tcp -m multiport --dports 21,22,80,443 -m conntrack --ctstate ESTAB
LISHED -j ACCEPT
-A OUTPUT -p tcp -m multiport --dports 21,22,80,443 -m conntrack --ctstate NEW,E
STABLISHED -j ACCEPT
```

Fig. (3). Iptables OUTPUT chain rules instructing the system to enable communication on ports 21, 22, 80 and 443

These packets are created on the basis of certain firewall rules, as seen in Fig. (2) and Fig. (3). If TCP packets from any source port on the outgoing interface have the RST (reset) flag set, the packets are dropped as RST indicates the need for connection termination. The RST iptables rule is dropped when the script stops running.

3.2.2. SSH Credential Logging Module

By default, the SSH protocol logs SSH login attempts, regardless of whether or not authentication is successful. However, since it uses an encrypted tunnel for all communication, it isn't possible to read the data being sent and the local logs do not record the passwords being used. Therefore, it isn't possible to log the login credentials being used via SSH with its default configuration. In order to overcome this, the SSH configuration present on the server has been altered as required.

The altered configuration has been achieved by executing the following as the root user:

1) Uninstall the SSH server and download from the source.
2) Insert a logit() function in the SSH authentication file "auth-passwd.c" at the location highlighted in Fig. (4).
3) Configure and install the SSH server as required.

```
int
sys_auth_passwd(struct ssh *ssh, const char *password)
{
    Authctxt *authctxt = ssh->authctxt;
    # ---- INSERT HERE ----
    logit("Honeypot_Log: Username and Password -> %s,%s",authctxt->user,password);
    # ^ INSERT HERE ^
    auth_session_t *as;
    static int expire_checked = 0;

        as = auth_usercheck(authctxt->pw->pw_name, authctxt->style, "auth-ssh",
            (char *)password);
        .
        .
        .
        }
        return (auth_close(as));
        }
    }
    #elif !defined(CUSTOM_SYS_AUTH_PASSWD)
    int
    sys_auth_passwd(struct ssh *ssh, const char *password)
    {
        Authctxt *authctxt = ssh->authctxt;
        # ---- INSERT HERE ----
        logit("Honeypot_Log: Username and Password -> %s,%s",authctxt->user,password);
        # ^ INSERT HERE ^
        struct passwd *pw = authctxt->pw;
        char *encrypted_password, *salt = NULL;
        .
        .
        .
        return encrypted_password != NULL &&
            strcmp(encrypted_password, pw_password) == 0;
    }
    #endif
```

Fig. (4). Credential logging function required in SSH server's password authentication file 'auth-passwd.c'

3.2.3. Notification Module

The need for timely, concise and easily accessible updates about possible attackers is extremely important for any organisation hosting a honeypot. Without it, there would be complete reliability in manually collecting traffic logs to detect and calculate all attempted connections to the honeypot. This would be slow, and prone to human errors. To accommodate this requirement, a Slack notification module has been included in this honeypot system. Slack is a messaging application used for team communications by businesses. It handles messages, files, third-party integrations such as Twitter, Dropbox, Google Docs, Trello, GitHub and dozens of other services all in one place. From large companies such as Pinterest, Airbnb and Shopify to smaller startups - all types of businesses use Slack - making it the ideal choice for an attack notification centre.

Slack's incoming webhook feature - a simple way to post messages from Slack applications to any channel - has been used to send updates about the number of connections attempted, to a Slack channel being used by the administrator (organisation). This has been achieved by reading all the source IP addresses from the traffic logs gathered by the honeypot, counting unique IP addresses found in the logs, and calculating which ones attempted the maximum number of connections. Using the source IP addresses and the number of times they sent connection requests (Top 1, Top 2 or Top 3), messages are created and sent to the Slack channel.

4. Observations

4.1. Results

The analysis of the gathered network traffic logs reveals information such as the attackers' geographical location, protocols being used, timestamps of the attacks, etc. The success of this study has been determined by the running of the honeypot, the level of deception it provides, and the variety of data it successfully collects. These results aim to encourage developers to work on security solutions for all types and sizes of organisations, supporting future research that would provide insights into the current state of available solutions.

4.1.1. Traffic Logging

The honeypot was deployed for 240 hours, from 21 October 2021 to 31 October 2021. Using the logs collected during this period, the following information was gathered:

Fig. (5). Logfile snippet

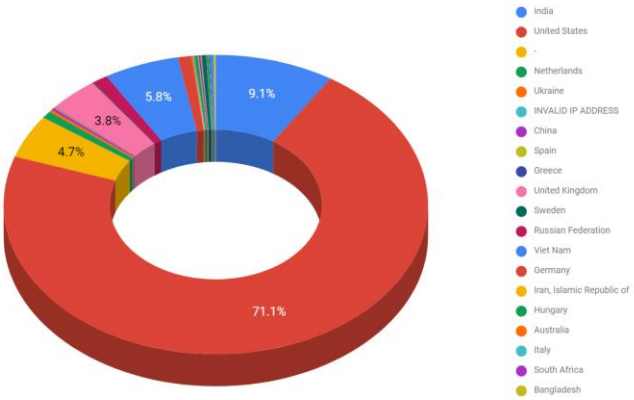


Fig. (6). Attack source (location) distribution

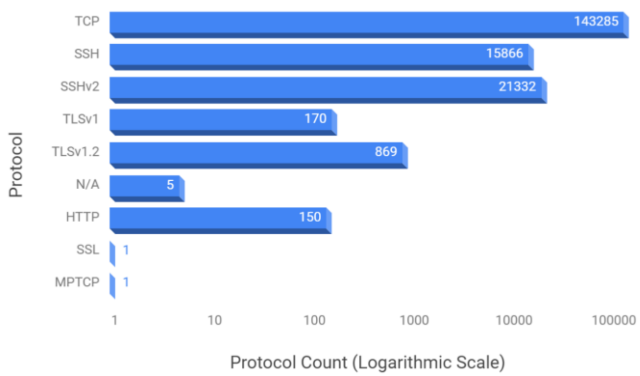


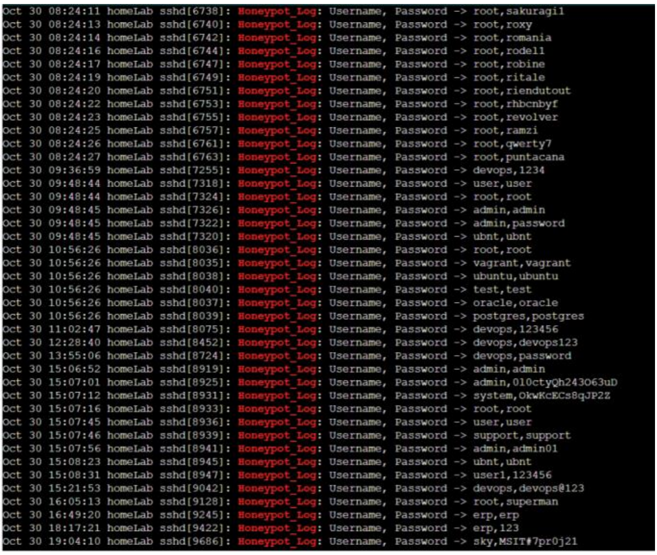
Fig. (7). Commonly exploited protocols

. Upon analysing the logs displayed in Fig. (5), it was observed that out of a total of 1,81,674 attempted connections, a strikingly large amount of traffic (71.1%) was generated from IP addresses mapped to the United States of America, while India reached the 9.1% mark - standing behind is Viet Nam at 5.8%. Other distinguishable locations included the United Kingdom (3.8%), the Russian Federation (1.2%) and the others (<=1%). Unidentifiable locations accounted for 4.7% of all traffic. While the difference in the amount of traffic generated by certain geographic locations may seem surprising in Fig. (6), factors such as technological advancement, infrastructure holding capacity and the usage of spoofed IP addresses or Virtual Private Networks must be kept in mind.

Overall, Fig. (7) shows a total of 1,43,285 TCP sessions, 2,13,323 SSHv2, and 15,866 SSH sessions. Across these sessions, the most commonly exploited protocol was HTTPS, with 1040 unique requests. HTTP was used for 150 unique sessions, while other protocols were very rare.

4.1.2. SSH Credential Logging

Following the custom SSH server configuration, the SSH local log file '/var/log/secure' not only contains records of attempted connections, but also the credentials used in those attempts - in plaintext, as evident in Fig. (8).



```
Oct 30 08:24:11 homeLab sshd[6738]: Honeypot_Log: Username, Password -> root,sakuragil
Oct 30 08:24:13 homeLab sshd[6740]: Honeypot_Log: Username, Password -> root,roxy
Oct 30 08:24:14 homeLab sshd[6742]: Honeypot_Log: Username, Password -> root,romania
Oct 30 08:24:16 homeLab sshd[6744]: Honeypot_Log: Username, Password -> root,rodell
Oct 30 08:24:17 homeLab sshd[6747]: Honeypot_Log: Username, Password -> root,robine
Oct 30 08:24:19 homeLab sshd[6749]: Honeypot_Log: Username, Password -> root,ritale
Oct 30 08:24:20 homeLab sshd[6751]: Honeypot_Log: Username, Password -> root,riendutout
Oct 30 08:24:22 homeLab sshd[6753]: Honeypot_Log: Username, Password -> root,rhbcnbyf
Oct 30 08:24:23 homeLab sshd[6755]: Honeypot_Log: Username, Password -> root,revolver
Oct 30 08:24:25 homeLab sshd[6757]: Honeypot_Log: Username, Password -> root,ramzi
Oct 30 08:24:26 homeLab sshd[6761]: Honeypot_Log: Username, Password -> root,qwerty7
Oct 30 08:24:27 homeLab sshd[6763]: Honeypot_Log: Username, Password -> root,puntacana
Oct 30 09:36:59 homeLab sshd[7255]: Honeypot_Log: Username, Password -> devops,1234
Oct 30 09:48:44 homeLab sshd[7318]: Honeypot_Log: Username, Password -> user,user
Oct 30 09:48:44 homeLab sshd[7324]: Honeypot_Log: Username, Password -> root,root
Oct 30 09:48:45 homeLab sshd[7326]: Honeypot_Log: Username, Password -> admin,admin
Oct 30 09:48:45 homeLab sshd[7322]: Honeypot_Log: Username, Password -> admin,password
Oct 30 09:48:45 homeLab sshd[7320]: Honeypot_Log: Username, Password -> ubnt,ubnt
Oct 30 10:56:26 homeLab sshd[8036]: Honeypot_Log: Username, Password -> root,root
Oct 30 10:56:26 homeLab sshd[8035]: Honeypot_Log: Username, Password -> vagrant,vagrant
Oct 30 10:56:26 homeLab sshd[8038]: Honeypot_Log: Username, Password -> ubuntu,ubuntu
Oct 30 10:56:26 homeLab sshd[8040]: Honeypot_Log: Username, Password -> test,test
Oct 30 10:56:26 homeLab sshd[8037]: Honeypot_Log: Username, Password -> oracle,oracle
Oct 30 10:56:26 homeLab sshd[8039]: Honeypot_Log: Username, Password -> postgres,postgres
Oct 30 11:02:47 homeLab sshd[8075]: Honeypot_Log: Username, Password -> devops,123456
Oct 30 12:28:40 homeLab sshd[8452]: Honeypot_Log: Username, Password -> devops,devops123
Oct 30 13:55:06 homeLab sshd[8724]: Honeypot_Log: Username, Password -> devops,password
Oct 30 15:06:52 homeLab sshd[8919]: Honeypot_Log: Username, Password -> admin,admin
Oct 30 15:07:01 homeLab sshd[8925]: Honeypot_Log: Username, Password -> admin,010ctyQh243063uD
Oct 30 15:07:12 homeLab sshd[8931]: Honeypot_Log: Username, Password -> system,OkwKcECs8qJP2Z
Oct 30 15:07:16 homeLab sshd[8933]: Honeypot_Log: Username, Password -> root,root
Oct 30 15:07:45 homeLab sshd[8936]: Honeypot_Log: Username, Password -> user,user
Oct 30 15:07:46 homeLab sshd[8939]: Honeypot_Log: Username, Password -> support,support
Oct 30 15:07:56 homeLab sshd[8941]: Honeypot_Log: Username, Password -> admin,admin01
Oct 30 15:08:23 homeLab sshd[8945]: Honeypot_Log: Username, Password -> ubnt,ubnt
Oct 30 15:08:31 homeLab sshd[8947]: Honeypot_Log: Username, Password -> user1,123456
Oct 30 15:21:53 homeLab sshd[9042]: Honeypot_Log: Username, Password -> devops,devops@123
Oct 30 16:05:13 homeLab sshd[9128]: Honeypot_Log: Username, Password -> root,superman
Oct 30 16:49:20 homeLab sshd[9245]: Honeypot_Log: Username, Password -> erp,erp
Oct 30 18:17:21 homeLab sshd[9422]: Honeypot_Log: Username, Password -> erp,123
Oct 30 19:04:10 homeLab sshd[9686]: Honeypot_Log: Username, Password -> sky,MSIT#7pr0j21
```

Fig. (8). Filtered view of SSH server log file '/var/log/secure'

With over 315 unique usernames and 1233 unique passwords, the highest frequency was calculated for the credentials (in any combination) present in Table 2:

Usernames	Passwords
user	information
port	remote
root	admin
tracerlab	pi

ftp	oracle
rustserver	sync
webmaster	hyjx
mike	aaron
db2inst2	amy
install	m
reboot	support1
matt	Azureuser123
tmp	web
ems	carlos
dillon	Guest123
printer	bruce
ayden	xbian
belkinstyle	albaunio
paul	ts3
epg	alpha
pierre	nobody
ghost	youssra
new	vanesa
full	transformer
ts3	street1

guestuser	saluttoi
coach	romania
nobody	qwerty7
12qwaszx	superman

Table 2. Most frequently used usernames and passwords with SSH

4.1.3. Slack Notifications

Useful in tracking down persistent attackers, the Slack notification module works to calculate the total number of connections attempted by IP addresses that interact with the honeypot frequently. Based on the logs collected during the above-mentioned duration, the top 3 IP addresses that interacted with the honeypot made a total of 1,11,984 requests - as shown in Fig. (9), and the required information was sent as a message to the associated Slack channel.

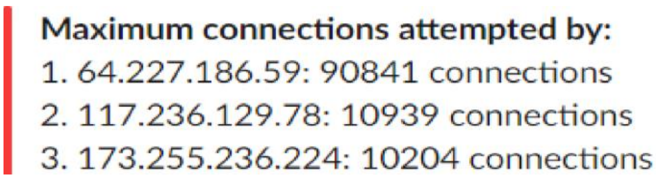


Fig. (9). Slack notification for the specified duration

4.2. Analysis

Since SSH logs all attempted connections, the IP addresses associated with failed connections have also been recorded, along with the username. When required, this data may be analysed separately. Additionally, the SSH protocol allows authentication using keys, instead of passwords. Analysis of the log file shows that 53 unique IP addresses attempted key-based authentication a total of 2857 times, in addition to password-based authentication - which has a total of 315 unique usernames with 1233 unique passwords in various combinations.

5. Conclusion

In this paper, we presented a user-friendly low-interaction honeypot. The honeypot is capable of running without manual intervention - once it has been deployed - and keeps track of each deployment session, without interfering with parallelly running processes (if any). The honeypot is capable of recording and analysing suspicious network traffic, as well as notifying the hosting organisation about the same. Additionally, it can collect login credentials used with SSH in plaintext, for a deeper insight into vulnerable keywords that may be blacklisted for increased security.

Challenges Faced

1) A large majority of currently available honeypots is built on outdated systems, with poor maintenance and irregular development cycles. They are predictable due to their limited adaptability and poor deception. Due to this, analysis of theory regarding fully functional honeypots that are user friendly enough to require minimal configuration, while being low interaction was difficult. However, by understanding the desirable aspects of multiple open-source honeypots, it was possible to integrate all the required functionality into one tool - while narrowing down on the exact architecture and tools needed for smooth functioning.
2) Default SSH logging of authentication attempts, while helpful, does not record passwords being used. Although this is a secure practice, it made the custom configuration of the SSH server on the honeypot a time-taking task. Taking inspiration from independent security researchers' attempts at implementing this idea [12], it was possible to create a solution that works with CentOS 8 x64 servers.

Future Scope

In order to make the honeypot more comprehensive, support modules for analysing network requests captured with the traffic could be added. Doing so would allow researchers to get notified about possible attack attempts such as HTTP-enabled backdoor installation. Additionally, platform support for a wider range of operating systems and environments could be added to reach a wider userbase.

References

[1] M. Kumar, "Security Issues and Privacy Concerns in the Implementation of Wireless Body Area Network" in *2014 International Conference on Information Technology*, 2014.

[2] Khanum, S., Pahal, S., Makkad, A., Panwar, A., & Panwar, A. (2018). Securing Onion Routing Against Correlation Attacks. Advances in Intelligent Systems and Computing, 573–580. https://doi.org/10.1007/978-981-13-1819-1_54

[3] R. C. Joshi and A. Sardana, "*Honeypots: A New Paradigm to Information Security*", 1st ed., Science Publishers, 2011.

[4] M. Tsikerdekis, S. Zeadally, A. Schlesener, and N. Sklavos, "Approaches for Preventing Honeypot Detection and Compromise" in *Global Information Infrastructure and Networking Symposium (GIIS),* 2018.

[5] R. M. Campbell, K. Padayachee, and T. Masombuka, "A survey of honeypot research: Trends and opportunities" in *10th International Conference for Internet Technology and Secured Transactions (ICITST)*, 2015.

[6] What Is a DMZ and Why Would You Use It? Fortinet. "Reference: Available from: https://www.fortinet.com/resources/cyberglossary/what-is-dmz" (Accessed 28 August 2021).

[7] G. E. J. Du, "A Study on Cyber Defense Honeynet Technology and Configuration Examples" in *International Journal of Simulation: Systems, Science & Technology*, 2016.

[8] J. Franco, A. Aris, B. Canberk, and A. S. Uluagac, "A Survey of Honeypots and Honeynets for Internet of Things, Industrial Internet of Things, and Cyber-Physical Systems" in *IEEE Communications Surveys & Tutorials*, 2021.

[9] Turpitka, D. (2020, January 28). When You Can't Stop Every Cyberattack, Try Honeypots. Forbes. "Reference: Available from: https://www.forbes.com/sites/forbestechcouncil/2020/01/28/when-you-cant-stop-every-cyberattack-try-honeypot" (Accessed 28 August 2021).

[10] M. Sharma, S. Pant, D. Kumar Sharma, K. Datta Gupta, V. Vashishth, & A. Chhabra. *"Enabling security for the Industrial Internet of Things using deep learning, blockchain, and coalitions. Transactions on Emerging Telecommunications Technologies*". 2020. – 8

[11] Pahal, Sudesh, NeeruRathee, and Brahmjit Singh. "A Deep Learning-Based Model for Link Quality Estimation in Vehicular Networks." *IETE Journal of Research* (2021): 1-10.

[12] Cole, J. (2011, December 3). SSH Password Logging. "Reference: Available from: https://www.jessecole.org/2011/12/03/ssh-password-logging/" (Accessed 19 October 2021).