An improved honeypot model for attack detection and analysis

Marwan Abbas-Escribano, Hervé Debar

An Improved Honeypot Model for Attack Detection and Analysis

Marwan Abbas-Escribano
Télécom Sud Paris & Sésame IT
France
marwan.abbas@telecom-sudparis.eu

Hervé Debar
Télécom Sud Paris
France
herve.debar@telecom-sudparis.eu

ABSTRACT

This paper presents a new model and design for honeypots, and the results obtained the implementation and exposure on the internet of an high interaction honeypot. We show that our model can allow higher interaction with attackers while preserving integrity and attractiveness. In our work, we use threat analysis based on the MITRE ATT&CK taxonomy to describe the design and supervision constraints of our honeypot with it's situation in our implemented architecture. We exposed our infrastructure during seventeen days and collected information about several actors and attack methods, from which we extracted previously undocumented Indicators of Compromise.

ACM Reference Format:
Marwan Abbas-Escribano and Hervé Debar. 2023. An Improved Honeypot Model for Attack Detection and Analysis. In *The 18th International Conference on Availability, Reliability and Security (ARES 2023), August 29–September 01, 2023, Benevento, Italy*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3600160.3604993

1 INTRODUCTION

Cyberattacks have now a high incidence and impact on activity and public image for long periods of time: it takes on average 315 days to detect and contain a data breach following a malicious attack and 23 days to recover from a successful ransomware attack according to industry surveys[2, 24]. Early attack detection is a vital topic for IT infrastructures, and obtaining hints about malicious activity early and continuously is increasingly important for successful cyber defense. This document presents the use of several deception techniques to detect attackers by enticing them to engage in repeated suspicious behavior. We define suspicious behavior as "*any behavior that differs significantly from the expected behavior of a genuine system*".

The main component of deception in cybersecurity is the honeypot, formally defined by Spitzner as a *"decoy computer resource whose value lies in being probed, attacked or compromised"*[28]. While often used in research and production environments, a drawback of honeypots is that they are built and deployed with a specific service or vulnerability in mind, directly exposed to the attacker, and offering little opportunity for interactions. They does not allow any genuine usage besides deception, an so almost any activity in an honeypot can be considered as suspect, leading to less false positive alerts. Furthermore, honeypots can generally be easily identified by attackers, and this identification shared in attackers circles, enabling them to avoid the deception, and leaving only automated processes to hit the honeypots. This leads to lower usability of the information collected by honeypots.

Our objective is to increase the level of interaction offered by a honeypot, thus potentially increasing its likeness with real information systems. In this work, we present an innovative model for honeypots, that includes a significant portion of the system not exposed directly to the Internet. Our objective in doing so is to increase the attractiveness of the honeypot by offering to the attacker successful attack pathways that enable him to progress in the honeypot, and obtain additional access. More specifically, our contributions are as follows:

- we propose a new design for honeypots, enabling a higher interaction level with attackers than currently existing in the state of the art, while preserving system integrity and attractiveness;
- we place this design in the scope of an up-to-date cyber-threat intelligence model, the MITRE ATT & CK;
- we implement such a design using open source tools, demonstrating the feasibility of our proposal;
- we expose the honeypot on the Internet and analyze the collected data, comparing our results to similar deployments in the literature.

Our work attempts to find an answer to three questions:

- Is it possible to implement an attractive honeypot using MITRE ATT & CK's taxonomy as a standard base of knowledge ?
- Can we use a virtualized architecture in order to deploy such honeypot while limiting resources consumption ?
- Can we impose constraints in our model and our implementation to canalize the actions of an attacker in order to extract pertinent IoCs with a reduced risk ?

2 STATE OF THE ART

Attack detection is a mature research topic. Many techniques have been described, with corresponding strengths and weaknesses[25]. Deception is used for this purpose in a variety of scopes and configurations[7, 16], ranging from pure attack detection[31] to extended technical analysis of trends and techniques used by actors[19, 23] or even behavior study with human participants[15].

The main advantage of honeypots for data collection is their capacity to generate unpolluted data, independent from their workload and with a reduced false positive rate[22]. Links between honeypots and existing cybersecurity frameworks, such as MITRE matrices or Cyberkillchain has been made in order to adapt their configuration to existing standards[26]. Some recent work focuses

ARES 2023, August 29–September 01, 2023, Benevento, Italy

ACM ISBN 979-8-4007-0772-8/23/08…$15.00
https://doi.org/10.1145/3600160.3604993

on using honeypot data to characterize attacker's behavior, hence improving existing current standards[14].

Depending on the amount of actions an attacker can perform on them, honeypots are divided according to their *interaction level*. In high-interaction honeypots, attackers have *"access to a real operating system where nothing is emulated or restricted"*[18]. This maximizes the range of actions an attacker can perform, but it also increases the resources needed for the deployment and the risk of attackers "escaping" the honeypot and altering data collection. Low-interaction honeypots focus only on minimal communication, often via emulated services, usually for detection and statistics collection. However, they offer significantly less analysis data, and are easily identified by attackers. In practice, honeypots are designed in a spectrum of interaction levels.

Previous studies exposed authentication services such as SSH as honeypots in order to collect data[17, 30]. They focused on topics such as brute force attacks and the collection of commands entered after successful login, while we use authentication honeypots as a first step for a persistent attack. Another difference in our work is that we don't use dedicated honeypot software, but a legitimate SSH demon with dedicated supervision.

Building a honeypot is done by assembling vulnerable elements, monitoring their activity and exposing the system. This approach is robust but restricts the scope of the honeypot as the attacker needs to be able to find and exploit the specific set of vulnerabilities implemented. The technology and theory about honeypot is now mature, and several commercial solutions exist, deploying lures with varying interaction levels and use cases, often based on defensive taxonomies[10, 11]

Honeypot construction is focused on the implementation of specific services. Commercial and open source solutions often present a configurable array of protocols that can be deployed[13]. In our work, we present a model for high interaction honeypots that allows us to deploy honeypots not based on specific vulnerabilities, but on a vulnerability analysis performed from an attacker point of view based on the MITRE ATT&CK taxonomy. This novel approach allows us to deploy realistic honeypots by exposing vulnerabilities creating specific attack paths for attackers, pushing them to progress in their Cyberkillchain in a tailored environment.

3 MODEL AND ARCHITECTURE

Instead of simply associating a vulnerability to a system, we design our honeypot model around MITRE's taxonomy of adversary tactics and techniques, the *MITRE ATT&CK*[1]. We chose it because it constitutes a standardized and updated knowledge base that aspires to cover extensively attack methods and is well known and used.

In our model, honeypots are defined by two parameters, the Tactics and Techniques (TTPs) they implement and their network reachability.

3.1 MITRE's Standards

The MITRE ATT & CK Matrix (MAM) constitutes *"a knowledge base of tactics and techniques based on real world observations"*[29], modeling adversary behavior. An attack process is composed of a Tactic (T), a technique (t) and one or several procedures:

Tactics The tactical objective of an attacker, the why of an attack.

Techniques The how of a tactical objective, the behavior pursuing an objective.

Procedures Specific implementations of the tactic and technique.

We do not explicitly reference MITRE's procedures in order to keep our model light. We thus focus on the attackers intentions rather than on the means they chose, but we will still use the *TTP* denomination, as it is the terminology used in the industry.

Techniques can be linked to different tactics. When we describe a specific technique, we must indicate the associated tactic. Tactics are picked from the eleven listed in the MAM. We define the TTPs associated to an honeypot as a tuple in the form *(Tactic, technique)*.

The techniques presented in the MAM describe the means by which an attacker can fulfill the objective defined by its corresponding Tactic. Techniques are necessarily described by a set of CAPECs (Common Attack Pattern Enumerations and Classifications). They may include a set of CVEs (Common Vulnerabilities and Exposures) to describe software vulnerabilities, and their associated CAPECs. We instantiate our *(Tactic,technic)* couple as a combination of a CAPEC and/or a CVE. Our implementation then requires that the corresponding vulnerability is embedded in one of our honeypots, including specific versions of vulnerable service implementations. The first parameter defining our honeypots is thus a list of CAPECs and/or CVEs.

The CAPEC is a catalog aiming to describe *common attributes and approaches employed by adversaries to exploit known weaknesses in cyber-enabled capabilities* [3]. It constitutes a standardized list of specific action trails performed by attackers in order to exploit known weaknesses and vulnerabilities that are more general than MAM's Procedures. The CVE system allows us to reference and describe vulnerabilities, with three main fields. These fields are CVSS (Common Vulnerability Scoring System), CWE (Common Weakness Enumeration) and CPE (Common Platform Enumeration). The CVSS is a detailed score that describes the criticality of the vulnerability ranging from 0 to 10. The CWE details the weakness, which allows the vulnerability to exist and the CPE describes the "targets" of the vulnerability, using a structured naming scheme.

3.2 Cyber Kill Chain implementation

As the objective of our honeypots is to obtain information concerning attacker's techniques, we must maximize our coverage of the Cyber Kill Chain (CKC), the model identifying which tasks attackers must perform in order to achieve their objectives [20], so we can discover more tools and procedures. In our deployment, attackers encounter two levels of CKC. The first level is the individual CKC for each independent equipment of our honeypot, and the second one is the general CKC for our whole system. The global CKC coverage is ensured by organizing a double progression of the attacker: we want it to progress in our infrastructure, accessing intermediary and internal honeypots from its access point, as much as we want them to progress in its CKC. The parallel progression path between our honeypot and the attacker's CKC is shown on Figure 1, a parallel also being made with the MAM Tactics. This progression mimics

the real process APTs (Advanced Persistent Threats) use to compromise organizations. In addition to the high interaction offered by the services embedded in the honeypots, the progression we expect from attackers adds another level of interaction possibilities. While we deploy three systems with decreasing interaction levels, their assembling provide a high interaction playground for attackers.

In essence, the model creates a longitudinal enclave around a set of parallel paths, representing TTPs available to the attacker. The model detects attempts against the enclave's walls and logs them, and only allows progression along predetermined TTPs paths. We thus expect to record attackers attempts and successes.

The attacker interactions are controlled thanks to our supervision and design choices (section 3.4). As we chose which techniques to "enable" based on strict supervision constraints, we reduce the risk of escape from the enclave while maximizing the amount of data collected relatively to our attack surface.

To incite attackers to progress both in space and CKC, we embed TTPs and vulnerabilities that force a progression, for example, we could make an entry level honeypot vulnerable to initial access, an intermediary honeypot present discovery and lateral movement possibilities and an internal honeypot contain exploitable services or valuable data to exfiltrate.

Another advantage of defining our honeypots configurations by the CKC progression of the attacker is that we control which kind of actions we expect them to perform in each honeypot. This allows us to specifically tailor our supervision to have extended logging on the accessible weaknesses, and be very strict in logging the rest of the activity. This also enhances the global safety of our solution, as the scope of exposure and the attack surface is restricted. This also helps in configuring the sensors with precise, deterministic events and reduces sensor noise such as false positives.

We must, for each position of the honeypots, select the TTPs that ensure attacker progression, as shown in our implementation (section 4). In addition, we must ensure that each TTP will be properly monitored to detect activity and collect logs, as explained in section 3.4.

3.3 Position

The second parameter used to implement the CKC of our honeypots is their position. For that, we consider a very simple one-dimensional model of our network. This model is entirely defined by the accessibility and exposure of the honeypot on the internet. We define four categories of depth:

Entry Level The honeypots on this position are directly accessible form the outside of our network. They form the first line attackers can probe and compromise in order to access the deeper layers, intermediary or DMZ. The entry Level honeypots are generally accessible from the Internet, but this is not an absolute requirement. They should be accessible to the attacker according to the threat model that needs to be monitored.

Internal Level The internal level honeypots are not accessible or even visible from the outside of the network, contrary to the entry level ones. They do not expose services the internet and are not even accessible from the Entry Level equipment.

Intermediary Level The third position level, the honeypots on the intermediary layer, have access simultaneously to Entry and Internal level honeypots. They serve as rebound points for the lateral movement phase of the attack.

Our honeypots are assigned to one of these positions. A honeypot's position imposes strong constraints on three elements: addresses management, visibility and supervision. IP ranges and subnets are not necessarily shared between honeypots of different levels and their IP address choice must be taken into account in the configuration and in the "cost" of an honeypot, as addresses have not the same scarcity in each network section.

The visibility constraints rely on the fact that an attacker attempting to access to another honeypot must perform reconnaissance techniques in order to get information about its targets. This constraint must be taken in account when choosing the TTP allowed by the honeypot deployed in an intermediary or internal position, allowing some reconnaissance or discovery techniques to be performed. The position of an honeypot thus plays a significant role in the determination of the TTPs we want to "enable" attackers to perform.

3.4 Supervision

We ensure honeypot integrity by mandating strong supervision of all honeypot components. We restrict the exposed services in order to enable specific TTPs and to setup adapted supervision systems. The deployment of any honeypot service is coupled with an appropriate specific supervision mechanism.

Supervision covers two dimensions. First, all the communications on the network level of the honeypot are monitored. Second, the internal state of the different machines hosting honeypots and the changes in the equipment themselves are logged. To achieve this, we define two categories of supervision mechanisms.

General Supervision (GS) ensures the overall integrity of the honeypot, and serves as a backbone for monitoring, data collection, and data storage and analysis. It monitors all the shared metrics of the honeypot. General supervision includes a specific environment (container in our implementation) for data management purposes, sensors for network filtering and monitoring for all network segments, and host-based agents for operating system monitoring.

Targeted supervision (TS) monitors a specific CAPEC or CVE in a specific honeypot (again container in our implementation), that we have chosen to offer for exploitation by the attacker. This dedicated host-based monitoring ensures that the specific exploitation details are logged when the attacker attempts to exploit the CAPEC or CVE in question. This monitoring can take the form of specific configurations added to the General Supervision, or of additional monitoring tools. TS logs are fed into the GS data collection system for full integration.

Figure 2 shows the two levels of design of our architecture and the associated elements and supervision.

Regardless of the technical solution used to implement supervision, its role is uniquely to monitor and log every event, and absolutely not to prevent any action from the attacker, what would harm our capacity to detect and log malicious activity.

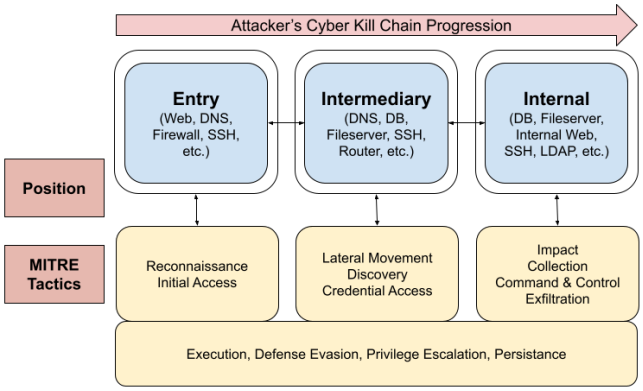

Figure 1: Parallels between progression in space and CKC for the attacker. Honeypots can belong to four positions according to their access to outer network. These positions reflect real network organisation and attackers progression on the Cyber Kill Chain

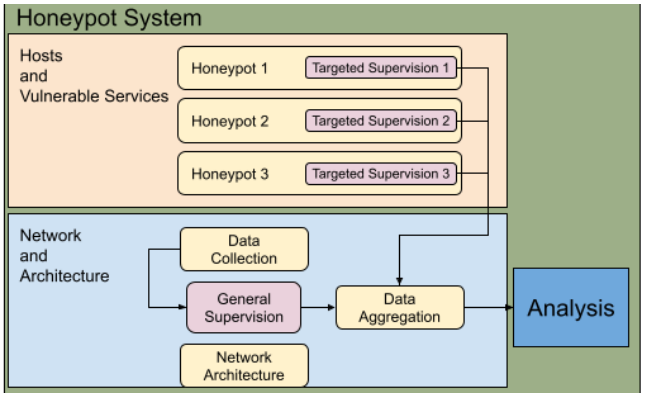

Figure 2: Levels of the honeypot architecture and constraints. The general supervision is a baseline at the Network and Architecture level, constrained by its components. On the opposite, honeypots' configuration on the Hosts and Vulnerable services are linked by their targeted supervision which is an integral part of the honeypot

4 PROTOTYPE IMPLEMENTATION

Our implementation is based on containers, using *iptables*, *wazuh* and an *elastic* stack [4] for log management.

4.1 Honeypot selection

Our prototype honeypot includes three simulated systems, hosted in containers. We use a virtualized infrastructure, with a baseline Virtual Machine (VM) embedding the supervision and data aggregation tools, and individual LXC containers [5] inside this VM simulating vulnerable systems. This allows us to use relatively modest hardware, a generalist dedicated server with 8GB of ram and a single multi-core processor. We also use only one public IP address.

We have chosen to deploy three honeypots at three different positions (entry, intermediary and internal). Scottberg and al. present different strategies for honeypot deployment [27], depending on the structure and the inclusion of the honeypots inside of a real network. According to his taxonomy, we deployed our honeypot as an *"Hacker Zoo"*, setting up an entirely deceptive environment filled with several equipment and services.

The first honeypot is an Ubuntu Linux Container exposed to the internet. It hosts a vulnerable implementation of the Apache Web Server and an up to date OpenSSH service configured with a common user/password that could allow attackers to brute-force the credentials. Concerning remote authentication, we also set up a Remote Desktop Protocol server for management and supervision purposes, but we took care of securing it by using an up to date version and robust credentials: that is why we expose an RDP server and monitor its activity, but we do not expect attackers to access to our system by this mean. The intermediary honeypot exposes a vulnerable implementation of Apache Druid, a database. The third honeypot, simulating the internal level, hosts a simulated DNS and LDAP server.

Our implementation is based around two objectives: easy portability and easy reset. Our objective is to be able to deploy our prototype virtual infrastructure on any server with a basic hypervisor and two network interfaces. The first one will be used for management and data collection. The second one must be accessible to attackers according to the position of the entry point, hosting the honeypots. In our current deployment, these two interfaces must be accessible from the Internet. For that, we embedded our honeypots in a single virtual machine which runs several LXC containers which host our honeypots and the supervision system. We use an Ubuntu Server 20.04.4 as our OS for the VM, hosted by an Oracle VirtualBox Manager. In order to simulate the progression of an attacker through the different position levels, we created a set of IPTables rules in order to restrict connections: internal honeypots are only accessible through intermediary ones, which are only accessible from entry ones. Figure 3 shows the architecture of our experiment's system. The description of each one of our honeypots according to our model, with an emphasis on vulnerability, supervision and position is shown in figure 4

4.2 Honeypot supervision implementation

We have implemented in our prototype a General Supervision monitoring general network traffic and covering all the components of the honeypot, as follows:

General Supervision is implemented using three components.

IPTables logs every new connection's IP packet's header. This information is useful by itself, and its correlation with higher level data obtained from the Targeted Supervision can allow us to make analysis with more detail.

Operating systems and containers are monitored using Wazuh, an open source XDR. It works with an Agent/Server architecture: every honeypot has an agent which is responsible for pushing log data to a server that will parse the data, store it and raise alerts. For its configuration, we choose to cover the largest perimeter, activating all the available alerts

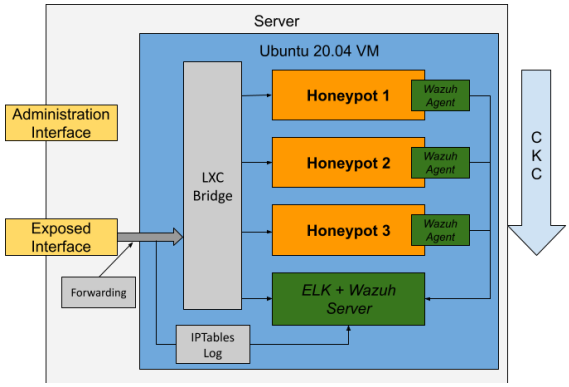

Figure 3: Architecture of our experimental honeypot. We integrate three honeypots in three LXC containers embedded in an Ubuntu virtual machine. The VM also handles the supervision and the data aggregation tools.

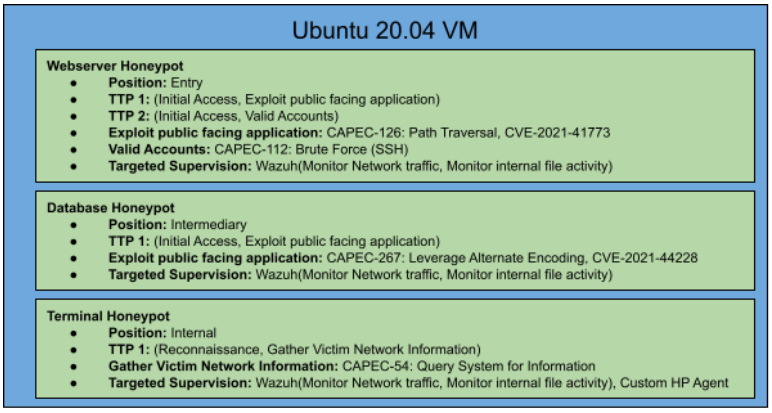

Figure 4: Details of the honeypots. Each exposed system is presented according to our model, empathizing on vulnerabilities, supervision and position

in order to collect all kind of information that could be useful for analysis.
Alerts and logs are managed using an Elasticsearch-Logstash-Kibana (ELK) Stack. This allows us to gather, manage and visualize our data in a convenient way.

Targeted Supervision is implemented by leveraging the Wazuh agent to collect and feed additional logs to the data management contained. We thus include logs from the SSH and HTTP daemons of honeypot 1. The targeted supervisions of honeypots 2 and 3 are covered by the default Wazuh configuration

Our honeypot may sound small in size, and indeed it is. However, the overall implementation based on containers and virtual machines enables some scalability. More importantly, it creates an environment where resources that are usually scarce in enterprise environments, such as routable IP addresses, are not necessary (or only in very limited numbers - one public IP in our case). This enables the creation of an entirely fictitious information system, that may not share the address plan of the environment it resides in. Thus, our implementation lifts this limitation of current honeypots or honeynets, the use of scarce (and pricey) resources.

4.3 Discussion

While the choice of IP tables as network sensor may appear limited, it enables the desired tight control and log over what is allowed to the attacker inside the network, while being sufficiently lightweight. We examined the potential use of an open source network intrusion detection system such as Suricata, but decided that the required additional coverage was properly provided by the wazuh agents in each honeypot.

5 RESULTS AND ANALYSIS

5.1 Overview

In order to test our concept, we exposed our honeypot on a public address during seventeen days, between 20/06/2022 and 06/07/2022. We collected almost 1.5 million events during this time frame, distributed between a constant noise of activity and occasional peaks from specific actors, as shown in figure 5.

These quantitative results are comparable to those of other honeypots exposing similar services, averaging the same order of magnitude of around one million hits per month[21, 26], even if it is hard to compare figures between experiments, as many variables may impact the number of hits. Nonetheless, we can state that the volume of data we collected fits with what is expected according to the literature.

During the time of exposition, no attacker was able to compromise our entry level honeypot. No rebound was performed, so all the data we analyzed focuses on the attempts to gain access to the first honeypot (Web server and SSH).

Each event we report corresponds to a new TCP connection established or to it's impact on the entry level honeypot. Our GS provide us information about the first TCP header of a stream, granting significant information for a quantitative analysis. The volume of these alerts allows us to identify 1492319 effective events giving us information about incoming IP packets.

We analyzed our data in three steps. First, we will discuss source IP addresses and attack attribution. Then, we will analyze the targeted ports and services in order to discuss about attackers' profiles. At last, we will analyse qualitatively the SSH brute force attempts before presenting some IoCs we were able to extract.

5.2 Attack Actors

The first interesting data is the source IP of the attack. We have aggregated 35971 unique addresses. Table 1 shows the top source addresses, identifying the most active actors. As some of them may hide behind several addresses, we aggregate data from IPs belonging to the same registered entity according to WHOIS information. We focus on actors representing more than 14000 events, either from an unique IP or from a registered address range.

As we see, there is a clearly dominant address in our analyzed data. It generated by far the highest number of connections to our honeypot. This address is allocated to "*China Telecom CHINANET Chongqing Province Network*", in China. It seems that the traffic from its address emerges from a Chinese data center. All these attributions show predictable actors unless the ones to RECYBER and Censys. All the other IPs belong to referenced ISPs or hosting providers so their traffic can be considered purely malicious. Concerning RECYBER, the *whois* description includes the following "remarks" field:

> *This net-block is not trying to hack you, we are only scanning for LEGIT purposes ONLY. This scanning is done by multiple security organizations. Please use https://www.recyber.net/opt-out to have your ip-address and/or netblock/as number white-listed and excluded from this project. If you have any further questions please contact email@recyber.net*

This indicates that this actor is actually not a real attacker, but an organization automatically scanning the internet, either for research or intelligence purposes. Nonetheless, even considering that the traffic is allegedly benign, we can express doubts about the necessity of almost 40000 probes merely for scanning purpose.

The distribution of the attacks performed by these IPs over time shows that despite a relatively homogeneous distribution, several high activity actors performed attacks on a concentrated period of time. We present the distribution of attacks by half-day and actor in Figure 6.

These IP ranges belong either to Service Providers or to organizations performing massive scale internet scans. It is hard to ascertain that related addresses belonging to the same provider are being used by the same actors, but the similarity of the attack patterns indicates so. Research organizations affirm that their traffic is only generated for scientific purpose, but among the vast majority of scanning traffic, we observe more focused attempts on the RDP and SSH ports. This latter activity implies at least some attempts of exploitation of insecure configuration. Regardless of its motivation, the traffic generated by allegedly genuine research organizations represents a significant percentage of the new connections we logged.

5.3 Services Targeted by Attackers

In this part, we focus on the destination ports targeted by attackers, which allows us to determine the intentions of attackers. Generally, we can sort our data by most used destination port, taking the ten most targeted ones, as shown in table 2.

Ports associated with brute-force attacks represent the a majority of the new connections established, with 56% of them targeting RDP and SSH services. As for data volume, these results are coherent with previous deployments made by researchers. Contrary to our SSH server, our RDP server was implemented with secure credentials and an up to date implementation as we wanted to limit our remote session attack surface to a single component. Even if several RDP related CVEs were published during the months before our experiment, we cannot correlate any of them to the activity we monitored. The important amount of RDP attempts is likely the result of opportunistic and unsuccessful activity.

The correlation between top IP addresses and ports displays three main profiles of attackers and operations, shown in table 3.

- **The RDP brute-forcers** Two of our three main IPs (Chinanet1 and Microsoft) trigger numerous RDP communications in a short period of time, attempting numerous new connections to port 3389 associated with the RDP protocol, in a clear brute-force attempt to identify the correct credentials combination to gain access to the honeypot.
- **The SSH brute-forcers** The IPs Chine United Networks, OVH and Chinanet2 have a similar behavior, but instead of RDP, the brute-force attempts target the SSH service on our entry level honeypot.
- **The scanners** The remaining IPs perform scanning campaigns. They target a large amount of different ports, each a few times, performing from one to forty tries for each one. They objective is not as straightforward to identify, as they could be motivated by legitimate intentions (research and cartography) or gathering information in order to perform more focused attacks. While former experiments implying honeypots deployment showed limiter port probing from scanners, our scanners seem to cover an extensive amount of ports. Dacier[12] reported 188 ports probed, while we observe tens of thousands.

This data and this classification of actors shows that the Tactics enabled for attackers in our entry level equipment, *Initial Access* and

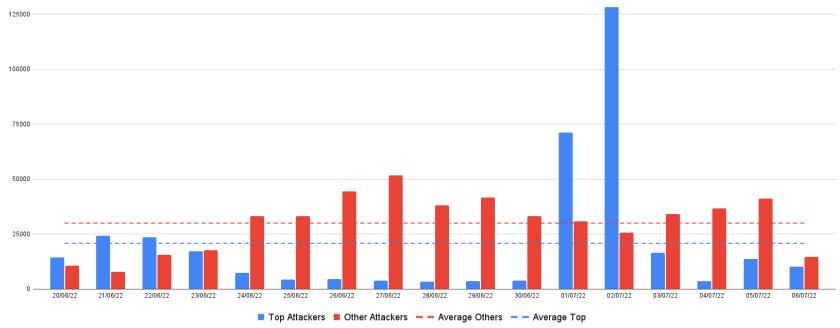

Figure 5: Time distribution of attacks from the 8 main attackers compared to all the other traffic, showed as events per day. The distribution shows localized activity spikes from top attackers, and a constant baseline activity

Actor	Nb of Events	Nb of IPs	Nb of DestPorts
Chinanet-1	113 610	1	1
Recyber	42 381	33	20078
Microsoft (UK)	38 480	1	1
Internet Solutions & Innovations LTD (ISI LTD)	26 834	8	12754
China United Networks	22 794	1	1
OVH (France)	21 256	1	1
Chinanet-2	15 268	1	1
Censys	14 122	182	6746

Table 1: Table representing the main actors that targeted our honeypot. It shows their name, their number of events, the number of IP addresses belonging to them that we identified and the number of distinct destination ports they tried to reach

Port	Number of events	Service
3389	228 482	Remote Desktop Protocol
22	222 306	Secure SHell
445	71 592	Microsoft attack detection/SMB
23	8 010	Telnet
6379	7 660	Redis Server
2222	6 136	Alternative SSH
80	3 341	HTTP
8080	2 149	HTTP
443	2 664	HTTPS
1433	2 156	SQL Server

Table 2: Number of events for the ten most targeted ports, and their associated service. Ports associated to remote authentication were heavily targeted in comparison to those linked to other services

Actor	Attack Procedure
Chinanet-1	RDP Bruteforce
Recyber	Port Scanning
Microsoft	RDP Bruteforce
ISI LTD	Port Scanning
China United Networks	SSH Bruteforce
OVH	SSH Bruteforce
Chinanet-2	SSH Bruteforce
Censys	Port Scanning

Table 3: Attack procedure per actor. The three main attack procedures are RDP Bruteforce, SSH Bruteforce and Port Scanning

Reconnaissance, were effectively exploited. The tactics we recorded include Active Scanning for Reconnaissance, as we don't know if any other external method was used, and Brute-Force and Valid Accounts for Credential Access and Initial Access. We recorded an effective TTP usage of the solutions enabled for the attackers. We notice that brute-forcers are completely focused on the service they target, with all their traffic directed to it.

While analyzing network level data, we noticed a peculiar behaviour. While the majority of the TCP traffic uses a random source port in the range allocated for this purpose, a few attacks use only a very small number of source ports. This likely indicate some kind of masquerading, implying the operations are run by several machines behind the same network equipment. For example, almost all the traffic from RECYBER's scan origins from source ports 46194 and 43304.

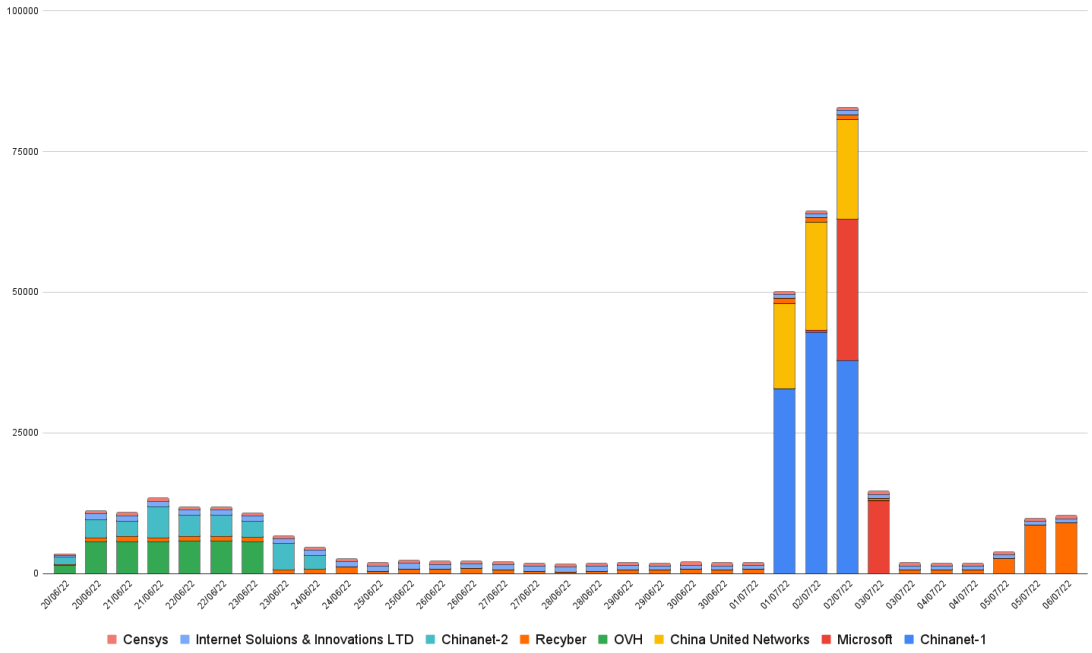

Figure 6: Time distribution of attacks from the 8 main attackers with the number of events per half-day for each actor.A few actors perform short campaign-like operations during a few days, while others maintain a continuous activity over the observation period

5.4 SSH Bruteforce

We recorded the usernames used in the brute-force SSH connection attempts. We identified 24782 unique usernames in almost 299k events. This volume is similar to the observations by Alata's high interaction SSH honeypot [8], but while they obtained 249k events in 131 days, we reached a higher amount in only 17 days. They were probed with more than 40K different usernames and more that 340 bruteforce campaigns were successful, while no attacker was able to guess our insecure password. In addition to these statistics, the usernames that were probed by attackers differ slightly in our work compared to theirs. The eight most used users in our experiment were listed in the table 4, compared with Alata's observations in 2006.

These results are in line with expectations, as the used credentials are expected. Attackers focus on priority well known or default credentials then focus on specific services names susceptible to be installed in the machine. The clearly dominant user name is "root", by a significant margin. We can notice that the unsafe username we set up for our honeypot, "*user*", was only tried around two thousand times, which may indicate why our credentials were not guessed, in addition to the country-specific choice we made for our password.

We also observed a strange pattern of usernames. Usually, brute-forcers perform dictionary attacks, consisting in the repeated test of credentials from a predefined list of words, but instead of enumerating different possible passwords for the same username, they enumerated different usernames, attempting a single try for each of them. We first suspected a method for identifying valid accounts, as some authentication systems present the known flaw of returning different error messages in case of incorrect password and unknown user, but it is not the case for our OpenSSH implementation. Our best hypothesis is human error in the configuration or usage of the automated tools by attackers, which is, as the chosen attack method, an indication of poor technicality and motivation from the SSH brute-forcer.

User	Attempts	Alata's observations and rank
root	157713	34121 (1/41530)
admin	17133	4007 (2/41530)
user	2524	1247 (4/41530)
test	2492	3109 (3/41530)
ubuntu	1972	-
1111	1366	-
oracle	1173	857 (8/41530)
ftbpuser	1074	-
postgres	1029	834 (9/41530)

Table 4: Main Usernames used for brute force attempts. We notice a clear dominance of "root", then common usernames, often linked to specific products or services. For comparison, we include the results obtained by Alata in the previously cited work

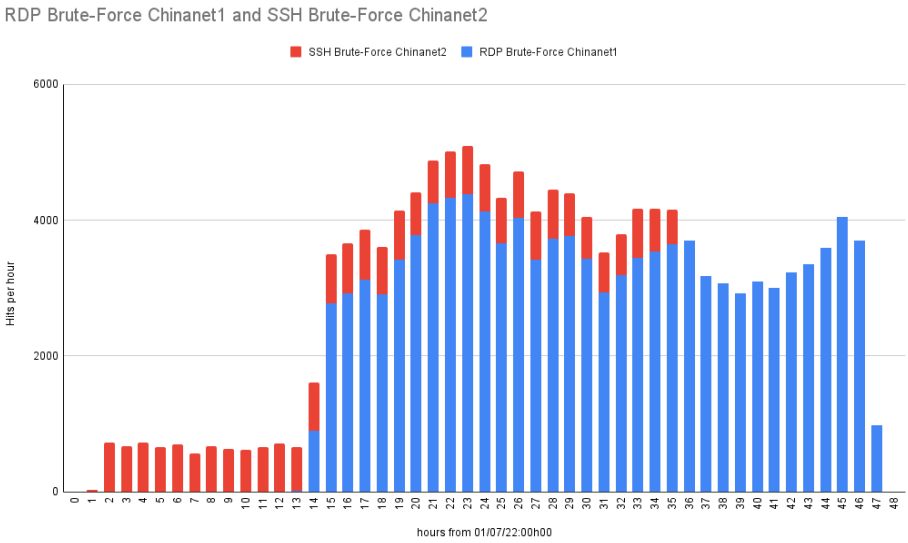

Figure 7: Time distribution, hour by hour, of the activity of the two more active brute-forcers. A high number of attempts were made during the 48h following the 01/07/2022

Ip Address	Attribution	Alerts	Comments
61.128.X.X	Chinanet1	1/94	1 Malware flag
51.142.X.X	Microsoft	0/94	No detection
116.131.X.X	China United Networks	4/94	3 Malware flags, 1 Malicious flag
54.38.X.X	OVH	0/94	No detection
61.177.X.X	Chinanet2	4/94	1 Malware flag, 3 Malicious flags

Table 5: Result of the Virustotal analysis of brute-forcer's IPs. These IPs are not identified as malicious by the VirusTotal tools.

We also analyze the details from the two China based brute force attacks. The RDP attack and the SSH attack by the IPs Chinanet1 and Chinanet2 were performed at the same moment, in a really narrow frame of time, as shown in Figure 7. Even without categorical evidence indicating a coordinated attack by the same actor using different source IPs, this indicates a potential coordinated attack, as shown in the Figure 7 histogram.

5.5 IoCs obtained

A secondary objective of our experiment was the potential determination of previously unknown Indicators of Compromise. We retrieved information about the unique IP addresses responsible of the most active brute force attacks from the community reference site Virustotal[6]. This site is an aggregator allowing the analysis of files, URLs and other indicators by a set of 94 (at the date of the experiment) antivirus and tools. The results are listed in table 5.

Table 5 shows that only the Chinese actors are identified as malicious or malware, but that a majority vote would lead to concluding that these addresses are innocuous. This might indicate that IP addresses have become less useful as IoCs, as attackers use hosting services (using compromised or even rented infrastructures) and move sufficiently rapidly to new infrastructures that their footprint remains under the detection threshold for most of the VirusTotal tools.

5.6 Lack of successful advanced penetration

The analysis of our data shows that despite the vulnerabilities integrated to our honeypot, no attacker was able to gain access inside our entry honeypot, and thus to perform rebound attacks inside of our honeypot system. While our honeypot included an unsecured SSH server with unsafe credentials (user:marseille) picked in the top 10 most used passwords used in France[9], no brute force attempt was successful at the time of guessing them. Even if we discovered pertinent data concerning the Entry level honeypot, we were unable to retrieve information about deeper honeypots in our architecture. That could be explained by the relative low interest of our honeypot, that is not correlated to any other organization or attractive asset, or by mere lack of success from the attackers who chose to target us, which seems to show low perseverance and technicality.

6 CONCLUSION

Taking our model and its constraints into account, we were able to build an honeypot and expose it to the internet with robust supervision and data collection features. This honeypot attracted hostile activity. It is hard for us to quantify the relative attractiveness of our honeypot in comparison to real critical systems because our deployment was made in an anonymous public hosting environment, but we collected an amount of data important enough for us to perform a detailed manual analysis. We were able to confirm our first two questions: it is possible to deploy for a low cost a honeypot based on the model we elaborated.

With this honeypot, we were able to detect and analyze hostile activity. This informed us about general trends on automated attack and allowed us to identify specific IP ranges as suspect or malevolent.

Our experiment have a few limitations. First of all, no attacker was able to pass through our poorly configured security measures. In addition, We studied the most relevant TTPs to expose in our machine, chose what version of an OS or a service we wanted to install in our honeypots, and thus what vulnerabilities we wanted to expose. Since the version of a service in an honeypot is fixed during the time period of its deployment, new vulnerabilities affecting the honeypot could emerge during the experiment phase. We didn't notice any new emerging vulnerability, but this has to be taken into account for longer honeypot deployments.

One last limitation can be found in the volume of data. Even if it is consequent for human analysis, 1.5M events are not enough for a effective automated analysis to be performed. The volume was limited by the fact that our experiment was sized for a certain volume of data, and that we had few hints about the average daily volume we would receive.

Further work could be performed on the same type of honeypot with a few differences, notably we could slightly reduce the security of the entry level honeypot, notably choosing a more internationally *guessable* password. Additional improvement could be made on the supervision, notably by monitoring extra services or outbound traffic and dimension the storage in order to gather larger datasets during a longer period of time so we could implement automated analysis using the larger dataset obtained.

REFERENCES

[1] 2022. Matrix - Enterprise | MITRE ATT&CK®. https://attack.mitre.org/versions/v11/matrices/enterprise/

[2] 2022. Purplesec 2022 Cyber Security Statistics Trends & Data. https://purplesec.us/resources/cyber-security-statistics/

[3] 2023. CAPEC - Common Attack Pattern Enumerations and Classifications. https://capec.mitre.org/

[4] 2023. Elastic, https://www.elastic.co/.

[5] 2023. Linux Containers. https://linuxcontainers.org/lxc/

[6] 2023. VirusTotal. https://www.virustotal.com/gui/home/upload

[7] Palvi Aggarwal and Gonzalez et al. 2016. Cyber-Security: Role of Deception in Cyber-Attack Detection. Cham. https://doi.org/10.1007/978-3-319-41932-9_8

[8] E. Alata, V. Nicomette, and M. et al. Kaaniche. 2006. Lessons learned from the deployment of a high-interaction honeypot. https://doi.org/10.1109/EDCC.2006.17

[9] Maxime Alay-Eddine. 2023. Richelieu. https://github.com/tarraschk/richelieu

[10] Commvault. 2023. *Metallic.io Threat Wise.* https://metallic.io/threatwise-cyber-deception

[11] CounterCraft. 2023. *CounterCraft.* https://www.countercraftsec.com/

[12] Marc Dacier and F. Pouget. 2012. Attack processes found on the internet. (03 2012).

[13] DinoTools. 2021. *Dionaea.* https://github.com/DinoTools/dionaea

[14] Lim et al. Djap, Ryandy. 2021. XB-Pot: Revealing Honeypot-based Attacker's Behaviors. https://doi.org/10.1109/ICoICT52021.2021.9527422

[15] Shade et al. Ferguson-Walter, Kimberly. 2018. *The Tularosa Study: An Experimental Design and Implementation to Quantify the Effectiveness of Cyber Deception.* Technical Report.

[16] Kheir et al. Han, Xiao. 2017. Evaluation of Deception-Based Web Attacks Detection. Dallas Texas USA. https://doi.org/10.1145/3140549.3140555

[17] et al Jorquera Valero. 2020. *Identification and Classification of Cyber Threats Through SSH Honeypot Systems:*. https://doi.org/10.4018/978-1-7998-2242-4.ch006

[18] Marty Roesch Lance Spitzner. 2001. The value of honeypots, part one: Definitions and values of honeypots. http://www.symantec.com/connect/articles/value-honeypots-part-one-definitions-and-values-honeypots

[19] C. Leita, V.H. Pham, and O. et al. Thonnard. 2008. The Leurre.com Project: Collecting Internet Threats Information Using a Worldwide Distributed Honeynet. Amsterdam. https://doi.org/10.1109/WISTDCS.2008.8

[20] Lockheed Martin. 2015. Gaining the advantage. https://www.lockheedmartin.com/content/dam/lockheed-martin/rms/documents/cyber/Gaining_the_Advantage_Cyber_Kill_Chain.pdf

[21] Ryan J McCaughey. 2017. *Deception using an SSH honeypot.* Technical Report. Naval Postgraduate School Monterey United States.

[22] Marcin et al. Nawrocki. 2016. A Survey on Honeypot Software and Data Analysis. http://arxiv.org/abs/1608.06249

[23] Kaâniche et al. Nicomette, Vincent. 2011. Set-up and deployment of a high-interaction honeypot: experiment and lessons learned. *Journal in Computer Virology* (2011). https://doi.org/10.1007/s11416-010-0144-2

[24] IBM Security Ponemon Institute LLC. 2020. Cost of a Data Breach Report 2020. (2020). https://www.ibm.com/downloads/cas/RZAX14GX

[25] Karthikeyan .K. R and A. Indra. 2010. Intrusion Detection Tools and Techniques –A Survey. *International Journal of Computer Theory and Engineering* (2010), 901–906. https://doi.org/10.7763/IJCTE.2010.V2.260

[26] Ryandy, Charles Lim, and Kalpin Erlangga Silaen. 2020. XT-Pot: EXposing Threat Category of Honeypot-Based Attacks. https://doi.org/10.1145/3429789.3429868

[27] B. Scottberg, W. Yurcik, and D. Doss. 2002. Internet honeypots: protection or entrapment? https://doi.org/10.1109/ISTAS.2002.1013842

[28] Lance Spitzner. 2003. *Honeypots: tracking hackers* (1. print ed.). Addison-Wesley, Boston Munich.

[29] Applebaum A. Strom B.E. 2020. MITRE ATT&CK: Design and Philosophy. https://attack.mitre.org/docs/ATTACK_Design_and_Philosophy_March_2020.pdf

[30] Solomon Z. Melese and P.S. Avadhani. 2016. Honeypot System for Attacks on SSH Protocol. *International Journal of Computer Network and Information Security* (2016). https://doi.org/10.5815/ijcnis.2016.09.03

[31] Li Zhang and Vrizlynn. L. L. Thing. 2021. Three decades of deception techniques in active cyber defense - Retrospect and outlook. (2021). https://doi.org/10.1016/j.cose.2021.102288