A distributed plateform of high interaction honeypots and experimental results (extended version)

Ivan Studnia, Vincent Nicomette, Mohamed Kaâniche, Eric Alata

A distributed platform of high interaction honeypots and experimental results (Extended Version)

Ivan Studnia[1,2], Vincent Nicomette[1,3], Mohamed Kaâniche[1,2], Eric Alata[1,3]
[1]CNRS, LAAS, 7 Avenue du colonel Roche, F-31400 Toulouse, France,
[2]Univ. Toulouse, LAAS, F-31400 Toulouse, France
[3]Univ. Toulouse, INSA, LAAS, F-31400 Toulouse, France
Email: {studnia,nicomett,kaaniche,ealata}@laas.fr

***Abstract*—The increase of various malicious activities spreading on the Internet network are today a crucial problem. In order to understand the motivations and operating modes of the attackers, it is necessary to collect data characterising these malicious activities. Their analysis enables to better face these attacks, anticipate new threats and better adpat the corresponding protection mechanisms. This paper proposes a distributed platform oh high interaction honeypots deployed for that purpose. The paper describes 1) the design and implementation of this platform, 2) the methodology used to collect and record data characterising the malicious activities and 3) the first analyses carried out on this data.**

I. Introduction

The Internet fast paced development (about two billion users in 2010 according to the Internet World Stats[1]) permitted the emergence of many online services and many communities of Internet users. This network is now so important that rules and even laws have been created in order to ensure its right behavior. Indeed, some people, through exploits of hardware or software vulnerabilities, misuse computers for malicious purposes, e.g. to obtain private information, take over computers or spread some malware (worms, trojans...). Therefore, countermeasures have been designed in attempts to fix these weaknesses and block the attacks, continuously forcing the hackers to find new vulnerabilites and new ways of using them. Thus, attackers and security experts are in a never-ending attack-defense race[1].

Today, it is crucial to know the strategies currently used by the attackers in order to efficiently counter incoming attacks and to design new, adapted protection mechanisms. To do so, one should collect the most recently available information about hackers. There are currently different ways used to do so. For example:

1) Collection and processing of the information issued by networking and security devices like routers or firewalls[2][3].
2) Setting up of "decoys" [2], in order to trap and monitor attackers' activities.

In this paper, we focus on the automation of the latter, through the concept of honeypots. The common definition was given by L. Spitzner [3]: *A honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource.* Here, this term refers to a computing system, connected to the Internet, deliberately designed to be vulnerable in order to be an attractive target for attacks so that we can analyze their characteristics (protocols being used, exploited weaknesses, executed programs...). There are many different ways to design a honeypot [4], [5]. They are usually grouped within two categories: high (for example [6]) or low (as in [7]) interaction, depending on the possibilities given to an attacker, although new trends are appearing (like in [8], [9] or [10]). We propose in this paper the description of a data collection platform using various high interaction honeypots set in different locations, along with our first analyses of this data. This deployment follows a previous experiment conducted with the same honeypot that was deployed at a single location (cf. [11]). The objective is to check if the results of our first experiment can be generalized.

This paper is arranged as follows. Section II presents an overview of the high interaction honeypot that we conceived and implemented within these experiments. Then the collected data and the way it is stored and sorted are described in section III. Section IV details the architecture of the distributed honeypot platform and explains how it works. deployed for this experiment. Section V presents the results of the first analyses performed on the collected data. Finally, section VI will conclude this work.

II. High interaction honeypot

The design and implementation of our high interaction honeypot is not the main contribution of this paper. As a consequence, we only present its main characteristics in this section. More detailed information can be found in [11].

Our high-interaction honeypot is especially designed to record malicious activities carried out by human beings. However it is also able to record activites carried out by automatic tools. For that purpose, we included vulnerabilities that can be easily exploited by human beings. Our choice was to use the `GNU/Linux` operating system and create user accounts with weak passwords, accessible through the `ssh` service.

So as to obtain the highest interaction level possible, several hosts can be accessed by the attackers. As "real" systems are

[1]www.internetworldstats.com/stats.htm
[2]www.dshield.org
[3]www.symantec.com/about/profile/universityresearch/sharing.jsp

costly and complex to manage, we used virtual hosts. The information recorded to analyse attack scenarios, is:

- The pairs (login/passowrd) tested by the intruder.
- The keystrokes typed by the intruder as well as the text displayed on his/her terminal. These characters allow us to reconstruct the commands entered by the intruder.
- The system calls generated by the activity of the attackers. This can be useful if the capture of the keystrokes entered by the intruder is not sufficient to identify the commands executed by the attacker (use of shortcuts, or program that calls other programs).

The kernel of each virtual machine is patched at two places. We instrumented: 1) the functions of the `tty` driver that allow us to record all the keystrokes and characters typed by the intruder while he/she has successfully penetrated the system and uses an interactive shell, and 2) the `exec` system call that allows us to intercept each system call executed by the intruder.

Additionally, in order to capture the usernames and the passwords tested by the attackers, we created a new system call in the kernel and modified the SSH server accordingly to use this new system call. All this information is logged on a dedicated area of the virtual host kernel memory. This choice is motivated by the fact that most of the patched code runs in privileged mode (ring 0) and has direct access to kernel memory space.

This collected information is then periodically copied into the real host: the content of the virtual host memory is directly accessible from the real host. The main advantage of this backup strategy is that it is difficult to identify by the attacker. It is a simple copy of data from a memory region to the hard disk of the real host and it is carried out only once a day at a fixed date (this operation lasts less than one second). Finally, the collected information is transferred to a database server where the analyses can be performed.

Figure 1 presents the architecture of the honeypot.

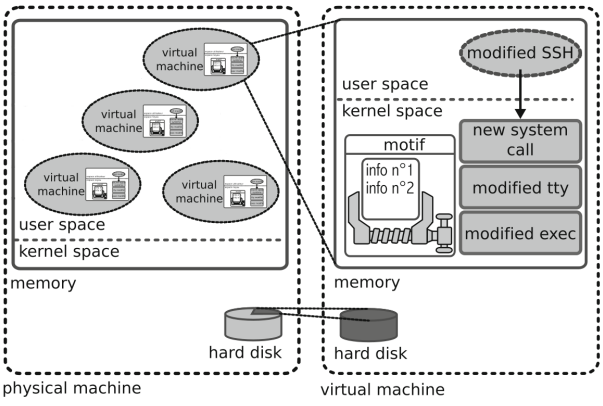

Fig. 1. Honeypot architecture

III. Data collection and storage

Data recorded by the honeypots is stored in a database and sorted to ease future accesses and subsequent analyses.

The database structure is summed up in figure 2. We will only describe here the most important tables used for our analyses.

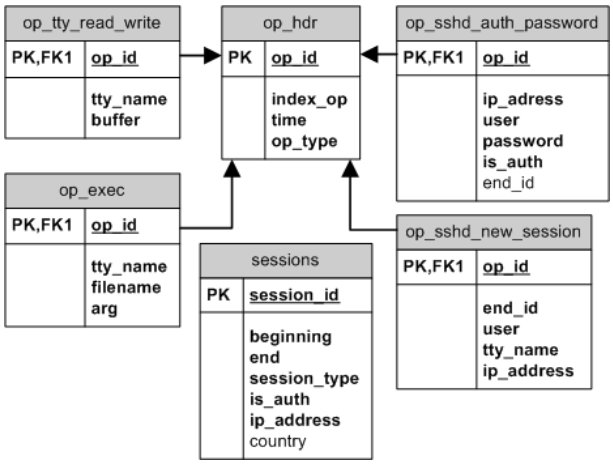

Fig. 2. Schema of the database corresponding to one honeypot

Table `op_sshd_auth_password` contains the data concerning every connection attempt. The meaning of each field is described hereafter:

- `ip_address` is the attacker's source IP address.
- `user` is the attempted login.
- `password` is the attempted password.
- `is_auth` shows whether the connection succeeded or not.
- `end_id` represents, in case of a successful connection, the id of the latest recorded action during the corresponding attack.

Table `op_sshd_new_session` is updated whenever there is a successful connection. Quite similar to `op_sshd_auth_password`, it only contains information about the attacks during which a terminal has been opened. Column `tty_name` shows which terminal was assigned to the attacker. This information is used to link the data given by the patched `tty` driver to the one given by the `ssh` server.

Table `op_tty_read_write` contains all the information recorded by the `tty` driver. Each line includes the content of the `tty` buffer and the name of the terminal to which it belongs. When the data contains input typed by an attacker, the buffer will only contain one character at a time. However, if the input has been copied and pasted, the buffer will contain several characters. It is therefore easy to make a first observation about the attacker's behaviour.

Table `op_exec` contains data related to the programs executed by the host during an intrusion. Each line contains the name of the executed program, the parameters it received and the terminal in which it was run.

Once the database is filled, tables called `sessions` can be created and updated. These tables store data resulting from the grouping of `ssh` connections into attack sessions. Data is regrouped as follows : close in time `ssh` connections from

the same IP are gathered into sessions[4]. in order to recreate an attacker's activities. We can distinguish three categories :

- The attacker succesfully logged in and commands have been issued. These are called *intrusions*.
- No commands were issued, but a large amount of (login/password) where tried in a short period of time. These are called *dictionnary attacks*.
- All other cases fall into the *others* category. They probably correspond to misconfigurations of accidental errors.

`sessions` tables then consists of:

- `beginning` and `end` which are the ids of the starting and ending events of a session.
- `session_type` which indicates whether the attack is an intrusion, a dictionary attack or none of these.
- `is_auth` which is set to 1 if at least one connection attempt succeeded during the session and which is set to 0 otherwise.
- `ip_address` which is the attack source IP.
- `country` which is the country this IP is from.

A graphical user interface for managing the base was developped in order to ease the processing and the analyses of the data recorded in this database.

IV. Honeypots distributed platform

Our goal is to deploy honeypots into several places over the world in order to compare the behaviours of different attackers. Our aim is to check if some global trends emerge or if activities are instead related to the location of the targeted honeypot. For this experiment, we have had access to four machines, each one with its own public IP address, located in three different places: one in Toulouse, France, another one in Rennes, France, another one in College Park, Maryland and the last one at LAAS.

This section presents the architecture of our honeypot platform.

A. Architecture

We had two different alternatives to deploy the high interaction honeypot on the different locations. A first possibility is to replicate the architecture of the honeypots used during our previous experiment and to install it on a computer located in those different locations. However, this idea has three major drawbacks. First, deploying a high interaction honeypot remains a risky operation, because it lets an attacker actually operate on the real system and possibly make damages on this system. Therefore, deploying several honepots in different remote sites imposes a lot of security constraints and administration overhead, all the more as we do not control the remorte network where the honeupots will be installed and configured. Secondly, each honeypot must be deployed under identical conditions because different parameters could bias the comparative analyses. This is important and must be taken into account when designing the platform. Finally, we do not want the networks hosting our honeypots to be infected. This means that we do not allow an attacker to execute his commands on computers that are not under our control. Our system have to make an attacker believe he is connected on a computer located in Toulouse, Rennes or College Park whereas he actually interacts with one machine at LAAS.

Thus, we chose an approach in which an attacker's connections will be rerouted towards virtual machines installed on one particular computer located at LAAS. This way we can keep control on the honeypots, guaranteeing a sufficient amount of security and more efficient ways to react to possible threats. Moreover, this allows us to minimize the influence of the hardware and the networks at the different locations by using these remote machines only as relays to our local setup.

Figure 3 provides an overview of the deployed architecture. We have three machines located into remote networks and one set up at LAAS, each one with a public IP address. All act as relays to our virtual machines, named VM1 to VM4. These are installed on the same host which also simulates a local network for each of those virtual machines. `GRE` (Generic Routing Encapsulation, as described in RFC 2784[5]) tunnels are created between each relay and the host, which links each virtual machine to its corresponding tunnel. We must however ensure that the replies sent to the attackers by our virtual machines will follow the same route that the requests they recieved (that is, going back through the tunnels and the relay), otherwise they could be blocked by some firewalls. Finally, routing rules on the host and the relays enable some selected part of the traffic targeting the relays public IPs to be rerouted to the virtual machines.

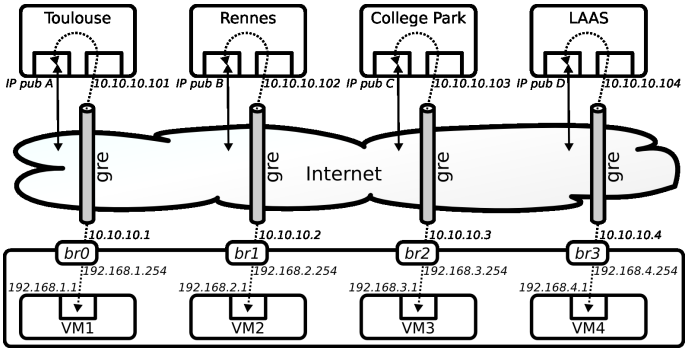

Fig. 3. Honeypots distributed platform architecture

B. Traffic management

Connections are monitored by the relays and the host at LAAS. Monitoring is done with `iptables`, allowing us to write rules to filter incoming and outgoing packets of a Linux system. The security policy enforced by the relays is as follows:

- Incoming connections on port 22 are allowed. It is the default port for `ssh`, and thus enables the attackers to connect to this service.

[4]This grouping is done according to a threshold for which we set the value at 20s, after analyses we will not detail here (see [12])

[5]www.ietf.org/rfc/rfc2784.txt.

- Outgoing connections through ports 53 (dns) and 123 (ntp) are allowed, so our virtual machines can use these services.
- An extra port plaas is opened only for machines with LAAS owned IPs, for remote administration purposes.
- GRE protocol is allowed in both directions.
- Every other connection attempts are dropped.

The security policy concerning the machine at LAAS hosting the virtual machines is as follows : La politique de sécurité au niveau de l'hôte des machines virtuelles est la suivante :

- GRE protocol is allowed in both directions.
- ssh connections on port plaas are allowed from LAAS, for administration purposes.
- ssh connections on port 22 are allowed from the tunnels to the virtual machines.
- dns and ntp are allowed from the virtual machines to the tunnels.

V. Experiment and data analysis

Our experiments were carried out in two steps. First, we deployed our honeypots for a month without any account created. This first step lasted from June 1^{st} to June 30^{th}. Analyses of the data collected during this month gave an overview of the login/password pairs tried by the attackers, as well as their frequency. These results have been used to create a list of accounts we knew to be often tried. The next step started with the creation of the aforementioned accounts on every honeypot. Thanks to these accounts, we hoped to quickly observe some intrusions. This second step started on July 1^{st}. In this section, we first give an overview of the results obtained during these two steps, distinguishing them if required. Then, we present some detailed analyses on this data. We consider the data recorded up to December 1^{st}.

A. Observed activities

In this part, we analyze the results of the observations made on the honeypots before any processing of the data. These analyses concern both steps of our experiment.

1) ssh *connections:*

Honeypot	Nb. connections	Nb. successful connections	Nb. distinct IPs
Toulouse	301948	58	385
Rennes	397462	119	387
College Park	10737	42	197
LAAS	150027	298	421
Total	860174	517	1207

TABLE I
Distribution of the ssh connections observed on the honeypots.

a) Overview: Table I shows how the observed connections are distributed between our honeypots on December 1^{st}. Each one of these connections corresponds to a login/password request sent to a ssh server.

The number of distinct addresses we get by considering all the honeypots together (1207) being smaller than the sum of the values obtained for each machine (1390), it appears that some IPs connected on more than one machine. Therefore, we will have to take these intersections into account during our analyses.

Table II shows which pairs were the most tried on each honeypot. Unsurprisingly, the root account (which is the administrator account on Linux) is the favorite target of the attacks, no matter which honeypot we consider. For the rest, we can see that the most tried pairs are quite simple, with a password often identical to the login. This confirms the observations made four years ago in [11]. We may then suppose that it is still common practice to use such logins/passwords (an attacker would have no interest in trying such combinations if they were very unusual). Moreover, there seems to be an interest for logins corresponding to default accounts usually created for the needs of some programs (oracle, mysql, postgres, nagios, etc.). Corresponding passwords are those initially created during the installation of such applications. It would be interesting to know if these programs are targeted only because they are widespread or if the interest lies in obtaining the files and the data they can use.

When we compare the results obtained on each honeypot, we can see some variations between the rankings, although some pairs appear in the first positions of every honeypot.

b) Calibration: At the end of the first phase, we identified the most tried combinations on each honeypot. Using this data (excluding root), we established a list of login/password pairs which has been used on all four honeypots. The pairs in this list were chosen according to the following criteria :

- Pairs being among the most tried ones on one honeypot but not on the others.
- Pairs often tried on every honeypot.
- Pairs in which the login differs from the password.
- Pairs in which the login is identical to the password.
- Pairs for which the login is related to software that could be installed on the computer (apache, mysql, etc.).

Using these criteria, we created the accounts shown in table III on every honeypot.

c) First connections: Let us call τ_1 the duration between the creation of an account and the first successful connection attempt to this account, and τ_2 the time spent since this attempt until the first connection on the same account where commands were entered. Values of τ_1 and τ_2 for every account created on our honeypots are given in table IV. In two cases, an attacker logged in with one of the 11 accounts succesfully managed to obtain root privileges through exploiting a system vulnerability. When this happened, we also give the time spent between the account creation and this event.

The table shows much higher values for τ_1 on the College Park honeypot. There are several possible explanations for this. First, the activity on this honeypot since the accounts creation has been less intense than on the three other ones. This could be due to the network configuration, which may implement some form of *rate limiting*, during a dictionary attack for example. Various tests indicate that this is not the case here, as it was later confirmed by the sysadmin of the corresponding site. We can also assume that the few seconds delay resulting from the rerouting of the requests from the

	Order	Toulouse Pair	Nb.	Rennes Pair	Nb.	College Park Pair	Nb.	LAAS Pair	Nb.
root included	1	root 123456	347	root 123456	386	root 123456	70	root 123456	247
	2	oracle oracle	282	root password	327	root root	61	root root	221
	3	root password	271	oracle oracle	304	root password	58	root password	203
	4	root qwerty	250	test test	297	root qwerty	47	root qwerty	201
	5	test test	247	root root	282	test test	36	oracle oracle	174
root excluded	1	oracle oracle	282	oracle oracle	304	test test	36	oracle oracle	174
	2	test test	247	test test	297	oracle oracle	32	test test	150
	3	mysql mysql	197	mysql mysql	257	admin admin	19	postgres postgres	140
	4	postgres postgres	193	postgres postgres	251	postgres postgres	16	mysql mysql	120
	5	test test123	159	user user	224	mysql mysql	13	admin admin	103

TABLE II
LIST OF THE 5 MOST TRIED PAIRS ON EACH HONEYPOT

Account	*login*	*pass*
C1	adam	adam
C2	alex	alex123
C3	apache	apache
C4	cary	cary
C5	eric	eric
C6	michael	michael
C7	mysql	mysql
C8	nagios	123456
C9	postgres	postgres
C10	test	test123
C11	user	password

TABLE III
LIST OF THE CREATED ACCOUNTS

Account	Toulouse		Rennes		C.P.		LAAS	
	τ_1	τ_2	τ_1	τ_2	τ_1	τ_2	τ_1	τ_2
C1	65h	-	48h	81h	80d	61h	25d	40d
C2	61h	-	48h	15d	81d	6h	14d	40d
C3	32h	-	72h	15d	73d	8d	33h	37h
C4	-	-	7d	14d	-	-	53d	30d
C5	65h	-	72h	99h	82d	-	6d	48d
C6	65h	70h	64h	7h	60d	-	6d	11d
C7	50h	1h	72h	6d	71d	-	55h	1h
C8	7d	19h	99h	99h	87d	-	100h	10d
C9	65h	70h	72h	28h	23h	6h	60h	2h
C10	56h	-	72h	9d	81d	-	55h	1h
C11	65h	-	72h	16d	-	-	7d	6d
root	6d		-		-		25d	

TABLE IV
τ_1 AND τ_2 DURATIONS FOR EACH ACCOUNT AND EACH HONEYPOT

United States to France was big enough to discourage part of the attackers. Finally, we can not rule out the hypothesis of an attacker being able to unveil the honeypot, putting its IP address on a "blacklist" of machines to be ignored.

We can see on this table that every account that has been found was not necessarily attacked afterwards. This may be due to several causes. First, as we deliberately chose to use frequently tried combinations, several of them may be found in one attack session (10 valid pairs were found during one single dictionary attack targetting the honeypot located in Toulouse). It happened that the attacker logged in with just one account then changed the password of the other ones from there, preventing other attackers from connecting afterwards. For example, we observed an attacker who changed 6 passwords in a row. Moreover, we had two cases where an attacker was able to acquire `root` privileges. From there, he changed the password of any account he could find on the machine. As `root`, he possessed every privilege on the attacked machine, thus he did not even need to know the original passwords in order to change them. After that, the new passwords being quite sophisticated, they were not found during later dictionary attacks. This kind of specific behaviours will be described in section V-C2.

The analyses of our results show that the discovery of our accounts happened quickly (except in College Park and for account C4), as expected, since in one week all but one logins and passwords were found. However, the time before a first intrusion on these accounts ranges from one hour to more than two weeks.

2) Geographical distribution: The recorded connections came from 1207 distinct IPs, originating from 78 countries. Table V shows the 5 countries from which we have seen the highest number of distinct IPs on each honeypot.

Rank	Toulouse	Rennes	College Park	LAAS
1	China 83	China 80	China 40	China 98
2	USA 50	USA 56	USA 39	USA 60
3	Germany 18	South Korea 37	Brazil 12	Romania 34
4	France 18	Germany 14	Japan 7	Russia 16
5	Netherlands 18	UK 13	UK 7	Germany 14
Nb. distinct IPs	385	387	197	421

TABLE V
ORIGIN OF THE MOST OBSERVED ATTACKS ON EACH HONEYPOT

China and the United States are the two top ranked countries on the four honeypots. Concerning the rest of the rankings, at this scale, the differences are not important enough to allow us to conclude.

Moreover, by comparing the IPs recorded by these four honeypots with those recorded by the honeypot previously installed at LAAS [11] (3230 distinct addresses recorded between January 2006 and August 2010), only four of them were seen during both experiments, and only on the honeypots located in France. This emphasizes a first interesting conclusion: it seems that the IP addresses used for these attacks have a limited lifespan.

B. Dictionary attacks

1) Overview: We call dictionary attacks a session during which at least 9 `ssh` connection attempts happened. This allows us to rule out quite safely the cases where the observed attempts actually correspond to connection mistakes. We obtained a total of 1479 dictionary attacks. These originated from 825 distinct addresses, from 71 different countries.

2) Vocabularies:

a) Definition: We call vocabulary of a session the set of login/password pairs used during this session. Each pair forms a word of this vocabulary. We call dictionary a set of vocabularies possessing common characteristics.

b) Observations: We created a global dictionary for each honeypot, which is actually the union of all the vocabularies observed on this honeypot. Thus, we call $D1, D2, D3$ and $D4$ the global dictionaries from the honeypots respectively located in Toulouse, Rennes, College Park and at LAAS. We call $D1', D2', D3'$ and $D4'$ the dictionaries created from $D1$,

$D2$, $D3$ and $D4$ and containing the words appearing only in these dictionaries :

$$Di' = \{m \in Di / \forall j \neq i, m \notin Dj\}$$

These are called the exclusive parts of those dictionaries. Table VI shows the amount of words those dictionaries have in common.

Despite the big differences of size between the different dictionaries, we can make some observations. First, one can notice that a large enough basis of common pairs exists, even if it changes within each honeypot. Indeed, the exclusive parts of the dictionaries $D1$, $D2$, $D3$ and $D4$ consist in 57%, 62%, 14% and 44% of their respective sizes.

Dictionary	Number of words	Dictionary	Nomber of words
$D1$	127226	$D1 \cap D2 \cap D3$	3732
$D2$	142956	$D1 \cap D2 \cap D4$	29277
$D3$	5953	$D1 \cap D3 \cap D4$	3172
$D4$	76102	$D2 \cap D3 \cap D4$	3224
$D1 \cap D2$	46893	$D1 \cap D2 \cap D3 \cap D4$	3025
$D1 \cap D3$	4218	$D1'$	72598
$D1 \cap D4$	36673	$D2'$	89463
$D2 \cap D3$	4546	$D3'$	825
$D2 \cap D4$	35262	$D4'$	33348
$D3 \cap D4$	3467		

TABLE VI
INTERSECTIONS OF THE FOUR GLOBAL DICTIONARIES

In a similar way, let us call $D0$ the dictionary containing every pair tried during the experiment described in [11], and $D0'$, its exclusive part regarding $D1$, $D2$, $D3$ and $D4$. Table VII gives us an overview of the evolution of current vocabularies compared to those observed during the previous experiment. We note that about 40% of the content from the recently observed dictionaries (except for $D3$, for which a higher proportion, around 80%, was observed) is also in $D0$. Considering the largest dictionaries, it seems that the content of the vocabularies tends to evolve through time. Nevertheless, it is early to generalize this conclusion due to the relatively short duration of the second experiment.

Dictionary	Number of words
$D0$	253287
$D0 \cap D1$	56333 (44% $D1$)
$D0 \cap D2$	51219 (36% $D2$)
$D0 \cap D3$	4653 (78% $D3$)
$D0 \cap D4$	33675 (44% $D4$)
$D0 \cap D1 \cap D2 \cap D3 \cap D4$	2720
$D0'$	172708

TABLE VII
COMPARISONS BETWEEN OLD AND NEW DICTIONARIES

3) Geographical distribution: Table VIII counts the IP addresses having performed dictionary attacks on several machines. We note that only two addresses have attacked all four honeypots. However, there is an important number of addresses which attacked at least two honeypots in France whereas a very few IPs were seen both in France and in the United States. It should be noted that the IPs corresponding to the French honeypots are quite close to one another but farther from the College Park address. Moreover, the lists of login/password pairs tried by a single address on several of our honeypots are very similar, and sometimes identical. It seems that machines dedicated to these attacks are given a range of IPs to target instead of scanning the whole spectrum. They then keep trying the same sequence of pairs on each machine before proceeding to the next one, or even restarting at the beginning of the range, as we witnessed several times.

Set	Number of attacks	Distinct IPs
Toulouse	358	297
Rennes	529	308
College Park	197	93
LAAS	395	225
Toulouse ∩ Rennes	120	54
Toulouse ∩ College Park	21	12
Toulouse ∩ LAAS	108	55
Rennes ∩ College Park	12	7
Rennes ∩ LAAS	84	37
College Park ∩ LAAS	9	6
Toulouse ∩ Rennes ∩ College Park	8	2
Toulouse ∩ Rennes ∩ LAAS	71	24
Toulouse ∩ College Park ∩ LAAS	6	2

TABLE VIII
DISTRIBUTION OF THE IPS INVOLVED IN DICTIONARY ATTACKS

C. Intrusions

This section is dedicated to the analysis of the successful connections that included the execution of some commands on the honeypot : the intrusions.

1) Identifying the attackers: We noted that almost half of the 131 intrusions we recorded came from the same European country P1 (62 intrusions). These results confirm what had been noticed during the previous experiment [11]. However, some of these addresses may actually be relays used by an attacker to hide himself behind. Nevertheless, analysis of the information given by the terminals used during the attacks showed that those attackers often tried to download programs on websites hosted in the same country. Besides, among the programs that were actually executed, several diplayed some text in the language from P1.

Moreover, we have seen in section V-A1a that some IP addresses were seen on several honeypots. The results of the analyses of the overlapping IPs presented in table IX show us that intrusions on different locations are rarely carried out from the same addresses. Indeed, only five IPs involved in intrusions were seen on several honeypots, more specifically on the French ones. Looking more in detail at what happened during these intrusions, we noted that for four of them, that targeted both the honeypots deployed in Toulouse and in Rennes, the altered password used during the account takeover (cf V-C2a) was always the same (the fifth address did not change the password of the attacked account). We can conclude from this observation that these four addresses therefore belong to the same individual or group of individuals. Furthermore, the number of distinct addresses is quite close to the number of detected intrusions but much higher than the number of attacked accounts, which means that most of the attackers do not connect twice with the same address on an account. In the case of accounts having been visited by several addresses, two

hypothesis (that do not exclude each other) can be considered :

- The attacker changed his address on each connection to hide his activities.
- There is actually a community of attackers sharing information about the machines they attacked, which for example allows them to relay if one of them cannot take over a machine due to a lack of technical skills.

Set	Number of intrusions	Distinct IPs	Nb. of attacked accounts
Toulouse	22	13	4 + root
Rennes	26	21	11
College Park	7	7	4
LAAS	76	51	11 + root
Toulouse ∩ Rennes	11	4	2 + 2
Rennes ∩ LAAS	3	1	1 + 1

TABLE IX
DISTRIBUTION OF THE IP ADDRESSES INVOLVED IN INTRUSIONS

2) Attackers' activities:

a) General trends: During the 131 observed intrusions, several behaviours common to a large number of attackers were frequently noted:

- Concern for discretion: the attacker checks if he is currently alone on the machine and often deletes the history files.
- Exploring the machine: the attacker tries to obtain information about the attacked machine : name and version of the OS, processor characteristics, etc.
- Account takeover: during the first connection to an account, the attacker always changes his password for a more sophisticated one in order to get control over the account, but takes the risk of being detected in case the legitimate user wants to use his machine.
- IP scan: the attacker installs an IP range scanning program in order to find which remote hosts can be accessed through `ssh` from the attacked computer.
- `IRC` client setup: this messenging protocol client is used to receive and execute instructions sent by an attacker from a remote server. The goal here seems to connect the infected machine to a *botnet*, the server thus sending its instructions to hundreds of compromised machines at the same time.
- Attempts to acquire administrator privileges: some attackers try to obtain administrator privileges in order to have a complete control over the compromised machine. To do so, they try to exploit security vulnerabilities thanks to various dedicated programs like `rootkits`.

b) Analysis of the "root" intrusions: We observed that two attackers successfully gained root access to the honeypot. We present in this section a description of the operations they ran once they obtained the root access.

Once the administrator privileges obtained, both the attackers immediately changed the `root` pasword. Then they installed customized software in order to get information about the "legitimate" users of the computer and also to open a new port so that they would be able to communicate with the machine even if they happened to lose the access through port 22 (a so called *backdoor*).

The first attacker thus installed the `rootkit` SHV4[6]. This software installs an `ssh` server if there is not one already, alters the `ssh` client executable to record the logins and passwords tried during connections to other hosts and installs several altered system executables which would normally permit to detect its presence, allowing it to stay undetected. This attacker likely wanted to easily find targets for his future attacks. However, he also changed the passwords of every account he had found on the machine thanks to his previous dictionary attack, but also by listing the folders located in the `/home` directory (but did not read the list of existing accounts, located for example in `/etc/passwd`). It seems odd that the attacker wanted to cover his activity and to collect the passwords typed by the legitimate users while also preventing the greatest possible amount of users from accessing their accounts.

The second attacker also replaced the `ssh` client binary by another version but we could not identify the changes actually implemented in the modified version. He however did not change the passwords of other existing accounts but created a new one, named "backup" and possessing administrator privileges. He then accessed again this account but did not alter anything else yet. However, he kept using the "user" account to install new `IRC` clients, probably because he could not establish a link with those he previously installed due to our network security policies.

VI. CONCLUSION

We have presented in this paper a distributed plateform for deploying high interaction honeypots as well as our first results regarding the analysis of the collected data.

These first results confirm, for most of them, the conclusions we had drawn from a previous experiment carried out with only one honeypot. These confirmed conclusions are: 1) the specialisation of the IP addresses used by the attackers, either used for dictionary attacks, or used for intrusions, but never for both activities; 2) the main activities and goals of the attackers when they interact with the honeypots and 3) the main country which is at the origin of the intrusions. We have also drawn new conclusions from this experiment: 1) the IP addresses that are used by the intruders are almost all renewed (only four IP addresses have been "seen" both during this experiment and during this previous experiment we carried out) and 2) the dictionaries used for the dictionary attacks seem to evolve, from the previous experiment to this new experiment. Of course, these analyses are still preliminary and need to be confirmed by other sets of data that we go on to collect.

REFERENCES

[1] B. McCarty, "The honeynet arms race," *IEEE Security and Privacy*, vol. 1, pp. 79–82, 2003.

[6]http://web.fhnw.ch/plattformen/ns/vorlesungsunterlagen-1/network-analysis-tools/shv4-analysis

[2] B. Cheswick, "An evening with berferd in which a cracker is lured, endured, and studied," in *Proceedings of the Winter 1992 USENIX Conference*, 1992, pp. 163–174.

[3] L. Spitzner, *Honeypots: Tracking Hackers*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002.

[4] F. Maggi and S. Zanero, "Analysis of the state of the art," *WOMBAT Project Worldwide Observatory of Malicious Behaviors and Attack Threats*, 2008, http://wombat-project.eu/workpackages/wp2-analysis-of-state-of-the-a/.

[5] NOAH, "Do1.1: Survey on the state of the art," *Deliverable of the European Network of Affined Honeypots*, 2005, http://www.fp6-noah.org/publications/deliverables/D0.1.pdf.

[6] E. Balas, "Know your enemy : Sebek," *The Honeynet Project*, 2003, www.honeynet.org/papers/.

[7] N. Provos, "A virtual honeypot framework," *Proceedings of the 13th conference on USENIX Security Symposium*, 2004.

[8] C. Leita, K. Mermoud, and M. Dacier, "Scriptgen: an automated script generation tool for honeyd," in *Proceedings of the 21st Annual Computer Security Applications Conference*. IEEE, 2005.

[9] G. Wagener, R. State, A. Dulaunoy, and T. Engel, "Self adaptive high interaction honeypots driven by game theory," in *SSS*, ser. Lecture Notes in Computer Science, R. Guerraoui and F. Petit, Eds., vol. 5873. Springer, 2009, pp. 741–755.

[10] P. Baecher, M. Koetter, M. Dornseif, and F. Freiling, "The nepenthes platform: An efficient approach to collect malware," in *In Proceedings of the 9 th International Symposium on Recent Advances in Intrusion Detection (RAID)*. Springer, 2006, pp. 165–184.

[11] V. Nicomette, M. Kâaniche, E. Alata, and M. Herrb, "Set-up and Deployment of a High Iinterfaction Hhoneypot: Experiment and Lessons Learned," *Journal in Computer Virology*, vol. 7, no. 2, pp. 143–157, Mai 2011.

[12] I. Studnia, E. Alata, M. Kâaniche, and V. Nicomette, "Observation et Aanalyse d'Attaques sur Internet," LAAS CNRS, Tech. Rep. RL 1160, 2011.