2011 17th IEEE Pacific Rim International Symposium on Dependable Computing

Characterizing Attackers and Attacks: An Empirical Study

Gabriel Salles-Loustau[1, 3], Robin Berthier[2], Etienne Collange[1, 4], Bertrand Sobesto[1], and Michel Cukier[1]

[1] Clark School of Engineering
University of Maryland
College Park, MD, USA
{gsallesl, ecollang, bsobesto, mcukier}@umd.edu

[2] Coordinated Science Laboratory
Information Trust Institute
University of Illinois
Urbana-Champaign, IL, USA
rgb@illinois.edu

[3] ENSI Bourges
Bourges, France
gabriel.sallesloustau@ensi-bourges.fr

[4] ENSEEIHT
Toulouse, France
etienne.collange@etu.enseeiht.fr

***Abstract*—This paper describes an empirical research study to characterize attackers and attacks against targets of opportunity. A honeynet infrastructure was built and deployed over 167 days that leveraged three different honeypot configurations and a SSH-based authentication proxy to attract and follow attackers over several weeks. A total of 211 attack sessions were recorded and evidence was collected at each stage of the attack sequence: from discovery to intrusion and exploitation of rogue software. This study makes two important contributions: 1) we introduce a new approach to measure attacker skills; and 2) we leverage keystroke profile analysis to differentiate attackers beyond their IP address of origin.**

Keywords- Empirical Research Study, Attacker Behavior, Attack Analysis, Honeypots.

I. INTRODUCTION

Security solutions should be deployed according to the threat facing the particular organization's network. However, few empirical research studies exist to quantify such threat. In this paper, we characterize attackers and attacks against targets of opportunity since our organization is a large public university. Moreover, we will restrict the discussion to attacks against UNIX computers where attackers gain access through SSH.

It is often difficult to find who is launching attacks. We divided the attack process between the initial brute force attack against SSH accounts and the intrusion step when the attacker logs in using the cracked username/password. For brute force attacks, we can only rely on the attacker's IP address. However, once the attacker logs in, we can also use the keystrokes and the attack type to try to characterize the attack further. In this paper, we analyze whether the attacker who launches the brute force attack is the same attacker than the one who logs in and whether a single attacker logs in or multiple attackers share the same compromised machine.

To expand our knowledge of attacker profiles, we assess the attackers' skills by introducing a list of ten criteria. We discuss the link between the attacker's skill and particular actions including hiding his/her malicious activities, changing the password or checking for the presence of other users. We also discuss the link between the attacker's skill and the attacker's capacity to successfully launch attacks from the compromised computers under different restrictions. Finally, we attempt to understand why the attack was launched by categorizing attacks in different classes according to the rogue software installed and exploited.

The empirical research study is based on data collected over five and a half months. The set of target computers consisted of three configurations that enforced increasing network limitations including a first fully functional environment, a second configuration where only the IRC port is blocked, and a third configuration where only a few services were allowed (HTTP, HTTPS, FTP, DNS, SSH). Attackers could compromise these three honeypot configurations incrementally during one month (the honeypots were backed up and redeployed at the beginning of each month), starting with the first configuration being deployed on 40 public IP addresses.

This paper is structured as follows. Section II discusses approaches for identifying the attacker and the attack. Section III describes the experimental setup including the target computers configurations and the data collection tools. Section IV presents and discusses the results. Section V reviews the related work. Finally, Section VI concludes the paper and proposes follow-up studies.

II. IDENTIFICATION OF ATTACKER AND ATTACK

In this section, we describe the methods developed to identify the attacker and the attack. The definition of these methods is important because it specifies the requirements used to design the data collection architecture.

A. Attacker Identification

A critical step to understand the threat landscape is to correctly identify attackers. Most of the time, an assumption is made by mapping each IP address to a single attacker. However, it is more realistic to assume that attackers can use multiple IP addresses to better hide their traces or because of

978-0-7695-4590-5/11 $26.00 © 2011 IEEE
DOI 10.1109/PRDC.2011.29

network constraints such as DHCP. In order to go beyond this assumption, we defined multiple indicators to build attacker profiles that we then use to try to uniquely identify human individuals behind each attack. In addition to the IP address, these indicators are, first, attacker AS routing number and attacker's geographic location: to potentially detect if an attacker comes from a single ISP that has changed the client IP address over the time window of our experiment. Second, attacker specific actions, in which three subcategories are considered:

- Rogue software origin: a same attacker often downloads his/her tool from a same location while compromising a target.
- Techniques to perform specific actions: for example, attackers who try to erase their traces can employ multiple commands. However, they usually use the same method from one attack session to another.
- Files accessed: many attackers install rogue software in hidden or complex locations. If two sessions access this kind of file, it is usually a hint about the attacker's identity.

Third, the comparison of keystrokes profiles that we describe further in the remainder of this section.

The goal of keystroke analysis is to help differentiating attackers. According to [6], delays due to the network for SSH connections in comparison to delays between human keystrokes can be neglected. Our analysis of keystrokes is based on this assumption. Extensive research exists on keystroke profiling [7, 9, 10]. We considered in particular the study presented in [9] because it introduces an approach adapted to our goals, which consist in user authentication and user recognition by analyzing data collected from various users typing on their workstations. The approach consists in grouping the recorded text following the most used syllables in English. Then, the graph of keystrokes latencies (i.e., time between successive keystrokes) and durations (i.e., length of time keys are pressed) as function of the syllables reveals the user's keystroke profile. In our case, we recorded session keystrokes latencies from attackers commands that are not copied and pasted text and we compared the delays of each matching pair of keystrokes between all the attack sessions recorded on a honeypot.

Our experiment differs from [9] because contrary to their experiment where the authors evaluated the efficiency of recognizing a user through his/her keystrokes, we don't know in advance from which user the session keystrokes originates. Therefore, keystrokes profiles are considered in our experiment as an indicator among other parameters (IP address, AS number, attacker's techniques to perform specific actions, files accessed and rogue software downloaded) to differentiate attackers. Moreover, contrary to their experiment, the quantity of keystrokes used to perform the comparison directly depends on the number of commands the attacker typed. In certain cases the recognition through keystrokes is not possible due to the lack of recorded keystrokes for a session.

B. *Attacker Skill*

To assess the attacker's skill over the entire recorded attack sequence, we discussed two approaches. The first one consists in asking an analyst to review each attacker session and to attribute a score. The second one introduces several criteria that correspond to various actions performed by attackers. We postulate that a variety of criteria fulfilled by an attacker indicates a higher skill level. As a result, we compute the skill level as a function of the criteria fulfilled. While the first approach is subjective, the second may be limited due to the definition of the criteria. In this paper, we opted for the second approach because it enables others to easily replicate our experiment.

We developed ten criteria based on four generic questions to evaluate the competences of the attacker. These questions are:

1) *Is the attacker careful about not being seen?* The fact that an attacker does not want to be noticed indicates that the attacker knows that such behavior increases the chances to maintain access to the target. It also indicates that the attacker knows how to reduce his/her traces. We identified four approaches in the criteria: erasing the files that attackers imported on the target from the Internet (Criterion 4, Table I), deleting the logs that contain traces of the attacker's activity (Criterion 1), restoring the logs not to catch attention of a user who could notice that the logs are missing (Criterion 2) and checking the presence of other users during the attack (Criterion 3).

2) *Does the attacker pay attention to the environment of the target before launching an attack?* Learning about the target and its environment (especially the network environment) is often important to carry the attack successfully. Therefore, we created Criterion 5, whether the attacker checks the environment. In addition, checking the presence of other users (Criterion 3), which is one of the criteria used for to evaluate the attacker's discretion, also contributes to answering this question.

3) *How familiar is the attacker with the rogue software he/she is using?* Some attackers attempted to use specific rogue software unsuccessfully. For example, a rogue software having network functionalities that is installed on a target where the corresponding network port is blocked (Criterion 10) indicates a poor knowledge of the software or a lack of expertise to detect blocked ports. On the contrary, editing the configuration file(s) (Criterion 6) before the installation shows that the attacker is aware of what he/she is doing and is not just reproducing an attack without understanding it. We can also link to the question the fact that the attacker changed the system to make the rogue software work (Criterion 7).

4) *Is the attacker protecting the compromised target?* Even once a target is compromised by an attacker, brute force attacks continue to be launched against this target. Since the attacker gained access through a brute force attack, it means that the attacker account credentials (login/password) are somehow weak. Therefore, we introduced two criteria to take in account if the attacker was

protecting his/her account, changing the password and creating a new user account (Criteria 8 and 9).

These ten criteria are evaluated with a value between 0 and 1 and summed over the entire period of the experiment, leading to an overall attacker' skill level between 0 and 10 for the global attacker's activity. We are able to identify all the sessions from a specific attacker through the profile analysis described previously (Section II-A). We now explain in more detail how we evaluated them.

Criteria 1, 2 and 3 are evaluated using ratios between numbers of sessions involving a given attacker. More precisely, we take into account whether these actions have been performed during each session or not, because the corresponding action(s) are more efficient if they are performed every time the attacker connects to the target. For example, to be the most efficiently hidden, an attacker needs to hide at each session, or if the attacker wants to verify who else is using the target, the attacker has to check the presence of other users at every session. The score for these criteria is the ratio of the number sessions where the attacker fulfilled the criteria over the total number of sessions, leading to any number between 0 and 1.

Most other criteria (4, 5, 6, 7, 8, and 9) are considered for the global activity of the attacker. The criterion is evaluated as either 0 or 1, depending if the attacker did the corresponding action(s) at least once among all sessions. Indeed, these actions do not need to be performed in each session to give an indication of the attacker's skill level.

The remaining criterion is the rogue software adequacy (Criterion 10). The idea is to assess whether the rogue software requirements match the target configuration. For each rogue software, we know its type from the rogue software analysis (see Section II-C) and the honeypot on which it has been run. Depending on the honeypot configuration, the rogue software is noted as adequate or not. We attribute 1 if more than half of the attacker's rogue software is adequate and 0 otherwise. This choice does not penalize too much an attacker who installed an inadequate software, realizes it, and installs another more adapted one. And gives 0 to an attacker that insisted in installing several inadequate rogue software.

Table I lists the different criteria with their definitions and the method used to evaluate them.

C. Attack Identification

To identify the attack tools installed by attackers, we developed an application in Java that automates a part of the analysis and stores the results in a database. This application performs two tasks, the static analysis and the dynamic analysis of a given attack tool.

Information obtained by static analysis includes: the name of the file that was downloaded on the honeypot (*filename*), the URL address from where it was downloaded (*url_origin*), the IP address corresponding to the URL (*ip_origin*), the time of the download (*download_time*), the session in which the software was downloaded (*session_id*), the file type (*file_info*), the file size (*filesize*), the number of files after unpacking if it is an archive (*nb_files*), the *roguesoft_id* of other rogue software already in the database that are similar (*matching_files*) and finally the files affected by the rogue software during execution and that were identified to be likely used as configuration files (*configuration_files*).

Information obtained by dynamic analysis includes: the ports that are being opened during an execution of the rogue software (*open_ports*), the files that are being accessed (*files_open*), the log of the network traffic generated (*log_iptables*), and the processes created (*new_processes*).

This dynamic analysis consists in replaying attacker sessions involving the rogue software in a sandbox identical to the honeypot environment. At this point human intervention is required to reproduce the sequence of commands entered by the attacker after he/she downloaded the rogue software. A full automation of the execution would have been possible if attackers were not making any mistakes while typing commands and if keystrokes were perfectly recorded by the keylog solution we used (see Section III-B). However, having to execute rogue software manually helps identifying them. Once we have reproduced the attack, the modifications to the sandbox resulting from the rogue software execution are saved and analyzed.

The information gathered statically and dynamically is reviewed to identify the type of rogue software. Ports opened during the execution give information on the protocol linked to the rogue software acting as a server. If no port has been opened, then this type of information can be retrieved from the network traffic traces in case the rogue software was used

TABLE I. ATTACKER SKILLS CRITERIA

Criterion ID	Criteria Name	Definition	Asssessment
1	Hide	Deletion of log files or deactivation of logging	Ratio of the number of sessions where the attacker hid
2	Restore Deleted Files	Restoration of deleted files	Ratio of the number of sessions where deleted files were restored
3	Check Presence	Observation of users	Ratio of the number of sessions where presence has been checked
4	Delete Downloaded File	Deletion of downloaded rogue software after usage	0 if downloaded file is not deleted in any session, 1 otherwise
5	Check System	Observation of system configuration or state	0 if the system has never been checked in any session, 1 otherwise
6	Edit Configuration File	Edition of rogue software configuration file	0 if configuration file is not edited in any session, 1 otherwise
7	Change System	Modification of system configuration or state to have a working attack	0 if the system has never been modified in any session, 1 otherwise
8	Change Password	Change user password	0 if the password was never changed in any session, 1 otherwise
9	Create New User	Creation of new user	0 if no new user is created in any session, 1 otherwise
10	Rogue Software Adequacy	Adequacy of rogue software	0 if less than half of the installed rogue software is adequate, 1 otherwise

as a client. The files accessed during execution are usually a relevant clue to identify a specific type of rogue software. Similarly, a particular process called during execution can reveal the type of malware.

It can happen that rogue software belongs to several categories (e.g., an archive containing several pieces of malware). In those cases, by replaying attackers actions, we only consider the observed usage(s) of the rogue software to determine the category(ies) it belongs to.

Different versions of a rogue software are indentified as distinct rogue software through the static analysis. However, they will belong to a same category if the dynamic analysis reveals that the attacker purpose while using it was the same.

Same rogue software used with different configurations are also distinguished. The matching_files field in the static analysis helps identify identical or similar files used by attackers.

The information gathered during this analysis process about attackers' tools and their behavior are stored in a database. This database helps to understand the scope of attacks. We note that unlike traditional malware analysis services like VirusTotal [15], our approach is not exclusively based on a comparison with a repository of already known dangerous files. Running both static and dynamic analysis often reveals the purpose of the attack tools instead of only giving a general idea about the dangerousness of the file.

III. EXPERIMENTAL SETUP

In this section we describe the configuration of the deployed honeypots and the tools used to collect attacks data.

The approach introduced in Section II relies on a specific data collection based on the following requirements: 1) to provide access to the honeypots through a frequently scanned and vulnerable entry point: a modified SSH service in our case; 2) to provide multiple honeypots for every attacker and for an extended period of time; 3) to configure the honeypots given to one attacker with increasing network limitations to build an increasing scale of difficulties for the attackers; and 4) to collect relevant data to characterize the attacker, such as network activity, keystrokes typed and rogue software downloaded.

A. Honeypots

These features are implemented through the use of two systems shown in Fig. 1: a network gateway in front of the Internet accepting connections from a range of public IP addresses on the SSH port, and a set of virtual honeypots that are OpenVZ containers running on a single host. These containers are configured with private IP addresses.

The network gateway has two network interfaces: one is connected to the Internet and configured with 40 public IP addresses from the University of Maryland, the other interface is configured with a private IP address. The operating system used is a Linux Ubuntu 9.10 server installed with modified versions of OpenSSH server, PAM-MySQL, NSS-MySQL and a specific shell provided to attackers when their brute force attack succeeds.

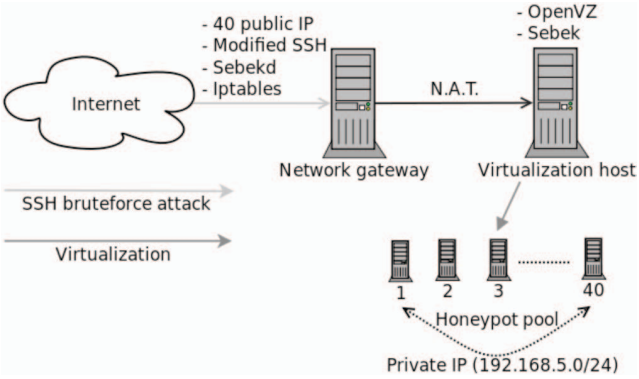

Figure 1. Testbed Architecture

OpenSSH was modified to reject SSH login attempts on its public IP addresses until a predefined number of attempts. When this predefined threshold is reached, OpenSSH skips password verification, creates a new user with the latest credentials attempted by the attacker and calls PAM, which is configured to use the modified PAM-MySQL module for user verification. The predefined threshold is randomly selected between 150 and 200. Moreover, to limit the number of deployed honeypots to three per attacker IP address, the modified version of SSH rejects any attempt from an IP address which has already deployed three honeypots.

PAM-MySQL initial purpose is to verify user credentials using MySQL as backend. In the modified version, PAM-MySQL creates user accounts for attackers and records on a MySQL database user information including: login, password, source IP address, targeted IP address on the gateway, a user ID and the path of the specific shell on the gateway which handles honeypot deployment.

NSS-MySQL is required to read users information from the MySQL database. This component is required by OpenSSH to access user account information before granting access to a session. We modified the module to handle conflicting cases of user entries with a same user login. The distinction between these entries is based on the targeted IP address of the SSH connection.

The specific shell provided to attackers on the gateway is a Bash script. It reads user information on the MySQL database, configures a new OpenVZ container on the virtualization host with users credentials and configures Network Address Translation (NAT) on the gateway between the public IP address targeted and the container private IP address. Network rules are applied on the network gateway for the deployed honeypot. Attackers, who have been identified according to their IP addresses, can deploy up to three honeypots that are configured as follows. The first honeypot (HP1) has no network limitation, the second one (HP2) has the main IRC port blocked (port 6667, incoming and outgoing traffic), the third one (HP3) has every port blocked except HTTP, HTTPS, FTP, DNS, and SSH.

This process is transparent from the attacker point of view. For the first attack session, when a given number of attempts has been reached, the attacker triggers the launch of the specific shell on the gateway that tunnels the connection

to a newly configured container on the virtualization host. For the following connection, a NAT is performed on the gateway to redirect all incoming traffic to the right container.

We used OpenVZ containers to provide a UNIX environment to attackers. OpenVZ was deployed on a host configured with a CentOS 5.4 system and the stable release of OpenVZ kernel 2.6.18-164.15.1.el5.028stab068.9. These containers are configured with two servers: OpenSSH and Apache.

The choice of OpenVZ as honeypot has three advantages. First, OpenVZ is a lightweight solution, which enables up to 40 containers to be run in parallel on a single host. Being able to run this number of honeypots in parallel is a priority in this experiment since we provide multiple honeypots per attacker. Second, we can administrate the containers from the host system at the honeypot deployment (i.e., add users and set network configurations). Both steps are realized through the use of the `vzctl` command. Moreover, OpenVZ virtual network device, *venet,* which implements a network device for the containers gives the assurance that attackers will not be able to change the IP address affected to the honeypot. Third, OpenVZ restricts attacker actions to the container they interact with. By design, attackers cannot escape from their container even if they have root privileges nor can they interact with the host's kernel. These restrictions ensure that attackers cannot find out the presence of a kernel based keylogger. However, attackers can find out that their environment is virtualized.

To observe attackers over an extended time period, we opted for a honeypot lifecycle of one month. This choice is discussed in Section IV-D. At the beginning of each month, honeypots are stopped, backed up and new honeypots are deployed.

This experimental setup provides two advantages: first, multiple honeypots can be deployed per attacker and these honeypots can run in parallel. Attackers can return to their environment at any time during one month. Second, it offers different environments to attackers. In this paper we focused on the influence of network restrictions on attacker's behavior. The choice of blocking the IRC port in HP2 was made in regard of the popularity of rogue software using IRC.

B. Data

We collected two main datasets: network traffic from the public IP address of the gateway, and attacker's keystrokes through the use of the Sebek keylogger from the Honeynet Project [17] on the virtualization host.

Sebek is a kernel based data capture tool that captures keystrokes by catching `read` syscalls. Note that to collect keystrokes from attackers and to determine from which container they come from, we modified Sebek to add the OpenVZ identification number (VEID) of the container of origin to Sebek packet headers.

The following information was extracted from the data:

- Network flows (especially SSH flows) that allow to map attackers IP addresses to keystrokes recorded from sessions,
- Rogue software installed using the Honeynet Project's Honeysnap tool [16] and Wireshark [18],
- Keystrokes processed into lists of commands for a study of attackers actions.

Two additional analyses were performed on keystrokes collected from Sebek. The first one aims at detecting copied and pasted text in attackers sessions to show how much the attacks are automated. The second one is an approach to evaluate how many different users exploit a same honeypot (see Section II-A).

Keystrokes recorded by Sebek are encapsulated in UDP packets and are sent to the network gateway. The packets include timestamps with a microsecond precision for the system calls. Attackers generate one packet per keystroke while typing commands through an interactive SSH connection, except for copied and pasted text where multiple keystrokes are sent in a same packet. This enabled us to collect individual keystroke times for interactive commands and also to determine whether the attacker copied and pasted commands in his/her shell. For such determination, we calculated the average delay between keystrokes for each command typed. A copied and pasted command is generally performed in two times: 1) the user copies and pastes the command in his/her shell, 2) the user hits the return key to launch the action. We removed the last keystroke interval between the last keystroke of the command and the return because of a possible excessive delay that could give the wrong results. To detect these commands containing copied and pasted text, we use a threshold of 100 millisecs of average delay interval between keystrokes. This threshold corresponds to the maximum speed of a skilled typist who can type up to 120 words per minute (i.e., 10 characters per second). We assumed that a human being could not go beyond such typing speed.

We note that attackers usually mix interactive commands and copied and pasted text, especially when they use long paths or URLs. The following sample attack session illustrates this behavior. Highlighted text indicates commands containing copied and pasted text and average time interval recorded are shown on the left-hand side.

```
[28 µs]   cat /usr/share/man/man1/.error
[>100 ms] cat /proc/cpuinfo
[>100 ms] history -c
[>100 ms] w
[>100 ms] ps -x
[>100 ms] cat /proc/cpuinfo
[42 µs]   wget http://download.microsoft.com/…
[>100 ms] ls
[>100 ms] rm -rf W2Ksp3.exe
[>100 ms] cd /usr/sbin
[28 µs]   wget http://sbebe.110mb.com/img
```

IV. Discussion of Results

This section first presents overall results, then focuses on the attacks characteristics. As previously mentioned, three types of honeypots were deployed: HP1 is fully functional, HP2 has the IRC port blocked for incoming and outgoing traffic, and HP3 has all ports blocked besides HTTP, HTTPS, FTP, DNS, and SSH. Moreover, the network is configured so that only attackers who gained access to HP1

can reach HP2, and once access is granted to HP2 attackers can reach HP3. The three honeypot configurations can be compromised through a similarly configured SSH server.

A. Overall Results

The experiment was conducted from May 17th 2010 to October 31st 2010. At the beginning of each month, the honeypots were reverted. During the experiment, 106 honeypots were successfully deployed, 56 HP1, 30 HP2 and 20 HP3. Table II shows the distribution of sessions per honeypot type and the number of computer compromise sessions (non-empty sessions).

TABLE II. DISTRIBUTION OF SESSIONS PER HONEYPOT TYPE

	Number of sessions	Number of non empty sessions
All honeypots	312	211 (68%)
HP1	160	110 (69%)
HP2	105	74 (70%)
HP3	47	27 (57%)

B. Attack Origin

We first considered the origin of the attacks by analysing IP addresses related information: the IP address itself, the AS number related to this IP and the country of origin.

It has been shown that attackers can be divided in different groups: those who conduct reconnaissance by brute-force scanning hosts, and those who attempt to actually gain control of hosts by compromising them [1]. We verified this statement by comparing the origins of both groups in the case of the SSH experiment. Fig. 2 shows the percentage of attackers who performed brute-force scanning but who did not compromise the computer (i.e., even if they found the login/password combination, they didn't use a shell), the percentage who did both, and the percentage who only compromised the target (i.e., they knew the login/password combination and got a shell at the first login attempt). Fig. 2.a shows the results when we assume that a single IP address is associated with a single attacker. Fig. 2.b shows the results when we assume that the attacker could have used another IP address that belongs to the same organization, identified by its AS number. Both figures confirm previous results because the majority of attackers are clearly divided into two groups and only a small fraction of attackers are associated with both brute-force scanning and compromising hosts.

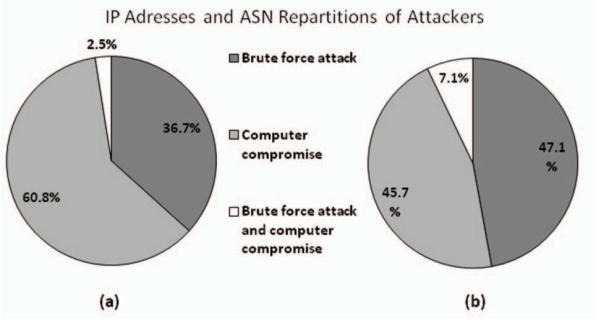

Figure 2. IP Addresses (a) and ASN (b) Distributions of Attackers

We also note that these results should be taken with caution because a given attacker could easily appear to connect from different IP addresses, different AS numbers and different countries.

C. Attack Characteristics

In this section we introduce the results about attackers differentiations and attackers skills. Once the attacker has compromised the honeypot, we used keystrokes profiles and the attackers keylogs to characterize the attack.

1) Identifying Attackers Through Keystroke Profile

Once the attacker compromised the target, the interactive shell sessions provide keystrokes to build profiles and keylogs. We applied a new approach of keystrokes analysis, discussed in Section II-A, to differentiate attackers.

Fig. 3 and Table III introduce an example of keystrokes analysis for three sessions of a given honeypot. According to the IP addresses, the attacks come from the USA (session 1), Israel (session 2) and Romania (session 3). Fig. 3 shows the delays between pairs of keystrokes as a function of the characters composing the pair. Values in Table III indicate the average delays interval between matching pairs of keystrokes for the three sessions. The smaller the value is, the closer the profiles are. In light grey we observe that profiles 1 and 2 are closer than profiles 1 and 3 or 2 and 3 are. Thus, the attacker from session 3 is likely to be different from the attacker in session 1 and 2.

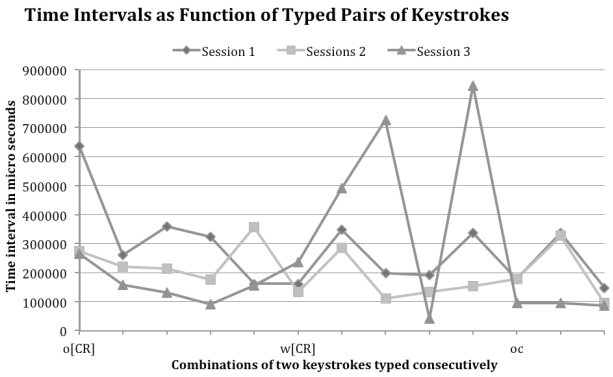

Figure 3. Time Intervals between Keystrokes for a Sample of Matching Pairs between Session 1, 2, and 3

In this case, further manual analysis of the attackers' keylogs sessions allowed distinguishing two different attackers: in session 1 and 2 the attacker used identical rogue software and executed the command "unset HISTFILE" at the beginning of both sessions whereas no similar actions were performed in session 3.

This method allowed to significantly improve the estimation of the number of attackers for each honeypot. Starting from 73 distinct IP addresses exploiting the honeypots over the five and a half months, we concluded that these were actually 39 different attackers. We note that we favored expert judgment over a traditional clustering technique to analyze those results, because the lack of

ground truth prevented us from identifying correct thresholds to build clusters from the different characteristics reviewed.

TABLE III. TABLE OF AVERAGE DELAYS INTERVAL IN MILLI SECONDS BETWEEN MATCHING PAIRS OF KEYSTROKES FOR THREE DIFFERENT SESSIONS

Session ID	1 (USA)	2 (Israel)	3 (Romania)
1 (USA)	0	57.7 (121 matching pairs of keystrokes)	197.5 (15 matching pairs of keystrokes)
2 (Israel)	X	0	163.8 (20 matching pairs of keystrokes)
3 (Romania)	X	X	0

2) Characteristics of Attack based on Attacker Skills

Based on the skill ranking technique introduced in Section II-B, Fig. 4 shows the distribution of attackers skill levels from 0 (no skill criterion observed) to 10 (all skill criteria observed) for the different honeypots. Fig. 4 seems to indicate that the configuration of the target is not linked to the attackers compromising it. This leads to the conclusion that deploying targets with very different configurations does not ensure to observe different types of attackers.

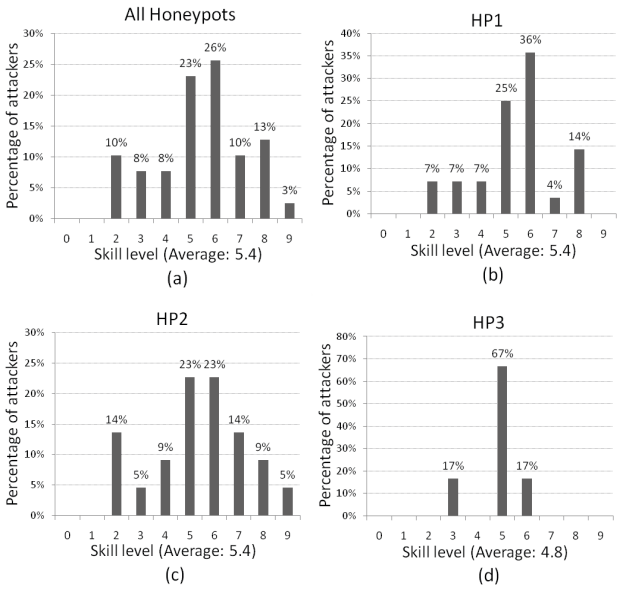

Figure 4. Distribution of attackers by skill levels for all honeypots (a), for HP1 (b), HP2 (c) and HP3 (d)

Having defined attackers' skills using ten criteria, we show the percentage of attackers who conducted each action in Fig. 5. We see that more than 90% of the attackers checked the system and the presence of other users on the target. Almost 80% of the attackers changed the account password. Less than 60% hid their actions. This number is surprising as it would have been expected that one of the attackers' main goal is to remain undetected. Less than 60% of the attackers did install correctly rogue software. Note also that new users are added only for about 15% of the attackers.

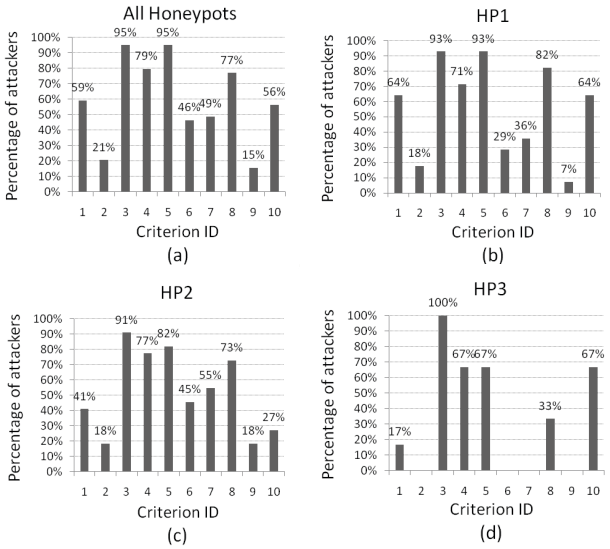

Figure 5. Percentages of Attackers Fulfilling Attacker Skill Criteria. For All honeypots (a), HP1 (b), HP2(c) and HP3(d)

Most of the attackers used copied and pasted commands in their session. To further expand the analysis, Fig. 6 represents the percentage of copied/pasted commands by skill level. One can observe that the most advanced attackers are more likely to paste directly command lines. One could expect that copied and pasted commands comes from less skilled attackers (a.k.a. script kiddies). However, to limit their visibility, it makes sense that skilled attackers prepare their attacks and limit the amount of time on the target launching the attack. However, another interpretation of this result could be that some attackers are simply using copied and pasted commands to execute a sophisticated attack without understanding it.

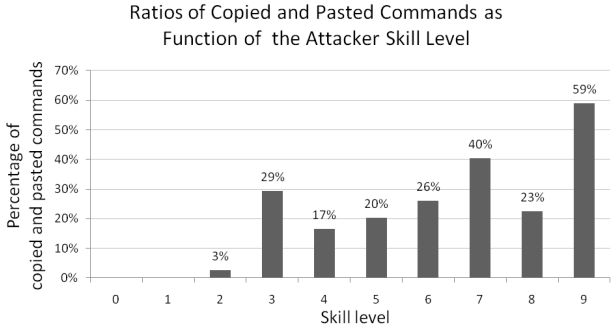

Figure 6. Percentages of Copied/Pasted Commands per Skill Level

To identify what actions are the most representative of the attacker's skill level, we calculate the skill distributions for groups of attackers who were observed performing specific actions. Figures 7 show the distributions for creating a new user (Fig. 7.a), hiding his/her actions (Fig. 7.b), changing his/her password (Fig. 7.c), correctly launching

rogue software (Fig. 7.d) and finally checking the presence of other users (Fig. 7.e).

On Fig. 7.a, we clearly notice that the population of attackers that created a new user has a higher average skill level (7.7) than the entire attacker population average (5.4) that can be seen on Fig. 4.a. This observation increases our confidence in selecting "Create new user" as a criterion for the skill level. As shown on Figures 7.b, 7.c and 7.d, similar cases can be observed to a lesser extent for a majority of the other criteria. However, Fig. 7.e shows that the average skill level of attackers that checked presence (5.5) is very close to the global average. This, as well as the fact that 95% of the attackers performed this action, shows that this action is performed by any type of attacker.

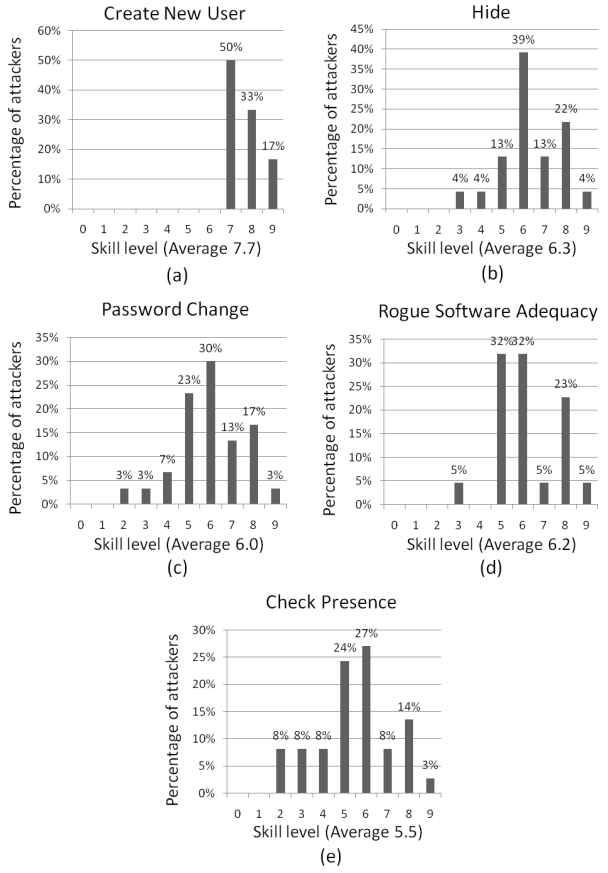

Figure 7. Distributions of Attackers by skill level for Attackers for Specific Attacker Actions

D. Attack Purpose

This section introduces the results about the second question that motivated this experiment, the attack purpose.

Table IV presents the list of categories of rogue software resulting from the analysis described in Section II-C. This analysis allowed concluding on the type of software for 90% of the rogue software collected. The remaining 10% could not be identified because they could not be correctly extracted from the network traces, either because of a prematurely stopped download or because of a broken network trace.

We observe that this list contains many categories that were expected based on the results found in [3], including IRC bots and IRC bouncers. In addition, we found some unexpected software, including a server for audio streaming.

TABLE IV. TYPE OF FILE OR ROGUE SOFTWARE USED BY ATTACKERS

IRCbot	Software used to enroll a compromised host in a botnet that uses the IRC protocol to communicate
Bouncer IRC	IP address spoofing software for the IRC protocol
Backdoor	Software allowing the attacker to come back on the host by another mean
Scanner	IP address or port scanning tool to look for potential vulnerability(ies)
Attack tools	Files download by the attacker to assist him/her during the attack (File editor, hiding script…)
Flooder	Software used to flood other IP address(es) with a large volume of packets (denial of service)
Privilege escalation	Tool to try to gain root privileges on the host
Download testing files	Big files used by attacker to test the speed of the Internet connection of the host
Library	Libraries added by attackers to make other rogue software work.
VoIP exploit	Tool to exploit vulnerabilities in VoIP software
Audio server	Server to stream Internet radios

By replaying every attack in a sandbox, we were able to gain a good understanding of the attack purpose. In Fig. 8, we see that installing IRC bot and IRC bouncers are still the most frequently observed categories. It is also interesting that in third position, attackers test the network bandwidth or the connectivity by downloading test files.

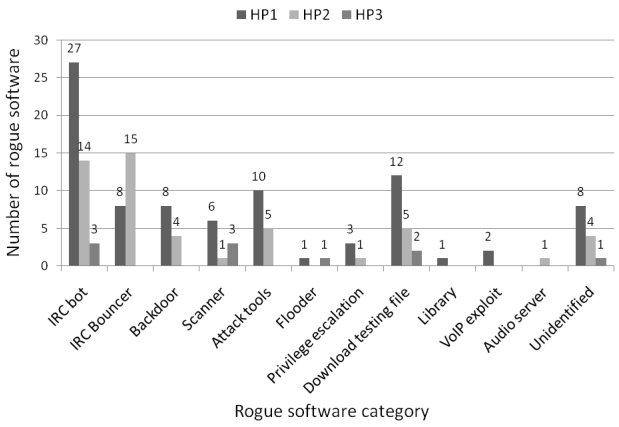

Figure 8. Number of Downloaded files organised per Category and Type of Honeypot

Fig. 8 also provides information with respect to the goal of understanding the attacker's behavior on the long term. By observing time intervals between sessions occuring on a same honeypot since the initial deployment triggered by a

successful brute-force attack, we notice that on average, attackers exploit the brute-forced host after 12 hours. However, for a significant number of honeypots, the following sessions occur on average several days after the deployement. This result is important to justify the extended period of time that we allocated to honeypot exploitation. Limiting honeypot life time to 24 hours [2, 3] would have significantly reduced the scope of our data collection. It is also important to note that the first session corresponds to the installation and the execution of a malicious program for 39% of the honeypots studied.

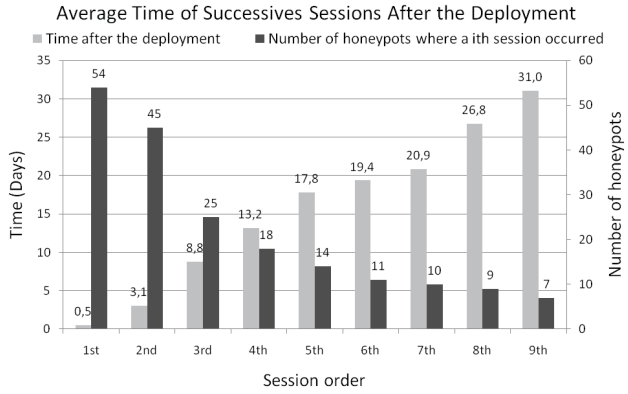

Figure 9. Average Time between Sessions and Number of Honeypots Involved

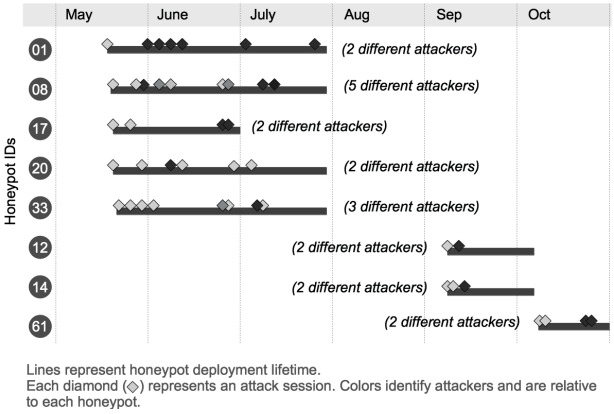

Figure 10. Timeline of Attacker Access to the Shared Honeypots

Another objective was to be able to determine whether or not and how often compromised hosts are shared among attackers. For the 60 honeypots that were targeted by 39 attackers identified in Section IV-C, 20% of attackers shared honeypots. Nine honeypots (15%) were exploited by more than one attacker. We found that seven honeypots had been exploited by two different attackers, one honeypot by three different attackers and one honeypot by five different attackers. This raises the important issue about how the access to honeypots was shared and why. Even though 77% of the attackers changed the password, 15% did share the access with at least another attacker.

Fig. 10 shows the timeline of activity for eight of the nine honeypots that were shared. Each diamond on Fig. 10 represents a session and each shade of grey a unique attacker. These eight honeypots present the particularity of all having an IRC bot installed during their lifetime. IRC logs were analyzed from these bots in order to evaluate the influence of IRC as a vector of transmission of honeypots credentials. We discovered that five of the eight honeypots used two identical IRC servers. We also found credential information being shared on IRC channels.

These results are important to show that attackers seem to be organized to compromise and exploit honeypots. However, we note that we observed this type of shared access for only a minority of honeypots.

V. Related Work

Over the past several years, honeypot-based experiments have significantly contributed to better understanding the threat landscape.

Regarding attacker behavior analysis, [1]-[3] leveraged SSH servers with easy-guessable passwords to collect and study malicious activity. In [1], the authors analyzed 38 attack sessions collected over a period of 131 days. Preliminary results about attackers skills and attack patterns were provided. However, no outbound connections were allowed, which limited attackers actions and prevented the analysis of rogue software. In [2], the authors built a state-machine of attacker actions based on 824 attack sessions. Outbound connections were authorized but honeypots were re-imaged after 24 hours. A more extensive study was conducted in [3], where the same authors studied 1,171 attack sessions and analyzed the 250 rogue software collected over a period of eight months.

Regarding binary analysis, [7] leveraged the SGNET honeypot project to conduct in-depth analysis of polymorphic malware. The authors introduced a clustering techniques based on static and behavioral characteristics analysis.

Regarding dynamic analysis of rogue software, [12] presents a tool for dynamically analyzing the behavior of Windows executables. This tool is available through an online service for analyzing malware [19]. However, only Windows binaries can be submitted for analysis. The same limitation applies to CWSandbox [20], that uses another automated dynamic analysis tool that was introduced by its authors in [11].

Regarding keystroke analysis, [7] considers the case of guessing typed text in an SSH connection by analyzing only the inter-arrival time of network packets. The authors show that network latency has little influence on the results. The tools [14, 15] implement a recognition of commands typed through SSH interactive session using keystroke intervals collected previously from a telnet session. Identification by keystroke analysis is tackled more broadly in [8, 9, 10], where the authors present different algorithms to perform keystroke recognition of users through an analysis of delays between keystroke and keystroke duration. Their approach consists on grouping keystrokes by pairs and observing time interval between this pairs.

CONCLUSIONS

This paper describes an empirical research study to gain insights on attackers launching attacks and the type of attacks launched. The experiment focuses on targets of opportunity where attackers face targets with different network capabilities. We introduced new approaches to characterize the attacker and the attack. In particular, we discussed how to use keystrokes profiles analysis and attacker behavior to have strong evidence that attackers are different. We also introduced criteria to assess attackers skills. We used a large set of different data to conduct our analysis: IP address, AS routing number, network traffic, keylogs, keystrokes profile, attack sequence, and rogue software downloaded.

We found that the main motivation behind attack is to install IRC-based botnets. We collected evidence to show that about 15% of attackers shared honeypot access with at least one other attacker. This represents nine honeypots among which seven of these honeypot had their password changed. This reveals that attackers seem to be organized and share credentials. We note that changing password was a frequent action for attackers (77% of attackers updated the target computer password). Only skilled attackers create a new user but any attacker checked the presence of users. We also noticed that the configuration of the target does not seem to impact the type of attacker attacking it.

The limitations of the experiment are the following. First, since the attackers and attacks evolve, we expect some of these results to change over time. Second, we have only looked at an SSH entry point with different network configurations. We will need to consider other entry points as well as other configurations changes to assess how attackers react to them (e.g., large disk space). Third, our empirical results cover a single organization. While we do not believe that the location has an impact on our results since these are targets of opportunity that were found using automated tools to gain access, it would be interesting to replicate our experiment in another organization to compare the results. Fourth, the keystroke analysis technique assumes that the path followed by the packets carrying attacker's commands is static and does not change over the course of the attack. We believe this assumption holds for the majority of attackers, but it would be interesting to validate it. Fifth, the experimental setup constrained the honeypots to be reset at the beginning of each month. This may have cut attacks occurring towards the end of the month.

ACKNOWLEDGMENTS

The authors thank Gerry Sneeringer and the Office of Information Technology at the University of Maryland for allowing and supporting the experiment and Jérémy Brun-Nouvion for his participation in this experiment.

REFERENCES

[1] E. Alata, V. Nicomette, M. Kaâniche, M. Dacier, and M. Herrb, "Lessons learned from the deployment of a high-interaction honeypot," In EDCC, volume 6, pages 18–20, 2006

[2] D. Ramsbrock, R. Berthier, M. Cukier, "Profiling Attacker Behavior Following SSH Compromises," DSN, 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07), pages 119–124, 2007

[3] R. Berthier, J. Arjona, and M. Cukier, "Analyzing the Process of Installing Rogue Software," in Proc. International Conference on Dependable Systems and Networks (DSN-2009), June-July 2009

[4] J. Zhuge, T. Holz, X. Han, J. Guo, and W. Zou, "Characterizing the irc-based botnet phenomenon," Peking University & University of Mannheim Technical Report, 2007

[5] C. Leita, U. Bayer, and E. Kird, "Exploiting diverse observation perspectives to get insights on the malware landscape," DSN, 40th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'10), pages 393–402, 2010

[6] D.X. Song, D. Wagner, and X. Tian, "Timing analysis of keystrokes and timing attacks on SSH," in Proceedings of the 10th conference on USENIX Security Symposium-Volume 10, page 25, USENIX Association, 2001

[7] M. Abu Rajab, J. Zarfoss, F. Monrose, and A. Terzis, "A multifaceted approach to understanding the botnet phenomenon," in Proceedings of the 6thACM SIGCOMM Conference on Internet Measurement, pages 41–52. ACM, 2006

[8] J. Ilonen, "Keystroke dynamics," Advanced Topics in Information Processing, Lecture, 2003

[9] F. Monrose and A. Rubin, "Authentication via keystroke dynamics," in Proceedings of the 4th ACM Conference on Computer and Communications Security, pages 48–56. ACM New York, NY, USA, 1997

[10] R. Joyce, G.K. Gupta, and J. Cook, "User authorization based on keystroke latencies," Dept. of Computer Science, James Cook University of North Queensland, 1989

[11] C. Willems, T. Holz and F. Freiling, "Toward Automated Dynamic Malware Analysis Using CWSandbox," IEEE Security and Privacy, pp. 32–39, March/April, 2007

[12] U. Bayer, C. Kruegel, and E. Kirda, "TTAnalyze: A Tool for Analyzing Malware," 15th European Institute for Computer Antivirus Research (EICAR 2006) Annual Conference, Hamburg, Germany, April 2006

[13] https://projects.honeynet.org/honeysnap, The Honeynet Project. (December, 2010)

[14] http://chaosreader.sourceforge.net/, Chaosreader network forensic tool (December, 2010)

[15] http://www.brendangregg.com/sshanalysis.html SSH analysis' Chaosreader plugin (December, 2010)

[16] http://www.virustotal.com (December, 2010)

[17] https://projects.honeynet.org/sebek/, Sebek keyloger website (December, 2010)

[18] http://www.wireshark.org/, Wireshark website (December, 2010)

[19] ANUBIS: Analyzing Unknown Binaries: http://anubis.seclab.tuwien.ac.at (December, 2010)

[20] CWSandbox: http://www.cwsandbox.org (December, 2010)