Profiling Attacker Behavior Following SSH Compromises

Daniel Ramsbrock
Department of Computer Science
University of Maryland, College Park
dramsbro@umd.edu

Robin Berthier, Michel Cukier
Center for Risk and Reliability
Department of Mechanical Engineering
University of Maryland, College Park
robinb@umd.edu, mcukier@umd.edu

Abstract

This practical experience report presents the results of an experiment aimed at building a profile of attacker behavior following a remote compromise. For this experiment, we utilized four Linux honeypot computers running SSH with easily guessable passwords. During the course of our research, we also determined the most commonly attempted usernames and passwords, the average number of attempted logins per day, and the ratio of failed to successful attempts. To build a profile of attacker behavior, we looked for specific actions taken by the attacker and the order in which they occurred. These actions were: checking the configuration, changing the password, downloading a file, installing/running rogue code, and changing the system configuration.

1. Introduction

Most security analysis experiments focus on methods for keeping attackers out of target systems but do little to address their behavior after a remote compromise. In this experiment, we focused almost exclusively on post-compromise attacker behavior. Our goal was to build a profile of short-term attacker behavior, capturing the actions in the minutes and hours after the initial compromise.

To achieve this goal, we utilized a set of honeypot computers running SSH on Linux. Attackers routinely scan for this service and use it for gaining both privileged and non-privileged remote access. The very nature of the experiment required us to observe a large number of successful compromises in order to draw conclusions about typical post-compromise attacker behavior. To ensure a large number of compromises, we used commonly tried passwords to attract attackers with a low level of sophistication (the so-called "script kiddies" who rely heavily on automated hacking tools and dictionary attacks).

Section 2 below describes the experimental setup, including the software configuration and usernames/passwords found on the honeypots, the data collection methods, and the typical lifecycle of a honeypot in this experiment. Section 3 presents the basic statistics we gathered as part of this experiment, focusing on the most commonly attempted usernames and passwords. Section 4 presents our findings, including the post-compromise attacker profile in the form of a state machine. Section 5 reviews related work in the area of honeypots and attacker behavior research, and Section 6 presents our conclusions.

2. Experimental setup

To collect attacker data, we used a set of four high-interaction Linux honeypot computers as part of the existing testbed architecture at the University of Maryland. The honeypots are on a separate network that limits outgoing connections to minimize damage but allows all incoming connections. For details regarding the testbed architecture, please refer to [1].

2.1. Software configuration

The four honeypots all ran on an identical Linux disk image: a slimmed-down install of Fedora Core 3, updated with the latest patches as of October 10, 2006. Since the primary interaction with the systems was via SSH, the install only included a text-mode environment (the X Window system and associated graphical programs were not installed).

To monitor attacker activity, we used the following tools: a modified OpenSSH sever to collect attempted passwords, syslog-ng to remotely log important system events, including logins and password changes, strace to record all system calls made by incoming SSH connections, and the Honeynet Project's Sebek tool [2] to secretly collect all keystrokes on incoming SSH connections.

The only modification to the OpenSSH source tree was the addition of a single line of code that uses syslog to record all passwords being tried.

37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07)
0-7695-2855-4/07 $20.00 © 2007 IEEE

2.2. User accounts and passwords

Each honeypot had one privileged root account plus five non-privileged user accounts. To get an idea about commonly tried usernames, we ran some initial experiments. Based on these results, we decided to use the following usernames: admin, mysql, oracle, sarah, and louise. These experiments also revealed that the most commonly tried passwords were '(username)', '(username)123', 'password', and '123456', where (username) represents the username being tried. We rotated among these four passwords for each username as follows: after a compromise, we re-deployed the honeypot and moved on to the next password in the list (see Section 2.4 for details regarding the re-deployment policy).

In order to encourage attackers to enter the non-privileged user accounts instead of the root account, two of the honeypots were set up with strong root passwords. The other two honeypots had root accounts which rotated among the four passwords 'root', 'root123', 'password', and '123456'.

2.3. Data collection

Two servers were responsible for collecting data: one was dedicated to syslog data and the other one collected Sebek data, strace data, and hourly snapshots of the `.bash_history` and `wtmp` files.

Sebek and syslog-ng were configured to send data to the servers continuously. To transfer the large amounts of strace data, we set up an automated, compressed hourly transfer. This was done via SCP using public keys and a hidden system account called 'sysadm'.

2.4. Honeypot lifecycle

To ensure quick turnaround after a compromise, we used a pre-built disk image and automated scripts to manage the deployment of the honeypots. We monitored the syslog messages coming from each honeypot at least every 24 hours to check for logins and password changes. In this context, we defined a compromise as an unauthorized login followed by a password change, rather than using the traditional definition of an unauthorized login only. Password changes typically happened every day, with the observed average time from honeypot deployment until the first password change being 11:25 hours.

Re-deploying immediately after an unauthorized login would have limited our results: due to the automated nature of the dictionary attacks, many attackers successfully gained access but did not perform any actions once they had a shell. On the other hand, keeping the honeypot running for more than a few hours after a password change is not productive for observing short-term attacker behavior: once the password has been changed for an account, all other attackers are locked out of it.

Following a password change, we waited at least one hour before we copied the disk image back onto the honeypot, re-ran the deployment script, and continued monitoring the live syslog data.

3. Attacker statistics

During the 24-day period from November 14 to December 8, 2006, attackers from 229 unique IP addresses attempted to log in a total of 269,262 times (an average of 2,805 attempts per computer per day). Out of these, 824 logged in successfully, and 157 changed an account password. The detailed figures for each honeypot are listed in Table 1.

Table 1. Login attempts per honeypot

Honeypot	Attempted	Successful	Password
HP1	66,087	267	49
HP2	69,044	228	43
HP3	72,953	159	31
HP4	61,178	170	34
Total	**269,262**	**824**	**157**

Despite the fact that we used commonly attempted usernames and passwords, we were surprised to find that only 0.31 percent of attempted attacks were successful. Even more surprisingly, only 22.09 percent of the time (in 182 out of 824 cases) did the attacker run any commands. In 25 cases, the attacker did not change the password despite running other commands. Overall, this resulted in only a 19.05 percent rate of password change among successful logins. This trend can possibly be explained by the automated nature of the attacks: if a low-skill attacker is using scripts to attack dozens of systems at once, he may not have time to take advantage of all compromised hosts.

Table 2. Top attempted usernames

Rank	Username	Attempts	Percent
1.	root	33,238	12.34%
2.	admin	4,392	1.63%
3.	test	3,012	1.12%
4.	guest	2,274	0.84%
5.	info	1,825	0.68%
6.	adm	1,563	0.58%
7.	mysql	1,379	0.51%
8.	user	1,317	0.49%
9.	administrator	1,205	0.45%
10.	oracle	1,169	0.43%

As described in Section 2, we logged all attempted

usernames and passwords. Among the most commonly tried usernames, the privileged root account was by far the most popular choice (see Table 2). Even though attackers attempted a total of 12,225 different usernames, the top 1,000 accounted for 72.45 percent of all attempts. System administrators should avoid these accounts when possible, or otherwise ensure that they have strong passwords. The root account is required, but SSH access to it should be disabled.

Table 3. Top attempted passwords

Rank	Password	Attempts	Percent
1.	(username)	115,877	43.04%
2.	(username)123	23,362	8.68%
3.	123456	19,177	7.12%
4.	password	5,742	2.13%
5.	1234	3,981	1.48%
6.	12345	3,890	1.44%
7.	passwd	3,793	1.41%
8.	123	3,682	1.37%
9.	test	3,564	1.32%
10.	1	2,925	1.09%

While compiling data on the most commonly used passwords (see Table 3), we noticed that attackers were trying variations on the username as the password. In many cases the attempted password was the username itself or the username followed by '123'. As a result, we specifically looked for patterns where the password contained the username, and it turned out that by far the most common password was the username itself. This combination accounted for almost half of all attempts, and the username followed by '123' was the second most popular choice. We also saw a third pattern of this type: the username followed by '321'. However, it did not occur frequently enough to appear in the top 10 list (2552 times, equaling 0.95 percent). Our pattern-based analysis of the attempted passwords provides a clearer picture of the underlying trends than do traditional methods, such as exact string matching. This result again emphasizes the point that a password should never be identical or even related to its associated username.

In a similar study by Alata and colleagues [3], the authors had the same results for the accounts being tried (Table 2 above). Not only were the top three accounts the same, but the percentages each was attempted were nearly identical.

4. Results

While basic statistics about attackers can provide some insight, the main purpose of this experiment was to build a profile of post-compromise attacker behavior. To do this, we developed a list of seven states that represent the typical observed actions (such as 'change password' and 'download file'). We then built a state machine showing the number of times attackers changed from one state to another. A state transition is an indication of sequence: an edge from state X to state Y indicates that the attacker engages in activity X first, then in activity Y (without engaging in any other activity Z in between).

4.1. State definitions

To build the state machine of attacker behavior, we defined seven states as follows.

1. **CheckSW** – 'Check software configuration.' This refers to actions that allow the attacker to gain more information about the system's software or its users. The specific Linux commands included in this state are: `w, id, whoami, last, ps, cat /etc/*, history, cat .bash_history, php -v`.
2. **Install** – 'Install a program.' This refers to new software being installed by an attacker. In most cases, this takes the form of untarring or unzipping a downloaded file, followed by other filesystem operations such as copying, moving, and deleting files, creating directories, and changing file permissions. The specific commands included in this state are: `tar, unzip, mv, rm, cp, chmod, mkdir`.
3. **Download** – 'Download a file.' This refers to remote file downloads by the attacker. Typically, attackers download TAR/ZIP files containing hacking tools such as SSH scanners, IRC bots, and password crackers. The specific commands included in this state are: `wget, ftp, curl, lwp-download`.
4. **Run** – 'Run a rogue program.' This refers to the attacker running a program that was not originally part of the system. To detect these programs, we looked for the `./` notation which usually precedes commands run from locations outside the system's binary path. However, some attackers modified the `PATH` environment variable so they could run their rogue program without the `./` notation. We were able to detect most of these cases because attackers repeatedly used the same kits, resulting in three commonly observed binary names: `cround, [kjournald], httpd`. Finally, some attackers used Perl scripts, so we also included `perl` and `*.pl` in this state.
5. **Password** – 'Change the account password.' This refers to changing the password of the compromised account. The only command included in this state is `passwd`.
6. **CheckHW** – 'Check the hardware configuration.' This refers to actions that allow the attacker to gain more information about the system's

hardware (uptime, network, CPU speed/type). The specific commands included in this state are: `uptime, ifconfig, uname, cat /proc/cpuinfo.`

7. **ChangeConf** – 'Change the system configuration.' This refers to attacker activity that permanently changes the state of the system. Typical examples of this were: setting environment variables, killing running programs, editing files, adding/ removing users, and running a modified SSH server (the one rogue program not considered part of the Run state because of its long-term effects on the system and its users). The commands included in this state are: `export, PATH=, kill, nano, pico, vi, vim, sshd, useradd, userdel.`

Table 4 provides a summary of how many commands matched each state. There are certain commands we did not include in any state because they are routine and have no significant effect on the system: `cd, ls, bash, exit, logout, cat`. These commands made up a large portion of the observed command set (34.08 percent) and are listed as (no-op) in Table 4. Including no-op commands, our state machine provided nearly full coverage of the observed command set (98.07 percent). It is interesting to note that a fairly narrow definition of states results in a high rate of coverage. The most likely explanation is that only a few different scripts accounted for most of the attacks.

Table 4. State machine coverage

State	Commands	Coverage
CheckSW	386	14.90%
Install	377	14.55%
Download	225	8.68%
Run	208	8.03%
Password	203	7.83%
CheckHW	157	6.06%
ChangeConf	102	3.94%
(unmatched)	50	1.93%
(no-op)	883	34.08%
Total	**2591**	**100.00%**

By inspection, we discovered that over half of the 50 unmatched commands were due to typographical errors by the attackers (they were close matches for valid commands). This shows us that while the attackers were most likely following predetermined command sequences, at least several of the attacks were being carried out manually.

4.2. Attacker profile

From the state definitions above, we constructed a profile to illustrate the typical sequence of actions following a compromise. We initially separated attacks on user and root accounts, hoping to see a clear difference between the two. However, we found no significant difference and decided to focus only on the combined dataset in order to make the trends clearer.

Figure 1 contains the state machine representing the typical post-compromise behavior of attackers. The number labeling each edge indicates how many times that state transition occurred, with the five most common shown in bold. The font size of each state indicates how many total command lines fit the state definition, with a larger font indicating a state with more attacker activity.

To make the diagram clearer and more concise, only the top 25 edges are shown, representing a total of 1,138 state transitions (84.11 percent of the total). The remaining 31 edges, representing 215 transitions (15.89 percent), are hidden. As a result, the in-degree and out-degree of each node will not be equal in most cases (though this is true for the full state machine).

The most popular course of action was to check the software configuration, change the password, check the hardware and/or software configuration (again), download a file, install the downloaded program, and then run it. The 'change configuration' action was less popular, though it occurred fairly equally at three different stages: 1) before and after checking the software configuration, 2) before running a rogue program, and 3) after installing software. Overall, the two most popular attacker activities were checking the software configuration and installing rogue software.

Due to our easy passwords and the fairly small set of commands the attackers ran, we can assume that most of them have a low skill level. Spitzner [4] also supports this contention: “Linux systems tend to be the focus of [attackers] ... who use commonly known vulnerabilities and automated attack tools.” Under this assumption, the observed behavior makes sense. The attackers are operating on memorized or automated sequences of commands, trying to build back doors into as many computers as possible. A possible explanation for this behavior is their intent to create botnets, which they can sell for profit. Given this motive, their main objectives are: 1) to check the machine's configuration to see if it is suitable for their purposes and 2) to install their rogue software, giving them full back door control of the machine or allowing them to identify other vulnerable hosts, for example. Most attackers appeared to be particularly concerned about detection while installing their software, repeatedly using the `w` command during their shell sessions. This command alone accounts for 8.11 percent of all commands issued, with only the no-op commands `cd` and `ls` having larger percentages.

The Alata study mentioned previously [3] also performed an analysis of post-compromise attacker

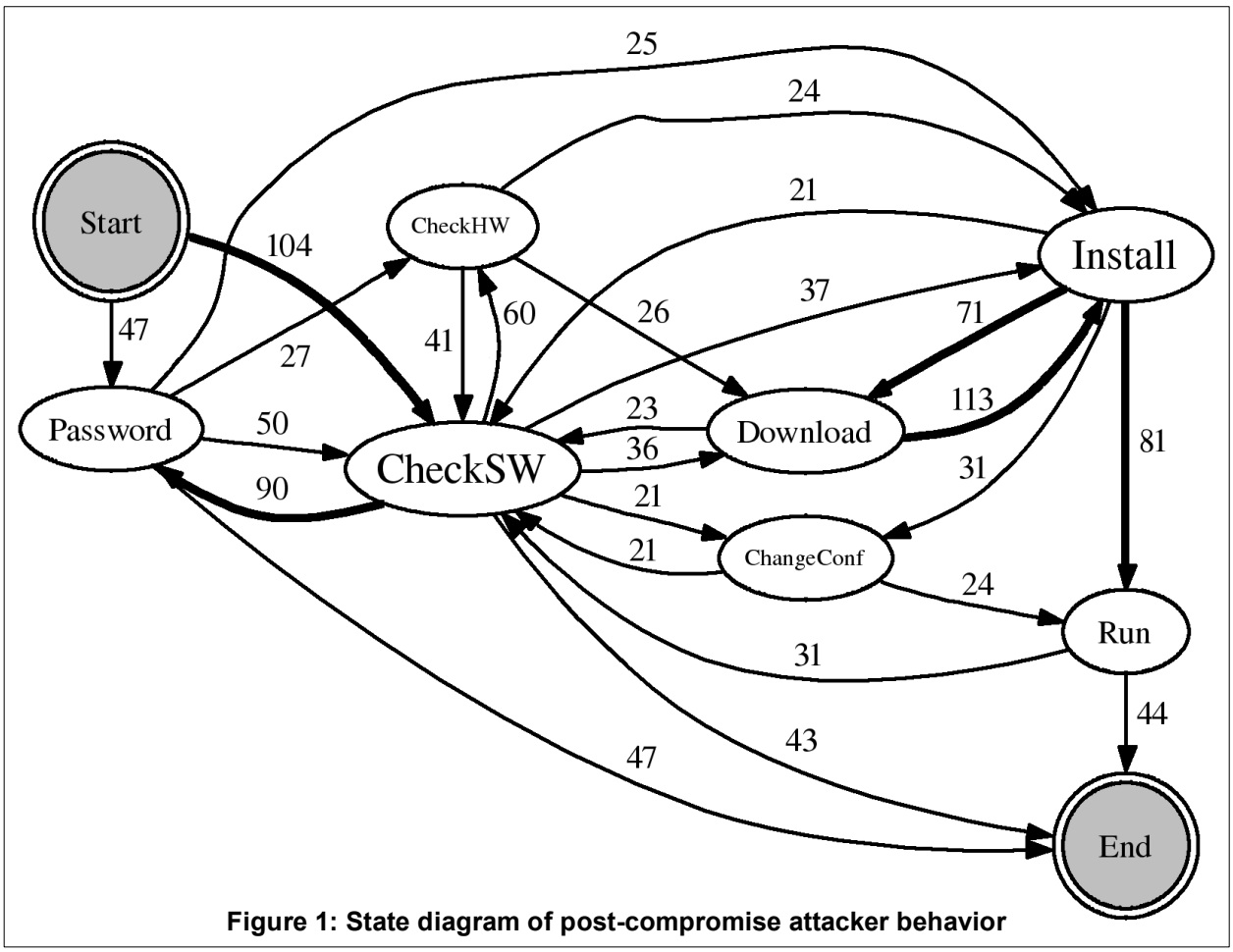

Figure 1: State diagram of post-compromise attacker behavior

behavior, and its findings are again very similar to ours. They also observed password change as the most common first step and reported that most attackers went on to download files (i.e. malicious programs) and then tried to install and run executables.

Another study similar to our experiment is [5]. Here, the authors performed an in-depth forensic analysis of post-compromise attacker behavior. They developed some general categories of attacker behavior: *discovery*, *installation*, and *usage*. However, these categories were much broader than ours and not precisely defined. The main difference between their project and our experiment is that we focused on a larger set of less sophisticated attacks. We gathered aggregate statistics about these attacks rather than investigating individual incidents in detail.

5. Related work

There have been many honeypot-related projects and papers in recent years, often appearing in the Honeynet Project's [6] "Know Your Enemy" series of papers [7].

The study that is most similar to our experiment is [3], where the authors collected both attempted login data and post-compromise attacker behavior. Their results closely match ours, although their study was based on a longer time period (131 days) and also included data from geographically distributed low-interaction honeypots. This suggests that even though our results are based on a smaller sample and shorter time period, they seem representative of overall trends.

Seifert [8] conducted a smaller-scale experiment collecting attempted usernames and passwords, with results roughly equal to ours. He recorded one successful login, providing some information about post-compromise attacker behavior.

Another study closely related to our experiment is [5], as mentioned in Section 4.2. The authors performed a detailed analysis of post-compromise attacker behavior, focusing on the individual actions of more sophisticated attackers rather than gathering summary data for a larger number of attackers.

Dacier and colleagues [9] conducted an extensive statistical analysis on malicious traffic using honeypots. Over a four-month period, they studied attacks from 6,285 IP addresses, averaging over two new sources of attack per hour. In another study, they observed 28,722 new attack sources over sixteen months [10]. In a third study, they analyzed data

collected over one year and conservatively estimated that 753 tools are available to launch attacks [11]. Finally, they found 924 attack sources per day in Germany during a multi-country study [12].

In 2003, Levine and colleagues [13] showed that a honeynet could be implemented on large-scale enterprise networks in order to identify malicious activity and pinpoint compromised machines.

6. Conclusions

In the course of our experiment, we built a profile of typical attacker behavior following a remote compromise and collected valuable data on commonly attempted usernames and passwords. Our findings are useful to the security community in two main ways.

First, these findings allow security and system administrators to adjust their password policies to ensure that no user accounts are open to trivial brute-force dictionary attacks. At minimum, all of the usernames and passwords presented in Section 3 should be avoided. Direct remote root logins should be disabled, only allowing select users to 'su' into the root account once logged on.

Second, these results can assist system administrators in choosing security tools to combat the most common attacker actions. Our results show that downloading/installing/running rogue software and checking the software configuration are the most common actions. Therefore, security tools and policies should focus on those areas. One possibility would be to restrict execution privileges only to registered programs, though this would require significant modification at the operating system level.

Most of our results will not come as a surprise to security professionals, but they are useful because they represent solid statistical evidence to support widely held beliefs about post-compromise attacker behavior. As expected, downloading/installing/running rogue software, checking the configuration, and changing the password were the most common actions following a successful attack. The two main unexpected results were 1) the very low percentage of successful attacks even with purposely weak passwords (0.31 percent) and 2) the low percentage of successful attacks which resulted in commands being run (22.09 percent). A possibility for future work in this area is to focus on finding explanations for these trends.

Acknowledgments

This research was inspired by a semester project conducted by Pierre-Yves Dion.

We thank the Institute for Systems Research and the Office for Information Technology for their support in implementing a testbed for collecting attack data at the University of Maryland. In particular, we thank Jeff McKinney, Carlos Luceno and Peggy Jayant for supporting us in this project with help, material, and space. We thank Gerry Sneeringer and his team for permitting the deployment of the testbed. We also thank Melvin Fields and Dylan Hazelwood for providing some of the computers used in the testbed.

We thank Rachel Bernstein for extensive help with editing, leading to significant improvements in clarity.

This research has been supported in part by NSF CAREER Award 0237493.

References

[1] S. Panjwani, S. Tan, K. Jarrin, and M. Cukier, "An Experimental Evaluation to Determine if Port Scans are Precursors to an Attack", in Proc. International Conference on Dependable Systems and Networks (DSN05), Yokohama, Japan, June 28-July 1, 2005, pp. 602-611.

[2] http://www.honeynet.org/tools/sebek/

[3] E. Alata, V. Nicomette, M. Kaâniche, M. Dacier, and M. Herrb, "Lessons learned from the deployment of a high-interaction honeypot", in Proc. European Dependable Computing Conference (EDCC06), Coimbra, Portugal, October 18-20, 2006, pp. 39-44.

[4] L. Spitzner, "The honeynet project: Trapping the hackers", *IEEE Security and Privacy*, 1(2), 2003, pp. 15-23.

[5] F. Raynal, Y. Berthier, P. Biondi, and D. Kaminsky, "Honeypot forensics", in Proc. IEEE Information Assurance Workshop, United States Military Academy, West Point, NY, June 10-11, 2004, pp. 22-29.

[6] http://www.honeynet.org/

[7] http://www.honeynet.org/papers/kye.html

[8] C. Seifert, "Malicious SSH Login Attempts", August 2006, http://www.securityfocus.com/infocus/1876.

[9] M. Dacier, F. Pouget, and H. Debar, "Honeypots: Practical Means to Validate Malicious Fault Assumptions," in Proc. 10th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC04), Papeete, Tahiti, French Polynesia, March 3-5, 2004, pp. 383-388.

[10] F. Pouget, M. Dacier, and V. H. Pham, "Understanding Threats: A Prerequisite to Enhance Survivability of Computing Systems," in Proc. International Infrastructure Survivability Workshop 2004 (IISW04), Lisbon, Portugal, December 5-8, 2004.

[11] F. Pouget and M. Dacier, "Honeypot-based Forensics," in Proc. AusCERT Information Technology Security Conf. 2004 (AusCERT04), Ashmore, Australia, May 23-27, 2004.

[12] F. Pouget, M. Dacier, and V. H. Pham, "Leurre.com: On the Advantages of Deploying a Large Scale Distributed Honeypot Platform," in Proc. E-Crime and Computer Conference 2005 (ECCE05), Monaco, March 29-30, 2005.

[13] J. Levine, R. LaBella, H. Owen, D. Contis, and B. Culver, "The Use of Honeynets to Detect Exploited Systems Across Large Enterprise Networks," in Proc. IEEE Workshop on Information Assurance, United States Military Academy, West Point, NY, June 18-20, 2003.