Final Research Project:

Effects of Password Hardening and Penetration Testing Tools on Black Hat Hacker Behavior

Nkunim Antwi, Ananya Nadig, Richard Strucko, Aman Thanvi

Summary:

Various tools are available to a hacker's tool belt for infiltrating a network. What if these tools are present from the first login? Little has been done to understand the threat of an ethical hacker's machine being compromised. This critical gap in research caused us to ask the question: what is the effect on hacker behavior as a result of entry hardening and penetration testing tools being present?

Our initial hypothesis operated on the presumption that, as an entry becomes more challenging, the realism of the system being entered increases to the attacker. This realism is the primary difference we theorize would affect hacker behavior. In comparison to the control instance, we theorize that a purely hardened instance increases the authenticity of the hacker, causing them to spend more time using the instance entirely, assuming they have more access to the broader system. In comparison to the control, an easy-to-access instance with pen-testing tools would decrease the realism to the hacker, likely causing them to conclude that such an instance is a honeypot and cause them to leave immediately to avoid wasting their own time. In a combined system containing both conditions, we theorize that the realism would be pretty convincing, likely causing the bad actor to leave immediately in fear that the user has the skills and know-how to track them down.

We tracked command history to correlate attacker behavior on and off the system, session times to associate with our hypothesis about the duration of logins, and the number of login attempts to determine if hardening is a proper deterrent. Unfortunately, as discussed later, some timestamps were not consistent, so that no valuable conclusions could be drawn. However, the remaining data was quite interesting.

Background:

The crux of this research came down to how to quantify hacker behavior. This requires understanding how hackers make use of their remote machines. The current theory on the psychology of hackers finds that one of their primary goals is exploitation (Gerritz, 2021). Whether that be through exploiting data found on the system like passwords, elements of identification, and personal data, or exploiting system resources for use in external hacking, all to obtain external rewards, a hacker will likely find some way to take advantage of a system upon gaining access. In the case of an ethical hacker's machine, both would likely interest the hacker. So the question becomes: how does one monitor the use of compromised machines by hackers for other attacks or general outbound traffic?

The research topic of extrusion detection answers the question above. This detection, from monitoring unauthorized file transfers off the system to outbound traffic towards innocent machines using malicious strategies, is already widespread (Peterson, 2008). However, most research on this topic is centered on detecting compromised systems inside corporate networks, where firewall logs can be used for analysis. Understanding the techniques used by these individuals for monitoring and preventing malicious outbound traffic will be immensely useful in preparing our research. The final question becomes: how do we safely test this hypothesis in a controlled environment?

The answer? Honeypots. Already quite popular in the industry, they lure hackers away from critical network infrastructure at large Fortune 500 companies or gather data on hackers, their tools, and their behavior. The primary advantage of a honeypot is its versatility and control. By using actual operating system kernels on top of existing administrative infrastructure, the functionality of an entire system can be falsely presented to a hacker while posing no real threat

to the system lying underneath. However, this relies on the presumption that the honeypot is appropriately configured and managed to prevent leaks to the outside network.

Design Modifications:

Our original intention was to model both independent variables (password hardening and pen-testing tools present) in all four possible combined iterations. Essentially, there would be a basic, barebones, easy Ubuntu instance to act as a control for data comparisons, a password hardened Ubuntu instance to implement a purely hardened system, and a default-config, non-hardened, “easy” Ubuntu instance with pen-testing tools, and finally, a hardened instance also containing pen-testing tools. However, this task proved more challenging than first anticipated.

Our inexperience with the tools presented to us caused us to initially reroute all internal traffic on the VM back to the VM itself rather than routing traffic to the containers. Following that correction and reboot, only three of our four instances were working, despite all running and being accessed with SSH and nearly identical networking and routing configurations. There were no differences in IPs, automatic routing access versus traditional SSH access, and the number of allowed attempts.

The non-working instance was our purely hardened instance; however, our initial research indicated that successful connection and session data for this instance would be limited due to the manual access required, a hypothesis supported by our limited session data for the hardened and pen-testing instance (from hereon referred to as the “double” instance), meaning that not much was lost here, leaving our central hypothesis left to test in regards to the effects of pen-testing tools being present.

Additionally, we planned for more significant variations in container configuration between the less secure and more hardened honeypots, such as in-depth YAML configuration files to manipulate how the container hosting software simulated available resources, different file protections, authorization levels, etc. However, due to team miscommunication, this was not fully implemented before deployment and would have meant losing even more time for data collection, so we opted for a more barebones configuration instead.

Data Collection:

There were three primary things we looked to collect regarding determining hacker behavior on the instances most clearly. This was:

- Session timestamps to assess the duration of malicious behavior
- Attacker keystrokes to determine hacker command behavior
- Number of login attempts to determine the persistence of hackers

Luckily, most of this data was collected and preserved automatically by the Man-In-The-Middle (MITM) tool supplied to us by the ACES program. This tool was used to manage the routing of the public IP pings to the honeypots through the VM they resided on, primarily to restrict off-network traffic and easy management of the number of attempts and automatic access. However, it also collected data on sessions, including session timestamps, attacker keystrokes and command outputs, and login attempts listed by IP address, username, and password/key used for entry.

Due to initial routing errors, we had five extra days from the control instance, with around two weeks worth of data for the other two working honeypots. Keystrokes were automatically logged and manually analyzed to determine the purpose of commands and who

used them. The number of IP addresses and their attempt numbers was done using a combination of common Unix processing commands, including cut, grep, sort, and uniq.

Some session times are not precise, as only the beginning of entry was logged, along with the end of the current recycling session of the honeypot, giving us maximums on session times but not concrete data. As a result of this issue, this analysis was ultimately scrapped in favor of an average number of attacks per IP to determine the persistence of users in gaining entry.

We exclude some early days of data due to hindsight in design on creating new honeypot snapshots based on current data, where we realized we might be preserving a compromised state of the honeypot, along with some data collected during our initial tests, like initial logins to test MITM and then immediately manually exiting the terminal, where “exit” was all that was logged.

Data Analysis:

We analyzed the data using data processing Unix commands to extract the attackers’ IP addresses and pull the commands they used during each session. We decided to analyze IP addresses because we wanted to observe attackers' behavior and see the difference in the number of login attempts between IP addresses of different regions. We also wanted to analyze commands to understand better what the attackers wanted to accomplish.

To produce the tables and graphs below (located in Appendix B), we first used the *“cat [honeypot login_attempts .txt file] | cut -d';' -f2 | sort”* command to extract the IP addresses from the login attempts of each honeypot. Unfortunately, we could not see any login attempts from our Hardened honeypot, so we could only gather attacker IP addresses from our Control, Double, and Pentest Honeypots. Once we got the list of sorted IP addresses for these three

honeypots, we saved those lists into a text file and used Python to read the files. Then, using Python, we imported Pandas, Matplotlib, and Numpy to save the data into a data frame and create frequency plots for each honeypot. Each frequency plot shows the top 10 attacking IP addresses, sorted by the number of attack attempts on May 7, 2022.

To find all the commands used during login attempts, we looked through MITM session data for each honeypot (in .gz files) and combined several .gz session data files used for each honeypot. Once we had the file of all session data for each honeypot, we specifically used *"cat honeypot-session.txt file | grep "^Noninteractive**" to look at the commands used alone during each session. We also used *"cat honeypot-session.txt file | grep -B7 "^Noninteractive**" to help associate the attacking IP address with the specific commands used. Unfortunately, we did not have any session data available for our Pentest, Double, and Hardened honeypots, so we could only pull three commands used by attackers from our Control honeypot and analyze those.

We have several theories for why there was no session data, but one of the most prevalent came down to a lack of attempts and an interesting pattern noticed in the Control honeypot for sessions. According to MITM, only four sessions had any logged commands of the nearly three weeks' worth of session data gathered from the Control honeypot. Most of these sessions compromised the honeypot, followed by immediately exiting. Two IPs returned later using the credentials they had logged. In contrast, others returned after resetting the honeypots, deleting their login data, effectively resetting their process, and making most of their sessions useless. Coupled with the fact that the successful compromise of the Double honeypot was virtually zero, and the Pentest honeypot was about a third that of the Control honeypot according to MITM standard output logs, it is possible that we would have only received only about one extra session log for the Pentest honeypot that would have been anything more than just entry and exit.

As of May 7, 2022, we ended up with 119,267 total login attempts for our Control honeypot, 8,608 total login attempts for our Double honeypot, and 10,885 total login attempts for our Pentest honeypot. When only looking at the top 10 attacking IP addresses, we had 7,291 login attempts for our Control honeypot, 5,036 login attempts for our Double honeypot, and 6,264 attempts for our Pentest honeypot. At the same time, our control honeypot had started collecting data about a week ahead of the other two because of misconfiguration issues on our end. This somewhat supports our initial hypothesis, as we assumed that our control instance would be easier to get into due to the lack of pen-testing software. We also expected the doubled instance to have fewer login attempts since that honeypot included a combination of pen-testing tools and a more difficult point of entry.

What we also found interesting from the session data for our control honeypot was that most of the attackers did not type any commands during most login attempts. We could only pull three commands where one of them was used twice ("cat /proc/cpuinfo | grep name | wc -l"). Attackers from IP addresses 80.248.67.11 and 178.128.220.159 entered this command on two different occasions, and we assumed that they were trying to get information about the CPU. Another long command was entered by an attacker from IP address 200.129.102.38, which attempted to remove the current SSH configuration on the honeypot and replace the authorized_keys file with their own RSA key. We believe this was with the intention of having a surefire way of reentering the system regardless of whether the system admin reset the password for the account. The specific command entered is featured in Appendix B.

The commands we gathered conflicted with our initial hypothesis since we expected to see fewer commands from the control compared to the double and hardened honeypots that

would demonstrate an increase in hacking behavior. However, we cannot draw sufficient conclusions due to the lack of session data for our hardened, pen-testing, and double honeypots.

After looking at the results from our graphs, we learned that our control and double honeypots had the most login attempts from an attacker using the IP address 171.232.4.198. That attacker attempted to log into the control honeypot 2,025 times and the double honeypot 2,021 times. In the pentest honeypot, that attacker came in 2nd with 915 login attempts, compared to 2,898 attempts from IP address 59.32.148.185. We looked up both IP addresses and learned that the first attacker (171.232.4.198) is in Vietnam, while the other (59.32.148.185) is from China. We found these results fascinating, as one of our team members had worked with an SSH project in the Summer of 2021, where he struggled with debugging SSH logs due to continuous attempts from Chinese hackers trying to access his machine. Our current working hypothesis based on this data and our prior experience is that Chinese hackers working in various nations in the area are attempting to gain entry to any and all machines for testing cyberattacks or finding useful "honey."

Conclusion:

Lacking certain aspects of the data we desired, such as session timestamps, means our conclusions are partially incomplete based on our hypothesis. However, we are reasonably sure that one of our hypotheses is accurate, and another is likely false from our data.

The number of attempts varies wildly between our three instances, with the most attempts on the control and the least on the double instance. Additionally, when looking at the login attempts on each of our instances, the average number of attempts per IP is significantly higher on the control than on the pen-test, which is also substantially higher than the double. This is

partially due to the extra few days of data on the control, but even the average number of attempts per day running is still higher. This means that the hackers were deterred in some way by the pen-testing and double instances.

There was no session data from the double, likely because of our point of entry challenge. The few bits of session data from the pen-testing instance consisted of gaining entry and immediately exiting. While this data also existed on the control, the control also featured actual commands such as those featured in Appendix B. This seems to imply that two different things caused the deterrence of the hackers from the other instances.

We believe that the smallest number of attempts being on the double instance is because of the challenge of entry, lowering the persistence of the hackers. While this does not directly disprove our hypothesis on hardened honeypots, given that we have no session data for their behavior once in the instance, it is likely false; given that attacks were far less persistent in gaining entry to the double instance, we feel it is likely that a hacker would ever even gain access to a hardened instance. This behavior is likely because easier targets exist, such as our easier honeypots. However, a consequence of this is that they are unlikely ever to reach a point where we could prove our hypothesis.

On the other hand, we believe that the slightly higher persistence seen on the pen-testing instance, coupled with no valuable session data, helps to prove our hypothesis on our undefended pen-testing instance partially. We theorized that an undefended instance containing standard penetration testing tools by default would appear suspicious to hackers, likely hinting that the device was not actual. This lack of realism would act as a deterrent, likely sending them off of the instance on the assumption that they had gained access to a honeypot and were trapped on a useless network regardless. This is different from the control. Several IPs returned to the

honeypot after resets kicked them from the system, indicating a higher level of persistence and realism of the control compared to the pen-testing instance.

As mentioned above, the lack of session data for behavior on the double honeypot leaves us unable to make presumptions on how both treatments affect hacker behavior, leaving a gap in our research that would best be resolved with a new method of hardening the point of entry, making it easier to access so that we would have real, practical data.

Ultimately, the most significant limitation in our research was the time constraints and the lack of session data, even for fully functional instances. Due to our inability to timely troubleshoot problems with three of our four instances, we shorted ourselves on data compared to our peers. We already were crunched for data with only five weeks to collect data as it is. Additionally, for a reason we could never discern fully, the number of sessions recorded by MITM was minimal, and most of them were useless instances of gaining access. Subsequently, exiting the system without any commands entered. This greatly limited our ability to decipher conclusions from our data.

With more time to collect data and design the research, we would revise the hardening of our point of entry so that hackers can genuinely gain access to the system, giving us session data on command usage that can be used to discern behavior. We would also spend more time designing data collection systems to ensure all of our required data is collected for proper analysis. For example, our timestamps were inconclusive, and our command history was nearly nonexistent for hackers who gained entry into our current setup.

Our conclusion as a whole was inconclusive without further concrete hacker behavior on the hardened instances. However, sufficient evidence supports our hypothesis on easy access with pen-testing tools present, implying that the machine is a trap or bait. If we could continue

this project, we would work to implement all the features we previously mentioned we had to drop due to time constraints. We would also want to collect and parse our data in more diverse and meaningful ways (i.e., developing more detailed and comprehensive metrics).

Appendix A

We learned that endeavors like these take longer than was allocated to us. We also learned that even the smallest blockers or hiccups could cause significant delays or problems down the road for the project if not resolved promptly and adequately. Overall, our group thought the project was exciting and meaningful. Despite our limited data due to previously mentioned roadblocks, regarding the attacks and attackers' behavior, we were shocked and intrigued to see that our initial presumptions and hypotheses regarding the presence of specific tools and configurations on a machine affect attackers' behaviors and interactions with a given machine. Some significant pitfalls and failures encountered during the experiment include the ones mentioned above in our conclusion. To be more specific, the most significant failures in our research were the lack of session data, even for fully functional instances, and our inability to timely troubleshoot problems with three of our four instances.

Appendix B

Top 10 Attacking IP Addresses on Control Honeypot

```
171.232.4.198      2025
202.40.190.10      1981
179.43.142.180      594
36.110.228.254      568
111.206.120.172     471
62.210.130.171      412
111.240.30.141      412
2.196.138.199       303
179.43.167.74       279
195.3.147.60        246
Name: IP Address, dtype: int64
```

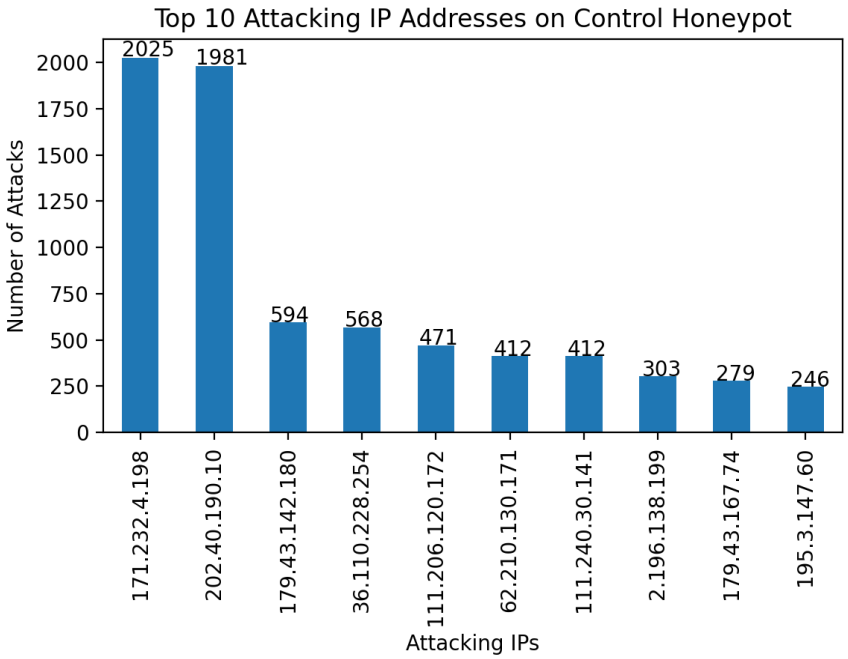
Top 10 Attacking IP Addresses on Double Honeypot

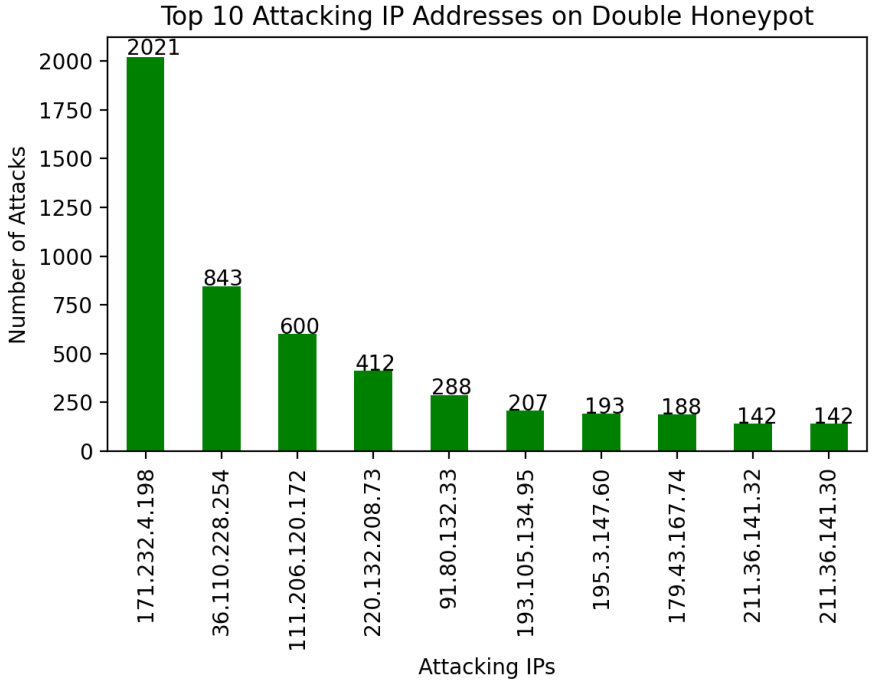
```
171.232.4.198      2021
36.110.228.254      843
111.206.120.172     600
220.132.208.73      412
91.80.132.33        288
193.105.134.95      207
195.3.147.60        193
179.43.167.74       188
211.36.141.30       142
211.36.141.32       142
Name: IP Address, dtype: int64
```

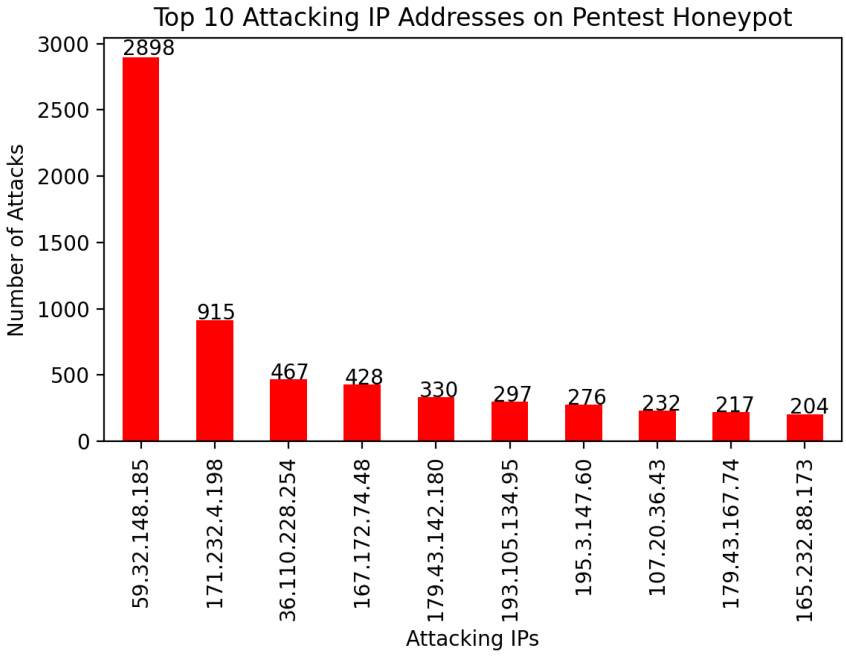
Top 10 Attacking IP Addresses on Pentest Honeypot

```
59.32.148.185     2898
171.232.4.198      915
36.110.228.254     467
167.172.74.48      428
179.43.142.180     330
193.105.134.95     297
195.3.147.60       276
107.20.36.43       232
179.43.167.74      217
165.232.88.173     204
Name: IP Address, dtype: int64
```

- Graphs







- Interesting scripts

```
root@lxc-host:~/MITM_data/sessions/control# cat control_sessions.txt | grep "^Noninteractive*"
Noninteractive mode attacker command: cat /proc/cpuinfo | grep name | wc -l
Noninteractive mode attacker command: cat /proc/cpuinfo | grep name | wc -l
Noninteractive mode attacker command: cd ~ && rm -rf .ssh && mkdir .ssh && echo "ssh-rsa AAAAB3N
zaC1yc2EAAAABJQAAAQEArDp4cun2lhr4KUhBGE7VvAcwdli2a8dbnrTOrbMz1+5O73fcBOx8NVbUT0bUanUV9tJ2/9p7+vD
0EpZ3Tz/+0kX34uAx1RV/75GVOmNx+9EuWOnvNoaJe0QXxziIg9eLBHpgLMuakb5+BgTFB+rKJAw9u9FSTDengvS8hX1kNFS
4Mjux0hJOK8rvcEmPecjdySYMb66nylAKGwCEE6WEQHmd1mUPgHwGQ0hWCwsQk13yCGPK5w6hYp5zYkFnvlC8hGmd4Ww+u97
k6pfTGTUbJk14ujvcD9iUKQTTWYYjIIu5PmUux5bsZ0R4WFwdIe6+i6rBLAsPKgAySVKPRK+oRw== mdrfckr">>.ssh/aut
horized_keys && chmod -R go= ~/.ssh && cd ~
Noninteractive mode attacker command: cat /proc/cpuinfo | grep name | wc -l
```

References

Gerritz, C., 2021. *Top 20 Most Common Hacker Behaviors*. [online] Security Boulevard. Available at: <https://securityboulevard.com/2021/03/top-20-most-common-hacker-behaviors/> [Accessed 3 May 2022].

Peterson, D., 2008. *Extrusion Detection to Detect Attacks - Dale Peterson: ICS Security Catalyst*. [online] Dale Peterson: ICS Security Catalyst. Available at: <https://dale-peterson.com/2008/05/14/extrusion-detection-to-detect-attacks/> [Accessed 3 May 2022].