Picky Attackers: Quantifying the Role of System Properties on Intruder Behavior

Timothy Barron
Stony Brook University
tbarron@cs.stonybrook.edu

Nick Nikiforakis
Stony Brook University
nick@cs.stonybrook.edu

ABSTRACT

Honeypots constitute an invaluable piece of technology that allows researchers and security practitioners to track the evolution of break-in techniques by attackers and discover new malicious IP addresses, hosts, and victims. Even though there has been a wealth of research where researchers deploy honeypots for a period of time and report on their findings, there is little work that attempts to understand how the underlying properties of a compromised system affect the actions of attackers. In this paper, we report on a four-month long study involving 102 medium-interaction honeypots where we vary a honeypot's location, difficulty of break-in, and population of files, observing how these differences elicit different behaviors from attackers. Moreover, we purposefully leak the credentials of dedicated, hard-to-brute-force, honeypots to hacking forums and paste-sites and monitor the actions of the incoming attackers. Among others, we find that, even though bots perform specific environment-agnostic actions, human attackers are affected by the underlying environment, e.g., executing more commands on honeypots with realistic files and folder structures. Based on our findings, we provide guidance for future honeypot deployments and motivate the need for having multiple intrusion-detection systems.

ACM Reference format:
Timothy Barron and Nick Nikiforakis. 2017. Picky Attackers: Quantifying the Role of System Properties on Intruder Behavior. In *Proceedings of ACSAC 2017, San Juan, PR, USA, December 4–8, 2017,* 12 pages.
https://doi.org/10.1145/3134600.3134614

1 INTRODUCTION

Despite our advancements in security policies and mechanisms, attackers are still able to compromise systems using a wide-range of techniques including social engineering, abuse of password leaks and credential reuse, and mere brute-forcing. The Mirai botnet [19], a large botnet which is composed of compromised webcams, routers, and other IoT devices, is a living testament that, even today, attackers can create botnets that are powerful enough to cause Internet outages at a country level, by merely abusing default login credentials of the aforementioned devices [10].

To understand how attackers compromise systems and what they do once they successfully break-in, researchers and security practitioners have been using honeypots. Honeypots are systems that are purposefully insecure, with the goal of luring attackers away from production systems and allowing researchers to study their techniques. Honeypots can vary depending on the level of interaction that they support (e.g. low, medium, and high-interaction) and the type of system that they purport to be (e.g. a general-purpose server vs. a SCADA system). Even though there is a lot of prior work in designing, deploying, and measuring the results of honeypots, in most cases the studies conducted were purely observational in nature [2, 16, 20, 28, 29, 33]. That is, one or more honeypots were deployed in one or more parts of the world and the researchers, at the end of their study, analyzed the recorded attacks describing their observations and discovered patterns of attacks.

In this paper, we report on the first large-scale study using medium-interaction SSH honeypots where, in addition to deploying honeypots and monitoring their activity, we change the environments of some honeypots in a controlled fashion and observe how these changes affect attackers. Specifically, we observe how the location of the honeypot, the difficulty of breaking in, and the file population on the compromised system has an effect on the actions of intruders. Over a span of four months, we deployed 102 honeypots in three different continents and received 12 million connections originating from 38K unique IP addresses. By analyzing the collected data, we observed patterns which we argue are of critical importance to future honeypot deployments and specific types of intrusion detection systems that rely on attackers interacting with monitored files, such as, tripwires and decoys [7, 31, 34]. Among others, we discovered that i) location and hosting provider matters, ii) attackers are very opportunistic (trying very few credential combinations before moving on to the next target), iii) the difficulty of break-in correlates with the eventual presence of a human attacker and iv) while human attackers are interested in finding and exfiltrating user files, bot attackers (the majority of attackers) are not. To rekindle interest in honeypot research and how subtle differences have a large effect on attackers, we will open source our code (modified honeypot software, Ansible scripts for VM preparation [4], and log-processing code) and collected data.

2 EXPERIMENTAL SETUP

In this section we begin by describing Cowrie, our choice of honeypot, providing details about our setup and the modifications made to test our hypotheses. We test three properties of honeypots corresponding to different phases of an attack. The first is the physical location of honeypots which impacts the target discovery/selection step *before login*. The second is the difficulty of break-in which affects attackers *during login*. The third is the population of files on the honeypot which is relevant to attackers' actions *after login*. Once an attacker has progressed to the next phase of the attack they are no longer directly affected by differences in the previous step, which allows us to minimize convolution of experimental parameters. Performing a systematic study of these three dimensions required the deployment of 102 different honeypots. By limiting ourselves to three properties we avoid combinatorial explosion in the number of honeypots required to make fair comparisons.

2.1 Our choice of honeypot

Honeypots are categorized based on how much they allow an attacker to interact with them, ranging from *low-interaction* honeypots which support the bare minimum of a protocol (e.g. showing a fake SSH prompt and recording the authentication attempts of attackers) to *medium-interaction* honeypots which allow much more freedom to an attacker but are still only simulating a real system, and *high-interaction* honeypots which are essentially real, non-simulated, systems. Due to space limitations, we refer readers who want more information about the advantages and disadvantages of different types of honeypots to the Honeynet project [26] and the book by Provos and Holz [27].

After evaluating several alternatives, we decided to use the open-source medium interaction honeypot called Cowrie [22]. Cowrie emulates a file system and shell for every user who logs in. It attempts to mimic the behavior of a real system as faithfully as possible, while logging every action that a user takes from connect to disconnect. Every session begins with an identical fresh setup and any changes, such as, the creation of new files, only last for the duration of that session. Moreover, each session is sandboxed so that the actions of different users have no impact on their peers.

While Cowrie is a medium interaction honeypot (and therefore does not support all the possible commands and operations that attackers may try) it does attempt to provide plausible reasons for the operations that it cannot support. For example, if attackers attempt to clone a git repository they will encounter the following error: `bash: git: command not found`. If they then attempt to use a package manager to install the missing `git` program, e.g., `apt-get install git`, Cowrie will show output as if it is installing git, but when an attacker attempts to use it they will get `git: Segmentation fault`. As such, an attacker may come to the conclusion that the machine they have broken into is one with configuration issues, e.g., buggy shared libraries, but not a honeypot. Overall, we argue that Cowrie is a good choice for investigating the overall motivation of attackers who break into our honeypots, without us needing to take the collateral risk typically associated with high-interaction honeypots. Finally, Cowrie is implemented in Python which gave us the ability to rapidly prototype the modifications described in the next sections.

2.2 Differences in file populations across honeypots

Contrary to prior work, our main motivation for this study is not to conduct one more observational study where we deploy honeypots for some time and then report our findings. Even though we welcome statistics about the number and location of attackers, the types of commands they use, and their overall interaction with our honeypots (and report some of them in later sections), our main goal is to understand whether an attacker's actions are dependent on certain properties of the compromised system. One such property is whether a system appears to be in-use, as evidenced by the files and folders present on its filesystem. Note that this is an "after-login" property since attackers will only encounter it after they have discovered and compromised a system.

To test how the population of files affects an attacker's actions (e.g. do they interact more with the filesystem when it appears to be in use?) we deployed three varieties of honeypots: *Vanilla, Random,* and *Curated.* A *Vanilla* honeypot is an unmodified Cowrie installation with the only change being that of a new password. This serves as our control. For the other two varieties we populated the home folders with files to make the honeypots seem like they had been used, i.e., real users have created directory structures and files within these directories.

In the *Random* variety of honeypot, we created between 1 and 3 new users and randomly populated their home folders with files and directories containing more nested files and directories. Each Random honeypot has separately generated files so they do not look the same, in case an attacker breaks into multiple honeypots. The home folders have between 5 and 8 recursively populated directories and between 2 and 4 files. Each nested directory has between 2 and 10 files and 0 to *depth* directories where *depth* starts at 4 and decreases by 1 at each level. Since we are not aware of any study that quantifies the number of files and directory structures of an average user, we chose these values empirically in an attempt to make the structure seem natural and varied. File and folder names were created using the natural filename generator library [30] (slightly modified for shorter names) and then randomly assigned extensions of common file types. The contents of the files were random bytes with a Gaussian size distribution centered around a general average for that file type, e.g., 6 MB for MP3 files.

The third type of honeypot (which we called *Curated*) is one where we put additional effort into making the files significantly more believable than random generation allows. We added between 1 and 3 home folders and each was populated with common directories, such as, Documents, Pictures, Music, Videos, and Downloads. Each of these contained random files with extensions matching what would be expected for that location, e.g., PDFs in Documents and MP3s in Music. We also added a directory called IMPORTANT with a small number of PDFs and DOCs to see if it would draw any more attention from attackers. Our exact choices were informed by our own file systems and the ones from colleagues who allowed us to inspect the structure of their home folder.

2.3 Honeypot location

To understand whether some honeypot locations are more preferable than others, and whether different locations attract different kinds of abuse, we deployed our honeypots in four different locations across the globe. One set was hosted on servers at our university campus (East Coast, US). The other three sets were placed on public clouds in northern California, USA (NA), Frankfurt, DE (EU), and Singapore (Asia). We chose Amazon Web Services (AWS) and Linode since both companies provide virtual private servers in all aforementioned locations, and deployed the same number of honeypots on each one. In each location/host we set up two honeypots of each type, in order to obtain redundant data and counter, as much as possible, the inherent randomness of honeypot break-ins. Combinatorially, this allowed us to deploy 36 honeypots on the two public clouds, and 6 more on our campus, bringing the total number of honeypots to 42.

2.4 Difficulty of breaking in

The final experimental parameter is that of break-in difficulty, i.e., how sophisticated must an attacker be in order to successfully break into our honeypots. To evaluate this parameter, we performed three different honeypot deployments changing the type of password that is required in order to login and whether the login method is through brute-force or prior credential leakage.

Round 1. In the first round of this experiment we wanted it to be easy for any attacker to log in so all honeypots accepted any username and password combination. Since it might have appeared suspicious if attackers always got in on their first try, instead they were accepted after a number of password attempts (the "correct" password was randomly chosen to be between the first one tried and the third one tried). This means any attacker who connected and made at least three login attempts was guaranteed a successful login. The password they were accepted with was cached with the attacker's IP address, so that if they returned later they were allowed in with only that same password. In terms of landing directories, when an attacker logged in they started in `/root` unless they happened to have guessed the user name of a home folder we randomly generated, in which case they started there. This round consisted of 42 honeypots monitored for a period of 41 days.

Round 2. In the second round we changed the way we allowed attackers in our honeypots by assigning specific credentials to each honeypot instance. All user names were root since that was the most commonly guessed user name in Round 1. Each password was chosen randomly from the RockYou password leak [8] which is one of the largest plain-text password leaks publicly available. We sorted the passwords in this list by their popularity and excluded the top 20 passwords, choosing a random password from the next 1,000 passwords. Our goal was to make it unlikely that an attacker happens to guess the password on the first try, but still allow for brute-force break-ins for attackers who use dictionaries with popular passwords. This allowed us to quantify how many attackers put the time into brute-forcing, how many gave up quickly, and to what extent this has an impact on the activity after successful login since we are, in principle, dealing with more dedicated attackers. It also allowed us to single out attackers that logged in successfully on their first attempt. In these cases, it is likely that a brute-forcing bot broke in, saved the credentials, and passed them along to a human for later inspection. As in the previous round, this round consisted of 42 honeypots monitored for a period of 43 days.

Round 3. In the last round we took inspiration from the recent work of Onaolapo et al. [21] who quantified the behavior of attackers on stolen webmail accounts by "leaking" them to public hacker forums and paste sites. As in the previous round, each honeypot was given a single set of credentials. User names were generated randomly from a list of celebrities with first and last names shuffled. In this round attackers start in the home folder of the user we created unless it is a *Vanilla* honeypot in which case there is no home folder for their user name and they will start in `/root`. Passwords were chosen randomly from the full RockYou list of 14,344,391 passwords. These passwords should be significantly harder to brute-force by chance in a reasonable amount of time, but also look like passwords realistic users would have rather than using entirely random strings.

The full set of user names, passwords, and IP addresses of the honeypots was posted to forums with a brief explanation. Similar to Onaolapo et al., we claimed that we were in the possession of thousands of stolen SSH credentials which we wanted to sell for exclusive access. To prove that we indeed were in the possession of these credentials, we provided a "sample" of 9 sets of credentials which attackers could try out before contacting us for more.

We posted our message to eight different hacker forums, (e.g. `bestblackhatforum.eu`, `offensivecommunity.net`, and `nulled.to`). Since our posts are behind registration walls and do not follow a well-defined format, we reason that the only successful connections against the honeypots of Round 3 must be from attackers who read the posts and decided to inspect the sample accounts.

A similar strategy was applied to paste sites where we posted the other set of credentials and the same story. Paste sites have been historically popular among hacking groups who break into websites and services and then make their spoils available through public pastes. As such, we reason that certain paste sites are monitored by attackers as a way of discovering stolen credentials. We used the most popular public paste sites, `pastebin.com`, `pastie.org`, `codepad.org`, `pasted.co`, and `slexy.org`. We reason that the only realistic way of finding our posts are through crawlers, which crawl the recent pastes that are publicly available, search for specific keywords, and alert their authors when potentially sensitive content is discovered. At the same time, we consider it highly unlikely that the bots have the ability to properly tokenize the text in arbitrary posts and, in our case, initiate SSH connections. Therefore, as for the credentials posted in our forums, we expect successful logins to be initiated by human attackers. Finally, to increase the probability that our posts would be captured by the aforementioned crawlers, we developed a Selenium-powered automated poster which posted new pastes twice a day.

Since we are actively advertising these credentials and do not rely on passive scanning from attackers, for this round, we decided to deploy our honeypots on a single public cloud (Linode) where we utilized 18 honeypots (9 for forum posts and 9 for paste sites) spread in three geographical locations (NA, EU, Asia).

2.5 Ethical Considerations

Before presenting our results, we want to touch upon our ethical considerations. To understand how attackers break into systems with weak credentials and how properties of these systems affect their behavior, we cannot avoid deploying honeypots in the real world and recording the actions of attackers on these honeypots.

Throughout our experiment, we have followed best practices as described in prior honeypot-related work [2, 16, 20, 28, 29]. By using medium-interaction honeypots that do not allow attackers to execute their programs and tunnel traffic to the outside world, we sacrifice some of our observing power in order to ensure that our servers cannot be abused to attack third parties. We do not collect any personally identifiable information from attackers and we only use the collected IP addresses for geolocation and unique-counting purposes. Moreover, we do not provide any software that attackers could download and execute on their own machines and we do not attempt to perform any sort of attack attribution. We submitted our experiment description and protocol to our institute's IRB which,

after evaluation, was deemed to be not "Human Subjects Related" (the board decided that the commands that attackers send to our honeypots do not constitute PII). Given the above, we are confident that our study did not have any adverse effects, neither to the attackers themselves, nor to any third-party service.

3 RESULTS

3.1 Effects of honeypot location

Figure 1a shows the average number of connections per day received by honeypots in different locations during Rounds 1 and 2. Honeypots in North America (NA) were slightly more popular than the rest. EU and Asia received roughly the same number of connections, and those on our campus received slightly less.

The host the honeypot was on had a much larger impact than geographic location. Figure 1b shows that honeypots on AWS received significantly more connections than their counterparts on Linode. Two factors may contribute to this result. First, AWS is a larger and more popular cloud provider which makes them a bigger target for attackers. Second, Amazon makes the IP ranges of their service publicly available [3] so it is easy for an attacker to scan the entire range for EC2 where they may assume that many machines will be running SSH servers. Looking at hosts explains the lower number of connections to the campus honeypots which in fact received a similar number of connections compared to all the Linode honeypots. This means the lower average in Figure 1a is due to the increased popularity of AWS rather than a lack of visibility or interest in the campus honeypots.

We noticed that the interquartile distance for honeypots on the same hosting service is significantly smaller for the Linode and Campus honeypots than the AWS honeypots. This is likely because the Amazon IP ranges are broader. With IP addresses closer together, an attacker who was scanning IP space and found one honeypot address was likely to also find the others thereby decreasing the variance for Linode and our campus.

Table 1 shows the total number of successful logins for each experiment. For Rounds 1 and 2 we give the average number per day as well to make up for the small difference in duration. The per day average is not a fair comparison for Round 3 because we leaked passwords at different times throughout the experiment rather than waiting for attackers to guess passwords set up at the start. By comparing the number of successful logins to total login attempts in Round 2, we find that only 6.25% of attackers are persistent enough to brute-force relatively common passwords. This is an unexpected finding because, as described in Section 2.4, we chose very popular passwords from password leaks which we theorized would be incorporated in virtually all SSH brute-forcing bots. One possible interpretation for this result is that there still exist so many insecure systems that, at least for non-targeted attacks, most attackers choose to proceed to other targets if their first few attempts do not provide them with access.

The numbers for Round 3 show that we successfully avoided the majority of brute-force attackers. Section 3.4 addresses how many of these logins are due to humans lured by leaked credentials and how many are due to exceptionally persistent or lucky brute-force attempts.

Round	Total	Per Day Average
Round 1	210,848	5,142.63
Round 2	1,383	32.16
Round 3 - Forums	86	N/A
Round 3 - Paste Sites	7	N/A

Table 1: Number of successful logins in each experiment.

Round	Successful Logins	Tunneling Sessions	% of Successful Logins
Round 1	210,848	139,746	66.28
Round 2	1,383	205	14.82
Round 3	93	16	17.20

Table 2: Number of tunneling sessions in each round.

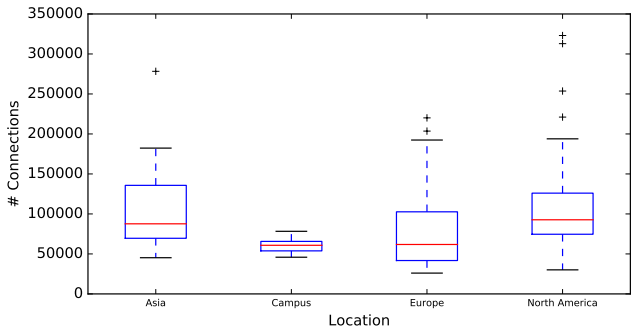
3.2 Tunneling

When attackers break into a machine, in addition to exfiltrating data and taking advantage of the machine's local resources, they can also use that machine as a stepping stone for conducting new attacks, e.g., more SSH brute-forcing, tunneling of traffic, and DoS attacks, while hiding their original IP address.

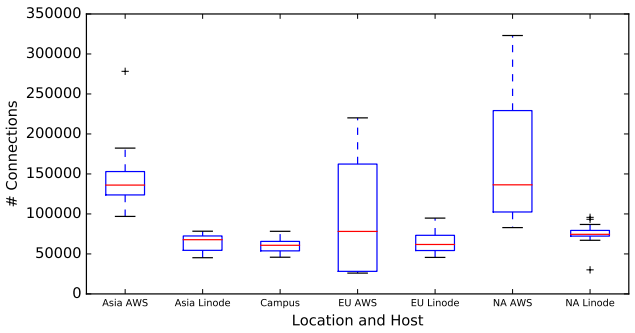
In this section we mostly focus on the tunneling activity from Round 1 because it had a much larger number of logins and a greater proportion of tunneling activity as shown in Table 2. These Round 1 results apply to the most general attackers who gained easy access to vulnerable machines.

When attackers used our honeypots as SOCKS proxies, cowrie logged the destination of the proxied traffic but did not actually forward the traffic to that destination (i.e. all connections fail before a handshake can be performed). Interestingly attackers continued to send requests despite the lack of appropriate responses. We found that over 65% of all successful logins were used as SSH tunnels. In addition we looked at attacker IP addresses who ran commands, tunneled, or both to get an idea of attackers' goals after logging in. 50.44% of unique attacker IPs who attempted to SSH tunnel did not run any commands. Meanwhile, 55.04% of unique IPs who ran commands also used the honeypots for tunneling. We argue that this is an important finding because it indicates that generic attackers value vulnerable servers primarily as proxies. If half of the attackers do not perform any commands once they break into a honeypot, this can affect the usefulness of deployed honeypots as ways of identifying attack trends. Contrastingly, we see that, for Rounds 2 and 3, less than 20% of attackers attempted to establish tunnels. This shows that by making it harder for an attacker to break into a machine, we filter-out those that are primarily interested in using honeypots as mere proxies. We discuss the implications of these findings in Section 4.

The location of our honeypots had a noticeable impact on the amount of tunneling activity. Figure 2a compares the number of sessions used for tunneling as a percentage of the total number of successful logins across the 7 locations/hosts. The Asia, Campus, and EU honeypots had similar proportions of tunneling sessions, but the NA honeypots had roughly 5% less. To account for the same attackers logging in multiple times we also looked at the percentage of successful login unique IPs that had tunneling activity. Figure 2b shows that the EU honeypots had tunneling from more unique

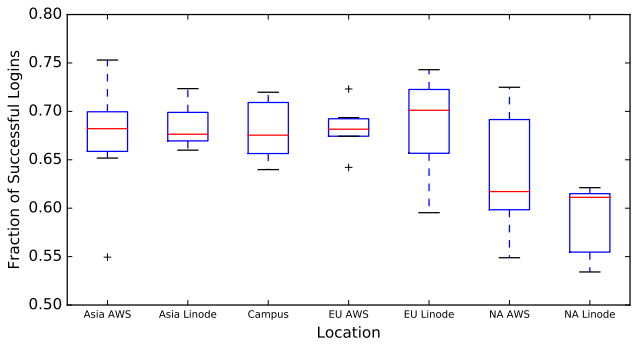


(a) Connections by location.

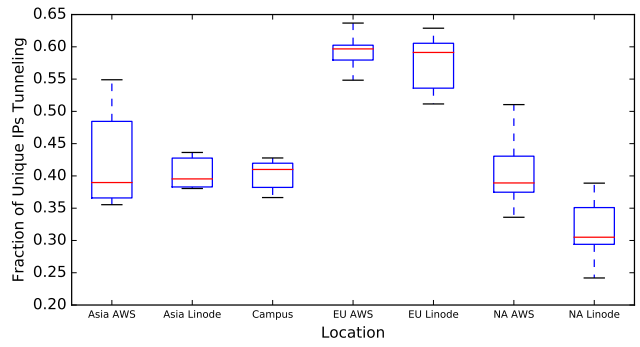


(b) Connections by location and host.

Figure 1: Number of connections per day by location and hosting provider for rounds 1 and 2.



(a) By session



(b) By unique attacker IPs

Figure 2: Fraction of successful logins used for SSH tunneling compared across locations and hosts.

attackers and confirmed that the NA Linode honeypots were used less for tunneling.

To obtain a rough approximation of the type of traffic that attackers were trying to tunnel through our honeypots, we analyzed the ports and IP addresses of their destinations. Port-wise, we found that 54% of forwarding traffic was standard web traffic (HTTP), 21% was encrypted web traffic (HTTPS), and another 21% was e-mail traffic (SMTP) demonstrating that most tunneling was done to disguise web traffic. In terms of destinations, we recorded over 158,000 unique forwarding destinations, but we found that a large amount of traffic was directed at specific targets. The top 4 destinations (according to amount of traffic sent) accounted for more than 11 million requests. Among the top 20 destinations we found various web hosting providers but also twelve SMTP servers (port 25) that belong to Yahoo. Because of the great focus of attackers on Yahoo, we hypothesize that this traffic may be related to the Yahoo data breaches announced shortly before and then again after Round 1 of our experiment [13, 24].

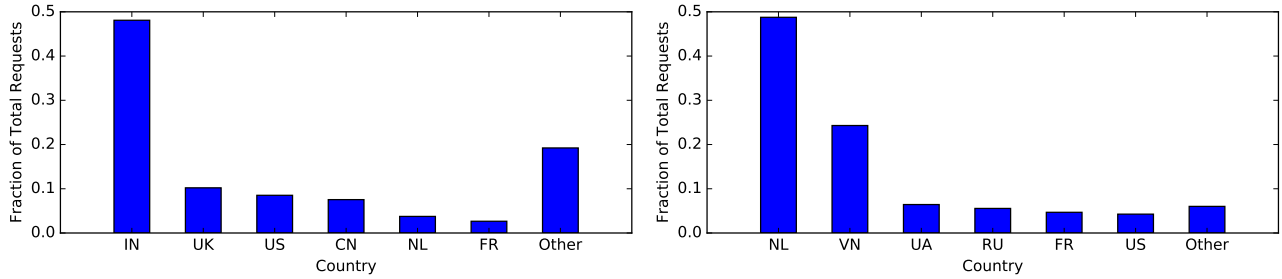
Next to the destinations that received the most traffic, we also discovered destinations that were common across a large number of attackers. The top destination (according to number of attacker IPs) was an IP owned by Kraft Foods Group that had requests from 1,331 unique IPs. The top 20 destinations also included several SMTP servers (ports 110, 143, 465, and 587) belonging to Yahoo, Orange, and Yandex, and a handful of servers in Ukraine and Bulgaria. The number of unique source IPs suggests that these requests were part of a coordinated attack against these destinations. We also found that many attackers utilized `ipinfo.io` or `httpbin.org`. These services provide attackers a simple tool to test the connection to the outside world and to verify the public-facing IP address of the compromised system. `ipinfo.io` also provides geolocation data which may be useful for assessing a system's value as a proxy.

Finally, by looking at the source location of attackers attempting to tunnel traffic, we found patterns that differ from attackers that ran commands. Figure 3 shows the top locations for attackers. We found that the Netherlands was the largest contributer accounting for nearly half of all tunneling sessions. Notably, there was no crossover between the top 4 countries in each group of attackers.

3.3 Brute-force and prior knowledge attackers

As described in Section 2.4, in Round 2 of our experiment we changed the setup so that attackers would have to brute-force the username and password to gain access to our honeypots. Since we used fairly common passwords we expected many attackers to successfully break in, but not in their first few attempts. In this section we investigate the sharing of credentials between IP addresses by looking at the number of attackers who managed to log in to our honeypots without making brute-force attempts (indicating that they were already aware of the correct password).

Here we define a *brute-force (BF) attacker* as a unique IP address that successfully logged in to a honeypot, but also had more than five failed login attempts on that same honeypot. We call IP addresses that successfully logged in with five or fewer failures, *prior knowledge (PK) attackers*. Allowing a small number of failed



(a) Attackers with commands. *(b) Tunneling attackers.*

Figure 3: Top locations of attackers that ran commands compared with attackers that tunneled traffic.

attempts leaves room for human error, such as mistypes. We chose five or less because it was the most conservative number of attempts that encompassed every Round 3 attacker in the prior knowledge group. Specifically, we noticed that attackers who logged in to our Round 3 honeypots, made up to five failed login attempts before successfully logging in. Given the difficulty of the passwords, we consider it extremely unlikely that this small number of failed attempts were from a BF attacker, hence we can conclude that five unsuccessful attempts is still realistic for an attacker who already knows the exact credentials.

Table 3 shows the breakdown of PK and BF attackers from the total number of successful logins. The primary quantity available in each cell of Table 3 is the number of unique IP addresses while the second quantity in parentheses is the total number of sessions from those IPs. We found that 40.6% of all successful attackers in Round 2 logged in without brute-forcing. This means that credentials must have been shared from those that did brute-force. We also found that the PK attackers were far more active with the credentials than the BF attackers. PK attackers had 75% more sessions per IP and 90.3% of PK attackers ran commands on the honeypots (PK+CMD), as opposed to only 4.7% of BF attackers (BF+CMD).

We propose two possible explanations about our attackers supported by these numbers. One is that attackers ran dedicated brute-forcing machines that searched IP address spaces, tried credentials until a successful login, then saved those credentials for a human who—from another machine—either logged in themselves, ran a non-interactive script, or initiated another bot to run commands. The other is that attackers brute-forced credentials and then either shared them with members of their hacking group or sold them to other attackers (as we pretended to do in Round 3).

As we will further explore in the next section, we have devised a three-step method for differentiating between human attackers and bot attackers. By applying our method to the attackers described in this section, we discovered that of the 40 human sessions found, 39 of them were from PK attackers and only 1 was from a BF attacker. This further supports the theory that attackers move in two passes, where dedicated bots brute-force credentials which are then passed on for manual exploration by human attackers.

3.4 Human attackers versus bots

In this section we propose a method to classify attackers as humans or bots and compare the number of humans across each round of

Round	Login IPs	PK IPs	BF IPs	PK+CMD	BF+CMD
Round 2	431 (1,383)	175 (754)	256 (629)	158 (542)	12 (19)
Forums	41 (86)	41 (86)	0 (0)	34 (54)	0 (0)
Paste Sites	7 (7)	7 (7)	0 (0)	6 (6)	0 (0)

Table 3: Breakdown of successful login IP addresses with 5 or less (PK) or more than 5 (BF) failed attempts per IP address. The values in parentheses are the number of sessions from those IPs.

experiments. We only consider interactive attackers, i.e., attackers who logged-in to one of our honeypots and then proceeded to "type" a series of commands. We do not consider attackers who use non-interactive sessions where one or more commands are sent together with an SSH login attempt, e.g., `ssh root@1.2.3.4 'bash -s' < attacker_script.sh`. Commands run in this way are still considered in Section 3.5 (where we explore the types of commands used by attackers), but we ignored them for the purpose of distinguishing humans and bots. Similarly, we also ignore attackers that only used the honeypot as an SSH tunnel (Section 3.2), since there was no interaction with the honeypot itself.

Cowrie saved all sessions as interactive/non-interactive TTY logs. For the interactive sessions, these logs contained timestamps for every keystroke entered on the honeypot. We used three methods to label a session as a human:

(1) The simplest and most effective was to check if the backspace or delete special characters were in the TTY log. It is very common for a human to make mistakes typing in a terminal and correct them as they type, whereas a bot would not make typos. Therefore if either of these characters are found we can be confident we have found a human attacker.
(2) By parsing Cowrie's TTY logs we were able to extract the time deltas between keystrokes, taking the max delta per session. The intuition is that humans type slowly and inconsistently while bots enter commands almost immediately. As such, human attackers will have a larger maximum delta between keystrokes. For each honeypot we took the set of max deltas and applied the median absolute deviation test to find outliers under the reasonable assumption that the majority of SSH attackers are bots (the alternative would be human attackers manually typing password after password hoping to guess the right one). Given this assumption, any outliers will be potential human attackers.
(3) Our final method was to apply a strict threshold for the max deltas. If the max delta was greater than the threshold then we

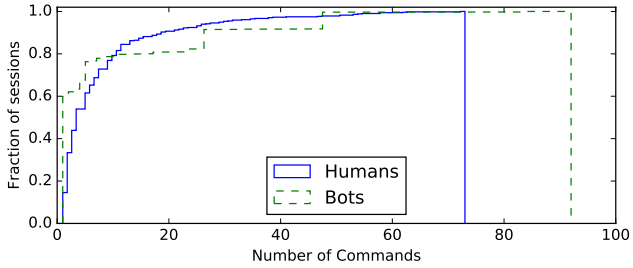


Figure 4: CDF of number of commands per session comparing humans to bots.

labeled the session as a human. The threshold was determined to be 0.1 seconds by manually inspecting histograms of deltas and examining sessions which were not marked as outliers, but were still outside the pack of presumable bots.

We considered these three methods complementary and labeled a session as belonging to a human if any one of these methods did so as well.

Since we had a smaller number of successful logins in Round 3 and we expected those attackers to be human, we manually inspected all TTY logs for Round 3 using a tool that replayed the session back in real time. We found that 42 out of 50 interactive sessions were made by humans. Using this set as the ground truth for humans we were able to test our algorithm and judge its accuracy finding that our algorithm was able to detect all 42 humans as humans and all 8 bots as bots. Based on this result we were confident applying our algorithm to Rounds 1 and 2 to classify the rest of the attackers as humans or bots. Even if there are corner cases that our algorithm does not capture, we argue that these would not result in enough misclassifications to affect the general findings of our analysis.

Table 4 shows the number of sessions labeled as humans for each round and the percentages they represent of interactive sessions, sessions with commands run, and all successful logins. Percentage of successful logins gave an idea of how many attackers were humans running commands, but percentage of sessions with commands may be the most fair comparison. This compared humans attempting to run commands to bots attempting to run commands (assuming all non-interactive sessions are bots) and ignored the many sessions only interested in SSH tunneling and brute-force bots that may do nothing but pass on credentials after logging in.

Based on our methods which were applied equally to each round we found an increasing percentage of humans in each experiment. This was expected in Round 3, but it was an interesting result for Round 2. Recall that the change from Round 1 to Round 2 was that we required attackers to brute-force the credentials. Therefore the increase in percentage of humans indicated that persistent brute-force bots were more likely to deliver credentials to a human than the less persistent bots in Round 1. In fact more than 7% of Round 2 sessions executing commands were from humans.

In Round 3, we found that 19% of successful attackers did nothing after logging in. We consider these to be attackers who found the credentials and were curious enough to test them, but did not continue to use the vulnerable system, perhaps because they were suspicious or because they postponed their investigation for some

Round	Classified As Humans	% of Interactive Sessions	% of Sessions with Commands	% of Successful Logins
Round 1	434	4.55	0.93	0.21
Round 2	40	31.50	7.13	2.89
Round 3	42	84.00	70.00	45.20

Table 4: Number of sessions classified as humans by our algorithm for Rounds 1 and 2 and by manual inspection for Round 3.

future time. Another 16% only used the honeypot as an SSH tunnel. The remaining portion of sessions were those that ran commands, and we found that humans made up 70% of this category. This showed that leaking credentials on the web attracts a significantly higher fraction of humans, but there will still be some bot activity which, because of the fact that parsing text and automatically extracting credentials from a generic forum is highly unlikely, we attribute to humans who chose to exploit these honeypots with scripts or use them as web proxies.

3.5 Categorizing attacker actions

In this section we investigate the commands run by attackers after they broke into our honeypots. We start by looking at number of commands run and then go deeper by categorizing the types of commands run in each session. We compare between sessions labeled as humans and bots, as well as between the 3 types of honeypots (*Vanilla, Random, Curated*). We do not compare across rounds because differences in round setup affected the way attackers logged in and whether they were mostly humans or bots, but on a per attacker basis it should not have impacted the way they interacted with the honeypots after logging in.

Figure 4 shows the number of commands per session as a cumulative distribution and compares between sessions labeled as humans and bots. We found that about 80% of both types had less than 10 commands. It was somewhat surprising to find that humans and bots ran a similar number of commands. Bots do not need a human typing each command which allows them to have more, but they are also more efficient. Each command is included to contribute to the bot's goal, whereas a human may execute more commands than they need as they explore the system, make decisions, or make typo commands which need to be retyped. What is strikingly different when comparing bots and humans is the smoothness of the CDF. This is most likely due to natural variability of humans which stands in sharp contrast with bots which will typically belong to specific families thereby running the same number of commands and producing vertical jumps in the cumulative distribution.

In order to understand what attackers did in their sessions, we defined seven categories of commands. For each session we labeled it with a category below if it contained at least one command that belonged to that category. Therefore, it was possible for a session to be labeled with anywhere from 0 to 6 categories.

- **Users:** Changing the password of the victim user account and/or creating a new user. These attackers attempt to ensure future access to the system by making sure there is an account to which they know the password.
- **Files:** Navigating directory tree, listing and opening files. These attackers interacted in some way with the file system of the honeypot. The files they looked at or used may have been their own.

- **System:** Investigating various system information such as, `top`, `free`, `uname -a`, `cat /proc/cpuinfo`, etc. These attackers tried to learn additional details about the system they had broken into, either to determine its value as a resource or to decide the best plan of attack.
- **Malware:** Downloading suspicious files from remote sources with `wget`, `ftp`, etc. These attackers attempted to fetch what was most likely malware from other machines.
- **Defenses:** Stopping defensive services such as firewalls. These attackers attempted to maximize the impact of their attack by first stripping away defenses.
- **Our Files:** Changing directory into the home folders of our fake users. These attackers show some interest in the *data* on the honeypot. We argue that the only reason an attacker would change directory to a user's home folder is if they intended to look at the files contained within. Instead of logging in and only following rote plans to download and execute malware, they show some interest in the value the victim machine might have for the information on it.

In Round 3 the attackers started in the home folder of the user whose credentials we leaked. This meant they did not need to change directory into the home folder of a user in order to see the files we created. To compensate for this, we manually inspected the TTY logs of the Round 3 attackers and labeled them as to whether or not they looked at our files. These sessions were added to the "Our Files" counts in Table 5. On the *Vanilla* honeypots where we did not generate any files, we still included any attempts to list files in a users' home directory since it conveyed the attackers' intent to investigate files if there were any.

Table 5 shows the percentage of sessions labeled with each category, split by human and bot attackers, and compared across the three types of honeypots. The first interesting result is that while human attackers were significantly more likely to change the password after logging in, it was uncommon across the board. This is in stark contrast to the findings of Nicomette et al. [20] where every attacker began by changing the password. Changing the password and/or adding a back-door account make it more obvious that an attack has occurred on a system in regular use. Our results suggest that attackers have adopted stealthier approaches in the last 10 years. This is likely due to increasing virtualization of web servers which makes it easier for an admin to recover a system after being locked out by an attacker. An attacker often has more to gain by maintaining their access to the system especially if they plan to add it to a botnet.

For bot attackers, "Files" and "Malware" were the largest categories. This showed that the majority of bots attempted to download and run malware, and only interacted with files related to their scripts. Some downloads appear to have failed due to connection problems with the remote hosts since their bots gave up after the attempt, but others continued to execute their downloaded files. We found that about 17% of bots did some sort of System investigation. While the value to a human attacker is clear, we were surprised to find so many bots in this category. It may be that the bots contained logic which allowed them to change attacks based on simple system information or that this information would be made available to the bot master for statistics, future exploitation, or botnet-renting purposes.

	Bots			Humans		
Category	**Vanilla**	**Random**	**Curated**	**Vanilla**	**Random**	**Curated**
Users	0.48	0.02	0.59	11.36	14.86	15.38
Files	70.82	72.73	70.91	59.09	65.71	67.79
System	16.33	16.12	17.29	72.73	60.57	66.35
Malware	74.24	74.31	72.50	31.06	35.43	35.10
Defenses	5.11	5.49	5.7	2.27	1.14	1.44
Our Files	0.00	0.00	0.01	10.61	8.57	14.42

Table 5: Percentage of sessions labeled with each category. Separated by human and bot sessions and by type of honeypot.

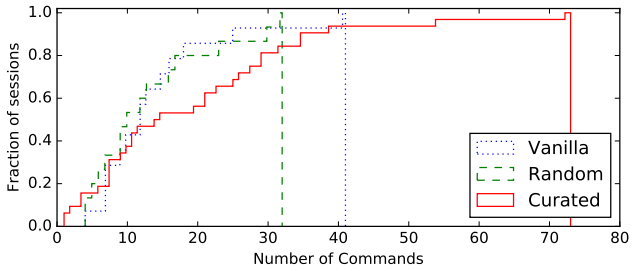


Figure 5: CDF of the number of commands executed by attackers who inspected a user's home directory. Each line represents sessions on one of the three honeypot types.

The human side of the table tells a different story. One of the biggest differences was the "System" category which was about 4 times larger for humans than it was for bots. This makes sense as human attackers make risk-to-value assessments during the attack so it would be useful for them to learn about the compromised system and decide on the best use for it. Human attackers also had a much smaller percentage of "Malware" commands and roughly 10% of them investigated at least one home folder of a user on the system. This demonstrates that humans were more focused on the information on the honeypot. Whether they were satisfying curiosity or planning on stealing information, their intentions were in stark contrast to those of the bots. Looking at the "Our Files" category, we found that of the 61 total sessions, 59 of them were classified as humans. Given the much larger number of bots and the fact that those two sessions could have been misclassified humans, this showed that essentially no bots have any interest in users' files present on the machine.

After categorizing commands and finding attackers interested in the files we placed on the honeypots, we revisited the number of commands run per session. The "Our Files" category allowed us to quantify whether attackers were interested in the files on the system. However, it did not provide a satisfying answer as to the impact of the different types of honeypots on the attackers' behavior after they looked at our files. To quantify the impact of the files we created, we examined the number of commands per session. The intuition is that an attacker who looked at our files and found them convincing may spend more time investigating. In Figure 5 we look at number of commands run in sessions which were labeled with "Our Files." This is a somewhat limited set of 61 sessions from Table 5, but it does cover all three honeypot types. These were the attackers that cared about files on the system and the total number of commands they ran gave us a metric to judge

the effect of their exploration on their behavior. We found that *Vanilla* and *Random* honeypots had similar numbers of commands, but the curated honeypots attracted more commands from attackers. This is an important finding because it allows us to conclude that, in order to entice human attackers to interact for a longer time with a honeypot, one must produce file populations that mimic real systems.

3.6 Attacker case studies

In this section we examine specific examples of typical human attacker sessions. The commands that were executed are shown in Appendix A. Each session started in the home folder of the compromised account and began by listing files. Session `Human_A` then immediately tried to install a visual file manager, midnight commander (`mc`). Each attacker continued to explore specific files/folders which gives us an idea of the names/locations which were most enticing. `Human_B` explored Videos, Downloads, Pictures, and opened the file `Downloads/20161216_work.txt`. `Human_A` explored Documents and was then drawn to a folder named IMPORTANT. `Human_C` also looked in Documents where they tried to open `faq-congo.pdf` and `index.html`. These three attackers were all on *Curated* honeypots and showed that Documents and Downloads were particularly enticing. `Human_D` was on a *Random* honeypot and investigated the randomly generated directory `situation_baghdad_047581`. They then tried copying it to the root directory with `scp` and while they discovered `scp` would not work, the attempt suggests an interest in exfiltrating the entire directory. While three of these attackers had a few other exploratory commands, it is clear that their main interest was in users' files. These types of attackers are the most dangerous to systems with sensitive information, and also the most vulnerable to tripwire/decoy file based intrusion detection systems [7, 31, 34].

We also examined sessions from bot attackers which due to space limitations we cannot fully describe (the commands of popular sessions are available in the Appendix). One key observation from comparing these sessions to the human sessions is that for bots, population and believability of the file system is irrelevant. Their actions are generic (e.g. downloading files and inspecting the system specifications) and unlike the human sessions, they do not include back and forth interaction with the compromised system.

4 DISCUSSION

In this section we present a series of key takeaways distilled from our experiments in Section 3 and then briefly describe the limitations of our study and potential directions for future work.

Key takeaways

- **Location and host matters.** Honeypots hosted in AWS in NA and Europe received more connections than the rest (Section 3.1). As such, if someone is operating on a constrained budget and wants to maximize the number of brute-forcing IP addresses collected, AWS appears to be the infrastructure that will facilitate this.
- **Most attackers are opportunistic.** The majority of brute-force attacks are very opportunistic to the extent that unless the password is an obvious one (allowing them to break-in in their first few attempts) they will move to the next target (Section 3.1). Our honeypots that had a password that was not in the top 20 most common passwords of users were broken into two orders of magnitude less than the ones that indiscriminately allowed attackers to login as long as they were willing to try three different passwords.
- **Difficulty of break-in correlates with human activity.** In Section 3.4 we showed that the fraction of human attackers was much larger in Round 2 than in Round 1 with evidence of credential sharing between the IP address that brute-forced a password and the human attacker who later came to inspect the compromised machine. Together with our earlier takeaway point, this suggests that when deploying honeypots one must strike a balance between choosing the greatest possible number of attackers and choosing to attract human attackers.
- **Most attackers are bots.** When deploying honeypots in generic IP address space, one must expect that the majority of break-ins will originate from bots which will not necessarily be followed by a human attacker (Section 3.3). This indicates that if one is interested in studying the attack patterns of humans (what they do once they break into a machine) they will have to deploy a large number of honeypots and use filters to remove automated bot-related activity.
- **Half of the attackers only break in to honeypots to use them as proxies.** Even though this could be seen as a negative, this would in fact allow defenders to monitor the web behavior of attackers, without the need of ISP collaboration, and therefore identify web attacks (e.g. password brute-forcing on popular websites) as they are happening. Note that, for ethical reasons, in our deployment we did not actually forward the proxied requests of attackers to their final destinations.
- **Hacking forums and paste sites can be used to learn more about human attackers.** Through our experiment with "leaking" honeypot credentials to hacking websites and paste-sites, we discovered that attackers indeed read our posts and proceeded to log in to our honeypots. Moreover, through manual labeling, we confirmed our intuition that the majority of these attackers interacting with the honeypots were human. Since researchers are already monitoring underground forums for a variety of purposes, e.g., identifying real break-ins [17], our results indicate that such forums could be used as one more source for gathering threat intelligence from real attackers and not scripted bots.
- **Humans care about user files, bots do not.** In Section 3.5 we witnessed that bots exhibit near zero interest in what files are present on user machines. Contrastingly, as much as 14% of human attackers inspected the file system and interacted, to a certain extent, with user files. These findings have implications on intrusion detection systems, specifically those that rely on booby-trapped files, such as, tripwires and decoys [7, 31, 34]. Our findings show that these systems would not be able to detect bot break-ins, suggesting that there need to be other monitoring systems in place that can account for malicious, bot-specific, behavior.
- **Curated file systems entice human attackers to interact more with them.** Spending time to create file systems with realistic looking files and directory structures pays dividends when it comes to enticing attackers to interact with them (Section 3.5). This interaction translates to human attackers spending more

time on a honeypot which is desirable, both from a research perspective, as well as giving more time for defenders to isolate that attacker from the rest of their network.

Limitations and Future Work

Cowrie, like any other simulating medium-interaction honeypot, is likely to be fingerprintable by dedicated attackers. It is therefore possible that our study was not able to capture a fraction of attackers who, upon the discovery of cowrie, logged out. While this could introduce a source of bias in our results, it is important to note that any bias would be present across all deployments/rounds/locations. As such, we argue that despite cowrie's potential limitations, we are still able to compare the activity of different attackers when exposed to different underlying environments. In future work, we plan to repeat our experiments with high-interaction honeypots, utilizing tools such as Bifrozt [15] or Dockpot [1], and compare our findings with the ones described in this paper.

Similarly, our results can generalize to the general public to the extent that honeypots located on public clouds and university campuses attract similar types of abuse as those located in companies and organizations. We are certain that, in addition to generic bot attackers, large companies experience much more targeted attacks that may deviate from the ones recorded by our honeypots. At the same time, we expect that these targeted attacks are, primarily, human driven (as opposed to bot-driven) and thus our observations about human attackers and how they interact with machines that they have compromised may, in fact, be similar to these targeted attacks. One of the reasons why we are open-sourcing our code and data is to allow researchers working in the industry to repeat our experiments and report their findings.

5 RELATED WORK

The area of intrusion detection in general and honeypots in particular, has a rich history of books, papers, deployments, and lessons learned from organizations which deploy individual honeypots as well as networks of honeypots.

The most popular project related to honeypots and networks of honeypots (honeynets) is the Honeynet Project which was founded in 1999 [25]. In [32], Spitzner described the implications of running networks of honeypots (honeynets) and described the organization's plan for the future. The Honeynet Project also hosts a series of whitepapers on attacker activity [26] that researchers were able to uncover because of honeypots as well as a multitude of deployable software related to honeypots. Spitzner [33] as well as Niels and Holz [27] provide complete treatments of honeypot types, deployment strategies, the detectability of honeypots and case-studies of forensic analyses that honeypots enable.

In 2004, Raynal et al [29] present a forensic analysis of 24 hours worth of network traffic to and from their honeypot and analyze the corresponding host-level activity. In the same year, Dacier et al. describe their analysis of the data collected by a virtualized network of honeypots and present, among others, the geographical information of attackers, their operating systems, and the ports that they target [11]. In 2006, Alata et al. [2] described their experience with the deployment of a single high-interaction honeypots for 131 days receiving connecting attempts from a total of 480 different IP addresses. A year later, Ramsbrock et al. [28] monitored SSH break-ins to four high-interaction honeypots for a month using accounts with easy-to-guess passwords. They presented the number of successful login attempts, commonly attempted credentials and the types of activity that attackers performed when they broke in. Nicomette et al. recount their experience running a high-interaction honeypot for more than a year [20]. Among others, the authors discovered that attackers are relatively unsophisticated (types of commands and errors incurred) and that they can identify types of attackers by the password dictionaries they use to brute-force their way in their honeypot. Koniaris et al. describe the findings of deploying the Kippo honeypot, on a virtual private server for four months [16]. Canali and Balzarotti investigated the post-exploitation actions of attackers and their goals when targeting vulnerable web applications [9], identifying common patterns of abuse.

In recent years, in addition to the traditional server honeypots (the topic of this paper), researchers have proposed honeypots for new and emerging technologies, such as, VoIP [14], instant messaging [5], mobile telephony [6], smartphones [18], SCADA systems [35] and even honeypot-like listings to find scammers on Craigslist [23]. Recently, Farinholt et al. presented the results of a study where they observed the activities of amateur operators of DarkComet, a popular malicious remote administration tool (RAT) [12]. Among others, the authors noted that RAT operators interacted more with the virtual machines that had the most *file system depth*. Our findings support this observation for the human portion of attackers breaking into our SSH honeypots suggesting a generic trait of human attackers who tend to exhibit more "curiosity" about the systems they compromise than bots.

The main difference between most of the aforementioned work and ours is that, in addition to performing an observatory study (where we set up honeypots and describe our findings) we change experimental parameters about our population of honeypots (i.e., their location, difficulty of break-in, and file population) and observe how these changes affect the actions of attackers who break into our honeypots. Moreover, inspired by the work of Onaolapo et al. [21], we actively advertise a set of honeypots to underground forums and paste-sites and describe how these attackers are different than the ones who autonomously break into our honeypots.

6 CONCLUSION

In this paper, we investigated how certain underlying properties of a system affect the actions of attackers who break into that system. To that extent, we conducted a four-month study where, through the deployment of 102 honeypots, we quantified whether the difficulty of break-in, the location of a system, and the population of files in a filesystem had an effect on attackers. Our results showed that the majority of attackers are bots which are interested in tunneling traffic and making our honeypots part of botnets and are not affected, with the exception of location, by other underlying parameters. At the same time, we found that harder-to-compromise systems have a higher probability of being later inspected by human attackers who are, in fact, interested in user files and tend to spend more time examining the system when the honeypot presents a realistic set of files and directory structures. Our findings can be used to guide future honeypot deployments and motivate the need for layered

intrusion detection systems that can account for bot-specific, as well as human-specific, activity on compromised hosts.

Availability: To rekindle interest in honeypot research and to allow other researchers to identify patterns that we may have missed, we will be open-sourcing all of our code and data.

Acknowledgments: We thank the reviewers for their valuable feedback. This research was supported by Office of Naval Research (ONR) under grant N00014-16-1-2264 and the National Science Foundation under grants CNS-1617902 and CNS-1617593. Some of our experiments were conducted with equipment purchased through NSF CISE Research Infrastructure Grant No. 1405641. We also thank Linode for providing us with virtual machines that made our large-scale experiments possible.

REFERENCES

[1] Ahmad Aabed. 2017. Dockpot. (2017). https://github.com/eg-cert/dockpot
[2] Eric Alata, Vincent Nicomette, Mohamed Kaâniche, Marc Dacier, and Matthieu Herrb. 2006. Lessons learned from the deployment of a high-interaction honeypot. In *Dependable Computing Conference, 2006. EDCC'06. Sixth European*. IEEE, 39–46.
[3] Amazon. 2017. AWS IP Address Ranges. (2017). http://docs.aws.amazon.com/general/latest/gr/aws-ip-ranges.html
[4] Ansible. 2017. Ansible is Simple IT Automation. https://www.ansible.com/. (2017).
[5] Spiros Antonatos, Iasonas Polakis, Thanasis Petsas, and Evangelos P Markatos. 2010. A systematic characterization of IM threats using honeypots. In *ISOC Network and Distributed System Security Symposium (NDSS)*.
[6] Marco Balduzzi, Payas Gupta, Lion Gu, Debin Gao, and Mustaque Ahamad. 2016. Mobipot: Understanding mobile telephony threats with honeycards. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*. ACM, 723–734.
[7] Brian M Bowen, Shlomo Hershkop, Angelos D Keromytis, and Salvatore J Stolfo. 2009. Baiting inside attackers using decoy documents. In *International Conference on Security and Privacy in Communication Systems*. Springer, 51–70.
[8] Ron Bowes. 2017. Leaked Passwords. (2017). https://wiki.skullsecurity.org/index.php?title=Passwords
[9] Davide Canali and Davide Balzarotti. 2013. Behind the scenes of online attacks: an analysis of exploitation behaviors on the web. In *20th Annual Network & Distributed System Security Symposium (NDSS 2013)*. n–a.
[10] Graham Cluley. 2016. These 60 dumb passwords can hijack over 500,000 IoT devices into the Mirai botnet. https://www.grahamcluley.com/mirai-botnet-password/. (2016).
[11] Marc Dacier, Fabien Pouget, and Hervé Debar. 2004. Honeypots: A Practical means to validate malicious fault assumptions. In *Dependable Computing, 2004. Proceedings. 10th IEEE Pacific Rim International Symposium on*. IEEE, 383–388.
[12] Brown Farinholt, Mohammad Rezaeirad, Paul Pearce, Hitesh Dharmdasani, Haikuo Yin, Stevens Le Blond, Damon McCoy, and Kirill Levchenko. 2017. To Catch a Ratter: Monitoring the Behavior of Amateur DarkComet RAT Operators in the Wild. In *38th IEEE Symposium on Security and Privacy (IEEE S&P)*.
[13] Vindu Goel and Nicole Perlroth. 2016. Yahoo Says 1 Billion User Accounts Were Hacked. (2016). https://www.nytimes.com/2016/12/14/technology/yahoo-hack.html
[14] Payas Gupta, Bharath Srinivasan, Vijay Balasubramaniyan, and Mustaque Ahamad. 2015. Phoneypot: Data-driven Understanding of Telephony Threats. In *NDSS*.
[15] Are Hansen. 2014. Bifrozt - A high interaction honeypot solution for Linux based systems. (2014). https://www.honeynet.org/node/1191
[16] Ioannis Koniaris, Georgios Papadimitriou, and Petros Nicopolitidis. 2013. Analysis and visualization of SSH attacks using honeypots. In *EUROCON, 2013 IEEE*. IEEE, 65–72.
[17] Brian Krebs. 2013. Cards Stolen in Target Breach Flood Underground Markets. http://krebsonsecurity.com/2013/12/cards-stolen-in-target-breach-flood-underground-markets. (2013).
[18] Steffen Liebergeld, Matthias Lange, and Collin Mulliner. 2013. Nomadic Honeypots: A Novel Concept for Smartphone Honeypots. In *Workshop on Mobile Security Technologies (MoST)*. San Francisco, CA.
[19] Lily Hay Newman. 2016. The Botnet That Broke the Internet Isn't Going Away. https://www.wired.com/2016/12/botnet-broke-internet-isnt-going-away/. (2016).
[20] Vincent Nicomette, Mohamed Kaâniche, Eric Alata, and Matthieu Herrb. 2011. Set-up and deployment of a high-interaction honeypot: experiment and lessons learned. *Journal in computer virology* 7, 2 (2011), 143–157.
[21] Jeremiah Onaolapo, Enrico Mariconti, and Gianluca Stringhini. 2016. What Happens After You Are Pwnd: Understanding the Use of Leaked Webmail Credentials in the Wild. In *ACM SIGCOMM Internet Measurement Conference*, Vol. 2016. Association for Computing Machinery (ACM).
[22] Michel Oosterhof. 2016. Cowrie SSH/Telnet Honeypot. https://github.com/micheloosterhof/cowrie. (2016).
[23] Youngsam Park, Jackie Jones, Damon McCoy, Elaine Shi, and Markus Jakobsson. 2014. Scambaiter: Understanding targeted Nigerian scams on craigslist. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*.
[24] Nicole Perlroth. 2016. Yahoo Says Hackers Stole Data on 500 Million Users in 2014. (2016). https://www.nytimes.com/2016/09/23/technology/yahoo-hackers.html
[25] The Honeynet Project. 2017. About The Honeynet Project. https://www.honeynet.org/about. (2017).
[26] The Honeynet Project. 2017. Papers. https://www.honeynet.org/papers. (2017).
[27] Niels Provos and Thorsten Holz. 2007. *Virtual honeypots: from botnet tracking to intrusion detection*. Pearson Education.
[28] Daniel Ramsbrock, Robin Berthier, and Michel Cukier. 2007. Profiling attacker behavior following SSH compromises. In *Dependable Systems and Networks, 2007. DSN'07. 37th Annual IEEE/IFIP International Conference on*. IEEE, 119–124.
[29] Frederic Raynal, Yann Berthier, Philippe Biondi, and Danielle Kaminsky. 2004. Honeypot forensics. In *Information Assurance Workshop*.
[30] Christian Rich. 2016. Natural filename generator. https://github.com/ChristianRich/natural-filename-generator. (2016).
[31] Malek Ben Salem and Salvatore J Stolfo. 2011. Decoy document deployment for effective masquerade attack detection. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 35–54.
[32] Lance Spitzner. 2003. The honeynet project: Trapping the hackers. *IEEE Security & Privacy* 99, 2 (2003), 15–23.
[33] Lance Spitzner. 2003. *Honeypots: tracking hackers*. Vol. 1. Addison-Wesley Reading.
[34] Jonathan Voris, Jill Jermyn, Nathaniel Boggs, and Salvatore Stolfo. 2015. Fox in the trap: thwarting masqueraders via automated decoy document deployment. In *Proceedings of the Eighth European Workshop on System Security*.
[35] Susan Marie Wade. 2011. SCADA Honeynets: The attractiveness of honeypots as critical infrastructure security tools for the detection and analysis of advanced threats. (2011).

A ATTACKER SESSIONS

Human_A
ls
mc
sudo apt-get install mc
apt-get install mc
mc
htop
top
ls
cd Do
cd /home/dickie/Documents
ls
cd /home/dickie/Documents/IMPORTANT
la
ls
cd /home/dickie/Documents
ls
ping www.terra.es
ifconfig
sudo ifcofnig
sudo ifconfig
exit

Human_B

```
ls
ls /home/alexis/Videos
ls /home/alexis/Downloads
less Downloads/20161216_work.txt
cat /home/alexis/Downloads/20161216_work.txt
ls /home/alexis/Downloads
ls
ls /home/alexis/Pictures
exit
```

Human_C

```
ls
ls
cd Do
cd /home/debby/Documents
ls
vi /home/debby/Documents/faq-congo.pdf
free
free -h
ping www.terra.es
ping www.elmundo.es
wget
wget www.terra.es
ls
vi /home/debby/Documents/index.html
sudo vi /home/debby/Documents/index.html
vi /home/debby/Documents/index.html
nano
ed
apt-get
sudo apt-get vi
apt-get vi
```

Human_D

```
ls
cd /home/ted/situation_baghdad_047581
ls
scp
scp /situation
cd /home/ted
scp /situation_baghdad_047581/ /
sudo apt-get install nbtscan
apt-get install nbtscan
nbtscan 45.118.133.235
dc
stop
shutdown
```

Bot session repeated 1,457 times.
Downloading and running executables.

```
wget http://212.92.127.196/.1 -O /tmp/..1
wget1 http://212.92.127.196/.1 -O /tmp/..1
chmod +x /tmp/..1
/tmp/..1 &
wget http://212.92.127.196/.2 -O /tmp/..2
wget1 http://212.92.127.196/.2 -O /tmp/..2
chmod +x /tmp/..2
/tmp/..2 &
wget http://212.92.127.196/.3 -O /tmp/..3
wget1 http://212.92.127.196/.3 -O /tmp/..3
chmod +x /tmp/..3
/tmp/..3 &
```

⋮

And so on up to file `/tmp/..12`

Bot session repeated 1,261 times.
Covers tracks and gathers information.

```
unset HISTORY HISTFILE HISTSAVE HISTZONE
      HISTORY HISTLOG WATCH
history -n
export HISTFILE=/dev/null
export HISTSIZE=0
export HISTFILESIZE=0
rm -rf /var/log/wtmp
rm -rf /var/log/lastlog
rm -rf /var/log/secure
rm -rf /var/log/xferlog
rm -rf /var/log/messages
rm -rf /var/run/utmp
touch /var/run/utmp
touch /var/log/messages
touch /var/log/wtmp
touch /var/log/messages
touch /var/log/xferlog
touch /var/log/secure
touch /var/log/lastlog
rm -rf /var/log/maillog
touch /var/log/maillog
rm -rf /root/.bash_history
touch /root/.bash_history
history -r  uname
free -m
ps -x
cat /proc/cpuinfo
```