Honeypots in the Cloud

Stephen Brown, Rebecca Lam, Shishir Prasad, Sivasubramanian Ramasubramanian, and Josh Slauson

University of Wisconsin - Madison

December 19, 2012

Abstract

Honeypots are systems used to trap, monitor, and identify erroneous requests within a network. For this project we conducted a study using honeypots within various cloud computing platforms (such as Amazon EC2, Windows Azure etc.) with the objective of learning more about what kind of packets they receive. We used various honeypots such as Dionaea, Kippo, and Amun on our cloud instances and gathered data about where attacks came from, what kinds of attacks were made, and differences among cloud instances. We discovered that most attack traffic comes from the US and China and that most attacks are on SSH and HTTP. We also found that for the most part, attack traffic among the clouds was quite similar. We also identified Dionaea and Kippo as the honeypots which are most effective in the cloud setting.

1 Introduction

With the rise of the internet, there has been a growing amount of traffic used for nefarious purposes. Thus, there is an growing need for detection and defense against the malicious traffic existing on the net. One tool used in network security is the honeypot. A honeypot is a decoy system used to attract and detect unauthorized or malicious traffic on the network. Honeypots are typically used for gathering information about attackers and gaining insight into their attack methodology.

1.1 Motivation

Cloud computing is an emerging market with different types of services such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). The inherent structure of the cloud presents security concerns unique to cloud networks. Users are allocated virtual machine instances that share physical resources with other users; this presents threats to each client's privacy and the integrity of the system as a whole. The clouds also serve as enticing targets for attackers, as they represent single locations where tousands of potential targets can be found. Thus, with its nascent popularity, cloud security is important both for companies that provide the service and those who rely on those services.

1.2 Contributions

In this paper we present a study of honeypots in the cloud setting with with an emphasis on characterizing attack traffic across multiple clouds. We focus mainly on identifying the following:

1. Where do attacks come from?
2. What kind of attacks are made?
3. Are there differences across different cloud providers?

The remainder of this article is organized as follows. Section 2 is a summary of related work. Section 3 discusses background on honeypots, and Section 4 discusses our experimental methodology and setup. In Section 5 we present our results, and in Section 7 we summarize our findings.

2 Related Work

Honeypots have been used in many research papers to profile and detect unauthorized traffic. A vast majority of them involve honeypots in a non-cloud setting used for a variety of purposes such as detecting spam and trapping database attacks. A number of papers [3][4] use honeypots to monitor botnet activity and use this information to detect and disrupt botnets. Alata et.al perform a study using a high-interaction honeypot to examine attacks on SSH [1]. Similarly, [6] examines attack traffic using both low and high interaction honeypots within a university network.

There have also been some papers that mention honeypots in a cloud setting. [2] suggests that cloud providers should provide honeypots as a profit-driven service to further their own security, while [7] describes the importance of honeypots as an emerging security tool for the cloud.

3 Honeypots

As mentioned before, honeypots are systems that emulate vulnerabilities with the purpose of trapping and examining attacker traffic. There are multiple types of honeypots: low interaction, medium interaction, high interaction, and pure. Low interaction honeypots simulate services and passively log connections. Medium interaction honeypots also simulate services, but they respond to the attacker. High interaction honeypots emulate an entire system, and pure honeypots are full-fledged machines in a network. One advantage of a low interaction honeypot over a high interaction one is that it is easier to deploy and maintain. However, they are more easily detected. In contrast, high interaction honeypots are more difficult to detect but are harder to maintain.[8]

3.1 Deployed Honeypots

1. **Dionaea** is a low interaction honeypot that detects automated malware by emulating different protocols such as SMD, HTTP, FTP, TFTP, MSSQL, MySQL, and SIP. It emulates a vulnerable Windows 2000 system.

2. **Kippo** is a medium interaction SSH honeypot that emulates the shell. It simulates common linux commands and a fake file system. Commands issued by attackers are logged and can be replayed.

3. **Amun** is a low interaction honeypot aimed at capturing autonomous spreading malware. It works by emulating a number of vulnerability modules,monitoring ports, and logging shellcode and downloads away for later analysis. Amun can also be extended with custom XML modules. [5]

4. **Artillery** is a low interaction honeypot that listens on common ports and detects connection attempts. When any connection is detected, the corresponding IP address is blacklisted via an iptables rule. It can also detect SSH brute force attacks and derail them by blacklisting the IP address. File monitoring is another service it provides; it can be configured to monitor certain directories and send an alert email if a change is detected.

5. **Glastopf** is a low interaction honeypot that emulates webserver vulnerabilities such as SQL injection and file inclusion. It works by providing a dynamic attack surface that can participate in multi-stage attacks. It fools attackers into thinking that their attacks were successful and captures the malware that is ultimately transmitted.

3.2 Other Honeypots

1. **Honeyd** is a very popular honeypot in off-cloud settings, and there are numerous studies attesting to its effectiveness. However, its dependence on DHCP prohibited us from including it in our study.

2. **Artemisa** is a VoIP honeypot which emulates a vulnerable VoIP client. It is no longer maintained, however, and we were unable to fully install it on our instances.

3. **HiHat** is an analysis tool which can transform any PHP application into a high-interaction honeypot. We launched several such instances, but none of them produced any meaningful traffic.

4. **Honeybot** is a Windows-based honeypot. While we were able to launch several instances of Honeybot, they produced very few results.

4 Methodology

4.1 Cloud providers

In this study, we focus mainly on Amazon EC2 and Windows Azure, although we did also set up honeypots in IBM Smartcloud and ElasticHosts. Table 1 summarizes the features of the different clouds, as well as the locations of our deployed instances.

Cloud Provider	Operating System	Service	Instances	Datacenters
Amazon EC2	Ubuntu 12.04 LTS	IaaS	22	North Virginia, Tokyo, Singapore, Ireland, North California, North Oregon, Sydney, Sao Paulo
Windows Azure	Ubuntu 12.04 LTS	IaaS	14	East US, East Asia, Southeast Asia, West Europe, West US
IBM Smartcloud	Redhat Enterprise Linux 6.3	IaaS	5	Canada, Germany, Japan, West US, Singapore
ElasticHosts	Ubuntu 12.04 LTS	IaaS	1	Los Angeles,USA

Table 1: A summary of cloud instances

4.1.1 p0f Fingerprinting Tool

p0f is a versatile passive OS fingerprinting tool. p0f can identify the system of machines that connect to your box, machines you connect to, and even machines that merely go through or near your box (even if the device is behind a packet firewall). We used p0f logs to extract the following data about the attacker:

1. Attacker's IP address
2. Operating System used by attacker to launch attacks e.g. Linux 2.6, Windows XP etc.
3. Port/Service targetted by attacker e.g. SSH, HTTP etc.
4. Attacker's mode of connection to internet e.g. ethernet/modem, DSL etc.
5. Number of network hops between attacker and honeypot instance
6. Uptime of attacker's machine
7. Time of attack

4.1.2 Analyzer Setup

To analyze the data generated across various honeypot instances, the following infrastructure was designed:

1. **Backend infrastructure**

   The backend infrastructure collects the data from various honeypot instances across the globe, processes it and persists it in a local database for subsequent analysis. It is written completely in Python. The main modules involved are:

(a) **Extractor** This module extracts the various log files like p0f log, honeypot-specific log files and honeypot database files from various honeypot instances. It is triggered daily once by a cron job and uses scp for securely copying the data from remote honeypot instances to the local machine.

(b) **Processor** This module processes the log and database files and extracts information relevant to analysing honeypot data. The main sub-modules comprising this data cleanup phase are:

i. **Converter** The local datastore is a MySQL database. But many honeypots store their data in sqlite3 databases. These has to be programatically converted to MySQL format to allow data integration.

ii. **Parser** The p0f log files have to be parsed to extract various parameters like attacker IP address, OS details, distance from honeypot instances etc.

iii. **Enricher** The extracted data is further enriched in this phase. For example, for each IP address found in the p0f logs, the country of origin is determined from a local IP address to country lookup database. Also, the md5 hashes of downloaded malwares are analysed using VirusTotal public API to determine the identity of the malware.

(c) **DataLoader** Once the raw data in the log and database files has been cleaned, processed, and enriched, relevant information is persisted in a local MySQL database to facilitate data analytics.

2. **Frontend infrastructure**

The frontend infrastructure analyses the data collected by the backend infrastructure, shows visualisations via charts and infers trends and attack patterns. It was designed using PhP and uses the Highcharts charting library extnesively for the visualization of our attack and attacker profile data.

5 Results

5.1 Attacker Profile

5.1.1 Geographical Location

The top countries from which attacks originated were essentially the same between EC2 and Azure. As can be seen in Figure 1 both clouds saw the most attacks from China and the United States, followed by Russia,

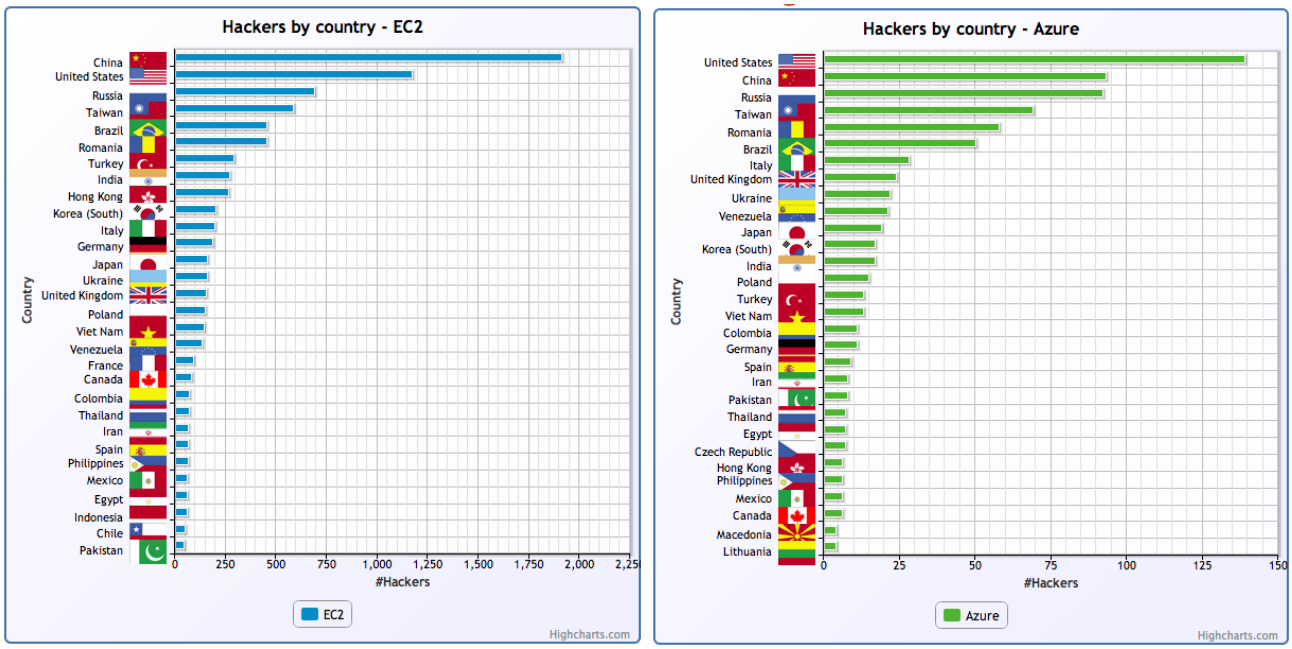

Figure 1: Attacks by Country

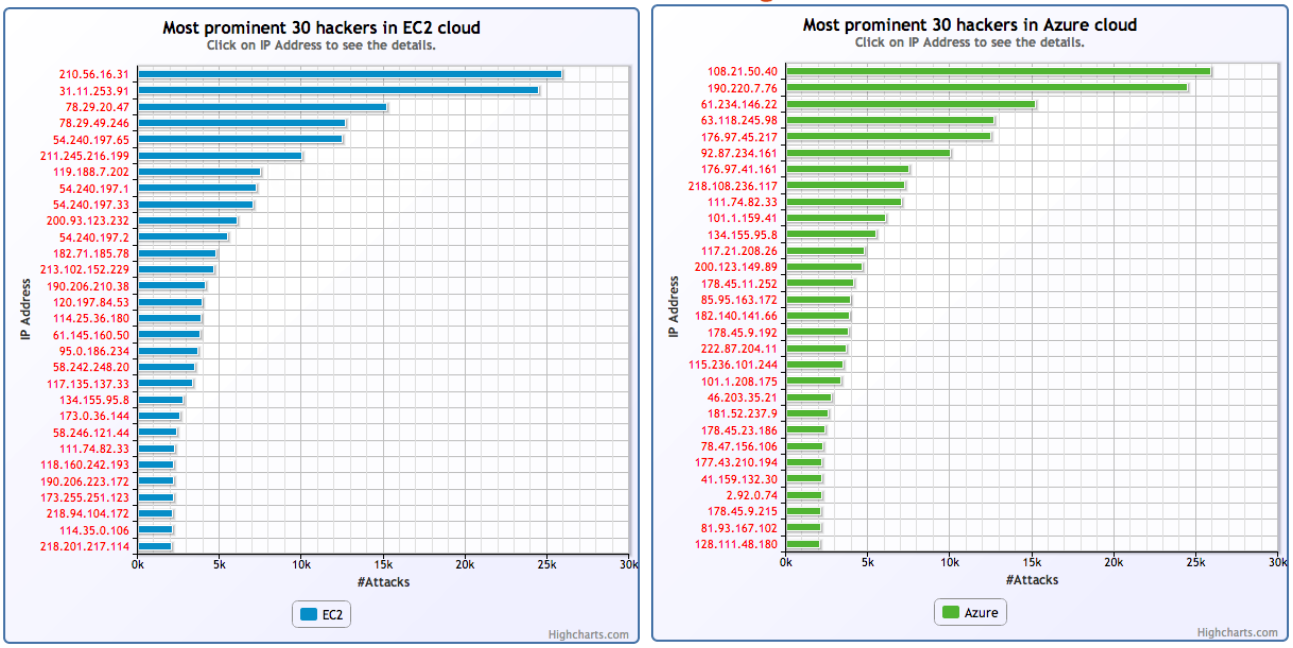

Figure 2: Attacker IP Addresses

Taiwan, Brazil, and Romania. Additionally, our instances were able to isolate a handful of IP Addresses that were responsible for thousands of connection attempts, as can be seen in Figure 2.

5.1.2 SSH Credentials

Both clouds saw a large number of automated SSH brute force attacks. The most common username was "root". Other usernames such as "sa" and "mysql" were attempted, but to a far lesser extent. As for passwords, attackers most commonly tried the empty sting (""), "123456", and "password." Attackers also

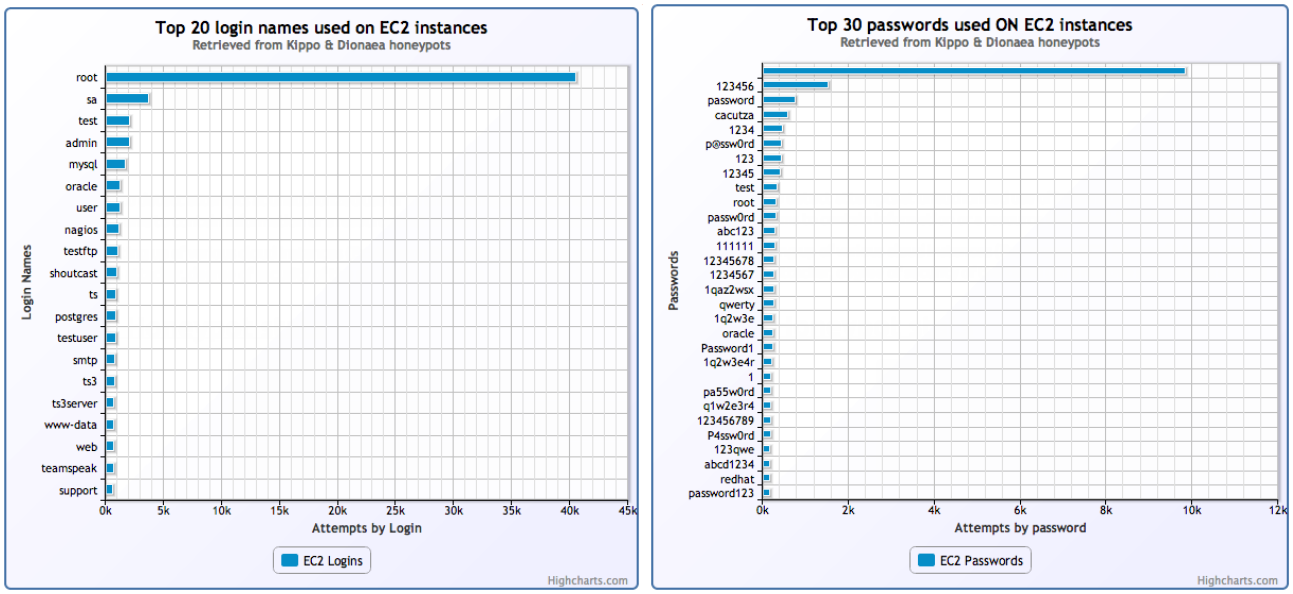

Figure 3: Common Attack Credentials

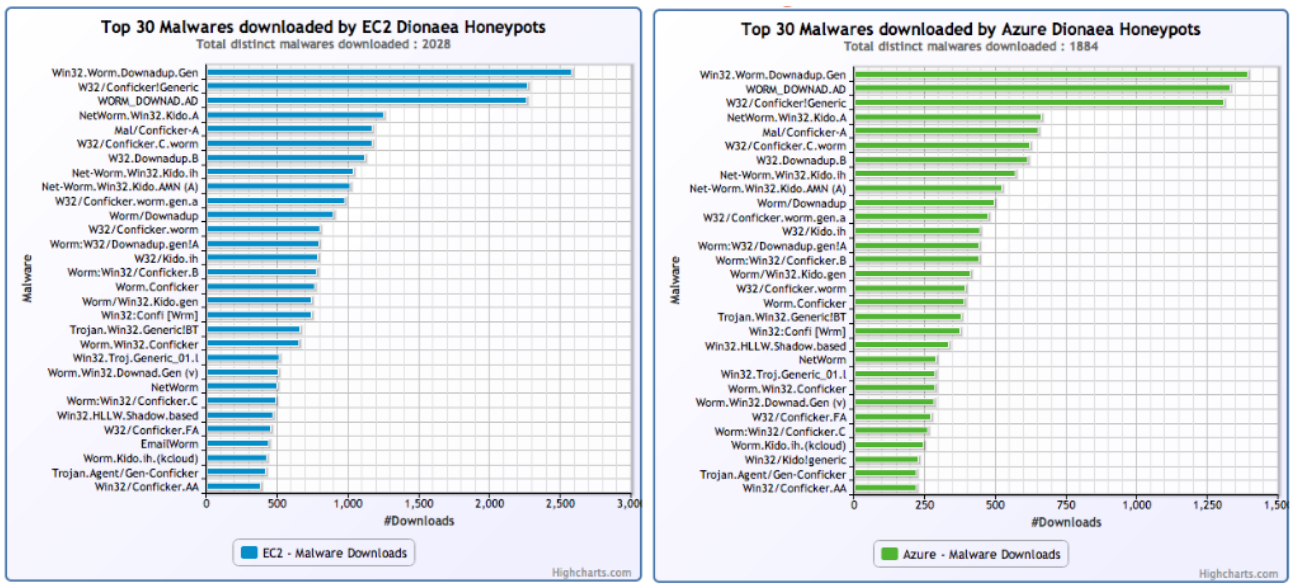

Figure 4: Most Downloaded Malwares

tried variations of common passwords that users find easy to type such as “qwerty” and “1qaz2wsx.” The results from EC2 can be seen in Figure 3. The results from Azure are nearly identical and are ommitted.

5.1.3 Malware

Our dionaea instances were successfully able to log the malware that attackers attempted to download onto our systems. As can be seen in Figure 4 most of these were variations of the Conficker / Downadup / Kido worm.

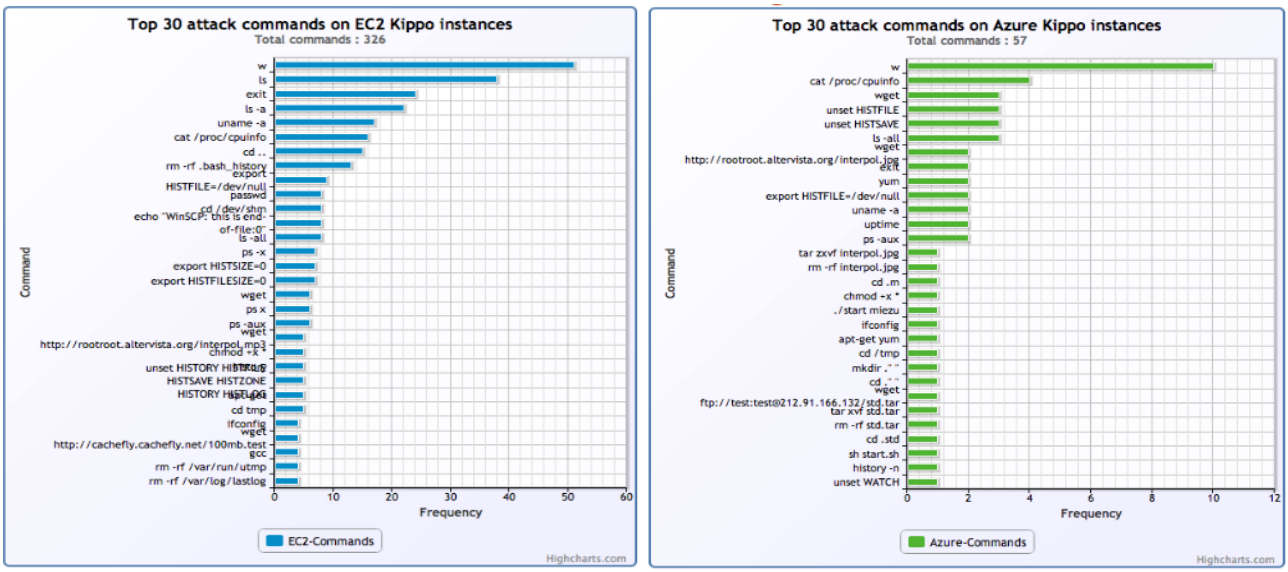

Figure 5: Common Attack Commands

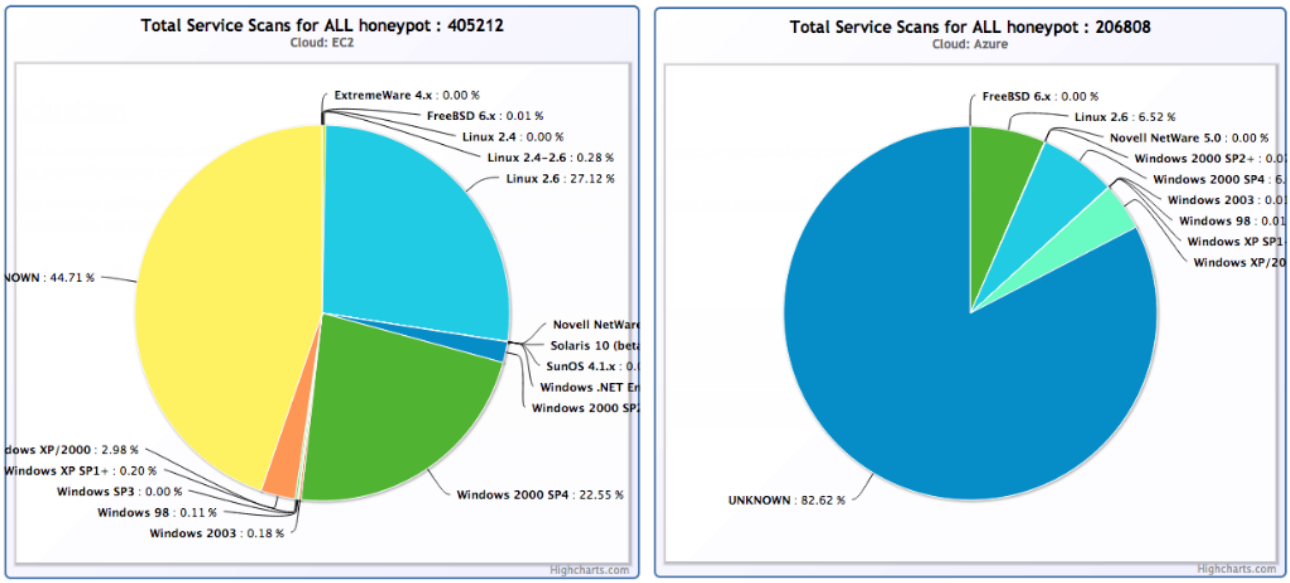

Figure 6: Attacker Operating Systems

5.1.4 Commands

Attackers who successfully logged into the emulated SSH terminals on our Kippo instances followed a consistent attack vector. Upon logging in, they usually would enter "w" to observe the other users currently logged into the system. From there, commands such as "ls" and "cpuinfo" were commonplace. "wget" was used frequently, ostensibly to fetch malicious data to install. The frequency distribution of attacker commands can be seen in Figure 5.

5.1.5 Operating Systems

A comparison of attacker Operating Systems can be seen in Figure 6. The honeypots in EC2 received roughly the same number of attacks from Linux-based attackers as Windows-based attackers. Our Azure instances received roughly 50% more attacks from Windows-based attackers.

Additionally, we found that many attacks originate from very old Operating Systems – this phenomenon was especially noticeable in EC2 and visible to a lesser extent in Azure. Roughly 27% of attackers in EC2 used Linux 2.6 or older, and 28% used Windows 2000 or XP. This proportion was much smaller in Azure: Linux 2.6 comprised only 6.52% of attackers, while Windows 2000 and XP made up 10.86% of attackers.

Lastly, the number of OS systems which were not detected by p0f v2 OS fingerprinting tool were considerably higher in Azure as compared to EC2. This indicates that the proportion of attackers using newer versions of OS like Windows Vista, Linux etc. and newer types of OS like Android OS is more in case of Windows Azure than EC2.

5.2 Cloud Discussion

5.2.1 Amazon EC2 vs Windows Azure

Amazon EC2 and Windows Azure services share a lot of similarities. They both offer infrastructure as a service (IaaS) where customers can launch virtual machines to be run in the cloud. Additionally, they both provide a wide variety of operating systems to use including many Linux and Windows distributions. They also have a few key differences listed below.

1. Azure puts a bigger focus on Windows-based operating systems than EC2. Windows Azure lists Windows operating systems first when selecting instance operating systems. Also, Azure charges the

same price per hour for Windows or Linux while EC2 charges significantly less for Linux operating systems.

2. The default way to access instances differs between them. Amazon EC2 provides users a private key file used to connect to instances via SSH. Windows Azure uses passwords for connecting to instances by default. These configurations can easily be changed after connecting to instances for the first time though.

3. Amazon EC2 easily allows all TCP or UDP ports to be opened for an instance. Windows Azure only allows ports to be allowed one by one.

Like the services themselves, our results across Amazon EC2 and Windows Azure were similar. We found that attacks originated from similar locations across the two services. Also, SSH login credentials and attacked services were comparable. Furthermore, we did not find any major differences between the time of attacks between Amazon EC2 instances and Windows Azure instances. They both had rather bursty attack patterns. There were a few minor differences between the two services though, which are listed below.

1. A higher proportion of the attacks on Azure originated from operating systems released after 2003. This was drawn from the the number of fingerprints marked as "unknown" by p0f v2 (which was released in 2003). These "unknown" operating systems are most likely ones which were released since then. The number of such systems was far greater on Azure than on EC2.

2. Azure saw a higher proportion of Windows-based attacks than did EC2. This may be due to Azure greater emphasis on Windows-based instances.

5.2.2 IBM Smartcloud

In addition to EC2 and Azure, we also had success in IBM Smartcloud, albeit to a lesser extent. We elected not to offer a comparision of IBM Smartcloud with the other clouds due to the lesser amount of collected data. We summarize our IBM Smartcloud results briefly by IP geographical location, SSH credentials, and attacker OS.

The six most common origin regions for the IP addresses that were used to communicate with our honeypots were China, United States, South Korea, Germany, India, and Taiwan. 40.3% of IP addresses were from China, 9.1% were from the US, and 7.8% were from South Korea. Germany, India, and Taiwan were tied at 3.9% of addresses.

In terms of SSH credentials, we found that the most common usernames by far was "root" which was used in 47% of the cases. After this were "www", "user", and "testuser" with 3.3%, 2.3%, and 2.3% frequency. Strangely, the most common password attempted was "⟩⟩radumadalina⟨⟨ " at 35.2% frequency, which was closely followed by the blank string "" at 28.6%. The next two most attempted passwords such as "123456" and "password" were used in less than 1% of the total cases.

As for the operating system, Windows and Linux were the most common with 94.0% and 4.5% of traffic, respectively. More specifically, Windows 2000, XP, and 2003 and Linux 2.6 were the most frequently used versions.

5.3 Honeypot Review

A honeypot-by-honeypot comparison of data can be seen in Table 2. Based on the number of attacks received, it is clear that Dionaea and Kippo are very effective honeypots to deploy in a cloud-based environment. The other honeypots faired relatively poorly, as they only received port scans and no attacks.

6 Limitations

There were several limitations of our study with respect to what and how we tested the honeypots in the various clouds.

Honeypot	Port Scans (EC2)	Port Scans (Azure)	Attacks (EC2)	Attacks (Azure)	Our Rating
Dionaea	262318	11,438	11,413	111	5 Stars
Kippo	61220	61,269	77,399	47803	5 Stars
Artillery	41,490	16,319	0	0	3 Stars
Amun	32775	18,782	0	0	2 Stars
Glastopf	7409	-	0	-	1 Star

Table 2: A summary of port scans and attacks

1. Our comparative study is limited to EC2 and Azure, since they are the only clouds which currently offer free, easy-to-use services. ElasticHosts allowed for a 5-day trial, which we used to launch a single instance. However, the narrow timeframe for this single instance constituted a prohibitively-small sample size, from which we were unable to derive any meaningful results. IBM Smarcloud provided free hosting for Red Hat Linux instances, but we ran into difficulty when launching instances. IBM Smartcloud offers very little in the way of pre-configured machine images; thus, the amount of required setup prevented us from launching more than a few instances. The number of successfully launched instances was comparatively few relative to EC2 and Azure, and thus we were unable to include IBM Smartcloud in our cross-cloud analysis.

2. Because of the limitations on free accounts, our study was further limited to micro-instances on EC2 and Azure. It is possible that many attackers ignored our micro instances, given their small size; larger instances would likely be more effective platforms from which to study attackers in the cloud.

3. Our study was limited to low interaction honeypots, with the lone exception of Kippo, a medium-interaction honeypot. Low and Medium interaction honeypots have the advantage of being easier to set up in the cloud, but are generally not as attactive to attackers. We investigated the use of high interaction honeypots such as HiHat; however, these instances did not generate any meaningful data.

4. Our honeypot study is confined to Linux-based instances. While we did launch several instances of Honeybot, a Windows-based honeypot, it produced very little meaningful data.

5. Many of the honeypots took a long time to configure correctly; this in turn limited the results we were able to collect. This difficulty can be attributed to a couple of reasons: 1) many of the honeypots are no longer maintained and have not received updates in many years, and 2) most of the honeypots lack up-to-date documentation, if any at all.

6. Our analysis of attacker operating systems was limited to pre-2003 systems. This limitation stems from our use of an outdated version of p0f. We used the version of p0f from the Ubuntu repository since it was easiest to configure and install across our many instances. However, that version was released in 2003 and thus, doe not contain fingerprint information for newer Operating Systems.

7 Conclusion

In our study we were able to determine a basic attacker profile by deploying a variety of honeypots in several cloud instances such as Amazon EC2, Microsoft Azure, and IBM Smartcloud. We found that attacks mostly come from China and the US. The most commonly attempted user was “root” and most commonly attempted passwords were “” and “123456”. The services that were targeted most frequently were SSH and HTTP. In terms of OS, Linux 2.6 and Windows 2000/XP were the most popular, and ethernet/modem was the connection protocol used the most. The timing of attacks was very random and bursty across all the regions and we could not identify any particular time of the day when hackers are most active.

Overall, we can say that the attack traffic in EC2 and Azure is rather similar. By most of our metrics (Country of Origin, Login Credentials, Targeted Services, and Protocols), the clouds presented few differences. The most significant distinction observed was the choice of attacker Operating System, where Azure saw a higher rate of Windows-based attacks.

We were also able to identify the honeypots which are most effective in the cloud setting. Dionaea and Kippo performed very well and produced copious amounts of useful data. The other honeypots – Amun, Artillery, and Glastopf – were not as effective, given that they received little traffic beyond port scans.

Group Member Contributions

1. Josh(slauson@cs.wisc.edu): Set up Kippo, Dionaea, and p0f on several instances in Amazon EC2 and Windows Azure. Also documented initial setup of Kippo. Researched and experimented with several other honeypots.

2. Rebecca(rjlam@cs.wisc.edu): Set up Artillery, Glastopf, Kippo, Dionaea on IBM Smartcloud. Experimented with Honeyd, HiHat, and Amun also on IBM Smartcloud. Performed data aggregation on IBM data. Wrote significant proportion of proposal and final paper.

3. Shishir(skprasad@cs.wisc.edu):

   (a) **Backend Infrastructure** Designed and created the backend data integration and enrichment infrastructure in Python for collecting the data from all the honeypot instances, parsing it and persisting it in a local database for data analytics.

   (b) **Frontend Infrastructure** Designed the PhP web application "Honeyweb - Know Thy Hackers" to analyse the data collected from various honeypot instances.

   (c) **Security Group Settings** Fixed the security settings for EC2 which helped in increasing the traffic to the honeypot instances ten-fold.

   (d) **p0f Fingerprinting** Researched and suggested usage of p0f fingerprinting tool for passively extracting lots of information about hackers.

   (e) **Honeypots** Researched and suggested the setup of Amun, Artillery, HiHat honeypot to test their effectiveness in luring hackers. Launched nine EC2 micro instances across the globe to collect honeypot data.

4. Sivasubramanian.R(sivas@cs.wisc.edu):

   (a) **Honeypots on Azure** Set Up instances of Dionaea, Kippo, Amun, Artillery, Hihat, Glastopf across the globe.

   (b) **Windows Honeypots** Researched and set up two windows bases honeypots on Azure - east asia (kfsensor and honeybot)

   (c) **Backend Infrastructure** Manipulation of queries of the Data Analytics tool for Kippo (Kippograph) to generate useful statistics and graphs.

5. Stephen(sbrown@cs.wisc.edu): Experimented with honeypot setup on ElasticHosts, researched and experimented with Artemisa, Honeybot, Honeyd, HiHat, and Glastopf honeypots. Hosted several of our EC2-based Dionaea, Kippo, and Glastopf instances across the globe. Wrote and edited significant portions of the final paper.

6. Everyone: Proposal Paper, Presentation, Final paper

References

[1] E. Alata, V. Nicomette, M. Kaâniche, M. Dacier, and M. Herrb. Lessons learned from the deployment of a high-interaction honeypot. In *Dependable Computing Conference, 2006. EDCC'06. Sixth European*, pages 39–46. IEEE, 2006.

[2] M. Balamurugan and B.S.C. Poornima. Honeypot as a service in cloud.

[3] R. Challoo and R. Kotapalli. Detection of botnets using honeypots and p2p botnets. *International Journal of Computer Science and Security (IJCSS)*, 5(5):496, 2011.

[4] E. Cooke, F. Jahanian, and D. McPherson. The zombie roundup: Understanding, detecting, and disrupting botnets. In *Proceedings of the USENIX SRUTI Workshop*, pages 39–44, 2005.

[5] J. Göbel. *Amun: A python honeypot*. Universität Mannheim/Institut für Informatik, 2009.

[6] R. McGrew. Experiences with honeypot systems: Development, deployment, and analysis. In *System Sciences, 2006. HICSS'06. Proceedings of the 39th Annual Hawaii International Conference on*, volume 9, pages 220a–220a. IEEE, 2006.

[7] SR Nithin Chandra and TM Madhuri. Cloud security using honeypot systems.

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