Dynamic Intrusion Deception in a Cloud Environment

Chia-Chi Teng
Cybersecurity
Brigham Young University
Provo, UT, USA
email: ccteng@byu.edu

Aaron Cowley
Cybersecurity
Brigham Young University
Provo, UT, USA
email: acow777@gmail.com

Ressel Havens
Cybersecurity
Brigham Young University
Provo, UT, USA
email: russel.havens@gmail.com

***Abstract*—As cyber-attacks become more sophisticated, Network Intrusion Detection Systems also need to adapt to counter the evolving advanced persistent threats. Security deception, such as Honeypot, is an emerging defense tactic for security operation in enterprise network or commercial cloud environment. A well designed Honeypot can fool attackers and malicious agents into a made-up system that is monitored by security operators who can safely observe the attacks and promptly develop counter measures. However, the availability of Anti-Honeypot technologies has made the deception defense more challenging. A dynamic deception method is necessary to counter the modern Honeypot detection systems. We propose a dynamic intrusion deception method designed to run in a public cloud environment. A prototype of Honeynet is built using the Microsoft Windows Azure Resource Group virtual machines and network management platform.**

Keywords—Cloud Computing; Intrusion Detection; Intrusion Deception; Honeypot; Honeynet.

I. INTRODUCTION

Since commercial cloud computing services became available over a decade ago, the cloud computing technology platforms have made significant advancement in research and development. Public cloud providers, such as Microsoft, Amazon, Google and IBM, are offering a large variety of services from infrastructure as a service (IaaS), platform as a service (PaaS) to software as a service (SaaS). Cloud computing is now the backbone of thousands of enterprises and organizations where a 2018 industry study shows 77% of enterprises have at least one application or a portion of their Information Technology (IT) infrastructure in the cloud [1] and trending up. While the industry continues to invest in cloud computing, the study also shows that about one-third of the IT decision-makers saying security concerns is one of their top challenges.

In addition to private enterprises and business, government agencies are also embracing the cloud computing technology to support their future infrastructure and services. For example, United States Department of Defense (DoD) listed cloud computing as one of the top priorities in their Digital Modernization Strategy [2]. DoD has also recently awarded a ten-year ten billion dollar (USD) contract to Microsoft for its Joint Enterprise Defense Infrastructure (JEDI) project [3].

With all the sensitive and classified information being stored in the cloud, the security requirement has also increased. It is important to understand the threat models, attack surfaces, and available controls in the cloud environment to effectively manage the security risks [4].

Intrusion Detection System (IDS) and Intrusion Prevention System (IPS) [5] are well-known security controls commonly deployed in enterprise or cloud computing environment. They can protect target systems based on network or host activities by denying access to malicious attacks. However, IDS/IPS do not usually attempt to discover additional information about the attacks or attackers, which is the primary function of the Honeypot technology.

Honeypots are usually designed to resemble valid systems or services with exploitable vulnerabilities to lure attackers to gain access. When working in conjunction with IDS, attackers' activities are monitored and analyzed by security operators once they are in the Honeypots. Valuable information can be discovered while an attack is taking place.

Unfortunately, Anti-Honeypot technologies have also been developed by spammers and other malicious parties to counter this defense measure. The unique capabilities of public cloud computing platforms can enable a new type of Honeypot that is more dynamic, realistic and cost effective in a way that defensive resources mimicking real targets can be instantiated and configured in real time as malicious attacks are being detected.

A design and prototype of a dynamic Honeypot system is presented below with a review of other recent work on cloud computing related Honeypot and Honeynet. As the system detects potential attacks, such as Brute-force SSH, it dynamically creates containers, re-routes malicious network traffic and actively engages with the attacker to gather information about the attack and attacker. While the preliminary work is implemented and tested on Microsoft Windows Azure platform, it can easily be ported to other public cloud providers.

II. BACKGROUND

Since the concept of Honeypot was first introduced in 1998 [6], its applications have steadily been gaining popularity and support as Honeypot evolved from a non-traditional tool to one of the commonly used security controls. For example, United States DoD Cloud Computing Security Requirement Guide [7] specifies Honeypot as one of the standard controls. Many varieties of Honeypot intrusion deception systems have been proposed over the years, which can be classified based on their level of interaction, scope, or targeted attack type [8]. A recent survey [9] of Honeypot systems in a cloud environment further classifies them based on architectures and functionalities. These cloud-based solutions include

- Honeynet [10]

 ISBN: 978-1-61208-786-3

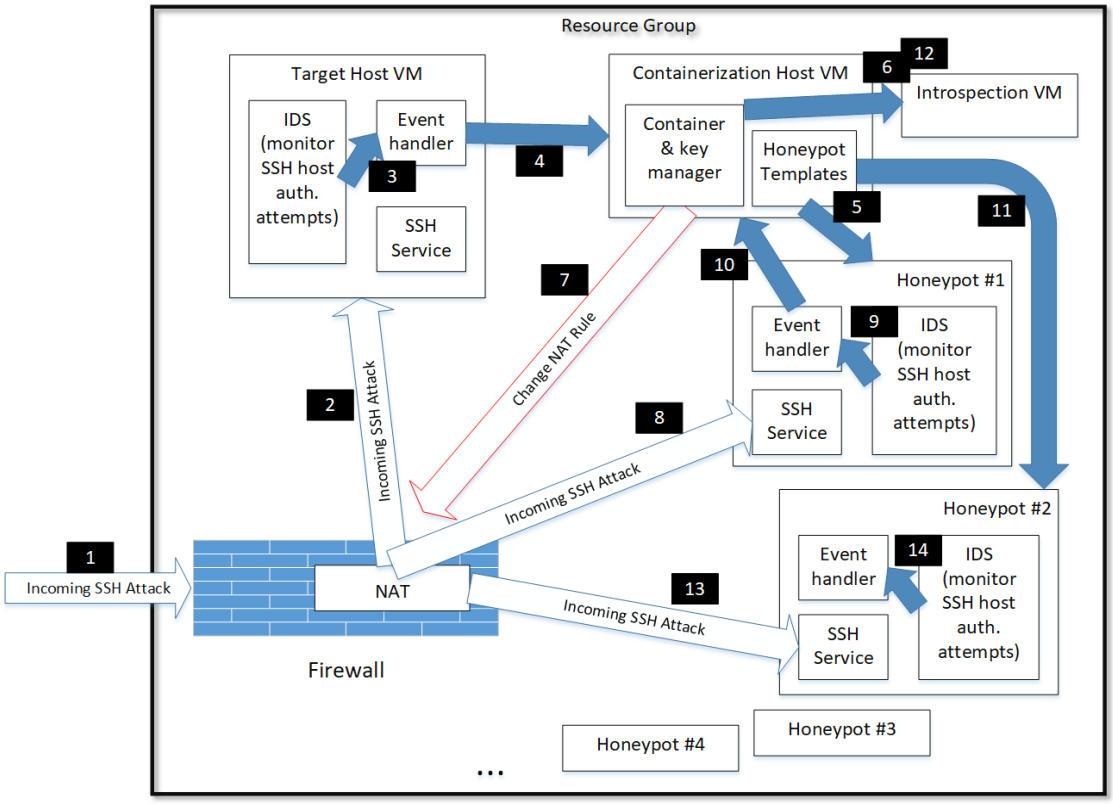

Figure 1. Dynamic Honenet System Diagram

- HoneyFarm [11]
- HoneyBrid [12]
- HoneyMix [13]
- HoneyProxy [14]

Honeypots are usually used as an integrated-component in an defense in depth strategy instead of a standalone security control. Study showed that combining Honeypot with IDS/IPS can increase the success rate of attack detection and prevention [15]. While IDS/IPS can deny access to malicious attack, Honeypot can further gather valuable information, such as attacker's identity and technique [16].

The security community has developed several open-source Honeypot packages where some of them are considered quite mature according to industry evaluation [17]. For example:

- Dionaea [18]: a low-interaction server-side Honeypot that collects network malware.
- Honeyd [19]: a multi-purpose low-interaction Honeypot using virtual hosts and services.
- Kippo [20]: low-interaction SSH Honeypot.
- Glastopf [21]: low-interaction web applications Honeypot.

CloudHoneyCY [22] further proposed an open-source framework to integrate a variety of existing Honeypot, such as the one listed above, to work together in a cloud environment.

SSH is one of the commonly used attack surface against services deployed in the cloud as it is the standard protocol for developers and operators to access the target system. Research shows that Brute-force/dictionary attacks against remote services, such as SSH is ranked among one of the most common forms of attacks that compromise servers [23]. Using Virtual Machine Introspection (VMI) technique with a VM-based SSH Honeypot can be effective in defending against such attacks [24].

Study [25] has shown that new containerization technology, such as Docker [26], can improve Honeypot systems with following advantages over VM-based solutions,

- Scalability: spinning up/down containers is easier.
- Performance: spinning up/down container is faster.
- Cross-platform: containerization is supported by all major cloud providers.
- Cost: contains have smaller CPU and memory footprint which incur lower cost than VM's in typical pay-per-use cloud environments.

The cloud containers will inevitably enable Honeypot or Honeynet to be more dynamic and looks more like a real system. Research and development of container introspection technology has drastically increase the capability of monitoring applications inside containers [27].

Today's cybersecurity is a cat-and-mouse game where threat actors constantly make improvement with their tools to by-pass defensive controls. Anti-honeypot technologies have been successful in identifying and countering low- and medium-interaction Honeypots [28], for example, Honeypot Hunter [29]

 ISBN: 978-1-61208-786-3

has been used by spammers to identify HTTP and SOCKS Honeypot proxies. Research [30] showed that a dynamic Honeypot can be effective in defending malicious attacks.

Leveraging the previous research and the latest cloud technology available, we propose a Honeynet system with the following characteristics,

- Dynamically provision and revoke Honeypots based on level of malicious network activities.
- High-interactivity Honeypots with dynamically configured SSH service.
- Use container technology, e.g., Docker, for increased performance and scalability.
- Easily deployable in a commercial cloud platform, e.g., Microsoft Azure.

The design and implementation of the proposed system are discussed below.

III. METHODS

This dynamic Honeynet design is currently targeting commercial public cloud computing services with large user base and mature technology platform. Leading providers, such as Microsoft Azure and Amazon AWS have the concept of "resource group", which is a logical collection of assets grouped together for effective management, such as provisioning, monitoring, and access control. The high-level design of the proposed cloud-based Honeynet system is shown in Figure 1 above in a logical resource group environment. As this system must rely on the cloud provider's resource management interface, e.g., Azure Resource Manager or AWS Resource Access Manager, the actual implementation might be somewhat platform dependent.

A resource group can be setup to include the following collection of items,

- A Firewall and programmable Network Address Translation (NAT) or reverse proxy layer.
- Regular service(s) which might be the initial target(s) of an SSH brute-force attack.
- A target VM containing real services which also monitors network intrusion with an IDS. Event trigger will take place when certain pre-defined malicious activity is observed.
- A VM host for the containerization software run-time, e.g., Docker, which also handles the IDS event triggered by an attack then dynamically provisions and configures Honeypot accordingly.
- Pre-built container template of Honeypot with SSH services and IDS.
- An Introspection VM for monitoring active Honeypots.

In the scenario of a SSH brute-force attack, the following step-by-step actions will take place as labelled in Figure 1.

1) Incoming SSH brute-force attack reaches Firewall and NAT.
2) Attack traffic directed to the target host.
3) After a number of failed SSH login attempts, an IDS event triggers indicating a SSH brute-force attack taking place.
4) The event handler invokes container host services.
5) An initial Honeypot service (#1) container is provisioned, which hosts a simulated SSH service and a pre-configured IDS.
6) Notify the Introspection VM to begin monitoring Honeypot #1.
7) Configure NAT to redirect attacker IP's SSH traffic to the Honeypot service.
8) Incoming SSH attacks now goes to Honeypot #1.
9) IDS detects brute-force SSH attack on Honeypot #1.
10) Invoke container service to create new Honeypot service (#2), generate authorized key for Honeypot #2, then allow attacker to successfully connect to the simulated SSH service where host and authorization information to Honeypot #2 can be found.
11) Honeypot #2 is provisioned.
12) Notify the Introspection VM to begin monitoring Honeypot #2.
13) Attacker attempts connection to Honeypot #2 with the host and authentication information found in Honeypot #1.
14) IDS detects attacker's activity on Honeypot #2 and create new Honeypots as needed.

Depending on the security operator's objective, the steps of detecting malicious activities, provisioning and directing attacker to a new Honeypot can be repeated as needed until certain information about the attacker is discovered, or until the attacker become inactive. The operator can configure the Honeynet based on available resources and desired level of interactivity.

The event handler in the Honeypots can be configured with a timer where it initiates the process of self-revocation or de-provision of the container if certain amount of time elapsed without the attacker actively engaged. The automated Honeypot provision and revocation feature can potentially make the Honeynet more dynamic and better at countering anti-Honeypot technology.

IV. RESULTS

Base on the design described above, a functioning prototype is successfully implemented with the following specifications as shown in Figure 2.

- Cloud computing platform: Microsoft Windows Azure with Azure Virtual Network, Azure Resource Group and Resource Manager.
- Target Host: Ubuntu Linux server running OpenSSH Daemon (sshd).
- IDS: Zeek (formerly Bro) Network Security Monitor.
- Network Address Translation (NAT): Azure Service Fabric Reverse Proxy.
- Container Host: Docker and its Command-Line Interface (CLI), such as "docker run" and "docker cp".

 ISBN: 978-1-61208-786-3

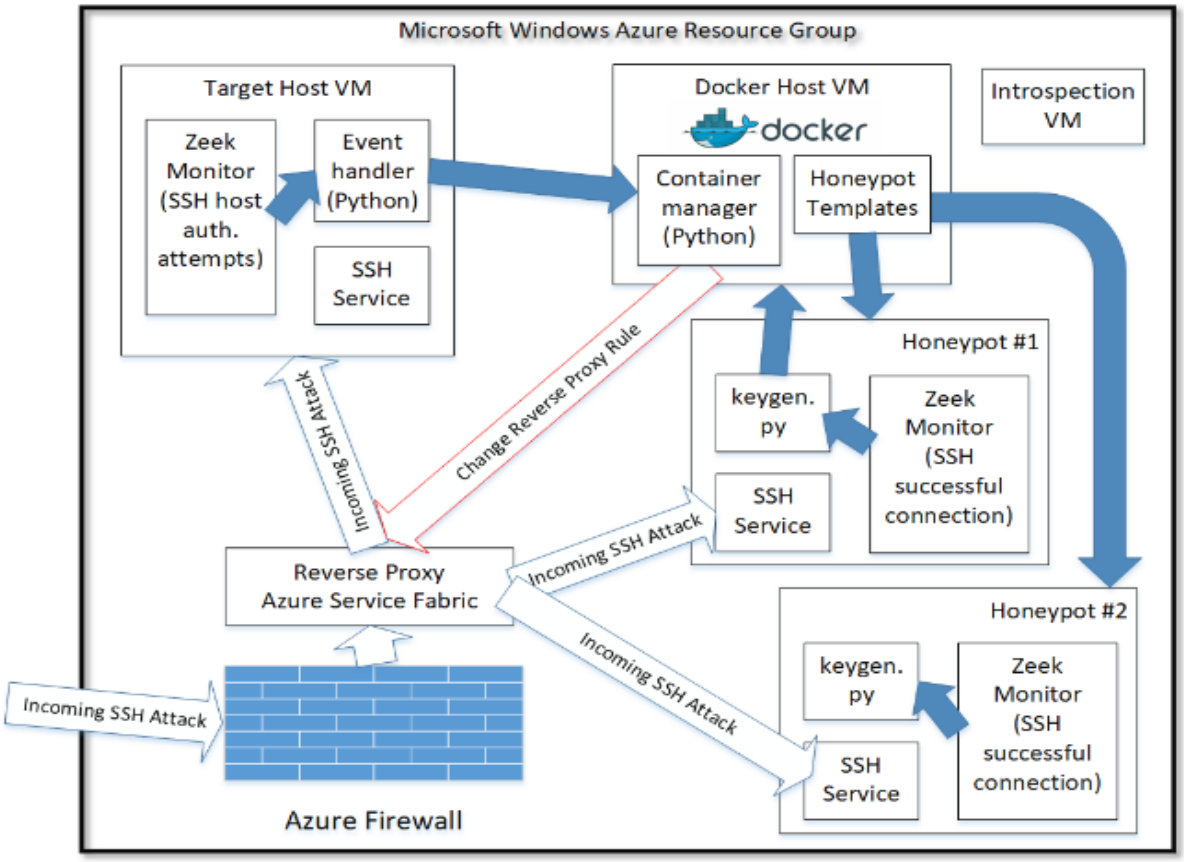

Figure 2. Functional Prototype as Implemented in Microsoft Windows Azure Cloud Environment.

- Two container templates/images: Honeypot #1 is the initial container where the SSH brute force attack is redirected to. Honeypot #2 is the secondary container where the attacker is lured to after allowed successful SSH connection to Honeypot #1.

In the test environment created as an Azure Resource Group, we configured a Zeek trigger to take place after ten failed SSH authentication attempts on the Target Host. The event handler is a Python script that connects to the Docker Host and invoke "docker run" to lunch Honeypot #1. The Python script also collects host information of the attacker and dynamically configure the Azure Reverse Proxy to redirect the SSH attack traffic to Honeypot #1.

If Honeypot #1 continues to see incoming SSH brute force attack, it will invoke another Python script that "docker run" Honeypot #2, then generate a new SSH authorized_keys file and "docker cp" to the newly provision container. At the same time, the Python script will leave bread crumb in Honeypot #1 containing the authorized key to Honeypot #2 for the attacker to find.

The completed Honeynet was blind tested by multiple penetration testers using tools such Ncrack [31]. The system ran successfully as designed in all instance where the attackers will reach Honeypots and attempt other exploits.

V. Conclusion

As more commercial and critical services and applications are migrating to the cloud infrastructure, it is imperative to design and implement proper controls to defend against external threats. While the underlying technologies for this Honeynet system may already exist, it is a novel attempt to integrate them in such a way that presents a more dynamic, scalable defense solution in the cloud environment.

While the preliminary results are promising, more work is needed to make it a complete solution. Potential future work includes,

- Integration with container introspection software, e.g., Prometheus [32].
- Honeypot for other common attack vectors, e.g., SQL injection.
- Working prototype with other public cloud platforms, e.g., Amazon AWS.

References

[1] L. Columbus, "State of Enterprise Cloud Computing," Forbes, 2018. https://www.forbes.com/sites/louiscolumbus/2018/08/30/state-of-enterprise-cloud-computing-2018/ [retrieved: Jun 2020]

[2] DoD, "DoD Digital Modernization Strategy," Department of Defense, U.S.A., 2019. https://media.defense.gov/2019/Jul/12/2002156622/-1/-1/1/DOD-DIGITAL-MODERNIZATION-STRATEGY-2019.PDF [retrieved: Jun 2020]

 ISBN: 978-1-61208-786-3

[3] Congressional Research Service, "DOD's Cloud Strategy and the JEDI Cloud Procurement," 2019. https://fas.org/sgp/crs/natsec/IF11264.pdf [retrieved: Jun 2020]

[4] N. Afshan, "Analysis and Assessment of the Vulnerabilities in Cloud Computing," International Journal of Advanced Research in Computer Science, vol. 8, no. 2, 2017.

[5] M. Rani and Gagandeep, "A Review of Intrusion Detection System in Cloud Computing," in proceedings International Conference on Sustainable Computing in Science, Technology & Management (SUSCOM 2019), pp. 770-776, 2019.

[6] F. Cohen, "The Deception Toolkit," Risks Digest vol. 19, 1998.

[7] DoD, "Department of Defense Cloud Computing Security Requirement Guide," Defense Information Systems Agency, U.S.A., 2017. https://rmf.org/wp-content/uploads/2018/05/Cloud_Computing_SRG_v1r3.pdf [retrieved: Jun 2020]

[8] C. K. Ng, L. Pan, and Y. Xiang, "Honeypot Frameworks and Their Applications: A New Framework," Springer Briefs on Cyber Security Systems and Networks, 2018.

[9] S. Krishnaveni, S. Prabakaran, and S. Sivamohan, "A Survey on Honeypot and Honeynet Systems for Intrusion Detection in Cloud Environment," Journal of Computational and Theoretical Nanoscience, vol. 15, pp. 2949-2935, 2018.

[10] L. Spitzner, "The Honeynet Project: Trapping the Hackers," IEEE Security & Privacy, vol. 1, no. 2, pp. 15-23, 2003.

[11] L. Spitzner, "Honeypot Farms," Symantec, 2003. https://www.symantec.com/connect/articles/honeypot-farms [retrieved: Jun 2020]

[12] R. Bertheir, "HoneyBrid: Combining Low and High Interaction Honeypots," Honeynet, 2009. https://www.honeynet.org/2009/05/27/honeybrid-combining-low-and-high-interaction-honeypots/ [retrieved: Jun 2020]

[13] W. Han, Z. Zhao, A. Doupe, and G. J. Ahn, "HoneyMix: Toward SDN-based Intelligent Honeynet," in proceedings 2016 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization (SDN-NFV Security '16), pp. 1-6, 2016.

[14] S. Kyung et al., "HoneyProxy: Design and implementation of next-generation honeynet via SDN," In proceedings 2017 IEEE Conference on Communications and Network Security (CNS 2017), pp. 1-9, 2017.

[15] S. Ravji and M. Ali, "Integrated Intrusion Detection and Prevention System with Honeypot in Cloud Computing," in proceedings 2018 International Conference on Computing, Electronics & Communications Engineering (iCCECE), pp. 95-100, 2018.

[16] P. A. Pandire and V. B. Gaikwad, "Attack Detection in Cloud Virtual Environment and Prevention using Honeypot," in proceedings of the International Conference on Inventive Research Applications (ICIRCA 2018), pp. 515-520, 2018.

[17] C. Polska, "Proactive Detection of Security incidents: Honeypots," ENISA, Tech. Rep., 2012.

[18] Dionaea, http://dionaea.carnivore.it/ [retrieved: Jun 2020]

[19] Honeyd, http://www.honeyd.org/ [retrieved: Jun 2020]

[20] Kippo, https://github.com/desaster/kippo/ [retrieved: Jun 2020]

[21] Glastopf, http://glastopf.org/ [retrieved: Jun 2020]

[22] H. Gjermundrod and I. Dionysiou, "CloudHoneyCY - An Integrated Honeypot Framework for Cloud Infrastructures," in proceedings 2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing, pp. 630-635, 2015.

[23] V. Singh and S. K. Pandey, "Revisiting Cloud Security Threat: Dictionary Attack," in proceedings of International Conference on Advancements in Computing & Management (ICACM 2019), pp. 175-180, 2019.

[24] S. Sentanoe, B. Taubmann, and H. P. Reiser, "Virtual Machine Introspection Based SSH Honeypot," in proceedings of the 4th Workshop on Security in Highly Connected IT Systems (SHCIS'17), pp. 13-18, 2017.

[25] N. Majithia, "Honey-System: Design, Implementation and Attack Analysis," PhD Thesis, Indian Institute of Technology, Kanpur, 2017.

[26] M. N. Khandhar and M. S. Shah, "Docker-The Future of Virtualization," International Journal of Research and Analytical Reviews, 6(2), pp. 164–167, 2019.

[27] T. Watts, R. G. Benton, W. B. Glisson, and J. Shropshire, "Insight from a Docker Container Introspection," in proceedings of the 52nd Hawaii International Conference on System Sciences (HICSS 2019), pp. 7194-7203, 2019.

[28] J. Uitto, S. Rauti, S. Laurén, and V. Leppänen, "A Survey on Anti-honeypot and Anti-Introspection Methods," Recent Advances in Information Systems and Technologies. WorldCIST 2017. Advances in Intelligent Systems and Computing, vol. 570, Springer, 2017.

[29] N. Krawetz, "Anti-Honeypot Technology," IEEE Security & Privacy, vol. 2, no. 1, pp. 76-79, 2004.

[30] K. R. Sekar, V. Gayathri, G. Anisha, K. S. Ravichandran, and R. Manikandan, "Dynamic Honeypot Configuration for Intrusion Detection," in proceedings 2nd International Conference on Trends in Electronics and Informatics (ICOEI 2018), pp. 1397-1401, 2018.

[31] Ncrack, https://nmap.org/ncrack/ [retrieved: Jun 2020]

[32] Prometheus, https://prometheus.io/ [retrieved: Jun 2020]

 ISBN: 978-1-61208-786-3