Cloud-native honeypot deployment

Authors
Ivan Beres, MSc (Royal Holloway, 2021)
Darren Hurley-Smith, ISG, Royal Holloway

Abstract

What is your organisation's incident response strategy when you are under attack? Do you ban all the offending IPs and hope the attacker stops, or do you try to collect information about the possible attack vector? What is the cost of shutting down some or all of your services?

Alternatively you can provision carbon copies of your cloud environment and quickly deploy a fake version of it using cloud-native services. Redirecting your attackers to this honeypot network lets you understand and mitigate attacks, gather threat analytics intelligence and, most importantly, gives you time to secure your environment. Exploring open problems facing cloud infrastructure, this article describes the implementation, feasibility, and benefits of cloud-native honeypots[a].

[a]This article is published online by Computer Weekly as part of the 2022 Royal Holloway information security thesis series `https://www.computerweekly.com/ehandbook/Royal-Holloway-Cloud-native-honeypot-deployment`. It is based on an MSc dissertation written as part of the MSc in Information Security at the ISG, Royal Holloway, University of London. The full thesis is published on the ISG's website at `https://www.royalholloway.ac.uk/research-and-teaching/departments-and-schools/information-security/research/explore-our-research/isg-technical-reports/`.

Introduction

Honeypots have been around in some form since 1989, deployed in information security defence and research. A honeypot is an information security tool designed to capture data on attacker behaviour, attack tactics and malware. Misconfigured and vulnerable services, servers or entire networks are set up on purpose to lure in attackers by exposing vulnerabilities for them to interact with, wasting their time and resources. This way, honeypots can detect and mitigate attacks and collect threat intelligence to be analysed to strengthen defences. Honeypots underpin the research on new attack vectors and exploits, aiding in keeping information security defences ready for the next attack.

Honeypots are servers or networks set up to lure in attackers.

The problem with honeypots, in general, is that they are easy for attackers to discover as decoys. Honeypots often do not fit (contextually) into the rest of the organisation's infrastructure or expose some outdated, vulnerable protocol. In the former case, attackers differentiate honeypots from real infrastructure by identifying disparities between available services on the honeypot and the expected services of the target. In the latter case, honeypots are identified by breaking best practice guidelines or policies internal to the organisation, but which may be public knowledge (or known to the attacker).

Attackers research their targets, and if they find a host on a network that does not quite match their mental model of what that network should look like, the game is up. When a honeypot is discovered, it becomes at best useless and, at worst, a liability. The information collected on how attackers exploit a vulnerable OpenSSL implementation may be useful for research but brings little to the table for an organisation. Attackers may move away from the honeypot when discovered to look for other targets, or they may try to attack and take over the honeypot to attack other hosts on the network.

Many businesses transitioning to the cloud are adopting cloud-native services. Deploying honeypots can also benefit from the integration and on-demand compute resources offered on cloud platforms.

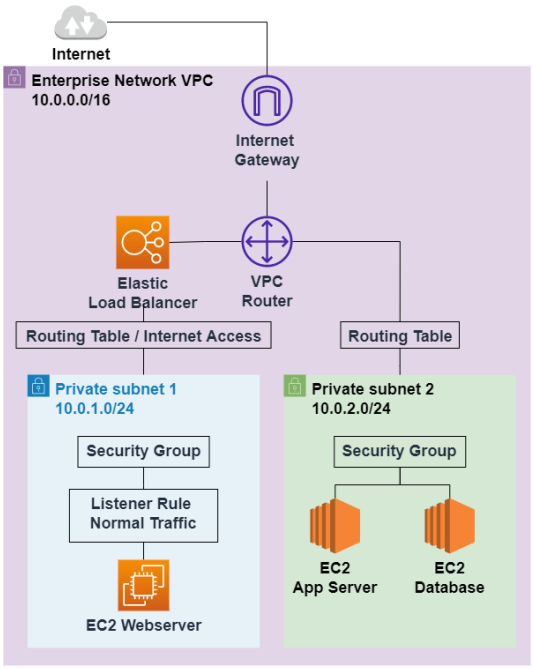

Figure 1: Architecture schematic in normal mode.

Cloud services make it possible to create separate virtual subnets and spawn honeypot hosts in them on the fly. Attackers are redirected to this fake subnet, where they can interact with a full stack of honeypot hosts. Because the subnet is a copy of the real network, attackers will have difficulty discovering that they are attacking a fake system. The information captured in the fake subnet is analysed to identify weak points and offer insight into the strategies and attack vectors attackers may use against the real system. The cloud makes it possible to set up a system where honeypots only spin up if an appropriate (hostile) event is detected. One example of such an event is: detecting a significant number of TCP or UDP requests over a wide range of ports in a given period of time. This would indicate host and service enumeration, an attempt by attackers to map potential targets for attack.

Implementation

We set up a simple test environment in AWS with a web server running Apache, a database server with MySQL and an application server EC2 instances in a Virtual Private Cloud (VPC). Only real hosts exist in the network in Normal Mode, and no honeypot instances are provisioned. Figure 1 represents this environment where all incoming requests are forwarded to the real webserver.

When malicious activities such as external scans are detected at the firewall level, honeypot instances spin up. Using python scripts wrapped in AWS's Lambda functions, we spin up web, application and database servers as copies of the existing servers based on Amazon Machine Images (AMI), as shown in Figure 2. Lambda is an AWS service that lets you run code without the need to provision or manage any server.

Lambda is an AWS service that lets you run code without the need to provision or manage any server.

Another Lambda function takes the offending IP addresses and routes attackers to the newly provisioned honeypot hosts. As shown in Figure 3, the Lambda function creates a new Elastic Load Balancer listener rule to redirect all requests coming from the offending IP to the honeypot webserver. Each new offending IP address is quarantined with a new listener rule. The load balancer has a static IP functionality, so the attacker is not aware of the changes happening in the background. It's common to have a load balancer in front of

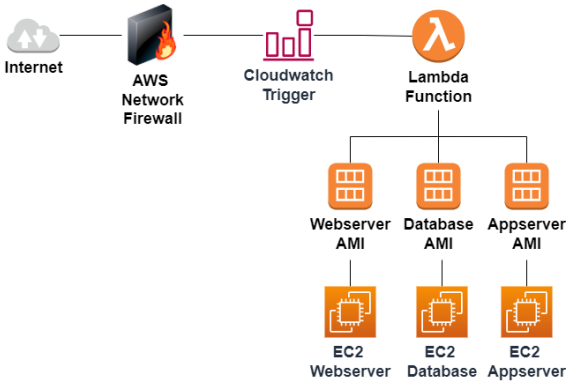

Figure 2: Honeypot provisioning process.

multiple internet-facing web servers. It is expected to have servers spin up and down depending on the load. Based on our tests, it is not possible to distinguish between the real servers and honeypots in this environment.

To summarise, the system goes into Defence Mode when a network anomaly is detected. This event triggers the provisioning of honeypots from AMIs and, subsequently, reconfiguring the Elastic Load Balancer listener rule to redirect requests from the offending IP address to the honeypot webserver, seen in Figure 4. The system switches from Defence Mode to Normal Mode when no network anomaly is detected for a certain amount of time. The list of offending IPs may or may not be discarded depending on preference. Because IPV4 addresses are often reassigned to other clients, it is reasonable to discard quarantined IP addresses to allow legitimate clients to connect to the real servers in the future. In Normal Mode, the honeypot servers are shut down and terminated. By switching between Normal and Defence modes, honeypots servers only consume resources when required, reducing costs and resource consumption.

In Defence Mode, the system can spin up new EC2 instances in under 40 seconds. Rapid honeypot provisioning enables honeypots to spin up on-demand, only when needed. There is no need for honeypots to be up and running all the time.

Timing tests

The speed of EC2 instance provisioning in AWS makes it possible to spin up a server before a single host can be scanned for open ports. This makes it possible to have an army of dormant honeypot servers waiting to be brought alive at the right time. Attackers must limit the speed of their port scans to avoid detection: network mapping and port scanning of large networks with numerous hosts may take hours. Depending on the number of targets to scan and the type of scan performed, it can take up to 21 minutes for a single host to be scanned. In our tests, scanning a single IP address with

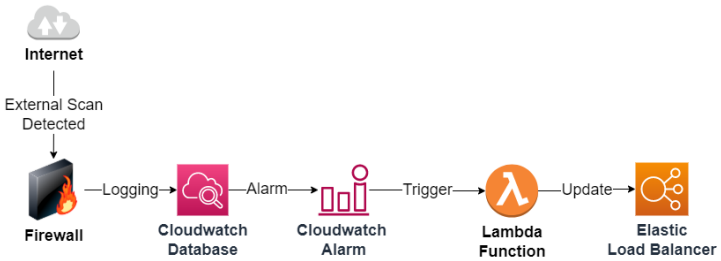

Figure 3: Elastic Load Balancer reconfiguration and quarantine.

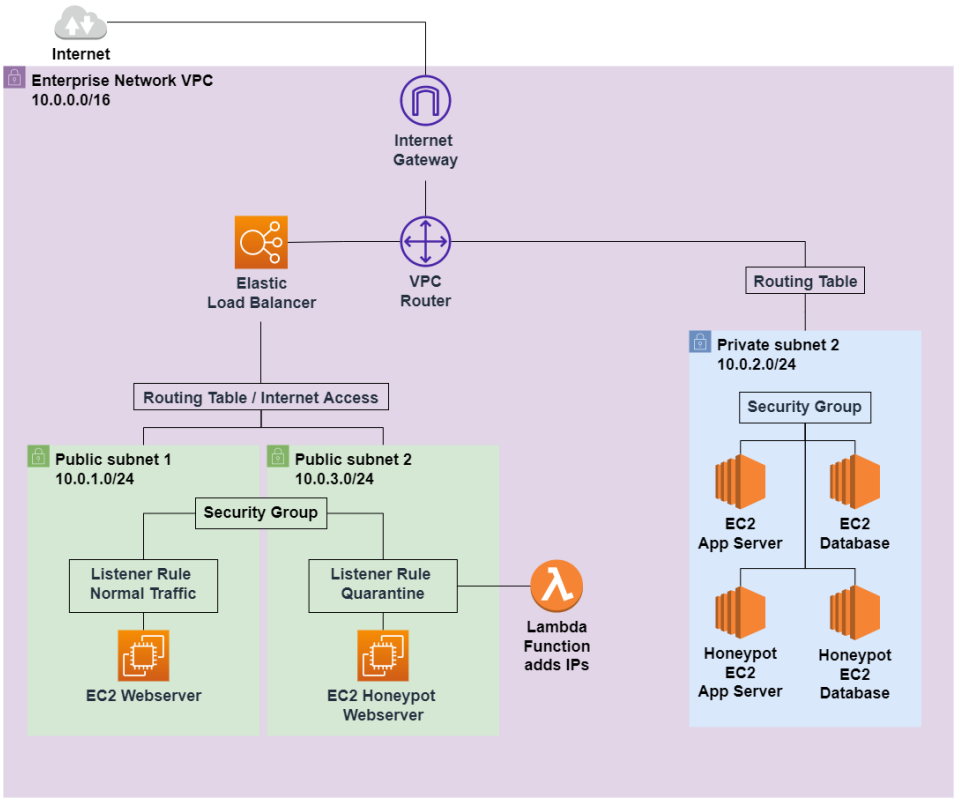

Figure 4: Architecture schematic in defence mode.

NMAP -sV (version detection) took 60 seconds. It takes around 40 seconds for a honeypot webserver instance running Linux to be provisioned from an AMI and accept requests.

Startup times vary depending on the instance type, the type and number of software packages that need to be started automatically baked into the AMI, infrastructure load on AWS and the kind of boot volume used. A database instance running on a Windows Server may take considerably longer to accept connections. For servers not facing the internet, boot time is of less importance.

Reconfiguring the Elastic Load Balancer's listener by adding a rule to forward requests from a specific IP address is instant. When Defence Mode is active, and honeypot instances are provisioned, new offending IP addresses are quarantined instantly. The time it takes the honeypot resources to become available allows more checks to be run to evaluate and quarantine IP addresses if necessary. Table 1 shows the mean of 5 runs of the test environment, all conducted from the same test machine over the same connection. Identifying the potential impacts of the attacker's position relative to the datacentre in which our cloud servers are hosted is the focus of ongoing work.

Test type	Net time
Spinning up an existing EC2 webserver	≈ 38 seconds
Spinning up an EC2 webserver from an AMI	≈ 46 seconds
NMAP -sV scan of a single IP address	≈ 60 seconds

Table 1: The mean of 5 runs all conducted from the same test machine over the same connection.

Potential issues and limitations

A possible issue with this system is that, after some time, denial-of-service attacks may reveal to the attacker that they are interacting with a fake system. If a DoS attack is successful and puts a heavy load on the system, requests from quarantined IPs may not be served or may be served considerably slower. Requests from regular IPs will still be served with normal response times by the real servers. An auto-scaling setup may mitigate this discrepancy. Provisioning more server resources would mimic the behaviour of a real system under a DoS attack.

Conclusion

The cloud offers new and enhanced ways for deploying and using honeypots in threat detection, analytics and incident response. Integrated, programmatically reconfigurable cloud services provide opportunities to overcome the shortcomings and change the role of honeypots in general. Honeypot believability is greatly enhanced by building honeypot servers that are exact copies of real servers in the network. Network scans conducted from the attackers' perspective prove that honeypot versions of real servers created from AMIs are indistinguishable from real servers. Security is achieved by network segmentation, as requests from malicious IP addresses are forwarded to a separate subnet, isolating attackers from interacting with other parts of the network. Relying on resources that don't exist at the time of an attack to become available enhances honeypot functionality, reduces resource and maintenance costs and opens new directions in which honeypots could be used in the future. The results are reproducible, and the design concept is transferrable to other cloud service providers.

Future work

Full-stack honeypot provisioning

The test configuration in this project is limited to an Apache webserver serving a static website. A full-stack design can be simulated by provisioning additional honeypot databases and application servers for maximum believability. In this case, data in the database will have to be anonymised to avoid leaking sensitive information in a successful data breach. The entire stack is duplicated in Defence Mode, serving requests running under the same configuration of slightly vulnerable software components. The honeypot web server, application and database servers are associated with their separate subnet to take advantage of network segmentation, virtually isolating the attacker from the rest of the Enterprise Network. This way, the entire application architecture is duplicated with anonymised data. Attackers are redirected and interact with the fake system, indistinguishable from the real system, in an isolated subnet. Such a system effectively wastes the attacker's resources and time, captures attack vector information for further threat analysis and acts as active defence shielding the real application, defending company assets at low maintenance and running costs.

Strategic honeypot positioning

Building on the current system, it may be possible to position honeypots in the network depending on the type and severity of an attack. By strategically positioning honeypots, attackers may be engaged longer and at a deeper level of interaction as the attack progresses.

Honeypot as code

A logical step in automated honeypot deployment in the cloud is to take advantage of infrastructure as code solutions offered by cloud platform providers. AWS CloudFormation is an infrastructure automa-

tion service that lets you deploy resources in a repeatable and predictable manner using templates. Like the rest of the infrastructure, honeypot infrastructure can be defined as code. The web, database and application servers, VPC network components are all defined as code, in JSON or YAML. This approach automates and simplifies resource management across the entire cloud infrastructure, maximising honeypot functionality.

Using Lambda as honeypot servers

It is possible to build serverless websites using Lambda in AWS. In this scenario, Lambda functions are the glue between AWS S3 for static content storage and DynamoDB, and the AWS managed NoSQL database system. This solution may further reduce costs and the need to provision, start and terminate honeypot servers using the EC2 service. Furthermore, Lambda functions can be used as a target group member for the Elastic Load Balancer. However, it may require more maintenance, and believability is questionable as attackers will not be interacting with a copy of a real server.

Biographies

Ivan Beres is a database and information security engineer at Gallup who has recently completed an MSc in Information Security at Royal Holloway University of London. His research interests are cloud security, security management and honeypot systems.

Darren Hurley-Smith is a Lecturer in Information Security, in the Information Security Group of Royal Holloway University of London. His research interests are statistical testing of Random Number Generators, RFID/NFC Security, Mobile Ad Hoc Network Security, and Edge-Cloud integrated mobile devices. He also has a keen interest in investigating moving target defence, and cyber-security related to cloud-implemented services.

Series editor: S.- L. Ng