Hardening AWS Environments and Automating Incident Response for AWS Compromises

Abstract

Incident response in the cloud is performed differently than when performed in on-premise systems. Specifically, in a cloud environment a responder can not walk up to the physical asset, clone the drive with a write-blocker, or perform any action that requires hands on time with the system in question. Incident response best practices advise following predefined practiced procedures when dealing with a security incident, but organizations moving infrastructure to the cloud may fail to realize the procedural differences in obtaining forensic evidence. Furthermore, while cloud providers produce documents on handling incident response in the cloud, these documents may fail to address the newly released features or services that can aid incident response or help harden cloud infrastructure.

In this paper, we detail the evidence an incident responder should collect and the mitigations that should be preformed for an AWS instance (host) or key compromise. We introduce ThreatResponse, a suite of tools to automatically collect this evidence, audit an AWS configuration for lapses in security posture, mitigate the host or key compromise, and provide a web based workstation, tailored to managing AWS compromises.

Authors & Contributors

Andrew Krug is a Senior Software Engineer at a large cyber security company. Krug has been Consultant, Network Architect, Systems Administrator, Operations Manager, Technical Trainer, and Software Engineer. Currently Krug works to develop gamified security education through security simulation scenarios.

Alex McCormack works in the design and implementation of capture the flag (CTF) competitions and training events. Alex has designed CTF challenges since 2013 and presented training since 2012. Prior to developing CTFs, Alex worked in Incident Response and Malware Analysis.

Joel Ferrier is the creator of Margarita Shotgun. Joel currently works as a Systems Administrator. Previously Joel worked as a Systems Engineer with a focus in Security Operations.

Jeff Parr is the Frontend Guru for the project. He has been developing web technologies for over 15 years; primarily Rails for the last 5. Parr has been a subcontractor through engineering/design firms, innovation consultant, and more; specializing in pairing, reviewing code, and contributing to open source.

Hardening AWS Environments and Automating Incident Response for AWS Compromises

Incident Response in AWS

At first glance, incident response within AWS may appear more challenging than incident response in an on-premise environment. Not only are servers not physically close by, but techniques associated with virtual machines such as taking a full disk and memory snapshot are not available.

However, by leveraging the APIs provided by the AWS SDK, organizations can prepare themselves to automatically collect evidence and mitigate compromises of AWS instances. Before implementing such a solution, the organization should understand what mitigations it needs to perform, and what evidence it needs to collect.

Mitigations to Preform

In March of 2016, Toni de la Fuente wrote a blog post detailing AWS steps to perform to collect evidence and mitigate a compromise. The mitigations included isolation, tagging, and shutting down an instance.

Isolation consists of creating a security group with exceptionally prohibitive access rules. Outbound traffic should be blocked completely, and inbound traffic should only be allowed by the specific IP address of the examiner.

The instance should then be tagged with a case number for record keeping and to alert other users that this instance should be treated with care.

Evidence is then collected and the instance is shut down.

Evidence to Collect

Two of the most important pieces of evidence to collect during an incident are disk and memory images. Disk image preservation is important because on the disk of a compromised host there may be host specific logs detailing what happened. There may also be copies of malware or other artifacts left by an attacker. Some attackers will alter the code of web applications to insert back doors, or collect sensitive data. Without forensic disk data, it may be impossible to determine what an attacker did after gaining access to the system.

Memory analysis is increasingly becoming a critical technique for forensic investigations. Memory analysis can be used to collect malware that may have been deleted from the disk, or never written to the disk in the first place. Memory analysis can be used to collect commands typed into a shell, discover programs hidden by rootkits, and much more.

Hardening AWS Environments and Automating Incident Response for AWS Compromises

In addition to disk and memory evidence, there are many AWS specific data points that may aid in an investigation. Metadata of an instance will reveal the public and private IP addresses of an instance, and the associated security groups. Console output and console screen shots may provide debugging messages from crashed services. VPC Flow logs may illustrate where an attack came from, and the destination of exfiltrated data. Finally, logs from CloudTrail may provide insights into actions performed by IAM users.

Automating IR with ThreatResponse Tools

Nothing can make handling an incident more frustrating than not having a plan to follow. Without a plan, responders may accidentally delete or fail to collect important evidence and inadequately remove access from the attacker. This could lead the responder to fail to understand what happened, and therefore prevent the attacker from getting back in.

Having an incident response plan can greatly reduce the stress of responding to an incident. Responders can walk through the plan and know they are doing the right thing. However, following the plan still introduces the risk of human error, as a responder may skip a step or perform sensitive data acquisitions out of order.

By automating the collection of evidence and compromise mitigations, organizations can be fully prepared should a compromise occur. To help organizations with this automation, we are releasing four distinct tools.

Margarita Shotgun: Capturing Memory from AWS Instances

Margarita Shotgun is a python module and a standalone command line tool that automates the process of aquiring memory from remote systems, both on premise and in Amazon Web Services. Margarita Shotgun makes heavy use of paramiko to securely connect to remote systems and secure memory in transit between the comprimised server and the incident responder workstation.

Margarita Shotgun makes use of the LiME Project to capture memory. A configurable repository of prebuilt LiME kernel modules is availible to streamline the memory aquisition process.

After determining the remote system's kernel version and architecture the appropriate LiME kernel module is loaded and system memory is streamed over an ssh tunnel to the incident responder's workstation. Memory can be saved to disk or streamed directly to an s3 bucket.

Memory aquisitions are preformed in parallel with the help of Python's multiprocessing library, decreasing the period that compromised instances must be left running.

AWS-IR: Automatic Evidence Collection and Mitigation

AWS-IR is a python module and standalone command line tool that can be used to collect evidence, mitigate compromises, or start an AWS specific incident response workstation. The command line tool has three subcommands: `host_compromise`, `key_compromise` and `create_workstation`.

AWS-IR is built on top of boto3, an AWS SDK for the python language. In order to use AWS-IR, a user should set up their environment for use with the AWS SDK or have the appropriate credentials in place.

When using one of the compromise commands, AWS-IR collects evidence and tags instances according to a case number. The case number can be provided with the `-n` flag, and if one isn't provided, a case number will be randomly assigned. The case is the basic organization of evidence and logging for a particular incident. All evidence collected by AWS-IR is stored in an S3 bucket for that specific case. Additionally, every action performed by AWS-IR is logged and stored with the case.

When the `host_compromise` subcommand, AWS-IR first starts collecting evidence and then mitigates the affected host. To use the `host_compromise` command, the user only needs to provide an IP address. An SSH username with root privileges and SSH key should also be provided if that information is available. AWS-IR will then start mitigating the compromised host by attaching a new, highly restrictive security group to the instance. This is designed to sever any existing session an attacker may have and stop the exfiltration of data. AWS-IR will then snapshot all attached volumes, attempt to capture memory, collect metadata about the instance, grab console output and save a console screenshot. After the evidence has been collected, AWS-IR will shutdown the instance and provide the case number.

When the `key_compromise` subcommand is used, the user must provide the AWS access key id of the compromised key. AWS-IR will locate the IAM account associated with the key and disable the key. The `key_compromise` subcommand should be used with the `-c` flag to automatically create a workstation so the user can perform more analysis to see what actions may have been taken with the compromised key.

Users can load the evidence of a case into an incident response workstation by using the `create_workstation` subcommand. Additionally, by specifying the `-c` flag, with either compromise subcommand, a workstation will automatically be created for the responder.

ThreatResponse-Web: An Incident Response Workstation for AWS

Incident responders can use the ThreatResponse-Web incident response workstation to both analyze the collected evidence, or collect additional evidence from other instances if the analysis leads them that way. After connecting to the workstation, a dashboard is displayed. The dashboard will illustrate the AWS regions instances are running in, recently launched instances, and the different types of AMIs those instances are running.

From the workstation a responder can take advantage of a full-text search to find other instances by AMI-ID, IP address or availability zone. If the responder determines another instance needs to be processed, that instance can be added to the case right from the workstation. When an instance is added to a case, either from inside the workstation or within AWS-IR, the workstation provide memory, disk, and configuration analysis capabilities.

The memory view allows the responder to analyze a memory dump using the volatility memory forensics framework. By leveraging a Javascript terminal, a volatility shell is provided inside the workstation. This allows the full feature set of volatility, and results can be copied and pasted out of the shell.

The disk analysis view leverages a full data analysis pipeline to automatically process the disk images collected. When a disk is analyzed, an EC2 instance is created and the snapshot of that disk is mounted as a volume to that instance. Plaso's Log2Timeline is then run on the attached volume to collect well formatted log files into a single `.plaso` file. Finally, TimeSketch reads the file to present a web view to the responder for further inspection. By leveraging AWS, multiple disks can be processed in parallel.

Finally, a view is available for an interactive configuration checking tool called ThreatPrep.

ThreatPrep: Preparing your environment for optimal evidence collection.

ThreatPrep is a tool to examine an AWS environment with two main objectives. Firstly, identify areas where the security posture could be increased and secondly, identify areas where the amount of forensic evidence could be increased.

The following items are currently checked by ThreatPrep:

- S3 Buckets have versioning and logging enabled, and do not allow public reading or writing.
- IAM users have MFA enabled, credentials have recently been rotated, and users are not attached to the AdministratorAccess policy.
- VPCs have flow logs enabled.
- Billing alerts are enabled.
- Multi-regional CloudTrail is enabled.

ThreatPrep can be used either from a command line, or used in a python project. In fact, the ThreatResponse-Web project includes output from ThreatPrep in the advise section.

ThreatPrep offers similar advice as AWS Trusted Advisor. Trusted Advisor is a service that checks for both security and performance improvements that could be made in an AWS environment. Trusted Advisor is currently only available in the console and can not be accessed programatically. Trusted Advisor is provided through the support plan of an account. AWS Trusted Advisor provides a few checks for accounts with the basic support plan, but the full set of checks is only available to Business or Enterprise support plans.

ThreatPrep and AWSConfig

ThreatPrep also shares similarities to AWS Config and Config Rules. Config deals with logging configuration details, called a *configuration item* for supported AWS resources whenever that resource is created, deleted, or changed. AWS Config rules is a distinct offering from Config. Config Rules is responsible for evaluating the configuration item against a set of predefined criteria and then alerting AWS users if that criteria is not met. AWS provides a set of configurable rules users may use, as well as the ability to make custom rules. Many of the predefined rules cover similar areas as Trusted Advisor. The price to store a configuration item is about a third of a cent so the cost of enabling AWS Config is unlikely to cause significant price increases. AWS Config Rules cost two dollars per month per rule, and possibly more depending on how many times the rule is evaluated.

Limitations and Future Work

AWS-IR's memory collection has focused exclusively on Linux based images. While most of the evidence collection is independent of the Operating System, the memory capture process assumes a Linux kernel.

Hardening AWS Environments and Automating Incident Response for AWS Compromises

In addition to incorporating a windows based memory collection procedure, future plans also include incorporating threat intelligence feeds, automatic processing of common volatility modules, and supporting plugins.

The ThreatResponse team encourages anyone interested in developing, contributing to, or providing feedback or ideas to connect with us on github or send us an email at info@threatresponse.cloud.

References and Prior Work

[blyx]: http://blyx.com/2016/03/11/forensics-in-aws-an-introduction/
[volatility]: http://www.volatilityfoundation.org/
[log2timeline]: https://github.com/log2timeline/plaso
[timesketch]: https://github.com/google/timesketch
[threatresponse.cloud]: http://www.threatresponse.cloud
[tr_github]: https://github.com/ThreatResponse
[boto3]: http://boto3.readthedocs.io/en/latest/
[aws_sdk_install]: http://docs.aws.amazon.com/cli/latest/userguide/installing.html
[LiME]: https://github.com/504ensicsLabs/LiME
[paramiko]: https://github.com/paramiko/paramiko