541.5
Automated Response Actions and CloudWars

SEC541.5 | Cloud Security Attacker Techniques, Monitoring, and Threat Detection

SANS

Automated Response Actions and CloudWars

Welcome to SEC541.5: Automated Response Actions and CloudWars.

TABLE OF CONTENTS

This page intentionally left blank.

Course Roadmap

- Section 1: Management Plane and Network Logging
- Section 2: Compute and Cloud Services Logging
- Section 3: Cloud Service and Data Discovery
- Section 4: Microsoft Ecosystem
- **Section 5: Automated Response Actions and CloudWars**

Microsoft Ecosystem

1. **Automated Response Actions**
2. IT Ops Workflows
3. Security Workflows
4. EXERCISE: Set Up AutoForensic
5. Constructing Response Actions
6. EXERCISE: Run AutoForensic
7. CloudWars

This page intentionally left blank.

Automated Response Actions (1)

Automation is a driver for organizations to enter the cloud space.

- Faster deployment
- Manage elasticity
- Reduce repetitive work
- Reduces errors (maybe)
- Support multi-account deployments

Automation tends to focus on infrastructure operations. In this section, we will look at automation to support security operations.

When it comes time to move to a cloud service provider, a main driver is the ability to build infrastructure when needed, on demand. Some companies only use the basic automations built into the platform, while other companies are automation driven, willing to automate any activity that has to be done more than twice. Automation provides a number of benefits to the corporation.

Deploy new and updated applications faster using an approved and prescribed methodology. This is the business driver of DevOps. DevOps is a combination of culture, practices, and tools that drive deployment to "high velocity". Imagine deploying changes to your production applications daily or even hourly? That requires a fully automated process.

Elasticity was the first business driver for the cloud environment. The idea that during the day you might need ten servers, but only three at night, requires that the environment shift and shape to needs. Cost becomes important. We are charged for what we use, so we want to make sure we are only running what we need. Entire industries have grown up around cloud cost analysis and consulting. Even the cloud providers themselves will help you reduce that unnecessary bill. Elasticity requires automation without human intervention, and the ability to scale based on load, or a schedule, or maybe only for a few days that a project is alive. There are two types of elasticity/scaling: horizontal and vertical. Horizontal scaling is increasing or decreasing some characteristic of a computing resource, such as memory or CPU. In practice, this type of scaling is used to right size the execution of a compute workload. Maybe you wrote an application, and you may need to increase/decrease the memory to ensure it runs without a lot of waste. The cloud providers have tools to make this fairly simple, however real automation is generally focused on horizontal elasticity. Going from ten servers during the day to three in the evening when the load goes down is a type of automated elasticity. The ability to automate the creation, configuration, and destruction of cloud resources will be helpful.

System administrators have been automating tasks with scripting languages and automation tools for years, but those tools were focused on the configuration of the virtual machine itself. The cloud APIs open a whole new world of automating the deployment of the resources themselves. Beyond virtual machines, this can include log collectors, storage resources, and even entire networks.

Automation may reduce errors, but they also might contribute to them. This is less about the automation itself than it is about the environment around it. It is difficult to properly automate the build and deploy and monitor the virtual machines properly if every development team gets to pick their own flavor of operating system. When considering automating, we will have to make changes to how other parts of the business operates.

In the olden days, the care and feeding of a database would take considerable time and energy. Doing it once was enough. But spinning up a new database for a new developer would be too much. By using automation and leveraging cloud environments, deploying one takes the same amount of effort as deploying many. This opens opportunities to have the same or similar infrastructure deployed in multiple locations. Each development team could have a smaller version of the production system. Each product team could get their own separate environment just for them. This type of segmentation also reduces the blast radius, application of different controls on production than in development, and allows teams to destroy unnecessary environments.

These operationally focused automations are where most organizations start. We will take those lessons learned and focus on how security operations, especially threat detection and monitoring processes, can be dramatically improved.

Automated Response Actions (2)

NIST Definition of Cloud Computing

- On-demand self-service
- Rapid elasticity
- Measured service

Enables automation of everything (nearly everything)

- Broad network access
- Resource pooling

Automation of a cloud vendor's resources uses similar method

Measure services means response actions are possible

As we saw in Section 1 of this class, there are five main characteristic of a cloud service, as defined by NIST. We investigated how a cloud vendor's API control plane gives us a unique ability to monitor activities happening in a cloud environment.

That control plane API automates elasticity of infrastructure and supports an on-demand self service of the environment.

Every resource that is configurable through the control plane, which is just about every resource, is automatable. The big cloud providers provide software development kits (SDK) to make it easier to write interactions in a particular language.

Command line tools are also helpful, enabling scripts to handle repeatable tasks. The authors of this course wrote a script that would build all class labs in order to test the virtual machines. (It didn't always work, but it certainly saved us some time.)

The cloud providers may also provide built-in services to help with automation.

AWS has CloudFormation, a separate service that takes a YAML or JSON file and interacts with the individual service APIs to manipulate services. Azure's Resource Manager has a similar purpose, but it is more tightly coupled to the API itself. Instead of another intermediary, the Resource Manager is the resource control plane, and is the endpoint from CLIs, SDKs, and JSON configuration files.
A cloud service must be a "measured service", using a metering capability and some abstraction to manage the resource activity. We as the consumer get access to that data, enabling a wide variety of response actions to just about any action in the environment.

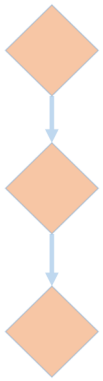
A cloud service will create a model for interacting with all of its services that should feel similar across the platform. The Azure CLI commands for creating a virtual machine are similar to the commands to create a blob store. This makes it easy to spin up on automation of complex services. In more traditional infrastructure, different "teams" were set up for networking, servers, and workstations because the skills to build and manage them were very different. Cisco switches and Windows 10 workstations are very different. With the cloud's common API methodology, we start to rethink about how we build and deploy these services.

References:
https://www.nist.gov/system/files/documents/itl/cloud/NIST_SP-500-291_Version-2_2013_June18_FINAL.pdf
https://aws.amazon.com/cloudformation/
https://docs.microsoft.com/en-us/azure/azure-resource-manager/management/overview
https://docs.microsoft.com/en-us/azure/virtual-machines/windows/quick-create-cli
https://docs.microsoft.com/en-us/azure/storage/blobs/storage-quickstart-blobs-cli

Automated Response Actions (3)

There are IT operations focused response actions we are building in the cloud. They tend to follow a similar flow:

- Some pre-defined activity or metric is observed in the environment, and a trigger is activated
- The trigger is predefined to perform an action in the environment, either *in* the resources, or in the control plane *with* the resources
- The action is tracked and reported



Next, we look at operational workflows that can be the basis of security focused workflows.

Any automated workflow tends to follow a similar pattern when looked at from a 10,000-foot view, although the details can get messy. When planning to build an automated workflow, it helps to think of these three groups separately, but connected.

First you have to decide on **what** will trigger your workflow. A pre-defined activity or metric that is observable, reliable, and repeatable. The triggering mechanism must have access to that data in a timely manner. It must understand the data and be able to determine when execution is appropriate. The cloud environment has services to help with this, including providing a common data format for logs and events.

The fun part of this is the actions. These actions could be against the control plane itself, such as spinning up another virtual machine. They could be against the resources, such as clearing out old objects from the managed object store.

Finally, the entire process, from trigger through action, to result, must be tracked and easily available. Humans may need to review an automation, especially if it fails. In a cloud provider, the workflow may be engaging multiple services, each with its own data logging approach. All of that data should be brought together and easily available. The other part of tracking progress is being able to detect when an automation fails, to what extent, and how the automation should handle that failure. Sophisticated ones may be able to roll back safely, while others may just have to throw up their digital hands and wait for human investigation.

Course Roadmap

- Section 1: Management Plane and Network Logging
- Section 2: Compute and Cloud Services Logging
- Section 3: Cloud Service and Data Discovery
- Section 4: Microsoft Ecosystem
- **Section 5: Automated Response Actions and CloudWars**

Microsoft Ecosystem

1. Automated Response Actions
2. **IT Ops Workflows**
3. Security Workflows
4. EXERCISE: Set Up AutoForensic
5. Constructing Response Actions
6. EXERCISE: Run AutoForensic
7. CloudWars

This page intentionally left blank.

Automated Response Actions (4)

Example 1: Elastic deployment of Virtual Machines

You are probably in the cloud so you can spin up resources when they are needed. Virtual machines tend to be the first automation.

- Monitoring determines CPU utilization has hit 70%.
- Automation is triggered, new VM is spun up and is healthy.
- What is "healthy", and how can that be automated?
- When CPU hits below 50%, the VM is spun down.

When an application is developed for elasticity, the security team now has more freedoms to respond to suspicious activity.

We already talked about elasticity being a core characteristic of a cloud service. That concept of elasticity drives companies to make the first leap into transitioning to cloud services. Virtual machine elasticity is implemented through the AWS Auto Scaling or the Azure Autoscale service.

First, we have to figure out when we would scale and by how much. Scaling could happen based on network traffic, memory utilization, CPU spiking, or even on a pre-defined timer. Knowing the application and how to respond to different loading scenarios is a must. Luckily, even that is automatable in the cloud.

For a web server, you may know that the server can only handle a certain amount of network traffic. A common metric is to monitor CPU utilization on the virtual machine. If the CPU utilization is 70% or more for five minutes, then bump up the number of virtual machines.

But it's not as simple as just spinning up a virtual machine. Looking deeper at the process, there are a number of steps that have to be accomplished. The auto scaler will start a new web server, but it takes a few minutes for the VM to finish coming online, be configured, and have the application start running. The developers need to devise some kind of http(s) accessible "health check" that will respond when the application is healthy. That also means the developer needs to build internal checks to determine if internal services are healthy. If your application connects to a database, should the health check also include a database connection test? We bring this up to demonstrate that building automation is not just about the cloud control plane but will also drive how the applications themselves are architected.

The scaling service will then need to know when to shut a virtual machine down. Picking the right metrics for kicking off workflows requires knowing the application and how it is used.

When an application supports creating and destroying virtual machines at will, it opens up a whole new avenue of security workflows. In traditional infrastructure, if there is an incident that needs to be investigated, the virtual machine in production still needs to operate and the investigator has to take care as they evaluate the server. Only when it is a confirmed breach is the virtual machine removed, and only if the business unit agrees. With our automated workflow, virtual machines can be pulled out of service for investigation without thinking twice.

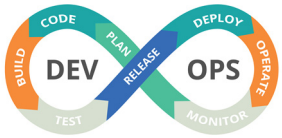
As we move through these operational workflows, think about how an operational improvement can also enable a security improvement.

References:
AWS Auto Scaling: https://aws.amazon.com/autoscaling/
Azure Autoscale: https://azure.microsoft.com/en-us/features/autoscale/
AWS Health checks: https://docs.aws.amazon.com/elasticloadbalancing/latest/network/target-group-health-checks.html
Azure Health checks: https://docs.microsoft.com/en-us/azure/app-service/monitor-instances-health-check
Secure image pipelines talk by course author Shaun McCullough:
https://www.youtube.com/watch?v=uamCoMY7cac

Automated Response Actions (5)

Example 2: DevOps Application Deployment

DevOps-first organizations automate deployment of new app versions. Requires automation of testing, application deployment, and publishing the new version.



Automation of testing can include security testing. Codify those security policies to be run daily, hourly, at every code change.

As we discussed, DevOps changes all aspects of how applications are built and deployed. This is a quick summary of the process:

- A workflow receives an alert that a new version of the application is ready for deployment.
- A series of automated tests take place. Local unit tests of the code itself, integration tests in a realistic environment, and stress tests ensure the new update won't crash the production environment.
- After tests, then the change must be deployed to production.
- After deployment, the service needs to be monitored. Is it operating properly? Is the infrastructure responding properly?

There are a couple of places that the DevOps workflow can have added security requirements. The first place is with the testing. Not only testing to see if the application operates as expected, but what about testing to see if any credentials were accidently left in the code? Or evaluating the app's included libraries to determine if a third-party library is vulnerable to some CVE? Security policies and procedures can be codified in the pipeline, rather than half-heartedly written on some word document.

DevOps also works to move the creation of infrastructure and testing into the developer's hands. No more developers passing a tool to some other group to click through it. Adding security checks to the developer's own test suites will help the developer create more secure code from the start.

These gates are built to ensure that only properly tested code can be deployed. It is up to the team to ensure that those tests are thorough. New tests can be added at any time. The workflow is also able to stop and roll back deployments that do not meet tests, or do not perform as expected. AWS and Azure have services to help with that continuous testing of live production systems.

DevOps for AWS and Azure:
https://aws.amazon.com/devops/what-is-devops/
https://azure.microsoft.com/en-us/overview/what-is-devops/

For additional training in DevSecOps automation, check out SEC540: https://www.sans.org/cyber-security-courses/cloud-security-devsecops-automation/.

Some third-party library testing tools:
https://www.blackducksoftware.com/
http://www.sonatype.com/
https://snyk.io/
https://www.owasp.org/index.php/OWASP_Dependency_Check
https://nodesecurity.io/

Automated Response Actions (6)

Example 3: Log collection

Use the deployment an automated workflow to implement logging inside resources, such as virtual machine's AWS CloudWatch agent, Azure Monitor agent, or rsyslog.

Ensure logging is always configured when each new system is turned on, removing configuration from a human.

We did this in our CDK applications and Terraform builds in the labs.

As we just saw, security policies can be codified if we implement automated testing at the right place. Proper configuration of assets can also be automated in the deployment workflows.

If virtual machines are deployed at a moment's notice, then there should be little to no post-deployment configuration required. Just-in-time configuration of a new system slows down the time to get a VM into service, and it increases the potential for post deployment failure significantly. Many times, the author's image build process has failed because it relied on some external configuration file that was suddenly unavailable.

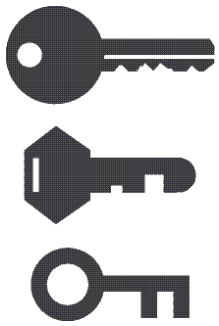
Every organization should have a policy and procedures for log collection, forwarding, analysis, and retention. As we discussed in Section 2 of this class, the cloud providers have their own agents that can get data from inside a resource and into the cloud service's eco system.

Consider this scenario. An application analyzes JSON files dropped into an S3 bucket and performs a transformation of that file. If the files are malformed, then an error needs to notify a human. Application logs from all deployed application resources must be pulled into a centralized resource.

In addition to logs from the resources, there are logging configuration considerations with the cloud accounts themselves. Consider creating an automated workflow that deploys an account when a new teams is formed. The security team can ensure that cloud API logging is deployed, tuned properly, and ships the logs to the security team's own account.

Automated Response Actions (7)

Example 4: Rolling Secrets



In DevOps, code and scripts manage deployments. If short term tokens are not possible, then a username/password or other long-term credentials needs to be stored in a key store, such as AWS Secrets Manager, Azure Key Vault, or HashiCorp Vault.

Credentials must be changed and redistributed without breaking the applications. Automating regular rotation, we can rotate when a security emergency arises.

Nothing induces panic in a developer faster than when they realize they checked a test script into GitHub with a username and password in it. Bots are digging through popular code sharing services looking for these infractions. As we saw in example 2, we can incorporate tools to help catch this.

Even better is to never have credentials, especially long-term credentials like a username/password, in a person's environment. For applications that can support short term tokens, move to them. For others, the username and password should be retrieved automatically when needed.

Imagine a docker container that must log into a database. Keeping the credentials out of the local file system, or provided on startup, means the docker container must go pull those credentials from a vault. AWS and Azure have their own vaults. Many companies rely on HashiCorp Vault, which can be run in multiple cloud and on-premise environments.

Credentials need to be rotated regularly. Build a workflow to automate new credentials and deploy those new credentials to the vault. Then wait for all in-process applications to grab the new credentials and retire the old credentials.

The workflow will be built and tested thoroughly. Once ready and deployed, the organization has reduced the risk of credential rotation. It can be done frequently, maybe monthly, to ensure credentials are always fresh.

When we automate this operational workflow, it is no longer a chore to rotate credentials. If done monthly, we can do it immediately if a breach happens, and credentials are somehow compromised. Use the exact same workflow for operations and security.

Vaults:
https://aws.amazon.com/secrets-manager/
https://docs.microsoft.com/en-us/azure/key-vault/general/overview
https://www.vaultproject.io/

Course Roadmap

- Section 1: Management Plane and Network Logging
- Section 2: Compute and Cloud Services Logging
- Section 3: Cloud Service and Data Discovery
- Section 4: Microsoft Ecosystem
- **Section 5: Automated Response Actions and CloudWars**

Microsoft Ecosystem

1. Automated Response Actions
2. IT Ops Workflows
3. **Security Workflows**
4. EXERCISE: Set Up AutoForensic
5. Constructing Response Actions
6. EXERCISE: Run AutoForensic
7. CloudWars

This page intentionally left blank.

Automated Response Actions (8)

The same tools and approaches to automating operational workflows can be applied to security and forensic data collection.

- Pre-defined activity or metric must be observable. Are there risks of false positives? What could that cause?
- Is the action going to be destructive? Does it require a human in the loop to approve?
- How will the action be reported? Can we automate interaction with security investigation teams?

Let's look at three scenarios of more security-minded workflows.

While looking at the operationally focused workflows we might build in our environment, we briefly discussed their security counterparts. We will now look at security-based automated workflows and how we build them.

Just as with operational workflows, security workflows need the right data with the right fidelity to determine when to launch the automation. This could be trickier than the operational workflows we have discussed. Data may need to come from inside the resource, from the cloud service logs, or a combination of both.

An atomic indicator, such as a bad IP address, is easy to detect, but requires a constant update of known bad indicators from some cyber threat intelligence feed.

Behavioral indicators tend to involve data from multiple sources, an analysis of data over time, or customized detections based on the application itself.

As with operational workflows, it is easier if the environment is homogeneous, and the resources are all the same. It would be very difficult to automatically configure all the virtual machines if each product team used its own unique operating system. The same applies with trying to detect and automatically remediate a security problem. If every product operates differently, it would be difficult to detect problems.

The author likes to bin response actions into three categories.

- Alerts: Identify an activity and send an alert or create a JIRA ticket for the security team. AWS and Azure have services focused on security detection and alerting, but most organizations will want to customize beyond that.
- Data Collection: Most security alerts are about a single dimension of data. The web server communicated with a known bad. When the security team takes the time to investigate, they start gathering related information. Automating that data gathering at the time of the event, grabbing logs, creating inventory reports, or even installing or configuring log collection agents can increase fidelity of data captured.
- Destructive: Our goal is not only detecting a security event, but to automate the remediation of the resource. Stop a suspicious resource, roll back an application that uses vulnerable third party, or quarantine the resource for analysis. This destructive workflow can have serious consequences to the company's production environment, so tread lightly. Maybe build up to this one.

Automated Response Actions (9)

Example 1: AWS bucket or Azure Storage resource is on the internet with sensitive data

This is bad and happens regularly.

What conditions would allow us to detect this automatically?

What actions would we want to take? Corrective or investigative?

Data leaks have been notorious in cloud environments, especially in AWS. AWS has increased the checks and balances of S3 buckets so accidental exposure is less likely today, but buckets, databases, ElasticSearch consoles, and Kubernetes control panels have been exposed.

How do you determine if an S3 bucket is accessible on the internet and should not be? It might be legitimate deployment. Is the data inside sensitive? Tools and services, such as Macie, investigate S3 buckets for common data formats like a social security number. But what about S3 buckets that contain code? Can an automated tool tell if that is public python programs meant for a client, or the company's core application? That is harder to do.

What are some automated response actions for S3 data exposure workflows that we might be interested in implementing?

- Scanning all S3 buckets in all accounts and reporting ones that are open may be noisy. That could be done regularly to see if there are any unknowns.
- When a new S3 bucket is deployed and exposed, or an S3 bucket configuration has changed to become exposed, an alert is created and sent to the security team with details gathered about that new deployment.
- Some organizations will set aside specific AWS accounts, or Azure subscriptions, in which public cloud resources reside. An S3 bucket in this account may reasonably be assumed to be public. However, the development team's account/subscription should never deploy an exposed S3 bucket. Because we have created this sandbox to separate exposable from private resources, we could implement a response action that will change the configuration of an exposed S3 bucket from that development account in near real time.

Many companies allow a single team to own the entire development and operations-the promise of DevOps. Without clear guidelines and consistent best practices in place, the automation of security becomes difficult. A balance has to be reached.

AWS Exposable Resources from SummitRoute:
https://github.com/SummitRoute/aws_exposable_resources

Automated Response Actions (10)

Example 2: Access token found in GitHub

We have looked at this story in the class. DevOps means everything must be in code or configurations, and secrets can slip in.

Tools like Microsoft's "detect-secrets" can trigger a secrets rollover and redistribution.

The response action could look like our rolling secrets from a few slides ago.

When the operations of a product is managed by the development teams, then deployment processes can start to fragment and be unique to every team. Security definition policies without providing solutions will make life harder for everyone.

When we can take an operational workflow, such as rolling secrets, and leverage it for security workflows, we have a better chance of getting buy in from the teams.

In our rolling secrets workflow, the trigger was a schedule. For the security workflow, the trigger is now a detected or reported security event. Attackers are quick to grab the data from a GitHub project and make fast use of it. Speed is of the essence. There are a number of tools that scan repositories and files for credentials of all kinds and can be augmented with your organizations unique credential data formats.

The goal of this workflow is to build it for operations but use it for security. In this case, the trigger needs to be somewhat separate from the action itself.

Reference:
https://microsoft.github.io/code-with-engineering-playbook/continuous-integration/dev-sec-ops/secret-management/recipes/detect-secrets/

Automated Response Actions (1 1)

GuardDuty alert: Detection of a VM hitting a known bad domain

We learned that GuardDuty can analyze VPC flow data, detecting connection to known bad sites.

We could shut down the system, or we can gather data for our investigators.

The control plane is available to our automations. SSM is pre-deployed so we can automate on host actions. We have logs.

In this last example, we will leverage a purpose-built security tool in AWS to detect an activity and kick off the workflow. As we say earlier in the class, tools like GuardDuty pull together logs from multiple sources and will alert on known bad actions. GuardDuty alerts can be noisy and may need to be tuned properly, but there are critical events that GuardDuty detects that we could certainly create workflows from.

As an example, GuardDuty has its own set of known bad domains it looks for in VPC Flow Logs and will create an alert. This is a specific and well understood problem. We may wish to implement a destructive workflow and stop the instance.

Compare that to receiving a GuardDuty alert that one of your VMs is initiating an SSH brute force attack. Is it really an attack? Is it a script that is stuck in a loop? Automatically destroying the system may not be warranted but gathering more data automatically would be helpful. When was it deployed, by whom, and what are the name and hash of all executables and scripts on the host machine?

Standing down the virtual machine will require interaction with the AWS control plane. Gathering the names of all the executables could leverage the SSM agent. Maybe a third-party automation service is leveraged.

We will take this idea of responding to a GuardDuty alert and build a real workflow in our lab.

Course Roadmap

- Section 1: Management Plane and Network Logging
- Section 2: Compute and Cloud Services Logging
- Section 3: Cloud Service and Data Discovery
- Section 4: Microsoft Ecosystem
- **Section 5: Automated Response Actions and CloudWars**

Microsoft Ecosystem

1. Automated Response Actions
2. IT Ops Workflows
3. Security Workflows
4. **EXERCISE: Set Up AutoForensic**
5. Constructing Response Actions
6. EXERCISE: Run AutoForensic
7. CloudWars

This page intentionally left blank.

Lab 5.1 | Set Up AutoForensic

Exercise Duration: 20 Minutes

Objectives

In our first lab, we will set up the AutoForensic workflow, and understand its main operation:

- Use CDK to build the AutoForensic workflow
- Review the origin of the workflow from Amazon
- Trigger the workflow through a GuardDuty alert

This page intentionally left blank.

Course Roadmap

- Section 1: Management Plane and Network Logging
- Section 2: Compute and Cloud Services Logging
- Section 3: Cloud Service and Data Discovery
- Section 4: Microsoft Ecosystem
- **Section 5: Automated Response Actions and CloudWars**

Microsoft Ecosystem

1. Automated Response Actions
2. IT Ops Workflows
3. Security Workflows
4. EXERCISE: Set Up AutoForensic
5. **Constructing Response Actions**
6. EXERCISE: Run AutoForensic
7. CloudWars

This page intentionally left blank.

Automated Response Actions (12)

When building a response action with an automated workflow, we typically look at four steps of the process:

- What will trigger the workflow?
- Does the workflow need outside resources or human intervention?
- What is the action?
- What follow up needs to happens?

Let's dive into these a bit and discuss the AWS and Azure services we might use.

The design of a workflow usually starts with a set of questions that has to be answered:

- What will trigger the workflow? What data, logs, or services do we have available that can fire when the right conditions are met?
- Does the workflow need a human to approve it, or does an outside service need to provide validation?
- What action do we want? How destructive? What will the effects on production services or access be?
- After the workflow has run, what is the reporting mechanism? What data is needed? How would we know if the workflow failed?

We will look at each of these questions and look at AWS and Azure services we may leverage.

Automated Response Actions (13)

In a security response workflow, the **trigger** will look for a particular condition, metric, or alert and start the execution of the workflow.

- Cloud providers have purpose build triggers, such as GuardDuty.
- The cloud service is metered, giving us data and telemetry about the environment. We can tap into that telemetry to detect problems.
- Will the trigger have false positives? Could a trigger occur over and over for the same problem?

Every workflow starts with a trigger—some condition that is observed, met, and kicks off execution of the workflow. That workflow may be as simple as sending an email, or as complex as taking a forensic snapshot of a virtual machine. No matter what the workflow, it starts with a trigger.

Cloud services provide triggering, or eventing capabilities, that may perform or support creating triggers. Services like GuardDuty will analyze data and create the event, from which you can create a trigger. Other services, such as Azure Monitor, will provide access and tooling to create your own triggers based on whatever conditions you wish.

The NIST cloud characteristics state that a proper cloud service is a "metered service", with details about most actions taking place in the environment, especially the cloud control plane. Nearly every new resource or resource configuration change is tracked, if we know where to look. Our trigger stage will make use of the control plan logs.

Our trigger may also leverage application telemetry that is not readily captured through the control plane logs. But AWS CloudWatch and Azure Monitor can use host agents to push those application logs into the cloud services log collection service.

Triggers must be tuned, monitored, and tested. A single failed SSH login attempt is not a brute force attack. The trigger for SSH Brute force may need multiple data metrics over a time period.

Let us take a public S3 bucket example. Say that when an S3 bucket is deployed, the workflow is triggered if that S3 bucket is created and that S3 bucket is accessible on the internet. If some S3 buckets **should** be accessible, then the filter needs more information to be able to ignore those; maybe a tag on the S3 bucket that says "Allow". Another option, which in the author's opinion is better, is to create AWS accounts specifically for public resources. All public resources have to reside in those accounts.

Let us look at some trigger services in AWS and Azure that we have not yet covered in class.

Automated Response Actions (14)

AWS EventBridge lets you respond to state changes in AWS resources.

- An **event** is generated by resources when actions happen.
- A **rule** catches the desired events.
- A **target** is a resource invoked by the rule.
- Event rules also trigger on CloudTrail events.
- An event rule can also be scheduled, rather than from an event.

AWS EventBridge is the service that helps you discover and manage a registry of AWS events across services and gives you the ability to connect external regions and SaaS services to your running environment. Previously called CloudWatch Events, the EventBridge service leverages the same services under the hood, but with additional discovery, registry management, and service integrations.

An "event" is a JSON object generated by most AWS services. The EventBridge service shows the schemas for **most** of the possible events. For some reason, GuardDuty and a few others are not visible in the EventBridge schema.

An event by itself is not interesting without a "rule". The event rule watches for a particular set of events, applies a filter, and looks for a match. The rule has an "event pattern"—JSON objects with properties to match the event to.

The event rule can invoke a "target". Targets are often Lambda Functions, Kinesis stream, or an SQS. EventBridge also supports targets of third-party SaaS services that are AWS Partners.

List of event types:
https://docs.aws.amazon.com/AmazonCloudWatch/latest/events/EventTypes.html

Event targets:
https://docs.aws.amazon.com/eventbridge/latest/userguide/eb-targets.html

Automated Response Actions (15)

Azure Event Grid routes events to services for execution.

- An **Event** is data traveling through Event Grid.
- **System Event Topic** is an Azure pre-defined event source. A **Custom Event Topic** can be an application or third party.
- **Event Subscription** tells the grid which topics to grab, with filters based on event type and a pattern.
- **Event handlers** is the destination of the event that will process the event.

Event Grid is built with custom events in mind, but for monitoring the changes in resources, we will use **system event topics.**

The Azure Event Grid is built on Azure Service Fabric and facilitates receiving events from Azure and custom sources and handing them off to other services. Event Grid is Azure's serverless event-driven service. It has the same goals of AWS Event Bridge, but with a bit more focus on customized events than maybe Event Bridge supports.

Events identify that an action has taken place, or that a service is notifying another service with data.

A publisher is a user or organization that decides to send events to the Event Grid. Microsoft is the publisher for Azure services, and you can create your own customized events from your applications as a publisher.

Event source is the service or resource that generates the event.

The topic is the endpoint that a publisher creates. The event source sends events to the topics. A System Topic is a built-in topic by Azure for things such as Azure Storage. They are the Azure generated events.

The event subscription tells event grid which events on a topic you are interested in receiving. It is the filter of events by type and subject pattern.

References:
https://azure.microsoft.com/en-us/services/event-grid/
System topics: https://docs.microsoft.com/en-us/azure/event-grid/system-topics
Subscription schema: https://docs.microsoft.com/en-us/azure/event-grid/subscription-creation-schema

Automated Response Actions (16)

There are times we may want a workflow to happen automatically.

But for some workflows, ones that might be destructive such as deleting a VM, we may need a human in the loop.

- Is the human available 24/7?
- Could the conditions of the environment change while waiting for the human?
- The approval/disapproval needs to be tracked and documented.

Although we may want the trigger and the workflow to be automated, there are times we may want a human in the loop. Most orchestration tools allow an email response, or maybe ChatOps tools, to provide an event to the human and pause the workflow until a response is received. This is especially desired if the workflow is "destructive" or will make a change to deployed resources. For instance, our workflow could give us an alert that a virtual machine is performing an SSH brute force attack. Would we want to automatically delete that VM? What if it's a mistake, or this is a testing VM? Or a script has gone bad? The workflow could send an email and wait for the human to click the "approve" or "disapprove" link, which will trigger the workflow to continue.

Make sure that the following is considered when building a human in the loop:

- Is the human available when needed? Work hours, weekends, holidays, and snow days should be considered as well. If the trigger is over email, maybe the approver will not even see it.
- If the workflow is to quarantine a virtual machine, and the approval comes two days after trigger, is that virtual machine even available? It's also possible to break up the destructive workflow into two parts. The first part of the workflow gathers information, takes snapshots of machine images, and gathers the logs. Then, an approval is needed to quarantine the machine.
- Who is approving? Is it a single point of failure? Email approval is best left for customers who are not inside the company's internal services. Leveraging ticketing systems with built in nagging services, or ChatOps if Slack or other services are a primary communication tool. Management should know how responsive these approvals are, to ensure that the workflows are operating in a timely manner.

Automated Response Actions (17)

The actual response action is where we spend the most time. There are a couple of types of response actions we may consider.

Notification: This is a bit boring but creating alerts and sending data to a ticket system is usually the first good step. Good way to tune the trigger.

Data Gathering: We can gather data and store for human analysis, especially if resources tend to change with elasticity.

Corrective Action: Deleting a resource, changing an IAM policy, or deactivating access are tempting, but your trigger must be perfect.

The heart of the response actions can be incredibly simple, or they can be complex. They can also be destructive or have a light touch, generic to the whole company or specialized for certain product lines. Whatever the flow, it is recommend to start off simple and with a light touch, moving the workflow to more complex and consequential actions.

All workflows should start with a notification that the workflow launched, the trigger, and the resources involved. Notification may go to a dashboard, ChatOps, or Jira board. While some workflows, like VM scaling operations, may not need close scrutiny, security workflows tend to need to be investigated. Which brings us to data gathering.

With the measured cloud service giving us volumes of information about the resources and how they are deployed, workflows could be built to gather cloud control plane logs and resource information and present them to the investigators in a report. Evaluate what the security team does when it performs an investigation and see if you can build tools to help automate those services.

Corrective action tends to be more destructive when it comes to security workflows. A virtual machine that is communicating with known bad domains could be shut down. S3 buckets that are exposed to the internet can be blocked, cycling a potentially compromised credential.

Workflows should be run regularly to ensure they work, services they rely on are still functioning, and security teams know how to react to them. Build automation tests with the workflows, so they can run against non-production resources, especially for destructive workflows.

Let's look at a few AWS and Azure services we will be using to run in the workflow.

Automated Response Actions (18)

AWS Lambda is a serverless compute service that lets you run arbitrary code when an event occurs.

- Native support for Java, Go, PowerShell, Node.js, C#, Python, and Ruby.
- Can invoke from SNS, SQS, DynamoDB, and other services events. Event rules, S3, and Lambda functions can also invoke a Lambda function.
- Lambda facilitated the "event-driven" architecture design pattern.

Lambda functions security should follow least privileged practices.

AWS Lambda is a "serverless" execution of customer created code that can do almost anything you want it to. The code is uploaded as a Lambda function, and AWS handles the configuration and execution of that code. Think of Lambda as executing in a container on a VM, where you do not have to manage the container executions or the VMs at all.

Easily run code based on Java, Go, PowerShell, Node.js, C#, Python, and Ruby. AWS also provides a runtime API so you can create support for additional programming languages. Lambda functions can also run Docker images.

Lambda can be configured to be invoked directly from SNS, SQS, S3, DynamoDB, and other services based on an event. EventBridge, Step Functions, API Gateway, and Lambda functions can also invoke a Lamba function.

The input data is a JSON object. Many times, this is the event data passed directly to the Lambda function for execution.

Lambda functions, just like other services, should be deployed with Least Privilege in mind. A Lambda function could be used to read and write data to an S3 bucket. The IAM policy on that Lambda function should be to read and write to only that S3 bucket, rather than read and write to all S3 buckets. Infrastructure as Code services such as Terraform and CDK make this a bit easier to manage when deployments get big with hundreds of Lambda functions.

References:
Lambda: https://aws.amazon.com/lambda/
Lambda Security White Paper: https://aws.amazon.com/lambda/security-overview-of-aws-lambda/

Automated Response Actions (19)

Azure Functions is Azure's serverless compute service to run code.

- Native support for C#, F#, Java, Node.js, PowerShell, Python, and Typescript.
- No concurrency limit advertised.
- Invoked by a timer, HTTP request, Blob upload/update, Queue, Cosmos DB, and Event Grid.

Functions can support service input and output bindings, making it more obvious what services the function will interact with.

Azure Functions is the Microsoft answer to AWS Lambda. They operate in much the same way with the same purpose. There are differences in billing, concurrency, and startup times. But for our purposes, they do the same job of executing customer written code in many of the same supported native languages.

AWS Lambda and Azure Functions are serverless, which means that the customer does not have to worry about how the function is executed. This allows for an architecture based on concurrency. Instead of a single application that performs twenty different jobs one after the other, you would create twenty different Azure Functions. Each function runs when an event happens. Those functions can be run concurrently rather than one at a time. This can drastically increase the throughput of the workflow, allowing your organization to do more work in the same amount of time. It will also change the way you architect your applications, providing a whole new way of operating.

Azure Functions supports resource binding for input and output. In the configuration of the Azure function, you specify the Azure Blob that data will be read from or written to. One great feature is that the internal function code does not need to specify **which** resource is the input/output.

References:
Azure Functions: https://docs.microsoft.com/en-us/azure/azure-functions/functions-overview
Azure Function security: https://docs.microsoft.com/en-us/azure/azure-functions/security-concepts
Azure Triggers and Bindings: https://docs.microsoft.com/en-us/azure/azure-functions/functions-triggers-bindings?tabs=csharp

Automated Response Actions (20)

AWS Lambda and Azure Functions are stateless. They execute and then forget.

- Functions generates logs in a silo, not connected to each other.
- Functions may detect a failure but cannot rewind previous actions.
- Single functions should be reusable in multiple workflows.

For an automated response workflow, we need a state managed service. AWS and Azure have services to support this.

AWS Lambda and Azure Functions are stateless operations. This means that they only know about their inputs, outputs, and execution. They have no idea what might have come before. This design really helps with creating concurrent, on-demand applications. Dealing with lots of small functions that are not directly connected to each other is a core part of serverless, event-driven microservice architectures.

In monolithic applications, the execution through the 100 functions likely carried state. Data storage is easily passed from function to function. Or the main program held state and then called each individual function to perform the 100 tasks. We need a way to replicate that orchestration of the functions.

AWS and Azure both have services that helps manage state across multiple Lambda and Azure Functions, sometimes orchestrating interactions with other cloud services. Let's look at a couple of them.

Automated Response Actions (21)

Azure Durable Functions is an extension of standard Functions.

A special Azure Function that can be written in C#, F#, Python, JavaScript, or PowerShell to perform the orchestration.

```
1   import azure.functions as func
2   import azure.durable_functions as df
3
4
5   def orchestrator_function(context: df.DurableOrchestrationContext):
6       data = yield context.call_activity("GetData", None)
7       result = yield context.call_activity("TransformData", data)
8       storage = yield context.call_activity("StoreData", data)
9       return result
10
11
12  main = df.Orchestrator.create(orchestrator_function)
```


Azure has Durable Functions, which is an Azure Function that manages the state and execution of the other Azure Functions in its workflow.

In this example, the “Durable function” will call three Azure Functions called GetData, TransformData,and StoreData.

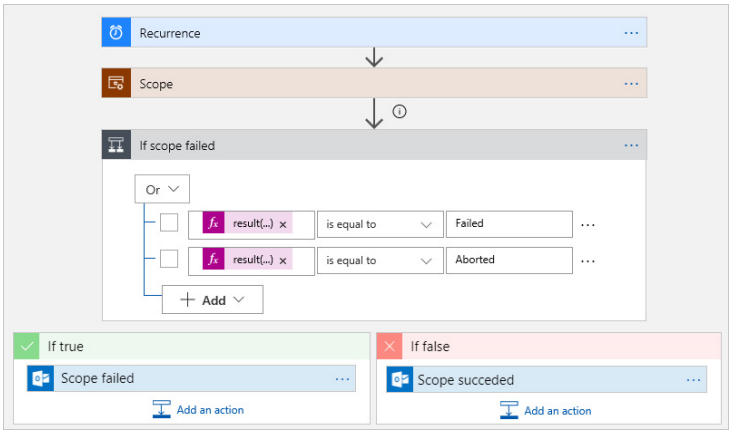
Building orchestrator in the programming language of choice feels very natural for the developer.

Azure provides a number of design patterns, including waiting for human interactions, aggregators, polling for an activity to finish, or function chaining as shown on this slide.

Reference:
Durable Functions: https://docs.microsoft.com/en-us/azure/azure-functions/durable/durable-functions-overview?tabs=python

Automated Response Actions (22)

- Azure Logic Apps is an orchestrator that provides an intuitive GUI
- Can also be configured with a config file
- Supports interactions with third party services



While Azure Durable Functions is designed for developers to orchestrate Azure Functions, Azure Logic Apps hits a different market. Logic Apps is a "designer first" model, with a declarative language and a visual workflow with an integrated designer tool. The drag and drop interface allows non-programmers to build complex workflows.

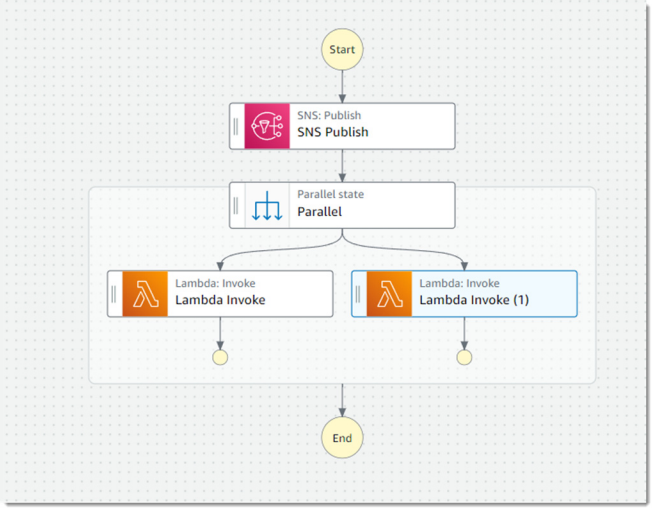
Logic Apps use "Azure Connectors" to interact with other Azure services, as well as third party services like tasking tools, SIEMs, and even AWS services. Those Logic Apps connectors enable interaction with more services out of the box than an Azure Function. That connection also supports authentication with the third-party services.

The Logic Apps services also supports a definition with configuration files, managing versions, and integration with the Visual Studio and IntelliJ suite of IDEs (among others).

Reference:
https://docs.microsoft.com/en-us/azure/logic-apps/logic-apps-overview

Automated Response Actions (23)

- Step Functions is AWS's Orchestrator
- Uses JSON to build the flow and connect services
- Step Functions now sports a visual editor like Logic Apps



The AWS Step Functions is AWS's main orchestrator. Similar to Logic Apps, it allows individual AWS Functions to be called while managing state, interacts with other AWS services, and provides callback functions for third party services.

Step Functions has been improving its tool building with a visual editor that allows drag and drop of individual "Tasks". Called "Workflow Studio", it allows you to create and edit these workflows.

Step Functions also integrates with the AWS SDK, allowing a step function to make specific SDK calls to over 200 endpoints. For instance, copying data from one S3 to another would normally be done from a Lambda function. Now, that command can be called directly from the Step Function task itself. This limits the number of Lambda functions that have to be deployed for a workflow.

These orchestrators can take some time to get used to the syntax, but they are a significant improvement over trying to deploy individual functions that are calling one another. Microservices can be difficult to debug and troubleshoot without some way of watching the flow of execution from service to service. These orchestration services in AWS and Azure not only tie together difficult tasks or functions, but they also help visualize and track the full state of the workflow.

Our lab centers around Step Functions to manage the execution of multiple lambda functions.

References:
https://aws.amazon.com/step-functions/
https://aws.amazon.com/about-aws/whats-new/2021/09/aws-step-functions-200-aws-sdk-integration/

Automated Response Actions (24)

After the workflow executes and the response action has fired, **validation** and logging should happen.

- The workflow should know if there were failures in the response action.
- The workflow may have been a false positive or destructive.
- The security team likely will need to follow up.

The entire flow of the workflow, with all resources touched, changed, or artifacts created must be easily available.

After execution, the success should be validated, logs created to track the entire execution, and any destructive activity raised to a higher alert. Collecting the logs will help when teams must reconstruct what happened, and how to improve defenses or security capabilities.

Collecting the reports and execution logs and providing them to the team should also be automated. Especially since, as we just saw, there could be different computational services brought to bear.

Course Roadmap

- Section 1: Management Plane and Network Logging
- Section 2: Compute and Cloud Services Logging
- Section 3: Cloud Service and Data Discovery
- Section 4: Microsoft Ecosystem
- **Section 5: Automated Response Actions and CloudWars**

Microsoft Ecosystem

1. Automated Response Actions
2. IT Ops Workflows
3. Security Workflows
4. EXERCISE: Set Up AutoForensic
5. Constructing Response Actions
6. **EXERCISE: Run AutoForensic**
7. CloudWars

This page intentionally left blank.

Lab 5.2 | Run AutoForensic

Exercise Duration: 45 Minutes

Objectives

The AutoForensic workflow should now be set up, and hopefully GuardDuty has kicked it off. Now we will look deeper into the workflow and how it executed.

- Look at how the workflow was executed
- Review the results of the workflow
- Investigate the artifacts it created

This page intentionally left blank.

Automated Response Actions (25)

When building automations, keep the following in mind:

- You can only automate a repeatable and describable process.
- Workflows must be treated as production software.
- The cloud, your environment, or external services can change. The workflow must be managed and updated regularly.
- IT engineers, operators, and security teams need to augment their skills with programmers and data analysts.

It takes time and energy to build an automation 1 time but will take little effort to run it 1,000 more times.

Automating security workflows can be difficult. Their success and complexity are directly related to how normalized the environment is. If development teams can choose which security policies to implement, then an alert or data collection, rather than destructive workflow, may work best.

Workflows are just like any other production software application you are building. The code must be managed properly, tested thoroughly, maintained, documented, and revised. Outside changes, such as new security policies, deployment scenarios, or changes in related services may require revision.

Breaking a workflow up into triggers, automated actions, and notifications may help. Triggers could be added and updated, but the automated action stays the same.

Less complex workflows may be built using cloud service provided tools and building blocks. A more complex service may require development experience, which is not typically a part of the IT operations team. In the world of cloud, software development becomes a necessary skill in all aspect of the operation and maintenance.

Building, managing, and operating workflows takes time. More time than clicking buttons yourself. Once its working as expected, that automation can run 1,000 times. The organization must be willing to invest the time to learn and build. It must also consider how an automated workflow may require changes in how the services are deployed or the applications are built. That is why tying the security workflow to the automated operations may help sell it to stakeholders. Sure, you are more secure, but we can run 1,000 widgets at a time. It broadens the appeal.

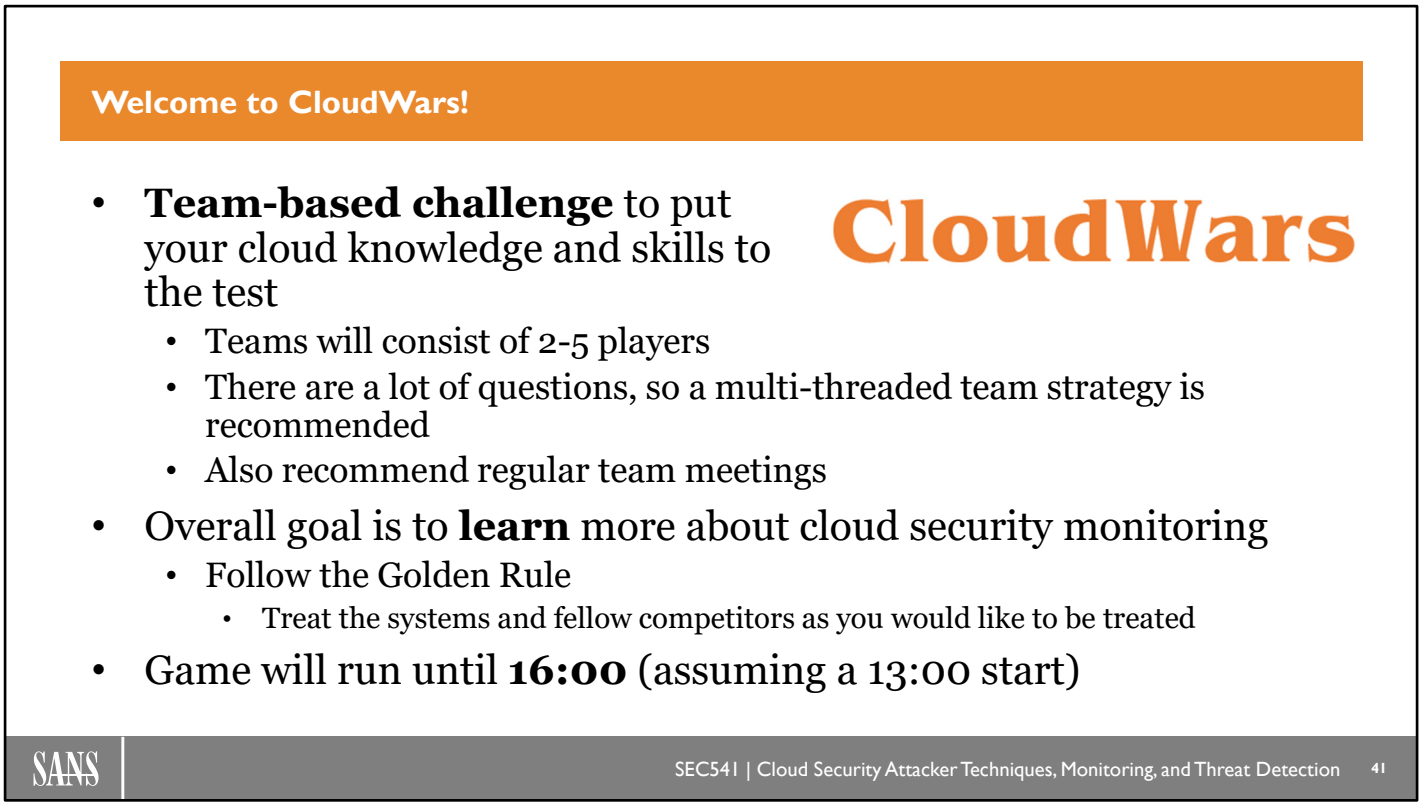
Course Roadmap

- Section 1: Management Plane and Network Logging
- Section 2: Compute and Cloud Services Logging
- Section 3: Cloud Service and Data Discovery
- Section 4: Microsoft Ecosystem
- **Section 5: Automated Response Actions and CloudWars**

Microsoft Ecosystem

1. Automated Response Actions
2. IT Ops Workflows
3. Security Workflows
4. EXERCISE: Set Up AutoForensic
5. Constructing Response Actions
6. EXERCISE: Run AutoForensic
7. **CloudWars**

This page intentionally left blank.



And now for a fantastic, team-based challenge to put your newly-discovered skills to the test! Welcome to CloudWars! If you are in a live class, you will hopefully have started pondering who to team up with in an attempt to correctly answer the forthcoming questions. If not, the time will be ticking very shortly! For all other variations of the course, you can participate individually as you will not have the same time crunch the in-person or live online classes will have—so take your time and enjoy!

If in a live class, you are going to have lots of fun competing with other teams to acquire the coveted SEC541 challenge coin! However, it is not all about the coin. You are here to have fun and learn more about cloud security monitoring and threat detection, through a series of hands-on challenges.

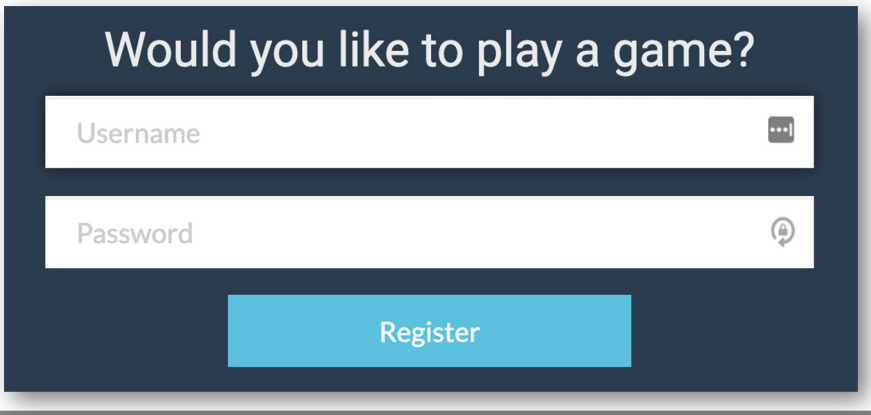
It should go without saying, but please be respectful of your fellow competitors, the course authors, instructor, and the supporting platform by playing the game as instructed. This means, first and foremost, no cheating, hacking the platform, or social engineering the instructor or fellow competitors. We are here to defend, not cause havoc.

The game will begin as soon as this short presentation is finished and will run, in a live setting, until 16:00 local class time unless the instructor says otherwise. This gives you almost three hours to complete as many of the challenges as possible. If you come up short, no worries. The course authors do not expect many teams to be able to complete all questions in the allotted time as the questions increase in complexity the further you move on—requiring more time to complete each successive question.

Signing Up

SANS will provide a URL for the CloudWars scoring server.

- Click on **Register** in the **top right corner** to begin the sign-up process.



To start competing, you will need to establish a username on the CloudWars server. The URL for this server was likely provided by the instructor just before this presentation began. Once you arrive, you will need to click on the Register link and populate the username and password field with values of your choosing. Protect these credentials, as anyone that can log into your account can see your answers and even cause you to lose points by incorrectly answering questions (more on this in a bit).

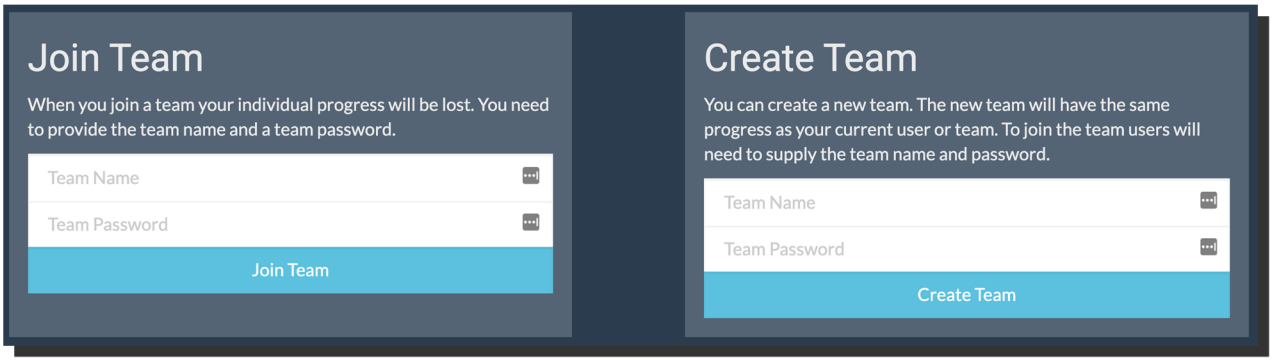
Once you created the username and password, it may appear that nothing happened as you are, again, presented with the username and password fields. You will notice, however, that this is no longer a registration page, but a login page. Make sure you credentials work by re-entering them in the username and password fields and clicking the Login button.

You should now arrive at the main questions page, but there are no questions. This is expected as the instructor did not yet start the game. As soon as the game is started, some questions will appear.

OnDemand students will find the URL in their My Labs section. Other students will be given the URL by the instructor.

Teams

- The first member of the team will click **Create Team**
  - Create a **team name** and **password** to share with team members
- The other members will click **Join Team**
  - Input the **team name** and **password** to join the team



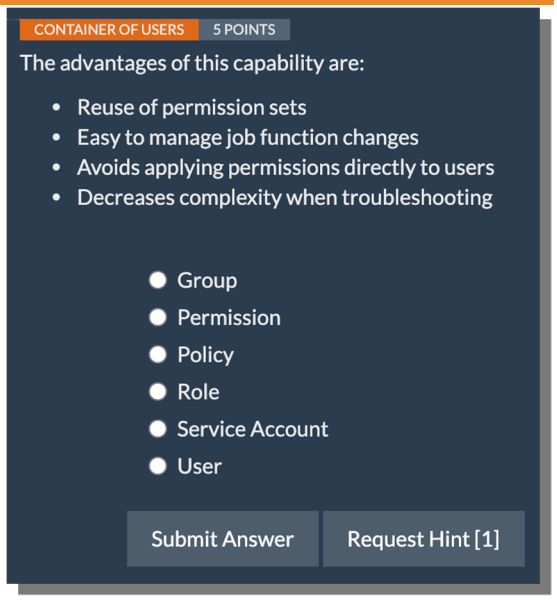
Once you have successfully created a login and you are participating in a live class, you must join a team. This can be done by clicking on the "Teams" button at the top of the page. If you are the first member of your team to get to this point, create a team name and password to share with your fellow team members. This can be done on the right side of this Teams page. Simply add a team name of your choosing in the top field and a team password in the second field. Don't forget this password as the other team members will need both the team name and password to join successfully—otherwise, the points they score will not be attributed toward the team. Click the Create Team button once those fields are populated.

If you are joining a team and have the team name and password sent to you by a fellow teammate, you will focus on the left part of the Teams page. Under Join Team, enter the team name and password and click Join Team.

If the team password is forgotten, please inform the instructor as they can reset this password for you.

Challenges

- Instructor will open the game soon
  - When opened, you will see challenges on the **Questions** page
- Four levels, which unlock as points are acquired
- You can get each answer wrong exactly once without a penalty
  - Incorrect answers 2-4 lose one point
  - Answers 5+ do not have a penalty



Once the game begins, some challenges will be available for your team to being answering. You will be presented with the challenge and the opportunity to unlock hints or answer the question. Pay very close attention to the question, and the format that the answer is expected to be in, as the result of several NetWars and CloudWars events came down to a misunderstanding in what the answer format is supposed to be. Had the contestant just read a little more carefully, they would not have lost the point that cost them their precious challenge coin.

When you see the questions, you may think, "Certainly, this cannot be all of the questions. I can get through these in 20 minutes, let alone 3 hours!". That is because you are only seeing the Level 1 questions. There are four total levels in this game and, as you progress, you will unlock the three other levels. There are also "gateway questions" which are questions that, when answered correctly, unlock follow-up questions that are related.

As you answer questions correctly, you will earn the points noted just above the question. Another very important item to note is that you can lose points as well. If you get any one question wrong, no points are lost, but every successive wrong answer, you will lose one point up to a maximum of three points lost per question.

Imagine you came into the troublesome question with 100 points and get the answer wrong the first time. You will still have 100 points. If you get it wrong again, you are down to 99 points. Wrong again—98 points. Wrong again—97 points. Wrong a fifth time—still 97 points. This is because you lost all the points and are free to guess at this point for this question. Please note that this is on a question-by-question basis.

Challenge Types and Hints

- **Challenge Types**
  - **Provide the answer as text**
    - Some of these are hands-on and you will interact with cloud resources
    - Pay close attention to the expected answer format
  - **Fill-in-the-blank**
  - **Multiple choice**
  - Some challenges **unlock** other challenges
- **Hints**
  - Cost points, so use them very carefully
  - Provide a nudge to help you retrieve the correct answer
  - Level 3 questions do not have any hints!

There are three different types of challenges to answer in this game. If you have played capture the flag (CTF) challenges before, the first type of question, "provide the answer as text", should look familiar. This question will ask you to retrieve a flag from your cloud environment, log entry, or some other location and provide this flag as you answer.

The second question type is fill-in-the-blank. Most of us should be familiar with these types of questions from grade school, but for those of us that have not been in grade school or taken a test where fill-in-the-blank was required, an example would be something like this:

_____ is the best class ever!

Obviously, the answer here would be SEC541.

All joking aside, let's move onto the third question type: multiple choice. These questions will ask you to select only one correct answer from a list of six options.

If you or your team are/is totally unsure of how to proceed with the challenge, you may elect to use hints. Hints do cost points to unlock (meaning you can no longer get a perfect score), so use them sparingly and ensure that the team is in agreement. Hints will begin to vanish as you progress, making the higher-level questions even more challenging.

And the Winner is...

- The game can end a few different ways:
  - It is 16:00
  - The leading team cannot be caught by any other team
  - A team acquires a perfect score of 541 points
- Remember, this game is designed to enhance your learning, so please be respectful of the environment and fellow players!
- **Any last-minute questions before we begin?**

The game will end when one of these conditions are met:

- It is 16:00, or the instructor-provided end time
- The leading team cannot be caught by any of the other teams (e.g., all other teams lost too many points to overtake the first-place team)
- A team achieves a perfect score of 541 points

One caveat is, in the event of a tie, there are tie-breakers to consider. The first tie-breaker is that the team that used the least number of hints will be in the higher position. The second tie-breaker (same score and same number of hints), the winner will be the team that reaches the score first. The platform takes care of this for us.

If you have any other questions, now would be the time to ask since the instructor will be opening the game very shortly.

Go!

Instructor will now open the game!

Let the games begin! Good luck and please reach out if you have any questions related to the game!

Section 5 Wrap Up

Automated Response Actions and CloudWars

- Discussed the purpose of automating response actions
- Looked at IT Ops Workflows
- Looked at Security Workflows
- Learned about the Cloud Services Used
- Built and executed an automated forensic workflow
- Played some CloudWars!

This page intentionally left blank.

Thanks, and Happy Hunting

"Good luck is when opportunity meets preparation, while bad luck is when lack of preparation meets reality."

- Eliyahu Goldratt

This page intentionally left blank.

COURSE RESOURCES AND CONTACT INFORMATION

AUTHOR CONTACTS

Shaun McCullough
smccullough@sans.org

Ryan Nicholson
ryananicholson@gmail.com

SANS INSTITUTE
11200 Rockville Pike, Suite 200
N. Bethesda, MD 20852
301.654.SANS(7267)

CLOUD RESOURCES
sans.org/cloud-security
Twitter: @SANSCloudSec

SANS EMAIL
GENERAL INQUIRIES: info@sans.org
REGISTRATION: registration@sans.org
TUITION: tuition@sans.org
PRESS/PR: press@sans.org

This page intentionally left blank.