



Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permitted without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Network Monitoring and Threat Detection In-Depth (Security 503)"
at <http://www.giac.org/registration/gcia>

Balancing Security and Innovation With Event Driven Automation

GIAC (GCIA) Gold Certification

Author: Teri Radichel, teri@radicalsoftware.com

Advisor: Richard Carbone

Accepted: March 12th 2016

Abstract

Many organizations struggle with the conflict between software developers who want to use new technology and security teams who want to prevent deployments that contain security vulnerabilities. Security teams blocking new technology risk hurting the company financially or being bypassed altogether. Based on the number and magnitude of recent security breaches, organizations that choose to bypass security recommendations face substantial risk. This paper presents an alternative approach to manual security review and overcomes bypassed security review, using security automation to respond to events in the environment. Amazon Web Services (AWS) cloud infrastructure and security tools are particularly well suited for event driven security automation and will be used to provide examples, but the concepts apply to any environment. A working framework demonstrates automated intrusion detection and response on AWS.

1. Introduction

Organizations seek innovation via use of new technology. In order to save money and deliver new products and features quickly, software development teams want to use open source software (Black Duck Software, 2015), public cloud platforms and continuous deployment strategies (Right Scale, 2016). These approaches introduce changes to networks and systems at a rapid pace. Security teams would generally prefer to vet changes introduced into an environment before deployment. If security teams continue to use manual processes, it is likely they will become a bottleneck in the innovation pipeline.

Event driven security automation can replace manual review in some cases, both removing the bottleneck and improving security. Based on the number of security breaches reported in the Verizon *Data Breach Investigations Report*, manual attempts to block what is bad from entering the environment alone is not eliminating attacks (2015). According to a presentation at the Seattle Cloud Security Alliance by Greg Reith and Brett Peppe on threat intelligence, adversaries are carrying out automated analytic attacks and security teams need to automate their response in order to be effective (2016). By responding to events that impact security in an automated fashion, problems can automatically be prevented or remediated.

Automated event response shifts blocking the “unknown” to blocking what is “known.” There are key points where vulnerabilities are introduced into an environment. Blocking a “known bad” at the entry point is an obvious first step to keeping an organization secure. Auditing for a “known bad” after it has been introduced can find what the blocking process missed. Rather than block “unknowns,” security teams can implement an automated event response system that analyzes the behavior of hosts using network traffic. Hosts that have been compromised will exhibit traffic patterns that can be used to detect a breach where automation can stop it.

Amazon Web Services (AWS) is a cloud platform which allows customers to deploy application resources including virtual machines, databases, file storage and other infrastructure that can scale up or down as needed. In the white paper *Overview of*

Teri Radichel, teri@radicalsoftware.com

Amazon Web Services (Amazon, 2015b), Amazon provides a description of cloud computing, its benefits and a brief description of some of the resources that can be deployed on the AWS cloud platform.

In order to take advantage of the platform's scalability, infrastructure needs to be deployed with cloud architecture in mind. The Amazon white paper, *Architecting for the Cloud: Best Practices* (Varia, 2011), provides an overview for properly architecting cloud applications. Building applications that can self-heal allows for new types of security responses to tackle problems found in the environment. Non-compliant hosts can automatically be terminated and replacement hosts can be re-deployed without human intervention. Netflix has developed Chaos Monkey, a tool that takes this concept to the point of randomly and intentionally terminating AWS resources to ensure self-healing is properly functioning (Bennett & Tseitin, 2012).

2. Event Driven Security Automation

2.1. Requirements for Event Driven Security Automation

In order for a security team to employ automation of event driven security, the security team must have access to data concerning events occurring in the environment that warrant a response. If the security team cannot track or does not have access to data indicating that events are occurring, response to those events cannot be automated. If someone can manipulate the event data, the data cannot be trusted. Missing data will lead to a lack of response. Inaccurate data may lead to a faulty response (OWASP, 2016). In order to mitigate risk of alteration, logs must be stored securely. NIST provides guidelines for log protection in *Guide to Security Log Management* (Kent & Souppaya, 2006).

For example, if hosts are missing from the host inventory, events generated by missing hosts will not be tracked and therefore no response can be automated. If the security team cannot monitor software deployments, automation cannot effectively block installation of software with known CVEs, a leading cause of security breaches. Verizon stated in the 2015 *Data Breach Investigations Report*: "We found that 99.9% of the

Teri Radichel, teri@radicalsoftware.com

exploited vulnerabilities had been compromised more than a year after the associated CVE was published” (p. 15). If the data that drives automated responses can be tampered with, event responses may in turn be inappropriate or bypassed. In this case the data cannot be trusted to secure the environment.

To employ event driven security, automation must be embedded into processes that introduce new threats into the environment. Highly innovative companies such as Netflix embed automated security into their build and deployment systems on AWS (King, 2015). Security teams must have a way to audit the environment in a secure manner and trigger events based on data patterns that indicate security risk or intrusion. Network data and traffic patterns can be used to trigger automated event responses, but only if the data can be trusted and the automated responses are built and secured to be tamper-resistant. Consider the results of the equation “ $1 + 2 = 3$.” If 2 changes to 5, then result 3 becomes 6. The results are different if the inputs change. In order to implement security automation with results that can be trusted, the inputs to this process, system logs, must be trusted and immutable.

Adequate training is required to ensure that the automation being deployed is not just automating processes, but that infrastructure is deployed in a manner that maintains a secure environment. Security automation teams must have developers who understand how to write software that is both secure and performs well in the environment. Secure software will follow security coding best practices (OWASP, 2010). Security tools that are not implemented with performance in mind can degrade performance of applications across the enterprise. Security professionals well versed in the security risk and threats an organization faces should help define requirements to ensure events and responses are correctly implemented. Quality assurance professionals who understand security should be involved in validating the automation.

Build systems that deploy automated systems must be secure. Automated systems that are not designed with appropriate build and deployment controls may actually increase security problems. For example, there is some evidence that in the Target Breach the build system may have been used to deploy malware to all the POS systems very quickly (Schwartz, 2014), which led to a massive data breach with far reaching effects.

Teri Radichel, teri@radicalsoftware.com

2.2. AWS Support for Security Automation

Amazon's AWS cloud platform offers many of the requirements for event driven security. Although a security team can automate compliance, intrusion detection, and response in any environment, AWS provides a platform where much of the heavy lifting is already done freeing security teams to focus on implementing security rules instead of generic system and platform components. Almost every type of resource that can be created on AWS can be instantiated using web service calls via code, instead of a human manually taking some action.

Some security teams have concerns about hosting systems or data on a third-party cloud due to a lack of visibility into all aspects of the cloud environment. The AWS security white paper describes a “Shared Responsibility Model” that defines which aspects of security Amazon provides and defines the customer's responsibility (Amazon, 2015a). Companies can review Amazon's security practices to see how they align with their own security practices and where gaps might exist between the two environments. By allowing Amazon to handle certain aspects of security, there is some loss of visibility into certain aspects of security. Amazon has been audited and certified by a number of third parties (Amazon, 2016b). Trust depends on an agreement and belief that Amazon will do what is stated in their contract, if those security policies and processes used by the cloud provider are acceptable (Cole, 2014).

While there is some loss of control due to not being able to go inside an Amazon data center or deploy certain security devices as has been done in the past, what is gained are new ways of implementing security that enable security automation teams become more effective. AWS is designed with security and automation in mind and Amazon continues to innovate in this area. At last year's annual AWS, conference Amazon introduced a number of new security tools (Stamp, 2015). The tools AWS provides can help security teams devise automated responses to security.

AWS offers inventorying, logging, separation of duties and programmatic controlling of all cloud resources. On AWS, a complete inventory should always exist by virtue of how the platform works. Every host, every network rule, every DNS entry and

Teri Radichel, teri@radicalsoftware.com

most other aspects of infrastructure can be queried in a matter of minutes. Every action taken by users on the platform is logged. Employees at Amazon cannot make changes to customer resources. Customers cannot make changes to something that has been logged in the AWS logging services (CloudWatch¹ and CloudTrail²), other than to completely delete the logs. Control to view and change or delete resources or logs can be controlled with the AWS IAM (Identity and Access Management)³ service (Prendergast, 2016).

All AWS services are accessible via an API (application programming interface)⁴ that allows programmatically creating, updating, and deleting infrastructure. Specific security and event trigger services such as Security Inspector⁵, Config Rules⁶ and CloudWatch Events⁷ can be used to automatically roll back or block non-compliant changes to the environment.

2.3. Caveat: AWS Security Best Practices

The ability to enhance security using the AWS cloud platform depends heavily on the proper use of its services in accordance with AWS best practices. A recent Gartner report predicts that most security problems on cloud provider platforms will be due to configuration problems and mismanagement of cloud resources (MacDonald & Young, 2015). A company named Code Spaces is completely out of business due to not following AWS best practices and failing to implement an appropriate backup and recovery process (Marks, 2014). Netflix, a company that has been very successful on AWS, explained in a presentation at AWS re:Invent 2015 that expecting software developers to be experts network engineering is likely not going to be successful in

¹ For more information, please consult <https://aws.amazon.com/cloudwatch/>.

² For more information, please consult <https://aws.amazon.com/cloudtrail/>.

³ For more information, please consult <https://aws.amazon.com/iam/>.

⁴ For more information, please consult <https://aws.amazon.com/tools/>.

⁵ For more information, please consult <https://aws.amazon.com/inspector/>.

⁶ For more information, please consult <https://aws.amazon.com/blogs/aws/aws-config-rules-dynamic-compliance-checking-for-cloud-resources/>.

⁷ For more information, please consult <https://aws.amazon.com/blogs/aws/new-cloudwatch-events-track-and-respond-to-changes-to-your-aws-resources/>.

Teri Radichel, teri@radicalsoftware.com

protecting their customer experience (Hahn, 2015). Deployment, networks, logging, access, and auditing must be set up in a secure manner just as in the traditional data center.

2.4. AWS Automation

To programmatically interact with AWS cloud resources, many tools exist that leverage commonly used programming languages⁸. The most basic tool that supports most AWS API calls is the AWS CLI (command line interface)⁹. This tool is used to run commands that create, describe, update, and delete resources. It will be used in subsequent examples as it is the easiest to install and use, and every new service is typically associated with a CLI command that enables automation of that service.

As an example, the most obvious use of a cloud service would be to create virtual machines on the cloud platform. A security team might then want to inspect details about running hosts and automatically take action if a host is non-compliant. On AWS, these virtual hosts are called EC2 instances. With the AWS CLI, the most basic syntax for these commands would be as follows:

To start an instance:

```
$ aws ec2 run-instances --image-id ami-xxxxxxx
```

To view details about an instance:

```
$ aws ec2 describe-instances --instance-id i-  
xxxxxxxxx
```

To terminate an instance:

```
$ aws ec2 terminate-instances --instance-id i-  
xxxxxxxxx
```

The same approach to programmatically managing an EC2 instance can be used to create, query, and delete other AWS resources in a similar fashion.

⁸ For more information, please consult <https://aws.amazon.com/tools/>.

⁹ For more information, please consult <https://aws.amazon.com/cli/>.

In addition to command line calls that specify all the properties for a resource on the command line, Amazon offers an AWS deployment tool called CloudFormation¹⁰. It is not available for every type of resource but when available, it is beneficial because it can help with dependency management and, where possible, the parallel deployment of resources. It is self-describing and decouples the definition of a resource from its deployment. CloudFormation files are written in JSON and can be deployed by the CLI. The call to deploy a CloudFormation template would look like this:

```
$ aws cloudformation create-stack --template-url [path]
```

AWS offers many other ways to automate deployments, security, and event response. These simple examples are provided as they are used in the subsequent sample security automation code.

2.5. An Overview of AWS Networking

AWS Networking can be deployed in a completely automated fashion as source code just as with almost any AWS resource. Amazon allows customers to create VPCs (Virtual Private Clouds)¹¹ within an account, which are a customer's own networks on the Amazon cloud platform. A VPC may have Internet access, depending on how it is configured. Within each VPC subnet¹², NACLs (Network Access Control Lists)¹³ can be defined. Subnets break up the network into smaller segments with specific access rules (NACLs) that define traffic allowed in and out of the subnet.

In addition to subnets, AWS offers Security Groups¹⁴. A Security Group is a set of rules, similar to a host based firewall. However, a security group does not reside on the

¹⁰ For more information, please consult <https://aws.amazon.com/cloudformation/>.

¹¹ For more information, please consult <https://aws.amazon.com/vpc/>.

¹² For more information, please consult http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Subnets.html.

¹³ For more information, please consult http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_ACLs.html.

¹⁴ For more information, please consult http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_SecurityGroups.html.

Teri Radichel, teri@radicalsoftware.com

host. The rules are applied outside the host, unlike a firewall deployed on the host itself, that could be altered by malware (Amazon, 2015a).

To allow traffic into an AWS VPC from the Internet an Internet Gateway is attached to the VPC. Other options for Internet connectivity are available but that topic is beyond the scope of this paper. This overview covers the networking components used in the sample security automation code provided with this paper.

Amazon provides a network traffic logging service called Flow Logs¹⁵. Logs can be turned on for an entire VPC, for a subnet or for a particular network interface. Flow logs capture information about IP traffic going to or from network interfaces within a VPC. Flow Log records will track the information, as shown in Table 1.

Table 1: Flow Log Records (Amazon, 2016a)

Field	Description
version	The VPC flow logs version.
account-id	The AWS account ID for the flow log.
interface-id	The ID of the network interface for which the log stream applies.
srcaddr	The source IP address. The IP address of the network interface is always its private IP address.
dstaddr	The destination IP address. The IP address of the network interface is always its private IP address.
srcport	The source port of the traffic.
dstport	The destination port of the traffic.
protocol	The IANA protocol number of the traffic. For more information, go to Assigned Internet Protocol Numbers .
packets	The number of packets transferred during the capture window.
bytes	The number of bytes transferred during the capture window.
start	The time, in Unix seconds, of the start of the capture window.
end	The time, in Unix seconds, of the end of the capture window.

¹⁵ For more information, please consult <http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/flow-logs.html>.

Field	Description
action	<p>The action associated with the traffic:</p> <ul style="list-style-type: none"> ● ACCEPT: The recorded traffic was permitted by the security groups or network ACLs. ● REJECT: The recorded traffic was not permitted by the security groups or network ACLs.
log-status	<p>The logging status of the flow log:</p> <ul style="list-style-type: none"> ● OK: Data is logging normally to CloudWatch Logs. ● NODATA: There was no network traffic to or from the network interface during the capture window. ● SKIPDATA: Some flow log records were skipped during the capture window. This may be because of an internal capacity constraint, or an internal error.

This data is sent to CloudWatch¹⁶, Amazon's monitoring service for cloud resources. Flow Logs data can be pulled from CloudWatch into other AWS resources for further processing, visualization and analysis. It can also be used to drive automated responses using AWS APIs as will be demonstrated further on in this paper.

Amazon does not offer packet level detail at this time. Packet level logging is possible by overlaying infrastructure on AWS (Wild Packets, 2013). Implementation of packet level logging on AWS is outside the scope of this paper, but the option was considered and discussed with AWS security and network architects. Secure packet logging requires network infrastructure on top of the AWS network. Implementation and maintenance requires a fair amount of time and expense, detracting from the benefits of AWS networking, which include a free programmable network that interacts with other AWS resources by design (Brandwine, 2015). Amazon uses customized packet headers and has built-in network security to help prevent certain types of attacks.

Many traffic patterns that demonstrate adversarial activities can be gleaned from traffic flows and endpoint modeling, rather than packet level inspection (Observable Networks, 2015). A mechanism could be deployed to add packet level logging when required for a particular instance or set of instances rather than an entire account.

¹⁶ For more information, please consult <https://aws.amazon.com/cloudwatch/>.

However, should an enterprise choose to implement packet level logging, the same concepts demonstrated for Flow Logs security response automaton, discussed in the remainder of this paper, could be applied to packet level inspection.

2.6. Integration of Security Automation

Event response can be integrated at different points in the software development lifecycle. The first approach would block anything that does not meet the current security standards. The second approach would be to reverse any changes that are made which are non-compliant, as quickly as possible. At the other end of the spectrum, a security team can simply respond to security incidents and events as they arise. Automation can be incorporated at one or all points of inspection.

2.6.1. Blocking Before Deployment

Blocking non-compliant changes before deployment will prevent a “known bad” from executing in the environment. To automate blocking non-compliant actions requires a mechanism for tracking deployment events and injection of security responses into the deployment process. If the security team does not have the ability to maintain and monitor the deployment pipeline this approach is not an option. AWS Principle Security Architect, Bill Shinn, recommended at AWS re:Invent 2015 that security teams should be using the same deployment pipeline as teams who are deploying applications (Rossman & Shinn, 2015). If the security team is not using the deployment pipeline, they may not be aware of the vulnerabilities that exist in it. By using the same pipeline, security teams will understand the challenges developers face when trying to use and deploy new technology.

If a decision is made to block certain events at the time of deployment, the consideration of perceived or potential versus real threats can be taken into account. Instead of blocking new software in a development environment, organizations might opt to allow new software to be deployed unless it is on a “known bad” list, or in other words a blacklist. At re:Invent 2015, AWS demonstrated the use of a new service called Security Inspector to scan and block a host from being deployed with a known CVE (Lucas, 2015).

Teri Radichel, teri@radicalsoftware.com

2.6.2. Audit and Respond to What Was Deployed

Events that introduce non-compliant changes into the environment can also be reversed after deployment. For example, the AWS Config Rules¹⁷ service provides a means to see if deployed AWS resources are compliant. Discovered non-compliant services can be terminated. Auditing tools such as Dome 9¹⁸ can roll back changes that were not approved or compliant. This approach will reduce the amount of time the problem exists in the environment.

The problem with this approach is the window of time in which the non-compliant changes exist in the environment as it can be used to exploit systems. For example, if a tool rolls back a wide-open network rule after three minutes, an attacker that knows this can simply make the change and exploit that window. If the change being audited introduces high risk into the environment, it is better to prevent the change in advance, if possible. Additionally if a tool rolls back a change incorrectly due to a flaw, bug or vulnerability the tool can potentially be exploited. It would be better to prohibit a known bad from entering the environment in the first place.

Instead of rolling back the change, auditing can also be used to alert and educate the person who caused the problem so they can fix it. This is the approach used by Intuit (Lietz & Bretan, 2015) to both find security problems and educate developers. For low risk changes, this allows developers to move forward with software development but be aware that they have to fix the problem before moving the system to production.

2.6.3. The Unknowns: Automated Intrusion Detection and Response

Known security problems can be detected through automation, as noted in the previous sections. The problem arises when developers want open access to a myriad of new open source tools, to make changes quickly using continuous integration and deployment (CICD). Continuous deployment strategies release new code to the environment multiple times per day. The risk posed by this new software may be

¹⁷ For more information please consult: http://docs.aws.amazon.com/config/latest/developerguide/evaluate-config_use-managed-rules.html.

¹⁸ For more information please consult: <https://dome9.com/product/>.

unknown. However, software developers, project managers, and product managers become impatient waiting for software to be reviewed by security before adding it to the deployment pipeline.

Unknown security problems can be addressed by a security review that may delay projects and put security teams at odds with innovators. Typically, the number of security team members in an organization is much smaller than the number of developers so security reviews will become a bottleneck in the software development lifecycle. Current data breach statistics indicate that even with the security reviews and processes in place today, many flaws that lead to security breaches are still entering the environment (Verizon, 2015). This could be because the security controls designed to prevent or rollback a breach, missed the problem, or that the controls were bypassed. Whatever the cause of the problem, the current tactics used to prevent security problems from entering the environment still have gaps.

Rather than attempting to block unknowns from entering the environment, security teams can consider focusing on automated intrusion detection and response. By considering commonalities shared by malware when it is running and focusing on detection and mitigation of threats as they arise, security teams can spot existing problems in the environment as well as problems introduced by new software. As problems are discovered via automated intrusion detection processes, they can be integrated into automated blocking of known problems before they enter the environment. This iterative approach may help security teams become more effective and efficient as more problems are fixed automatically without human intervention.

Malware has common network traits. In order for malware to operate, it must traverse the network to receive commands, scan the network and exfiltrate data. If network logs capture all traffic, actions taken by malware that traverses the network will be in those logs. If known bad behavior is exhibited, automation can take immediate action to prevent a breach.

A paper by Lockheed Martin outlines the steps of an attack and shows how the network must be involved (Hutchins, Cloppert, & Amin, Ph.D., 2011). The initial point

Teri Radichel, teri@radicalsoftware.com

of entry for any exploit will be to send traffic over the network to deliver a payload. Software will be delivered and installed onto a host. Once the software is installed, it will “phone home” to a command and control server. After doing so it will receive commands and will likely start scanning the network to find other hosts that have vulnerabilities it can exploit. By repeating this process the malware may eventually obtain access to critical systems and attempt to exfiltrate sensitive data over the network to hosts outside the organization.

Some known bad traffic patterns require packet level logging and inspection; however, many types of bad traffic can be gleaned from traffic flows alone. In *Network Security Through Data Analysis: Building Situational Awareness*, Collins (2014) finds flows to be more effective than individual packets to find fumbling, which he defines as a failed attempt by a host to access a resource (Kindle, Loc 4839). A failed attempt can be identified by a rejected connection, if networks are properly configured to only allow valid traffic and rejections are logged in a secure manner. A few other obvious patterns found in an AWS Flow Log can be used to determine if a host has been compromised:

- A host that generates a REJECT in Flow Logs is performing an unwanted action;
- A host that should not send outbound traffic initiates outbound traffic;
- A host communicating with a known bad address may be compromised. For example, Alert Logic publishes a list of the top 10 bad IP addresses. Other lists of known bad IP addresses can be found at Lenny Zeltzer’s web site:

<https://zeltser.com/malicious-ip-blocklists>.

These examples are used only to demonstrate patterns in general, rather than provide a comprehensive list. They will be used to demonstrate automated intrusion detection and response in a honeypot deployed using AWS resources. The patterns used to automate detection and response may vary depending on an organization’s specific rules for network traffic flows.

Further analysis and automation could be performed if Flow Log data was pulled into an alternate data store to perform aggregated queries and cross-flow analysis. An organization may have a threshold for the number of sessions, length of sessions or

Teri Radichel, teri@radicalsoftware.com

number of packets that trigger an automated action against the offending host (Cole, 2014). In his book, Collins (2014) provides many good examples and Python scripts that can be used to set up automated intrusion detection and response based on network traffic patterns. Events can trigger full packet capture as well as further analyze traffic. The focus of the following example is to provide a completely automated intrusion detection and response framework. This framework can be modified and extended to support many other tools and patterns.

2.7. Example: Automated Detection and Response

2.7.1. Objective

The goal of this example is to demonstrate automated instantiation of resources, detection of traffic matching predefined rules, and automated response that stops the malicious activity. The code automatically builds network and hosts. Automated processes automatically detect unwanted traffic patterns and respond to the event. The response should both prevent further damage but also maintain evidence to determine the root cause of the problem.

Automation can prevent known bad changes and software from entering the environment. As noted in Section 2.6.3, an alternative to blocking unknowns entering the environment could be to automate the detection of and response to network patterns that indicate malicious activity. With a framework similar to this example in place, developers can download and use new open source code in a non-blocking manner. Misbehaving hosts that match defined known bad traffic patterns are automatically detected and terminated. On-going traffic analysis would allow security teams to refine rules and responses to automatically find and thwart malware before they can do serious damage.

Before introducing the code, an explanation of infrastructure as code is presented, as well as the AWS resources that will be used in this example. The remaining sections explain what is deployed, how to run the code, and what the code does when it is run. To view the code visit: <https://github.com/tradichel/AWSSecurityAutomationFramework>.

Teri Radichel, teri@radicalsoftware.com

2.7.2. Infrastructure as Code

AWS users commonly refer to the phrase “infrastructure as code” (Wittig, 2015). The goal is to completely automate deployments, with all the required code stored in a source code repository. This practice is well known to software developers, but is a new concept to some people in other fields such as security.

One obvious benefit of writing code for deployments is the prevention of human error that occurs in manual processes. Automated deployments are also faster. It will take some time to create the automation; however, the aggregate savings in time and reduction of errors after automation can be substantial. The code can be shared so others can leverage it. In addition to sharing code, a source code repository ensures that changes can be tracked and audited.

In the event of a non-compliant or inadvertent change, it can be rolled back. In the case of a disaster where the environment is destroyed or damaged for some reason, the source code can be run again to restore the environment. Additionally with an automated configuration, it is easy to build up and tear down environments when not in use, which saves money on a third party cloud provider.

2.7.3. AWS Resources Deployed In This Example

The following table is a brief description of resources created by this sample code and a link to the AWS documentation.

Table 2: AWS Resources Used In Sample Code

Resource	Description
Aws Account	Customers may have multiple AWS accounts. An AWS account is created when a customer registers at https://aws.amazon.com .
AWS CLI	The AWS CLI or Command Line Interface is a tool that can be used to manage AWS resources. The sample code includes scripts that execute commands using this tool. https://aws.amazon.com/cli/
CloudFormation	CloudFormation is a tool that deploys AWS resources based on a JSON template file that contains resource configuration. CloudFormation separates deployment of resources from definition and provides some management of resource dependencies. https://aws.amazon.com/cloudformation/
Regions and Availability Zones	An AWS region is a physical location in the world where a group of data centers exist. An AWS availability zone generally refers to one of the data centers in a particular region. https://aws.amazon.com/about-aws/global-infrastructure/

Teri Radichel, teri@radicalsoftware.com

VPC	A VPC or Virtual Private Network is a network created within a customer account. VPCs may be completely private or open to the Internet. A VPC exists only in one region. Multiple VPCs can be created within a customer account in the same or different regions. A VPC is similar to VRF (Virtual Routing and Forwarding) (Brandwine, 2015) https://aws.amazon.com/vpc/
Subnet	Subnets can be created in an AWS VPC to segregate the VPC network. A subnet exists in one availability zone. A subnet is basically a VLAN (Brandwine, 2015). http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Subnets.html
Route Table	A route table is created and attached to a subnet to specify allowed source and destination traffic routes. The same route table can be attached to multiple subnets. http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Route_Tables.html
Internet Gateway	An Internet Gateway is created to allow Internet traffic in or out of a VPC. The Internet Gateway is added to a route in the route table of a subnet to allow traffic to or from the Internet for that particular subnet. http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Internet_Gateway.html
NACL	A NACL or Network Access Control List on AWS specifies traffic that is allowed in or out of a subnet. The same NACL can be applied to multiple subnets. NACLs can have allow and deny rules. NACLs are not stateful so both inbound and outbound rules can be specified. Rules are processed in order. This allows for finer grained control when a subset of a larger network needs to be blocked or allowed. http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_ACLs.html
Security Group	A security group is created in a VPC and consists of a set of rules specifying what traffic is allowed in or out of an AWS resource in that VPC. The rules act as a firewall around resources. Multiple security groups can be applied to an AWS resource. The same security group can be applied to resources in different availability zones in a VPC. A security group consists only of allowed traffic. Rules to deny traffic cannot be created. Security groups are stateful. This means that response traffic for any request traffic will be allowed. http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_SecurityGroups.html
AMI	An AWS AMI is a machine image that specifies a particular operating system and configuration for a new virtual machine. Amazon provides some machine images or customers may create their own. http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMIs.html
ENI	An ENI (Elastic Network Interface) is a network interface that can be attached to a virtual machine or database on the AWS cloud. An ENI can be detached from one AWS resource and added to another. http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-eni.html
EC2 Instance	An EC2 Instance is the AWS name for a virtual machine on the AWS cloud platform. An EC2 instance is created based on a particular AMI. It

Teri Radichel, teri@radicalsoftware.com

	is assigned to a particular subnet and an ENI is created, or ENI(s) are assigned to it when it is created. Security groups are assigned to the instance. https://aws.amazon.com/ec2/
EBS Volume	An EBS volume is basically a virtual drive that can be attached to an EC2 instance. https://aws.amazon.com/ebs/
S3 Bucket	Scalable storage for files and objects in locations are called buckets. Other AWS resources can access objects stored in S3 buckets. Each bucket name must be unique across all existing AWS S3 buckets. https://aws.amazon.com/s3/
IAM Roles	<p>IAM is the AWS Identity and Access Management Service. Within that service the concept of roles exists to limit the action of a user or resource assigned to that role. http://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html</p> <p>Use of roles with an EC2 instance is an AWS IAM best security practice. http://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html</p>
CloudWatch	A repository for logs and metrics for AWS resources running on AWS. Alarms can be triggered from CloudWatch events. https://aws.amazon.com/cloudwatch/
Flow Logs	Flow Logs capture IP traffic to and from network interfaces in a VPC. Flow Logs can be configured to capture traffic for an entire VPC, subnet or a network interface. Flow Logs are sent to CloudWatch. http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/flow-logs.html
Lambda	AWS Lambda provides a mechanism for running code on the cloud without deploying servers or other infrastructure. The code will run on infrastructure that scales as required. https://aws.amazon.com/lambda/
CloudWatch Subscription	CloudWatch Subscriptions feed CloudWatch events to resources subscribed to receive them. http://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/Subscriptions.html
Snapshot	A snapshot is an exact replica of an EBS volume. When multiple snapshots are taken each new snapshot records the differences since the previous snapshot. http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSSnapshots.html

2.7.4. Overview

The entire stack creation is automated through command line scripts using the AWS CLI and CloudFormation. The code has three modes: CREATE, DELETE and PINGTEST. When run in CREATE or PINGTEST mode, resources are deployed. When run in DELETE mode, resources are deleted.

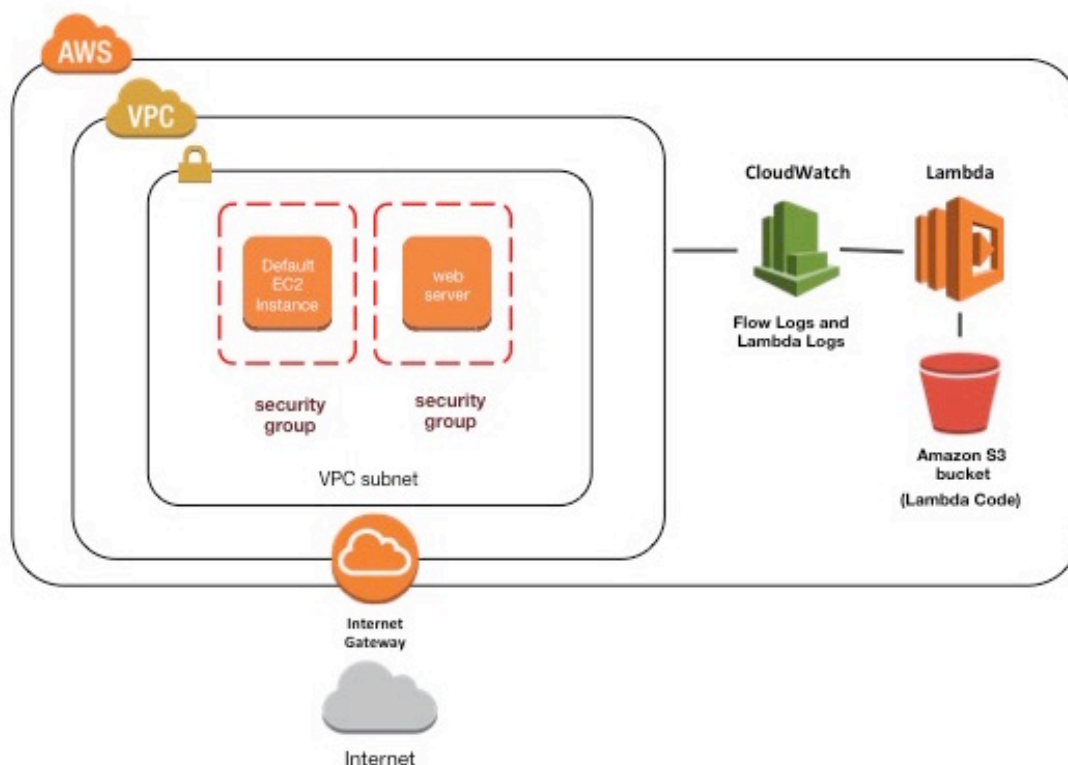
When the code is run to create or test the stack, an environment is set up that creates a honeypot with instances in a wide-open network. A honeypot is used simply to

Teri Radichel, teri@radicalsoftware.com

demonstrate the code and receive sample traffic from the Internet, which will start appearing minutes after all resources are deployed. This code could be altered and used in a real environment, in which case the network and the hosts would be appropriately secured. The honeypot VPC is set up with almost no traffic rules, except that the instances are in two separate subnets that prevent the instances from communicating with one another. One instance is configured to run an Apache web server and the other is run with no modifications to the default AWS Linux instance. The honeypot VPC is set up with almost no traffic rules, except that the instances are in two separate subnets that prevent the instances from communicating with one another. One instance is configured to run an Apache web server and the other is run with no modifications to the default AWS Linux instance.

The honeypot exists in one VPC with Flow Logs turned on. Flow Logs send traffic to CloudWatch. A CloudWatch subscription feeds Flow Logs data to a Lambda function that takes action based on data in a single flow. The Flow Logs data is parsed and output is logged when traffic matches specified criteria. In some cases, a security response is triggered by undesirable events. An S3 Bucket is created to hold the Lambda files. IAM roles are created to define permissions for the various resources in the stack. This is shown in Figure 1.

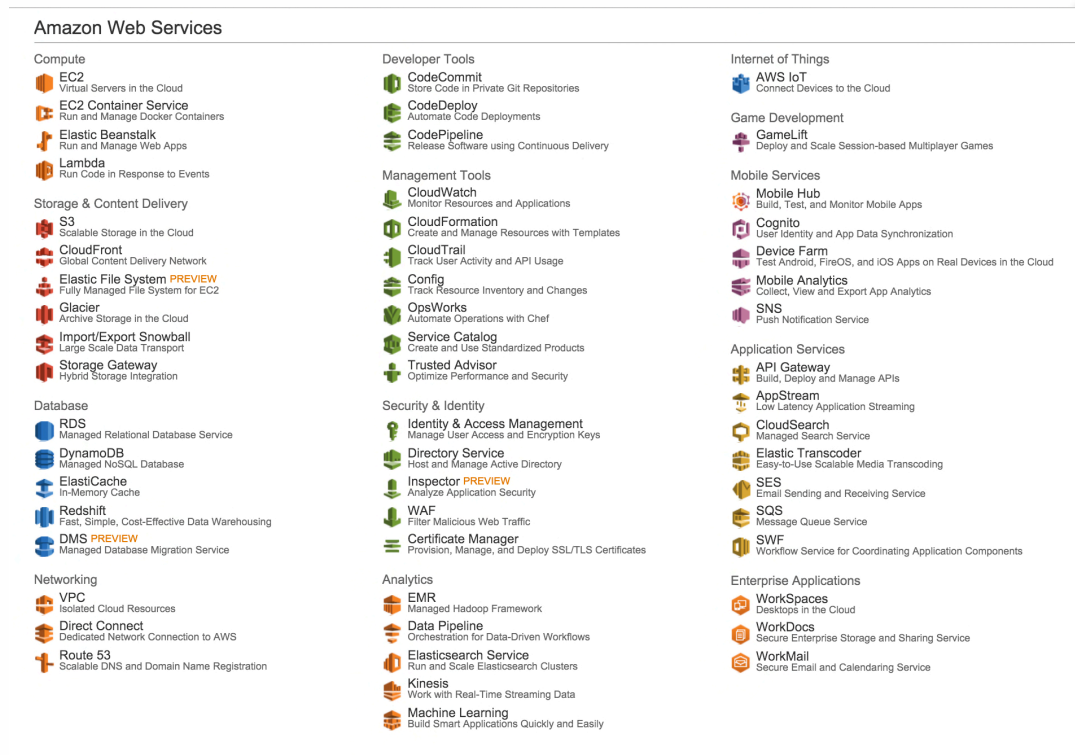
Figure 1: AWS Resources Created by Sample Code



Teri Radichel, teri@radicalsoftware.com

Resources created can be viewed by logging into the AWS console at <https://aws.amazon.com>, as shown in Figure 2.

Figure 2: AWS Console



Click on a service icon in the AWS console to view resources for that service. For example, the first icon, as seen in Figure 1, is “EC2.” Click on EC2 to view the EC2 instances that were created by the scripts. This is shown in Figure 3.

Figure 3: EC2 instances in AWS Console

<div> <div>Launch Instance</div> <div>Connect</div> <div>Actions ▾</div> </div>					
<div> <div>Filter by tags and attributes or search by keyword</div> </div>					
<input type="checkbox"/>	Name ▾	Instance ID ▴	Instance Type ▾	Availability Zone ▾	Instance State ▾
<input type="checkbox"/>	Radical Software Beta 2	i-e3bce739	t2.micro	us-west-2c	● running
<input type="checkbox"/>	Radical Software Beta 1	i-ecbce736	t2.micro	us-west-2c	● running

Teri Radichel, teri@radicalsoftware.com

In Figure 2, click VPC to view the VPC, Subnets, Route Table, Security Groups, NACL, and Internet Gateway that was created. Click on VPC on the left to view the VPCs. This is shown in Figure 4.

Figure 4: VPCs in AWS Console

<input type="checkbox"/>	Name	VPC ID	State	VPC CIDR	DHCP options set	Route table	Network ACL	Tenancy
<input type="checkbox"/>		vpc-765f6f1f	available	10.0.0.0/16	dopt-a76a5cce	rtb-485f6f21 p...	acl-495f6f20	Default
<input checked="" type="checkbox"/>	HoneyVpc	vpc-e4b34280	available	10.20.0.0/24	dopt-a76a5cce	rtb-8aa074ee	acl-23dc1947	Default

vpc-e4b34280 (10.20.0.0/24) | HoneyVpc

Summary | **Flow Logs** | Tags

You can create flow logs on your resources to capture IP traffic flow information for the network interfaces for your resources. [Learn more about flow logs](#).

Create Flow Log

Flow Log ID	Filter	CloudWatch Logs Group	IAM Role ARN	Creation Time
fl-4237d02b	ALL	autosec751499613737	arn:aws:iam::751499613737:role/FlowLogRole-FlowLogRole-15UZJNV48KU3Y	March 6, 2016 at 11:21:20 PM UTC-5

In Figure 4, to view Flow Logs for the VPC, click on the “CloudWatch Logs Group” link. This is shown in Figure 5.

Figure 5: Flow Logs in for ENIs (Elastic Network Interfaces)

Log Groups > Streams for autosec751499613737

Search Events | Create Log Stream | Delete Log Stream

Filter: Log Stream Name Prefix x

<input type="checkbox"/>	Log Streams	Last Event Time
<input type="checkbox"/>	eni-cc19d191-all	2016-03-07 00:33 UTC-5
<input type="checkbox"/>	eni-e01cd4bd-all	2016-03-07 00:21 UTC-5

In Figure 5, click on an ENI (Elastic Network Interface) to view the logs for that particular ENI. This is shown in Figure 6.

Figure 6: Flow Logs for a single ENI

Log Groups > Streams for autosec751499613737 > Events for eni-cc19d191-all

Filter:	Search for events	Date/Time:	2016/03/07 05:15:47 UTC (GMT)
Event Data			
▼ 2	751499613737	eni-cc19d191	5.58.76.88 10.20.0.9 39231 22 6 2 88 1457327747 1457327801 ACCEPT OK
▼ 2	751499613737	eni-cc19d191	91.224.161.42 10.20.0.9 47700 3392 6 2 80 1457327747 1457327801 ACCEPT OK
▼ 2	751499613737	eni-cc19d191	10.20.0.9 91.224.161.42 3392 47700 6 1 40 1457327747 1457327801 ACCEPT OK
▼ 2	751499613737	eni-cc19d191	10.20.0.9 132.163.4.101 123 123 17 2 152 1457327747 1457327921 ACCEPT OK
▼ 2	751499613737	eni-cc19d191	58.218.204.225 10.20.0.9 43861 3128 6 1 40 1457327810 1457327861 ACCEPT OK
▼ 2	751499613737	eni-cc19d191	10.20.0.9 58.218.204.225 3128 43861 6 1 40 1457327810 1457327861 ACCEPT OK
▼ 2	751499613737	eni-cc19d191	10.20.0.9 64.6.144.6 123 123 17 3 228 1457327944 1457328221 ACCEPT OK

The other resources created by this sample code can be viewed in a similar manner by clicking on the icons in the console for the related service. In Figure 2, click on Lambda, CloudWatch, IAM and S3 to view each of the resources created after executing the sample code. Click on CloudFormation to monitor creation and deletion of stacks.

2.7.5. Running the Sample Code

1. Set up an account at <http://aws.amazon.com/>. There is a free trial period for new users up to a certain limit.
2. Install Git from <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>
3. Clone the repository to get the code by running this command from the directory where the code should be downloaded:

```
$ git clone
https://github.com/tradichel/AWSSecurityAutomationFramework.git
```

4. Install the AWS CLI:

<http://docs.aws.amazon.com/cli/latest/userguide/installing.html>

5. To create the resources only execute this command:

```
$ ./run.sh CREATE
```

At the end of the command a message will be printed:

Teri Radichel, teri@radicalsoftware.com

```
--SUBSCRIPTION--
The subscription cannot be created until logs exist.
To check for logs: Log into AWS console, click CloudWatch,
click logs on left, then click on this log
group:  autosecxxxxxxxxxxxxx
Once logs appear run this script again
----END----
```

Follow these instructions.

6. Run this command to delete resources created by the above command:

```
./run.sh DELETE
```

7. Run the ping test which creates an instance that breaks the rules and gets automatically terminated after a snapshot has been created.

```
./run.sh PINGTEST
```

2.7.6. Code Execution – CREATE Mode

If the script is run to create the stack, the target instances are spun up in the honeypot VPC and do nothing but wait for traffic. Traffic will appear within a short amount of time as the instances are scanned by random IP addresses on the Internet.

The Lambda function receives the log stream and parses the log entries to look the patterns mentioned above:

- A REJECT in the logs will trigger the code to snapshot the instance and terminate it. A REJECT indicates that the instance-initiated traffic is not expected or acceptable.
- Outbound traffic on unexpected ports is logged. This traffic is not currently terminated because more analysis would be needed to determine which traffic is invalid or valid. Once this termination was made, instances exhibiting invalid traffic patterns could also be terminated.
- If an instance contacts a bad IP address in the Alert Logic top 10 list, a snapshot will be made and it will be terminated. The code uses the top 10 IP addresses sent to the Alert Logic mailing list in February 2016 to flag bad IP addresses. This

Teri Radichel, teri@radicalsoftware.com

could be handled differently by pulling it from a database, a configuration file, or some other means of matching traffic against a known bad IP list.

Figure 7 shows sample output logged by the Lambda function. This data could be in any format or logged to a database instead of written to logs. The point is simply to demonstrate that the data can be automatically analyzed near real-time and code can be written to respond to events in the logs.

Figure 7: Lambda Function Logs

Filter: Search for events	Date/Time: 2016/02/24 10:56:39 UTC (GMT)
Event Data	
▼ START RequestId: 4796337d-dae5-11e5-ad6e-a7cda8d9e5fc Version: \$LATEST	
▼ 2016-02-24T10:56:40.951Z 4796337d-dae5-11e5-ad6e-a7cda8d9e5fc OUTBOUND source 10.20.0.25 source port 80 destination 94.102.49.79 destination port 53 interface eni-9f50b0c2 instanceid i-034905d9	
▼ 2016-02-24T10:56:40.988Z 4796337d-dae5-11e5-ad6e-a7cda8d9e5fc EVENTS: 15 OUTBOUND ERRORS: 1 REJECTS: 0 BAD IPS: 0 NTP: 4 NTP DST: 64.71.128.26, 171.66.97.126, 173.230.144.109, 66.228.42.59, PORT 0: 2 PORT 0 DST: 94.102.48.193, 58.218.204.225,	
▼ END RequestId: 4796337d-dae5-11e5-ad6e-a7cda8d9e5fc	
▼ REPORT RequestId: 4796337d-dae5-11e5-ad6e-a7cda8d9e5fc Duration: 1809.36 ms Billed Duration: 1900 ms Memory Size: 128 MB Max Memory Used: 18 MB	
▼ START RequestId: 95042c58-dae6-11e5-932d-f52757e7aee0 Version: \$LATEST	
▼ 2016-02-24T11:05:58.348Z 95042c58-dae6-11e5-932d-f52757e7aee0 EVENTS: 10 OUTBOUND ERRORS: 1 REJECTS: 0 BAD IPS: 0 NTP: 8 NTP DST: 104.156.99.226, 107.170.224.8, 131.107.13.100, 216.218.254.202, 64.71.128.26, 171.66.97.126, 173.230.144.109, 66.228.42.59, PORT 0: 3 PORT 0 DST: 37.203.214.106, 94.102.48.193, 58.218.204.225,	
▼ END RequestId: 95042c58-dae6-11e5-932d-f52757e7aee0	
▼ REPORT RequestId: 95042c58-dae6-11e5-932d-f52757e7aee0 Duration: 16.37 ms Billed Duration: 100 ms Memory Size: 128 MB Max Memory Used: 18 MB	
▼ START RequestId: fa9a12bd-dae7-11e5-bef1-3f23e69a6672 Version: \$LATEST	
▼ 2016-02-24T11:15:58.256Z fa9a12bd-dae7-11e5-bef1-3f23e69a6672 EVENTS: 18 OUTBOUND ERRORS: 1 REJECTS: 0 BAD IPS: 0 NTP: 12 NTP DST: 131.107.13.100, 107.170.224.8, 104.156.99.226, 216.218.254.202, 104.156.99.226, 107.170.224.8, 131.107.13.100, 216.218.254.202, 64.71.128.26, 171.66.97.126, 173.230.144.109, 66.228.42.59, PORT 0: 7 PORT 0 DST: 80.90.88.235, 216.218.206.100, 184.105.247.223, 89.163.249.88, 37.203.214.106, 94.102.48.193, 58.218.204.225,	
▼ END RequestId: fa9a12bd-dae7-11e5-bef1-3f23e69a6672	
▼ REPORT RequestId: fa9a12bd-dae7-11e5-bef1-3f23e69a6672 Duration: 1.20 ms Billed Duration: 100 ms Memory Size: 128 MB Max Memory Used: 18 MB	

Connections triggering a REJECT or traffic to a bad IP are logged one line per event. These two actions trigger a snapshot of the instance initiating the connection and termination of the instances. When a snapshot is made and an instance is terminated, these actions are logged with the corresponding snapshot id and instance id. In addition, most types of outbound traffic will be logged on a single line per event.

Outbound connections for NTP and port 0 are logged in aggregate on one line rather than on individual lines. These entries were so prevalent in the logs they were a distraction. By default AWS instances contact public servers from the pool.ntp.org project so much of the traffic will be NTP traffic to those IP addresses. A significant amount of traffic to port 0 was present in this test. Traffic flows in VPC Flow Logs cannot pinpoint malicious traffic in the port 0 or NTP connections. Packet level inspection could be added by incorporating some other type of logging to determine if that traffic is valid or the traffic could simply be blocked or eliminated. NTP on the

Teri Radichel, teri@radicalsoftware.com

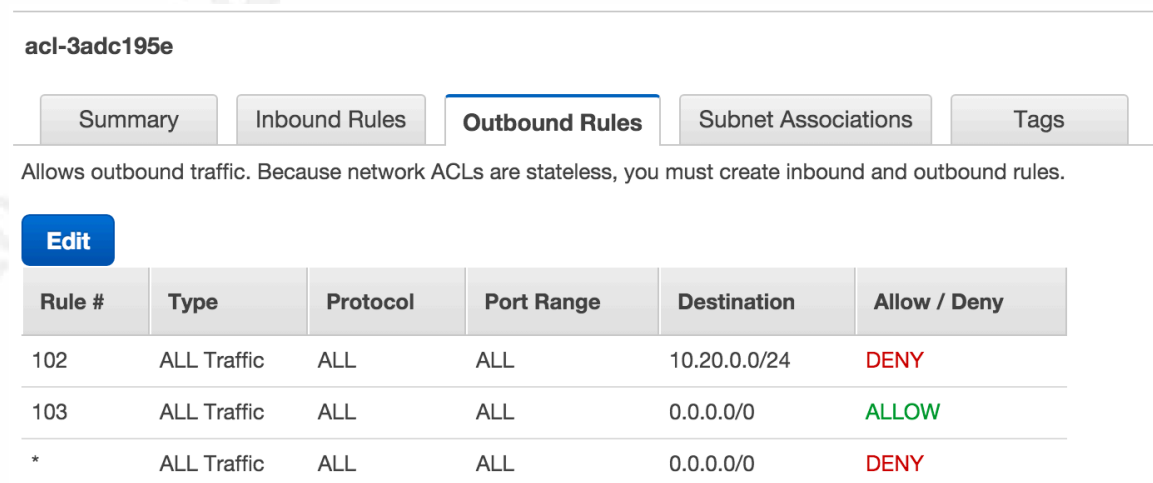
instances could be changed to use an internal, trusted NTP server instead of reaching out to the Internet. Port 0 could be blocked since that traffic is invalid.

As can be seen in Figure 7, the second log entry shows some interesting traffic. One of the instances is making an outbound connection on port 80 to an external IP address with a destination port of 53. That is quite odd and possibly, depending on the expected traffic patterns, something that could trigger instant snapshot and termination of the instance, as will be demonstrated in the following ping test. Initially the code was simply written to log outbound connections, which are unexpected from a web server or default instance with no additional services installed. An iterative approach to security automation would involve analyzing the logs and adjusting automated responses based on abnormal or non-compliant traffic patterns as they are discovered.

2.7.7. Code Execution – PINGTEST Mode

When the code is run in PINGTEST mode traffic is generated to force a REJECT. The instances reside in two subnets. The subnet rules created by this sample code disallows traffic between the two subnets via NACL rules with DENY entries. The VPC is created with CIDR 10.20.0.0/24. Traffic is denied to this CIDR block, as shown in rule 102 in Figure 8. If traffic is initiated by an instance to that CIDR, a REJECT will appear in the VPC Flow Logs.

Figure 8: Subnet NACL Denies Intra-VPC Traffic



The screenshot shows the AWS Network ACL console for 'acl-3adc195e'. The 'Outbound Rules' tab is selected. A description states: 'Allows outbound traffic. Because network ACLs are stateless, you must create inbound and outbound rules.' Below this is a table of rules:

Rule #	Type	Protocol	Port Range	Destination	Allow / Deny
102	ALL Traffic	ALL	ALL	10.20.0.0/24	DENY
103	ALL Traffic	ALL	ALL	0.0.0.0/0	ALLOW
*	ALL Traffic	ALL	ALL	0.0.0.0/0	DENY

Teri Radichel, teri@radicalsoftware.com

When run in PINGTEST mode, code is executed on one of the instances to ping the other instance in the background. This code can be found in the */resources/instances.json* file:

```
"UserData":
  { "Fn::If" :
    [
      "PingMe",
      { "Fn::Base64":
        { "Fn::Join": [ "", [
          "#!/bin/bash -e\n",
          "echo ping ",
          { "Fn::GetAtt" : [ "Ec2Instance1" , "PrivateIp" ] },
          " > /tmp/ping.sh\n",
          "cd /tmp\n",
          "chmod 777 ping.sh\n",
          "nohup ./ping.sh &\n"
        ] ] } },
      { "Ref" : "AWS::NoValue" }
    ]
  }
}
```

The instance that runs this code on instantiation will continue to ping the other instance until the Flow Logs generate a log event with a REJECT that is received by the Lambda function. The Lambda function will then parse the REJECT entry in the VPC Flow Logs, snapshot and terminate the instance. New Flow Logs will appear about every five minutes. An entry with a reject will appear:

```
2 751499613737 eni-ce6da593 10.20.0.5 10.20.0.25 0 0 1 27 2268 ... REJECT OK
```

The Lambda function will receive the reject when the logs are forwarded by the subscription. This may take some time depending on the configuration of the subscription. Once processed the snapshot will appear in the Lambda logs:

```
2016-03-07T07:02:45.923Z 973a3461-e432-11e5-8e57-ebad8623ef5e Created snapshot for volumeid: vol-8da27a34
snapshot id: snap-fff7fead
```

An example of this instance snapshot is shown in Figure 9.

Figure 9: Instance Snapshot

Name	Snapshot ID	Size	Description	Status	Started
	snap-fff7fead	8 GiB		completed	March 7, 2016 at 2:02:45 A...

Finally, the instance termination will appear in the logs and only one instance will remain running.

```
2016-03-07T07:22:44.523Z 626936b5-e435-11e5-83c2-
e30aae0d6c63 Terminated instance id: i-c3c99219
```

Note that for the purposes of this example the instance was simply terminated. In a production environment, AWS provides AutoScaling Groups that would provide a mechanism for automatically creating a new instance with the same configuration when an instance is terminated. For more information, please consult <https://aws.amazon.com/autoscaling/>.

3. Conclusion

Event driven security can help security teams be more effective and organizations to be more innovative at the same time. Event driven responses can prevent more known bad inputs from entering the environment, find non-compliant items faster, help pinpoint, and stop malware from traversing a network.

In order to automate, security teams need access to the data that drives the event driven processes. The data must be trusted and secure. Security teams need to be able to inject security review into points in the software development life cycle that allows review, detection, and blocking of known bad inputs in a read-only manner.

Security automation is the convergence of software and security in such a way that new forms of teams are required. A security automation team will consist of

Teri Radichel, teri@radicalsoftware.com

seasoned security professionals, not just software developers who are not well versed in security. The security automation will also have software developers and QA team members who understand the software development life cycle and can ensure the security automation software is well engineered and tested.

Shifting from a blocking to a non-blocking approach, without sacrificing security completely, can help organizations become more secure and more innovative at the same time. Rather than block software and changes, or let all software and changes in without any form of security review, security teams can automate review, intrusion detection, and response. Analysis of traffic patterns indicative of compromise are events on the network right now, versus new software or changes which may or may not pose a threat.

Teri Radichel, teri@radicalsoftware.com

4. References

- Amazon. (2016a). *Amazon Virtual Private Cloud User Guide*. Retrieved February 1, 2016, from aws.amazon.com:
<http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/vpc-ug.pdf>
- Amazon. (2015a, August). *Amazon Web Services: Overview of Security Processes*. Retrieved February 1, 2016, from aws.amazon.com:
<https://d0.awsstatic.com/whitepapers/aws-security-whitepaper.pdf>
- Amazon. (2016b). *AWS Assurance Programs*. Retrieved February 1, 2016, from aws.amazon.com: <https://aws.amazon.com/compliance/pci-data-privacy-protection-hipaa-soc-fedramp-faqs/>
- Amazon. (2015b, December). *Overview of Amazon Web Services*. Retrieved February 1, 2016, from Amazon: <https://d0.awsstatic.com/whitepapers/aws-overview.pdf>
- Bennett, C., & Tseitlin, A. (2012, July 30). *Chaos Monkey Released Into The Wild*. Retrieved February 1, 2016, from Netflix:
<http://techblog.netflix.com/2012/07/chaos-monkey-released-into-wild.html>
- Black Duck Software. (2015, April 15). *The Ninth Annual Future of Open Source Survey*. Retrieved February 28, 2016, from Black Duck Software:
<https://www.blackducksoftware.com/future-of-open-source>
- Brandwine, E. (2015, October). *AWS re:Invent 2015 | (NET403) Another Day, Another Billion Packets*. Retrieved February 1, 2016, from YouTube:
<https://www.youtube.com/watch?v=3qln2u1Vr2E>
- Cole, D. E. (2014, February). *SEC401: Security Essentials Bootcamp Style*. Phoenix, AZ.
- Collins, M. S. (2014). *Network Security Through Data Analysis: Building Situational Awareness*. Sebastopol, CA: O'Reilly Media.
- Hahn, D. (2015, October 12). *AWS re:Invent 2015 | (DVO203) A Day in the Life of a Netflix Engineer*. Retrieved February 1, 2016, from YouTube:
<https://www.youtube.com/watch?v=-mL3zT1iIKw&ebc=ANyPxKrCGn8UyJSkcH8RtC3Kkb5q4BXYe0caaLKAMNy cSCnfZGiBU7yQpsI4xj3dV8fyPUy8ytTm>

Teri Radichel, teri@radicalsoftware.com

- Hutchins, E. M., Cloppert, M. J., & Amin, Ph.D., R. M. (2011). *Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains*. Retrieved February 1, 2016, from Lockheed Martin: <http://www.lockheedmartin.com/content/dam/lockheed/data/corporate/documents/LM-White-Paper-Intel-Driven-Defense.pdf>
- Kent, K., & Souppaya, M. (2006, September). *NIST*. Retrieved February 1, 2016, from Guide to Computer Security Log Management: <http://csrc.nist.gov/publications/nistpubs/800-92/SP800-92.pdf>
- King, R. (2015, June 1). *How Netflix Manages Security in the Age of DevOps*. Retrieved February 1, 2016, from Wall Street Journal: <http://blogs.wsj.com/cio/2015/06/01/how-netflix-manages-security-in-the-age-of-devops/>
- Lietz, S., & Bretan, M. (2015, October). *AWS re:Invent 2015 | (SEC402) Enterprise Cloud Security via DevSecOps 2.0*. Retrieved February 1, 2016, from YouTube: <https://www.youtube.com/watch?v=fqjwlKKA-V4>
- Lucas, A. (2015, October). *You Tube*. Retrieved February 1, 2016, from AWS re:Invent 2015 | (SEC324) New! Introducing Amazon Inspector: https://www.youtube.com/watch?v=HjuEtMrWc_w
- MacDonald, N., & Young, G. (2015, April 15). *Best Practices for Securing Workloads in Amazon Web Services*. Retrieved February 1, 2015, from Gartner: <https://www.gartner.com/doc/reprints?id=1-2K4XVSV&ct=150729&st=sb>
- Marks, H. (2014, July 3). *Code Spaces: A Lesson In Cloud Backup*. Retrieved February 1, 2016, from Network Computing: <http://www.networkcomputing.com/cloud-infrastructure/code-spaces-lesson-cloud-backup/314805651>
- Observable Networks. (2015, December 3). *Endpoint Modeling 101 - A New Approach to Endpoint Security*. Retrieved February 1, 2015, from Slideshare.net: <http://www.slideshare.net/ObservableNetworks/endpoint-modeling-101-a-new-approach-to-endpoint-security>
- OWASP. (2016, January 20). *Logging Cheat Sheet*. Retrieved February 1, 2016, from OWASP: https://www.owasp.org/index.php/Logging_Cheat_Sheet

Teri Radichel, teri@radicalsoftware.com

OWASP. (2010, November). *OWASP*. Retrieved February 1, 2016, from OWASP Secure Coding Practices Quick Reference Guide:

https://www.owasp.org/images/0/08/OWASP_SCP_Quick_Reference_Guide_v2.pdf

Prendergast, T. (2016, February 6). *AWS Security How-To: Protecting CloudTrails Data*. Retrieved February 15, 2016, from The evident.io Blog:

<http://blog.evident.io/blog/2015/2/3/aws-security-how-to-protecting-cloudtrails-data>

Reith, G., & Peppe, B. (2016, January 27). Seattle Cloud Security Alliance. *Threat Intelligence*. Seattle, WA.

Right Scale. (2016, 2 9). *Cloud Computing Trends: 2016 State of the Cloud Survey*.

Retrieved February 28, 2016, from Right Scale Cloud Management Blog:

<http://www.rightscale.com/blog/cloud-industry-insights/cloud-computing-trends-2016-state-cloud-survey>

Rossman, H., & Shinn, B. (2015, October). *YouTube*. Retrieved February 1, 2016, from AWS re:Invent 2015 | (SEC303) Architecting for End-to-End Security in the Enterprise: <https://www.youtube.com/watch?v=nqaL5zJqFuo>

Schwartz, M. J. (2014, January 1). *Dark Reading*. Retrieved February 1, 2016, from Target Hackers Tapped Vendor Credentials: <http://www.darkreading.com/attacks-and-breaches/target-hackers-tapped-vendor-credentials/d/d-id/1113641>

Stamp, P. (2015, October 7). *New Security Services Launched at AWS re:Invent 2015—Amazon Inspector, AWS WAF, and AWS Config Rules*. Retrieved February 1,

2016, from Amazon Web Services Security Blog:

<https://blogs.aws.amazon.com/security/post/Tx2XKTVE4JWD0F9/New-Security-Services-Launched-at-AWS-re-Invent-2015-Amaon-Inspector-AWS-WAF-an>

Varia, J. (2011, January). *Architecting for the Cloud: Best Practices*. Retrieved 1 2016, February, from Amazon:

https://media.amazonwebservices.com/AWS_Cloud_Best_Practices.pdf

Teri Radichel, teri@radicalsoftware.com

- Verizon. (2015, 4 13). *2015 Data Breach Investigations Report*. Retrieved February 28, 2016, from Verizon: <http://www.verizonenterprise.com/DBIR/>
- Wild Packets. (2013, January 13). *The Cloud: Are You Getting What You Paid For?* Retrieved February 28, 2016, from Wild Packets Network Analysis and Monitoring Blog: <http://blog.wildpackets.com/tag/amazon-aws>
- Wittig, A. (2015). *Amazon Web Services in Action*. Shelter Island, NY: Manning Publications