Afstudeerverslag
Bijlage Document

Student:	Niels Loozekoot
Student Nummer:	15081214
Onderwijsinstelling:	De Haagse Hogeschool, Delft
Opleiding:	HBO-ICT: NSE
Afstudeerperiode:	2019-02-04 t/m 2019-05-31
Begeleidend Examinator:	Pieter Burghouwt
Expert Examinator:	Tony Andrioli
Bedrijf:	Secura
Afdeling:	Security Assessment (SA)
Begeleider:	Pim Campers
Opdrachtgever:	Ralph Moonen
Versie:	0.4
Datum:	2019-05-30

INHOUDSOPGAVE

APPENDIX A. DEFINITIES EN AFKORTINGEN

Hieronder bevind zich een lijst van verschillende afkortingen or termen die mogelijk niet bekend zijn door de lezer.

Definitie / Afkorting	Betekenis / Uitleg
AWS	Amazon Web Services (Cloud provider)
Azure	Microsoft Azure (Cloud provider)
gCloud	Google Cloud (Cloud provider)
EC2	Amazon Elastic Compute Cloud, a VPS
VPS	Virtual Private Server
Baseline	Hoofdstuk 5.1. / Appendix C. Research Automated Pentesting
OS	Operatie Systeem (Windows, Linux, MacOS)

Plan van aanpak

Cloud Security Pentesting Automation

Bedrijf: Secura

Laatste bewerking: 2019-02-08
Geschreven door: Niels Loozekoot
StudentNr: 15081214
Voor: De Haagse Hogeschool

INHOUDSOPGAVE

1. INLEIDING

De afstudeerstage bij Secura zal door Niels Loozekoot uitgevoerd worden tussen 4 februari en 31 mei. Dit document begint met het beschrijven van de opdracht en de eisen die gesteld zijn door de opdrachtgever. Vervolgens wordt de aanpak van de stage beschreven met een samenvatting en overzicht van de planning die tijdens de opdracht uitgevoerd zal worden.

2. OPDRACHT

Secura voert regulier pentests uit op Cloud omgevingen, deze kosten veel tijd en kunnen op een andere manier uitgevoerd worden per medewerker. Secura wilt de cloud omgeving waarin de systemen zich bevinden kunnen testen op configuratie fouten. Ook willen ze een groot deel van de crystal box pentests automatiseren en standaardiseren zodat alle systemen aan een basis ofwel 'baseline' voldoen ongeacht de hoeveelheid systemen binnen een bedrijf. De specialisten kunnen met deze resultaten verder werken en hoeven zich geen zorgen meer te maken over de configuratie van services. De tests worden allemaal op dezelfde manier uitgevoerd waardoor er geen verschil meer zit tussen de tests die uitgevoerd worden voor verschillende bedrijven.

3. EISEN

De eisen die opgedaan zijn uit een interview met de begeleider en opdrachtgever zijn onderverdeeld in de categorieën: Must have, Should have en Cloud have.

3.1. Must have

Deze lijst bevat de verschillende functies die de tool 'moet' bevatten volgens de opdrachtgever.
– Een lijst van systemen en services produceren die binnen de cloud omgeving draaien.
– Netwerk en logging van de cloud omgeving testen op configuratie fouten en baselines.
– De software detecteren die draait op OS'en (Bijv. Apache of NGINX onderscheiden).
– Modulair zijn voor het toevoegen van nieuwe baselines, software en operatie systemen.
– Kan baselines definiëren aan de hand van de softwareversie, OS of Service.
– De 'regels' van configuratie bestanden te vergelijken met een baseline.
– Default waardes van configuratie meenemen met de baseline.
– Bewijsvoering per baseline (Kort).
– Output in LaTeX naar de reporting tool.

3.2. Should have

De lijst die hieronder beschreven staan bevat onderdelen die de opdrachtgever graag zou willen zien indien de mogelijkheid zich voorstelt.

– kunnen detecteren wat het doel is van elke server (Mailserver, Website, etc).
– Controleren of interactie tussen services via SSL loopt.
– Alle PaaS instanties / services controleren op configuratie fouten en baselines.
– ACL's en Firewalls controleren op configuratie fouten.

3.3. Could have

In de onderstaande lijst zijn de eisen beschreven die in de tool verwerkt kunnen worden indien de tijd het toelaat.

– Uitgebreide bewijsvoering & uitleg per baseline voor reporting, bijv. waarom iets niet voldoet.

4. RISICOANALYSE

In dit hoofdstuk wordt behandeld welke risico's het project kan bevatten, wat de kansen zijn dat deze voorkomen en hoe deze voorkomen kunnen worden.

Omschrijving Risico	Kans	Minimalisatie Gevolgen	Prioriteit	Impact
De hoeveelheid tijd of werk die in het project gestoken dient te worden wordt onderschat	Klein	Het project goed analyseren en rekening houden met het omvang van het project.	Hoog	Hoog
Resultaat voldoet niet aan de verwachtingen van de opdrachtgever	Klein	PoC en Prototype maken die de opdrachtgever kan beoordelen.	Hoog	Hoog
Het ontwerp heeft fouten	Gemiddeld	Het ontwerp laten nakijken door de begeleidende examinator.	Gemiddeld	Gemiddeld
Opdrachtgever voegt nieuwe wensen toe	Gemiddeld	Goed noteren wat de eisen zijn en deze een tweede keer doornemen na het PoC.	Klein	Klein
Eisen blijken niet haalbaar te zijn	Klein	In de ontwerpfase kijken of het ontwerp kan voldoen aan de eisen die gesteld zijn door de opdrachtgever	Klein	Gemiddeld

5. AANPAK

Tijdens dit project zal een er een onderzoek plaatsvinden die als doel heeft om de API's van de 3 populairste Cloud providers te onderzoeken. Er wordt hierbij gekeken naar de mogelijkheden om informatie uit de API's te halen die nodig is om de opdracht uit te voeren. Ook zal er tijdens dit onderzoek gekeken worden naar verschillende talen, libraries en software die gebruikt kunnen worden om dit te realiseren met oog op onderhoudbaarheid voor het bedrijf.
De drie populairste Cloud providers volgens het onderzoek "Cloud Survey: Dutch Skies" door Computable in 2018 zijn:
– Amazon (AWS)
– Microsoft Azure
– Google Cloud
De projectmethode die toegepast wordt om dit project uit te voeren is een V-Model waterval. Deze is te zien in de onderstaande afbeelding waar hij gevisualiseerd is per stap. Dit model geeft weer hoe de normale waterval methode toegepast kan worden samen met input van de opdrachtgever. In dit model zal de VRAAG gerepresenteerd worden door de opdrachtgever die zijn wensen kan specificeren. Het AANBOD zal vervuld worden door Niels Loozekoot, de stagiair, die de opdracht maakt. Dit model moet van links naar rechts gelezen worden

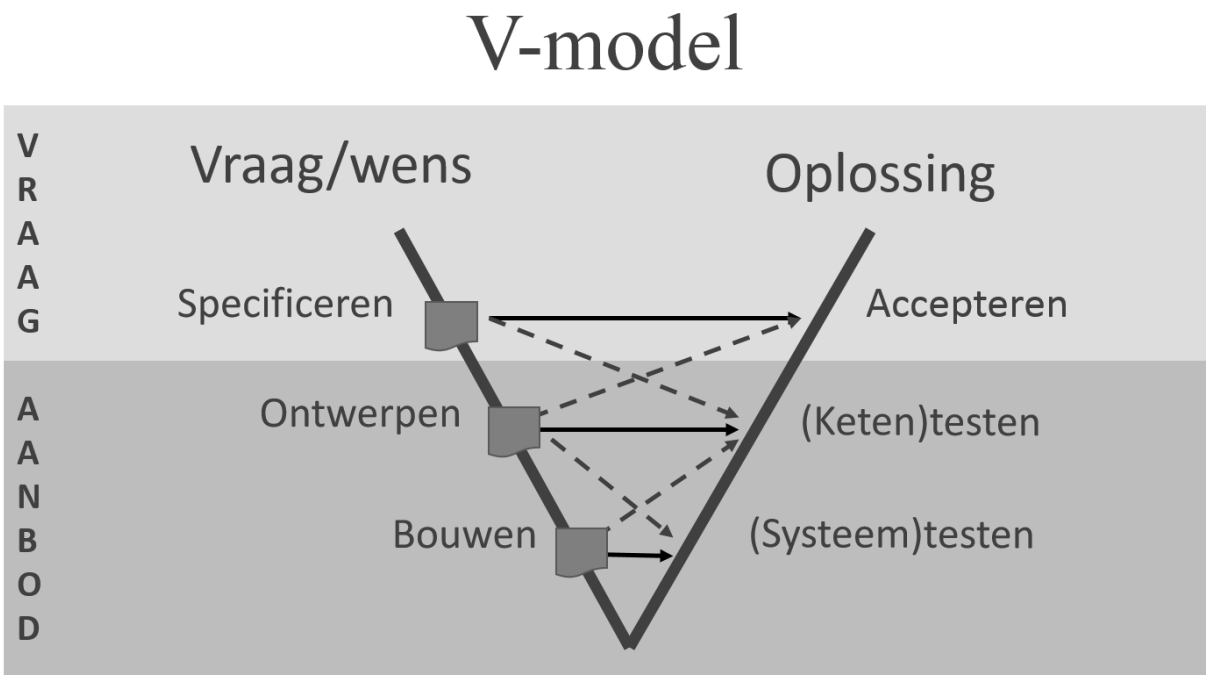

Image source: http://www.house-of-control.nl/watervalmodel-v-model.html

De waterval methode zal opnieuw uitgevoerd worden voor elk softwareproduct dat opgeleverd wordt: PoC, Prototype, Productierijp product. Hierdoor wordt het project verder uitgebreid voor iedere stap dat het doorloopt. De specificatie zal de vorm aannemen van 'feedback' in de producten Prototype en Productierijp product omdat specificatie van het product niet meer nodig is, maar de opdrachtgever wel de mogelijkheid geeft om feedback te geven op het vorige product.

5.1. PoC

Het Proof of Concept is een klein product dat één kernfunctie van de penetratie test automatisch uitvoert. Hiermee kan bewezen worden dat de functionaliteit haalbaar is en dat er met de onderzochte methode aan de eisen kan voldoen. Dit dient als bijproduct voor het onderzoek en als voorbereiding op het prototype.

5.2. Prototype

Dit product zal het merendeel van alle kernfunctionaliteit bevatten en een grote basis zijn van het productierijp product. Het is nog niet klaar voor productie maar geeft de opdrachtgever een duidelijk beeld van hoe het uiteindelijke product eruit ziet.

5.3. Productierijp Product

Dit product zal in staat zijn om aan de eisen van de opdrachtgever te voldoen en kan in zijn volledigheid de taken automatisch uitvoeren die nodig zijn om een baseline op te stellen bij cloud security pentesting.

6. PLANNING

Het project zal bestaan aan een aantal verschillende fases die afgerond worden met een product. De fases duren individueel 1-2 weken om af te ronden, en het formaat van de fase hangt af van de activiteiten.

Datum	Op te leveren product	Omschrijving
#1 Ma 4 Feb – Vr 8 Feb	Plan van Aanpak	Inlezen in opdracht en opstellen van de eisen
#2 Ma 11 Feb – 15 Feb	N.v.t.	Meelopen en onderzoeken hoe het proces binnen het bedrijf verloopt.
#3 Ma 18 Feb – Vr 1 Maart	Resultaten van onderzoek	Literatuur onderzoek en onderzoeken wat de beste taal / software is om deze opdracht mee uit te voeren
#4 Ma 4 Maart – Vr 15 Maart	Ontwerpen PoC Proof of Concept	Ontwerpen en bouwen van het PoC met een demo om af te sluiten.
#5 Ma 18 Maart – Wo 17 April	Ontwerpen Prototype Prototype	Ontwerpen en bouwen van het Prototype + verwerking van PoC.
#6 Do 18 April – Vr 24 Mei	Ontwerpen Productierijp Productierijp Product Documentatie	Ontwerpen, bouwen, demo'en en documentatie maken van productierijp product.
#7 Ma 27 Mei – Vr 31 Mei	N.v.t.	Pentest uitvoeren met klanten indien van toepassing en afronden product (buffer ruimte)

Appendix C. Haalbaarheidsonderzoek

Cloud Pentest Automation

Comparing Cloud Environments

Cloud Pentest Automation
Niels Loozekoot (Intern SA)
Version 3
2019-05-16

VERSIONHISTORY

Version	Version Date	Review by	Distributed to
V1	2019-02-18	Niels Loozekoot Pim Campers	
V2	2019-02-22	Niels Loozekoot Pim Campers	Expert Examinator Begeleidend Examinator
V3	2019-05-16	Niels Loozekoot Pim Campers	Expert Examinator Begeleidend Examinator

TABLE OF CONTENTS

1. INTRODUCTION

In this document a research by Niels Loozekoot will be conducted as part of his graduation internship for Secura. The research will be focusing on: “What possibilities do cloud providers offer that could help automate pentests and what information is required to realize this?“. At the start of this paper, a description of the project will be given to grant insight into the project and its goals. A boundary is set for what cloud providers will be inspected during this research; namely Amazon AWS, Microsoft Azure and Google Cloud.

A setup is made of what is required from the providers in order to complete this project, this is mostly related to data or actions that interact with the cloud environment. Once the project and its requirements are described, a research is performed into the solutions of the 3 most popular cloud infrastructure providers. This part of the research will see how the solutions function and what data can be retrieved or what actions can be taken. It concludes with a verdict if a solution, such as an API is suitable for the project or what steps need to be taken in order to make it usable.

As a big part of the project, but not of the research itself, appendix B contains research into the selection of a programming language / software package to use when developing the tool.

This document contains multiple terms and concepts relating to IT(-Security) or cloud hosting that can be hard to grasp for inexperienced developers or non-IT personnel.

2. HIGH LEVEL SUMMARY

This research looks to give a broad overview of the three Cloud Environments that Secura is looking to automate cloud security pentests for. The research analyzed what data is required from each company to allow for the required functionality to be implemented into the automated tool. Results from the research show that all three companies have an extensive API available that allow for full control of the cloud environment. Comparing the capabilities of the APIs with the list of required data a conclusion can be drawn if the cloud environment is suitable for automating penetration tests. Based on the results from the tests it can be concluded that all three cloud environments satisfy the requirements and can be used to automate tests. A table comparing the requirements to the results can be found in the Summary.

3. PROJECT DESCRIPTION

Secura is looking to automate a large part of the pentests on cloud environments specifically relating to configuration and environment settings in the cloud infrastructure and its machines. This project will provide a tool to Secura which will allow for a standardized way to test Cloud environments for security issues in network / PaaS / IaaS configurations. This will result in the knowledge that all systems within the environment adhere to a baseline of security which can then be used by the Secura security assessment team to further test for security threats.

An example of a baseline is that FTP servers do not allow guests or anonymous users to log in. The reason (for this is that it allows (partial) access to the system that is otherwise restricted to specific individuals. By looking at the configuration and default values we can determine if the FTP service running on a specific system adheres to this baseline. This is illustrated in image 3.1, in this image a Server hosts an FTP service. A security researcher discovers that the default configuration of that FTP service allows for anonymous logins. The way to resolve this problem is by setting a value on the configuration to false. If the configuration entry name is “Allow_anonymous” our baseline could be: “FTP Service X requires Allow_anonymous to be false”. By comparing the current value of the config to the baseline we can determine if the server adheres to this baseline or not.

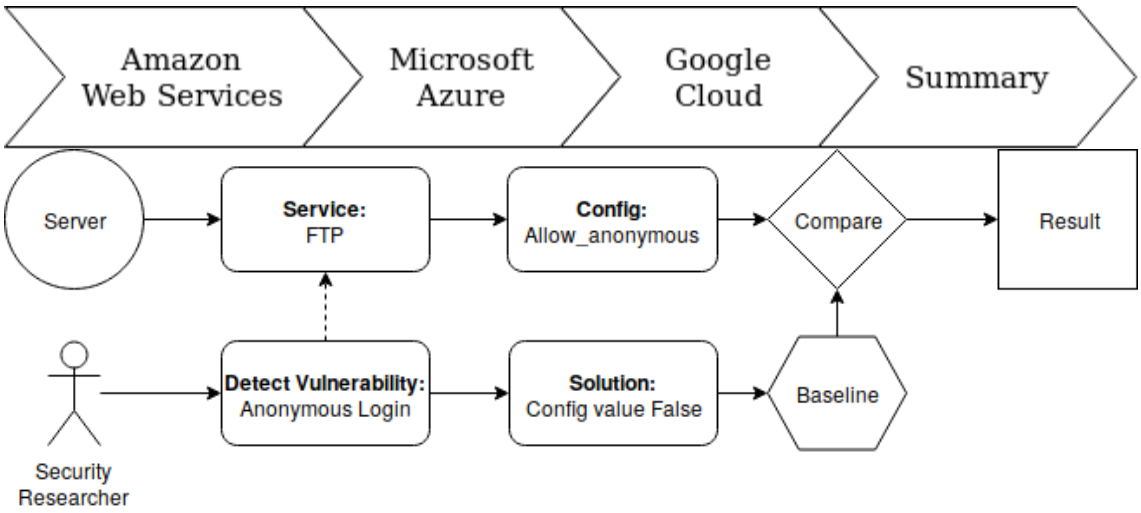

Image 3.1: Sketch of baseline FTP anonymous login

4. DATA REQUIREMENTS

This section will describe the data that is required from a cloud service provider in order to perform an automated penetration test. This data is based on experience with a regular penetration test and the requirements presented by the product owner. Should the cloud provider not be able to provide pieces of these data. Then sections of security might not be testable automatically for the specific cloud provider. Future cloud providers added to the tool's functionality should be able to provide this data in order to be fully integrated.

4.1. *Cloud infrastructure test*

Cloud setups are new and many companies are not familiar with the security issues that can come with miss-configuration of a PaaS setup or virtual network. This section of the data requirements focuses on the data that is required from the cloud service providers in order to perform a test on these items.

Server List

In order to automate the tool and allow it to scan all systems within an environment it needs to be able to retrieve a list of all IaaS / PaaS instances within the environment. Failure to retrieve this list means that the security analyst needs to manually provide a list of servers / services to investigate.

PaaS Configuration

In order to test PaaS instances for configuration errors, it is required that we can extract configuration values from them. Each type of PaaS instance can contain different configuration settings, if any. But can play a vital role in security if they revolve around access to resources.

Security Group List & Rules

Security Groups and their respective rules are important information to know about a cloud environment to depict if some or all ports are available from the outside, even though that might not be intended. Allowing the tool to look into the access control lists of the environment will enable the prevention of unauthorized access by blocking the routes to the target.

Users / Groups / Roles information

To gain insight into what users have access to the systems, what permissions they have and if any accounts might have unneeded settings / permissions it is important that we are able to extract the related information. The data retrieved should be a list of users and settings / permissions. A list of groups and their permissions as well as any other roles that might influence the access a user has.

Change- / Access- / Security logging

To prevent / detect abuse, detect unused users/groups/roles and much more, it is important that logging is enabled in its entirety and that these logs are accessible by the tool such that it is possible to view items such as, but not limited to:

- Login events
- Password changes
- Security alerts (from internal (N)IDS)

4.2. Crystal box test

Virtual machines running an operating system can be configured by a company in an unfathomable amount of ways. To test its security extensively all configuration files must be examined in order to determine if no security essential configurations have been adjusted. To do this to its fullest extent requires a lot of reading and comparing. The below list displays the data that is required to perform a crystal box test on a machine.

Server List

In order to automate the tool and allow it to scan all systems within an environment it needs to be able to retrieve a list of all IaaS / PaaS instances within the environment. Failure to retrieve this list means that the security analyst needs to manually provide a list of servers / services to investigate.

Execute commands

If it's possible to execute commands on the system and read the output, it will be possible to create a list of running processes / services and read configuration files using system commands like "ps -aux", "netstat --tulpan" or "cat" respectively. This will allow us to run any kind of command or script on the target as required, given that the permissions of the executing account are high enough.

5. CLOUD PROVIDERS

Although many companies offer forms of IaaS, taking all companies and their solutions into account is unfeasible. So basing our choices on a 2018 research by Computable called: “Cloud Survey: Dutch Skies” a focus will be put on the three most popular cloud service provider. Amazon Web Services, Microsoft Azure, Google Cloud Services concluding with a small summary of features that the cloud providers are capable of. In order to see if systems are suitable for test automation I will test the three cloud service providers against the data requirements mentioned above in 4. Inability to provide pieces of data can result in a diminished ability to test the environment for security vulnerabilities. API returns are not part of the research due to the sheer size of the return values (some returns are 4 pages worth of formatted JSON).

5.1. Amazon Web Services

Amazon Web Services (AWS) is a cloud platform by Amazon. It contains a multitude of PaaS solutions such as storage, databases as well as an IaaS solution called EC2. The ways to interact with AWS are via its website (web console) and an extensive API. In favor of efficiency and programming best-practices I will be focusing on the API that AWS offers. The API offered is extensive enough that it covers (almost) all functionality that the website offers. To ease usage Amazon provides a command line interface to interact with the API, we will be making use of this to shorten documentation and ease understanding of the actions taken.
Ubuntu: “apt install awscli” will install the Amazon CLI tool, further instructions can be found on the official Amazon website. It is installed by default on EC2 instances.

An important thing to understand about Amazon Web Services is that different names are used than people in the IT field are used to. An example of this is what Amazon calls “EC2” which stands for Elastic Compute Cloud. An instance of EC2 is an IaaS and closely resembles a VPS (Virtual Private Server).

Cloud infrastructure test

Server List

If a user is given API access to instances found on the AWS environment, they can use the API to retrieve a list of said instances via the “describe-instances”[1] API call.
AmazonCLI: “aws ec2 describe-instances”

This API call returns a list of EC2 instances the user has access to. Similar API calls exist for PaaS’s (i.e. “aws rds describe-db-clusters” returns a list of DBClusters in RDS (relational database service)).

PaaS Configuration

Amazon contains multiple API calls that can be used to retrieve configurations from PaaS instances[7]. An example of this is the command: “aws s3api get-bucket-acl". Using this command in combination with the “aws s3api list-buckets” will allow for the program to obtain a list of buckets and for each bucket get the configuration and access control list associated with it. This will ultimately allow for the extraction of the ACL configuration for S3 Storage buckets. These types of calls are available for other types of instances too.

Security Group List & Rules

The API that Amazon provides has the ability to describe all security groups for EC2 instances, showing what ports are allowed through the firewalls for both input and output rules. This can be used to check for mistakes in the ACL configuration or security groups.

Users / Groups / Roles information

AWS Identity and Access Management (IAM)[8] section of the API contains many different calls allowing for the management of users, groups, roles, policies, keys and more. These can be used to gain the required information.

Change- / Access- / Security logging

Logging for IaaS and PaaS is possible but requires a service called CloudWatch, charges can apply for using this service, but it does allow for monitoring events and alerts in access logs as well as keep track of authentication. So, it is possible, but not for free out of the box as with the other API calls. Implementing this into the box might require a check if the service is being used or not before continuing with testing specific baselines against it.

5.1.1. Crystal box test

Server List

If a user is given API access to instances found on the AWS environment, they can use the API to retrieve a list of said instances via the “describe-instances”[1] API call.

AmazonCLI: “aws ec2 describe-instances”

This API call returns a list of EC2 instances the user has access to. Similar API calls exist for PaaS’s (i.e. “aws rds describe-db-clusters” returns a list of DBClusters in RDS (relational database service)).

Execute Commands

Amazon seems to include a few methods of running commands on the servers. If their “SSM Agent” (AWS Systems Manager Agent) is installed on the server (installed by default on newer instances). It is possible to call the “send-command”[2] API call. This allows the API to execute a command on the server via the API as the ‘root’ user. An example of the Amazon CLI API Call to execute a command is shown below. In this example, the SSM is used to send the “ifconfig” command.

AmazonCLI: “aws ssm send-command --instance-ids "i-077ab1e66f3809687" --document-name "AWS-RunShellScript" --comment "IP config" --parameters commands=ifconfig --output text”

This command returns a command-id which can then be used in other commands to gain information about the results of the command. The API call “list-commands”[3] provides insight into the status of commands and what instances they are running on. Looking into the output of a specific command is possible with the command “get-command-invocation”[4] which specifies a single instance to look at and retrieves details relating to the command. Finally, the API call “list-command-invocations”[5] can be used to retrieve the results of an API call on one or multiple instances and optionally displays all the outputs of each instance the command was run on. An example of the command is seen below.

AmazonCLI: “aws ssm list-command-invocations --command-id "ab3bca4c-46c6-49fc-9e08-7388445bb06d" –details”

The second method to approach executing commands on an EC2 instance is via SSH authentication. With a ssh public-private SSH keypair bound to an instance it is possible to remotely execute commands on the server. This allows us to to provide the list of services and configuration file contents via normal commands without using the API. This is cleaner and faster than the API but does require a public key to be assigned to each instance that needs testing. This is not preferred as it requires extra steps to be taken and opens the server up to an extra user.

Amazon SSM is a requirement for executing commands on IaaS instance

5.2. Microsoft Azure

Microsoft Azure is a cloud computing service created by Microsoft. It provides different Iaas, PaaS and SaaS services for customers to use without hosting them themselves. Microsoft Azure, similar to AWS offers a web interface to interact with the environment as well as a REST API. Also similar to AWS, they provide a clean tool called "Azure CLI"[6][9] that we will be using here as well.

Microsoft Azure makes use of Resource groups to assign resources to networks or machines. This can be seen as a "group" with multiple services such as a virtual machine, a network interface, a public IP address and a virtual network assigned to it. These listed are all created simultaneously when creating a virtual machine as they are parts of that virtual machine, but you can also create an SQL database service and add that to the same resource group.

5.2.1. Cloud infrastructure test

Server List

Azure has an API call available to list all SQL databases, storage accounts and VMs, this includes the type of VM. For the VM, retrieving the IP address requires a second command.

Azure CLI: "az vm list"
Azure CLI: "az vm list-ip-addresses" or
Azure CLI: "az vm nic list -g resourcegroup --vm-name vmNameHere"

Azure contains a few different API calls that result in different responses but ultimately contain the same information.

PaaS Configuration

Azure allows for the retrieval of special PaaS configurations from the systems depending on the PaaS type there is a different set of API calls. For example, SQL databases have threat-policies whilst storage accounts have network-rule and container immutability-policy as endpoints. More endpoints exist for different types of configuration, but these endpoints do exist and they are retrievable via the Azure API.

Security Group List & Rules

The API provided by Microsoft Azure allows for the listing / showing details of security alerts generated by the security center. It also allows us to obtain a list of security "Just in Time" network policies or workspace settings.

Users / Groups / Roles information

Microsoft Azure offers the management of access to resources via roles. This happens through the use of RBAC (Role-Based access control) and can be found in Azure's role API branch[10][11].

Change- / Access- / Security logging

Azure offers a (paid) service called Azure Monitor Service. Through the user of this service, users can aggregate logs and create/manage alerts. The API is able to hook into these settings and view logs / alerts. If employed/deployed within the environment, this data can be checked.

5.2.2. Crystal box test

Server List

Azure has an API call available to list all SQL databases, storage accounts and VMs, this includes the type of VM. For the VM, retrieving the IP address requires a second command.

Execute Commands

The API Azure offers contains the possibility to execute commands using the “run-command” endpoint. An example of running the “ID” command (for Linux) on a VM using the Azure CLI can be done doing this:
Azure CLI: az vm run-command invoke -g Trial -n testvm --command-id RunShellScript --scripts "id"

This uses the resource group “Trial” to locate the relevant IP, VM and resources and the “-n” parameter to specify the VM name. The output of the command is given as soon as execution completes.

5.3. Google Cloud

Google Cloud is a cloud service provider created by Google, the subcompany of Alphabet. It provides a multitude of Iaas, PaaS and SaaS services for customers to use without hosting these themselves. Similar to both other services, Google Cloud also offers a web interface, an API and an additional command line interface tool that allows for easy access to the API called "gcloud"[12][13].

5.3.1. Cloud infrastructure test

Server List

A Google account that has the given permissions to look into and manage the environment can, within the Google cloud environment create a listing of all services per type. An example of this is the listing of all 'compute', or 'SQL' instances (IaaS/PaaS respectively).

Google CLI: "gcloud compute instances list"
Google CLI: "gcloud sql instances list"

PaaS Configuration

Google cloud offers extensive information regarding the configurations of PaaS. Via the API both the listing and modification of an instance is possible.

Google CLI: "gcloud sql instances describe testdatabase" where testdatabase is the instance name.
This lists all configuration settings of a SQL server instance such as backup configurations.

Security Group List & Rules

Security groups do not directly exist, but service accounts and roles take their place. The API allows for listing of both the accounts and the roles. It also allows for the full management of the firewalls and ACL's surrounding it.

Google CLI: "gcloud iam service-accounts list"
Google CLI: "gcloud iam roles list"
Google CLI: "gcloud compute firewall-rules list"

Users / Groups / Roles information

The solutions that Google cloud offers allows for full management of users, permissions and settings surrounding them. Google accounts are used to manage access at a high level.
Google CLI: "gcloud auth list"

The above command is a command that lists the user accounts that have privileges on the Google Cloud Platform.

Change- / Access- / Security logging

Logging inside of Google happens by default and mostly for free, unlike the services that Amazon and Azure offer. But from Niels' research, the logging does seem to be more limited than the services that both Amazon and Azure have available. The logging is available via the API where it can be read.
Google CLI: "gcloud logging logs list"

5.3.2. Crystal box test

Server List

A Google account that has the given permissions to look into and manage the environment can, within the Google cloud environment create a listing of all services per type. An example of this is the listing of all ‘compute’, or ‘SQL’ instances (IaaS/PaaS respectively).

Google CLI: “gcloud compute instances list”
Google CLI: “gcloud sql instances list”

Execute Commands

Command execution is a slightly different scenario than what Amazon AWS and Microsoft Azure offer. The execution of commands does not seem to be possible to do directly via the API itself. Instead the API allows for the execution of commands via SSH. This is possible on both AWS and Azure but not preferred as it creates modifications on the customers’ file system. Google however has a management system in place for the SSH public/private keypairs used. So via the API it can automatically add SSH keys to the authorized_keys file but at the same time use the metadata API calls to remove the SSH Key from the project which will then proceed to remove keys from the authorized_keys files.

Google CLI: “gcloud compute ssh instance-1 --command id”

The command shown above is an example of how the command “id” can be execution on the system “instance-1”.

6. SUMMARY

During this research we tried to answer the question: What possibilities do cloud provider offer that could help automate pentests? It was determined that all cloud service providers only had an API available as solution that can be used to 'easily' automate testing. Another method of access included the website of the provider. All three of the providers also offered a command line interface tool to easily utilize the API. Looking at all the data requirements for the project and each cloud provider we can determine if they are suitable to use for this project. In the table below is an overview of all cloud providers and each data requirement. The suitability of the provider is based on the amount of 'Yes' values for data requirements.

Data Requirements	Amazon Web Services	Microsoft Azure	Google Cloud
Server List	Yes	Yes	Yes
PaaS Configuration	Yes	Yes	Yes
Security Group List / Rules	Yes	Yes	Yes
Users, Groups, Roles: Info	Yes	Yes	Yes
Logging	Yes (Paid)	Yes (Paid)	Yes
Execute Commands (IaaS)	Yes	Yes	SSH Only
Suitable	**Yes**	**Yes**	**Yes**

Looking at the results of the research we can determine that it is indeed possible to obtain the required data from the cloud service providers. With this knowledge we can look into methods of automating the cloud security assessments. In Appendix B. a research has been performed utilizing the results from this research. It focusses on the actual implementation of the tool, looking at the different programming languages and software kits that are available and what their pros / cons are in relation to the subject. Results from this research conclude that the best approach to satisfy all requirements and allow for future adjustments is to make use of the Python programming language to develop the tool.

REFERENCS

[1] https://docs.aws.amazon.com/cli/latest/reference/opsworks/describe-instances.html
[2] https://docs.aws.amazon.com/cli/latest/reference/ssm/send-command.html
[3] https://docs.aws.amazon.com/cli/latest/reference/ssm/list-commands.html
[4] https://docs.aws.amazon.com/cli/latest/reference/ssm/get-command-invocation.html
[5] https://docs.aws.amazon.com/cli/latest/reference/ssm/list-command-invocations.html
[6] https://docs.microsoft.com/en-us/cli/azure/install-azure-cli-apt?view=azure-cli-latest
[7] https://docs.aws.amazon.com/cli/latest/reference/s3api/index.html#cli-aws-s3api
[8] https://docs.aws.amazon.com/cli/latest/reference/iam/index.html
[9] https://docs.microsoft.com/en-us/cli/azure/reference-index?view=azure-cli-latest
[10] https://docs.microsoft.com/en-us/azure/role-based-access-control/role-assignments-cli
[11] https://docs.microsoft.com/en-us/cli/azure/role?view=azure-cli-latest
[12] https://cloud.google.com/sdk/docs/quickstart-debian-ubuntu?hl=nl
[13] https://cloud.google.com/sdk/gcloud/?hl=nl
https://aws.amazon.com/answers/security/aws-securing-ec2-instances/
https://d1.awsstatic.com/whitepapers/Security/AWS_Security_Best_Practices.pdf

APPENDIX C.A. PRODUCTION APPROACH RESEARCH

1. INTRODUCTION

Secura is looking to develop a security tool that is capable of automatically performing a cloud penetration test based on the configurations of its components. In order to realize this, a research was conducted into the functionality of the most common cloud security platforms. The research question: “What software or computer language can best be used to automate cloud security pentests” is a continuation of the previous research on “What possibilities do cloud provider offer that could help automate pentests?“.

In order to start the active development of the tool a research must be conducted to deduce the programming language or software packet that can best be used. Using the search engine: Google, Niels looked for software that could be used to automate different functions (inside a cloud environment). These will be compared against each other based on their features and their ability to fulfill the requirements set by the product owner.

2. REQUIREMENTS

Based on the requirements of the project, a list of requirements was created for the different solutions to test on. These will not be tested but will rather be judged based on an estimated guess. This will be performed by collecting information from official sources and looking online at information provided by the community.

1. Able to detect systems /services on an unknown environment
2. Able to detect the type of software running on a system
3. Able to run baselines based on attributes of the system (version/OS)
4. Able to integrate with one or multiple cloud environments
5. Able to integrate new types of services (PaaS)
6. Able to create and compare baselines
7. Able to output results to reporting files/API

3. SOLUTIONS

Based on multiple google searches for automating tasks in corporate systems or cloud environments the following few software packages piqued interest as they mentioned features that matched the requirements of this project, such as updating configurations (which requires accessing/reading the file) or updating software (which requires executing commands).

- Ansible[3]
- Chef[2]
- Rudder[1]

The rest of this research will focus on these 3 software packages as possible solutions. Writing the software ourselves is not preferred but considering the requirements it is a feasible solution to match them all. So the solutions will also be compared against these software packages and should the results indicate that writing the software ourselves is a better solution, an investigation into the language to use for writing the program will be conducted.

4. COMPARISON

In this chapter a comparison between these 4 solutions (3 software, 1 custom program) will be made. This will be done based on the functionality that both the website and documentation advertise as well as the requirements set up by the product owner.

4.1. Ansible

Ansible is a software package that allows for automating the deployment of applications and the management of systems. They allow for rolling out complex systems and environments in an easy fashion. Through the use of their special modules they can integrate with all 3 of our platforms as well as many others, boasting integration of over 750+ built-in modules. Ansible supports automation for VMWare, Cisco and Windows as well.

4.1.1. Comparing Requirements

Ansible is capable of completing most of the requirements. Especially with the built in modules Ansible seems to be capable of automating most of the required functionality. Some of the features that Ansible seems to miss are:

1. Able to detect systems /services on an unknown environment
3. Able to run baselines based on attributes of the system (version/OS)
5. Able to integrate new types of services (PaaS)

Although Ansible does seem capable of network discovery, research indicates that automating both discovery and checking for baselines seems nearly impossible for the flexible workstyle that Secura works with, having multiple customers, all with their own platform, network setup and settings.

4.1.2. Major Pro's

Some special features or functionality that Ansible seems to offer might not be related to the requirements of the project but should still be considered a big selling point in the way of comparison.

The services that Ansible offers are currently available for free with a paid version available for more functionality and support. Paid version's pricing is around $10,000 USD per managed node per year going up to $17,500[5]. The free version offers a command line tool and framework whilst the 'Tower' (Paid) version offers a full web application for simplification of its usage.

Source code that Ansible offers is available on GitHub. Having open source code and over 25 commits in the last hour (at time of checking) makes it an insanely active repository and good option when choosing software. Github Repository: https://github.com/ansible/ansible

4.2. Chef

Chef is a software package that offers powerful infrastructure automation, ensuring that systems are configured consistently and continuously in any environment, at any scale. Chef works with defining resources which is practically an action with some extra options such as the installation of a package with a specific version.
Example:

```
package 'tar' do
  version '1.16.1'
  action :install
end
```

These can be combined with other items such as templates to create automatic scripts called cookbooks. Both the cookbooks and recipes can be configured with other plugins to interact with cloud environments such as Azure.

4.2.1. Comparing Requirements

Chef is a software package capable of completing some of the requirements. Chef seems to lack its functionality in automatic discovery and adjustability to flexible / multiple environments. Some of the features that Chef seems to miss are:

1. Able to detect systems /services on an unknown environment
3. Able to run baselines based on attributes of the system (version/OS)
5. Able to integrate new types of services (PaaS)
6. Able to create and compare baselines

Unlike Ansible, Chef seems to lack any kind of discovery of both network devices and cloud instances.

4.2.2. Major Pro's

Some special features or functionality that Chef seems to offer might not be related to the requirements of the project but should still be considered a big selling point in the way of comparison.

The services that Chef offers are currently available for free with a paid version available for more functionality and support. A quick glance at these functions seem limited. Paid version's pricing is $72 or $137 per node[4].

Source code that Chef offers is available on GitHub. Having open source code and 3 commits in the last hour (at time of checking) makes it an extremely active repository and good option when choosing software. GitHub repository: https://github.com/chef/chef

4.3. Rudder

Rudder is a continuous configuration solution which combines configuration management and continuous audit. It is dedicated to production infrastructure needs.

4.3.1. Comparing Requirements

Rudder is a software package that seems to be the most lacking in matching the requirements. The free version has no support for customized reporting, Debian/RHEL 5 etc. and Windows 2012 R2 or higher meaning that an annual price per machine must be paid in order to obtain those features. This would be impossible for Secura to realize as the machines change per company they work with. Some of the features that Rudder seems to miss are:

1. Able to detect systems /services on an unknown environment
2. Able to detect the type of software running on a system
3. Able to run baselines based on attributes of the system (version/OS)
4. Able to integrate with one or multiple cloud environments
5. Able to integrate new types of services (PaaS)
6. Able to output results to reporting files/API

4.3.2. Major Pro's

Some special features or functionality that Rudder seems to offer might not be related to the requirements of the project but should still be considered a big selling point in the way of comparison.

The services that Rudder offers are currently available for free with limits imposed and a paid version available with unlocked functionality and SLA style support. The price increases per machine but pricing is not publicly available and requires contact with the business first.

Source code that Rudder offers is available on GitHub. Having open source code and 3 commits in the last day (at time of checking) makes it an active repository and good option when choosing software.
GitHub repository: https://github.com/Normation/rudder

4.4. Custom Program

A program written in a language such as C/C++, Python, PHP, Perl, Java, Ruby or one of the many other languages gives a lot of opportunity. New features can be added on the fly and all requirements that are made possible by the cloud provider should be achievable in a self-written program. The downside to this ordeal is that this requires constructing the program. Including designing the software and programming it with all functionality. From all these languages, Python, followed by C/C++ are the most likely choices because they are most widely known by Developers and all three cloud providers offer an extensive software development kit to work with their API.

4.5. Comparison diagram

Based on the results of the previous information collected table 4.1 has been constructed showing the requirements. For limits in display only numbers of the requirement will be used in the table. Below are the requirements repeated:

1. Able to detect systems /services on an unknown environment
2. Able to detect the type of software running on a system
3. Able to run baselines based on attributes of the system (version/OS)
4. Able to integrate with one or multiple cloud environments
5. Able to integrate new types of services (PaaS)
6. Able to create and compare baselines
7. Able to output results to reporting files/API

Requirement	Ansible	Chef	Rudder	Custom
1	✓	X	X	✓
2	✓	✓	X	✓
3	X	X	X	✓
4	✓	✓	X	✓
5	X	X	X	✓
6	✓	✓	✓	✓
7	✓	✓	X	✓
Usable	**No**	**No**	**No**	**Yes**

Table 4.1

5. SUMMARY

From the three software packages that were researched, Ansible came out as the strongest choice of the three based on the requirements it fulfilled, it however lacked a lot of functionality unlike creating a self-made program. From this research we notice a consistent lack of ability to adjust and adapt to new networks and handle them in the same way. Most use SSH as a form of access, which is a less preferred method, even though Amazon AWS and Microsoft Azure have a method of performing the tasks without being given access to the server. To summarise: All three researched options consistently lack in functionality to match the requirements.

6. CONCLUSION

Based on personal experience with the language. Developing a new application for Secura in Python is the best option. This application is able to meet all requirements set out by the product owner and due to the knowledge base of Secura and its employees, support can be given for upgrades even after the intern completes the project.

REFERENCES

[1] https://www.rudder.io/en/ 2019-02-20
[2] https://www.ansible.com/ 2019-02-20
[3] https://www.chef.io/chef/ 2019-02-20
[4] https://www.g2crowd.com/products/chef/pricing 2019-02-20
[5] https://www.ansible.com/products/tower/pricing 2019-02-20
[6] https://www.rudder.io/en/pricing/subscription/ 2019-02-21

https://technology.hcs-company.com/a-short-comparison-of-ansible-chef-puppet-and-saltstack/

Appendix D. Program Documentation

Git .MD files

Cloud Pentest Automation
Niels Loozekoot (Intern SA)
Version 1
2019-05-28

CONFIDENTIAL

1. README.MD

Cloud Pentest Automation

Usage: python3 main.py -h

```
[Secura] Automated Cloud Pentesting Tool

Usage:
  main.py [--company=Secura] [--debug=1]
  main.py (-h | --help)
  main.py --version

Options:
  -h --help                 Show this screen.
  --version                 Show version.
  --debug=N                 Debug Level [0-2] [default: 1].
  --company=Secura          Company title [default: Secura].
```

Credential Setup

All credentials can be entered into the settings.conf file.

AWS

https://console.aws.amazon.com/iam/home?nc2=h_m_sc#/security_credentials Create access key

Setting the scope

In order to only scan the hosts required for the test, a scope can be defined. The scope, located in `scope.conf` is a list of IP addresses and identifying names / resource groups that can be used when comparing against the data retrieved from the cloud provider.

Currently uses CIDR notation for IP addresses. If no IP is present, a name MUST be specified.
No Wildcard or Regex available yet.

```
# To scan ALL use 0.0.0.0/0 as scope.
instances:
  - 0.0.0.0/0
  - testbucket-abcf534
```

2. INSTALL.MD

Install MySQL

```
sudo apt install mysql-server
sudo mysql_secure_installation
```

Setup Mysql

```
    y/Y for password plugin
    0 for password strength (Use 1 or 2 for production)
    Set up strong MySQL root password (preferably random)
    y/Y remove anonymous user
    y/Y disallow root login remotely
    y/Y remove test database and access to it
    y/Y reload privilege table
```

```
sudo mysql
mysql> CREATE DATABASE cloud_pentest_automation;
mysql> GRANT ALL PRIVILEGES ON cloud_pentest_automation.* TO 'pentest_manager'@'localhost' IDENTIFIED BY 'STRONGPASSWORD';
```

STRONGPASSWORD is the password to use, ensure this is a random password of minimum 16 random alphanumeric characters. (can include special characters if properly escaped). Recommended entropy is 80-127 bits. Example: (EUb5Qa8p,zTeD6!jGw) use: https://www.random.org/passwords/?num=3&len=21&format=plain&rnd=new when in doubt. Especially in production.

Import Database

```
sudo mysql cloud_pentest_automation < database.sql
```

Creating Development environment:

```
python3 -m venv ../venv
source ../venv/bin/activate
pip3 -V
pip3 install -r requirements.txt
```

Install SSM Agent (Required for AWS)

Ubuntu has the SSM agent installed by default, but Amazon

CentOS / Redhat

```
sudo yum install -y https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux_amd64/amazon-ssm-agent.rpm
sudo systemctl start amazon-ssm-agent
```

Windows

```
https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/windows_amd64/AmazonSSMAgentSetup.exe
```

openSUSE

```
sudo zypper install https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux_amd64/amazon-ssm-agent.rpm
sudo systemctl start amazon-ssm-agent
```

3. DEVELOPMENT.MD

New Cloud Provider

1. Copy AWS.py (no template available yet)
2. import relevant instances and serializers
3. add relevant settings to settings.conf
4. create a 'session' if possible
5. set the `self.name`
6. set the `self.config` provider correctly
7. program `discover_instances` to correctly identify the instance types
8. program `get_config` to grab 'provider wide' settings

New IaaS Instance Setup

1. Copy instance_template_iaas.py to a new file called: instance_.py Example: instance_ec2.py
2. Update the class name to match the new instance
3. Update the docstrings
4. Set the `TemplateInstance` in `super(TemplateInstance, self).__init__(instance_id)` to your new class variable
5. Correctly set the provider and product name. (Case sensitive)
6. Program `get_os()` to correctly determine linux vs windows without executing commands.
7. Program the `create_command()` to create and execute a command
8. Program the `get_command_output()` to retrieve the output of the `create_command()`
9. Program the `execute_command()` to execute a command on an instance (combine create and execute)
10. In the respective cloudprovider, import the new service and add a statement to create an instance of the class.

New PaaS Instance Setup

1. Copy instance_template_paas.py to a new file called: instance_.py Example: instance_s3.py
2. Update the class name to match the new instance
3. Update the docstrings
4. Set the `TemplateInstance` in `super(TemplateInstance, self).__init__(instance_id)` to your new class name
5. Program `get_config()` to correctly retrieve relevant settings
6. In the respective cloudprovider, import the new instance and add a statement to create an instance of the class.

New Service Setup

1. Copy service_template.py to a new file called: service_.py Example: service_mysql.py
2. Correctly set the class name to match the new service
3. set `self.name = "Static Name"` to the name of the service (process name?)
4. set `self.config_files = ["/tmp/x"]` to the correct file locations that contain (relevant) configuration settings
5. In the respective IaaS instance classes, import the new service and add a statement to create an instance of the class. *Note: less files / folders = faster execution, be strict and limited.*

New Report

1. To create a report, make use of the 'global' report variable passed down from Main()
2. Set the title of the report with 'report_type' specifying the type of data
3. Set the data itself with `data=` this is any type of `list` or `dict` and is JSONified

```
self.report.create_entry(report_type='vpc', data=serialized_vpc)
```

Naming convention:

- Type is the type of file. I.e. instance, service and name being the type of instance, service, etc.

File Names: `<type>_<name>`

```
File names use the type first to create an easy overview via alphabetical sorting.
Examples: instance_ec2 service_apache.
```

Class Names: Camel Case `<name><type>`

```
Classnames use a camel casing and can always be recognized inside of the code as such.
Examples: `ApacheService`, `EC2Instance`
```

Function Names: Underscore

```
Function names make use of underscores as 'spaces' and a longer naming convention. This improves readability.
Examples: `vpc_serializer`, `get_config`, `test_baselines`
```

Variable Names: Underscore

```
Function names make use of underscores and a longer naming convention. This improves readability.
Examples: `good_message`, `config_file`, `resource_group`
```

EC2

`AmazonEC2RoleforSSM` permission is required on the linked IAM role in order to execute commands on the EC2 instance.

APPENDIX E. COMPETENTIES

A1 Analyseren probleemdomein & opstellen probleemstelling

Bewezen tijdens de Initialisatie en definitie fase door de opdracht verder te definiëren en het probleem vast te stellen.

A2 Informatie vergaren, analyseren, beoordelen & verwerken

Door onderzoek te doen naar penetratie test automatisering en cloud automation is deze competentie uitgevoerd.

C6 Ontwerpen Software

Om de cloud pentests te automatiseren is een tool ontworpen om ervoor te zorgen dat deze goed, structureel opgebouwd is.

D15 Testen

De functionaliteit van de tool is constant getest tijdens de ontwikkelingsfase om te controleren of de code functioneert. Aan het einde van de iteraties zijn de eisen getest tegen de functionaliteit van de code om ervoor te zorgen dat deze voldoet.

D16 Realiseren Software

Het realiseren van software is gebeurd tijdens de realisatie fases van de Agile iteraties. De resultaten van de realisatie is het opgeleverde product.

Gc Kritisch, onderzoekend en methodisch werken

Tijdens de afstudeerstage is er gebruik gemaakt Agile, GOTHIK en projectmatig werken om de fasering en werkmethode van de stage te beheersen. Dit bewijst dat er methodisch te werk gegaan is. Tijdens mijn opdracht zijn er verschillende keuzes naar voren gekomen. In de onderzoeken is hier kritisch gekeken naar de onderzochte mogelijkheden en een keuze gemaakt aan de hand van een vergelijking.

Gf Leren leren: voorbereiden op volgende studiefase en beroep

Door onderzoekend bezig te zijn met nieuwe onderwerpen is nieuwe kennis opgedaan op het gebied van Cloud omgevingen. Aan de hand van feedback en reflecties zijn nieuwe inzichten en ervaringen opgedaan die toegepast kunnen worden op mijn volgende studiefase en beroep.

Appendix F. Suggestion Report
Cloud Pentest Automation

Suggestion Report
Cloud Pentest Automation
Niels Loozekoot (Graduate Intern)
Version 1
2019-05-21

CONFIDENTIAL

Secura B.V.
Vestdijk 59
5611 CA EINDHOVEN
Netherlands

Karspeldreef 8
1101 CJ AMSTERDAM
Netherlands

T +31 (0)40 23 77 990
E sales@secura.com
W secura.com

TABLE OF CONTENTS

SUMMARY

To further the marketing goals of Secura, an initiative was put forth to create a tool that would automate crystal box penetration tests on cloud environments. These are currently 'manually' provided by security specialists and consume a considerable amount of time most of which are done partially (instead of scanning all systems). The goal of this document is to give suggestions based on the tool created by the graduate intern as well as give further suggestions to improve, optimize or add-on to the program.

To start off this document, a small explanation will be given about the developed tool and its abilities. This is followed by a suggestion on how its usability can be simplified and integrated in the hacking workflow of the cloud security specialists. Finally, a few suggestions will be given on developing new functionality.

1. CLOUD PENTEST AUTOMATION – TOOL

The Cloud Pentest Automation tool was developed by Niels Loozekoot in his graduate internship of 2019 between January and Mei. It was built in Python3 with the intent of being as modular as possible allowing Secura to develop new pieces of software that can easily be added and implemented into the tool and increase its usability. In the current stage the tool is able to support EC2 instances and S3 buckets from Amazon AWS. With Amazon AWS it is capable of extracting:

- VPC Data (Virtual Private Cloud)
    - Subnets
    - Instances
    - Security Groups
    - Routing tables
    - ACL's (Access Control Lists)
- S3 settings
    - Default Encryption
- Service settings for:
    - Apache (Apache2, httpd, WAMP)

With Support for:

- Windows
- Ubuntu
- RHEL
- SUSE
- Amazon's custom OS

The tool is capable of executing commands specified in a database and assigning them to a 'setting' name. This setting can then be compared to one or multiple baselines (also specified in a database). The results of which can be used to create a Report.

2. FILLING IN THE BLANKS – SECURITY ASSESSMENT

To expand the coverage of the tool, adding new baselines is one of the first steps that the Security Assesment team can take in furthering the tool. By putting a priority on the 'cloud provider' baselines they can be re-used for all clients, regardless of what systems they have / use. Following this, putting a priority on the most common services / instances will further increase coverage. Once a solid set of baselines exist inside of the tool, an effort can be made to make the baselines compliant with an official set of regulations.

3. WEB PLATFORM – USABILITY

In its current stage the tool contains two types of data in an SQL database:

1. Commands to execute
2. Baselines to compare settings to

This database must currently be set up and maintained by each user individually and for each command and baseline that is manually added with an SQL command, it must be re-distributed to all users of this tool. To combat these issues, I have the following recommendations:

To prevent unsynchronized databases and outdated data, a central database used by everyone in the company will resolve this as everyone modifies and maintains the same set of data. Following this, a web application can be built around the central database to allow for the easy addition or removal of baselines / commands. This allows the security professional to add or remove baselines from the tool whilst they are performing their normal tests which results in a steady increase of coverage.

Optionally, to preserve integrity and quality, a review system can be built in to require one or multiple peer reviews before the baseline or command is fully implemented in the system.

4. NEW FUNCTIONALITY – DEVELOPMENT

There are many different parts of the tool that can be adjusted and tweaked to add new functionality. First, the recommendation is to finish adding to the existing AWS functionality. Implementing new services (such as FTP or SSH), adding support for multiple PaaS instances (such as S3 buckets) and assisting cloud security researchers in adding new baselines to the tool.

Re-implementing the Azure cloud environment, which is one of the larger parts of the structure, is something that could be completed after the AWS functionality. This will require the development of new services as a result which is why these require extra time combined with new credentials and documentation. Based on the Proof of Concept branch / code stored in Gitlab a shortcut can be achieved in obtaining a functioning implementation of Microsoft Azure.

APPENDIX G. AFSTUDEERPLAN

Informatie afstudeerder en gastbedrijf
Afstudeerblok: 2019-1.1 (start uiterlijk 4 februari 2019)
Startdatum uitvoering afstudeeropdracht: 2019-02-04
Inleverdatum afstudeerdossier volgens jaarrooster: 31 mei 2019
Versie: 2.3
Studentnummer: 15081214
Achternaam: Loozekoot
Voorletters: N.J.H.
Roepnaam: Niels
Adres: Holterberg 13
Postcode: 2716 EZ
Woonplaats: Zoetermeer
Telefoonnummer: 0798887265
Mobiel nummer: 0612164894
Mijn mobiele nummer zal naar alle waarschijnlijkheid veranderen in de komende maanden, Ik zal hierover informatie
verstrekken zodra het nieuwe nummer bij mij bekend is.
Privé emailadres: n.loozekoot@hotmail.com
Opleiding: HBO-ICT: NSE
Locatie: Delft
Variant: voltijd
Naam studieloopbaanbegeleider: Hans de Vreught
Naam begeleidend examinator: Pieter Burghouwt
Naam expert examinator: Tony Andrioli
Naam bedrijf: Secura
Afdeling bedrijf: Security
Bezoekadres bedrijf: Karspeldreef 8
Postcode bezoekadres: 1101 CJ
Plaats: Amsterdam
Telefoon, bedrijf: +31 (0)40 23 77 990
Internetsite, bedrijf: https://secura.com
Achternaam opdrachtgever: Mw. Van Kolck
Voorletters opdrachtgever: A.H.
Titulatuur opdrachtgever: MSc
Functie opdrachtgever: Manager Service Line Security Assessments
Doorkiesnummer opdrachtgever: n.v.t.
Email opdrachtgever: antal.vankolck@secura.com
Achternaam, bedrijfsmentor: Dhr. Campers
Voorletters, bedrijfsmentor: P.J.F.P
Titulatuur, bedrijfsmentor: BSc
Functie bedrijfsmentor: Security Specialist
Doorkiesnummer, bedrijfsmentor: n.v.t.
Telefoonnummer bedrijfsmentor: 0031 618 293 126
Email, bedrijfsmentor: pim.campers@secura.comDoorkiesnummer, afstudeerder: n.v.t.
Functie afstudeerder (deeltijd/duaal): Fulltime
Titel afstudeeropdracht:
Geautomatiseerd cloud-security pentesten
Opdrachtomschrijving

1. Bedrijf
Secura is een onafhankelijke, gespecialiseerde adviseur op het gebied van Securityadvies, -testing, -training en -certificering. Het bedrijf heeft ongeveer 70 medewerkers verdeeld over twee locaties (Amsterdam en Eindhoven) en is opgericht in 2000. Secura is in een groeifase en verwacht veel nieuwe medewerkers te krijgen in de komende jaren. (Onafhankelijk, in deze context wilt zeggen dat ze geen specifieke software of certificering promoten bij het geven van bijvoorbeeld adviezen)

2. Probleemstelling
Secura voert regulier pentests uit op verschillende Cloudomgevingen; deze kosten veel tijd voor securityspecialisten om goed uit te voeren. Om tijd te besparen wil Secura deze tests en bijhorende rapporten zo veel mogelijk automatiseren.

3. Doelstelling van de afstudeeropdracht
De opdracht zal een onderzoek zijn naar hoe pentests op cloud omgevingen het best geautomatiseerd kunnen worden en scripts / programma's maken die in staat zijn om deze automatische pentests uit te voeren. Een groot deel van de cloud pentest zal geautomatiseerd worden en rapporten zullen gegenereerd worden aan de hand van de resultaten. Hiermee kunnen de pentesters verder aan de slag om de pentest af te ronden. De geautomatiseerde pentest zal proberen om overeen te komen met de technische eisen van de CSA Cloud Control Matrix (Internationaal framework voor beoordeling van cloud services) om te testen of er een 'baseline' security standaard aanwezig is.

4. Resultaat
Met de resultaten van de opdracht is het bedrijf in staat om sneller en efficiënter cloud pentests uit te voeren. Hierdoor kunnen pentesters hun tijd aan andere onderdelen besteden binnen de cloud pentest met als gevolg dat er een beter resultaat opgeleverd kan worden.

5.Uit te voeren werkzaamheden, inclusief een globale fasering, mijlpalen en bijbehorende activiteiten
Globale fases:

- Oriëntatie op de opdracht – 1.5 weken
- Onderzoek hoe Secura 'cloud investigations' uitvoert – 0.5 weken
- Literatuuronderzoek voor automatisering van pentests en tooling – 1 week
- Proof of concept bouwen voor enkele test – 4 weken
- Prototype van scripts / plugins maken voor testautomatisering – 3.5 weken
- Product productierijp maken – 4.5 weken

Activiteiten:

- Experts raadplegen over automatiseren van tests – 3 Dagen
- Plan van aanpak schrijven – 2 Dagen
- Meelopen met pentester voor een cloud pentest – 2-3 Dagen
- Onderzoeken hoe cloud investigations nu uitgevoerd worden – 3 Dagen
- Literatuuronderzoek – 5 Dagen
- (Optioneel) Volgen van cursussen binnen Secura – (0-5 dagen)
- Proof of concept bouwen + leren omgeving – 10 Dagen
- Demo van Proof of concept – 1 Dag
- Aanvullend onderzoek & ontwerpen prototype – 5 Dagen
- Prototype bouwen – 15 Dagen
- Documentatie voor prototype – 2 Dagen
- Demo Prototype – 1 Dag
- Aanvullen / aanpassen ontwerpen aan de hand van feedback – 5 Dagen
- Procutierijp maken – 15 Dagen
- Documentatie voor Productierijp product – 3 Dagen
- Demo Productierijp product – 1 Dag
- Pentest uitvoeren met pentesters en gebruikmaken van de geschreven scripts – (0-3 dagen)

Tussen deze activiteiten door zal ik actief werken aan mijn stageverslag

6. Op te leveren (tussen)producten

- Plan van aanpak
- Onderzoek naar methodes om geautomatiseerde cloud pentests uit te voeren
- PoC voor geautomatiseerde test
- Prototype voor geautomatiseerd testen
- Geautomatiseerde script(s)
- Script / Tooling documentatie

7. Te demonstreren competenties en wijze waarop

- **A1 Analyseren probleemdomein & opstellen probleemstelling**
  Tijdens de afstudeerstage zal ik het probleemdomein van mijn opdracht verder definiëren en een probleemstelling schrijven.

- **A2 Informatie vergaren, analyseren, beoordelen & verwerken**
  Onderzoek doen naar penetratie testing technieken, test automatisering en de manier hoe deze toegepast kunnen worden op de opdracht.

- **C6 Ontwerpen Software**
  Om de cloud pentests te automatiseren zullen er scripts ontwikkeld moeten worden; het ontwerpen van scripts/software zorgen ervoor dat ze gestructureerd opgebouwd worden.

- **D15 Testen**
  De opdracht zal een aantal verschillende scripts en proofs of concept opleveren. Deze zullen getest worden op o.a. fouten. Ook zullen systemen en netwerken getest worden op security fouten

- **D16 Realiseren Software**
  De scripts die de pentests zullen automatiseren zullen deze competentie verantwoorden.

- **Gc Kritisch, onderzoekend en methodisch werken**
  Tijdens deze opdracht zal ik een onderzoek volgen en een methodische aanpak gebruiken die in het plan van aanpak beschreven zal staan. Tijdens mijn onderzoek zal ik kritische afwegingen maken met de product eisen in oog en zal reflecteren op de gemaakte keuzes aan het einde van de opdracht.

- **Gf Leren leren: voorbereiden op volgende studiefase en beroep**
  Tijdens deze opdracht zal ik veel nieuwe kennis opdoen op het gebied van security en zelfstandig bezig zijn met het leren van nieuwe technieken of software. Ik zal keuzes maken aan de hand van onderzoek en ervaring. Ik zal reflecteren op de keuzes en de manier waarop deze gemaakt zijn om te leren van eventuele fouten of hoe de opdracht effectiever uitgevoerd had kunnen worden.

APPENDIX H. EVALUATIEFORMULIER BEGELEIDER

In te vullen door opdrachtgever c.q. bedrijfsmentor(en)
Student: Niels Loozekoot
Periode: 04 Februari tot en met 31 Mei
Bedrijf c.q. instelling: Secura
Bedrijfsmentor: Pim Campers
Plaats: Eindhoven
Datum: 20-05-2019

1. Heeft de student zich zelf snel en goed ingewerkt in het bedrijf en de uit te voeren afstudeeropdracht?
Vanwege de complexiteit van de opdracht is er in het begin van de opdracht nauwe samenwerking geweest tussen Niels, de CTO (Ralph Moonen) en mijzelf. Hierna is door Niels zelfstandig gewerkt aan de realisatie van doelen en producten. Tussentijds zijn er een aantal momenten afgesproken waarin de verschillende producten gevalueerd zijn geworden en waarin mogelijk wijzigings of verbeter voorstellen zijn gedaan. Deze zijn daarna door Niels op een correcte manier opgepakt en verwerkt in de uiteindelijk opgeleverde producten.

2. Hoe beoordeelt u de communicatieve vaardigheden van de student (in de samenwerking met collega's, in contacten met de opdrachtgever, bij mondelinge presentaties, schriftelijke rapportages)?
Als ik kijk naar de overleggen die gevoerd zijn door Niels met de organisatie zijn deze meer als voldoende al is op te merken dat het in het begin wat onwenning was voor Niels maar gedurende het project was te merken dat hij zich steeds comfortabeler begon te voelen met het voeren van overleggen. Kijkende naar zijn preseteren kan ik alleen maar lovend zijn ondanks dat hijzelf aangeeft presenteren vervelend te vinden was de presentatie die hij intern heeft gehouden helder en verduidelijkend over de afstudeer opdracht en het uiteindelijke product dat Niels heeft opgeleverd. Samenvattend beoordeel ik zijn communicatie vaardigheden dan ook als goed.

3. Hoe heeft de student tijdens het uitvoeren van de opdracht gefunctioneerd?

Qua verantwoordelijkheid	~~goed~~ / **voldoende** / ~~matig~~ / ~~onvoldoende~~
Qua zelfstandigheid	**goed** / ~~voldoende~~ / ~~matig~~ / ~~onvoldoende~~
Qua planmatig werken	~~goed~~ / **voldoende** / ~~matig~~ / ~~onvoldoende~~
Qua creativiteit	**goed** / ~~voldoende~~ / ~~matig~~ / ~~onvoldoende~~
Qua productiviteit	**goed** / ~~voldoende~~ / ~~matig~~ / ~~onvoldoende~~
Qua samenwerken met collega's	~~goed~~ / **voldoende** / ~~matig~~ / ~~onvoldoende~~
Qua draagvlakontwikkeling	~~goed~~ / **voldoende** / ~~matig~~ / ~~onvoldoende~~
Qua inspelen op bedrijfscultuur	**goed** / ~~voldoende~~ / ~~matig~~ / ~~onvoldoende~~
Qua rekening houden met de specifieke context van het bedrijf	**goed** / ~~voldoende~~ / ~~matig~~ / ~~onvoldoende~~
Qua het op gang brengen van de nodige veranderingen	~~goed~~ / **voldoende** / ~~matig~~ / ~~onvoldoende~~

4. Hoe beoordeelt u de kennis en kunde van de student in verhouding tot wat u verwacht van een bijna afgestudeerde?
Kijkende naar de complexiteit van de opdracht en de verschillende aspecten die hierbij komen kijken zoals bedrijfsbelangen, technische mogelijkheden van de cloud-diensten en de producteisen. Denk ik dat de kennis van Niels volledig na verwachting is van een bijna afgestudeerde. Hierbij is zijn programmeer kennis bovenverwachting en heeft hij zich verder snel wegwijs gemaakt in de API's en de werking van de verschillende cloud providers.

5. Hoe beoordeelt u de kwaliteit van de opgeleverde (tussen)producten?
Zoals eerder ook aangegeven zijn er verschillende tussenproducten geweest voor het uiteindelijk product. Hierin is per product een steigende lijn te zien qua kwaliteit van het product. Dit is voornamelijk terug te herleiden naar het duidelijk krijgen van de mogelijkheden en beperkingen van de verschillende clouddiensten. Het uiteindelijke product beoordeel ik dan ook met een voldoende tot goed.

6. Bent u tevreden over het opgeleverde (eind)product?

- **In hoeverre heeft u gekregen wat is afgesproken?**
  Tijdens het project zijn eisen opgesteld en in de tussentijdse evaluatie van de verschillende producten zijn sommige eisen bijgesteld voor de kwaliteit en haalbaarheid van het product. Dit is door Niels op een juiste manier opgepakt en aangekaart geworden. De conclusie is dan ook dat het uiteindelijke product aan de eisen voldoet.

- **In hoeverre voldoet het (eind)product aan uw verwachtingen?**
  Kijkende naar de opgestelde eisen van het product is aan de eisen van het product voldaan. Het product dient als framework waar op basis van verder functionaliteit en baselines aan toegevoegd zullen worden zodat het een volwaardig product is. We concluderen dan ook dat het product aan de verwachtingen voldoet.

- **Wat is de bruikbaarheid en onderhoudbaarheid hiervan?**
  Zoals al aangegeven is het eindproduct onderdeel van een veel groter product dat verder ontwikkeld zal worden op basis van het huidige eindproduct van Niels. Hij heeft in deze opdracht namelijk een framework gebouwd waarin de verschillende clouddiensten op basis van alleen een API sleutel bevraagd kunnen worden op verschillende security eisen. Verder is het product volledig modulair opgebouwd (dit was een van de product eisen) dit zorgt ervoor dat alle onderdelen los van elkaar te onderhouden zijn en dat het toevoegen en wijzigen van API requests op een eenvoudige manier is te realiseren.

- **Wat gebeurt er met het opgeleverde (eind)product?**
  Het framework zal worden uitgebreid met de verschillende baselines voor het bevragen van de clouddiensten op security richtlijnen (Zoals de CIS baselines). Omzodoende op een geautomatiseerde en een duidige manier omgevingen van klanten te kunnen onderzoeken.

- **Kunt u direct met het opgeleverde product aan de slag?**
  Het betreft een framework daarom dient er op dit moment nog de baselines aan toegevoegd te worden om inzetbaar te zijn. Daarbij dient er nog iets bedacht te worden om van de resultaten van het framework naar een rapport te gaan. Het framework is daarom voor de organisatie direct te gebruiken voor verdere ontwikkeling van het geautomatiseerde product binnen de organisatie.

7. Zijn er nog aspecten voor u van belang die nog niet aan de orde zijn geweest?
Als bedrijfsmentor heb ik Niels zien groeien binnen de organisatie waar in het begin te merken was dat het allemaal wat onwennig was. Merk ik nu aan het einde dat Niels zich op een goede en juiste manier binnen de organisatie weet te bewegen. Hierbij zie je de ontwikkeling op zelfstandigheid en initiatief goed terug.