BUG DEFENDER

Group: Cloud Based Antivirus

Authors: Gayatri Bias , Manasa Kulkarni, Manju Yadav, Smit Topiwala, Vinaya Chaparala

Email addresses: [gchandel@uab.edu](mailto:gchandel@uab.edu) ; [manasak5@uab.edu](mailto:manasak5@uab.edu) ; [manju@uab.edu](mailto:manju@uab.edu) ;

[smit@uab.edu](mailto:smit@uab.edu) ; [cvinaya5@uab.edu](mailto:cvinaya5@uab.edu)

# Abstract

Keywords: Anti-Virus, N-version software,

Cloud Security, Malware detection

Abstract - Antivirus software is a commonly used tool for detecting and stopping malicious and unwanted files. But, the long-term effect of traditional host-based antivirus is questionable. Antivirus software fails to detect many new threats, and its growing complexity has resulted in vulnerabilities that are being exploited by malware. In this project, we worked on a malware detection model on end hosts as an in-cloud service. This model enables the identification of malicious and unwanted software by multiple detection engines running in parallel. The N-versioning approach gives several benefits, including better detection of malicious software, enhanced forensics capabilities. In cloud computing, malware detection includes a lightweight, cross-storage host agent and a network service. This project combines detection techniques, static signatures analysis, and dynamic analysis detection [1]. We found that cloud- malware detection provides better detection coverage against recent threats than a single antivirus engine. In this project, we have also reviewed some of the works based on the N-version software approach. So, this project report provides summarized information about the basic requirements for implementing N-version software to increase detection accuracy and reduce the load on the client computers. The potential benefits of the N-version approach are also discussed.

# Introduction

Computer viruses are software programs that are deliberately designed to interfere with computer operations: record, corrupt or delete data, spread themselves to other computers and throughout the internet [2]. Computer viruses can be detected and removed by an antivirus software. Antivirus (AV) software is a computer program that detects, prevents and takes actions to remove malicious software programs, such as computer viruses and worms [2]. AV software is a predominant way to keep computers safe from malware. The first ever virus recorded in history to infect a computer was called the Creeper virus dated back in 1971 [3]. A program called ‘The Reaper’ was created to delete the virus from the computer [3]. It was considered the first ever antivirus, but in fact ‘The Reaper was just another virus to remove the Creeper virus. Since then, several other viruses have followed the Creeper virus. There was several other software as well to detect these viruses.

Over the years, the malware has evolved so much that some of them are not detectable by the AV software. Frederick B. Cohen, best known as the inventor of computer virus defense techniques, demonstrated in one of works that there is no algorithm that can perfectly detect all possible viruses [4]. Traditional AV software extracts the signature of all the detected malware and adds them to AV database so the software can easily detect when it comes across a same one again. But the viruses which can be categorized as polymorphic, metamorphic or oligomorphic are updating themselves at a deeper level that they encrypt a part of the code so that the extracted signatures do not match the signatures in the database of the antivirus.

In this project, we create a cloud-based antivirus, Bug Defender, which scans files for malware. This AV software uses 3 antivirus engines in parallel to detect the viruses. As this is based on N-Version programming technique, the chances of detecting a virus is a lot higher than that of a single AV software. The Bug defender user interface is easy to use. The user can first register in the application and can login after registration. The user can upload as many files at the same time and the files will be scanned by 3 different antivirus engines and provides results if the files are safe to use or if they are infected.

# Background

To implement BugDefender, we need to have an AWS account. We need to have some knowledge on python for working on AWS Lambda function and on Boto3. We also must have some knowledge of Linux commands to install the antivirus engines on the instances. We have discussed more clearly about the technical approach in section 4.

# Technical Approach

For detecting and debugging purpose we have used three different antiviruses running on three different EC2 instances, we have used Amazon S3 to store uploaded files and DynamoDB for storing the scanned results. The brief details about the Amazon services that we have used for our application development are given below:

* Amazon EC2:

Amazon EC2(Amazon Elastic Compute Cloud) is a web service that provides safe computing capacity in the cloud which is reliable and scalable. Amazon EC2 makes computing easier for developers.

We have used three EC2 instances with three different antivirus engines running on the virtual machines.

* Amazon S3:

Amazon S3(Amazon Simple Storage Service) provides service for storing the objects that offers scalability, security, performance and data availability. This is being used by almost all industries who wants to store and secure large amount of data like mobile applications, websites, IoT devices, archive and big data analytics. Amazon S3 also provides features which are easy-to-use by user. User can provide access control to S3 bucket as per their requirements.

Amazon S3 is designed for a maximum durability and can store data for millions of applications for organizations all around the world. In this application, we have used S3 for storing the uploaded files.

* Boto3:

Boto3 is the Amazon Web Services (AWS) Software Development Kit (SDK) for Python, which allows Python developers to write software that makes use of services like Amazon S3 and Amazon EC2.

* Lambda:

Lambda is an Amazon service which provides an interface for coding for an application. We have used python language for configuring EC2 instances, upload files to S3. Files that need to be scanned are sent to EC2 instances through SSM.

* SQS:

SQS is an Amazon queuing service. There are two types of queues, Standard and FIFO. SQS allows to decouple and scale serverless applications and distributed system etc. When a file is uploaded into S3 in our application, using event notification property, it triggers to SQS.

* SSM:

SSM is an Amazon Systems Manager. This service enables user to view and have control on his AWS infrastructure. It even allows to perform task operations automatically. It also helps in updating, maintaining, and securing EC2 instances. The output of the scan results are sent back to Lambda through SSM.

* Cognito:

Amazon Cognito provides authentication and authorization for web based and mobile based applications. It also has ability to provide complete user identity management system for application.

* DynamoDB:

Amazon DynamoDb is a NoSQL database service that is provided by Amazon Web Services. It is fully managed, fast, and highly scalable. The results from the instances are updated into database. We have used it for storing scan results.

The front end of this application is developed using Qt Designer. The brief details on the Qt Designer is given below:

* Qt Designer:

It is an open source Qt tool which is used to develop graphical user interfaces.

# Design and Implementation

The design approach to solving the detection and debugging of viruses/malware was based on the following key ideas,

* Completely define the problem.
* Decide the fastest and most cost-effective way to solve the problem.
* Develop with trouble shooting tools or methods that will aid in trouble shooting problems.

The overall architecture of our application is given in Fig. 1. and explained briefly in section 5.1.

The user logins into the application. Amazon Cognito will provide user authentication and access control to the application. Once the user is logged in, he/she can upload the files that need to be scanned.

## Architecture:

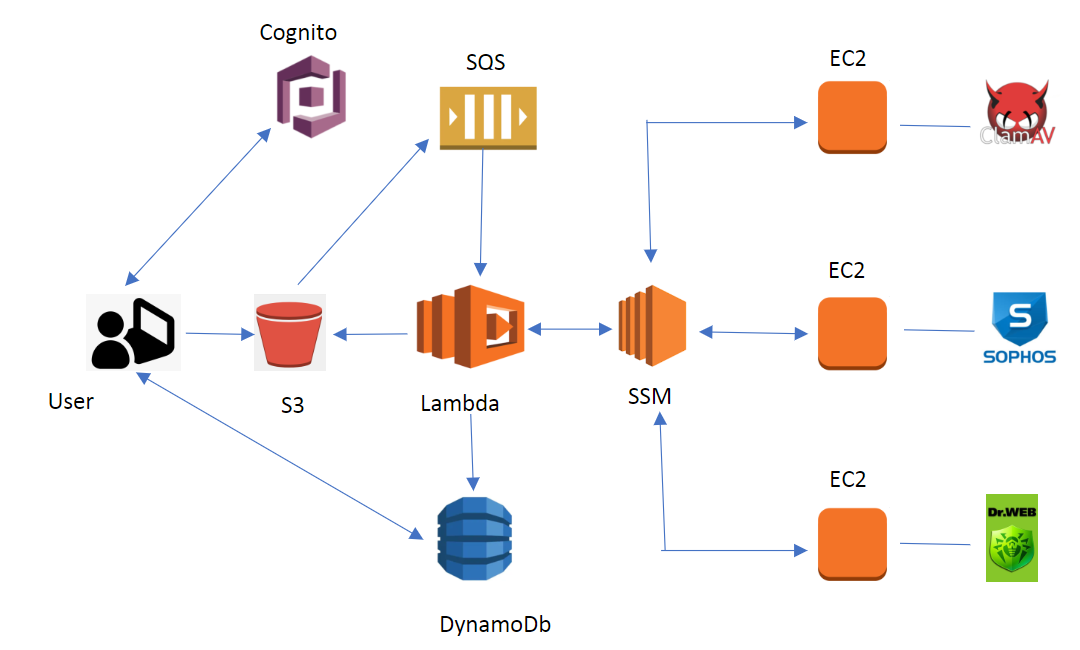


Fig. 1. Architecture

These files are uploaded into Amazon S3 and are sent to Amazon SQS using S3 event notification property. Then SQS notifies the lambda function using lambda trigger. In Lambda function, we have written a logic that will handle multiple EC2 instances with different antivirus engines running in parallel with the help of Amazon Systems Manager. SSM makes possible to manage and secure EC2 instances.

We have created three EC2 instances. One instance is installed with ClamAv antivirus, the other with Sophos and the third instance with Dr.Web. The output of the scanned result from all instances are sent to Lambda. The results of the scan are stored in database. User can also view the results in user interface. The scan status of the file for each antivirus engine will be shown to the User in parallel.

We have provided some screenshots of our application process for a more visual understanding in section 5.2.

## Screenshots:

We have provided some screenshots of our desktop application. Fig.2 is the login page of our application. Fig.3 is the home page of our application, the screenshot shows that the files are getting scanned by different Antivirus engines. Fig.4 is the screenshot of the table in DynamoDb where scan results are stored.

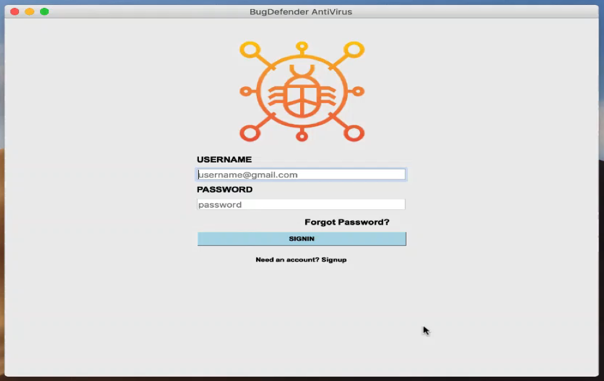


Fig. 2. Login Page

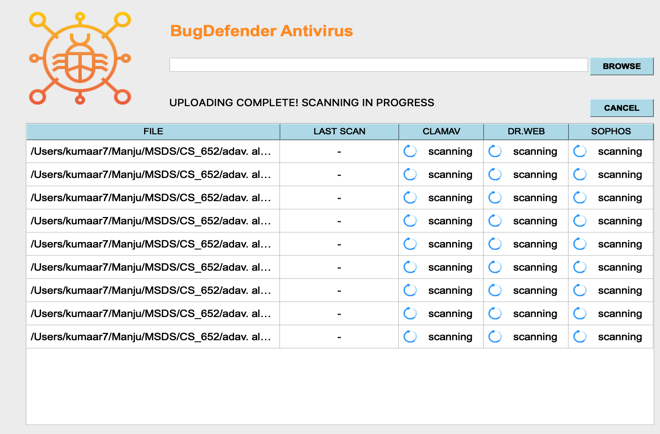


Fig. 3. Home Page

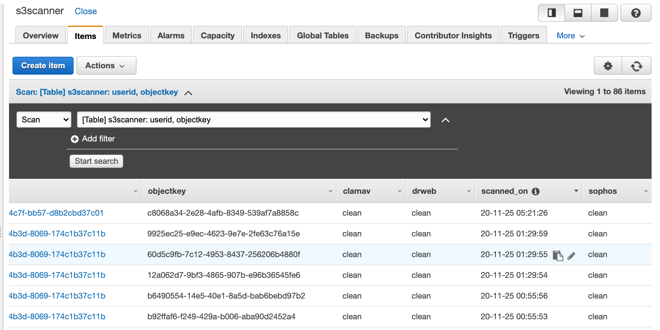


Fig. 4. Scanned result stored in database

## Implementation:

Steps to create BugDefender BackEnd:

* First, we need to create a DynamoDB table to save scan data; by providing the primary key and sort key.
* We also need to create an SQS queue to S3 to upload event notifications. While creating the SQS (Simple Queue Service), choose STANDARD QUEUE. Enter the Queue name. In the configuration section, set visibility timeout to 3 Minutes. In Access policy, add the below policy document mentioned in Fig (5)

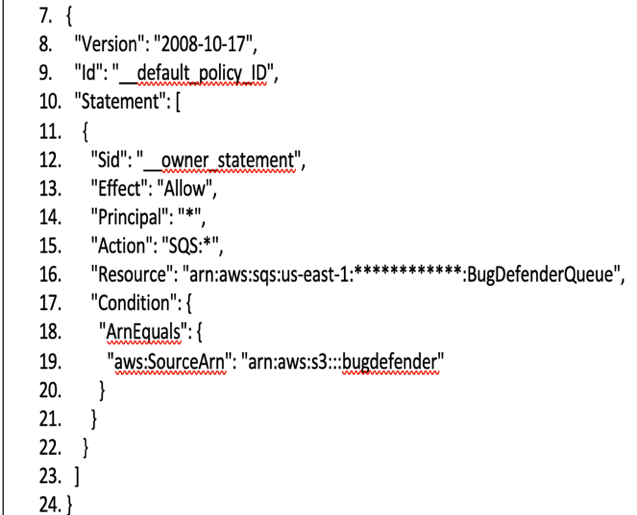


Fig. 5. Access Policy

* Once we have created the SQS queue, we need to create S3 Bucket, and we have to set up the event notification for SQS by selecting SQS Queue as destination in the Destination card. We also need to create a role for lambda to have the permissions for all the required services. This role includes the following permissions:
* AmazonS3FullAcess
* AmazonEC2ReadOnlyAccess
* AmazonDynamoDBFullAccess
* AmazonSSMFullAccess
* AWSLambdaSQSQueueExecutionRole
* AWSLambdaExecute
* Now using the above-created role, we need to create SQS triggered Lambda function.
* To set the Lambda function to SQS Lambda triggers, we need to select the already created SQS Queue, and we need to add a trigger for the above-created lambda.
* We need to set up Cognito for user authentication and IAM Policies to upload to s3 and read from DynamoDB. For the policy document, we have updated the template as given below.



Fig.6. Authenticated Cognito user Policy

In a similar manner we have created the policy for unauthenticated Cognito user.

* Now create the user pool so that users can sign in to app through Amazon Cognito. Make note of the Pool Id. Create an application client for signing up and signing in the users. Make note of the App client id. To create unique identities for your users and federate them with identity providers we need to create federated identities pool. With an identity pool, you can obtain temporary, limited-privilege AWS credentials to access other AWS services.
* While creating federated identities pool Cognito will inform you that Cognito identities need access to your resources via IAM roles. So, we need to create new roles for both authenticated and unauthenticated identities. Enter names for each one, authenticated" and unauthenticated" (At this point we left the policies as Cognito defaults but replaced those policies with the ones we made above.) Make note of the Federated Identities pool id.
* Now we need to update Cognito roles (Authenticated and unauthenticated) with previously created authenticated and unauthenticated policies. Also Create Role and policies for EC2 instances.
* Create IAM role for lambda with AmazonS3FullAcess and Similarly, we have created a policy for unauthenticated Cognito users. Now create the user pool so that users can sign in to the App through Amazon Cognito. Make a note of the Pool Id. Create an application client for signing up and signing in the users. Make a note of the App client id.
* To create unique identities for your users and federate them with identity providers, we need to create a federated identities pool. With an identity pool, we can obtain temporary, limited-privilege AWS credentials to access other AWS services.

While creating a federated identity pool Cognito will inform us that Cognito identities need access to our resources via IAM roles. So, we need to create new roles for both authenticated and unauthenticated identities. Enter names for each one, authenticated" and unauthenticated." (At this point, we left the policies as Cognito defaults but replaced those policies with the ones we made above.) Make a note of the Federated Identities pool id. We need to update Cognito roles (Authenticated and unauthenticated) with previously created authenticated and unauthenticated policies.

* Create Roles and policies for EC2 instances. Create IAM role for lambda with AmazonS3FullAcess and AmazonSSMManagedInstanceCore Policies. Choose lambda from common use case and create role.
* The next step is to launch EC2 instances by selecting Ubuntu Server 18.04 LTS (HVM), SSD Volume Type 64-bit(x86). Choose IAM role created from the dropdown "Antivirus\_Worker\_Role" and launch instance.
* After launching the instances, we have to update the Linux instances and swap each instance's memory to provide some extra memory. Next, we need to install pip3 and boto3 on the instances. After that, we mount different antivirus scanner scripts directly to instances. Once this is done, we Installed three different Antivirus on three running instances with installation instructions [5], [6], [7].

# Discussion:

The main aim to build this application is to check files in the user’s system are safe from malware or not. We have used N-version programming approach for more accurate results. In this process, we have faced few challenges which are discussed in section 6.1. We have few limitations too in our application discussed in section 6.2 which can be overcome in future work.

## Challenges:

* It was initially hard for us to understand the overall architecture and to implement it.
* Finding open-source antivirus engines were also challenging and their installation on instances.
* To combine results of all instances and generate output was also tough part to figure it out.
* As we were facing memory issues in our instances, we had to add swap memories to each instance to avoid the issue.
* Simplifying user interface.

## Limitations:

* We have used only 3 different antivirus engines. More antivirus engines can be added for more accurate results.
* We have restricted file size in our application. File size greater than 100 mb cannot be uploaded.
* Sophos antivirus is old and slow. Also does not provide accurate results at times.
* Lack of Detection Engine in our application. It means that infected files cannot be deleted automatically by our Bug Defender.

## Lessons Learned

* In this process of application development, we have learned about how Qt designer works and how-to create desktop application using python code.
* There was a lot to learn about S3 bucket, how to make it public and about the access control, different properties associated to S3 bucket like event notification etc. So, we played around with S3 buckets and gained more knowledge on its working.

Our primary lesson learned was to dig deeper into the actual use of the final product.

# Task Distribution:

All members of the team have contributed and interacted well with each other throughout the project. Everyone was prompt in completing their assigned task and submitting it back to the group for review. Submitting tasks back to the group gave teammates an opportunity reviews the work and provide appropriate feedback.

* Gayatri Bais : Application Design - Frontend Development, Research, Report, Testing.
* Manasa Kulkarni : AWS S3, Lambda, PPT and Report.
* Manju Yadav : Backend Design, Frontend & Backend integration, Demo Recording, Report.
* Smit Topiwala : Frontend Design, AWS S3, PPT, Report, Research on different Antiviruses and Implementation.
* Vinaya Chaparala : Backend Design, Research and Implementation, Report, Testing.

# Conclusion:

To conclude, with the completion of Bug Defender application, we have successfully implemented and demonstrated the project solution that allows users to upload data files in the cloud for detecting and debugging if there is any virus or malware in the file and keep the computer system free from harmful viruses and malwares in order to protect the computer by simply using a desktop application.

## Future Enhancements:

Bug Defender application can be enhanced in the following areas,

* Better UI: The user interface can be modified to make the user experience smoother on the application and make the features more accessible.
* More functionalities (Folder upload, drag drop feature.): The desktop application can be modified to add Drag and Drop feature where users can simply drag their files and drop them for uploading

# References:

1. Oberheide, Jon & Cooke, Evan & Jahanian, Farnam. (2008). CloudAV: N-Version Antivirus in the Network Cloud. Proceedings of the 17th USENIX Security Symposium. 91-106.
2. [https://web.archive.org/web/20110411203211/http://www.microsoft.com/security/resources/antivirus-whatis.aspx](https://web.archive.org/web/20110411203211/http:/www.microsoft.com/security/resources/antivirus-whatis.aspx)
3. [From the first email to the first YouTube video: a definitive internet history | Internet | The Guardian](https://www.theguardian.com/technology/2009/oct/23/internet-history)
4. [An Undetectable Computer Virus (archive.org)](https://web.archive.org/web/20110604155118/http:/www.research.ibm.com/antivirus/SciPapers/VB2000DC.htm)
5. <https://linuxize.com/post/how-to-add-swap-space-on-ubuntu-18-04/>
6. <https://download.drweb.com/linux/?lng=en>
7. <https://github.com/maliceio/malice-av/raw/master/sophos/sav-linux-free-9.tgz>
8. <https://aaronbrighton.medium.com/installation-configuration-of-clamav-antivirus-on-ubuntu-18-04-a6416bab3b41>