Cyber ChatBot
ChatBots in cyber world

ABHIJEET SRIVASTAVA, University of Delaware
ANUPAM BASU, University of Delaware

In order to create a disruption in how cyber security experts do their work we have investigated how a Cyber ChatBot can be created which would be helpful to cyber analysts. It is estimated that there will soon be a shortfall of a million cyber security analysts needed to protect computer systems worldwide. There are many chat bots in the market which allow continuous dialogues with them. We have created a cyber chatbot that has natural language processing (NLP) capabilities and can answer questions about a file that is uploaded. The idea behind a powerful cyber chatbot is that it should be able to perform menial or repetitive tasks. It should also be able to work along side human analysts significantly increase the effectiveness of security analysts.

CCS Concepts: •**Computer systems organization** → **Embedded systems;** *Redundancy;* Robotics; •**Networks** → Network reliability;

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Wireless sensor networks, media access control, multi-channel, radio interference, time synchronization

ACM Reference format:
Abhijeet Srivastava and Anupam Basu. 2017. Cyber ChatBot. 1, 1, Article 1 (April 2017), 8 pages.
DOI: 0000001.0000001

1 INTRODUCTION

The phase 1 of the project was to build a cyber chatbot that would answer virus related questions about a file. We are using Amazon Web Services(AWS)[5] to deploy ec2-user linux machine. We made a bot using claudia[8] framework which allows us to focus on the functionality where as it focuses on workflow. Claudia.js framework is very powerful framework which deploys AWS lambda functions for each bot. Virus total and flask API is used to ask questions about the file. Each API is answering different types of questions for a particular file.

We have created slack slash commands[17] and have integrated them with slack application that we created. There is a slack bot that we created which is listening to any file upload, upon any file upload it uploads the file to AWS S3 bucket. When answering questions we pick the file from S3 and calculate its hash on the go. The user also has option of giving a hash and ask question about that hash and in that case we do not look at the hash of the particular file.

Figure 1 shows a basic outline of our project. Slack is our front end for the cyber chatbot. S3 bucket in AWS is where we save our uploaded files. We have hosted our server in the ec2 machine which connects with virustotal or flask API to get the answers related to virus.

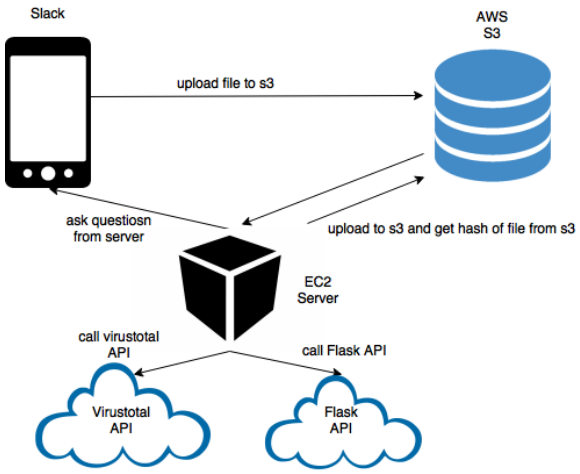

Fig. 1. Our front end is a slack application which interacts with the server hosted on ec2 machine. The files are uploaded in s3 by ec2 server. Once server receives a request from flask it sends and receives response from Virustotal or Flask API depending on the question

The whole project has been uploaded in GitHub [10]. We have put the sample input and output in data folder of the github link. The results obtained in the phase 1 of this project is quite interesting. We have implemented natural language processing which allows users to ask the same question in more than one ways. Also a user can keep on having conversations about one file like how a chatbot would actually do.

2 FRAMEWORK: CLAUDIA.JS

Claudia[8] makes it easy to deploy Node.js projects to AWS Lambda and API Gateway. It automates all the error-prone deployment and configuration tasks, and sets everything up the way JavaScript developers expect out of the box.

This means that you can get started with Lambda microservices easily, and focus on solving important business problems instead of dealing with AWS deployment workflows.

Advantages of using Claudia are as follows-

(1) Deploy and update using a single command- We can use the command **claudia create** for the first time and **claudia update** if we have already once deployed the bot.
(2) Just use standard npm packages, no need to learn Swagger - We have used npm packages like crypto for getting the hash of a file, natural for natural language processing.
(3) Skip all the boilerplate and just focus on your work - Claudia takes care of the deployment of the code as AWS lambda. We can focus on our solution.
(4) Manage multiple versions easily - Claudia update command allows us to easily update a bot.
(5) Get started in minutes, with a very low learning curve - Claudia setup is staight forword and can be configured with some background knowledge about IAM.

There are various tutorials(https://claudiajs.com/tutorials/index.html) about claudia and an active community to help others facing problem in claudia(https://gitter.im/claudiajs/claudia).

3 IAM ACCESS

Identity and Access Management (IAM) in AWS[5] is a web service which enables us to grant permissions to the users we create to access AWS's other resources. This web service is critical in maintaining security of the AWS account, and creating users and groups with custom capabilities and permissions. By managing permissions, one can control which actions the created user is allowed to take. In order to follow best practices, instead of using only the root user in AWS, we have decided to create users in IAM, with select permissions to have access to services like Lambda and Simple storage Service (S3), which is essential for the chatbot. The IAM policy on Lambda allows us to connect and create our slash command files (Lambda functions utilizing the Access key and secret access key which we will generate) from EC2 through ClaudiaJS.

The IAM policy on S3 gives permission for the user to access S3 bucket and store files when the user shares the file with the chatbot. It allows slash commands program to retrieve those files.

4 VIRUS TOTAL

VirusTotal is a subsidiary of Google. It is a free online service which analyzes files, URLs and domains and enables the identification of viruses, worms, trojans and other kinds of malicious content detected by anti virus engines and website scanners. At the same time, it can also be used as a means to detect false positives, i.e. innocuous resources detected as malicious by one or more scanners.

We are using public API of virus total which is free. The public API is limited to 4 requests per minute and takes more hours to answer about a file which is uploaded using public key access.

4.1 Virustotal public API

VirusTotal's Public API[18] allows upload and scan of files, submission and scanning URLs, accessing finished scan reports and making automatic comments on URLs and samples without the need of using the HTML website interface. In other words, it allows to build simple scripts to access the information generated by VirusTotal.

The chosen format for the API is HTTP POST requests with JSON object responses and it is limited to at most 4 requests of any nature in any given 1 minute time frame.The public API is a free service, and must not be used commercially as per their license.

4.1.1 Response basics. The API response format is a JSON object containing at least the following two properties:

(1) response code: if the item searched is not present in VirusTotal's dataset this result will be 0. If the requested item is still queued for analysis it will be -2. If the item was indeed present and it could be retrieved it will be 1.
(2) verbose message: provides verbose information regarding the response code property.

4.1.2 Sending and scanning files. The VirusTotal API allows you to send files. In order to send a file you must perform an HTTP POST request to the following URL:
https:www.virustotal.comvtapiv2filescan

The response json can be read and parsed to get the desired output. There are different types of questions that are being answered using the response from virus total. Virus total API also has the ability to answer questions apart from file. It can answer about URL, domains etc.

Npm package node-virustotal is a very efficient package available. This API provides factory methods which make connection objects, which act as job queues. There are 5 kinds of connections: with the public API, public API with honeypot permissions, private API, email API, Intel API.

4.2 Questions answered by VirusTotal

The following questions can be answered by Virustotal-

(1) is this a malware ?
(2) what does macafee say about it?
(3) how many antivirus say it is a virus?
(4) what more can you tell about this file?

We have implemented NLP so that users can ask the same question in more than one way.

5 FLASK

Flask is the other API which is being utilized in the chatbot. It is a part of University of Delaware, John Cavazos' Cyber 20/20 Analytics Service. Flask is a component in FCAS backend of that project, which we communicate via http post.

The analysis, file, latest, submission are the categories of API which you can search for in flask, with the input being file hash containing sha256 hash format of the file. We are using the analysis category by using the URL with the API key provided by the lab for this project cisc850. Using this API helps determine whether our file can be found in their malware database. This is the URL which is used:

http://fcas-test-0.us-east-1.elasticbeanstalk.com/cisc850/search/analysis

5.0.1 JSON Request. The JSON format necessary for posting to Flask API must contain the following:

(1) **filehash** must contain file in the hash format of SHA256.
(2) **select** must contain a list of fields like malware,hash ,family ,date, version, malware_model. The Response JSON will be structured based on this object.
(3) **search_type** must contain LIKE

5.0.2 JSON Response. The JSON response from Flask is a result object array which contains the list of malware matching with the hash sent in the json http request. If the sha256 of the file being searched for was not present in Cyber 20/20 dataset, the result object will be empty. The values of the malware with matching hash depends on the fields given in the select object in the JSON request.

Using the response from Flask, the following questions can be answered by Flask-

(1) Is this a virus?
(2) what is the category of this malware?

6 NATURAL LANGUAGE PROCESSING

Natural language processing (NLP) is a field of computer science, artificial intelligence, and computational linguistics concerned with the interactions between computers and human (natural) languages and, in particular, concerned with programming computers to fruitfully process large natural language corpora.

We have used **natural** npm package to implement NLP. Natural is a general natural language facility for nodejs. Natural allows tokenizing, stemming, classification, phonetics, tf-idf, WordNet and string similarity. We are using tokenization to find all the tokens in a string. Once we have all the tokens we are doing a spell check for incorrect words. By correcting the incorrect words we can easily get the names of correct antivirus.

In NLP module we have created a function of our own that based on various words in a sentences figures out the type of question asked by the user. For example if mcaffee is present in the sentence the type of question is "is this a virus according to mcaffee". If there are words that could be in more than one type of question we have implemented an algorithm similar to k-means clustering where we try to figure out the proximity of a question to various types of questions. The type of question to which it is most similar is the one we answer.

Using regular expression we have also figured out the hash in a question. Any word that is a alphanumeric string (along with other constraints) is considered as hash in a string and then we do not query a question for the uploaded file but answer according to the hash provided in the question string.

Apart from natural npm package there are others- speakeasy and stanfordnlp which are either too simple(as in speakeasy) for our usage or have too many functions(as in stanfordnlp) which we do not need.

7 FILE HANDLING

A cyber chatbot should have the capability of scanning a file, and returning an appropriate response when asked about it by the user. The following sections will elaborate on how our chatbot retrieves and stores the file shared by the user. It also elaborates on what happens to the file when we ask questions about it.

7.1 Real Time Messaging

The **Real Time Messaging** (RTM) is a websocket API in Slack[17]. This API enables us to retrieve events in real time from Slack, and make custom responses or workflow for those events. This capability can be extended to almost all types of events from user messaging to files uploaded by the user. We are using this feature to capture files uploaded by the user in the channel of the chatbot. This is done by starting our program which acts like a server listening to this particular event. The file on being shared is uploaded and saved in the following module S3.

7.2 S3 Bucket

AWS's Simple storage service (S3)[5] is a web service which is used for storing and retrieving data. It is usually used in cloud. It is highly scalable and inexpensive form of data storage. S3 service stores its data in units of storage containers called bucket. Our file upload module accesses the last file from the channel shared between the chatbot and the user, and uploads it to our S3 bucket as chatbot. We later retrieve chatbot file, when the user uses slash commands to ask questions about the file.

7.3 File to Hash

A hash[11] or a digest in the cryptographic sense is an alpha-numeric string of fixed length processed and returned when a **Cryptographic Hash function** (a type of hash function) takes in an input of any format (e.g. file or string). There are numerous types of cryptographic hash functions which are utilized in today's world like SHA-1, SHA-256, MD5, etc.

In order to make flask or virus total scan a file and return whether the file is a virus, we are packaging the file in the form of SHA-256 (Secure Hash Algorithm 256) hash result which is a fixed 256-bit length. We retrieve the file from the bucket where it was uploaded by the file upload function.

We are utilizing a node package called **Crypto** which takes filestream as input and returns SHA-256 of the filestream. It internally uses the cryptographic hash function to calculate the result. This result in turn is sent back to its respective caller function- flask slash command or virustotal slash command. The further process happens in respective modules.

8 SLACK INTEGRATION

Slack[17], the popular acronym of "Searchable Log of All Conversation and Knowledge" [7] is a cloud-based messaging platform which has been popular form of team collaboration. This platform is an ideal team based platform which we have chosen to build our cyber chatbot. Due to its popularity as a work place platform, security teams working in an organization are more likely to use it to communicate with each other.

8.1 Slack Slash Commands

Slack Slash Commands are commands on the Slack messenger which differ from normal messages sent by the user, since they behave and respond differently. There are some built-in slash commands which can be immediately accessibly to the user. But Slack has given the ability to developers to create their own slash commands, which can be customized to unique responses depending upon the type of request from the user. Everything written after a slash command is sent as content to that specific API linked to that command. This API in turn formats and returns an answer based on the content. We are currently using custom slash commands with its unique workflow to ask questions to either flask or virus total.

8.2 Slack App

In slack, creating an Application initializes the process of building a bot with custom functions. Slack Commands can be augmented onto application, to further increase its capabilities with its workflows. It also allows the creating of bot users, which we have utilized in listening to user actions like file upload.

8.3 Bot User

Bot users are users which can be created to have names, profile, bios just like a normal human user in slack. Bots have the capability of listening to almost all the activity happening in slack. Hence we are utilizing custom bot user in order to intercept file exchange events between itself and the user. This has been achieved by **RTM** API of slack. This way the user can perform slash commands to do a follow up questions on the file which was uploaded.

9 RELATED WORK

A central theme in a chatbot is that, it covers an aspect in Natural Language Processing called Dialogue system. Two agents can have an ongoing conversation or dialogue. In the case of a chatbot, one of the participants is human while the other is a Computer system, which should have the capability of Natural Language understanding and Natural Language Generation.

Efforts have been made in order to make chatbots which can tackle conversations with a user based on context implied by the user. A model of social chatbot [9] was proposed by Agnese Augello [4], Manuel Gentile, Lucas Weideveld and Frank Dignum[2]. Their work relies on the premise [3] that social context is a key factor in the Natural language understanding aspect of the agent, hence creating a suitable response in a dialogue[1].

Furthermore improving upon the dialogue generation for conversational agents Allan Ritter, Will Monroe and Dan Jurafsky have developed a neural network model leaning towards Deep learning [6] coupled with reinforcement learning [13]. Instead of human and agent testing , their test was conducted test two virtual agents conversing with each other. Their conversation was judged [16] on variety, length, ease of answering and human evaluation [14]. They had performed SEQ2SEQ model [19], as their supervised learning for the agents through training , hence initializing them. The subsequent actions [15] would be responses (with the help of neural networks) measured on a reward based system (reinforcement learning) [12] keeping track of the previous conversations as a part of the state.

In terms of cyber security based chatbots, there has been two of mention. Demisto, a recent start-up had developed a bot on top of slack called dbot, which performs cyberanalytic tasks and informs users whether their files shared are safe. Dropbox had also recently announced deploying a cyber chat bot of its own, called Securitybot on slack. This Securitybot has a detection alert and notification based system which alerts users when they have committed a potential malicious act.

10 FUTURE WORK

The result of phase 1 of Cyber-Chatbot is that it can handle same question in more than one ways using natural language processing. It also has the capability of having conversations pertaining to the last uploaded file. The future work which is phase 2 has the prospects of adding three features which will improve upon our existing project.

(1) The ability to interact with multiple users at the same time. The chatbot right now has the ability to interact with one user at the moment, and answer questions regarding the last file uploaded by the user. Adding the ability to handle many users simultaneously would make this chatbot even more useful to security teams.
(2) The second feature that can be added to the chatbot involves other operations related to uploaded files from multiple users e.g. notifying one or more slack channels when a file is corrupted. There would be the possibility of sharing reports amongst two or more users from the chatbot.
(3) The last feature to the cyber chabot is creating a rich text format with which the chatbot will respond to the user. The rich text format is a way to further improve the format of responses from the chatbot like showing a mini preview of the links shared.

Apart from the existing functionalities if any of the new features are implemented in phase 2, our cyber chatbot will become a viable tool for security teams.

11 CONCLUSION

We have created a cyber chatbot that can answer basic virus related questions about a particular file or hash. We have also implemnted natural language processing enabling users to ask same question in more than one way. In a future version of this cyber chatbot more types of questions can be answered along and more functionalities can be implemented.

The cyber chatbot would enable a security analyst to be more efficient by doing menial and repetitive tasks.

ACKNOWLEDGMENTS

The authors would like to thank Dr. John Cavazos for providing the initial idea of the project Cyber-ChatBot. We would also like to thank Tristan for his continuous feedback. The work is supported by Amazon's AWS-free tier package available for free.

We would also like to thank the online open source community which gave us direction.

REFERENCES

[1] Sameera A Abdul-Kader and John Woods. 2015. Survey on chatbot design techniques in speech conversation systems. *Int. J. Adv. Comput. Sci. Appl.(IJACSA)* 6, 7 (2015).
[2] Agnese Augello, Manuel Gentile, Lucas Weideveld, and Frank Dignum. 2016. A model of a social chatbot. In *Intelligent Interactive Multimedia Systems and Services 2016*. Springer, 637–647.
[3] Agnese Augello, Giovanni Pilato, Alberto Machi, and Salvatore Gaglio. 2012. An approach to enhance chatbot semantic power and maintainability: experiences within the FRASI project. In *Semantic Computing (ICSC), 2012 IEEE Sixth International Conference on*. IEEE, 186–193.
[4] Agnese Augello, Antonella Santangelo, Salvatore Sorce, Giovanni Pilato, Antonio Gentile, Alessandro Genco, and Salvatore Gaglio. 2007. A multimodal interaction guide for pervasive services access. In *Pervasive Services, IEEE International Conference on*. IEEE, 250–256.
[5] AWS. 2011. AWS- Amazon Web Services. (2011). https://aws.amazon.com/iam/ [Online; accessed 10-May-2017].
[6] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).
[7] Businessinsider. 2011. Business Insider. (2011). http://uk.businessinsider.com/where-did-slack-get-its-name-2016-9?_ga=1.138509299.1175501442.1475074973?r=US&IR=T [Online; accessed 9-May-2017].
[8] Claudia. 2011. Framework-Claudia.js. (2011). https://claudiajs.com/tutorials/ [Online; accessed 9-May-2017].

[9] Virginia Dignum and Frank Dignum. 2014. Contextualized planning using social practices. In *International Workshop on Coordination, Organizations, Institutions, and Norms in Agent Systems*. Springer, 36–52.
[10] GitHub. 2011. GitHub link for Cyber-ChatBot project. (2011). https://github.com/cavazos-lab/spring-2017-CISC850-chatbot [Online; accessed 9-May-2017].
[11] Hash. 2011. Cryptographic Hash function. (2011). https://www.tutorialspoint.com/cryptography/cryptography_hash_functions.htmHASHsection [Online; accessed 11-May-2017].
[12] Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf. 2015. Deep reinforcement learning with a natural language action space. *arXiv preprint arXiv:1511.04636* (2015).
[13] Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541* (2016).
[14] Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv preprint arXiv:1603.08023* (2016).
[15] Yi Luan, Yangfeng Ji, and Mari Ostendorf. 2016. LSTM based Conversation Models. *arXiv preprint arXiv:1603.09457* (2016).
[16] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 311–318.
[17] Slack. 2011. Slack, The team collaboration tool. (2011). https://en.wikipedia.org/w/index.php?title=LaTeX&oldid=413720397 [Online; accessed 11-May-2017].
[18] Virus Total. 2011. VirusTotal: Public API. (2011). https://www.virustotal.com/en/documentation/public-api/ [Online; accessed 9-May-2017].
[19] Wojciech Zaremba and Ilya Sutskever. 2015. Reinforcement learning neural Turing machines. *arXiv preprint arXiv:1505.00521* 362 (2015).