Volume: 7 Issue: V Month of publication: May 2019

DOI: https://doi.org/10.22214/ijraset.2019.5098

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.177
Volume 7 Issue V, May 2019- Available at www.ijraset.com

Automation in Cyber-Security

Sonakshi Kothiyal[1], Dr. Priyanka Sharma[2], Aditya Chaudhary[3]
[1] Post Graduation, Cyber Security, M.Tech, Raksha Shakti University, Ahmedabad, Gujarat, India
[2] Professor & Head Dept of IT & Telecommunication Dept, Raksha Shakti University, Ahmedabad, Gujarat, India
[3] Senior Security Engineer, Lucideus, Delhi, India

Abstract: Making a Cyber-Security policy should seek solutions that leverage the expertise of both the private sector and federal Government. With the deluge of modern vulnerabilities, automated tools are needed to keep-up, but designing such application needs a distributed architecture to support scanning and testing. In response to this need, a distributed Framework has been designed that scans the network endpoints, collects and processes the scan data and provides an in-depth security analysis of the end points without any human intervention. The end result of this work is the creation of a tool for reducing manual effort involved in security assessment of a network.

This paper is based on Framework that provides in-depth security of network endpoints without any human intervention. It is a tool that integrates all the open source scanners that are available. It will ensure that the output of scanned data are readily available as all the processing is done on the server side. A run time server is required that is supported by RabbitMQ. RabbitMQ is a popular open-source and commercially- supported pub/sub systems that have been around for almost a decade and have seen wide adoption.

Keywords: Cyber-security, automated tools, multithreading, open source scanners, Publish/Subscribe Systems.

I. INTRODUCTION

In today's world it seems that everything relies on computers and the internet now used for shopping (online stores, credit cards), communication(email), medicine (equipment, medical records) entertainment (digital cable, mp3), transportation (airplane navigation) and the list goes on. Because of the most use of internet challenge in developing cyber security also increase. A user for one of these platforms might start spending weeks/months on scanning the system/network, outputs are collected and manually analyzed and build a report before scanning another system/network again. This cycle can continue hundreds or thousands of times and divert limited resources. As such, it is highly beneficial to build an automated tool that scans the system, collects the data, analyze the output and stores the results. In this paper, application runs on the server for scanning and testing the system. Run time server is required that is supported by RabbitMQ. RabbitMQ is message broker which is an open source. It is a queuing server that can be used to let disparate applications share data via a common protocol or to simply queue jobs for processing by distributed workers. RabbitMQ middleware supports many messaging protocols, among which the most important are STOMP: Streaming Text Oriented Messaging Protocol and AMQP: Advanced Messaging Queuing Protocol. Every message received by the RabbitMQ always is placed in a queue. Messages in queues can be stored in memory (memory-based) or on a disk (disk- based). Second important elements of the RabbitMQ are exchanges - the delivery service for messages. The exchange used by a publish operation determines if the delivery will be direct or publish and-subscribe, for example: When the message is published, client chooses the exchange which is used to deliver each message. The exchange looks at the information in the headers of a message and selects where they should be transferred to. This is how AMQP brings the various messaging idioms together. Clients can select which exchange should route their messages Given the popularity to this system and the fact that it is branded as pub/sub systems it reduces the burden of application designers by providing dedicated middle ware infrastructure.

II. BACKGROUND: PUB/SUB SYSTEMS

This section is highlight on the main concepts of the publish/subscribe paradigm. The primary purpose of this section is to establish a common framework/language. Knowledge- able readers may skip it.

A. Core Functionality

Publish/subscribe is a distributed interaction paradigm well adapted to the deployment of scalable and loosely coupled systems. The most fundamental functionality of a pub/sub system is decoupling the publishers and subscribers. The pub/sub coordination scheme provides along the following three dimensions: (1) Entity decoupling: publishers and consumers do not need to be aware of each other. The publisher/subscriber infrastructure terminates the interaction in the middle. (2) Time decoupling: The interacting parties need not to be active, at the same time. (3) Routing logic: pub/sub systems decide if and where a packet that is coming from a producer will end up at a consumer. Routing logic is mainly of two types:

1) Topic-based subscription is characterized by the publisher statically tagging the message with a set of topics, that can then be used very efficiently in the altering operation that decides which message goes to which consumer.
2) Content-based subscription does not need the producer to explicitly tag the message with routing context. All data and meta data fields of the message can be used in the altering condition.

III. DESIGN

We require RabbitMQ for building an RPC system as the function is running remotely on different computer. To analyze efficiency, there can be many agents as per the requirement and one command execution control which were continuously sending messages at a rapid pace to agents. The RabbitMQ Message bus was implemented in Python using the following steps:

1) *Establish:* Establish a TCP oriented connection between communicating parties that are command control center and agents.
2) *Create/Declare:* Create an exchange and a queue to store the given instructions.
3) *Command Control Center:* The instructions are pushed in the queue by the command center and it is received by the agents.
4) *Response Back:* Command center will get back the response from agents.

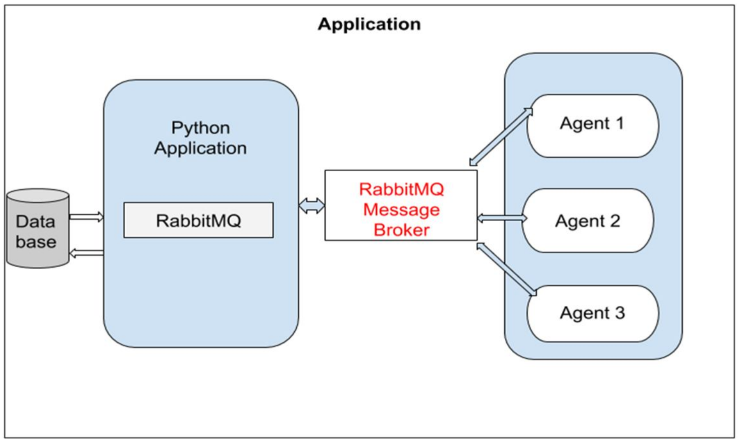

Fig. 1. Layout

RabbitMQ was deployed on a machine with the following specifications: HARDWARE DETAILS:

a) Processor: 8x Intel Core i7-6700K CPU @ 4GHz Memory (RAM):16 GB
b) Hard disk: 1 TB SOFTWARE DETAILS:
c) Operating System: Ubuntu 16.04.2 LTS
d) Programming Language: Python (pika package required for RabbitMQ APIs)
e) Software to be installed: RabbitMQ server.

The Command control center and agents were deployed on a machine with the following specifications: HARDWARE DETAILS:

a) Processor: Intel Core i5-6200U CPU @ 2.30GHz 4 Memory (RAM): 7.2 GiB
b) Hard disk:483.3 GB SOFTWARE DETAILS:
c) Operating System: Kali GNU/Linux Rolling Programming Language: Python
d) Soft-wares to be installed: None

IV. IMPLEMENTATION

In this RabbitMQ is a message broker, it accepts and forwards messages. Code is written in python language. RabbitMQ runs on the developer machine. Every developer should have access to a personal instance of the systems that they need to integrate with, as far as possible. The developers should have their own databases, message queues, web containers, and so on. As per my experience the easiest way to achieve this is to install the systems on each developer machine.

For implementation, RabbitMQ is being used to build an RPC system: an agent and a scalable RPC command control center. We are going to create a RPC service that returns the instructions.

1) *Agent Interface:* An agent class is created which is going to expose a method named call which sends an RPC request and blocks until the answer is received: rpc_calling= Agent() response=rpc_calling.call(instruction) print(" [.] Got %r" % response)
2) *The Instruction Includes:* command, file, any message, batches, status, and activities. Concept of multithreading is used to perform multiple task simultaneously. An agent sends a request message and command control center replies with a response message. In order to receive a response agent needs to send a 'callback' queue address with the request. We are creating a single callback queue per client to set a unique value for every request. When we receive a message in the callback queue and based on that we'll be able to match a response with a request. If an unknown correlation id value is seen, we may safely discard the message - it doesn't belong to our requests.
3) *Command control center Interface:* Command control center act as a server. It starts by establishing the connection and declaring the queue. Command class is created which waits for the request. We declare our instruction function. We declare a callback, the core of the RPC server. It's executed when the request is received. It does the work and sends the response back. We might want to run more than one command center process. In order to spread the load equally over multiple command center we need to set the prefetch count setting.

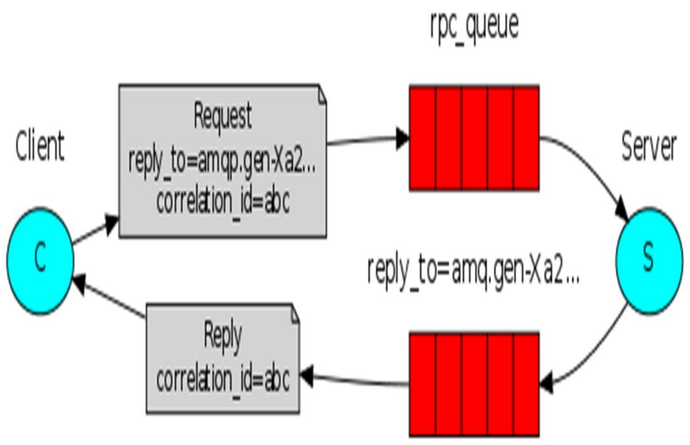

Fig. 2. Implementation

Our process will work like this

a) When the Agent starts up, it creates an anonymous exclusive callback queue.
b) For an RPC request, message is sent by the Agent with two properties: reply to, which is set to the callback queue and correlation id, which is set to a unique value for every request.
c) An rpc queue is a queue in which sends the request.
d) The RPC worker (aka: Command control center) is waiting for requests on that queue. When a request appears, it does the job and sends a message with the result back to agent, using the queue from the reply to field.
e) The Agent waits for instruction on the callback queue. When a message appears, it checks the correlation id property. The response is returned to the application if the value from the request.

V. CONCLUSION

In this paper, we have designed a Framework that integrates all the available APIs and open source scanners. We are trying to utilize/incorporate the concept of multithreading. This is done by testing and scanning through command center and automation. Even the users can plugin their modules or upgrade the packages by simply making the python script and zipping it in a file and pushing it to the agent. By doing this automated assessment can be done from the company itself without going anywhere.

REFERENCES

[1] http://www.rabbitmq.com
[2] http://blogs.vmware.com/vfabric/2013/02/choosing-your-messaging- protocol-amqp-mqtt-or-stomp.html
[3] http://en.wikipedia.org/wiki/RabbitMQ
[4] Ionescu, Valeriu Manuel. ”The analysis of the performance of RabbitMQ and ActiveMQ.”RoEduNet International Conference- Networking in Education and Research (RoEduNet NER), 2015 14th. IEEE, 2015.
[5] Peng, Zhen, Zhao Jingling, and Liao Qing. ”Message oriented mid- dleware data processing model in Internet of things.” Computer Science and Network Technology (ICCSNT), 2012 2nd International Conference on. IEEE, 2012.
[6] Rostanski, Maciej, Krzysztof Grochla, and Aleksander Seman. ”Eval- uation of highly available and fault-tolerant middleware clustered architectures using RabbitMQ.” Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on. IEEE, 2014
[7] http://www.cloudampq.com
[8] Rong Chen. ”Preliminary exploration on enterprise application integra- tion based on message-oriented middleware.” Consumer Electronics, Communications and Networks (CECNet), 2011 International Confer- ence on. IEEE, 2011.
[9] Tina AlSadhan, Joon S. Park. ”Security Automation for Informa- tion Security Continuous Monitoring:Research Framework.” World Congress on Services Computing IEEE,2016.
[10] Sanjit A. Seshia, Senior Member, IEEE, Shiyan Hu, Senior Member, IEEE,Wenchao Li, Member, IEEE, and Qi Zhu, Member, IEEE. ”De- sign Automation of Cyber-Physical Systems:Challenges, Advances, and Opportunities.” IEEE TRANSACTIONS ON COMPUTER- AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, VOL. 36, NO. 9, SEPTEMBER 2017.
[11] Jie Lin, Wei Yu, Nan Zhang, Xinyu Yang, Hanlin Zhang and Wei Zhao
, A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications IEEE Internet of Things Journal(Volume: 4, Issue: 5, Oct. 2017)
[12] https://www.python.org/
[13] Jorge E.Luzuriagaet al,A comparative evaluation of AMQP and MQTT protocols over unstable and mobile networks,Conference: Consumer Communications and Networking Conference (CCNC),2015 Annual IEEE, At Las Vegas,NV,Volume:12th.
[14] Carl Burchet al,Django,a web framework using Python:tutorial pre- sentation,Journal of Computing Sciences in College archive, Volume 25 Issue, May 2010 Page 154-155
[15] Ibarra I., Ward D.D., Ruddle A.,Threat analysis and risk assessment in automotive cyber security, SAE Technical Paper 2013-01-1415, 2013, DOI:10.4271/2013-01-1415
[16] C. W. Ten, et. al, Vulnerability Assessment of Cyber security t, IEEE Trans. On Power Systems, vol. 23, no. 4, pp. 1836-1846, 2008.
[17] R. Montesino and S. Fenz, Information security automation: how far can we go? In 2011 Sixth International Conference on Availabil- ity,Reliability and Security (ARES), pp. 280-285