**University of Victoria**

**Engineering & Computer Science Co-op**

**Work Term Report**

**Term (Summer) 2018**

**Implementation and Security Analysis of OSPF**

**Routing Protocol**

**NOKIA**

**A000001803/4L AA EP Sec Threat Intel Lab Alcatel-Lucent**

**Kanata, Ontario, Canada**

**Maninder Singh**

**V00879900**

**Work Term Number-2**

**Masters in Telecommunication and Information Security**

[**maninder.singh@uvic.ca**](mailto:maninder.singh@uvic.ca)

**August 20, 2018**

**In partial fulfillment of the academic requirements of this co-op term**

**Supervisor's Approval: To be completed by Co-op Employer**

This report will be handled by UVic Co-op staff and will be read by one assigned report marker who may be a co-op staff member within the Engineering and Computer Science Co-operative Education Program, or a UVic faculty member or teaching assistant. The report will be either returned to the student or, subject to the student’s right to appeal a grade, held for one year after which it will be destroyed.

I approve the release of this report to the University of Victoria for evaluation purposes only.

Signature: Position: Date:

 Name (print): ­­­ E-Mail: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For (Company Name)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Maninder Singh

94 Acklam Terrace

Kanata, Ontario, K2K 2J1

August 20, 2018

Dr. Kin Li

Dept. of Electrical & Computer Engineering

University of Victoria

P.O. Box 1700

Victoria, B.C. V8W 2Y2

Canada

Dear Dr. Kin Li

Please accept the accompanying Work Term Report entitled “Implementation and Security Analysis of OSPF Routing Protocol.”

This report is the result of second work term completed at Nokia (formerly Alcatel-Lucent) Kanata, Ontario. I was working in the position of malware research assistant in Nokia’s Threat Intelligence Lab. As a Co-op student I was responsible to read research articles related to the latest malware attacks, then write signatures that alert in the field if that kind of malware is active, doing passive analysis for the existing signatures to make sure they alert on right kind of traffic with no false positives.

Through this course of the term, I got the opportunity to learn valuable concepts about malware, their working behaviour, how they spread, and strategies to stop them. I feel this knowledge will be very beneficial for me in the future co-op terms and further in my career.

I would like to thank my manager Mr. Paul Edwards, security engineer Mr. John, and Mr. Pierre for sharing their knowledge and experiences with me.

Sincerely,

Maninder Singh

NOKIA, Kanata, ON

Attachment: Implementation and Security Analysis of OSPF Routing Protocol

TABLE OF CONTENTS

LIST OF FIGURES v

SUMMARY vi

[1.0 INTRODUCTION 1](#_Toc523399855)

[2.0 ATTACKS IN OSPF 2](#_Toc523399856)

[2.1 REMOTE FALSE ADJACENCY ATTACK 2](#_Toc523399857)

[2.2 DISGUISED LSA ATTACK 3](#_Toc523399858)

[2.3 MaxAge ATTACK 3](#_Toc523399859)

[2.4 ADJACENCY SPOOFING ATTACK 4](#_Toc523399860)

[2.5 SINGLE PATH INJECTION 5](#_Toc523399861)

[3.0 SECURITY STRENGTHS AND VULNERABILITIES IN OSPF 7](#_Toc523399862)

[3.1 SECURITY STRENGTHS: 7](#_Toc523399863)

[3.2 VULNERABILITIES IN OSPF 9](#_Toc523399864)

[4.0 REAL EXAMPLES OF OSPF ATTACKS 11](#_Toc523399865)

[4.1 MULTIPLE CISCO PRODUCTS OSPF LSA MANIPULATION VULNERABILITY 11](#_Toc523399866)

[4.2 MaxAge LSA VULNERABILITY IN OSPF PROTOCOL IN SOME HUAWEI PRODUCTS 12](#_Toc523399867)

[5.0 SIMULATION 14](#_Toc523399868)

[5.1 OSPF ROUTING PROTOCOL IMPLEMENTATION 14](#_Toc523399869)

[5.2 IMPLEMENTATION OF DISGUISED LSA ATTACK 18](#_Toc523399870)

[6.0 APPROACHES TO DETECT AND RECOVER FROM OSPF ATTACKS 20](#_Toc523399871)

[6.1 ATTACKS DETECTION SYSTEM BASED ON PACKET ANALYSIS 20](#_Toc523399872)

[6.2 DETECTING AND PROTECTING FROM OSPF ATTACKS USING ANALYSER ROUTER 22](#_Toc523399873)

[7.0 CONCLUSION 24](#_Toc523399874)

[8.0 RECOMMENDATIONS 25](#_Toc523399875)

[REFERENCES 26](#_Toc523399876)

**TABLE OF FIGURES**

[Figure 1: Remote false adjacency attack 3](#_Toc523063847)

[Figure 2: Procedure for adjacency spoofing attack 5](#_Toc523063848)

[Figure 3: Single Path Injection Attack network topology 6](#_Toc523063849)

[Figure 4: Packet header of OSPF packet 7](#_Toc523063850)

[Figure 5: Cisco OSPF LSA manipulation vulnerability 11](#_Toc523063851)

[Figure 6: Huawei OSPF MaxAge vulnerability 13](#_Toc523063852)

[Figure 7: Network topology for OSPF implementation 14](#_Toc523063853)

[Figure 8: OSPF Hello packet captured 15](#_Toc523063854)

[Figure 9: OSPF Database Description (DD) packet captured 16](#_Toc523063855)

[Figure 10: OSPF Link State Update (LSU) packet captured 16](#_Toc523063856)

[Figure 11: Route traced to 20.0.0.2 17](#_Toc523063857)

[Figure 12: Network topology for Disguised LSA attack 18](#_Toc523063858)

[Figure 13: LSA packet captured to create Disguised LSA 19](#_Toc523063859)

[Figure 14: Disguised LSA sent by attacker 19](#_Toc523063860)

[Figure 15: Acknowledgement received by router 6 20](#_Toc523063861)

[Figure 16: Attack detection system architecture 21](#_Toc523063862)

[Figure 17: Packet Analysis and Detection Module flowchart 22](#_Toc523063863)

**SUMMARY**

Routing protocols like RIP, OSPF, EIGRP etc. forms the basis of internet and this fast-growing need for internet applications proposes a great challenge for internet security. So, in order to enforce network security, security at routing and switching mechanism as well as routing protocols is vital. OSPF is one of the widely used link state, open standard routing protocol which uses Dijkstra’s algorithm to initially construct a shortest path route and then populate the resulting routing tables with best shortest path. Without this correct routing table information there would be inefficient network transmission which may cause failure of the network. Also, there are attacks where a hacker can use to poison these routing table with false link status information which in turn may paralyse the network and can further lead to DNS spoofing, phishing, man-in-the-middle attacks.

So, in this project, we would first implement OSPF in packet tracer cisco tool in order to understand the basic operations of OSPF: neighbor and adjacency initialization, LSA flooding, SPF tree calculation. Then, we would emphasis on security of OSPF by studying various poison routing table attacks which bypass “fight-back” mechanism of OSPF and then implementing disguised LSA attack in packet tracer tool. Finally, an approach would be learned which would be used to detect and recover from these kinds of attacks.

# **1.0 INTRODUCTION**

DevOps started as a culture and set of practices to support collaboration and communication across development and operations, and to apply automation to key phases of the software delivery process. It has been popularized by successful new companies developing business models and related applications empowered by the cloud (cloud-native applications). More recently, large, established enterprises have recognized the need to deliver innovation faster to stay relevant and capitalize on industry disruption, while also improving operational metrics for application quality and cost.

# DevOps is a set of practices that automates the processes between software development and IT teams, in order that they can build, test, and release software faster and more reliably. The concept of DevOps is founded on building a culture of collaboration between teams that historically functioned in relative siloes. The promised benefits include increased trust, faster software releases, ability to solve critical issues quickly, and better manage unplanned work.

**History of DevOps**

The DevOps movement started to coalesce some time between 2007 and 2008, when IT operations and software development communities got vocal about what they felt was a fatal level of dysfunction in the industry.

They railed against the traditional software development model, which called for those who write the code to be organizationally and functionally apart from those who deploy and support that code.

Developers and IT/Ops professionals had separate (and often competing) objectives, separate department leadership, separate key performance indicators by which they were judged, and often worked on separate floors or even separate buildings. The result was siloed teams concerned only with their own fiefdoms, long hours, botched releases, and unhappy customers.

Surely there’s a better way, they said. So the two communities got together and started talking – with people like Patrick Dubois, Gene Kim, and John Willis driving the conversation.

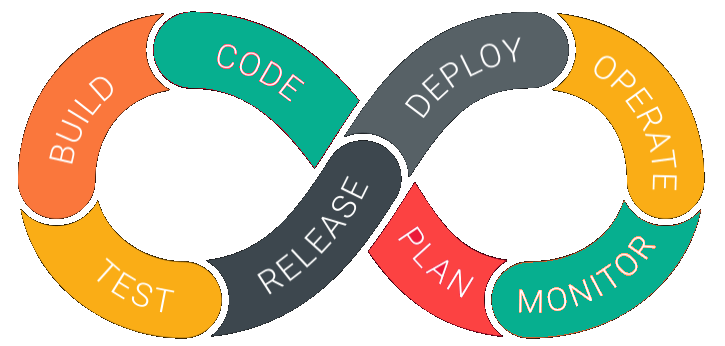
What began in online forums and local meet-ups is now a major theme in the software zeitgeist, which is probably what brought you here! You and your team are feeling the pain caused by siloed teams and broken lines of communication within your company.

You’re using agile methodologies for planning and development, but still struggling to get that code out the door without a bunch of drama. You’ve heard a few things about DevOps and the seemingly magical effect it can have on teams and think “I want some of that magic.”

The bad news is that DevOps isn’t magic, and transformations don’t happen overnight. The good news is that you don’t have to wait for upper management to roll out a large-scale initiative. By understanding the value of DevOps and making small, incremental changes, your team can embark on the DevOps journey right away. Let’s look at each of these benefits in detail.

# **2.0 DevOps Life Cycle**

DevOps Lifecycle

[](https://www.guru99.com/images/2-2017/092917_0812_DevOpsTrain2.png)

DevOps is deep integration between development and operations. Understanding DevOps is not possible without knowing DevOps lifecycle.

Here is a brief information about the Continuous DevOps life-cycle:

1. Development

In this DevOps stage the development of software takes place constantly. In this phase, involves continuous iteration of development and testing in the SDLC process. This software development method emphasizes on iterative, incremental, and evolutionary development. the entire development process is separated into small development cycles. This benefits DevOps team to speed up software development and delivery process.

2. Testing

QA team use tools like Selenium to identify and fix bugs in the new piece of code.

3. Integration

In this stage, new functionality is integrated with the prevailing code, and testing takes place. Continuous development is only possible due to continuous integration and testing. In Continuous Integration after a code commit, the software is built and tested immediately. In a large project with many developers, commits are made many times during a day. With each commit code is built and tested. If the test is passed, build is tested for deployment. If deployment is a success, the code is pushed to production. This commit, build, test, and deploy is a continuous process and hence the name continuous integration/deployment.

A Continuous Integration Pipeline is a powerful instrument that consists of a set of tools designed to host, monitor, compile and test code, or code changes.

Continuous delivery picks up where continuous integration ends. CD automates the delivery of applications to selected infrastructure environments. Most teams work with multiple environments other than the production, such as development and testing environments, and CD ensures there is an automated way to push code changes to them. CD automation then performs any necessary service calls to web servers, databases, and other services that may need to be restarted or follow other procedures when applications are deployed.

Continuous integration and delivery requires continuous testing because the objective is to deliver quality applications and code to users. Continuous testing is often implemented as a set of automated regression, performance, and other tests that are executed in the CI/CD pipeline.

4. Deployment

In this phase, the deployment process takes place continuously. It is performed in such a manner that any changes made any time in the code, should not affect the functioning of high traffic website.

A mature CI/CD practice has the option of implementing continuous deployment where application changes run through the CI/CD pipeline and passing builds are deployed directly to production environments. Teams practicing continuous delivery elect to deploy to production on daily or even hourly schedule, though continuous delivery isn’t always the optimal for every business application.

5. Monitoring

In this phase, operation team will take care of the inappropriate system behavior or bugs which are found in production.

# **3.0 DevOps Principles**

# Here, are six principles which are essential when adopting DevOps:

# 1. Customer-Centric Action: DevOps team must take customer-centric action for that they should constantly invest in products and services.

# 2. End-To-End Responsibility: The DevOps team need to provide performance support until they become end-of-life. This enhances the level of responsibility and the quality of the products engineered.

# 3. Continuous Improvement: DevOps culture focuses on continuous improvement to minimize waste. It continuously speeds up the improvement of product or services offered.

# 4. Automate everything: Automation is a vital principle of DevOps process. This is not only for the software development but also for the entire infrastructure landscape.

# 5. Work as one team: In the DevOps culture role of the designer, developer, and tester are already defined. All they needed to do is work as one team with complete collaboration.

# 6. Monitor and test everything: It is very important for DevOps team to have a robust monitoring and testing procedures.

# **4.0 Roles, Responsibilities, and Skills of a DevOps Engineer**

# DevOps engineers work full-time. They are responsible for the production and ongoing maintenance of a software application's platform.

# Following are some expected Roles, Responsibilities, and Skills that is expected from DevOps engineer:

# Able to perform system troubleshooting and problem-solving across platform and application domains.

# Manage project effectively through open, standards-based platforms

# Increase project visibility thought traceability

# Improve quality and reduce development cost with collaboration

# Analyse, design and evaluate automation scripts & systems

# Ensuring critical resolution of system issues by using the best cloud security solutions services

# DevOps engineer should have the soft skill of problem-solver and quick-learner

# **5.0 SDLC**

# The entire SDLC process divided into the following stages:

# <https://www.guru99.com/images/1/080118_0641_SDLCSoftwar1.png>

# Phase 1: Requirement collection and analysis

# Phase 2: Feasibility study:

# Phase 3: Design:

# Phase 4: Coding:

# Phase 5: Testing:

# Phase 6: Installation/Deployment:

# Phase 7: Maintenance:

# In this tutorial, I have explained all these phases

# Phase 1: Requirement collection and analysis:

# The requirement is the first stage in the SDLC process. It is conducted by the senior team members with inputs from all the stakeholders and domain experts in the industry. Planning for the quality assurance requirements and recognization of the risks involved is also done at this stage.

# This stage gives a clearer picture of the scope of the entire project and the anticipated issues, opportunities, and directives which triggered the project.

# Requirements Gathering stage need teams to get detailed and precise requirements. This helps companies to finalize the necessary timeline to finish the work of that system.

# Phase 2: Feasibility study:

# Once the requirement analysis phase is completed the next step is to define and document software needs. This process conducted with the help of 'Software Requirement Specification' document also known as 'SRS' document. It includes everything which should be designed and developed during the project life cycle.

# There are mainly five types of feasibilities checks:

# Economic: Can we complete the project within the budget or not?

# Legal: Can we handle this project as cyber law and other regulatory framework/compliances.

# Operation feasibility: Can we create operations which is expected by the client?

# Technical: Need to check whether the current computer system can support the software

# Schedule: Decide that the project can be completed within the given schedule or not.

# Phase 3: Design:

# In this third phase, the system and software design documents are prepared as per the requirement specification document. This helps define overall system architecture.

# This design phase serves as input for the next phase of the model.

# There are two kinds of design documents developed in this phase:

# High-Level Design (HLD)

# Brief description and name of each module

# An outline about the functionality of every module

# Interface relationship and dependencies between modules

# Database tables identified along with their key elements

# Complete architecture diagrams along with technology details

# Low-Level Design(LLD)

# Functional logic of the modules

# Database tables, which include type and size

# Complete detail of the interface

# Addresses all types of dependency issues

# Listing of error messages

# Complete input and outputs for every module

# Phase 4: Coding:

# Once the system design phase is over, the next phase is coding. In this phase, developers start build the entire system by writing code using the chosen programming language. In the coding phase, tasks are divided into units or modules and assigned to the various developers. It is the longest phase of the Software Development Life Cycle process.

# In this phase, Developer needs to follow certain predefined coding guidelines. They also need to use programming tools like compiler, interpreters, debugger to generate and implement the code.

# Phase 5: Testing:

# Once the software is complete, and it is deployed in the testing environment. The testing team starts testing the functionality of the entire system. This is done to verify that the entire application works according to the customer requirement.

# During this phase, QA and testing team may find some bugs/defects which they communicate to developers. The development team fixes the bug and send back to QA for a re-test. This process continues until the software is bug-free, stable, and working according to the business needs of that system.

# Phase 6: Installation/Deployment:

# Once the software testing phase is over and no bugs or errors left in the system then the final deployment process starts. Based on the feedback given by the project manager, the final software is released and checked for deployment issues if any.

# Phase 7: Maintenance:

# Once the system is deployed, and customers start using the developed system, following 3 activities occur

# Bug fixing - bugs are reported because of some scenarios which are not tested at all

# Upgrade - Upgrading the application to the newer versions of the Software

# Enhancement - Adding some new features into the existing software

# The main focus of this SDLC phase is to ensure that needs continue to be met and that the system continues to perform as per the specification mentioned in the first phase.

# **6.0 STLC**

There Software Testing Life Cycle (STLC) is defined as a sequence of activities conducted to perform Software Testing.

It consists of series of activities carried out methodologically to help certify your software product.

[](https://www.guru99.com/images/stories/software-test-life-cycle.jpg)

Diagram - Different stages in Software Test Life Cycle

Each of these stages have a definite Entry and Exit criteria; , Activities & Deliverables associated with it.

What is Entry and Exit Criteria?

Entry Criteria:Entry Criteria gives the prerequisite items that must be completed before testing can begin.

Exit Criteria: Exit Criteria defines the items that must be completed before testing can be concluded

You have Entry and Exit Criteria for all levels in the Software Testing Life Cycle (STLC)

In an Ideal world you will not enter the next stage until the exit criteria for the previous stage is met. But practically this is not always possible. So for this tutorial, we will focus on activities and deliverables for the different stages in STLC life cycle. Lets look into them in detail.

Requirement Analysis

This is the very first phase of Software testing Life cycle (STLC). In this phase testing team goes through the Requirement document with both Functional and non-functional details in order to identify the testable requirements.

In case of any confusion the QA team may setup a meeting with the clients and the stakeholders (Technical Leads, Business Analyst, System Architects and Client etc.) in order to clarify their doubts.

Once the QA team is clear with the requirements they will document the acceptance Criteria and get it approved by the Customers.

Activities to be done in Requirement analysis phase are given below:

• Analyzing the System Requirement specifications from the testing point of view

• Preparation of RTM that is Requirement Traceability Matrix

• Identifying the testing techniques and testing types

• Prioritizing the feature which need focused testing

• Analyzing the Automation feasibility

• Identifying the details about the testing environment where actual testing will be done

Deliverables (Outcome) of Requirement analysis phase are:

• Requirement Traceability Matrix (RTM)

• Automation feasibility report

Test Planning

Test Planning phase starts soon after the completion of the Requirement Analysis phase. In this phase the QA manager or QA Lead will prepare the Test Plan and Test strategy documents. As per these documents they will also come up with the testing effort estimations.

Activities to be done in Test Planning phase are given below:

• Estimation of testing effort

• Selection of Testing Approach

• Preparation of Test Plan, Test strategy documents

• Resource planning and assigning roles and responsibility to them

• Selection of Testing tool

Deliverables (Outcome) of Test Planning phase are:

• Test Plan document

• Test Strategy document

• Best suited Testing Approach

• Number of Resources, skill required and their roles and responsibilities

• Testing tool to be used

Test Case Development

In this phase the QA team write test cases. They also write scripts for automation if required. Verification of both the test cases and test scripts are done by peers. Creation of Test Data is done in this phase.

Activities to be done in Test Case Development phase are given below:

• Creation of test cases

• Creation of test scripts if required

• Verification of test cases and automation scripts

• Creation of Test Data in testing environment

Deliverables (Outcome) of Test Case Development phase are:

• Test cases

• Test scripts (for automation if required)

• Test Data

Test Environment setup

This phase includes the setup or installation process of software and hardware which is required for testing the application. In this phase the integration of the third party application is also carried out if required in the project.

After setting up the required software and hardware the installation of build is tested. Once the installation of build is successful and complete then the Test Data is generated.

After the creation of Test data the Smoke testing is executed on the build in order to check whether the basic functionalities are working fine or not. This phase can be done in parallel with the Test Case Development phase.

Activities to be done in Test Environment Setup phase are given below:

• As per the Requirement and Architecture document the list of required software and hardware is prepared

• Setting up of test environment

• Creation of test data

• Installation of build and execution of Smoke testing on it

Deliverables (Outcome) of Test Environment Setup phase are:

• Test Environment setup is ready

• Test Data is created

• Results of Smoke testing

Test Execution

Before starting the Test Execution phase the Test Environment setup should be ready. In Test Execution phase the test cases are executed in the testing environment.

While execution of the test cases the QA team may find bugs which will be reported against that test case. This bug is fixed by the developer and is retested by the QA.

Activities to be done in Test Execution phase are given below:

• Execution of Test Cases

• Reporting test results

• Logging defects for the failed test cases

• Verification and retesting of the defect

• Closure of defects

Deliverables (Outcome) of Test Execution phase are:

• Test execution Report

• Updated test cases with results

• Bug Report

Test Cycle Closure

In order to start the Test Cycle Closure activity the Test Execution phase should be completed. In Test Cycle phase the QA team will meet and discuss about the testing artifacts.

The whole intent of this discussion is to learn lessons from the bad practices. This will help in future projects.

Activities to be done in Test Cycle Closure phase are given below:

• To evaluate the test completion on the basis of Test Coverage and Software Quality

• Documentation of the learning from the project

• Analyzing the test results to find out the distribution of severe defects

• Test Closure Report preparation

Deliverables (Outcome) of Test Cycle Closure phase are:

• Report of Test Closure

**7.0 Why is DevOps used? (How DevOps is different than traditional IT**

DevOps allows Agile Development Teams to implement Continuous Integration and Continuous Delivery. This helps them to launch products faster into the market.

Other Important reasons are:

1. Predictability: DevOps offers significantly lower failure rate of new releases

2. Reproducibility: Version everything so that earlier version can be restored anytime.

3. Maintainability: Effortless process of recovery in the event of a new release crashing or disabling the current system.

4. Time to market: DevOps reduces the time to market up to 50% through streamlined software delivery. This is particularly the case for digital and mobile applications.

5. Greater Quality: DevOps helps the team to provide improved quality of application development as it incorporates infrastructure issues.

6. Reduced Risk: DevOps incorporates security aspects in the software delivery lifecycle. It helps in reduction of defects across the lifecycle.

7. Resiliency: The Operational state of the software system is more stable, secure, and changes are auditable.

8. Cost Efficiency: DevOps offers cost efficiency in the software development process which is always an aspiration of IT companies' management.

9. Breaks larger code base into small pieces: DevOps is based on the agile programming method. Therefore, it allows breaking larger code bases into smaller and manageable chunks.

**7.0 DevOps Automation Tools**

It is vital to automate all the testing processes and configure them to achieve speed and agility. This process is known as DevOps automation.

The difficulty faced in large DevOps Team that maintain large huge IT infrastructure can be classified briefly into six different categories.

1. Infrastructure Automation

2. Configuration Management

3. Deployment Automation

4. Performance Management

5. Log Management

6. Monitoring.

Let's see a few tools in each of these categories and how they solve the pain points–

Infrastructure Automation

Amazon Web Services (AWS): Being cloud service you do not need to be physically present in the data center. Also, they are easy to scale on-demand. There are no up-front hardware costs. It can be configured to provision more servers based on traffic automatically.

Configuration Management

Chef: It is a useful DevOps tool for achieving speed, scale, and consistency. It can be used to ease out complex tasks and perform configuration management. With this tool, DevOps team can avoid making changes across ten thousand servers. Instead, they need to make changes in one place which is automatically reflected in other servers.

Deployment Automation

Jenkins: This tool facilitates continuous integration and testing. It helps to integrate project changes more easily by quickly finding issues as soon as a built is deployed.

Log Management

Splunk: This is a tool solves the issues like aggregating, storing, and analyzing all logs in one place.

Performance Management

App Dynamic: It is DevOps tool which offers real-time performance monitoring. The data collected by this tool helps developers to debug when issues occur.

Monitoring

Nagios: It is also important to make sure people are notified when infrastructure and related services go down. Nagios is one such tool for this purpose which helps DevOps teams to find and correct problems.

**7.0 Jenkins**

Jenkins is an open source Continuous Integration server capable of orchestrating a chain of actions that help to achieve the Continuous Integration process (and not only) in an automated fashion.

Jenkins is entirely written in Java. Jenkins is a widely used application around the world that has around 300k installations and growing day by day.

It is a server-based application and requires a web server like Apache Tomcat. The reason Jenkins became so popular is that of its monitoring of repeated tasks which arise during the development of a project. For example, if your team is developing a project, Jenkins will continuously test your project builds and show you the errors in early stages of your development.

By using Jenkins, software companies can accelerate their software development process, as Jenkins can automate build and test at a rapid rate. Jenkins supports the complete development lifecycle of software from building, testing, documenting the software, deploying and other stages of a software development lifecycle.

Advantages of using Jenkins

• Jenkins is being managed by the community which is very open. Every month, they hold public meetings and take inputs from the public for the development of Jenkins project.

• So far around 280 tickets are closed, and the project publishes stable release every three months.

• As technology grows, so does Jenkins. So far Jenkins has around 320 plugins published in its plugins database. With plugins, Jenkins becomes even more powerful and feature rich.

• Jenkins also supports cloud-based architecture so that you can deploy Jenkins in cloud-based platforms.

• The reason why Jenkins became popular is that it was created by a developer for developers.

Disadvantages of using Jenkins

Though Jenkins is a very powerful tool, it has its flaws.

• Its interface is out dated and not user friendly compared to current UI trends.

• Though Jenkins is loved by many developers, it's not that easy to maintain it because Jenkins runs on a server and requires some skills as server administrator to monitor its activity.

• One of the reasons why many people don't implement Jenkins is due to its difficulty in installing and configuring Jenkins.

• Continuous integrations regularly break due to some small setting changes. Continuous integration will be paused and therefore requires some developer attention.

**7.0 CONCLUSION**

In this project, we have first implemented OSPF routing protocol in packet tracer to understand its working and basic terminologies of OSPF. Then various loop-holes and vulnerabilities were identified. We then implemented Disguised LSA attack in packet tracer. After analyzing the various attacks, we came to the point that these attacks are very easy to implement in real life scenario and industry people using OSPF are not aware of these attacks. So, proper prevention and detection techniques need to be implemented so as to protect a great data breach or attack in future.

////////

• In Continuous Integration, after a code commit, the software is built and tested immediately

• Jenkins is an open source Continuous Integration server capable of orchestrating a chain of actions

• Before Jenkins when all Developers had completed their assigned coding tasks, they used to commit their code all at same time. Later, Build is tested and deployed.

• After Jenkins the code is built and test as soon as Developer commits code. Jenkin will build and test code many times during the day

• By default, Jenkins comes with a limited set of features. If you want to integrate your Jenkins installation with version control tools like Git, then you need to install plugins related to Git

• The biggest pros of Jenkins is that it is managed by the community which holds public meetings and take inputs from the public for the development of Jenkins projects

• The biggest con of Jenkin is that Its interface is out dated and not user friendly compared to current UI trends.

# **8.0 RECOMMENDATIONS**

Based on the vulnerabilities in OSPF, it is recommended to use OSPF v.3 wherever possible as it is not affected by such vulnerabilities. Other recommendations of using OSPF v.2 includes using OSPF authentication, probably MD5 authentication which is recommended as a best practice to mitigate such attacks. Moreover, it is recommended not to process OSPF packets without a valid key so as to prevent replay attacks.

# **REFERENCES**

[1] Kirshon A, Gonikman D, Nakibly G. Owning the routing table—new OSPF attacks. BlackHat Briefings and Trainings USA+ 2011. 2011:1-8.

[2] P. R. Tadimety and SpringerLink (Online service), OSPF: A Network Routing Protocol. 2015.

[3] J. T. Moy, OSPF: Anatomy of an Internet Routing Protocol. Reading, Mass: Addison-Wesley, 1998.

[4] Sangroha D, Gupta V. Analyzer Router: An Approach to Detect and Recover from OSPF Attacks. In International Symposium on Security in Computing and Communication 2014 Sep 24 (pp. 370-378). Springer, Berlin, Heidelberg.

[5] Y. Song et al, "Novel attacks in OSPF networks to poison routing table," in 2017,. DOI: 10.1109/ICC.2017.7996829.

[6] W. Ming-Hao, "The security analysis and attacks detection of OSPF routing protocol," in 2014, DOI: 10.1109/ICICTA.2014.200.

[7] <https://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-20170727-ospf>, last accessed on 24th August, 2018.

[8] <http://www.huawei.com/en/psirt/security-advisories/huawei-sa-20170720-01-ospf-en>, last accessed on 24th August, 2018.

[9] T. Lammle, CCNA: Routing and Switching, 2017 (reprinted), p.745-770.

[10] Zhijun Q. Research on OSPF Routing Protocol’s Security [D][D]. Northeast Petroleum University, 2007.

[11] Wang, Feiyi and Wu, etc., “On the vulnerabilities and protection of OSPF routing protocol,” in Computer Communications and Networks, IEEE, 1998.