Uncovering Router Protocol Vulnerabilities through Intelligent Fuzzing

By

Asad Mehdi

A thesis submitted to the faculty of Information Security Department, Military College of Signals, National University of Sciences and Technology, Rawalpindi in partial fulfilment of the requirements for the degree of MS in Information Security

October 2018

THESIS ACCEPTANCE CERTIFICATE

It is certified that final copy of MS Thesis written by **Asad Mehdi** Registration No. **00000170761**, of **Military College of Signals** has been vetted by undersigned, found complete in all respect as per NUST Statutes/Regulations, is free of plagiarism, errors and mistakes and is accepted as partial, fulfillment for award of MS degree. It is further certified that necessary amendments as pointed out by GEC members of the scholar have been also incorporated in the said thesis.

Signature: _____

Name of Supervisor:_____

Date: _____

Signature (HOD):_____

Date: _____

Signature (Dean/Principal):_____

Date: _____

Declaration

I hereby declare that no portion of work presented in this thesis has been submitted in support of another award or qualification either at this institution or elsewhere.

Dedication

This thesis is dedicated to MY FAMILY, TEACHERS AND FRIENDS for their love, endless support and encouragement.

Acknowledgement

First of all, I would like to thank Allah Almighty for His countless blessings. After that I want to express my appreciation to my family, my friends, colleagues and the faculty for providing their enormous support to help me to do this research. Without their relentless support, assistance and prayers, I would not have reached culmination point in a peaceful state of mind.

I would like to convey my gratitude to my supervisor, Dr.Mehreen Afzal, for her supervision and constant support. Her invaluable help of constructive comments and suggestions throughout the experimental and thesis work are major contributions to the success of this research. I'm greatly admired by her dedication, sincerity towards work and humble attitude with her students.

I'm thankful to my committee members; Dr.Fawad Khan, Asst Prof Mian Muhammad Waseem Iqbal for their support.

I am highly thankful to my friend Capt Arslan Shah and course mates Maj Shahid Rafiq, Lt Cdr Kaleemullah, Mr. Arsalan, Mr. Abdul Rehman, Mr. Haseeb Javed, Mr. Akash Gerard, Mr. Muhammad Waqas and Mr. Syed Adeel Shah who helped me during different phases of this research and course work.

Asad Mehdi

October 2018

Copyright Notice

Abstract

Simple Network Management Protocol(SNMP) is a renown network management protocol. In an IP network, it is used for gathering information from, and configuring network devices. Due to its extensive usage and presence in critical network devices such as routers, switches,servers etc, its security preservation is a major concern. Any security flaw in software implementation of such protocol may lead to catastrophic situations. A lot of SNMP vulnerabilities have been reported in the past. The modern vulnerability assessment tools only look for known vulnerabilities but lack zero-day detection.

Fuzzing, is an automated software security testing technique which is typically known for finding zero-day, buffer-overflow and memory corruption vulnerabilities effectively. Researches on finding unknown flaws in protocol implementations of network devices through fuzzing is still immature. The existing open-source tools do not cater fuzz testing of complex protocols due to their data modeling complexity.

In this research, fuzz testing of Simple Network Management Protocol(SNMP) implementation in Cisco routers is performed. A simple approach for generating malformed test-cases is also proposed. During fuzz testing experiments several memory corruption and a known DoS vulnerability is exposed. Analysis of SNMP vulnerabilities for renown vendors in scope of fuzz testing and evaluation of prominent network protocol fuzzing tools based on certain criteria is also part of this thesis.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

1.1 Introduction

This chapter gives an overview of the basic concepts underlying behind this research such as Fuzz testing, Router Protocols with their brief working and different forms of attacks on them. Aim, motivation, scope and contributions have been explained later in the chapter and finally the thesis organization.

1.2 Network Protocols

Protocols, in context of digital networks, are the set of rules that are needed to be followed by any two entities on a network to communicate with each other. Since this research revolves around the routers, it only discuss the different types of network protocols which are implemented in a router which extends its limited capability of performing routing of packets only to serving application layer communication. Protocols in routers are mainly of two types: **Routing Protocols** and **Application layer Protocols**

Routing Protocols: Routing protocols form the rules that allow the exchange of routing and data information. These are layer-3 i.e. network layer protocols hence routers are also referred as layer-3 network devices. Efficiency of a router is most critical to any network. Routing protocol has vital impact on efficiency of router. The purpose of

any routing protocol is to determine the best path for a packet to reach its final destination. Routing protocols also called as Dynamic Routing Protocols because of their adaptive nature. They have the flexibility to adapt to all sorts of conditions, such as traffic congestion, poor link speed, or a complete disconnection of a particular route.

Dynamic Routing protocols are further differentiated based on what and how much information they share with other nodes in the network and upon which attributes routing decisions are made. Figure 1.1 shows a diagram of how routing protocols are differentiated.

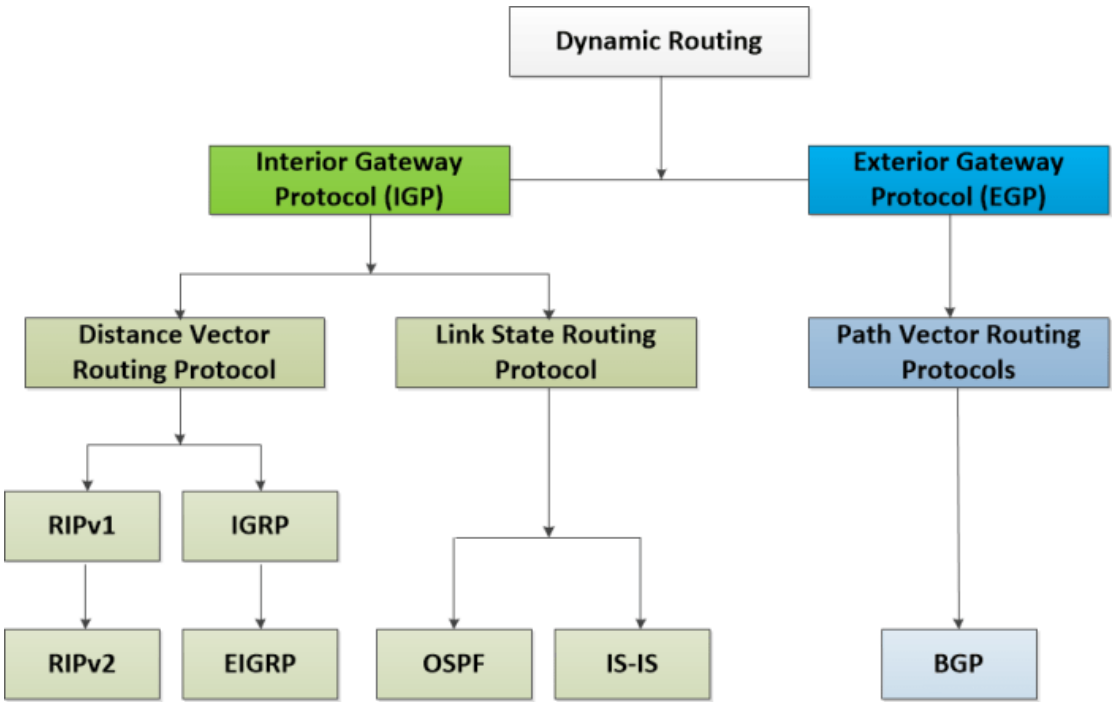

Figure 1.1: Dynamic Routing Protocols

Data networks are mainly divided into private and public network. The private part is an internal network of an organization while the public part is internet. To facilitate communication between these two networks there are different set of protocols. A protocol used to enable communication within a local network is referred as **Interior Gateway Protocol** whereas for data transactions outside the network is achieved with the help of **Exterior Gateway Protocol**.

Interior Gateway Protocol is a generalized term for the protocols used inside a LAN. As mentioned earlier, different protocols use different attributes, network information

and approaches to perform routing. Hence, there are two main classes of IGPs. DVRPs make routing decisions based on the distance or hop count, hence named so. The distance from one router to the other is counted as one hop. For instance, if there exist more than one path to the same destination, the path with less number of routers or hops is preferred. These protocols don't have full knowledge of network topology. are commonly used DVRPs inside an AS. There exists two versions of RIP. The latter version i.e. RIPv2 only differs, that it supports VLSM. EIGRP is Cisco's proprietary protocol with VLSM support for LAN routing. Due to interoperability issues, network administrators avoid the deployment of proprietary protocols.

LSRPs are widely deployed in enterprise networks due to their enormous capacity of handling larger networks. As compared to DVRPs they don't just rely on hop count to make routing decisions rather collect other metrics, such as reliability, bandwidth, delay and path-cost to build their routing tables. Combining all these metrics enable them to make efficient routing decisions. Open Shortest Path First(OSPF) is a LSRP which uses Dijkstra algorithm for finding shortest path between two nodes. It is a non-proprietary routing protocol that uses a number of advertisement and acknowledgment packets to inform only its neighbors regarding routing information.

Exterior Gateway Protocols play their role when it comes to traversing of packets over the internet or public network. **Path Vector Routing Protocol** is the only class of EGP.

Application Layer Protocols: Application Layer takes the seventh position in OSI model. It is the only layer with which the humans can interact. There are tens of application layer protocols which set the rules for software applications to communicate or exchange the data with their peers or to avail the desired service. A few well known application layer protocols are:**HTTP/HTTPS, FTP, SMTP, TELNET, SSH, SNMP, DHCP, DNS** and others. Take an example of a user wants to access a specific web page. To accomplish this task, the browser application at the back-end uses an HTTP/HTTPS request message to fetch that content from the web server. Similarly, for remotely accessing any device TELNET or SSH protocol is used. Email communication is setup by SMTP, FTP facilitates transferring of files, SNMP is used to remotely manage the

nodes on a network. For dynamic allocation of ip addresses DHCP plays its role and DNS to resolve domain names to ip addresses. The list is never ending, but the focus is only on those application layer protocols which are made a part of routers and the need for making them so.

TELNET, is a famous utility among network administrators to access different nodes on network, which eases the task of making any configuration changes without being physically there. **SSH** does the same but with added security. Both of these protcols justify their place to be a part of any routing device because of the services they offer. **DHCP** enabled routers makes it easier to allocate ip addresses to other hosts on the network but it is not commonly practiced. **HTTP/HTTPS** are also implemented in router OS to receive vendor specific updates. **Simple Network Management Protocol(SNMP)** helps to remotely manage the network devices, to check configurations, current state of the device and any modifications due to topological changes. The clients or agents can also be set to report the server if there happens any change. Network professionals widely adopt SNMP for centralized management of routers and other devices on the network. There are lots of other features offered by SNMP which are briefly discussed in Chapter No. 4

1.3 Attacks on Router

Routers are the core components of any packet-switching network. A security breach or compromise of these devices may result in catastrophic situation. Routers can be compromised either by DoS/DDoS attacks, gaining access and modifying their configurations, poisoning the routing table or flooding the bandwidth. All of these attacks can be hit-and-run or persistent which are discussed below.

Denial of Service Attacks: DoS/DDoS attack is one of the common type of attack in the wild. The prime objective of this attack is to put a device at halt in-order to prevent it from entertaining any legitimate requests. Such an attack on router may result in complete network jam for the subnets connected to it. The situation can be even worse

if the attack victim is border router or firewall.

Hit-and-Run Attacks: Hit-and-Run Attacks are referred to the type of attack in which a single or few malformed packets result in compromise of a device. The attack may also result in complete device crash or making certain service unavailable. Such attacks require high level of expertise and thorough knowledge of the victim. Due to the same reason, these attacks are harder to spot than DoS attack.

Persistent Attacks: Persistent Attacks are the ones in which an attacker gains unauthorized access to system or network and remains there undetected, for a long period. The detection possibility of these attacks is much higher because for an attacker to maintain access there is an ongoing stream of packets to analyze. These attacks can be carried out through routing table poisoning or by leveraging the protocol implementation vulnerabilities.

Packet Mistreating Attacks: Packet Mistreating Attacks involve modification of actual data packets and then injecting them. These malformed packets introduce confusion and result in mishandling or mistreating of legitimate packets by the router. Such an attack is confined to a part of network and does not affect the whole network. The outcome of such an attack may include DoS and network congestion.

Routing Table Poisoning: Routers make routing decisions with help of routes stored in a database known as Routing Table. RTP attacks occur when an attacker injects fake routes into a routing table. When these fake routes are cycled throughout the network, the routing table of neighboring nodes also get affected. This series of events have drastic impact on the network components and may result in suboptimal routing, congestion, partition, overwhelmed host and unauthorized access to data.

Protocol Attacks: Protocol Attacks involve leveraging by design vulnerabilities and injection of control protocol messages to attack routing protocols. Disruption of internal communication is achieved when such an attack occurs. Most common examples are:

DNS Poisoning, Interior Routing Protocols Injections (OSPF and EIGRP Injections), ARP Poisoning and BGP Hijacking.

1.4 Fuzz Testing

Fuzzing or Fuzz testing, a proven black-box testing technique which is used to find known and unknown software vulnerabilities where a target is fed with fuzzed data to process, and then monitored in parallel for any unexpected behavior [3] .The technique is favorable for automated testing of the products for which the source code is not available such as Cisco IOS (Internetwork Operating System) which runs as a single binary image and decompressed at boot-time only [4]. One of the main challenge of fuzz testing is to generate efficient test cases which may produce reliable verdicts. Test cases are normally constructed in one of the three fashions: totally random input, mutation of existing data and generation of malformed data from scratch with prior knowledge of target [3]. The latter approach is complex to implement but results in better code coverage, hence yields fruitful result [5].

1.5 Motivation and Problem Statement

Routers are the core component of any digital network which primarily act as a gateway for any data to be travelled in or outside the network. Due to advancements in technology there are multiple features being integrated into routers which turns it into a complex device. With several vendors in the market and tens of protocols implemented to aid respective services, manufacturers and developers often miss on considering the security flaws that are embed in those devices at development phase due to flawed coding practices, complex architecture of protocols and improper testing. In current era of cyberwar, these neglected flaws may aid hackers and across the border agencies and enemies to carry out cyber-attacks such as network hijacking, data exfiltration, covert monitoring, network sabotage and others. As earlier disclosed by Wikileaks in its Vault 7 series how CIA was in control of 318 different models of Cisco switches and routers

[6]. Therefore, there is greater need to develop and adopt effective testing techniques which must be practiced prior to utilization of such products.

1.6 Aims and Objectives

1. To explore the effectiveness of Fuzzing technique for vulnerability analysis of network protocols.

2. Evaluation of well-known fuzzing frameworks for uncovering router protocol vulnerabilities.

3. To propose generalized fuzzing methodology for exposing known/unknown vulnerabilities of router protocols.

4. Practical validation of proposed methodology by targeting Simple Network Management Protocol (SNMP) in Cisco routers.

1.7 Thesis Contributions

This section enlists the major contributions of this thesis.

Fuzzing effectiveness for Network Protocols: Fuzzing technique in its current state is thoroughly studied with the prime focus on measuring its effectiveness and feasibility to fuzz network Protocols. Evaluation of renowned Open Source fuzzing frameworks(Peach, Spike, Kitty and Sulley) is performed based on certain metrics.

Generalized Methodology for Fuzzing Network Protocols: A generalized approach is proposed to fuzz test network protocols after encountering a common limitation offered by current open source fuzzing tools. The approach utilizes a free-ware network traffic sniffer: Wireshark and a binary editor to mutate the protocol fields.

Fuzz testing Cisco router: The proposed methodology is adopted to fuzz test Cisco router by emulating the real-world Cisco IOS in Dynamips. SNMP with all its three versions is tested. Several memory corruption and a known DoS vulnerability is exposed during the test process.

1.8 Thesis Organization

This research document is divided into six chapters. Following is the summary of each chapter.

- **Chapter 1:** This chapter includes introduction to topic, brief explanation of network protocols and their attacks, research aims and objectives. It also highlights contributions of this research.
- **Chapter 2:** Comprehensive literature review of the topic is covered in this chapter. Fuzzing in its current state and its different approaches proposed over the time are touched. Also, survey of the potential risk carried by Cisco routers and their exploitation methods presented by different researchers are discussed.
- **Chapter 3:** This chapter contains detailed insight of the Fuzzing technique and its different types. Various open-source fuzzers with their purpose are discussed. Evaluation of open source network protocol fuzzing frameworks is the dominant excerpt of this chapter.
- **Chapter 4:** The chapter elaborates our target protocol i.e. SNMP, its all three versions with detailed working and also the analysis of potential vulnerable points in SNMP to fuzz efficiently. To the date reported SNMP vulnerabilities of different vendors are also analyzed in scope of their exploitation possibility through fuzz testing.
- **Chapter 5:** This chapter includes the proposed methodology for fuzzing network protocols, fuzzing experiments, obtained results and exploitation of SNMP vulnerabilities in Cisco routers.

- **Chapter 6:** Concludes this research and proposes future directions.

Chapter 2

LITERATURE REVIEW

2.1 Introduction

This chapter contains comprehensive review of fuzzing and different techniques adopted to maximize its efficiency. Fuzzing applicability for network protocols and challenges in path of fuzzing complex protocol with their existing solutions are discussed. Later in the chapter, cisco router exploitation methods proposed by different authors and the recent researches for fuzzing Cisco routers are reviewed.

2.2 Fuzzing - A software vulnerability testing technique

This section includes compact review of fuzzing origin, evolution of fuzz testing techniques over the time, its different approaches, areas of application and potential protocol vulnerabilities exposed with the aid of fuzzing.

Fuzzing is a dynamic software vulnerability testing technique which can effectively be used for hunting bugs in network protocols as well. Barton Miller developed the first ever fuzzer for robustness testing of UNIX utilities which used random input mutations [7]. The main challenge of fuzz testing is the generation of input test cases which may produce reliable verdicts. There exist three approaches for generating test cases: Random, mutation-based and generation based. The generation-based approach is proven to be more effective and time saving but requires handful knowledge of the target [8].

Over the time different test case generation techniques have been introduced to fuzzing for improving its efficiency which includes Grammar Representation, Scheduling Algorithms, Dynamic Symbolic Exectuion, Coverage Feedback and Dynamic Taint Analysis. Apart from evolution of test case generation techniques, modern day fuzzers are also featured to monitor the targets, restart the target if a crash occurs, log test-cases which produced exception and provide summary at the end of fuzzing session.

2.3 Network Protocols Fuzzing

Fuzzing as an automated testing technique for finding bugs and vulnerabilities has also been widely adopted for testing network protocols to ensure reliable communication.

2.3.1 Stateful Protocol Fuzzing

When it comes to fuzzing network protocols there exist several challenges which are needed to be addressed and overcome in order to develop an effective fuzzer. Complex protcols are usually comprised of different protocol states. SNOOZE: a stateful network protcol fuzzer was developed in [9]. Session Initiation Protocol(SIP) was tested with initial prototype of SNOOZE and successfully exposed several real-world vulnerabilities. PULSAR [10] is another stateful protocol fuzzer which is able to fuzz generic as well as proprietary protocols. The proprietary protocol fuzzing by PULSAR is performed by inferring a protocol model from captured protocol traces. Several known and unknown vulnerabilities were exposed while testing OSCAR, a proprietary protocol implemented in many instant messengers.

2.3.2 Checksum Aware and Encrypted Protocol Fuzzing

Network Protocols usually contain fields which keep on updating such as checksums and hashes. An intelligent fuzzer should incorporate such features for in depth testing of complex protocols. In [11] a checksum aware fuzzing tool named TaintScope was developed by incorporating dynamic taint analysis and symbolic execution techniques.

Evaluation was done on real world applications which included Adobe Acrobat, Google Picasa, Microsoft Paint, ImageMagick and exposed 27 unknown vulnerabilities. Similary for encrypted protcols, a light-weight, yet effective technique is proposed in [12]. IKE(Internet Key Exchange) protocol was tested and found two new vulnerabilities.

2.3.3 Wireless Fuzzing

Fuzzing can also be performed for security testing of wireless protocols. Laurent Butti at Black Hat Europe(2007) made a practical demonstration of fuzzing 802.11 (wi-fi) and revealed several unknown vulnerabilities [13]. A case study of fuzzing Bluetooth, Wi-fi and WiMAX technologies is performed in [14] with aim of exploring their implementation vulnerabilities.

2.3.4 AutoFuzz: Automated Network Protocol Fuzzing

Serge Gorbunov and Arnold Rosenbloom introduced an open source automated network protocol fuzzing tool named AutoFuzz [15]. The tool works by adopting man-in-the-middle approach and captures the traffic between client and server. Autofuzz learns a protocol implementation by constructing a Finite State Machine(FSM). The authors were able to discover all known and few unknown vulnerabilities in a variety of FTP server implementations. The major limitation of this tool is its ability to fuzz only text-based protocols such as FTP, SMTP and HTTP.

2.3.5 Application Layer Protocol Fuzzing

Fuzzer developement of application layer protocolos is quite usual among security researchers because it is the only layer with which humans directly interact. In [16] TFTP fuzzer was developed and tested on a number of windows based TFTP applications. A number of open source HTTP, SMTP and FTP fuzzers are available in the wild but currently there exist no open source fuzzing tool which can fuzz complex binary protocols.

2.4 Cisco Router Exploitation and Fuzzing

This section focuses on different methods and techniques suggested by security professional for exploiting Cisco routers and also includes to the date literature review of fuzzing for router vulnerability mining.

In the last ten years or so, methods for discovering vulnerability and similar study on routers have been advancing rapidly, and most research focuses on Cisco router. Felix Linder from Phenoelit examines various Cisco IOS vulnerabilities and different exploitation techniques [17]. Michael Lynn proposed a technique which showed how to completely hijack an IOS-based router, with the help of buffer overflow or a heap overflow, two types of memory vulnerabilities [18]. Gyan Chawdhary and Varun Uppal proposed a method which shows how to write shell codes with GNU debugger and debug Cisco IOS, which makes router attack feasible [19]. Sebastian Muniz and Alfredo Ortega presented a tool which facilitates debugging and reverse engineering process of Cisco IOS by allowing the integration with widely used debugging and disassembler tools such as GDB and IDA Pro [23]. The tool is basically modification of original dynamips version. This research utilizes the same tool for IOS emulation and debugging purpose.

Fuzzing applicability on routers is not much explored. There exists only few researches for mining router vulnerabilities through fuzzing. In [24] an ICMP fuzzer was developed and validated on cisco router which resulted in successful disclosure of buffer overflow vulnerability. The test-cases were generated by adopting SFTCG(Simple Fuzzing Test Case Generation) approach. RPFuzzer [25], a closed source fuzzer for discovering protocol vulnerabilities in routers was developed. TFTCG(Two Stage Fuzzing Test Case Generation) method was adopted for creating malformed inputs. SNMP protocol in Cisco router was targeted for practical validation and claimed to uncover four known and two unknown vulnerabilities across all three versions of protocol.

The major limitation of above two researches is: the tools are not made open source which leaves a question mark on their practical validation by research community.

2.5 Conclusion

This chapter highlights the most effective fuzz testing tools and frameworks proposed for performing implementation test of network protocols. It can be concluded from existing researches that fuzzing can be handy in finding unknown flaws in network protocol implementations. But specific to the routers, researches only focus on how to exploit them instead of automated discovery of zero-day vulnerabilities. A lot of research is still needed to be done for implementation testing of complex network protocols.

Chapter 3

Fuzzing and Evaluation of Network Protocol Fuzzing Tools

3.1 Introduction

This chapter is one of the vital pieces of this research because the whole work revolves around the technique being explained in this chapter. At first the basic idea of fuzzing is described with its different types. The fuzzing engine, its different parts which make up an actual fuzzer and also the techniques adopted to improve fuzzing efficiency are briefly explained. Later in the chapter, an evaluation of network protocol fuzzing tools is also performed based on some metrics which may aid the penetration testers and QA assurance professionals to choose the right tools as per their skills and requirements.

3.2 Fuzzing

Fuzzing is an automated testing technique for exposing potential vulnerabilities in software or network protocol by intentionally feeding malformed data to system under test (SUT). The technique involves from sending totally random data to specifically crafted packets which can effectively discover flaws in the code.

In other words, the prime purpose of fuzz testing is to reveal security critical flaws

which may result in service degradation, denial of service, or other anomalous behavior. During fuzz testing, the main focus is to find such inputs which can trigger unexpected behavior. Combining randomness and protocol knowledge coupled with certain degree of automation may yield fruitful results with low labor cost. As compared to manual testing, such attempts may expose promising flaws.

The first ever fuzzer was developed by Barton Miller in 1989 for robustness testing of UNIX utilities[7]. The provoking incident behind the concept of fuzz testing was line noise induced in author‘s cable modem during a storm. The randomness introduced by noise caused the program to crash. This observed behavior was later termed as ‘fuzz’.

Currently, fuzz testing is adopted by computer security analysts to expose and report bugs, by Quality Assurance professionals to enhance the quality of their software products and by hackers to uncover and exploit the hidden flaws. Fuzz testing is also adopted by renown software companies in their development lifecycle of softwares[26] to come up with better products.

3.2.1 Types Of Fuzzing

The three basic types of Fuzzing are differentiated as below:

White-box Fuzzing: The source code of the target is known to the analyst in white-box fuzzing. Fuzz testing of applications developed in-house and open source applications comes under umbrella of white-box fuzzing.

Grey-box Fuzzing: Grey-box fuzzing is performed on targets for which the source code is not available. Testing is done by injecting binary/assembly code using a debugger and monitored in parallel. Most of the vendor developed softwares are closed source, this is what fuzzing is mainly focused at.

Black-box Fuzzing: In black-box fuzzing, the testing is performed by randomly sending the input to the target because no source code or information about the target is

available.

3.2.2 Test-case Generation Approaches

One of the challenging task in process of finding vulnerabilities through fuzzing is generation of test-cases. In-order to produce reliable and meaningful verdicts, test cases can be generated in one of the following two fashions:

Mutation Based: In mutation based fuzzing test cases are generated by mutating fields of already available data samples or packet traces.

Generation Based: Test cases are generated from scratch by modeling target protocol or file format with the help of RFCs or available documentation.

Research proves, the latter approach to be more efficient in finding loopholes, but its more complex to implement as it requires complete knowledge of the target [27]. Fuzzers are also differentiated based on their test-cases generation approach (i.e. Mutation based, or Generation based fuzzers)

3.2.3 Structure of a Fuzzer

Moder day fuzzers do not just focus completely on techniques for generating test-cases. Fuzzers nowdays are equipped with several features such as automation, reporting, failure analysis and others.

Protocol Modeler: To incorporate the ability of modeling various data types, message formats and protocol messages, protocol modeler is used. The trivial models are based on templates, on the contrary complex models can make use of context-free grammars.

Anomaly Library: It is the collection of inputs known for triggering the vulnerabilities in software. Modern day fuzzers include this feature to expose existing vulnerabil-

ities.

Attack Simulation Engine: Utilizes existing anomalous data heuristics or learns from them. The heuristics available in the tool coupled with randomness are applied to form malformed test cases for fuzz test.

Runtime Analysis Engine: Performs monitoring of target during fuzz test. Several methods can be utilized to interact with the target under test.

Reporting: The testing results are needed to be presented in a meaningful way, which can help identify the exact loopholes. Few tools do not facilitate report while others perform complex bug reporting.

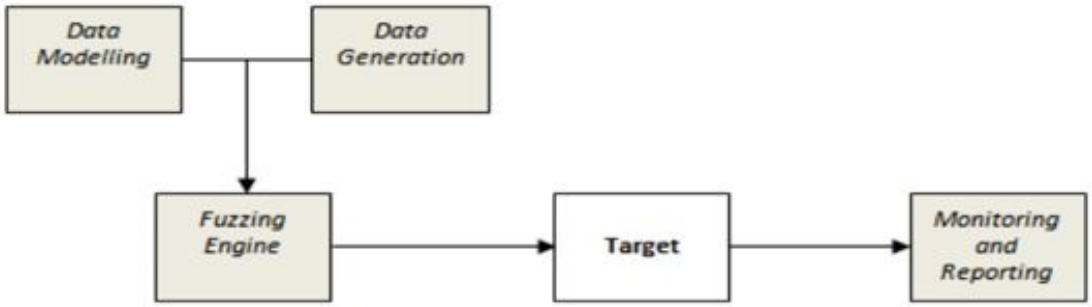

Figure 3.1: Generic Fuzzer model

3.2.4 Fuzzing Process

A common process of fuzz test comprises stream of messages (requests and responses) which are sent to the target. The resulting modifications and unexpected behavior can then further analyze or can be completely ignored as shown in Figure 3.2. Following are the responses of a typical fuzz test:

Valid response: A normal behaviour.

Error response: (Can be a legit reply from protocol point of view).

Anomalous response: (unexpected behavior but not critical).

Crash or other failure: Due to unexpected input or memory corruption.

Fuzzing is way more than just sending and receiving messages. Generation of test cases is first done and then sent to the target. Target is constantly under monitoring during the test.

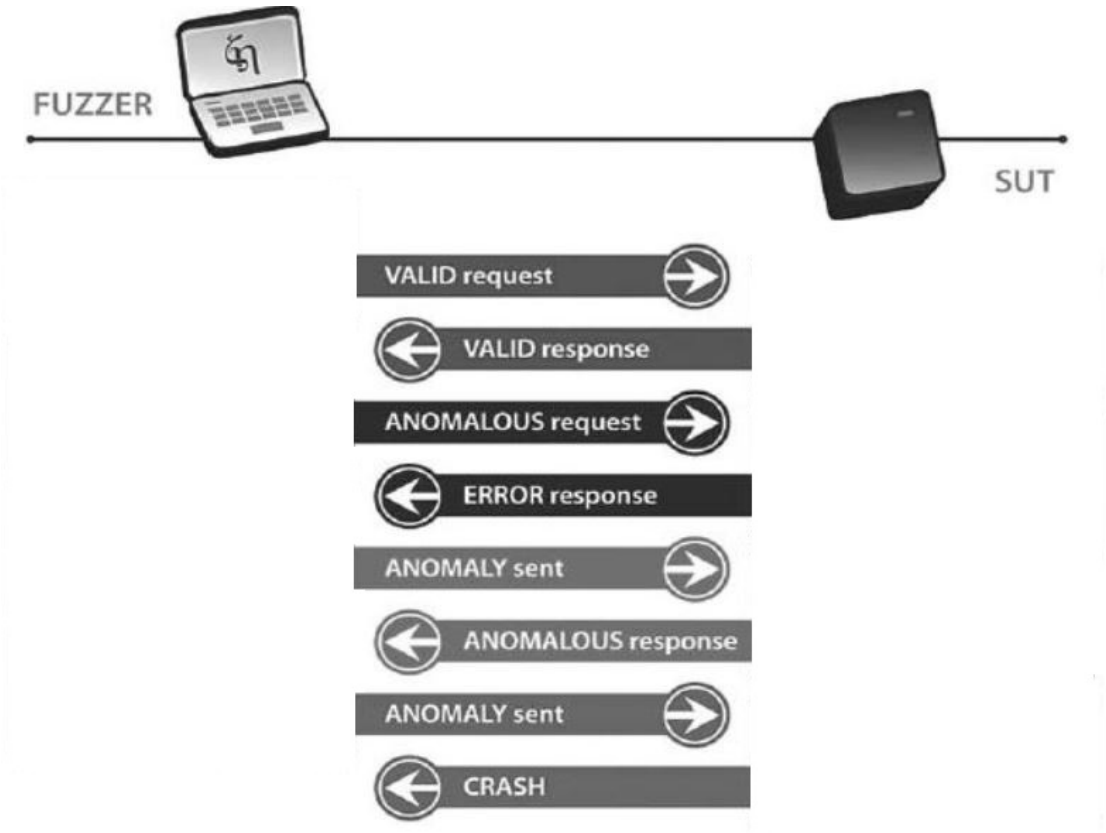

Figure 3.2: Example fuzz test cases and resulting responses from an SUT [28]

3.3 State of the Art Fuzzing Tools

A number of fuzzing tools have been developed for a wide range of protocols and different scopes. Over 300 tools are available in the wild which aid fuzz testing of different targets [29]. Fuzzer can be divided into two types: fuzzing frameworks and fuzzers which target specific protocols. Fuzzing frameworks facilitate fuzzer development. Peach, Spike and Kitty are well known examples of network protocol fuzzing

frameworks.

3.4 Types of Fuzzers

Fuzzers are mainly differentiated based on the type of target they can fuzz. Following are the main classes of fuzzers.

3.4.1 File Format Fuzzers

Every application both server or client, deal with input and output file. For example, antivirus gateways often need to parse com-pressed files to diagnose what lies within them. Another example is an office productivity suite that needs to open a document. Such applications can be susceptible to vulnerabilities that occur when parsing maliciously crafted files.

This is where file format fuzzing comes in. A file format fuzzer will dynamically create different malformed files that are then launched using the target application. Although the specific methods used in file fuzzing are not exactly the same as other types of fuzzing, the general idea is the same.

3.4.2 Remote Fuzzers

Softwares that listens on a network interface can be target through remote fuzzers. Network-enabled applications are likely the most targeted fuzzing target. With the growth of Internet, virtually all corporations now have publicly accessible servers to deliver Web pages, e-mail, Domain Name System (DNS) resolution, and so on. Such vulnerability could provide an attacker with access to confidential data or a launch pad to carry out further attacks.

3.4.3 Network Protocol Fuzzers

Network protocol fuzzers can be broken into two major categories: those that target simple protocols and those designed for complex protocols. Following definitions differentiate between each type.

Simple Network Protocols: Simple network protocols mostly contain simple authentication or no authentication at all. They are often based on ASCII printable text as opposed to binary data. A simple protocol will not contain length or checksum fields. Additionally, there are typically not many states within the application.

An example of a simple protocol is FTP. In FTP, all control channel communications are in plain ASCII text. For authentication, only a plain text user-name and a password is required.

Complex protocols: Complex network protocols are typically comprised of binary data with the occasional human-readable ASCII string. Authentication might require encryption or some form of obfuscation and there might be several complex states involved.

As in this research, the targeted protocol i.e. SNMP is a good example of a complex protocol despite the name "Simple Network Management Protocol". SNMP is a binary protocol. The protocol requires length description fields and fragmentation, encryption and authentication (SNMPV3). Overall, it is not a fun protocol to implement for fuzzing.

3.4.4 Web Application Fuzzers

Web applications have become popular as a convenient way to access back-end services such as e-mail and bill paying. With the advent of Web 2.0, traditional desktop applications such as word processing are moving to the Web.

When fuzzing Web applications, the researcher is primarily looking for vulnerabilities unique to Web applications such as SQL injection, Cross Site Scripting (XSS), and so

on. This necessitates fuzzers capable of communicating via Hypertext Transfer Protocol (HTTP) and capturing responses for further analysis to identify the existence of vulnerabilities.

3.5 Well-Known Fuzzing Tools

A number of renowned fuzzing tools are listed in below table . The table also contains the type of target they can fuzz and the nature of the product i.e. Open/closed-source.

Fuzzing Tools	Target	Open/ Closed Source
QuickFuzz American Fuzzy Loop (AFL) BFF FOE go-Fuzz OFuzz Radamsa	File	Open Source
Backfuzz Dizzy Spike Sulley Pulsar Netzob Kitty	Protocol	Open Source
Peach	Protocol	Closed Source
beStorm	Protocol+File	Closed Source+Paid
Wfuzz Filebuster Rfuzz	Web	Open Source
TriforceAFL Passive Fuzz Framework OSX	OS	Open Source
Syzkaller Kernel Fuzzer Trinity IOCTL	OS Kernel	Open SOurce
Wadi Grinder NodeFuzz	Browser	Open Source
LibFuzzer	Library	Open Source
Csmith LangFuzz	Compiler	Open Source
Syntribos	API	Open Source
KEMU Fuzzer	Visual Machine	Open Source
Diffy	Service	Open Source
Dranzer	ActiveX	Open Source
Js-Fuzz	Java Script	Open Source
PyJFuzz	JSON data	Open Source
Nightmare BrundleFuzz FuzzFlow ClusterFuzz	Distributed	Open Source
Regex-Fuzzer	Regular Expressions	Open Source

Table 3.2: Well-Known Fuzzing Tools.

3.6 Comparison of Network Protocol Fuzzers

Since the objective of this thesis is mainly to test network protocols through fuzzing technique, a comprehensive comparison of few well-known network protocol fuzzers is performed in-order to make it easier for the readers to get an insight of the features of each tool and help penetration testers to pick up the right tool for fuzzing network protocols based on their skill set and distinguished features of fuzzers.

Fuzzers Comparison Factors: Factors and criteria are derived based on the requirements of target user: A software developer or penetration tester with little experience of security testing beforehand, no financial assistance for security testing and a little time to acquaint himself with the tool, and with the objective of hunting at least most trivial security loopholes of already implemented or custom network protocols. The three factors are **Ease of Use, Practical Features of the Fuzzer** and **Fuzzing Intelligence**.

Criterion for each of the factors is mapped with description in the following tables:

1. **Ease of Use**

Ease of use criteria	Description
Executable availability	Ready to install executable or needed to compile?
Documentation	Readme or PDF document about components and usage of the tool with trivial examples.
Tool Assistance	Yes
Environmental requirements	Platform independent or OS/hardware limitations?

Table 3.4: Criterion for Ease-of-Use of the Fuzzer

2. **Practical features of the Fuzzer**

Practical features criteria	Description
Automation Support	Support for automation of test runs?
Target Monitoring	Built-in support for monitoring the target during the test available or not?
Customization	Options for tool customization?

Table 3.6: Criterion for Practical features of the Fuzzer

3. **Fuzzing Intelligence**

Fuzzing intelligence criteria	Description
Test data generation options	Totally random, data mutation or generation of data from scratch.
Fuzzing heuristics utilization	Yes
Data modeling complexity	Feasibility to perform generation-based fuzzing.

Table 3.8: Criterion for Fuzzing Intelligence

3.6.1 Selected Fuzzers

Selection of the fuzzers to be compared was made as an exploratory research using most dominant search engine, Google and on the experience gained during the experimental phase of this thesis for implementing or testing the widely adopted SNMP protocol. Since the research focus is on network protocols, therefore only network protocol fuzzers and fuzzing frameworks which allow to build own fuzzer are selected. Based on their popularity and search engine results, the selected contenders for network protocol fuzzers comparison are: **Peach, Spike, Kitty** and **Sulley**.

3.6.2 Individual Evaluation of Network Protocol Fuzzers

The tools are first evaluated independently based on above mentioned three factors

PEACH: Peach is a cross-platform smart fuzzing framework capable of performing both generation and mutation-based fuzzing. The framework lets you develop your own file and network protocol fuzzers. Peach has been under active development for seven years and is in its third major version, actively developed by Michael Eddington of Deja vu Security. Initial version was written in Python, later moved on to Microsoft .NET Framework, primarily C#.

Criteria and possible advantages of Peach:

Ease of Use	**Possible advantage**
Executable availability	Yes(Binaries). Source code is available for previous versions which are buggy.
Documentation	Yes. Comprehensive documentation with tutorials.
Tool Assistance	Yes.
Environmental prerequisites	Cross-platform i.e. runs under Windows, Mac OS-X and Linux.
Practical Features of Fuzzer	**Possible Advantage**
Automation Support	Yes. Multiple fields mutation.
Target Monitoring	Yes, through agents which can be run locally or remotely.
Customization	Yes.
Other features	Fuzzer is extensible, Distributed Fuzzing, Error logging
Fuzzing Intelligence	**Possible Advantage**
Test data generation options	All three approaches available i.e. Random, mutation, data-modeling.
Fuzzing heuristics utilization	Yes. Both smart and random mutations are included.
Data modeling complexity	Fairly easy. XML editor is used to create Pitfiles which contain all the information needed for Peach to perform fuzzing.

Table 3.10: Evaluation of Peach Fuzzer

KITTY: Kitty is an open-source modular and extensible fuzzing framework written in python by Cisco SAS team. Just like Peach, Kitty is a fuzzing framework which includes common functionality of every fuzzing process required to write your own fuzzer. It doesnâĂŹt contain implementation of specific protocol or communication

channel but provides a baseline for developing custom protocol fuzzers.

Criteria and possible advantages of Kitty:

Ease of Use	**Possible advantage**
Executable availability	Yes. Source code is available.
Documentation	Yes. Comprehensive documentation with few examples.
Tool Assistance	Yes.
Environmental prerequisites	Cross-platform i.e. runs under Windows, Mac OS-X and Linux.
Practical Features of Fuzzer	**Possible Advantage**
Automation Support	Yes. Multiple fields mutation.
Target Monitoring	Yes, through agents which can be run locally or remotely.
Customization	Yes.
Other features	Stateful fuzzing, fuzzer is extensible, error-logging
Fuzzing Intelligence	**Possible Advantage**
Test data generation options	All three approaches available i.e. Random, mutation, data-modeling.
Fuzzing heuristics utilization	Yes. Both smart and random mutaions are included.
Data modeling complexity	Moderate. Requires good knowledge of Python and its associated libraries.

Table 3.12: Evaluation of Kitty Fuzzer

SULLEY: Sulley is a python based fuzzing framework that can be used to fuzz file formats, network protocols, command line arguments, and other codes.

Criteria and possible advantages of Sulley:

Ease of Use	Possible advantage
Executable availability	Yes. Source code is available. Lots of dependencies are required for installation.
Documentation	Yes. Comprehensive documentation with no examples.
Tool Assistance	Through Mail. No active development.
Environmental requirements	Cross-platform i.e. runs under Windows, and Linux only.
Practical Features of Fuzzer	**Possible Advantage**
Automation Support	Yes.
Target Monitoring	Yes, through agents.
Customization	Yes.
Other features	Stateful fuzzing, fuzzer is extensible, error logging, Parallel fuzzing.
Fuzzing Intelligence	**Possible Advantage**
Test data generation options	All three approaches available i.e. Random, mutation, data-modeling.
Fuzzing heuristics utilization	Yes. Both smart and random mutaions are included.
Data modeling complexity	Moderate. Requires good knowledge of Python and its associated libraries.

Table 3.14: Evaluation of Sulley Fuzzer

SPIKE: Spike is one of the basic and pioneer fuzzer creation kit, providing an API that allows a user to create their own fuzzers for network-based protocols using the C programming language.

Criteria and possible advantages of Spike:

Ease of Use	Possible advantage
Executable availability	Yes. Source code is available.
Documentation	Poorly documented.
Tool Assistance	Through Mail. No active development.
Environmental requirements	Linux only.
Practical Features of Fuzzer	**Possible Advantage**
Automation Support	Yes.
Target Monitoring	No.
Customization	Yes.
Other features	Very basic fuzzer but extensible.
Fuzzing Intelligence	**Possible Advantage**
Test data generation options	All three approaches available i.e. Random, mutation, data-modeling.
Fuzzing heuristics utilization	Yes. Both smart and random mutaions are included.
Data modeling complexity	Moderate. Requires good knowledge of C language and its associated libraries.

Table 3.16: Evaluation of Spike Fuzzer

which the data or packets are transmitted in human readable form (ASCII characters). Examples of text-based protocols are FTP ,SMTP and HTTP. Perhaps due to same reason Peach has its commercial version of fuzzer which lets one to fuzz binary as well as plain-text protocols.

Due to the same mutual drawback no fuzzing tool is used to fuzz the target protocol i.e. SNMP with these tools and had to adopt manual packet mutations for testing the target protocol (SNMP). Details for which can be found in subsequent chapters.

3.7 Fuzzing Limitations

Fuzzing has limitation in types of vulnerabilities it can find. Following are the vulnerabilities which typically go undiscovered by a fuzzer.

Access Control Flaws: Some applications have privilege layers to support multi accounts. Consider a software which has read only rights for normal user while read/write rights for administrator. While fuzzing such software there is no way for the fuzzer to detect that at some point during fuzzing it has got administrator privileges due to access control flaw. Because fuzzers are not aware of the logic of the software.

Implementing logic-aware functionality into the fuzzer is plausible.

Poor Design Logic: Fuzzers are also not the best tools for identifying poor design logic. Consider a software that has a database to store users information. If there exist no authentication mechanism for accessing the database, this is a security flaw by software design logic. There is no way for the fuzzers to detect such design logic flaws.

Back-doors: For a fuzzer with limited or no knowledge of target application logic, a back-door is seen no different. While fuzzing, there is no way for the fuzzer to identify if there is backdoor embed in the target software like hard-coded passwords, any data ex-filtration mechanism or others.

But if one think the other way, Fuzzing is also famous for uncovering the zero-days. It

is efficient in finding DoS, buffer overflow, integer overflow, coding flaws which might have been made a part of an application intentionally to be exploited at later stage. It can be concluded that due to ability of fuzzing for finding only certain flaws, the technique might not be able to explore some specific back-doors but not all.

Multistage Vulnerabilities Exploitation is not always as simple as attacking a single weakness. Fuzzing might be useful for identifying the individual flaws but not generally be valuable for chaining together a series of minor vulnerabilities to identify multi-vector attack.

3.8 Conclusion

In light of advantages and limitations offered by fuzzing, it can be concluded that the said technique can be fruitful for automated discovery of unknown flaws in network protocols. But, based on the evaluation of open-source network protocol fuzzing tools performed in this chapter, they lack several features for fuzzing a complex target. Improvements in the existing tools are still required to make them feasible utility for researchers to fuzz test network protocol target.

Chapter 4

SNMP Vulnerabilities Exploitation through Fuzzing

4.1 Introduction

This chapter is all about SNMP and its vulnerabilities. At first SNMP protocol is comprehensively explained covering all three versions, SNMP architecture, its operational model and components, internals of SNMP packets etc. Later in the chapter SNMP is analyzed from security point of view to uncover its by design vulnerabilities and potentially weak areas in SNMP packets. Cisco, Juniper, Huawei products specific SNMP vulnerabilities are also discussed and their feasibility to be explored through fuzzing is also inspected.

The chapter is of great importance as it will help to design test cases for fuzzing SNMP and also the performed analysis will aid to narrow down the scope of exploring the vulnerabilities which are exploitable through fuzzing.

4.2 Simple Network Management Protocol (SNMP)

Modern networks are larger in size, robust in processing and more advance than their predecessors. Networks become more complex and difficult to manage as they expand,

speed up and enhance. To manage such huge and complicated network infrastructure there is an immense need for decent network management technologies which may enable centralized, robust and reliable management and monitoring of the network. Simple Network Management Protocol (SNMP) is an Internet Standard protocol which allows to organize and gather information from managed devices on TCP/IP networks and modification of that information to remotely change device behavior. SNMP operates at application layer of OSI model and takes advantage of UDP port 161 and 162 (trap messages) for transmission and reception of SNMP messages. Three major versions of SNMP have been developed and deployed so far with SNMPv1 being the original version of protocol. Latest versions include SNMPv2c (c stands for "community") and SNMPv3 with improvements in features, performance and security.

4.2.1 SNMP Versions

SNMP version developed in 1988 is the "Old Faithful" SNMP version which was widely adopted. Despite several revisions introduced to the initial standards and additional MIB modules incorporated over time, but for several years the base technology remained unchanged.

As with the passage of time and advancement in technology and growing attack vector of cyber threats, SNMPv1 was labelled as insecure by its users. By design "trivial" (as RFC 3410 defines) authentication scheme used by SNMP version 1 called as a "community string" [34]. Community string is a field in a SNMP packet which enables the agent or manager to authenticate if the request is from valid user or not. Another drawback which makes SNMPv1 more vulnerable is transmission of packets involving no encryption. An insider may obtain community string by sniffing the packets and can later impersonate as administrator which may lead to retrieval of sensitive information and configuration modification of managed network devices.

In-order to add more features and to overcome above mentioned security issues SNMP version 2 was published in April 1993 with the idea of party-based security, documented in RFCS 1441 through 1452. This feedback and confusion lead to the advent of vari-

ants of SNMPv2. There came SNMPv1.5, SNMPv2c (Community-Based SNMPv2), SNMPv2u (User-Based SNMPv2) but none of the variant was universally adopted.

To return the universality of SNMP and to resolve the outstanding issues a work began in 1996. SNMP version 3 was introduced in 1998, incorporated with significant enhancements in previous versions of SNMP and received universal acceptance.

The latest version of SNMP is SNMPv3 which is still under active revisions. The dominant change in SNMPv3 is a more organized and robust way of handling various security approaches to SNMP. SNMPv3 adopts SNMPv2 PDU (Protocol Data Unit) message format and protocol operations. The protocol is enriched with distinguished security methods such as encryption and authentication.

4.2.2 Basic Terms and Components of SNMP

There are few basic terminologies which must be explained in-order to grasp operational working of SNMP.

Managed Nodes: Regular or conventional TCP/IP devices which can be managed through SNMP.

SNMP Agent: A piece of software resides on managed devices for implementing SNMP. It allows to share manageable information with NMS and receive instructions from it.

Network Management Station (NMS): A dedicated node on a network equipped with particular software which permits to manage the SNMP enabled entities.

SNMP Manager: It is a software implementation of SNMP protocol, which enables the NMS to instruct and obtain information from SNMP enabled agents.

SNMP Manager, SNMP Agent, SNMP Application, MIB comes under a single umbrella and are called as SNMP entities. Figure 4.1 illustrates a simple SNMP operational

Model.

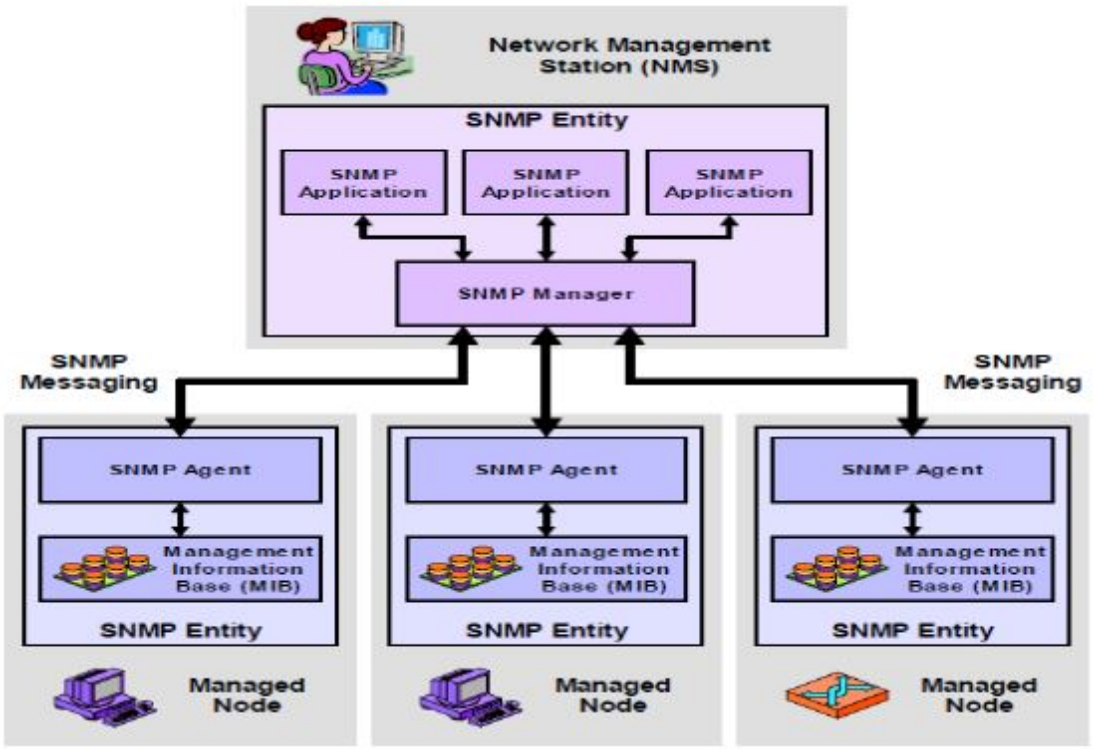

Figure 4.1: SNMP Operation Model [31]

4.2.3 SNMP Messages and Protocol Data Units (PDUs)

Like any other network protocol, the information flow in SNMP protocol is carried out by exchanging SNMP packets. The term SNMP PDUs is also referred for SNMP packets because it uses UDP.

Six different PDUs were initially defined for SNMPv1. The introduction of new features and changes in SNMPv2 and SNMPv3 expanded the count of PDUs. Table 4.2 shows the main SNMP PDUs with respect to their versions.

GetRequest: A request made by manager to an agent for obtaining value of variable or variables list.

SetRequest: A request by manager to change variable value or list of variables in an agent.

GetNextRequest: A request by manager to find out the available variable and their values.

GetBulkRequest: A manager initiated request for various iterations of GetNextRequest.

Response: Manager initiated requests such as GetRequest, GetNextRequest, SetRequest, InformRequest and GetBulkRequest are entertained by agent through response messages. For error reporting, error-index and error-status fields are used.

Trap: Asynchronous alert by an agent to a manager.

InformRequest: Acknowledged asynchronous notification.

SNMP PDU Classes	Description	SNMPv1 PDUs	SNMPv2/ SNMPv3 PDUS	Port
Read	To perform management information retrieval	GetRequest-PDU, Get-Next-Request-PDU	Get-Request-PDU, Get-Next-Request-PDU, Get-Bulk-Request-PDU	UDP (161)
Write	To modify management informa-tion(effects device‘s operation)	SetRequest-PDU	Get-Request-PDU, Get-Next-Request-PDU, Get-Bulk-Request-PDU	UDP (161)
Response	Packet sent in response of re-quest	GetResponse-PDU	Response-PDU	UDP
Notification	Packet sent to SNMP manager containing interrupt-like alert	Trap-PDU	Trapv2-PDU, InformRequest-PDU	UDP (162)

Table 4.2: SNMP PDU types and Classes

4.2.4 SNMP Architecture

In an SNMP architecture, there are two prominent entities (Manager and Agent) and two channels for communication (UDP port 161 and 162). The following figure shows communication flow between a Manager and an Agent and the ports they use depending on type of PDUs.

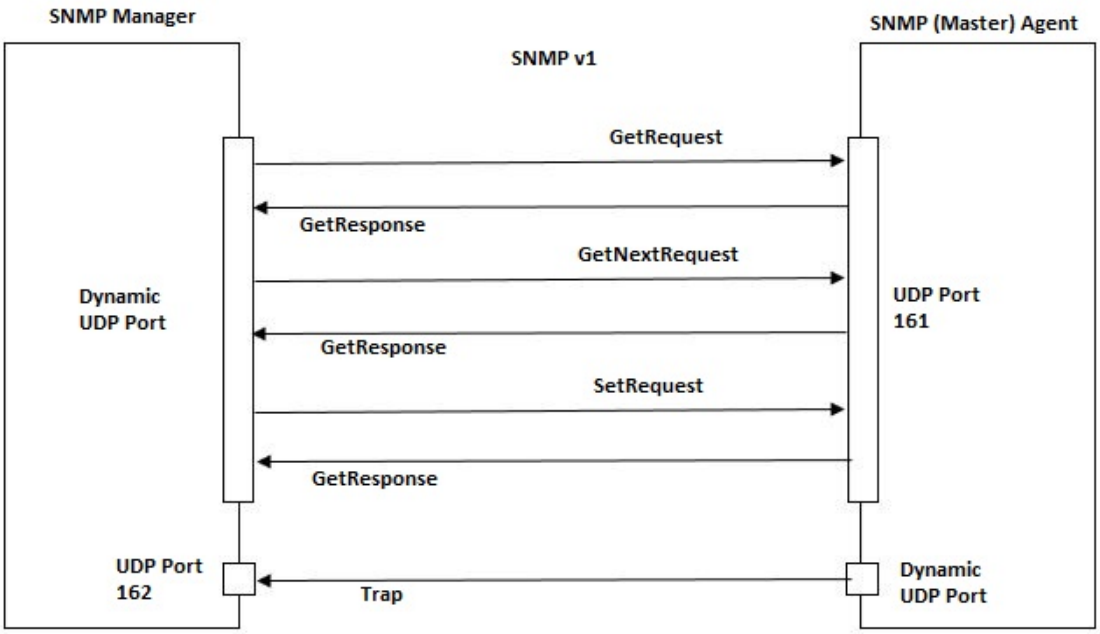

Figure 4.2: Simplified SNMP architecture

4.2.5 General SNMP Message Format

An SNMP packet has two major fields, SNMP message and SNMP PDU. It is important to know the difference between these interchangeably used terms because these are not the same. SNMP PDU is contained within an SNMP message which also has message header. The following figure illustrate general SNMPv1 message format.

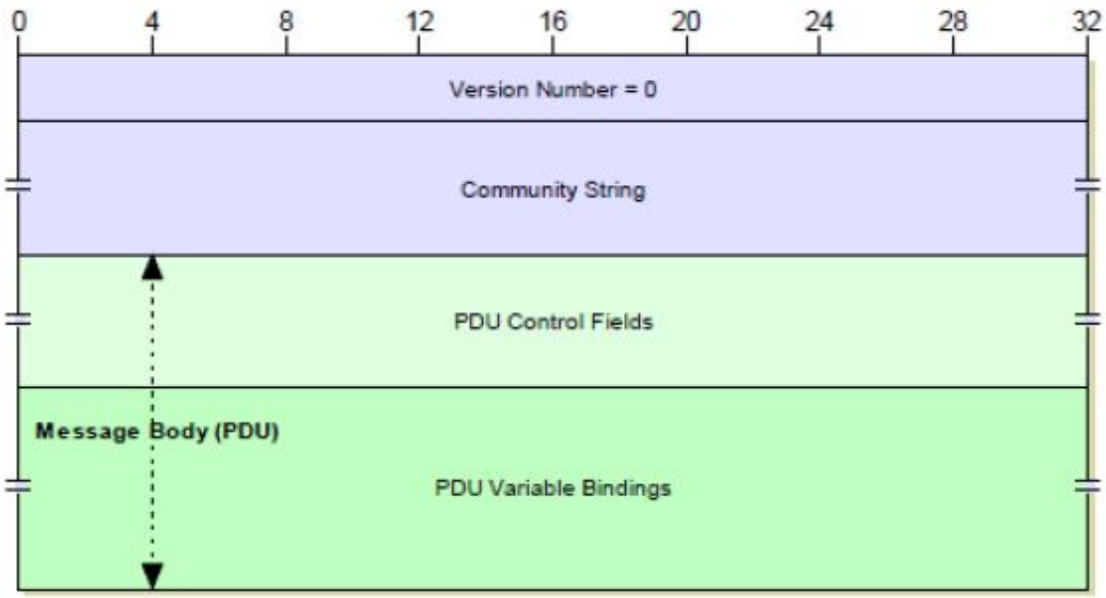

Figure 4.3: SNMP Message Format [31]

Version number identifies the SNMP version while community string field in the message header is used for authentication.

PDU Control Fields: Communication between two SNMP enable entities is done with PDU Control Fields. It also describe SNMP PDU.

PDU Variable Bindings: Variable bindings field in SNMP PDU describes MIB object.

All the fields, except SNMP version number field which is 32-bit fixed value, are variable in size. Version number=0 in the above message indicates SNMPv1. Number of fields and their sizes may vary from version to version due to more features and added security.

4.3 Factors Behind Choosing SNMP as Target Protocol

As discussed in the earlier chapters that our target for fuzz testing a network protocol in routers is SNMP with all three versions. There are certain factors for which the protocol has been chosen as our test target which are discussed below.

SNMP is renown protocol for managing individual and network devices in IP networks. Due to this fact, there are various implementations and abundant of installation across different vendor products of it. Although, if SNMP is not enabled inside a network, there is fair chance of its presence in individual devices operating on default configurations.

SNMP enabled devices can be of greater concern for network infrastructure. Because a simple DoS attack against these devices may have serious threats to network availability. Therefore, uncovering and fixing flaws in protocol implementation should be taken seriously.

Flaws in decoding unexpected BER encoding fields can be critical because they may lead to conditions where access control mechanism such as authentication becomes

useless. Since, parsing of the packets takes place before authentication.

4.4 SNMP vulnerabilities by Design

Apart from product or vendor specific vulnerabilities which will be discussed later, there exist some by design vulnerabilities of SNMP which are as under.

SNMP uses UDP on port 161 which is connectionless protocol. No prior setup is required by SNMP agents and trap-aware NMS to accept incoming requests and traps. This makes it a feasible target for packet injection attacks [32].

For first two versions i.e. SNMPv1 and SNMPv2c, community string which is used for authentication is being sent in plain-text with each SNMP packet.

Connectionless communication makes it vulnerable to brute force attack. An attacker can keep guessing the community string by spoofing the Ip-address.

The above vulnerabilities favor impersonation attack: An attacker may act as NMS and can retrieve sensitive information from Agent devices.

ASN.1 (Abstract Syntax Notation 1) with BER (Basic Encoding Rule) is the required encoding scheme for SNMP PDUs, which also has some vulnerable points in it (discussed later in detail).

4.5 Analysis of Vulnerable fields in SNMP

The below mentioned vulnerable areas in SNMP are analyzed manually and empirically. Fuzzing is not able to find certain vulnerabilities, such as poor design logic, access control flaws, backdoors and multistage vulnerabilities [3]. For this reason, we just consider the vulnerabilities that Fuzzing can find. In accordance with the analysis on historical vulnerabilities and the SNMP protocol, there are five types of vulnerable points on SNMP.

ASN.1 BER parse: ASN.1 with BER is the required scheme by SNMP for encoding its PDUs. In BER rules, possible vulnerable points are invalid encodings, including invalid types, abnormal lengths and malformed values [33]. An invalid type/length/value encoding means replacing right encodings with malformed encodings. For example, the encoding of Integer is 0x02, we can replace it with 0x04(OCET String) or 0x05(NULL).

Integer overflow: Integer overflow is caused by malformed Integer values including boundary value, large Integer number and other values than Integer, besides, the transformation from signed Integer value to unsigned may lead to anomaly. E.g. large Integer number 256+1 or (-256)-1 is likely to cause Integer overflow. Although Integer overflow never happened on SNMP before, it is indispensable for Fuzzing test.

Buffer overflow: Buffer overflow is caused by incorrect input strings, which include long character strings and format character strings. For SNMP, vulnerable points about buffer overflow could be the field Variable-Bindings according to historical statistics. So, we can test this field with zero-length Object Identifier (OID), overlong single or multiple format character strings, overlong OIDs with many branches.

Empty UDP packets: Empty packets include empty UDP packets, empty IP packets, and empty SNMP packets. The data of above packets can be tested with 0x00. There are a lot of vulnerabilities caused by empty packets. Such as CVE-2001-0566, its root cause is to send an empty packet to port 161(SNMP).

A large number of packets: Sending abundant packets to routers could allow attackers to launch denial of service or gain privileges, such as CVE-2002-0012 and CVE-2002-0013. This is a significant cause for denial of service attack. Hence, we should test routers with a large number of all kinds of SNMP packets.

4.6 Products Specific SNMP Vulnerabilities

The following tables will list down the SNMP vulnerabilities specific to Cisco, Juniper and Huawei products and their possibility of exploitation through fuzzing. The CVE (Common Vulnerability Exposure) data has been take from NVD's (National Vulnerability Database) official website [2].

Note: Comments and exploitability chances of vulnerabilities listed in the below table are based upon the knowledge acquired about Fuzzing during this thesis work. Not all but few vulnerabilities of Cisco products are exploited practically (results and methodology can be found in subsequent chapter).

Cisco Products Specific SNMP Vulnerabilities Analysis

No.	CVE ID	Vulnerability Type	Score	Exploitability through Fuzzing	Comments
01	CVE-2018-0329	+Info	5.0	Yes. By fuzzing the community string field only.	
03	CVE-2018-0161	DoS	6.3	Yes. By fuzzing OID (Object Identifier Field)	
04	CVE-2018-0160	DoS	6.3	Yes, but not recommended.	Credentials required.
05	CVE-2017-12278	DoS	5.2	Yes.	Credentials required.
06	CVE-2017-12211	Mild DoS	6.3	Yes.	By Polling IPv6 info.
07	CVE-2017-6783	+Info	4.0	No.	
08	CVE-2017-6744 to 6736	Execution code overflow	9.0	Yes. Fuzzing is good at finding buffer overflow vulnerabilities	Credentials required
09	CVE-2017-6615	DoS	6.3	Maybe. By fuzzing OID field	Under certain conditions.
10	CVE-2017-3820	DoS	6.8	Maybe	Not enough information available

11	CVE-2016-6366	Execution Code overflow	8.5	Yes. Fuzzing is good at finding buffer overflow vulnerabilities	
12	CVE-2016-1473	+Info	10.0	No. Fuzzing cannot identify hardcoded passwords/strings	
13	CVE-2016-1452	Integrity	6.4	No.	
14	CVE-2016-1432	Dos	6.8	No.	
15	CVE-2016-1428	Dos	6.8	Yes.	
16	CVE-2016-1333	Dos	6.8	Yes. By Fuzzing OID field for unspecified Bridge MIBs	
17	CVE-2015-6308	Dos	4.0	Yes. By Fuzzing OID field that doesnâĂŹt exist	
18	CVE-2015-6260	Dos	7.8	Yes	
19	CVE-2015-4238	DoS	6.8	Yes. By performing stress testing through fuzzing	
20	CVE-2015-4204	DoS	6.8	No.	
21	CVE-2015-0687	DoS	6.3	Yes.	
22	CVE-2015-0686	DoS	6.3	No.	
23	CVE-2015-0661	DoS	4.0	Yes.	
24	CVE-2015-0617	DoS	5.0	Yes.	
25	CVE-2014-3377	DoS	4.0	Yes.	

26	CVE-2014-3341	+Info	5.0	No.	
27	CVE-2014-3269	DoS	6.8	Yes. By frequent fuzzing	
28	CVE-2014-2103	DoS	6.8	Maybe.	Not Enough information available.
29	CVE-2013-6700	DoS	5.0	Yes. By Fuzzing OID field for unspecified MIB.	
30	CVE-2013-1217	DoS overflow	6.8	Yes. By frequent Fuzzing	
31	CVE-2013-1216	DoS +Info	4.0	Yes.	
32	CVE-2013-1204	DoS	5.0	Yes. By sending UDP on port 162	
33	CVE-2013-1180	Execution Code overflow	9.0	Yes. Only overflow can be caused through Fuzzing.	
34	CVE-2013-1179	Execution Code overflow	9.0	Yes. Only overflow can be caused through Fuzzing	
35	CVE-2013-1105	Auth. Bypass	9.0	No.	
36	CVE-2012-5030	DoS	6.8	Yes. Can be exploited through legit input.	
37	CVE-2012-1365	DoS	4.0	Yes.	
38	CVE-2011-4023	DoS	7.8	Maybe.	
39	CVE-2010-2982	+Info	7.1	No.	
40	CVE-2010-2976	+Info	10.0	No.	

41	CVE-2010-1574	+Info	10.0	No.	
42	CVE-2009-0625	DoS	7.8	Yes.	
43	CVE-2009-0624	DoS	6.8	Yes.	
44	CVE-2008-3807	+Info	9.3	No.	
45	CVE-2008-1746	DoS	7.8	Yes.	
46	CVE-2007-2036	+Info	10.0	No.	Hardcoded strings are not detected by Fuzzing
47	CVE-2007-1257	Code Execution	10.0	No.	
48	CVE-2007-0967	DoS	7.8	Maybe.	Not enough information available.
49	CVE-2006-4950	+Info	10.0	No.	
50	CVE-2005-3803	+Info	5.0	No.	Hardcoded strings are not detected by Fuzzing
51	CVE-2005-0612	DoS	7.5	No.	Hardcoded strings are not detected by Fuzzing
52	CVE-2004-1434	DoS	5.0	Yes.	
53	CVE-2004-0714	DoS Memory Corruption	5.0	Yes.	
54	CVE-2003-1003	DoS	7.8	Yes.	
55	CVE-2003-1002	DoS	5.0	Yes.	

56	CVE-2002-1555	+Info	6.8	No.	Hardcoded strings are not detected by Fuzzing.
57	CVE-2002-0013 CVE-2002-0012	DoS	5.0	Yes.	
58	CVE-2001-0711	DoS	5.0	No.	
59	CVE-2001-0566	DoS	5.0	Yes	
60	CVE-2000-0955	+Priv	7.5	No.	

Table 4.3: Cisco Products Specific SNMP Vulnerabilities

The above listed vulnerabilities are across different Cisco products such as Routers, Switches, IP-Phones, Firewalls and ASAs. There is no software available which can emulate all these devices. Thus, it is impossible to test all of them in scope of fuzzing. The comments and probability of exploitation remarks for these vulnerabilities are made keeping in view the ability and limitations of fuzzing. Not all, but few of the above vulnerabilities have been exposed through fuzzing in chapter 5.

Juniper Products Specific SNMP Vulnerabilities Analysis

No.	**CVE ID**	**Vulnerability Type**	**Score**	**Exploitability through Fuzzing**	**Comments**
01	CVE-2018-0019	DoS	4.3	Maybe.	Not enough information available
02	CVE-2017-10611	DoS	4.3	No.	Not enough information available.

03	CVE-2017-2345	DoS Code Execution	7.5	No.	Not enough information available.
04	CVE-2009-3487	XSS	3.5	Yes.	Cross-site Scripting vulnerabilities can be spotted by Fuzzing
05	CVE-2008-0960	Bypass	10.0	No.	Fuzzing in no way can detect authentication bypass.

Table 4.4: Juniper Products Specific SNMP Vulnerabilities

Huawei Products Specific SNMP Vulnerabilities Analysis

No.	**CVE ID**	**Vulnerability Type**	**Score**	**Exploitability through Fuzzing**	**Comments**
01	CVE-2013-4613	DoS overflow	7.8	Yes	
02	CVE-2012-3268	+Info	8.5	No.	

Table 4.5: Huawei Products Specific SNMP Vulnerabilities

In comparison to Cisco, Juniper and Huawei products specific SNMP vulnerabilities are quite few in number. The reason being Cisco is the first and leading network device manufacturer in market. Also, their products are extensively deployed under different environmental conditions which result in exposure of these vulnerabilities. For the same reason and due to availability of Cisco emulator we chose Cisco as our target product for fuzzing.

4.7 Conclusion

The significance and extensive usage of SNMP protocol in digital network infrastructure makes it one of the most sought after target by attackers. To make it zero-day prone, fuzzing can be utilized to discover unknown flaws by targeting vulnerable fields in SNMP protocol as identified in this chapter. Our analysis of existing SNMP vulnerabilities exploitation in scope of fuzzing for renown vendor products shows that these vulnerabilities could have been exposed prior to deployment if careful fuzz testing had been performed.

Chapter 5

Experimental Results

5.1 Introduction

This chapter initially discusses the methodology adopted to carry out fuzz testing of all three versions of SNMP. All the steps involved in fuzz testing are briefly explained with the tools and their usage. Afterwards, experimental setup and the step-by-step procedure of testing process with visual graphics is demonstrated. Later part of this chapter consist of the results and vulnerabilities exposed during this whole process of testing.

5.2 Methodology of Experiments

The methodology adopted to perform fuzz testing of SNMP protocol in Cisco router is straight forward. The target router is first configured with SNMP protocol and then fed with the malformed test cases, generated by mutating the vulnerable fields in SNMP packets. The target is monitored in parallel for any exceptions. In case of an exception, the test-case resulted in anomalous behavior marked as malicious and further analysis is carried out with the help of attached debugger. If the target device crash, router is rebooted manually. The technical explanation of each step-followed to perform testing is elaborated in later sections along with the tools used.

5.3 Flowchart

Below is the flowchart of the methodology followed to perform fuzz testing of SNMP in Cisco router.

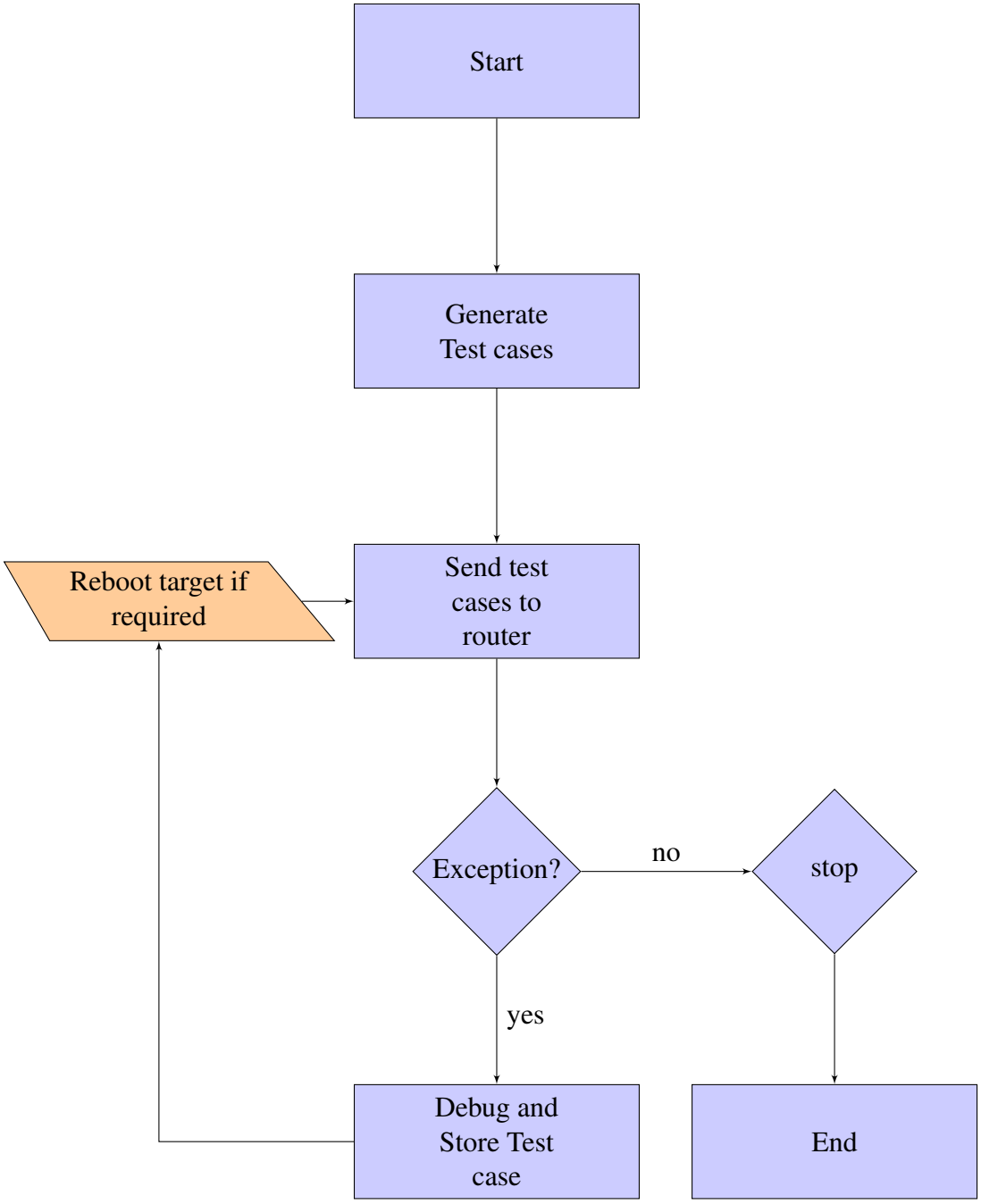

Figure 5.1: Methodology flow chart

5.3.1 Generation of Test-Cases

As discussed in chapter 4, SNMP is a complex protocol as compared to other plain-text protocols such as FTP, SMTP and TFTP. SNMP uses ASN.1 Basic Encoding Rules(BER).The available open-source fuzzing tools are not equipped with BER encoder and decoder thus, prevent SNMP fuzzing. Due to this drawback mutation based fuzzing is adopted i.e first capturing handful of legit SNMP messages and then mutating the vulnerable fields identified in section 4.5.

5.3.2 Test-Cases Injection

Following the generation of test-cases the next step is injection of test case to the target. For this purpose an existing UDP injector tool is utilized, which was developed by researchers at University of OULU in Finland while testing SNMPv1 implementation across different products [35]. The tool is fully automated i.e. it reads the malformed test cases from database and sends them to specified ip address.

Rest of the steps pictured in flow chart can be well understood by going through implementation process in subsequent sections.

5.4 Tools and Utilities

Wireshark: Wireshark is utilized for capturing SNMP traffic and extracting application layer data from sniffed packets.

Dynamips: Dynamips is an emulator computer program that was written by Christophe Fillot to emulate Cisco routers. It can run on FreeBSD, Linux, Mac OS X or Windows. Dynamips emulates Cisco routing series hardware which allows booting of actual Cisco IOS. Cisco router platforms that can be emulated with Dynamips are 1700, 2600, 2691, 3600, 3725, 3745 and 7200.

A modified version of Dynamips proposed in [4] is used which allows to attach external

debugger such as GDB or IDA-Pro.

SNMP MIB Browser: SNMP MIB Browser is specifically utilized as SNMP server i.e. generates snmp queries to SNMP agent in Cisco router. Those queries and their replies from agent router were sniffed with wireshark which aided in collection of legitimate SNMP packets.

Hex Editor: Hex editor is a binary editor which allows editing of byte level data. Mutations in SNMP packets is performed using this tool.

Protos UDP Packet Injector: The packet injection is performed with Protos tool. It was developed by Oulu University Secure Programming Group (OUSPG), Finland during security testing of different applications and protocols. It is a command line utility with different switches available to perform extended function. The tool is developed in java.

Scapy: Scapy is an open-source Python based tool, developed for manipulating data packets. It can be used to craft and decode packets, transmit, capture, and match replies and requests.

5.5 Experimental Setup

The fuzzing experiments are carried out on Intel Core-i5 machine with Windows 10 as host operating system. Ubuntu 16.04 LTS and Kali Linux (2017 release) are installed on virtual machines, created with VMware workstation. Host and the guest machines are connected via virtual switch in VMware, having same network address.

Wireshark, SNMP MIB Browser, Hex Editor and Protos UDP Injector are installed in Windows 10 while modified version of Dynamips in Ubuntu 16.04 LTS to emulate our target router. The purpose of Kali Linux is to utilize built-in version of Scapy in it. The topology of experimental setup is shown in below figure.

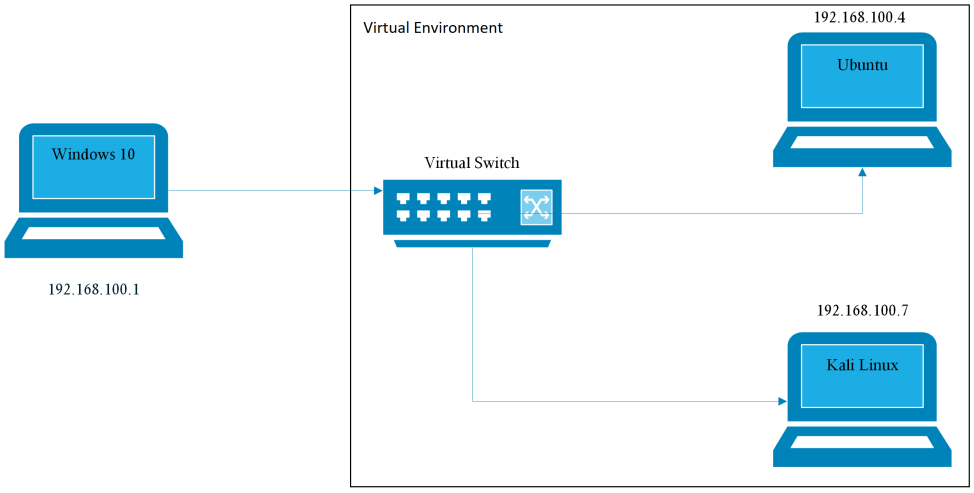

Figure 5.2: Topology of Experimental Setup

5.6 Fuzzing SNMP in Cisco Router

This section dives into the technical details on how fuzz testing of SNMP implementation in Cisco router is performed.

5.6.1 Capturing SNMP Traffic and Test-Cases Generation

As mentioned earlier that mutation based fuzz testing approach is adopted to fuzz the target, for this legitimate SNMP traffic is required. All types of SNMP PDUs exchanged between SNMP MIB Browser(acting as SNMP server) and SNMP agent(configured in Cisco 2600 series router) are first captured with wireshark as shown in below figure.

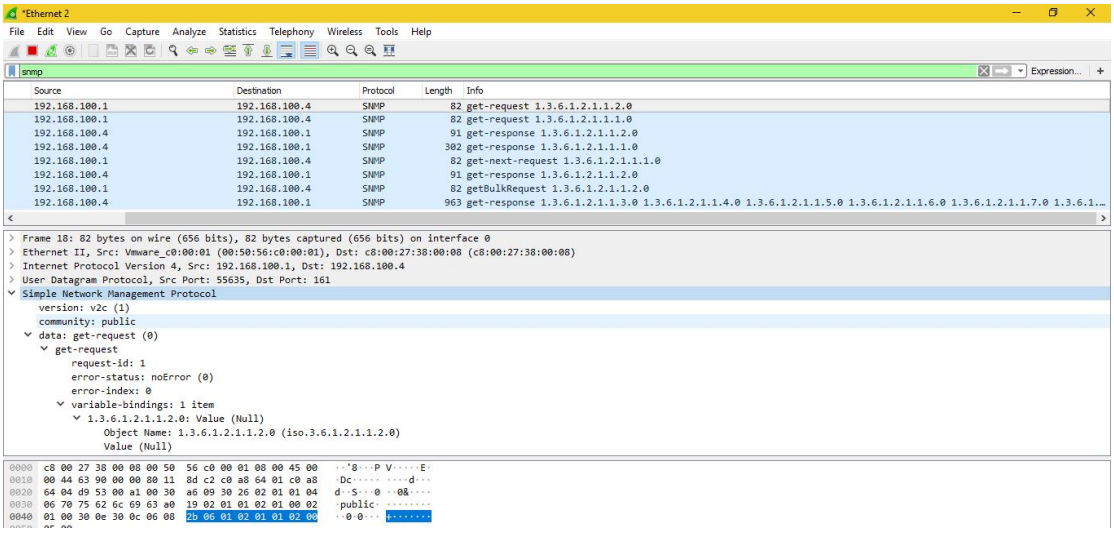

Figure 5.3: SNMP traffic capture with wireshark

The above snapshot shows capture of SNMP version 2c PDUs. Same procedure is followed for capturing SNMP-v1 and SNMP-v3 traffic.

After traffic capture, application layer data i.e. SNMP bytes are extracted from captured packets of each type. Wireshark offers this feature of extracting the data for any of the layers. The extracted data packets are saved as binary file i.e. in **.bin** format. Binary data packets are then opened with Hex editor to perform desired byte level mutations.

In order to perform intelligent mutation potential vulnerable fields analyzed in section 4.5 are targeted. For identification of the same vulnerable fields in binary data, ASN.1 BER chart (shown in below Table) along with byte level highlighting feature of wireshark is used.

Primitive ASN.1 Types	**Identifier in Hex**
Integer	02
Bit String	03
Octet String	04
Null	05
Object Identifier	06
Constructed ASN.1 Type	**Identifier in Hex**
Sequence	30
Primitive SNMP Application Types	**Identifier in Hex**
Ip Address	40
Counter (Counter 32 in SNMPv2)	41
Guage (Guage 32 in SNMPv2)	42
Time Ticks	43
Opaque	44
NaapAddress	45
Counter64 (only in SNMPv2)	46
Uinteger32 (only in SNMPv2)	47
Context-Specific Types within an SNMP Message	**Identifier in Hex**
GetRequest-PDU	A0
GetNextRequest-PDU	A1
GetResponse-PDU	A2
SetRequest-PDU	A3
Trap-PDU	A4
GetBulkRequest-PDU	A5
InformRequest-PDU	A6
SNMPv2-Trap-PDU	A7

Table 5.2: ASN.1 with BER

An example of SNMP Get-Request PDU with fields highlighted by referring the above chart is shown in below figure.

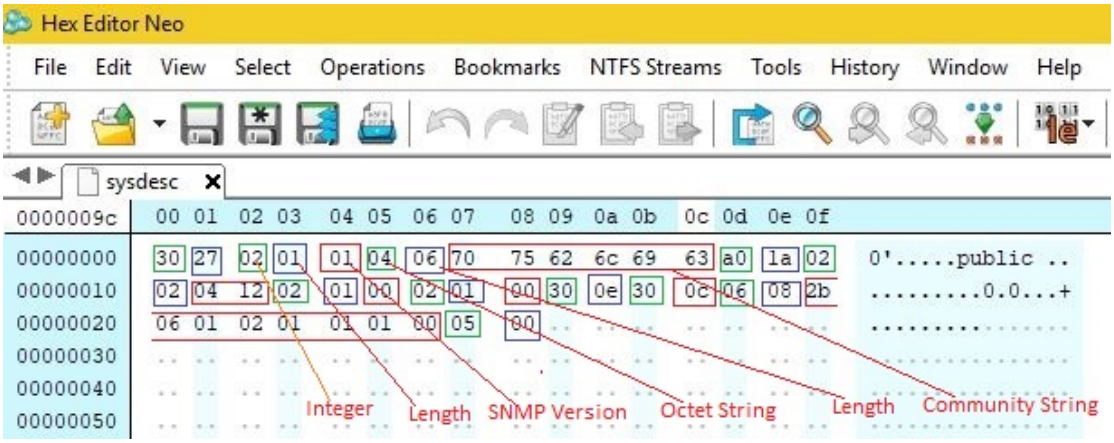

Figure 5.4: SNMP PDU encoding and packet fields identified

It can be clearly seen in above figure that every SNMP field has some encoding byte before it. These encoding bytes are as per ASN.1 Basic Encoding Rules(shown in Figure 5.5). Every field is represented in TLV (Type, Length, Value) format.

5.6.2 Target Prepration

Before starting actual fuzzing process, it is to be made sure that the target is up and debugger is successfully attached. The below figure shows that the target is up, gdb server is listening on port:4321 and thread is activated.

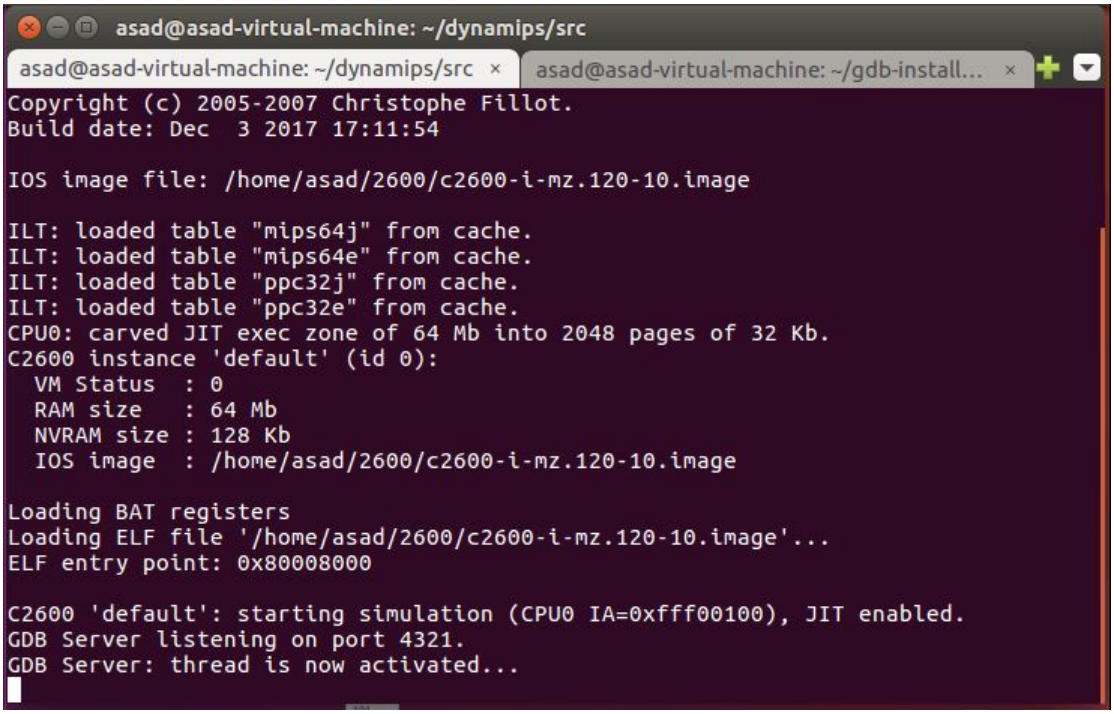

Figure 5.5: GDB server thread activated

5.6.3 Injection of Test-Cases to Target

Once the malformed test cases are generated and target is ready, the next step towards fuzzing is injection of test-cases.For feeding malformed packets, Protos UDP Packet Injection tool is used. The below figure shows injection of packets through Protos tool.

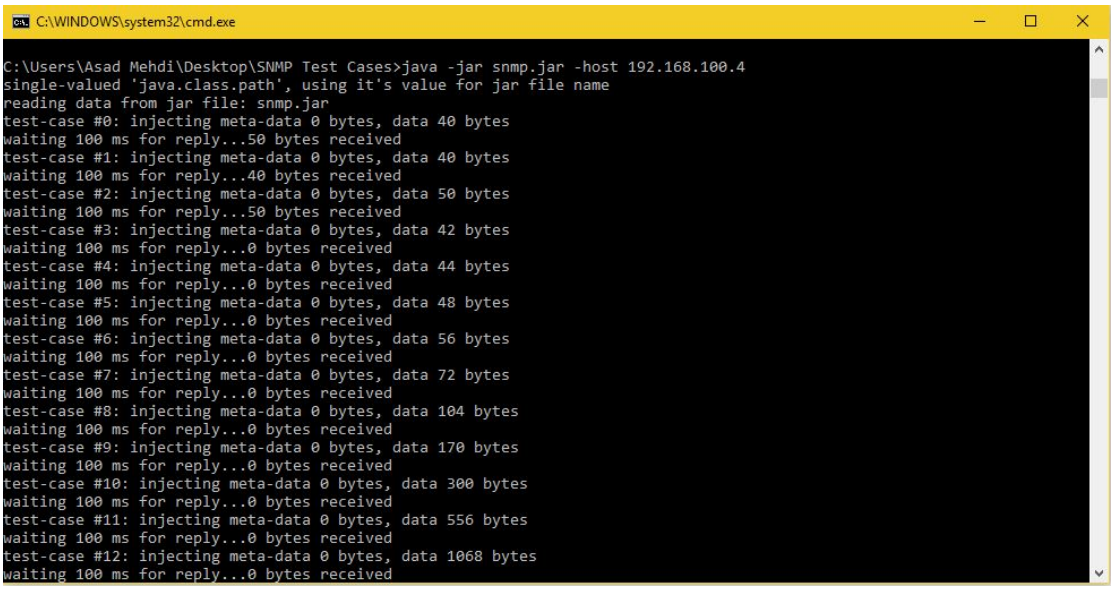

Figure 5.6: Test Cases injection with Protos UDP Injector

5.6.4 Target Monitoring

The System Under Test must be monitored in parallel with fuzzing process to identify any anomalous behavior or a device crash. One way to check if the target still alive is to continuously ping the target. To verify the proper working of the target protocol, Protos has the feature of sending zero-test-case, which can be thought of as an SNMP ping.

5.7 Testing Results

The procedure described in previous section is adopted to carry out fuzzing of all three versions of SNMP in cisco routers. There are almost 60 vulnerabilities reported to the date for SNMP across all Cisco devices. Due to software and hardware limitations it is not possible to test all these vulnerabilities in scope of fuzzing. The target router details are mentioned in below table.

Target	Cisco Router
IOS version	12.10 & 15.2
Platforms	C2600 & C7200
Tested Protocol	SNMP
Adapter	tap device
GDB Port	4321

Table 5.4: Target Details

All experiments were carried out on Cisco C2600 and C7200 series routers by emulating these platforms in Dynamips. Tap device is used to bind router interface with Ubuntu's ethernet interface.

5.7.1 Memory Corruption Vulnerability

Memory corruption refers to type of vulnerability in computer system when its memory is altered without explicit assignment. Programming errors result in modification of memory contents which enables arbitrary code execution. Memory corruption vulnera-

bilities are severe in nature because they may result in program crash causing denial of service or complete compromise of a device through remote code execution.

During the fuzzing experiments Protos test cases for SNMP-v1 are used. The same test cases are referred to test SNMP-V2c. Because, the packet structure of SNMP-V1 and V2c is similar except some additional PDUs , same vulnerabilities were reproduced as in SNMP-v1 testing on Cisco router. The following figures show test cases which caused memory corruption during SNMP-V1 and SNMP-V2c testing.

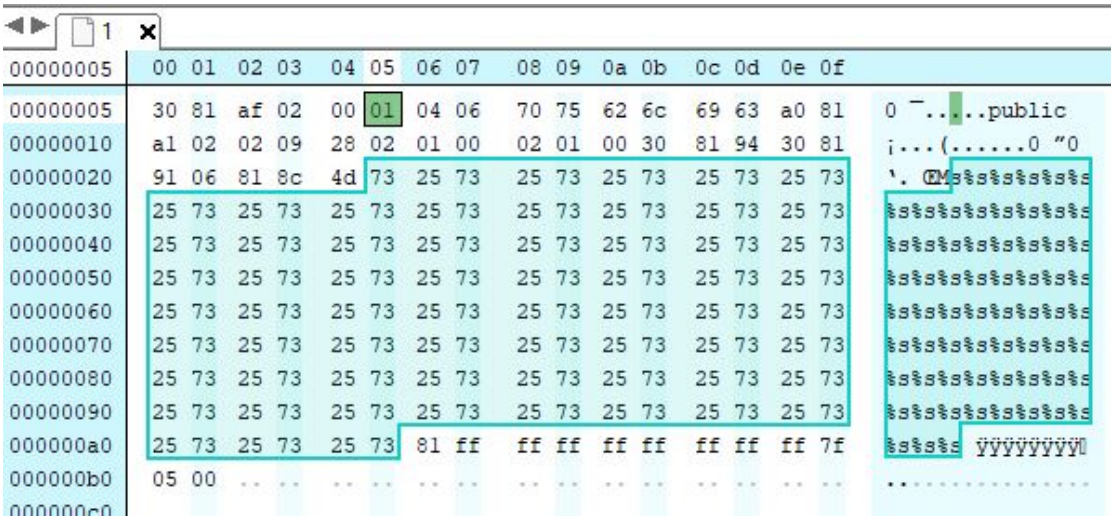

Figure 5.7: Test-case-1 caused memory corruption

The packet is of total 178 bytes out of which 129 bytes field (highlighted) is the fuzzed OID field of SNMP version-2 packet.

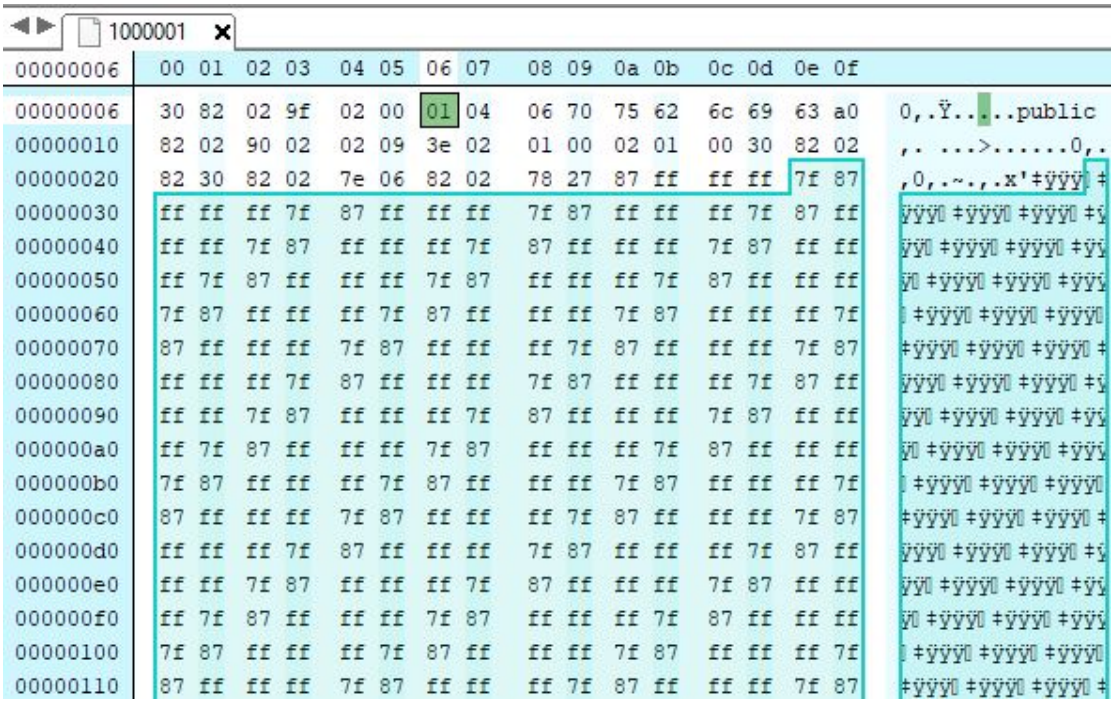

Figure 5.8: Test-case-2 caused memory corruption

Packet size 675 bytes with 629 bytes of fuzzed OID field.

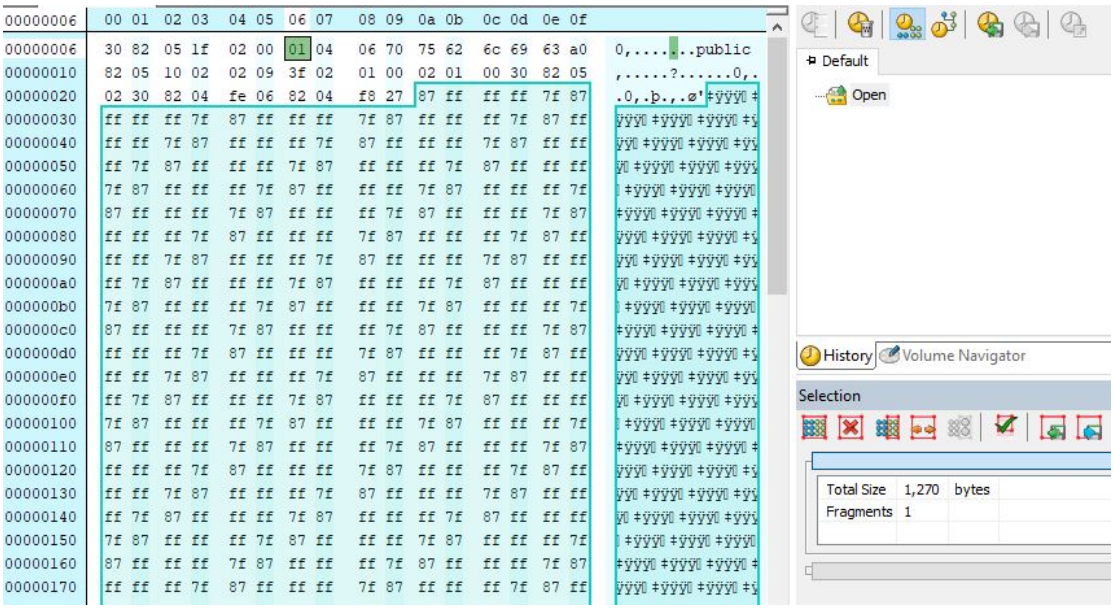

Figure 5.9: Test-case-3 caused memory corruption

Packet size of 1315 bytes with 1270 bytes of fuzzed OID field.

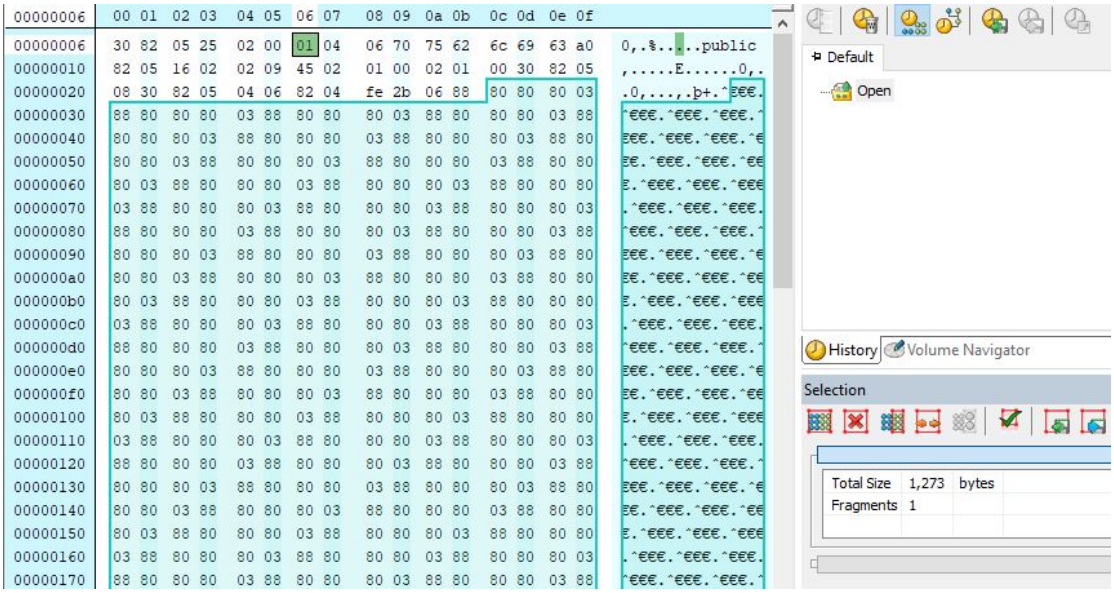

Figure 5.10: Test-case-4 caused memory corruption

Packet size of 1321 bytes with 1273 bytes of fuzzed OID field.

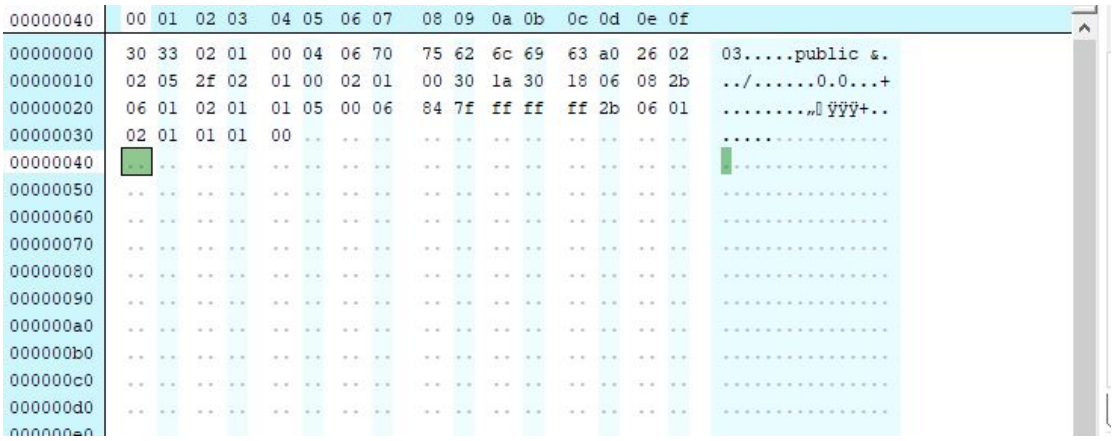

Figure 5.11: Test-case-5 caused memory corruption

The above test case is formed by manipulating the encoding bits which, when injected, also produced router crash. It can be inferred from the results that memory corruption can be caused by fuzzing SNMP fields as well as encoding bits.

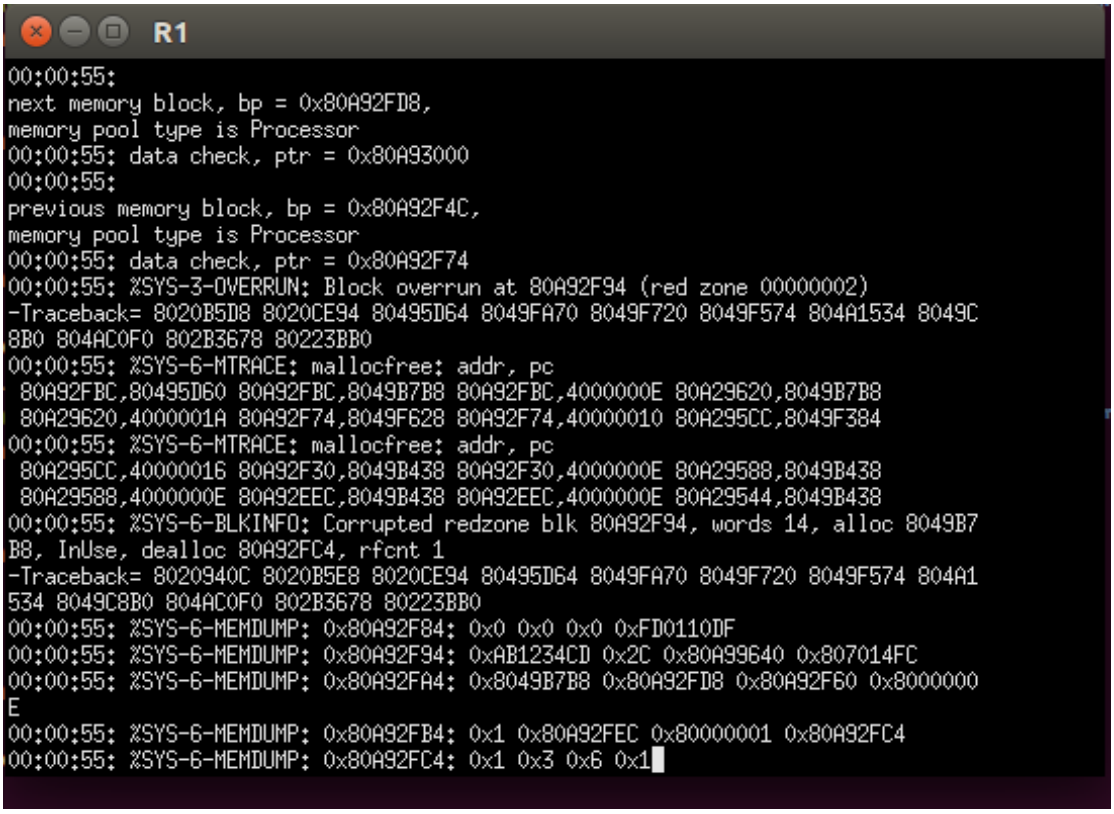

Figure 5.12: Router Console: Memory Corruption Error

The output at router console when each of the above test-cases injected is shown in above figure.

5.7.2 Denial of Service vulnerabilities

Empty UDP Packets: As stated in **CVE-2001-0566** and **CVE-2001-1097** , when empty UDP packets i.e UDP packets with no payload are sent on port 161(SNMP) results in DoS or Mild DoS for IOS version series 12. For exposing the same vulnerability Scapy is used to generate and send large number of UDP packets to Cisco 2600 router with IOS version 12.10 which resulted in Mild-DoS.

Mild-DoS is a variation of DoS in which the victim device CPU utilization is less than 100% but equal or greater than 60%. The unusual CPU utilization causes device to go unresponsive for certain periods in time. The below figures show the generation of empty UDP packets through scapy and corresponding CPU utilization.

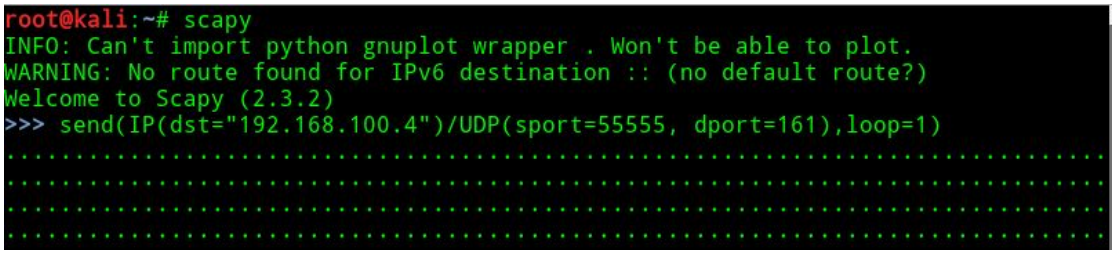

Figure 5.13: Empty UDP packets generation with Scapy

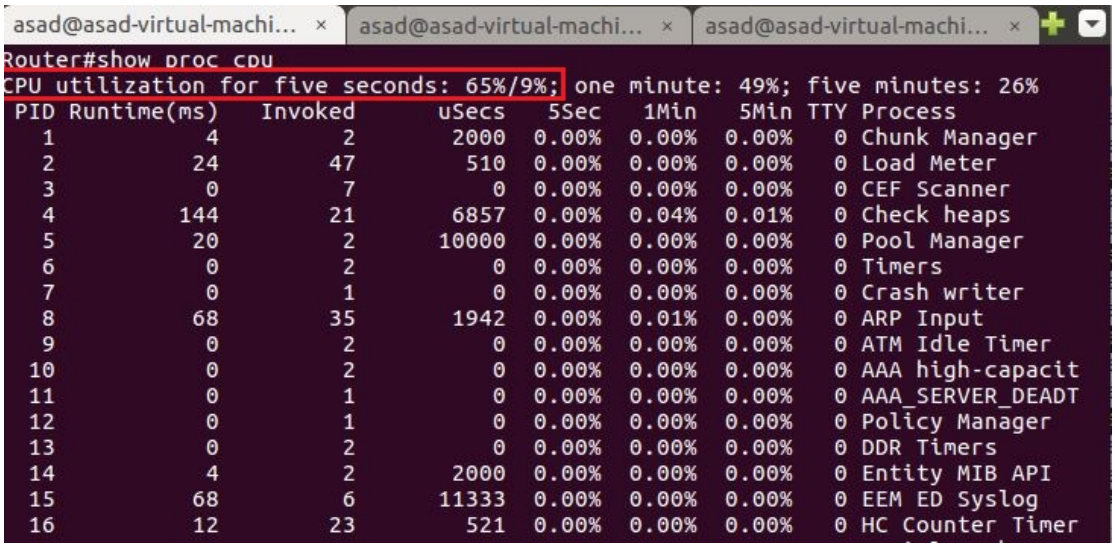

Figure 5.14: CPU utilization: C2600 router

A large number of SNMP Requests As stated in **CVE-2002-0013** that: A large number of GetRequest, SetRequest, GetNextRequest, GetBulk Request (SNMPv1) requests cause denial of service. In order to test this vulnerability we generated and sent these packets through Scapy but could not encounter DoS condition except CPU utilization around 30-40%. Packet generation and CPU utilization are shown in below figure.

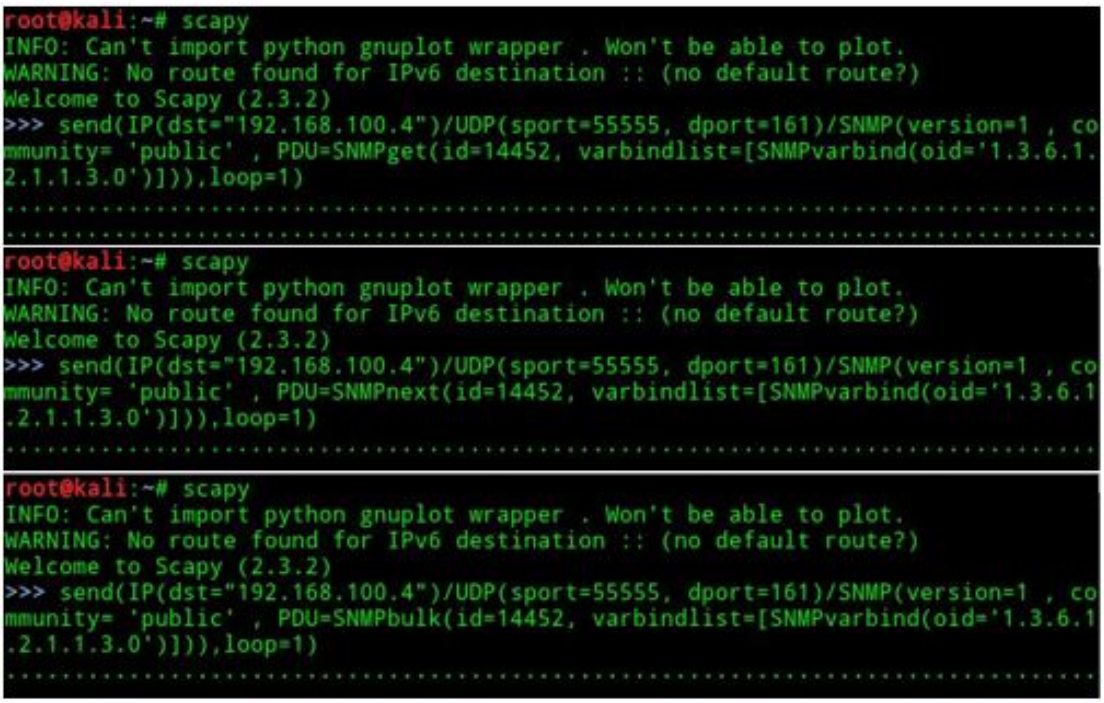

Figure 5.15: Large Number of SNMP requests generation with Scapy

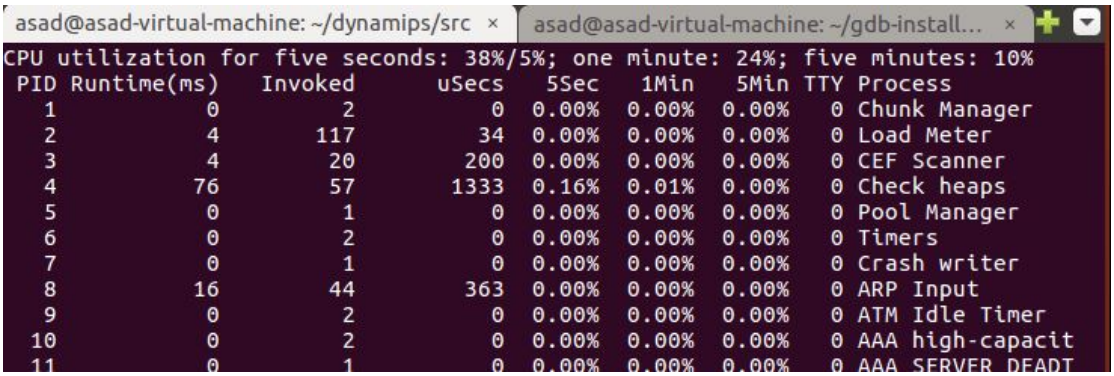

Figure 5.16: CPU utilization for large number of requests

5.7.3 SNMPv3 Fuzzing

SNMPv3 offers three different modes in terms of security i.e. Authentication only, Authentication + Data Privacy or No Authentication + No Data Privacy. As discussed earlier in Chapter.No.2 protocols with such features are a real challenge for a fuzzing task. Because these fields are dynamic.

In order to test SNMPv3 implementation No-Authentication + No-Privacy mode is used. The packets were first generated and captured then manual mutations were per-

formed with HexEditor. Several malformed test-cases are generated with long strings of OID field but could not find any flaw in Cisco router(c7200) with IOS version 15.2. The snapshot of malformed sample is shown in below figure.

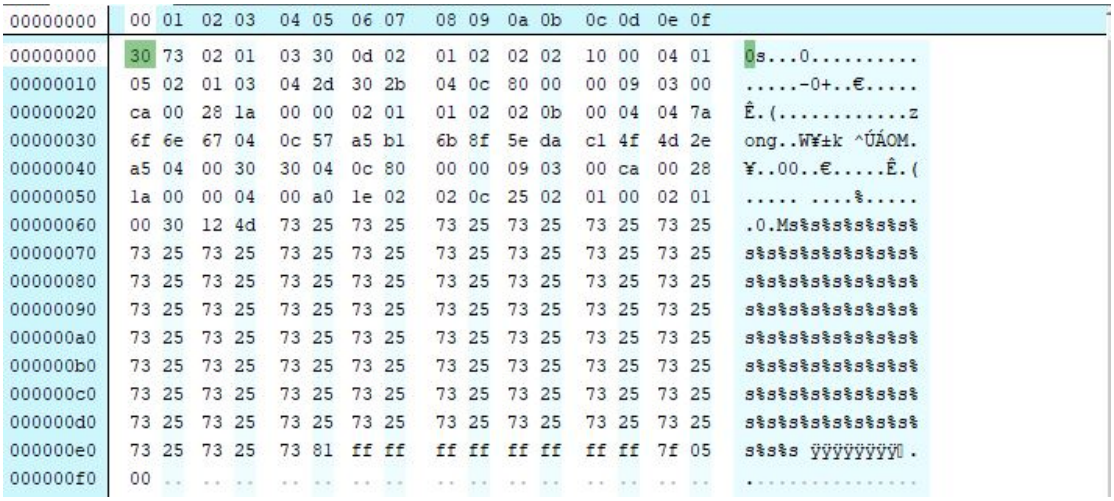

Figure 5.17: SNMPv3 Malformed test-case

5.8 Conclusion

In this chapter, fuzz testing on all three versions of SNMP is performed. It demonstrates how malformed test-cases can be generated through legitimate traffic and the injection process as well. By following the said methodology several memory corruption vulnerabilities in SNMPv1 and SNMPv2 are exposed. An existing Mild-DoS vulnerability is also exposed while fuzzing SNMPv1 with empty UDP packets.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In modern world of rapid technology evolution, the concentrations and energies of the manufacturers and developers are highly directed towards bringing new and efficient equipment which usually results in negligence of proper security testing of their products. In a globally connected network infrastructure, the devices such as routers, switches and servers which form critical part of any network are at greater risk. Therefore, proper security testing of these devices is a topmost concern. Techniques and procedures are needed to be adopted which not only ensure that these devices are not prone to existing vulnerabilities but to the zero-days as well.

Fuzzing, an automated software testing technique, is renowned for finding known and unknown flaws. For its ability to find zero-days, we tried to explore this technique for finding vulnerabilities in protocol implementations in Cisco routers.

6.1.1 Network Protocol Fuzzers Evaluation

While choosing an open-source network protocol fuzzer to test Simple Network Management Protocol(SNMP) in Cisco routers a major mutual drawback was encountered. The available open-source fuzzers lack the fuzzing ability for complex protocols. Although, commercial fuzzing solutions are available for complex protocol fuzzing but

the cost of those products is a major barrier for research community. To ease the task of choosing suitable fuzzer for the researchers in future, a comparison of famous network protocol fuzzing tools on the basis of certain criteria is performed in chapter 3.

6.1.2 Products Specific SNMP Vulnerabilities Analysis

Along with fuzzers comparison, analysis of to the date reported SNMP vulnerabilities in major network equipment manufacturers: Cisco, Juniper and Huawei is carried out. Based on the details provided by National Vulnerability Database(NVD) [2] for those vulnerabilities, an analysis for the probability of their exploitation in scope of fuzzing is performed.

6.1.3 SNMP Fuzzing

SNMP falls under the category of complex protocols. Due to same reason none of the available open source network protocol fuzzing frameworks are found useful to perform fuzz testing of SNMP. Therefore, a simple yet effective methodology is adopted by utilizing free of cost available tools as demonstrated in chapter 5. All three versions of SNMP protocol are tested. Several memory corruption vulnerabilities are exposed in Cisco C2600 router with IOS version 12.10. A known Mil-DoS vulnerability is also encountered when empty UDP packets injected to the target router. SNMPv3 implementation is found flawless.

6.2 Limitations

Although the proposed methodology is simple and can be adopted to fuzz any of the standard protocol. There comes some limitations with it. The manual mutations to generate malformed test cases is the major limitation. Moreover, it also does not cater the encrypted protocols and the protocols with dynamically changing packet fields such as checksum and hashes. For the same reason fuzz testing of SNMPv3 is done in no-authorization and no-privacy mode.

6.3 Future Work

For future work the proposed approach can be extended for fuzz testing of any other UDP based protocols. Another step forward can be to find the possibility of carrying out remote-code execution for the memory corruption vulnerabilities exposed in the experiments.

Extending open source fuzzing frameworks for fuzz testing complex protocols can also be a huge contribution to research community.

List of Abbreviations and Symbols

Abbreviations

AS	Autonomous System
ISP	Internet Service Provider
IGP	Interior Gateway Protocol
DVRP	Distance Vector Routing Protocol
RIP	Routing Information Protocol
VLSM	Variable Length Subnet Mask
EGP	Exterior Gateway Protocol
BGP	Border Gateway Protocol
SSH	Secure Shell
DHCP	Dynamic Host Control Protocol
DNS	Domain Name Service
SNMP	Simple Network Management Protocol
SMTP	Simple Mail Transfer Protocol
IOS	Internetwork Operating System
SIP	Session Initiation Protocol

OSCAR Open System for Communication in Realtime

SUT System Under Test

References

[1] Group, S.R. Cisco‘s Dominant Share of Switching and Routers Holds Steady. [Accessed 24 Dec 2017]https://www.srgresearch.com/articles/ciscos-dominant-share-switching-routersholds-steady.

[2] NVD. http://nvd.nist.gov/

[3] M. Sutton, A. Greene, and P. Amini, “Fuzzing: Brute Force Vulnerability Discovery,”Pearson Education, 2007.

[4] S. Muniz and A. Ortega. Fuzzing and debugging Cisco IOS. Barcelona, Spain, 2011.

[5] MILLER, C., AND PETERSON, Z. N. J. “Analysis of Mutation and Generation-Based Fuzzing.”Tech. rep., Independent Security Evaluators, Mar. 2007.

[6] https://arstechnica.com/information-technology/2017/03/a-simple-command-allows-the-cia-to-commandeer-318-models-of-cisco-switches/

[7] Bryan So Barton P. Miller Lars Fredriksen. An Empirical Study of the Reliability of UNIX Utilities. Tech. rep. University of Wisconsin-Madison, Dec. 1990.

[8] MILLER, C., AND PETERSON, Z. N. J. “Analysis of Mutation and Generation-Based Fuzzing.”Tech. rep., Independent Security Evaluators, Mar. 2007.

[9] Banks, Greg, et al. “SNOOZE: toward a Stateful NetwOrk prOtocol fuzZEr.”International Conference on Information Security. Springer, Berlin, Heidelberg, 2006.

[10] Gascon, Hugo, et al. “Pulsar: Stateful black-box fuzzing of proprietary network protocols.”International Conference on Security and Privacy in Communication Systems. Springer, Cham, 2015.

[11] Wang, Tielei, et al. “TaintScope: A checksum-aware directed fuzzing tool for automatic software vulnerability detection.”Security and privacy (SP), 2010 IEEE symposium on. IEEE, 2010.

[12] Tsankov, Petar, Mohammad Torabi Dashti, and David Basin. “SECFUZZ: Fuzz-testing security protocols.”Proceedings of the 7th International Workshop on Automation of Software Test. IEEE Press, 2012.

[13] Butti, Laurent. “Wi-Fi advanced fuzzing.”Black Hat Europe (2007).

[14] Petajasoja, Sami, et al. “Case Studies from Fuzzing Bluetooth, WiFi and WiMAX.”ISSE/SECURE 2007 Securing Electronic Business Processes. Vieweg, 2007. 188-195.

[15] Gorbunov, Serge, and Arnold Rosenbloom. “Autofuzz: Automated network protocol fuzzing framework.”IJCSNS 10.8 (2010): 239.

[16] LIU, Qi-xu, and Yu-qing ZHANG. “TFTP Vulnerability Exploiting Technique Based on Fuzzing [J].”Computer Engineering 20 (2007): 051.

[17] Felix Lindner, “Cisco vulnerabilities-yesterday, today and tomorrow,”in Proc. of BlackHat, Virginia, USA, September 29-October 2, 2007

[18] M. Lynn, “The holy grail: Cisco IOS shellcode and exploitation techniques,”in Proc. of BlackHat, Las Vegas, USA. July, 2005

[19] Gyan Chawdhary and Varun Uppal, “Cisco IOS shellcode,”in Proc. of BlackHat, Las Vecas, USA, August, 2008.

[20] Felix Linder, “Cisco IOS attack and defense the state of art,”in Proc. of 25th Chaos Communication Congress (25C3), Berlin, Germany, December, 2009.

[21] Felix Linder, “Cisco IOS router exploitation,”in BlackHat, Las Vecas, USA, July, 2009.

[22] A. Cui, J. Kataria and S.J. Stolfo, “Killing the myth of Cisco IOS diversity,”in Proc. of USENIX Worshop on Offensive Technologies, San Francisco, CA, USA, August, 2011.

[23] S. Muniz and A. Ortega, “Fuzzing and debugging Cisco IOS,”in Proc. of Black-Hat, Barcelona, Spain, March, 2011.

[24] Li, Fengjiao, Luyong Zhang, and Dianjun Chen. "Vulnerability mining of Cisco router based on fuzzing." Systems and Informatics (ICSAI), 2014 2nd International Conference on. IEEE, 2014.

[25] Wang, Zhiqiang, Yuqing Zhang, and Qixu Liu. "RPFuzzer: A Framework for Discovering Router Protocols Vulnerabilities Based on Fuzzing." KSII Transactions on Internet and Information Systems 7.8 (2013).

[26] Microsoft.com. Microsoft Security Development Life Cycle. Feb. 2012. url: http://www.microsoft.com/security/sdl/discover/verification.asp.

[27] Zachary N. J. Peterson Charlie Miller. Analysis of Mutation and Generation-Based Fuzzing. Tech. rep. ISE Independent Security Evaluators, 2007.

[28] Takanen A, Demott JD, Miller C. Fuzzing for software security testing and quality assurance. Artech House; 2008.

[29] softscheck.com. Fuzzelarbeit - Identifizierung unbekannter Sicherheitsluecken und Softwarefehler durch Fuzzing. Mar. 2012. url: http://www.softscheck.com/publications/ProfDrHartmutPohl_Identifizierung _unbekannter_Sicherheitsluecken_und_SoftwareFehler_durch_Fuzzing_kes20115.pdf.

[30] Mauro, Douglas, and Kevin Schmidt. Essential SNMP: Help for System and Network Administrators. " O’Reilly Media, Inc.", 2005.

[31] Kozierok, Charles M. The TCP/IP guide: a comprehensive, illustrated Internet protocols reference. No Starch Press, 2005

[32] Jiang, G., 2002. Multiple vulnerabilities in SNMP. Computer, 35(4), pp.supl2-supl4.

[33] O.Tal, S.Knight and T.Dean, âĂIJSyntax-based vulnerability testing of frame-based network protocols,âĂİ in Proc. of 2nd Annual Conference on Privacy, Security and Trust, pages 155âĂŞ160. Citeseer, 2004.

[34] Case, J., R. Mundy, D. Partain, and B. Stewart. “Introduction and Applicability Statements for Internet-Standard Management Framework - RFC 3410.” Internet Engineering Task Force (2002).

[35] https://www.ee.oulu.fi/roles/ouspg/PROTOS_Test-Suite_c06-snmpv1